Giorgio Nunzi
Caterina Scoglio
Xing Li (Eds.)

IP Operations and Management

9th IEEE International Workshop, IPOM 2009
Venice, Italy, October 29-30, 2009
Proceedings

Springer

Volume Editors

Giorgio Nunzi
NEC Europe Ltd.
Kurfürsten-Anlage 36
69115 Heidelberg, Germany
E-mail: Giorgio.Nunzi@nw.neclab.eu

Caterina Scoglio
Kansas State University
Electrical and Computer Engineering Department
2069 Rathbone Hall, Manhattan
Kansas 66506-5204, USA
E-mail: caterina@ksu.edu

Xing Li
Tsinghua University
CERNET Center
Beijing 100084, China
E-mail: xing@cernet.edu.cn

Library of Congress Control Number: 2009936051

CR Subject Classification (1998): C.2, C.2.2, C.2.5, C.2.6, C.4, B.8

LNCS Sublibrary: SL 5 – Computer Communication Networks and Telecommunications

ISSN	0302-9743
ISBN-10	3-642-04967-2 Springer Berlin Heidelberg New York
ISBN-13	978-3-642-04967-5 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12775927 06/3180 5 4 3 2 1 0

Preface

On behalf of the IEEE Communications Society, Technical Committee on Network Operations and Management (CNOM), Manweek 2009 Organizing Committee, and members of the IPOM Technical Program Committee, it is our pleasure to present the proceedings of the 9th IEEE Workshop on IP Operations and Management (IPOM 2009), held as part of Manweek 2009 during October 26-30, 2009, Venice, Italy.

Recently, there has been a heated discussion followed by highly ambitious, thus, high in risk, international research undertakings and programs, GENI and FIND in USA, FIRE in Europe, u-Japan in Japan to name a few, aiming at defining the Future Internet. Building on the success of the previous events, IPOM 2009 focused on network management challenges for the current Internet as well as Future Internet Design (FIND). We are particularly interested in network management issues related to virtualization, mobility, service provision, security, multimedia, wireless networking, and P2P applications. Building on the success of the previous events, we wanted IPOM 2008 to focus on network management challenges for the current Internet as well as future Internet design.

Like the previous four IPOM workshops, IPOM 2009 was co-located with several related events as part of Manweek. The other events were the 20th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 2009), the 12th IFIP/IEEE International Conference on Management of Multimedia and Mobile Networks and Services (MMNS 2009), the 4th IEEE International Workshop on Modeling Autonomic Communications Environments (MACE 2009), and the 6th International Workshop on Next Generation Networking Middleware (NGNM 2009). Co-locating those events provided the opportunity for an exchange of ideas among separate research communities working on related topics, and also allowed participants to forge links and exploit synergies.

This workshop attracted 35 paper submissions through an open call for papers. A rigorous review process was followed with typically three reviews per paper. The Technical Program Co-chairs decided to have a strong program with single-track sessions. With this in mind, the submitted papers were discussed based on the reviews received as well their relevance to the workshop. These proceedings present 12 accepted full papers, thus, the acceptance rate is 34%. Authors of accepted papers are from 12 different countries spanning three continents; they were invited to present their work at the conference. Additionally, five papers were selected as short papers to present highly interesting work-in-progress.

We thank members of the IPOM Technical Program Committee, IPOM Steering Committee, Manweek 2009 Organizing Committee, and reviewers for their hard work that made this workshop possible. In particular, we thank Tom Pfeifer and Alberto Gonzalez, Manweek 2009 Publications Chairs, and the Springer LNCS team for their excellent handling in the production of these proceedings.

Finally, we thank the patrons of Manweek 2009, and the TSSG for their financial and in-kind contributions to this workshop.

October 2009

Giorgio Nunzi
Caterina Scoglio
Xing Li

IPOM 2009 Organization

Workshop and Program Co-chairs

Giorgio Nunzi	NEC Europe, Germany
Caterina Scoglio	Kansas State University, USA
Xing Li	Tsinghua University, China

Steering Committee

Prosper Chemouil	OrangeLabs, France
Tom Chen	Southern Methodist University, USA
Petre Dini	Cisco Systems, USA
Andrzej Jajszczyk	AGH University of Science and Technology, Poland
G.-S. Kuo	National Chengchi University, Taiwan
Deep Medhi	University of Missouri-Kansas City, USA
Curtis Siller	IEEE ComSoc, USA

Publication Advisory Chair

Tom Pfeifer	Waterford Institute of Technology, Ireland

Publication Chair

Alberto Gonzalez Prieto	Royal Institute of Technology (KTH), Sweden

Finance Chair

Raouf Boutaba	University of Waterloo, Canada

Infrastructure Chair

Sven van der Meer	Waterford Institute of Technology, Ireland

Local Arrangements Chair

Massimo Foscato	Telecom Italia Labs, Italy

Registration Chair

Idilio Drago	University of Twente, The Netherlands

Publicity Co-chair

Carlos Becker Westphall	Federal University of Santa Catarina (UFSC), Brazil

Manweek 2009 General Co-chairs

Aiko Pras	University of Twente, The Netherlands
Roberto Saracco	Telecom Italia Labs, Italy

Manweek 2009 Advisors

Raouf Boutaba	University of Waterloo, Canada
James Hong	POSTECH, Korea
Aiko Pras	University of Twente, The Netherlands

IPOM 2009 Technical Program Committee

Alexander Clemm	Cisco Systems, USA
Ana Elisa Goulard	Texas A & M, USA
Andrzej Jajszczyk	AGH University of Science and Technology, Poland
Baek-Young Choi	University of Missouri-Kansas City, USA
Catalin Meirosu	Ericsson Research, Sweden
Charalabos Skianis	University of the Aegean, Greece
David Malone	University Maynooth, Ireland
Deep Medhi	University of Missouri-Kansas City, USA
Edmundo Madeira	State University of Campinas, Brazil
George Michailidis	University of Michigan, USA
Gerhard Hasslinger	T-Systems, Germany
Gerry Parr	University of Ulster, UK
Haobo Wang	Broadcom, USA
Joel Rodrigues	UBI, Portugal
Jun Bi	Tsinghua University, China
Lixia Zhang	UCLA, USA
Luciano Paschoal Gaspary	Federal Univ. Rio Grande Sul, Brazil
Marcus Brunner	NEC Europe Ltd., Germany
Michal Pioro	Warsaw University of Technology
Pascal Lorenz	Université de Haute Alsace, France
Petre Dini	Cisco Systems, USA
Sasitharan Balasubramaniam	Waterford IT, Ireland
Stefano Bregni	Politecnico di Milano, Italy
Thomas Magedanz	Fraunhofer Fokus, Germany
Timothy Gonsalves	Indian Institute of Technology Madras, India
Toshiaki Suzuki	NICT, Japan
Xiuduan Fang	Google, USA

Table of Contents

Scheduling IPTV Content Pre-distribution

Per Kreuger and Henrik Abrahamsson

Swedish Institute of Computer Science
Box 1263, SE-164 29 Kista, Sweden
{piak,henrik}@sics.se

Abstract. Recently many new IPTV providers have appeared, typically telephone and broadband companies, that now distribute scheduled TV channels, sometimes high-definition TV and video-on-demand (VoD) over IP in their own network. From a network management perspective, distribution of IPTV can be seen as a scheduling problem with the network links and storage facilities as resources and the objective to balance the overall load in the network. We show that scheduling and optimisation techniques can be used to make the distribution of IPTV more efficient by pushing content out to caches in the network to the right place at the right time.

1 Introduction

IPTV, were television is distributed over the Internet Protocol (IP) in a single operator network, has now become reality. Several large telecom companies have become TV providers including AT&T U-verse, Verizon Communications, Deutsche Telekom, TeliaSonera, British Telecom and China Telecom. It is estimated that there are about 50 IPTV-operators today with a total of more than 10 million subscribers and that these figures will increase significantly [7,13].

The TV service provided often includes tens of TV channels, sometimes high definition feeds, and video-on-demand (VoD). Most operators distribute the TV channels over their network as continuous streams. But they also often employ a system of peripheral storage devices (caches) located closer to the customers for time-shift TV and for more permanent storage of VoD content. Often the network is also used for consumer Internet offerings which means that the overall network traffic load has to be taken into account when distributing TV content.

From a network management point of view, distribution of TV style content to the caches is a scheduling problem with the network links and storage facilities as resources and the objective to balance the overall load in the network. TV data is a prime candidate for load balancing since availability and demand for the content is highly predictable. Also, since it is still to a large extent distributed in packages analogous to the traditional TV channels with the same content appearing in more than one "channel" and several times over e.g. a week, there is considerable room for reducing overall traffic load.

Figure 1 shows an example with TV programs in 16 channels during 2 days in January 2009. The programs are categorised as live, non-live and reruns. Only

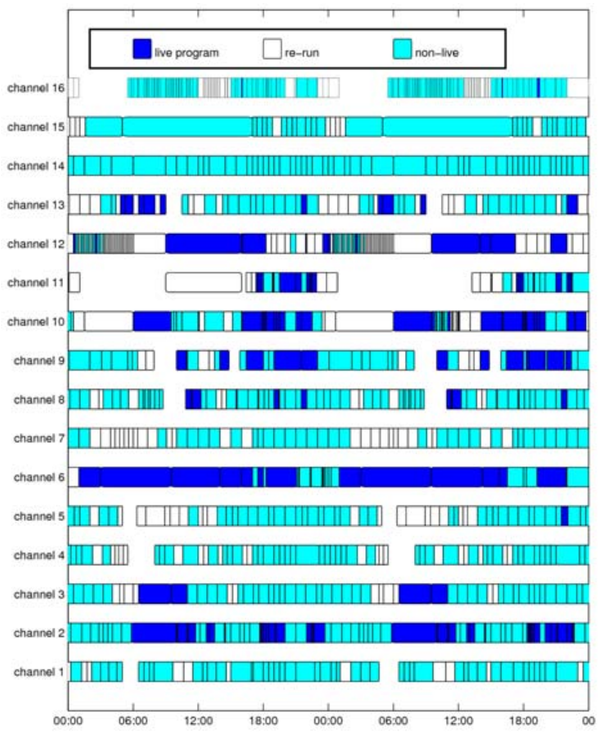


Fig. 1. Programs in a TV package of 16 channels during 2 days. Only a small part of the programs are sent live while many are reruns and transferred several times.

a small part of the programs are live: some TV shows, news and sports. Most programs are pre-recorded. Some have been recorded the same day, while most have been ready for distribution a long time before they are shown. In many channels there are also a lot of reruns of previous days programs. Often the prime time programs are shown again during night or during the next day.

If we regard the TV programs as individual information objects (rather than just part of a streamed channel), these could be stored in caches closer to the customers. If we store the data of the reruns in the caches we can filter out about 21% of the TV traffic in the backbone in the example in Figure 1.

However, since peak hours of the channels tend to coincide, the bandwidth made available by the filtering is not well distributed with respect to the other demands on the network, such as that for VoD and consumer Internet offerings. The peak load of the example (with reruns taken away) is still 100% of the 16 channel live demand. This suggests the idea of *pre*-distributing recorded content to the caches, but releasing it to the consumer at a later announced time.

Internet traffic loads and video demands in an operator network varies a lot during the day but in predictable cycles. Video-on-demand traffic typically have peak hours in the evening of each day [18]. Exploiting this fact, we show that, given an announced TV-schedule and release times at a distribution centre, pre-recorded TV-data can be pre-distributed to peripheral servers to minimise the maximum TV traffic load in the network, or to minimise the TV traffic during peak hours for VoD and Internet data traffic. With many channels and many hundreds of programs, the issue of when to best transfer the content from the distribution centre to the caches becomes quite a challenging scheduling problem.

The contributions of this paper are:

- we propose and evaluate two scheduling models for IPTV transfers and show that it is feasible to schedule a large number of channels and programs.
- we show that scheduled pre-distribution can lower the maximum bandwidth requirements for TV traffic and substantially limit the TV traffic during peak times for VoD and data traffic.

We verify the models against data extracted from real TV schedules. For a case with 16 channels during 5 days we can reduce the total bandwidth requirements by 21% just by avoiding to send the reruns. By scheduling the transfers of content to the caches we can reduce the maximum bandwidth requirement by 25% or, perhaps more interestingly, reduce the bandwidth requirement during prime time periods by 62%. Combining these two objectives is also possible.

The rest of the paper is structured as follows: In section 3 we present the network and traffic scenario. The scheduling models are presented in section 4, and are evaluated in section 5. Related work is described in section 2 and the paper is concluded in Section 6.

2 Related Work

A thorough introduction to constraint problems and methods to solve them is given in Apt [2]. The scheduling methods used in this work are well-proven techniques in application areas such as vehicle routing and timetabling but to the best of our knowledge this is the first work to apply these techniques to IPTV transfers.

Similar to our work, Agrawal *et.al* [1] present an IPTV distribution model. But their objective is to develop a general framework for planning an IPTV service deployment. They look at ways to maximize the number of new subscribers that can be added to a given infrastructure in different scenarios.

Cha *et.al* [5] presents a measurement study of viewing behaviour including channel popularity and channel switching in a large operational IPTV network.

There has been a vast amount of research on different server scheduling techniques and proxy caching strategies and combinations of the two for video-on-demand systems and content distribution networks [8,12,15,16,17]. These works are similar to ours in that they all study methods for minimizing bandwidth requirements for media content distribution and investigate the trade-offs between network bandwidth and cache storage resources. But our work is fundamentally different in that it considers IPTV with a schedule that is known in advance.

Recently much research has focused on peer-to-peer techniques for TV distribution [6,9,19] and VoD [10,11,14]. Cha *et.al* [6] analyse the TV viewing behaviour in an IPTV system and explore how P2P-techniques can complement existing Telco-managed IPTV architectures.

Huang *et.al* [11] describe the experiences with the P2P-VoD system deployed by PPLive. Suh *et.al* [14] present a Push-to-Peer VoD System. This work has some similarities with ours in that network providers can push out content to caches in the network during periods with low network load.

3 IPTV Scenario and Problem Description

We consider a scenario with distribution of traditional scheduled TV over IP in an infrastructure based single-operator environment. We assume an architecture

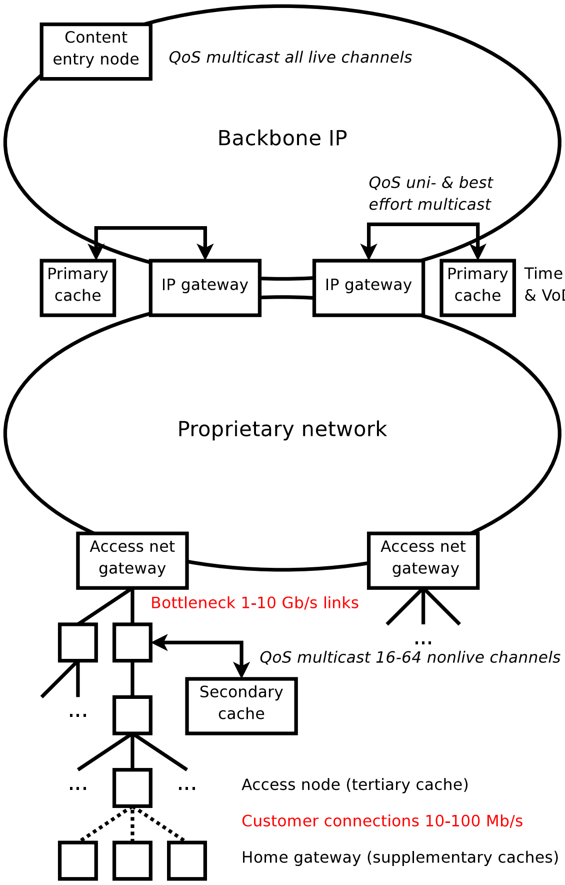

Content entry node
QoS multicast all live channels
Backbone IP
QoS uni- & best effort multicast
Primary cache
IP gateway
IP gateway
Primary cache
Time shift & VoD srv
Proprietary network
Access net gateway
Access net gateway
Bottleneck 1-10 Gb/s links
QoS multicast 16-64 nonlive channels
Secondary cache
Access node (tertiary cache)
Customer connections 10-100 Mb/s
Home gateway (supplementary caches)

Fig. 2. Schematic network model

similar to the IPTV systems described in [4,5] with a central node (distribution centre), where the IPTV content enter the network, and a number of local nodes with peripheral storage. The TV content can be distributed to the local nodes using multicast. This case is illustrated in Figure 2. For maximum benefit the caches should be placed downstream from the bottleneck link resources.

We assume that the operator have control over both the distribution centre with all non-live content stored and a number of peripheral servers closer to the end users with a limited storage capacity.

The main tasks of a content distribution management system for such a scenario include mechanisms to:

- schedule transfer of content from a distribution centre to peripheral servers.
- allocate storage of released (and pre-distributed) content on peripheral servers.
- schedule removal of content from servers based on assessed demand.

These problems are in turn constrained by:

1. The *release time* when content becomes available at distribution centre.
2. The scheduled *streaming start* when the content is released to the consumers
3. *Link capacity* on the backbone network
4. *Storage capacity* of peripheral servers
5. Expected *lifetime* of the content on the peripheral server.

Our models of this problem regard each peripheral node as a storage resource with a fixed capacity (i.e. a cumulative resource) and each backbone link as an independent bandwidth resource. The release time at the distribution centre of each unit of content is a lower, and the streaming start at the peripheral server plus some estimate of network latency, an upper bound on the start of the corresponding transfer task.

The option to pre-distribute the content before the execution of the streaming schedule amounts to a trade off between efficient use of bandwidth and storage resources. In this work we focus on schedules for a single transfer resource. Further studies on scheduling of the storage resources and hierarchically organised caches are left for future work.

To produce a straightforward but still interesting case we have made certain assumptions about both the network and the demands:

1. All content is stored and made available at the distribution centre at a specified release time and will remain available there at least until its scheduled streaming start.
2. All peripheral servers receive the same data through multicast from the distribution centre.
3. The available bandwidth on the links between the distribution centre and each peripheral server is constant and identical for all peripheral servers
4. Each unit of content is stored at the peripheral server from the time of its first transfer and at least until the end of its last announced rerun.

Assumptions 2, 3 and 4 together imply that we may consider only a single link resource and a single peripheral server, since all the others will be treated equivalently. Not all of these assumptions are completely realistic but the resulting case is anyhow a core problem in any more general setting.

4 Scheduling Models

4.1 A Unit Task Cumulative Scheduling Model

Model parameters (inputs). Each unit of content is specified in terms of the following parameters:

$\{r_i\}$ release time for the complete unit at the distribution centre
$\{s_i\}$ streaming start time at the peripheral server
$\{d_i\}$ duration of streaming content in seconds
$\{z_i\}$ size of the content in bits
$\{Lc\}$ link capacity in b/s

Unit constraint model. Each unit of content is mapped to a single transfer task on the link resource.

Transfer tasks. Each transfer task i is represented by the variables:

$\{t_i^s\}$ start of the transfer task
$\{t_i^d\}$ duration of the transfer task
$\{t_i^r\}$ bit rate of transfer (assumed to be constant for the duration of the transfer)

Linear transfer task constraints. The bounds of transfer task variables are calculated from the inputs as follows:

$t_i^s \geq r_i$ transfer start at or later than release time
$t_i^s \leq s_i$ transfer start at or earlier than streaming start
$t_i^s + t_i^d \leq s_i + d_i$ transfer end at or earlier than streaming end

Note how the latest transfer start coincides with the streaming start and that the end of a transfer task has to be completed before the streaming end. This means that if we delay the transfer as much as possible i.e. $t_i^s = s_i$, the transfer has to be made at at least live rate i.e. $t_i^d \leq d_i$, to guarantee that streaming will be possible at the peripheral server.

Transfer task resource constraints. The capacity requirement of each task i corresponds to the transfer rate t_i^r which is related to the transfer duration t_i^d by the (quadratic) constraint $t_i^r t_i^d = z_i$. The resource constraint amounts to ensuring that each transfer task is fixed in time such that the capacity of the transfer resource is not exceeded by the cumulative rates of the set of tasks executing at any one time.

This is achieved by a cumulative constraint that reasons on a geometric placement problem where each task is represented as a rectangle of length t_i^d and height t_i^r. This combinatorial constraint guaranties that for any valid assignment of the scheduling variables, the task rectangles can be placed so that the sum of heights of all rectangles occupying any given x-position never exceeds the link capacity Lc.

4.2 Model Analysis

A property of the model described above is that each unit of content must be transferred as a continuous block at an constant rate. The transfer rate for each unit is part of the decision problem, for which the solver has to select both rate and duration for each task. The relation between these two parameters is given by the fact that their product should be equal to the constant size of the program data. This quadratic constraint for a fixed discretisation of both time and bandwidth is one important source of complexity even though it turns out that our implementation of this model scales very well.

One way of illustrating what is going on is by visualising the number of valid assignments of the durations and rates for a unit of given size and fixed time and bandwidth resolutions. Consider the case of a 30 minute real time unit of content. This could e.g. be transferred with a rate corresponding to 1 real time channel in 30 minutes, in 15 minutes using a rate corresponding to 2 real time channels and in 10 minutes using 3. We could also transfer it at less than the real time rate, e.g. in rates corresponding to $\frac{2}{3}$ and $\frac{1}{3}$ of a channel. Figure 3 shows 6 valid

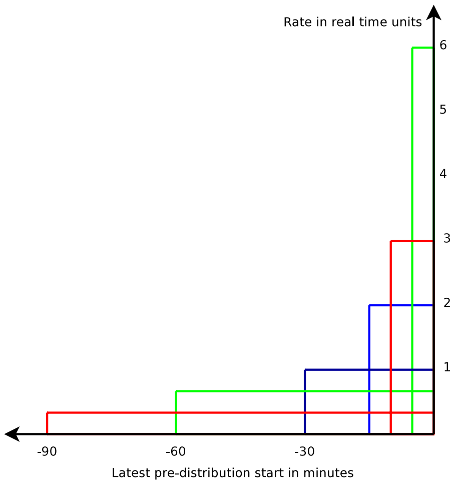


Fig. 3. 6 valid assignments of rate and duration for one particular discretisation of a 30 minute unit

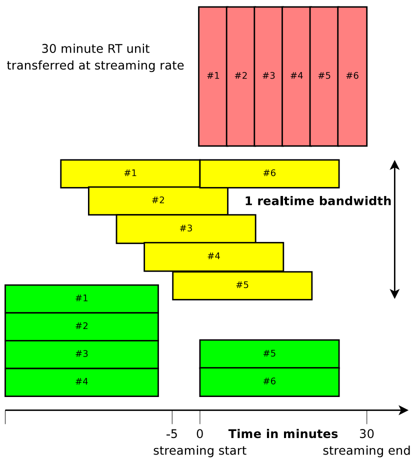


Fig. 4. Splitting a unit of content into independent parts and scheduling their transfer corresponding to variable rate and pre-emption for the complete unit

assignments of rates and durations for this case. This type of formulation makes two implicit but related assumptions that upon reflexion are not necessary.

1. The data transfer rate is constant for a given unit of content
2. The transfer of the complete program data is non-pre-emptive

Relaxing these, suggests an alternative discrete model where fragments of the content data have individual deadlines and can be transferred independently.

4.3 A Fragment Task Cumulative Scheduling Model

Dividing units into smaller fragments will enlarge the scheduling problem, but there may be other advantages to doing so. We can e.g. vary the transfer rate of a complete unit during the transfer which should give more flexibility in the scheduling problem and, at least in theory, allow for better solutions.

To achieve a problem with tasks of fixed duration and bandwidth consumption we will also need to decide on a discrete time resolution and fix a transfer rate of each fragment. Assume that we use minutes as the smallest time unit and 1/6 of a channel as the smallest bandwidth unit. A 30 minute unit can then be split into 6 individual 5 minute fragments and each fragment transferred individually at any rate, as long as we set the deadline of the completion of the fragment transfer to coincide with the start of the streaming start of that fragment.

One such case is illustrated in figure 4 where the streaming schedule is given at the top, the latest possible schedule for the individual fragments in the middle and one alternative schedule illustrating some of the options for variable rates and pre-emptions of the transfer at the bottom.

Model parameters

t smallest time unit in seconds $t = \gcd\left\{t_i^d\right\}$.
b the number of discrete bandwidth units in one streaming channel.

in addition to all the parameters listed in section 4.1.

Time abstraction. To limit the impact of the increased size of the scheduling problem we will introduce a time abstraction that significantly reduces the search space for valid schedules. Even though, in reality, the transfer time of each fragment has to be scheduled down to t minutes, we show that we do not need to do that at the level of weekly planning.

It suffices to determine the start time of its transfer in terms of a longer duration $\hat{t}$. Each fragment scheduled during a particular interval of duration $\hat{t}$ can be considered to occupy a fragment of the bandwidth resource corresponding to a rate of $1/b$ during that period.

In the empirical below section we have used $\hat{t} = 15$m and a fragment size of corresponding to 5m of streaming transfer which could be 375MB for a high quality compressed HD stream of 10Mb/s. The impact of this level of abstraction on the objective value of the optimal solutions is minor.

Fragment deadlines. We specify how to break each unit into fragments and assign latest start times to tasks corresponding to each fragment so that a schedule respecting these deadlines will reserve enough bandwidth on the link so that streaming is possible at the peripheral server. Algorithm 1 assigns latest start

Algorithm 1. Assigning fragment task deadlines

```
for all $<units\ i>$ do
    $n = \left\lceil \frac{d_i b}{\hat{t}} \right\rceil$ // no. of fragments of unit $i$
    for $j = 1; j \leq n; j{+}{+}$ do
        $s_{ij} = \left\lfloor s_i + j\frac{\hat{t}}{b} \right\rfloor$ // latest start of fragment $ij$ at given resolution
    end for
end for
```

times s_{ij} for all fragments ij such that if the required completion of the transfer of the fragment falls in the kth $\hat{t}$-long time interval, its latest start time will be k. Requiring in addition that start times t_i^s are fixed to multiples of $\hat{t}$ completes the abstraction and allows us to search efficiently for (but possibly excluding some) valid solutions to the original problem.

Fragment constraint model. Each unit of content is mapped to a number of separate transfer tasks on the transfer resource. All of these have unit resource consumption and duration $\hat{t}$. The only unknown left is the start time of each fragment transfer which is represented by the variable t_{ij}^s.

Linear transfer task constraints. The bounds of the transfer task variable are calculated from the inputs as follows:

- $r_i \leq t_{ij}^s \leq s_{ij}$ transfer start between release time and assigned deadline
- $t_{ij}^s = k_{ij}\hat{t}$ where k_{ij} is an auxiliary integer variable.

This constraint ensures that t_{ij}^s can only be assigned to multiples of $\hat{t}$. Enumerating the k_{ij} variables instead of the t_{ij}^s reduces the search space significantly.

Transfer task resource constraints. Given that each fragment has a duration of $\hat{t}$ and resource consumption of 1, the resource constraint amounts to ensuring that each fragment transfer task is fixed in time such that the capacity of the transfer resource is not exceeded by the cumulative rates of the set of tasks executing at any one time. This is achieved in the same way as in the unit model by a cumulative constraint. The main difference now is the increased number of tasks and the uniformity of the tasks.

5 Evaluation

5.1 TV Data Set

We have evaluated the two models on a variety of problems extracted from actual IPTV schedules. We use a data set of 16 channels during 5 days in January 2009.

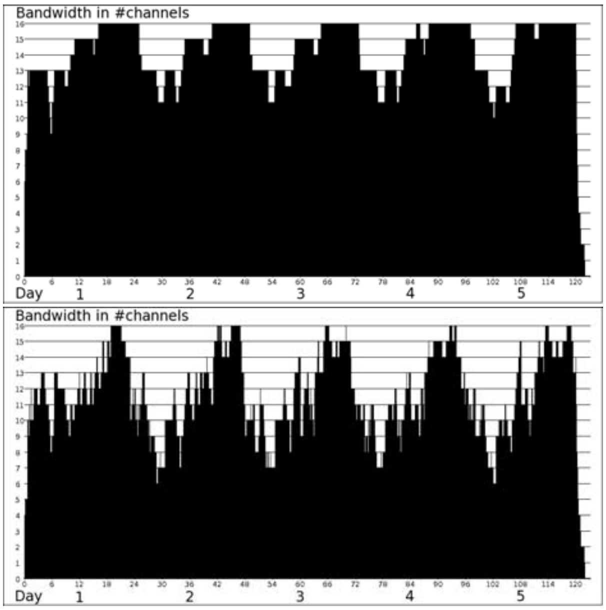


Fig. 5. Load histograms for streaming schedule of 16 channels before and after duplicate filtering

These channels constitutes an IPTV offering of TeliaSonera. It includes some of the most popular TV channels in Sweden and is a mixture of public service channels, commercial channels, sports, music and documentary channels. For each program in the set we determined the scheduled start time, program length, if the program was sent live or, if not, an estimate of when it could have been released for transfer.

The channels include a lot of reruns. Reruns and runs of the same program in different channels were filtered so that content is only transferred once. Figure 5 shows load histograms of the bandwidth resource before and after filtering. Filtering this set gave a reduction of the total number of hours of streaming transfer from 1 784 to 1 412, a reduction of 21%. As seen in the figure, the peak load is still the same as the bandwidth requirement of streaming 16 channels (shown in whole channel bandwidth units and hours) simultaneously. The resulting data consists of 1899 individual units of content (programs).

5.2 Comparison of the Methods

For the performance analysis we have used two measures of the quality of the solutions produced by our implementation: Reduction in the peak bandwidth requirement over the full 5 days, and reduction of the bandwidth requirement during the prime time periods (between 18:00 and 24:00) of each day.

For the *peak metric* we try to minimise the maximum TV traffic load. The idea with the *prime time metric* is to investigate the possibility to limit the TV

Table 1. Performance comparison of the two models

Unit model

chnls	peak	time	prime	time	pk+prm	time
3	0%	1s!	0%	1s!	0+0%	1s!
4	25%	8s!	25%	6s!	25+0%	16s!
5	33%	26s!	40%	21s!	33+6%	70s!
8	33%	125s	50%	183s!	29+20%	513s
16	25%	954s	62%	4797s!	12+50%	7319s

Fragment model

chnls	peak	time	prime	time	pk+prm	time
3	0%	2s!	0%	2s!	0+0%	2s!
4	25%	65s!	25%	56s!	25+0%	116s!
5	33%	239s	40%	384s!	33+6%	837s
8	33%	1725s	50%	2326s!	29+20%	4057s
16	16%	8748s	35%	2857s	-	-

traffic during certain periods of the day. This is because TV traffic might share the network with Internet data traffic and VoD traffic that varies in cycles during the day. VoD traffic volumes typically increase a lot during prime time [18]. The prime time is also the most challenging period to limit the TV traffic since all channels then have scheduled programs many of which are live.

The implementation and evaluation of the models were done in the constraint programming system SICStus Prolog [3] using the built-in linear and combinatorial constraints on various subsets of the data set using a standard 32bit 1.6 MHz laptop with 2GB of main memory.

Table 1 summarises the result of the comparison. For each of the models, the reduction of peak and prime time load is given in percent of the bandwidth requirement for streaming. In the "time" columns the time to find a feasible solution of that quality is given in seconds. An "!" following the time value indicates that the given value was proved optimal within the given time.

The combined metric given in the third column has two values: the peak load of the solution and the additional reduction of the load during prime time period. The time given in this case is for first finding the optimal prime time value and then reducing the peak value as much as possible without increasing the value of the prime time measure. If the prime time value could not be proved optimal within a reasonable time, no solution for the combined metric is given. The bandwidth resolution b used in these runs was 3 units/channel and time resolution 15 minutes, which is a realistic scale. In the experiments we assume all content to be HD quality video transferred over 10Gb/s links.

Figure 6 shows two load histograms for the full set of sixteen channels. These were produced by the unit method and are directly comparable to those of figure 5 which should give a good indication of the kind of gains possible.

The peak load was reduced by 25% of the bandwidth required for streaming. This result is reached in about 15 minutes of run time, but not proved optimal within an hour. Minimising the load during the prime time periods gives a

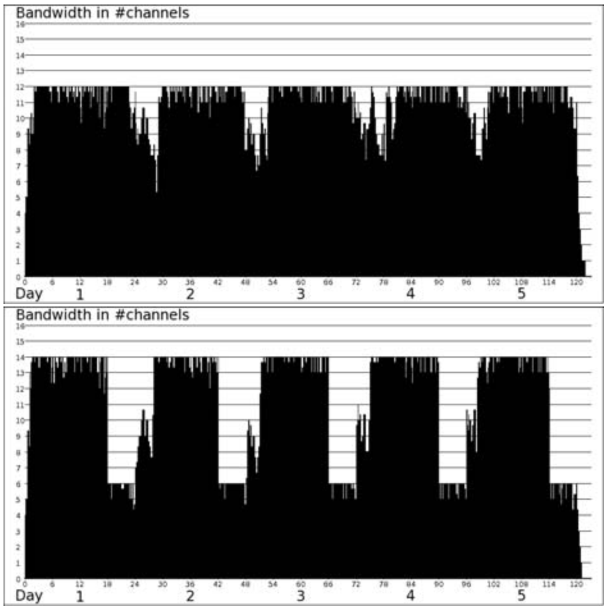


Fig. 6. Load histograms for near optimal solutions for 2 different metrics

reduction of 62% which was found and proved optimal in about 80 minutes. The best solution for the combined measure, reducing the peak load for the optimal prime time solution, by an additional 12% was found in about 2 hours, but not proved optimal after 2.5 hours. This solution is shown in the right histogram of figure 6 and gives a very nicely balanced combination of loads for the two measures. Note that the valleys in the load histograms occur at prime time of each day, preserving transfer capacity for other types traffic in the network during these periods.

The performance of the two methods are comparable for the chosen resolutions even if the unit model is generally quite a bit faster. The expected increase in solution quality from the fragment model has *not* been realized for this data set. Using a finer grained time resolution increases the run times significantly, especially for the fragment model.

These results indicate that the unit model is superior since even though the fragment model should, in theory at least, allow for better solutions, this gain cannot be realised in practice, at least not using our current implementation. Still, the fragment model remains a candidate for adaptation to a integer programming IP) implementation which may be needed when addressing larger problems of more elaborate caching architectures.

6 Conclusions and Future Directions

In this work we consider the distribution of traditional scheduled TV over IP in an infrastructure based single-operator environment. We show that if TV distributors have access to the TV program content (and not only TV streams) they could save considerable amount of bandwidth by using caching and scheduling. The bandwidth requirements on bottleneck link resources can be decreased by avoiding to send reruns and by pushing content out to caches during periods with low traffic load. We compare and evaluate two scheduling models, a unit task and a pre-emptive model, and show that it is feasible to schedule a large number of channels and programs. Judging from the empirical investigation the unit model is superior in performance and even though the pre-emptive model should in theory allow for better solutions this is not realised on the investigated data set. Our evaluation, using real TV-schedule data for 16 channels over 5 days, show that scheduled pre-distribution can lower the maximum bandwidth requirements for TV traffic and substantially limit the TV traffic during peak times for VoD and data traffic.

Future work include a reformulation of the fragment model as an Integer Program using a packing abstraction to further improve scaling and extending the model to allow several storage and transfer resources.

References

1. Agrawal, D., Beigi, M., Bisdikian, C., Lee, K.: Planning and Managing the IPTV Service Deployment. In: Proceedings of 10th IFIP/IEEE International Symposium on Integrated Network Management (IM 2007), Munich, Germany (May 2007)
2. Apt, K.R.: Principles of Constraint Programming. Cambridge Univ. Press, Cambridge (2003)
3. Carlsson, M., et al.: SICStus Prolog Users Manual. SICS (1995), ISBN 91-630-3648-7, `http://www.sics.se/isl/sicstus/docs/`
4. Cha, M., Choudhury, G., Yates, J., Shaikh, A., Moon, S.: Case Study: Resilient Backbone Design for IPTV Services. In: Proceedings of the IPTV Workshop, International World Wide Web Conference (May 2006)
5. Cha, M., Rodriguez, P., Crowcroft, J., Moon, S., Amatriain, X.: Watching Television Over an IP Network. In: Proceedings of Internet Measurement Conference (IMC) 2008, Greece (2008)
6. Cha, M., Rodriguez, P., Moon, S., Crowcraft, J.: On Next-Generation Telco-Managed P2P TV Architectures. In: Proceedings of International workshop on Peer-To-Peer Systems, IPTPS (2008)
7. Telcos Tunin. In to IPTV, News@Cisco, `http://newsroom.cisco.com/dlls/2007/ts_121107.html`
8. Eager, D., Ferris, M., Vernon, M.: Optimized Regional Caching for On-Demand Data Delivery. In: Proceedings of Multimedia Computing and Networking (MMCN 1999), San Jose, California (January 1999)
9. Hei, X., Liang, C., Liang, J., Liu, Y., Ross, K.W.: A measurement study of a large-scale P2P IPTV system. IEEE Transactions on Multimedia (2007)
10. Huang, C., Li, J., Ross, K.W.: Can Internet VoD be Profitable? In: Proceedings of ACM Sigcomm 2007, Kyoto, Japan (2007)
11. Huang, Y., Fu, T., Chiu, D., Lui, J., Huang, C.: Challenges, Design and Analysis of a Large-scale P2P-VoD System. In: Proceedings of ACM SIGCOMM 2008, Seattle, USA (August 2008)

12. Ramesh, S., Rhee, I., Guo, K.: Multicast with Cache (Mcache): An Adaptive Zero-Delay Video-on-Demand Service. In: Proceedings of IEEE INFOCOM 2001, Anchorage, Alaska (April 2001)
13. IMS Research. IPTV: A Global Market Analysis - 2008 edn. `http://www.imsresearch.com`
14. Suh, K., Diot, C., Kurose, J., Massoulie, L., Neumann, C., Towsley, D., Varvello, M.: Push-to-Peer Video-on-Demand System: Design and Evaluation. IEEE Journal on Selected Areas in Communications 25, 1706–1716 (2007)
15. Venkatramani, C., Verscheure, O., Frossard, P., Lee, K.W.: Optimal proxy management for multimedia streaming in content distribution networks. In: Proceedings of NOSSDAV 2002, Miami, USA (2002)
16. Verscheure, O., Venkatramani, C., Frossard, P., Amini, L.: Joint server scheduling and proxy caching for video delivery. In: Proceedings of the Sixth International Workshop on Web Caching and Content Distribution, Boston, USA (June 2001)
17. Wang, B., Sen, S., Adler, M., Towsley, D.: Optimal proxy cache allocation for efficient streaming media distribution. In: Proceedings of INFOCOM 2002, USA (2002)
18. Yu, H., Zheng, D., Zhao, B., Zheng, W.: Understanding user behavior in large-scale video-on-demand systems. In: Proceedings of EuroSys 2006, Belgium (2006)
19. Zhang, X., Liu, J., Li, B., Yum, T.: Coolstreaming/donet: A data-driven overlay network for efficient live media streaming. In: Proceedings of IEEE INFOCOM 2005, Miami, FL, USA (March 2005)

VoIP Measurement Architecture Using Data Mediation

Atsushi Kobayashi and Keisuke Ishibashi

NTT Information Sharing Platform Laboratories
3-9-11 Midori-cho, Musashino, Tokyo 180-8585, Japan
akoba@nttv6.net, ishibashi.keisuke@lab.ntt.co.jp

Abstract. A VoIP measurement architecture using a flow-based measurement method, such as IPFIX/PSAMP, is presented. In particular, we focus on an ACL-based metering function adaptable to existing router/switch capabilities. We also applied a mediation function handling the ACL-based filtering configuration and correlating both SIP and RTP packet-based data records to build the VoIP measurement architecture. This architecture enables network operators to measure VoIP quality as needed in large-scale networks. Finally, we demonstrated the feasibility of the measurement system by evaluating a prototype.

Keywords: IPFIX, PSAMP, VoIP, SIP.

1 Introduction

In recent years, time-sensitive streaming traffic, such as IPTV and voice-over-IP (VoIP), has been increasing in IP backbone networks. Therefore, network operators have been exploring how to make passive measurements of the quality of service (QoS) performance of real networks more easily. When network failure occurs or customers inquire about VoIP service conditions, network operators try to investigate failure points between the caller and callee by evaluating the results of a ping, the call detail record (CDR), and logging data. However, there is no way to investigate the failure points immediately. In addition, investigating is more difficult if the RTP session passes directly between the caller and callee without a media gateway (MGW) and session border controller (SBC) because the network devices (routers and switches) passing through the session do not identify the RTP packet. Therefore, a more flexible, scalable, and adaptable measurement method for any VoIP network topology is required.

These days, flow measurement technologies, such as NetFlow [1] and sFlow [2], are widely used in several kinds of networks. However, the technologies do not have QoS monitoring capabilities. IP Flow Information eXport (IPFIX) [3] and Packet SAMPling (PSAMP) [4], flow- and packet-based passive measurement methods, respectively, have been developed in IETF to expand the existing applicability covered by NetFlow and sFlow. Therefore, IPFIX and PSAMP, which are applicable to the QoS monitoring in Ref. [5], have developed the primitive techniques. However, QoS monitoring for specific application layers, such as a VoIP or IPTV session, has not been developed in IPFIX. This paper illustrates the limitation of existing IPFIX/PSAMP features, and proposes a VoIP measurement architecture using mediation function and improved configuration scheme to overcome the limitation.

In VoIP measurement, there are three main challenges: adapting to any network topology, adapting to large-scale networks, and correlating Session Initiation Protocol (SIP) signaling data and RTP session data. The proposed system uses access control list (ACL)-based metering functions adaptable to existing commercial routers and switches and improves the IPFIX configuration data model through those functions. We also evaluated the feasibility of the approach by developing a prototype of the measurement system.

The organization of this paper is as follows: section 2 explains the work related to our approach, section 3 gives an overview of IPFIX and PSAMP, section 4 summarizes the challenges for VoIP measurement methods, section 5 describes the proposed measurement method to cope with these challenges, section 6 presents the measurement logical architecture, section 7 describes the prototype measurement system and the result of evaluating its feasibility, and finally, section 8 presents the conclusion and future work.

2 Related Work

A VoIP measurement using IPFIX has been proposed by several groups. In Ref. [6], an end-to-end VoIP measurement method and template data structure including new information elements related to RTP packets was presented. Because this approach requires all end devices in a network domain to have an exporting IPFIX function, its scalability is limited. In Ref. [7], the QoS monitoring method for RTP sessions using nProbe was presented. Because this approach assumes that SIP signaling and the associated RTP sessions pass through the same network segments, it is suitable for enterprise networks rather than carrier-class backbone networks. In Ref. [16], an end-to-end VoIP monitoring method of correlating SIP signaling data and RTP session data was presented. This approach also has limited scalability. In Ref. [8], another correlation method between these data using IPFIX mediation and new information elements related to SIP signaling and RTP packets was presented. This approach adopts the distributed measurement architecture adaptable for carrier-class backbone networks and has similarities to our proposed architecture. However, adapting to any network topology seems to be difficult, because the approach assumes the RTP packets pass through MGW. And its feasibility has not been evaluated yet. For further efficiency, a method using the general metering function hosted in commercial routers or switches is required. This work helps illustrate the feasibility of the VoIP measurement architecture and improves it by adapting the IPFIX configuration scheme in Ref. [9]. In addition, we extend the work in Ref. [10] to extract the RTP session using an ACL-based metering function by using the improved IPFIX configuration scheme.

3 IPFIX/PSAMP Summary

IPFIX mainly defines the protocol that delivers flow records and packet reports from IPFIX exporters, such as routers or switches, to IPFIX collectors storing them for visualization or analysis. A flow indicates a set of packets, which have common properties in a given time interval at an observation point. A flow record indicates flow-based traffic data including summed bytes and packet fields. A packet report indicates packet-based traffic data including generally the entire header of the IP layer and transport layer.

3.1 IPFIX and PSAMP Features

The IPFIX protocol in Ref. [3] has been developed as extended NetFlow version 9 in Ref. [1]. NetFlow version 9 uses a flexible and extensible template structure instead of a fixed format style, such as NetFlow version 5. Other than template structure, IPFIX has the following features.

- Reliable SCTP/TCP transport session protocols can be used.
- Enterprise-specific information elements can be applied as vendor-specific extensions. Therefore, each vendor can create any unique information element except for IPFIX-specified information elements.
- Variable length information elements can be applied to the template. They allow a packet report to be exported through the IPFIX protocol.

PSAMP in Ref. [4] mainly defines several sampling and filtering techniques to measure packet-based traffic behavior. The combination of these techniques provides more elaborate traffic measurement. Extracted packet property data as a packet report is also delivered by the IPFIX protocol. Basic PSAMP techniques are as follows.

- Sampling: systematic sampling, random sampling
- Filtering: property match filtering, hash-based filtering

Property match filtering is similar to the ACL-based filtering. However, PSAMP supports only the “AND” operation; it does not support access-list description.

3.2 IPFIX Mediation and Configuration

IPFIX mediators in Ref. [11,12] as intermediate devices between the exporter and collector, and IPFIX configuration in Ref. [9] have recently been discussed in the IPFIX Working Group. The mediator functionalities improve the existing functional limitation of IPFIX/PSAMP, and compensate for the missing capabilities of the exporter and collector. These functions are as follows:

- Aggregating flow records/packet reports
- Correlating a set of flow records/packet reports
- Filtering and selecting flow records/packet reports
- Modifying flow records/packet reports, including changing the transport protocol that carries IPFIX messages.

With regard to QoS monitoring, the correlation function creates new metrics, such as one way delay, on the basis of the correlation of packet reports from different exporters along a specific path.

An IPFIX/PSAMP configuration data model being discussed in the IPFIX Working Group is also expected to further expand IPFIX flexibility by adjusting the sampling and filtering parameters. The data model is presented through an XML or YANG scheme. In either way, if a security incident or link failure occurs, the IPFIX configuration allows network operators to change the selection parameters dynamically to investigate the traffic. Although the data model expands IPFIX flexibility, it does not support the ACL-based filtering scheme according to PSAMP specifications. Because of this, the implementation of this model seems to be limited.

4 Challenges for VoIP Measurement

As mentioned in the introduction, the main requirement of a measurement system for network operators is to keep track of the behavior of VoIP sessions on large-scale networks at any time. Measurement systems are assumed to be utilized for trouble-shooting when an incident occurs or customers inquire about the service conditions. We describe the requirements to keep track of VoIP traffic as follows.

- Requirement #1: Adaptable to any network topologies

The measurement system needs to adapt to any network topology regardless of device types. Even if the network domain does not have MGW and SBC, the system needs to measure the QoS performance on the basis of the traffic data exported from general network devices, such as routers and switches. We need to introduce an easier implementation method for commercial routers and switches.

- Requirement #2: Large scale measurement

In large-scale networks, the observed packets in a few thousand VoIP sessions at multiple observation points generate a large amount of traffic data. Thus, monitoring all streaming packets and estimating the precise service quality seems difficult. We need to utilize the PSAMP functionalities and monitor serious network failure affecting service quality.

- Requirement #3: Correlation between SIP signaling and RTP sessions

SIP packets and RTP packets pass through different paths respectively between caller and callee. It is generally difficult for the router and switch to identify the RTP packets, because the UDP port number of RTP session is not fixed. Therefore, prior to capturing the RTP session, we need to extract the Session Description Protocol (SDP) information presenting the media description from the SIP signaling. Thus, the system needs a function correlating SIP signaling and RTP sessions on large-scale networks.

5 Proposed Measurement Method

Therefore, we explored a measurement method meeting the requirements.

- Requirement #1: Adaptable to any network topologies

To execute VoIP measurement from a general router and switch, the technologies used by the method need to be similar to existing router/switch capabilities. To achieve the requirements, we explored easy implementation techniques for general routers. Measuring QoS performance through detecting packet loss and disorder by keeping track of the sequence number of an RTP header requires packet-based traffic data rather than flow-based traffic data. Thus, ACL-based filtering seems suitable for extracting wanted packets prior to sampling. Similar approaches already released are ACL-based sFlow [14] and Input Filter NetFlow [15]. Thus, we focused on an

ACL-based metering function that selects a wanted packet and then exports its packet-based data to an appropriate collector. The ACL-based metering function is compatible with existing network management systems and router functionality. In addition, the techniques can be utilized for other traffic monitoring approaches, such as IPTV traffic monitoring and security inspection.

- Requirement #2: Large scale measurement

We focus on the hierarchical structure to increase the scalability of the measurement system. A single collector's performance is overwhelmed by the huge amount of VoIP traffic data, including SIP/RTP packet-based data. Thus, the IPFIX mediator is suitable for preprocessing metric computations prior to a collector.

Technologies reducing the amount of traffic data rather than targeting all VoIP sessions for QoS measurement are also required. One is the on-demand QoS measurement method that selects the VoIP session on the basis of the SIP-URI of the caller or callee, if needed. The method allows operators to monitor the call of a specific customer who complains of service quality degradation. Another is the random sampling of VoIP sessions to select the VoIP session to measure. The method allows operators to monitor the service quality condition with an overview of the situation.

- Requirement #3: Correlation between SIP signaling and RTP sessions

There are two approaches to correlating the SIP session and RTP session. One is a method correlating SIP session data with RTP session data after capturing SIP packets and RTP packets, respectively, at different devices. The other is a method triggering QoS measurement of an RTP session after capturing SIP packets and extracting the SDP information. From the viewpoint of scalability and using a commercial router's capability, the latter method is suitable for a large-scale network and any network topology.

We can consider the following scenario by using the ACL-based metering function. The router close to the SIP proxy captures the SIP packets by ACL filtering on the basis of the port number and exports the SIP packet data containing SDP data. Then the ACL filter condition associated with the RTP session is configured on appropriate routers. In the case of different paths being taken in the SIP signaling and RTP session, we need to consider a method determining which routers that RTP packets pass through. To cope with this, the system needs to choose routers close to the caller/callee by looking up the routing information on the basis of the destination/source IP addresses, respectively.

6 VoIP Measurement Logical Architecture

We focused on the hierarchical structure to increase the scalability with regard to components including the knowledge plane, control plane, and monitor plane. The structure can adapt to the size of network domain by adjusting the number of devices on monitor or control plane. As the representative hierarchic structure model, the knowledge plane architecture taking on the concrete cognitive system [16] is famous. System architecture for access network monitoring using the knowledge plane was proposed in Ref. [17]. Our proposed architecture, shown in Fig.1, can be adapted to SIP relevant services, such as VoIP and Video services.

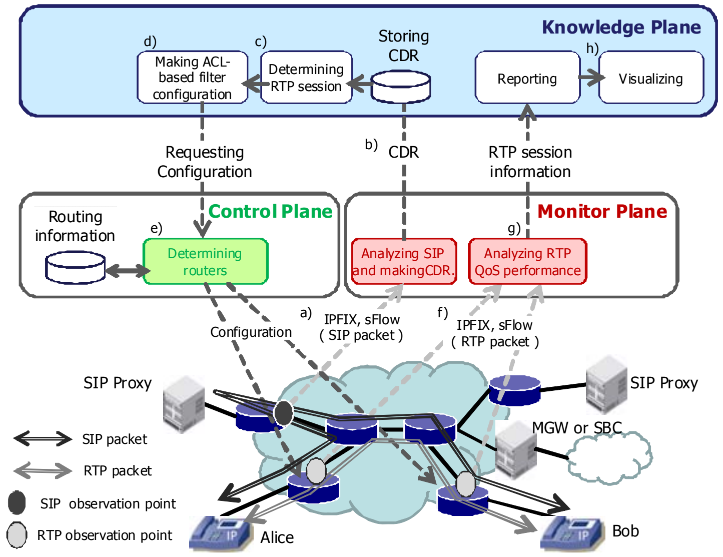


Fig. 1. VoIP measurement logical architecture

The system can operate as follows.

a) The router that enables the ACL-based metering function captures SIP packets and then exports the relevant packet report including the SDP information as the “ipHeaderPacketSection” field defined in Ref. [18].

b) The monitor plane parses the relevant “ipHeaderPacketSection” field. After detecting the established status of the SIP session, the monitor plane makes a CDR and exports it to the knowledge plane. The CDR consists of the following fields: timestamp, Call-ID, caller SIP-URI, callee SIP-URI, media type, and relevant media information (source/destination IP address, source/destination port number, and protocol).

c) The knowledge plane determines the VoIP session to be measured by random sampling of VoIP sessions or on-demand QoS measurement. In the case of on-demand QoS measurement, the network operators need to configure the target SIP-URI in advance.

d) After determining the RTP session to be measured, the knowledge plane configures the ACL-based filter to capture the RTP packets.

e) The control plane determines the target router taking on the configuration.

f) The router that enables the ACL-based metering function captures RTP packets and then exports the relevant packet report including the RTP header information as “ipHeaderPacketSection”.

g) The monitor plane parses the relevant “ipHeaderPacketSection” field, computes some QoS metrics, and then reports the data to the knowledge plane.

h) The knowledge plane reports to the operator when QoS performance degradation occurs. The knowledge plane allows network operators to investigate the failure point, report the VoIP session, and visualize the results of the investigation.

According to SIP signaling, the system operation procedure is shown in Fig.2.

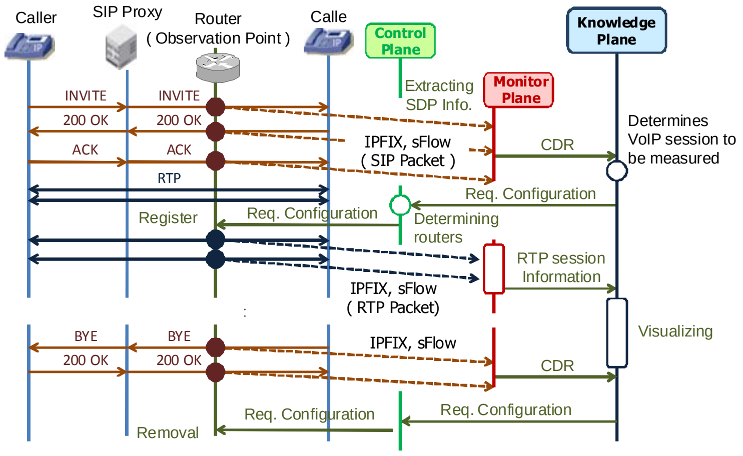


Fig. 2. System operation procedure according to SIP signaling

7 Implementation for VoIP Measurement System

7.1 Development of Prototype System

In accordance with the logical architecture, we developed the prototype system and evaluated it. We developed prototype routers having the ACL-based metering function based on the commercial router platform. The function can choose the maximum length of the "ipHeaderPacketSection" and determine to which device traffic data are exported on the basis of the ACL-based filtering. In the case of monitoring SIP packets, the maximum length would be set to about 700 bytes including the SDP information; in the case of monitoring RTP packets, the maximum length would be set to about 50 bytes including the RTP header information. To minimize the development time period, we utilized the ACL-based mirroring function and sFlow agent.

We also developed the mediator hosting the monitor/control plane functionalities. The mediator has the following capabilities.

a) Web console for ACL-based filtering configuration
b) Receiving the ACL-based filtering configuration file presenting the XML schema
c) Selecting the application module to be monitored, including the SIP module or RTP module, in accordance with the ACL-based filter configuration
d) Regarding the SIP module, parsing the SIP packets and making a CDR
e) Regarding the RTP module, creating QoS metrics and exporting traffic data.

