Cross-Layer optimization in a satellite communication network

SABRINA DUBROCA

KTH Information and
Communication Technology

Degree project in
Communication Systems
Second level, 30.0 HEC
Stockholm, Sweden

Master Thesis

Cross-Layer optimization in a satellite communication network

Sabrina Dubroca

August 28, 2013

Examiner:
Professor Gerald Q. Maguire Jr.
Supervisors:
Boris Buiron, Michel Delattre, Luc Loiseau, Eric Vitureau
Thesis performed at
Thales Communications

School of Information and
Communication Technology
KTH Royal Institute of Technology
Stockholm, Sweden

Abstract

This thesis aims to improve a satellite communication network which carries both data streams and Voice over IP (VoIP) communication sessions with resource reservation. The resource reservations are made using the standard protocols for Traffic Engineering: MPLS-TE and RSVP-TE. The goal of this thesis project is to optimize the number of concurrent VoIP calls that can be made, in order to use the available bandwidth while maintaining a guaranteed Quality of Service (QoS) level, which is not possible in the existing system.

This thesis proposes and evaluates a solution to this optimization problem in the specific context of a satellite modem system that was developed by Thales Communications. This optimization improves the system's ability to carry VoIP communications through better use of the available transmission resources. A solution to this problem would also increase the flexibility in bandwidth allocation within the modem system, and could provide a framework for future development.

The proposed solution allows all of the reservable bandwidth to be used. The amount of reservable bandwidth must be at least a little lower than the channel's available bandwidth in order to avoid congestion. Some areas of future work are proposed.

Keyword: QoS, Traffic Engineering, RSVP-TE, MPLS-TE, source routing, resource reservation, aggregation.

Sammanfattning

Detta projekt har försökt förbättra ett datornätverk bestående av satelliter som används till både data och Voice over IP (VoIP) kommunikation. VoIP använder sig av resursreservation som bestäms av standardprotokollen för Traffic Engineering, MPLS-TE och RSVP-TE. Målet är att optimera antalet samtidiga VoIP samtal så att det mesta av den befintliga bandbredden kan utnyttjas samtidigt som Quality of Service (QoS) kan garanteras. Detta är omöjligt i det befintliga systemet.

Projektet föreslår en lösning för problemet med modemet som utvecklas av Thales Communications och utvärderar därefter lösningen. Dessa optimeringar förbättrar systemets förmåga att driva VoIP kommunikationer genom att bättre använda de befintliga resurserna. En lösning för det här problemet skulle höja systemets flexibilitet och kunna användas som underlag för kommande utvecklingar.

Tack vare lösningen kan hela utsedda bandbredden reserveras. Antalet bandbredd som kan reserveras måsta vara minst lite lågre än total befintling bandbredd för att undvika överbelastning. Även några möjliga idéer för vidare undersökning föreslås.

Résumé

Ce projet a pour but d'améliorer un réseau de communication par satellite utilisé pour transporter des flux de données ainsi que des sessions de communication Voix sur IP (VoIP) avec réservation de ressources. Les réservations sont prises en charge par les protocoles standard de Traffic Engineering que sont MPLS-TE et RSVP-TE. L'objectif de ce projet est d'optimiser le nombre d'appels VoIP pouvant être passés en parallèle afin d'utiliser autant de bande passante que possible tout en offrant un niveau de Qualité de Service (QoS) garanti, chose impossible dans le système actuel.

Ce rapport propose et évalue une solution à ce problème d'optimisation dans le contexte spécifique du modem satellite développé par Thales Communications. Ces optimisations amélioreraient la capacité du système à transporter des communications VoIP grâce à une meilleure utilisation des ressources disponibles pour la transmission. Une solution à ce problème rendrait aussi l'allocation de ressources plus flexible au sein du système, et pourrait fournir une base à de futurs développements.

La solution proposée permet l'utilisation de toute la bande passante réservable. La quantité réservable doit être un peu inférieure à la bande passante totale disponible afin d'éviter la congestion. Les résultats de ces évaluations sont exposés. Enfin, ce rapport propose de futurs développements possibles.

Acknowledgements

First of all, I would like to thank my examiner and supervisor Professor Gerald Q. Maguire Jr., for his guidance during this project. His feedback has been invaluable, always quick, precise, thourough, and very helpful.

I also want to thank my supervisors at Thales Communications, who gave me the great opportunity to work on this project. I deeply appreciate the knowledge and the help they offered me throughout the course of the project, the time they took to guide me and provide crucial advice and ideas. I am very grateful for their kindness and the warm reception they gave me.

I would also like to thank my friends for their support and encouragement. I could not have succeeded without you.

Lastly, I want to express my gratitude to my parents. Thank you for always supporting me, never doubting me, and just being here for me.

Contents

List of Figures

List of Tables

Abbreviations and Acronyms

ACM	Adaptive Coding and Modulation
AS	Autonomous System
ATM	Asynchronous Transfer Mode
BE	Best Effort
CBR	Constraint-Based Routing
CFI	Canonical Format Indicator
CSPF	Constrained Shortest Path First
DAMA	Demand Assigned Multiple Access
DCCP	Datagram Congestion Control Protocol
DiffServ	Differentiated Services
DSCP	Differentiated Services Code Point
FEC	Forwarding Equivalence Class
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IntServ	Integrated Services
IP	Internet Protocol
IS-IS	Intermediate System To Intermediate System
LDP	Label Distribution Protocol
LER	Label Edge Router
LLC	Logical Link Control
LSA	Link State Advertisement
LSP	Label Switched Path
LSR	Label Switching Router
MPLS	Multiprotocol Label Switching
NHLFE	Next Hop Label Forwarding Entry
OSI	Open Systems Interconnection
OSPF	Open Shortest Path First
PBR	Policy-Based Routing
PCP	Priority Code Point
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RFC	Request For Comments

RSVP	Resource ReSerVation Protocol
RTCP	RTP Control Protocol
RTP	Real-Time Transport Protocol
SBC	Session Border Controller
SCTP	Stream Control Transmission Protocol
SIP	Session Initiation Protocol
SLA	Service Level Agreement
TCP	Transmission Control Protocol
TE	Traffic Engineering
TPID	Tag Protocol Identifier
TTL	Time to Live
UA	User Agent
UDP	User Datagram Protocol
VID	VLAN Identifier
VLAN	Virtual Local Area Network
VoIP	Voice over IP
VPN	Virtual Private Network

Chapter 1

Introduction

This chapter gives a general introduction to the topic of this thesis. The project's goal and the structure of this thesis are presented.

1.1 Introduction

Speaking is the most common and important form of human communication. Even over long distances and via traditonal telecommunication networks, voice has been a major form of communication. Telephones have been used for over a century. The first mobile phone was created over 60 years ago, and nowadays in industrialized countries and in many developing countries, a very large fraction of the population has a mobile phone.

Voice communication networks have evolved over the years, from the early days of the Public Switched Telephone Network (PSTN) to the ongoing transition to Voice over IP (VoIP). IP is becoming the base for most communication technologies and services, and both voice and data traffic are being carried over the same network.

According to the TCP/IP or OSI layered models, different technologies can coexist transparently at the link layer and different media can be used. However, since they have different characteristics, most importantly in terms of available bandwidth and delay, the physical and link layers have different impacts on the communications occurring at the upper layers. For example, satellite links offer different capabilities compared to a cellular network, a Wi-Fi interface, or a wired network. Some of these links support broadcast communication, cover different distances, and vary in terms of bandwidth, transmission delay, and reliability.

Moreover, a single network can be used for completely different types of communication. Some applications, such as file transfer, might demand a lot of bandwidth, but accept high delay. Others, such as multimedia streaming, require low delay with possibly high bandwidth. Voice over IP and interactive multimedia streams require both a bounded delay and a low jitter, i.e., low variation of the delay over time.

High network load – caused by too much traffic – represents a mismatch of the network's capabilities and the current traffic requirements, this requires specific measures to be taken in order to ensure a certain level of usability of all the services deployed on a network, and to guarantee a level of quality to the users. This set of measures is implemented to provide so-called Quality of Service (QoS). The goal is to achieve a better level of service than what is offered through best-effort services by prioritizing some of the traffic flowing through a network [*]. Another way to provide QoS on a network is to reserve resources. Both approaches can be combined in a QoS strategy.

The simplest approach to solve any QoS problem is to add resources so that the load is always low: low utilization and few users compared to what would be the maximum possible load for a given system. Although this approach might be realistic in some cases, for example, by adding fibers or wavelengths on an existing fiber to a network or CPUs to a server, it cannot be done in certain systems and under some conditions. Specifically, it is a problem to increase the data capacity of many satellite networks, hence the focus of this thesis project is on better management and utilization of the existing resources of such a network in order to achieve the users' objectives.

1.2 Different types of networks

This thesis focuses on mobile networks that can be deployed directly in the field of operations. These mobile networks are very different from fixed infrastructure networks in most aspects.

Infrastructure networks are mostly wireline (fiber optics or copper wires) networks with high throughput and low delay. They are designed to serve a large number of users, and power & size requirements for the equipment are not a major issue[†]. They are also mostly insensitive to external conditions such as weather, and tend to be highly connected, thus offering redundant paths in the core network. Additionally, adding more resources, either bandwidth or processing power, is reasonably convenient and not too expensive.

In contrast, mobile networks primarily utilize wireless links and have a lower throughput. Throughput can vary depending on external conditions, which leads to unreliable data rates and uncertain connectivity. The transmission delay tends to be higher than in infrastructure networks, especially when using a geostationary satellite orbiting at $36000km$, leading to a delay of about $240ms$ or more from one earth station to another. In order to be easily deployed in the field, these mobile stations have tight

[*] The resulting QoS for the unprioritized traffic will be lower under high network loads.
[†] There is currently a major effort to decrease the power consumed by this infrastructure [17].

power requirements and need to be reasonably compact. Rapidly deployable mobile communication infrastructures have been particularly important for military operations and for civilian disaster recovery efforts [*].

1.3 General principles of Quality of Service

QoS measures can either provide a higher quality of service under high network loads, or a *guaranteed service* level [26]. The first measure is based on prioritization of traffic. However, under very high loads, prioritization alone might not be sufficient. For example, if all the flows in the network are of the highest priority and collectively they require more resources than are available, these flows will eventually suffer packet loss.

This creates the need for providing a guaranteed service level, which is implemented using a combination of *resource reservation* and *admission control*. Resource reservation works by specifically allocating resources to a network flow, so that as long as the flow stays within the reservation's boundaries, the committed quality of service will be guaranteed. Admission control is the mechanism by which resource reservations are granted or denied. If the resources available in the network are insufficient to allow a reservation to be established, then the admission control procedure will signal that the reservation cannot be established, in order that the committed quality of service can be guaranteed to all the existing reserved flows. If there are sufficient resources, then the required resources will be reserved in the network and admission will be granted.

1.4 Overview of the problem

This thesis focuses on a communication system for both data and VoIP using SIP over a satellite network. The goal in this project is to improve the system's ability to carry VoIP calls. The satellite modem system offers variable bandwidth that can become lower than the capacity required to carry all the calls that need to take place at a point in time. To prevent congestion, resource reservations are setup using RSVP.

In the satellite modem system, each peer is connected via a distinct VLAN. This makes it seem like peers are connected through distinct physical link that each have a separate bandwidth allocation, when in fact all the VLANs on a node share a single pool of bandwidth. The solution presented in this thesis aims to make better use of this pool of available bandwidth to increase the number of simultaneous phone calls, while preventing congestion.

[*] See for example the Wireless Local Area Network in Disaster Emergency Response (WIDER) [16] project.

In the initial state of the system, the bandwidth that can be reserved for phone calls is limited to the minimum of bandwidth that is always available on each of the virtual links, which is only a small fraction of the available bandwidth under good transmission conditions (see section 2.2.1).

1.5 Thesis outline

Chapter 2 presents some previous related work and established standards in related fields such as VoIP, QoS, cross-layering, and resource management. It gives an overview of the specific satellite communication system that the thesis focuses on. Chapter 3 describes the problems caused by the duality of scale in resource management. Some solutions to these problems are proposed in chapter 4, and a decision upon the solution implemented during this project is presented and justified. The design and implementation of the chosen solution is described. Chapter 5 presents the tests and evaluations conducted. In chapter 6 the conclusions and ideas for future work and presented.

Chapter 2

Background and Related Work

This chapter introduces the communication system this thesis project aims to improve and the protocols used in the system. It also presents related work in cross-layering and resource management.

2.1 Voice over IP

There is a high level of interest in Voice over IP (VoIP) technologies in the networking world. One of the main benefits of VoIP is that it allows a reduction in the number of the public interfaces to the network: only an IP network is required, instead of having an additional telephone network interface to the outside world. Several standards have been defined to provide telephony functions to network users, such as the Session Initiation Protocol (SIP) and H.323. SIP has gained a lot of support in recent years.

2.1.1 Session Initiation Protocol (SIP)

SIP [40] is a text-based signaling protocol for establishing, tearing down, and modifying multimedia sessions, which can be of any type, and in particular can be VoIP sessions. We will focus on the use of SIP for VoIP.

A SIP architecture can range from a very simple setup with only a few nodes (*SIP phones*) enabling users to establish calls, to a rather complex system with many functionally different entities. The main SIP entity is the *User Agent*, an end-point for a SIP session. The most common example of a SIP user agent is a SIP phone, implemented either as software installed on a computer (a softphone), or as a piece of hardware similar to a standard telephone, but with an IP interface to the rest of the network rather than a circuit-switched interface. User agents can communicate directly with each other if the network allows them to.

Other SIP entities can provide interesting features to the users. A SIP *proxy* is an entity that makes SIP requests on behalf of the caller, performs policy control ("Is this user allowed to make this call?") and routes incoming and outgoing calls toward their destination. *Registrars* offer location services: the user agents register with this entity, which will then answer requests from proxy servers when they need to discover where to route a specific call. *Session Border Controllers* (SBCs) are intermediary nodes that offer various functionalities. SBCs are described in RFC 5853 [21], and can provide very different services, depending on what is needed: topology hiding, NAT traversal, QoS, etc. SBCs may handle both SIP messages and the media stream.

2.1.2 Real-Time Transport Protocol (RTP)

RTP [41] is a transport protocol for multimedia streams. RTP runs at the application layer, mostly over UDP, but can also be carried over TCP, Stream Control Transmission Protocol (SCTP), or Datagram Congestion Control Protocol (DCCP). RTP is coupled with the RTP Control Protocol (RTCP), which is used to exchange monitoring information about the transmission.

2.2 Satellite Networks

This thesis focuses on a satellite network used to transmit both voice and data communications. While the upper communication layers depend mainly on standard protocols that will be described in the following sections, the transmission system was developed from scratch and designed for military applications. Note that in this thesis we will focus on the use of satellites in an orbit at $36000km$, hence the station to satellite delay will be approximately $240ms$. These types of satellites are called geosynchronous, or geostationary if their orbit is directly above the equator*.

2.2.1 Modems

The satellite modems used are link layer devices. They can be equipped with a variable number of demodulators to receive data from a variable number of peers. To improve the resilience to external conditions, such as bad weather and jamming, the modems use an Adaptive Coding and Modulation (ACM) technique.

The modems are able to transmit to several other modems. The destination modem for an input frame is identified by an IEEE 802.1Q VLAN tag: each destination modem has a specific tag value. The IEEE

* Geostationary satellites have a fixed position relative to the ground, whereas other geosynchronous satellites oscillate with a 24 hours-period.

802.1p Priority Code Point (PCP) field included in the VLAN header is used to identify a queue in the modem [*]. There are three groups of queues:

Premium Guaranteed service. All the packets in these queues will be sent if the destination modem is reachable.

Assured Volume-oriented service. Better than “best effort” service, but offers no guarantee.

Best-Effort Basic service, to the best of what the available bandwidth allows. When bandwidth is insufficient, packets will be dropped.

2.2.2 System

The network nodes are divided into different clusters. Each cluster has up to 32 nodes, one of which is a *Controller Node*. All the other nodes are *Member Nodes*. The Controller plays a particular role in resource allocation within the cluster. A cluster is configured to work in either a *star topology* or a *mesh topology* (see Figure 2.1). In the first case, the member nodes send their data to the hub located at the controller node, which then dispatches the communications to the other member nodes if necessary. In the second case, the member nodes can communicate directly with each other. However, the topology can be a partial mesh, in which case more than one satellite hop will be necessary to reach the destination.

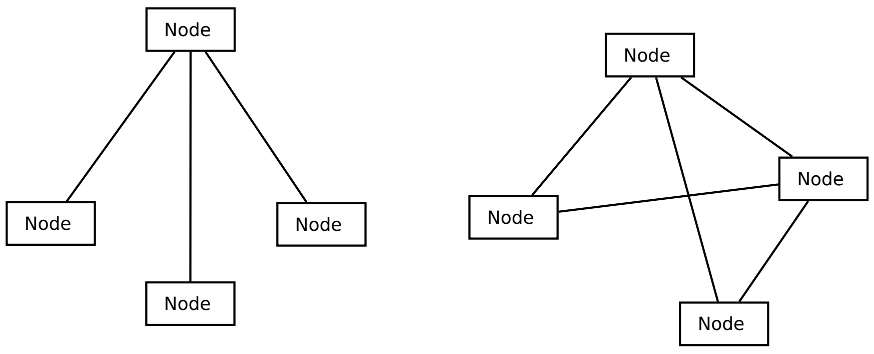

Figure 2.1: Two possible configurations: star (left), and partial mesh (right).

A cluster is allocated an amount of bandwidth (n MHz). All things being equal, the more bandwidth a cluster has, the higher the available throughput. This bandwidth is then shared between all the nodes in the cluster, according to their needs. The Controller Node is in charge of this allocation [†].

[*] details of these standards will be given in section 2.3 [†] This bandwidth assignment method is called Demand Assignment Multiple Access (DAMA)

All the nodes inform the controller of the size of their queues at each resource allocation period [*], and a sharing algorithm is then run to allocate the bandwidth for the next period.

During a planning phase, prior to the field deployment, the topology is specified and the modems are configured. One of the parameters is the *guaranteed throughput* for the logical links, which are the point-to-point links between two modems. This is the lowest throughput that will be available at any time if the modem is working. If this throughput cannot be achieved between two modems, the link is considered down.

2.3 Standards and protocols for Quality of Service

In this section we introduce the existing standards and protocols that can be used to implement QoS in a network. QoS can be implemented at different layers in a network, and different levels of quality may be offered. This section offers a technical overview of what can be done.

2.3.1 IEEE 802.1p and VLANs

The IEEE 802.1Q standard [24] describes VLAN Tagging and allows a single physical network, such as Ethernet, to be segmented into a number of virtual networks, all of which are logically independent. This standard defines a new header (see Figure 2.2) to add to the frame header. In the case of Ethernet, this header in placed before the Ethertype field. VLANs can also be used with Logical Link Control frames (LLC, defined in IEEE 802.2 [23]), in which case the IEEE 802.1Q header is inserted between the lower link layer header and the LLC header.

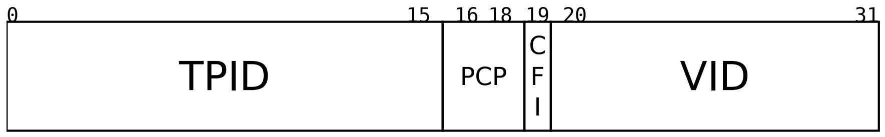

Figure 2.2: IEEE 802.1Q VLAN header

TPID Tag Protocol Identifier, 16 bits. Always set to 0x8100, identifies a 802.1Q frame.

