University of Twente
The Netherlands

PERFORMANCE EVALUATION OF RMD (RESOURCE MANAGEMENT IN DIFFSERV) WITHIN NSIS (NEXT STEPS IN SIGNALING)

Master of Science Thesis

By Desislava C. Dimitrova

Committee:

Dr. ir. G. Karagiannis[1] (first supervisor)

Prof. dr. Hans van den Berg[1,2]

Dr. ir. P. T. de Boer[1]

Date:

July 6, 2006

[1] University of Twente, Faculty Electrical Engineering, Mathematics & Computer Science, Chair Design and Analysis of Communication Systems, Enschede, The Netherlands

[2] TNO ICT, Delft, The Netherlands

Abstract

A new signaling framework is developed by the NSIS (Next Steps In Signaling) working group within IETF. One of the purposes of this framework is to support quality of service provisioning. A particular signaling protocol to deliver quality of service to end users is the RMD-QOSM protocol of the NSIS framework.

The main goal of this research is to evaluate the performance behavior of the RMD-QOSM protocol in a realistically simulated network. A simulation model within the Network Simulator (ns) version 2.29 is developed. The model includes traffic generation, the RMD-QOSM protocol behavior, and the transmission media.

The RMD-QOSM performance is tested by using VoIP simulated flows with three priorities – low, medium and high.

In the first set of experiments the impact of dropping marked bytes on the severe congestion detection and handling solution is observed when multiple severe congestion points are occurring. In the second set of experiments the effect of using an ingress-egress pair aggregate on the severe congestion performance is observed. In both experiments unidirectional flows are considered when the flows pass through different ingresses but the same egress.

In the next three sets of experiments bi-directional reservations are considered. In the third set is observed the influence of the reservation sizes of the forward and reverse directions on severe congestion situations on either the forward or the reverse path. The fourth set tests how flow termination, based on reservation sizes, affects the performance of the severe congestion solutions. In the fifth set of experiments the impact of flow termination, based on the severe congestion state of the ingress, on the performance of the severe congestion solutions on the forward and on the reverse direction is observed.

The last set of experiments concentrates on the state scalability of the RMD-QOSM protocol when compared to the pure QoS NSLP protocol.

For each set of experiments a realistic network topology is defined and performance measures, such as link utilization and time to detect and solve the congestion, are monitored. Special attention is given to the handling of flows with different priorities.

The first set of experiments shows that when data packets are dropped the severe congestion is solved in a slower time frame than if there are no drops. The proposed optimization, using dsRED queuing discipline, solves these issues. The second set of experiments shows that when the ingress-egress pair aggregates (a new proposal) is used then the handling of priority flows might be disturbed. From the third set of experiments it is concluded that the size of the reservations in both directions has a major influence on the link utilization. Besides when both directions are severely congested an utilization undershoot might occur, that is too many flows are stopped. An optimization in the mechanism, which uses knowledge on the number of terminated flows, is proved to perform better.

Samenvatting

De IETF werkgroep Next Steps In Signaling (NSIS) is verantwoordelijk voor de ontwikkeling van een nieuw raamwerk voor signaleren. Één van de doelen van dit raamwerk is om de kwaliteit van service in te schatten. Een signalerend protocol voor kwaliteit van service aan de eindgebruikers is het RMD-QOSM protocol dat deel uitmaakt van het NSIS raamwerk.
Het hoofddoel van dit onderzoek is de evaluatie van de prestaties van RMD-QOSM in een realistisch gesimuleerd netwerk. Naar aanleiding hiervan is voor de netwerksimulator (ns) een simulatiemodel ontwikkeld. Het model bevat een pakketgenerator, het RMD-QOSM protocol en een transmissie medium.
Voor het evalueren van de RMD-QOSM QoIP zijn netwerkstromen met drie prioriteiten gebruikt, namelijk: hoog, gemiddeld en laag.

In de eerste groep experimenten is het effect van gemarkeerde (‘drop’) gegevenspakketten en het oplossen van sterke congestie onderzocht, wanneer deze congestie plaatsvindt op meerdere verbindingen. De tweede groep van experimenten richt zich op het monitoren van het oplossen van sterke congestie als een complex *ingress-egress* paar gebruikt wordt. In beide experimenten is aandacht besteed aan eenrichtingsstromen, die afkomstig zijn van verschillende *ingress* knooppunten, maar arriveren bij hetzelfde *egress* knooppunt.
In de volgende drie groepen experimenten zijn tweerichtingsreserveringen gebruikt. Het derde experiment beschouwt de invloed van de reserveringsgrootte van de beide richtingen aangaande het oplossen van sterke congestie, wanneer deze optreedt in één van de twee richtingen. Voor de vierde groep experimenten is een geoptimaliseerd mechanisme voor afbreking van de netwerkstroom toegepast. Het mechanisme maakt gebruik van de reserveringsgrootte. De prestatie bij sterke congestie is geëvalueerd. In de vijfde groep experimenten is het mechanisme voor het oplossen van sterke congestie geoptimaliseerd en zijn de prestaties bij tweerichtingsreserveringen geëvalueerd in de situatie met sterke congestie in beide richtingen.
Het laatste experiment betreft de schaalbaarheid van RMD-QOSM in vergelijking met het pure QoS NSLP protocol.

Door de eerste groep experimenten wordt duidelijk dat een daling van gemarkeerde gegevenspakketten de tijd voor het detecteren en oplossen van sterke congestie verlengd. De tweede groep experimenten bewijst dat bij toepassing van een *ingress-egress* paar het effect kan zijn dat de stromen niet meer verwerkt worden volgens hun prioriteiten. Uit de derde groep experimenten blijkt dat de grootte van de reserveringen in beide richtingen grote invloed heeft op het gebruik van de verbinding. Bovendien bestaat er een kans dat meer stromen gestopt worden dan feitelijk nodig is. Het is bewezen dat de voorgestelde optimalisatie betere prestaties levert onder de voorwaarde dat de knooppunten beschikken over voldoende informatie over de netwerkstromen.

Резюме

Нова група протоколи за сигнализация се разработва от NSIS (Next Steps in Signaling), работна група към IETF. Една от поставените цели е предоставяне качество на обслужване (QoS) на крайните потребители, което е задача на RMD-QOSM протокола от новоразработената групата. Този протокол работи с DiffServ мрежова област.

Главната цел на настоящето изследване е да се оцени работата на RMD-QOSM протокола в реалистично симулирана комуникационна мрежа. Симулационен модел на протокола е разработен за мрежовия симулатор (ns) версия 2.29. Моделът включва генератори на трафик, поведението на протокола и комуникационна среда.

Работата на протокола е тествана, като са симулирани VoIP потоци – потоците имат три приоритета: нисък, среден и висок.

Първата група експерименти изследва влиянието на загубата на маркирани пакети върху работата на протокола при пренатоварване в мрежата, при втората се наблюдава ефектът от използването на т.нар. вход-изход агрегат (потоците между един входен и един изходен възел от DiffServ мрежовата област), върху процеса по понижаване на пренатоварването. И в двата случая са симулирани еднопосочни потоци.

Следващите три групи експерименти са за двупосочни потоци. В първия случай се следи за промяна на натоварването в мрежата при симулирани различни големини на потоците, при условие че пренатоварване се случва или само в правата, или само в обратната посока. Втората група експерименти изследва същата ситуация, но при условие, че се използват два различни метода за спиране на потоците. В последната група от експерименти се наблюдава ефектът от използването на оптимизиран механизъм за понижаване на мрежовото пренатоварване, като е последното се случва едновременно и в двете посоки на комуникация.

Направени са допълнителни експерименти с цел да се прецени колко добре новият протокол може да се приложи за големи мрежи, като протоколът RMD-QOSM е сравнен с по-общия QoS NSLP протокол за сигнализация.

След проведените експерименти могат да се направят следните изводи:

1. При загуба на маркирани пакети мрежата остава по-дълго пренатоварена. Решение на проблема е предложената оптимизация с използването на dsRED опашки.

2. При използването на вход-изход агрегат е възможно да не се спази приоритетът на потоците.

3. Големината на потоците влияе в значителна степен върху мрежовото натоварване в двете посоки на комуникация. Допълнително, ако пренатоварване се случи в двете посоки и се използва съществуващият механизъм, прекалено много потоци могат да бъдат спрени. Експериментите потвърждават, че предложената оптимизация, където броят на текущо спрените потоци се следи и използва, работи по-ефективно.

Preface

Allow me to welcome everyone that is interested in this research. If the dear reader, after he/she has read the introduction and the discussions chapter, is still eager to continue to the depths of this report I have successfully accomplished the mission to bring out an appealing topic.

It is a fact that, during the last decade, new applications and services for fixed and mobile devices are developed at high rate. This requires the standardization of new network protocols to provide optimal operation for these novelties. One very important network aspect is service differentiation and quality of service provisioning. A new framework of protocols is being developed to answer the divers signaling needs, one of which is quality of service delivery. The protocol in question is called RMD-QOSM and the motive behind this research is the performance evaluation of this RMD-QOSM.

This research was conducted during my master thesis within the DACS group of the University of Twente, Enschede, The Netherlands, where I was lucky to do my Master program. My first supervisor, dr. ir. G. Karagiannis, is one of the designers of the new protocol and has helped me in understanding how it operates. Together with him goals of the research were set and finally successfully accomplished. Finally he has helped me a lot of feedback about the scientific look of the thesis. My other two supervisors, prof. dr. H. van den Berg and dr. ir. P.T. de Boer, have always answered my questions and have provided me support during the writing of this thesis.

The preparation and the realization of this research required a lot of dedication but fortunately also improved a lot of my capabilities. I have fought my way through programming in C++ and OTcl tutorials [ns1, www; ns2, www]. I got to know how to use the ns2, a simulation environment to test protocols, even with the help of the ns2 manual. Most importantly I have learned what it means to design a protocol and to create a simulation model for it. The hours of simulation have thought me never to underestimate the things that can go wrong.

Finally, I want to thank to all the people that have stand behind me during the master assignment. Beginning with my family, my mum and dad, Filip and Ruben, which have literally survived me. Thanks to Simon and Adrian, my second family. I bid my gratitude to Jan Schut, my study supervisor and my flat-mates and friends that actually believed I am capable of graduating. Special thanks to Gerjan Stokking and Andras Csaszar, which have helped me with the simulation model and the simulator.

Desislava
July 6, 2006

Abbreviations

DSCP	DiffServ Code Point
GIST	General Internet Signaling Transport
IETF	Internet Engineering Task Force
IP	Internet Protocol
MBAC	Measurement Based Admission Control
MRI	Message Routing Information
NSIS	Next Steps In Signaling
NSLP	NSIS Signaling Layer Protocol
NTLP	NSIS Transport Layer Protocol
PDR	Per Domain Reservation
PHB	Per Hop Behavior
PHR	Per Hop Reservation
QNE	QoS NSIS Entity
QNI	QoS NSIS Initiator
QNR	QoS NSIS Responder
QoS	Quality of Service
QOSM	Quality Of Service Model
QSPEC	Quality
RII	Request Identification Information
RMD	Reservation Management in DiffServ
RMD-QOSM	Resource Management in DiffServ Quality Of Service Model
RMF	Resource Management Function
RODA	RMD On DemAnd
RSN	Reservation Sequence Number
RSVP	Resource ReserVation Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

Table of Contents

1 Introduction

Quality of Service (QoS) support in the Internet plays increasingly significant role. That is due to the combination of more available network capacity and high resource demands from real time services, such as voice over IP (VoIP). First of all why bother to provide quality of service on the Internet? Each application type has its own requirements towards the media characteristics, some cannot tolerate delay, whereas others cannot tolerate packet drop. Classification among the applications is needed which results in differentiation in the provided QoS. For example, if you want to play a multiplayer game, for instance World of War Craft, you can survive temporal change in the scenery but being dead and not knowing it can be quite annoying. In other words, you would like your application packets to arrive on time rather than without packet loss.
Soon a need arose to provide different treatment within one QoS category. To give an example when your house is on fire you would rather prefer to call the fire brigade than your parents. The former if classified as an emergency call and the latter as a domestic call, which can be assigned different priorities.
Quality of Service can be provided if a packet classification mechanism exists and a way to manage the network resources is provided. How the applications should be classified in QoS groups is described in QoS frameworks. The definition of how the network nodes are informed on the type of the supported application and on how the network resources should be managed is the task of the so called protocols for quality of service signaling and provisioning.
The first QoS framework that has been standardized by the IETF (Internet Engineering Task Force) is IntServ (Integrated Services) [RFC1633], which uses for QoS signaling support, the Resource Reservation Protocol (RSVP) [RFC2205]. Another QoS framework that has been standardized by the IETF is DiffServ (Differentiated Services) [RFC2475].

1.1 IntServ framework and RSVP protocol

RSVP is a protocol specified to mainly work with the IntServ QoS framework. Its main goal is to allow data flow initiators to inform network nodes, on the path of data flow (i.e., data path), about the QoS requirements of the flow. RSVP is used to make resource reservations (reservation state) only by the nodes that are located on the path followed by the user data. It is therefore, denoted as a path-coupled or path oriented reservation protocol. RSVP is a soft state protocol, which means that nodes have to initiate periodically refresh messages to keep the reservation state active. A reservation session can be released if it is not refreshed and the lifetime of the reservation state expires or by explicit release of the reserved resources.
In the IntServ/RSVP protocol data flows achieve quality of service by announcing their service requirements to the network nodes and making resource reservations. That is done on a per flows basis and unfortunately does not scale efficiently in the global Internet [BaKa05].
The major problems of IntServ/RSVP are described in [FuBa05, BaKa05]:

- lack of fragmentation causing limited length of the transport units and lower link resource utilization;
- reliability problems due to the use of IP or UDP as transport layers, for the transport of the messages, instead of using e.g., TCP. The message delivery is assured only by retransmissions. This imposes constraints on the signaling;
- lack of support for network mobility, which is one of the biggest problems currently in the wireless and ad-hoc networks in particular;
- discovery and signaling message delivery are combined in one step which does not allow RSVP to make use of the available security solutions for Internet.

1.2 DiffServ framework

To support scalability in a more efficient way, a new QoS framework/architecture has been developed by the IETF, called Differentiated Services or DiffServ [RFC2475]. In DiffServ each flow is classified in one of fixed range traffic classes, which are identified by the Type of Service (ToS) field in the IPv4 protocol or by the traffic class field in the IPv6 protocol. These fields are denoted as DS fields. Each traffic class has predefined QoS parameters and all flows classified in the same class are called DS aggregate traffic. Every DS aggregated traffic receives the same level of quality of service, denoted as a Per Hop Behavior (PHB). Packets that are using the same PHB are marked with the same DS code point (DSCP), see below, and receive the same forwarding behavior. DiffServ is scalable for core networks because it provides QoS to DS aggregated traffic, instead of providing QoS to each individual flow.
The DiffServ terminology used in this report is listed below, see also [RFC2475].

DS edge node – a DS node that connects one DS domain to a node either in another DS domain or in a domain that is not DS-capable.
DS-capable – capable of implementing differentiated services as described in this architecture; usually used in reference to a domain consisting of DS-compliant nodes.
DS code point – a specific value of the DSCP portion of the DS field, used to select a PHB.
DS domain – a DS-capable domain; a contiguous set of nodes which operate with a common set of service provisioning policies and PHB definitions.
DS egress node – a DS boundary node in its role in handling traffic as it leaves a DS domain.
DS ingress node – a DS boundary node in its role in handling traffic as it enters a DS domain.
DS interior node – a DS node that is not a DS boundary node.
DS field – the IPv4 header TOS octet or the IPv6 Traffic Class octet. The bits of the DSCP field encode the DS code point, while the remaining bits are currently unused.
Per-Hop-Behavior (PHB) – the externally observable forwarding behavior applied at a DS-capable node to a DS behavior aggregate.

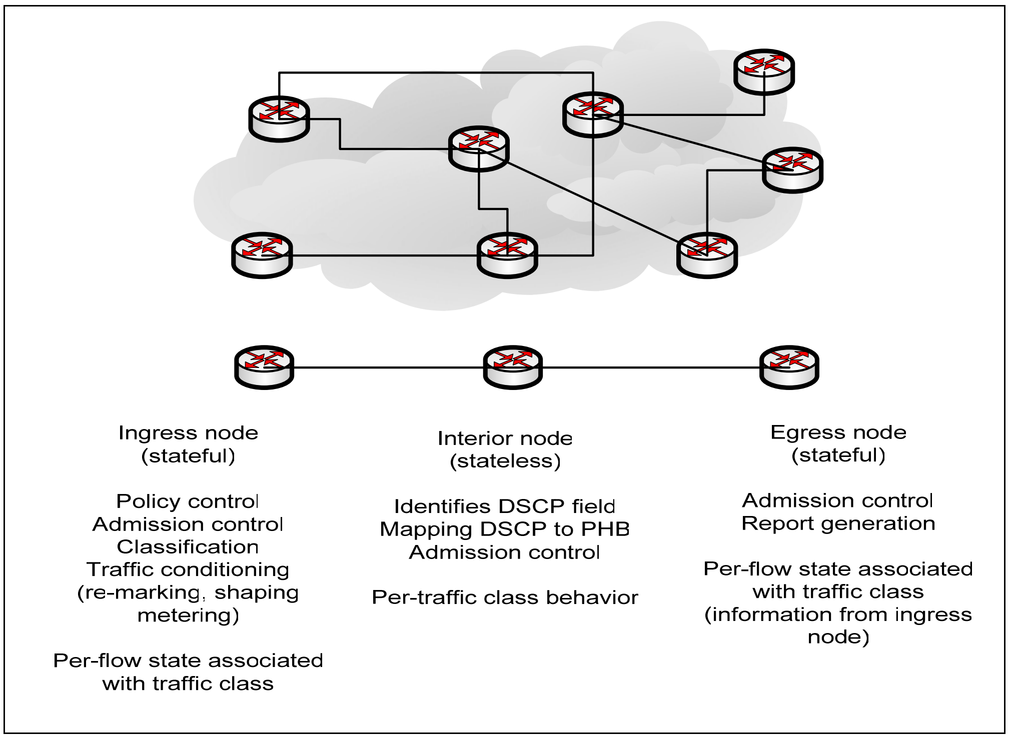


Figure 1.1 General overview of a DiffServ domain

The operation of a DiffServ domain is based on simple general principles and is presented in Figure 1.1. A flow enters the domain at the ingress node, where first admission control is applied. Second the flow is examined and classified in an aggregate behavior and it is given a DS code point (DSCP). To perform the classification a traffic classification policy is used. Subsequently, sometimes traffic conditioning might be needed. This includes shaping (if the flow generates more date that is allowed for its traffic class), metering (collecting data for the behavior of the flow), remarking and scheduling. Remarking may, for example, happen on the border of two domains when the first is non-DiffServ domain but the second is. Thus the edge nodes need not only maintain information of its own domain but also from the connected domains. Interior nodes need not have such knowledge since they only handle intra-domain traffic.
After the flow passes the ingress node, the user data packets associated with the flow, are forwarded via the DS interior nodes to the DS egress node. Each node, by checking the DSCP, can associate each user data packet with a predefined PHB. When the egress node receives a packet it can associate it with the original flow.

1.3 The NSIS protocol framework

The use of the DiffServ mechanism when applied with RSVP has limitations. Therefore a new working group within IETF, Next Steps In Signaling [NSIS, www], took the responsibility of creating a new IP signaling protocol that would solve some RSVP limitations. The intention of the NSIS working group is to provide a general signaling framework, which can support a diversity of mechanisms for QoS support, one of them being DiffServ. The approach taken during the development of the framework is separation of the generation of signaling application messages from the transportation of the messages. This results in two layer model.

The lower layer, denoted as NSIS Transport Layer Protocol (NTLP), provides the transport support of the signaling application messages. The NTLP defines what information should be included in the NTLP layer header and which transport mechanisms are used, i.e. reliable, unreliable, should be used, etc. The NTLP carries the signaling application messages that are generated by the upper layer, i.e., NSIS Signaling Layer Protocol (NSLP). At this layer the rules of generation and processing of signaling messages are defined along with the description of the signaling messages header format. NSLP is meant to be a general signaling protocol, one of which purposes is to signal for QoS provisioning. This special case of NSLP, called QoS NSLP, is responsible for generation of signaling messages used to support the QoS signaling. The QoS signaling can be associated with information related to the used flow classification, the QoS requirements on e.g., delay, bandwidth, jitter, and the administrative permissions assigned to flows that request QoS, etc. The QoS NSLP signaling messages carry a QoS Specification (QSPEC) object, which is dedicated to the description of the QoS parameters.
The QoS NSLP can support several QoS frameworks. These QoS frameworks determine which QoS parameters are used, what their possible values are, how the network nodes should use the parameters and eventually how the network resources should be managed to provide the desired QoS level. Within NSIS the QoS frameworks are called QoS Models (QOSM).
In NSIS, the RMD-QOSM model represents the combination of the DiffServ QoS framework with the Resource Management in DiffServ (RMD), see chapter 4. The specifics on how the quality of service is signaled and provided are included in the RMD QSPEC.
To summarize, imagine a network where the nodes support the NSIS framework. After the signaling messages are processed at the lower network layers they arrive first at the NTLP layer. There, the NTLP header is processed and removed. The upper signaling application layer is recognized as QoS NSLP and the resulting message is passed to it. At the QoS NSLP layer the QoS NSLP header is processed and removed. One part of it is the (RMD) QSPEC, which tells the QoS NSLP layer that RMD-QOSM is used. The RMD QSPEC is passed to the RMD-QOSM for processing after which the node knows how to reserve network resources. This is done is every RMD-QOSM aware node.

1.4 Goal and objectives of the assignment

Project goal: The main goal of this assignment is to accomplish the performance evaluation of the main severe congestion mechanisms used in the RMD-QOSM by using simulation experiments.

The choice of simulation environment is made because to test the protocol in a real network would be more complicated and could affect the network performance and the network user satisfaction.

In order to satisfy the main goal of the assignment several objectives have been identified:

- Study of the protocol specifications of QoS NSLP, QSPEC and RMD-QOSM.
- Study available simulation models on the topic.
- Design of a simulation model that can be used for performance evaluation of RMD-QOSM.
- Implementation of the simulation model in the Network Simulator (ns) environment.
- Define and perform experiments to evaluate the performance of RMD-QOSM.
- Analyze the experiment results, draw conclusions and provide feedback.

1.5 Organization of the report

Chapter 2 presents the analysis of the research topic and possible approaches are presented. Chapter 3 describes the introduced protocols and their components. Note that the introduced protocols are discussed in detail, i.e. message types, header formats, because a correct simulation model can be built only if their specifications are implemented correctly. The information given in Chapter 3 is used in the development of the simulation model. Before this is done, the existing and available simulation model is examined and presented to the reader in Chapter 4. Chapter 5 describes the comparison between the existing and available simulation model with the simulation model that has to be developed in this assignment. At the end of Chapter 5 recommendations on the necessary modifications are given. Theses recommendations are used during the design of the simulation model that is described in Chapter 6. Via the use of C++ [wiki, www] and OTcl [wiki, www] programming languages the design simulation modules are transformed in implementation simulation blocks. Chapter 7 discusses the implementation of the simulation model and how the simulation model can be used to accomplish the simulation experiments that are described in Chapter 8. The simulation experiments are first defined and subsequently their results are presented and discussed. Finally, Chapter 9 presents the conclusions and proposes topics for future research.

2 Problem analysis

A simulation model that is to be used for performance evaluation should include the functionality of the tested protocol along with functionalities supported by the network topology, where the protocol is to be evaluated. The implementation of the evaluated protocol, when used for performance analysis, should represent closely the protocol specifications without having to include details that are not relevant to the performance or have supportive role. Some simplifications, assumptions and abstractions can be done. When the protocol has a complex nature it has to be determined which mechanism should be included and which can be ignored.