Network operators can easily capture the SIP packets by setting the port number on the Web console. Other management nodes, such as the knowledge plane, can dynamically change the configuration by sending the configuration file. The configuration data model is considered on the basis of the IPFIX configuration, but it has the following additional capabilities.

a) Compatibility for ACL-based filtering
b) Configuration data model including the parameters on both the mediator and exporter sides

The following figure shows an example of configuration data model.

```
<AccessList operation="">
  <FlowId>RTP PACKET FILTER SAMPLE</FlowId>

<FilterMatch id="1">
    <ActionProcess actionType="Exporter" id="Exporting"/>
    <ActionProcess actionType="Mediator" id="RTP_module"/>

    <infoElementValue>
      <sourceIPv4Address>192.168.0.0/32</sourceIPv4Address>
      <destinationIPv4Address> 172.16.0.1</destinationIPv4Address>
      <protocolIdentifier>16</protocolIdentifier>
      <sourceTransportPort>1024</sourceTransportPort>
      <destinationTransportPort>2048</destinationTransportPort>
    </infoElementValue>

  </FilterMatch>

</AccessList>
```

```
<ActionProcess id="Exporting">
   <samplingInterval>1</samplingInterval>
   <collectorIPaddress>10.0.0.1</collectorIPaddress>
   <collectorPort>9996</collectorPort>
</ActionProcess>

<ActionProcess id="RTP_module">
   <transmitApplication>rtp</transmitApplication>
</ActionProcess>
```

Fig. 3. Example of IPFIX configuration

In Fig. 3, the configuration example on the left for ACL-based filtering defines action process identifier on both of exporter and mediator as well as one filtering list including packet property parameters. The "ActionProcess" tag fields contains a reference to the configuration example on the right, which are shown in detail how to process the associated packet or traffic data on both of the exporter and mediator sides. With the data model, the monitor plane and control plane in the mediator can share the configuration information, and the mediator can correlate the received traffic data with configuration information and then select the appropriate module. The

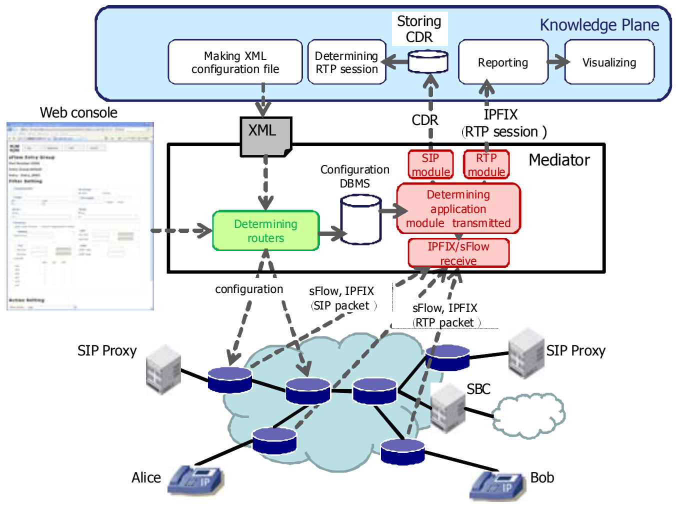


Fig. 4. VoIP measurement prototype system architecture

0	15	16 31
Set Id = 2		Length
Template Id = 258		Field Count = 6
0	Bytes Id = 1	Field Length =4
0	Packets Id = 2	Field Length = 4
0	protocolIdentifier Id = 4	Field Length = 4
0	sourceTransportPort Id = 1	Field Length = 2
0	ingressInterface Id = 10	Field Length = 4
0	destinationTransportPort Id = 11	Field Length = 2
0	sourceIPv6Address Id = 27	Field Length = 16
0	destinationIPv6Address Id = 28	Field Length = 16
0	ipVersion Id = 60	Field Length = 1
0	exporterIPv4Address Id = 130	Field Length = 4
0	droppedPacketDeltaCount = 133	Field Length = 4
0	flowStartMilliSeconds = 152	Field Length = 4
0	flowEndMilliSeconds = 153	Field Length = 4
1	RTP Interval (Ave.)	Field Length = 4
Enterprise Number		
1	RTP Interval (Max)	Field Length = 4
Enterprise Number		
1	RTP Interval (Min)	Field Length = 4
Enterprise Number		

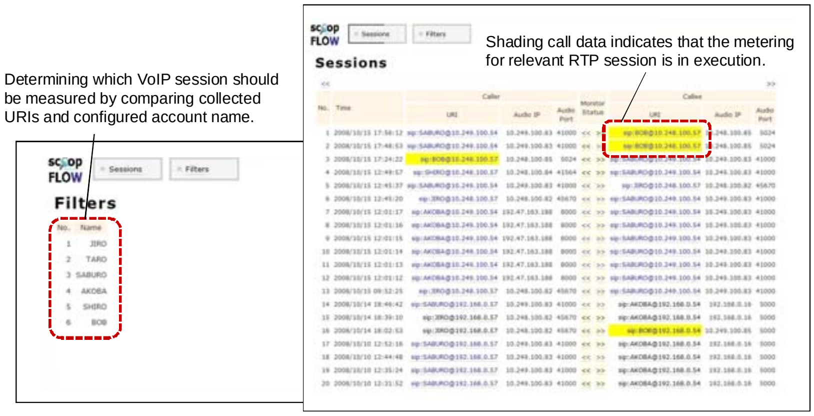
Fig. 5. Template data structure related to RTP sessions

prototype system structure is shown in Fig. 4, and the template data structure transmitting from the RTP module to the knowledge plane is shown in Fig. 5. This template structure includes the packet loss field and several fields related to RTP packet interval time. In the future, it will include one way delay.

The upper layer device indicating the knowledge plane has the following capabilities.

a) Visualizing the CDR
b) Setting the customer account for QoS measurement
c) Determining the RTP session for QoS measurement
d) Creating the XML filtering configuration
e) Visualizing the result of QoS measurement for RTP sessions

The web console showing the CDR is in Fig. 6.


Determining which VoIP session should be measured by comparing collected URIs and configured account name.
Shading call data indicates that the metering for relevant RTP session is in execution.
scoop FLOW
Sessions
Filters
<<
>>

Filters

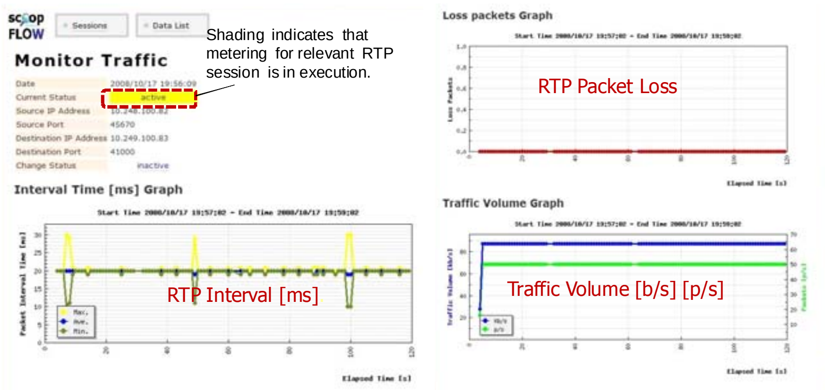
No.	Name
1	JIRO
2	TARO
3	SABURO
4	AKOBA
5	SHIRO
6	BOB

Sessions

No.	Time	Caller			Monitor Status	Callee		
		URI	Audio IP	Audio Port		URI	Audio IP	Audio Port
1	2008/10/15 17:56:12	sip:SABURO@10.249.100.54	10.249.100.83	41000	<< >>	sip:BOB@10.248.100.57	10.248.100.85	5024
2	2008/10/15 17:48:53	sip:SABURO@10.249.100.54	10.249.100.83	41000	<< >>	sip:BOB@10.248.100.57	10.248.100.85	5024
3	2008/10/15 17:24:22	sip:BOB@10.248.100.57	10.248.100.85	5024	<< >>	sip:SABURO@10.249.100.54	10.249.100.83	41000
4	2008/10/15 12:49:57	sip:SHIRO@10.248.100.57	10.248.100.84	41564	<< >>	sip:SABURO@10.249.100.54	10.249.100.83	41000
5	2008/10/15 12:45:37	sip:SABURO@10.249.100.54	10.249.100.83	41000	<< >>	sip:JIRO@10.248.100.57	10.248.100.82	45670
6	2008/10/15 12:45:20	sip:JIRO@10.248.100.57	10.248.100.82	45670	<< >>	sip:SABURO@10.249.100.54	10.249.100.83	41000
7	2008/10/15 12:01:17	sip:AKOBA@10.249.100.54	192.47.163.188	8000	<< >>	sip:SABURO@10.249.100.54	10.249.100.83	41000
8	2008/10/15 12:01:16	sip:AKOBA@10.249.100.54	192.47.163.188	8000	<< >>	sip:SABURO@10.249.100.54	10.249.100.83	41000
9	2008/10/15 12:01:15	sip:AKOBA@10.249.100.54	192.47.163.188	8000	<< >>	sip:SABURO@10.249.100.54	10.249.100.83	41000
10	2008/10/15 12:01:14	sip:AKOBA@10.249.100.54	192.47.163.188	8000	<< >>	sip:SABURO@10.249.100.54	10.249.100.83	41000
11	2008/10/15 12:01:13	sip:AKOBA@10.249.100.54	192.47.163.188	8000	<< >>	sip:SABURO@10.249.100.54	10.249.100.83	41000
12	2008/10/15 12:01:12	sip:AKOBA@10.249.100.54	192.47.163.188	8000	<< >>	sip:SABURO@10.249.100.54	10.249.100.83	41000
13	2008/10/15 09:52:25	sip:JIRO@10.248.100.57	10.248.100.82	45670	<< >>	sip:SABURO@10.249.100.54	10.249.100.83	41000
14	2008/10/14 18:46:42	sip:SABURO@192.168.0.57	10.249.100.83	41000	<< >>	sip:AKOBA@192.168.0.54	192.168.0.16	5000
15	2008/10/14 18:39:10	sip:JIRO@192.168.0.57	10.248.100.82	45670	<< >>	sip:AKOBA@192.168.0.54	192.168.0.16	5000
16	2008/10/14 18:02:53	sip:JIRO@192.168.0.57	10.248.100.82	45670	<< >>	sip:BOB@192.168.0.54	10.249.100.85	5000
17	2008/10/10 12:52:16	sip:SABURO@192.168.0.57	10.249.100.83	41000	<< >>	sip:AKOBA@192.168.0.54	192.168.0.16	5000
18	2008/10/10 12:44:48	sip:SABURO@192.168.0.57	10.249.100.83	41000	<< >>	sip:AKOBA@192.168.0.54	192.168.0.16	5000
19	2008/10/10 12:35:24	sip:SABURO@192.168.0.57	10.249.100.83	41000	<< >>	sip:AKOBA@192.168.0.54	192.168.0.16	5000
20	2008/10/10 12:21:52	sip:SABURO@192.168.0.57	10.249.100.83	41000	<< >>	sip:AKOBA@192.168.0.54	192.168.0.16	5000

Fig. 6. Web console for stored CDR data

When the network operators want to measure the VoIP session of a specific customer and click on the associated SIP-URI, the measurement data in Fig. 7 appear after about 10 seconds.


scoop FLOW
Sessions
Data List
Shading indicates that metering for relevant RTP session is in execution.
Interval Time [ms] Graph
Start Time 2008/10/17 19:57:02 - End Time 2008/10/17 19:59:02
Packet Interval Time [ms]
Max.
Ave.
Min.
RTP Interval [ms]
Elapsed Time [s]
Loss packets Graph
Start Time 2008/10/17 19:57:02 - End Time 2008/10/17 19:59:02
Loss Packets
RTP Packet Loss
Elapsed Time [s]
Traffic Volume Graph
Start Time 2008/10/17 19:57:02 - End Time 2008/10/17 19:59:02
Traffic Volume [kb/s]
Packets [p/s]
kb/s
p/s
Traffic Volume [b/s] [p/s]
Elapsed Time [s]

Monitor Traffic

Date	2008/10/17 19:56:09
Current Status	active
Source IP Address	10.248.100.82
Source Port	45670
Destination IP Address	10.249.100.83
Destination Port	41000
Change Status	inactive

Fig. 7. Web console for result of QoS measurement

7.2 Experimental Results

We evaluated the feasibility of the prototype system.

a) Time between VoIP session initiation and capture of RTP packets

The measurement system configures the ACL-based filtering on routers after detecting the establishment of a SIP session. With regard to the router's configuration, there generally is an interval of time for this filter to become available. Our system would require a shorter time. We evaluated the time between the VoIP session establishment time and the capture of relevant RTP packets. The result depends on the number of filtering lists configured at the same time. The result of the evaluation is shown in Fig. 8. The x-axis shows the number of ACL filtering lists as a variable parameter.

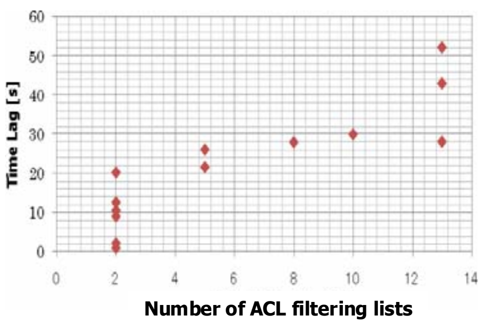


Fig. 8. Interval between time of VoIP establishment and time of initial RTP packet capture

The results show that the time for one session is shorter than one for multiple sessions. Although the time for 2 sessions takes 10–20 seconds, this was determined to cover almost all VoIP sessions by comparing it with the average call duration (ACD) of 30–400 seconds described in Ref. [18]. However, ACD depends on the time of day and the situation. The system requires the improvement of the configuration scheme to cover short calls.

b) Degradation detection for route change

When the interface on a router goes down and a route change occurs, we can see the result of RTP session QoS measurement metrics as follows.

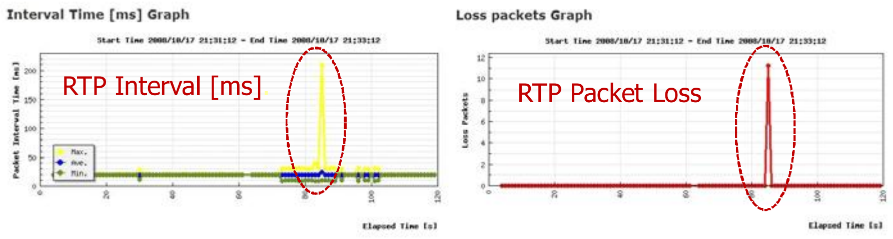


Fig. 9. Web console shows degradation detection in case of route flap

8 Conclusion

The architecture of a VoIP measurement system using an ACL-based metering function, mediator, and configuration was proposed. The logical architecture can be adopted to any SIP-based service and IPTV in any network topology. We illustrate the scalable architecture model to provide easy operation in a cost-effective way. The proposed configuration data model improves the limitations of the IPFIX configuration data model and is suitable for an exporter-mediator-collector structure. Regarding QoS monitoring, the mediation functionalities are the key components, and the combination of mediation functionalities and configuration expands the flexibility of traffic monitoring. Thus, IPFIX is expected to develop further supporting the monitoring of specific application layers by using the mediation and configuration.

We also evaluated the feasibility of a prototype of the developed system. This architecture can be adapted to any packet-based exporting protocol, such as IPFIX, PSAMP, or sFlow, and to traffic measurement for SIP-based session control services. In the future, we will try to measure the one way delay by correlating a pair of packet-based traffic data using this architecture.

Acknowledgments. This study was supported by the Ministry of Internal Affairs and Communications of Japan.

References

[1] Claise, B., et al.: Cisco Systems NetFlow Services Export Version 9. RFC 3954 (October 2004)

[2] Phaal, P., et al.: InMon Corporation's sFlow. RFC 3176 (September 2001)

[3] Claise, B., et al.: Specification of the IPFIX Protocol for the Exchange of IP Traffic Flow Information, RFC 5101 (January 2008)

[4] Zseby, T., et al.: Sampling and Filtering Techniques for IP Packet Selection. RFC 5475 (March 2009)

[5] Zseby, T., et al.: IP Flow Information Export (IPFIX) Applicability. RFC 5472 (March 2009)

[6] Lee, B., et al.: End-to-End Flow Monitoring with IPFIX. In: Ata, S., Hong, C.S. (eds.) APNOMS 2007. LNCS, vol. 4773, pp. 225–234. Springer, Heidelberg (2007)

[7] Deri, L.: Open Source VoIP Traffic Monitoring, http://luca.ntop.org/VoIP.pdf

[8] Anderson, S.: SIPFIX Using IPFIX for VoIP monitoring. In: IRTF-NMRG Workshop on Netflow/IPFIX Usage in Network Management Munich (October 2008)

[9] Muenz, G., Claise, B.: Configuration Data Model for IPFIX and PSAMP (February 2008), <draft-muenz-ipfix-configuration-04>

[10] Kobayashi, A., Toyama, K.: Method of Measuring VoIP Traffic Fluctuation with Selective sFlow. In: SAINT Workshops (2007)

[11] Kobayashi, A., et al.:"IPFIX Mediation: Problem Statement (April 2009), <draft-ietf-ipfix-mediators-problem-statement-03>

[12] Kobayashi, A., Claise, B., Nishida, H.: IPFIX Mediation: Framework (February 2009), <draft-ietf-ipfix-mediators-framework-02>

[13] Lindh, T., Roos, E.: Monitoring of SIP-Based Communication Using Signaling Information for Performance Measurements. In: Proceedings of the International Conference on Internet Surveillance and Protection (August 2006)

[14] BROCADE, ACL-based sFlow, http://www.brocade.com/

[15] Cisco, NetFlow Input Filters, http://www.cisco.com/

[16] Clark, D., et al.: A Knowledge Plane for the Internet. In: ACM SIGCOMM 2003, Karlsruhe, Germany (August 2003)

[17] De Vleeschauwer, B.: On the Enhancement of QoE for IPTV Services through Knowledge Plane Deployment. Broadband Europe (December 2006)

[18] He, Q(A.): Analyzing the Characteristics of VoIP Traffic. University of Saskatchewan (2007)

One-Against-All Methodology for Features Selection and Classification of Internet Applications

José Everardo Bessa Maia and Raimir Holanda Filho

Department of Statistics and Computing, State Univ. of Ceará – UECE
Master's Course in Applied Computer Sciences, Univ. of Fortaleza - UNIFOR
jmaia@larces.uece.br, raimir@unifor.br

Abstract. Traffic classification by Internet applications, even on off-line mode, can be interesting for many applications such as attack identification, QoS prioritization, network capacity planning and also computer forensic tools. Into the classification problem context is well-known the fact that a higher number of discriminators not necessarily will increase the discrimination power. This work investigates a methodology for features selection and Internet traffic classification in which the problem to classify one among *M* classes is split in *M* one-against-all binary classification problems, with each binary problem adopting eventually a set of different discriminators. Different combinations of discriminators selection methods, classification methods and decision algorithms could be embedded into the methodology. To investigate the performance of this methodology we have used the Naïve Bayes classifier to select the set of discriminators and for classification. The proposed method intends to reduce the total number of different discriminators used into the classification problem. The methodology was tested for classification of traffic flows and the experimental results showed that we can reduce significantly the number of discriminators per class sustaining the same accuracy level.

Keywords: Traffic classification; features selection; statistical discriminators.

1 Introduction

Internet traffic classification based on application type is one of the most relevant problems on computer networks. Problems of workload characterization and modeling, capacity planning, route provisioning and network management benefit from the accurate classification of network traffic.

Specifically into network management, activities of traffic classification can be used to: flow priority, traffic policing and diagnostic monitoring. For instance, a network manager can analyze the traffic changing as consequence of applications such as peer-to-peer (P2P) file sharing applications and evaluate the impact of these applications over a network infrastructure.

In the last years, many techniques have been proposed to Internet traffic classification based on application type. The most traditional method to Internet traffic classification has used the application mapping into well-known port numbers [1]. Despite the large utilization in the past, recent studies have showed that port-based identification of network traffic is ineffective [2], [3]. For P2P traffic, for instance, this inefficiency is very clear as consequence of the use of dynamic port numbers.

A second approach used to classify applications was based on signature matching [4]. In this case, the packet payload is opened and data are read [5]. However, this technique has two serious drawbacks. The first is the privacy problem, in the sense that the analysis is not restricted to the TCP/IP packet headers. The second problem arises as consequence that significant portions of the Internet traffic is encrypted, becoming impossible the payload analysis.

Heuristics applied to the transport layer also have been used for traffic classification [5]. Transport layer information has been used to overcome the limits of classification based on port numbers and payload. However, the definition of heuristics for each type of application is a hard work and the results obtained with this technique sometimes are not satisfactory.

In [6], the author presents a relationship between two traffic classes and their statistical properties, describing the bytes and flow packets distribution for a specific number of applications. Recently, statistical methods combined with machine learning techniques have been used for traffic classification [7]. These methods have shown that are not affected by the problems described above and present very expressive results to a large number of applications.

Statistical discriminators and cluster analysis techniques were used in [8], [9], [10], [28] to identify specific applications like P2P and Attacks. The obtained results show that traffic data analysis based on statistical methods produces a level of accuracy in the classification similar of other approaches with higher computational complexity.

Recently, Kim et al. [20] into a large comparative analysis showed that a Robust Support Vector Machine (SVM) classifier overcome many methods cited earlier in terms of accuracy and precision and not less important, showed that for classification based on flows features, the TCP port continues being an important feature. This recent work also show that a definitive solution for the Internet traffic classification is not yet dominated and some time will be wasted to overcome it.

Finally, the authors in [19] evaluated a deep-packet inspection method as a strategy of pipeline classification which combine an observation window and a machine learning method (C4.5) that shows a good performance on online, real-time classification of applications.

Our proposed work investigates a methodology of Internet traffic classification in which the problem of classification of one class among *M* classes is broke it up into *M* binary classification problems of type one-against-all [23], [24], with the adoption of a set of eventually different discriminators by each binary problem. Different combinations of discriminators selection methods, classification methods and decision algorithms could be embedded into the methodology. To investigate the performance of this methodology is used Naïve Bayes classifier [21], [22] to select the set of discriminators and for classification. A wrapped discriminator selection approach is used. The paper [25] shows that the division of a multi-classes problem in pairwise Naïve Bayes classifiers has the same discrimination power that a multi-classes Naïve Bayes classifier and demonstrated, using a simple example, that the one-against-all approach can produce different outcomes.

The methodology was tested in the classification of traffic flows and the experimental results show that we can obtain a significant reduction in the number of discriminators without a proportional loss of discrimination power.

The remainder of the paper is structured as follows. In section 2, we present the proposed methodology and also we describe the sequence of steps to be executed. In section 3, we explain, for the training and validation phases, the obtained results and the accuracy verification in the classification to validate our proposal. At last, in section 4 we present the most important conclusions of our work.

2 Methodology

Previous works, that applied a statistical approach to the Internet traffic classification, tried to solve the following problem: how to select the minor set of statistical features to classify the Internet traffic. Our work follow a different way based on the statistical approach. First of all, we start from the hypothesis that the best set of discriminators used to classify each traffic class does not match the best set of discriminators to classify all the traffic classes simultaneously. After that, we try to obtain for each traffic class a set of features that better identify that class against all others.

The classifier training consists then, in the selection of a set of discriminators to each one of the binary problems one-against-all and the calculation of the weights to be used into the decision process in the cases of samples are assigned to more than one class. When a sample is not classified into one of the classes, a randomly selection occurs or the sample are not classified. The next paragraphs describe the implementation of the methodology investigated in our work.

After obtained a number of variables in a set, for each binary problem, the first step consists in the calculation of the F-ratio of each variable into the set and their sorting. The F-ratio is an efficient measure for the separation between groups. The F-ratio uses two estimates, the mean variance of inter-group elements S_b^2 and the mean variance of intra-group elements S_w^2 , defined as follows:

$$F-ratio = \frac{S_b^2}{S_w^2} = \frac{n\dfrac{\sum_j (\bar{x}_j - \bar{\bar{x}})^2}{(k-1)}}{\dfrac{1}{k(n-1)}\left[\sum (x_i - \bar{x}_1)^2 + \ldots + \sum (x_i - \bar{x}_k)^2\right]} \qquad (1)$$

with x_i being the sample, $\bar{x}$ the sample mean, $\bar{\bar{x}}$ the average sample mean, *k* the number of groups and *n* the number of group samples.

With the variables sorted into a decreasing order, the discriminators selection runs as follow: for each one of the *M* binary problems is applied a supervised classification to the set of training data using the Naïve Bayes classifier and the rightness percentage is registered. The procedure is repeated aggregating a new variable following the decreasing order defined by F-ratio in a cumulative way until the rightness percentage profits are no more significant or a pre-defined rightness percentage are reached.

The final rightness percentages obtained by the training set will be used as a decision criterion in the cases that a sample is assigned to more than one class.

In the classification phase the procedure consists in take a new flow, classify it according to the Naïve Bayes classifier of each binary classification problem using the related discriminators of each class. The following three situations can occur as a final result. First, a flow was classified into a unique class. In this case, the class is defined. Second, the procedure assigned the flow to more than one class. In this case, the decision is weighed by the rightness percentage obtained in the training phase. At last, in the case of a sample is not assigned to any class, a class is chosen randomly based on the frequency of the classes into the training sample or, whether the application supports the alternative, the sample is not classified.

2.1 Naïve Bayes Classifier

In this work, we have used the Naïve Bayes technique [27]. Consider a collection of flows $x = (x_1, \dots, x_n)$, where each flow x_i is described by *m* discriminators $\{d_1^{(i)}, \dots, d_m^{(i)}\}$ that can take either numerical or discrete values. In the context of the Internet traffic, $d_j^{(i)}$ is a discriminator of flow x_i, for example it may represent the mean inter-arrival time of packets in the flow x_i. In this paper, flows x_i belong to exactly one of the mutually-exclusive classes. The supervised Bayesian classification problem deals with building a statistical model that describes each class based on some training data, and where each new flow y receives a probability of getting classified into a particular class according to the Bayes rule below,

$$p(c_j\,|y) = \frac{p(c_j)\,f(y|c_j)}{\sum_{c_j} p(c_j)\,f(y|c_j)} \tag{2}$$

where $p(c_j)$ denotes the probability of obtaining class c_j independently of the observed data, $f(y \mid c_j)$ is the distribution function (or the probability of y given c_j) and the denominator acts as a normalizing constant. The Naïve Bayes technique that is considered in this paper assumes the independence of discriminators $d_1, \dots, d_m$ as well as the simple Gaussian behavior of them.

3 Results

Traces of IP traffic were used to validate the proposed methodology. The trace contains 10 different applications. Moore [11] showed that, among the 10 classes, the P2P and Attack classes are the hardest to classify. Results presented in [8], [9] e [10] show that the P2P traffic confound mainly with the WWW traffic, the Attack traffic confound with the MAIL traffic and that P2P e ATTACK have a large intersection. The methodology was evaluated using three classes: P2P, Attack and Others. In this description, Others represents the following joint classes: Games, Ftp, Web, Multimedia, Mail, Interactive, Services and Database.

3.1 Traces Description and Sampling

In this work were used pre-processed traces that were available to the scientific community and were described in [17] and [11]. The method of data collection is described in [18], having been used the data available in [7] and [11]. These data were collected from a network with 1,000 users connected through a full-duplex Gigabit Ethernet Internet connection and based on a 24-hour period. Ten archives were

Table 1. Flows by trace

(TRACE 1 TO TRACE 10)

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
ATTACK	122	19	41	324	122	134	89	129	367	446
P2P	339	94	100	114	75	94	116	289	249	624
Others	24434	23707	22832	22171	21573	19113	55566	55017	65600	63943
Total	24312	23535	22655	21791	21413	19341	55771	55435	66216	65013

created, each one related to a period of 1,680 seconds (28 minutes) and available to the scientific community. Table 1 shows the amount of Attack, P2P and Others flows in each one of the ten traces.

We used the traces described above in which the tasks of conversion of packet sequence to flows and the manual identification of the traces were fulfilled. In [11], a set of statistics was generated for each flow. In [17], this set of statistics are called discriminators and the authors generated 249 discriminators that include simple statistics about the size and packet delay and TCP protocol information, such as *Syn* and *Ack* packet counters.

The methodology was evaluated using a cross-validation approach based on an equally balanced sample. All flows with missing data were deleted and neither normalization or filtering of outliers was done. We observed that into the 10 traces, the classes with minor number of flows are the P2P class with 2094 flows and the attack class with 1782 flows. Hence, we adopted a random sample of 1782 flows and ran 40 times. With this sample defined, were separated randomly 1425 (80%) flows from each class to perform the features selection and training and 357 (20%) flows to perform the validation phase. The results are presented in terms of mean values and standard deviation. The data were manipulated using WEKA [21] and the algorithms used a script language into the MATLAB [22].

3.2 Training

The real beginning of our work are the 249 discriminators that were calculated and tabulated. In accordance with our methodology, the discriminators selection was based on those variables that presented large F-Ratio values. The tables 2, 3 and 4 present 10 discriminator candidate variables based on large F-Ratio values used to identify the P2P, Attack and others traffics respectively. The table does not include the TCP port feature which is used for all cases. Also, in the same tables, we show the accumulated hit ratio for the three classes as consequence of the discriminator selection process when using that features with the TCP port. The set of variables that discriminates the P2P, Attack and Others is also selected from tables 2, 3 and 4. Variables are inserted to this set when they increase significantly the discrimination power. The number of discriminators for the three classes cited above are 5, 4 and 3.

Table 2. Candidate discriminator variables (P2P)

Candidate Variable	F-Ratio	Acum Hit Ratio
Third quartile of packet inter-arrival time. (server→client)	249,75	45,85%
Maximum idle time, calculated as the maximum time between consecutive packets seen in the direction. (client→server)	223,44	50,28%
FFT of packet IAT (arctan of the top-ten frequencies ranked by the magnitude of their contribution). (server→client)	172,84	57,52%
Third quartile of total bytes in IP packet. (client→server)	160,47	88,37%
The total number of bytes sent in the initial window. (client→server)	153,21	98,43%
Maximum of bytes in (Ethernet) packet.	124,33	-
The average throughput calculated as the unique bytes sent divided by the elapsed time. (client→server)	100,84	-
Median of total bytes in IP packet. (client→server)	79,53	-
The time spent idle (where idle time is the accumulation of all periods of 2 seconds or greater when no packet was seen in either direction).	76,13	-
Median of bytes in (Ethernet) packet.	62,62	-

Table 3. Candidate discriminator variables (Attack)

Candidate Variable	F-Ratio	Acum Hit Ratio
The minimum segment size observed during the life-time of the connection. (client→server)	826,21	74,06%
The total number of bytes sent in the initial window. (client→server)	444,72	88,26%
The average segment size observed during the lifetime of the connection calculated as the value reported in the actual data bytes field divided by the actual data pkts reported. (server→client)	328,42	94,59%
The total number of bytes sent in the initial window. (server→client)	263,71	98,26%
Maximum of bytes in (Ethernet) packet.	163,40	98,58%
Minimum number of total bytes in IP packet. (client→server)	112,72	-
Median of total bytes in IP packet. (client→server)	54,79	-
Variance of bytes in (Ethernet) packet. (server→client)	38,19	-
The average throughput calculated as the unique bytes sent divided by the elapsed time. (client→server)	17,97	-
The count of all the packets seen with the PUSH bit set in the TCP header. (client→server)	14,85	-

Using the training data from each class and the selected features in the previous phase, Naïve Bayes classifiers are trained for each binary problem. After that, to observe the discrimination power of these binary classifiers, the same training data were used in a classification process.

Table 4. Candidate discriminator variables (Others)

Candidate Variable	F-Ratio	Acum Hit Ratio
Third quartile of packet inter-arrival time.	233,87	64,35%
The minimum segment size observed during the life-time of the connection. (client→server)	217,36	77,95%
The total number of bytes sent in the initial window. (server→client)	206,85	84,35%
The average segment size observed during the lifetime of the connection calculated as the value reported in the actual data bytes field divided by the actual data pkts reported. (server→client)	187,39	84,85%
Maximum idle time, calculated as the maximum time between consecutive packets seen in the direction. (client→server)	104,53	-
Median of total bytes in IP packet. (client→server)	98,18	-
Third quartile of total bytes in IP packet. (client→server)	95,35	-
The average throughput calculated as the unique bytes sent divided by the elapsed time. (client→server)	81,59	-
Minimum number of total bytes in IP packet. (client→server)	78,48	-
FFT of packet IAT (arctan of the top-ten frequencies ranked by the magnitude of their contribution). (server→client)	64,99	-

The tables 5, 6 and 7 show the confusion tables for mean values (av) and standard deviation (sd) for each one of the binary classification problems for a set of 40 runs. Only integer values were registered in these tables. With the purpose to establish control parameters, the classification process was applied to training data and the results are presented on table 8. From these tables we can obtain the following values to the training phase: false positive ratio of 26.49% ((382 + 92) / 1789) and false negative ratio of 7.71% ((82 + 28) / 1425) to P2P; false positive ratio of 17.23% ((82 + 121) /1178), and false negative ratio of 31.57% ((382 + 68) / 1425) to Attack and false positive ratio of 7.33% ((68 + 28) / 1308), and false negative ratio of 14.94% ((92 + 121) / 1425) to Others. In this procedure, 1.23% of all flows were assigned to more than a class and 2.54% of all flows were not assigned to any class. In both cases, the flows were classified in the class with higher pertinence probability.

Table 5. Confusion Table for the binary problem P2P (Training Phase)

	P2P		Attack and Others	Flows Total
	AV	SD		
P2P	1315	231	110	1425
Attack and Others	474	94	2376	2850
Total	1789	-	2486	4275

Table 6. Confusion Table for the binary problem Attack (Training Phase)

	Attack		P2P and Others	Flows Total
	AV	SD		
Attack	975	272	450	1425
P2P and Others	203	96	2647	2850
Total	1178	-	3097	4275

Table 7. Confusion Table for the binary problem Others (Training Phase)

	Others		P2P and Attack	Flows Total
	AV	SD		
Others	1212	314	213	1425
P2P and Attack	96	108	2754	2850
Total	1308	-	2967	4275

Table 8. Confusion Table for Final Classification (Training Phase)

	P2P		Attack		Others		Flows Total
	AV	SD	AV	SD	AV	SD	
P2P	1315	231	82	8	28	3	1425
Attack	382	58	975	272	68	14	1425
Others	92	24	121	19	1212	314	1425
Total	1789	-	1178	-	1308	-	4275

3.3 Validation

The validation was carried out submitting the classification to a different set of the training phase. The confusion tables to this phase, for the binaries problems and for the final classification, are showed in the tables 9, 10, 11 and 12. From these tables we obtained the following values: false positive ratio of 33.05% ((140 + 18) / 478) and false negative ratio of 10.36% ((28 + 9) /357) to P2P; false positive ratio of 20.55% ((28 + 24) / 253) and false negative ratio of 43.69% ((140 + 16) / 357) to attack; and false positive ratio of 7.35% ((9 + 16) / 340) and false negative ratio of 11.76% ((18 + 24) / 357) to Others.

In order to calculate the results we defined the following parameters: *trust*, *accuracy* and *application accuracy.* These parameters are defined as follow:

$$Trust = \frac{num.of\ \ flows\ correctly\ classified\ into\ application\ clusters}{total\ \ flows\ into\ application\ clusters} \tag{3}$$

$$Accuracy = \frac{num.of\ \ flows\ correctly\ classified\ into\ clusters}{total\ of\ \ trace\ \ flows} \tag{4}$$

$$Application\ accuracy = \frac{num.of\ \ application\ \ flows\ correctly\ classified\ into\ clusters}{total\ of\ \ application\ \ flows\ into\ trace} \tag{5}$$

The final results to these parameters, to the validation phase, obtained from table 12 are: P2P trust = 66.95%; Attack trust = 79.45%; Others trust = 92.65%; mean accuracy = 78.05%, P2P Accuracy = 89.63%, Attack Accuracy = 56.30% and Others Accuracy = 88.23%. In the validation phase, a mean of 6.75% of all flows were assigned to more than a class and 3.06% of all flows were not assigned to any class. In both cases, the flows were classified in the class with higher pertinence probability.

Table 9. Confusion Table for the binary problem P2P (Validation Phase)

	P2P		**Attack and Others**	**Flows Total**
	AV	**SD**		
P2P	320	28	37	357
Atack and Others	158	44	556	714
Total	478	-	593	1071

Table 10. Confusion Table for the binary problem Attack (Validation Phase)

	Attack		**P2P and Others**	**Flows Total**
	AV	**SD**		
Attack	201	23	156	357
P2P and Others	52	16	662	714
Total	253	-	818	1071

Table 11. Confusion Table for the binary problem Others (Validation Phase)

	Others		**P2P and Attack**	**Flows Total**
	AV	**SD**		
Others	315	33	42	357
P2P and Attack	25	26	689	714
Total	340	-	731	1071

Table 12. Confusion Table for Final Classification (Validation Phase)

	P2P		Attack		Others		Flows Total
	AV	**SD**	**AV**	**SD**	**AV**	**SD**	
P2P	320	28	28	14	9	3	357
Attack	140	25	201	23	16	2	357
Others	18	4	24	5	315	26	357
Total	478	-	253	-	340	-	1071

3.4 Discussion

The reduction of the number of variables in each binary problem produced a faster classifier in the training as also in the classification phase if compared to a single Naïve Bayes multi-class classifier, since, for example, the calculation of the Mahalanobis distances, four times, with matrices of order 4 is more economic than the calculation, once, with a matrix of order 16. There was not, in any case, problems of poor conditioning of matrices. It is not correct to say, however, that the total number of features to be extracted from the sample will be reduced in all cases. This fact will depend of a specific problem.

The larger amount of double classification cases are between P2P/Attack with 4.15% followed by P2P/others with 2.60%. This fact confirms the results of other works [11] which showed that, in this workload, the largest intersections are between P2P and Attack and between P2P and HTTP (others).

We found also that the small number of samples that were not assigned to any class (3.06%) were mainly composed by outliers from various classes not represented in the training set. Given the known robustness of the Naïve Bayes classifier, the outliers were not filtered in our experiments.

Finally, as showed on standard deviation column on table 12, we found a large variability among runs.

4 Conclusions

This paper presented a methodology for features selection and Internet traffic classification based on one-against-all approach and on Naïve Bayes classifiers. The method presented here uses a small number of statistical discriminators from traffic flows in each application to classify Internet applications. Specifically, in the traffic flows classification, for the used data, the proposed methodology reduces the number of discriminators to approximately 4 variables per application.

According to [20], there is nowadays a hardness to compare Internet traffic classification results as consequence of the data diversity and of the evaluation criterion applied on these works. However, we emphasize that the results presented here are better than that showed in [11] for similar statistical methods.

We believe that the methods presented, although they can be improved, present promising perspectives in real time applications in a not stationary environment. This becomes more clear when we want to separate a specific class such as in attack detection applications or P2P traffic monitoring.

As future work, we are planning to apply the presented methodology to own traces collected from two ISPs and extend the methodology to other classes of traffic. At this moment, we are experimenting different classification techniques and decision rules embedded in the methodology, including kernel methods.

References

1. Moore, D., et al.: CoralReef software suite as a tool for system and network administrators. In: In Proceedings of the LISA 2001 15th Systems Administration Conference (2001)
2. Sariou, S., Gummadi, K., Dunn, R., Gribble, S., Levy, H.: An analysis of Internet content delivery systems. In SIGOPS Oper.Syst. Rev., 315–327 (2002)
3. Sen, S., Wang, D.: Analyzing peer-to-peer traffic across large networks. In: ACM SIGCOMM Internet Measurement Workshop (2002)
4. Sen, S., Spatscheck, O., Wang, D.: Accurate, Scalable In-Network Identification on P2P Traffic using Application Signatures. In: WWW 2004: Proceedings of the 13th International Conference on World Wide Web (2004)
5. Karagiannis, T., Broido, A., Faloutsos, M., Claffy, K.: Transport Layer Identification of P2P Traffic. In: Proceedings of IMC 2004 (2004)
6. Paxson, V.: Empirically derived analytic models of wide-area TCP connections. IEEE/ACM Trans. Netw., 316–336 (1994)
7. Auld, T., et al.: Bayesian Neural Networks for Internet Traffic Classification. IEEE Transactions on Neural Networks (2007)
8. Carmo, M.F.F., Maia, J.E.B., Holanda Filho, R., de Souza, J.N.: Attack Detection based on Statistical Discriminators. In: IEEE International Global Information Infrastructure Symposium, 2007, Marrakech. Proceedings of the IEEE International Global Information Infrastructure Symposium (2007)
9. Paulino, G., Maia, J.E.B., Holanda Filho, R., de Souza, J.N.: P2P Traffic Identification using Cluster Analysis. In: IEEE International Global Information Infrastructure Symposium, 2007, Marrakech. Proceedings of the IEEE International Global Information Infrastructure Symposium (2007)
10. Holanda Filho, R., Maia, J.E.B., do Carmo, M.F.F., Paulino, G.: An Internet Traffic Classification Methodology based on Statistical Discriminators. In: IEEE/IFIP Network Operations & Management Symposium, 2008, Salvador, Brazil. Proceedings of the NOMS 2008 (2008)
11. Moore, A., Zuev, D.: Internet Traffic Classification Using Bayesian Analysis Techniques. In: Proceedings of the 2005 ACM Sigmetrics International Conference on Measurements and Modeling of Computer Systems, Alberta, Canada (2005)
12. Anderson, T.W.: An Introduction to Multivariate Statistical Analysis. John Wiley Sons, New York (1958)
13. Johnson, D.: Applied Multivariate Methods for Data Analysis. Brooks/Cole Publishing Co. (1998)
14. Kaufman, L., Rousseeuw, P.: Finding Groups in Data: An Introduction to Cluster Analysis. Wiley and Sons, Inc., Chichester (1990)
15. Jain, R.: The Art of Computer Systems Performance Analysis. John Wiley Sons, Inc., Chichester (1991)
16. MacQueen, J.B.: Some Methods for classification and Analysis of Multivariate Observations. In: Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, pp. 281–297. University of California Press, Berkeley (1967)

17. Moore, A., et al.: Discriminators for use in flow-based classification. RR-05.13 Department of Computer Science. University of London (2005)
18. Moore, A., et al.: Architecture of a Network Monitor. In: Passive & Active Measurement Workshop, PAM (2003)
19. Wei, L., et al.: Efficient application identification and the temporal and spatial stability of classification schema. Computer Networks 53, 790–809 (2009)
20. Kim, H., et al.: Internet Traffic Classification Demystified: Myths, Caveats, and the Best Practices. In: ACM CoNEXT 2008, Madrid, SPAIN, December 10-12 (2008)
21. WEKA: Data Mining Software in Java
22. `http://www.cs.waikato.ac.nz/ml/weka/`
23. `http://www.mathworks.com/support`
24. Hand, D.J., Yu, Y.: Idiots Bayes - not so stupid after all? International Statistical Review 69, 385–389 (2001)
25. Zhang, H.: The optimality of naive Bayes. In: Proceedings of the Seventeenth Florida Artificial Intelligence Research Society Conference, pp. 562–567. AAAI Press, Menlo Park (2004a)
26. Beygelzimer, A., Langford, J., Zadrozny, B.: Weighted One-Against-All. In: Proceedings of the 20th National Conference on Artificial Intelligence (AAAI), pp. 720–725 (2005)
27. Sulzmann, J.-N., Furnkranz, J., Hullermeier, E.: On pairwise naive bayes classifiers. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) ECML 2007. LNCS (LNAI), vol. 4701, pp. 371–381. Springer, Heidelberg (2007)
28. Witten, I.H., Frank, E.: Data Mining. Morgan Kaufmann Publishers, San Francisco (2000)
29. de Oliveira, M., Valadas, R., Pacheco, A., Salvador, P.: Cluster analysis of Internet users based on hourly traffic utilization. IEICE Transactions on Communications E90-B(7), 1594–1607 (2007)

A Labeled Data Set for Flow-Based Intrusion Detection

Anna Sperotto, Ramin Sadre, Frank van Vliet, and Aiko Pras

University of Twente
Centre for Telematics and Information Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
P.O. Box 217, 7500 AE Enschede, The Netherlands
{a.sperotto,r.sadre,a.pras}@utwente.nl,
d.f.vanvliet@student.utwente.nl

Abstract. Flow-based intrusion detection has recently become a promising security mechanism in high speed networks (1-10 Gbps). Despite the richness in contributions in this field, benchmarking of flow-based IDS is still an open issue. In this paper, we propose the first publicly available, labeled data set for flow-based intrusion detection. The data set aims to be *realistic*, i.e., representative of real traffic and *complete* from a labeling perspective. Our goal is to provide such enriched data set for tuning, training and evaluating ID systems. Our setup is based on a honeypot running widely deployed services and directly connected to the Internet, ensuring attack-exposure. The final data set consists of 14.2M flows and more than 98% of them has been labeled.

1 Introduction

Considering the increasing number of security incidents and discovered vulnerabilities per year [1], it is not surprising that intrusion detection (ID) has become an important research area in the last decade. A large number of ID techniques have been proposed and many of them have been implemented as prototypes or in commercial products. Moreover, the research community has recently focused on flow-based approaches.

When proposing a new intrusion detection system (IDS), researchers usually evaluate it by testing it on labeled (or annotated) traffic traces, i.e., traffic traces with known and marked anomalies and incidents [2]. Labeled traces are important to compare the performance of diverse detection methods, to measure parameter effectiveness and to fine-tune the systems. Ideally, a labeled traffic trace should have the following properties: it should be realistic (opposed to “artificial”), completely labeled, containing the attack types of interest and, not less importantly, publicly available. Despite the importance of labeled traces, research on IDS generally suffers of a lack of shared data sets for benchmarking and evaluation. Moreover, we have no knowledge of any publicly available *flow-based* traffic trace that satisfies all these criteria.

Several difficulties prevent the research community to create and publish such traces, in first place the problem of balancing between privacy and realism. It is natural that the most realistic traces are those collected “in the wild”, for example at Internet service providers or in corporate networks. Unfortunately, these traces would reveal privacy sensitive information about the involved entities and hence are rarely published. On the other hand, artificial traces, i.e., traces that have not been collected but artificially generated, can avoid the problem of privacy but they usually require higher effort and deeper domain knowledge to achieve a realistic result. Moreover, labeling is a time consuming process: it could easily be achieved on short traces, but these traces could present only a limited amount of security events. Therefore, most publications use non-public traffic traces for evaluation purposes. The only notable exception is the well-known DARPA traces [3,4,5], which still are, despite their age, the only publicly available labeled data sets specifically created for intrusion detection systems evaluation.

In this paper, we present the first labeled flow-based data set. We describe how a suitable traffic trace can be collected and how it can be labeled. A flow is defined as "a set of IP packets passing an observation point in the network during a certain time interval and having a set of common properties" [6]. Our final data set will contain only flows and full-packet content will be used only as additional information source during the labeling process. Concentrating on flow-based data sets is in our opinion important for mainly two reasons. First, it substantially reduces the privacy concerns compared to a packet-based data set: since there is no payload, it is possible to deal with privacy issues by anonymizing the IP addresses. Second, several promising results have been recently proposed in the flow-based intrusion detection research community (for example, [7,8,9]): we feel that especially now there is the need for a shared flow data set.

Publications on the labeling of non-artificial data traces are rare and, as said, we are not aware of any concerned flow-based data sets. WebClass [10], a web-based tool that allows people to contribute to manually label traffic time-series, has not found much interest in the community. More effort has been done in the creation of artificial data sets. The already mentioned DARPA trace consists of generated traffic, equivalent to the traffic at a government site containing several hundreds of users on thousands of network hosts. In [11], the authors propose the synthetic generation of flow-level traffic traces, however without discussing its technical implementation. The malicious-traffic generation in the MACE [12] and FLAME [13] frameworks are performed by mixing background traffic with the output of different attack and anomalies modules. In MACE, it is up to the user to create a realistic traffic trace by providing an appropriate configuration file.

The remainder of the paper is structured as follows. In Section 2, we describe and discuss the data collection setup. The processing of the collected data and the labeling of the found intrusions and anomalies is described in Section 3. The resulting data set and its characteristics are discussed in Section 4. Finally, the paper concludes with Section 5.

2 Data Collection

A proper measurement setup depends on the requirements we expect the labeled data set to meet. In our case, we want the data set to be *realistic*, as *complete* in labeling as possible, *correct*, achievable in a acceptable *labeling time* and of a sufficient *trace size*. In our research, we studied diverse approaches to data collection. In Section 2.1

we present our operational experience, explaining strengths and drawbacks of possible measurement infrastructures according to our data set requirements. In addition, Section 2.2 describes the technical details of our measurement setup.

2.1 Operational Experience in Data Collection

This subsection will discuss the impact of both *measurement scales* and *flow collection location* on our requirements.

Measurement scale. Flows can be collected at different measurement scales. The University of Twente has a 10 Gbps optical Internet connection with an average load of 650 Mbps and peaks up to 1.0 Gbps. Several hundred million flows are exported per day [8]. Traces collected on this network would for sure be realistic, but we cannot accomplish the goals of completeness in labeling and reasonable creation time. Labeling network-wide traces suffers of scalability issues.

Let us now concentrate on a small subnetwork that is primarily used by us for research purposes. Due to the limited number of users, it should be easy to distinguish trusted IP addresses from unknown ones, leaving out only a small fraction of suspicious traffic to be further analyzed. However, our findings show that more than 60% of the connections cannot easily be categorized as malicious or benign. Collecting on a small subnetwork ensures us to have a realistic data set, but, as for the network scale, it would be neither complete nor achievable in a short time.

A different setup, and the one that we finally chose, is based on monitoring a single host with *enhanced logging* specifically tuned to track malicious activities, e.g., a *honeypot*. A honeypot can be defined as an "environment where vulnerabilities have been deliberately introduced to observe attacks and intrusions" [14]. In this case, the trace size would be smaller and the labeling time limited. Moreover, an everyday service setup would ensure the traffic to be realistic. Finally, the logs would ensure us to achieve both completeness and correctness in labeling.

Flow collection location. We have different options about the flow collection location, while we collect logs on our honeypot. A possibility is to collect the flows generated by a Netflow enabled router, in our case the University one. However, decoupling the flow creation from the log location introduces errors in measurements, such as timing delays and split TCP sessions. These issues make impossible to properly associate a security event with the flow that caused it. A data set based on these premises would have a serious lack in completeness and correctness. Another possibility is therefore to dump only the traffic reaching the honeypot and to create the flows off-line after the data collection is completed. This decision allows to have complete control over the flow-creation process, overcoming the problem of the session splitting.

Discussion. We will now summarize the methods we studied for the data set collection, showing if they meet our requirements. Please note that, since monitoring the university network or a subnetwork does not scale, we did not explore further the options of online/offline flow creation. All our approaches ensure the trace to be realistic. Moreover, monitoring the University network or a subnetwork would also provide sufficiently large traces. After we measured the traffic reaching our honeypot, we can say that also in this case this requirement can be met. Regarding completeness and correctness, large network traces reduce the trust we can have on the labeled data set. The honeypot approach is more reliable since it offers additional logging information, but in this case, the flow collection/creation is crucial. As previously in this section, relying on external flow sources can introduce measurement errors. Creating the flows offline, on the other hand, allows to have flows that better suit the needs of a labeled data set. Finally, large infrastructures suffer of scalability in labeling time, while the honeypot setup overcomes this problem.

From this section, we can conclude that monitoring a single host that has enhanced logging capabilities is a promising setup for flow-based data sets.

2.2 Experimental Setup

Our honeypot was installed on a virtual machine running on Citrix XenServer 5 [15]. The decision to run the honeypot as a virtualized host is due to the flexibility to install, configure and recover the virtual machine in case it is compromised. In addition, a compromised virtual machine can be saved for further analysis. Diverse scenarios are possible, in terms of number of virtual machines, operative systems and software. Our experimental setup consisted of a single virtual machine, on which we installed a Linux distribution (Debian Etch 4.0). In order to keep the setup simple, controllable and realistic, we decided not to rely on honeypot software, but to configure the host by ourselves. In particular, the following services have been installed:

- `ssh`: OpenSSH service [16]. Beside the traditional service logs, the OpenSSH service running on Debian has been patched in order to log sessions: for each login, the *transcript* (user typed commands) and the timing of the session have been recorded. This patch is particularly important to track active hacking activities.
- Apache web server: a simple webpage with a log in form has been deployed. We relied on the service logging capabilities for checking the content of the incoming `http` connections.
- `ftp`: The chosen service was `proftp` [17]. As for `http`, we relied on the `ftp` logs for monitoring attempted and successful connections. `proftp` uses the `auth/ident` service running on port 113 for additional authentication information about incoming connections.

Along with the service logs, we decided to dump all the traffic that reached the honeypot during our observation period. The virtual machine ran for 6 days, from Tuesday 23 September 2008 12:40:00 GMT to Monday 29 September 2008 22:40:00 GMT. The honeypot was hosted in the university network and directly connected to the Internet. The monitoring window is comprehensive of both working days and weekend days. The data collection resulted in a 24 GB dump file containing 155.2 million packets.

3 Data Processing and Labeling

The labeling process enriches the data trace with information about (i) the type and structure of the malicious traffic, (ii) dependencies between single isolated malicious

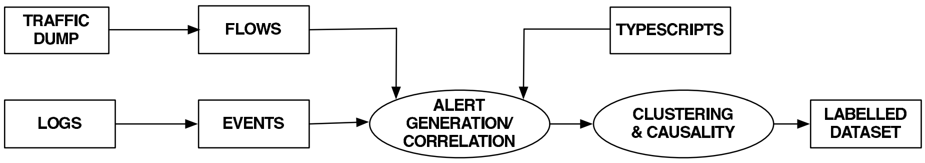


Fig. 1. From raw data (packets and logs) to the labeled data-set

activities. The latter is particularly important for a flow-based data set where by design no further detail on the content of the communication is available to the end user. In this section, we describe the processing of the collected traffic data and how the labeling information is generated and structured.

Figure 1 gives an overview on the whole data processing and labeling. As a first step, shown in the left part of the figure, the collected traffic dumps are converted into flows. In addition, the data extracted from the diverse log files is converted into a common format (log events) that simplifies the following processing steps. The resulting flow records and log events feed the alert generation/correlation process. Finally, a post-processing step generates additional information, namely the so-called cluster alerts and the causality information. The different steps are explained in the following Sections 3.1 through 3.4.

3.1 From Packets to Flows

The first step is the creation of flows from the traffic trace. In our data set, a flow closely follows the Netflow v5 definition and has the following form:

$$F = (I_{src}, I_{dst}, P_{src}, P_{dst}, Pckts, Octs, T_{start}, T_{end}, Flags, Prot)$$

where the unidirectional communication is defined by the source and destination IP addresses I_{src} and I_{dst}, the employed ports P_{src} and P_{dst} (in case of UDP/TCP traffic), and the level 3 protocol type $Prot$. The fields $Pckts$ and $Octs$ give the total number of packets and octets, respectively, in the data exchange; the TCP header flags are stored as a binary `OR` of the flags in all the packets of the flow (field $Flags$); the start and end time of the flow are given in T_{start}, respectively, T_{end} in millisecond resolution. The flow creation has been performed using a modified version of `softflowd`[18].

3.2 From Log Files to Log Events

Information about attacks against our honeypot can be extracted from the log files of the services we were monitoring. In order to simplify the alert generation process, the relevant data found in the various log files is converted into log events. A log event consists of the following information:

$$L = (T, I_{src}, P_{src}, I_{dst}, P_{dst}, Descr, Auto, Succ, Corr)$$

where T gives the timestamp of the attack (as found in the logs), I_{src} and P_{src} give the IP address and used port (if available) of the attacker, and I_{dst} and P_{dst} give the attacked

IP address (our honeypot) and port number. In addition, a deeper analysis of the log files reveals whether an attack was automated or manual and succeeded or failed (flags $Auto$ and $Succ$, respectively). The field $Descr$ allows us to enrich the data set with additional information retrieved from the log files. The field $Corr$ is always initialized to $false$ and later used by the alert generation process.

3.3 Alert Generation and Correlation

Goal of this step is to generate so-called alerts and to correlate each generated alert to one or more flows. An alert describes a security incident and is represented by the following tuple:

$$A = (T, Descr, Auto, Succ, Serv, Type)$$

The fields T, $Descr$, $Auto$, $Succ$ are defined as for the log events (see Section 3.2). The field $Serv$ gives the service addressed by the security incident, for example `ssh` or `http`. The $Type$ field describes the type of the incident. The alert generation process consists of three steps that are explained in the following.

Alerts from log events. For attacks toward the honeypot, alerts can be directly generated from the log events. The fields T, $Descr$, $Auto$, $Succ$ of an alert record are set to the values of the corresponding fields of the log event. The $Serv$ field is set accordingly to the destination port field of the log event, for example $Serv =$ `ssh` if $P_{dst} = 22$. The $Type$ field is always set to the value `CONN`, indicating that the alert describes a malicious *conn*ection attempt.

In order to correlate an alert with a flow, we have to find the flow that corresponds to the log event from which the alert has been generated. This is not a trivial task since the timing information extracted from a log file may not be aligned with the flow's one. In addition, we would like to not only correlate the incoming (as seen from the honeypot) flow to the alert but also the response flow of the honeypot.

We use the flows as starting point for the alert generation and correlation. This avoids that a flow is correlated to more than one alert. The resulting procedure for a service s is shown in Algorithm 1. As a first step, a best matching response flow is selected for each flow of the considered service (lines 3-6). The matching is made based on the flow attributes. If there are more than one candidate flows, the closest in (future) time is chosen. It is possible, nevertheless, that such a flow does not exist, for example in the case in which the target was a closed destination port. Since the flow tuple source/destination address/port may appear multiple times in the data set, the parameter δ ensures that an incoming flow is not correlated with a response too late in time. Values of δ from 1 to 10 seconds are possible.

After searching for a response flow, the algorithm proceeds with retrieving the best matching log event (lines 7-10). The log event must match the flow characteristics. The timing constraint in this case forces the log event to be in the interval between the beginning and the end of the flow. Moreover, since log files do not provide us with millisecond precision and multiple alerts can be generated by the same host in the same second, we require the matching log event to not have been consumed before (line 9). If the matching event exists, the algorithm will create the alert and correlate it with the

Algorithm 1. Correlation procedure

```
1: procedure ProcessFlowsForService ($s$ : service)
2: for all Incoming flows $F_1$ for the service $s$ do
3:     Retrieve matching response Flow $F_2$ such as
4:     $F_2.I_{src} = F_1.I_{dst} \wedge F_2.I_{dst} = F_1.I_{src} \wedge F_2.P_{src} = F_1.P_{dst} \wedge F_2.P_{dst} = F_1.P_{src} \wedge$
5:     $F_1.T_{start} \leq F_2.T_{start} \leq F_1.T_{start} + \delta$
6:     with smallest $F_2.T_{start} - F_1.T_{start}$ ;
7:     Retrieve a matching log event $L$ such as
8:     $L.I_{src} = F_1.I_{src} \wedge L.I_{dst} = F_1.I_{dst} \wedge L.P_{src} = F_1.P_{dst} \wedge L.P_{dst} = F_1.P_{src} \wedge$
9:     $F_1.T_{start} \leq L.T \leq F_1.T_{end} \wedge$ not $L.Corr$
10:    with smallest $L.T - F_1.T_{start}$ ;
11:    if $L$ exists then
12:        Create alert $A = (L.T, L.Descr, L.Auto, L.Succ, s, \texttt{CONN})$.
13:        Correlate $F_1$ to $A$ ;
14:        if $F_2$ exists then
15:            Correlate $F_2$ to $A$ ; $L.Corr \leftarrow \mathbf{true}$ ;
16:        end if
17:    end if
18: end for
```

flow and, if possible, with the response flow (lines 11-16). Finally, the log event will be marked as consumed (line 15).

Alerts for outgoing attacks. Some of the incoming attacks against the `ssh` were successful. As a consequence, the attacker used the honeypot machine itself to launch `ssh` scans and dictionary attacks against other machines. In order to generate alerts for these outgoing attacks, we have analyzed the typescripts of the `ssh` sessions. This allowed us to reconstruct the times and the destinations of the different attacks launched from the honeypot. Similarly to the previous step, we have used this information to find the corresponding flows and to correlate them to alerts of type `CONN`.