PCP Priority Code Point, 3 bits.

CFI Canonical Format Indicator, 1 bit. Always set to 0 for an Ethernet switch.

[*] This period is $500ms$.

VID VLAN IDentifier, 12 bits. Contains the VLAN tag.

The PCP field contains the IEEE 802.1p priority, with values between 0 and 7 describing 8 possible classes of service. The meaning for each value is not defined by the standard, so the interpretation is to be chosen by each network administrator. The IEEE has proposed in [24] a mapping of these values to specific traffic, as shown in Table 2.1.

Table 2.1: PCP values proposed by the IEEE (Table G-2, page 282, in [24])

Value	Signification
1	Background
0	Best Effort
2	Excellent Effort
3	Critical Applications
4	Video ($< 100ms$ latency and jitter)
5	Voice ($< 10ms$ latency and jitter)
6	Internetwork Control
7	Network Control

2.3.2 IP Integrated Services and Differentiated Services

IntServ was developed by the Internet Engineering Task Force (IETF) in the mid-1990s as an end-to-end QoS solution which provided guarantees for each application. It relies on the RSVP protocol, described in the next section, to signal resource reservations along the path followed by the flow. Three *service classes* (required levels of quality of service) are possible: guaranteed service, which is a hard guarantee [47]; controlled load, in which the flow is handled like best effort traffic in a lightly loaded network [42]; or best effort, i.e. standard IP service.

However, IntServ has several major problems. Guaranteed service causes the network to be underused, and induces additional work for the routers since the routers each need to manage a distinct queue for each flow. Controlled load does not provide the same level of QoS but limits these drawbacks. In both cases, for each flow, a state must be maintained in each router along the path. This can amount to an overwhelmingly large amount of state for a large number of flows for a router in the core of a large network, and is therefore *not* a scalable approach.

Because of these problems, IntServ is not actually used in large-scale networks. Differentiated Services were introduced in the late 1990s as an alternative approach to solve the QoS problem. Instead of handling each flow separately, flows are divided into classes. Each class is handled according to a configuration defined by the network administrator. Since policies are defined by each administrator, when a packet leaves the *domain* from which it originated (a network managed by a single authority), no quality of service can be guaranteed [*]. However, agreements [†] can be reached between two networks so that a traffic flow is offered some quality of service in the next network. A particular class of service is marked in each corresponding packet using a Differentiated Services Code Point (DSCP) [34]. This DSCP is a part of the Type of Service field in the standard IP header (see Figure 2.3) [35].

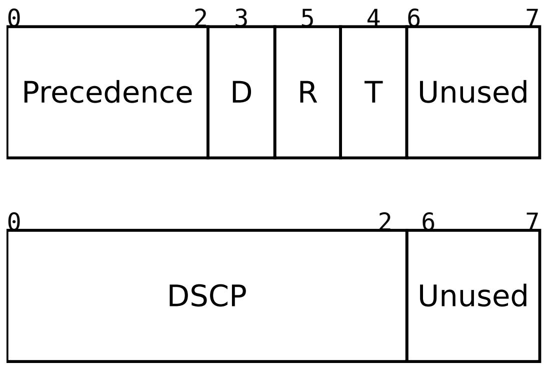

Figure 2.3: Type of Service field from the original IP header (top, with the *Precedence* field and the *Low Delay* (D), *High Throughput* (T), and *High Reliability* (R) bits), and DSCP (bottom) redefinition of this field

2.3.3 RSVP: Resource ReSerVation Protocol

RSVP was introduced as the signaling protocol for the IntServ QoS standard and is defined by the RFC 2005 [9]. Additionally, some types of objects contained in RSVP packets are defined in RFC 2210 [47]. RSVP is used to establish resource reservations and flow states in a network. RSVP signaling includes a description of the flow for which the reservation is established, in terms of the source and destination IP addresses and port numbers for UDP or TCP. It also includes the definition of the reservation, in which the most important information is the bandwidth and peak data rate for the flow. When the reservation is established, the protocol, source and destination IP addresses, and source and destination port numbers allow each router along the path to detect the packets belonging to a particular reservation and enqueue them according to their flow, the queue being configured according

[*] This could be a problem if you are a service provider and are selling a service with a specific QoS guarantee to some of your clients and they realize that you cannot provide the expected service outside of your own network. [†] These agreements are called *Service Level Agreements* (SLAs).

to the reservation request.

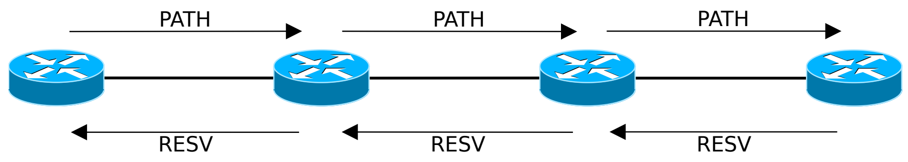

Figure 2.4: RSVP reservation process

A reservation is established using the following process – as shown in figure 2.4: the sender node sends a PATH message containing the information necessary for the reservation to the receiver node. A key element in PATH messages is the flow descriptor. A flow descriptor is the combination of a flow spec object with a filter spec object. The flowspec is used to match specific traffic to a reservation, while the filter spec defines the QoS to be provided for the corresponding flow. Along the path followed by this message, which is determined by standard IP routing, all the routers initialize a new state corresponding to this specification. If a router does not have enough reservable bandwidth to establish this new flow, it sends an error message to the previous node, thus providing *admission control* features. Otherwise, the path continues to be established downstream. When the PATH message reaches the destination node, it generates a RESV message. While the PATH message only traces the path that will later be followed by all the packets belonging to the flow, the RESV packet establishes the actual reservation. RESV packets are sent upstream, from router to router, according to the path set up with the PATH messages. At each hop, the router forwards the RESV packet to the node from which it received the PATH message matching this RESV. When the RESV packet reaches the sender node, the reservation is established.

2.3.4 Multiprotocol Label Switching (MPLS)

MPLS is a layer 2.5 protocol that allows the creation of circuits similar to ATM virtual circuits, independent of the capabilities of the lower layers. In particular, this protocol makes it possible to create a circuit to carry IP packets over any underlying network*. Additionally, the path followed by the circuit can be chosen to be different from the path that the IP packets would follow using standard routing in the network. MPLS is not a standalone protocol. MPLS takes care of the “data” traffic, but requires a distinct control and signaling protocol, such as LDP or RSVP-TE, to establish these MPLS tunnels.

* For networks that do not offer frame handling, such as optical networks with multiple wavelengths, a generalized fork of MPLS – called GMPLS – can be used. See for example [43].

The MPLS header contains the following fields (see figure 2.5):

Label Value 20 bits label, identifying a specific Label-Switched Path (LSP) on a given link.

Traffic Class 3 bits field to specify a QoS value.

End of Stack (EoS) 1 bit field. When set to 1, the current label is the end of the stack.

TTL 8 bits field specifying the Time To Live (TTL) for the packet, similar to an IP TTL.

MPLS labels can be *stacked*, which means that two or more labels can be prepended to a packet. Only the first (outer) label will be considered during switching. The `stack` field identifies the last label of the stack – the first label that was added to the packet.

An example MPLS network is shown in figure 2.6. An MPLS path is called an LSP. It is set up between two Label Edge Routers (LER). The intermediate nodes are called Label Switching Routers (LSR). When an IP packet that should travel through a specific LSP reaches the LER, an MPLS header is added before the IP header. The headers to be prepended are determined according to the Forwarding Equivalence Class (FEC) this packet matches. From now on, the IP header is no longer considered when forwarding until the packet reaches the other LER. This is why LSPs are called "tunnels".

At each LSR, a *Next Hop Label Forwarding Entry* (NHLFE) matching the MPLS label for the incoming packet is looked up [39]. It contains the next hop for this packet, the label stack operation: either remove the label, change the label, or change the label & prepend a new one. The most common operation for an intermediate LSR is *label switching*: the label value is *swapped* to the outgoing label, the TTL is decremented, and the packet is sent on the outgoing interface. At the last LSR of the path, the label is *popped*. If there is no MPLS label left, the packet will then be forwarded according to the layer 3 protocol, for example IP forwarding.

MPLS was designed to improve network performance. Label switching was supposed to require fewer resources on the router than IP routing.

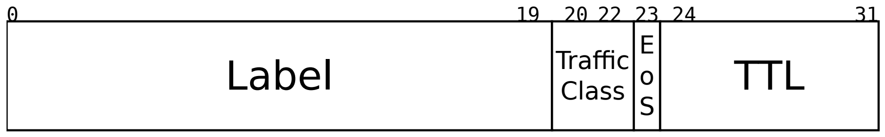

Figure 2.5: MPLS header

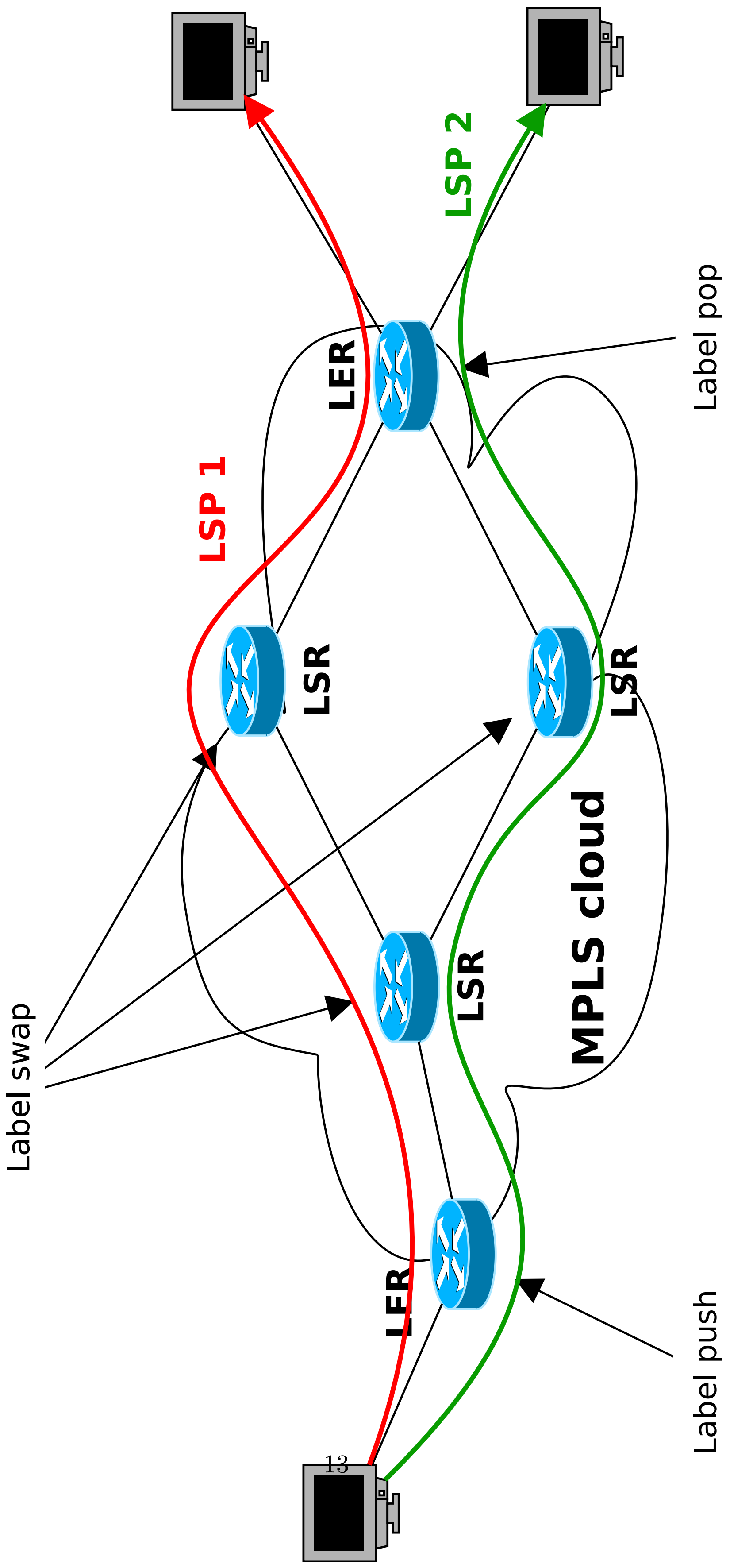

Figure 2.6: Example MPLS network

However, due to the increase in processor speed, this is not, in practice, a reason to deploy MPLS in a network these days. The primary reason for using MPLS today is to create Virtual Private Networks (VPN) and to provide some traffic engineering services.

2.4 Traffic Engineering

Traffic Engineering (TE) is a set of measures aiming at optimizing performance and resource utilization in a network, and improving communication reliability. These measures are implemented through the following mechanisms:

- flow identification according to some specific criteria;
- resource reservation on end-to-end paths;
- different priority levels;
- preemptions; and
- source routing and explicit path specification by the sender.

2.4.1 TE attributes

Dynamic routing algorithms rely on information about the network topology to compute the shortest path to any known subnet, and in particular the bandwidth available or some other metric for each link. The specific metric used for computing the best path can often be configured in the routers by the network administrator.

TE aims to improve performance and increase resource utilization, hence some new aspects of routing and forwarding need to be considered. With a standard dynamic routing algorithm, the shortest path to a node will first be computed, and then all the traffic to this destination will follow along this path. If too much traffic is sent, the link will eventually become congested, which will cause packet loss and increased transmission times. A redundant link could be available and sit unused because of a higher weight, when it would in fact be a better choice for part of the traffic. There are dynamic routing protocols which enable the use of multiple paths to a destination concurrently, such as Cisco's IGRP [10] and EIGRP [12] [11].

If we want to optimize the utilization of network resources and increase QoS for some traffic, we need to take a holistic view of the network in order to find more global optima rather than local optima. To achieve a more global optimum we utilize TE to measure resource usage, reserve resources, and manage the overall state of the network. A fraction of the bandwidth available on a link is marked as being *reservable*, and depending on the established reservations, the reserved bandwidth for each level of priority can be specified.

A new parameter is introduced to identify links based on administratively-chosen criteria. Depending on the document, this parameter is called either *affinity* (in RSVP-TE), *class color* (in MPLS-TE), or *administrative group* (in OSPF-TE). These three protocols will be presented in the following sections of this document. The parameter is actually the same parameter in these different protocols, and is coded in a 32-bit field. The significance of the bits is determined by the network administrators *. This parameter can play a role in the path decision mechanism described in section 2.4.4.2.

2.4.2 MPLS-TE

RFC 2702 [4] briefly describes TE in general, and how MPLS can be used to implement it. In this section, we will describe how MPLS with traffic engineering extensions (MPLS-TE) works.

In MPLS-TE, a single LSP can be configured with several alternative paths, which can be either automatically computed or chosen by an administrator. In the later case the paths may be completely defined, so that all nodes between the source and the destination are specified, or partially defined, in which case the rest of the path will be automatically computed.

MPLS-TE also defines the affinity attribute – without specifying any format. Such an attribute could be used to include only links with a specific value, or to exclude some links. The default policy would be to allow any link if no affinity information is specified.

MPLS-TE also offers several capabilities that can prove useful in the context of TE, such as the re-optimization of LSPs when a better path becomes available, or load-sharing between LSPs with the same source and destination but using different links.

In RFC 3036 [1], a protocol called Label Distribution Protocol (LDP) was defined. Its goal was to distribute labels to establish LSPs. However, as indicated in RFC 3468 [2], the IETF MPLS Working Group decided to "focus on RSVP-TE as signalling protocol for traffic engineering applications for MPLS". Thus, in practice, LDP is not used.

2.4.3 RSVP-TE

RFC 3209 [3] defines *Extensions to RSVP for LSP Tunnels.* As per RFC 2205 [9], RSVP allows some kind of resource reservation in a network, but some information is lacking to establish MPLS tunnels.

Four new essential objects were introduced in RFC 3209, and three others are modified to convey MPLS-specific information instead of information required to identify the flows, as in IntServ. In particular, the original

* The set of bits can be used as a bit-field or any other definition that the administrators choose.

RSVP protocol needed to be extended to allow label binding between any pair of nodes along the path. This is done by the `LABEL_REQUEST` and `LABEL` objects. The `LABEL_REQUEST` is sent downstream in the PATH messages. In the RESV message for the same session, the downstream node will put a `LABEL` object, containing the label it will expect for this tunnel, on this particular link. The upstream node saves this label in its NHLFE.

The third essential object defined in RFC 3209 is the `EXPLICIT_ROUTE` object. This object encapsulates a sequence of sub-objects, each of which represents a node, either by an IP (IPv4 or IPv6) prefix or an Autonomous System number. Several intermediary hops may be needed between two successive nodes enumerated in the `EXPLICIT_ROUTE` object. This object allows the specification of a path totally different from what IP routing would produce.

The last new object is `SESSION_ATTRIBUTE`. It carries the priorities related to the tunnel being set up, some flags, and a human-readable name for the session. The priorities allow an established reservation to be preempted by a new reservation with a higher priority when the reservable bandwidth becomes insufficient. Priorities are coded in 3 bits, ranging from 0 to 7, where 0 is the highest priority and 7 the lowest. If a new reservation at a low priority is required on a link with insufficient bandwidth, this reservation will be rejected unless it is possible to obtain enough bandwidth to satisfy the demand through preemptions. To compute preemptions, the *setup priority* of the new reservation is compared with the *hold priority* of previously established tunnels.

2.4.4 OSPF-TE

OSPF-TE is one way to implement TE in a network. IS-IS-TE, the set of TE extensions to the IS-IS routing protocol, can also be used to achieve the same goals.

2.4.4.1 Introduction to OSPF

OSPF (Open Shortest Path First) is a standard intra-autonomous system routing protocol used in the Internet. OSPF is described in RFC 2328 [32]. It allows the dynamic configuration and propagation of routes in a network. OSPF works by having the routers exchange messages containing routing information so that all the routers in an area know the whole topology of the particular area they find themselves in. Knowing this topology, each router runs Dijkstra's algorithm on its copy of the graph to fill its routing table.

The messages exchanged between OSPF nodes are called Link State Advertisements (LSAs). They are used to inform other routers about the topology: routers attached to a subnet and subnets connected to the routers.

2.4.4.2 Traffic Engineering Extensions to OSPF

OSPF is a good protocol to compute IP routes and offer a best effort service. However, it does not consider resource usage and congestion, nor does it offer any TE functions. These functions were added in RFC 3630 [27].

OSPF-TE is based on Opaque LSAs, which are described in RFC 2370 [13]. Opaque LSAs are specific messages distributed according to the standard OSPF mechanisms used for the transmission of LSAs. They can distribute data used directly by OSPF itself, or data to be used by some other application [*].