RMD-QOSM is such protocol. It consists of several mechanisms, which in co-operation deliver the end protocol behavior. Major mechanisms are the algorithms for admission control, severe congestion detection and handling and quality of service provisioning. Others have supportive role, such as mechanisms for authentication and authorization, flow classification to predefined level of service, security issues, etc. What is even more important is that RMD-QOSM is a combined product of a general signaling protocol, QoS NSLP, and a specific QOSM, see Chapter 1. To get familiar with the protocol behavior one has to study the specifications [NSLP] and [QSPEC], which cover the general processing rules at the QoS NSLP layer, and the [RMD NSLP] specification that concentrates on the specifics of the used QOSM.

RMD-QOSM uses the idea of Resource Management in DiffServ (RMD) [WeJa03] to deliver the required quality of service. The way of how resources are reserved and the mechanisms for admission control and severe congestion are applied in RMD are very similar if not identical as for RMD-QOSM. A simulation model of RMD already exists, which has been developed by András Császár and Atilla Tákacs from Ericsson Hungary [Ericsson, www] and is implemented in the network simulator (ns) environment [ns, www]. This RMD simulation model implements several possibilities for admission control, severe congestion detection and handling procedures and reservation methods to support quality of service provisioning. The mechanisms are described in diversity of papers with the above mentioned as authors [CsTa04, CsTa04]. The model was extended by Gerjan Stokkink to implement the use of DSCP and preemption priorities within the same DSCP class.

Along with modeling the RMD-QOSM element the other network elements also should be represented. Considerable amount of time and work can be saved by re-using already implemented components when this is possible. A good choice of simulation environment results in support of common elements. Among the variety of available simulators the network simulator (ns) is chosen. The ns environment is especially developed for the simulation of communication networks and includes many network modules, which are tested and ready to use. Besides this the existing RMD simulation model has already been implemented in ns.

Due to the similarities in the behaviors of RMD and RMD-QOSM, the simulation model that can be used for RMD-QOSM could partly use the existing RMD simulation model. Before that, a comparison should be done, to determine which parts of the existing simulation model could be re-used and which modifications would be necessary.
To determine what functionality should be included in the RMD-QOSM simulation model it should be clear what part of the behavior will be evaluated. The goal formulated in chapter 1 is broad and is therefore divided in sub-goals. Their number is limited due to the broad functionality of RMD-QOSM and the little available time for research. The chosen sub-goals are listed below:

- Performance evaluation of severe congestion situations in unidirectional operation. Several experiments on the topic were already done but there are still unexamined aspects, e.g., severe congestion on two consecutive links. This point of the research aims at finding optimization solutions to the existing mechanisms.
- Performance evaluation of severe congestion in bi-directional reservations. Experiments on the severe congestion occurrence only in one direction or in both are to be performed. As result of this sub-goal, a feedback to the specification about the bi-directional operations and possible optimizations of the algorithms, are expected.
- Performance evaluation of the admission control mechanism. A new approach towards the admission control mechanism is taken to support preemption priorities. It should be included in the simulation model of RMD-QOSM and be tested with simulated real flows generation.
- Performance evaluation in the means of protocol scalability. The RMD-QOSM protocol can be evaluated as a whole, and one of the aspects is how scalable it is. A research on that topic is provided without being exhaustive. The research is narrowed to the comparison between RMD-QOSM and the general QoS NSLP protocol.

Once the goals are clearly stated the approach towards the research can be presented. First of all the protocol specifications are carefully studied to determined what part of the RMD-QOSM functionality has to be implemented. As second step the existing RMD simulation model is examined and a comparative analysis is done. After it is observed what exists and what not, the new RMD-QOSM simulation model can be designed and subsequently implemented. To achieve a performance evaluation and reach the goals, finally a set of experiments is defined. Their results will provide feedback and will be used to optimize the behavior of RMD-QOSM.

3 NSIS framework signaling protocols

3.1 Introduction

In chapter 1 the need for quality of service provisioning in current IP based networks was discussed along with the need of new signaling protocol solution that will allow quality of service to be supported. As response to this need the NSIS working group of IETF [NSIS, www] is currently busy on developing a signaling protocol suite to support a variety of signaling applications.

NSIS stands for Next Steps In Signaling and this chapter presents a general description of the solution provided by the IETF working group in section 3.2 together with a short introduction of the first transport layer. Subsequently the second signaling layer of NSIS is reviewed in section 3.3 and detailed explanation of the particular protocol of interest is given in section 3.4. Section 3.5 is dedicated to the unidirectional operation of the protocol while section 3.6 concentrates on the bi-directional scenarios.

This chapter is based on the QoS NSLP protocol specification version 9, QSPEC specification version 8 and the RMD QoS NSLP, i.e., RMD-QOSM, protocol specification version 5. All additional changes introduced in newer versions of the specifications are not considered in this research assignment.

3.2 Next Step In Signaling (NSIS) framework

3.2.1 The NSIS protocol suite

The new protocol suite developed by NSIS is split into two protocol layers. The first, lower layer, i.e., NTLP, is responsible for the transport of the signaling messages and the second, upper layer, i.e., NSLP, is responsible for the generation of signaling messages with a predefined format and according to strict rules [BaKa05, KaBa04]. NTLP may use different transport and security protocols and it can operate in two modes – connection orientated (or reliable) and datagram (or unreliable) mode. Which transportation mode is to be used depends on the type of requested connection and on the information passed by the NSLP. NSLP supports the general features of a signaling protocol but each concrete NSLP realization is dependant on the signaling application to be served. The format and semantics of the messages used by different signaling applications are application specific. For example, messages used for quality of service signaling can be processed only by the nodes that support the signaling application for delivering quality of service.

In a real network situation the backbone of the NSIS suite is the NTLP layer. It must be present in every node where NSLP layer is to be used. Which NSLP specific implementations are installed in the NSIS nodes is the decision of the network operator and depends on the purpose of the node. Many NSLP realizations can be installed in the same node without interfering with each other. Currently three NSLP specifications are being developed and tested. These are QoS NSLP to ensure predefined quality of service per session; Network Access Translator (NAT)/Firewall NSLP; and NSLP functionality for metering entities. A schematic presentation of the protocol suite [FuBa05] is given in Figure 3.1.

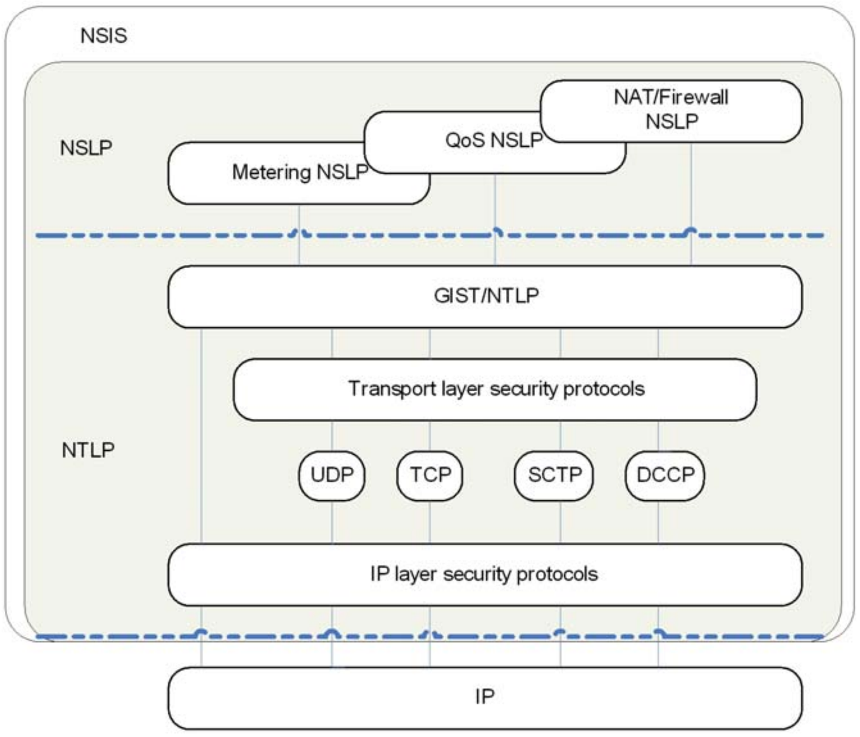


Figure 3.1 NSIS architecture

The most distinguishable features of the NSIS protocol suite, see [FuBa05], are described below:

Transport of signaling messages: The separation of the protocol suite into two layers makes the transportation of the signaling messages independent from their generation at the signaling application. In other words the upper layer is responsible for the creation of the messages and gives them signaling application specific meaning, while the lower layer only transports them throughout the NSIS aware nodes. This separation of duties allows on one hand different signaling application implementations to be developed and on the other hand different transport modes to be used.

Reservation model. The initiator of the reservation can be the sender or the receiver of the data path, which makes NSIS more flexible. In NSIS it is as well possible nodes different from sender or receiver of the data path to initiate messages. Therefore not only end-to-end communication is supported but also edge-to-edge and edge-to-end.

As a *soft state* protocol NSIS nodes keep states that have a certain lifetime. When the lifetime expires the state is no longer kept. To ensure that the node will process a session identified by a particular state the state has to be refreshed every predefined period of time. As result all nodes on the path will keep the particular state and the session will not be interrupted.

Scoping (or range) of signaling. The scoping (or range) of the signaling messages represents the support of connections between end nodes, between edge nodes in a domain, or between an end node and an edge node

Table 3-1 Comparison between RSVP and NSIS

	RSVP	**NSIS**
Protocol structure	Single layer	Two layers
Transport	IP or UDP	Reliable, datagram
Reservation initiator	Receiver	Sender or receiver
States	Soft, explicit release	Soft, explicit release
QoS models	IntServ, DiffServ	IntServ Diffserv, other
Scope of signaling	End-to-end	End-to-end, host-to-edge, edge-too-edge
Multicast	Yes	No
Bi-directional	No	Yes
Mobility	No	Yes
Aggregation	Yes	Yes
Summary refresh	Yes	Yes
Priority	Yes	Yes

.

NSIS does not support *multicasting*, which simplifies the functionality that have to be supported by each NSIS aware node.

Bi-directional reservations are supported by the NSIS suite as the bound reservations initiated and maintained simultaneously in the forward and reverse direction between a QNI (QoS NSLP Initiator) and a QNR (QoS NSLP Receiver).

A *binding mechanism* allows, for example, the identifiers of the two directions to be related and therefore a stateful node knows which forward session to which reverse session corresponds.

Mobility support. NSIS uses for the identification of a flow, a Session ID, instead of using the flow ID used by RSVP. Note that the flow ID is a set of five parameters, i.e.,: IP address of sender, IP address of receiver, port number of sender, port number of receiver and protocol ID. When a user is roaming, the flow ID changes, while the Session ID remains the same. This means that when the reservations are associated with Session IDs, the reservations that were initiated and maintained before roaming will also be kept after roaming.

Along with all above the NSIS protocol suite aims at high s*ecurity level* by implementing security mechanisms in the suite itself or by using available security protocols. *Separation of node discovery and transport of signaling messages* opens a possibility to use different security protocols. Therefore, NSIS can make use of already existing well tested security protocols.

The characteristics of NSIS are listed in Table 3-1 along with their presence or absence in the RSVP protocol [FuBa05]. The latter allows for a fast comparison and the advantages of the NSIS suite are easily recognized.

In the base of the NSIS characteristics and its advantages is the special way to manage connections, used by the NSIS protocol suite. As it was mentioned, a connection in NSIS is identified not by a flow identifier, but by a Session ID. A Session ID is a randomly generated number, supposedly unique, which is not dependent on the flow ID or the end nodes IP address. In fact one session ID can be associated with one or more flow identifiers. As result NSIS supports node mobility, tunneling/bypassing and multi-homing [FuBa05].

3.2.2 The transport supportive layer GIST

The lower layer in the NSIS architecture defines a common protocol that all kind of signaling applications can use. Application specific functionality is given by the signaling protocols that form the upper NSIS layer. The main protocol used by NTLP to provide the transport of signaling messages is the General Internet Signaling Transport (GIST). GIST must be present if an upper layer NSIS protocol needs to be supported by a node. If some node on the sender-receiver path is not GIST enabled, then all NSIS messages are considered to be ordinarily data packets. The GIST terminology used in this report can be found in [ScHa06] and is given below:

Data flow: A set of packets identified by some fixed combination of header fields. Flows are unidirectional (a bidirectional communication is considered a pair of unidirectional flows).
Session: A single application layer flow of information for which some state information is to be manipulated or monitored. It is identified with a Session ID (SID) parameter
Sender: The node in the network which is the source of the packets in a flow. Could be a host, or a router (e.g. if the flow is actually an aggregate).
Receiver: The node in the network which is the sink for the packets in a flow.
Downstream: In the same direction as the data flow.
Upstream: In the opposite direction to the data flow.
Adjacent peer: The next node along the data path, in the upstream or downstream direction, with which a GIST node explicitly interacts. The GIST peer discovery mechanisms implicitly determine whether two nodes will be adjacent.

GIST has two major goals – one to provide routing and second, transportation of signaling messages. The *routing* determines how to reach the adjacent peer along the data path, and can be done independently for each direction of the connection. Two NTLP states are used for the routing – a routing state, used in the forwarding of the messages, and a message association state, used to relate incoming messages to a particular saved session. A *Message Association* is a connection between two explicitly identified GIST adjacent peers and a message, arriving from the signaling application, is connected to an established message association via the SID parameter of the message header.
Transportation is the delivery of signaling information from peer to peer [ScHa06]. The signaling message delivery is divided in two transport modes, the Datagram Mode (D-mode) and the Connection Mode (C-mode). The *Datagram Mode (D-mode)* sends GIST messages between nodes without using any transport layer state or security protection and uses UDP encapsulation [ScHa06]. The *connection Mode (C-mode),* on other hand, sends GIST messages directly between nodes using by default TCP as transport protocol. The choice of mode depends on the routing state and on the requirements coming from the signaling application. The GIST lower layer and the application layer on top of it

communicate vie the interface, or API, defined between them. The primitives passed at the interface are of three groups – reliability or what transportation mode is desired; security or what security mode is required; and local processing or what special processing has to be done like prioritization, etc.
The messages generated at the GIST layer are see [ScHa06]:

- *GIST-Query messages* are used in the first phase of the discovery procedure, 3-way handshake. It is always sent in datagram mode and leads to creation of routing state for the flow and also message association state if necessary.
- *GIST-Response message* is used in the second phase of the handshake and can be sent in datagram or connection mode. If a message association is needed but it is not created this is accomplished during this phase.
- *GIST-Confirm message* is the last phase of the discovery procedure an also can be in datagram or connection mode. If connection mode is used a message association must be established during the transfer of the previous two message types.
- *GIST-Data message* is used to encapsulate all messages coming from the NSLP layer.
- *GIST-Error message* reports errors occurring at the GIST level.
- *GIST-MA Hello message* is used to keep a message association state.

For complete description of each message, see [ScHa06].
When a connection is to be established first the GIST discovery procedure is started. The procedure is between two peers and uses the GIST messages Query, Response and Confirm. As result the peers on the data path sender – receiver are discovered. Subsequently the signaling NSLP messages and the data are encapsulated in GIST Data messages. The GIST discovery procedure can be combined with the NSLP signalization to establish connection.

3.3 QoS NSLP protocol

The signaling layer protocol is tightly connected with the user applications in the end nodes. The tasks of NSLP are two – to create signaling application specific messages with well defined format and to give instructions how these messages are to be handled in the nodes. It was already mentioned that three signaling applications – NAT/firewall NSLP, metering NSLP and Quality of Service NSLP, are developed. The further discussion is concentrated in QoS NSLP.

QoS NSLP dictates the general common processing [MaKa06] of the signaling messages for quality of service provisioning. Different quality of service models (QOSM) can be used. The *Resource Management Function (RMF)* is responsible for processing specific information for a particular QOSM. A special object within the QoS NSLP message, QSPEC, carries information on the quality of service parameters used by the different QOSM. *QoS Model (QOSM)* is the collective quality of service parameters and RMF processing rules to deliver certain level of service to a connection. A QOSM can be local within a domain, global or specific for a particular organization domain. When two or more domains with different QOSM participate in the same connection they have to interoperate. One way is to use stacking – the local domain messages have local QSPEC pushed on top of the original QSPEC from the initiator. Another way to apply local domain

quality of service policy is tunneling where the initiator message travels to the end of the domain unprocessed and an additional local message is constructed.
The terminology used in this chapter is specified in [MaKa06] and is given below:

QNE: an NSIS Entity (NE), which supports the QoS NSLP.
QNI: the first node in the sequence of QNEs that issues a reservation request for a session.
QNR: the last node in the sequence of QNEs that receives a reservation request for a session.
QoS reservation state: State used/kept by Resource Management Function to describe reserved resources for a session.
Flow ID: This is essentially included in the Message Routing Information (MRI) in GIST for path-coupled signaling. Note that QoS-NSLP, currently, supports only path-coupled signaling.
QoS NSLP operation state: State used/kept by QoS NSLP processing to handle messaging aspects. It includes non-persistent and persistent state. The non-persistent state keeps information only during the processing of a message of the session. The persistent state is active during the whole existence of the session and is organized as a table [MaKa06].

3.3.1 Objects and messages

Four message types are specified for QoS NSLP as each has a common header and a body. The body consists of three groups of message objects, which according to the specification [MaKa06] are:
Control information objects: are common for all quality of service models that can be used on top of QoS NSLP. The objects semantics and processing are standardized by the QoS NSLP functionality. The objects are presented in Table 3-2.
QoS specification object: carries the QOSM specific information over resource management and includes the parameters to describe the requested quality of service. The QSPEC object is described in detail in section 3.3.
The content of a *Policy object* is used in the authentication and authorization of the initiator (QNI). The policy control is performed by separate functionality, usually at the boundaries of an administrative domain.
The *common header* provides information on the message type and the possible flags to be set. It precedes all other QoS NSLP objects and its format is given in Table 3-3.
The *RESERVE* message is the only message that can manipulate – create, refresh, modify or release a reservation state in a node. The reservation message can have set the generic flag Scoping and the message specific flags Tear, Acknowledge and Replace. RSN is the only mandatory object, since it is used to maintain up-to-date reservation state. In case a response is required the RII is included. Each Reserve message can have at the most two QSPEC objects – the original and the local QSPEC. The Refresh_period, Bound_session_ID and Policy_data objects are also part of the message format.

Table 3-2 Control information objects

Type	Description
Request Identification Information (RII)	A random, unique per-session number that is used every time a response is desired. The response has to come from the destination, thus it is different form the A flag (see Table 3-3) and has end-to-end importance. It is used to connect a Response for to a Reserve or Query.
Reservation Sequence Number (RSN)	A random number, unique between two peers that is increased every time a modify message is sent from the peer to the next one. It has local, peer-to-peer importance.
Refresh_period	The period used by the node to generate refresh messages in the soft state operation mode.
Bound_session_ID	Contains the Session Identifier (SID) of the session(s) that are bound with the current one. It is used in aggregations or when traversing a domain and using layering or tunneling.
Info_spec	Organized in error classes and error codes which identify a occurrence of event, very often error.
Packet_clasifier	Provides information on the Message Routing Method used and additional information in the form of flags.

Table 3-3 Common header

Field	Flag	Description
Message type	RESERVE	A message used for session initialization.
	QUERY	A message used for network resources check-up.
	RESPONSE	A message used to respond back to previous request.
	NOTIFY	A message used to inform for special events.
Specific flags	Tear (T)	An existing state (reservation and operation) must be terminated.
	Reserve-init (R)	A reverse initiation is required.
	Acknowledge (A)	An explicit confirmation about the state installation is required.
	Replace (R)	The MRI carried in the message replaces the old one, which can be torn down. Used in rout changes.
Generic flags	Scoping (S)	The message is no be forwarded only to the next hop and not the whole path.

A QUERY message is generated for either receiver initiated reservation or if a node wants to check up the available network resources. The free resources are carried in the QSPEC. The Scoping flag and the message specific Reserve-Init flags can be set for this message. If the message is used for informing about network resources an RII object is mandatory. The Response to the Query will have the same RII object and the initiator node can connect the Response to the particular Query. The Query format [MaKa06] also includes the Bound_session_ID and Policy_data objects.

The *RESPONSE* message carries the result of the Reserve or Query message processing on the signaling path [MaKa06]. Only the Scoping flag can be set for a response message. The RSN/RII object relates the Reserve/Query message.
NOTIFY message is initiated asynchronous and is not dependant on previous message or state processing. It is generated upon a network event, most frequently an error. This message has no specified flags [MaKa06].

3.3.2 Quality of service definition (QSPEC)

The Quality of service SPECification (QSPEC) is introduced to represent the quality of service model specifics in terms of message parameters and RMF handling [AsBa06]. The QSPEC is transparent for the QoS NSLP node functionality and is processed by the RMF. The resource management function should define processing scenarios for all QoS models that are currently implemented in the node.
The terminology specified in [AsBa06] and used in this chapter is given below:

QSPEC Control information is QOSM specific and it controls the way of how RMF should process the QSPEC.
The *QSPEC Description* carries the desired by a flow quality of service, represented by QSPEC objects.
A QSPEC Object is a cumulative building block used to represent quality of service level. A QSPEC Object is identified by its Object ID and can be Control information, QoS Desired, QoS Available, QoS Reserved and Minimum QoS. QSPEC objects are generated by the initiator of the signaling session and can be:

- *mandatory objects* that must be present in the QSPEC and the receiver must be able to interpret them;
- *optional objects* which can be included in the QSPEC and should be interpreted by the receiver if present.

QSPEC Parameter is a formal way of describing the quality of service required by a used application. One QSPEC Object can have many QSPEC Parameters. The QSPEC Parameter is identified by a Parameter ID and can be:

- a *read-write parameter* that can be changed by the RMF in every node on the session path;
- a *read-only parameter* that cannot be modified by the intermediate nodes but only by the initiator and the receiver node of the QSPEC.

Control information field gives general instructions to the resource management function such as used QOSM, which procedure to process the QSPEC should be followed, etc.
The *QoS Description* is a collective name of the four QSPEC objects with information on the reservation resources. Only the QoS Desired object is mandatory object while the others are optional. Since QoS Desired is used in the RMD-QOSM the object is presented in details. The optional objects are given general definition.

Table 3-4 QoS Desired object

Parameter	Description
Traffic description	Information for the characteristics of the traffic given by the desired Bandwidth and the Token bucket. Desired in this case is equivalent to generated or the parameters values are taken from the initiator's traffic.
QoS Class	Defines which class the initiator requires to be handled like. Three parameters are possible. Which one is used and its value is QOSM specific.
Priority	Three types of priorities are defined and used to classifies the traffic within the node and its manner of processing.
Path latency	Optional parameter, which when available indicate the node should include then in the QoS Available object. They are used for path evaluation in the End node.
Path jitter	
Path BER	

QoS Desired object transports the initiator request for quality of service. The parameters of the object, see Table 3-4, are read-only and inform every node on the signaling path for the requested QoS.
The *QoS Available object* carries information about the available network resources that is updated by the nodes on the signaling path. The parameters of this object are read-write. In the case of RESERVE or QUERY messages when the QoS Desired is evaluated the local node resources are checked. If they are less than the desired resources modification is done to the QoS Available object. In case of RESPONSE message the QoS Available object transports values of the available network resources.
A *QoS Reserved object* represents the actual resources, reserved on the data path and it has as parameters traffic description, QoS class and priority.
The *Minimum QoS object is* defined by the initiator with read-only parameters. It gives the lower boundary on the quality of service parameters, accepted by the initiator, and allows resource negotiation. This object is evaluated at the receiver and compared with the QoS Available object parameters. If the values of the available resources are below the values in the minimum QoS object parameters the connection cannot be established. If the Minimum QoS object is absent the end node will compare the desired and available objects and if the values of the available object are smaller than the ones of the desired object the connection procedure is aborted.