Alerts for side effects. Several attacks towards and from the honeypot have caused non-malicious network traffic that we consider as "side effects". Most notably, `ssh` and `ftp` connection attempts caused ICMP traffic and traffic directed to the `auth/ident` service (port 113) of the attacker, respectively, the honeypot. Furthermore, one attacker installed an IRC proxy on the honeypot. For these flows, we have created alerts of type `SIDE_EFFECT` with $Serv = \texttt{ICMP}$, respectively, $Serv = \texttt{auth}$ or $Serv = \texttt{IRC}$.

3.4 Generation of Cluster Alerts and Causality Information

The alerts described in the previous section represent single security incidents and are directly correlated with one or more flows. In addition to those basic alerts, we also generate so-called cluster alerts and extra-information describing the causal relationships between alerts.

Cluster alerts are used to label logical groups of alerts. For example, in the case of an `ssh` scan consisting of 100 connection attempts, we create basic alerts of type `CONN` for each connection, plus *one* cluster alert of type `SCAN` for the entire scan operation.

More formally, a basic alert A belongs to a scan alert C, if (i) the alert A has the same source IP and type as the other alerts in the cluster, and (ii) the alert is not later than γ seconds after the latest alert in the cluster. In our case we set $\gamma = 5$ seconds.

As a final step, we manually add causality information. In the current setup, that means that we have created (i) links between the alert representing an attacker's successful log-in attempt into the honeypot via `ssh` and all alerts raised by the attacker during that `ssh` session, and (ii) links between the alerts of the ICMP and `auth/ident` flows and the alerts of the `ssh` and `ftp` flows that caused them.

Our data set has been implemented in a MySQL database. The structure of the database reflects the alert and flow structure and the relations between these categories.

4 The Labeled Data Set

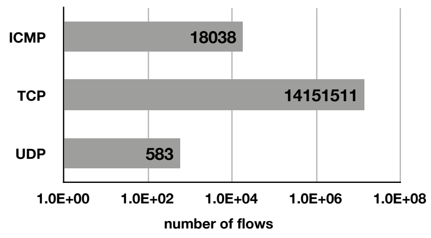
The processing of the dumped data and logs, collected over a period of 6 days, resulted in 14.2M flows and 7.6M alerts.

4.1 Flow and Alert Breakdown

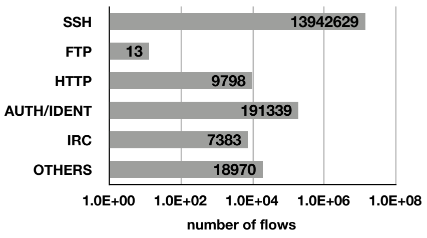
Figure 2(a) and 2(b) present a breakdown of the flows according to the level 3 protocols and the most active services, respectively.

At network level, the collected data set presents a subdivision in only three IP protocols: ICMP, TCP and UDP. The majority of the flows has protocol TCP (almost 99.9% of the entire traffic). A second slice of the data set consists of ICMP traffic (0.1%). Finally, only a negligible fraction is due to UDP traffic. The level 3 protocol breakdown is consistent with the services flow breakdown, shown in Figure 2(b). The honeypot most active service has been `ssh`, followed on the distance by `auth/ident`.

Figures 3(a) and 3(b) show the alert breakdown according to malicious connections and scans on the most active services. `ssh` and `auth/ident` are the only services for which the honeypot has been both a target and a source. The honeypot received several thousand `ssh` connections and it has been responsible of millions of outgoing ones. On the contrary, it has been the target of extensive `auth/ident` requests, but it produced only 6 outgoing connection to port 113. As shown in Figure 3(b), the `ssh` alerts can be grouped into 45 scans, 10 incoming and 35 targeting remote destinations. We also have

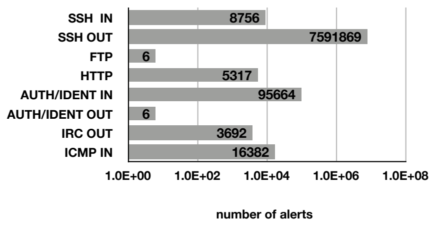


(a)



(b)

Fig. 2. Flow breakdown according to the level 3 protocol (a) and the service traffic (b)



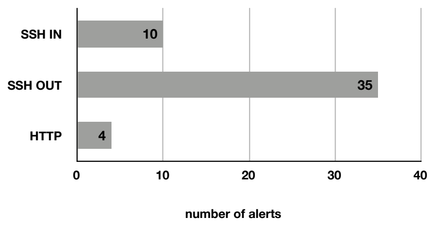


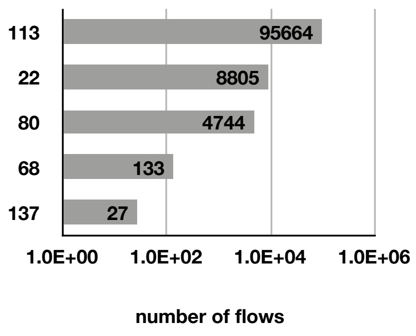
Fig. 3. Alert repartition for basic (a) and cluster (b) types

been the target of 4 `http` scans. None of the `ftp`, `auth/ident` or `irc` alerts can be clustered into scans.

4.2 Honeypot Target and Source Behavior

Figure 4(a) presents the five most contacted ports on the honeypot. Port 113 (`auth/ident` service) is the most targeted one. Among the services we monitored, the most often contacted ones are `ssh` (port 22) and `http` (port 80). On these ports we actually received malicious traffic, confirming our initial idea that daily used services are normally target of attacks. The incoming traffic on port 68 is due to periodical renew of the dynamically assigned IP address (`dhcp`). Finally, we received traffic on port 137 due to the `netbios` protocol, to which we never responded.

Figure 4(b) presents the top 5 ports contacted by the honeypot on remote destinations. The hit list is opened by `ssh`, confirming the alert breakdown in Figure 3. Traffic directed to port 67 has been generated by `dhcp` and it matches the incoming traffic on port 68. The traffic on port 53 (`dns`) was directed to the University `dns` servers. The outgoing traffic towards port 80 consists of only RST packets. The dumped traffic shows that remote hosts contacted us with SYN/ACK packets from port 80. Since the



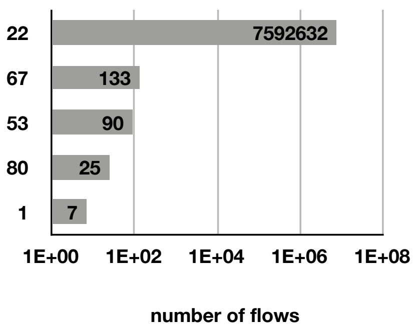


Fig. 4. Top 5 contacted ports. Figure (a) presents the incoming traffic, while Figure (b) the outgoing one (logarithmic scale).

honeypot never initiated this interaction, it reset the connections. For its characteristics, this traffic seems not to be related to attacks: we suspected it is a form of background radiation [19,20], or part of a SYN/ACK scan. Traffic to port 1 is negligible (`tcpmux`).

4.3 Discussion on the Data Set

This section will discuss the results we obtained as outcome of the correlation process, pointing out the characteristic of both the labeled and unlabeled traffic. Our correlation process succeeded in labeling more that 98.5% of the flows and almost the totality of the alerts (99.99%).

Malicious traffic. All the labeled traffic is related to the monitored services. Since we did not interfere in any way with the data collection process, i.e., we avoided any form of attack injection and we did not advertise our infrastructure on hacker chats, we can assume that the attacks present in the data set reflect the situation on real networks.

The majority of the attacks targeted the `ssh` service and they can be divided into two categories: the automated and the manual ones. The first ones are well-known automated brute force scans, where a program enumerates usernames and passwords from large dictionary files. This attack is particularly easy to observe at flow level, since it generates a new flow for each connection. Attacks of the second type are manual connection attempts. There are 28 of these in our trace and 20 of them succeed.

The `http` alerts labeled in our data set are automated attacks that try to compromise the service by executing a scripted series of connections. These scans are executed using tools like Nikto and Whisker, both easily available online. Differently from the `ssh` connections, no manual `http` attacks are present.

Regarding the `ftp` traffic, the data set contains only 6 connections to this service on the honeypot, during which a `ftp` session has been opened and immediately closed. Even if the connections did not have a malicious content, this behavior could be part of a reconnaissance performed by an attacker gathering information about the system.

Side-effect traffic. Part of the traffic in our data set is the side effect of attacks but cannot be considered by itself malicious. `auth/ident`, ICMP and `irc` traffic fall in this category. The scanning activities have been responsible of the majority of the flows to and from port 113. The service is supposed, indeed, to retrieve additional information about the source of a `ssh` connection. Regarding the ICMP traffic, more that 120,000 incoming packets have type *time exceeded* and *destination unreachable* and come from networks that the honeypot scanned.

The analysis of the honeypot showed that a hacker installed an IRC proxy that received chat messages from several channels. It does not appear anyway that any message has been sent from our honeypot or that the channels have been used for any malicious activity. Even if this traffic is due to an application that we did not originally install on our machine, we decided not to label it as malicious but as a side effect.

Unknown traffic and uncorrelated alerts. For a small fraction of the data set, we cannot establish the malicious/benign nature of the traffic. This traffic consists of three main components: (i) `ssh` connections for which it has not be possible to find a matching alert, (ii) heterogeneous traffic, containing all the flows having as destination a closed honeypot port and (iii) some not malicious connections to the `http` and `vnc` services. The `vnc` service was accessible because of the XEN virtualization software running on the honeypot. Strangely, no attacker identified this service and tried to compromise it by using known vulnerabilities or performing a brute-force dictionary attack on the password . The `vnc` flows in the data set are originating from two hosts which contacted the service but did not complete the connection.

Regarding the alerts, for few of them (0.01%) no matching traffic has been found. This makes us aware that during the collection phase some packets that reached our honeypot have not been recorded by tcpdump.

5 Conclusions

The main contribution of this paper is to present the first labeled data set for flow-based intrusion detection. While approaching the problem of creating a labeled data set, we consider the choice of the proper data collection infrastructures a crucial point. It has indeed impact not only on the feasibility of the labeling, but also on the reliability of the result. We studied several data collection infrastructures, enlightening strengths and drawbacks. We conclude that, in the case of flow-based data sets, the most promising measurement setup is monitoring a single host with enhanced logging capabilities. In the specific context of this experiment, the host was a honeypot. The information collected permitted us to create a data base of flows and security events (alerts).

The paper also describes a semi-automated correlation process that matches flows with security events and, in addition, reflects the causality relations between the security events themselves. The results of the correlation process show that we have been able to label more that 98% of the flows in the trace. The correlation process also proves that, although we limited our measurements to a single host, labeling remains a complex task that requires human intervention.

The approach we propose in this paper allows us to capture unexpected security events, such as the behavior of a compromised host. However, the presented trace mainly consists of malicious traffic. For evaluation of an IDS, this means that our data set allows to detect false negatives but not false positives. In the future, we aim at extending our monitoring setup to more challenging scenarios. An example would be a server that is daily accessed by benign users. Monitoring such a server would result in a trace with a more balanced content of malicious and normal traffic. We believe that an approach similar to the one we presented in this paper will be useful also in this scenario. Finally, the data set is available on e-mail request to the authors.

Acknowledgments. This research has been supported by the EC IST-EMANICS Network of Excellence (#26854).

References

1. CERT Coordination Center (January 2009), `http://www.cert.org/certcc.html`
2. Mell, P., Hu, V., Lippmann, R., Haines, J., Zissman, M.: An overview of issues in testing intrusion detection systems. Technical Report NIST IR 7007, National Insititute of Standards and Technology (June 2003)

3. Lippmann, R., Fried, D., Graf, I., Haines, J., Kendall, K., McClung, D., Weber, D., Wyschogrod, D., Cunningham, R., Zissman, M.: Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation. In: Proc. of the DARPA Information Survivability Conf. and Exposition, DISCEX 2000 (2000)
4. Lippmann, R., Haines, J., Fried, D., Korba, J., Das, K.: The 1999 DARPA off-line intrusion detection evaluation. Computer Networks 34 (2000)
5. Haines, J., Lippmann, R., Fried, D., Zissman, M., Tran, E., Boswell, S.: 1999 DARPA Intrusion Detection Evaluation: Design and Procedures. Technical Report TR 1062, MIT Lincoln Laboratory (February 2001)
6. Quittek, J., Zseby, T., Claise, B., Zander, S.: Requirements for IP Flow Information Export (IPFIX). RFC 3917 (Informational)
7. Lakhina, A., Crovella, M., Doit, C.: Characterization of network-wide anomalies in traffic flows. In: Proc. of 4th ACM SIGCOMM Conf. on Internet measurement, IMC 2004 (2004)
8. Sperotto, A., Sadre, R., Pras, A.: Anomaly characterization in flow-based traffic time series. In: Akar, N., Pioro, M., Skianis, C. (eds.) IPOM 2008. LNCS, vol. 5275, pp. 15–27. Springer, Heidelberg (2008)
9. Strayer, W., Lapsely, D., Walsh, R., Livadas, C.: Botnet Detection Based on Network Behavior. Advances in Information Security, vol. 36 (2008)
10. Ringberg, H., Soule, A., Rexford, J.: Webclass: adding rigor to manual labeling of traffic anomalies. In: SIGCOMM Computer Communication Review, vol. 38(1) (2008)
11. Ringberg, H., Roughan, M., Rexford, J.: The need for simulation in evaluating anomaly detectors. In: SIGCOMM Computer Communication Review, vol. 38(1) (2008)
12. Sommers, J., Yegneswaran, V., Barford, P.: A framework for malicious workload generation. In: Proc. of the 4th ACM SIGCOMM Conf. on Internet measurement, IMC 2004 (2004)
13. Brauckhoff, D., Wagner, A., Mays, M.: Flame: a flow-level anomaly modeling engine. In: Proc. of the Conf. on Cyber security experimentation and test, CSET 2008 (2008)
14. Pouget, F., Dacier, M.: Honeypot-based forensics. In: Asia Pacific Information technology Security Conference (AusCERT 2004) (May 2004)
15. 5, C.X.: (April 2009), `http://www.citrix.com/`
16. OpenSSH: `http://www.openssh.com/`
17. proftp: `http://www.proftpd.org/`
18. Softflowd: (April 2009), `http://www.mindrot.org/projects/softflowd/`
19. Moore, D., Shannon, C., Brown, D., Voelker, G., Savage, S.: Inferring internet denial-of-service activity. ACM Trans. Comput. Syst. 24(2) (2006)
20. Pang, R., Yegneswaran, V., Barford, P., Paxson, V., Peterson, L.: Characteristics of internet background radiation. In: Proc. of the 4th ACM SIGCOMM Conf. on Internet measurement, IMC 2004 (2004)

Fast Failure Detection in Multipoint Networks

Wouter Tavernier[1], Dimitri Papadimitriou[2], Bart Puype[1], Didier Colle[1], Mario Pickavet[1], and Piet Demeester[1]

[1] Ghent University, IBCN - IBBT, Department of Information Technology
Gaston Crommenlaan 8 bus 201, B-9050 Gent, Belgium
{wouter.tavernier,didier.colle,mario.pickavet, piet.demeester}@intec.ugent.be
[2] Alcatel-Lucent Bell
Copernicuslaan 50, B-2018 Antwerpen , Belgium
dimitri.papadimitriou@alcatel-lucent.be

Abstract. Bridged Ethernet and IP routing and forwarding are without any doubt the most deployed network technologies, mainly because of their robust, low-cost and easy-to-deploy-character. Solutions to provide fast recovery over these technologies are thus a highly desired feature. Fast recovery not only needs a fast switchover (e.g. using IP-FastReRoute), but also requires fast failure detection. However, few efficient failure detection mechanisms are available over Ethernet networks, i.e. multi-access broadcast capable networks. We present an efficient multipoint Bidirectional Forwarding Detection (BFD) scheme running directly over Ethernet. We prove its value through emulation and evaluate its time performance and resource consumption.

1 Introduction

The growth of the Internet and networks in general, has been significantly supported by the two crown jewels of the networking economy: IP and Ethernet. The first being the layer 3 technology used for inter-network addressing, routing and forwarding, the second being the layer 2 technology of choice for intra-network forwarding, especially in LAN environments. The success of both technologies was emphasized last decades by the numerous efforts to extend them in terms of traffic engineering capacity, resilience possibilities and scalability. This resulted into highly advanced (but also more expensive) technologies relying on constraint based routing to provide extended traffic engineering capabilities including extended protection and recovery features, tunneling and aggregation functionality. A well-known example of this category of complex control technologies is Generalized MultiProtocol Label Switching (GMPLS). Currently, GMPLS-extensions are being developed to allow the setup and teardown of Ethernet-data paths, referred to as Carrier Ethernet (e.g. Ethernet Label Switching [14] and Provider Backbone Bridge-Traffic Engineering (PBB-TE) [12]).

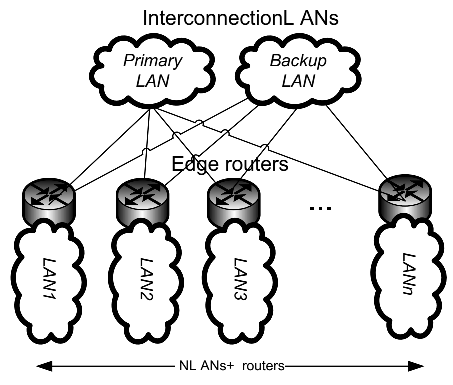
One way to avoid the high expenses and complexity of control suites such as GMPLS, and still provide highly resilient networks using IP over Ethernet, is to use a technique such as IP Fast ReRoute (IPFRR). IPFRR defines a re-routing mechanism at the IP layer which provides protection against a link or router failure by invoking locally pre-determined loop-free alternate paths (a.k.a. repair paths). An important aspect of such a mechanism is the prevention of packet loss caused by transient loops which can occur during the reconvergence of the network following a failure. IPFRR thus provides a highly desirable mechanism to minimize packet loss during failure events, especially for real-time services such as Voice-over-IP (VoIP) or IPTV. However, IPFRR needs to be triggered by a failure detection mechanism (or a failure notification mechanism). In a landscape being dominated by switched Ethernet positioned as networking layer device interconnection layer 2 technology, practice learns that almost no fast failure detection mechanism can be used to efficiently trigger re-routing mechanisms such as IPFRR. Failure notification mechanism being inherently slower than failure detection mechanisms, they are no further investigated in the context of this paper.

Therefore, this paper proposes a new failure detection mechanism applicable at the Ethernet level. The objective consists in enhancing the interplay between the failure detection mechanisms for multi-access broadcast capable Ethernet networks (or LAN[1]-segments) interconnecting IP routers and IPFRR. The structure of the paper is as follows: in Section 2 we discuss existing failure detection methods for use over Ethernet networks on the given scenario of dual-homed LANs and characterize their shortcomings. In the next Section (Section 3) we present a new multipoint BFD (MP BFD) variant for Ethernet. The last Section (Section 4) describes the implementation of the failure detection mechanism presented in Section 3 in an emulation environment, as well as its performance regarding time and resources. The paper concludes with Section 5, by summarizing our main results and outlining possible future direction of investigations.

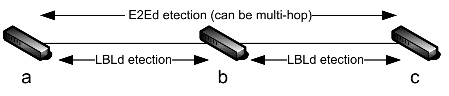
2 Failure Detection over Ethernet Networks

This paper focuses on failure detection mechanisms, i.e. the mechanisms involved in the detection phase that occurs after e.g. a link or node failure, but before the actual actions initiated by the recovery mechanism. In the set of phases involved in the global failure recovery process, failure detection positions thus as turning point between failure occurrence and the recovery operation itself by acting as a trigger. Let us consider the scenario depicted in Figure 1a: n Ethernet LANs attached to n edge IP routers connected to a primary and a backup interconnection LAN. Edge routers thus communicate to each other via either the primary or the backup Ethernet LAN. In the ideal situation, from the

[1] In this paper the term Ethernet LAN (Local Area Network) refers to a network consisting of one or more Ethernet segments (point-to-point or multipoint) interconnected via Ethernet bridges, using the multipoint mechanisms of broadcasting, flooding, learning and learned forwarding (see reference [4] for the standard specification).



(a) Base scenario



(b) LBL vs. E2E failure detection

Fig. 1. Failure detection over Ethernet

moment interconnectivity between two edge routers fails via the primary LAN, the backup LAN is used as fast as possible for communication. IPFRR (see [13] and [2]) can be used as an effective switchover mechanism at the IP layer to recover from a detected failure in the primary LAN. However this requires a failure detection mechanism between the edge routers over the primary and/or backup LAN, to trigger the IPFRR mechanism.

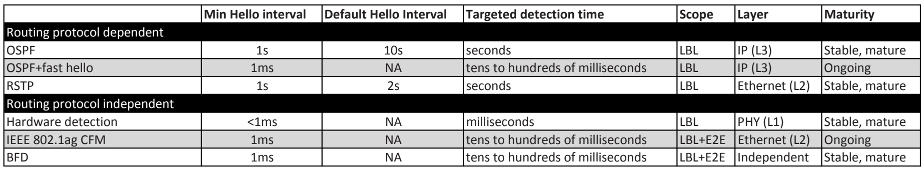
Before existing failure detection technologies are introduced, let us define the *scope of the detection*. We distinguish two scopes: detection on a *link-by-link (LBL)* basis and detection on an *end-to-end* (E2E) data path basis (see Figure 1b). The first single hop scheme allows a link to be monitored independently of the data paths crossing that link. From Figure 1, (Ethernet) data paths are defined by the set of nodes crossed by the Ethernet flow in between a pair of edge routers within the Ethernet interconnection LANs. As such, the single task of LBL detection is to signal whether a link is up, or down. Whereas this provides a perfect mean to determine if a link, and what link exactly has failed in a network, it does not directly indicate the data paths or traffic streams over the network that are affected by the failure. To tackle the latter, the failure detection mechanism needs to adapt its scope to a specific end-to-end data path within the network. This allows to indicate if a traffic stream following the monitored path from a given source to a given destination, is affected by a failure. This type of detection, referred to as *E2E failure detection*, does not give any information on the exact cause (what link or node) of a failing end-to-end data path.

In the remainder of this section, we subdivide our analysis of existing failure detection mechanisms into i) routing protocol dependent, and ii) routing protocol independent failure detection mechanisms. The former includes Open Shortest Path First (OSPF) Hello messages, whereas the latter includes Ethernet data link layer detection mechanisms and Bidirectional Forwarding Detection (BFD).

2.1 Routing Protocol Related Mechanisms

Open-Shortest Path First protocol (OSPF). is a link-state routing protocol which maintains neighbor relationships (e.g. between the edge routers of Figure 1) by periodically sending OSPF Hello messages at every HelloInterval (Type 1 packets, see [11]). Routing adjacencies are formed between selected neighboring routers for exchanging routing information. The RouterDeadInterval is used to determine the number of seconds before the router's neighbors will declare it Down when they stop hearing the router's Hello Packets (by default, 4 times the HelloInterval). This event typically triggers the reconvergence of the routing protocol for the new topology. By default, Hello timers can be configured in the order of seconds (default value of 10s), with a minimum of 1 second, resulting in a failure detection time ranging from 2 to 4 seconds. Such detection time is too slow to allow seamless recovery of real-time services as outlined in the Introduction. We validated these numbers by means of emulation (see Section 4). Initiatives have thus been undertaken to increase the frequency of transmitting Hello packets in OSPF (see e.g. [10]). However these approaches never found broad acceptance, because OSPF was not initially designed for performing fast link failure detection (but only detecting and maintaining neighbor relationships). Indeed, decreasing the Hello timers would result into a high risk of overloading the routing process because of the frequent sending and receiving of Hello messages (which would be from 10 to 1000 times smaller than initially planned). In addition, the lack of handshake mechanism to agree on the HelloInterval to be used between neighboring routers would result into manual configuration of the HelloInterval, which is undesirable in larger networks.

(Rapid) Spanning Tree Protocol. The (R)STP protocol is a distance-vector routing protocol used in (VLAN-)bridged Ethernet to i) construct a loop-free logical tree topology originated at the root bridge (the tree spans all bridges of the entire Ethernet broadcast domain or sub-domain), and ii) avoid broadcasted frames of being endlessly looped through the network. STP, specified in IEEE 802.1d, is based on a break-before-make paradigm (hence slow by construction). Subsequent attempts such as RSTP, specified in IEEE 802.1w, consider a make-before-break approach with faster convergence time. The RSTP failure detection mechanism is also based on Hello timers (as in OSPF). The Hello interval can be configured between 1 and 10 seconds, having a default value of 2 seconds. A link is considered Down after a given number of subsequent Hello messages is lost (3 by default, see [4]). The failure detection mechanism of RSTP is still too slow given the Hello-interval granularity in terms of seconds for the recovery of real-time services. More generally, none of these additional mechanisms/extensions fundamentally solve the root cause of slow convergence resulting from the count-to-infinity problem intrinsic to distance-vector protocols. In practice, this means that only if the remaining topology after failure does not include any cycle - and if the root bridge is not affected by the failure - the re-convergence time is acceptable. If manufacturers declare quick(er) convergence times, this is because of combination with hardware-detection (see next section), and because they do



	Min Hello interval	Default Hello Interval	Targeted detection time	Scope	Layer	Maturity
Routing protocol dependent						
OSPF	1s	10s	seconds	LBL	IP (L3)	Stable, mature
OSPF+fast hello	1ms	NA	tens to hundreds of milliseconds	LBL	IP (L3)	Ongoing
RSTP	1s	2s	seconds	LBL	Ethernet (L2)	Stable, mature
Routing protocol independent						
Hardware detection	<1ms	NA	milliseconds	LBL	PHY (L1)	Stable, mature
IEEE 802.1ag CFM	1ms	NA	tens to hundreds of milliseconds	LBL+E2E	Ethernet (L2)	Ongoing
BFD	1ms	NA	tens to hundreds of milliseconds	LBL+E2E	Independent	Stable, mature

Fig. 2. Failure detection overview

not take into account worst-case topologies prone to i) count-to-infinity, or ii) topologies with slow convergence properties such as linear/ring topologies.

2.2 Routing Protocol Independent Mechanisms

Hardware detection. The most obvious example of routing protocol independent failure detection is performed at the PHY level (layer 1). Various PHY technologies support the detection of the loss of signal at link level. The Ethernet PHY allows to detect decreases in voltage between two links. While this provides for a sound and fast method for failure detection (in the order of tens of microseconds), a major issue is that it can only detect link-errors, meaning that remote link failures are by definition undetectable in case the signal is terminated before reaching the destination.

Ethernet Connectivity Fault Management (CFM). The ongoing IEEE 802.1ag standard defines a set of OAM mechanisms to help operators in managing their bridged Ethernet environments. The CFM suite comprises loopback, traceroute, and continuity check functions. In particular, the last function is interesting for failure detection in our research space. The idea consists in supporting a multicasted heartbeat (short-length protocol data unit) along the network, that is configurable at millisecond granularity. Unfortunately, the details of IEEE 802.1ag protocols, are not yet standardized, nor implemented.

Bidirectional Forwarding Detection (BFD). BFD, specified by the Internet Engineering Task Force (IETF, see [8]), positions itself as a protocol to detect faults in the bidirectional path between two forwarding engines, including interfaces, data link(s), and to the extent possible the forwarding engines themselves, with potentially very low latency (tens to hundreds of ms). In its *asynchronous mode* (base mode), BFD defines a Hello-protocol which sends control packets between two end points at a negotiated transmission (TX) rate and reception (RX) rate. Both rates range in order of microseconds. A three-way handshake mechanism sets up BFD sessions. Sessions are declared Down if control packets at one of either side arrive beyond a computed level of negotiated transmission rates (using a multiplication factor with the remote TX). The *demand mode* allows a system, once a BFD session is established, to ask the other system to stop sending BFD control packets, except when the system decides to explicitly verify connectivity. In this case a short sequence of BFD Control packets is exchanged, and then the system acknowledges the reception of this sequence of packets. An additional *Echo function* provides an independent detection mode by sending control packets to the remote system that itself loops them back through its forwarding path towards the sender. BFD is being standardized for IP and MPLS-environments ([1]). BFD can be used over IP by encapsulating BFD control packets into UDP. When running over connectionless forwarding environments such as IP, BFD supports link-by-link sessions (LBL or single hop, [5]), and end-to-end sessions (E2E or multi-hop [6]).

Now, let us reconsider a part of the network from Figure 1a: a set of n LAN's having their edge routers connected to a (primary) Ethernet LAN. If we want to apply failure detection between the edge routers using BFD over IP as currently specified, we can set up $n(n-1)/2$ LBL IP BFD sessions[2], i.e. one session per pair of routers. Though this scheme is valid, from the resource consumption perspective this mode of operation is not efficient, as it results into $(n-1)$ BFD sessions running over every interface connecting an edge router to the LAN (N-square scalability problem). Given the multi-access broadcast capable nature of the inter-connection network (Ethernet), this results in a waste of resources, because (1) the outgoing direction of the interface connecting a router to the multi-access network does not take advantage of the fact that the (multi-access) LAN network itself can replicate its BFD messages, and (2) every router needs 2 timers per other edge router: one for scheduling the BFD message to be sent, and one to check if a BFD message has been received sufficiently recent, while only the last one is needed. In addition the Address Resolution Protocol (ARP) is needed to retrieve the MAC addresses to be used for setting up the BFD sessions over IP.

The observation that the BFD definition from [8] does not account for the special nature of broadcast and multicast capabilities of certain multi-access network technologies, is also acknowledged by the multipoint BFD effort at IETF (see [7]). By relying on the multicast nature of the underlying technology (again targeting IP or MPLS), a unidirectional multipoint BFD session can be set up from a head-end node to a set of tail-end nodes enclosed by a multicast address. Sending BFD Control packets at regular intervals, as determined by the TX value in the BFD Control packet header, allows tail-end nodes to detect loss of connectivity with the head-end node. Three possibilities to notify the head-end node are discussed in [7] in case of loss of multicast connectivity: (1) (unreliable) asynchronous BFD feedback of the tails in case of failure, (2) the head periodically triggers the tails to reply (poll-sequence) with session status, and (3) request a poll-sequence but in case of failure initiate a unicast BFD session-setup (on-demand unicast BFD session). Whereas these multipoint BFD extensions constitute definitely a step forward, they still inherit on the characteristics of the IP layer (or the MPLS layer) and do not account for the specific nature of an Ethernet network. For the given network depicted in Figure 1a, this makes the

[2] No matter the underlying Ethernet LAN topology, at the IP layer nodes are only at single hop distance from each other, resulting into the applicability of LBL BFD sessions.

detection mechanism traffic dependent: point-to-point BFD sessions (for unicast traffic) or multipoint BFD sessions (for multicast traffic). Thus point-to-point BFD sessions are still required for detecting liveliness of links over which unicast traffic is transmitted towards its corresponding destination address. In particular, on a multi-access link/network, BFD Control packets are transmitted with the source and destination addresses set as part of the corresponding IP subnet.

Therefore, we suggest to extend the BFD specification such as to take benefit of the properties specific to Ethernet technology and keep BFD mechanism independent of the traffic exchanged between the two end-points of the session. It is important to underline that the proposed mechanism is not transposable at the IP level because multipoint BFD sessions can only "detect" failures for multicast traffic forwarding, i.e. upon failure, no conclusion can be drawn for what it concerns unicast traffic.

3 BFD over Ethernet

Multi-access broadcast capable environments such as Ethernet LAN segments inherently allow for reaching any port (identified by their MAC address) within the network. Instead of encapsulating the BFD packets into UDP messages over IP datagrams, we suggest to encapsulate BFD packets directly into an Ethernet MAC frame. In this scheme, the end points of the BFD sessions are directly identified by the MAC addresses of the corresponding interfaces[3]. As depicted in Figure 1a, whereas the edge routers are at multi-hop distance in the Ethernet layer. If we apply BFD failure detection at the Ethernet layer between the given edge routers at the Ethernet layer, we have two choices: (1) using E2E BFD sessions over Ethernet, and (2) using Multipoint BFD (MP BFD) sessions over Ethernet. The first option results again into the N-square scalability problem as described in Section 2.2. Indeed, except the addressing specifics, LBL BFD sessions over IP are analogue to E2E BFD sessions over Ethernet. The second option is further detailed in Section 3.1.

3.1 Bi-directional Failure Using Multipoint BFD over Ethernet

True failure detection in an Ethernet LAN network as illustrated Figure 1a, needs bidirectional failure detection capability between all nodes, taking advantage of the multi-access property of the medium, independently of the properties of the traffic being sent on the network. Setting up a single multipoint BFD (MP BFD) session per node at the Ethernet layer is therefore sufficient. The scheme works as follows: (1) Each node sets up a single MP BFD session for which control packets have as source MAC address the local outgoing interfaces address and use a well-known group MAC address as destination MAC address, (2) Each node listens to the group MAC address on which they periodically receive MP BFD control packets from $(n-1)$ sources. For this purpose, a well-known group

[3] Source and destination addresses are only needed during BFD session setup, running sessions use the discriminator field (see [8]).

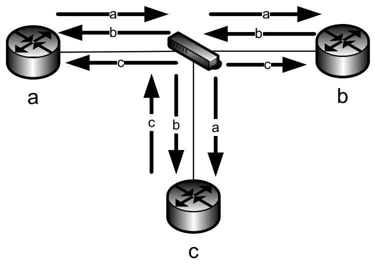


Fig. 3. Asynchronous mode in MP BFD

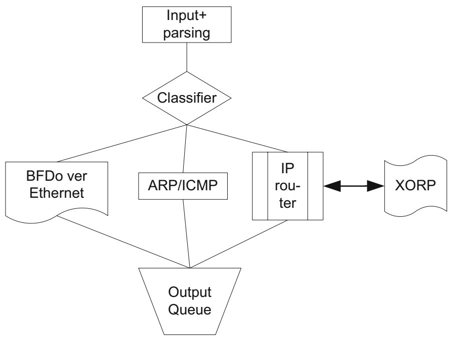
MAC address is dedicated to the BFD protocol such that "edge routers" can listen from senders. The actual group MAC address is taken from the set of reserved code-points for MAC group addresses as specified by IEEE standards[4]. Setting up the MP BFD session can be performed either manually for every node, or triggered for all nodes once a first MP BFD session has been set up (the group MAC address from the first session can then be re-used in order to trigger the other MP BFD sessions). The MP BFD session reaches the Up-status, when all outgoing MP BFD Control packets declare the session to be Up. Every node connected to the Ethernet network receives thus $(n-1)$ MP BFD Control Packets having an Up-status. In case one of these nodes does not receive any BFD Control Packet from one (or more) sources after the calculated Detection Time, it declares the MP BFD session Down. This means that the declaring node has lost connectivity with the source of the MP BFD session. The Detection time is defined as the period of time without receiving BFD packets. As for BFD for asynchronous mode, the Detection Time is calculated as follows: at the receiving (local) node, this time is equal to the value of DetectMult received from the remote node, multiplied by the agreed transmit interval of the remote node[5]. The DetectMult value determines the (minimum) number of missing packets in a row before declaring the session Down. Figure 3 illustrates the scenario where 3 edge routers are interconnected by a single switch[6]. Each edge router connected to the multi-access broadcast capable network maintains thus one transmitting MP BFD head session and keeps listening tail sessions to $(n-1) = 2$ other MP BFD head sessions.

On the contrary to existing mechanisms (see Section 2), the formulated Ethernet-specific MP BFD scheme allows a given receiver to detect unreachability of the sender, once the receiver does not hear from one (or a set of) sources, i.e. it does not receive BFD control packets during a pre-determined period. Upon detection (and declaration of the corresponding session down), the receiver assuming bidirectionality in the Ethernet link layer, declares the logical link to that address Down and can trigger re-routing at the IP layer

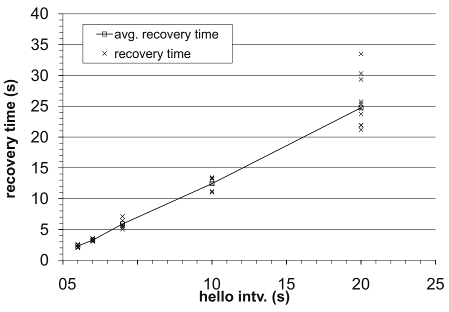
[4] http://standards.ieee.org/regauth/groupmac/tutorial.html

[5] The greater of bfd.RequiredMinRxInterval and the last received Desired Min TX Interval (see [8]).

[6] $n = 3$ and replace the primary LAN by a single switch from Figure 1.



(a) Node architecture in emulation



(b) OSPF reconvergence time

Fig. 4. Emulation

(e.g. IPFRR). Applied on the network from Figure 1a, re-routing can imply to route over the backup LAN instead of routing over the primary LAN (or vice versa).

4 Performance in Emulation

In order to evaluate the performance of the proposed MP BFD protocol scheme over Ethernet, we implemented it in an emulation environment. This experimental implementation allows us to benchmark the added value of the proposed mechanism in terms of time and resource consumption efficiency.

4.1 Architecture and Environment

The implementation of the MP BFD protocol over Ethernet is designed as an element in the Click Modular Router ([9]) framework. The Click software architecture allows for building flexible and configurable routers. A Click router is assembled from packet processing modules called elements that can be linked to each other into a directed graph representing the router configuration. Because Click does not comprise an OSPF-element as part of its suite of components, we interconnected the Click router tables with the OSPF daemon from XORP ([3]). As a result, an edge router from Figure 1 can be emulated by a Linux PC, running both Click and XORP. A simplified node architecture of such router is shown in Figure 4a. The resulting emulation network set up within the Emulab environment ([15]) runs on the virtual wall of ilab.t at IBBT (where each node is emulated on a dual core CPU of 2.0GHz).

4.2 BFD over Ethernet

BFD failure detection has integrated into a new element of the Click Modular Router software. As described in Section 3, the implementation does not rely on UDP/IP for encapsulation of the BFD control packet. Instead, the BFD control packet is directly encapsulated into an Ethernet frame using a custom (configurable) Ethertype[7]. It is important to underline once again that the encapsulation scheme is not an implementation specific decision but plays a critical role in the protocol architecture. Indeed, running MP BFD over UDP/IP would limit its applicability outside of the targeted scope of this paper. The element supports the following BFD over Ethernet modes: Link-By-Link (LBL) BFD sessions, End-to-End (E2E) BFD sessions, E2E BFD sessions over Carrier Ethernet tunnels (see [14]), and MP BFD sessions (3.1). The Click handler interface triggers the setup of BFD sessions. The setup can also be triggered using a socket interface, such that new BFD sessions can be triggered from external applications (such as Dragon GMPLS [14]). With the latter it is possible to send a feedback signal using the same socket interface. The arguments needed to set up a BFD session are: (1) the MAC address of the local interface (source node) and the remote interface (destination node), the latter can be set to a group address for multipoint BFD as explained in Section 3.1), and (2) the local interface identifier that leads to the remote interface (e.g. eth1).

To test the CPU performance of our implementation, we set up a single hop BFD session and hold it for 30 min using equal transmission (TX) and receiver (RX) timer value. The upper left side of Figure 5 shows the average CPU usage compared to the configured RX and TX values. Even with an experimental implementation, the trend shows an exponential decay for decreasing RX and TX values.

In order to determine the lowest possible detection times (time performance) that are possible using our implementation, the following experiment was set up on a ring topology of 19 nodes. E2E BFD sessions over Ethernet were set up with increasing hop-sizes, varying from 1 to 18 hops. Using equal values for the transmission (TX) and receiving (RX) timers, we started setting both values at both ends at 200 ms, using a multiplier equal to 3 (see Section 2.2). Next we waited for 15 minutes. If the session did not time out for a single time, we considered the session to be stable, then we decreased both timers in steps of 5 ms. Meanwhile traffic streams varying from 100 Mbps to 1Gbps were sent through the Click nodes. Using this methodology, the lowest, stable RX and TX values that we found compared to the length of the paths they monitored (number of hops) is shown in the lower left side of Figure 5. This figure shows that failure detection times of 15 ms were possible on our emulated network.

4.3 XORP OSPF Performance

A preparatory experiment consisting of 8 individual runs using XORP and Click on the topology, confirmed the theoretical reconvergence times stated in Section 2.1. As illustrated in Figure 4b we found the fastest reconvergence time of about

[7] The Ethertype field of the Ethernet frame header determines what payload is carried by the Ethernet frame. Standard values are 0x800 for IP or 0x8100 for VLAN-tagged Ethernet frames.

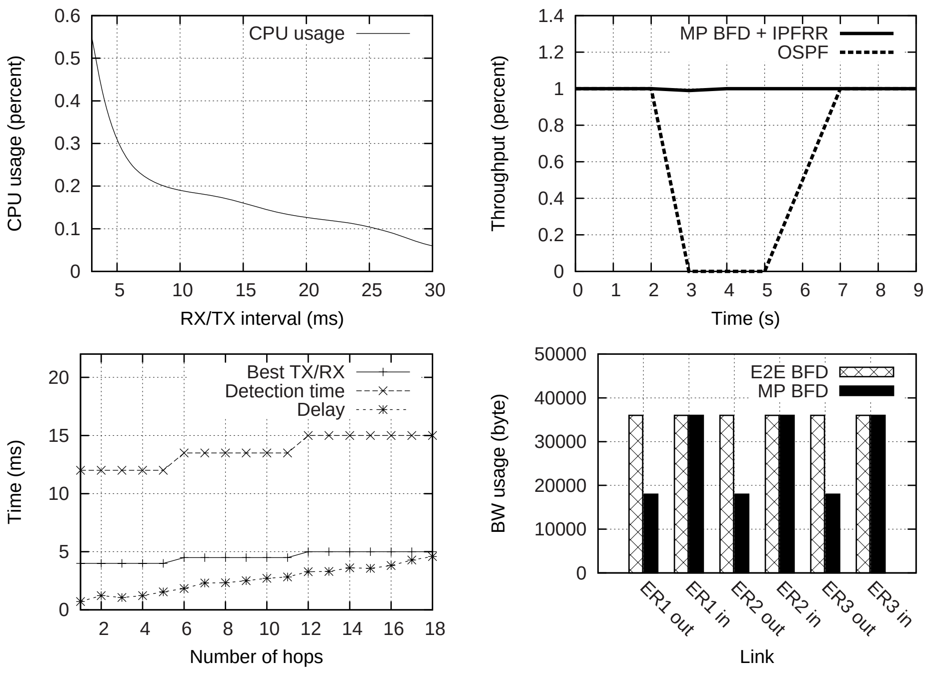


Fig. 5. Emulation results

2.5 seconds on average by using 1 second for the HelloInterval and the double for the RouterDeadInterval. When the HelloInterval was increased, we found a linear trend in the reconvergence time, as it can be observed from Figure 4b.

4.4 OSPF vs. MP BFD over Ethernet with IPFRR

For the purpose of performance comparison between OSPF Hello vs. MP BFD over Ethernet in combination with IPFRR recovery mechanism, we consider the topology depicted in Figure 1a using 3 LANs/edge routers ($n = 3$). The goal of this experiment is to compare throughput performance of standard OSPF-based solution vs. MP BFD over Ethernet in combination with IPFRR when a link fails in the primary LAN. In our setup, the TX and RX timers of the MP BFD sessions are set to 5ms (with multiplier=3) and a failure is introduced at the third second. The upper right side of Figure 5 shows the throughput graph (for constant traffic flow of 500 Mbps) of a pure OSPF-based solution compared to the approach combining MP BFD with IPFRR. Where less than a percent packet loss happens for the latter during the third second (failure detection time took 15 ms, IPFRR-switchover took another 15 ms), the pure OSPF-based solution resulted in 100 percent packet loss for more than 2.5 seconds (resulting from the reconvergence time). This confirms the expected gain in time performance. Figure 6 illustrates the time that is resulting from OSPF failure detection, OSPF

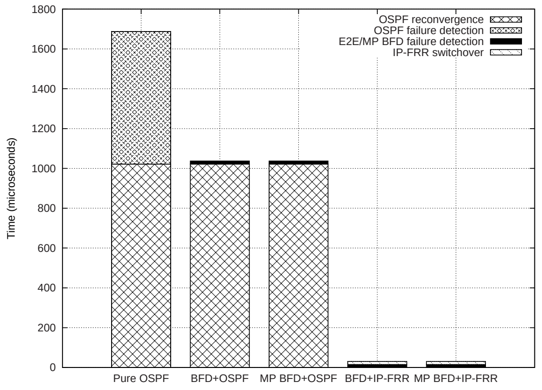


Fig. 6. Time comparison

reconvergence, (MP) BFD failure detection and IPFRR switchover. The lower right side of Figure 5 shows the gain in resources, comparing MP BFD with E2E BFD over Ethernet for the given base network with 3 edge routers (ER's). The lower right part of Figure 5 makes clear that MP BFD consumes half of the bandwidth less on the outgoing direction of the link towards the primary LAN. This obtained result is as expected, because for the outgoing direction MP BFD relies on the multiplication functionality of the Ethernet LAN.

In summary, the conducted experiments have proven the gain in terms of resource consumption and time performance of our implementation running (MP) BFD over Ethernet. This closes the gap of a highly needed feature of IP routers interconnected by LAN networking needing fast recovery.

4.5 Scalability Analysis of MP BFD

Here below, we analyze the scalability of the MP BFD detection mechanism. Given our base scenario of Figure 1a, investigating the scaling of the proposed mechanism involves two main dimensions:

- The *size (of the primary and backup LAN) topology*, which influences: i) the ability to recover within the LAN using RSTP-based mechanisms and ii) the values to be used for the RX and TX timers for the running BFD-sessions between the edge routers. Because neither failure detection, nor recovery using RSTP is envisioned (for reasons specified in Section 2.1) MP BFD does not suffer from the scaling effects of the LAN-topology with respect its influence on RSTP. However larger LAN-topologies do have an effect on the RX and TX-timers that can be used in the MP/E2E BFD configuration, as these timers need to be higher than the total delay (sum of propagation, transmission, forwarding, and scheduling delay) from sender to receiver.
- *Interconnecting more edge routers* (increasing n) to both the primary and the backup LAN, which scales as follows (independently of the underlying LAN topology): i) Linearly increasing set of transmitting MP BFD sessions (using

the LANs broadcasting properties): 1 transmitting MP BFD session per edge router, ii) Squared increase with respect to the number of listening MP BFD sessions: every edge router listens to all other edge routers transmitting MP BFD frames, i.e. n(n-1) sessions.

With respect to the transmitting side, the solution has better scaling properties than any existing failure detection mechanism between n network nodes. The number of edge routers does not affect the timers' values that can be used for setting up the MP BFD sessions because their settings mainly depend on the diameter of the network topology. Indeed, independently of the number of edge routers, from the moment a failure is detected at the receiving nodes, the MP BFD session is declared Down by that node. This detection time is only determined by the DetectMult multiplied by the TX-time whose setting depends on the diameter of the LAN topology. This diameter determines the total delay incurred by the BFD packets transmitted from the sending node and the receiving nodes. The LAN's inherent broadcasting capability ensures that only one BFD frame needs to be transmitted from the detecting node while guaranteeing that the frame will be multiplied on branching points within the LAN network.

5 Conclusion and Future Work

This paper has shown the shortcomings of existing mechanisms for detecting failures between IP routers interconnected by Ethernet LANs. Therefore, a new Multipoint BFD variant over Ethernet has been proposed and evaluated by means of emulation through Click and XORP. The presented MP BFD scheme: (1) acknowledges the special character of the underlying data plane, by running directly over Ethernet, (2) allows faster failure detection than the classical OSPF Hello mechanism, and (3) uses less resources compared to E2E BFD sessions. The defined mechanism enables a faster trigger of recovery switching from a primary to a backup Ethernet LAN in case a failure is detected in the former. Nevertheless, it does not take into account the possibility that only connectivity between a subset of the edge routers needs to be recovered, as it declares the entire MP BFD session Down and triggers the switchover for all edge routers. Future work could consist in detecting and determining which edge routers are affected by the failure by correlating the errors, such that only a partial switchover is needed. Another aspect that requires further investigation is the verification of the scaling performance that would allow confronting the analysis detailed in Section 4 against a large-scale execution of the proposed MP BFD scheme with IPFRR.

Acknowledgments

This research is partly funded by the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen) through the TIGER project in the European CELTIC framework and the FP-7 project BONE.

References

1. Aggarwal, R., Kompella, K., Nadeau, T., Swallow, G.: BFD For MPLS LSPs. Internet-Draft draft-ietf-bfd-mpls-07, Internet Engineering Task Force, Work in progress (June 2008)
2. Atlas, A., Zinin, A.: Basic Specification for IP Fast Reroute: Loop-Free Alternates. RFC 5286, Internet Engineering Task Force (September 2008)
3. Handley, M., Hodson, O., Kohler, E.: Xorp: an open platform for network research. SIGCOMM Comput. Commun. Rev. 33(1), 53–57 (2003)
4. IEEE. Draft standard for local and metropolitan area networks: Media access control (mac) bridges (revision of ieee std 802.1d -1998 incorporating ieee std 802.1t -2001 ieee std 802.1w -2001) (replaced by 802.1d-2004). IEEE Std P802.1D/D4, pages– (2003)
5. Katz, D., Ward, D.: BFD for IPv4 and IPv6 (Single Hop). Internet-Draft draft-ietf-bfd-v4v6-1hop-08, Internet Engineering Task Force, Work in progress (March 2008)
6. Katz, D., Ward, D.: BFD for Multihop Paths. Internet-Draft draft-ietf-bfd-multihop-06, Internet Engineering Task Force, Work in progress (January 2008)
7. Katz, D., Ward, D.: Bfd for multipoint networks. Internet-Draft draft-katz-ward-bfd-multipoint-01, Internet Engineering Task Force, Work in progress (January 2008)
8. Katz, D., Ward, D.: Bidirectional Forwarding Detection. Internet-Draft draft-ietf-bfd-base-08, Internet Engineering Task Force, Work in progress (March 2008)
9. Kohler, E., Morris, R., Chen, B., Jannotti, J., Kaashoek, M.F.: The click modular router. ACM Trans. Comput. Syst. 18(3), 263–297 (2000)
10. Kou, Z.: Update to OSPF Hello procedure. Internet-Draft draft-kou-ospf-immediately-replying-hello-02, Internet Engineering Task Force, Work in progress (January 2007)
11. Moy, J.: OSPF Version 2. RFC 2328, Internet Engineering Task Force (April 1998)
12. Reid, A., Willis, P., Hawkins, I., Bilton, C.: Carrier ethernet. IEEE Communications Magazine 46(9), 96–103 (2008)
13. Shand, M.: IP Fast Reroute Framework. Internet-Draft draft-ietf-rtgwg-ipfrr-framework-08, Internet Engineering Task Force, Work in progress (February 2008)
14. Tavernier, W., Papadimitriou, D., Colle, D., Pickavet, M., Demeester, P.: Emulation of gmpls-controlled ethernet label switching. In: International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCom), Washington, USA (2009)
15. White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., Hibler, M., Barb, C., Joglekar, A.: An integrated experimental environment for distributed systems and networks. In: Proc. of the Fifth Symposium on Operating Systems Design and Implementation, pp. 255–270. USENIX Association, Boston (2002)

Traffic Classification Based on Flow Similarity[*]

Jae Yoon Chung[1], Byungchul Park[1], Young J. Won[1], John Strassner[1,2], and James W. Hong[1]

[1] Dept. of Computer Science and Engineering, POSTECH, Korea
[2] Waterford Institute of Technology, Waterford, Ireland
{dejavu94,fates,yjwon,johns,jwkhong}@postech.ac.kr

Abstract. Due to the various masquerading strategies adopted by newer P2P applications to avoid detection and filtering, well-known port mapping techniques cannot guarantee their accuracy any more. Alternative approaches, application-signature mapping, behavior-based analysis, and machine learning based classification methods, show more promising accuracy. However, these methods still have complexity issues. This paper provides a new classification method which utilizes cosine similarity between network flows.

Keywords: Traffic classification, traffic monitoring.

1 Introduction

The large amount and diversity of today's Internet traffic requires more efficient and accurate traffic classification methods. For example, many modern network applications, such as Voice over IP (VoIP), use features such as dynamic port allocation, [*]incorporate ephemeral port allocation (e.g. peer to peer (P2P) and web storage service applications); some applications also allocate their traffic to well-known ports to avoid detection (e.g. P2P and YouTube). Therefore, port-based classification [1] cannot guarantee that it can correctly identify applications by inspecting the ports used [5]. To overcome this issue, alternative traffic classification techniques, such as signature-based classification [4][7][8][13], behavior-based modeling [3][6][12][18], and machine learning-based classification [9][10][11], are introduced.

In this paper, we propose a new traffic classification methodology that translates packet payloads into vectors and classifies the application traffic based on the result of similarity comparisons among the series of vectors. Our work is inspired by document classification [15], which is widely used in the Natural Language Processing (NLP). The basic idea of comparing packet payload vectors is analogous to document classification; however, there are fundamental differences between document content and packet payloads. Whereas a document consists of basic components called words which have precise meaning, packet payloads consist of binary data, of which most has no pre-defined meaning. Therefore, we try to convert packet payloads into a series of vectors reflecting this difference.

[*] This work was partly supported by the IT R&D program of MKE/IITA [2008-F-016-02, CASFI] and WCU program through the KSEF of MEST, Korea [R31-2008-000-10100-0].

The remainder of this paper is organized as follows. Section 2 provides an overview of application traffic classification methodologies. Section 3 describes our classification methodology based on flow similarity. The evaluation results with real-network traffic traces are presented in Section 4. Finally, the concluding remarks and future work are discussed in Section 5.

2 Related Work

Many previous studies on application traffic identification have considered well-known network ports [1] as the primary means for identifying traffic. However, these traditional port mapping methods now have a much lower identification accuracy, due to three factors: (1) increased use of dynamic port allocation, (2) the increased use of ephemeral ports, and (3) the use of well-known ports (e.g., a P2P application may use port 80, even though it is not using HTTP) to avoid detection and filtering. Such phenomena have degraded the accuracy of port mapping mechanisms. Moore et al. [2] hypothesized that the accuracy of port mapping methods is not more than 50~70%. Our technique does not use port identification for these reasons.

An alternative approach, signature-based classification, has been proposed to address the drawbacks of port-based classification [4][7][8][13]. Packet signature-based classification inspects the packet payload to identify the application. Other types of signatures-based mechanisms construct an application signature based on communication patterns or other identifying features. In either case, a signature is static and can be used to distinguish applications from each other. Once a reliable signature is available, this approach can identify the target application very accurately. However, generating the signature requires exhaustive searching and time consuming.

Other research focused on behavior-based classification methods [3][6][12][18], which characterize the behavior of communicating nodes using connection related information, such as tuples that identify the particular entity (e.g., in terms of the source destination addresses as well as other metrics, such as protocol number). BLINC [3] profiles host behavior using destination IP addresses and port allocation patterns; application traffic is then identified by comparing it with existing profiles. Behavior-based classification is efficient for detecting anomalies or worm-like P2P traffic [6], and is available with encrypted packet payloads. These methods do not rely on any information present in the payload; thus, its accuracy is often questionable. In addition, the time complexity is problematic due to obscure behavior patterns when dealing with a high volume of traffic.

More recently, machine learning-based approaches [9][10][11] using statistical information of the transport layer have been introduced. These classification methods can use unsupervised, supervised, reinforcement, or hybrid learning mechanisms to automatically classify packets or even flows based on (for example) their statistical characteristics, instead of fixed data like port numbers. In other words, different types of applications can be distinguished by their communication characteristics such as connection duration, inter-packet arrival time, and packet size. Since most machine learning-based classification methods only use statistical information, these methods can also be used with encrypted traffic. However, machine learning approaches have their own set of issues that our approach does not have, such as effective traffic feature selection. We focus on how to utilize the payload data without heavy packet inspection while achieving reasonable accuracy.

3 Proposed Methodology

This section explains our traffic classification method. Its strength is that it does not rely on extracting any packet information (e.g., from the IP header). We illustrate our payload vector converting, vector comparison, and flow comparison methods.

3.1 Vector Space Modeling

Vector Space Modeling (VSM) is an algebraic model from the NLP research area that represents text documents as vectors, and is widely used for document classification. The goal of document classification is to find a subset of documents from a set of stored text documents D which satisfy certain information requests or queries Q. One of the simplest ways to do this is to determine the significance of a term that appears in the set of documents D [17]. This is quite similar to traffic classification, which identifies and classifies network traffic according to the type of application that generated the traffic. The main issue in VSM is how to determine the weight of each term. Salton et al. [15] have proposed some recommended combination of term-weighting components: (1) Term Frequency Component (b, t, n), (2) Collection Frequency Component (x, f, p), and (3) Normalization Component (x, c). However, these combinations are not suitable for traffic classification, because these were derived based on the needs of document classification. We therefore have to consider other weighting methods in order to use VSM for traffic classification; these methods should correspond to those portions of the traffic payload that are application-specific and disregard other data.

3.2 Similarity

Once packet payloads are translated into vectors, the similarity between packet payloads can be calculated by computing the similarity between vectors using many metrics. We used one such measure – the cosine similarity, which measures the similarity between two vectors of *n* dimensions by finding the cosine of the angle between them. The similarity value ranges from -1 (meaning exactly opposite) to 1 (exactly same) and 0 indicates independence. We normalized payload vectors so that the similarity can be calculated by the dot product, and it approaches one if the vectors are similar and zero otherwise. Formula (1) is a mathematical definition of cosine similarity and Figure 1 illustrates the cosine similarity between payload vectors.