In OSPF-TE there are two types of additional LSAs. One contains the *Router Address* for a node, which is an address that will be reachable as long as the router is connected – typically, an address set on a loopback interface[†]. The other type of OSPF-TE LSA is the *Link Information* LSA. It describes the TE characteristics for the link: bandwidth, reservable bandwidth, reserved bandwidth for each priority, and *link color.*

2.4.5 Constrained Shortest Path First (CSPF)

Constraint-based routing is briefly defined in RFC 2702 [4]. Unlike standard routing algorithms which aim at providing the shortest possible path to a destination, constraint-based routing algorithms need to compute a path between two endpoints that satisfies some requirements over a set of attributes. These algorithms start by pruning all the links that do not have the required attributes, and then run a shortest path algorithm on the remaining topology.

CSPF was documented in an expired IETF draft [31]. The CSPF algorithm relies on a database obtained through OSPF or IS-IS with TE extensions. CSPF provides an explicit path between the specified endpoints, with the required characteristics.

2.5 Cross-Layer techniques

According to both the OSI and TCP/IP layered models, the network functions are split into several independent layers. This allows for greater flexibility, since the role for each layer is clearly defined and the interfaces between layers are specified. Functional separation, modularity, and specified communication interfaces between the modules are considered to be good practices when it comes to designing complex systems such as complete networking stacks. A layer N protocol can run over any layer N - 1 protocol transparently. IP does not care whether the underlying link

[*] For an example of this use by other applications see [37]. [†] A loopback interface is used because it is reachable regardless of which one of the multiple links of this router is connected. This is described in paragraph 2.4.1 of RFC 3630 [27].

it uses is Ethernet or not, or if the media is a copper wire, a wireless link, or a free space or fiber based optical link. TCP works the same way over IPv4 or IPv6 networks, or any other network-layer protocol. The same TCP implementation can be used in all these cases, and TCP is not even aware of the differences in the lower layers – except for details that impact TCP itself, such as MTU, throughput, loss rate, and round-trip delay.

However, this layer separation has a cost. Due to the lack of communication and interaction between layers, the upper layer protocols cannot take advantage of specific features of the lower layers. Cross-layer techniques, that allow use of knowledge about another layer to influence how one layer works, can be designed to dynamically optimize the configuration and function of the system.

Srivastava and Motani [44] propose a definition of cross-layering as a *violation of a reference layered communication architecture.* They classify each of the possible cross-layer designs into one of four groups. The first group involves the creation of new interfaces, either reinjecting information from an upper layer into a lower layer (*downard interface*), from a lower layer into an upper layer (*upward interface*), or *back and forth* – a combination of both upward and downward interfaces. The second group is a set of techniques in which two or more adjacent layers are merged and offer standard interfaces to the upper and lower layers. The third category involves replacing an existing layer with a new one, designed specifically to take advantage of the particularities of the other "crossed" layer. No new interface between layers is added. *Vertical calibration* techinques is the last group described, and involves adjusting parameters at several layers, either statically or dynamically. Additionally, some implementation ideas are proposed. The cross-layering could be implemented through different methods:

- direct communication between the layers;
- a database containing information shared between all the layers involved in the cross-layering; or
- by creating a new, unlayered, overlay model.

Kawadia and Kumar [28] remind us of the many issues and dangers when designing a cross-layer system. Among these issues is the loss of isolation between the protocols. There is therefore a risk for bad design, often refered to as "spaghetti design". The search for performance optimization is a short-term goal, whereas an architecture is a long-term quest, and should not be neglected in favor of immediate payback. Additionally, they warn us about unintended consequences. A cross-layer system designed with a specific goal in mind could in fact have a negative impact on the performance of the system, at least in some cases. Moreover, a cross-layer technique could interact with a new technique being developed, and may create loops between them that could negatively impact stability of the network. The designers should therefore maintain a holistic view of the network on which they are working.

Ojanperä [33] describes a cross-layer architecture to optimize video streaming to multi-homed wireless devices that handles handovers between networks, and takes advantage of the specificities of video encoding, such as the relative importance of frames and bitrate adaptation, and improved utilization of multi-access networks through better handovers and load-sharing across different networks paths available simultaneously. The architecture is based on *cross-layer signaling* to exchange and distribute information and *cross-layer controllers* that take decisions.

Some effort has been put into this field in the last years, as the number of research articles shows. However, most of the research seems to focus on the lower layers of the network, specifically cross-layering between the physical and link layers – in wireless networks, and does not apply to this thesis. Additionally, this thesis aims to improve an existing commercial system, specifically the Thales Communications Modem 21e. Current research topics can provide interesting insights into the subject, but the ideas they propose are often too theoretical or too far from a practical solution to be applied directly by industry. The study of these research papers provided some interesting ideas and insights, and the warnings offered by the authors need to be kept in mind during this thesis project.

2.6 Management of resources

When network resources are limited, it is necessary to watch carefully how they are used and shared between the users. Additionally, resource usage by in-band signaling protocols, i.e. control protocols that share the available bandwidth with the users of the network, must be kept as low as possible so that these protocols do not reduce the bandwidth actually available to users excessively, while maintaining network functionality.

2.6.1 Centralized and decentralized systems

There are two main approaches when it comes to management of resources shared between a set of "users": centralized management and decentralized management. Additionally, the resource management system can be all-knowing – aware of all the needs and uses within the group – or have a partial view of the situation. The resource management system can provide different levels of performance, involve different difficulties of implementation, and require more or less exchange of information between the users. The resource to manage in the context of this thesis is bandwidth in the network. The decisions to be made concern admission control and preemptions.

In a centralized system, one node – the *controller node* – is in charge of handling all requests and managing all the resources available. This node is therefore aware of the complete situation. This complete knowledge enables the controller node to make the best possible decision during the allocation phase. However, keeping the controller's database up to date can be a problem, as this may require extra traffic or a complex protocol design. Additionally, sending all the reservation requests to a centralized server induces a delay of one trip to the controller.

The decision system can also be decentralized. Each node could decide for itself to make a reservation or not, relying on the information that is currently available to it. The amount of information that is currently available could range from full knowledge of the complete system state to a local view. For example, knowing only about the available bandwidth and established reservations on this local node. In the first case, the decision could be the same as in the centralized case, provided the synchronization mechanism distributes knowledge to all the nodes. Such a synchronization could require more information exchange, perhaps via a dedicated protocol. An example of such a protocol is OSPF, which uses messages to distribute local knowledge, enabling each node to build a global view of the network topology. In the second case of only local knowledge, the decision would likely be sub-optimal: if the decision process is not aware of the reservations made by other nodes, preemptions can only be made of reservations made by the local node rather than preempting the lowest priority within the whole bandwidth-sharing area.

2.6.2 Frequency of the control loop

The protocols that make the network work consistently – routing protocols, keepalive methods, and signalling protocols – use refresh mechanisms to maintain an up-to-date view of the network and flows. The frequency at which these refresh mechanisms operate can often be configured to a value different from the default, in order to improve performance in specific cases.

Increasing the frequency can provide better reactivity (positive effect) to the events in the network – a link going down, for example – but causes an increased load in the network and on the nodes (negative effect). Conversely, a lower frequency limits the load caused in the network by the signalling protocols, but the resulting reactivity is not as good. Moreover, setting an update period lower than the usual evolution period over the network causes unnecessary overhead and would not significantly improve the service quality.

Chapter 3

Presentation of the Problem

This chapter gives more detail on the system and describes the problem that this thesis project aims to solve.

3.1 Network Architecture and Mechanisms

The entity deployed in the field is a *station*. It can have either one or more modems, each of which participates in one cluster. A station can be equipped with a number of computers and SIP phones. These equipment are plugged into a standard commercial router, itself connected to the modem(s), and any other link(s) that might be available.

A SIP Proxy/Registrar is added to each station to provide call-control features. Outgoing calls, to SIP phones outside the local station, are forwarded through a local *Session Border Controller* (SBC). This SBC acts as a QoS intermediary. When a local user wants to call a remote user, the following sequence takes place:

1. User dials the remote SIP UA's number
2. The SIP phone initiates a connection to the proxy
3. The proxy detects a remote number, then routes the call through the SBC
4. The SBC detects the need for a reservation of resources
    (a) The CSPF algorithm computes the route for the reservation (from the local station to the remote station)
    (b) A reservation request is sent using RSVP
5. SIP signalling is sent to the remote SBC
6. The remote SBC starts to establish a reservation (from the remote station to the caller's station)

    (a) The CSPF algorithm computes the route for the reservation
    (b) A reservation request is sent using RSVP

7. Reservations are established in both directions

8. The SIP signalling is routed from the SBC to the proxy (remote node) and to the remote phone

9. The remote phone rings

This sequence is documented in RFC 3312 [19]. It has a major drawback: if the callee does not answer the call, or if the number is invalid, the reservations have been established needlessly, and may have caused unnecessary preemptions. However, the order of the steps in this sequence prevents the occurence of a situation where the callee picks up his phone and the call is cancelled just as he answers the call because of a lack of bandwidth – and the "this system does not work" reaction.

When the call ends, the reservations are torn down by the SBCs on both sides.

With the current system, reservations are only allowed in the *guaranteed* portion of the link's bandwidth, as defined in section 2.2.2.

3.2 Overview of the problem

The end-goal of this master's thesis project is to provide a significant improvement in the QoS perceived by the user of a VoIP service running over links to geosynchronous satellites via Thales Communications Modem 21e modems. This improvement in QoS will be measured in terms of the number of simultaneous voice calls. Other metrics could include quality of transmission (packet loss, delay, and jitter) and resilience to changes in the environment (weather, jamming). These metrics were not measured during the course of this project.

The system must be able to simultaneously transmit both VoIP and data communications, which means both real-time and non-real-time streams. This system should also be able to support different levels of priority for each type of traffic. Some non-real-time streams could be more important than other real-time streams, thus a low-priority voice call should, ideally, leave room for a high-priority data transfer. Since basic data flows do not use reservations, in part because of the nature of these flows, which often consist of bursts of data rather than a continuous bandwidth requirement over the duration of the communication, enforcing such a priority policy is a problem that requires careful consideration of the whole priority system.

The bandwidth available to one node may vary in time (see Figure 3.1), but never drops below a minimum guaranteed value (section 2.2.2). This guaranteed bandwidth is usually set at a low value, to provide basic communication capability even under adverse transmission conditions. When only one node needs to communicate, it can be allocated a large amount of bandwidth. During jamming or in case of bad weather, the available bandwidth can change quickly. If several nodes want to communicate at the same time, the bandwidth must be shared among them.

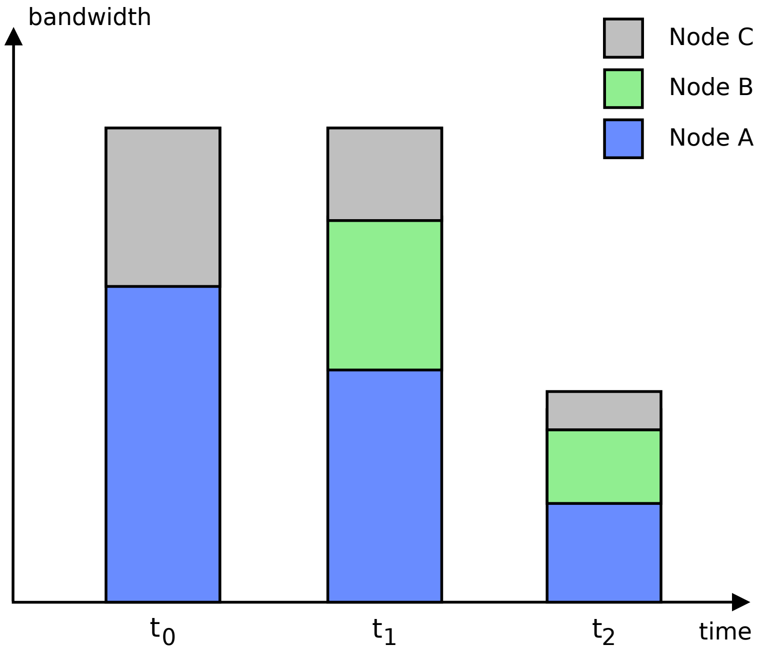

Figure 3.1: *Bandwidth allocated to nodes A, B and C at different times.* The bandwidth allocated to one node depends on its own needs and the needs of other nodes in the network. Initially (t_0) only A and C are sending data, hence they share all the available bandwidth. When B needs to send data at t_1, the bandwidth available to A and C shrinks. If the transmission conditions become worse, the global pool of bandwidth is reduced, as occurs at t_2.

Additionally, as noted earlier, the modems allow the nodes to request more bandwidth if they need it. This need for additional bandwidth will often be related to an increase in the number of simultaneous voice calls. The path for these calls will be computed at the source. However, the only information available to the source-routing algorithm is the bandwidth that is available at the moment, or rather, when the last OSPF-TE message was sent. This would be fine in a system that does not allow bandwidth sharing between the nodes ("node A always has bandwidth a" and so on), but since the bandwidth allocated to a node is related to its needs, this could cause problems. More bandwidth than has been allocated might be available for this node, either because other nodes are transmitting low priority data, or because no one is communicating, but if the source-routing algorithm is not aware of this potentially available bandwidth, it might not allow the link to be used. When several alternative paths are available, lack of detailed knowledge about the precise current utilization of a link could cause a link to be eliminated from consideration when it could in fact be used, thus unnecessarily overloading some other parts of the network (Figure 3.2).

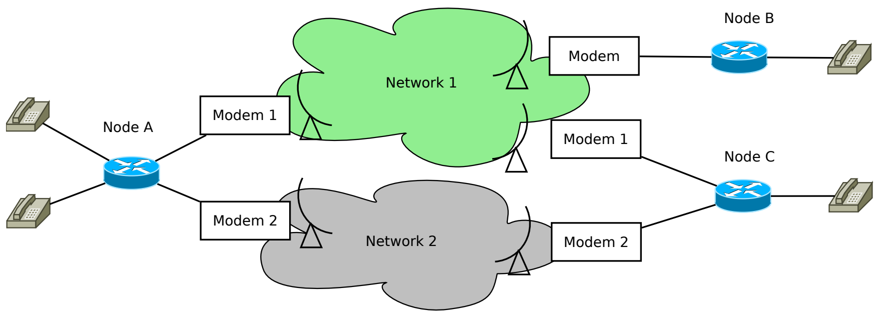

Figure 3.2: *Example of a possible configuration.* With two possible routes from A to C, not being aware of bandwidth sharing in network 1 could cause all traffic between A and C to go over network 2

Among the reserved bandwidth – for voice over IP – preemptions can occur. One goal would be to avoid preempting other reservations needlessly, while still allowing higher priority reservations to be established. Preempting communication via a link when the node could have been allocated sufficient bandwidth is an example of unnecessary preemption (Figure 3.3).

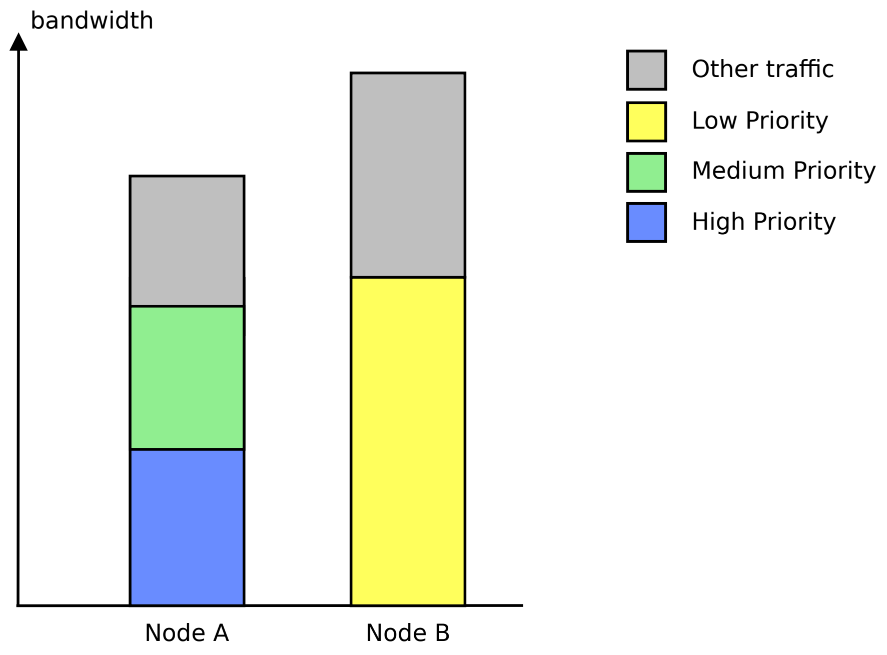

Figure 3.3: *Bandwidth allocated to nodes A and B and reservations.* The bandwidth actually available for more reservations is the gray band, but for a new high priority reservation, the available bandwidth is the sum of all gray, green and yellow band. However, if the reservation can be established using only the gray band, this is desirable, as it prevents preemption.

Chapter 4

Presentation of the Ideas Considered and Implemented

During the course of this thesis project, a number of ideas were investigated. These ideas are presented in this chapter. The solution that was implemented is the RSVP snooper, which is described in section 4.2.

4.1 Ideas examined and rejected

This section introduces the various ideas that have been spawned from reflections on the issues presented in section 3.2. Additionally, one needs to keep in mind some characteristics of this system:

Coupling of pairs of reservations When a user from the node A calls a user on the node B, two resource reservations are setup: one from A to B, and one from B to A. They should never exist independently. When one reservation is preempted, its source detects this preemption and tears down the other reservation. This means that the bandwidth for *both* reservations is freed. The corresponding SIP session is also shut down with the appropriate messages.*

Quantized chunks of bandwidth Each communication session uses a fixed amount of bandwidth in each direction.†

Multi-hop calls Sometimes, the callee is not directly within reach of the current node, for example because this destination node belongs to a different cluster, or because direct communication between the two stations cannot be established within the cluster, as would be the case in a star topology. Although the conversational interactivity of these

* This is only the case for bi-directional communication systems. In a video broadcasting system for example, only downstream reservations are required. † Again, this is the case for this particular system, but if video-conferencing was implemented this might not be the case.

calls is not as good as for single-hop calls, multi-hop calls can occur. In this case, a single reservation can use resources several times in the same cluster[*]. This needs to be considered for resource allocation.

Adaptive Coding and Modulation (ACM) The coding/modulation couple could change from node to node, and on a single node, from direction to direction. Thus, we may need to consider allocating some kind of "tokens" instead of bandwidth measured in kb/s, B/s, or any other unit. These tokens could be frequency bands (Δf in units of MHz).

Delay for resource allocation The central resource allocator for a cluster needs some time to allocate more resources to a modem that suddenly receives a flow of data for transmission.

Centralized resource allocation at the physical layer The frequency bands for each node, as well as the coding and modulation, are selected centrally by the resource allocator. One has to wonder whether it is consistent to define a completely distributed resource reservation system when the system does centralized allocation of physical resources. Moreover, there could be stability or availability issues with the central allocator. Additionally, if a central reservation system is selected, does it make sense to locate it at some other node, rather than co-locating it with the physical resources allocator?

Network load The bandwidth allocated to a node may be very low at times ($64kb/s$). Therefore, we do not want to overload it with signaling and network protocols, as the goal is to carry user data, information exchanged between the nodes has to be somewhat limited. Moreover, most reservation problems will occur when the available bandwidth is low [†], but we particularly need to limit the signaling protocols during these periods of limited available resources.

CPU load The central resource allocator already has an important role and a complex algorithm to run during each allocation phase. If another centralized function needs to be performed by this node, the available computing power will be quite limited. A hardware upgrade could solve this family of problems, but is not always feasible.