3.3.3 Example of QoS-NSLP operation

To give an overview of the QoS NSLP operation an example is presented. The detail explanation of them and more additional examples can be found in [MaKa06]. A sender initiated reservation is given in Figure 3.2. Note that the QoS-NSLP messages are encapsulated into GIST messages.

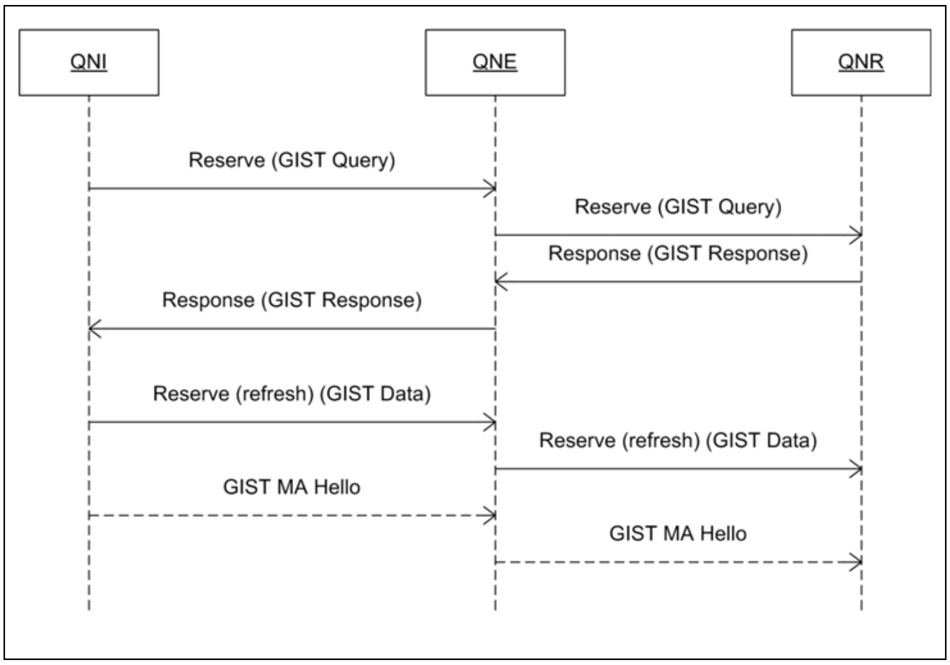


Figure 3.2 Sender initiated reservation

The initiator of the request (QNI) is the sender of the data flow. QNI generates a RESERVE message with the initiator QSPEC. At each intermediate QoS NSLP aware node first authentication and policy control are performed. Second, the control information is processed and the QSPEC is sent to the RMF. The RMF performs the resource check-up and if the reservation is admitted, a reservation and an operational state are installed and the RESERVE message is sent to the next peer. If any of the nodes do not have sufficient available resources, then no states are installed and a RESPONSE message is returned right away to the initiator. When the receiver (QNR) gets RESERVE message if the RII object is included a RESPONSE is initiated. QNR also installs states if the requested resources are free.

3.4 RMD model within QoS NSLP

The RMD QOSM is used when a QoS NSLP message has to traverse a RMD domain. It this case tunneling/bypassing is used and the original RESERVE message is sent unchanged to the egress node while a local RESERVE message is used to make the reservation within the domain. These local messages comply with the format of QoS NSLP messages as presented below. The message objects are as explained in section 3.3.1.
The nodes at the edge of the RMD domain, i.e., ingress and egress, are stateful nodes and keep reservation state per each session. The interior nodes are reduced state and keep reservation state only per PHB class. All nodes in cooperation apply the admission control and congestion detection, notification, handling and solving mechanisms. The functionality of the edge nodes is more complicated that the one of the interior nodes. Both type of nodes have different responsibilities. These mechanisms are specific for each quality of service domain and for RMD are presented in sections 3.4.2 and 3.4.3.

Common header	RSN	RII	Refresh period	Bound Session ID	Policy data	Packet classifier	RMD QSPEC

Figure 3.3 RESERVE message format

Common header	RII	Bound Session ID	Policy data	Packet classifier	RMD QSPEC

Figure 3.4 QUERY message format

Common header	RSN/ RII	INFO SPEC	RMD QSPEC

Figure 3.5 RESPONSE message format

Common header	INFO SPEC	RMD QSPEC

Figure 3.6 NOTIFY message format

3.4.1 RMD-QOSM QSPEC

In the case of RMD-QOSM framework the used model is the Resource Management in DiffServ. In order to reflect the RMD quality of service provisioning the QSPEC consists of three objects: QoS Description, PHR container and PDR container [WeCs02]. The combination of them determines what quality of service is required by the application. Each one of the objects is described below.

3.4.1.1 QoS Description

Only one of the defined in section 3.3.2 objects is included in the QoS Description, carried by a RESERVE message, and that is the QoS Desired object, with two parameters Bandwidth and PHB class. Note that a RESPONSE message carries a QoS Reserved object. The Bandwidth parameter, with ID 3, signals how much resource would be needed by the flow and its format is presented in Figure 3.7. The PHB class parameter, with ID 7, currently carries information of the DSCP value of the flow. The DSCP value is used in the routers to distinguish between different traffic classes. The parameter format is presented in Figure 3.8.

Flags	Parameter ID = 3	Reserved	Length = 1
Bandwidth			

Figure 3.7 Bandwidth parameter format

Flags	Parameter ID = 7	Reserved	Length = 1
DSCP	0	Reserved	

Figure 3.8 PHB Class parameter format

3.4.1.2 Per-hop reservation (PHR) container

The Per-Hop Reservation (PHR) container supports the resource reservation procedure and is processed by all nodes on the flow path. Its format in presented in Figure 3.9 and its parameters are described in Table 3-5

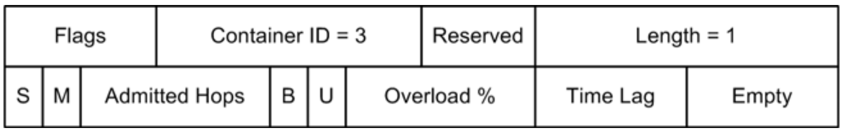


Figure 3.9 PHR container format

3.4.1.3 Per-domain reservation (PDR) container

The Per-Domain Reservation (PDR) container provides additional support to the PHB container for the connection establishment. The PDR container is in the base of the end-to-end communication. It also passes all nodes but is processed only by the edge nodes. Its format is presented on Figure 3.10 and the description of its fields in Table 3-6.

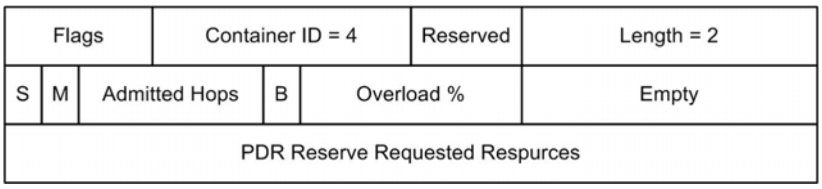


Figure 3.10 PDR container format

Table 3-5 PHR container fields

Parameters	Description
Flags	Flags for service use to support parameter handling.
Container ID 1 2 3	 PHR_RESOURCE_REQUEST Used for resource reservation PHR_REFRESH_UPDATE Used for resource refresh PHR_RELEASE_REQUEST Used for resource release
Reserved	Bits reserved for future use.
Length	The length of the parameter in bytes.
S	Indicates severe congestion occurrence. 0 for no congestion, 1 for congestion.
M	Indicates node possibility to reserve resources. If 1 insufficient resources.
Admitted hops	Counts the number of nodes in which the reservation was successful. Set to 0 in the ingress node.
B	If set bi-directional reservation is requested.
U (Hop_U)	If set indicates the admitted hops counts must not be increased (in case of unsuccessful reservation).
Overload %	Indicates the level of overload detected. Every node checks its own level and if necessary updates it.
Time lag	Used in the refresh procedure.
Empty	All zero bits.

Table 3-6 PDR container fields

Parameters	Description	
Flags	Flags for service use to support parameter handling.	
Container ID		
4	PDR_RESERVATION_REQUEST	Used for resource reservation
5	PDR_REFRESH_REQUEST	Used for resource refresh
6	PDR_RELEASE_REQUEST	Used for resource release
7	PDR_RESERVATION_REPORT	Result of reservation procedure
8	PDR_REFRESH_REPORT	Result of refresh procedure
9	PDR_RELEASE_REPORT	Result of release procedure
10	PDR_CONGESTION_REPORT	Indicates severe congestion
Reserved	Bits reserved for future use	
Length	The length of the parameter in bytes.	
S	Indicates severe congestion occurrence. 0 for no congestion, 1 for congestion.	
M	Indicates node possibility to reserve resources. If 1 insufficient resources.	
Max admitted hops	Counts the number of nodes where the reservation was successful. Set to 0 in the ingress node.	
B	If set bi-directional reservation is in place.	
Overload %	Indicates the level of overload detected. Every node checks its own level and if necessary updates it.	
Empty	All zero bits.	
PDR Reverse Requested Resources	Indicated the resources needed for the reverse direction in case of bi-directional reservation.	

3.4.2 Admission control

Under the term admission control are united procedures that specify how network resources can be checked, when a reservation can be made and when it should be rejected. Within RMD-QOSM two types of admission control are defined, depending on how the network resources are represented and how the reservations are organized.

Measurement based

The measurement based method uses real bandwidth measurement of the link utilization. No reservation state is kept in the interior nodes. The only information kept is the total reserved resources per PHB class and a threshold for each class. The threshold is the maximum number of resources that this PHB class can reserve. Usually, the thresholds are configured as percentages of the total bandwidth that can be supported by a link.

The measurement based approach can be applied in two ways. In the first possibility RESERVE messages are regularly sent in the RMD domain and used as probes. When the threshold is exceeded the message is marked and local policy determines whether new requests can be accepted.

In the second scenario information on the current traffic level per PHB class in a node is kept along with the threshold specified for the PHB class. If a new request summed with the measured occupied bandwidth is above the threshold it is rejected. In the pervious described admission control method, the interior nodes are not NSIS aware, while here the interior nodes are NSIS aware. This means that these nodes can process the RMD-QOSM information carried by the NSIS signaling messages.

Reservation based

In RMD, usually, the link bandwidth is represented by resource units with a pre-defined bandwidth value. In this case the interior nodes keep reservation state per each PHB class. This state is represented in means of resource units (or bandwidth) and is a soft state, which has to be refreshed periodically. Upon each RESERVE message the new requested resources are added to the existing reservation only if this sum does not exceed a predefined PHB resource/bandwidth threshold.

3.4.3 Severe congestion

When a network link, or router, breaks the flows that it transfers have to be re-routed via other links. These other links support their own traffic and if other flows get re-routed through them, then it is possible that these links become severely overloaded and congested. Such situations are denoted as severe congestion situations and can be very harmful for the communication network. If the overload is not solved fast enough, then all sessions that use the particular link will suffer unacceptable degradation in the quality of service. Therefore it is crucial that severe congestion situations are quickly detected solved. The detection process can use the fact that the load on a line gets above a pre-defined threshold. If this happens certain measures have to be taken so the load drops back to link capacity. Severe congestion is typically detected in the interior nodes. Since these are normally reduced state nodes they do not maintain per flow information and cannot make decisions on flows terminated. Therefore state aware edge nodes are notified about the bandwidth that causes the overload and then they can reduce it, i.e., they provide severe congestion handling. There are two methods of severe congestion notification, one based on signaling messages, another – on marking of the data packets.

Severe congestion notification and handling by marking (refresh) RESERVE messages

When for the severe congestion detection signaling messages are used a (refresh) RESERVE message is marked as severe congested (S flag marked) and also carries the degree of the congestion (Overload %). The reasoning of this is as follows: if the requested resources in the (refresh) RESERVE message cannot be confirmed, but the flow was accepted then there has occurred change in the link load that is endangering the existing flows. Therefore, the edge node should be informed and try to solve the problem. That is done by the congestion handling mechanisms.

Severe congestion notification by marking data packets

For congestion notification the data packets can also be used. In this case two additional DSCPs can be used. The one will only inform that a data packet has passed a congested

node, denoted as “affected DSCP”, and the second will be used to inform the rate of congestion, denoted as “encoded DSCP”. The mechanism used is called rate proportionate marking. In other words the number (or rate) of marked data packets is proportional to the level of the severe congestion.
Each interior node counts how many bytes in total arrive in one measurement period (N_{tot}). Then it calculates how many bytes are above the congestion threshold (N_{sc}). The level of congestion is calculated as the ratio between the bytes above the threshold and the total passing bytes – N_{sc}/N_{tot}. The node marks number of bytes equal to the congestion level (N_m) and remembers that number. In the next measurement period the same procedure is followed only the number of bytes to mark is decreased with N_m, resulting in $N_{sc}/N_{tot} - N_m$. This is a sliding window mechanism where the window size is the number of periods for which the node keeps memory. If the memory is not big enough the node will mark more than the necessary bytes to solve the congestion or in other word it will undershoot.

Severe congestion handling when data packets are marked
In order to solve the severe congestion flows that contribute to the severe congestion, have to be terminated. Edge nodes can do that based on the proportion of severe congestion they learn from the interior nodes. The proportion of the severe congestion is calculated by measuring the rate of the marked (as “encoded DSCP”) packets that are arriving at the egress node. The total number of marked bytes gibes the total congested bandwidth.
Each flow has reserved a certain bandwidth. The number of flows to be terminated should be chosen such that the sum of their reserved bandwidth is larger than the total congested bandwidth. A flow is chosen for termination and its bandwidth is subtracted from the total congested bandwidth. If the result of the subtraction is still positive, another flow has to be terminated and so on. Given that certain PHB class messages are marked as severe congested, only flows from the same PHB class should be terminated.
When more than one flow priorities have to be supported within one PHB class flow termination will start from the flows with lower priorities. If their bandwidth is not enough to solve the severe congestion, the calculation step moves one priority level above. The mechanism is specified in [BaWe06].
The interior nodes will mark bytes per PHB group not considering the packet priority. In order to keep the priority principle the edge node, after all marked flow are stopped, should move to the affected flows from the same priority before it goes on to the next priority level. The particular criterion to choose flows is an implementation decision. It is possible to begin from the biggest flows within a priority class, from the smallest flows or with the flow which bandwidth is the closest to the total severe congested bandwidth.

3.5 RMD QoS NSLP Unidirectional Operation

All signaling situations that can occur in the case of one way communication between two end nodes – unidirectional operation are presented with the reservation orientated admission control. For precise explanation how do the fields of the RMD containers change during the message exchange the reader is advised to look at the protocol specification [BaWe06]. The edge nodes keep per flow reservation state and the interior nodes reduced state and as such they keep aggregated reservation per PHB class.

3.5.1 Successful reservation procedure

A reservation procedure is the sequence of processes connected with initiation of a reservation request, its propagation in the data path, the check up of available resources in the interior nodes on the data path, creation of reservation states and generation of response for the result of the resource request. When all nodes on the data path can support the reservation request, the reservation procedure is called to be successful. The message sequence and the state changes are shown on Figure 3.11.

The ingress node, after it admitted the requested resources and has installed a reservation state, generates two messages – one end-to-end and one local intra domain RESERVE message. The end-to-end RESERVE message should not be processed in the interior nodes and therefore it uses different RAO (Router Alert Value) than typically. The local RESERVE message is initiated by the ingress node and it carries a RMD-QOSM QSPEC. It is the responsibility of the ingress node to bind the two messages and to inform the egress about that relation using the BOUND_SESSION_ID.

The interior nodes apply the admission control mechanism and during a successful reservation, the reservation state is increased with the requested resource units. When the egress receives the local RESERVE it binds it with the corresponding end-to-end RESERVE message. The end-to-end RESERVE is forwarded (the reservation is successful) but with removed marking, used to bypass the interior nodes.

When a successful RESPONSE is received at the egress it is sent back directly to the ingress. The ingress has to transmit the RESPONSE message further so the data initiator is informed and can start transmission of the user data. Using the BOUND_SESSION_ID the local RESPONSE can be related to the proper per-flow end-to-end reservation state in the edge nodes.

3.5.2 Unsuccessful reservation procedure

In the cases when one or more of the nodes on the data path cannot accept the RESERVE message the reservation procedure fails. This is called unsuccessful reservation procedure and it is presented in Figure 3.12. In such situations additional measures should be taken so the data initiator is notified about the failed request and the network state is the same as before the request was made.

In case the admission control fails in a node the M flag in the RESERVE message is set and no states are installed. Further, the flag Hop_U has to be set that to instruct the subsequent nodes not to increase the Admitted hops value. This is to be used in the release procedure. All nodes receiving M marked RESERVE do not process it. The egress generated RESPONSE message with set M flag and the value of Admitted hops taken from the RESERVE message. The M marked RESPONSE starts in the ingress node a release procedure, which is described in Section 4.5.4.2.

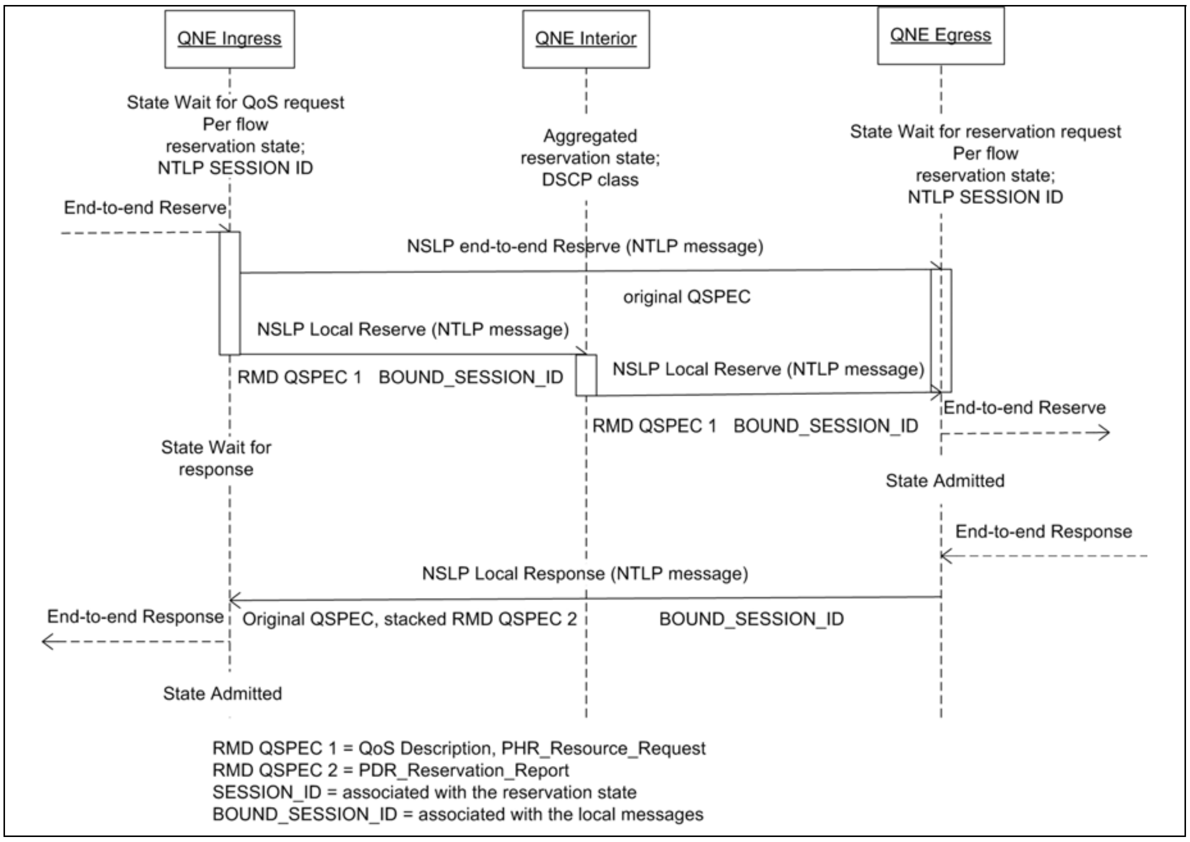

QNE Ingress
QNE Interior
QNE Egress
State Wait for QoS request
Per flow
reservation state;
NTLP SESSION ID
Aggregated
reservation state;
DSCP class
State Wait for reservation request
Per flow
reservation state;
NTLP SESSION ID
End-to-end Reserve
NSLP end-to-end Reserve (NTLP message)
original QSPEC
NSLP Local Reserve (NTLP message)
RMD QSPEC 1 BOUND_SESSION_ID
NSLP Local Reserve (NTLP message)
End-to-end Reserve
RMD QSPEC 1 BOUND_SESSION_ID
State Wait for response
State Admitted
End-to-end Response
NSLP Local Response (NTLP message)
End-to-end Response
Original QSPEC, stacked RMD QSPEC 2
BOUND_SESSION_ID
State Admitted
RMD QSPEC 1 = QoS Description, PHR_Resource_Request
RMD QSPEC 2 = PDR_Reservation_Report
SESSION_ID = associated with the reservation state
BOUND_SESSION_ID = associated with the local messages

Figure 3.11 Successful reservation procedure

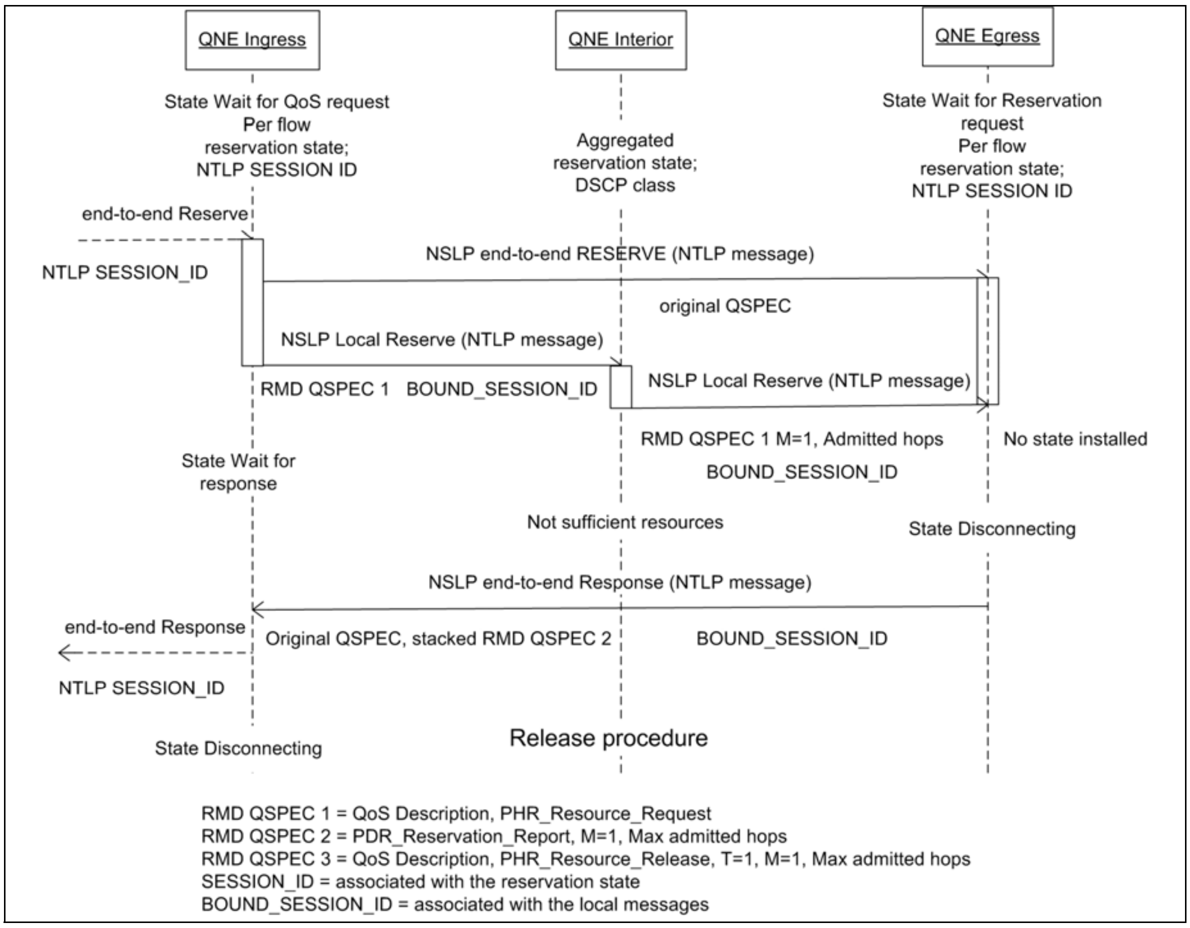

QNE Ingress
QNE Interior
QNE Egress
State Wait for QoS request
Per flow
reservation state;
NTLP SESSION ID
Aggregated
reservation state;
DSCP class
State Wait for Reservation
request
Per flow
reservation state;
NTLP SESSION ID
end-to-end Reserve
NTLP SESSION_ID
NSLP end-to-end RESERVE (NTLP message)
original QSPEC
NSLP Local Reserve (NTLP message)
RMD QSPEC 1 BOUND_SESSION_ID
NSLP Local Reserve (NTLP message)
RMD QSPEC 1 M=1, Admitted hops
BOUND_SESSION_ID
No state installed
State Wait for response
Not sufficient resources
State Disconnecting
NSLP end-to-end Response (NTLP message)
end-to-end Response
Original QSPEC, stacked RMD QSPEC 2
BOUND_SESSION_ID
NTLP SESSION_ID
State Disconnecting
Release procedure
RMD QSPEC 1 = QoS Description, PHR_Resource_Request
RMD QSPEC 2 = PDR_Reservation_Report, M=1, Max admitted hops
RMD QSPEC 3 = QoS Description, PHR_Resource_Release, T=1, M=1, Max admitted hops
SESSION_ID = associated with the reservation state
BOUND_SESSION_ID = associated with the local messages

Figure 3.12 Unsuccessful reservation procedure

3.5.3 Refresh procedure

As it was mentioned, QoS NSLP is a soft state protocol and as such the states that it installs have limited lifetime. When this lifetime has expired if the signaling application has not re-confirmed the reserved resources they are freed and can be occupied by other sessions. To keep the reservation active a refresh procedure is initiated – before the reservation lifetime has expired a (refresh) RESERVE message is sent to re-confirm the used resources.