Our payload vectors have very high dimensionality (for example, 65536), so probabilistically they are very sensitive (the detail of vector converting is described in the next section). If two payload vectors are generated by the different applications, the contents of each payload consist of distinct word (or binary) sequence and their vectors are also very different. Because most signatures of the application traffic are a small portion of the payload data, we may ignore the other part of the payloads signatures which is actually arbitrary binary data.

$$Similarity(\mathrm{p}_1, \mathrm{p}_2) = \frac{\mathrm{V}(\mathrm{p}_1) \cdot \mathrm{V}(\mathrm{p}_2)}{|\mathrm{V}(\mathrm{p}_1)||\mathrm{V}(\mathrm{p}_2)|}. \quad (1)$$

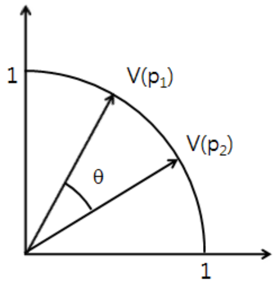


Fig. 1. Cosine similarity illustrated

3.3 Payload Vector

We define the *word* of a payload as follows. If a window size is longer, the payload vector can reflect the order of signature in the payload. However, we have to handle the high dimension vectors caused by increasing length of words.

- **Definition 1:** We define *word* as a payload data within an i-bytes sliding window where the position of the sliding window can be 1, 2 … n-i+1 with n-length payload. The size of whole representative words is 2^{8*i}.

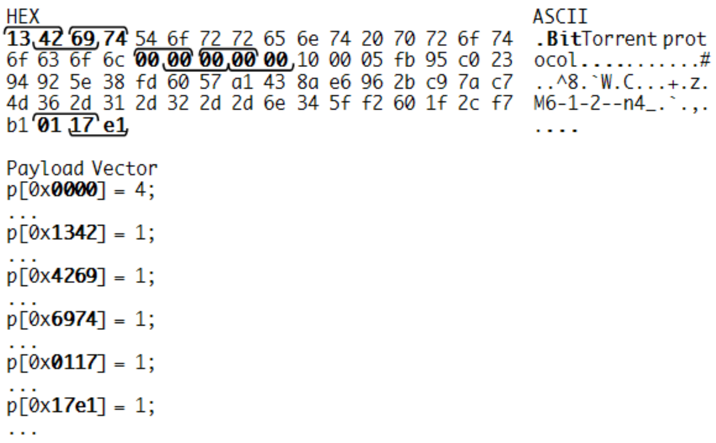
We can make the term-frequency vector by counting words within a payload. This vector is called the *payload vector*, which is the same as the document vector in NLP research.

- **Definition 2:** When w_i is the count of the i-th word that appears repeatedly in a payload, the *payload vector* is

$$Payload\ Vector = [w_1\ w_2\ ...\ w_n]^T. \quad (2)$$

where n is the size of whole representative words.

Our study focuses a window size of two, which means that the word size is two, because it is the simplest case for representing the order of content in payloads. When the word size is 2-bytes, the size of all representative words is 2^{16}=65536. Thus, the payload vector has 65536 dimensions. Figure 2 is part of a payload vector that is converted from a BitTorrent packet. In most of cases, we can see that the vector is very sparse because the dimension of the payload vector is much higher than the length of the payload.



```
HEX                                                ASCII
13 42 69 74 54 6f 72 72 65 6e 74 20 70 72 6f 74   .BitTorrent prot
6f 63 6f 6c 00 00 00 00 00 10 00 05 fb 95 c0 23   ocol...........#
94 92 5e 38 fd 60 57 a1 43 8a e6 96 2b c9 7a c7   ..^8.`W.C...+.z.
4d 36 2d 31 2d 32 2d 2d 6e 34 5f f2 60 1f 2c f7   M6-1-2--n4_.`.,.
b1 01 17 e1                                       ....

Payload Vector
p[0x0000] = 4;
...
p[0x1342] = 1;
...
p[0x4269] = 1;
...
p[0x6974] = 1;
...
p[0x0117] = 1;
...
p[0x17e1] = 1;
...
```

Fig. 2. Converting BitTorrent packet into payload vector

3.4 Packet Comparison

We have to calculate cosine similarities between payloads. In the document classification research, some studies enhance not only words that appear very frequently in a document, but also words that appear very rarely in the documents. When we calculate similarities, we have to define the term weight differently from the recommendations in [15]. However, if we apply the 'term frequency – inverse document frequency' weighting rule in our packet comparison process, the classification accuracy can be lower than that of typical signature-based classification approaches due to frequent appearances of signature. Therefore, we define weight values as the number of word appearances in a payload. These weight values are empirically updated.

If a packet is composed by w*, which appears in more than half of the payload length, then the similarity between a given packet and another packet that has at least one w* is always high. This is true even if those packets are generated by different applications. Conversely, it is the most important evidence for identifying a flow if a certain packet is composed of one word. To overcome mis-classification caused by improper weighting to a dominant word, we define the following exceptional case. We use the hard clustering approach; thus, the similarity between the packets containing the same w* is either zero or one.

- **Exception Case:** Let w* be the dominant word in a packet. If a packet is mostly filled with w*, the content of another packet in being comparison would also be mostly filled with w*.

3.5 Flow Comparison

In the scope of this paper, traffic classification relies on how to define and distinguish the similar flows. Flow here refers to bidirectional-flow containing both in and out packets from host.

3.5.1 Payload Flow Matrix

Formula (3) defines the *payload flow matrix (PFM)*. The i-th row of a PFM is the i-th packet of the flow. PFM is a $k \times n$ matrix, where k is the number of packets and n is the dimension of payload vectors. It is defined as follows.

- **Definition3:** *Payload flow matrix (PFM)* is

$$PFM = [\,\overrightarrow{\mathrm{p}_1}\,\overrightarrow{\mathrm{p}_2}\,...\,\overrightarrow{\mathrm{p}_k}\,]^{\mathrm{T}} \tag{3}$$

where $\overrightarrow{\mathrm{p}_\mathrm{i}}$ is *payload vector* as defined in Section 3.3.

3.5.2 Collected PFMs

Figure 3 illustrates a collection of m PFMs ($k \times n \times m$ matrix). Each layer represents a typical flow of each application. The collected PFMs are empirically accumulated according to formula (4), where α is a constant weight value. They are not only the basis for classifying application traffic but also for defining the term frequency weighting. For comparison aspects, we do not need a large amount of PFMs as a training set, but we need to know the typical flows of each application.

$$Collected\ PFMs = \alpha * new\ PFM + (1-\alpha) * Collected\ PFMs \tag{4}$$

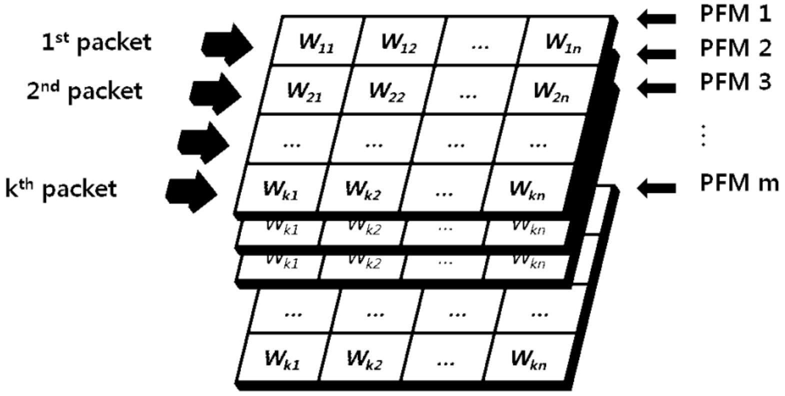


Fig. 3. Collected payload flow matrices

3.5.3 Flow Similarity Calculation

Figure 4 shows how to compare a new flow with two previously collected PFMs belonging to two different applications. The vertically linked circles refer to the packets in each flow. The dash line implies the comparison sequence between the packets. The first packet of the new flow is compared to only the first packets of each PFM. It means that we keep the order of packets within the flows. We assume that the two flows are well matched even if we keep the order of comparison sequences. Algorithm 1 is the pseudo-code representation of flow similarity calculation according to

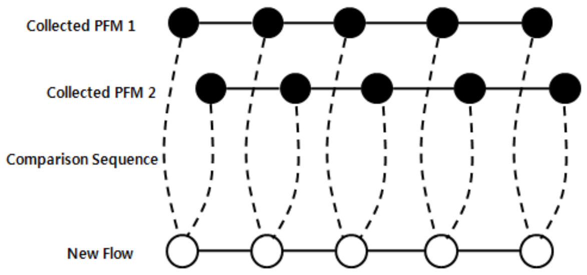


Fig. 4. Flow and packet comparisons

Algorithm 1. Pseudo code for flow similarity calculation

```
flow_similarity(collected_flows, input_flow)
begin
    N = the first N packets in the flow;
    n = the number of flows in the collected_flows;

    for i=1 to n,
        compared_flow = collected_flows(:,:,i);
        for j=1 to N
            weight = compared_flow(j,:)/row_sum(compared_flow(j,:));
            weighted_flow = input_flow(j,:).*weight;
            pkt_sim(j,i) = similarity(compared_flow(j,:), weighted_flow);
        end
    end

    /*
     * W_* is scoring weight
     * M is number of scoring packets
     */
    flow_similarity =
        pkt_sim(1, :)*W_1 + pkt_sim(2, :)*W_2 + ... + pkt_sim(M, :)*W_M;

    return flow_similarity;
end
```

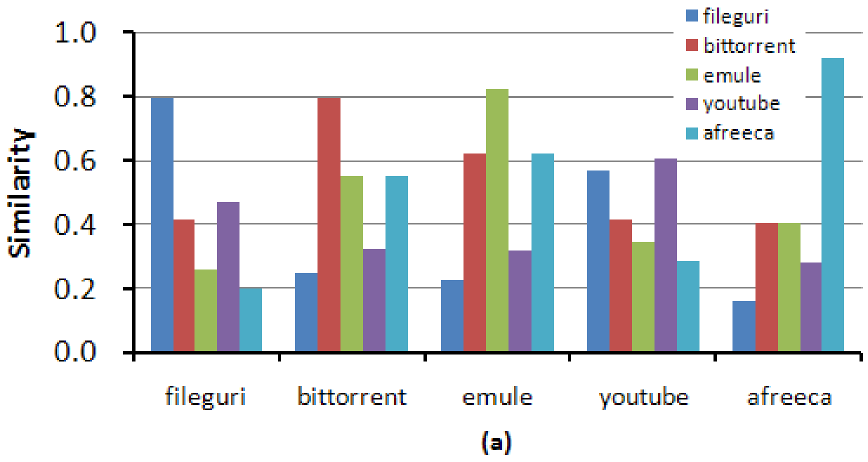
Figure 4. We can determine the similarity scores for each collected PFM. If the score between the PFM 1 and new flow is higher than the score between the PFM 2 and new flow, we conclude that the new flow belongs to the application of PFM 1. However, when its similarity score does not exceed a certain threshold value, the new flow becomes unknown.

4 Evaluation

Our evaluation divides into two steps. The first step is to determine the best-fitting parameters for accurate traffic classification. Then, we evaluate the accuracy of our method using real network traffic from our campus backbone.

For flow similarity calculation, the following parameters should be determined: (1) weighting scheme, and (2) the number of first N packets in a flow. We select five target applications and compare the similarity scores for every pair of target applications with respect to various parameter settings. The selected target applications are BitTorrent [19], Emule [20], YouTube [21], Fileguri (P2P file-sharing application) [22], and Afreeca (P2P video streaming application) [23]. They are chosen based on their usage popularity in our campus network. In addition, YouTube and Afreeca are video streaming services, which enable us to check whether our method is valid for web-based video streaming detection.

We generated five flows of each application at an end host and *collected PFMs* (which were described in section 3.5.2). We captured another flow generated by one of target applications and calculated the similarity between this flow and five *collected*



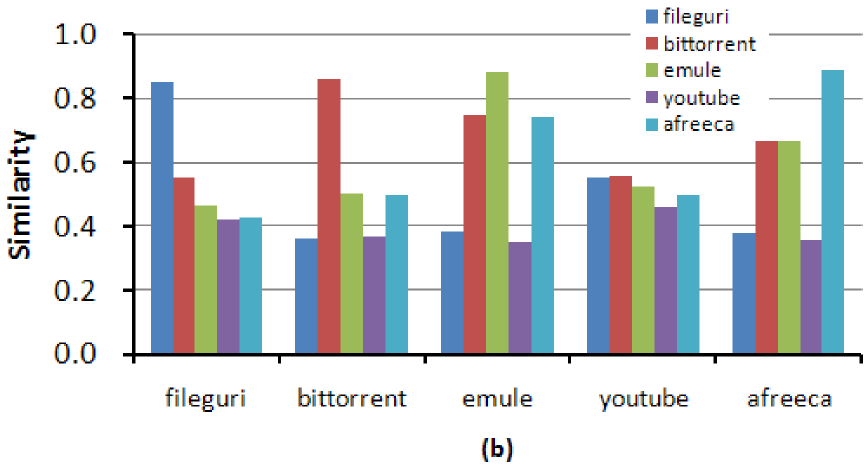
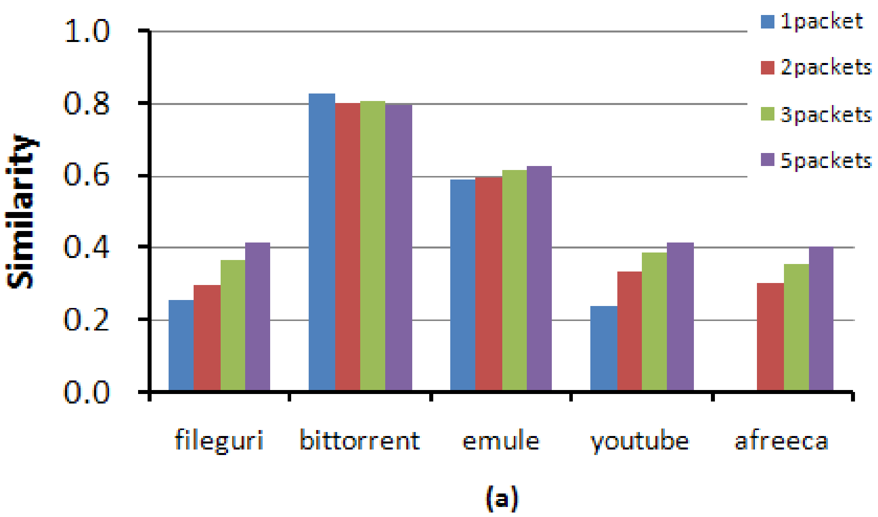


Fig. 5. Exponentially decrease weight (a) vs. uniform weight (b)

PFMs with two different weighting schemes; our two schemes are (a) exponentially decreasing weight and (b) uniform weight. Figure 5 shows the variance of similarities between application traffic is larger if the weights are exponentially decreased. In other words, the results demonstrate the fact that the first few packets have remarkable characteristics or signatures that distinguish each flow [14].

We calculated the similarities between the single flow of a target application and *collected PFMs* with various N. Figure 6 (a) shows that the BitTorrent flow and the *collected PFM* of BitTorrent is almost the same, although N is changed from 1 to 5.



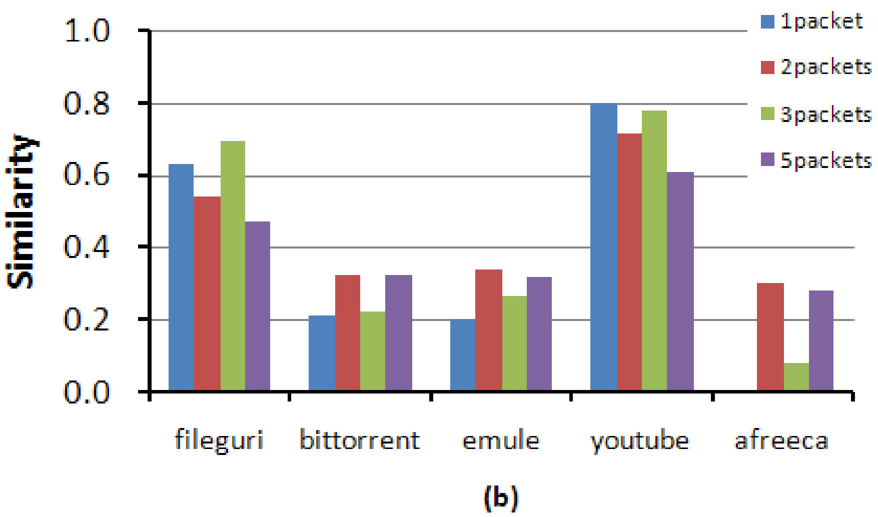


Fig. 6. Similarity with respect to changing first N (=1, 2, 3, 5) packets when BitTorrent (a) and YouTube (b) are the flows being compared to

Figure 6 (b) shows that YouTube flows are basically similar to Fileguri flows because both applications use the HTTP protocol to request download files or videos; this results in almost identical payload formats. Figure 6 (b) infers that a large N can help to distinguish flows that are generated by different applications having similar content. If N is too large, however, the later packets of a flow can contain arbitrary binary data. This can decrease the similarity, even when the flows were generated by the same application. From these observations, we chose exponentially decreasing weighting and N=2 (excluding data packets) as the best-fitting parameters for classifying selected target applications.

We validated our approach using the campus backbone traffic. The traces were collected from one of the two Internet junctions at POSTECH, a university with a user population of about 3500 people. To avoid any possible packet loss, the monitoring probe equipped with Endace DAG 4.3GE [24] was used to monitor 1 Gbps Ethernet link. The collected trace covers 40 minutes period (23:00 to 23:40) in early 2009. We took a sample of traffic from several end hosts and estimated the results within the whole backbone traffic.

The ground truth traffic is obtained by Traffic Measurement Agent (TMA). TMA is a Windows based client program and is deployed at target endpoints to determine the exact flow information at the origin of the traffic. TMA monitors the network interface of the host. It also summarizes traffic and records log data, including five flow tuples, process name, packet count, and time. The log data is compared with the results of classifications from POSTECH's Internet junction traffic; from this, we can calculate false positives and false negatives and remove them to compute the overall accuracy, as defined by formula (5). Table 1 explains the results of the classification of four applications: BitTorrent, LimeWire, YouTube, and Fileguri.

$$Overall\ Accuracy = \frac{Total\ traffic - (FP\ traffic + FN\ traffic)}{Total\ trafflc} \tag{5}$$

BitTorrent has the most accurate results because it is well classified using the payload signature-based approach. Note that we intentionally mixed YouTube traffic with HTTP traffic used for web browsing to check whether our method can classify YouTube traffic from normal web traffic correctly. Table 2 shows a part of the HTTP and YouTube signaling packets. The similarity between the first packets of HTTP flow and YouTube flow is relatively high, about 0.79. Thus, it is difficult to decide a clear boundary between HTTP traffic and web-based application traffic, such as video streaming and web-storage.

We compare the overall accuracy of proposed method with that of LASER [4], which relies on application-level signatures to classify the traffic (Table 3). The overall accuracy of computing flow similarity improves slightly compared to LASER. However, the false negatives for each application are lower than LASER except Fileguri due to some packet loss while capturing.

Table 1. Classification accuracy using TMA

Application	Classified Traffic (kB)	False Negative (kB)	False Positive (kB)	Overall Accuracy (%)
BitTorrent	202,018	3,361	0	98.36
LimeWire	87,678	2,951	0	96.74
Fileguri	95,804	9,691	0	90.81
YouTube	16,061	0	3,775	80.96
TMA Log Traffic	421,339 kB			

Table 2. HTTP packets vs. YouTube signal packets

HTTP packet contents	YouTube signal packet contents
GET / HTTP/1.1 User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) ... ... Connection: Keep-Alive	GET /videoplayback?sparams=id%2C expire%2Cip%2Cipbits%.... ...... HTTP/1.1 User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) ...... Connection: Keep-Alive

Table 3. Flow similarity approach (FSA) vs. LASER [4]

Application	Approach	False Negative (%)	False Positive (%)
BitTorrent	FSA	1.66	0
	LASER	10.40	0
LimeWire	FSA	3.36	0
	LASER	8.42	0
Fileguri	FSA	10.11	0
	LASER	0.39	0
Overall Accuracy	LASER	97.39%	
	FSA	96.01%	

Finally, we estimate the population proportion with 95% confidence. The confidence interval is (6). The estimated classification accuracy is 96.01±0.0011.

$$\hat{p} - z_{\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}} < p \le \hat{p} + z_{\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}, \tag{6}$$

where $z_{\alpha/2} = 1.96$.

5 Conclusion

This paper proposes a traffic classification approach based on payload vector similarity. We suggest converting payloads into vector representations. Then, we apply a variation of simple document classification approach to classify traffic classification. For validation, we classify several target application traffic from our campus network traffic. The overall classification accuracy shows about 96%. Web-based application traffic is also identified, but there exists a small portion of false positives because it is very similar to HTTP traffic.

The proposed methods are based on the similarity between payload vectors, which is a metric of defining how similar flows are to each other. The exhaustive task of extracting the payload signatures from traffic is not required, and we can represent the similarity between payloads or flows as a simple numerical value. Our approach shows a reasonable performance even when high-volume applications, such as P2P traffic and web-based video streaming, are being measured.

For future work, we will increase the number of applications and develop a real-time agent program. We also plan to perform the same experiments using other similarity metrics.

References

[1] IANA, IANA port number list, `http://www.iana.org/assignments/port-numbers/`

[2] Moore, A.W., Papagiannaki, K.: Toward the Accurate Identification of Network Applications. In: Passive and Active Measurement Conference, Boston, MA, USA, March 31-April 1 (2005)

[3] Karagiannis, T., Papagiannaki, K., Faloutsos, M.: BLINC: Multilevel Traffic Classification in the Dark. In: ACM SIGCOMM 2005, Philadelphia, PA, USA, August 21-26 (2005)

[4] Park, B., Won, Y.J., Kim, M.-S., Hong, J.W.: Towards Automated Application Signature Generation for Traffic Identification. In: IEEE/IFIP Network Operations and Management Symposium (NOMS 2008), Salvador, Brazil, April 7-11, pp. 160–167 (2008)

[5] Karagiannis, T., Broido, A., Brownlee, N., claffy, K., Faloutsos, M.: Is P2P Dying or just Hiding? In: IEEE Globecom 2004, Dallas, Texas, USA, November 29-December 3 (2004)

[6] Kim, S.S., Reddy, A.L.N.: Image-Based Anomaly Detection Technique: Algorithm, Implementation and Effectiveness. IEEE Journal of Selected Areas in Communications 24, 1942–1954 (2006)

[7] Haffner, P., Sen, S., Spatscheck, O.: ACAS: Automated Construction of Application Signatures. In: ACM SIGCOMM 2005, Philadelphia, PA, USA, August 21-26 (2005)

[8] Sen, S., Spatscheck, O., Wang, D.: Accurate, Scalable In-Network Identification of P2P Traffic using Application Signatures. In: International World Wide Web Conference, NY, USA, May 19-21, pp. 512–521 (2004)

[9] Moore, A.W., Zeuv, D.: Internet Traffic Classification Using Bayesian Analysis Techniques. In: International Conference on Measurements and Modeling of Computer Systems, Banff, Alberta, Canada, June 6-10, pp. 50–60 (2005)

[10] Erman, J., Mahanti, A., Arlitt, M., Williamson, C.: Identifying and Discriminating Between Web and Peer-to-peer Traffic in the Network Core. In: International World Wide Web Conference, Banff, Alberta, Canada, May 8-12, pp. 883–892 (2007)

[11] Erman, J., Arlitt, M., Mahanti, A.: Traffic Classification Using Clustering Algorithms. In: SIGCOMM Workshop on Mining Network Data, Pisa, Italy, September 11-15, pp. 281–286 (2006)

[12] Karagiannis, T., Broido, A., Faloutsos, M., claffy, K.: Transport Layer Identification of P2P Traffic. In: Internet Measurement Conference, Taormina, Sicily, Italy, October 25-27, pp. 121–134 (2004)

[13] Choi, T.S., Kim, C.H., Yoon, S., Park, J.S., Lee, B.J., Kim, H.H., Chung, H.S., Jeong, T.S.: Content-aware Internet Application Traffic Measurement and Analysis. In: IEEE/IFIP Network Operations and Management Symposium (NOMS 2004), Seoul, Korea, April 23, vol. 1, pp. 511–524 (2004)

[14] Gummadi, K.P., Dunn, R.J., Saroiu, S., Gribble, S.D., Levy, H.M., Zahorjan, J.: Measurement, Modeling, and Analysis of a Peer-to-Peer Filesharing Workload. In: ACM Symposium on Operating Systems Review, December 2003, vol. 27, pp. 314–329 (2003)

[15] Salton, G., Buckley, C.: Term-weighting Approaches in Automatic Text Retrieval. Information Processing and Management 24(5), 513–523 (1988)

[16] Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, Cambridge (2008)

[17] Luhn, H.P.: A Statistical Approach to the Mechanized Encoding and Searching of Literary Information. IBM Journal of Research and Development, 309–317 (October 1957)

[18] Iliofotou, M., Pappu, P., Faloutsos, M., Mitzenmacher, M., Singh, S., Varghese, G.: Network monitoring using traffic dispersion graphs. In: Internet Measurement Conference, San Diego, CA, USA, October 24-26 (2007)

[19] BitTorrent, `http://www.bittorrent.com/`

[20] Emule, `http://www.emule-project.net/`

[21] YouTube, `http://youtube.com/`

[22] Fileguri, `http://www.fileguri.com/`

[23] Afreeca, `http://www.afreeca.com/`

[24] Endace, DAG 4.3GE, `http://www.endace.com/`

Managing Network-Aware Grid Services on Metropolitan Scale: The SCoPE Experience

Francesco Palmieri[1] and Silvio Pardi[2]

[1] Università degli studi di Napoli Federico II,
[2] INFN Sezione di Napoli
Via Cinthia, 5 – 80126, Napoli, Italy
{fpalmier,spardi}@unina.it

Abstract. An increasing number of high-performance scientific and industrial applications, ranging from tera-scale data mining to complex computational genomics, weather forecast and financial modeling, are starting to take great advantages from the mesh of geographically distributed computing, network and data management resources, commonly referred to as “Grid”. In this paper, we discuss opportunities and challenges in the Grid resource management framework and network control plane design, critical to the provision of network-assisted extensible Grid services empowering a real high performance distributed computing system built on metropolitan-scale dark fiber transport networks, administered within a single domain and offering plenty of cheap bandwidth. Our primary goal is to enhance our existing grid computing infrastructures, currently focused on CPU and storage, to include the network as an integral Grid component that offers reliable, and if possible guaranteed, levels of service. As a proof of concept, we realized within the SCoPE High Performance Computing environment the prototype of a basic “Metropolitan Area” Grid architecture by implementing on the local scale a novel centralized network resource management service supporting a flexible Grid-application interface and several effective network resource reservation facilities.

Keywords: Data Grid, Service management, MPLS, Network-aware Grid.

1 Introduction

The Internet was developed to meet the need for a common communication medium between large, federally funded computing centers. These communication links led to resource and information sharing between these centers and eventually to provide access to them for additional users. The current Grid technology can be viewed as an extension or application of this framework to create a more generic resource sharing context. Such context requires the orchestration of the right data, to the right computation, at the right time and on the right site belonging to the same or to cooperating organizations. Thus the distribution of information and data over time, space, and organizations becomes crucial for Grid applications, more and more often involved in the transfer of immense amounts of information between geographically spread data centers. Unfortunately, the Grid resource management logic has been traditionally based on a partially or totally network oblivious approach in which the computational resources can be chosen from different sites participating to the Grid without inspecting their connectivity characteristics, since Grid services had no visibility of the underlying network. As a direct consequence the involved applications can be subject to failure if the network connections between the computing and service nodes do not perform as required. Hence, a new network-aware approach based on a strict integration between the Grid resource management logic, the requesting application and the network entities that offer the connectivity services is needed. This requires administrative access to network entities and can only applied in a limited scope like a local network or a single administrative domain. Today, beside the big national and international GRID infrastructures, the increasing availability of high performance dark fiber metropolitan area networks (MAN) connecting several large computing farms located on a medium scale geographical distance and operating under the control of a single administrative authority, give us the great opportunity of realizing a new concept of GRID based distributed computational systems, implemented on metropolitan scale, that we can call Metropolitan Area Grids (MAG). Since all the computing and storage nodes will be mutually connected within a single domain, where the organization managing the transport network directly owns administrative control on the underlying network elements, the resulting computing infrastructure will behave as an highly connected distributed system deployed on a LAN, without the performance limitations and oddities introduced in the wide area. Such approach makes the transport infrastructure the main enabling factor of a novel Grid vision aiming at unifying all the distributed computational and storage resources into a common “virtual site” abstraction, so that they can cooperate as if they were in the same Server Farm and LAN. In response to the above requirements, we address some of the key technical challenges related to the provisioning of on-demand end-to-end network connectivity across a metropolitan high-performance Grid infrastructure, by focusing on the deployment of advanced management/brokering services that allow the underlying transport infrastructure to be treated as a first class resource. We implemented a management framework that aims at engineering the most reliable and cost-effective combination of the available networking options, to optimize, as possible, the data transfer operations for all the Grid applications running in an integrated environment. Our MAG transport infrastructure has been enabled, by means of specific network-level technologies, architectural choices and properly designed Grid-to-network interfaces to provide on demand bi-directional data paths between the computing farms and hence between the cooperating Grid nodes. These paths will be either dedicated Layer-2 channels, realizing the abstractions of a transport network behaving as a single virtual switching device, or Layer-3 paths with guaranteed bandwidth, delay, etc. To implement the above facilities we work on a pure “peer-based” model based on the Multi-Protocol Label Switching (MPLS) technology that introduces a circuit-switching paradigm on top of the basic IP packet-switching framework. All the involved services have been almost transparently made available to Grid applications, by means of properly crafted Web Services interfaces, in a way that is totally independent from the underlying network protocols and communication technologies. As a proof of concept some simple and effective MAG prototype services have been implemented in the context of the SCoPE high performance computing project to satisfy the connectivity demands of the most network-hungry e-science applications.

2 The SCoPE Environment

SCoPE (Italian acronym for high Performance, Cooperative and distributed System for scientific Elaboration) is a high performance computing project that aims at developing several applications in the field of fundamental research. It has been co-funded by the Italian Ministry for Education and Research (MIUR), under the EU PON program, on a competitive base, according to the 1575/2004 tender. The SCoPE architecture has been conceived to provide the unification of all the main computational and storage resources already available in the participating sites, located in the Naples urban area, by creating an open and multidisciplinary distributed computing infrastructure based on the state-of-the-art Grid paradigm. The Grid's connectivity is supported by our metropolitan multi-ring shaped optical network, empowered by the MPLS technology, that offers high performance communication facilities to the four main university research sites: Medicine, Engineering, CSI (Center for Information Services) and the Monte S. Angelo Campus (see fig. 1).

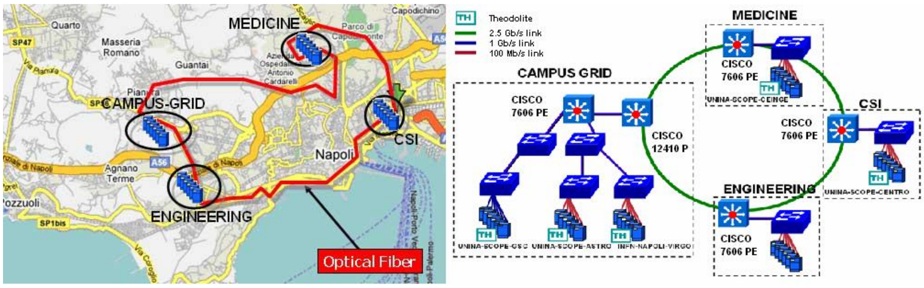


Fig. 1. The SCoPE network layout

The backbone is built on a fully meshed core realized between four high performance Cisco routers (a 12410 GSR and three 7606 OSRs), each acting as an access aggregation point (POP) in the metropolitan area. We realized two distinct independent rings between the core nodes using, for connecting each node to another both the primary and secondary branches on the ring. belonging to the primary ring backbone are made on POS STM-16 long range interfaces and the links belonging to the secondary (or backup) backbone ring and with the leaf access nodes are built on Gigabit Ethernet interfaces. The leaf nodes implementing access services for the MAG are implemented through high performance Cisco 6509 optical layer-3 switches, connected to one of the POPs through a differentiated two-way fiber path (access or drop ring). All the connections between the backbone routers and access switches are made with single mode optical fiber between Ethernet interfaces.

2.1 The Grid Infrastructure

The SCoPE Grid solution is based on the INFN-GRID [1] middleware, realizing a structured architecture built on a customization of the LCG storage and computing infrastructure [2] projected for the high-energy physics community using the Large Hadron Collider (LHC) and the gLite middleware [3] developed within the EGEE project context. The actual implementation provides the creation of a central Virtual Organization (VO) managed by a local Virtual Organization Membership Services (VOMS) that collect all the users of the project and that is enabled in each shared resource. In the current setup 12 local sites have been activated for a total of more than 3000 CPU cores and 500 TB of storage. All the local sites provide one or more traditional Computing Element (CE) batch execution systems, with several associated Worker Nodes offering the needed computing power. The SCoPE Grid implements storage access through multiple storage elements (SE) via the LCG Disk Pool Manager and Storage Resource Manager interface.

2.2 The Network Monitoring Platform

On-line Network monitoring is essential for the correct functionality of our MAG model. We need to understand the usage patterns of our network infrastructure and its performance both in real-time and historically to enable the network as a managed, effective and robust component of our Grid infrastructure. The GlueDomains prototype service [4], supporting the domain-based INFN Grid network monitoring activity, is used for this sake. The above service is supported by a dedicated central server node, placed in the main MSA Campus Grid site. This server manages the central database used to configure the topology measurement and the network monitoring activity of each "Theodolite" agent actually deployed to perform active network monitoring on five sites of the SCoPE Grid, as shown in the above fig. 1. Theodolite agents are installed within the Storage Elements and configured in a full mesh, making available Round-trip Time, Packet Loss Rate and One-Way Jitter performance measurements for each domain-to-domain path. We deployed a real time network resource control and monitoring agent on each site belonging to the MAG. Such agents also provide the functionality to modify the logical network topology by interacting with the MPLS label switching nodes to create on demand specific end to end traffic engineered paths with guaranteed bandwidth and/or QoS features.

3 Implementing the MAG Services

In the proposed MAG scenario, the network turns out to be a resource as important as computation and/or storage. As such, the Grid requires the same level of control towards subsets of well-defined connection services for the entire duration of a specific Grid task. A chief goal of this control is to turn the network into a virtualized grid resource that can be acted upon and controlled by other layers of software, realizing a service plane available to applications. Such service plane is typically concerned with path allocation, optimization, monitoring, and restoration across one or more domains. The service plane must be designed to be extensible from the ground up. It should allow adaptation of various control plane interfaces and abstract their network view, or element set, into its service portfolio. Viewing network entities as Grid Services allow us to consider the management of connections as a software transaction problem. Grid middleware must support this by relying on information models responsible for capturing structures and relationships of the involved entities. To virtualize networks into software components, we formally divide Grid resources into two types:

1. Network Elements (NEs), e.g. switches, routers, fibers/links;
2. Other resources, e.g. Storage Elements or Computing Elements.

Type 2 resources are already modeled with Grid Services. We extend the same metaphor to the Network Elements and provide the network as a software component. The end-to-end bandwidth reservation Grid Service is constructed using these components. Such service abstracts the domain as a virtual switch device. Accordingly applications operating within the SCoPE MAG must be able to trigger, with or without human interaction, three fundamental facilities:

– a bandwidth on demand layer-3 virtual tunnel service based on the setup of a couple of dynamic LSPs
– a point-to-point layer-2 service modeling a pseudo-wire between its endpoints
– a multipoint layer-2 on IP virtual LAN service modeling a virtual switch.

All these services must provide advance reservation capabilities: the ability to reserve in advance network resources for a future use. Advance or immediate network resource reservations have properties which make them somewhat different from the classical per-flow guarantees that have been demanded for multimedia services – the service may not be used immediately after its reservation and the flows are elastic. Realizing such per-flow bandwidth guarantees is not easy. Even when fine-grain reservation mechanisms like RSVP [5] would be available, providing them is an effort for the network, meaning that it will not be done for free. On the other hand, differentiating between a protected traffic aggregate and "all other traffic" is much easier, and can for instance be done by switching a pre-configured type of traffic (with classification via CoS, for instance) onto a specific MPLS virtual circuit/LSP.

3.1 Implementing Site-to-Site Bandwidth on Demand Services

MPLS control plane protocols allow on-demand requests for rate-guaranteed connectivity between multiple points in the network. These features make MPLS-based networks well suited to serve our MAG transport infrastructure in supporting the realization of bandwidth guaranteed on-demand dedicated virtual layer-3 circuits between any two sites in the common Grid environment. Guaranteed bandwidth LSPs with dynamic path selection and auto-rerouting capability can thus be established, when needed, for each pair of PE nodes connecting a couple of Grid sites, thus generating a mesh of LSPs, implementing individual virtual site-to-site connections with the required bandwidth assurances. Specific traffic flows can be explicitly conditioned to be routed on these LSPs through policy routing. Clearly, we need on each involved PE LSR a policy routing instance for each pair of nodes (and eventually service/port), clusters or entire sites we want to connect in a dedicated end-to-end fashion. Together with the requested bandwidth, Grid applications can also ask for a specific QoS treatment on a connection, defined by a proper Class of service (CoS) value. To mark MPLS traffic for specific QoS handling within our MPLS transport network the CoS information is carried in the experimental (EXP) field of the MPLS shim header. This field allows for eight different CoS markings. Label Switched Paths (LSPs) using this approach are called E-LSPs, since their QoS information is inferred from the EXP field. The signaling scheme for triggering LSP or lightpath set-up and reserving the needed bandwidth or wavelength resources along the path may be directly inherited by the TE-RSVP protocol [6]. To make a reservation request, the source node needs the path and the bandwidth that it is trying to reserve. Thus, the request is sent by the source along with path information. At every hop, the involved node determines if adequate bandwidth is available in the onward link. If the available bandwidth is inadequate, the node rejects the requests and sends a response back to the source. If the bandwidth is available, it is provisionally reserved, and the request packet is forwarded on to the next hop in the path. If the request packet successfully reaches the destination, the destination acknowledges it by sending a reservation packet back along the same path. As each node in the path sees the reservation packet, it confirms the provisional reservation of bandwidth. In order to accept/reject an incoming request, every network entity must have full knowledge of the available and reserved bandwidth and wavelengths on each outgoing link. This implies that every node needs to run a distributed control-plane protocol that keeps up-to-date information about the complete network topology and available resources.

3.2 Implementing Pseudo-wire Services

The point-to-point layer-2 or “pseudo-wire” service is functionally equivalent to provide emulated leased lines on the underlying MPLS backbone. The use of pseudo-wires enables integration and transport of diverse types of network traffic, as well as coexistence with other types of encapsulation. Pseudo-wires encapsulate Layer-2 protocol datagram units (PDUs), for transport across the MPLS domain. The establishment of emulated virtual circuits is specified in the IETF drafts usually known as “drafts Martini” [7] and [8]. These drafts define how MPLS can be used to support Layer-2 protocols such as Ethernet, Frame Relay or ATM. The draft [7] concentrates on encapsulation methods, while the other [8] specifies signaling to set up point-to-point layer-2 circuits over an MPLS network. The IETF draft “An architecture for L2VPNs” [9], proposes a layer-2 VPN solution, which is based on the tunnel-based emulation of layer-2 circuits. This draft can be seen as an evolution of a previous one - “MPLS-based Layer-2 VPNs” [10] - now obsolete, which originally described how to build layer-2 connections using MPLS in the provider core. The draft [9] that is the reference for our Cisco-based implementation is based on the “Martini” drafts for data frame encapsulation and for the signaling used to setup and maintain the emulated virtual circuits. The need to specify an auto-discovery mechanism is indicated but no solution is proposed for the time being.

3.3 VPLS or “Virtual Switch” Services

Traditionally, Layer-2 connection services have been provided only point-to-point. With new Layer-2 architectures like Virtual Private LAN Service (VPLS), the multipoint nature of the Ethernet LAN can be extended over an MPLS network, adding enhanced packet replication facilities and the ability to learn source-based MAC addresses for multipoint Layer-2 capabilities. This is an attractive option for MPLS-based networks because it uses a traditional MPLS Layer-2 service provisioning architecture to offer multipoint Ethernet connections that can join multiple sites within a MAN. Specifically, the PPVPN IETF group has reutilized the VPLS concept (following a term originally defined in RFC2764 [11]) as a layer-2 service that emulates a LAN across a WAN [12] [13]. The basic purpose of a VPLS is to offer layer-2 connectivity to multiple Grid sites in a manner that is totally transparent to the involved sites or computing nodes. With VPLS, each customer Grid site only requires a single connection to the network edge, and the involved PE nodes provide full multipoint connectivity. In our implementation, based on the [13] draft, the MPLS backbone is responsible for transporting customer Layer-2 frames and switching them across the MAG network between the Grid sites, and LDP is used as the signaling facility of choice. From the customer's point of view the service is equivalent to connecting the Grid devices via a switch, i.e., all in the same broadcast domain/LAN segment. Thus, using VPLS, we can create the new concept of a Layer-2 "virtual switch" dedicated to the Grid application over the MAG core.

3.4 The Grid Interface

When network connection facilities are considered as resources to be managed on the Grid it becomes necessary to specify exactly what is meant for these resources, how to encapsulate them into Grid services and how to manage these services. A Grid service is a self-contained, self described application that can be published, located and invoked over a network through properly designed interfaces. Accordingly, the network resource reservation services and enhanced communication facilities presented in the previous sections must be managed by a new centralized component introduced at the collective services layer in the SCoPE base middleware stratum, offering well-defined secure transport service interfaces to the Grid applications and support services. This will result in an overlay Grid-driven network management and control facility built on top of the MPLS control plane, whose configuration, security policies and functional behavior are assumed to be totally independent. A fundamental construct underlying many of the required attributes and functions of the above facility is service virtualization that underpins the ability to map through properly crafted interfaces common service semantic behavior onto native platform facilities. A natural choice for implementing these interfaces within the Grid middleware is the Web Service Resource Framework (WSRF) [14] that aims at implementing Grid services in the form of web services enhanced for Grid applications. Such interfaces are compliant with the GGF's OGSA specification [15] and, in addition, conform to widely used Web Services standards (WSDL, SOAP, and XML). A centralized Network Resource Manager (NRM) implemented within the MSA SCoPE Central Services farm, with the role of collective broker for network connectivity requirements keeps track of the resources and interfaces available on the whole MAG to cope with all the necessary network operations. For example, a dedicated bandwidth may be reserved between cooperating applications identified by their IP addresses and ports so that based on network condition, Grid middleware can request, through the NRM, a new bandwidth constrained tunnels between relevant MPLS network elements. The NRM also interacts with the GlueDomains Theodolites to supply network performance information to Grid applications and is integrated into the MAG information system (IS) by publishing its operating and service basic information via LDAP on the top BDII server of the SCoPE infrastructure.

Each network operation, resource or node can be described by a set of XML interface elements. Note that every interface can be characterized by a set of user-layer attributes (i.e. CoS, Bandwidth required, and traffic flow identifier) that in turn is implemented at the control plane layer by a couple of unidirectional traffic engineered LSP tunnels together with some flow-specific routing policies. The ability of the Service Interface to hide the complexity of the service provisioning permits to define simple XML-based messages capable of supporting high level services (fig. 2).

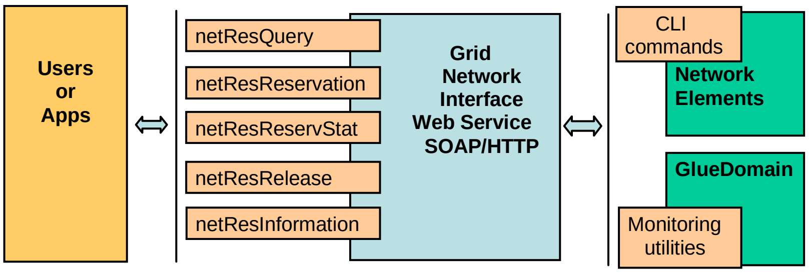


Fig. 2. NRM Web Services Architecture

In particular, the five messages exchanged are:

- *netResourceQuery(host_src, src_prot_ports, host_dst, dst_prot_ports, proto, service, band, class):* can be used in order to know if a certain network resource is available. This method will trigger an NRM-network device interaction needed to gain a specific picture of the resource usage between the involved PEs.
- *netResouceReservation (host_src, src_prot_ports, host_dest, dest_prot_ports, proto, service, band, class):* This is the main method realizing the end-to-end connections supporting the required bandwidth and Class of Service. It provides a unique, overloaded interface for all the reservation services. The reservation process is asynchronous, the user ask for a service by specifying all the necessary parameters, and retrieves a specific token. After that, the NRM synchronizes with the network control plane to create the required LSPs and activate the services.
- *netResourceReservationStatus (token):* allows the user to obtain information about the status of a Resource Reservation request, by identifying it through the token received at the reservation request time from the NRM.
- *netResourceRelease (token):* releases the resources identified by token by removing the associated LSPs.
- *netResourceInformation (theodolite):* This method is used by the user to know the network performance measurement performed by a specified Theodolite. The web service directly contacts the GlueDomains instance by forwarding the user request and retrieving on-line the associated output.

The NRM supports these functions within Grid middleware by relying on information models responsible for capturing structures and relationships of the involved entities. To cope with the heterogeneity of the network infrastructure resources when making advance reservations we proposed a new technology-independent network resource abstraction: the Traffic Engineered Tunnel, modeling the available PE-to-PE LSPs on the underlying networks that can be used for virtual connection transport. Thus the NRM is responsible for the dynamic creation, modification and deletion of the needed LSPs and is the only device interacting with the network elements. It uses a simple Perl-based programmatic interface that permits Grid applications to dynamic control and manage the underlying network resources according to the cooperation agreements stipulated between the Grid organization and the Federico II network operation center. Every basic service function is in turn mapped to a set of Cisco CLI commands for network resource setting, submitted to the network elements by using the `Net::Telnet::Cisco` standard Perl interface. Each invocation of specific function triggers the execution of a Perl script using a dedicated CLI session for the duration of its execution. Configuration requires the definition of the LSP identifier, the associated bandwidth and possibly some additional terms such as the CoS. The setup of intermediate routers is done automatically by LDP/RSVP signaling [16]. In order to identify the device to configure, the NRM uses an internal topology database from which network devices and routing information can be accessed. On-demand allocation of dedicated connections requires on-line discovery of network resources available to accommodate, new layer-3 or layer-2 associations on existing LSPs between the terminating PE or create, if needed new ones.

4 Performance Evaluation and Analysis

In this section we present some simple performance evaluation experiences done on our SCoPE MAG prototype to show how the applications distributed in the different sites of the metropolitan e-infrastructure can greatly benefit from the MAG facilities, and hence to demonstrate the effectiveness of the implemented architecture in providing QoS or bandwidth guarantees. We report the average results of the tests ran for performance analysis sake on five major sites of the SCoPE DataGrid testbed by performing replication of sample files. Specifically the involved sites are:

– UNINA-SCOPE-GSC (MSA Campus Grid new Data Center)
– UNINA-SCOPE-ASTRO (MSA Campus Grid)
– INFN-NAPOLI-VIRGO (MSA Campus Grid)
– UNINA-SCOPE-CENTRO (Centro Storico)
– UNINA-SCOPE-CEINGE (Medicine):

All the above sites are connected each other through the dark fibre metropolitan area ring described in section 2, where the involved link capacities vary from 2.5 Gbps POS STM-16 to the single or multiple Gigabit Ethernet connections. Each site participating to the testbed is provided with at least a Computing Elements and a Storage Element. To better emphasize the above benefits and improvements to application behavior, we performed our tests under real world extreme traffic load conditions, by working on the peak traffic hours between the UNINA-SCOPE-GSC site, actually the largest data centre in the scope Grid - providing computational resources to more than 150 internal users and external communities, and all the other sites. More precisely, the presented results have been obtained, by calculating the average values on several data replication sessions between one of the storage elements in UNINA-SCOPE-GSC and those located respectively in UNINA-SCOPE-CENTRO, INFN-NAPOLI-VIRGO and UNINA-SCOPE-CEINGE, performed with and without the support of MAG end-to-end network optimization services. All the involved storage elements are connected to their respective access switches through a 1 Gbps full-duplex Ethernet link. Here, for simplicity sake, we consider for each storage element two transfer sessions moving a total of 600 GB data:

– Replication of 250 files (average dimension 1.2 GB) with "BE" behavior.
– Replication of 250 files (average dimension 1.2 GB) with a bidirectional 1 Gbps bandwidth reservation and "EF" service class.

The data manager asks for a privileged end-to-end connection between the above storage elements with a bandwidth reservation of 1000Mbit/s and "expedited forwarding" service class, before starting the data replication, by issuing, for example, the `netResouceReservation (scopegsc.unina.it, 2119, se01.dsf.unina.it, 2119, tcp, BAND, 1000, EF, NIL)` request.

Such action triggers the creation, through the NRM interface, of a pair of dynamic traffic-engineered LSPs (one for each required direction) on the involved PE nodes. These paths are properly characterized with the required bandwidth, to be constantly ensured through the head-end auto-rerouting capability and a specific EXP-inferred Expedited forwarding (EF) class of service value, implying a differentiated low-latency queuing treatment on each hop of the MPLS cloud. Since the CoS value is encoded as part of MPLS header and since this value remains in the packets until the header is removed when the packets exit from the egress router, all LSRs along a

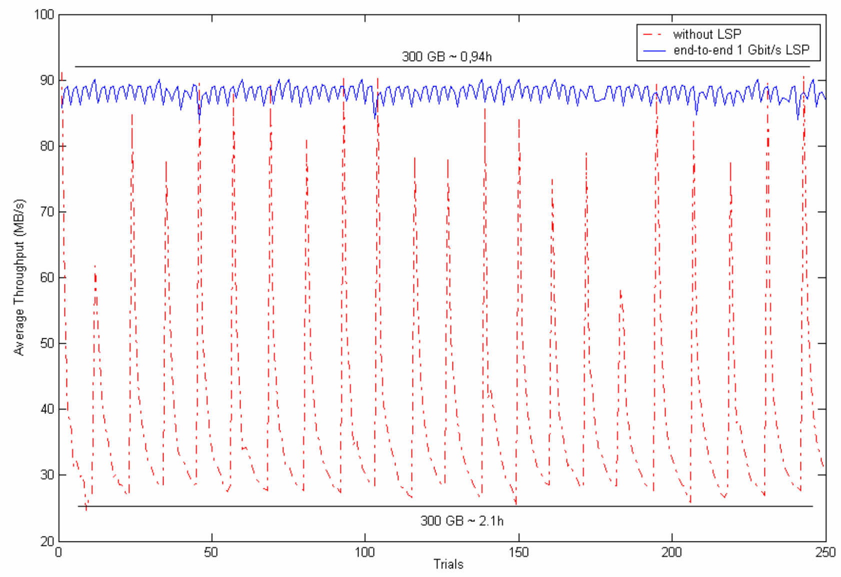


Fig. 3. Transfer time improvements with End-to-end bandwidth provisioning

specific LSP utilize the same CoS value set at the ingress router, and treat packets in the same manner accordingly. During the first replication test, performed without the aid of any MAG facility, our control plane routing protocol picks the best but most crowded route (due to the strong utilization of the involved links in peak hours) through the main branch of the metro ring so that we are able transfer 300GB of data in about 2.1 hours (average 31 seconds/file) with an average throughput of 38 MB/s and a maximum of 91 MB/s. During the second test, the above end-to-end LSP are automatically routed through the secondary (and almost unused) branch to guarantee the required bandwidth commitment and hence we can observe that the whole 300GB data transfer is completed in 0.9 hours (13 sec/file) with an average throughput of 88 MB/s and minimum of 84 MB/s. This simple test evidences how our MAG control plane can be effective in performing end-to-end bandwidth management and traffic engineering as it can be seen in fig. 3 below.

We also used the same replication tests to evaluate the end-to-end CoS support facilities provided by our Grid service plane demonstrating that it can effectively prioritize packets based on their service classes, and hence give the appropriate treatments to time critical traffic which can be extremely latency (end hence jitter) dependent. Accordingly we plotted (see fig. 4 below) the jitter measured during each transfer operation performed with both best effort and EF service class. We observed respectively an average jitter value of 20674 μs for the BE transfer and a much lower value of 2626 μs when Expedited Forwarding per-hop-behavior is required. This demonstrated the low-latency and transmission stability capabilities experienced on the Grid-controlled QoS-constrained end-to-end connection.

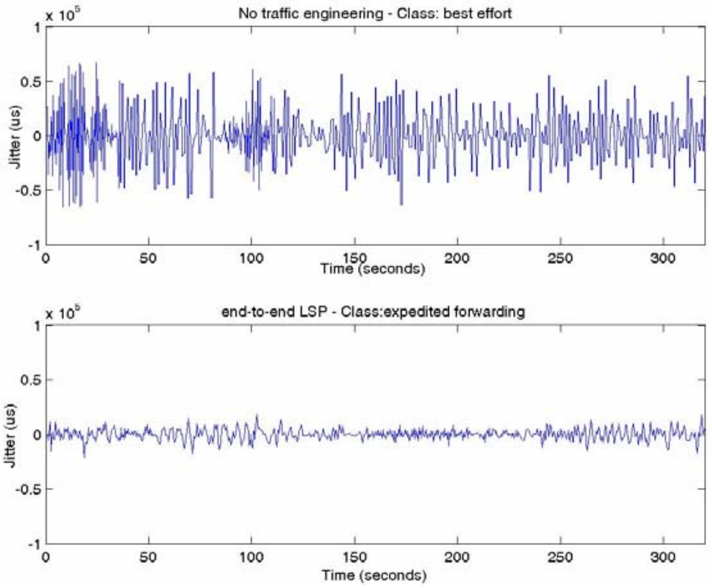


Fig. 4. Class of service support – BE vs. EF jitter values

5 Related Work and Discussion

To the best of our knowledge, very few experiences about Metropolitan Area Grids can be found in literature. The MAG concept has been introduced in [17] where a limited-scale grid computing environment called “mini-grid” that can be easily deployed on a MAN has been proposed as a very flexible and effective testbed for high performance Grid application. The problem has also been approached in a rather theoretical way in [18] by formalizing a centralized resource scheduling model for MAGs based on Stochastic Process Algebras. A more practical experience can be found in [19] presenting a real MAG deployed in the Shanghai area to realize a metropolitan-area information service infrastructure and establish an open standard for widespread upper-layer applications from people and the government. Our MAG implementation, working entirely in the electronic domain, has been designed to be flexible in network resource usage and to easily offer the hooks for inter domain-cooperation between neighboring MAG infrastructures. Clearly, since the presented solution is essentially empowered by the ownership on a large quantity of inexpensive parallel fiber strands on the metro-area, it aims in reducing equipment cost (that is the most critical factor at this scale) more than optimizing the usage of the available infrastructure by using expensive wavelength-division multiplexing devices. This implies that such an approach, that has been demonstrated to be successful for small-to-medium sized organizations spanning a limited geographical area, becomes less suitable for commercial carriers aiming at maximizing revenue from their owned fiber infrastructure.

6 Conclusions

We presented a new network-centric management architecture allowing the programmed pre-allocation of bandwidth and QoS-guaranteed virtual end-to-end connections on a Metropolitan Area Grid scale. In such architecture the traditional Grid resource reservation facilities can be extended with enhanced end-to-end connectivity services that are totally under the control of the Grid application. Accordingly we proposed and developed a new service oriented abstraction that, based on the existing Web Services architecture and built on the WSRF framework, introduces a new MAG-specific network control layer between the Grid customers and the network infrastructure decoupling the connection service provisioning from the underlying network infrastructure implementation. Such approach changes the perspective of scientific computing and data handling on a small geographical scale by focusing on the advantages of metropolitan/regional transport infrastructures in outreaching and supporting local scientific communities and Small & Medium Enterprises while decreasing the infrastructure cost of ownership. As a proof of concept, a simple and successful MAG prototype has been implemented in the context of the SCoPE high performance computing project for e-science applications.