Special use of MPLS-TE tunnels MPLS-TE tunnels were designed as traffic trunks. In this particular system, however, each tunnel is assigned to a single reservation. Therefore, they have the following characteristics:

[*] For example, in a star topology: from the source node to the hub, and from the hub to the destination node [†] For example, if the available bandwidth for a node is $1Mb/s$, it should be possible to establish a dozen or more VoIP sessions.

- No increase or decrease in the bandwidth reservation;
- One pair of tunnels for each communication, instead of several similar sessions in each tunnel;
- Short-lived reservations and tunnels;
- A single path for each tunnel; and
- No tunnel rerouting: if one of the links fails, then the tunnel goes down and the communication is interrupted.

4.1.1 Current situation: no sharing

By allowing only a well-defined portion of the bandwidth allocated to a modem to be reserved, the bandwidth sharing problem is avoided: a fixed amount of bandwidth, labeled premium bandwidth, as defined in 2.2.1, will always be available for each direction as long as the peer node can be reached, and is marked as the only reservable bandwidth.

This solution is highly sub-optimal, as explained in section 3.2.

4.1.2 Fair repartition

Another way to solve the sharing problem is to split the reservable bandwidth fairly between the destinations. If a modem is allocated a bandwidth B, it could share this bandwidth between N output directions or neighbors. Each direction could be allowed to make reservations up to B/N.

This solution is still sub-optimal, since one station may need many simultaneous voice sessions at some point in time, while no other station has an on-going session.

Another fair repartition scheme could be defined by assigning different weights to the nodes. If the node i has the weight w_i in the repartition scheme, it would be allowed to make reservations up to:

$$B_i = \frac{w_i}{W} B, \text{ with } W = \sum_{j=1}^{N} w_j$$

This scheme may not provide much improvement. If one station needs to supports many simultaneous calls part of the time, and another station has the same requirements only when the first one does not place calls, we cannot find a fixed weight repartition that will allow both stations to make the desired calls (see figure 4.1).

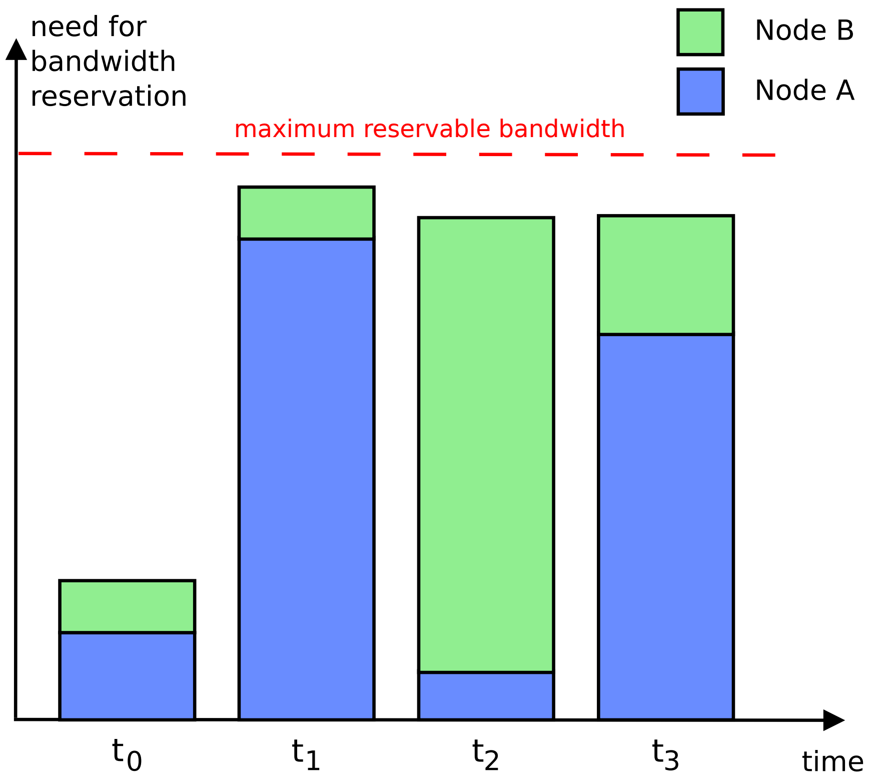

Figure 4.1: Possible evolution of bandwidth need for two nodes A and B over time. If both A and B require a large part of the reservable bandwidth alternatively, a fair repartition cannot satisfy the needs of the stations.

4.1.3 Computing the reservable bandwidth

Provided the central resource allocator is informed of the bandwidth used by the reservations in the whole cluster, it could compute an estimation of the bandwidth that is still reservable. This estimation would provide the bandwidth that can be reserved if all the remaining resources were used to transmit only on one link. By using the current state of reservation in the cluster as the input for the resource allocation algorithm, the bandwidth that could be reserved if all the remaining network resources are used on a single link can be determined. For each modem in the cluster, and for each outgoing link for this modem, the algorithm is run with the following parameters:

- Bandwidth requirement for the selected link: maximum possible bandwidth for the link at a best effort priority;
- Bandwidth requirement for any other link from the same modem, currently null; and
- Bandwidth requirement for any link from any other modem – currently reserved bandwidth, with an assured priority.

This way, the algorithm will allocate as much bandwidth as possible to the selected link, while allowing the current reservations to be preserved.

However, there are several issues with this solution. To provide useful information, the computation would need to be performed for each priority level, which would require knowledge of the reserved bandwidth for each priority in the cluster. This information could be obtained from the OSPF-TE messages or by designing a new signaling protocol. Running the allocation algorithm nine times (8 priorities, plus the actual allocation) for each link requires quite a lot of CPU power.

Additionally, another problem is raised by this solution: excess bandwidth can be reserved. The value computed for each link is indeed the bandwidth that can be reserved if *no other reservation* is setup at the same time. If two reservations are established simultaneously, they could exceed the bandwidth that is actually available in the system.

4.1.4 Solving the overbooking problem

To try to solve the overbooking problem, three ideas are proposed. Each of these ideas is described in a paragraph below.

4.1.4.1 Ghost LSP

The first idea involves a "Ghost" LSP. When an LSP is created in a given direction, the corresponding amount of bandwidth needs to be marked as “reserved” with the same priority in the other directions (figure 4.2). These ghost LSPs always go one hop along the path to one of its neighbors*, and are never actually used, as they do not carry any data. Marking the bandwidth as “reserved” is easy. However, preempting and tearing down reservations becomes more difficult, as we have to make sure that all the ghosts have been cleaned up to avoid the system entering an inconsistent state. Additionally, detecting that a ghost LSP needs to be created from an intermediary node in a multi-hop reservation to each of its neighbors would be difficult, as it would require either some scripting on the routers or sending a command from one node to a router at a remote node.

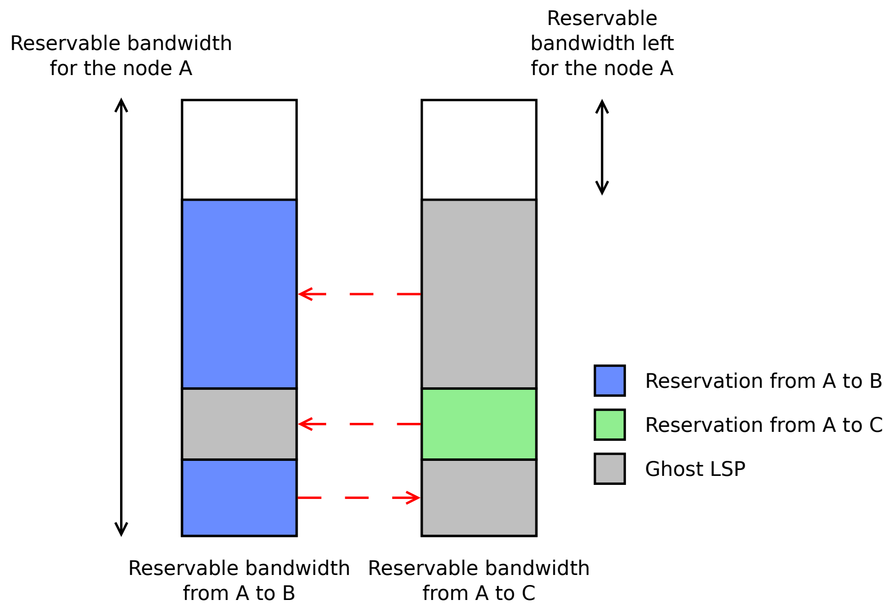
Reservable bandwidth for the node A
Reservable bandwidth left for the node A
Reservation from A to B
Reservation from A to C
Ghost LSP
Reservable bandwidth from A to B
Reservable bandwidth from A to C

Figure 4.2: Real LSPs and their corresponding ghost LSPs

4.1.4.2 Another way to limit the reservable bandwidth

A new LSP priority class could be created, with a (virtually) negative priority†. We can then configure the reservable bandwidth to any value (for example, $10Mb/s$) for each direction. This value, B, is not related to the bandwidth that is actually available. In each of the directions, one LSP is created with the new priority and a bandwidth C, so that the bandwidth that is actually reservable is $R = B-C$. This solution would require frequent updates of the reservable bandwidth, and the signaling protocols could use quite a bit of network resources.

* See the multi-hop reminder at the beginning of this section. † Hence, the priority for the new class is higher than the priority for any real class of LSP

4.1.4.3 Passive approach

Another way to deal with this overbooking problem is to allow the LSP to be set up. When the traffic starts to flow through the new LSP and the modem does not actually have sufficient bandwidth, the queues in the modem will grow. This growth can be detected and we can signal the router or the SBC that it needs to tear down one of the reservations. The problem with such a passive approach is that there is a delay before the problem is detected, during which congestion will occur. During this congestion period, packets will be lost or delayed, which will result in a loss of quality one or more users.

4.1.5 Two routers at each station

By installing two routers at each station, the sharing problem would be solved. The bandwidth sharing problem comes from the fact that each direction is represented by a different VLAN on the link between the router and the modem, and that bandwidth cannot be shared between the virtual interfaces* created for each VLAN on an interface. If two routers are used, the outer router is connected to the modem with the usual VLANs for each direction. These VLANs can be assigned a large value as their reservable bandwidth. The limitation would be configured on the link between the inner and outer routers, which does not require any VLAN.

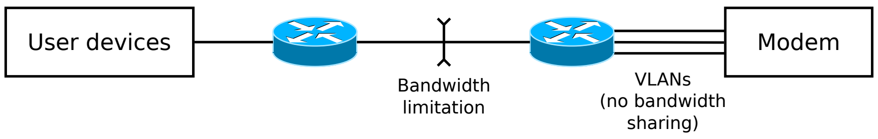

Figure 4.3: Configuration with 2 routers on a node.

This solution is unfortunately physically and financially unacceptable. The hardware for a station would become significantly heavier, larger, use more power, and be notably more expensive. An attempt at realizing this function on a single router has been made, but realizing it has not been feasible.

4.1.6 Getting rid of the VLANs

Since the VLANs seem to be the issue, it is tempting to try to remove them altogether and make a cluster look like a single LAN. With no VLAN between the router and the modem, the bandwidth sharing problem is solved. However, this solution works well only in a topology where all the

* In the Cisco terminology, these interfaces are called “sub-interfaces”.

nodes can reach any other node, such as a standard LAN. In the system this thesis focuses on, the only topology where this assumption can be made is a full-mesh topology with all the links up.

In a star topology or in a partial mesh, at least one node needs to send its data through an intermediary node to reach one of the destinations in the cluster. If we try to make our network look like a single LAN to the routers, they will detect that a packet that came through one interface needs to be routed and leave through the same interface, and they will send an ICMP redirect message back to the sender, as illustrated on figure 4.4. Thus, this solution cannot be considered.

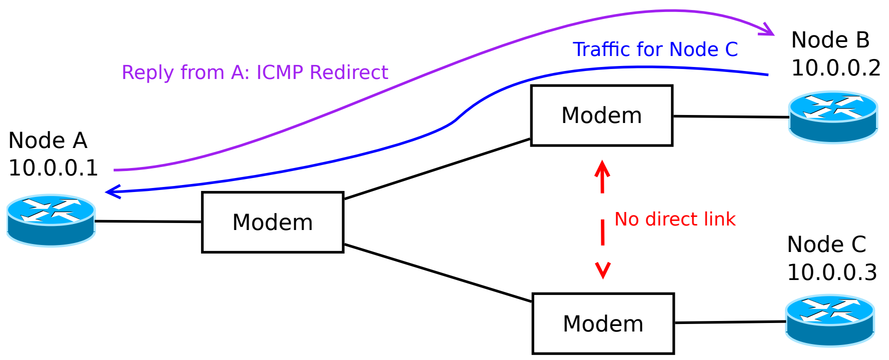

Figure 4.4: ICMP Redirect issue when a star topology is represented as a LAN.

We can try to solve the ICMP redirect issue by maintaining an LAN for each link, as in the original setup, without VLANs *. However, due to the behavior of the modem, which uses the VLAN tags as peer identifiers, we need to implement a bridging feature on the modem's input. The Linux kernel bridge [30][29] provides the desired feature, but a new issue appears with this solution: when the satellite link to a particular peer goes down, the router is unable to detect this event. As a result, the router never decides to route through another path, and the system does not work correctly. Integrating the bridging feature within the modem might solve the issues presented in this section, but was not considered feasible during the time frame of this thesis, as this would require much work on the modem's internals.

* Actually, one VLAN is present because the PCP field, described in 2.3 is required by the modems.

4.1.7 Changing the meaning of the VLANs

As we saw above, the network cannot be considered as a simple LAN. The idea of removing the VLANs to avoid the problem of the bandwidth sharing is still tempting. Another idea would be that each node's router has a single outgoing VLAN, that it would use to send data to all the other nodes, and as many incoming VLANs as needed, on which it would receive data coming from the other nodes and send some signaling messages. This idea is illustrated in figure 4.5. The bandwidth sharing problem is solved directly by the router, since all the reservations leaving the node are made on the same VLAN.

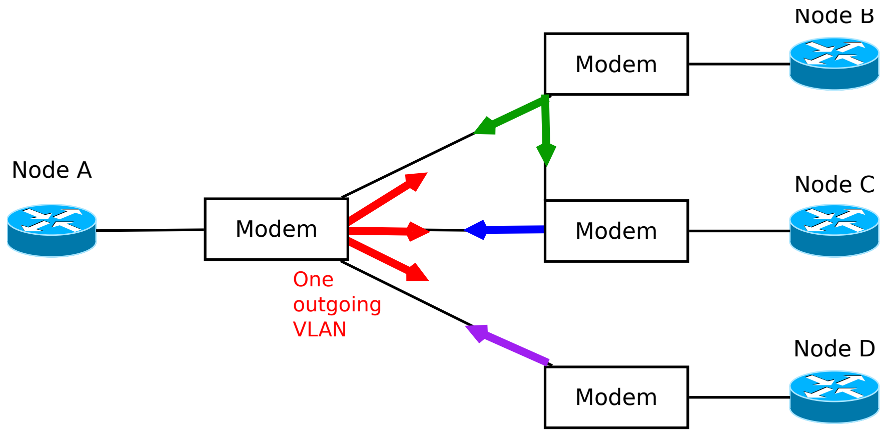

Figure 4.5: One outgoing VLAN for all the directions.

This idea seemed promising, and provided the expected improvements in a simulation environment. However, it provides less flexibility and evolution possibilities than another idea that was also considered, and was therefore rejected.

4.2 Description of the solution implemented

After examining the solutions proposed in the previous section, they all had to be rejected for various reasons: some were financially unacceptable, some would involve dirty hacks that could easily lead to an inconsistent state and an unmaintainable implementation, and some did not promise a significant improvement. Another idea was examined during this process.

4.2.1 RSVP snooper

The bandwidth sharing problem can be considered from a local point of view. In this case, a local solution can be provided by adding a functional entity that would perform admission control for the reservations using the local modem. This entity would need to aggregate the resource requests in all the directions and compare this aggregated value to the reservable bandwidth. Preemptions between the directions would also be performed by the entity. The ideal location for such a function is between the router and the modem. This way, both reservations made by this node to another node and multi-hop reservations for which the current node is just an intermediary node would be processed.

This feature can be implemented by tracking all the RSVP reservations when they are being set up. The RSVP messages contain all the necessary information to perform this accounting: requested bandwidth, priorities, and the data necessary to uniquely identify any reservation. This tracking is mostly passive: as long as there are enough resources to allow all the reservations to be admitted, the agent will simply record the new reservation and let the signaling messages pass through. However, when the bandwidth becomes insufficient, the entity will reject the new reservation, or make suitable preemptions. Since the entity is not a real MPLS-TE router, it has no IP address and is not present as a next hop in the `EXPLICIT_PATH` object. Thus, the agent takes the identity of the next hop* on the path to signal the problem to the upstream node.

This entity, called a *snooper*, is a combination of a partial RSVP implementation that tracks the reservations and spoofs the required RSVP messages to signal the preemptions or admission failures, and a traffic control entity that will do the accounting, decide whether a reservation should be allowed or not, and compute the necessary preemptions. The RSVP snooper is illustrated on figure 4.6.

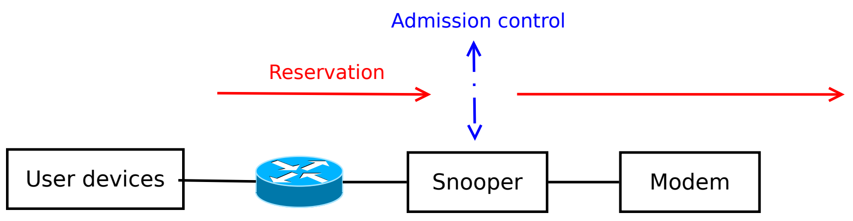

Figure 4.6: Role and integration of the RSVP Snooper in the system

* There will always be such a node, since our reservation has not reached the destination router at the time it passes through the snooper.

4.2.2 CSPF modifications

In combination with some of the previous solutions, and particularly with the RSVP snooper, the path computation algorithm may also need to be modified to take bandwidth sharing into account. The affinity field (see section 2.4.1) can be used to identify a set of links that share the same bandwidth. The CSPF algorithm could then sum the bandwidth reserved for each level of priority (this is obtained via the OSPF-TE LSAs) for all the links that share some bandwidth. The reservable bandwidth for each priority can then be computed for each link, and the usual path computation can then take place, using the accurate reservable bandwidth information that has just been computed.

4.2.3 Reasons for this choice

The snooper solution seemed very promising after an analysis of the features it could provide. The accounting and admission control part can be designed to adapt precisely to the particularities of the current system, but could also be modified to be used in a different system presenting similar problems if necessary. By tracking currently established reservations, it also allows future developments which could request additional resources be allocated to a node, and the flexibility the snooper provides can be used as the basis for further improvements of the system.

4.2.4 System and integration

The snooper entity is inserted in the existing system as an intermediary node, between the router and the modem for the station it belongs to. This is shown on figure 4.7. To avoid direct modification of the existing system, the snooper ran on a distinct physical entity during this project. However, in the future, it could be integrated in the modem itself, or in another computer that constitutes a complete node in the system. This would bring back the cost, weight, and power requirements to their original level.