The refresh procedure appears only in the reservation based method since it keeps reservation states. A full description of the procedure and the message fields values are given in [BaWe06].

The ingress node is responsible for the generation of the (refresh) RESERVE messages. Each interior node refreshes the resources specified in the Bandwidth parameter of the refresh message. The reservation state in each refresh period is actually rebuilt or the reserved resources per each class start from zero and are re-created based on the refresh messages. In case the refresh is unsuccessful, the message is M marked to inform the edge node about the unsuccessful refresh.

The Egress node has the responsibility to generate a RESPONSE message, built upon the fields of the (refresh) RESERVE message. The egress node uses the SESSION_ID, carried by the (refresh) RESERVE message, to identify the ingress that has to receive the RESPONSE.

3.5.4 Release procedure

When an existing reservation is no more requested or due to some network changes it is rejected, the reservation state has to be deleted. A special (release) RESERVE message is used to free the resources used by the particular session.

During the release procedure an interior reduced state node that receives a (release) RESERVE message should release the resources for the signaled session. This is done by subtracting the resource units specified in the message from the current reservation state for the PHB class. The PHB class reservation state is recognized using the DSCP value of the (release) RESERVE message. Before the state is released, it has to be calculated if the release is for this refresh period or for the previous one. If the message was sent for the previous refresh period the resources were automatically released and when the node decreases the current reservation state it actually releases resources of another session.

A special value, called Time Lag is carried by the (release) RESERVE. Time Lag contains the time difference between the last refresh and the release generation. The same difference is calculated in each node processing the release message. The resources will be released only if the value withdrawn from Time Lag is smaller than the calculated. The exact calculation procedure can be found in [BaWe06].

The release procedure can be triggered by an explicit (release) RESERVE message from the data sender (initiator) when the reservation is no more required; or by special marked RESPONSE or NOTIFY messages, when the network cannot support the reservation anymore; or by M marked RESPONSE in the special case of unsuccessful reservation [BaWe06]. Since some of the nodes have possibly created reservation state and consequently this state has to be removed. The latter procedure is called partial release.

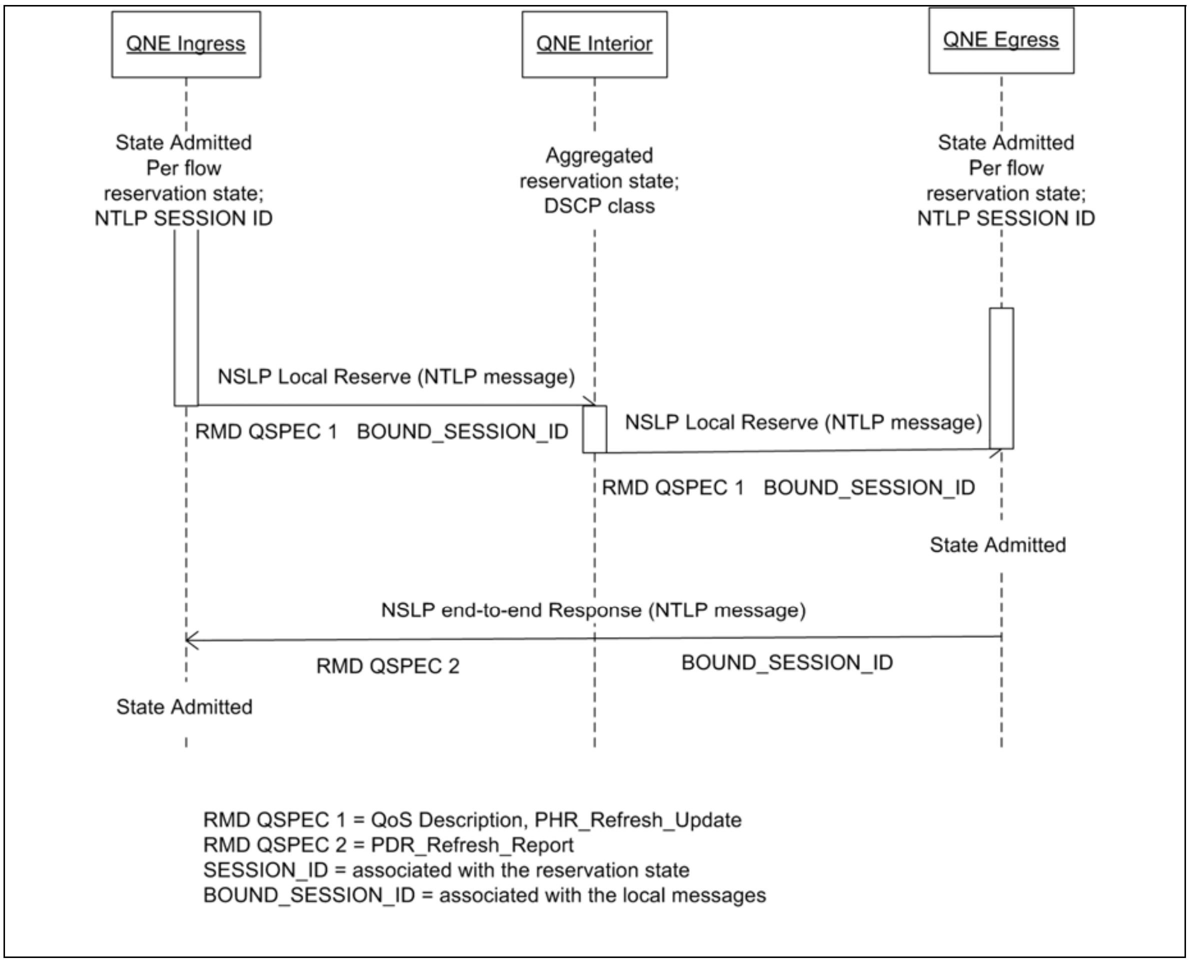

QNE Ingress
QNE Interior
QNE Egress
State Admitted Per flow reservation state; NTLP SESSION ID
Aggregated reservation state; DSCP class
State Admitted Per flow reservation state; NTLP SESSION ID
NSLP Local Reserve (NTLP message)
RMD QSPEC 1 BOUND_SESSION_ID
NSLP Local Reserve (NTLP message)
RMD QSPEC 1 BOUND_SESSION_ID
State Admitted
NSLP end-to-end Response (NTLP message)
RMD QSPEC 2
BOUND_SESSION_ID
State Admitted
RMD QSPEC 1 = QoS Description, PHR_Refresh_Update
RMD QSPEC 2 = PDR_Refresh_Report
SESSION_ID = associated with the reservation state
BOUND_SESSION_ID = associated with the local messages

Figure 3.13 Refresh procedure

3.5.4.1 Release due to T marked RESERVE message

A RESERVE message with set TEAR flag indicates desire of the initiator to stop the session. The TEAR flag informs the NSIS aware nodes that a release procedure should start (Figure 3.14). The procedure is very familiar to the reservation establishment procedure with some difference in field settings and the reservation state management. The (release) RESERVE will trigger precondition check and afterwards release of resources. This process happens in each node on the path of the message.
Along with a local RESERVE message an end-to-end RESERVE message is sent. At the egress node both messages are coupled and the end-to-end message is processed further only if the local message has arrived. The end-to-end message, just as in the reservation request scenario, is marked to bypass the interior nodes in the domain.

3.5.4.2 Release due to M marked RESPONSE

The release procedure is started by a RESPONSE message with marked M flag and is presented in Figure 3.15. That occurs as result of unsuccessful reservation procedure. The ingress copies the value of Max admitted hops from the RESPONSE message in the new (release) RESERVE message. This messages triggers check of the Time Lag value and release of resources in each node it reaches. In this case the combination S=0 and M=1 makes each node to compare the values of Max admitted hops and Admitted hops. If these are equal the node is the last one that made successful reservation. The release message should not be forwarded otherwise it would release resources of other sessions.

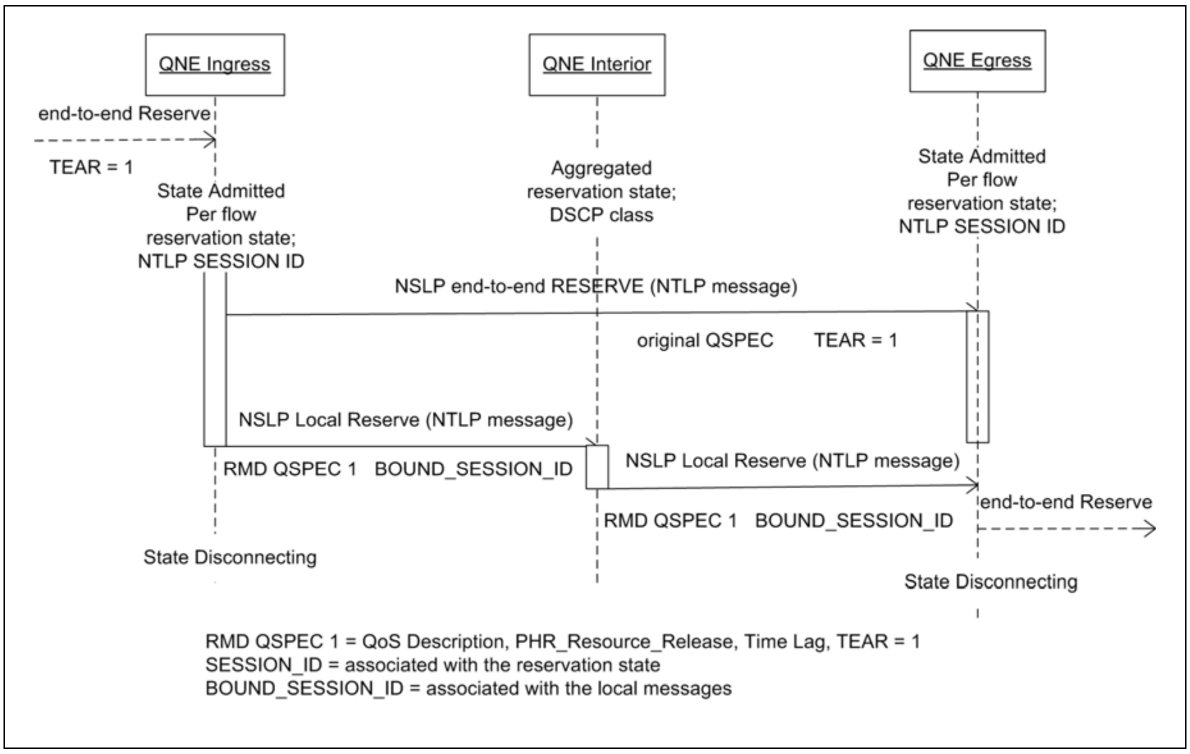

QNE Ingress
QNE Interior
QNE Egress
end-to-end Reserve
TEAR = 1
State Admitted
Per flow
reservation state;
NTLP SESSION ID
Aggregated
reservation state;
DSCP class
State Admitted
Per flow
reservation state;
NTLP SESSION ID
NSLP end-to-end RESERVE (NTLP message)
original QSPEC TEAR = 1
NSLP Local Reserve (NTLP message)
RMD QSPEC 1 BOUND_SESSION_ID
NSLP Local Reserve (NTLP message)
RMD QSPEC 1 BOUND_SESSION_ID
end-to-end Reserve
State Disconnecting
State Disconnecting
RMD QSPEC 1 = QoS Description, PHR_Resource_Release, Time Lag, TEAR = 1
SESSION_ID = associated with the reservation state
BOUND_SESSION_ID = associated with the local messages

Figure 3.14 Release procedure triggered by T marked RESERVE

3.5.4.3 Release due to S marked RESPONSE

In a congestion detection procedure using the refresh messages a RESPONSE is S marked. The (release) RESERVE message will be marked with M=1 and S=1 and it will not be terminated in the interior nodes. The message has to release resources on the whole path and to reach the egress node. Again the reservation state for the correct PHB class is decreased by subtraction and using the Time Lag. The message exchange is presented on Figure 3.16 and the header precise filed setting can be found in [BaWe06].

3.5.4.4 Release due to NOTIFY message

In case of severe congestion detection, a NOTIFY message will be sent to the ingress node. It will contain an Error class: Transient failure, Error code: Transient RMF related error and Error sub-code: Severe congestion. The ingress node will then initiate a release procedure similar to the release by S marked RESPONSE. In this case the S flag is also set. Otherwise the process of decreasing the reservation state in each node and the use of the Time lag is the same as discussed so far. The message exchange is shown in Figure 3.17.

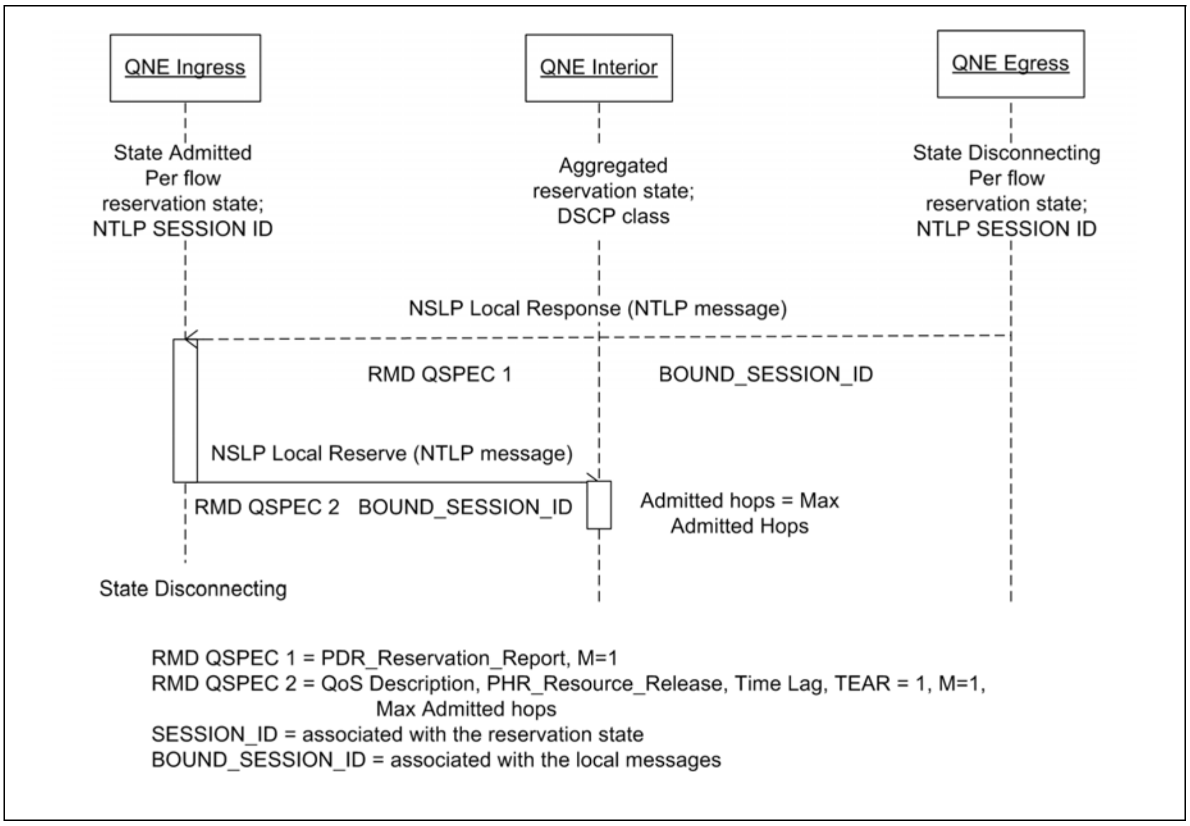

QNE Ingress
QNE Interior
QNE Egress
State Admitted Per flow reservation state; NTLP SESSION ID
Aggregated reservation state; DSCP class
State Disconnecting Per flow reservation state; NTLP SESSION ID
NSLP Local Response (NTLP message)
RMD QSPEC 1
BOUND_SESSION_ID
NSLP Local Reserve (NTLP message)
RMD QSPEC 2 BOUND_SESSION_ID
Admitted hops = Max Admitted Hops
State Disconnecting
RMD QSPEC 1 = PDR_Reservation_Report, M=1
RMD QSPEC 2 = QoS Description, PHR_Resource_Release, Time Lag, TEAR = 1, M=1, Max Admitted hops
SESSION_ID = associated with the reservation state
BOUND_SESSION_ID = associated with the local messages

Figure 3.15 Release procedure triggered by M marked RESPONSE

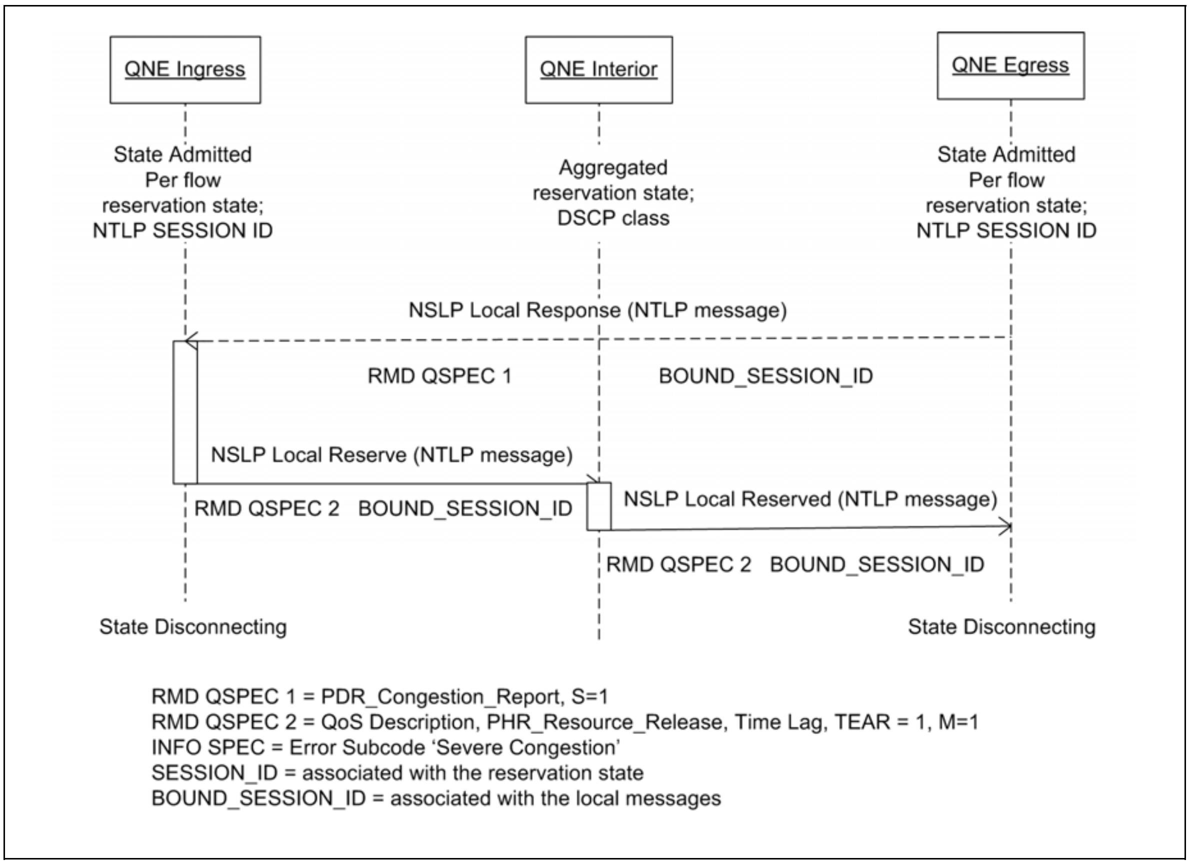

QNE Ingress
QNE Interior
QNE Egress
State Admitted Per flow reservation state; NTLP SESSION ID
Aggregated reservation state; DSCP class
State Admitted Per flow reservation state; NTLP SESSION ID
NSLP Local Response (NTLP message)
RMD QSPEC 1
BOUND_SESSION_ID
NSLP Local Reserve (NTLP message)
RMD QSPEC 2 BOUND_SESSION_ID
NSLP Local Reserved (NTLP message)
RMD QSPEC 2 BOUND_SESSION_ID
State Disconnecting
State Disconnecting
RMD QSPEC 1 = PDR_Congestion_Report, S=1
RMD QSPEC 2 = QoS Description, PHR_Resource_Release, Time Lag, TEAR = 1, M=1
INFO SPEC = Error Subcode 'Severe Congestion'
SESSION_ID = associated with the reservation state
BOUND_SESSION_ID = associated with the local messages