References

1. INFN Grid, `http://grid.infn.it/`
2. LCG middleware, `http://lcg.web.cern.ch/LCG/`
3. gLite middleware, `http://glite.web.cern.ch/glite/`
4. Ciuffoletti, A., et al.: Architecture of monitoring elements for the network element modeling in a grid infrastructure. In: Proc. of Workskop on Computing in High Energy and Nuclear Physics (2003)
5. Braden, R., et al.: Resource Reservation Protocol (RSVP) - Version 1 Functional Specification, IETF RFC 2205 (1997)
6. Berger, L.: Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource Reservation Protocol-Traffic Engineering (RSVP-TE) Extensions, IETF RFC 3473 (2003)
7. Martini, L., et al.: Encapsulation Methods for Transport of Layer-2 Frames Over IP and MPLS Networks, IETF draft, draft-martini-l2circuit-encap-mpls-12.txt (2006)
8. Martini, L., et al.: Transport of Layer-2 Frames Over MPLS, IETF draft, draft-martini-l2circuit-trans-mpls-19.txt (2006)
9. Rosen, E., et al.: An architecture for L2VPNs, IETF draft, draft-ietfppvpn-l2vpn-00.txt (2001)
10. Kompella, K., et al.: MPLS-based Layer-2 VPNs, IETF draft, draft-kompellampls-l2vpn-02.txt (2001)
11. Gleeson, B., et al.: A Framework for IP Based Virtual Private Networks, IETF RFC 2764 (2000)
12. Andersson, L., et al.: PPVPN L2 Framework, IETF draft, draft-andersson-ppvpn-l2-framework-01.txt (2002)
13. Agustyn, W., et al.: Requirements for Virtual Private LAN Services (VPLS), IETF draft, draft-augustyn-vpls-requirements-02.txt (2002)
14. Czajkowski, K., et al.: From Open Grid Services Infrastructure to WS-Resource Framework: Refactoring and Evolution (2004), `http://www.globus.org`
15. Foster, I., et al.: The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, Globus Project (2002), `http://www.globus.org/research/papers/ogsa.pdf`
16. Jamoussi, B., et al.: Constraint-Based LSP Setup Using LDP, IETF RFC 3212 (2002)
17. Brooke, J., Foster, M., Pickes, S., Taylor, K., Hewitt, T.: Mini-Grid: Effective Test-Beds for Grid Application. In: Buyya, R., Baker, M. (eds.) GRID 2000. LNCS, vol. 1971, pp. 158–169. Springer, Heidelberg (2000)
18. Wang, G., Lin, C., Liu, X.: Performance Modeling and Analysis for Centralized Resource Scheduling in Metropolitan-Area Grids. In: Shen, H.T., Li, J., Li, M., Ni, J., Wang, W. (eds.) APWeb Workshops 2006. LNCS, vol. 3842, pp. 583–589. Springer, Heidelberg (2006)
19. Li, M., et al.: ShanghaiGrid: an Information Service Grid. Concurrency and Computation: Practice and Experience 18(1), 111–135 (2006)

Extending Routers Utilisability and Life Cycle through Automated Configuration Management

Francisco Rodríguez, Mickaël Hoerdt, and Laurent Mathy

Computing Department, InfoLab21
Lancaster University, Lancaster, UK
{f.rodriguez,m.hoerdt,laurent}@comp.lancs.ac.uk
http://www.comp.lancs.ac.uk

Abstract. We present the design of a distributed router platform aimed at consolidating multiple hardware routers. The goal of the approach is twofold: firstly decouple the logical routing and forwarding functionality from the limitations of the hardware that runs it, through automated configuration management only; and secondly, give component routers a longer lease of life, as constituting parts of a larger router system. We focus on the logical intra-domain routing function provided by routers, and show the need for a centralized intra-domain route server.

Keywords: Aggregated IP Router, Router Management Automation, Routing Management, Route Server.

1 Introduction

The ever increasing demands for higher forwarding throughput [1,2] force ISPs to improve their network capacity. In an aggregation network architecture, as the one used by ISPs, network capacity is mainly determined by the routers forwarding capacity. In such a scenario, replacing all routers for improving network capacity becomes a very expensive and unthinkable solution. Instead, ISPs have developed a different and rather characteristic replacement pattern. They introduce new, high performance routers in the core of their network, pushing existing routers towards their network edge, with routers already at the edge being often decomissioned before the end of their hardware life cycle.

Furthermore, routers are usually seen as single role entities only interacting with each other for creating a network of monolithic packet routing devices. The main reason for this is that most routers manufacturers usually sell their routers without the capabilities to be reprogrammed, effectively preventing their owners to perform any structural modification if there was the need to.

In this paper, we present a technique that has the potential to change the way routers are replaced and managed by ISPs nowadays. We propose a method aimed at extending the duty cycle and usefulness of edge routers, by consolidating these hardware routers in order to build a bigger (logical) router with higher port density and increased forwarding capacity. We show the feasibility of using the routers themselves as the basic building block for such a logical router only by acting at their configuration level. The benefits of such a system are not only limited to reduced costs. More specifically, these logical routers will be more resilient by being composed of multiple devices. Additionally, port density and forwarding capacity can easily be increased by simply adding more hardware routers, increasing its scalability. We determine that it is possible to build what we called an Aggregated IP Router (AGIR) out of standard router devices without relying on their internal structure. This makes our proposal applicable to most routers. This solution offers the means to compose and recompose logical routers out of heterogenous hardware boxes while being independent from any particular implementation.

It is important to highlight that an AGIR is not a new router architecture or a technique to split in a predefined fashion router's tasks through a set of hardware components, but a technique to build bigger routers out of a set of heterogenous routers without modifying their internal structure. In a sense, the routers themselves are components of an AGIR. Therefore, all the component routers have to be configured cleverly and automatically in order to complete all the tasks required to provide a single unified router function.

This approach of building logical routers by aggregating several hardware routers can be further extended to provide several independent logical routers onto a same aggregate (cluster) of hardware routers. The purpose here is that through appropriate control, the logical router entities can be decoupled from the hardware routers: a logical router can be mapped onto several component routers, while parts of several logical routers can be hosted onto a single hardware router.

For instance, consider the case of Internet Exchange Points (IXPs). IXPs can be more than a place and a switch where ISPs interconnect their equipment. It can be transformed into a whole service which not only provides its current mission, but also a whole router leasing service. This could bring many benefits to the ISPs considering the complexity and high costs that they have to take on for implementing and managing a router installed in an IXP. This idea is not only associated with IXPs, but also with ISPs' points of presence (POPs), as is the case in which the POP could be able to share its physical hardware resources providing routers as a service to multiple customers or even multiple ISPs. By doing so, we provide the technical platform for enabling a new business model for ISPs.

Note, however, that this paper focuses on the mechanisms for the provision of a single logical router onto a hardware router aggregate.

The remainder of the paper is organized as follows. Section 2 describes the concept of Aggregated IP Routers. Section 2.1 presents the challenges that need to be solved for building them. Section 3 describes the proposed design for providing our Aggregated IP Routers with intra-domain routing. Section 4 describes our implementation and evaluation in which we use as a case of study the Open Short Path First (OSPF) protocol. Section 5 provides an overview of the related work. Finally, in section 6, we highlight our conclusions and give directions for future work.

2 Aggregated IP Routers

In essence, AGIRs resemble distributed routers and, as them, offer the same advantages derived from a component based architecture [4,5,6], but with a *significant* variation. Instead of designing hardware boxes with specific software, tasks, and new protocols, we take advantage of the already existing technology, uniquely relying on the basic configuration interface provided by all routers. By doing this, we investigate the level of flexibility that current routers can offer when composed with each other at the configuration level in order to provide the same functionalities of a router.

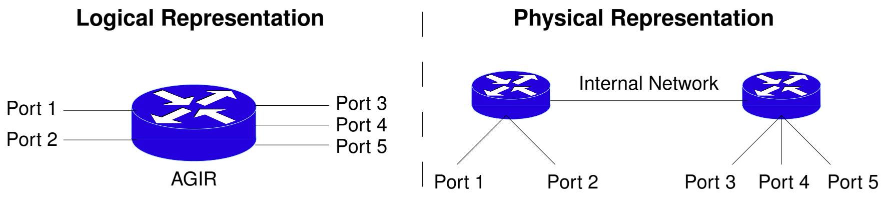


Fig. 1. Aggregated IP Router Abstraction

An AGIR is a "logical" IP router that has the same functionalities of a router. An AGIR is composed by a set of routers configured specifically to perform as a single router. These routers are interconnected by a switch which interacts between them at the IP level. Such an AGIR design can seems to appear as a network of routers, but still be regarded one single router from an external point of view. In principle, any routers regardless its architecture, i.e. hardware routers, software routers and virtual routers, could be included in the composition.

2.1 Challenges

Building a AGIR out of closed router boxes[1] poses many challenges, since routers are designed merely to interact between each other at the network layer (IP) in order to build networks, **not** to compose other routers.

Internal networking: Since AGIRs are integrated by N routers residing in N hardware boxes, also referred as AGIR members, they require to be interconnected for communicating between each other. This basic requirement can be fulfilled by using a high speed switch. This switch will provide the means to create an internal network, namely Forwarding Private Channel (FPC). This FPC represents the equivalent to a router's backplane. It is important to highlight that having such internal network introduces new failures scenarios that currently routers do not consider, e.g. the partial failure of the internal network due to a switch's port failure or a member's port failure.

[1] Any router regardless its type and make capable of performing routing and forwarding functions.

Port awareness: All AGIR's members lack of a complete AGIR's ports awareness as well as a lack of forwarding paths port awareness to all other members. AGIR's members need to have a complete knowledge of all the ports which are contained in the AGIR as well as the forwarding path to reach them.

Heterogenous configurations: Another important challenge to solve is to define a way to represent AGIR's configuration. AGIRs are integrated by multiple members, so it is crucial to have a way to represent and identify them. Furthermore, it is important to create adequate reconfiguration policies for AGIR's members in order to reflect AGIR configuration changes. Moreover, AGIR's members might require configuration changes during its operation time; changes that required to be reflected in all AGIR's members.

Unified Routing: One of the principal challenges for building AGIRs is to provide them with one of router's main capabilities: routing. This capability represents routers' capacity for computing routes with other routing devices outside AGIR. From the outside, no difference should be seen in comparison with router when exchanging routing packets with AGIRs. The way in which routing protocols have to be configured in AGIR does not appear to be a straight forward process. Furthermore, we have to consider the different types of routing, static and dynamic, in order to understand the difficulties that each type presents when implementing them in AGIRs.

Static routing or static routes are entries in routers configuration that contain well-known establish route paths. These route paths are called static because they do not change except by manual configuration. Due to its persistent state and implementation simplicity, providing AGIRs with static routing capabilities does not present a problem. They just need to consider AGIRs composition since the routes specifications might differ from AGIR member to AGIR member.

Dynamic routing change their routing decisions to reflect changes in the network topology. Dynamic routing protocols differ in where they get their information, when they change the routes and the metric used for optimization. These adaptive algorithms are also called routing protocols. Based on where the routing protocols are used, they can be classified in two different categories: inter-domain and intra-domain.

Inter-domain routing protocols are designed for use between Autonomous Systems (ASes). They are configured in routers when there is a need to exchange routing information between ASes. This means that they calculate the forwarding path in terms of which AS(es) passes through, and do not calculate the route through individual routers within an AS. Running an inter-domain routing protocol, like BGP, in a AGIR does not present a problem. Considering one of BGPs main principles, namely to give neighbours ASes a single view of the advertised AS through a distributed set of border routers, AGIRs can easily support inter-domain routing. For instance, AGIR members can be configured to run IBGP between them. In this way, the ASes connected to the AGIR via an AGIR member will have exactly the same view of the AS advertised by AGIR giving the impression of a single router by not exposing the internal structure of

AGIR. In this case, the nature of inter-domain routing protocols works in favour of supporting them in AGIRs in an easily and transparently manner.

Intra-domain routing protocols are used within an AS or routing domain. They calculate their paths based on router hops within an AS. These routing protocols keep an exact track of how to get from one destination to another inside a network or set of networks that are under the same AS management. Running an intra-domain routing protocol in AGIRs represents many challenges. The main reason behind these challenges is the fact that essentially intra-domain routing protocols gather a complete picture of the network topology where they are ran. In such a situation, AGIRs internal structure would be exposed to the external neighbour routers.

Scalability: Aggregating several routers together requires a design which takes into account the scalability issues related to the possible large number of nodes composing the AGIR, the link layer technology cost connecting the nodes together as well as the complexity of the backplane topology. Despite this challenge, a carefully designed interconnection can potentially load balance flows on the routers composing the AGIR to exploit the aggregated resources more efficiently than if they were used on its own.

Forwarding delay: Aggregating several routers together to forward packets introduces additional delay to the processed packets when they are crossing multiple routers composing the AGIR. If a packet only needs to be forwarded by one router, no additional delay will be added to the forwarding operation except the usual one introduced by a stand-alone router. Careful attention must be taken when designing AGIR's backplane topology to avoid extra forwarding delay.

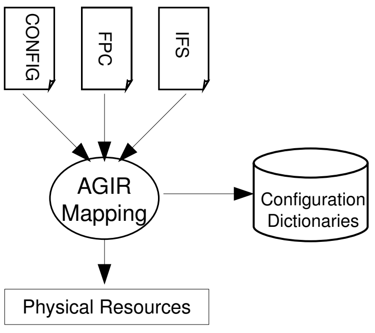
3 Designing an Aggregated IP Router

The proposed AGIR's design is driven by the challenges mentioned in section 2.1. The construction of an AGIR includes: 1) mapping the AGIR to its physical resources[2], 2) computing and maintaining AGIR's routes and AGIR members' forwarding table, and 3) monitoring AGIR's state. Following, we discuss separately each one of these steps in the remainder of the section.

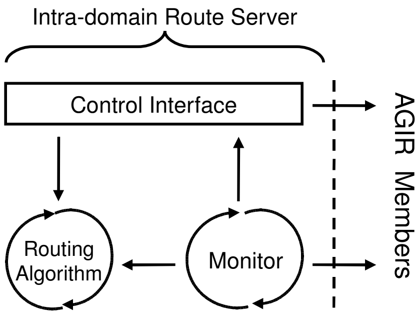
3.1 Mapping the AGIR's Members Resources

For tackling AGIR's representation problem, we organized the AGIR's information in three different repositories. These repositories, shown in Fig.2(a), store all the necessary information for assembling an AGIR. User's AGIR's configuration is represented by the **Config** repository. This repository contains an AGIR's simplified view and stores essential information for mapping correctly to the hardware routers the configuration settings of the user. These settings covers all the information related to AGIR's ports, such as IP information, as

[2] AGIR's members.



(a) AGIR's Representation



(b) Route Server Abstraction

Fig. 2. AGIR's Design

well as the routing protocols settings and other applications, e.g. NAT, packet filtering, etc.

AGIR's internal network is represented by the **FPC** repository. This repository contains all the necessary data for setting the communication channels betwen AGIR members. The data stored in this respository describes how an AGIR is composed by listing all its members' ports with its corresponding address within AGIR's internal network.

For solving the port awareness problem, we define the **IFS** repository. This repository stores the location and description of all AGIR's ports. Using the information in IFS, we propose the utilization of static routes for specifying AGIR's ports in every AGIR's member. In this way, AGIR's members will be aware of all the ports that composed the AGIR to which they belong. We have chosen to use static routes in our AGIR design due to the flexibility that they offer for mapping routes at the configuration level.

As AGIR members are composed by multiple heteregenous hardware routers, we need a way to translate their various configuration settings into a unified AGIR configuration. This is represented in our AGIR design by a collection of configuration dictionaries. These are used to set and modify the different configuration settings needed in AGIR members for representing AGIR's configuration settings.

3.2 Computing and Maintaining Aggregated IP Routers Routes

In order to overcome the limitation that intra-domain routing protocols present for implementing them in AGIRs, we propose to utilise an intra-domain route server (RS) for AGIR. By doing the latter, we would be able to hide the internal structure of our AGIR providing the same external view of any other type of router. Additionally, we avoid to run in each AGIR member a different intra-domain instance.

Since the RS acts as AGIR's intra-domain routing engine, the AGIR's RS will need to perform the same tasks that routers perform when running an intra-domain routing protocol, i.e. receive and transmit routing packets, calculate AGIR's routes, calculate and install AGIR members correspondant forwarding tables, as well as to guarantee consistent intra-domain routing announcing and AGIR members forwarding tables consistent with AGIR's real state. We have splitted and grouped the previous tasks, as shown in Fig.2(b), through three different processes: intra-domain routing algorithm, monitor, and configuration policies.

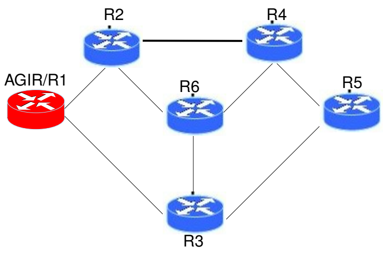
The **routing algorithm** process represents the core of our RS design. This process is in charge of computing AGIR's intra-domain routes by using the corresponding algorithm. The algorithm to run by the RS would depend on the configured routing protocol, e.g. Djiskstra algorithm for OSPF. Furthermore, it is in charge of receiving the incoming routing packets through AGIR's members. Likewise, it is responsible for generating and sending the corresponding routing packets.

One of the main issues, as well as advantages, of using a distributed architecture in routers is that they are integrated by multiple devices. This helps to improve routers' resiliency at the time that decreases its failure probability [3]. In AGIR's case, not only the latter is true but also introduces a new problem to its RS. Since all AGIR components are disseminated through different hardware boxes, AGIR's RS has to monitor them for detecting any failure and consequently compute correctly AGIR's routes and every AGIR member forwarding table as well as announcing AGIR's real state to its neighbours. RS's **monitor** process assignments are the prompt detection of routing changes as well as the prompt detection and location of AGIR's failures. Under these circumstances, it will turn on the respective alarm.

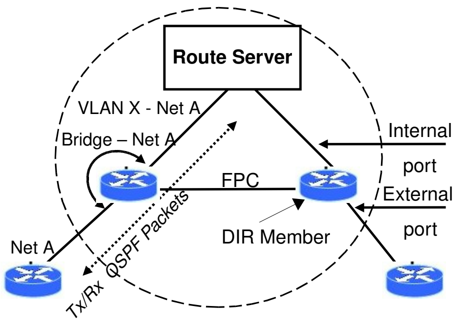
As its name suggests, the **control interface** is RS's interface for interacting with its intra-domain routing algorithm process as well as with AGIR members. This interface contains the policies that need to be applied for modifying AGIR members forwarding tables and/or the intra-domain routing algorithm process behaviour based on the type of alarm received from the monitor process.

4 Implementation and Evaluation

In order to show the feasibility of the proposed AGIRs, we have implemented a AGIR mapper and a AGIR intra-domain RS. The AGIR mapper has been developed using Python programming language. Its main purpose is to map AGIR's configuration settings into AGIR's members based on the three different repositories, described in section 3. The commands used by the AGIR mapper for configuring the routers are included in the implemented configuration dictionaries. At the moment, we have implemented only the configuration dictionaries for IOS based Cisco Systems routers and PCs running Linux Debian OS.



(a) Network Topology



(b) Aggregated IP Router

Fig. 3. Testbed

For AGIR's RS implementation we have developed its monitor and control interface using Python programming language; as for the routing algorithm we have only implemented Djiskstra algorithm for supporting the OSPF routing protocol. For implementing this routing algorithm, we have used an adapted version of XORP[15] software.

The first problem that we have faced while running our experiments was to cross the OSPF packets through the AGIR members. The reason behind this problem is OSPF packet's TTL value of 1[13]. This value prevents routers to further forward such packets. For overcoming this problem, we have configured AGIR members as bridge-routers[14]. In this way, OSPF packets can cross through AGIR members and all other packets are routed based on AGIR member's forwarding table, as illustrated in Fig.3(b).

Our evaluation tests have been carried out in a virtual environment. Therefore, we have emulated the network topology, shown in Fig.3(a), using Kernel-base Virtual Machine (KVM) software running in two Dell PowerEdge 1950 servers with 2 64-bit Quad-Core Intel Xeon processors. Our evaluation methodology has been to run tests in our virtual environment using uniquely routers, XORP software running in Linux Debian 2.6.22-3 virtual machines (vms), generating a reference point to compare against the introduction of one AGIR in the same network topology. For composing AGIRs, we have used Linux Debian 2.6.22-3 vms for all its components. For this study, we have used an AGIR composed by two members and one RS, as the one illustrated in Fig.3(b). We have defined in our emulated network, shown in Fig.3(a), R1 as the device to observe through our experiments, and consequently its neigbhours R2 and R3. For AGIR's studies, we have replaced R1 for a AGIR.

We have studied two different aspects of AGIRs using our implementation. Firstly, we have measured the impact of using AGIRs in OSPF networks. Secondly, we have evaluated RS's performance during all the possible failures that AGIRs could present, ensuring that it provides AGIR with consistent intra-domain routing.

4.1 Impact in OSPF Networks

In order to measure the impact of using AGIRs in an OSPF network, we have studied its behaviour in the event of a failure within the network. In this context, we have simulated in R3 a failure in its port connected to R6 while capturing the generated link state advertisement (LSA) packets due to the failure. We ran the experiment several times and plotted the average number of observed LSA packets during the failure and its time distribution.

The results, plotted in Fig.4(a), using and without using an AGIR where the same. When R3's port failure occurs, R3 detects it and floods its remaining reachable neighbours with an LSA update. This event triggers an SPT recomputation in all routers and an announcement later, after 5 seconds on average, of its new state. Using a RS as AGIR's routing engine do not have any impact in OSPF networks when having to recompute the Shortest Path Tree (SPT) after a network failure. The use of AGIR members as bridge-routers does not introduce significant packet distribution variation in the reception and in the transmission of the LSA packets to and from the RS towards its neighbours.

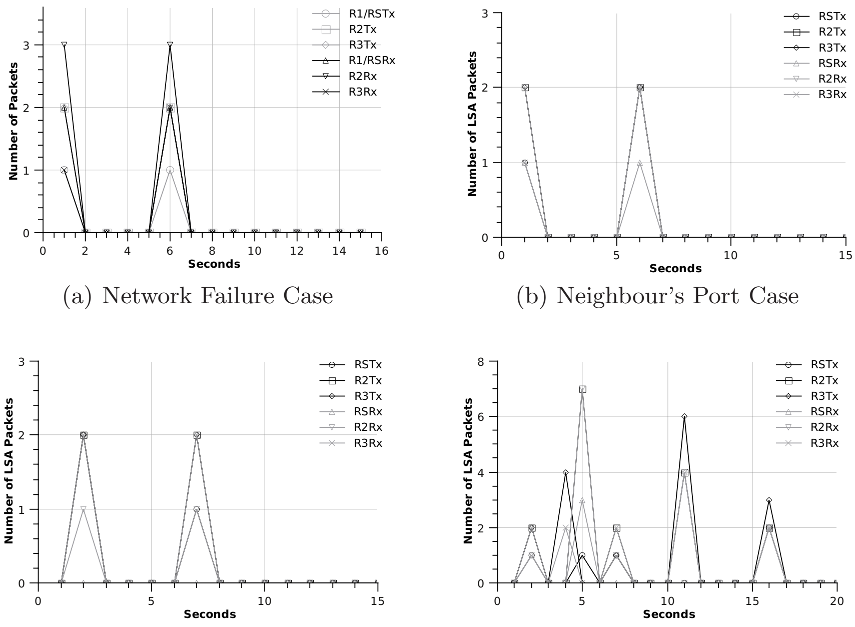
4.2 Providing Consistent Intra-domain Routing

As in section 4.1, we have carried out a comparative study between the different types of failures presented in stand-alone routers vs AGIR's failures. Using the same methodology of the previous study, we have simulated the different types of failures for both types of routers.

Neighbour's port failure are not failures presented within the AGIR router, but still affect its behaviour. The results from our experiments when simulating this type of failure were the same in the AGIR and the non-AGIR cases. The LSA packet distribution graph, shown in Fig.4, describes OSPF behaviour. Based on results, we can say that AGIRs do not have a negative impact at the routing level. Nevertheless, the utilisation of a RS introduces a natural delay when a SPT recomputation takes place at AGIR's forwarding level. This delay is mainly caused by RS's additional tasks: computing and installing the corresponding forwarding tables for every AGIR member.

To determine the mentioned delay, we have measured the reconvergence time for both cases in our emulated environment by running the following experiment. On one router (RO), we have generated every 100 ms an ICMP request to a learned OSPF route. Then, we have simulated a port failure in another router (RF) through which the ICMP packets go through to reach its final destination. In this way we defined router's convergence time as: $\Delta T = T2 - T1$; where T1 is the time at which RO stopped receiving ICMP replies and T2 the time at which RO started receiving ICMP replies after the failure.

We ran several times the previous experiment and obtained that the average OSPF convergence time given by a stand-alone router, in our emulated environment, is 3.1 seconds. For AGIR's case the convergence time was significantly above previous case, about 1.2 seconds more. Analyzing our RS implementation, we realized that waiting for the OSPF process to converge before starting



(a) Network Failure Case (b) Neighbour's Port Case

(c) AGIR Member's Port Case (d) AGIR's FPC Case

Fig. 4. LSA Packet Distribution

to compute AGIR members forwarding table was too costly. Consequently, we decided to integrate into our RS prototype a routine which precomputes AGIR members' forwarding tables in the event of a failure within the AGIR. By doing so, RS OSPF's convergence time when having a failure is composed as follows: $RSCT = DT + IT$; where DT is the time for RS to detect the existance of a failure and IT the time for the RS to install in AGIR members the corresponding forwarding updates. By using this pre-computation technique, we have reduced the average DT for a neighbours port failures to 1.3 seconds while the average IT was 1.2 seconds. So on average, our RS takes 2.5 seconds to update AGIR members' forwarding table.

AGIR's port failures can be classified into two different types of failures: those happening on the **RS's ports** and those happening on the **AGIR member's ports**. Although both failures are the equivalent of having a router's port failure they do not have the same effects in RS behaviour; therefore we have analyzed them separately.

The LSA packets distribution for router's port failure and RS's port failure are fairly the same. The results from both cases present the same behaviour as the one for router's neighbour case, illustrated in Fig.4. The reason for this is that the failure, in both cases, occurred in the physical device running OSPF. During a RS's port failure its very important the DT component, since this will trigger the action for updating AGIR members forwarding table. We have measured such time and obtained that on average the DT for this type of failure is 1.4 seconds. So, in the event of this type of failure our RS prototype can offer a convergence time of 2.6 seconds.

From the results, shown in Fig.4(c), we realized that in the event of a **AGIR member's port failure**, the LSA packet distribution suffers a shift to the right in comparison to all previous cases. This shift to the right represents RS's DT since the failure did not occur in the device where the routing computations takes place. According to our measurements, DT for this failure type is on average about 1.1 seconds.

A **failure in AGIR's internal network** is the equivalent to a failure in router's backplane. In stand-alone routers this failure will completely halt its packet forwarding capabilities. In contrast, such failure in AGIRs would only prevent AGIR members to forward packets between them but allowing AGIRs RS to keep exchanging routing information with its OSPF neighbours; consequently, generating routing and forwarding inconsistency in AGIR. For avoiding the latter, we have designed and implemented in our RS a solution for dealing with this very particular case. This solution consists in splitting the OSPF route calculation in the same number of OSPF processes in which AGIR has been splitted by the failure.

We have simulated several times a failure in AGIR's internal network and noted from results, shown in Fig.4(d), that there is a delay in announcing the failure caused by DT. As in all previous failure cases, we have used the approach of precomputing AGIR members' forwarding table for avoiding to wait for OSPF to reconverge, which on average can take up to 11 seconds. DT for this type of failure, based on our measures, is on average 1.2 seconds.

5 Related Work

The use of single logical router entities for representing a set of confederated routers to provide a service is not a new concept in networking; it has being widely used with different purposes. Digital was one of the pioneers using and developing this concept in [7]. Its original idea of providing a fast failover mechanism for IP routers or line failures rapidly evolved and converted into what we know nowadays as VRRP. VRRP used logical routers as a way of monitoring a router in case of a failure providing a backup method for the network [8]. The reasons behind VRRP differs from ours, namely to consolidate multiple heterogenous hardware boxes conforming a single logical router.

Another work related to the use of logical router entities is what we know as virtual routers [9,10]. The main aim of virtual routers is to maximize the hardware resources utilization of high-end routers. As mentioned in [9], virtual routers not only allow routers to maximize hardware resources, but also provide a flexible platform to offer a broad diversity of services e.g. MPLS VPNs[10]. These virtual routers differ from our main purpose inversely. While we are trying to consolidate a set of hardware routers to conform one router, virtual routers try to split the hardware resources of a hardware router in N virtual routers.

Furthermore, virtual routers running in commercial platforms are usually limited to reside on a single hardware box. Moreover, AGIRs provides the same functionality obtained by deploying a router, situation that differs in MPLS VPNs case since some of the functionalities provided by the router, e.g. routing, are entirely entitled to the ISPs.

Another benefit that logical routers have proved to provide is the one exposed in [11,12]. Both works aim to propose an architecture for decreasing network downtime during network maintenance by using cooperatively ISP's infrastructure. They removed the static binding that customers have with ISP's hardware, looking at customers more as a service and less as another hardware box to interconnect. They achieved the latter by providing to ISP's customers logical routers and not hardware routers as done nowadays. Their purpose and the way they conceived logical routers, in a single hardware box, differs from our main aim and approach for providing AGIRs.

ForCES framework [4] is the closest work in comparison to our AGIR proposal. In a sense, both works try to provide the means for composing routers from multiple hardware boxes, but differing very much in their proposed method. While ForCES defines an architecture for creating a protocol that allows the utilization of heterogenous NE's components, i.e. from different vendors, for conforming a router, AGIR solution considers a different method, namely to manipulate the routers in a clever and automatic fashion. ForCES objectives follow the line to enable its idea by incorporating a new IPC protocol into the hardware boxes, while our work suggests a new and more effective router's configuration management method. It is also important to highlight, that AGIR considers all types of routers regardless their architecture, so eventually routers built under ForCES framework could be utilised for conforming AGIRs by just adding its corresponding configuration dictionary.

6 Conclusions and Future Work

In this paper, we have described the design of an aggregated IP router platform, aimed at consolidating a set of heterogenous hardware routers connected by a high-speed local interconnect. We have mainly focused on the issues associated to the support of a single logical intra-domain OSPF entity on such platform.

We have analyzed the different failures scenarios in which such platform might appeared and seen, that in most of the cases, it provides an acceptable level of performance. We observed that such platform posses a high level of availability and flexibility, which are presented when using a set of routers in contrast with individual routers performance. Our results demonstrate that AGIR's approach is promising and encouraging. Taking the latter into consideration, we believe that the flexibility brought by our aggregated IP routers platform, along with the extension of routers' life-cycle, open up the prospect of various new applications for the platform.

References

1. Odlyzko, A.M.: Internet traffic growth: Sources and implications. In: Optical Transmission Systems and Equipment for WDM Networking II, August 2003, vol. 5247, p. 115. SPIE, San Jose (2003)
2. Shapiro, R.J.: The Internet's capacity to handle fast-rising demand for bandwidth, White Paper, US Internet Industry Association (September 2007)
3. Ramjee, R., Ansari, F., Havemann, M., Lakshman, T.V., Nandagopal, T., Sabnani, K.K., Woo, T.Y.C.: Separating control software from routers. In: First International Conference on Communication Systems Software and Middleware, New Delhi, India (January 2006)
4. Yang, L., Dantu, R., Anderson, T., Gopal, R.: Forwarding and Control Element Separation (ForCES) Framework, RFC 3746 (April 2004)
5. Hagsand1, O., Hidell, M., Sjdin, P.: Design and Implementation of a Distributed Router. In: IEEE International Symposium on Signal Processing and Information Technology, Athens, Greece (December 2005)
6. Hidell, M., Sjdin, P., Hagsand Router, O.: Architectures Tutorial at Networking 2004, Athens, Greece (May 2004)
7. Higginson, P.L., Shand, M.C.: Development of router clusters to provide fast failover in IP networks. Digital Tech. J. 3(9) (January 1997)
8. Hinden, R.: Virtual Router Redundancy Protocol (VRRP), RFC 3768 (April 2004)
9. Juniper Networks. Intelligent Logical Router Service, White Paper (October 2004)
10. Cisco Systems. Introduction to Cisco MPLS VPN Technology, `http://www.cisco.com`
11. Agrawal, M., Bailey, S.R., Greenberg, A., Pastor, J., Sebos, P., Seshan, S., van der Merwe, K., Yates, J.: RouterFarm: towards a dynamic, manageable network edge. In: INM 2006: Proceedings of the 2006 SIGCOMM workshop on Internet network management, Pisa, Italy (September 2006)
12. Wang, Y., Keller, E., Biskeborn, B., van der Merwe, J., Rexford, J.: Virtual routers on the move: live router migration as a network-management primitive. SIGCOMM Comput. Commun. Rev. 38(4) (October 2008)
13. Moy, J.: OSPF: Anatomy of an Internet Routing Protocol, January 1998. Addison-Wesley, Reading (1998)
14. Cisco Systems. Understanding and Configuring VLAN Routing and Bridging on a Router Using the IRB Feature, `http://www.cisco.com`
15. Handley, M., Kohler, E., Ghosh, A., Hodson, O., Radoslavov, P.: Designing Extensible IP Router Software. In: The proceedings of the 2nd USENIX Symposium on Networked Systems Design and Implementation, Boston, Massachusetts, USA (May 2005)

Autonomic Virtual Routers for the Future Internet

Wajdi Louati, Ines Houidi, and Djamal Zeghlache

TELECOM SudParis, 9 rue Charles Fourier, 91011, Evry, France
{wajdi.louati,ines.houidi,djamal.zeghlache}@it-sudparis.eu

Abstract. This paper provides the design and the architecture of autonomic virtual routers to support automated provisioning and management of virtual networks. The Autonomic Virtual Routers (called AVR) combines the IETF ForCES (Forwarding and Control Element Separation) principle with the autonomic computing concept to build standardized, programmable, open and extensible virtual routers. The AVR can automatically manage and organize control and forwarding components and their interactions to deliver on demand virtual router services. The objective is to automatically create virtual routers in substrate nodes to support on demand virtual networks, without any human intervention.

1 Introduction

Network Virtualization has been proposed as a promising way for diversifying the Future Internet architecture. Network virtualization enables the coexistence of separate logical networking environments, called *Virtual Networks*, on a shared *substrate network* enabling the deployment of both evolutionary and revolutionary networking designs for the Future Internet [1][2][3][4][5][6]. Using the virtualization concept, a substrate node is split into *virtual nodes* (e.g. virtual router/switch) and a substrate link is partitioned into *virtual links* (e.g. Layer 2 or 3 tunnels). A virtual network (VN) is hence defined as a group of virtual nodes interconnected via virtual links.

This paper focuses on the case where a VN is supported by a set of *Virtual Routers*(VRs). A virtual router [7][8][9][10][11][12][13] is an emulation of a physical router including mechanisms and tools for management, configuration, monitoring and maintenance. Many VR instances may run on a single substrate node by using hardware/software-based virtualization approaches. The VR instances have independent IP routing and forwarding tables and are isolated from each other.

To provision and manage VNs for users, the providers (e.g. VN provider [2], Infrastructure and Connectivity Providers [5]) are responsible for allocating, configuring and managing the VRs forming the VNs. In existing virtual routers, the provisioning (including creation, reconfiguration and removal) and management of VRs require the manual intervention of providers (e.g. manual configuration of routing tables and protocols, network interfaces, etc). The manual involvement of providers is costly, time-consuming and can even increase provisioning and management complexity with possible router misconfigurations. This is difficult if not infeasible for highly dynamic virtual networks that can change not only in state but also in composition (VR failures, VR migration, Mobility, Dynamic reconfigurations, etc). Two examples are provided below that show clearly the need of an *automated* provisioning and management of VNs and their VRs to deal with dynamic networking environments:

– Dynamic reconfiguration of VNs are expected and needed to handle, for instance, routing protocol changes, dynamic topology updates, new service deployments, etc. VN users may also require full control over their VNs and associated VRs (without any intervention of providers). In this case, a customizable self-service interface is required to allow VN users to create, control and manage their VRs.
– VR Migration: To support user mobility or to simplify the management, a VR may migrate from a substrate node to another without changing the virtual network topology or disrupting the network traffic [9][11].

1.1 Requirements for Future Virtual Routers

Virtual routers should be flexible, modular, open, autonomic and standardized to enable automated provisioning and management of VR services. Virtual Routers should be disaggregated into modular and fine-granular components with standard interfaces, communication protocols and APIs. These components should be automatically aggregated to offer *online* virtual router instances. Virtual routers should also manage themselves (i.e. self-management and autonomic capability [15]) in accordance with high-level goals and policies (representing business-level objectives) from providers or users. The expected capabilities for future virtual routers are:

– Self-provisioning: The VR instances should be created, configured and removed online and automatically to satisfy VN deployment and provisioning requests as they occur.
– Self-awareness: The VRs need detailed knowledge of their states, components, and behaviors and should be aware of their logical and physical resources that can be shared, isolated, allocated or borrowed. The VRs should monitor, maintain and adjust their operations, resources and components during automated VRs provisioning.
– Self-optimization: The VRs should seek ways to optimism itself to improve themselves execution and to ensure that all VRs are running at optimum levels.
– Self-healing: The VRs should discover and repair localized problems, resulting from bugs or failures in software and hardware, to ensure that all VR instances run smoothly.
– Self-protection: The VRs should be capable of identifying, detecting and protecting its resources from malicious attacks.

1.2 Objective and Contribution

To design a virtual routers system that meets the requirements presented above, an Autonomic Virtual Routers (AVR) is proposed to facilitate online and automated VR provisioning and management. The proposed AVR combines the IETF ForCES (Forwarding and Control Element Separation) principles [14] with the autonomic computing concept [15] to design a modular, standardized, composable, customizable and self-management virtual routers. ForCES is defining a set of standard protocols and models to separate the control and forwarding planes for next generation routers. The aim of ForCES is to replace proprietary interfaces with a standard protocol and programmable APIs.

This paper does not address how the AVR would achieve all the previously specified requirements but focuses only on the self-provisioning capability and on the automatic creation of VR instances. A VR instance is considered in this work as a *service* composed of a set of software router components including management, control and forwarding components. The AVR is composed of a pool of virtual resources, logically separated, made available to build the software router components forming the VR services. The interaction between virtual router components relies on the ForCES protocol. The AVR receives provisioning requests from both providers and users, analyzes the requests and allocates the appropriate software router components (i.e. most suitable for the purpose) needed to create, configure and manage the VR services. The AVR integrates an autonomic framework to self-organize the VR service components and their interactions without any human intervention.

Section 2 of this paper presents the generic design of the AVR supporting multiple ForCES based virtual routers. A step-by-step scenario describing the automated provisioning of a VR service in the AVR is provided in section 3. Section 4 concludes this work and provides future work.

2 Autonomic Virtual Routers

This section presents the design of the proposed AVR system. The VR services are provided on demand through the self-provisioning of VR instances. Figure 1 depicts the generic design of the AVR which is composed of three main building blocks:

– *Logical Resource Partitions* (i.e. virtualized resources) which handle *Autonomous Router Components (ARCs)* communicating via standard mechanisms and protocols and forming multiple *ForCES based Virtual Router* instances.
– an *Autonomic Framework* that supports the AVR self-provisioning behavior. The autonomic framework is responsible for organizing the virtualized resources and the Autonomous Router Components to create and configure VRs online.
– an *AVR Interface* used between the AVR itself and the providers, users and other AVRs. The AVR Interface integrates a policy based management framework responsible for receiving and analyzing VR provisioning

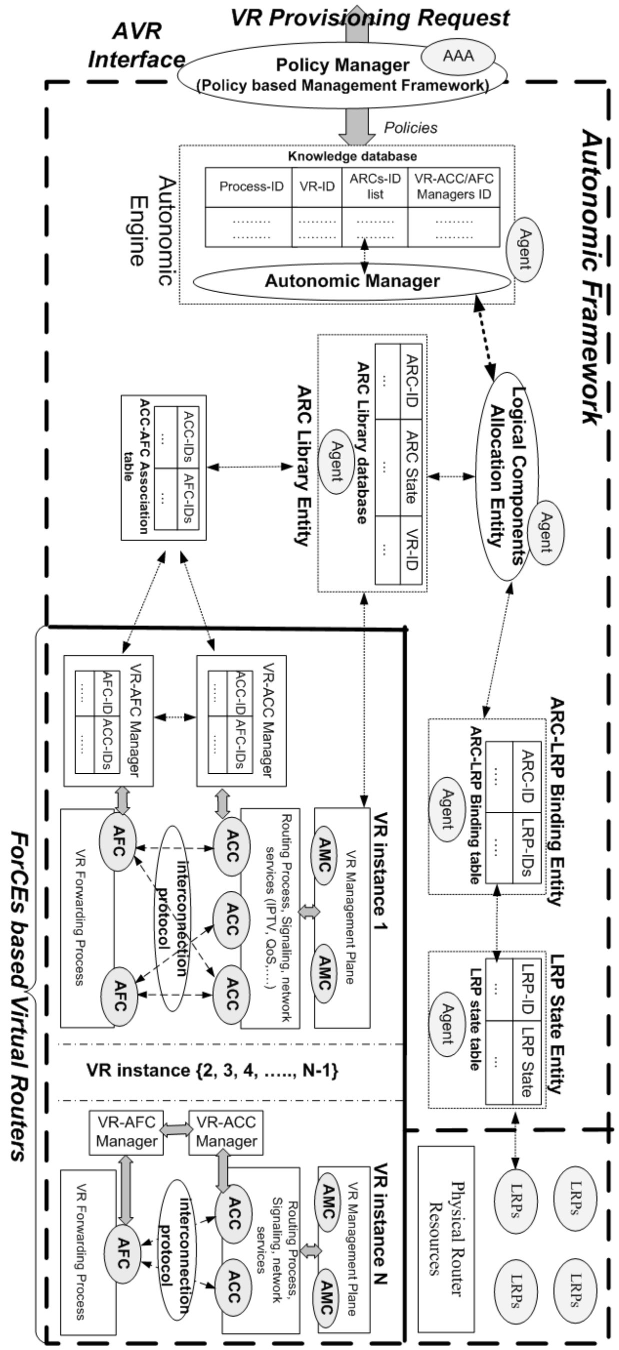


Fig. 1. A generic design of the Autonomic Virtual Routers System

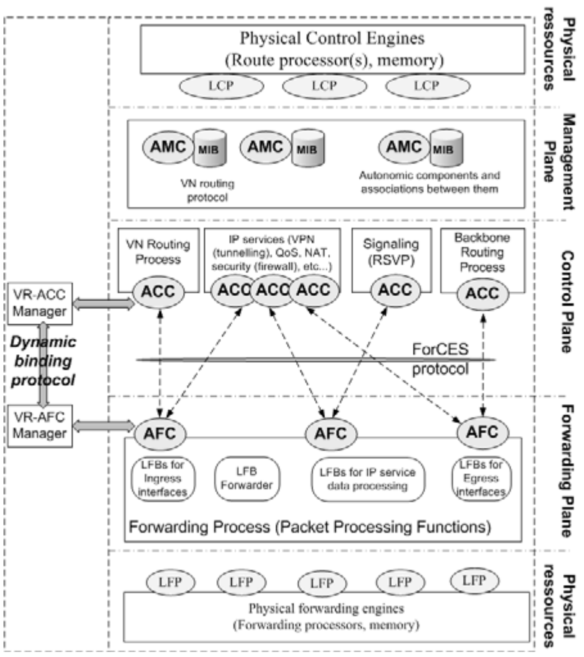

Physical ressources
Physical Control Engines (Route processor(s), memory)
LCP
LCP
LCP
Management Plane
AMC
MIB
AMC
MIB
AMC
MIB
VN routing protocol
Autonomic components and associations between them
Control Plane
VN Routing Process
IP services (VPN (tunnelling), QoS, NAT, security (firewall), etc...)
Signaling (RSVP)
Backbone Routing Process
ACC
VR-ACC Manager
Dynamic binding protocol
VR-AFC Manager
ForCES protocol
Forwarding Plane
AFC
LFBs for Ingress interfaces
LFB Forwarder
LFBs for IP service data processing
LFBs for Egress interfaces
Forwarding Process (Packet Processing Functions)
Physical ressources
LFP
Physical forwarding engines (Forwarding processors, memory)

Fig. 2. Design of one VR using the ForCES model

and management requests. This interface includes also a AAA (Authentication, Authorization and Accounting) module responsible for authenticating and authorizing providers and users to provision VR services.

The next sections detail each AVR building block and explore how virtual routers are provisioned automatically.

2.1 ForCES Based Virtual Router

Logical Resource Partitions: Multiple VR instances may share the physical resources of a physical network node including control components (e.g. memory, route processors, CPU), forwarding components (e.g. network processors) and network interfaces. The virtualization of physical resources provides separate *Logical Resource Partitions* (LRPs). The forwarding components and the control components are divided into Logical Control Partitions (LCP) and Logical Forwarding Partitions (LFP), respectively. As shown in figure 2, a VR instance is supported by a set of LCPs (e.g. CPU and memory slices) and a set of LFPs (e.g. forwarding processor slices, virtual network interfaces, etc).

Design of a ForCES based Virtual Router: In this work, the design of a VR instance is partially based on the router disaggregation concept proposed by the IETF ForCES architecture to standardize the interfaces, protocols and the exchange of information between the control and forwarding components running in the VR. However, the *Control Element* and the *Forwarding Element* defined in ForCES are not expected to have the intelligence to manage their own states and behaviors and to interact with other router components in accordance with policies and rules. VR components should be autonomous and intelligent enough to deal with dynamic composition and aggregation of VR services.

As depicted in figure 2, a ForCES based VR is composed of: *Autonomous Router Components (ARC)* (including forwarding, control and management components), a *VR-ACC Manager* and a *VR-AFC Manager*.

- *Autonomous Forwarding Component (AFC)*: supported by LFPs, operates in the VR forwarding plane. One or multiple AFCs perform the VR forwarding functionalities like packet processing, Forwarding Information Bases (or FIBs), etc. Each AFC is composed of logical processing functions, called LFBs (Logical Function Blocks) [16] .
- *Autonomous Control Component (ACC)*: supported by LCPs, operates in the VR control plane. The ACCs perform VR control functionalities like Routing Information Bases (or RIBs), VN routing protocols, signalling protocols, etc.
- *Autonomous Management Component (AMC)*: supported by LCPs, operates in the VR management plane. The AMCs handle Management Information Objects and Databases responsible for maintaining management information for monitoring, security policies, interfaces and interactions between VR components.
- The VR-ACC Manager maintains the list of ACCs performing the VR control plane and is responsible for the AFC discovery process by determining with which AFC(s) a ACC should communicate within one VR instance. A AFC discovery process is required to provide the capabilities of AFCs available in the AVR.
- The VR-AFC Manager maintains the list of AFCs performing the VR forwarding plane and is responsible for the ACC discovery process by determining with which ACC(s) a AFC should communicate within the VR instance. The ACC discovery process is needed to provide the ACCs capabilities.

The VR-ACC Manager and the VR-AFC Manager communicate via a dynamic binding protocol [17] to discover, associate and exchange ACCs and AFCs capability information. The ForCES protocol [18] is used as a transport protocol between ACCs and AFCs to standardize the information exchange between them. The modular and standard architecture of ForCES provides the flexibility, simplicity and reliability to automatically aggregate VR components.

2.2 Autonomic Framework

The aim of the *Autonomic Framework* is to build virtual routers capable of discovering, configuring, controlling and managing automatically the ARC components and their interactions. As depicted in figure 1, the Autonomic Framework is composed of: *Autonomous Entities* and an *Autonomic Engine*.

Autonomous Entities. The Autonomous Entities are responsible for maintaining, controlling and managing the ARCs. Each Autonomous Entity integrates an Agent responsible for communicating with other Autonomous Entities, receiving/sending requests and manipulating components running in the AVR. The Autonomous Entities include:

– *Logical Resource Partition (LRP) State Entity*: composed of an Agent and an LRP State table maintaining all LRPs of the AVR. Each entry of this table is composed of the LRP identifier and the LRP state: "Allocated" or "Available" state.
– *Autonomic Router Component-Logical Resource Partition (ARC-LRP) Binding Entity*: maintains a pool of predefined ARCs (maintained by the ARC-LRP Binding table) already associated to LRPs. These ARCs are ready for any further allocation procedure conducted by the Logical Components Allocation Entity.
– *ARC Library Entity*: a logical entity that maintains all ARCs in the AVR. The ARC Library Entity is composed of an Agent and a ARC Library database. Each entry in the database is composed of a ARC identifier, the state of this ARC ("Ready" or "Allocated"), and the identifier of the VR integrating the ARC.
– *Logical Components Allocation Entity*: a logical entity responsible for allocating ARCs. The Logical Components Allocation Entity transfers all predefined ARCs from the ARC-LRP Binding Entity to the ARC Library Entity (ARC state: "Ready"). Once ARCs are allocated, they will be defined as "Allocated" in the ARC Library database.
– *ACC-AFC Association Entity*: includes the association table between all ACC identifiers and AFC identifiers in the AVR. This table ensures interpretability between router components. The ACC-AFC Association table interacts with all VR-ACC Managers and VR-AFC Managers of VRs.

Autonomic Engine. The Autonomic Engine integrates an autonomic manager responsible for receiving high-level policies from the AVR interface, analyzing policies, making decisions, monitoring operations, executing actions, coordinating and managing the Autonomous Entities. This engine integrates a Knowledge database maintaining entries composed of:

– Process ID: representing a unique identifier composed of: the user or provider identifier, the VN membership identifier and the identifier of the automated provisioning process.
– VR-ID: a unique identifier of the VR instance
– ARCs-ID list: include the lists of all ARC identifiers involved in the VR instance.
– VR-ACC Manager-ID and VR-AFC Manager-ID associated to the VR instance.

2.3 AVR Interface

The AVR interface receives VR provisioning requests in the following format:

[**Header**: [Request Identifier] [Request Object (VR creation, VR configuration, VR removal)][Allocation process]] [**Data**: [VR creation information], [VR configuration information]]

The Request Object field provides the object of the request that can be: VR creation (creates a VR for the customer if no VR is already installed in the AVR), VR configurations (configures and changes ARCs components in a VR instance) and VR removal (removes a VR instance already installed in the AVR).

The VR creation and VR configuration are achieved via a logical components allocation process: the provider or the user needs, in this case, to allocate ARCs components available in the AVR. The Autonomic Manager, based on the data part of the provisioning request, allocates the required ARCs from the ARC Library Entity via the Logical Components Allocation Entity.

The VR creation information field is related to the VR creation object and provides parameters and capabilities information needed for the VR creation. Based on the VR creation data information, the Autonomic Manager allocates, if possible, the required ARCs available in the AVR to create the VR.

The VR configuration information field is related to the VR configuration object and provides information about updates, changes and modifications of already created VRs. Based on the VR configuration data information, the Autonomic Manager updates, modifies or changes the appropriate ARCs constituting the VR.

3 Scenario of Automated VR Provisioning in the AVR

3.1 VR Provisioning

To understand better the autonomic and self-organizing aspect of the AVR, Figure 3 provides a flow diagram illustrating the interaction between the Autonomic Engine and Autonomous Entities to support automated provisioning of VR services.

During the initial state of the AVR, the Logical Components Allocation Entity interacts with the ARC-LRP Binding Entity to retrieve all available ARCs in the AVR. These ARCs are transferred to the ARC Library Entity (ARC state: "Ready").

Once the AVR receives the VR provisioning and management requests (step 2) the following steps are performed:

1. The Autonomic Engine analyzes the high-level policies and determines, first, the "Request Identifier" information and checks the Knowledge database.
2. The Autonomic Engine retrieves next the "Request Object" (VR creation, VR configuration, VR removal) and self-organizes the interactions between the Autonomous Entities to support the required VR provisioning:

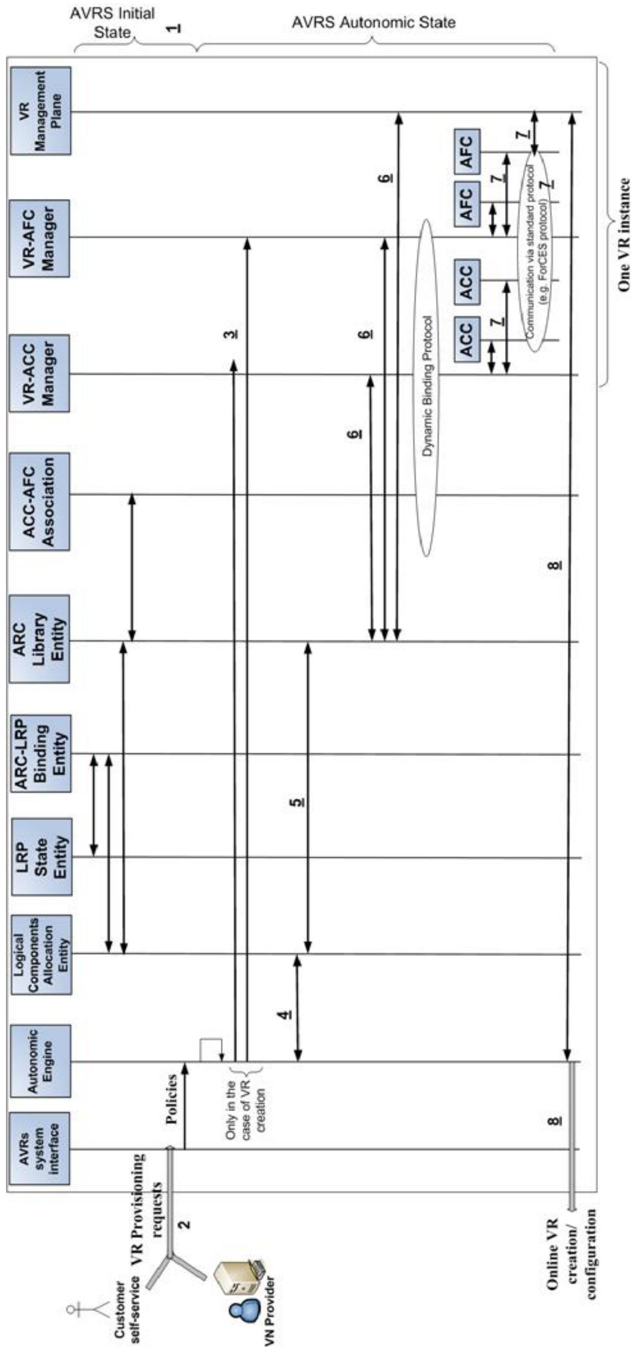


Fig. 3. Automated VR provisioning scenario using the AVR

  - If the "Request Object" is related to the "VR creation" , the Autonomic Engine creates a new VR identifier and adds it in the Knowledge database. Next, the Autonomic Engine instantiates at run time new VR-ACC Manager and new VR-AFC Manager for the new VR control and forwarding planes (step 3).
  - If the "Request Object" is related to the "VR removal", the entry related to the Process-ID specified in the provisioning request will be removed entirely from the Knowledge database. The Autonomic Engine frees, then, all ARCs allocated to the VRs and updates the Autonomous Entities.
3. The Autonomic Engine retrieves the data information included in the provisioning request ([VR creation information] or [VR configuration information]) and interacts with the Logical Components Allocation Entity (step 4) to allocate the required ARCs from the ARC Library Entity (step 5). Once the ARCs are allocated, they will be flagged as "Allocated" (ARC state) and assigned to the appropriate VR-ID in the ARC Library database.
4. The ARC Library Entity, ACC-AFC Association Entity, VR-ACC Manager and VR-AFC Manager communicate via a dynamic binding protocol to decide which ACCs and AFCs will finally form the VR (step 6). The final and specific list of ARCs identifiers constituting the VR instance is sent to the Autonomic Engine to complete the Knowledge database.
5. The VR-ACC Manager activates, next, the allocated ACCs and informs them of the association list of AFC(s) that they should communicate with inside the VR. Likewise, the VR-AFC Manager activates the AFCs and informs them of the association list of ACC(s) that they should communicate with inside the VR (step 7).
6. Finally, a standard protocol (such as the ForCES protocol) interconnects automatically the ACCs and AFCs inside the VR instance and updates the MIB tables, handled by AMCs, in the VR management plane(step 7). A notification request is finally sent to the provider or user to inform about successful provisioning of the VR (step 8).

3.2 VR Migration

As introduced in section 1, a VR instance can migrate from a substrate node to another without changing the virtual network topology. Figure 4 depicts a scenario illustrating the interaction between two AVR systems AVR1 and AVR2 migrating a virtual router instance VR-A from AVR1 to AVR2. AVR1 formulates and sends a VR provisioning request to AVR2 including a "Request Object = VR-A creation" field and information (or profile/context) about VR-A (step 1). Upon receiving the VR provisioning request, AVR2 executes the steps provided in subsection 3.1. Once the virtual router instance VR-A is created (step 2), AVR2 sends a request to AVR1 with "Request Object" tagged as "VR-A Removal" (step 3). Finally, AVR1 removes the VR-A instance from the substrate node (step 4).

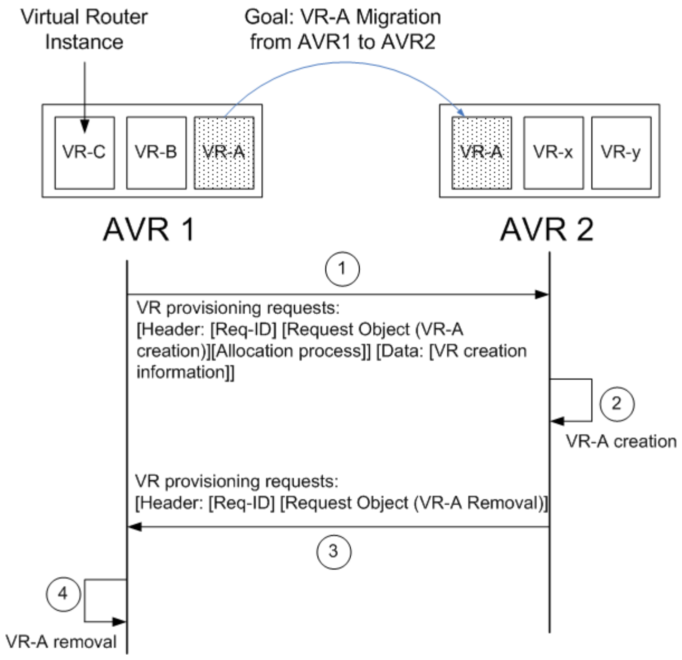

Virtual Router Instance
Goal: VR-A Migration from AVR1 to AVR2
VR-C
VR-B
VR-A
VR-A
VR-x
VR-y
AVR 1
AVR 2
1
VR provisioning requests:
[Header: [Req-ID] [Request Object (VR-A creation)][Allocation process]] [Data: [VR creation information]]
2
VR-A creation
VR provisioning requests:
[Header: [Req-ID] [Request Object (VR-A Removal)]]
3
4
VR-A removal

Fig. 4. VR migration scenario

4 Conclusions and Future Work

This paper presented the design of an Autonomic Virtual Router capable of supporting automated provisioning of on demand Virtual Router services. The proposed AVR system maintains a library of routing and forwarding components to fulfil on demand VR instances expressed by providers and users. The dynamic and automated composition and cooperation of the VR components relies on an autonomic framework responsible for self-organizing and self-managing the interactions between the VR components and the AVR autonomous entities operating these VR components.