The snooper has two traffic interfaces: one facing the router and the other facing the modem. These two interfaces are bridged at the snooper. The bridge is implemented with the Linux kernel bridge module. This bridge passes all the packets from one side of the snooper to the other. However, the RSVP packets, must only pass through if they satisfy the admission control procedure. They must therefore be blocked at the bridge in certain cases.

The snooper also has a configuration interface. This interface is dedicated to setting the new reservable bandwidth in real-time. Whenever a new value is entered, the snooper reads it, and if the new value is lower than the bandwidth that is currently reserved, some preemptions occur to

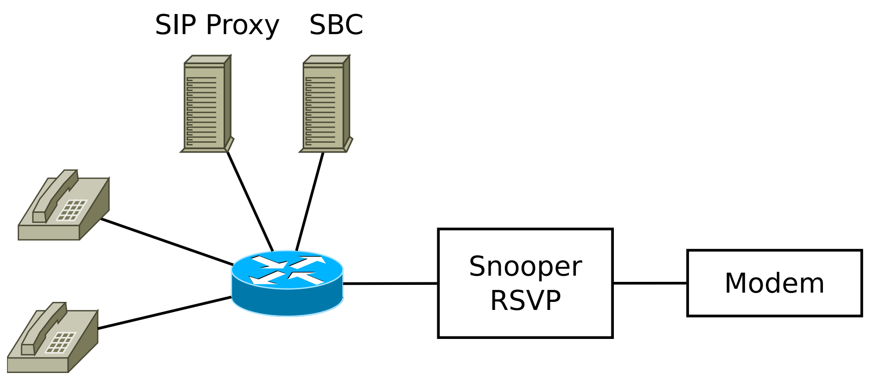

Figure 4.7: Station with a snooper

go back to a satisfactory situation, where the reservations are lower than the available bandwidth.

4.2.5 Software design

The snooper's code is organized around several modules that perform distinct tasks. These modules are shown in Figure 4.8.

- The **protocol** module is in charge of handling the network protocol stack and headers (IP, RSVP, VLAN) and RSVP messages. It keeps track of all the established reservations and builds new packets to be sent.
- The **traffic control** module performs admission control and summarizes bandwidth usage. It is also in charge of computing preemptions when they must occur. Reservations, along with their associated bandwidth and priority, are registered by this module.
- The **configuration** module listens for configuration requests. A new reservable bandwidth value can be entered, and will be taken into consideration immediately.
- The **network** module handles packet sending and determines the direction of a packet going through the bridge (either entering the local node or leaving this node).
- The **timer** module manages a set of timers, each associated with a reservation, so that cleanup is performed if a reservation is not refreshed as it should (i.e, when a *timeout* occurs the reservation is removed in keeping with RSVP's soft-state approach).

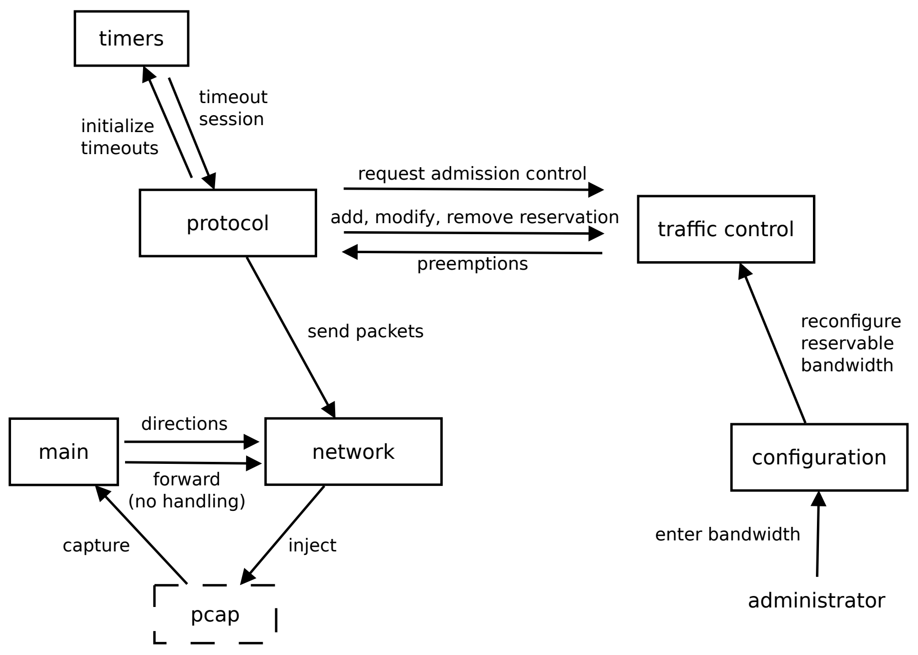

Figure 4.8: Snooper modules and interactions

- The **main** module initializes the modules and libraries, and then starts listening for packets.

Packets are captured using the *pcap* library. This implies that the snooper must either be run as *root* on the machine, or that *capabilities* must be set [*]. Security risks and vulnerabilities are therefore an important consideration during the design and implementation of this software.

Iptables, the Linux kernel firewall, offers packet inspection and captures features with NFQUEUE [25]. This mechanism has not been studied during this thesis project, as careful study of the *pcap* library showed it to provide the features required to develop the snooper.

4.2.6 Traffic Control module

This module is in charge of three main tasks:

- admission control: accept or refuse a reservation, depending on the requested bandwidth, the request's priority, and the reservations that have already been setup;

- registering reservations: when a reservation is established, register the new state, and free it when the reservation is torn down; and

[*] Capabilities grants a user the permission to greater interaction with a system, for example access to administrative-level functionalities of the network stack.

- preemptions: if some bandwidth must be freed to allow a new reservation to be established, select the most appropriate reservation(s) and tear them down.

The traffic control module keeps a list of all the current reservations. When a new request arrives or when admission control must be performed, the module first checks the currently available bandwidth. If there is enough available bandwidth, the reservation is accepted immediately. Otherwise, preemptions are necessary to allow the reservation to proceed. The module then checks if enough bandwidth can be obtained through preemptions. A reservation with priority p [*] can only preempt other reservations that have a priority r such that $p < r$. If the bandwidth requirement can be reached by preempting other reservations, admission control succeeds and the reservation is accepted. If the request was to actually setup the reservation, then the preemptions are executed.

Preemptions are signaled by a pair of RSVP messages, sent upstream, toward the node that originated the reservation. The first message is a RSVP PATH ERROR message, that signals a lack of available bandwidth. The second message is a RSVP RESV TEAR message, which signals that the reservation needs to be torn down. Both these messages are built by the snooper based on the information associated with the reservations that need to be removed. The messages are sent following the process described in section 4.2.9.

4.2.7 Packet handling overview

A packet captured by the snooper via the *pcap* library is first verified for basic validity. Packets entering the snooper are expected to be of one of the following formats:

ETH IP RSVP

ETH VLAN IP RSVP

ETH IP GRE IP RSVP

ETH VLAN IP GRE IP RSVP

The headers are checked so that they follow the recommendations for each protocol. In particular, RSVP and RSVP-TE specify a set of objects that must be present in the RSVP message, depending on the message type. If a packet does not follow this header stack or if one of the headers is not valid, then the packet is dropped at the snooper. Any packet that passes this verification step is known to have one of the above formats and to contain

[*] With $0 <= p <= 7$

a valid RSVP message. This validation step aims to avoid possible buffer overflows or security risks when processing the message.

Not all packets must be handled specifically by the snooper. The snooper's role is to combine the bandwidth used by all the reservations leaving the node it runs on. Reservations entering the node must be ignored, as they do not use any bandwidth allocated to the node. Since the RSVP protocol uses messages that go both ways (downstream and upstream) to setup a reservation, the messages must be filtered by the snooper before being parsed. Table 4.1 shows whether a RSVP message going through the snooper will be handled, depending on its type. Some message types are not handled at all, as they are only informative or related to parts of the RSVP protocol that are not implemented in the snooper.

Table 4.1: RSVP message treatment depending on message type and direction. Only messages marked with an X are processed.

Direction / Message type	Entering	Leaving
PATH		X
RESV	X	
PATH ERR	X	
RESV ERR		X
PATH TEAR		X
RESV TEAR	X	
CONFIRM		
ACK		
SREFRESH		
Direction detail	modem to router	router to modem

4.2.8 RSVP messages handling

The handling of RSVP messages roughly follows the guidelines given in RFC 2209 [8]. Since the snooper is not a real RSVP node, many simplifications can be done to the process. However, since the snooper considers the TE extension to RSVP, some processing must be added to these guidelines to handle the new objects.

Some error cases that should, according to the RFCs, be handled by a full RSVP node, are not implemented by the snooper. The snooper is always an intermediate node between two RSVP nodes, thus the next RSVP node can take care of the error processing and reply with a PATH ERR or RESV ERR message, which is then processed by the snooper. This choice makes error handling a little slower due to the added satellite hop in some cases, but this choice simplifies the RSVP implementation of the snooper, makes the code easier to follow and understand, and will most probably prove to be useful to other people involved in this project (and even other future projects).

4.2.8.1 Message types description

A PATH message is used to initiate a state in all the RSVP routers along the path.

A PATH TEAR message is sent when a node wants to tear down a reservation downstream, toward the destination for this reservation.

A PATH ERR message is sent in the direction of the sender to indicate that the path does not satisfy the requested parameters.

A RESV message is sent upstream, toward the sender, and actually sets up the reservation on a link, according to the parameters specified in the PATH messages. The links used are the exact links that the PATH message followed downstream during its propagation from the sender to the receiver, but in the opposite direction.

A RESV TEAR message is sent upstream to tear down a reservation. It is therefore sent by a receiver or an intermediate router.

A RESV ERR message is sent downstream when an error is detected during reservation setup, if the path does not satisfy the requested parameters.

4.2.8.2 Introducing some data structures

Three main data structures are introduced in RFC 2209: the Path State Block (PSB), the Reservation State Block (RSB), and the Traffic Control State Block (TCSB). They represent essential information needed to process RSVP messages.

The PSB structure holds the path state obtained from a received PATH message. It represents the information associated with a specific (session, sender) pair. The data contained in this structure is the SESSION, SENDER_TEMPLATE, SENDER_TSPEC, and HOP objects from the PATH message, as well as an identifier of the “outgoing” interface for this message.

The RSB structure holds the reservation request obtained from a received RESV message, and contains, in the case of RSVP-TE, information related to a triple (session, next hop, filter_spec). The data held is the SESSION, FILTER_SPEC, FLOWSPEC objects from the RESV message, the next hop’s IP address, and the outgoing interface for the reservation.

The TCSB structure holds the reservation specification that can be handed to the traffic controller for an (outgoing) interface, and is associated with a (session, outgoing interface, filter_spec) triple. It contains the

SESSION, FILTER_SPEC, FLOWSPEC (for RSVP-TE), outgoing interface identifier, and traffic control handle for the reservation.

4.2.8.3 Outgoing interfaces

There is currently only one outgoing interface for the snooper. Although the different VLANs can be seen as distinct interfaces, this is not what we want here. The snooper's goal is to collect the bandwidth needs for the different peers (identified by a VLAN for each peer) and see these reservations as sharing a pool of bandwidth (the bandwidth allocated to the modem or a fraction of this bandwidth).

However, the snooper could be modified to handle two pools of bandwidth, each one corresponding to a specific list of VLANs (for example, VLANs 5 to 8 sharing one bandwidth pool, and VLANs 12 to 17 sharing another pool). In this case, each pool would become a different logical "outgoing interface" for the traffic control module.

4.2.8.4 PATH message processing

The processing of a PATH message starts with looking for a PSB that matches the SESSION and SENDER_TEMPLATE objects from the message. If a PSB is found, its fields are updated as necessary. If no match is found – which means that this message is initializing a new path state along the way – a new PSB is created and filled with the data from the message being processed. Admission control is performed to ensure that a reservation is not allowed to proceed if the snooper cannot provide enough bandwidth. This behavior prevents resources being used downstream when we already know that they will not be needed. If admission control fails, a PATH ERR message indicating a lack of bandwidth is sent, the new PSB is deleted, and the PATH message is dropped.

Finally, the expiration timer associated with this PSB is reset and the PATH message is forwarded.

4.2.8.5 PATH TEAR message processing

When a PATH TEAR message is received, the snooper must tear down the reservation associated with the information in this message and then clear all the states related with it. A PSB matching the SESSION and SENDER_TEMPLATE objects is looked up. If no match is found, the message is dropped and processing is interrupted. Otherwise, all the RSBs matching the message are deleted and the reservations associated with these RSBs are cleared. The PSB is then deleted, and the PATH TEAR message is forwarded.

4.2.8.6 PATH ERR message processing

When a PATH ERR message is received, limited action need be taken by the snooper. The snooper looks for a PSB matching the message. If no PSB is found, the message is dropped. Otherwise, the snooper forwards the message.

4.2.8.7 RESV message processing

When a RESV message is processed, the snooper looks for a PSB that matches the contents of the RESV message. If no match is found, the path for this reservation has not been established and the reservation cannot proceed: a RESV ERR message is sent and the RESV message is dropped. If a match was found, processing continues and the snooper looks for an RSB that matches this message. If a matching RSB is found, then the reservation already exists and should be updated if necessary [*]. If no such RSB is found, the reservation is new and an RSB is created with the data contained in the RESV message. The *update traffic control* procedure is then executed to actually reserve the bandwidth. If the traffic control procedure fails, the RSB is deleted and a RESV ERR message is sent. Otherwise, an expiration timer associated with this RSB is set or reset, and the RESV message is forwarded.

4.2.8.8 RESV TEAR message processing

Upon receiving a RESV TEAR message, the snooper looks for an RSB that matches the data in the message. If there is no such RSB, the message has no effect on the snooper and is dropped. If a match is found, the tear-down concerns an existing reservation, that must be removed. The RSB is deleted and the *update traffic control* procedure is executed to clear the reservation related to this RSB. The snooper then forwards the RESV TEAR message to the previous hop, which is the hop stated in the message.

4.2.8.9 RESV ERR message processing

When a RESV ERR message is received, the snooper looks for a PSB and an RSB that match its contents. If both are found, the message is forwarded. Otherwise, it is dropped.

4.2.9 Sending packets

Packets are sent using *pcap*. Complete packets, including the Ethernet frame header and the VLAN header when needed, are written by the snooper to the output of the interfaces connected to the bridge.

[*] This should not happen in the case of the snooper, since reservations are static: there are no rerouting operations and no bandwidth changes.

All RSVP messages are blocked on the bridge using *iptables*. The messages that must be forwarded are recreated by the snooper.

Since the Ethernet, VLAN, IP, GRE, and IP transport headers for upstream-flowing and downstream-flowing messages are identical for every packet, except for the length and checksum fields, the snooper stores this header list for each session in each direction to make the generation of packets simpler.

When a message needs to be sent by the snooper, the content of the RSVP message is generated by concatenating the required objects. The RSVP header is then created, and the RSVP message is appended to one of the headers stored (either an upstream or downstream header, depending on the direction this message needs to take). The checksums and lengths are computed and the packet is added to a send queue. The packets in this queue are sent at the end of the processing of each RSVP message or after a bandwidth change.

4.2.10 Reconfiguration of the reservable bandwidth

The *configuration* module takes care of any update of the reservable bandwidth. It opens a TCP port and listens for incoming connections. A client is expected to provide a numerical value followed by a new line character. The TCP connection is then shut down.

Due to *pcap* blocking the main thread waiting for captured packets, the configuration module runs in a different process. The value that was received is passed to the main snooper process via another TCP socket, and then a signal is sent to interrupt the *pcap* wait function so that the new bandwidth value is considered. This means that the snooper is a single-threaded process, which greatly simplifies all the processing that occurs on the snooper by avoiding the need for locks and concurrency checks.

When a bandwidth change occurs, two different situations can arise. As long as the new bandwidth is greater than the currently reserved bandwidth, the bandwidth change must be noted by the snooper but nothing more happens. This situation occurs when the new bandwidth in greater than the previously configured bandwidth, or when the new bandwidth is lower than the old bandwidth, but still higher than the amount currently being occupied by the various reservations that have been set up. The second situation occurs when the new bandwidth is lower than the bandwidth currently occupied. In this case, some preemptions (see 4.2.6) are necessary to maintain an acceptable state of the system. The snooper will browse through its list of established reservations, and then will preempt the lowest priorities reservations until the bandwidth currently reserved is at most equal to the configured bandwidth.

Chapter 5

Testing and Performance Evaluation

This chapter describes the testing process and the criteria used to evaluate the snooper's performance. The snooper was tested in two environments: a testbed that allowed only a basic setup and a simulated environment that allowed more complex configurations.

5.1 Introduction to the testing methods considered

When designing and implementing a piece of software, such as the snooper that was developed during this thesis project, which aims to *improve the performance* of an *existing system*, several points need to be evaluated. First of all, the software itself needs to be tested in the same way that any software needs to be tested. This will be described in section 5.1. Then, the complete system, with the new software included, has to be tested. These tests are described in section 5.2. Following this section 5.3 describes the performance measurements that have been carried out. Sections 5.4.3 and 5.6 describe the testing in real conditions and in simulated scenarios (respectively).

5.2 Software testing

Software testing is a very important topic for any kind of software development. Software testing has been studied widely and many approaches have been designed to ensure that everything works smoothly and as expected.

Unit testing is the process of testing small portions of the code, in isolation from the rest of the software. This is used to make sure that each part, taken independently, works as expected, before putting all the pieces together. The result of the combination of the various modules also has to be tested as a complete piece of software.

Multiple kinds of testing have to be performed: check that no bug causes the software to crash, check that the result provided by the software is the result we expect it to produce, and that this result actually is what we needed it to do. These tests need to be done both at the unit level and at different steps during the integration of all the modules.

The software should be tested outside its real environment, to allow invalid input (that should not occur in the actual system, but could happen anyway) to be submitted.

Additionally, testing that the software has no memory leak can be an important consideration, especially if it is expected to run for extended periods of time or in a system that does not have much memory. Valgrind [45] is a powerful tool to assist developers in this task, and this tool has been used throughout the development phase.

5.3 System testing

After the software has been tested and debugged in its "sandbox" environment, it can be integrated into the actual system. The first tests to be conducted are *regression tests*, where we want to make sure that no loss of functionality has occurred due to the addition of the new element. The (hopefully) improved system then needs to be tested to verify that the new addition lives up to the expectations: better performance, offering the desired new features, etc.

In the case of this thesis, the regression tests needed to prove that a given number of VoIP sessions could be established over the satellite network. When there is insufficient bandwidth, a new session initiation will either be rejected (if the priority is lower or equal to the lowest priority of the established reservations) or accepted (if the priority is higher than the lowest priority for an established reservation), in which case the new session will preempt some other reservation before being established. An established communication session should not suffer any packet loss due to a lack of resources (although packet loss could occur due to random bit errors in the packets during transmission).

5.4 Performance measurement

To measure the performance improvement provided by the solutions proposed during this project, three aspects were considered:

Test cases Test cases typical of how the system is expected to be used need to be specified and used to test the system.

Metrics Some relevant sets of metrics need to be chosen. These metrics will subsequently be calculated based upon measurements of the system during performance testing, and analyzed afterwards.

Reference point The current performance of the system needs to be measured, in order to compare this current performance to the results of the new system's performance.

5.4.1 Test cases

The solution was tested in two environments: a realistic environment, which featured the actual equipment running on an experimental platform, and a simulation environment.