Figure 3.16 Release procedure triggered by S marked RESPONSE

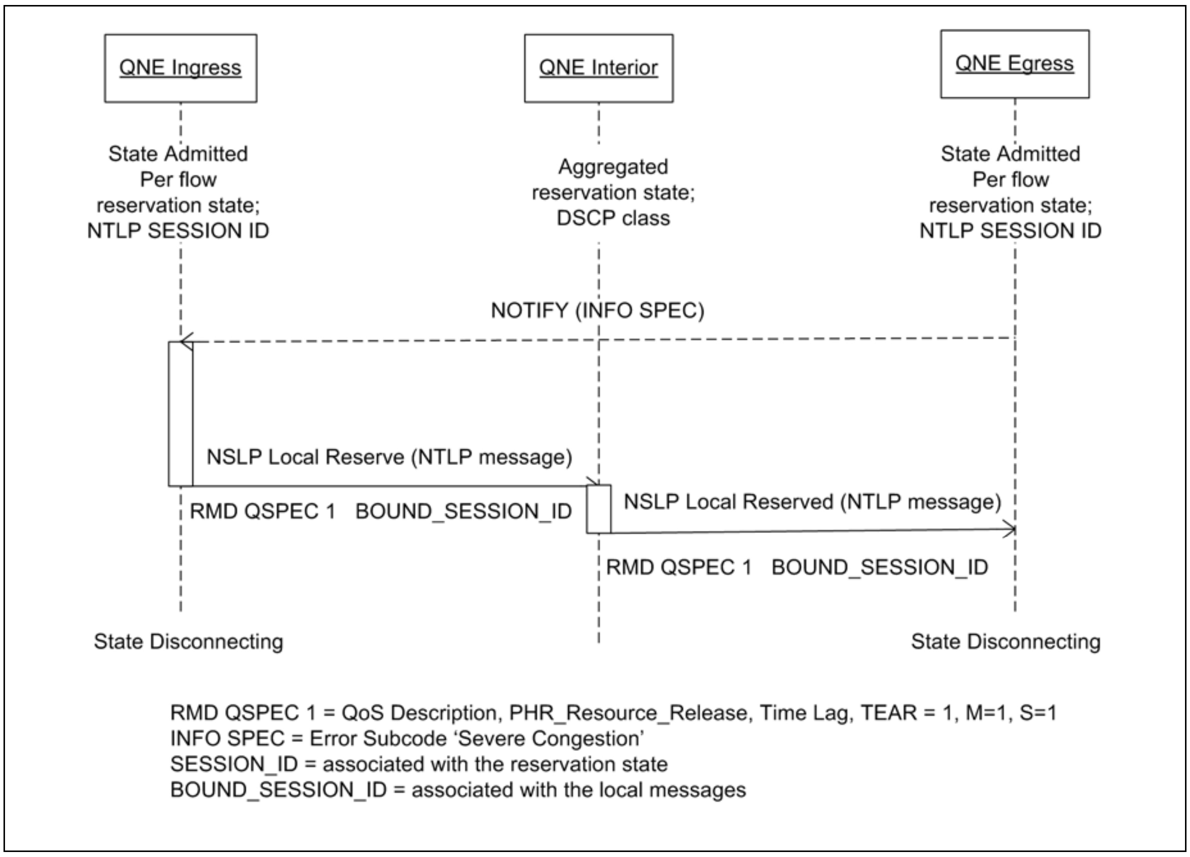

QNE Ingress
QNE Interior
QNE Egress
State Admitted Per flow reservation state; NTLP SESSION ID
Aggregated reservation state; DSCP class
State Admitted Per flow reservation state; NTLP SESSION ID
NOTIFY (INFO SPEC)
NSLP Local Reserve (NTLP message)
RMD QSPEC 1 BOUND_SESSION_ID
NSLP Local Reserved (NTLP message)
RMD QSPEC 1 BOUND_SESSION_ID
State Disconnecting
State Disconnecting
RMD QSPEC 1 = QoS Description, PHR_Resource_Release, Time Lag, TEAR = 1, M=1, S=1
INFO SPEC = Error Subcode 'Severe Congestion'
SESSION_ID = associated with the reservation state
BOUND_SESSION_ID = associated with the local messages

Figure 3.17 Release procedure triggered by NOTIFY

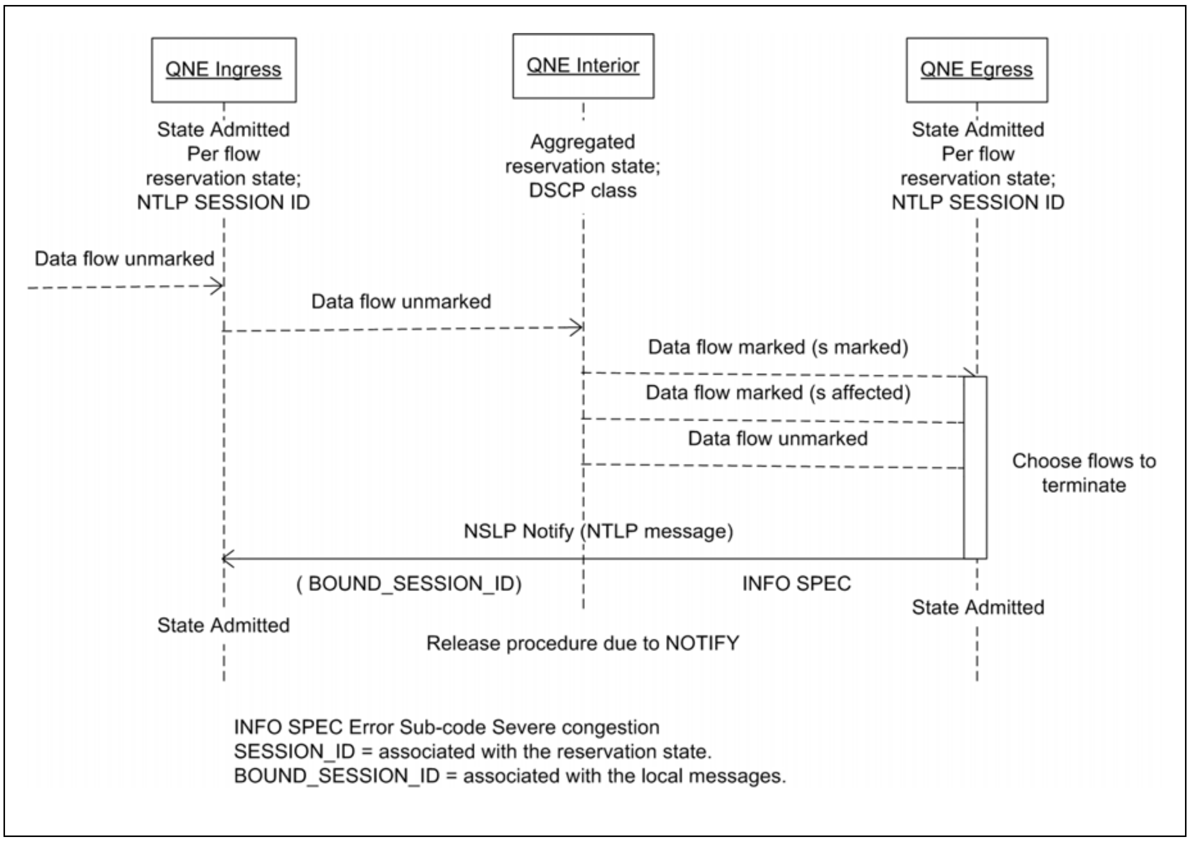

QNE Ingress
QNE Interior
QNE Egress
State Admitted Per flow reservation state; NTLP SESSION ID
Aggregated reservation state; DSCP class
State Admitted Per flow reservation state; NTLP SESSION ID
Data flow unmarked
Data flow unmarked
Data flow marked (s marked)
Data flow marked (s affected)
Data flow unmarked
Choose flows to terminate
NSLP Notify (NTLP message)
(BOUND_SESSION_ID)
INFO SPEC
State Admitted
State Admitted
Release procedure due to NOTIFY
INFO SPEC Error Sub-code Severe congestion
SESSION_ID = associated with the reservation state.
BOUND_SESSION_ID = associated with the local messages.

Figure 3.18 Severe congestion procedure using data packets

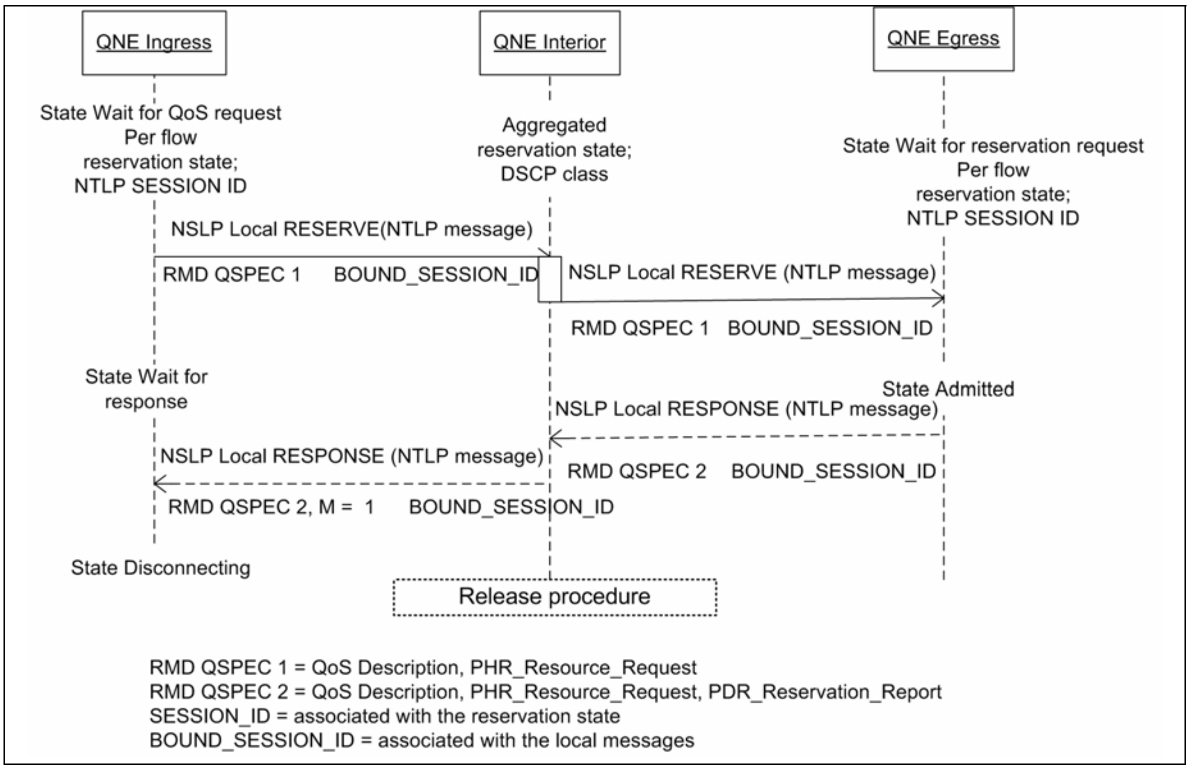

QNE Ingress
QNE Interior
QNE Egress
State Wait for QoS request Per flow reservation state; NTLP SESSION ID
Aggregated reservation state; DSCP class
State Wait for reservation request Per flow reservation state; NTLP SESSION ID
NSLP Local RESERVE(NTLP message)
RMD QSPEC 1 BOUND_SESSION_ID
NSLP Local RESERVE (NTLP message)
RMD QSPEC 1 BOUND_SESSION_ID
State Wait for response
State Admitted
NSLP Local RESPONSE (NTLP message)
RMD QSPEC 2 BOUND_SESSION_ID
NSLP Local RESPONSE (NTLP message)
RMD QSPEC 2, M = 1 BOUND_SESSION_ID
State Disconnecting
Release procedure
RMD QSPEC 1 = QoS Description, PHR_Resource_Request
RMD QSPEC 2 = QoS Description, PHR_Resource_Request, PDR_Reservation_Report
SESSION_ID = associated with the reservation state
BOUND_SESSION_ID = associated with the local messages

Figure 3.19 Unsuccessful reservation procedure for reverse direction

3.5.5 Severe congestion procedure

The message sequence of the severe congestion procedure based on data marking, as presented in section 3.4.3.2, is depicted in Figure 3.18.

An interior node can discover a severe congestion situation occurrence by using the severe congestion detection mechanism. It will then use the specified marking procedure to inform the egress node about the level of the severe congestion. In the egress the severe congestion handling mechanism will be used. For every flow that has to be stopped, the egress sends NOTIFY towards the ingress node. The NOTIFY message will have an INFO SPEC with the following information – Error class: Transient failure, Error code: Transient RMF related error and Error sub-code: Severe congestion. It is very important is that the original DSCP values of the data packets should be restored, otherwise the severe congestion handling mechanism can be applied by the next domain, which may cause problems.

The last step is the release procedure in the ingress node when the marked flows are stopped as it was discussed in chapter 3.5.4.4.

3.6 QoS NSLP Bidirectional Operation

3.6.1 Successful and unsuccessful reservation procedure

The reservation of the resources on the data path is the same as in the case of unidirectional reservations, only this procedure has to be applied for the forward and the reverse path. Therefore, RESERVE messages have to be sent in both directions. That leads to a change in Figure 3.11, where the egress does not send RESPONSE message back but a RESERVE message with a PDR container response and a PHR container reservation request. The

reservation procedure can be also seen in Figure 3.19 with the difference that the last RESESRVE message is not M marked.
A node that does not have enough available resources can be located on the forward as well as on the reverse path. Thus two scenarios of unsuccessful reservation procedures are possible. When the reservation fails on the forward path the procedure is the same as presented in Figure 3.12. If the reservation fails on the reverse path the RESERVE message from egress to ingress will be M marked (Figure 3.19). The full description of the procedures is given in [BaWe06].

3.6.2 Refresh and release procedure

Both procedures are very similar to the procedures for the unidirectional scenario. The schemes that were presented in sections 3.5.3 and 3.5.4 can represent the bi-directional operation with only one difference – the egress sends to the ingress a RESERVE message that carries in its PDR container the response for the forward direction. The RESERVE messages on the reverse path are processed by all nodes on the path. When the ingress receives the RESERVE it learns the result for both directions – the PDR container informs for the forward and the PHR container for the reverse. For in detail explanation of the changes in the message fields [BaWe06] can be consulted.

3.6.3 Severe congestion procedure

Just as in the case of reservation procedure the data flow goes in two directions – forward and reverse. These two paths can differ in intermediate nodes in other words they can traverse different links. A link break can occur on the forward or on the reverse path or on both. As result, severe congestion situations can occur on both paths and consequently data marking too. If a link on the forward path breaks the scenarios is identical as the one in the unidirectional situation, Figure 3.18. The severe congestion procedure with link break on the reverse path is presented in Figure 3.20. In this case the marked data packets arrive at the ingress that chooses flows to terminate. Since the ingress is the node that stops the flows no NOTIFY message is generated. The full severe congestion procedure and the format of the messages are described in [BaWe06].
In bidirectional reservations there is the possibility severe congestion situation to happen on both paths of the data exchange. Data packets are marked on the forward and on the reverse path [Figure 3.21]. The severe congestion handling is applied in both edge nodes. The egress node calculates the overload, chooses flow for termination and sends NOTIFY for each of these flows a NOTIFY message is sent and when it reaches the ingress the flow is released. At the same time the ingress also receives marked data packets and calculates severe congestion level on the reverse path. Flows are chosen and stopped right away. At the end the traffic load on both paths – forward and reverse, is stabilized bellow or equal to the link capacity.

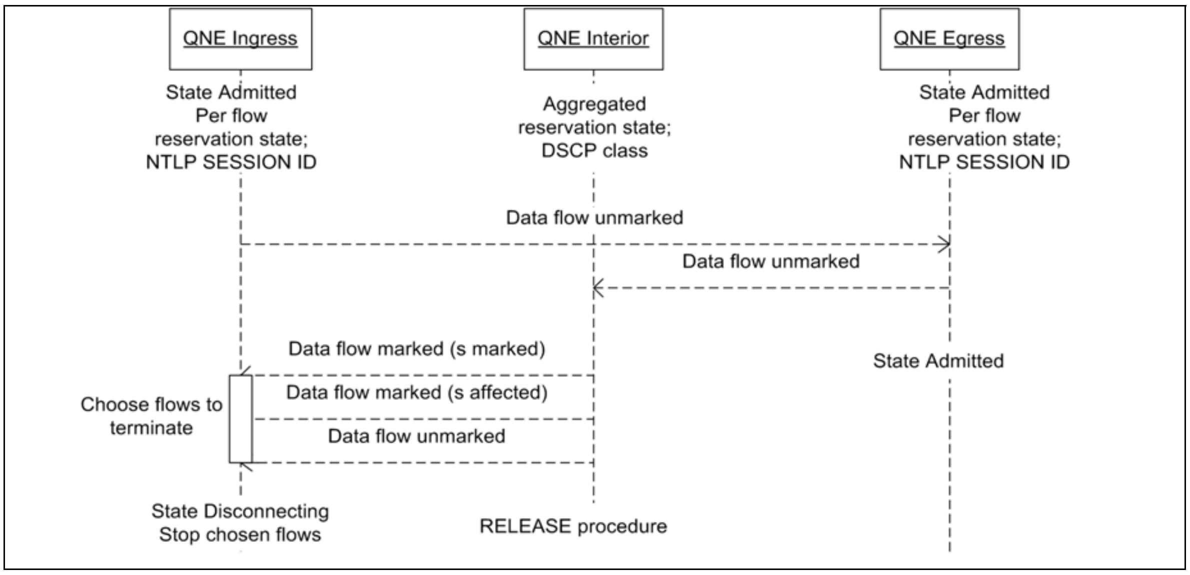


Figure 3.20 Severe congestion procedure on the reverse path

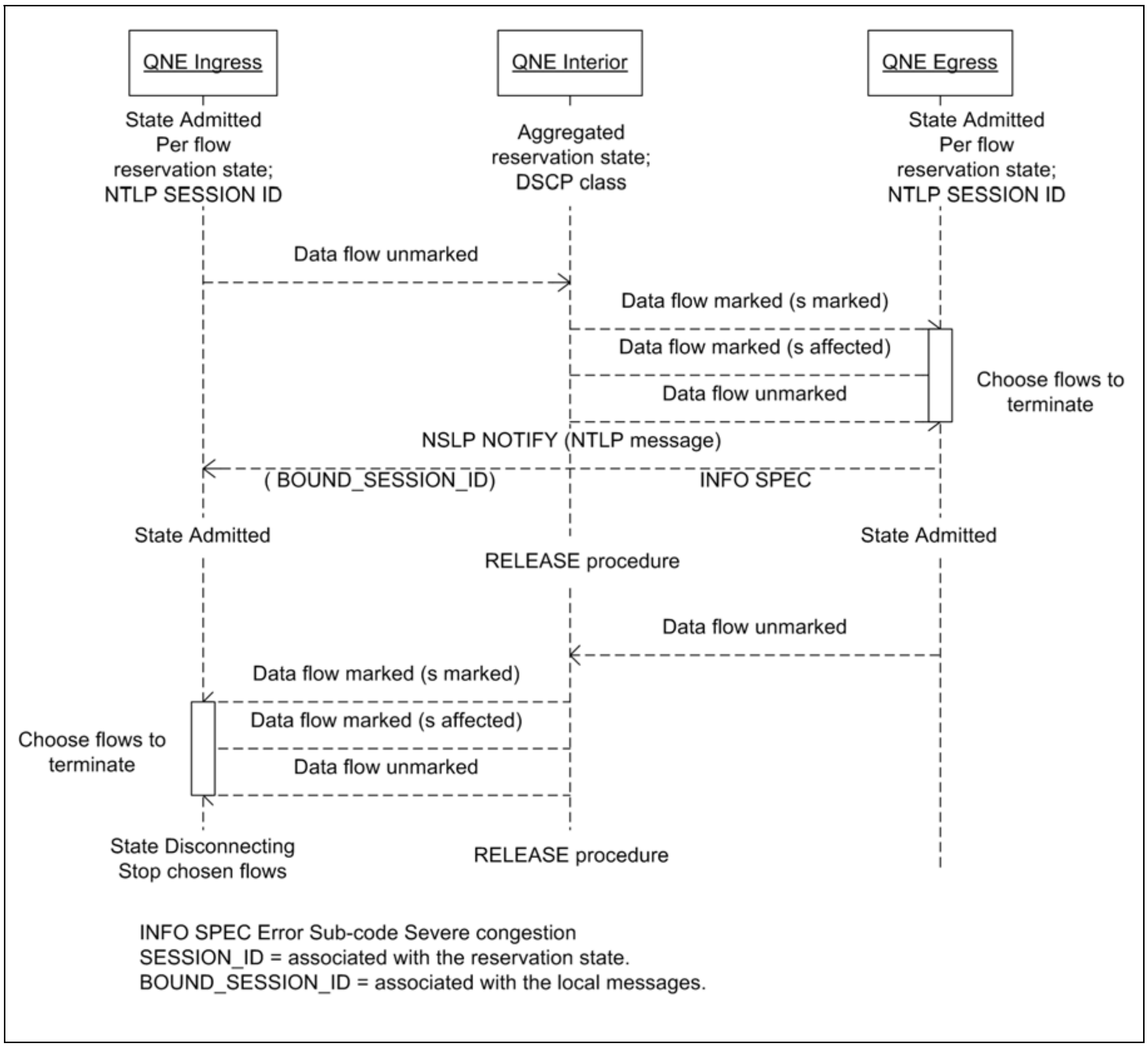


Figure 3.21 Severe congestion on both paths

4 Base RMD simulation model

In chapter 1 the DiffServ idea was introduced. A special protocol framework, Resource Management in DiffServ (RMD), is developed to specify how the network nodes should apply the DiffServ idea and how resources should be managed. In order to reach these goals in RMD two protocols are used – Per Hop Reservation protocol and Per Domain Reservation protocol. The former is manages the resource reservations within a DiffServ domain interior nodes, and the latter at the borders of the domain, edge nodes. The main functional principles of the RMD framework are described in section 4.1.

The RMD framework also includes processes of admission control, or under what conditions a request can be granted, and severe congestion handling, or how severe congestion can be detected and solved. However no particular mechanisms to apply these processes are specified. A variety of possible mechanisms are discussed in the literature. Some of them, used in the base RMD simulation model, are presented in section 4.2 and their general operation is explained.

Based on the RMD framework [WeJa03] a simulation model was developed to evaluate the RMD performance. This initial model, see Chapter 2, was continuously extended to include newer versions of the above mentioned mechanisms for admission control and severe congestion detection and solving. Eventually it is characterized with relative complexity and is multifunctional. Section 4.3 describes the RMD simulation model implementation in ns2 simulator, with sub-chapters dedicated to each particular RMD functionality. In section 4.3.1, first the node implementation is presented. Section 4.3.2 presents the used communication links and their queuing models. Sections 4.3.3 concentrates on the edge nodes state transitions, i.e., state machine. The next three sections from 4.3.4 to 4.3.6 describe the implementation of admission control, severe congestion detection and the severe congestion solving mechanisms, respectively. Section 4.3.6 presents some classes that have supportive monitoring function. The message sequence diagrams used by the RMD framework simulation model are presented in section 4.3.7.

The RMD simulation model, called also base simulation model, is of interest because it is used as a starting base for the creation of the new RMD-QOSM simulation model, goal of this research. This chapter is useful on its own too as documentation for the existing RMD model, something that was not done so far.

Finally section 4.4 uses a state transition test to verify whether the existing model conforms to the RMD framework specification.

4.1 RMD protocol framework

The RMD protocol framework [WeJa03] aims at provisioning of resource management in a DiffServ domain. Under resource management it is understood reservation and release of resources and control of the reservations to provide optimal network utilization.

There are two ways to reserve resources in a RMD domain – measurement based approach and reservation based approach. For both approaches the edge nodes are ststeful, or a reservation state for each flow is kept. In the measurement based approach the interior nodes are stateless, meaning that no reservation states are kept in the nodes. For each PHB

class the following are maintained: the measured traffic associated with the PHB class and the maximum allowed traffic for that PHB class. This approach relies on a measurement algorithm for admission control and congestion detection.
In the reservation based approach each interior node maintains an aggregated reservation state for each PHB class. The link capacity is split into resource units with fixed size and the aggregated reservation state represents the resource units occupied by flows of certain PHB class. Each per PHB class has a maximum allowed reservation threshold that is used for admission control and congestion detection.