In this preliminary overview and design of autonomic virtual routers, only self-provisioning capabilities have been considered. Future work will address other key AVR capabilities such as self-awareness, self-optimization, self-healing and self-protection (including security and isolation paradigms). An implementation is also planned to evaluate thoroughly the AVR system. A multi-agent based approach will be used to build the AVR autonomic engine. Reinforcement learning is also envisaged to facilitate adaptation, evolution and confer the needed autonomic properties to the agent based AVR system.

Since the AVR system enables network virtualization and ensures customized and automated network configuration and management, it can be extended and envisaged as a basic component in data centers, in cloud networking and service oriented infrastructures for the future Internet.

References

[1] Global Environment for Network Innovations (GENI), `http://www.geni.net`
[2] The FP7 4WARD Project, `http://www.4ward-project.eu`
[3] Anderson, T., Peterson, L., Shenker, S., Turner, J.: Overcoming the Internet impasse through virtualization. IEEE Computer Magazine 38(4), 34–41 (2005)
[4] Feamster, N., Gao, L., Rexford, J.: How to lease the Internet in your spare time. SIGCOMM Comput. Commun. Rev. 37(1), 61–64 (2007)
[5] Zhu, Y., Zhang-Shen, R., Rangarajan, S., Rexford, J.: Cabernet: Connectivity architecture for better network services. In: Proc. Workshop on Rearchitecting the Internet (December 2008)
[6] Bavier, A., et al.: VINI veritas: Realistic and controlled network experimentation. In: Proc. ACM SIGCOMM, Pisa, Italy (September 2006)
[7] Cisco XR 12000 Series Router, `http://www.cisco.com/en/US/products/ps6342/`
[8] Kolon, M.: Intelligent logical router service. White Paper Juniper Networks (October 2004)
[9] Louati, W., Zeghlache, D.: Network based Virtual Personal Overlay Networks using Programmable Virtual Routers. IEEE Communications Magazine 43(8), 86–94 (2005)
[10] Heller, B., et al.: Virtualization for a Network Processor Runtime System. Technical Report: WUCSE-2006-62 (November 2006)
[11] Wang, Y., et al.: Virtual routers on the move: Live router migration as a network-management primitive. In: Proc. ACM SIGCOMM (August 2008)
[12] Fu, J., Rexford, J.: Efficient IP-address lookup with a shared forwarding table for multiple virtual routers. In: Proc. CoNext (December 2008)
[13] Egi, N., et al.: Towards high performance virtual routers on commodity hardware. In: ACM CoNEXT 2008 (December 2008)
[14] Yang, L., et al.: Forwarding and Control Element Separation Framework. RFC 3746 (April 2004)
[15] Horn, P.: Autonomic Computing: IBM's perspective on the State of Information Technology. IBM Corporation (October 2001), `http://www.research.ibm.com/autonomic/`
[16] Halpern, J., et al.: ForCES Forwarding Element Model. Internet Draft, work in progress (October 2008)
[17] Lakshman, T.V., et al.: The SoftRouter architecture. In: Proc. ACM SIGCOMM Workshop on Hot Topics in Networking (November 2004)
[18] Doria, A., et al.: ForCES Protocol Specification. Internet Draft, work in progress (March 2009)

Ontological Configuration Management for Wireless Mesh Routers

Iván Díaz[1,⋆], Cristian Popi[2,⋆⋆], Olivier Festor[2,⋆⋆], Juan Touriño[1,⋆], and Ramón Doallo[1,⋆]

[1] Computer Architecture Group Department of Electronics and Systems, University of A Coruña Campus de Elviña s/n, 15071 A Coruña, Spain
`{idiaz,juan,doallo}@udc.es`

[2] MADYNES - INRIA Nancy Grand Est - Research Center 615, rue du jardin botanique 54602 Villers-les-Nancy, France
`{popicris,Olivier.Festor}@loria.fr`

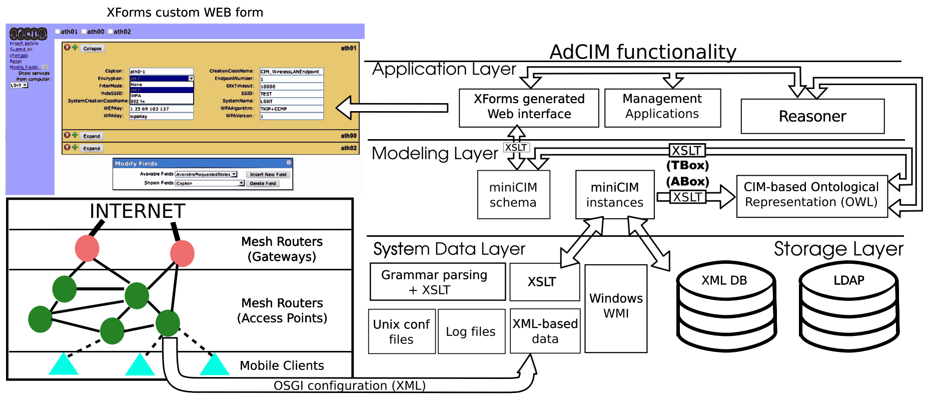
Abstract. Wireless mesh networks (WMNs) are a category of wireless networks that are self-organized, robust and which offer more flexible client coverage with less equipment requirements than wired networks. In WMNs, mesh routers constitute the network's "backbone". The distributed, ever-changing and ad-hoc nature of these networks poses new challenges in configuration management. In order to face them, we modelize the configuration and semantics of a preexisting mesh router using the CIM model and OWL ontology language and implementing XSLT transformations from the original configuration format to CIM/OWL and back. We thus represent it in a higher level of abstraction, an ontological representation that supports configuration semantic checking, policy enforcing and reasoning on the configuration of WMN nodes. We also use the capabilities of our AdCIM framework for persistence and the generation of web configuration interfaces.

1 Introduction

Wireless Mesh Networks [1] replace the classical wired network distribution with a wireless, self-organizing and self-healing infrastructure. This allows for an easy and cheap deployment of access points in places where cabling is costly. A WMN consists of access points, and client nodes. Access points (or mesh routers) form a mesh of fixed nodes, the "backbone", and have a double function: providing access to roaming clients, and relaying data for other routers and other networks (see left side of figure 1). The coverage of a wireless mesh network is extended by means of multi-hop communications. Therefore, mesh routers have additional

⋆ This work was funded by the Ministry of Education and Science of Spain under Project TIN2007-67537-C03, and by the Galician Government (Xunta de Galicia) under Project PGIDIT06PXIB105228PR.

⋆⋆ This work was partially supported by the French Ministry of Research project, AIRNET, under contract ANR-05-RNRT-012-01.



ath01

Caption:	ath0-1	CreationClassName:	CIM_WirelessLANEndpoint
Encryption:	WEP	EndpointNumber:	1
FilterMode:	None	GtkTimeout:	10000
HideSSID:	WEP	SSID:	TEST
SystemCreationClassName:	WPA	SystemName:	LSIIT
	802.1x		
WEPKey:	1 35 69 103 137	WPAAlgorithm:	TKIP+CCMP
WPAKey:	wpaKey	WPAVersion:	1

Modify Fields
Available Fields: AvailableRequestedStates
Shown Fields: Caption

Fig. 1. Application of the AdCIM framework to WMN management

functions to support mesh networking (i.e. routing capabilities), functions which are very important to manage for the performance and health of the network.

There is currently a lack of frameworks for the integrated configuration of WMN routers, so this work explores the application of the AdCIM framework [2] to the configuration of WMN routers. Our approach is shown in figure 1. The router configuration is mapped to the CIM object model [3] and then converted using XSLT into CIM instances (in our custom miniCIM format). Then the functionality of the AdCIM framework is exploited, including persistence in LDAP directories and the generation of web forms to manipulate these instances. The modified data is reverted to its original format and updated in the router, or transformed to an OWL [4] based semantic representation, to check internal consistency in the configuration and infer new data.

Using the CIM model and its OWL representation opens new possibilities to diagnose mesh network problems or to simulate the effect of a proposed configuration change globally. It also provides a higher level view that hides implementation details and that is generalizable to routers with other architectures which could then be managed homogeneously. We chose the configuration of a modular wireless router developed by the Image Sciences, Computer Sciences and Remote Sensing Laboratory (LSIIT) RP team [5]. This router configuration is managed using the OSGi [6] Java-based framework, which supports on-the-fly management and deployment of modules, and to support starting, stopping and uninstalling them independently.

This paper is structured as follows. First, Section 2 details background information and persistence and user interface aspects of the AdCIM framework related with this work. Section 3 contains an analysis of the router configuration and the entities it represents. After that, Section 4 elaborates about the mapping of these entities to CIM and the implementation of this mapping in CIM. The transformation from CIM to the ontological representation, and its use in semantic checking are developed in Section 5. Section 6 explores works related with this paper, and, finally, Section 7 the conclusions.

2 Background and Management Infrastructure

This section outlines some technologies used in this work and the management infrastructure provided by AdCIM.

From the W3C we use three standards: XSLT, XForms, and OWL. XSLT [7] is a template language for XML transformations. Its declarative nature allows transformations to be further optimized and parallelized by the interpreter. XForms [8] is designed as a replacement for HTML forms and uses XML data as input and output, supports dynamic form changes and off-line validation without server intervention or Javascript support. OWL (Web Ontology Language) [4] represents formally knowledge domains organized as hierarchical classifications and supports reasoning tasks on them, such as concept satisfiability and consistency. OWL supports three flavors, OWL Lite, OWL-DL, and OWL Full, which represent various compromises between expressivity and computability.

CIM (Common Information Model) [3] is a standard from the Distributed Management Task Force (DMTF) that defines an object-oriented and extensible information model to represent configuration data. It covers a vast spectrum of configuration information, ranging from the logical (such as device capabilities, operational status, software dependencies) to the physical (e.g., temperature, physical location, cabling, card placement), and relates all these entities via associations, which represent much of the semantical information in CIM.

To manage these CIM instances and associations, in this case modelling the configuration of the router, we use our AdCIM framework [2], of which figure 1 shows a rough overview. This framework supports the extraction of configuration data as CIM instances, even from unstructured sources. It validates and stores these data in LDAP directories and generate user interfaces via XForms. OWL data for the reasoning processes that will be presented in Section 5 are also generated using XSLT and divided into ABox (declarative) statements deriving from the instances, and TBox (terminological) statements deriving from the schema, as depicted in the figure.

Data persistence in AdCIM is modularized, but the preferred solution is LDAP [9], since it supports partial replication and scalability for efficient decentralized management. Also, it is more flexible than relational databases to store and retrieve miniCIM instances. LDAP storage of miniCIM data is handled with XSLT stylesheets that transform XML to and from LDAP format according to the miniCIM schema and the DMTF recommendations in [10].

Since the transformation to and from directory data must be called at every query, the stylesheet processor is configured to cache the CIM schema to ensure optimal response times and stylesheets make intensive use of the XSLT `<xsl:key/>` operator, as well as other optimization techniques.

AdCIM generates forms that retrieve the entities represented by CIM classes from the repository, modify them honoring schema restrictions, and support adding or deleting supported fields and instances. These forms are generated on-the-fly from the schema and a preexisting template, and then further processed to generate a standard HTML+Javascript form compatible with all major browsers.

```
<configurations>
  <level name="device">
    <level name="ethernet">
      <level name="interface">
        <configuration name="br31">
          <String name="ConfigurationType">interface</String>
          <Integer name="UpdateType">0</Integer>
          <String name="device.deviceType">ethernet</String>
          <String name="device.ethernet.broadcast">0.0.0.0</String>
          <Short name="device.ethernet.flags">1</Short>
          <String name="device.ethernet.ip">130.79.91.223</String>
          <Boolean name="device.ethernet.ipDesactivated">false</Boolean>
          <StringArray name="device.ethernet.ipv6addresses"/>
          <Integer name="device.ethernet.mtu">1500</Integer>
          <String name="device.ethernet.netmask">255.255.254.0</String>
          <Boolean name="device.ethernet.usingDHCP">false</Boolean>
          <String name="device.interfaceName">br31</String>
          <String name="device.virtualName">br31</String>
          <String name="service.bundleLocation"> file:ap-bundles/devmng_eth.jar </String>
          <String name="service.pid"> device.ethernet.interface.br31 </String>
        </configuration>
      </level>
    </level>
  </level>
</configurations>
```

Fig. 2. OSGi-based mesh router configuration for an IP interface

```
<CIM_OSGiConfSettingData namespace="dc=udc">
  <BundleLocation>
    file:ap-bundles/devmng_eth.jar
  </BundleLocation>
  <InstanceID>device.ethernet.interface.br31</InstanceID>
  <ConfigurationType>interface</ConfigurationType>
  <UpdateType>0</UpdateType>

<CIM_IPProtocolEndpoint namespace="dc=udc">
  <SystemCreationClassName>
    CIM_ComputerSystem
  </SystemCreationClassName>
  <SystemName>LSIIT</SystemName>
  <CreationClassName>
    CIM_IPProtocolEndpoint
  </CreationClassName>
  <Name>device.ethernet.interface.br31</Name>
  <Caption>br31</Caption>
</CIM_IPProtocolEndpoint>

<CIM_ElementSettingData namespace="dc=udc">
  <IsCurrent>Is Current</IsCurrent>
  <ManagedElement>
    <ref classname="CIM_IPProtocolEndpoint"
         namespace="dc=udc">
      <CreationClassName>
        CIM_IPProtocolEndpoint
      </CreationClassName>
      <Name>device.ethernet.interface.br31</Name>
      <SystemCreationClassName>
        CIM_ComputerSystem
      </SystemCreationClassName>
      <SystemName>LSIIT</SystemName>
    </ref>
  </ManagedElement>
  <SettingData>
    <ref classname="CIM_OSGiConfSettingData"
         namespace="dc=udc">
      <InstanceID>
        device.ethernet.interface.br31
      </InstanceID>
    </ref>
  </SettingData>
</CIM_ElementSettingData>
```

```
<CIM_IPAssignmentSettingData namespace="dc=udc">
   <InstanceID>br31</InstanceID>
   <AddressOrigin>Static</AddressOrigin>
</CIM_IPAssignmentSettingData>
<CIM_StaticIPAssignmentSettingData namespace="dc=udc">
   <InstanceID>br31-static</InstanceID>
   <IPv4Address>130.79.91.223</IPv4Address>
   <SubnetMask>255.255.254.0</SubnetMask>
   <GatewayIPv4Address>130.79.91.254</GatewayIPv4Address>
</CIM_StaticIPAssignmentSettingData>

<CIM_ElementSettingData namespace="dc=udc">
  <ManagedElement>
    <ref classname="CIM_IPProtocolEndpoint"
         namespace="dc=udc">
      <CreationClassName>
        CIM_IPProtocolEndpoint
      </CreationClassName>
      <Name>device.ethernet.interface.br31</Name>
      <SystemCreationClassName>
        CIM_ComputerSystem
      </SystemCreationClassName>
      <SystemName>LSIIT</SystemName>
    </ref>
  </ManagedElement>
  <SettingData>
    <ref classname="CIM_IPAssignmentSettingData"
         namespace="dc=udc">
      <InstanceID>br31</InstanceID>
    </ref>
  </SettingData>
</CIM_ElementSettingData>

<CIM_ConcreteComponent namespace="dc=udc">
  <GroupComponent>
    <ref classname="CIM_IPAssignmentSettingData"
         namespace="dc=udc">
      <InstanceID>br31</InstanceID>
    </ref>
  </GroupComponent>
  <PartComponent>
     <ref classname="CIM_StaticIPAssignmentSettingData"
          namespace="dc=udc">
        <InstanceID>br31-static</InstanceID>
     </ref>
  </PartComponent>
</CIM_ConcreteComponent>
```

Fig. 3. Excerpt from output of transforming fig. 2 configuration into miniCIM format

To find inconsistencies in the miniCIM data shown by these forms we use the programmatic Java interface of the Pellet reasoner.

3 OSGi Configuration Analysis

The router subject to study provides a naming schema and structure for the configuration attributes that conform to the standard OSGi Configuration Service which is the component of OSGi tasked with managing the settings of other services and their persistence. It defines *configuration objects* that contain *configuration dictionaries*, a collection of name-value pairs that represent the settings of an OSGi service; objects also have a *PID* (persistent identifier) as primary key. OSGi services can register themselves to a PID to receive a dictionary, or to a *configuration factory* to receive an arbitrary number of dictionaries registered in the factory.

The LSIIT router stores its configuration objects as XML data in the format seen in figure 2. Configuration objects have structured PIDs used for references and located in a hierarchy similar to that of Java packages. OSGi properties name, type and value are codified as an XML element attribute, name, and value, respectively. The router conceptual entities mapped by these objects are:

Services. There are three services in the router configuration: SNMP, Bridging and Telnet. Each has a very different configuration: SNMP only needs to be set as started or stopped, Bridging needs a list of network interfaces; so the mapping of each service is different in each case.

IP Interfaces. These entities represent various virtual interfaces on top of the wireless interfaces. Their set of properties includes IP address and net mask and DHCP configuration. While located in level `device.ethernet.interface`, they mostly represent IP configurations, with two properties (*MTU* and *Flags*) representing transport level properties. They can be related to wireless interfaces or be implemented by the bridging service.

Logical Wireless Interfaces. They represent the aspects of wireless interface configuration that reside in a higher level than physical configuration. These aspects include encryption algorithms and settings, Radius server configuration, VLAN configuration, SSID (wireless network name), MAC filtering and related parameters like the link quality level, window size or link hysteresis control. Each one of these entities are generally associated with an IP interface and a physical wireless interface.

Physical Wireless Interfaces. They represent the low-level settings of a wireless interface and are bound to logical interfaces. These settings include transmission channel or frequency and transmission power.

Other entities. Other entities in the configuration include a generic IP routing default gateway setting and password information. There are also entities to represent virtual LAN settings.

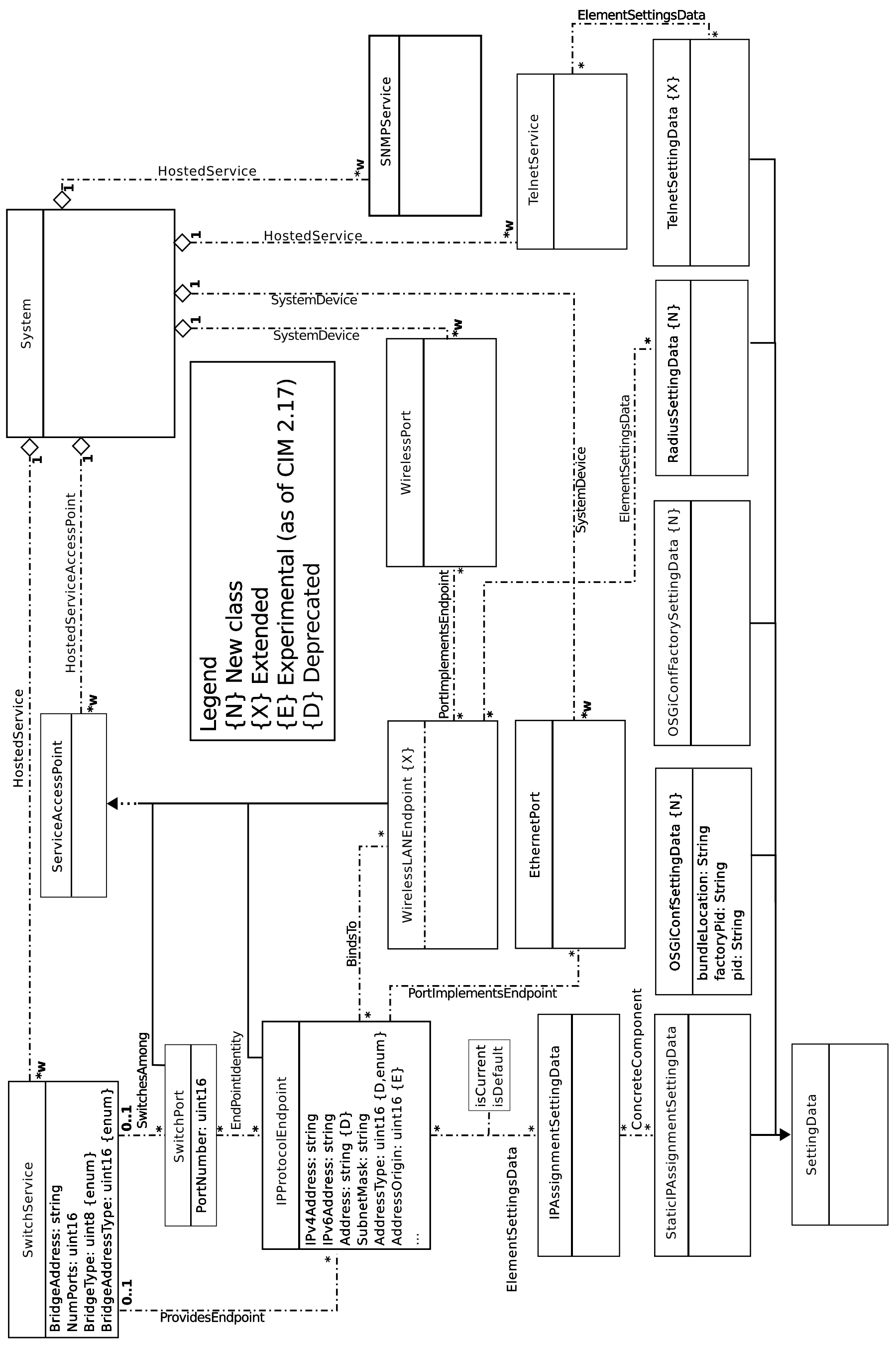


Fig. 4. CIM mapping class hierarchy

4 Mapping of the Router Configuration to CIM

This section covers the mapping from the original format of the router configuration to the CIM model that is later translated to OWL. According to the classification presented in Rivière et al [11], the mapping in this work follows the "recast" philosophy, since "concepts" are mapped. It also follows the principle of abstract translation, since redundant information is removed. Finally, the organization is independent, since the resulting model is standard. In figure 4, we show the CIM class hierarchy used to map the router configuration. We will explore both this mapping and the XSLT templates that implement it.

The CIM mapping of figure 4 can be separated in two abstraction levels, one including the `OSGIConfSettingData` and `OSGIConfFactorySettingData` classes, both representing OSGi-related structures and identifiers, and the other level representing more abstract entities. The two abstraction levels simplify the recovery of the original configuration file and, at the same time, avoid pollution of OSGi specific attributes on abstract entities.

The second abstraction level includes some specialized service classes, such as `SNMPService`. `SwitchService` represents the bridging facility and is associated to a list of `SwitchPort` instances, each associated to an `IPProtocolEndpoint`. The bridging service is also accessed as an IP interface of its own, related with `ProvidesEndpoint`. These `IPProtocolEndpoints`, which represent IP interfaces, are related to an `IPAssignmentSettingData` which indicates if the address setting is static or via DHCP. In the first case, it is further associated with a `StaticIPAssignmentSettingData` which contains the IP data. There are cases in which no IP assignment data will be given.

`IPProtocolEndpoint` instances can be associated to a `WirelessLANEndpoint` instance which represents logical wireless interfaces. Each one can have several Radius configurations, represented by the `RadiusSettingData` class. The IP interfaces maximum transfer unit value is moved to the `EthernetPort` class. Finally, physical wireless interface data are included in `WirelessPort` instances, related to `WirelessLANEndpoint` by the `PortImplementsEndpoint` association.

The properties of an OSGi dictionary are usually directly mapped to CIM properties, but there are some properties that are not mapped at all to CIM, such as boolean values that control the expression of others. For example, `device-.ethernet.ipDesactivated` shows if the IP interface has a valid IP configuration. Similarly, invalid fields can be omitted instead of being represented by dummy values.

4.1 XSLT Implementation

The implementation of the transformations is done with two XSLT stylesheets: one that transforms the XML OSGi configuration data into CIM data and another one that does the opposite process. The result of the application of the first one to the configuration shown in figure 2 can be seen in figure 3. We use the miniCIM XML format (more detailed in [2]), which is a custom format that stores schema data separately, being much more compact and efficient than CIM-XML [12], the official XML mapping representing CIM data.

Internally, the first stylesheet can create any particular CIM association with the association endpoints and properties as arguments. Depending on the configuration PID of each entry, appropriate templates that create CIM classes and associations are invoked and a second pass adds relationships which would require backtracking in the first pass. The second template, which converts CIM data back to the OSGi configuration, can similarly follow CIM associations and thus rebuild the OSGi configuration retrieving the CIM instances pointed by OSGi-related associations. This template can also recover the level structure of the file by parsing their PID values. These two templates use pattern matching to allow extensibility: recognized elements trigger special case processing and unknown elements only are mapped at low level, without aborting the transformation.

5 Ontology Representation

Section 4 showed a semi-formal representation that covers taxonomical classification and domain knowledge, but this representation is not formal because many domain constraints and metadata are not expressed explicitly, so is not possible to infer and reason over the data without a priori knowledge of the semantics of the domain (see Quirolgico et al. [13]). For example, in the domain of WMNs, the channel information of a wireless interface actually maps into a range of frequencies that might be unusable because of national regulation or interference with other nearby equipment. The existence of several usable bands, and proprietary wi-fi protocols further complicates the issue. The use of an ontology and a reasoner (a program implementing logical reasoning) allows to deduce a conflict in those situations.

Configuration semantic checking is another motivator. Sinz et al. [14] and Glasner et al. [15], verify logical constraints in the Apache configuration file. These constraints are not concerned with mere well-formedness, but with semantic integrity. Checking this with ontologies has many advantages. For instance, problems in higher levels of abstraction are traced logically by the reasoner to lower-level causes and other configuration formats can be expressed with the same model, without changing the underlying logic.

Other advantage is the formal enactment of policies. A rule language like SWRL [16] allows to specify Horn-like rules of the form $H \leftarrow B_1, \ldots, B_n$, in which the head H is asserted if all the body atoms $B_{1 \ldots n}$ are true. Nevertheless, Motik et al. [17] show that the naïve combination of OWL-DL and unrestricted SWRL rules is undecidable (not guaranteed to end in the worst case), but it is decidable if rule variables are restricted to known individuals. SWRL rules extend OWL when more expressivity is needed (e.g. role composition like in a hypothetical property *uncleOf*) or there is no reasoner support (e.g. reasoning and mathematical operations with datatypes).

We chose OWL-DL as the format for our ontologies. Among the OWL flavors, OWL-DL has the best balance between expressivity and efficiency; Lite is too restrictive and Full is undecidable and inefficient. All are based on description logics, that are fragments of first order logic (FOL), in turn, propositional logic with existential and universal quantifiers. Full FOL is not used because of its undecidability and computational intractability. Description logics differ in the operations retained (or added) from FOL. OWL-DL supports these operators:

$$C \rightarrow \neg C \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid A \mid \exists R.C \mid \forall R.C \mid \geq nS \mid \leq nS \mid a_1, \ldots, a_n \quad (1)$$

where C and C_i are concepts or classes, A an atomic concept, $R.C$ describes a binary role or property, S is a property name, n is an integer number that indicates its minimum or maximum cardinality, and a_i are named individuals. Thus, the operations include negation, set union, set intersection, existential quantification, universal quantification, number restrictions and named individuals. Individuals can belong to several concepts and roles represent logic predicates involving concepts that can be transitive. Known facts or "axioms" are grouped in the *knowledge base*, divided in the TBox ("terminological box") which hierarchically groups axioms about concepts and roles and their mutual inclusion, and the ABox ("assertional box"), which contains knowledge about individuals (and their inclusion in a concept). A reasoner can perform several tasks with that information, such as classifying individuals in concepts, restructuring the concept hierarchy and detecting inconsistencies. We have used the open-source reasoner Pellet [18], which has support for both OWL and SWRL.

5.1 CIM Transformation to OWL

Works like the ones by Heimbigner [19], Majewska et al. [20] and García et al. [21] have already implemented mappings of CIM to OWL-DL, and remark the lack of equivalences for some CIM constructs in OWL. Our mapping is closer to the second and third works, and is implemented with XSLT. Our mapping approach from the CIM schema to OWL TBox is roughly described as:

CIM entity	**OWL TBox mapping**
Class	`<owl:Class>`, defined as a closed set of individuals using `<owl:OneOf>`
SubClass	`<rdfs:subClassOf>`, all subclasses declared as disjoint to one another
Properties	`DataProperties` with appropriate types to map CIM types
References	cardinality 1 `ObjectProperty`. Inverse properties are inferred automatically by the reasoner
Association	subclasses of `Association`. Each instance limited to `DataProperty` and two cardinality 1 `ObjectProperty`
Cardinality	Normal in `DataProperty`. `ObjectProperty` in associated classes by limiting the cardinality of inverse object properties
Key	Not implemented, causes undecidability without being needed

We adapted some of the ideas of previous works and rejected others, for example, the approach for mapping property names in [19] is very cumbersome and we chose instead to append the class name as a prefix. Mapping CIM key properties requires OWL Full, which is undecidable, so instead a unique identifier is assigned for each instance conserving the semantics. Our approach of "splitting" associations tries to simplify their closure and preserves the semantics of cardinality. Since OWL properties are binary, we require separate instances for associations to house possible association attributes.

Closing the world. Our mapping provides axioms to obtain world closure. OWL reasoners by default operate by the *Open World Assumption*, so unstated facts are not false, merely unknown. This also makes membership by negation (or *negation as failure*) very hard to verify, so it only works if there is no possible new knowledge that invalidates the negation. In this case, since the configuration file is a closed universe of discourse, we prevent this by declaring explicitly the inexistence of additional instances, restricting the cardinality of associations, and defining all individuals pairwise disjoint. This closes the world and allows negation as failure. This can be viewed as a drawback of OWL, but it also allows to open parts of the world, e.g. parts modifiable by unknown external imports.

5.2 OWL Reasoning Implementation

Once the information contained in the miniCIM schema and instances is translated to TBox and ABox axioms, additional axioms are introduced in an included file that verify some conditions. This section shows some examples of applicable restrictions for configuration checking. To better understand this section, refer to figure 4.

The first example shows a simple case in which unconfigured entities are detected; in this case, ports in a switching service that are not configured. This avoids the possibility of the switching service failing due to misconfiguration and causing a malfunction in the node.

$$SwitchPort\_Undefined \equiv SwitchPort \cap \tag{2a}$$

$$\neg(\exists EndpointIdentitySystemElement^{-}.EndpointIdentity) \tag{2b}$$

This restriction declares an *undefined* port in a switching service as a port not represented with a network endpoint. This is checked by the presence of an inverse property from association `EndpointIdentity`, x^{-} in 2b. The values of these inverse properties are inferred automatically by the reasoner. Since this restriction uses negation as failure, the possible instances of the `EndpointIdentity` association and `SwitchPort` must be closed for it to work.

The effects of a configuration error are made to cascade to other entities defining intermediate classes (OWL-DL does not support composition of properties). The reasoner automatically determines the proper evaluation order, and the cascading can be made arbitrarily deep. The type of errors detected by this process are important in wireless nodes, since obscure high-level errors might have simple motivations solvable on-the-fly:

$$Not\_current\_IP\_setting \equiv \tag{3a}$$
$$ElementSettingData \cap \tag{3b}$$
$$(\exists ElementSettingDataIsCurrent = false \,|\, xsd : string) \cap \tag{3c}$$
$$(\exists ElementSettingDataSettingData.IPAssignmentSettingData) \tag{3d}$$

$$Unconfigured\_IP\_Endpoint \equiv IPProtocolEndpoint \cap \tag{3e}$$
$$((\forall ElementSettingDataManagedElement^{-}.Not\_current\_IP\_setting) \tag{3f}$$
$$\cup\, (\neg(\exists ElementSettingDataManagedElement^{-}.IP\_setting))) \tag{3g}$$
$$BindsTo\_Unconfigured\_IP\_Endpoint \equiv \tag{3h}$$
$$BindsTo \,\cap (\exists BindsToAntecedent.Unconfigured\_IP\_Endpoint) \tag{3i}$$

The term 3c selects IP setting instances that are not currently used, term 3d deselects `ElementSettingData` instances not related to IP Settings and the special intermediate class `Unconfigured_IP_Endpoint` is defined as one Endpoint with no IP settings (3e), or one in which none are current (3f). Finally, term 3h declares a subclass of the association `Binds_To` grouping those instances that bind with `Unconfigured_IP_Endpoint` instances. In that way, semantic errors are propagated so they can help diagnose problems in top-level entities.

Policies are also implemented by SWRL rule chaining. SWRL allows both straightforward composition without intermediate classes, and performing inequality comparisons with datatype ranges (instead of only supporting equality comparisons) . Some OWL reasoners translate the rules and relevant OWL axioms to another rule engine, sometimes changing the semantics, but the Pellet reasoner integrates them fully with OWL axioms. These rules, for example, detect wireless ports that have illegal frequencies depending on the legislation of the country:

$$WirelessPortChannel(?x, ?y) \wedge swrlb : multiply(?y5, ?y, 5) \tag{4a}$$
$$\wedge\, swrlb : add(?z, ?y5, 2412) \rightarrow WirelessPortFrequency(?x, ?z) \tag{4b}$$

$$located(Japan, ?y) \;\wedge\; ComputerSystem(?y) \wedge \tag{4c}$$
$$\wedge\; SystemDeviceGroupComponent(?z, ?y) \wedge \tag{4d}$$
$$\wedge\; SystemDevicePartComponent(?z, ?a) \;\wedge\; WirelessPort(?a) \wedge \tag{4e}$$
$$\wedge\; WirelessPortChannel(?a, ?b) \;\wedge\; swrlb : greaterThanOrEqual(?b, 2500) \tag{4f}$$
$$\rightarrow IllegalFrequency\_WirelessPort(?a) \tag{4g}$$

$$located(USA, ?y) \;\wedge\; ComputerSystem(?y) \wedge \tag{4h}$$
$$\wedge\; SystemDeviceGroupComponent(?z, ?y) \wedge \tag{4i}$$
$$\wedge\; SystemDevicePartComponent(?z, ?a) \;\wedge\; WirelessPort(?a) \wedge \tag{4j}$$
$$\wedge\; WirelessPortFrequency(?a, ?b) \;\wedge\; swrlb : greaterThanOrEqual(?b, 2467) \tag{4k}$$
$$\rightarrow IllegalFrequency\_WirelessPort(?a) \tag{4l}$$

Since the legal spectrum depends on the country, it first determines the location of the system housing the wireless ports (4c, 4h), and then finding their frequency (4d-4g, 4i-4k). Since these frequencies will be usually expressed as channels, terms 4a and 4b calculate the frequency by using SWRL built-in operations. Finally, the frequency of each port is compared with the legal maximum (term 4f, 4k) and minimum (not shown). If this value is out of limits (for example, channel 13 in the USA), the ports are classified in the `IllegalFrequency_WirelessPort` subclass.

More complex policies and taxonomies are defined using SWRL to define new properties. In the following example, `WirelessPort`s are defined as interfering one another (i.e. a symmetric property) if their frequency difference is less than 25Mhz (5g). This property can be used in turn in more complex rules. For example, if throughput degradation between two ports is detected, interference can explain its origin.

$$ComputerSystem(?y) \wedge SystemDeviceGroupComponent(?z, ?y) \wedge \tag{5a}$$
$$\wedge\ SystemDevicePartComponent(?z, ?a) \wedge \tag{5b}$$
$$\wedge\ SystemDeviceGroupComponent(?z, ?y1) \wedge \tag{5c}$$
$$\wedge\ SystemDevicePartComponent(?z, ?a1) \wedge WirelessPort(?a) \wedge \tag{5d}$$
$$\wedge\ WirelessPort(?a1) \wedge WirelessPortFrequency(?a, ?b) \wedge \tag{5e}$$
$$\wedge\ WirelessPortFrequency(?a1, ?b1) \wedge swrlb : subtract(?delta, ?b, ?b1) \wedge \tag{5f}$$
$$\wedge\ swrlb : lessThanOrEqual(?delta, 24) \rightarrow interferes(?a, ?a1) \tag{5g}$$

6 Related Work

Some works have been done in the area of managing wireless mesh networks, for example, a simulator-based scheme for troubleshooting faults in WMNs by Qiu et al [22]. Zhang et al [23] present an attack–resilient security architecture for WMNs.

For the configuration and accounting of WMNs, commercial solutions are available from Nortel [24] or LocustWorld [25]. Staub et al [26] tackle the challenges of defective configurations or errors in mesh routers, and propose a distributed automated reconfiguration architecture. The approach uses cfengine [27] to distribute configuration and updates among the nodes in the WMN backbone, but it is based on the over-writing of the current configuration and does not allow for extraction and analysis of the current state of the network.

Other works use ontologies or logic models for configuration. Sinz et al. develop in [14] a CIM-based formal model similar to description logics to check for inconsistencies on the Apache configuration, as does Glasner et al. in [15] but using a custom OWL model and with more emphasis on decidability. García et al. [21] transform the CIM model into OWL and use SWRL rules, but do not give details about their implementation or undecidability. Quirolgico et al. [13] is an earlier exploratory effort using RDFS, but lacks cardinality restrictions, used in [14][15] and in our work for certain structural constraints.

7 Conclusions

We have presented support in our AdCIM framework for the configuration of wireless mesh networks as well as reasoning processes on their management data. We have implemented it analyzing the configuration of a real mesh network router, separating its format and underlying entities, and defining a CIM mapping that supports the complete expression of this configuration, taking special care to store format intricacies without losing abstraction. This mapping and its opposite was implemented with XSLT templates. AdCIM is model-driven, based on standards and supports automatically generated forms and LDAP-based persistence.

Our approach represents the current state of the WMN in an integrated manner that allows off-line analysis, useful to prevent broken configuration states before deployment. This analysis is supported with an ontological model that detects semantic errors and conflicts and supports policy enforcement. This semantical representation is exploited by discovering semantical equivalences (i.e., inferring individuals class pertenence and property values), and by navigating the associations in the underlying CIM model (e.g. using both the physical location and vendor of a product to enforce rules). We are currently working on the integration of our approach with other mesh routers, and reusing the same basic ontology and reasoning processes with other configuration formats and domains.

References

1. Akyildiz, I.F., Wang, X., Wang, W.: Wireless Mesh Networks: A survey. Computer Networks 47(4), 445–487 (2005)
2. Diaz, I., Touriño, J., Salceda, J., Doallo, R.: A framework focus on configuration modeling and integration with transparent persistence. In: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS 2005). Workshop on System Management Tools for Large-Scale Parallel Systems, Denver, Colorado, USA, April 2005, p. 297a (2005), http://adcim.des.udc.es
3. DMTF. Common Information Model (CIM) Standards, http://www.dmtf.org/standards/cim (accessed July 2009)
4. W3C. OWL 1.0 (2004), http://www.w3.org/TR/2004/REC-owl-features-20040210/ (accessed July 2009)
5. Image Sciences, Computer Sciences and Remote Sensing Laboratory Research Unit, http://lsiit.u-strasbg.fr/ (accessed July 2009)
6. OSGi Alliance, http://www.osgi.org/Main/HomePage (accessed July 2009)
7. W3C. XSL Transformations (XSLT) Version 1.0 (1999), http://www.w3.org/TR/xslt (accessed July 2009)
8. W3C. XForms 1.0 (2003), http://www.w3.org/TR/xforms (accessed July 2009)
9. Loshin, P.: Big book of LDAP RFCs. Morgan Kaufmann, San Francisco (2000)
10. Wood, E.: Guidelines for CIM-to-LDAP directory mappings (2000), http://www.dmtf.org/standards/documents/DEN/DSP0100.pdf (accessed July 2009)
11. Rivière, A., Sibilla, M.: Management information models integration: From existing approaches to new unifying guidelines. Journal of Networks and System Management 6(3), 333–356 (1998)

12. DMTF. Specification for the Representation of CIM in XML (2002), `http://www.dmtf.org/standards/documents/WBEM/DSP201.html` (accessed July 2009)
13. Quirolgico, S., Assis, P., Westerinen, A., Baskey, M., Stokes, E.: Toward a formal Common Information Model ontology. In: Bussler, C.J., Hong, S.-k., Jun, W., Kaschek, R., Kinshuk, Krishnaswamy, S., Loke, S.W., Oberle, D., Richards, D., Sharma, A., Sure, Y., Thalheim, B. (eds.) WISE 2004. LNCS, vol. 3307, pp. 11–21. Springer, Heidelberg (2004)
14. Sinz, C., Khosravizadeh, A., Kuchlin, W., Mihajlovski, V.: Verifying CIM models of Apache web-server configurations. In: Proceedings of the 3rd International Conference on Quality Software, QSIC 2003, Dallas, USA, November 2003, pp. 290–297 (2003)
15. Glasner, D., Sreedhar, V.C.: Configuration reasoning and ontology for web. In: Proceedings of the 4th IEEE International Conference on Services Computing, SCC 2007, Salt Lake City, USA, July 2007, pp. 387–394 (2007)
16. W3C Member submission. SWRL (2004), `http://www.w3.org/Submission/SWRL/` (accessed July 2009)
17. Motik, B., Sattler, U., Studer, R.: Query answering for OWL-DL with rules. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 549–563. Springer, Heidelberg (2004)
18. Clark, K., Parsia, B.: Pellet: The Open Source OWL DL Reasoner, `http://clarkparsia.com/pellet/` (accessed July 2009)
19. Heimbigner, D.: DMTF - CIM to OWL: A case study in ontology conversion. In: Proceedings of the 16th Conference on Software Engineering and Knowledge Engineering, SEKE 2004, Banff, Canada, June 2004, pp. 470–473 (2004)
20. Majewska, M., Kryza, B., Kitowsky, J.: Translation of Common Information Model to Web Ontology Language. In: Shi, Y., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2007. LNCS, vol. 4487, pp. 414–417. Springer, Heidelberg (2007)
21. García, F., Martínez, G., Botía, J., Gómez-Skarmeta, A.: On the application of the Semantic Web Rule Language in the definition of policies for system security management. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM-WS 2005. LNCS, vol. 3762, pp. 69–78. Springer, Heidelberg (2005)
22. Qiu, L., Bahl, P., Rao, A., Zhou, L.: Troubleshooting wireless mesh networks. ACM SIGCOMM Computer Communication Review 36(5), 17–28 (2006)
23. Zhang, Y., Fang, Y.: ARSA: An Attack-Resilient Security Architecture for multihop wireless mesh networks. IEEE Journal on Selected Areas in Communications 24(10), 1916–1928 (2006)
24. Nortel WMN solutions, `http://www.nortel.com` (accessed July 2009)
25. LocustWorld, `http://www.locustworld.com` (accessed July 2009)
26. Staub, T., Balsiger, D., Lustenberger, M., Braun, T.: Secure remote management and software distribution for wireless mesh networks. In: Proceedings of the 7th International Workshop on Applications and Services in Wireless Networks. ASWN 2007, Santander, Spain (May 2007)
27. Burgess, M.: Cfengine: A site configuration engine. Computing Systems 8(3), 1–29 (1995)

A Feasibility Evaluation on Name-Based Routing

Haesung Hwang[1], Shingo Ata[2], and Masayuki Murata[1]

[1] Graduate School of Information Science and Technology, Osaka University
1–5 Yamadaoka, Suita, Osaka 565–0871, Japan
{h-hwang,murata}@ist.osaka-u.ac.jp
[2] Graduate School of Engineering, Osaka City University
3–3–138 Sugimoto, Sumiyoshi-ku, Osaka 558–8585, Japan
ata@info.eng.osaka-cu.ac.jp

Abstract. The IPv4 addressing scheme has been the standard for Internet communication since it was established in the 1960s. However, the enormous increase in Internet traffic usage has been leading in the past to issues such as increased complexity of routing protocols, explosion in routing table entries, provider-dependent addressing, and security problem, demonstrating the need for a redesign in advanced router technologies. The past proposals have limitations when it comes to establishing the foundations of future-generation networks, which require more sophisticated routing protocols, like content-based routing. Furthermore, those previous approaches were not conceived to fully utilize the advantages of TCAM, which is a type of memory capable of performing high-speed lookups that is already implemented in high-end routers. In this paper, we show that routing based on domain names is already a feasible technology on the Network Layer and we evaluate the necessary network and hardware resources needed to implement name-based routing strategies. We present a routing scheme and propose three methods for equally balancing the routing information in the TCAM of multiple routers. The results show that this routing scheme is scalable and that the required number of routers is two orders of magnitude smaller than the number of currently existing routers.

Keywords: Domain name routing, Future-generation network, Network/hardware resource, Routing protocol, Ternary Content Addressable Memory (TCAM).

1 Introduction

The Internet Protocol (IP) has been predominantly used for communication among heterogeneous devices in the Internet. Despite its popular usage today, problems arise because of significant differences in the traffic carried at the time it was developed and current trends toward high-volume multimedia communication. In the following, some of the major issues are listed: (i) Routing protocols are increasing in complexity and the routing table size is exploding [1], (ii) The IP address acts simultaneously as an 'ID' that distinguishes different nodes and a 'locator' that specifies the location of a node in the Internet, which makes the end node addresses depend on the Internet Service Provider (ISP), (iii) sophisticated routing schemes such as those based on names or content cannot be used. In this case, it is necessary to use an overlay or the Domain Name System (DNS), which results in extra resource consumption and propagation delay, and (iv) there are security problems such as Distributed Denial-of-Service (DDoS) attacks.

Semantic Routing [2], Content Addressable Network (CAN) [3], and Locator/ID Separation Protocol (LISP) [4] partially resolve the problems mentioned above, but most of those studies use overlay networks or routing based on agent nodes, in both cases using IP addresses as before. On the other hand, next-generation network projects, such as FIND [5], FP7 [6], and AKARI [7] aim at designing a post-IP network architecture with one of the main trends being the effort to design a network that can perform routing using names and semantic information instead of simply numbered addresses. One of the motivations for name-based routing is to eliminate the need for DNS servers. Resolving Fully Qualified Domain Names (FQDNs) to IP addresses may require queries to multiple servers which depends on the cache updating policy. Locating a specific host under frequent changes of an IP address is difficult since DNS does not premise dynamic updates. In addition, name-based routing can separate ID and locator without using an IP address and provide an unlimited address space.

The final goal of our research is to propose a network architecture that uses existing resources such as port numbers, applications, and types of information. The mechanism for finding, querying, and fetching the desired information can be performed within the routers themselves, which eliminates the process of forwarding packets to specific data servers or domain name servers. As a first step, we propose in this paper a routing scheme that achieves routing by domain names utilizing the advantage of the router's Ternary Content Addressable Memory (TCAM) [8]. Achieving name-based routing with TCAM has been considered unrealistic in the past because it was presumed that a lot of hardware resources are required. This paper is the first work to our knowledge that suggests the feasibility of name-based routing by evaluating and estimating the required TCAM resources.

The rest of this paper is organized as follows. Section 2 surveys existing studies that propose routing with names instead of the conventional IP addresses. Section 3 presents the proposed system structure and routing scheme. Section 4 presents three methods to store the domain names in the TCAM of routers. Section 5 evaluates the necessary network and hardware resources for the methods presented in Sect. 3 and Sect. 4. Section 6 concludes this paper by briefly summarizing the main points and mentioning future work.

2 Related Work

In the Name-Based Routing Protocol (NBRP) [9], the IP address acts only as a temporary routing tag and the Uniform Resource Locator (URL) is used for identifying end nodes. When a user requests a certain content, a content router (CR), which simultaneously acts as a conventional IP router as well as a name server, returns the address of the server that has the content. However, although that proposal can eliminate the multiple levels of redirection that is inherent to DNS, the necessary memory size of the CR is evaluated by DRAM, which is not optimal for high-speed packet forwarding because it can only perform exact binary matches. Partial matches of the prefix should be able to correspond to the network level of the packets in order to forward the packets rapidly to the next hop. Furthermore, names are used only while the connection is being established and not for routing the actual data packets. In [10], another name-based routing scheme is proposed. They compare the performance of their proposal with IP in terms of the creation, search, and update time of routing tables, as well as the required memory. The biggest problem with the proposal mentioned in that paper is the storage requirement, which is improved through caching and aggregating domain names. However, the work is still in its initial phase and the paper does not fully explain whether this is a feasible technology in terms of hardware resources. Moreover, the algorithm is evaluated using only a single router and does not state how the overall system should be designed nor do the authors elaborate on details of the routing method.

In this paper, we focus on evaluating the general feasibility of routing using domain names instead of IP addresses. This evaluation is done not only for the network resources, but also for the hardware resources using TCAM.

3 Routing by Domain Names

This section describes the overall structure of the proposed system and the routing scheme with a purpose of listing the preliminaries and requirements for the design of a new routing scheme.

3.1 Hierarchical Architecture

Our proposed system has a hierarchical structure in order that local information is routed in a closed network instead of being transmitted over a widespread area. In addition, it is possible to take advantage of one specific characteristic of domain names: they are already separated into different levels by dots. There are some advantages to having many levels in a hierarchical structure: each level consist of only few nodes, making the routing table of each node small. However, there are also some disadvantages such as overhead for traversing multiple levels and nodes in end-to-end communication. On the other hand, if there is only a single level, routing information of a lot of nodes have to be handled by each node, which makes the routing table of each node very large. In research on hierarchical routing for P2P applications [11,12], two-level structures are commonly considered. In addition, as the number of hierarchy levels increases the reduction in routing table size is most effective when the structure has two or three levels [13]. Considering the maintenance cost, it is inefficient to have more

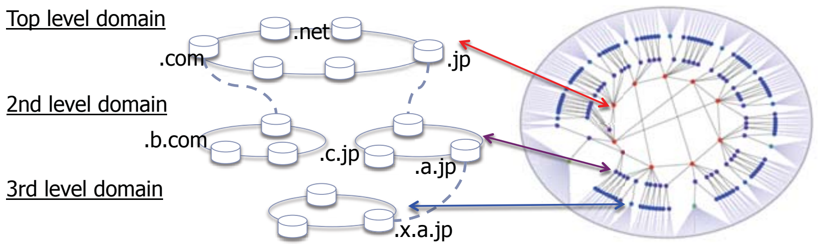


Fig. 1. Hierarchical to real topology

than three levels. Therefore, we chose the maximum number of levels of the hierarchy in the system to be three. Since the number of FQDNs in different Top Level Domains (TLDs) is not equal, not all TLDs have three underlying levels of routers. If the router in the highest level can handle all of the routing information within this TLD, then a single level is sufficient for that particular TLD.

The network topology inspired by the Abilene Network [14] and the hierarchical topology used in this paper are shown in Fig. 1. The Abilene Network is known to have similar characteristics to the real Internet topology [15]. It groups routers into three levels from the center outwards to the edge: backbone, local gateway, and edge routers. The architecture in this paper also groups routers into three levels: routers managing TLDs, routers managing 2nd-level domain names, and routers managing 3rd-level domain names. We map this hierarchical topology to the Abilene-inspired topology correspondingly. Since there are fewer than 300 kinds of TLDs, it is possible to statically configure which router manages a TLD. For local gateway and edge routers, we use dynamic configuration to write the routing information, which is explained in detail in Sect. 4.

3.2 Name Registration

Domain name routing differs from IP routing in terms of the registration because of the ID/locator separation. As IP serves as both an ID and locator, it simplifies the aggregation of the downstream traffic because IP assignment is the same in the hierarchical and in the Internet topology. However, domain names are not always location dependent which makes a separate registration process necessary. A node that intends to register its domain name injects a 'registration message' into the network and this message will eventually reach a router. The router that returns a 'destination unreachable' message because it does not have the path information is the place where the new domain name will be registered.

Each router has several TCAMs of predetermined size. In order to ensure that there is always enough room for routing table updates, a threshold value is used. After a certain threshold value is reached, the routing information must be written in a new router. A router's threshold of 0.75 means that the initial storage available for existing domain names is 75% of the router's memory capacity. The remaining 25% is reserved for routing table updates. When a new FQDN is registered, the utilization ratio of the router is checked. If it is below the threshold, the domain name is registered in that router; otherwise, there are two possibilities: split the original router's routing table in half and divide it between two routers or leave the original router in its current status and start storing new domain names in a new router. These 'new routers' are designated as candidates from the beginning and they are only used when a split table needs to be stored.

3.3 Routing Table Exchange

Within an Autonomous System (AS) [16], one or more interior gateway protocols are used. An exterior gateway protocol such as BGP-4 [17] is used to route packets between ASs. In domain name routing, an AS can be regarded as a set of routers that manage the routing information of one or more TLDs. For example, a set of routers that reside in Japan and mainly have `.jp` as the TLD can be regarded as an AS. In order for the routers to have the network reachability information, it is necessary for the routers to exchange the initial routing table that was created at the time of the domain name registration described in Sect. 3.2. In the case of IP, routers exchange information about which subnets the network can reach, such as 192.168.0.0/16, which means hosts from 192.168.0.0 to 192.168.255.255 can be reached. Similar behavior can be achieved in domain name routing by exchanging information such as `*.osaka-u.ac.jp`, which means that all domains having `osaka-u.ac.jp` as their suffix can be reached. The exchange of routing tables is achieved by extending BGP-4.

3.4 Packet Forwarding

Routers forward packets on the basis of the information contained in the routing table stored in its TCAM. An example of packet forwarding is shown in Fig. 2. When the packets are forwarded from `computer.ist.osaka-u.ac.jp` to `biz.yahoo.com`, the former sends packets to the router R1. This router's routing table consists of three parts for domain names in the upper level, the same level, and the lower level. Since the packets are destined for a higher level than router R1, they are forwarded to port P. Router R2 goes through a similar process and sends the packets to port Q. Since router R3 is in the highest level of the hierarchy, the packets are sent to port X to reach the router that has the more specific address of the `.com` suffix. Router R4 refers to its routing table to send the packets to port A and the packets finally reach R5, which has the direct routing information for the destined domain name. The process is also done by extending BGP-4, using character information instead of the IP address.

4 Balanced Distribution of Domain Names in Routers

Compared with the IP routing tables, domain name routing tables are larger because of the variable length of domain names. Therefore, it is important to balance the routing information among multiple routers. In this section, we propose three methods for equally distributing the routing information.

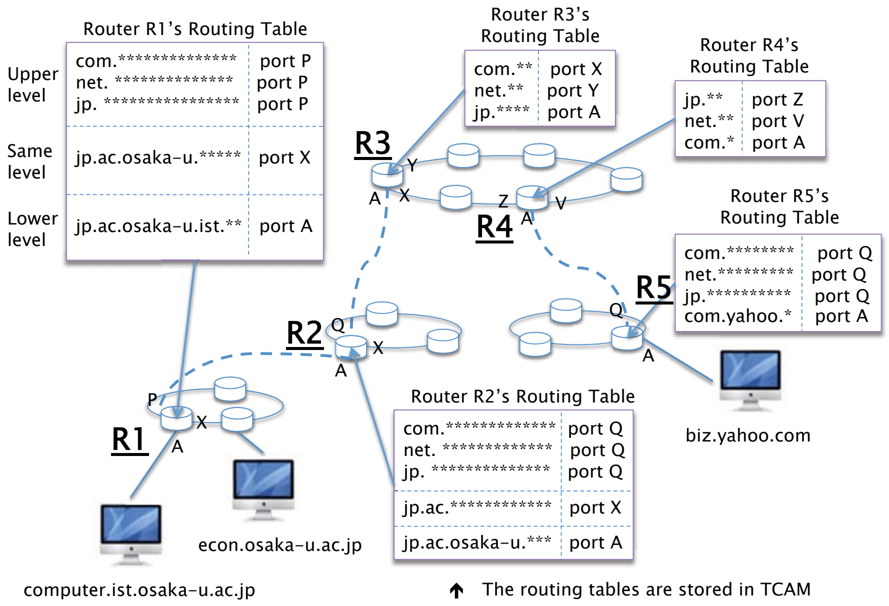

Router R1's Routing Table
Upper level
com.************** port P
net. ************** port P
jp. **************** port P
Same level
jp.ac.osaka-u.***** port X
Lower level
jp.ac.osaka-u.ist.** port A
Router R3's Routing Table
com.** port X
net.** port Y
jp.**** port A
Router R4's Routing Table
jp.** port Z
net.** port V
com.* port A
Router R5's Routing Table
com.******** port Q
net.********* port Q
jp.********** port Q
com.yahoo.* port A
Router R2's Routing Table
com.************* port Q
net. ************* port Q
jp. ************** port Q
jp.ac.************ port X
jp.ac.osaka-u.*** port A
R1
R2
R3
R4
R5
P
A
X
Q
Y
Z
V
computer.ist.osaka-u.ac.jp
econ.osaka-u.ac.jp
biz.yahoo.com
The routing tables are stored in TCAM

Fig. 2. Packet forwarding

4.1 Hash-Based Distribution

The simplest of the three proposed distribution methods is hash-based distribution. This has the best performance in terms of balanced distribution of the routing table. An FQDN is made 'flat' by hashing it with SHA-1 where 'flat' means that it is no longer hierarchical. Since every FQDN is considered to have equal status, we can obtain the total number of routers required for domain name routing by multiplying the number of domain names by the number of bits needed per entry and then dividing the result by the TCAM size in a router. The advantage of this method is that the characteristics of SHA-1 allow a large database to be equally distributed into a given number of groups. However, domain names that end with country code TLDs (ccTLDs) lose locality information after hashing. In addition, when the nodes' physical locations have no relation to their IDs, it may lead to the problem of *long stretch* which is defined in [18] as the ratio of the number of physical hops taken by the protocol to reach the destination node from the source node and their shortest distance in terms of physical hops. Therefore, although it is considered ideal in terms of achieving a nearly equal distribution, but it is not realistic for implementation.