In the experimental platform, three modems were used to create a *star* network. Two client stations were used, and the third node, the hub, was not equipped with client equipment (i.e., it did not have SIP phones, SIP proxy, SBC). This setup demonstrates that the solution can be integrated into the actual desired system, and that no functional errors were caused by integrating the snooper into the system.

In the simulation environment, the possibilities of configuration were more flexible. A star setup with more than 2 client stations could be used, along with a mesh configuration.

5.4.2 Metrics

Each communication session uses a pair of tunnels with the same bandwidth *. Therefore, communications could be considered as a quantized amount of data (with symmetric bandwidth requirements). What matters most is the number of communication sessions that could be established at a point in time during the testing (as one of the goals of the project was to increase the number of simultaneous communication sessions).

While measuring performance the metrics of delay, jitter, and packet loss are interesting to evaluate transmission quality. However, these measurements do not provide an accurate description or basis for evaluation of the work done during this thesis project. The system at hand aims to *prevent* congestion before it happens, so that the transmission links are never used at their maximum capacity, thus jitter and packet loss are expected to be minimal. If congestion occurs, since data traffic associated with reservations is limited to bandwidth that is actually available, it is caused by additional traffic outside of the reservations. Traffic is prioritized on the modem according to the PCP field (section 2.3), so that the traffic for reserved bandwidth is always fully transmitted. The improvements provided by the solutions proposed in this thesis project are related to making use of the reservation of resources. Therefore, the only relevant number is the number of communication sessions that can successfully occur

* Though this was the case during this thesis project, it could change with future uses of the system.

concurrently under specific bandwidth allocations (and which must meet the same minimum QoS criteria in order to be considered acceptable sessions).

5.4.3 Bridge performance

The Linux Bridge kernel module is used in the snooper. The goal of this thesis project is not to establish the performance of the bridge module. Given performance measurements performed on the Linux Bridge on older hardware [48] at low throughputs and on modern hardware [36] at higher throughputs show that the module would not be a bottleneck on modern hardware at the bandwidth the satellite modem system offers.

5.5 Testing in real conditions

During the thesis project, the snooper and the patched version of the CSPF algorithm were tested using a testbed configuration with real modems and routers. Tests were run on a basic testbed setup, using only a star configuration (see section 2.2.2).

5.5.1 Reference test: without the snooper

As shown in figure 5.1, the test setup consists of 2 end-user nodes and a hub. Only the client nodes are equipped with phones. All the modems were configured with 256 kbps of bandwidth, and the voice CODEC chosen requires 100 kbps in each direction for a single call. Thus the reservations are for 100 kbps for each session, allowing exactly one call to be placed within the available transmission capacity.

The router interfaces were all configured with 256 kbps of reservable bandwidth. Therefore, the hub's router allows complete use of the available bandwidth. It also allows excess reservations, since 2*256 kbps of bandwidth can be reserved while only 256 kbps can be transmitted by the modem. In this configuration, two calls can be placed, but since there is not enough bandwidth, congestion occurs very quickly and both calls quickly are terminated.

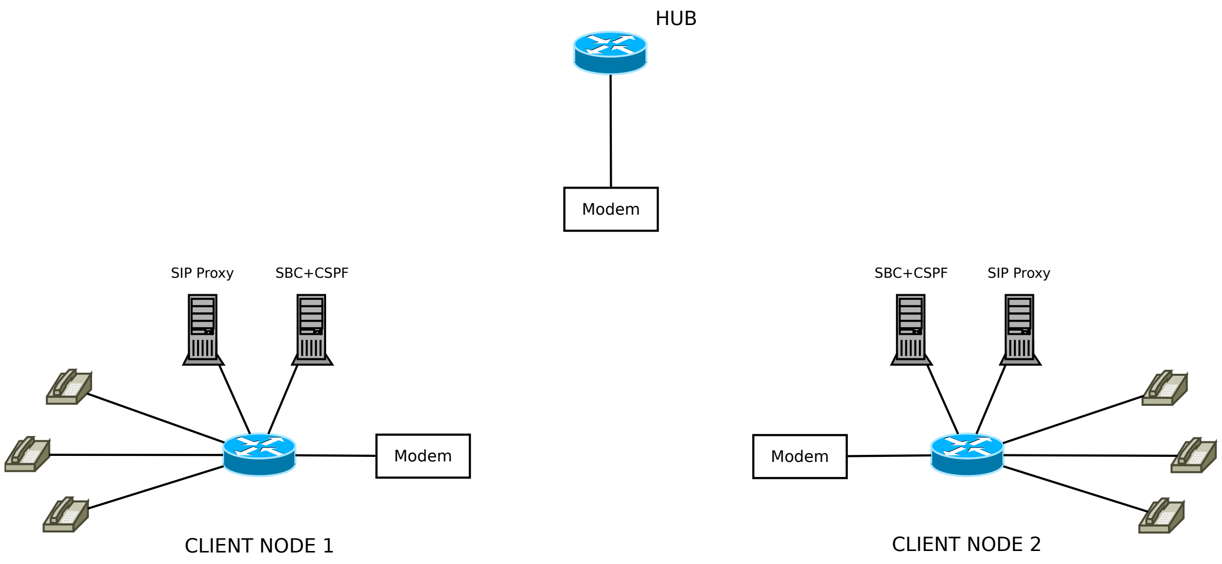

Figure 5.1: Test configuration on the experimental platform

5.5.2 Test with the snooper

The snooper is installed on the hub, and CSPF modifications are applied to both client nodes. This modified setup in shown in figure 5.2. Bandwidth is allocated in the same manner as in the reference test.

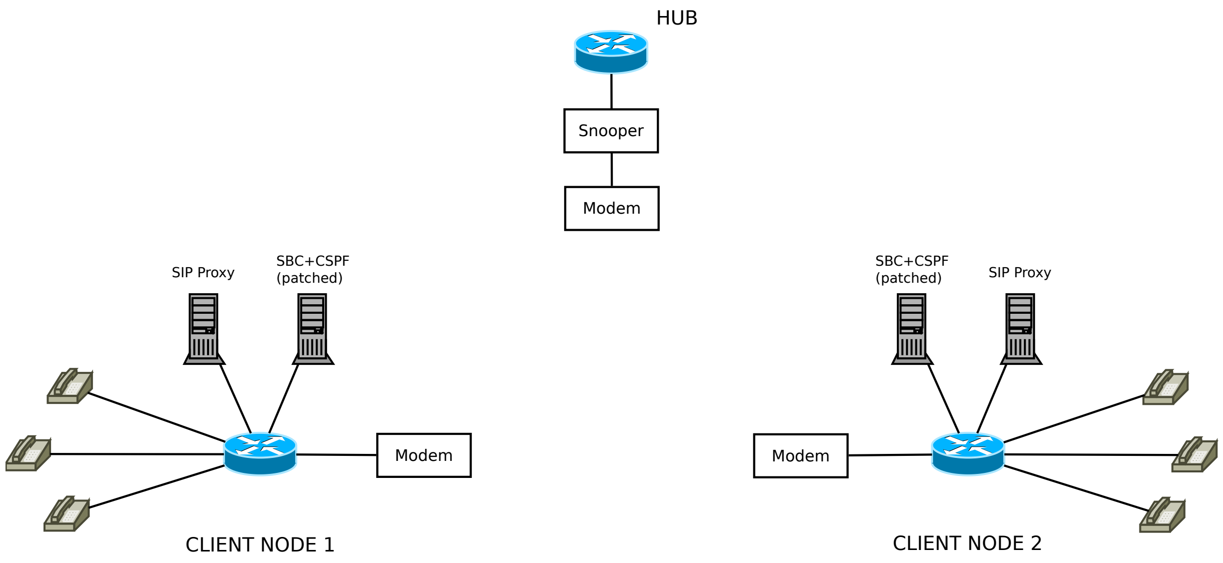

Figure 5.2: Modified test configuration on the experimental platform

When this configuration was used, one call could be placed. If a second call, with a priority lower than the priority for the first call, is initiated by a user, it is not allowed to proceed. However, if a third call is placed with a higher priority than that of the first call, this new call is allowed to proceed and it preempts the first call, so that only one call is established at a time, and no more than 256 kbps of bandwidth are used at any time. A summary of these test scenarios and the number and priority of calls are shown in Table 5.1.

Table 5.1: Scenario and sequence of calls for the snooper demonstration. The highest priority is 0, the lowest priority is 7.

Sequence	Priority	Status
1	4	Allowed
2	7	Blocked
3	2	Allowed, call 1 preempted by the snooper

During this test scenario, the following sequence occurs as seen by the CSPF algorithm and the snooper (the evolution of the reserved bandwidth is shown in Figure 5.3):

First call: no resource reservation is currently established. 256 kbps of bandwidth are reservable on all the router interfaces (node 1 $\leftrightarrow$ hub, node 2 $\leftrightarrow$ hub).

- The CSPF algorithm at node 1 computes a route through the Hub to reach the destination.
- The RSVP messages proceed. The snooper registers that 100 kbps are used on the up-link for the Hub.
- The CSPF algorithm at node 2 computes a route through the Hub to reach node 1.
- The RSVP messages proceed. The snooper registers that an additional 100 kbps are used.
- The communication session is initiated.

Second call: the reservations for the first call are in place. At this point in time, 56 kbps of bandwidth are available at priorities $p >= 4$.

- The CSPF algorithm at node 1 tries to find a route with 100 kbps of reservable bandwidth at priority 7. As there is no such route, the algorithm returns an error code.
- The reservation is not set up and the communication session is not initiated.

Third call: this situation is similar to the second call with 256 kbps of bandwidth available at priorities $p < 4$.

- The CSPF algorithm at node 1 computes a route through the Hub.
- The RSVP messages are forwarded. The snooper detects that a preemption is necessary to allow this reservation to be set up, and it preempts either one of the existing reservations (called R thereafter). The snooper registers the new reservation of 100 kbps at priority 2.
- The SBC that originated reservation R detects that the reservation was torn down and shuts down the corresponding communication session.
- The other SBC for this session shuts down the second reservation. The snooper registers that the reservation has been torn down, so that only one reservation remains.
- The CSPF algorithm at node 2 computes a route through the Hub.
- The RSVP message is forwarded. The snooper registers that an additional 100 kbps (priority 2) are used.
- The communication session is initiated.

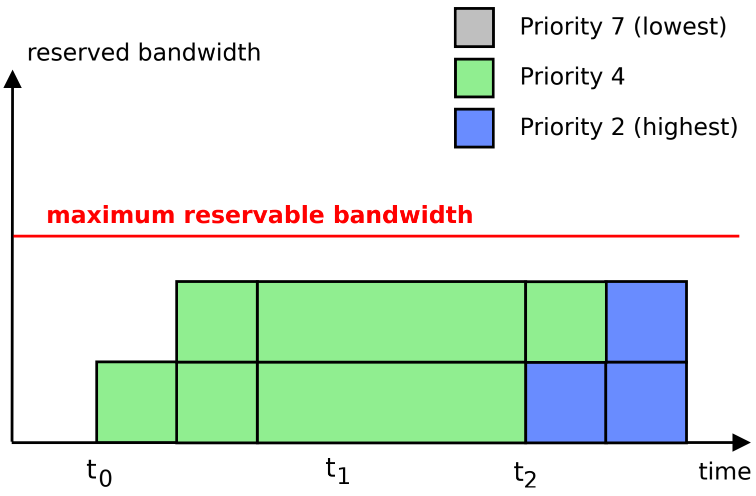

Figure 5.3: Evolution of the bandwidth reserved at the snooper over time. Times t_0, t_1, and t_2 are the times at which the calls are placed. The reservations for the second call are never established. The reservations for the first call are preempted when the third call is placed.

This specific test would have worked just as well without a snooper. By allowing only 128 kbps to be reserved on each of the outgoing interfaces at the hub (128 kbps on the hub → node 1 link, and 128 kbps on the hub → node 2 link), the router would have "shared" the bandwidth satisfactorily, since for each reservation on one link, an identical reservation will be made on the second link. However, if the hub was equipped with phones, or utilized a broadcast service with reservations (for example, for video broadcasting) that did not provide content in the same way to nodes 1 and 2, or if additional client nodes were added, such as in a *star* configuration with three or more nodes, then the utility of the snooper would become more obvious. This more complex scenario has been simulated.

5.5.3 Conclusions of the tests

This set of tests, run on the demonstration hardware, proved that the snooper can easily be integrated into the system and does not require modifications to the current modems, except for the addition of a host (between the router and its modem) that runs the snooper. This demonstration also proved that the original level of functionality is maintained: calls can still be placed in exactly the same way. The improvements provided by the snooper are not demonstrated in this test, since this particular configuration of the system allowed it to perform in a correct way. However, with the configuration that was used for the test presented in section 5.5.2, the snooper's and the improved CSPF's actions were demonstrated.

5.6 Simulations

The modem system allows the deployment of a network much more complex than could be set up using only the testbed configuration that was available during this thesis project. These complex configurations were tested in a simulation environment to evaluate the performance of the snooper in more realistic situations. Tests were conducted in "star" configurations with several nodes and in both full and partial mesh configurations. These configurations were chosen because they provide a minimal subset of configurations that replicate all the possible test scenarios.

5.6.1 Simulation environment

The simulations were designed to simulate as closely as possible a probable application environment. These simulations were run on a Linux machine, using the network simulator GNS3 [20] in conjunction with Dynamips [15]. GNS3 is a graphical network simulator that allows the programmer to implement a number of different topologies using real router models. GNS3 also provides an interface to the Linux Kernel's networking stack, so that the simulated network can be connected to a real network, to virtual machines, or to virtual network interfaces using the TAP driver *. Dynamips is an emulator that enables the user to run a complete Cisco IOS system on a Linux or Windows computer.

The snooper was run directly on the computer used for the simulations, using TAP interfaces between the simulated network and the snooper. The CSPF algorithm was also run on the same computer after its code was modified.

In these simulations, the CSPF algorithm was *not* used. The paths were decided and entered manually. The client nodes are simulated using a router – that represents the actual router that would be needed for this node – and a second router, which acts as the TE agent at the SBC. This second router runs OSPF(-TE), RSVP(-TE), and MPLS(-TE). When present, the Hub node consists of a simulated router and a snooper.

5.6.2 Simulations in the "star" configuration

The tests with the star configuration all use the same topology (see figure 5.4). It is composed of a Hub and four client nodes. Each client node can establish reservations to any other client node.

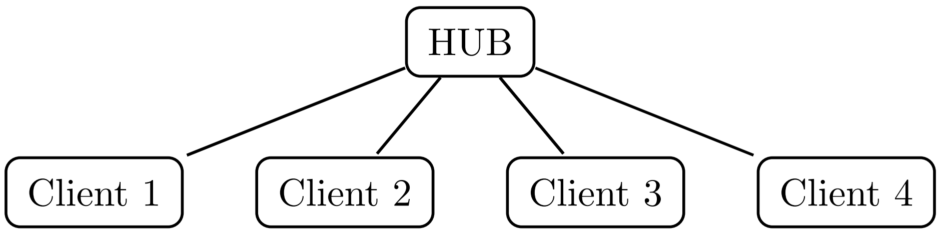

Figure 5.4: Topology used for the simulations with the star configuration

5.6.2.1 Test 1: two client nodes

This first test reproduces the test run on the real system as was described in section 5.5.2. The whole system only has enough resources to set up one call. This test is described in Table 5.2.

The only difference between the same situation running on the test platform and this simulation is the absence of the CSPF algorithm. Therefore, the RSVP messages for the reservations of the second "call" are generated and pass through the snooper. At the snooper, the excess

* TAP interfaces behave like other network interfaces on a Linux computer, thus are compatible with the PCAP library used by the snooper without modification.

Table 5.2: Scenario and sequence of calls for the first test case in the star configuration

Sequence	Priority	Source	Destination	Status
1	4	1	2	Allowed
2	7	1	2	Blocked by the snooper
3	2	1	2	Allowed, call 1 preempted by the snooper

reservation is detected and the admission control procedure is executed, leading to the reservation being refused. The bandwidth utilization is the same as was shown in figure 5.3.

5.6.2.2 Test 2: more client nodes

In this second test, a third node is used. The test case variations demonstrate a reservation failure and a preemption between pairs of nodes that are not the pair of nodes between which the first reservations pair is established. The test sequences for the two variations of this test are described in tables 5.3 and 5.4.

Table 5.3: Scenario and sequence of calls for the second test case in the star configuration, first variation

Sequence	Priority	Source	Destination	Status
1	4	1	2	Allowed
2	7	2	3	Blocked by the snooper
3	2	2	3	Allowed, call 1 preempted by the snooper

In test 2, the snooper acts in the exact same way as in the previous test. The snooper does *not* handle reservations differently depending on variations of the source or destination.

In this case, however, the test would fail without the snooper. With the same bandwidth allocation that was used for this test but no snooper to manage the pool of bandwidth at the hub, the second call of each sequence would not be blocked, since there is still more than 100 kbps of reservable bandwidth on each of the three outgoing links, while almost all the allocated bandwidth (200 kbps out of 256 kbps) is already reserved. The preemption

Table 5.4: Scenario and sequence of calls for the second test case in the star configuration, second variation

Sequence	Priority	Source	Destination	Status
1	4	1	2	Allowed
2	7	3	4	Blocked by the snooper
3	2	3	4	Allowed, call 1 preempted by the snooper

in the third call of the first scenario would occur because all the bandwidth at node 2 is reserved, but two calls would remain (calls 1 and 3), exceeding the capacity of the system. In the second scenario, no preemption would occur, and three calls are set up simultaneously in a system that only has enough capacity for one. The fair repartition scheme would allow less than 100 kbps to be reserved on each link, which means that no call could be set up.

5.6.2.3 Test 3: two concurrent calls

In this third test, the bandwidth set at the Hub is 450 kbps. Each of the nodes have 200 kbps of reservable bandwidth. This allows two concurrent calls to be placed. This test has two variations, for which the sequence of the reservations are listed in tables 5.5 and 5.6.

Table 5.5: Scenario and sequence of calls for the third test case in the star configuration, first variation

Sequence	Priority	Source	Destination	Status
1	7	1	2	Allowed
2	7	3	4	Allowed
3	4	1	3	Allowed, call 1 or 2 preempted by the snooper
4	2	2	3	Allowed, call 2 or 1 preempted by the snooper (depending on which one was preempted during the previous step)

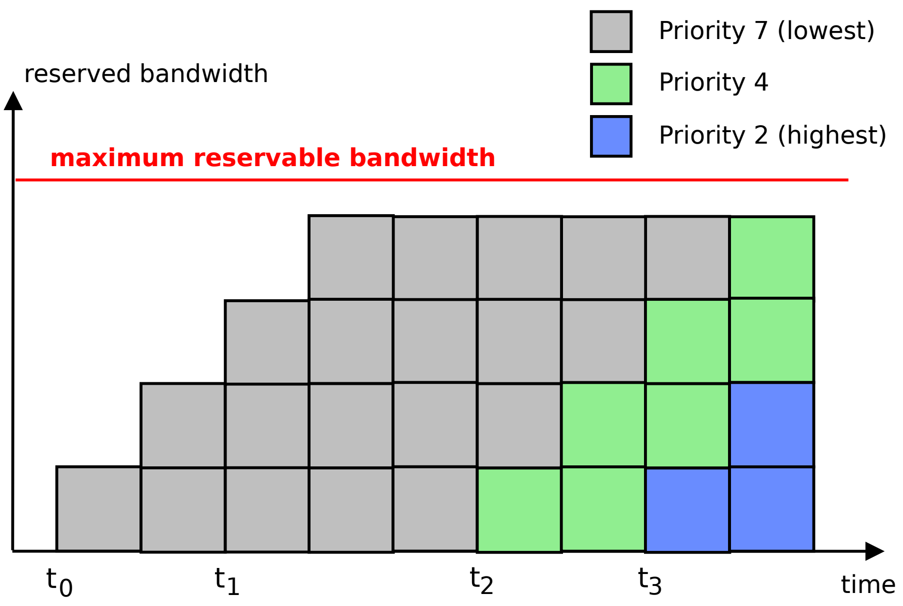

Figure 5.5: Evolution of the bandwidth reserved at the snooper over time in the third test case for the star topology. Times t_0, t_1, t_2, and t_3 are the times at which the calls are placed.