4.1.1 Nodes

In the RMD framework three types of nodes are used – ingress and egress nodes (edge nodes) and interior nodes. The edge nodes process PDR and PHR messages and handle the end-to-end communication in the domain. The interior nodes work with the PHR protocol, which is responsible for the management of the resource reservations. Since the resource reservation management is related to admission control and congestion detection the PHR protocol is designed to support these mechanisms. A per flow congestion handling solution cannot be performed by the interior nodes since they maintain only per PHB class reservation state and cannot recognize separate flows. The congestion handling, i.e., choosing flows to terminate in order to solve the occurred congestion is done by the edge nodes.
The mechanisms that are used for admission control and congestion handling are discussed in section 4.2.

4.1.2 Messages

This section presents the message types defined for the PHR and PDR protocols and their semantics. Each message has a PHR type and corresponding PDR type so it can be processed by interior and by edge nodes. Sometimes additional message types are introduced in the RMD simulation model. These types are not part of the specification and their use in the simulation model is purely formal to simplify the message recognition in the node processing. The PHR message types are:
PHR_Resource_Request is used in the beginning of a connection to request the necessary resources.
PHR_Refresh_Update is used to maintain the soft reservation PHB state in the reservation based approach, which is sent every refresh period.
PHR_Resource_Release (called *PHR_Release_Request* in the simulation model) is used to explicitly release resources in case of the reservation based approach.
PHR_Reserved (additional message used in the simulation model): In ns2 each packet has all defined headers in the simulator and even data messages have an RMD header. This fact presents an opportunity data packets to be easily recognized in the simulation model by the use of special combination of PHR and PDR message type, PHR_RESERVED and PDR_NO_MESSAGE. Data packets are used in severe congestion handling procedures and they should be processed by the nodes.
PHR_Release_Refresh (additional message used in the simulation model) is used to release resources after they were refreshed once, while the PHR_Release_Request releases

resources before they were refreshed. Two messages are used in the simulation model out of convenience for the implementation of the admission control mechanism.
PHR_Resource_Reinit (additional message used in the simulation model) is not used.

PDR message types:
PDR_No_Message (additional message used in the simulation model) is used in combination with PHR_Reserved as explained.
PDR_Reservation_Request corresponds to the PHR_Resource_Request.
PDR_Reservation _Report is generated by the egress node to inform the ingress about the status of the connection establishment.
PDR_Refresh_Request corresponds to the PHR_Refresh_Update.
PDR_Refresh_Report is generated by the egress to inform the ingress about the status of the refresh procedure.
PDR_Request_Info corresponds to the PHR_Release_Request/PHR_Release_Refresh.
PDR_Congestion_Report is used by the egress to notify the ingress when a flow has to be terminated due to severe congestion.

4.1.3 Header format

An RMD framework implementation in a real network should be built on top of the IP header using the IP option header field, the latter being specified in [WeKo03]. As a result the RMD header should include fields for its own identification within the IP header, followed by the RMD specific fields. In the simulation model an additional RMD header is defined that includes only the RMD specific fields. In ns2 all headers are defined in one common space. When a new header is needed it is created as an addition to the other headers. This does not change the protocol behavior, contributes to the readability of the code and keeps all RMD fields in one place.
The RMD header fields are given in Table 4-1 and some of them need explanation.
The delta_T and send_time parameters are explained when the Release scenario is presented. The parameters t and pdr_ttl are used during the partial release procedure.
The dscp parameter represents the value of the PHB class the packets belongs to. Every time a packet arrives in a node first its dscp field is checked to classify the packet in a PHB class. Normally the DSCP field will be included in the Type of flow IPv4 field [WeKo03] but in the simulation model it is included in the RMD header to keep all RMD functionality in one place.
Further differentiation within one traffic class is possible when preemption priorities are used. That provides the possibility that in severe congestion situations first low priority flows are terminated, protecting the high priority flows from the congestion. The packet priority is given by the preemption_priority field.

Table 4-1 RMD header

Field	Description
PHR message type	The type of the PHR message.
PDR message type	The type of the PDR message.
S	Signals severe congestion.
M	Signals insufficient resources.
t	Signals unsuccessful reservation.
requested	Number of requested resource units.
pdr_ttl	TTL value in the last node where reservation was successful.
delta_T	The difference in time between a release message sent and the last refresh started.
send_time	The time when a release message is sent.
dscp	The DSCP class of the packet.
preemption_priority	The preemption priority of the packet.

4.2 RMD protocol mechanisms

4.2.1 Admission control mechanisms

There are two basic approaches for admission control. The one is a simple measurement based admission control (MBAC), where PHR_Resource_Request messages are used as probes whether the requested resources are available. The second approach is reservation based admission control (RODA), where the media capacity is split in resource units and reservation states are kept in the nodes.

For the reservation based admission control the sliding window algorithm from [CsTa04] is applied. An important part of the mechanism is the admission control in case of re-routing. It is possible re-routed flows to be detected using calculation of all reserved resources in the current refresh period and comparing with the reservations of the last refresh period. If the reservation of refresh messages in the current refresh period is higher than the reservation of refresh messages in the previous refresh period new flows have arrived on the link and these can be only re-routed flows.

In fact the refresh period is split into cells and during each cell statistical information is collected. At the end of the cell and each time a message is processed the information is updated to reflect the changes in load. The details of the algorithm can be found in [CsTa04].

It is very likely that the re-routing leads to congestion in which case three different methods can be applied for admission of new reservations. Each method has an admission criterion, which overrides the standard admission criterion. If an incoming reservation request will result in severe congestion due to re-routed flows the overload criterion is applied. If this is not the case the standard admission criterion continues to be used and it checks if the flow requested resources are not more than the allowed maximum for the class.

The three overload modes are bandwidth measurement based, greedy blocking and refresh estimation based access control [CsTa04]. The measurement based method checks if the new requested resources, summed with the current link load, will be above the overload threshold. Depending on how precisely the current link load can be measured, severe congestion situations can be avoided. In the greedy blocking method if re-routed flows are detected no new reservations are accepted for the next refresh period. When the estimation based method is used, estimation of the expected re-routed flows reservation is calculated. If re-routing has happened the estimated load is taken into account and the new request is rejected only if it will cause link overload. The last method gives good link utilization without rejecting flows when they can be accepted.
The admission control algorithm also handles the release of reservation resources. This is because release of resources affects the reservation sizes and the possibility new requests to be accepted. With the use of cells in the sliding window mechanism care should be taken that the link resources are released in the correct cell. Otherwise reservation resources of other than the released flow can be freed. The number of the correct cell can be calculated, in which process the RMD header fields delta_t and send_time are used.

4.2.2 Severe congestion detection mechanism

A severe congestion handling begins with a phase of overload detection. Overload can be detected by the interior nodes. In RMD these are reduced sate for the reservation based mode and stateless for the measurement based mode. The latter means that interior nodes cannot solve the severe congestion but edge nodes have to be notified. For the notification re-marking of passing PHR messages or of data packets can be used. In the latter case several possibilities exist. Of all possibilities the rate proportionate marking is discussed.
An interior node collects information on the link load of data packets during a predefined measurement period [CsTa05]. At the end of each period the bandwidth on the link is calculated and it is compared with a fixed threshold level. The link bandwidth can be calculated as only during the expired period or also values from previous periods can be used for better estimation. If the calculated bandwidth value is above a fixed severe detection threshold a process of marking data packets is started.
This is the base operation on top of which two methods for data packets marking can be applied [CsTa05]. In the first method referred as Rate Proportionate Marking (RPM) the number of data packets to mark is calculated as the number of bytes above the congestion detection threshold. The calculated data packets are marked with “encoded DSCP”. This calculation happens each measurement period. The marking of data packets continues until no overload is detected on the link. This method only informs the edge node for occurred severe congestion but about its level.
The second method, referred to as Dampened Rate Proportionate Marking (DRPM), is an improved version of RPM that uses a sliding window mechanism for the marking [CsTa05]. The initial number of bytes to mark is calculated as the number of bytes above the severe congestion restoration threshold. The restoration threshold represents the load value under which the severe congestion is considered as solved. At the same time the node remembers the bytes marked during previous measurement periods, the number of the periods depending on the window size. As result the final number of bytes to be

marked is the initial calculated number of bytes minus the number already marked bytes in the window. The calculated number of data packets are marked with “encoded DSCP” and announce the level of severe congestion. All other data packets carry the “affected DSCP” and only inform that severe congestion has happened.
The DRPM method aims at informing also the level of severe congestion to the edge nodes and the sliding window is used to prevent possible undershoot due to over-marking of packets.

4.2.3 Severe congestion solving mechanism

The second phase of the severe congestion handling is the congestion solving. The reason for severe congestion is often the overload on a link due to re-routing. Therefore to solve the severe congestion part of the existing flows have to be stopped and no new flows to be accepted until the utilization of the link drops back to its capacity. The criterion used by the egress to terminate flows depends on a local policy and therefore it can differ.
When the dampened proportional marking, section 4.2.2, is used the egress node determines the level of severe congestion by counting the number of data packets with “encoded DSCP”. The same measurement period as for the severe congestion detection is applied for the counting. The total number of bytes from the counted “encoded DSCP” data packets should be enough to solve the congestion.
The egress node also counts the bytes that are from “affected DSCP” data packets. Each flow with “encoded DSCP” or “affected DSCP” data packets is recorded in the egress node. At the end of the measurement period the egress node chooses the flows to terminate according to established local policy. In order the severe congestion to be solved the sum of the reservations of the terminated flows has to be equal or bigger than the total number of marked bytes.
As it was mentioned preemption priorities are implemented in the model and each generated flow is assigned one. The severe congestion solving mechanism should always terminate flows in order of their preemption priority stating with the lowest one. Therefore flows with the lowest priority are stopped first. If the total amount of bytes for this priority class is smaller that the total marked bytes, flows from the next priority level has also to be terminated. Within each priority class first flows with “encoded DSCP” data packets are chosen. If the number of bytes from “encoded DSCP” marked flows cannot solve the congestion then flows with “affected DSCP” have to be terminated. The egress node actually sends a congestion report message to the ingress, which can stop the flow. Such end handling of the flows is called partial termination because part of the flows is terminated.

4.3 RMD simulation model

The simulation model is written in C++ and OTcl [ns manual, www]. Four software major programming classes are defined for the RMD edge nodes, i.e., admission control, congestion handling, dscp mapping and monitoring. The new header used in the ns2 simulation model has the format as described in section 4.1.3 and the used messages are described in section 4.1.2. Since the model is characterized with intricacy and multiple implemented solutions, each one of the major classes is discussed separately. The correspondence between the used class names in the text and the real class names with the files where they can be found is given in Appendix A.3.

4.3.1 Nodes

As mentioned in section 4.1.1 interior nodes apply admission control for every reservation request and monitor the link for severe congestion occurrence, i.e., the PHR protocol functionality. These functionalities are implemented and associated with the outgoing link of a node (Figure 4.3).

The edge nodes on the other side in addition to the support of the functionality of the PHR protocol have also to support the PDR protocol functionality, such as the support of the severe congestion handling. In particular, there is a dedicated class RMDEdge that implements their functionality with two sub-classes, i.e., the MBACEdge and the RODAEdge. The former implements the measurement based behavior of the edge nodes and the latter – the reservation based. The hierarchy is presented in Figure 4.1, along with the severe congestion handler and the used timers. The measurement based behavior and the re-allocation behavior are not of interest for the RMD-QOSM. In the re-allocation behavior all flows with marked packets are stopped and subsequently for each flow again a request is made. In this case the congestion handling software class does not calculate how many of the marked flows should be stopped.

Concentrating on RODAEdge, the available functionality is for connection establishment, refresh and release and severe congestion handling. The means to distinguish between the ingress and egress functionality is accomplished by using functional states that are explained in section 4.3.3. An egress edge agent can connect to a severe congestion handler, which manages all activities for severe congestion solving by terminating flows.

Each flow, scheduled in the simulator, has its own pair RODAEdge agents – one agent acting as ingress and one agent as egress. Therefore one physical node can have many ingress/egress agents. To each RODAEdge agent a transport agents and a traffic generator/sink are attached. The resulting protocol stack is presented in Figure 4.2.

After this initialization phase the exchange of signaling messages is started. The admission control takes place on each node of the signaling path. As a result if a connection is established the traffic source is started and the data exchange phase begins. If the reservation is not successful the edge agents are deleted.

The manner in which the edge agents are connected with the flows, one to one, creates inconvenience for handling the severe congestion. No single egress agent can know how many flows there are in its node and whether they are severe congested. Therefore the severe congestion handler is added to monitor all flows in one node. This addition is further described in section 4.3.5.

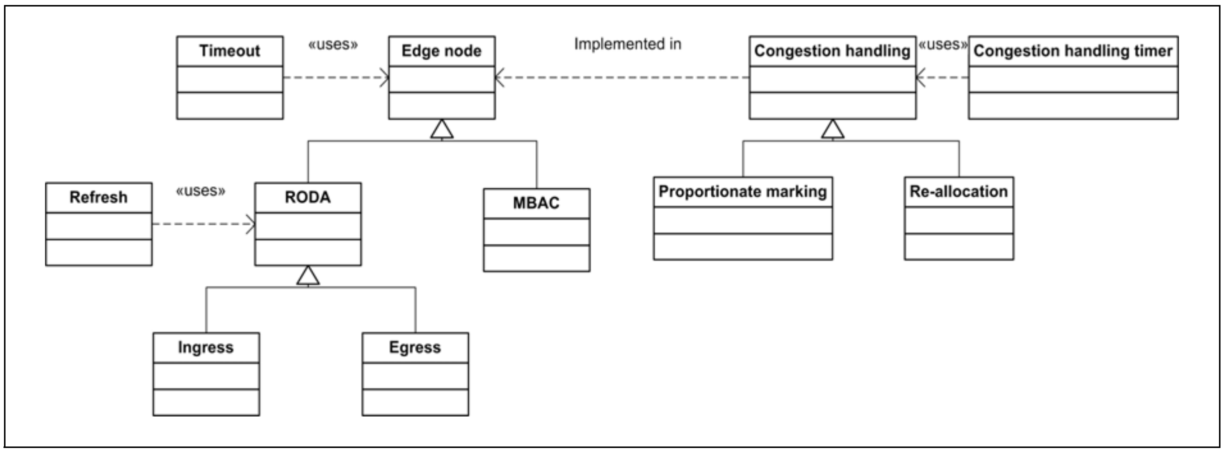


Figure 4.1 Hierarchical structure of a node.

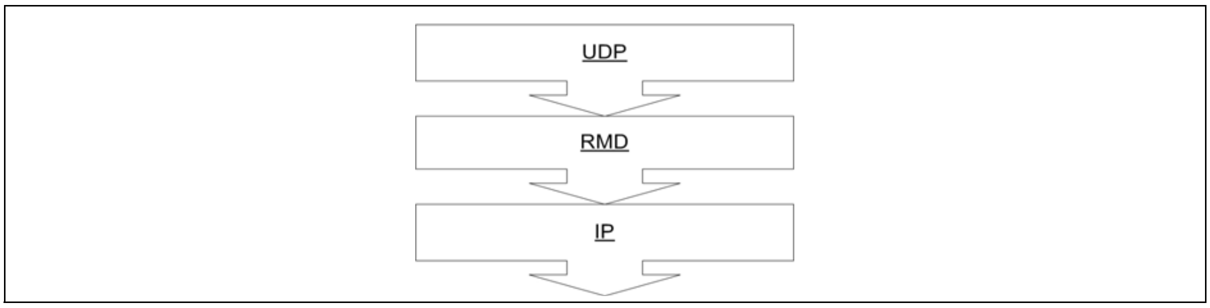


Figure 4.2 Simulation protocol stack

4.3.2 Links

The two way signaling requires duplex links. Such duplex links can be defined in the ns2, which actually consist of two simplex links in forward and reverse direction. On top of the simplex link a software class "Link Monitor" is added. The monitor class is an umbrella class that calls methods of two other classes (Figure 4.3) – an "Admission control" class and a "Severe congestion detection" class. These two classes implement the RMD interior nodes behavior and process each packet that enters or leaves the link. Additionally a "Measurer" (i.e., meter) software class, with supportive function, is attached to the link to gather information on the entering link traffic per PHB.

The queuing models used on the link are of type Class Based Queuing (CBQ), for which traffic classes with different priorities are defined [ns manual, www, section 7.3]. In the simulation model signaling messages are assigned to traffic class with higher priority and data packets to traffic model with lower priority. In this way the signalization will always be passed by the link.

4.3.3 States

Two types of functional states are defined in the class of the edge agents – for the ingress and for the egress agent. A functional state, a pure implementation feature, is used to identify different stages of the signaling and to detect incorrect message type arrival i.e. refresh in disconnecting state. The states and possible state transitions are given in Appendix A.1.

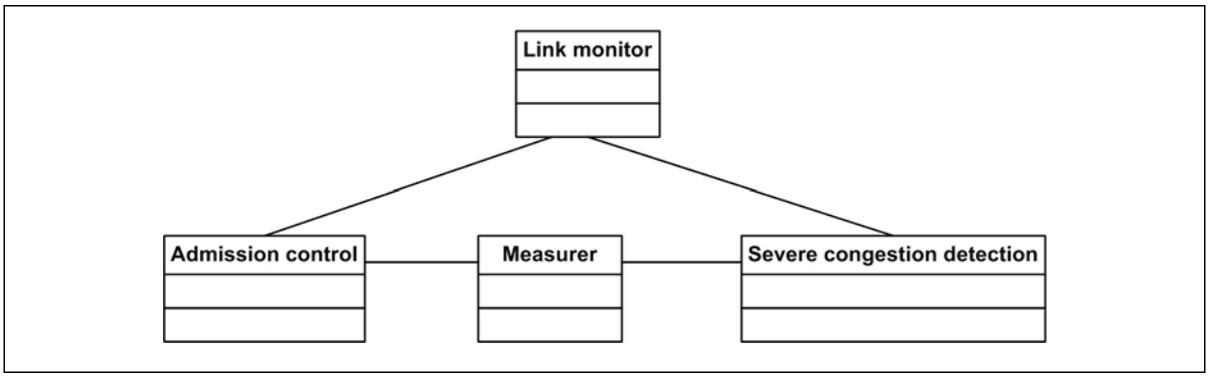


Figure 4.3 Link hierarchy

The ingress node states are:

- initial "Wait for QoS request" state in which the ingress node awaits request for connection establishment. The state is left when an instruction to start a session is given and a PHR Request message is sent. The state changes to
- "Wait for Reservation report" state when the ingress agent waits until PDR Reservation Report message for the initial PHR Request is received. After timeout or if the PDR Reservation Report is negative the state is changed to "I_Disconnecting". If the PDR Reservation Report is positive the session can be started and the ingress changes state to
- "I_Admitted" state when the session is accepted and the data flow is started. Each predefined period of time the ingress agent initiates refresh message and moves temporally to
- "Wait for refresh report" state. This is the state of the ingress agent during the refresh procedure.
- Final "I_Disconnecting" state that is changed to at the end of the session or when the session was prematurely terminated from the network.
- For the re-allocation mechanism three additional states are used: "I severe congestion re-initialization release", "I severe congestion re-initialization reserve" and "I severe congestion re-initialization back-off".

For the egress node the states begin with:

- "Wait for reservation request" is the state the egress agent is "born" with. The state is left only if a PHR Request message is received. Any other messages would be arriving in an illegal state. The arrival of PHR Request can cause transition to "E_Disconnecting" (unsuccessful reservation procedure) or to:
- "E_Admitted" state follows a successful reservation on the data path. This is the normal working stat e of the egress agent and is left only when a release message is received. The state after it is
- "E_Disconnecting" which is the final state.
- Additional state for a re-allocation procedure – "E severe congestion reinitialization".

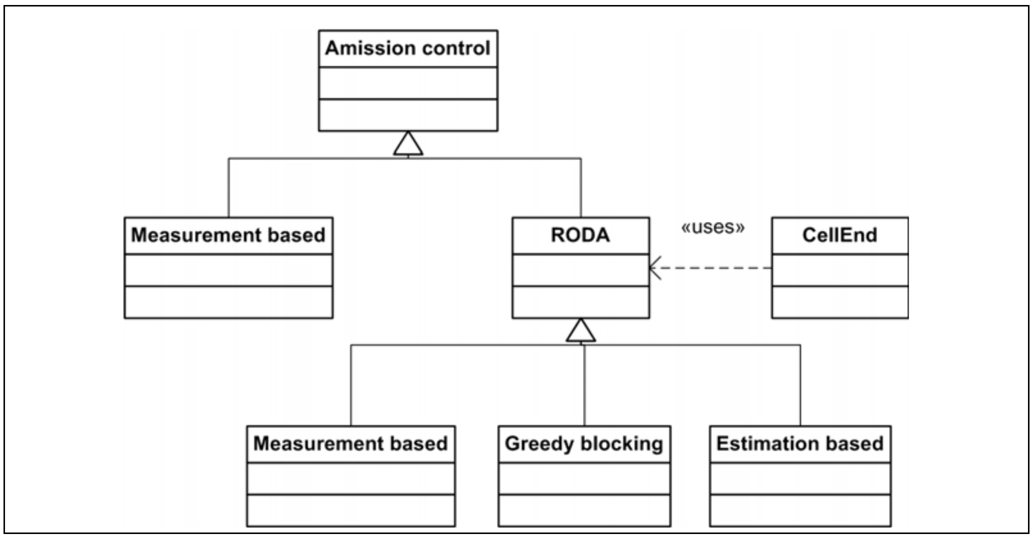


Figure 4.4 Admission control classes hierarchy

4.3.4 Admission control

The admission control methods implemented in the code were described in section 4.2.1. The hierarchy of methods is presented in Figure 4.4 where each method is implemented in software class.

The measurement based admission control (MBAC) is implemented in the software class Measurement based. Software class RODA, with its three sub-classes, implements a reservation based admission control that uses the sliding window mechanism from section 4.2.1 and its three override modes. An additional software timer class "Cell end" is introduced that is used to delimit the end of cell duration in the refresh period.

The correspondence between the names of the software classes and the C++ classes in the source code is given in Appendix A.3.

4.3.5 Severe congestion detection and solving

The congestion detection class hierarchy is presented on Figure 4.5. A part of the code concerning token bucket measurements (TBMeasured) is still in research phase and is not discussed. The software class "BWMeasured“ implements all common operations when the link load is measured to discover severe congestion. A software timer class "BWUpdate" is attached to it to delimit the end of the measurement period. Two sub-classes stem from the "BWMeasured" class. The one, "ThresholdBasedMarking" is not of interest in the current research, and the other, "RatePropotionalMarking" represents the RPM method described in section 4.2.2. Finally the upgrade method DRPM is implemented in the "DampenedRateProportionalMarking".