4.2 Hierarchical Longest Alphabet Matching

In order to store domain names in memory, hierarchical longest alphabet matching represents domain names in ASCII code. The characters used in domain names are 26 case insensitive letters of the English alphabet and the 10 numbers from 0 to 9 plus hyphens and dots [19]. Inspired by the longest prefix matching scheme, longest alphabet matching is applicable when the domain names are represented in binary format. In addition to the good searching speed, aggregation of multiple domain names with the same suffix is possible, which reduces the occupied memory space. Hierarchical longest alphabet matching takes full advantage of TCAM, which can represent a don't care value '`*`' in each cell in addition to `0` or `1`. To calculate the total number of routers required, we first compute the number of routers required in each TLD. The pseudocode for this calculation is shown in Alg. 1.

Algorithm 1. Numbers of routers for Hierarchical longest alphabet matching

```
$n_{L2} \leftarrow$ number of routers in 2nd level
$n_{L3} \leftarrow$ number of routers in 3rd level
$t \leftarrow$ threshold of TCAM utilization
$entry_l \leftarrow 180$ (bits per one entry)
$router_c \leftarrow 18 \times 10^6 \times 10 \times t$
$u \leftarrow$ entries with unique 2nd-level domain names
$e \leftarrow$ entries in the TLD
**if** e $\times$ $entry_l \leq router_c$ **then**
   **RETURN** $n_{L2} \leftarrow 1$, $n_{L3} \leftarrow 0$
**else if** u $\times$ $entry_l > router_c$ **then**
   divide $u$ into groups (dynamic configuration using 2nd-level domain name)
   **RETURN** $n_{L2} \leftarrow$ number of groups
   **for each** group **Func_Calculate** $n_{L3}$
**else**
   $n_{L2} \leftarrow 1$, **Func_Calculate** $n_{L3}$
**end if**
**Func_Calculate** $n_{L3}$
divide $e$ into groups (dynamic configuration using 3rd-level domain name)
**RETURN** $n_{L3} \leftarrow$ number of groups
**END**
```

For each TLD, the first step is to determine whether all e domain name entries would fit into a router. Each entry is multiplied by 180 bits ($entry_l$) and divided by the router's available TCAM size ($router_c$). We assume that one entry consumes 180 bits in order to make comparisons with the result in Sect. 4.1, where 160 bits were the hashed value plus 20 bits of output port plus 4 parity bits. Again we define a router as having 10 TCAMs, each TCAM having a capacity of 18 Mbits. Threshold t is explained in Sect. 3.2.

If e multiplied by $entry_l$ is less than $router_c$, then the total number of routers required in this TLD is set to 1, and the process ends. Otherwise, u unique 2nd-level domain names are written in the 2nd level of the hierarchy in the form of `TLD.uniq.*`. The routers in the highest level are written with TLD information, which we ignore for the moment. Here, u domain names are written dynamically by referring to ASCII table. The basic idea is that names are grouped in alphabetical order, and when a router overflows, it starts storing the domain names in a new router. An additional idea is to take advantage of the prefix values. Names are first divided into two groups: digits and letters. If the size is still too large, the letters are divided again into smaller groups, a group of `110****` and a group of `111****`. This process is repeated until every router is well within the *router*$_c$. One example combination of the grouping is, hyphen and digits (`01*****`), a to c (`11000**`), d to g (`11001**`), h to o (`1101***`), p to q (`111000*`), r to s (`111001*`), t to w (`11101**`), and x to z (`1111***`).

When the partitioning of the 2nd-level domain names is over, the grouping is done with 3rd-level domain names. However, since the maximum number of levels in the hierarchy considered in this work is three, routers are written with FQDNs instead of with the unique 3rd-level domain names. The number of groups in each level corresponds to the number of routers. Therefore, the total number of routers required is $n_{L2} + n_{L3}$.

4.3 Hybrid Distribution

In this section, TLDs are distinguished by 9 bits because there are fewer than 300 TLDs. The hash function is applied to 2nd and 3rd-level domain names separately. To calculate the total number of required routers, we first compute the number of routers required in each TLD. The pseudocode is shown in Alg. 2.

Algorithm 2. Number of routers for Hybrid distribution

```
(Refer to Alg. 1 for the variables)
if e × $entry_l \le router_c$ then
    RETURN $n_{L2} \leftarrow 1$, $n_{L3} \leftarrow 0$
else if u × $entry_l \le router_c$ then
    RETURN $n_{L2} \leftarrow 1$
    RETURN $n_{L3} \leftarrow (e \times entry_l) / router_c$
else
    $n_{L2} \leftarrow (u \times entry_l) / router_c$
    hash(2nd level domain name) % $n_{L2}$ (divide routers in 2nd-level into groups)

    for each group calculate $n_{L3}$
    RETURN $n_{L3} \leftarrow (e_{group} \times entry_l) / router_c$
end if
END
```

The process is the same as Alg. 1 until the list of u does not fit into a single router. We obtain n_{L2} routers required in the 2nd level in the hierarchy by multiplying u by $entry_l$ and dividing it into $router_c$. In other words, the number of groups based on the 2nd-level domain names is equal to hashing 2nd-level domain names modulo n_{L2} to the most significant character. Then we obtain groups that have nearly equal entries independent of the alphabets. The next step is to calculate n_{L3} routers required in the 3rd-level; these routers are located under the corresponding routers in the 2nd level of the hierarchy. The router in which each FQDN is stored depends on the hash value of the FQDN's 2nd-level domain names. To simplify the calculation of n_{L3}, each entry is multiplied by $entry_l$ and divided by $router_c$. This was done assuming that hashing has the characteristic of equally distributing a large amount of data among groups.

5 Evaluation and Discussion

We evaluate the network and hardware resources required for domain name routing with entries obtained from the ISC database in July 2008 [20].

5.1 Network Resources

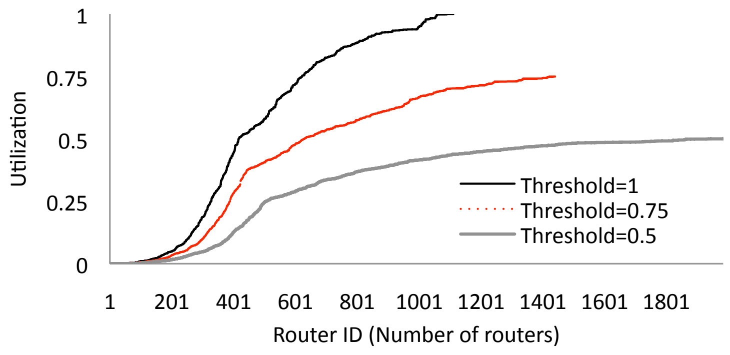
Figure 3 shows the distribution of domain name entries in routers, arranged in ascending order of the routers' utilization ratio. The utilization is defined as the number of entries in a router multiplied by 180 bits divided by the router's memory size. The graph shows the results for hierarchical longest alphabet matching and hybrid distribution for threshold values of 1, 0.75, and 0.5. The total number of required routers is smallest when the threshold is 1, letting the expected number of routing hops to be minimum. However, since there is no space left for updating routing tables, setting the threshold to 1 is not a realistic choice.

For each threshold, the required number of routers is 29.1% smaller on average for hybrid distribution than for hierarchical longest alphabet matching. In addition, hybrid distribution is likely to make a better use of the memory space because there are more routers with a utilization ratio close to the threshold.

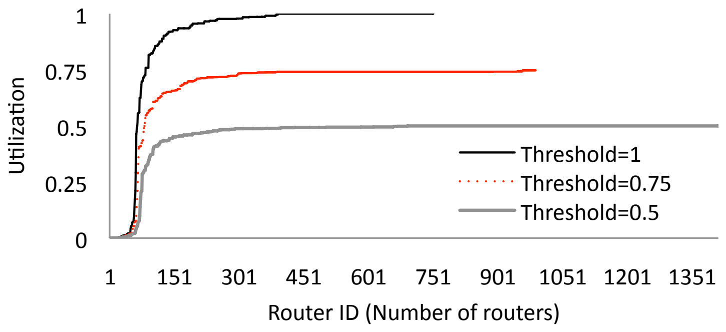
The performance of both methods can be improved by aggregating those entries in the routers that have a low utilization ratio. Since both algorithms assign at least one router for a single TLD, approximately 150 TLDs out of 270 TLDs have a utilization ratio of less than 0.1%. Those entries are collected and stored in only a small number of routers. That reduces the total number of routers required to an average of 14.2% and 16.5% for hierarchical longest alphabet matching and hybrid distribution, respectively. Therefore, statically assigning routers in backbone routers, as proposed in Sect. 3.1, needs careful consideration of the growth rate of domain names in each country.

Comparing these two algorithms, we can see that hybrid distribution needs less routers, but hierarchical longest alphabet matching utilizes the advantage of TCAM better. If TCAMs can handle Distributed Hash Table (DHT) data, then hybrid distribution can also be considered to be a feasible method.

The number of routers required for different thresholds in routers are shown in Fig. 4. Depending on the update rate of the routing table, it is possible to estimate how the required number of routers will increase. The result shows



(a) Hierarchical longest alphabet matching



(b) Hybrid distribution

Fig. 3. Distribution of domain name entries in routers for different thresholds

that the initial storage of existing domain names with a threshold value of 0.2 would require approximately 4,000 routers for both hierarchical longest alphabet matching and hybrid distribution.

5.2 Hardware Resources

When hash-based distribution is used, the average number of entries among 4,096 routers is 145,976. We obtain the memory size of 26 Mbits by 180 bits (SHA-1 output (160 bits) + output port (16 bits) + parity bits (4 bits)) $\times$ 145,976. Thus, having two 18-Mbit TCAMs is sufficient when there are 4,096 routers. In other words, if there are ten 18-Mbit TCAMs in a router, a total of 597 routers ((145,796 entries $\times$ 180 bits $\times$ 4,096 routers) / (18 $\times$ 10^6 bits $\times$ 10 TCAMs)) are required to store the existing domain names.

The number of routers required in hierarchical longest alphabet matching is 1,396, assuming that one entry consumes 180 bits. 7-bit ASCII code is sufficient to distinguish the characters used in domain names. Furthermore, since there are fewer than 300 TLDs, they can be differentiated using 9 bits. Whereas it is easy to adjust the bit length in each entry in hash-based distribution because each

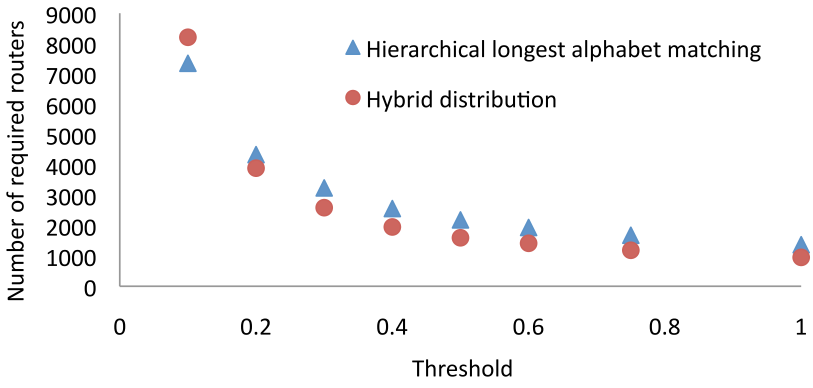


Fig. 4. Number of necessary routers required for different thresholds

entry is hashed, this is not the case in hierarchical longest alphabet matching. To satisfy the static length (180 bits) that we used to evaluate the required number of routers throughout this paper, 171 bits must be free for use, excluding the 9 bits reserved for the TLD. When one character uses 7 bits, approximately 25 characters can be written which only consists of 30% of the FQDNs [20]. To store 99% of the FQDNs, the TCAM must be able to store entries having up to 50 characters. Although the search speed decreases when the lookup size increases [8], designing the TCAM to suit longer bit lengths does not impose much of a technological difficulty.

The number of routers required for hybrid distribution is 952. Hierarchical longest alphabet matching and hybrid distribution require 2.4 and 1.7 times as many routers as hash-based distribution, respectively. However, they both use hierarchical structures, which have the advantage of restricting the local information to be routed in a closed network instead of causing it to be forwarded over a widespread area. Although hierarchical longest alphabet matching cannot achieve a nearly equal distribution of domain names among the routers, because it does not use a hashing function, it should reduce the burden of applying hashing to every entry.

In [21,22], the authors use the work from [23] and estimate that the number of routers deployed world-wide is approximately 228,260. Therefore, the router numbers required for the methods proposed in this paper are realistic values that indicate that a name-based routing architecture is feasible.

6 Conclusion and Future Work

In this paper, we used the domain name in routing as a substitute for the conventional IP address. Through statistical evaluations of currently existing domain names we showed that the proposed method is feasible in the Network Layer by estimating the required network and hardware resources. Future work involves evaluating the routing table updates by observing the evolution of the registered domain name entries over time. This includes estimating the rate of increase of entries as well as that of the number of newly added names. In addition, the effect of eliminating the current domain name servers should be investigated.

Acknowledgements. This research was supported by National Institute of Information and Communications Technology (NICT) of Japan.

References

1. BGP Reports, `http://bgp.potaroo.net/`
2. Inoue, K., Akashi, D., Koibuchi, M., Kawashima, H., Nishi, H.: Semantic Router using Data Stream to Enrich Services. In: 3rd International Conference on Future Internet Technologies (2008)
3. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Schenker, S.: A Scalable Content-Addressable Network. In: ACM Special Interest Group on Data Communication, pp. 161–172 (2001)
4. Farinacci, D., Fuller, V., Oran, D.: Locator/ID Separation Protocol (LISP). IETF Network Working Group, work in progress (2008), `http://tools.ietf.org/html/draft-farinacci-lisp-10`
5. FIND (Future InterNet Design), `http://www.nets-find.net/`
6. FP7 (Seventh Framework Programme), `http://cordis.europa.eu/fp7/home_en.html`
7. AKARI: Architecture Design Project that Illuminates the Path to the New Generation Network, `http://akari-project.nict.go.jp`
8. Renesas Technology Corporation: Network Address Search Engine (9 M/18 M-bit Full Ternary CAM), `http://documentation.renesas.com/eng/products/others/rej03h0001_r8a20211bg.pdf`
9. Gritter, M., Cheriton, D.R.: An Architecture for Content Routing Support in the Internet. In: 3rd USENIX Symposium on Internet Technologies and Systems, pp. 37–48 (2001)
10. Shue, C.A., Gupta, M.: Packet Forwarding: Name-based Vs. Prefix-based. In: 10th IEEE Global Internet Symposium, pp. 73–78 (2007)
11. Garcés-Erice, L., Biersack, E.W., Felber, P.A., Ross, K.W., Urvoy-keller, G.: Hierarchical Peer-to-Peer Systems. In: Euro-Par., pp. 643–657 (2003)
12. Kersch, P., Szabo, R., Kis, Z.L., Erdei, M., Kovács, B.: Self Organizing Ambient Control Space: An Ambient Network Architecture for Dynamic Network Interconnection. In: 1st ACM workshop on Dynamic Interconnection of Networks, pp. 17–21 (2005)
13. Lian, J., Naik, S., Agnew, G.B.: Optimal Solution of Total Routing Table Size for Hierarchical Networks. In: 9th IEEE Symposium on Computers and Communications, pp. 834–839 (2004)
14. Abilene Network, `http://www.internet2.edu/network/`
15. Li, L., Alderson, D., Willinger, W., Doyle, J.: A First-Principles Approach to Understanding the Internet's Router-level Topology. In: ACM Special Interest Group on Data Communication, pp. 3–14 (2004)
16. Hawkinson, J., Bates, T.: RFC 1930: Guidelines for Creation, Selection, and Registration of an Autonomous System, AS (1996)
17. Rekhter, Y., Li, T.: RFC 1771: A Border Gateway Protocol 4 (BGP-4) (1995)

18. Lu, G.H., Jain, S., Chen, S., Zhang, Z.L.: Virtual Id Routing: A Scalable Routing Framework with Support for Mobility and Routing Efficiency. In: ACM International Workshop on Mobility in the Evolving Internet Architecture, pp. 79–84 (2008)
19. Mockapetris, P.: RFC 1035: Domain names - Implementation and Specification (1987)
20. Internet Systems Consortium, `https://www.isc.org`
21. Yook, S.H., Jeong, H., Barabási, A.L.: Modeling the Internet's Large-scale Topology. In: Proceedings of the National Academy of Sciences of the United States of America, pp. 13382–13386 (2002)
22. Lakhina, A., Byers, J.W., Crovella, M., Matta, I.: On the Geographic Location of Internet Resources. IEEE Journal on Selected Areas in Communications 21(6), 934–948 (2003)
23. Govindan, R., Tangmunarunkit, H.: Heuristics for Internet Map Discovery. In: 19th IEEE INFOCOM (2000)

NCClient: A Python Library for NETCONF Client Applications

Shikhar Bhushan, Ha Manh Tran, and Jürgen Schönwälder

Computer Science, Jacobs University Bremen, Germany
{s.bhushan,h.tran,j.schoenwaelder}@jacobs-university.de

Abstract. The NETCONF protocol provides sound mechanisms for configuring network devices. While support for the protocol has been implemented by several network device vendors, there is a lack of supporting tools and libraries for NETCONF client applications. This paper presents NCClient, an open source Python library providing features and facilities for scripting and application development. The architecture, design and interoperability testing of the library is described and some examples of its usage are provided.

Keywords: configuration management, NETCONF, Python.

1 Introduction

The Network Configuration (NETCONF) protocol [1] is a relatively recent protocol standardized by the Internet Engineering Task Force (IETF) for network configuration management, an area that has been dominated by proprietary command line interfaces. Automation has often taken the form of ad-hoc scripting using tools like the Unix `expect` utility [2]. Since this "screen-scraping" approach has proven to be unreliable and unmaintainable [3], NETCONF has been designed from the ground-up to support robust programmatic use [4].

With this "programmatic use" motivation in mind, we developed NCClient, an API for robust application development around the NETCONF protocol. NCClient utilizes Python [5] programming language features to make programming NETCONF applications a natural and intuitive task. NCClient is an extensible implementation, in that new transport mappings and capabilities can be added easily. Error conditions are well-defined to ensure robust interaction. Features worth mentioning here are request pipelining, asynchronous RPC requests, Python context managers for sessions and locks, and full support for the operations and capabilities set out in RFC 4741 [1].

In order to make the paper concise and precise, we refer to NETCONF operations such as `get-config` by typesetting the operation name in teletype font. The names of Python classes and methods are typeset in italics. The rest of the paper is structured as follows: In Section 2, we present an overview of the NETCONF protocol. Section 3 describes the structure of the NCClient library with regard to its architecture and design details. Our interoperability testing efforts of the library are summarized in Section 4. Section 5 provides usage examples of the library and describes how the library can be extended. We discuss related work in Section 6 before concluding in Section 7.

2 NETCONF Protocol Overview

The NETCONF protocol [1] uses a simple remote procedure call (RPC) layer running over secure transports to facilitate communication between a client and a server. The Secure Shell (SSH) [6] is the mandatory secure transport that all NETCONF clients and servers are required to implement as a means of promoting interoperability [7].

The NETCONF protocol has a layered architecture as shown in Figure 1. The transport layer provides a secure communication path between the client and server. The RPC layer provides a mechanism for encoding RPCs and the operations layer residing on top of the RPC layer comprises the operations invoked as RPC methods with XML-encoded parameters to manipulate configuration state. The content layer relates to the data-model specific entities like configuration data.

The base NETCONF specification [1] specifies several operations for manipulating configuration datastores. The set of basic protocol operations consists of `get-config`, `edit-config`, `copy-config`, `delete-config`, `lock`, `unlock`, `get`, `close-session`, `kill-session`, `discard-changes`, `validate` and `commit`. Multiple configuration datastores are supported. A configuration datastore is defined as the set of configuration objects required to get a device from its initial default state into a desired operational state. The `running` datastore is present in the base model and provides the currently active configuration. In addition, NETCONF supports a `candidate` datastore, which is a buffer that can be manipulated and later committed to the `running` datastore, and a `startup` configuration datastore, which is loaded by the device as part of initialization when it reboots or reloads.

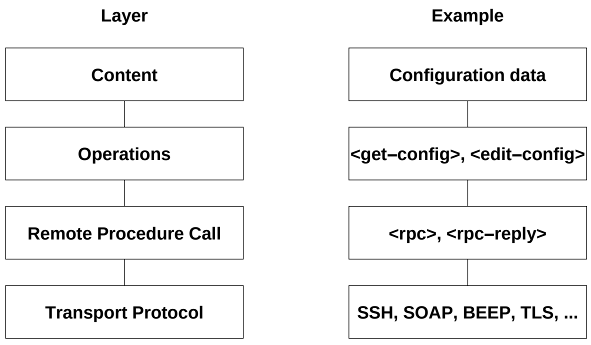


Fig. 1. NETCONF protocol layers [1]

NETCONF protocol has the notion of capabilities. A capability is some functionality that supplements the base NETCONF specification and is not required to be supported by all implementations and devices. Capabilities are identified by a uniform resource identifier (URI). NETCONF peers exchange device capabilities when a session is initiated: When the NETCONF session is opened, each peer (both client and server) must send a `hello` message containing a list of that peer's capabilities. Following RFC 4741, we denote capabilities by the capability name prefixed with a colon, omitting the rest of the URI.

3 Structure of the NCClient Library

The NCClient library closely follows the layered architecture of NETCONF. The transport layer corresponds to NCClient's *transport* module. The RPC and operations layers both map to the *operations* module. The base NETCONF specification [1] deems the actual content being exchanged as out of its scope and so have we. The NCClient API presently treats configuration data as an opaque entity. Facilities for the content layer can be implemented on top of the NCClient API.

We have initially implemented NETCONF over Secure Shell (SSH) [7], since this is the mandatory transport implementation. All operations and capabilities specified in the base NETCONF specification are also implemented, as shown in Table 1. Table 2 indicates the operation interface exposed by the high-level *Manager* API (described below).

There exists a separate specification for NETCONF event notifications [8], specifying the `:notification` and `:interleave` capabilities. While support for the `:notification` capability is planned, it can be asserted that the `:interleave` capability is already supported in principle, since the design allows for `rpc-reply` and `notification` messages to be interleaved.

The following is a summary of the main modules contained in the NCClient library. Not described below is the *xml_* module, which provides utility functions for XML serialization and parsing, and the *capabilities* module, which has

Table 1. NETCONF capabilities supported by NCClient

Capability	Supported
`:writable-running`	✓
`:candidate`	✓
`:confirmed-commit`	✓
`:rollback-on-error`	✓
`:startup`	✓
`:url`	✓
`:validate`	✓
`:xpath`	✓
`:notification` [8]	
`:interleave` [8]	

a container class that eases refering to standard capability URI's with abbreviated names, e.g., `:candidate`, or `:candidate:1.0` in case version information is significant.

3.1 Transport Module

The transport module implements the SSH transport for NCClient. Concrete transport implementations derive from the *Session* base class, which is itself a *Thread* class. Threading has been utilized to support request pipelining, asynchronous RPC requests, and asynchronous protocol messages such as notifications. The implementation of a *Session* thread's main loop is left to subclasses.

Inter-thread communication takes the form of messages to be sent and messages that have been received and are to be dispatched. Messages to be sent are put in a thread-safe queue data structure, which is meant to be consumed by a concrete implementation's main loop. Received messages are dispatched to attached *SessionListener* instances. The *Session* class thus acts as a subject for passive *SessionListener* instances, which provide an interface for *callback* and an *errback* function (called in case of transport error).

The *SSHSession* class uses the *paramiko* SSHv2 library [9]. It inherits the functionality of *paramiko* such as reading typical SSH configuration files, support of publickey, password, and keyboard-interactive authentication methods, and authenticating using *ssh-agent*. Unknown host key verification is the basis for security in SSH and required special thought. The *connect()* function accepts an optional callback which is invoked with the host key fingerprint when it cannot be verified. This allows the decision to rest with the API user.

The `netconf` SSH subsystem is invoked after connection establishment and control is given to a base class method which handles the `hello` message exchange. The method initializes the session with the server's capabilities and assigned `session-id`. RFC 4742 stipulates a character sequence that is sent

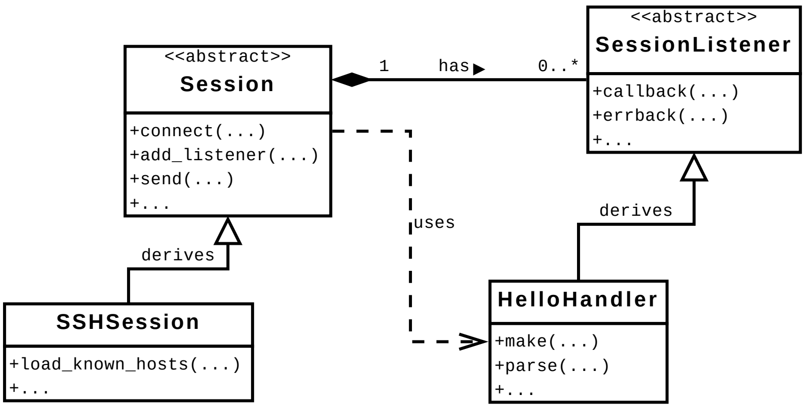


Fig. 2. Transport module represented as a UML diagram

after every XML document in the NETCONF exchange. *SSHSession* uses this as a basis for delimiting messages, and tries to parse this sequence efficiently.

3.2 Operation Module

All operations derive from the *RPC* base class, which handles making the request and taking delivery of the reply. An *RPC* object is created with a *Session* instance. *RPC* objects register with *RPCReplyListener*, which maintains a `message-id` to *RPC* mapping. *RPCReplyListener* is a so-called multiton, in that there may only be one instance per *Session*.

A common interface for requesting the operation is provided, although parameters can vary. The request is built as an XML document and sent over the session. In case the RPC was requested as asynchronous, it returns immediately; otherwise, the function blocks till the response is received or an error event takes place.

When *RPCReplyListener* encounters a `rpc-reply`, it is delivered to the correct `RPC` object based on the `message-id` attribute. All RPC objects registered with it are also notified in case of an error event in the transport layer, so that they do not block forever waiting for a reply.

An *RPCReply* instance represents a `rpc-reply` message, and only concerns itself with the success or failure of the RPC. In case of parsed `rpc-error` elements, *RPCError* objects that capture all the error information are retained. *GetReply* is the reply class associated with the *Get* and *GetConfig* operations that additionally maintains a parsed `data` element, since this is of primary interest in case of these operations.

It is ensured that *RPCReply* parsing does not take place in the same context as the *RPCReplyListener*'s callback, so that the session thread is not impacted. Operations can also specify their dependencies on capabilities, which are verified

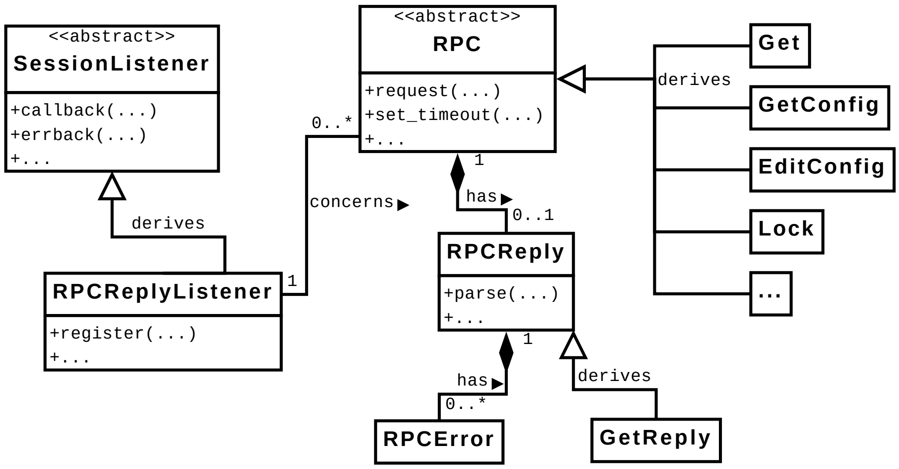


Fig. 3. Operations module represented as a UML diagram

Table 2. High-level API for NETCONF operations

API	Parameters
get_config()	*source[, filter]*
edit_config()	*target, config[, default_operation, test_option, error_option]*
copy_config()	*source, target*
delete_config()	*target*
lock()	*target*
unlock()	*target*
get()	*[filter]*
close_session()	
kill_session()	*session_id*
discard_changes()	
validate()	*source*
commit()	*[confirmed, timeout]*

during object initialization, or at request time in case the dependency only arises from an optional, user-supplied parameter.

3.3 Manager Module

The manager module provides a façade for the low-level API. It is a thin layer of abstraction that exposes all of the features of the library. A *Manager* instance is created by a factory function that corresponds to session creation. It offers an API of function calls for operations, as shown in Table 2. Function calls and parameters correspond to operations and arguments defined in RFC 4741. Optional parameters are indicated within square brackets.

4 Interoperability Testing

Interoperability has been tested with Cisco 1802 routers running IOS version 12.4(19.18)T, Tail-f ConfD version 2.8.1, and Netopeer version 1.0. The ConfD software produced by Tail-f is an extensible development toolkit that allows developers to add new components by writing a configuration specification for a data model and loading the generated object and schema files for the components. The Netopeer software [10] is an open-source implementation of the NETCONF protocol used for configuring the IP traffic flow monitoring probe (FlowMon). ConfD and Netopeer were installed and configured to run on Linux XEN virtual machines [11]. A Juniper implementation of NETCONF could not be tested due to SSH version incompatibilities.

The Tail-f and Netopeer implementations were found to comply to the NETCONF standard and thus easy to work with. However, a few problems were encountered with the Cisco implementation relating to the use of XML namespaces. The `hello` message is not namespaced, and in all other messages the base NETCONF namespace is incorrectly represented.

Table 3. NCClient test results on three NETCONF implementations

Operation	Tail-f	Cisco	Netopeer
get-config	√	√	√
edit-config	√	√	√
copy-config	√	√	√
delete-config	√	√	√
lock	√	√	√
unlock	√	√	√
get	√	√	√
close-session	√	√	√
kill-session	√	√	√
discard-changes	√		
validate	√		
commit	√		

Table 3 summarizes the result of our tests on these systems. We invoked the high-level API corresponding to each operation with appropriate parameters and verified the response. The Tail-f system supports all capabilities except the `:startup` capability, thus all the operations could be tested. The Cisco and Netopeer systems support fewer capabilities and operations relying on the `:candidate` and `:validate` capabilities could not be tested.

5 Practical Usage

In this section, we discuss some code fragments in order to illustrate the usage of the NCClient API and we explain how a new operation can be added to the library.

5.1 Sample Code Fragments

The following code examples use the high-level *Manager* API. Figure 4 shows how the `running` configuration datastore can be updated with a new configuration. The *connect()* factory function used in line 3 returns a *Manager* instance. In the absence of other arguments, it will connect on the default NETCONF port (830), verify host key, and for authentication use all mechanisms available such as *ssh-agent*. A *Manager* instance is a Python context manager, therefore the session will automatically be closed with the `close-session` operation when the context indicated by the *with* statement in line 4 finishes. Line 5 is redundant since the use of a URL as a source or target would cause the manager to verify the capability, and is present simply to indicate how a given capability can be verified. Lines 7-9 cause a lock to be acquired on the `running` datastore, and after taking a checkpoint of the current configuration, it is updated with a new configuration. If a `rpc-error` element is encountered in the response, this is raised as an *RPCError* exception by default. The lock will be automatically released in this case, due to the use of a lock context.

```
from ncclient import manager
from ncclient.operations import RPCError

with manager.connect("hostname", username="user") as m:
  assert(":url" in m.server_capabilities)
  try:
    with m.locked(target="running"):
      m.copy_config(source="running", target="file:///old.conf")
      m.copy_config(source="file:///new.conf", target="running")
  except RPCError as e:
    print(e) # stub
```

Fig. 4. Replacing `running` configuration

```
1  expected_fp = "c1:32:d6:5d:28:ed:c5:2c:8a:96:47:d8:dc:56:e3:80"
2  cb = lambda host, fp: fp == expected_fp
3  config = "<config><some-config-data-here/></config>"
4  m = manager.connect("hostname", 22, username="username",
                       key_filename="~/.ssh/id_dsa",
                       password="passphrase", unknown_host_cb=cb)
5  m.set_timeout(60)
6  try:
7    m.discard_changes()
8    with m.locked(target="candidate"):
9      m.edit_config(target="candidate", config=config,
                     test_option="test-then-set")
10     m.commit()
11     with open("local_backup", "w") as f:
12       f.write(m.get_config(source="candidate").data_xml)
13 except RPCError as error:
14   if error.severity == "error": # contrived example
15     print(error.info)
16 finally:
17   m.close_session()
```

Fig. 5. Editing `candidate` configuration

Figure 5 shows usage of the `edit-config` operation for editing the `candidate` datastore, and then committing to the running configuration. Connection is established in line 4 using publickey SSH authentication, and host key verification is accomplished through the callback argument. With line 5, a timeout of 60 seconds for synchronous RPC requests is set. After connection establishment the RPC requests are performed within the try-except-finally block. This is equivalent to using the `Manager` instance as a context manager as shown in the previous example. Before acquiring a lock in line 8, a `discard-changes` RPC is requested to ensure locking does not fail because of the `candidate` datastore being in an inconsistent state. In line 9 the `edit-config` RPC is requested specifying that the XML literal assigned in line 3 be used to update the `candidate` datastore, using the `test-then-set` option, which validates the changes before applying. Here as

```
from ncclient.operations import RPC
from xml.etree import ElementTree as ET

pc_uri = "urn:liberouter:params:xml:ns:netconf:power-control:1.0"

class RebootMachine(RPC):

    DEPENDS = [pc_uri]

    def request(self):
        spec = ET.Element("{%s}reboot-machine" % pc_uri)
        return self._request(spec)
```

Fig. 6. Example implementation of a new operation

well, capability-verification is performed automatically. Provided this operation succeeded and did not result in an exception, the `commit` operation in line 10 engages this change. Finally, the configuration is retrieved as an XML literal using the `data_xml` attribute of the `get-config` RPC result, and saved to a local file.

There is presently no explicit API support for transactional changes, however Figure 7 illustrates this using NCClient's asychronous API. The example easily scales to an arbitary number of hosts. In asynchronous mode, instead of an *RPCReply* each request returns immediately with a *RPC* object corresponding to that operation; as in lines 24-25. Error-checking needs to be done explicitly as shown in the *check()* function defined in lines 4-8: an error could arise either from the transport-layer or due to `rpc-error` elements in the `rpc-reply`.

5.2 Extension

Extensibility was a primary design consideration. The modular, decoupled architecture of NCClient eases adding transport implementations and capabilities. In addition to the default SSH transport, transport mappings are currently defined for NETCONF over BEEP [12], SOAP [13], and TLS [14]. New capabilities are being defined by the IETF and it may be desirable to support vendor-specific capabilities.

A new operation can be implemented simply by deriving from the *RPC* class and implementing the *request()* method that builds the request as an XML document based on given parameters (if applicable). The rest is handled by the base class. A custom reply class that derives from *RPCReply* can also be supplied. Figure 6 shows how a new RPC `reboot-machine` defined by the Netopeer implementation can be implemented.

A transport implementation is more involved, yet well-defined. It would derive from the *Session* class, and implement methods that take care of connection setup and teardown, and additionally the main loop for the thread that would be responsible for sending queued messages and requesting that received messages be dispatched.

```
TIMEOUT = 10
HOSTS = ["host1", "host2", "host3", "host4", "host5"] # .. hostN

def check(op):
    if op.error is not None: # e.g. transport layer error
        return op.error
    elif not op.reply.ok: # <rpc-error>(s) present
        return op.reply.error # first <rpc-error>

managers = []

try:

     for host in HOSTS:
         mgr = manager.connect(host, username="common_username")
         mgr.set_timeout(TIMEOUT)
         mgr.discard_changes()
         mgr.lock(target="candidate")
         mgr.set_async_mode(True)
         managers.append(mgr)

    ops = []
    for mgr in managers:
        copy_op = mgr.copy_config(source="ftp://config",
                                  target="candidate")
        commit_op = mgr.commit(confirmed=True)
        ops.append((copy_op, commit_op))

    for idx, host in enumerate(hosts):
        copy_op, commit_op = ops[idx]
        copy_op.event.wait(TIMEOUT)
        commit_op.event.wait(TIMEOUT)
        err = check(copy_op)
        if err is None: err = check(commit_op)
        if err is not None:
            print("Error on [%s]: ", (host, err))
            raise err
    else:
        for mgr in managers:
            mgr.set_async_mode(False)
            mgr.commit(confirmed=True)

finally:
    for mgr in managers:
        mgr.close_session();
```

Fig. 7. Transactional changes using asynchronous API

6 Related Work

ConfM is a proprietary Java library developed by Tail-f Systems that supports network-wide configuration change transactions, validates, and rollbacks. Juniper provides their customers a Perl module (Net::Netconf::Manager) that can be used to invoke operations on systems running the JunOS operating system.

EnSuite [15] and Netopeer [10] are open-source network management systems based on NETCONF. EnSuite presents an extensible implementation on the server side, and a web-based management interface called YencaP Manager is provided. EnSuite appears to not have been actively developed beyond the prototype stage. Netopeer is a more recent platform-neutral NETCONF implementation, presently used for configuring an IP traffic flow monitoring tool (FlowMon Probe). It includes a command-line client application.

NCClient provides an open source Python API for the development of NETCONF client applications. A previous paper [16] examines interoperability of several NETCONF servers. We expect that NCClient will prove useful for interoperability testing and prototyping new protocol features.

7 Conclusions

We have developed NCClient, an open source Python library that provides an intuitive API for writing scripts and developing NETCONF client applications in Python. The library provides several features such as request pipelining, asynchronous RPC requests, Python context managers for sessions and locks, and full support for the operations and capabilities set out in RFC 4741. The design of NCClient closely aligns with the layers of the NETCONF protocol. The library and documentation can be downloaded at the project homepage [17].

We have performed interoperability testing of the NCClient library on various NETCONF implementations. The results show that all the operations supported according to the capabilities of the servers successfully execute on the three systems under test (Tail-f, Cisco and Netopeer). We have also described the usage of NCClient for two scenarios: replacing configuration data using the `copy_config` operation and editing configuration data using the `edit_config` operation. In addition, we have demonstrated asynchronous usage and extensibility.

NCClient has been carefully designed with the consideration of extensibility and usability for API clients. There are several possibilities for future development. More capabilities can be supported, such as `:notification`, which was not implemented due to lack of a test subject. At the time of writing, there are Internet-Drafts for capabilities defining operations that provide fine-grained locking primitives, knowledge of default configuration values, and retrieval of schema information. The library could be extended with more high-level API constructs supporting network-wide configuration management transactions. The focus so far has been on individual device configuration.

Acknowledgments

The work reported in this paper is supported by the EC IST-EMANICS Network of Excellence (#26854).

References

1. Enns, R.: NETCONF Configuration Protocol. RFC 4741 (December 2006)
2. Libes, D.: Expect: Curing Those Uncontrollable Fits of Interaction. In: Proc. Summer 1990 USENIX Conference, Anaheim, June 1990, pp. 11–15 (1990)
3. Schönwälder, J.: Overview of the 2002 IAB Network Management Workshop. RFC 3535 (May 2003)
4. Schönwälder, J., Björklund, M., Shafer, P.: Network Configuration Management using NETCONF and YANG. IEEE Communications Magazine (2009)
5. van Rossum, G., Drake, F.: The Python Language Reference Manual. Network Theory Limited (2003)
6. Ylonen, T., Lonvick, C.: The Secure Shell (SSH) Protocol Architecture. RFC 4251 (January 2006)
7. Wasserman, M., Goddard, T.: Using the NETCONF Configuration Protocol over Secure Shell (SSH). RFC 4742 (December 2006)
8. Chisholm, S., Trevino, H.: NETCONF Event Notifications. RFC 5277 (July 2008)
9. Paramiko - SSH2 Protocol for Python, `http://www.lag.net/paramiko/` (last access in May 2009)
10. Krejčí, R., Lhotka, L., Čeleda, P., Špringl, P.: Secure Remote Conguration of Network Devices - A Case Study. In: Proc. CESNET Conference 2008, Prague, Czech Republic, CESNET, September 2008, pp. 77–84 (2008)
11. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A.: Xen and the Art of Virtualization. In: Proc.19th ACM Symposium on Operating Systems Principles (SOSP 2003), October 2003. ACM, New York (2003)
12. Lear, E., Crozier, K.: Using the NETCONF Protocol over the Blocks Extensible Exchange Protocol (BEEP). RFC 4744 (December 2006)
13. Goddard, T.: Using NETCONF over the Simple Object Access Protocol (SOAP). RFC 4743 (December 2006)
14. Badra, M.: NETCONF over Transport Layer Security (TLS). RFC 5539 (May 2009)
15. Cridlig, V., Abdelnur, H.J., Bourdellon, J., State, R.: A NetConf Network Management Suite: ENSUITE. In: Magedanz, T., Madeira, E.R.M., Dini, P. (eds.) IPOM 2005. LNCS, vol. 3751, pp. 152–161. Springer, Heidelberg (2005)
16. Tran, H.M., Tumar, I., Schönwälder, J.: NETCONF Interoperability Testing. In: Proc. 3rd International Conference on Autonomous Infrastructure, Management and Security (AIMS 2009), pp. 83–94. Springer, Heidelberg (2009)
17. NCClient - Python library for NETCONF clients, `http://code.google.com/p/ncclient` (last access in July 2009)

An IPSec Mediation Approach for Safe Establishment of Inter-domain VPNs

Alexandre Matos, Fernando Matos, Paulo Simões, and Edmundo Monteiro

CISUC – Informatics Engineering Department, University of Coimbra
Coimbra, Portugal
{aveloso,fmmatos,psimoes,edmundo}@dei.uc.pt

Abstract. In this paper we propose a new solution to increase the security of BGP/MPLS IP VPNs established across multiple domains. In general, layer 3 VPNs already present a number of security risks when used in single domain scenarios, since they are vulnerable to attacks originated inside the provider backbone. In order to overcome these risks, IPSec tunnels are recommended. In multi-domain scenarios, however, the safe establishment of such IPSec tunnels is much more difficult, due to need to set up proper Security Associations in an open environment. The solution we present in this paper not only solves this problem but also improves the dynamic composition of multi-domain VPNs, thus reducing the effort and time required to provide such VPNs.

Keywords: Inter-domain VPN, IPSec, BGP/MPLS, SOA, Business Layer.

1 Introduction

A Virtual Private Network (VPN) is a solution to restrict communications among a set of endpoints, aiming to provide confidentiality even on public networks. Internet Service Providers (ISPs) currently offer managed VPN services to a wide range of clients, as a means of market expansion and revenue generation.

BGP/MPLS IP VPN [1] is a technology through which an ISP may provide a VPN to customers through an IP backbone. While BGP (Border Gateway Protocol) is used to distribute routing information between edge devices of the provider networks (designated as Provider Edge: PE), MPLS (Multiprotocol Label Switching) is used to forward VPN traffic, based on policies and bandwidth requirements.

Basic security requirements are addressed by BGP/MPLS IP VPNs. However, the inherent lack of encryption may compromise integrity, confidentiality and authentication. In a multi-domain scenario a VPN crosses distinct Autonomous Systems (AS). It implies that border routers (ASBRs – Autonomous System Border Routers) exchange critical information such as VPN IPv4 addresses of endpoints. According to [1] this may be accomplished by three approaches. The first requires fixed connections among ASBRs, which does not scale well. The second requires a previous trust relationship between involved ASs so that labeled routes may be exchanged. The third approach is a solution for multi-hop distribution of labeled routes. This information will travel, non-encrypted, through distinct domains. The implicit risk of content inspection indirectly compromises the VPN security.

A solution recommended by IETF is to add cryptography [3-5], more specifically IP Security Protocol (IPSec). However, IPSec requires the establishment of Security Associations (SA), which may turn into a complex problem when considering multi-domain scenarios. This means that even when following IETF recommendations a challenge remains: how to securely establish IPSec SAs across multiple domains.

In a previous work [2] we presented a Business Layer (BL) approach (the Global Business Framework: GBF) for support of dynamic, on-demand multi-domain VPNs. GBF provides a way to dynamically build VPNs, based on automated negotiation of technical and business parameters between providers and collaborative management of the VPN lifecycle. This solution drastically reduces the time and cost of current procedures, based on ad-hoc, manual interactions between providers. GBF is a general business framework, in line with initiatives such as IPsphere [6] and with concepts like Next Generation Networks (NGN) and Future Internet (FI).

In this paper we propose a solution to the problem of safely providing SA for the dynamic establishment of managed VPNs in multi-domain scenarios, where a key challenge needs to be addressed: how can a reliable communication channel be established between entities unknown to each other?

The rest of this paper is organized as follows: Section 2 presents the current approaches to the dynamic establishment of SA. In Section 3 there is a discussion of how a dynamic IPSec SA can be established via a BL support. In Section 5 we evaluate the proposed framework and Section 5 concludes the paper.

2 Related Work

The authors of [7] propose an approach to distribute keys and establish SAs along the path. Security Policies Databases (SPD) and a Security Associations Database (SADB) are used for this intent. SPDs and SADB must be previously established, taking into account keys and certificates distributed by a centralized Certificate Authority (CA). To the best of our knowledge, this is the first attempt to implement IPSec features on inter-domain BGP/MPLS VPNs. Despite the contribution, this work involves intense human intervention to provide relationships among domains.

IPSec encryption is proposed by [8] to address risks associated with BGP route advertisements, such as hijacking of TCP connections, corruption of route updates, replay attacks and intruder masquerading. In this approach, BGP traffic flows through IPSec tunnels before the establishment of the BGP/MPLS VPN. According the authors, the evaluation tests did not present a negative affect at overall performance.

Li et al. [9] exchanges extended X.509 certificates carrying IPSec policies in order to achieve a common trust basis and identification of endpoints. These policies are generated according to the role of the endpoint. After each endpoint completes its certificate specification, this will be sent over to the other endpoint, which will decide whether to adopt the underlying policies or not. Associated interactions are supported by Policy Servers, and the effective establishment of an IPSec tunnel depends on the acceptance of each certificate. Other servers are present in this framework – the PKI Server and the VPN Server. SSL tunnels are used by each endpoint in order to reach these servers. Although this is an innovative solution for the establishment of SAs, it has some shortcomings from the scalability point of view, namely the growing number of SSL tunnels required to link endpoints to each server and the complexity of the insertion of a new endpoint (that requires a new extended X.509 certificate and inclusion on the VPN server).

A compliance checker that evaluates whether a packet conforms to the underlying policies of a SA is proposed in [10]. This approach guarantees that only authorized entities are allowed to exchange specific traffic. However, it does not address solutions for fine-grained security policies such as the duration of a SA. Another drawback is the fact that many applications are forced to duplicate at the service layer cryptographic functions already provided at the infrastructure layer.

Despite contributions of former works, there is in general a lack of dynamic and scalable solutions to address the management of IPSec security on inter-domain VPNs [11]. In the next Section we propose the establishment of SAs between distinct domains as a NGN-like service, based on our previously developed framework [2].

3 Proposed Approach

GBF [2] is a BL-oriented approach to handle dynamic creation of services assembled using sub-services (or Elements) from different providers (ISPs, carriers, content providers, etc.) which may assume two distinct roles: Service Owner (SO) and Element Owner (EO). A SO is the provider that is contacted by the customer to provide the (composed) service. Each EO is another provider that is “subcontracted” by the SO to provide a part of the service (Element). The discovery of potential Element providers is supported by an UDDI-based (Universal Description, Discovery and Integration) service publishing mechanism, and the description of service components (as well as the negotiation between the SO and the EOs) is based on the exchange of service templates.

There are two types of service templates: Service Specification Template (SST) and Element Specification Template (EST). While the SST describes the general parameters of the service, the EST is used to define, with EOs, more complete technical and business details of the participation in the composition of a service. These details include is its public key, with certificates signed by trusted CAs.

Communication between the SO and each EO is protected through the guidelines of WS-Security [12]. As shown in Figure 1, two components of the GBF BL are responsible for the template exchange: the Service Instance Manager (SIM) and the Element Instance Manager (EIM). SIM coordinates service provisioning and, for this reason, it determines which service parameters – including security credentials – need to be known and enforced by each EIM for the SOAP transactions.

Figure 2 presents an excerpt of the EST Template. Some of the fields of this Template (*FirewallTraversal, BGPAdvertisement, IPSecService...*) are crucial to achieve an inter-domain IPSec SA. However, those fields correspond to sensitive information that requires a reliable channel. This is accomplished using the public keys of each EO and the digital signature of the SO. While EO public keys (and underlying certificate) are gathered from EOs, the SO fills each EST template with security information. This way each EO only retrieves service templates (including sensitive information) after the SO checks the EO digital certificate associated with its public key.

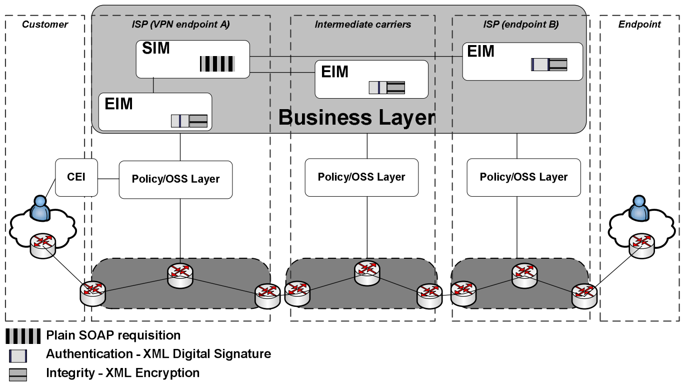


Fig. 1. WS-Security approach applied to the GBF BL

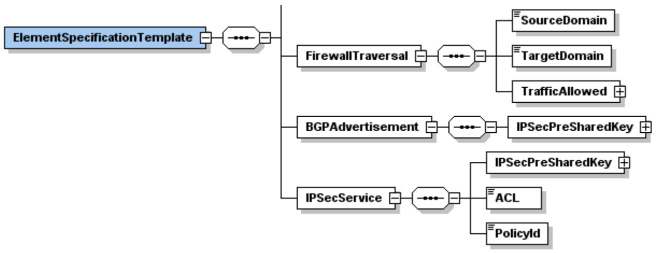


Fig. 2. Excerpt of an Element Specification Template (EST)

As discussed in [2], each EO plays a distinct role in the composition of the VPN service. One of those roles associates an EO to a transport and connection element, enabling distinct domains to be interconnected trough ASBRs. In this case, the SO must fill *BGPAdverstisement* fields with a distinct *IPSecPreSharedKey* element. This information is then used by the underlying domain to provide an IPSec tunnel to carry safe advertisement of BGP routes and updates.

In our prototype the WS-Security recommendations were implemented with the help of Apache WSS4J [13], which enables authentication and integrity checking during the EST exchange process.

In that prototype a secure tunnel was implemented within each EO domain. This "vertical" channel transports decisions from the secure BL to the underlying Network and Infrastructure Layer (NIL). A similar approach is also found in previous work [9].

4 Evaluation

In order to assess the performance penalty resulting from the proposed IPsec mediation approach, we used an experimental testbed where inter-domain VPNs are safely established between three different domains (Figure 3).

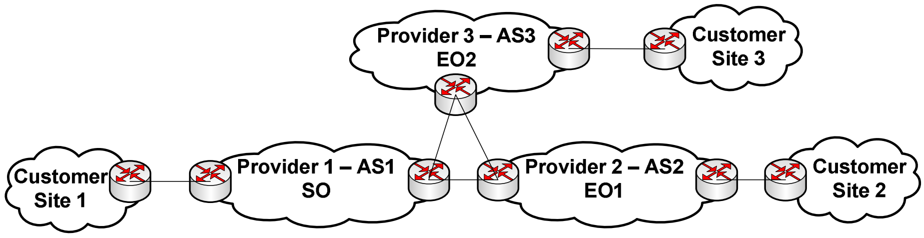


Fig. 3. Inter-domain VPN Testbed (Basic Scenario)

Previous work [2] already focused on the validation of the general GBF framework for dynamic establishment of multi-domain VPNs. We now focused on comparing traffic overhead and performance penalties induced by the GBF framework in two distinct situations: with and without the proposed IPsec mediation scheme.

Table 1 presents the traffic exchanged between the SO and each EO. There is an increase in traffic overhead, introduced by the IPsec mediation, but within acceptable limits. Performance is also slightly affected: the average VPN establishment latency raised less than 9%, from 1,200 ms (without IPsec mediation) to 1,300 ms (with IPsec mediation).

These measurements show that IPsec mediation does affect performance and network traffic but well within reasonable values, considering its security benefits.

Table 1. VPN Establishment With and Without IPSec Mediation (traffic overhead generated by the GBF framework)

Link	*VPN establishment (without IPSec)*	*VPN establishment (with IPSec)*
SO / EO 1	*47 packets (17721 octects)*	*62 packets (23376 octects)*
SO / EO 2	*47 packets (17721 octects)*	*62 packets (23376 octects)*

5 Conclusions

In this paper we presented a new mechanism to safely create inter domain BGP/MPLS IP VPNs. This mechanism was built on top the GBF, thus allowing the dynamic provisioning of on-demand VPNs.

This proposal, inline with recommendations from IETF and ITU-T, allows the safe application of IPSec tunnels to build BGP/MPLS IP VPNs, bypassing a number of intrinsic vulnerabilities. Since building trust relationships across multiple domains can become a hard task, the Business Layer of GBF was used to establish high-level statements for the service (namely the establishment of the Security Association).

The original EST template used by GBF to describe service instance parameters was extended in order to include additional information, for instance to secure BGP advertisements.

Future work will focus on trust management, more specifically in the improvement of trust mechanisms between EOs and SOs, leading to a more dynamic trust negotiation approach based on SOA.

Acknowledgments. This work was partially funded by FCT (scholarship contracts SFRH/BD/16261/2004 and SFRH/BD/29103/2006) and supported by the IST FP6 OpenNet Project.

References

1. Rosen, E., et al.: BGP/MPLS IP Virtual Private Networks (VPNs). RFC 4364 (2006)
2. Matos, A.V., Matos, F.M., Simões, P., Monteiro, E.: A Framework for the Establishment of Inter-Domain, On-Demand VPNs. In: 11th IEEE/IFIP Network Operations and Management Symposium – NOMS, pp. 232–239 (2008)
3. Behringer, M.: Analysis of the Security of BGP/MPLS IP Virtual Private Networks (VPNs). RFC 4381 (2006)
4. Rekhter, Y., Bonica, R., Rosen, E.: Use of Provider Edge to Provider Edge (PE-PE) Generic Routing Encapsulation (GRE) or IP in BGP/MPLS IP Virtual Private Networks. RFC 4797 (2007)
5. Rosen, E.: Applicability Statement for BGP/MPLS IP Virtual Private Networks (VPNs). RFC 4365 (2006)
6. Alateras, J. (ed.): IPsphere Framework Technical Specification – Release 1 (2007), http://www.ipsphereforum.org/Files/IPSF_R1_Spec.pdf
7. Ren, R., Feng, D., Ma, K.: A detailed implement and analysis of MPLS VPN based on IPSec. In: Proceedings of International Conference on Machine Learning and Cybernetics, vol. 5, pp. 2779–2783 (2004)
8. Pezeshki, J., et al.: Performance Implications of Instantiating IPsec over BGP Enabled RFC 4364 VPNs. In: IEEE Military Communications Conference - MILCOM, pp. 1–7 (2007)
9. Li, Q., Xu, M., Xu, K.: Toward A Practical Scheme for IPSec Management. In: International Conference on Information Networking - ICOIN, pp. 1–5 (2008)
10. Blaze, M., Ioannidis, J., Keromytis, A.: Trust Management for IPSec. ACM Transactions on Information and System Security 5(2), 95–188 (2002)
11. Masmoudi, K., Afifi, H.: Building identity-based security associations for provider-provisioned virtual private networks. Journal of Telecommunication Systems 39(3), 215–222 (2008)
12. Nadalin, A., Kaler, C., Monzillo, R., Hallam-Baker, P.: Web Services Security: SOAP Message Security 1.1 (WS-Security 2004). OASIS Standard Specification (2006)
13. Apache WSS4J, http://ws.apache.org/wss4j/

Optical Switching Impact on TCP Throughput Limited by TCP Buffers

Giovane C.M. Moura, Tiago Fioreze, Pieter-Tjerk de Boer, and Aiko Pras

University of Twente
Centre for Telematics and Information Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
Design and Analysis of Communications Systems (DACS)
Enschede, The Netherlands
`{g.c.m.moura,t.fioreze,p.t.deboer,a.pras}@utwente.nl`

Abstract. In this paper, we observe the performance of TCP throughput when self-management is employed to automatically move flows from the IP level to established connections at the optical level. This move can result in many packets arriving out of order at the receiver and even being discarded, since some of these packets would be transferred more quickly over an optical connection than the other packets transferred over an IP path. To the best of our knowledge, so far there is no work in the literature that evaluates the TCP throughput when flows undergo such conditions. Within this context, this paper presents an analysis of the impact of optical switching on the TCP CUBIC throughput when the throughput itself is limited by the TCP buffer sizes.

1 Introduction

The *Transmission Control Protocol* (TCP) is the main protocol used for end-to-end communications, representing approximately 90% of all traffic across the public Internet currently [1]. Given its major role, knowledge about TCP performance is important to understand the overall picture of service performance for IP networks. Due to that, its performance has been widely investigated in the networking community [2] [3], and most of the research works identify packet loss as one of the major impacting factors on TCP throughput [4].

Nowadays, higher transmission rates can be achieved by the advent of hybrid communications. One example is optical switching, in which network flows can be forwarded at the optical level instead of relying solely on traditional IP routing. Approaches currently employed to manage optical switched networks rely on human operators to: (*i*) select flows to be moved to the optical level, and (*ii*) create and release the necessary optical connections for such flows.