Table 5.6: Scenario and sequence of calls for the third test case in the star configuration, second variation

Sequence	Priority	Source	Destination	Status
1	7	1	2	Allowed
2	**6**	3	4	Allowed
3	4	1	3	Allowed, call 1 preempted by the snooper
4	2	2	3	Allowed, call 2 preempted by the snooper

In these cases, two calls are first set up. When a new call with a higher priority is placed, the lowest priority call is preempted to free enough bandwidth. If there are several calls at the same priority, we cannot predict with certainty which one is going to be preempted. The snooper does not favor reservations based on their age.

Without the snooper, this test would have failed. First, in the fair reservation scheme, the bandwidth on each outgoing link at the hub would

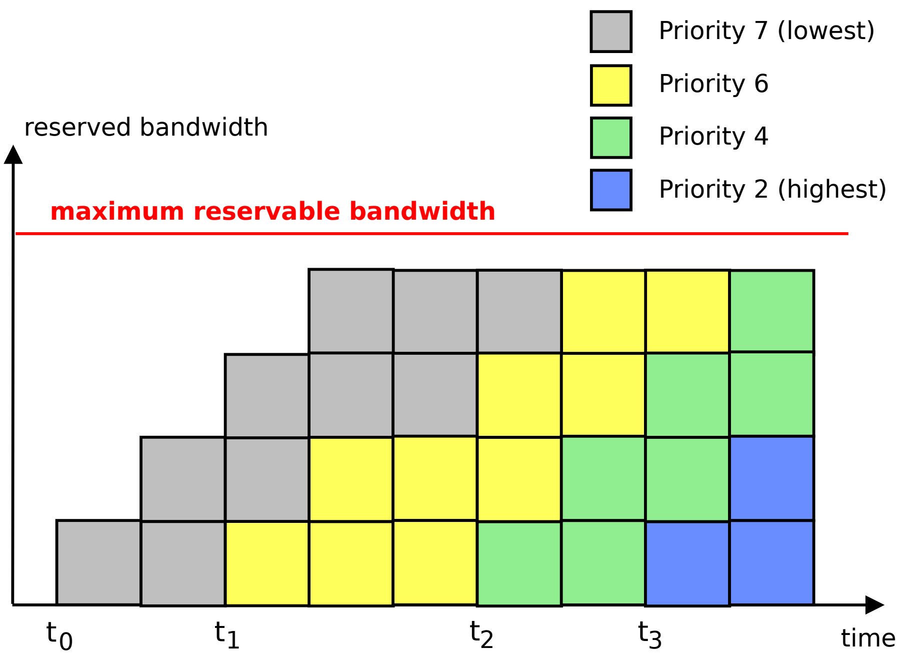

Figure 5.6: Evolution of the bandwidth reserved at the snooper over time in the third test case (second variation) for the star topology. Times t_0, t_1, t_2, and t_3 are the times at which the calls are placed.

allow a single reservation to be established. The reservations for calls 1 and 2 could be established, occupying all the reservable bandwidth on each of the four links. The router at the hub would allow call 3 to be set up, and preempt the first two calls, as they take up bandwidth on links hub–1 and hub–3. The last call would also be established, and the reservations for the third call would be preempted. The system performs better with the snooper, allowing two calls to be established simultaneously in the second half of the test.

Using the same bandwidth allocation that was used for the test but removing the snooper, we would get congestion. The first two calls would be established as in the presence of the snooper. The third call would be established but no preemption would happen, since to the hub's router, only 100 kbps out of 450 kbps are reserved on each of the four links. When the last call is set up, reservations for the second call would be preempted *by node 3*, since it has enough capacity for only two reservations. During sequence items 3 and 4, three calls are set up, thus the system is over capacity.

5.6.3 Simulations in the "mesh" configuration

The tests with the mesh configuration use the topology shown in figure 5.7. There are four client nodes. Each client node can set up reservations to any other client node. Each node is equipped with a snooper.

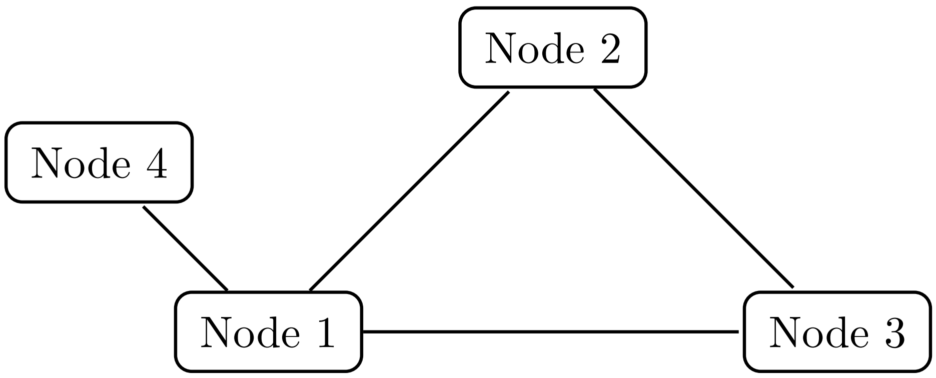

Figure 5.7: Topology used for the simulations with the mesh configuration

5.6.3.1 Test 1: three nodes

In this test, each node is allocated 350 kbps of bandwidth, which means that three outgoing reservations can be set up.

Table 5.7: Scenario and sequence of calls for the first test case in the mesh configuration

Sequence	Priority	Source	Destination	Status
1	5	1	3	Allowed
2	4	2	3	Allowed
3	3	1	2	Allowed
4	2	2	3	Allowed
5	7	1	2	Blocked by the snooper, not enough bandwidth available at node 2
6	0	1	2	Allowed, call 2 preempted by the snooper at node 2

As we can see in figure 5.8, when the last reservation, with priority 0 (sequence number 6), is set up, a preemption is necessary. The reservation

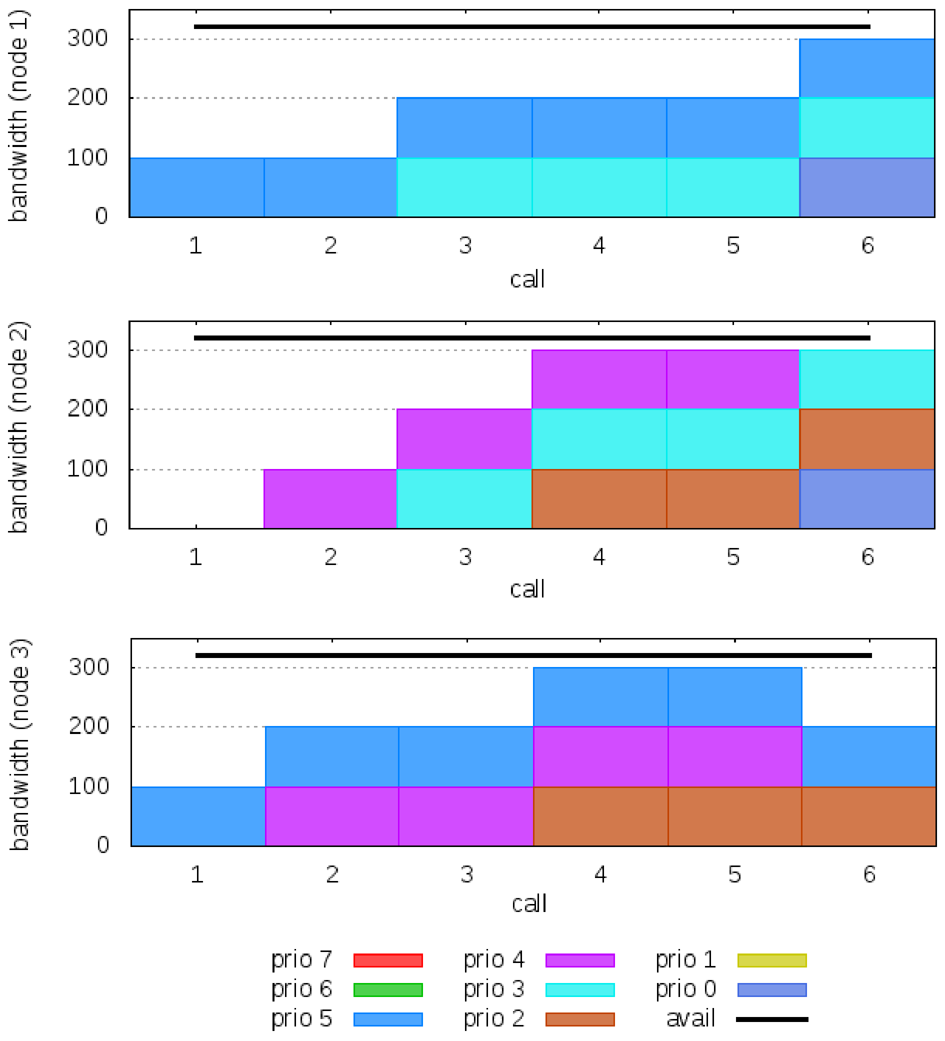

Figure 5.8: Evolution of the bandwidth reserved at the snooper over time in the first test case for the mesh topology.

with priority 4 (reservation 2) is torn down, and the second reservation in this pair of tunnels is torn down too, as the graphic for node 3 shows.

Without the snooper, if the same bandwidth allocation is maintained, reservations 5 and 6 can both be established without causing any preemption, which means that 400 kbps of bandwidth are reserved at node 1 and 500 kbps are reserved at node 2, causing congestion.

5.6.3.2 Test 2: four nodes

In this test, the fourth node in the topology (see figure 5.7) is used. This node is only able to communicate with node 1. Therefore, to communicate with node 2 or node 3, the path must go through node 1 and use some of its bandwidth. As in the previous case, each node has 350 kbps of reservable bandwidth.

Table 5.8: Scenario and sequence of calls for the second test case in the mesh configuration

Sequence	Priority	Source	Destination	Status
1	5	1	3	Allowed
2	4	2	3	Allowed
3	3	1	2	Allowed
4	2	4	2 (via 1)	Allowed, call 1 preempted by the snooper at node 1
5	7	4	2 (via 1)	Blocked by the snooper, not enough available bandwidth on the node 1
6	0	4	3 (via 1)	Allowed, calls 3 and 4 preempted by the snooper at node 1

The bottleneck in this test case is node 1. As an intermediary node between 2 and 4 and 3 and 4, for each pair of reservations between these nodes, two reservations are necessary at node 1. When reservation 4 is set up, reservation 1 is preempted. Reservation 5 is blocked because node 1 already has three reservations of higher priority (one for call 3, and two for the fourth call).

Without the snooper, all calls would proceed without causing any preemption, causing congestion at nodes 1 and 2, with respectively 8 and 4 established reservations.

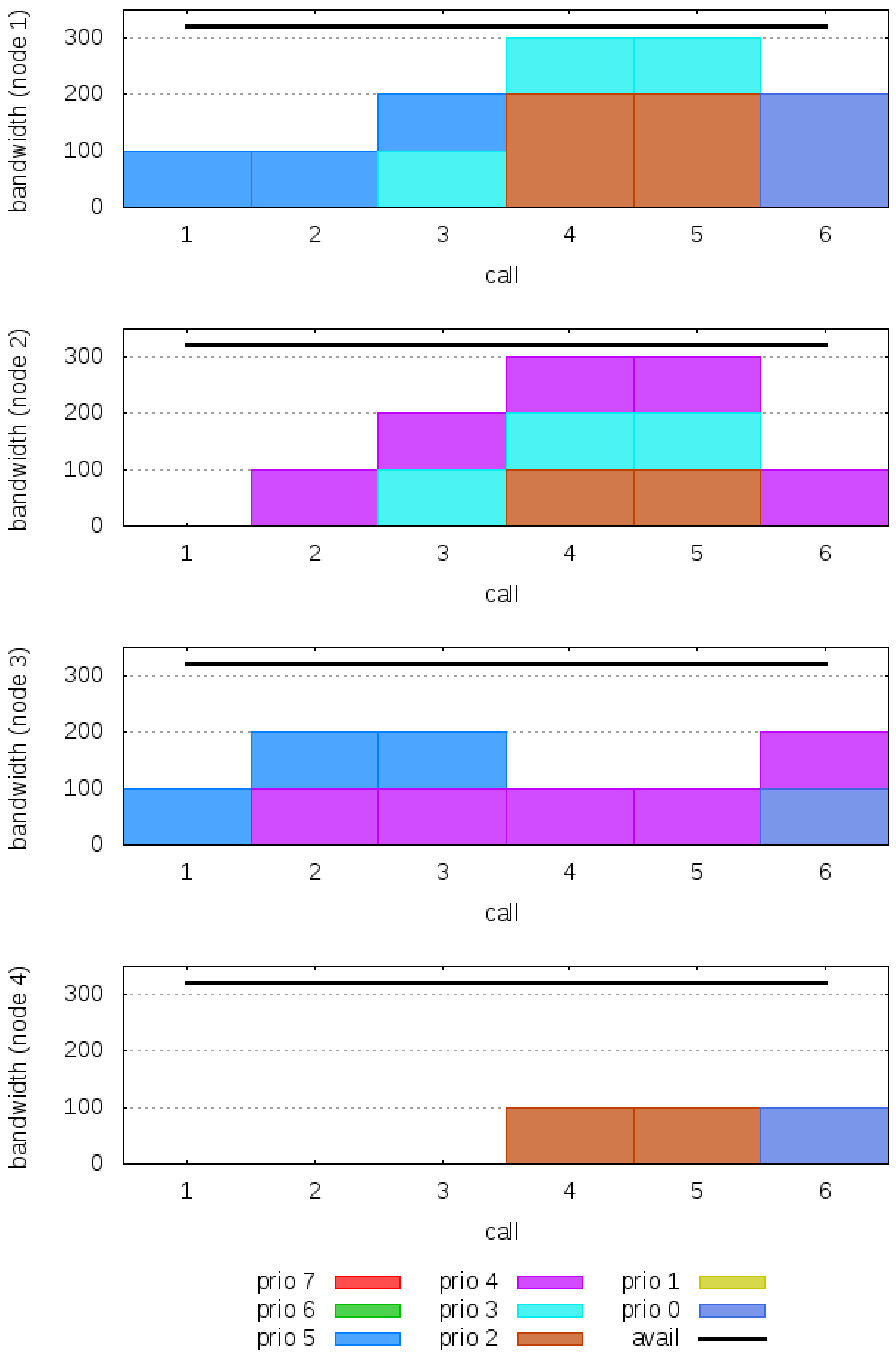

Figure 5.9: Evolution of the bandwidth reserved at the snooper over time in the second test case for the mesh topology.

5.6.3.3 Test 3: link availability

During this test, the link between nodes 1 and 3 will become unavailable. Thus, all traffic between these two nodes will be routed through node 2 while the link is unavailable. The bandwidth for each node is still 350 kbps. This simulation does not use the fourth node.

Table 5.9: Scenario and sequence of calls for the third test case in the mesh configuration

Sequence	Priority	Source	Destination	Status
1	5	1	3	Allowed
2	4	2	3	Allowed
3	3	1	2	Allowed
4	2	1	3	Allowed
5	Link 1-3 goes down			Reservations 1 and 4 are torn down
6	6	1	3	Blocked by the snooper at node 2, not enough available bandwidth
7	1	1	3	Allowed, reservation 2 preempted by the snooper at node 2
8	Link 1-3 goes up			
9	7	1	3	Allowed

The state of bandwidth reservations is shown in figure 5.10. When the link between 1 and 3 goes down, the reservations using that link are torn down, since there is no alternative path for these reservations. Node 2 becomes the bottleneck for this test case, since all the communications must go through it. The sixth reservation is denied (two existing reservations with higher priorities leaving node 2, the maximum is three), but the seventh reservation is allowed to proceed, preempting the reservations for call number 2. When the link between 1 and 3 is up again, the next reservation (sequence number 9) can use it, but the reservations that were set up when this link was not available continue to use the path through node 2, as there is no re-routing of the tunnels.

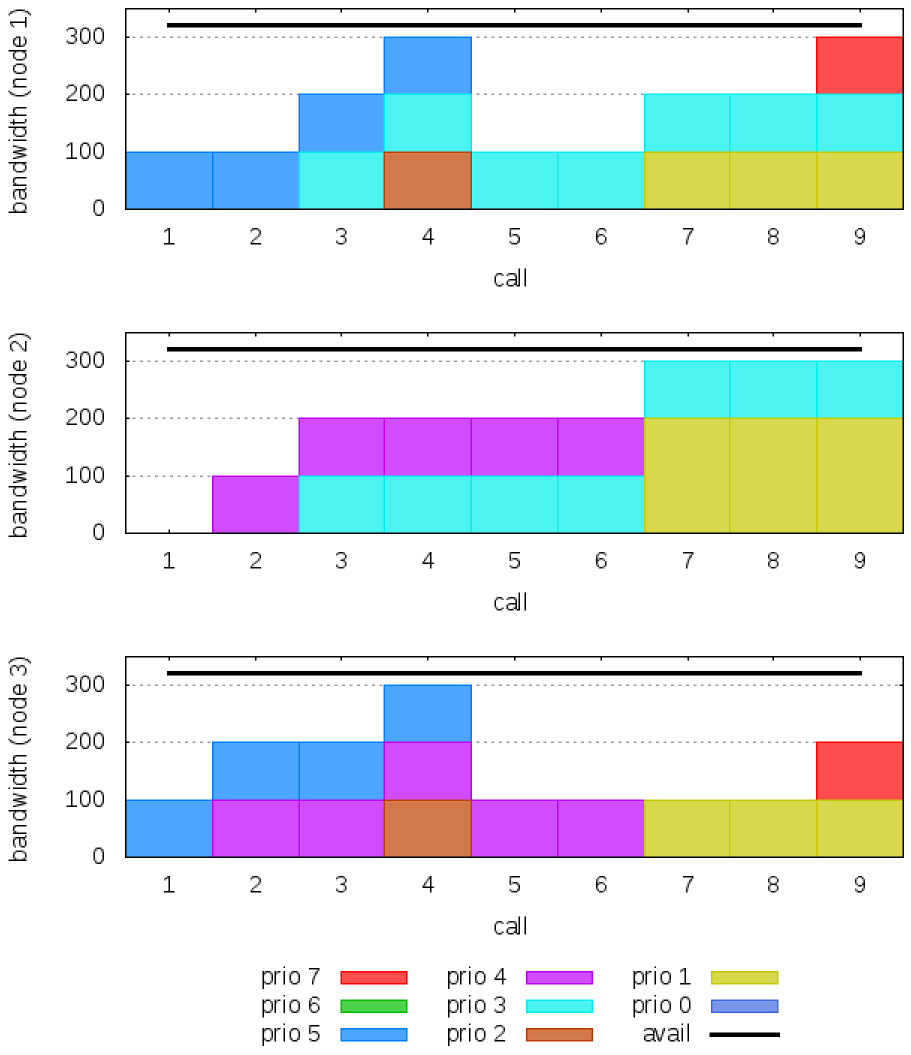

Figure 5.10: Evolution of the bandwidth reserved at the snooper over time in the third test case for the mesh topology.