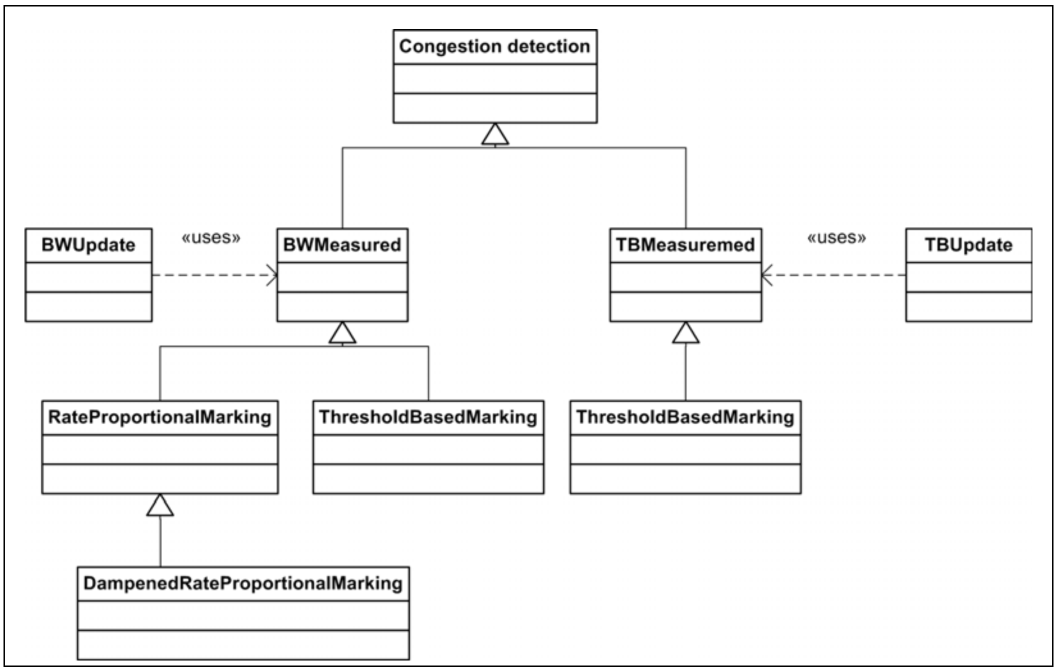


Figure 4.5 Congestion detection classes hierarchy

The implemented severe congestion solving mechanism is as described in section 4.2.3. A local policy at the egress how flows are chosen or termination is also specified. In the simulation model the flows that have to be stopped are chosen based on their size. The flow with size closest to the severe congestion level but lower than it is chosen as first. The severe congestion level is decreased with the size of the flow. Subsequently the same principle is applied for the remaining flows. As result of this policy minimum number of flows is stopped.
The severe congestion solving mechanism is implemented in the “Congestion handling” software class, see section 4.3.1.

4.3.6 Monitor support

A special class is developed to collect statistical information on the link load from signaling and data messages. This information on bandwidth measurements can be used to evaluate the operation of the RMD protocol in different situations. The ns2 implementation of the monitor is the C++ class EnhancedFlowMonitor [enhanced-flow-mon.h/cc]. An instance of the class is attached to the link of interest. The priority field of the IP header is used to collect information on signaling messages, high priority, and data packets, low priority. The data load statistics are classified in three groups depending on the preemption priority of the packet, low, medium or high.

4.3.7 Scenarios

In all scenarios interior nodes are represented by using Link monitor object, see Figure 4.3. As it was shown the interior nodes behavior in the model is spread over the two objects for admission control and congestion detection, forming a monitor. The Link monitors are responsible for performing the resource reservation and marking packets if severe congestion occurs.

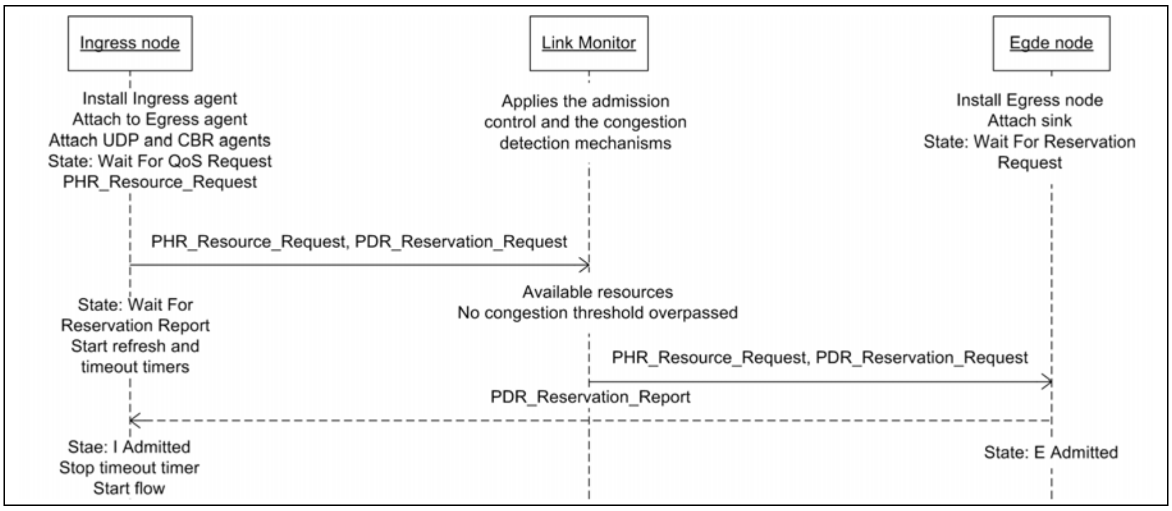

Ingress node
Link Monitor
Egde node
Install Ingress agent
Attach to Egress agent
Attach UDP and CBR agents
State: Wait For QoS Request
PHR_Resource_Request
Applies the admission control and the congestion detection mechanisms
Install Egress node
Attach sink
State: Wait For Reservation Request
PHR_Resource_Request, PDR_Reservation_Request
State: Wait For Reservation Report
Start refresh and timeout timers
Available resources
No congestion threshold overpassed
PHR_Resource_Request, PDR_Reservation_Request
PDR_Reservation_Report
Stae: I Admitted
Stop timeout timer
Start flow
State: E Admitted

Figure 4.6 Successful reservation scenario

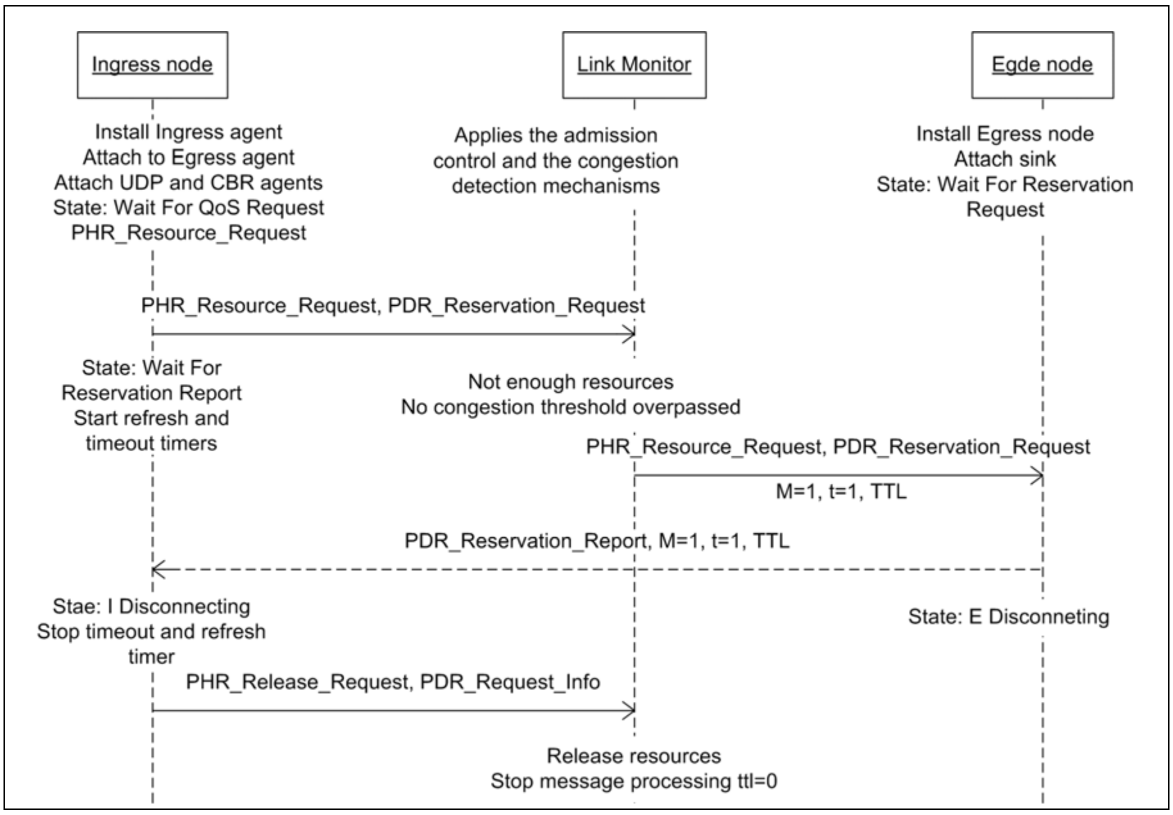

Ingress node
Link Monitor
Egde node
Install Ingress agent
Attach to Egress agent
Attach UDP and CBR agents
State: Wait For QoS Request
PHR_Resource_Request
Applies the admission control and the congestion detection mechanisms
Install Egress node
Attach sink
State: Wait For Reservation Request
PHR_Resource_Request, PDR_Reservation_Request
State: Wait For Reservation Report
Start refresh and timeout timers
Not enough resources
No congestion threshold overpassed
PHR_Resource_Request, PDR_Reservation_Request
M=1, t=1, TTL
PDR_Reservation_Report, M=1, t=1, TTL
Stae: I Disconnecting
Stop timeout and refresh timer
State: E Disconneting
PHR_Release_Request, PDR_Request_Info
Release resources
Stop message processing ttl=0

Figure 4.7 Unsuccessful reservation scenario with following resource release

Successful reservation procedure

Under successful reservation is understood that all nodes on the data path have available resources to support the session and have installed reservation states for it. The resources are requested using the PHR Request message and the egress agent when it evaluates the reservation as successful returns positive PDR Reservation Report to the ingress node. The procedure is and the message exchange can be followed in Figure 4.6.

Unsuccessful reservation procedure with partial release

Specific for this scenario is the partial release mechanism, as described in [BaWe06] and presented in Figure 4.7. Whenever a node can not support a request, it M marks the message and includes the current IP *ttl* value in the RMD header. The ingress node knows the default *ttl* value to reach the egress. When the ingress receives M marked reservation response it has to initiate partial release procedure. The *ttl* value of the release message is the difference between the stored default *ttl* and the ttl value received from the response message. Using this mechanism it is ensured that the release will reach only nodes where reservation was successful and will stop before the node where it failed.
Another characteristic of the release procedure is the *delta_T* parameter – the difference in time between the release message generation and the beginning of the current refresh period. This difference is also calculated by each interior node to find the correct cell of the refresh period where the resources have to be released.

Successful and unsuccessful refresh procedure

The procedure for successful refresh is presented in Figure 4.8. A refresh message is sent from the ingress on the data path to egress agent. The message is processed in all intermediate nodes and the reservation state is refreshed. The egress agent returns to the ingress response for successful refresh which is not processed inside the domain.
When a refresh message request cannot be satisfied in an interior node, due to severe congestion, the message should be M marked but nevertheless the resources are always refreshed. This is done because two possibilities for unsuccessful refresh exit. In the one when the flow is stopped an explicit release procedure is initiated. If the reservations were not refreshed the release message will affect incorrectly reservations other flows. It can be chosen that after M marked refresh no release procedure is started in which case the resources will be released after the refresh period has expired. Figure 4.9 shows the first variant when explicit release is initiated.

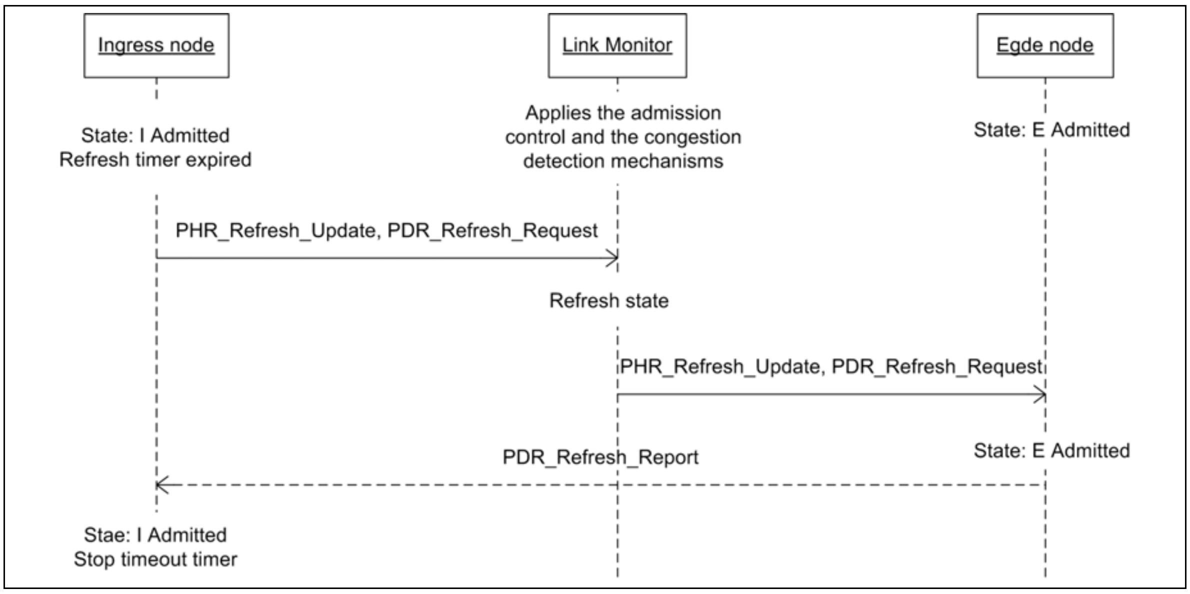


Figure 4.8 Successful refresh scenario

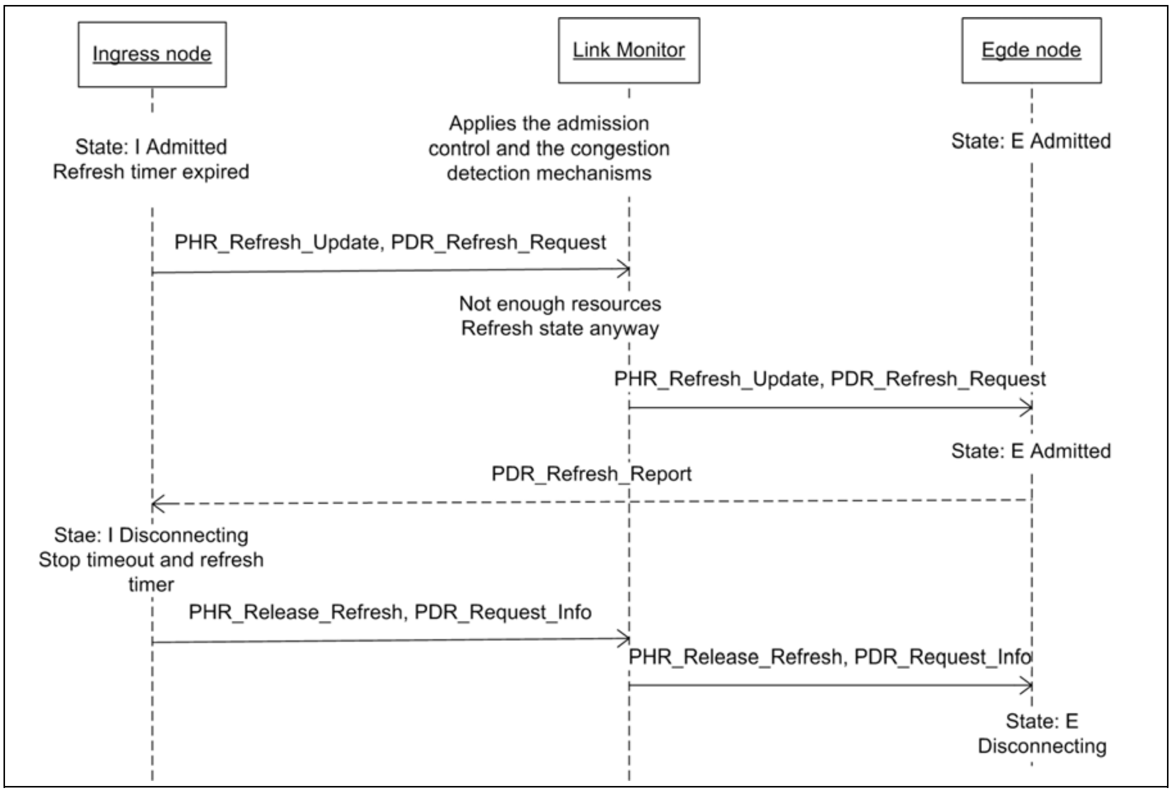

Ingress node
Link Monitor
Egde node
State: I Admitted
Refresh timer expired
Applies the admission control and the congestion detection mechanisms
State: E Admitted
PHR_Refresh_Update, PDR_Refresh_Request
Not enough resources
Refresh state anyway
PHR_Refresh_Update, PDR_Refresh_Request
State: E Admitted
PDR_Refresh_Report
Stae: I Disconnecting
Stop timeout and refresh timer
PHR_Release_Refresh, PDR_Request_Info
PHR_Release_Refresh, PDR_Request_Info
State: E Disconnecting

Figure 4.9 Unsuccessful refresh scenario

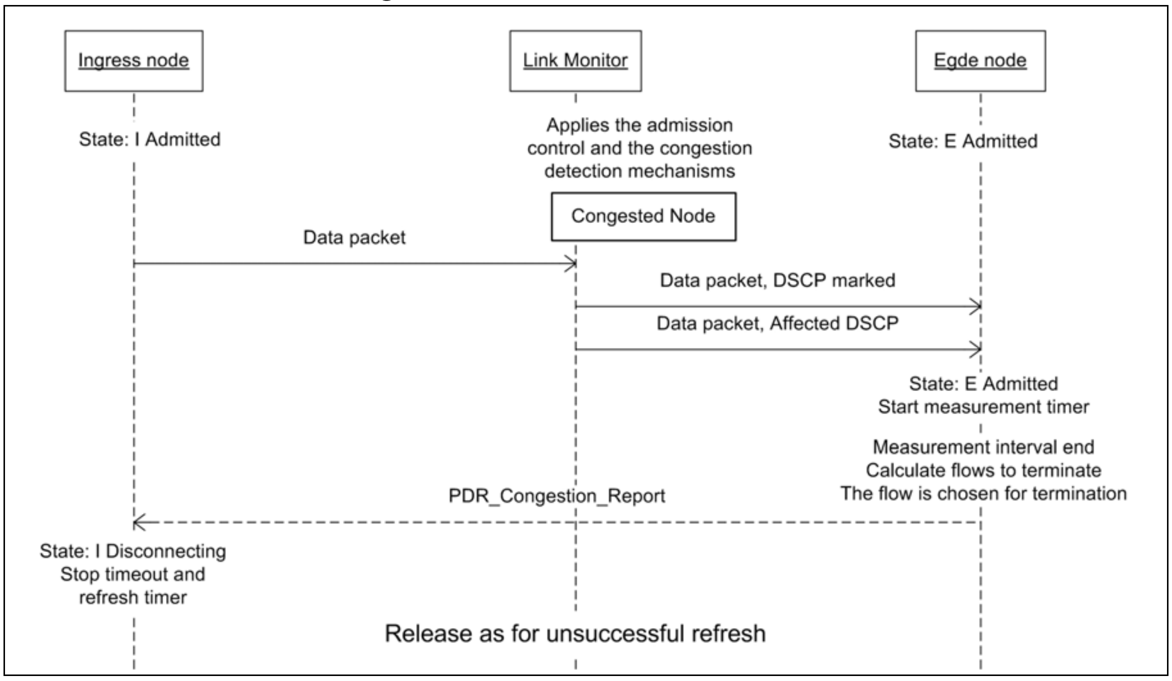


Figure 4.10 Severe congestion scenario

Severe congestion scenario

On Figure 4.10 the severe congestion handling for one flow is shown. When an interior node receives a data packet the node checks if congestion has occurred and whether the packet have to be either "encoded DSCP" or "affected DSCP" marked. The rule what marking should be used is as explained in section 4.2.2. If the data packet is marked with "encoded DSCP" its size is remembered in the sliding window.

4.4 Protocol – model conformance

The simulation model presented in section 4.3 should represent correctly the behavior of the PHR and PDR protocols in the RMD framework briefly described in Section 4.1. The best way to verify the relation between the specification and the implementation is to formally verify and validate the model, which falls out of the scope of this assignment. Nevertheless since the RMD simulation model is used as a base for the development of a new model care should be taken that RMD simulation model conforms to the specification.
The conformance is tested by comparing the message exchange and state transitions defined in the RMD framework and the ones implemented in the RMD simulation model. This manner to test conformance is straightforward, it is not time consuming and provides trustworthy results. The mechanisms for admission control and severe congestion handling are verified only intuitively. These mechanisms are relatively complicated and intuitive comparison is not sufficient, therefore formal verification is recommended.

4.4.1 Used methods to test conformance

In the process of testing the conformance protocol specification – simulation model state machines are used to represent the possible conditions of the nodes/agents and specifying the factors causing transitions between the states. Using state machines allows the state diagrams used for the presentation of the RMD framework here, to be directly compared to the state diagrams for the simulation model, presented in section 3.4. Furthermore state machines provide easier transition to later formal verification.
A back box approach is used where the end user is represented as a system of three components – an application module, a timer module and a network node module. Each of the modules is considered to be black box and only the inputs and outputs of the box are visible. The network node module includes the RMD edge node functionality and it will be tested. For this module the inputs can come from the application module – data traffic event, the timer module – timeout, or another network node module – signaling messages.
Based on the RMD specification two state machines are developed one for the ingress node and one for the egress node. Both state machines are presented in tables with components begin state, input, output and end state, see Appendix A.2. All combinations begin state – input are presented no matter if the combination is treated as correct or erroneous in the specification. All erroneous combinations should produce an ERROR output.
In the simulation implementation invalid combinations are detected with an explicit check. For each incoming message the agent checks if the state is correct. If the state does not have the coming input as possible it rejects the message, an ERROR output is generated and the simulation aborts.
The state machines are used to create test cases, which are organized in test suits. A test case usually represents a desirable behavior, or what output is expected upon given input. Therefore each entry in the tables in Appendix A.2 can be a test case, and the sum of all entries form a test suite. The test suites for ingress and egress agents are presented in Appendix A.2 in the form of a graphical representation. Only the test cases ending in state different than ERROR are presented to keep the scheme readable.
The conformance RMD framework – RMD simulation model is tested as each test case is checked against the state diagrams of the RMD simulation model in section 4.3.3. If all test

cases are supported by the implementation then the simulation model reflects the specification correctly.
Before the results of the conformance check are presented explanation of the used terminology is given.
State space is the multitude of functional states in which a node can be, section 4.2.3.
Input space is the multitude of input stimuli that can arrive at the entrance of the system and can cause response reaction to take place. Most of the time the input stimuli are messages.
Output space is the multitude of reactions generated as result of the occurrence of one of the input stimuli. The reactions are in the form of messages.
In sections 4.4.2 and 4.4.3 before the conformance test to be explained the above defined spaces are populated with valid members for the object of test.

4.4.2 Ingress node

States space: Wait for QoS request, Wait for response, Admitted, Final state (-), Abort
Inputs space: QoS request, PDR_Reservation_Report, PDR_Reservation_Report M=1, PDR_Reservation_Report S=1, PDR_Refresh_Report, PDR_Refresh_Report S=1, PDR_Congestion_Report, Refresh timeout, Stop traffic, Data packet.
Outputs space: PHR_Resource_Request, PHR_Refresh_Update, PHR_Resource_Release, Data packet, ERROR

After the test cases from Appendix A.2 were compared against the state machine for the simulation model the following observations were made. First in the model two additional valid states are defined: Disconnecting and Wait for refresh response. Disconnecting state corresponds to the final state used here. The “Wait for refresh report” state can be seen as a sub-state of the Wait for report state. So far the simulation model only extends the specification and therefore it still represents it correctly.
Two test cases can not be followed in the simulation model since they were not implemented – case 5 and 7 or congestion detection with protocol messages. Currently the model does not support these two mechanisms. This difference is not crucial for the work of the protocol and does not affect its proper behavior.
About case 6 the implementation can distinguish two cases – marked or unmarked refresh. Upon received marked refresh a release message can be sent so the flow is stopped. This possibility is again extension to the specification and does not violate the correct operation of the protocol. Further all test cases coincide with the simulation model.