However, self-management approaches have been investigated with the goal to move automatically *on-the-fly* IP flows to optical connections [5] [6]. By avoiding the IP routing, it is expected that packets switched to the optical level would be delivered more quickly than the remaining others still being forwarded at the IP level. This could cause packets belonging to the same flow to arrive out of order at their destination or even being discarded, which could result in performance problems with the TCP throughput.

Based on the foregoing, the question one could pose is: *what is the impact on the TCP throughput when flows are moved on-the-fly to optical connections?* It is known that TCP performs packet reordering, but little is known about its limits under such circumstances. Thus, the goal of this paper is to present a study on the impact of this *on-the-fly* move on TCP CUBIC throughput, when the throughput itself is limited by the TCP buffer sizes of both sender and receiver.

To address this question, we conducted a series of simulations using the ns-2 network simulator [7], version 2.33. The first step was to review the literature about the different versions of TCP used. Out of many TCP flavors, we chose TCP CUBIC (version 2.1) [8] because it was specially designed for high-speed networks and it is the default version employed in recent Linux kernels. Next, we defined the simulation scenario and finally the simulations were conducted.

The rest of paper is structured as follows. Section 2 introduces the simulation setup used in our simulations. Following that, Sections 3 presents the results obtained and Section 4 describes our conclusions and proposes future work.

2 Simulation Setup

Figure 1 shows the topology used in our simulations. It consists of three routers (`r1`, `r2`, and `r3`) and two nodes (`Sender` and `Receiver`) connected by two different paths: the IP path (`r1-r2-r3`) and the optical path (`r1-r3`).

The simulation starts with the sender opening a single TCP connection with the receiver forwarding data only via the IP path. After reaching pre-defined throughput values, the router `r1` performs the transition of the TCP flow to the optical level, and starts to forward all the data to the receiver via the optical path. It is worth observing that the transition affects only one direction of the flow (`sender` → `receiver`), and that the acknowledgment packets (ACKs) in the reverse direction (`receiver` → `sender`) continue to use the IP path. For a short period of time after the switch – the *transient phase* – there will be data packets on both the IP and optical paths. After the transient phase, the simulation reaches a new phase, in which there are only data packets on the optical path and only ACKs on the IP path. The simulation finishes soon afterwards.

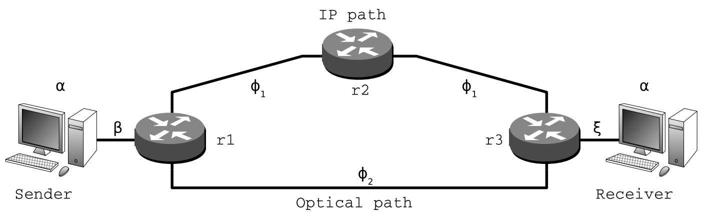


Fig. 1. Topology used in the simulations and limiting factors (Greek letters)

In order to evaluate how the TCP flow is affected by its transition from the network level to the optical level, we analyzed the trace files generated by the ns-2 to compute the throughput perceived by the receiver machine. We are interested not only in observing the maximum throughput, but also in determining *how the throughput changes* after the moment the transition is triggered.

In general, the TCP throughput can be delimited by several factors, such as the kind of application utilized, the TCP buffers sizes employed, and the link capacity [9]. In this work we focus however on the impact of moving flows *on-the-fly* on TCP throughput when flows are limited by TCP buffer sizes.

2.1 Simulation Scenario

Before executing the simulations, we had to calculate the TCP buffer sizes to be used. This value can be obtained by multiplying the flow data rate by the RTT value before the move ($flowRate \times RTT$), and indicates the maximum amount of data on the network at any given time. This value is also the minimum size that TCP buffers must have to handle the specified rates.

Due to that, we had to decide *the moment* at which a flow should be moved to the optical level ($flowRate$) and the RTT values (RTT) to be used. For the evaluated scenario, we have configured the TCP buffers to restrict flows to three data rates (100 Mbps, 1 Gbps, and 10 Gbps) and then we performed the flow transition at each one of these data rates. RTT, in turn, is usually a function of the physical distance between the sender and the receiver. In this study, we have simulated three different situations in which sender and receiver are located in hypothetical cities: (*i*) close to each other, (*ii*) relatively far from each other, and (*iii*) very far from each other. In each case, we assumed that the RTT value after the move would be smaller than before, since packets could avoid the routing process at the IP level. We then have used the RTT values described in Table 1.

After that, we could finally determine the required buffer sizes. Table 2 summarizes the values used in our simulations, where each line represents a single simulation. For each simulation, we have calculated the values for the buffer sizes (αs of Figure 1). Due to space constraints, we present in this table only the cases where RTT is equal to 10 ms, even though an equal number of simulations was conducted for 100 ms and 1000 ms RTT cases. Since we were interested in having TCP buffer sizes as the limiting factor, we over-provisioned the other limiting factors (link speed: β, ϕ, and ξ in Figure 1) to allow the network to handle all the data generated by the sender. These values are presented in Table 2.

The next section shows the results obtained from our simulations.

Table 1. RTT values employed in the simulations

Distance	**RTT before moving (ms)**	**RTT after moving (ms)**
Close	10	6
Relatively Far	100	60
Very far	1000	600

Table 2. Values used in simulations for RTT equal to 10 ms

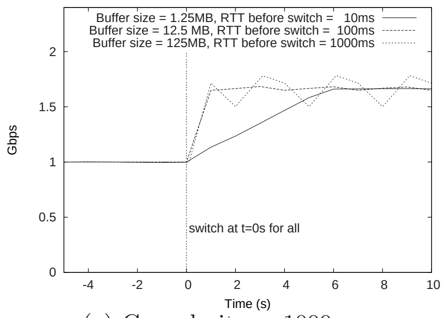
Limiting rate	**Buffer size (α) with RTT=10ms**	**Link speed ($\beta = \phi_1 = \phi_2 = \xi$)**
100 Mbps	0.125 MB	622.08 Mbps
1 Gbps	1.25 MB	2.488 Gbps
10 Gbps	12.5MB	39.813 Gbps

3 Simulation Results

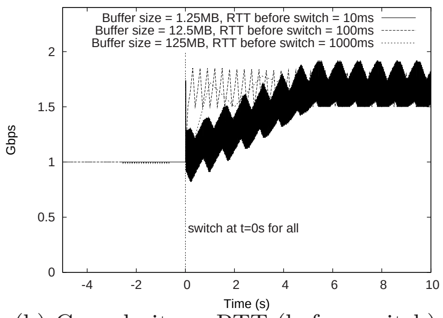
This section shows the results of our simulations when the data rate is limited by the TCP buffers. In these simulations, the flow transition was performed at the time $t = 0$s, denoted by a vertical line. One important factor when computing throughput is the granularity (*i.e.*, the time interval used to average the data rate transmitted). If the granularity is very coarse, it may mask some effects that might occur in the network. If it is too fine, it may present a too detailed view, which makes the acquisition of a general view of the throughput more complex. Figure 2 shows the results when the TCP buffers limit the data rate to 1 Gbps.

As shown in Figure 2(a), the flows present a stable rate equal to 1 Gbps before the flow transition. However, after the transition ($t = 0$s), it is possible to note that no throughput reduction is observed. In fact, we can see that CUBIC reacts very well to the new configuration, and the throughput increases quickly to the expected new theoretical value of 1,667 Gbps (obtained by dividing the buffer size by the RTT).

Figure 2(b), in turn, presents the throughput results obtained from the same simulation of Figure 2(a), but with different granularity values (equal to the RTT of each case before the switch). With these granularity values, it is possible to see that, in fact, flows experience a reduction on the throughput just after the transition, but they quickly recover from that. What happens is that many packets arrive out-of-order at the receiver, causing many duplicate ACKs to be



(a) Granularity = 1000 ms

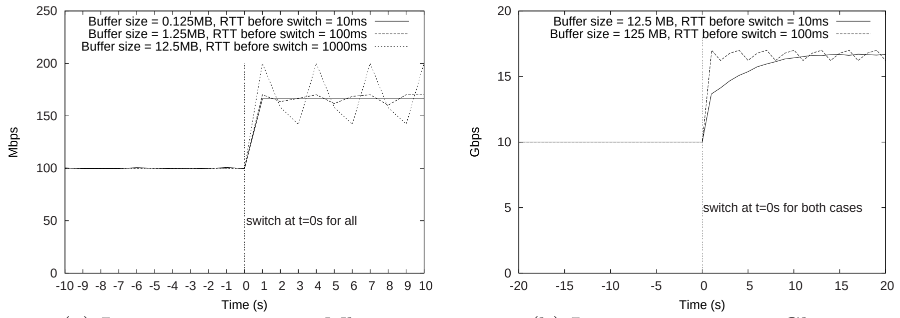


(b) Granularity = RTT (before switch)

Fig. 2. Throughput of TCP flows in Scenario A (1 Gbps limiting rate)

transmitted to the sender (for example, 1372 duplicate ACKs for the 10ms RTT case). Before this simulation, it was not clear how TCP CUBIC would reduce its congestion window (and, ultimately, cause throughput reduction) in face of many packets arriving out-of-order. This simulation however shows that TCP CUBIC performs well and recovers quickly from the reordering conditions imposed by the flow transition.

Figures 3(a) and 3(b) show the throughput when TCP buffers limit the data rate to 100 Mbps and 10 Gbps, respectively, using a granularity of 1000ms. Similar to 1 Gbps case, the throughput also quickly increases after the switch. In Figure 3(b), we have not included the throughput when the RTT is 1000ms because the TCP throughput does not reach the rate selected for comparison (10Gbps) before the transition.



(a) Limiting rate = 100 Mbps (b) Limiting rate = 10 Gbps

Fig. 3. Throughput of TCP flows in Scenario A

Despite the granularity used, we can conclude that when the buffer size is the limiting factor, the users will experience a brief reduction on the throughput followed by quick increase when a flow transition occurs.

4 Conclusions and Future Work

In this paper, we presented an analysis of the impact of moving flows *on-the-fly* on TCP throughput when TCP buffer sizes act as the limiting factor for the throughput. TCP CUBIC was the TCP flavor employed in our analysis due to its design for high-speed networks and its use in recent versions of Linux kernels. The analysis shown in this paper was performed by using the ns-2 simulator.

When the size of the TCP buffers in the sender and receiver is the limiting factor, we observed that the TCP throughput is slightly reduced just after the flow transition (*i.e.*, the transient phase). However, we observed that the throughput significantly increases when the transient phase is over, due to the smaller RTT values existing in the optical path. The smaller RTT value allowed the sender to have a higher transmission rate in our simulation. Another important observation from our simulations is that TCP CUBIC quickly reacts to new the network conditions available after the flow transition and does not suffer from the massive reordering.

As future work, we plan to evaluate the impact of moving flows *on-the-fly* to the optical level when the limiting factors for the TCP throughput are the link capacities.

Acknowledgments. This research work has been supported by the EC IST-EMANICS Network of Excellence (#26854).

References

1. Kim, M.-S., Won, Y.J., Hong, J.W.: Characteristic Analysis of Internet Traffic from the Perspective of Flows. Computer Communications 29(10), 1639–1652 (2006)
2. Grieco, L.A., Mascolo, S.: Performance evaluation and comparison of Westwood+, New Reno, and Vegas TCP congestion control. SIGCOMM Comput. Commun. Rev. 34(2), 25–38 (2004)
3. Koutsonikolas, D., Dyaberi, J.M., Garimella, P., Fahmy, S., Hu, Y.C.: On TCP throughput and window size in a multihop wireless network testbed. In: WINTECH, pp. 51–58 (2007)
4. Chandran, K., Raghunathan, S., Venkatesan, S., Prakash, R.: A feedback-based scheme for improving TCP performance in ad hoc wireless networks. IEEE Personal Communications 8(1), 34–39 (2001)
5. Fioreze, T., Pras, A.: Self-management of lambda-connections in optical networks. In: Bandara, A.K., Burgess, M. (eds.) AIMS 2007. LNCS, vol. 4543, pp. 212–215. Springer, Heidelberg (2007)
6. Fioreze, T., van de Meent, R., Pras, A.: An architecture for the self-management of lambda-connections in hybrid networks. In: Pras, A., van Sinderen, M. (eds.) EUNICE 2007. LNCS, vol. 4606, pp. 141–148. Springer, Heidelberg (2007)
7. ns2. The Network Simulator NS-2, `http://www.isi.edu/nsnam/ns/`
8. Ha, S., Rhee, I., Xu, L.: CUBIC: a new TCP-friendly high-speed TCP variant. SIGOPS Oper. Syst. Rev. 42(5), 64–74 (2008)
9. Timmer, M., de Boer, P.-T., Pras, A.: How to identify the speed limiting factor of a tcp flow. In: Proceedings, 4th IEEE/IFIP Workshop on End-to-End Monitoring Techniques and Services, April 2006, pp. 17–24 (2006)

Security Considerations for Intrinsic Monitoring within IPv6 Networks

Lei Shi and Alan Davy*

Telecommunication Software&Systems Group,
Waterford Institute of Technology, Ireland
{lshi,adavy}@tssg.org

Abstract. Intrinsic Monitoring is a method of collecting and disseminating node specific monitoring data throughout an IPv6 network by using the IPv6 extension headers as a carrier medium. The advantages of such a monitoring mechanism can be invaluable to a network operator, offering a wide range of performance and accuracy enhancements over traditional SNMP based or active probing based approaches. This paper discusses previous proposals related to Intrinsic Monitoring and highlights a number of security considerations that must first be resolved for such an approach to be deployable within an operational IP network. The paper offers initial contributions towards addressing these challenges.

1 Introduction

Nowadays network monitoring systems are crucial to communication networks. They periodically collect network performance metric values, identify performance anomalies, and determine root causes for the problems. Their effectiveness and efficiency determine the quality of network services. The most important performance metrics include connectivity, delay, packet loss rate, location of congested network nodes, and bandwidth information.

Intrinsic Monitoring is a method of collecting and disseminating node specific monitoring data throughout an IPv6 network using the IPv6 extension headers as a carrier medium. The advantages of such an approach can be invaluable to a network operator, offering a wide range of performance and accuracy enhancements over traditional Simple Network Management Protocol (SNMP) and IP Flow Information Export (IPFIX) based collection and dissemination methods. The approach is especially effective in collecting intermediate node specific information associated with the paths of particular monitored flows between source and destination pairs. Such a procedure can incur quite a lot of overhead for traditional approaches, such as the identification of appropriate nodes along the flow path and dissemination of data between each node and the associated collector. With an intrinsic monitoring approach, the packet is used to carry the

* This publication is based on work performed in the framework of the European Commission ICT/FP7 project EFIPSANS. The authors wish to thank all the collaborators in EFIPSANS for their efforts and support for completing this work.

monitoring data collection request and store the collected data for dissemination to the collector.

The basic idea is that the performance of a data flow can be monitored between a source-destination pair by inserting specific information in an extension header of selected IPv6 packets in the data flow. By initiating an extension header at a source, and updating the extension header at any intermediate nodes along the source-destination path, a destination node can have a performance evaluation of select nodes in a network based upon the reported data in the IPv6 extension header. However there are associated security concerns regarding intrinsic monitoring with regards to privacy of information.

The Internet is operated by thousands of interconnected Internet Service Providers (ISPs), each ISP would like to keep their operational information confidential, such as IP address allocation, packet loss rate, etc. Moreover, monitored information is vulnerable to malicious attacks and data corruption due to the complexity of networks. It is, therefore, important to ensure security properties, such as data confidentiality and integrity of the monitored data carried within the IPv6 extension header and authentication of origin. In this paper, we propose an IPv6 hop-by-hop extension header and present a secure method for collecting and disseminating monitored information.

The rest of the paper is organized as follows. Section 2 summarizes related work. In Section 3, we define the requirements for secure intrinsic monitoring. In Section 4, we first present the design of the *hop-by-hop monitoring extension header*, then we discuss system assumptions and propose our method to protect the monitored data. Finally Section 5 concludes the paper and discusses future work.

2 Related Work

The utilization of IPv6 hop-by-hop extension headers have been studied in [1,6,7]. A method is proposed in [7] to accurately and efficiently determine the available bandwidth in IPv6 networks through the use of a proposed IPv6 timestamp hop-by-hop extension header. In RFC draft [1] the RR6 option is defined as another new IPv6 hop-by-hop extension header. Based on that, a "Record Route for IPv6" mechanism is described. In RFC draft [6], IPv6 hop-by-hop extension header is used to record information along a communication path. The collected information includes interface attributes and statistics such as IP address, speed, number of transmitted packets and so on. These RFC drafts were rejected mainly because of the lack of security consideration.

As previous work does not consider the implications of the security, deployment of such features has seen little progress. IPv6 has its own IP Security (IPsec) [5] protocol suite to provide security services at the IP layer. It enables communication nodes to establish mutual authentication, negotiate cryptographic keys, select required security protocols and determine the algorithms to use. Two fundamental elements of IPsec are Encapsulated Security Payload (ESP) [4] and Authentication Header (AH) [3]. However, neither ESP nor AH has been designed to encrypt and authenticate the augmentable IPv6 hop-by-hop extension header.

3 Security Considerations for Intrinsic Monitoring

Information security is critical for intrinsic monitoring. In this section, we focus our security considerations on the confidentiality and integrity of monitored data and authentication of origin. Confidentiality means stored or transmitted monitored data cannot be read or altered by an unauthorized party. ISPs consider monitored data proprietary and are unwilling to disclose their internal network structure and performance information details. Integrity means any alteration of transmitted or stored monitored data can be detected. The receiver should be able to detect altered monitored data to prevent misinterpreting the meaning of the original data or even malicious attacks. Authentication of origin ensures the monitored data from trusted sources.

There are few requirements for doing the authentication and encryption to achieve the information security purpose. First, the amount of encrypted data should be reduced to a minimum because encryption is computationally expensive. Each network node should only encrypt the data it inserts to reduce the encryption overhead. The second is to limit the authentication overhead to a minimum. The network node should only authenticate the data it inserts to reduce the authentication overhead. Both the encryption and authentication is computationally expensive, so it is important to use suitable encryption and authentication algorithms to reduce the requirements for computing resource.

4 Protecting the Hop-by-Hop Extension Header

A *hop-by-hop monitoring extension header* is composed of a header and multiple performance metric records as shown in Table 1. This design aligns all the data into a 32-bit word boundary.

It is flexible to use the IPv6 Hop-by-hop extension header to collect monitoring data. Firstly, the *hop-by-hop monitoring extension header* can indicate all performance metrics it is interested in collecting. Example identifier values could be number of packets forwarded, timestamps, congestion points, etc. Second, it can also specify the nodes it is interested in monitoring. Routing header options can be used together to monitor designated intermediate nodes. Third, an IPv6 address can also be included in the *hop-by-hop monitoring extension header* to indicate a specific node and only the information from this node be collected. Fourth, it is also possible to collect all the intermediate nodes information because IPv6 hop-by-hop extension header will be processed by every router along the path.

Maximum Transmission Unit (MTU) discovered by the sender is also included into the header to facilitate the intermediate router to detect packet size. The intermediate router can dump the collected data records to an ICMP packet which

Table 1. IPv6 Hop-by-hop Monitoring Extension Header Format

Next Hdr	Hdr Ext Len	Option type	MTU Len
Sequence Number			
Record Count	Num of Metrics	Flags	Record Hdr Length
Identifier 0	Identifier 1	. . .	Padding
IPv6 address(optional)			
SPI			
Node Position 0	Num of supported Metrics	Authentication data length	Record Length
Identifier 0	Identifier 1	. . .	Padding
Identifier0 Data			
Identifier1 Data			
. . .			
IdentifierN Data			
Padding and padding length			
Authentication Data			
. . .			
SPI			
Node Position N	Num of supported Metrics	Authentication data length	Record Length
Identifier 0	Identifier 1	. . .	Padding
Identifier0 Data			
Identifier1 Data			
. . .			
IdentifierN Data			
Padding and padding length			
Authentication Data			

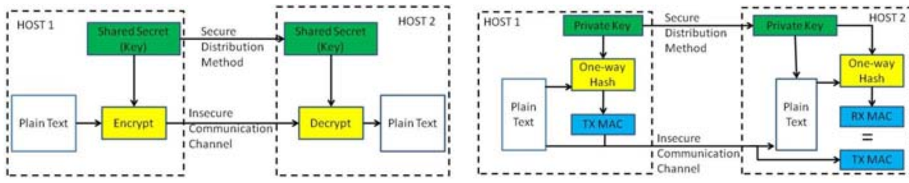
will be sent back to the collector and the packet will continue being forwarded to the destination.

Now we present our method of protecting the *hop-by-hop monitoring extension header*. As Internet Key Exchange version 2 protocol (IKEv2) [2] is integrated as a core component of IPv6 protocol stacks, we assume that it is supported by every network node. A protocol, for example an extended ICMPv6 protocol, initiates IKEv2 protocol for all nodes along the path in order to create IPsec Security Associations (SAs) with the destination node. SA includes keys, key lifetime, encryption and authentication algorithm, and related parameters.

IKEv2 offers a reliable and efficient key exchange scheme to provide a secure data transport. It also defines procedures to establish, negotiate, modify, and delete SAs. The advantage of IKEv2 is that it enables on-demand creation of keys for SAs and facilitates the use of keys in a large distributed system.

Our method is a combined encryption and authentication method. After the SAs are created, each node along the path shares a secret key with the destination node. Asymmetric algorithms use a significant amount of computational resources in comparison to their symmetric counterparts and therefore are not chosen to encrypt monitored data.

The symmetric encryption algorithms use a single key to encrypt and decrypt the data. The advantage is that symmetric algorithms use significantly less computational resources than their asymmetric counterparts. This single key is exchanged securely before the secure communication using key management protocols in IPsec framework. Typical key sizes are 64, 128, or 192 bits. Fig 1(a) shows how a shared secret key is used for protecting confidentiality of



(a) Symmetric Cryptography. (b) Message Authentication Code.

Fig. 1. Encryption and authentication

each record. The data portion starting from *node position* until *padding length* field (inclusive) is encrypted.

Message Authentication Code (MAC) algorithms are used to ensure authentication and data integrity. The MAC is the *authentication data* field of each record in the IPv6 *hop-by-hop monitoring extension header* which is computed from the monitored data using the shared key. The authenticated data starts from *SPI* until *padding length* field (inclusive). After the receiver gets the message, it recomputes the *authentication data* field using the authentication algorithm and shared key, then compares the result with the received *authentication data* field. If they are equal, the authentication is successful. The process is shown in Fig 1(b). The shared secure key authenticates the sender, and the hashed result ensures data integrity.

In a typical scenario, there are one sender, one receiver, and multiple intermediate nodes. The procedures can be summarized as follows: The sender initiates monitoring by creating a *hop-by-hop monitoring extension header*, which indicates the interested performance metrics. the sending pattern will be specified according to the monitoring purposes, e.g. from a specific application.

An intermediate router receives the packet with *hop-by-hop monitoring extension header*, it recognises it is a monitoring packet. After decapsulating the packet, it collects its performance metric values according to the requirements and generates a monitoring record. It then encrypts this record using the shared key. The authentication data will be generated for the encrypted record only. Finally, the encrypted data and the authentication data will be inserted into the *hop-by-hop monitoring extension header* as a record following the format defined in Table 1.

The destination node receives the monitoring packet and processes records one after another. Each record is associated with a *SPI* field, which identifies the SA to which this monitoring record belongs. The encryption and decryption is very computationally expensive, so the decryption should be done for the receiver after the successful authentication. The destination node authenticates the monitoring record using the key from corresponding SA, applies the hash algorithm to the encrypted monitoring record, and if the results match, then both the authenticity of the sender and the integrity of this monitoring record are assured. Then the decryption algorithm and key associated with the SA decrypt the monitoring record. The monitoring record will be processed by the destination node one by one until it finishes or error occurs.

5 Conclusions and Further Work

In this paper, we have proposed a secure method for collecting monitored network information utilizing the IPv6 hop-by-hop extension header. A generic IPv6 hop-by-hop extension header which can be easily extended to include all the possible monitored information has been designed. We suppose that nodes are deployed with IPsec framework support, and a protocol, such as extended ICMPv6, initiates the key distribution and SA negotiation between the nodes along the flow transfer path and the destination node. Based on that, we have also discussed a light-weight information encryption and authentication scheme to securely transport collected monitoring information. To our knowledge, it is the first try to offer a secure IPv6 hop-by-hop extension header for all kinds of monitoring purpose, which is a significant step for intrinsic monitoring towards real deployment.

Several issues will need further investigation: 1) Due to the expensive computational cost of the IKEv2 exchange, which mainly includes the Diffie-Hellman key exchange, certificate handling and the authentication processing steps, IKEv2 requires quite some time for session establishment. The monitoring may conduct periodically to collect performance information to ensure the optimized operations of communication networks. Essential future work will seek fast re-establishment of the sessions when session expires or network problems occur. 2) An efficient and optimized compression method for monitored information in order to transport collected performance records efficiently.

References

1. Toutain, L., Durand, A.: Ipv6 traceroute option, IPv6 Working Group Internet Draft (June 1997)
2. Kaufman, C.: Internet Key Exchange (IKEv2) Protocol, RFC 4306 (Proposed Standard), Updated by RFC 5282 (December 2005)
3. Kent, S.: IP Authentication Header, RFC 4302 (Proposed Standard) (December 2005)
4. kent, S.: IP Encapsulating Security Payload (ESP), RFC 4303 (Proposed Standard) (December 2005)
5. Kent, S., Seo, K.: Security Architecture for the Internet Protocol, RFC 4301 (Proposed Standard) (December 2005)
6. Kitamura, H.: Connection/link status investigation (csi) for ipv6 hop-by-hop option and icmpv6 messages extension, Internet Draft, Work in Progress (1999)
7. Crocker, J.B.M., Lazarou, G., Picone, J.: A bandwidth determination method for ipv6-based networks. International Journal of Computers and Applications (2009)

A Methodology of Traffic Engineering to IP Backbone

José E. Bessa Maia[1], Arnoldo N. da Silva[2], Jorge L.C. Silva[1], and Paulo R.F. Cunha[2]

[1] Department of Statistics and Computing, State Univ. of Ceará – UECE
[2] Informatics Center, Federal Univ. of Pernambuco - UFPE
{jmaia,jlcs}@larces.uece.br, {ans2,prfc}@cin.ufpe.br

Abstract. It is essential for the network operator to anticipate events leading to network node and link capacity breakdown in order to guarantee the Quality of Service (QoS) contract. Traffic prediction can be undertaken based on link traffic (aggregate), or on origin-destination (OD) traffic that presents better results. This work investigates a methodology for traffic engineering based on multidimensional OD traffic, focusing on the stage of short-term traffic prediction using Principal Components Analysis and a Local Linear Model. The results validated with data on a real network present a satisfactory margin of error for its adoption in practical situations.

Keywords: Traffic Engineering, Traffic Prediction, Principal Components Analysis, Local Linear Model.

1 Introduction

To deliver the Quality of Service (QoS) contract it is essential for the network operator to manage how traffic flows in the network. The growing demand for link bandwidth and node capacity is a frequent phenomenon in IP network backbones. Traffic engineering is a data flow control process in a network, such as to optimize the use of its resources. It works, therefore, with currently installed network resources, and seeks to determine the best manner of extending the present system's useful life. Traffic prediction can be conducted in different time scales. This work considers from one to several days anticipation as the short-term for traffic engineering predictions, while in capacity planning the so-called long-term predictions seek to anticipate network link and node capacity breakdown by 6 to 12 months or several years.

The traffic engineering methodology considered here decomposes the problem into three large stages: (i) data acquisition and processing, (ii) traffic prediction and tendency analysis, and (iii) diagnosis, engineering, sizing and expansion planning. The three steps in the methodology presented is commonly used for the traffic engineering task. What is different from others [1,2,3] is the combination of techniques used in each step. A critical step in this process is the traffic prediction activity on which the whole future planning and sizing depend. This task will receive attention in this paper while the other phases will be discussed in another opportunity.

Traffic prediction can be conducted based on link traffic (aggregate), or on origin-destination (OD) traffic that presents better results. There are two difficulties for predictions based on OD traffic: the first is obtaining the OD Traffic Matrix time series, and the second is the high dimension of the problem in big networks, which grows to the square of the number of nodes in the network. This fact leads practitioners to frequently work with individual predictions by link, an approximation which does not take into consideration the correlation between flows, which can be considered as a naïve approach.

It is understood that OD traffic constitutes a more direct and structural information about network dynamics than the aggregated traffic in the links. Lakhina et al.[4] shows that through principal components analysis (PCA) OD flows can be modeled using a small number of independent components. With the same approach [5] shows, through PCA, that traffic characteristics in one router are also found in other routers in the same backbone, observing that intrinsic OD flow structures in large networks tend to follow a pattern.

In the context of time series prediction, some papers use a combination of clustering techniques, such as k-means or SOM (Self-Organizing Map) with local models [6]. Clustering methods work with subsequences of the initial time series. Such subsequences are vectors of successive past values that can be extracted from the series using a sliding window. Though the SOMs are usually considered as a classification or recognition tool, there are a few works where SOMs were used in forecasting tasks. For example, some authors use SOMs to create clusters in the regressor space, eventually associating each cluster to a linear local model or a nonlinear one [7].

This work investigates a methodology for traffic engineering based on multidimensional OD traffic, using Principal Components Analysis as a technique for dimensionality reduction and a Local Linear Model as a technique for trend analysis. Using PCA, the prediction is applied to a small number of principal components (PCs) that retain a significant part of the data properties, allowing identification of the behavior profile of traffic common to several flows. Traffic behavior trends and prediction are applied to the time series of scores calculated for each PC using SOM and a local linear regression model. In this manner, only the flows tending to increase and reduce demand for bandwidth will be analyzed.

The remainder of the work is organized as follows: Section 2 details the steps of the methodology; Section 3 presents and discusses the results of the tests conducted to validate the procedure; the article is concluded in Section 4.

2 Traffic Engineering Methodology

The three stages in methodology will be described in the following paragraphs. Since the methodology adopts multidimensional OD traffic prediction, the stage of data acquisition and preparation is critical for the success of the procedure. This paper does not analyze the tasks routing, configuration and sizing. A diagnosis example is show in next section.

2.1 Acquisition and Pre-processing of Traffic, Topology and Routing Data

The objective of this stage is to obtain a time series of OD traffic data that is representative of real behavior of traffic on the network. Technically, OD traffic can also be measured directly. Direct measurement would involve mapping origin IP and destination IP addresses of all packets that leave a node. However, the inexistence of adequate infrastructure, and the cost involved, lead operators to seek other solutions. The solution most researched nowadays consists in estimating the OD Traffic Matrix based on measurements of traffic in the links [8]. This task consists in solving the following linear inverse problem for X: Y = AX, where Y is the traffic vector measured in the links, A is a routing matrix, and X is a vector representing the OD traffic matrix. This is a badly conditioned problem and several solution methods have been tested in attempt to obtain a satisfactory solution. Medina [8] informs that estimation with an error less than 10% would be satisfactory for use in traffic engineering.

Setting up the estimation problem mentioned before also depends on information about network routing and topology. If the network adopts dynamic routing the problem becomes much more complex. Some proposals already exist in literature to address this problem, but which still do not appear satisfactory. Other issues related to this stage are: scheduling, programming, gathering (centralizing) and storing measurements, filtering, sampling intervals and traffic aggregation, definition of the data window and outlier treatment. In fact, depending on the granularity of data collected, the same data can be used in traffic engineering or in capacity planning, making the aggregation adequate. This work uses direct measurement data from a real network in the validation phase [9].

2.2 Traffic Prediction and Trend Analysis

This step initially works with the reduction of data bulk by means of Principal Components Analysis [4,5]. The objective of PCA is to transform a quantity of n correlated original variables into non-correlated variables called Principal Components. In this manner PCA can reduce the dimension of the original variables using the information in the more relevant k components (k < n). The k PCs are chosen according to the explained percentage of variability for each component. Each of these components is a linear combination of the original variables.

$$Y_j = a_{j1} X_1 + a_{j2} X_2 + \ldots \quad + a_{jn} X_n \qquad j = 1, \ldots, n$$

The vector formed by the coefficients (a_{j1}, a_{j2}, … , a_{jn}) in each PC is called an eigenvector. Applying the values of the time series of the original variables (OD flows) in each PC generates a series of scores called an eigenflow. The behavior of the series in the eigenflow allows the identification of behavior profiles of traffic on the OD flows. The values of the elements of the eigenvector act as weights, informing how much of the eigenflow behavior is present in the OD flow, in other words, weights that surpass a threshold indicate presence of the behavior. This threshold is calculated from the equation $1/\sqrt{n}$ specified by [4], where n is the number of random variables.

The trend analysis phase is done, in this stage, using a Local Linear Model (LLM) [6,7]. Based on the splitting of the past dynamics into clusters using SOM, local models are built to capture the possible evolution of the series given the last known values. The method can be applied to any time series prediction problem, but is particularly suited to data showing non-linear dependencies and cluster effects, as many traffic network series do. According to the type of trend presented, the eigenflows are selected and the flows with greater weights are analyzed. Based on this calculus, it is possible to verify the set of flows that contain the sought trend. The route chart locates the links where these flows pass through, thus identifying the flows responsible for the load growth trend on the link. The results expected for this phase are the estimates for the Hurst parameters, the mean values (tendency), and the variance, necessary to link and buffer dimensioning. The methodology consists in computing the Hurst parameters and the variance of the PCs and using the corresponding reconstruction weights (eigenvectors) in order to obtain the values for each OD flow.

3 Results and Discussion

To validate the traffic prediction technique, data from a real network was used. The scenario used in the case study belongs to the GEANT network, a backbone of the European National Research and Education Networks (NRENs). The topology of GEANT shows a network with 23 nodes and 37 links. The set of data for analysis was collected over 4 months at 15-minute intervals provided by Uhlig et al. [9].

3.1 Traffic Prediction

The technique here demonstrated for predicting traffic is applicable to both traffic engineering and capacity planning. The results presented are directed to the short-term prediction adopted in traffic engineering.

In the pre-processing phase filtration and outliers elimination was carried out. Filtration used first order exponential smoothing with coefficients [1.0, 0.11]. These coefficients were determined by the following method: the frequency spectrum of the series were superposed on a single graph, and by visual inspection of the resulting blot the cut frequency was defined, and consequently the coefficients [1].

Applying PCA initially allows verifying the presence of atypical data (outliers). In the graph of the first with the second PC, the outlying dots of the most concentrated regions are considered outliers. PCA for the data, after removing the outliers, shows that the graph between the two first PCs (Figure 1l) now shows dispersion with different regions of dot concentration. PCA generates 529 principal components, where the first 10 PCs explain 89.4% of the total variability (Figure 1l), allowing obtaining a good representativeness of traffic data.

The daily seasonal was maintained and a time series of traffic was obtained for each hour of the days. So we can get the curve for the prediction of daily traffic. The results are shown only for the greatest movement hour.

The width of the window used both in the prediction and the trend analysis was of 8 measures. An expressive number of experiments was conducted with the window starting at random points along the series, and the results presented here are representative of these experiments. Data aggregation was tested in hourly and daily measurements and the series of the highest daily values being used as representative traffic, which also generated good predictions. However, the results of this work considered a series with samples taken every hour.

The overall performance of the forecasting model is evaluated by Mean Absolute Percentage Error (MAPE), given by [4,6]:

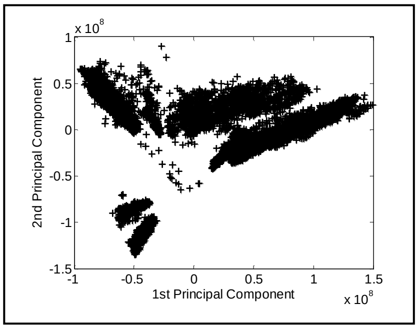
$$MAPE_h = \frac{1}{N}\sum_{i=P+1}^{P+N} \frac{|y_i - \hat{y}_{i,i-h}|}{|y_i|} \times 100\%$$

where y_t denotes the desired value at the time t; $y_{t,p}$ the predicted value for period t and computed at period p; P is the present time and N the number of forecasts.

Tables 1 summarize the results of this work with the final diagnosis of the network. The results of the prediction in the first 10 eigenflows for h = 1, 2, 5 and 10 days in advance presented maximum error of 21.20%. Table 1l shows the explained variance and the Hurst parameters of each of the eigenflows. The average MAPE values obtained over the entire trace in good position for h = 1, 2, 5 and 10 were, respectively, 7.62%, 8.80%, 14.02% and 19,1%.

Two rules for obtaining the Hurst parameter for each final OD flow, adopted for sizing the network, were tested: the average of the Hurst parameters of the eigenflows weighted by the scores of the corresponding eigenflows used in the reconstruction of the OD flows and to use the largest Hurst parameter between PCs. The latter was adopted. Table 1r summarizes the final network diagnostic.

Figure 1r shows a sketch of the trend analysis in the eigenflow of PC1 for the 10 following days. The LLM model [6] used here is detailed in another paper. The calculated projection follows a growing trend, corresponding to the real series in the same period. The flows with the most strongly presence behavior in PC1 are identified according to the values of the scores.



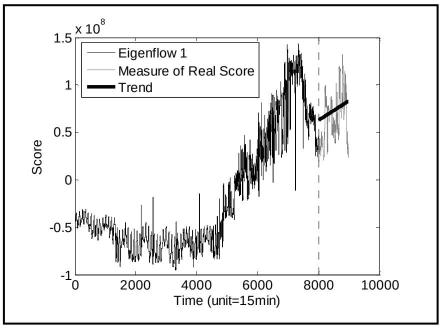


Fig. 1. First PC x Second PC without outliers (l) and Percentage of explained variability (r)

3.2 Discussion

We tested the clustering techniques k-means and SOM. The critical point of the training model is the adjustment of the subsequence size of the PC time series (eigenflows) leading to lower MAPE values in the OD flows prediction. The program consisted of a loop that varied the subsequence size in the interval [4, 40] samples of the PC time series and the initial time series, while the estimated value of the MAPE index is calculated in the OD flows prediction. This procedure was repeated for h = 1, 2, 5 and 10. In this regard there is some margin for optimization of the process. The paper presents the most favorable results and used the SOM algorithm with local linear regression. Figure 1r is only one example of the representation that can be achieved when applying local linear regression on subsequences of time series.

The principal components calculation using standard routines is based on eigenvalues and eigenvectors of the covariance matrix. The programming was done in MATLAB© with the support of the Statistic Toolbox©. The individual maximum errors in eigenflows prediction reached 21.20% in eigenflow 10 for h = 10. However, note that the reconstruction of OD flows is obtained by a linear combination of weighted eigenflows, which explains why the maximum average error obtained was 8.71% ,Table 1l. Draws attention the small variance values reported in Table 1l for both prediction error and Hurst parameter.

Table 1. Explained variance and Hurst parameter for the first 10 PCs (left) and network diagnosis (right)

	Var.(%)	Hurst Param.
1	35.9265	0.7811
2	21.6029	0.8617
3	9.8312	0.6627
4	5.2462	0.8547
5	4.9846	0.7261
6	3.6100	0.8548
7	2.5283	0.6591
8	2.1914	0.7209
9	1.9529	0.5826
10	1.5567	0.6012

		Flows with prediction of growth	Flows with prediction of reduction	Flows with prediction of stability	Total
Error	Prediction	28	29	472	529
	Real	28	29	472	529
Average Traffic Prediction	Av.(%)	8.31	7.45	8.60	8.52
	Var.(%)	0.17	0.51	0.69	0.66
Hurst Parameter Prediction	Av.(%)	5.21	3.35	5.40	5.22
	Var.(%)	0.19	0.42	0.71	0.69

4 Conclusions

This work presented a solution for trend analysis in traffic engineering applied to OD flows. OD traffic matrix estimation, routing, configuration and sizing will be subjects of other papers. The case study used a real network with 529 OD flows. After the process of removing atypical data from the traffic matrix, PCA showed that only 10 PCs were sufficient to represent the flow characteristics. From the PCs it was possible to identify behavior profiles, and those with a growth or reduction trend were selected, automatically locating flows with greater bandwidth demand for the links utilized. The results, when compared with real data, resulted in an average MAPE values obtained over the entire trace in good position for h = 1, 2, 5 and 10 of, respectively, 7.62%, 8.80%, 14,02% and 19,1%, identifying flow trend (growth, reduction and stability) with 100% accuracy. Considering a homoscedastic (constant variance) traffic scenario, this approximation allows including, in the process, the prediction of second order parameters necessary to sizing, such as the Hurst parameter that measures long-range dependency, with an error of 5.22%.

Is planned, as a future work, to apply the presented methodology to other examples for validation. At this moment, we are experimenting different trend analysis techniques including non-linear ones. The goal is to use the approach proposed here for producing a traffic engineering software tool.

References

[1] Babiarz, R., Bedo, J.: Internet Traffic Mid-term Forecasting: A Pragmatic Approach Using Statistical Analysis Tools. In: Boavida, F., Plagemann, T., Stiller, B., Westphal, C., Monteiro, E. (eds.) NETWORKING 2006. LNCS, vol. 3976, pp. 110–122. Springer, Heidelberg (2006)

[2] Lv, J., Li, T., Li, X.: Network Traffic Prediction Algorithm and its Practical Application in real network. In: 2007 IFIP Int. Conf. on Network and Parallel Computing (2007)

[3] Gunnar, A., Abrahamsson, H., Söderqvist, M.: Performance of Traffic Engineering in Operational IP-Networks - An Experimental Study. In: Magedanz, T., Madeira, E.R.M., Dini, P. (eds.) IPOM 2005. LNCS, vol. 3751, pp. 202–211. Springer, Heidelberg (2005)

[4] Lakhina, A., Papagiannaki, K., Crovella, M., Diot, C., Kolaczyk, E., Taft, N.: Structural analysis of network traffic Fows. In: Proc. ACM SIGMETRICS (2004)

[5] Cortez, P., Rio, M., Rocha, M., Sousa, P.: Internet Traffic Forecasting using Neural Networks. In: 2006 IJCNN, pp. 16–21 (2006)

[6] Vesanto, J.: Using the SOM and Local Models in Time-Series Prediction. In: Proc. WSOM 1997, pp. 209–214 (1997)

[7] Barreto, G.A.: Time Series Prediction with the Self-Organizing Map: A Review. Studies in Computational Intelligence (SCI) 77, 135–158 (2007)

[8] Medina, A., Taft, N., Salamatian, K., Bhattacharyya, S., Diot, C.: Traffic matrix estimation: Existing techniques and new directions. In: Proc. ACM SIGCOMM (2002)

[9] Uhlig, S., et al.: Providing public intradomain traffic matrices to the research community. ACM SIGCOMM Computer Communication Review 36(1) (2006)

A Network Optimization Model for Multi-layer IP/MPLS over OTN/DWDM Networks

Iyad Katib and Deep Medhi[*]

Computer Science & Electrical Engineering Department
University of Missouri-Kansas City, USA
{IyadKatib,DMedhi}@umkc.edu

Abstract. The operational model for large Internet service providers is moving to a multi-layer architecture consisting of IP/MPLS coupled with OTN/DWDM. While there has been significant work on multi-layer networks, the explicit modeling in IP/MPLS over OTN/DWDM has not been addressed before. In this paper, we present a detailed network optimization model for the operational planning of such an environment that considers OTN as a distinct layer with defined restrictions.

1 Introduction

With the proliferation of Internet traffic caused by new Internet users and applications that require high bandwidth and QoS guarantees, the current IP backbone network is facing new challenges. Optical Transport Network (OTN), defined in [1,2], is a new-generation transmission layer technology that supports high-speed optical signals, and is emerging as promising for the next-generation transport networks featuring large-granule broadband service transmissions. It combines the benefits of both SONET (synchronous optical network) and DWDM (dense wavelength-division multiplexing) while offering viable solutions to their shortcomings. Some advantages of OTN include more productive multiplexing and switching of high-bandwidth (around 100 Gbit/s) signals and the capability of cross connect dispatching of wavelengths and sub-wavelengths resulting in efficient wavelength utilization. In addition, OTN offers a "digital wrapper" layer that defines signal overhead to support powerful performance-monitoring, fault detection, and forward error correction (FEC); this is useful to limit the number of required regenerators in the network and hence, extends the transmission distances at a lower cost. For this and many other benefits, large Internet providers are recognizing that IP/MPLS-over-OTN/DWDM is an emerging architecture that bridges integration and interaction between the optical layer and the IP layer [3]. The operational implementation of OTN is slated to be on top of DWDM systems in the future.

Multi-layer networks have been an important research topic in recent years. The vast majority of research considered the two-layer architecture: IP-over-WDM. They mostly concentrated on the resiliency or traffic engineering of the

[*] Supported in part by NSF Grant No. 0916505.

network [4,5,6]. A two-layer network architecture that consists of a traffic layer over a DWDM transport layer, such as IP/MPLS-over-DWDM, suffers from a major limitation. In this architecture, the core routers are connected through the physical optical fibers and DWDM provides the transmission channels. Adding wavelengths to increase the capacity of the backbone network would, in turn, require extending the capacity of the routers. However, approximately 60-70% of the core routers found in this case are to be used for forwarding services instead of processing local add/drop services on the nodes [3]. This is where OTN comes into the picture. The OTN layer, as a middle layer, separates the logical from the physical topologies. IP/MPLS routers will be connected based on the logical topology while OTN/DWDM provides connections based on the physical topology. As a result, a demand that requires more than one logical link at the IP/MPLS layer can be accommodated in a fewer number of links at the OTN/DWDM layer, and thus, significantly reduces the forwarding services that the core routers perform.

On the other hand, previous work that considered the OTN system, such as [7], have embedded the OTN in the DWDM layer implicity. They have not taken into consideration the OTN layer as a separate layer that imposes its own restrictions. This does not reflect a precise view of the next-generation architecture nor does it produce accurate results of the models or simulations based on that assumption. However, the functionalities each technology provides are distinguishable, and this impels us to model each of them separately, i.e., to visualize the relation between the two layers with OTN as the optical logical layer on top of the physical DWDM layer.

In this paper, we introduce an explicit architecture for IP/MPLS-over-OTN-over-DWDM as depicted in Fig1. Considering OTN as a separate layer with its own restrictions, we model this architecture as a three-layer network. Each layer becomes a demand on the layer underneath it. In this architecture, we consider the network operational planning problem and present an explicit network optimization formulation, including the cost structure for different interfaces. Our formulation uses [8] as the basis of understanding and presents a novel explicit optimization model for IP/MPLS over OTN/DWDM networks.

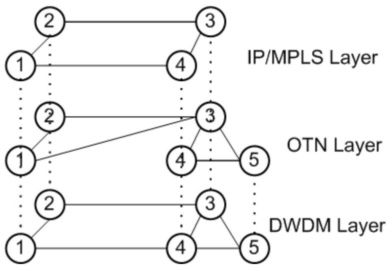


Fig. 1. IP/MPLS over OTN/DWDM Network

The rest of the paper is organized as follows. In section 2, we give a brief overview of the OTN signals bit rates and the multiplexing rules. In section 3, we present the problem formulation; in section 3.1, we discuss the cost model associated with the objective function. Finally, in section 4, we briefly outline our direction for future work.

2 OTN Signals and Multiplexing

The OTN system consists of multiple layers [2]. Each layer is distributed along the network and is activated at its termination points to perform specific activities. For the scope of this paper, we are concerned with the Optical Data Unit (ODU) layer that provides an end-to-end functionality of digital path signals for transporting the client data that can be of diverse formats such as STM-N, Ethernet, IP/MPLS, ATM, etc. The ODU layer supports three bit-rate client signals, i.e. 2.5, 10, and 40 Gbit/s, which are referred to as ODUk ($k = 1, 2, 3$), respectively. We are also interested in the ODUk time division multiplexing. It allows several lower bit-rate signals to be multiplexed and transported over a higher bit-rate optical channel and maintains an end-to-end trail for the lower bit-rate signals. This may occur when a client signal does not occupy an entire wavelength. The multiplexing of signals is defined as follows: up to 16 ODU1s or 4 ODU2s can be multiplexed to an ODU3, and 4 ODU1s can be multiplexed to an ODU2. While it is possible to mix ODU1s and ODU2s in an ODU3, only one stage multiplexing is allowed to reduce the overall network complexity. For example, it is possible to perform the multiplexing of (ODU1 $\rightarrow$ ODU2) or (ODU1 and ODU2 $\rightarrow$ ODU3), but not (ODU1 $\rightarrow$ ODU2 $\rightarrow$ ODU3).

Let U_1, U_2, and U_3, denote ODU1, ODU2, and ODU3, respectively. Then for the multiplexing process we can say: $4U_1 = U_2$, $4U_2 = U_3$, and $16U_1 = U_3$. Furthermore, U_1 and U_2 can be multiplexed into a U_3 signal according to the following rule: $U_3 = SU_2 + (4 - S)4U_1$ where $(0 \leq S \leq 4)$.

Table 1 shows the OTN different signals bit-rate, and how OTN signals can be carried over a single wavelength assuming that the maximum wavelength bit rate is 40 Gbit/s.

Table 1. OTN Signals, Data Rates and Multiplexing

ODU Signal	Bit-Rate (Gbit/s)	Max. ODUs in a wavelength
ODU1	2.5	16
ODU2	10	4
ODU3	40	1

3 Problem Formulation

In this section, we provide a link-path formulation to describe the multi-layer network optimization problem. The key point of the model is that each upper layer imposes demands on the neighboring lower layer, while explicitly considering all technological restrictions. In our example of Fig. 1, the demand volume is realized by the means of flows assigned to paths of layer IP/MPLS. The summation of flows passing through each link in the IP/MPLS layer determines the capacity of the layer. Next, the capacity of each link of the IP/MPLS layer becomes a demand realized by the means of flows assigned to paths in the OTN layer. And if we sum up the flows through each link of the OTN layer, the resulting loads determine the capacity of the layer. The last step is analogous for the DWDM. Table 2 lists the notations used in our formulation. We first discuss each constraint separately.

Table 2. Notations used in Multi-layer Network Optimization Formulation

Notation	Discription
D	Set of demands between source-destination pairs of the IP/MPLS layer
P_d	Set of candidate paths for demand $d \in D$
E	Set of links of the IP/MPLS layer
Q_e	Set of candidate paths of OTN layer for realizing capacity of link $e \in E$
G	Set of links of the OTN layer
Z_g	Set of candidate paths of DWDM layer for realizing capacity of link $g \in G$
F	Set of links of the DWDM layer
J	Set of modular interfaces of OTN $1, 2, 3$
Constants	
h_d	Volume of demand d
δ_{edp}	=1 if link e belongs to path p realizing demand d; 0, otherwise
γ_{geq}	=1 if link g belongs to path q realizing capacity of link e; 0, otherwise
ϑ_{fgz}	=1 if link f belongs to path z realizing capacity of link g; 0, otherwise
M	Module size for IP/MPLS layer
U_j	Module size for OTN layer link capacities $j \in J$
N	Module size for DWDM layer link capacities
η_e	Cost of one capacity unit of module M of the IP/MPLS layer link e
β_{gj}	Cost of one capacity unit of module type U_j of the OTN layer link g
ξ_f	Cost of one capacity unit of module N of the DWDM layer link f
Variables	
x_{dp}	IP/MPLS flow variable realizing demand d allocated to path p (non-negative, continuous or binary)
m_{eq}	OTN flow variable allocated to path q realizing capacity of link e (non-negative integral)
k_{gjz}	DWDM flow variable allocated to path z realizing capacity of link g of interface j (non-negative integral)
y_e	Number of modules M to be installed on link e in the IP/MPLS layer (non-negative integral)
w_{gj}	Number of modules U_j to be installed on link g in the OTN layer (non-negative integral)
b_f	Number of modules N to be installed on link f in the DWDM layer (non-negative integral)

Constraints

We assume that an IP demand $d \in D$ can be carried over different tunnels P_d, and the fraction of the demand volume for d to be carried on tunnel p is x_{dp}. This can be expressed as follows:

$$\sum_{p \in P_d} x_{dp} = 1 \qquad d \in D \tag{1}$$

defining the IP/MPLS layer demands. It may be noted, however, that if each demand is to be carried on a single tunnel, then we can consider the same constraint and define x_{dp} to be a binary decision variable instead. Thus, either requirement on the flow can be captured by the above constraint

Next, we consider the IP/MPLS layer capacity feasibility constraints (2). These assure that for each IP/MPLS layer link $e \in E$, its capacity is allocated in modules of size M and is not exceeded by the flow using this link as shown below:

$$\sum_{d \in D} h_d \sum_{p \in P_d} \delta_{edp} x_{dp} \leq M y_e \qquad e \in E \tag{2}$$

Here, M is the allowable granularity of each MPLS tunnel.

The constraints (3) below specify how the capacity of each IP/MPLS layer link $e \in E$ is realized by means of flow m_{eq} and is allocated to its candidate paths from the routing list in the OTN layer.

$$\sum_{q \in Q_e} m_{eq} = y_e \qquad e \in E \tag{3}$$

We next consider the OTN layer capacity feasibility constraints, shown below(4). They assure that all flows routed on each OTN layer link $g \in G$ do not exceed their capacity that is allocated in modules of sizes U_j, which represent the three modular interfaces of OTN.

$$M \sum_{e \in E} \sum_{q \in Q_e} \gamma_{geq} m_{eq} \leq \sum_{j \in J} U_j w_{gj} \qquad g \in G \tag{4}$$

The following constraints (5) specify how the capacity of each OTN layer link $g \in G$ is realized by means of flow k_{gjz}, allocated to its candidate paths from the routing list in the DWDM layer.

$$\sum_{z \in Z_g} k_{gjz} = w_{gj} \qquad j \in J, g \in G \tag{5}$$

These next constraints (6) are the DWDM layer capacity feasibility constraints. They assure that for each physical link $f \in F$, its capacity allocated in modules of size N is not exceeded by the flow using this link. Note that N is the module size of the DWDM layer link capacity that is equal to the number of wavelengths per fiber, and b_f would be the number of fibers to be installed on link f.

$$\sum_{g \in G} \sum_{j \in J} U_j \sum_{z \in Z_g} \vartheta_{fgz} k_{gjz} \leq N b_f \qquad f \in F \tag{6}$$

The above then completes all the constraints in our multi-layer design model, along with the requirements on the variables as listed in Table 2.

Objective

The objective in our design model is to minimize the total network cost. Thus, the goal is to

$$\text{Minimize} \qquad \sum_{e \in E} \eta_e y_e + \sum_{g \in G} \sum_{j \in J} \beta_{gj} w_{gj} + \sum_{f \in F} \xi_f b_f \tag{7}$$

subject to the set of constraints (1)–(6).

3.1 Cost Model

Our objective function (7) tries to minimize the cost of network resources over all three layers. The final solution is the optimal number of tunnels (IP/MPLS layer), lightpaths (OTN layer), and fibers (DWDM layer) needed to satisfy the demands. Each layer has a different cost structure. This is now briefly described.

For the IP/MPLS layer, η_e is the unit cost of link $e \in E$; this is defined as the sum of the interface cost for the upper layer (η_e^U) and lower layer (η_e^L) ends of the connection between the IP/MPLS layer node and the OTN layer node, i.e., $\eta_e = 2\eta_e^U + 2\eta_e^L$, where 2 is for both ends. At the OTN layer, β_{gj} is the unit cost of link $g \in G$, and is equal to the cost of establishing a new lightpath on link g (β_g^L) plus the cost of multiplexing OTN signals (β_g^j), i.e., $\beta_{gj} = \beta_g^L + \beta_g^j$. For the DWDM layer, ξ_f is the cost of link $f \in F$, and is equal to the interface cost for line-cards connected to the transport end of a physical node to another physical node (ξ_f^I) plus a physical link distance cost (Δ_f), i.e., $\xi_f = 2\xi_f^I + \Delta_f$.

4 Future Work

The model presented here has a large number of constraints and variables even for a small network problem. Furthermore, the problem is $\mathcal{NP}$-hard, since simpler forms of network design problems, such as the single-path flow allocation (i.e., x_{dp} is binary) or modular link design, are shown to be $\mathcal{NP}$-hard [8]. Thus, we plan to verify the computational limits of our proposed model and develop efficient heuristic algorithms to solve the problem for large networks. This will be reported in a future work.

References

1. ITU-T Recommendation G.872: Architecture of optical transport networks (November 2001) (Amendment 1 December 2003, Corrigendum 1 January 2005)
2. ITU-T Recommendation G.709/Y.1331: Interfaces for the optical transport network (OTN) (March 2003) (Amendment 2 November 2007)
3. Bhatta, M.: Four challenges in backbone network. Huawei Communicate Issue 44, 40–42 (2008)
4. Cui, X., Wang, J., Yao, X., Liu, W., Xie, H., Li, Y.: Optimization of multilayer restoration and routing in IP-over-WDM networks. In: Conference on Optical Fiber communication/National Fiber Optic Engineers (OFC/NFOEC 2008), February 2008, pp. 1–10 (2008)
5. Gouveia, L., Patricio, P., de Sousa, A.F., Valadas, R.: MPLS over WDM network design with packet level QoS constraints based on ILP models. In: Proc. of IEEE INFOCOM 2003, March 2003, pp. 576–586 (2003)
6. Kuipers, F., Dijkstra, F.: Path selection in multi-layer networks. Communications 32, 78–85 (2009)
7. Bigos, W., Cousin, B., Gosselin, S., Foll, M.L., Nakajima, H.: Survivable MPLS over optical transport networks: Cost and resource usage analysis. IEEE Journal on Selected Areas in Communications 25(5), 949–962 (2007)
8. Pioro, M., Medhi, D.: Routing, Flow, and Capacity Design in Communication and Computer Networks. Morgan Kaufmann Publishers, San Francisco (2004)

Author Index