5.6.4 Conclusions for the simulations

The test cases studied in the previous sections demonstrate a significant range of possible situations. They consider a star configuration with four nodes, which is sufficient to simulate any situation, since these cases show communication sessions between independent pairs of nodes. The test cases also cover a mesh configuration with three parts: several nodes in a full-mesh setting, between which the communications can be direct; an isolated node, which needs to communicate with some of the nodes through an intermediate station; and a link failure in a full-mesh setting, turning the network into a partial mesh.

As we can see in the simulations results, the reservable bandwidth that is configured in a snooper is a hard limit, which is what the snooper was expected to provide. We absolutely wanted to avoid any congestion risk, so it is very important never to reserve bandwidth beyond the configured limit, and to remove excess reservations when the available bandwidth decreases. The tests also demonstrate that the preemption mechanism works as expected.

Using the same test scenarios, the fair repartition scheme prevents congestion but does not allow as many calls to be set up as what is possible when the snooper is used. If bandwidth allocations are kept identical to the configurations involving the snooper, over-reservation occurs in all test cases, causing congestion.

Chapter 6

Conclusions and Future Work

6.1 Conclusion of the testing

The snooper was tested during its development to eliminate bugs and memory leaks. The code was also tested against invalid input to make sure that it would not crash if it received unexpected packets during its operation.

An integration test was performed to prove that the snooper could be added to the actual system and work as we expected from the basic simulations and tests.

Furthermore, the operation of the snooper was tested in more complex cases to prove that it allows optimal utilization of the resources available at any given time, while respecting at all times the hard limit that was set. In all cases, the snooper enabled complete use of the transmission capacity for VoIP communications, whereas the unmodified system only allowed the guaranteed bandwidth, which is usually set to low values (see section 3.2), to be used for reservations.

6.2 Future work

This project provided a working solution to the stated problem.

RFC 2961 [6] documents an extension to RSVP that allows a reduction in the signaling traffic after the reservation has been set up. Through the use of a new field, `MESSAGE_ID`, a number of reservations can be refreshed at the same time. When the reservation is set up, an identifier is associated with the reservation. Afterwards, instead of sending a full PATH message for each reservation, which amounts to about 250 bytes of data, a router simply sends an SREFRESH message to its neighbor, with a list of reservations that are to be part of this refresh. A SREFRESH message for a small number of reservations amounts to 100 to 150 bytes. This extension thus significantly reduces the overhead due to the signaling protocols.

The original standard defining the RSVP protocol [9] introduces an `INTEGRITY` object, but does not define it. The actual definition of this object was introduced in RFC 2747 [5]. The `INTEGRITY` object contains a cryptographic hash of the contents of the RSVP message to guarantee that it has not been tampered with by a malicious intermediary node. Although the purpose of the snooper is perfectly safe, it acts as a kind of man-in-the-middle since it modifies the RSVP messages that go through it: the snooper passes itself off as the previous hop or the next hop along the path for a reservation when it generates a new RSVP message. This is precisely what the `INTEGRITY` object aims to prevent. If this security measure was to be implemented in the system, the snooper would need to be provided with the cryptographic keys for all the routers that it passes itself off as in order to be able to compute a valid cryptographic hash. Otherwise, any RSVP message generated by the snooper would be dropped when it reaches the neighboring RSVP router. This should be relatively easy to achieve in practice since both the snooper and the actual next hop are running as part of the same network and should be managed by the same administrators.

The modem system allows the definition of a "favored" group, so that the nodes that can get a better throughput because of better transmission factors (for example, antenna size, weather, or absence of jamming) are able to take advantage of their better conditions. When this group is used, the time slots are divided in two parts. One part is used only by the favored nodes to communicate, and the other part of the time slots are used to communicate with an "unfavored" node, which includes communication between a favored node and an unfavored node and communication between two unfavored nodes. At the moment, the snooper does not take into account the different destinations to which a reservation is made. Support for this group of favored nodes within the snooper will require some additional work.

6.3 Some additional ideas for future investigations

The ideas proposed in this section are some reflections about what could be possible, but they raise some new issues, such as stability or an excessively high reservation setup delay. They were not considered for a practical implementation during the time frame of this thesis project, as they were expected to require too much work on the whole modem system before a working solution could be found. However, since they consider the system as a whole, instead of focusing on a more restricted subset of the system as some of the other ideas presented earlier, they could potentially provide a significant improvement for the whole system.

6.3.1 Requesting more resources

The CSPF algorithm currently only computes paths that already have enough bandwidth to setup the reservation. However, more bandwidth might be available upon request, for example by using bandwidth that is currently allocated to best effort data flows, but if the CSPF is not aware of this possible reallocation, it will not allow the call to be made. If the central resource allocator could signal to the nodes that there is indeed some bandwidth available for new reservations, then the CSPF algorithm could compute a path that would likely provide the desired resources upon request. In this case, an entity such as the RSVP snooper would notice that the reservation cannot be satisfied with the bandwidth available at the moment, but knowing that the cluster is able to provide more bandwidth to this node, would request the modem to request resources from the central allocator.

6.3.2 Requesting preemptions of reservations that do not go through the local node

This issue arises in a very simple situation. Let us assume that the system only has enough capacity for two reservations. Two reservations are currently in place: one from A to B, with a medium priority, and one from C to D, with the lowest priority. Now, we want to add another reservation, from A to B, with the highest priority. The medium priority reservation can be preempted quite easily, since it has been set up by the local node, A. But at a global level, preempting the lowest priority reservation would seem to be a better choice.

However, this remote preemption has a cost. Let us look at the steps needed to do such a preemption, and then setup the new reservation:

1. Decide to preempt the reservation from C to D;
2. Send the order to the router at the node C;
3. Router C tears down the reservation;
4. The resource allocator detects that the node C requires less bandwidth and reallocates its resources; and
5. Router A now has enough bandwidth.

This enumeration assumes that the allocator can somehow indicate that there is enough bandwidth in the system to allow a new reservation to be made.

Sending the order to a remote node, in itself, takes at a minimum the amount of time necessary to reach a node through at least one satellite hop – maybe more than one hop if the remote node is not directly within reach.

The reallocation of resources within the cluster also takes time – about 2 seconds. Finally, the reservation can be made, which also takes a bit of time. For a node that is one satellite hop away, it would take about one second: 2 satellite hops, plus twice the time spent waiting for the next time slot – half the length of the time slot, on average. Having to wait for more than three seconds before the remote phone starts ringing is quite long. We expect people to pick-up their phone within 2 or 3 seconds, and voice mail often kicks in after about 5 seconds in a cellular network. This could cause a rather bad user experience unless one or more of these expectations were to change.

Another issue, a technical one, is caused by such a design: a race condition can occur. During the time between the preemption at the node C and the time when the new reservation is finally made, another call can be placed by a user, resulting in the resources being used before the new reservation is up. Before the preemption request arrives at the router at node C, the lowest-priority reservation could already have been preempted, either by another low priority reservation, or by a high priority reservation. In the first case, that new reservation could also be preempted, but in the second case, it might not be possible. It could be possible to setup a lock, so that no new reservation is made before the reservation that caused the preemption in the first place is established. However, distributed locking schemes can be quite dangerous, especially when link reliability is not very good, which would probably be the case if preemptions are needed: when the satellite links are reliable, they probably provide enough bandwidth that preemptions are not really necessary. Missing an unlock request and setting the system in a unstable state is quite possible with such a locking mechanism.

6.3.3 Distributed preemptions system

A solution that would appear to provide a significant increase in performance would be to allow each node to choose the path and the necessary preemptions in the whole cluster. This solution would require all the reservations lists to be distributed over the cluster. Such an information exchange can consume a significant amount of bandwidth, especially when resources become quite scarce in the system. It also depends on the ability to request preemptions from a remote node, which could prove to be quite challenging, as has been discussed above.

6.3.4 Central preemptions system

Another solution would be to centralize all the reservations at one node. This node might or might not provide admission control, but the full knowledge of the reservations would allow it to compute optimal preemptions and reallocate resources faster than a distributed reservations scheme that needs to interact with a central (physical) resource allocator. Since the preemptions requests come from a central point, the race condition risk should be reduced or disappear completely. However, even though it should be faster and more reliable than the previous solution, it would cause an additional load to be added to the existing central allocator.

6.4 Conclusions

This thesis gives an account of the ideas that have been proposed and discussed collectively with some of the engineers working on the modem system and on related topics, as well as a background study of the concepts and standards involved in this project. Based on this reflection and on considerations of feasibility, a solution was chosen and implemented. This solution was tested and evaluated during the project, both through simulation of complex scenarios and integration of the solution in the actual setting it was designed for.

The issues analyzed during this master thesis project are complex and could not be completely solved during the time frame for this project. However, a significant improvement of the system's capacity in terms of the number of VoIP communication sessions was provided by the solution this thesis documents. Before implementing the RSVP snooper, the reservable bandwidth in the system was limited to a small quantity that is always available. With the snooper, almost all of the available bandwidth can be reserved. Furthermore, this thesis provided more than a partial solution to the problem posed at the beginning of the project, but also suggests future improvements that could, through extensive study and additional reflection, lead to a more complete solution and further optimizations of the use of scarce network resources. The software developed during this project provides the basis for the implementation of such solutions, and this project provides, in this sense, a framework for future improvement of the studied system.

6.5 Required reflections

This project allowed me to apply and improve my knowledge of network systems, programming, and engineering skills. During this project, I investigated different solutions and evaluated the improvements they brought to the system and their drawbacks, and used this information to decide which of the solutions would be implemented and integrated in the system. I was confronted with network technologies I was not very familiar with beforehand, and with designs quite specific to the system on which this project is based. My work involved reading documentations, standards, and recent research. I made decisions on the relevance of the documentation and research to the system at hand, and decided how to implement subsets of standard protocols in the specific context for this project. I also improved my skills in software design, C programming, and shell scripting. I worked with Linux extensively and used various tools such as Awk and Sed, gnuplot, and the network simulator dynamips. I worked with Cisco routers and network protocols, specifically RSVP-TE and MPLS-TE. Finally, I designed and conducted functional tests. Under the guidance of my supervisors, I proposed and implemented a solution to a very complex problem that could not be completely solved in the limited amount of time I had for this project. Although this solution is not entirely optimal, it provides interesting improvements to the system's performance and a base for future developments. This project provided a great learning opportunity through all its stages.

On the environmental standpoint, the solution proposed is mostly software-based, and does not necessitate replacing and recycling massive amounts of hardware. A single additional computer needs to be added to each deployed node to implement the feature this thesis documents. This feature could be integrated in one of the existing computers at each node, so that no additional hardware is required.

Being a communication system designed for military applications, usual ethical concerns that apply to all military systems are relevant. Improved communication capabilities in dangerous situations and during peacekeaping or humanitarian interventions could help save lives. This same capability can also be used for destructive purposes.

Bibliography

[1] L. Andersson, P. Doolan, N. Feldman, A. Fredette, and B. Thomas. LDP Specification. RFC 3036 (Proposed Standard), January 2001. Obsoleted by RFC 5036.

[2] L. Andersson and G. Swallow. The Multiprotocol Label Switching (MPLS) Working Group decision on MPLS signaling protocols. RFC 3468 (Informational), February 2003.

[3] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow. RSVP-TE: Extensions to RSVP for LSP Tunnels. RFC 3209 (Proposed Standard), December 2001. Updated by RFCs 3936, 4420, 4874, 5151, 5420, 5711.

[4] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus. Requirements for Traffic Engineering Over MPLS. RFC 2702 (Informational), September 1999.

[5] F. Baker, B. Lindell, and M. Talwar. RSVP Cryptographic Authentication. RFC 2747 (Proposed Standard), January 2000.

[6] L. Berger, D. Gan, G. Swallow, P. Pan, F. Tommasi, and S. Molendini. RSVP Refresh Overhead Reduction Extensions. RFC 2961 (Proposed Standard), April 2001.

[7] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Service. RFC 2475 (Informational), December 1998. Updated by RFC 3260.

[8] R. Braden and L. Zhang. Resource ReSerVation Protocol (RSVP) – Version 1 Message Processing Rules. RFC 2209 (Informational), September 1997.

[9] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification. RFC 2205 (Proposed Standard), September 1997. Updated by RFCs 2750, 3936, 4495, 5946.

[10] Charles L. Hedrick. An Introduction to IGRP. http://www.cisco.com/warp/public/103/5.html, August 1991.

[11] Enhanced Interior Gateway Routing Protocol. http://www.cisco.com/warp/public/103/eigrp-toc.html, April 2005.

[12] Introduction to EIGRP. http://www.cisco.com/en/US/tech/tk365/technologies, August 2005. Technology White Paper.

[13] R. Coltun. The OSPF Opaque LSA Option. RFC 2370 (Proposed Standard), July 1998. Obsoleted by RFC 5250, updated by RFC 3630.

[14] Dynagen website. http://www.dynagen.org/, December 2010.

[15] Dynamips website. http://www.ipflow.utc.fr/index.php/Cisco_7200_Simulator, December 2010.

[16] Ericsson. Ericsson Response - disaster relief and connectivity. http://www.ericsson.com/article/ericsson-response_965785785_c, October 2010.

[17] European Commission, Joint Research Centre, Institute for the Energy, Renewable Energy Unit. Code of Conduct on Energy Consumption of Broadband Equipment, November 2008.

[18] A. Farrel, J.-P. Vasseur, and J. Ash. A Path Computation Element (PCE)-Based Architecture. RFC 4655 (Informational), August 2006.

[19] G. Camarillo, W. Marshall, J. Rosenberg. Integration of Resource Management and Session Initiation Protocol (SIP). RFC 3312 (Proposed Standard), October 2002. Updated by RFCs 4032, 5027.

[20] GNS3 website. http://www.gns3.net/, December 2010.

[21] J. Hautakorpi, G. Camarillo, R. Penfield, A. Hawrylyshen, and M. Bhatia. Requirements from Session Initiation Protocol (SIP) Session Border Control (SBC) Deployments. RFC 5853 (Informational), April 2010.

[22] S. Herzog. RSVP Extensions for Policy Control. RFC 2750 (Proposed Standard), January 2000.

[23] IEEE Computer Society. IEEE Standard for Information Technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific Requirements – Part 2: Logical Link Control. *IEEE*, 1998.

[24] IEEE Computer Society. IEEE Standard for Local and metropolitan area networks: Virtual Bridged Local Area Networks. *IEEE*, 2005.

[25] Iptables NFQUEUE. `http://www.netfilter.org/projects/libnetfilter_queue/index.html`.

[26] James F. Kurose, Keith W. Ross. *Computer Networking: A Top-Down Approach*, chapter 7.6: Providing Quality of Service Guarantees. Pearson Education, 5th edition, 2009.

[27] D. Katz, K. Kompella, and D. Yeung. Traffic Engineering (TE) Extensions to OSPF Version 2. RFC 3630 (Proposed Standard), September 2003. Updated by RFCs 4203, 5786.

[28] Kawadia, V., Kumar, P. R. A Cautionary Perspective on Cross Layer Design. *IEEE Wireless Communications*, 12, February 2005.

[29] The Linux Bridge Project. `http://sourceforge.net/projects/bridge`.

[30] Linux Bridge guide. `http://www.linuxfoundation.org/collaborate/workgroups/networking/bridge`, November 2009.

[31] Manayya KB. Constrained Shortest Path First. *IETF Draft*, February 2010.

[32] J. Moy. OSPF Version 2. RFC 2328 (Standard), April 1998. Updated by RFC 5709.

[33] Tiia (nee Sutinen) Ojanperä. Cross-layer optimized video streaming in heterogeneous wireless networks, 2013.

[34] K. Nichols, S. Blake, F. Baker, and D. Black. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. RFC 2474 (Proposed Standard), December 1998. Updated by RFCs 3168, 3260.

[35] J. Postel. Internet Protocol. RFC 791 (Standard), September 1981. Updated by RFC 1349.

[36] Radim Roška. Performance evaluation of GNU/Linux network bridge. Master's thesis, Czech Technical University in Prague, Faculty of Electrical Engineering, Department of Computer Science and Engineering, May 2011.

[37] Amir Roozbeh. Resource monitoring in a Network Embedded Cloud : An extension to OSPF-TE. Master's thesis, KTH, Radio Systems Laboratory (RS Lab), 2013.

[38] E. Rosen, D. Tappan, G. Fedorkow, Y. Rekhter, D. Farinacci, T. Li, and A. Conta. MPLS Label Stack Encoding. RFC 3032 (Proposed

Standard), January 2001. Updated by RFCs 3443, 4182, 5332, 3270, 5129, 5462, 5586.

[39] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching Architecture. RFC 3031 (Proposed Standard), January 2001.

[40] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261 (Proposed Standard), June 2002. Updated by RFCs 3265, 3853, 4320, 4916, 5393, 5621, 5626, 5630, 5922, 5954, 6026.

[41] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550 (Standard), July 2003. Updated by RFCs 5506, 5761, 6051.

[42] S. Shenker, C. Partridge, and R. Guerin. Specification of Guaranteed Quality of Service. RFC 2212 (Proposed Standard), September 1997.

[43] Pontus Sköldström. Multi-region GMPLS control and data plane integration. Master's thesis, KTH, Communication Systems, CoS, 2008.

[44] Srivastava, V., Motani, M. Cross-Layer Design: A Survey and the Road Ahead. *IEEE Communications Magazine*, 43, December 2005.

[45] Valgrind website. `http://valgrind.org`, December 2010.

[46] J. Wroclawski. Specification of the Controlled-Load Network Element Service. RFC 2211 (Proposed Standard), September 1997.

[47] J. Wroclawski. The Use of RSVP with IETF Integrated Services. RFC 2210 (Proposed Standard), September 1997.

[48] James T. Yu. Performance evaluation of linux bridge. Technical report, 2004.

Appendix A

Outputs

This appendix describes the work produced of this Master Thesis.

A.1 Snooper

A RSVP Snooper software has been writen during this thesis. It comes with a Makefile to build it, a documentation on how to use it and how it works. A set of scripts to help use the snooper have also been writen.

A.2 Future work ideas

Some ideas that have not been implemented during this project could be considered for further research.

A.3 Tests conducted, simulations, and tools

The tests conducted during this thesis, and particularly the scenarios and ideas they represent, could be used during a further development of the product. Some of the simulations could also be useful in the future, as could the scripts and configuration files writen for them.

A.4 The present report

This report itself, the list of accronyms and the bibliography, as well as the figures designed, are a part of the deliverables for the project.

TRITA-ICT-EX-2013:206