4.4.3 Egress node

States space: Wait for request, Admitted, Final state(-), Abort
Inputs space: PHR_Resource_Request, PHR_Resource_Request M=1, PHR_Resource_Request S=1, PHR_Refresh_Update, PHR_Refresh_Update S=1, PHR_Resource_Release, Data packet
Outputs space: PDR_Reservation_Report, PDR_Reservation_Report M=1, PDR_Reservation_Report S=1, PDR_Refresh_Report, PHR_Refresh_Update S=1, PDR_Congestion_Report, ERROR, Data packet

The test suite for the egress node is presented in Appendix A.2. As in the case of the ingress node the Disconnecting state is defined to reflect the final state. Also as for the ingress node the PHR message marking for severe congestion is not presented in the model therefore test cases 4 and 6 cannot be followed in the simulation model. Further all state transitions of the RMD framework are represented in the simulation model.

Based on the test suites for the ingress and the egress node, applied correspondingly to the ingress and egress agents from the implementation model, it can be concluded that the model misses some functionality. With respect to the latter, it can be said that the missing functionality is not a crucial part of the protocol but an alternating choice and does not cause malfunctioning of the implementation. Further the model extends the specification which is also not an erroneous situation, as long as the additions do not conflict with the behavior defined in the specification.

5 Comparison of base RMD simulation model and RMD QOSM

Main principle in protocol engineering is reusability of components and building on top of them instead of ‘re-inventing the wheel’. An already implemented base simulation RMD model was presented in Chapter 4. Using the model as a start base, saves a lot of efforts and allows reusing already tested mechanisms. In order the base model to be re-used, it should be clear what is implemented, what the differences with the desired behavior are and therefore what should be added. Comparing each functionality in the base model and in the desired new model, it can be concluded what has to be done additionally.
The scope of this chapter is a comparative analysis between the two models, starting with section 5.1 where it is reasoned why the RMD framework and the RMD-QOSM model can be compared. In section 5.2 the analysis is done step by step, beginning with the QoS NSLP protocol and moving to the RMD-QOSM used on top of it. Recommendations on necessary functionality for the new RMD-QOSM simulation model are given in section 5.3. At the end of this chapter it should be clear what modifications and additions the development of the new RMD-QOSM simulation model requires.

5.1 Grounds for comparison

Why would a simulation model for one protocol be used to represent another protocol? Because both protocol share similarities in their behavior. Understandably modifications and adaptations might be necessary. A simulation model representing the RMD protocol behavior already exists, see Chapter 2, and it was introduced in Chapter 4. This research studies the behavior of the RMD-QOSM by creating simulation model for it. RMD-QOSM actually uses the same RMD concepts but applied on top of a general QoS NSLP signaling protocol. The RMD-QOSM was introduced in section 3.3 and its operation in sections 3.4 and 3.5. It can be concluded that the already implemented RMD simulation model and the desired RMD-QOSM simulation model will share functionalities.
RMD, in its original for, supports quality of service provisioning with DiffServ concepts. QoS NSLP, on the other side, defines a general frame for quality of service delivery, on top of which different models can be used. One of the models, i.e., RMD-QOSM, adopts the RMD idea with slight changes to allow interoperability with the signaling QoS NSLP layer. This implies that the majority of mechanisms, used in both protocols, are similar but still differences exist. Consequently the base RMD simulation model can be used for the evaluation of the RMD-QOSM behavior but first some changes might be necessary. For example, the QoS NSLP functionality part of RMD-QOSM, as a new part, would have to be additionally implemented in the new simulation model.
A good starting point for the development of the new simulation model is a comparative analysis to be done. The already implemented functionalities, RMD simulation model, can be compared with the required functionalities, RMD-QOSM specification. A comparison can be made because both, RMD and RMD-QOSM, operate in the same environment, DiffServ domain and both have the same purpose, to provide reservation management for quality of service delivery

5.2 Comparative analysis

First the new aspects of the simulation model, connected with the QoS NSLP protocol, are described. Subsequently, a comparison of the base simulation model with the RMD-QOSM model is provided. A point-by-point manner of comparison is used of the major protocol elements – nodes behavior, header formats, and message exchange.

It is important to note that the NSIS framework includes the NTLP and NSLP protocol layers. This assignment concentrates on the performance evaluation of the NSLP, and in particular of the RMD-QOSM model. Therefore, in order to reduce the amount of necessary simulation implementation work, it is decided that the simulation model will not include the NTLP functionality. It is assumed that the NTLP layer operates in an ideal manner and provides an ideal bridge to the lower layers of the IP protocol stack.

In ns2 UDP is implemented as an existing agent with well defined interface functions. The base simulation mode and the newly developed one use UDP transport agent and it is trusted that the module is correctly implemented. The UDP agent also processes the IP header that is attached to each message.

In this section the term *base simulation model* is used to represent the available RMD simulation model and the term *specification* – the RMD-QOSM and QoS NSLP protocol specifications. It is assumed that the base simulation model corresponds to the protocol specification.

5.2.1 New functionality for QoS NSLP

QoS NSLP, as novice functionality, is not implemented in the base simulation model and therefore all its components have to be added. Beginning with the message types, the four QoS NSLP message types – Reserve, Query, Response and Notify will be introduced. The use of these messages is general and does not depend on the quality of service model to be used. Note that even if some of the QoS NSLP functionality is not used by the RMD-QOSM simulation model, it is still included. This will allow in the future an easier adaptation of the simulation model to serve other quality of service models.

The QSPEC object, see section 3.3.2, will include information on requested resources, DSCP value for the particular request and specifics for each particular quality of service scheme used, RMD-QOSM model in this case. When the base simulation model is applied all necessary information is included in the PHR and PDR message fields. These messages, in the new simulation model will be implemented as objects of the newly defined QoS NSLP messages. Further the fields of the PDR/PHR messages and the PDR/PHR QSPEC containers are similar.

The new messages use a new common header, the QoS NSLP header. In order to support a modular implementation of the simulation model, two different headers will be defined, the QoS NSLP header and a header that can represent the QSPEC object. In the researched simulation model, the latter header carries all RMD specific information. All other standard QoS NSLP header information is specified and carried in the former header and remains unchanged within the RMD domain. Such separation of the headers allows new quality of service models to be easily implemented in the simulation model, by only specifying a new header representing their QSPEC object.

In the base simulation model an RMD reservation state is kept. In the simulation model implementation of the QoS NSLP, an additional operational state will be added. Furthermore, only defining the new messages and QSPEC header is not enough – they have to be processed correctly in the nodes.
The RMD-QOSM simulation model uses the concept of session ID. It plays a very important role in identifying a session, when for example, tunneling, flow aggregation or bi-directional reservations are applied. All three possibilities are missing in the base simulation model. How the session ID can be implemented in the new simulation model is discussed section 5.2.2.
Another important feature is the policy control functionality of the QoS NSLP. To apply policy control additional software classes in the simulation model are required. The policy control module is still under research and therefore, it can be left for a later phase of the simulation model implementation process.
As it was shown above, many features required by the QoS NSLP protocol will have to be implemented. That can be an interesting task since the protocol is developed to support a lot of QoS applications that can be applied in different network scenarios. As result implementing the complete functionality of QoS NSLP simulation model can be an independent project. For the goals of the current research all functionality necessary to support the correct operation of the RMD-QOSM model will have to be added.

5.2.2 Required modifications for the RMD-QOSM implementation

The information carried by the PHR and PDR messages, in the base RMD simulation model, and the one carried by the PHR and PDR QSPEC containers, in the RMD-QOSM specification [BAWE06], is the same. The comparison of their parameters shows that the Admitted hops, Max Admitted hops and Hop_U fields from the RMD-QOSM specification are missing, see section 3.4.1. The fields can be represented by using the PHR-TTL and PDR-TTL and the T flag from the base simulation model. These are two different ways to perform the same task. It would be the simulation model implementer's choice either to use the available simulation model implementation and add/modify the implementation to support the needs of the RMD-QOSM specification or to displace the base simulation model with a completely new one implemented to support the main features of the RMD-QOSM specification.
Furthermore, the RMD-QOSM specification uses an additional preemption priority. The preemption priority feature is currently included in the new version of the RMD-QOSM specification and it provides the possibility flows to be differentiated within one DSCP class.

The node functionality can be divided in two groups – for edge nodes and for interior nodes. As mentioned in previous sections, the interior nodes can be stateless, when used for the measurement based admission control, or they can be reduced state, when used for the reservation based approach. Both admission control mechanisms, see section 4.2.1, are implemented in the existing code and it is a researcher choice which of the mechanism to be used and tested.

Interior nodes are also responsible to detect severe congestion situations and to notify the edge nodes if such situations have occurred. Severe congestion detection and handling using PHR refresh messages [WeJa03] is not implemented in the simulation model and has to be added if required. Notification by marking data packets is implemented with several solutions [CsTa05] of which the Dampened Rate Proportional Marking class (section 4.3.5) is chosen, after a comparison with the mechanism described in the specification [BAWE06]. Using the simulation model implementation, the described algorithms in the RMD-QOSM specification could be refined. This is done in the new specification.
A difference between the base simulation model and the RMD-QOSM specification is that in the simulation model the interior node functionality is performed by a Link monitor. Note that in ns2 the links can be seen as queuing and scheduling modules. A Link Monitor can integrate the functionality that processes the received messages and data packets and provide admission control and severe congestion detection decisions.
Edge nodes perform a more complex functionality than the interior nodes. Edge nodes are considered to be stateful and they should be able to support the complete QoS NSLP functionality and the RMD-QOSM functionality. In particular, they maintain reservation states for each flow and manage the signaling within the domain. Edge nodes are implemented in the simulation model as agents – ingress and egress.
Interior nodes can inform the egress agents about severe congestion situation by marking of data packets. Calculating the rate of the marked bytes, the egress agent can decide which flows should be terminated, such that the severe congestion situation can be solved. Section 4.6.1.6.2.2 of [BAWE06] provides recommendations on how the severe congestion should be handled by the egress node. In that section also the preemption priority is used in the termination of flows. In both, base simulation model and RMD-QOSM specification only “encoded DSCP” marked or/and “affected DSCP” marked flows are terminated, by starting with the lowest priority. The base simulation model implementation provides a particular manner on how flows should be chosen. Other choices are also possible. The RMD-QOSM specification is not strict on that.

The biggest difference in the base simulation model and the RMD-QOSM specification is the use of session ID. To implement the new behavior associated with the session handling two approaches are possible. In the first case a new single edge agent can be created to manage all flows that belong to him. That closely represents a real edge node since one agent is attached to one node in the simulation model. This approach however requires re-writing a major part of the base simulation model. The second approach builds on top of the existing implementation. In the base simulation model for each flow two edge agents are used, each agent maintaining only the session ID it is responsible for. Therefore, an additional functionality is needed that will maintain information on the session IDs of all edge agents attached to one physical node. This new functionality will be responsible for the management of the sessions and for performing all necessary check ups on identifying the session ID of a particular edge agent. Due to time constraints, the second approach is followed. If tunneling, aggregation of reservations and bi-directional reservations are all implemented in the simulation model, this new functionality should be capable to differentiate between the use of the session ID in each of these scenarios.

By comparing the message sequence diagrams used in the RMD-QOSM specification and the one used in the base simulation model, it can be deduced that the RMD-QOSM specification uses messages that have to be propagated on an end to end basis. These messages are not processed in the interior nodes and have to be added to the simulation model along with their processing by the edge nodes. No further differences in the communication between nodes are observed.

5.2.3 Necessary extensions for the base simulation model

Before the results of the comparative analysis are presented as conclusions, some recommendations towards the potential designer of the new RMD-QOSM simulation model are made.

It is advised that all missing protocol functionality is considered at the design phase of the simulation model. As end result, a complete simulation model design will be available to support later implementation. If a modular principle is used, the functionalities can be implemented in different stages depending on the goals of the simulation model.

A good implementation strategy would be the QoS NSLP functionality to be split from the particular quality of service one. The modular approach, if used, can result in having relatively independent simulation model implementations of the common signaling protocol and of the different quality of service models. It would also prepare a clear structure of the implementation for later extensions of the simulation model with the underlying GIST transport layer. To achieve a split simulation model implementation, two types of protocol agents have to be used – one of the QoS NSLP and one for the QoS model used. Accordingly two headers – RMD-QOSM and the QoS NSLP common header should be attached to the signaling messages. Each agent will be responsible for processing the corresponding header.

A modular principle would also be used for the implementation of the node behavior. As it was shown, a QoS NSLP node can perform a lot of different procedures with well defined relations. If a module is used to implement each of these procedures the design and implementation of the simulation model will be easy to understand by every new researcher working on it. Furthermore adding new functionality to the simulation model will be simplified and made less error prone.

Based on the analysis it can be concluded that a good starting base for simulation model implementation exists. As first step of the RMD-QOSM simulation model implementation, the operation of the base simulation model should be understood. As second, a good simulation model design should be made, and it should be followed by a correct implementation.

5.3 Conclusions

Finally the results of the comparative analysis are presented as conclusions.

- The QoS NSLP messages have to be included in the RMD-QOSM simulation model. That can be done by defining a new header, which carries parameters for the general and the message specific flags, and the QoS NSLP objects – RSN, RII, BOUND_SESSION_ID and INFO SPEC.
- Additional header to represent the QSPEC object has to be used. Therefore, the current existing rmd header has to be adapted to carry quality of service model specific information.
- The agents implemented in the simulation model, process only the RMD-QOSM header. Additional functionality has to be added for the processing of the QoS NSLP header. That can be done by creating a new QoS NSLP agent or modifying the behavior of the implemented agents.
- Modification of the state management in the agents is necessary. In the base simulation model only the RMD reservation state is introduced. Therefore, the QoS NSLP operational state has to be added.
- Addition to the edge nodes behavior that processes local and end-to-end messages should be provided. This is necessary to simulate the tunneling/bypassing of the end to end QoS NSLP messages within the RMD domain.
- The session ID has to be introduced and maintained in the simulation model. It will be done by a common, per node, software object that handles all sessions in a node and that binds the end-to-end messages with the local domain messages.

When at least two of the cases – tunneling/bypassing, QoS NSLP aggregation and bi-directional reservations are operating together then the new session ID module should be capable of identifying for which of them the BOUND_SESSION_ID is used.

- The mechanisms for admission control, congestion detection and congestion handling and solving for RMD-QOSM have been implemented. If a different mechanism is of interest it can be added at a later stage of the implementation.
- Additional behavior should be added to the node agents if bi-directional reservations are of interest, since they were not supported. Bi-directional operation requires two additional header fields – B flag and Reverse requested resources, and modifications of the nodes behavior to comply with the RMD-QOSM specification [BAWE06].
- If the outside domain communication is of interest the edge nodes should be capable of classifing incoming requests into a predefined DSCP. This can be done by introducing an independent model class, which performs the flow classification and returns to the agent the flow DSCP value.
- When the policy management specification is finalized, it should be added to the simulation model. A good way of doing this is to define a new software class for the policy control. The software objects of the class will have functions to handle the policy management and will provide an interface for the node agents.

6 Simulation model design

The goal of the current research, as defined in Chapter 1, is to develop a simulation model that will allow the performance evaluation of the RMD-QOSM protocol. This goal directly influences the nature of the simulation model, its level of detail and its development. The latter is organized in the phases of design and implementation of the available protocol specification and a feedback phase, where new ideas for the protocol operation have been proposed. The ideas have been also subsequently implemented and tested. The design and the feedback are presented in this chapter while the implementation is in the next Chapter 7.

Before the design of the simulation model is presented, the reader is provided with background on the principles and concepts used in the design phase. First in section 6.1 it is explained why the design of the model is split into modules. Subsequently, the goals of the design are presented, section 6.2, and the reader is then slowly led in the design process. Section 6.3 presents the general design of the simulation model and goes further with detailed discussion on some of the design modules. Central point in this chapter is the design of the module which represents the RMD-QOSM functionality. This part of the design, explained in section 6.4, contains new contributions to the existing work. The last section 6.5 is dedicated to the feedback phase. It presents innovative approaches that were considered during the simulation model implementation, Chapter 7, and initial test phase. Each approach, and the issue it addresses, is described along with the possible changes the innovation would cause to the protocol specification and simulation model design.

6.1 Used simulation model principles

The simulation model design is developed using Unified Modeling Language (UML) [LeLa01] presentation concepts. Such approach allows different abstraction levels and comprehensible simulation model representation. Another reason of using UML is the formalism of the language, which should ease the transition to formal verification and validation of the simulation model. The latter can be applied via different tools and use different formal languages. Therefore a common representation by using a well known language is a good starting point for an independent research in any of these directions.

Another concept applied in the design phase is the modularity principle. Clear separation of duties between modules improves the simulation model design and implementation. Consequently later design changes in one module can be implemented fast and without extensive modifications of other modules. In addition, the modularity principle allows each module to be tested separately and the factors that affect its performance to be isolated. Together, the UML and modularity concepts increase the simulation model design and implementation readability.

UML provides a large diversity of representations of which the design phase uses *use case diagrams*, *sequence diagrams* and *class diagrams*. The whole design process starts with the use case diagram which is built on the RMD-QOSM protocol specification. Each use case in the diagram represents one protocol activity. The protocol activities are realized by protocol entities and *a sequence diagram* presents the communication between them in the terms of the message exchange. On the base of the sequence diagrams *class diagrams* can be built. A *class diagram* models the protocol entities or parts of them as software classes which are often bound together to perform one task.

6.2 Goals of the design

The aim of the current research is to evaluate the RMD-QOSM performance in a simulated realistic network topology and based on this evaluation the protocol behavior to be optimized. To achieve the former the simulation model should implement the existing RMD-QOSM protocol functionality. To achieve the latter novice features of the protocol functionality have to be considered and included in the model. These goals allow the design of the simulation model to be simplified to include only functionality of specifications that is crucial for the performance evaluation phase.

An example of a realistic network topology is presented in Figure 6.1. In this network topology three types of components are identified – end users, network nodes and transmission link/media**.** Each of them has to be included in the simulation model. According to the modular principle each one of the networks parties is modeled independently and has well defined interfaces for communication. Some of the network parties are already available in the used simulator. For them only decisions should be made which of the simulator components can represent the parties. On the other side, these network modules that are not yet implemented have to be designed.

The network nodes can be classified in two groups – interior nodes that do not have links outside the domain and edge nodes that are bridge nodes between the end users and the interior nodes. An edge node can be ingress or egress or both. An ingress node is where a flow enters the domain and the egress node is where the flow leaves the domain.

The only part of the network that needs additional design is the RMD-QOSM, which is part of the network node functionality (Figure 6.1). Since RMD-QOSM comprises of QoS NSLP protocol and RMD functionality on top, in its design both specifications [MaKa06] and [BaWe06] are used. QoS NSLP supports a large diversity of features and broad functionality, which makes its design and implementation worth for an independent research project. In the current research only the QoS NSLP features crucial for the operation of the RMD-QOSM are considered. Nevertheless unused functional modules are included in the design phase at a high abstraction level to allow future research. On the other hand the bigger part of the RMD-QOSM specification [BaWe06] is used in the design.

It can be concluded that the design of the simulation model should allow representation of the components of a real network and should support the development of performance evaluation simulation scenarios (described in Chapter 8).

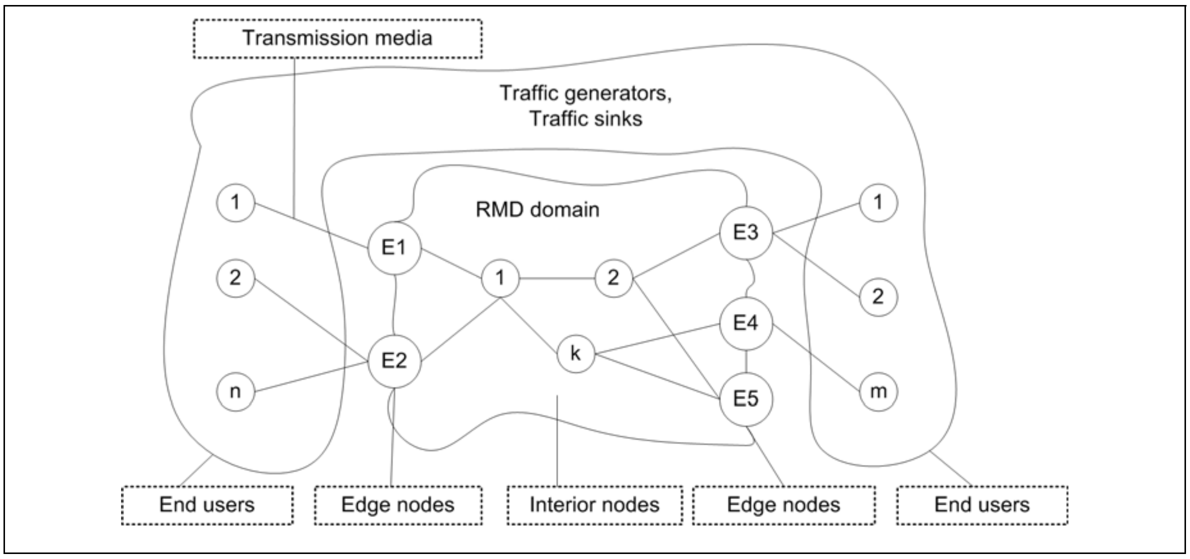


Figure 6.1 Design of the simulation model

6.3 Simulation model design

Three major design modules were mentioned in the previous section 6.2. The *end user* (Figure 6.1) that can be modeled by a workload model that uses a traffic generator (sender) or a traffic sink (destination) attached on top of edge node module. The traffic generator is the originator of the flows/sessions[3], the signaling messages and the data traffic. The traffic sink represents the destination on the data path. Both are available in the used simulator (ns2) and their interfaces allow them to be configured as required using a particular simulation scenario. In ns2 a diversity of *traffic generators* are implemented [ns manual, www, section 37.3]. The traffic generators can generate flows/sessions by using a inter generation time with a certain distribution [ns manual, www, section 24.1.2.1], such as exponential, uniform distribution and with a holding time that can have distributions, such as, exponential distribution, mix of normal distributions. By using the functionality for random variable generation possibly more distributions can be modeled, such as normal or pareto. In this assignment, the severe congestion experiments are using a uniform distribution for the flow inter-generation time with flow holding times set to infinity. All flows/sessions are generated within a very short time period, i.e., between 5 and 35 seconds simulation time. The above choices are made because the goal of the experiments is to consider what happens to ongoing flows/sessions when a link/router fails and when re-routed packets of such flows cause a severe congestion.

In addition to generating flows/sessions the traffic generators are also the originators of signaling messages and data packets that can be associated with generated flows/sessions. In ns2, the rate of data packet generation can be selected to be for example, Constant Bit Rate (CBR), Variable Bit Rate (VBR). The generation of the signaling messages depends on the message sequence chart used by the RMD-QOSM protocol.

In this assignment, for the needs of the performance evaluation, a Constant Bit Rate (CBR) traffic generator is attached to the node that sends the data. CBR traffic is chosen because major part of the experiments need to use CBR voice traffic. Furthermore, the data packets

[3] In a network it is talked about flows but inside a RMD domain in RMD-QOSM about sessions. Since in the design the edge nodes are also end users the combined term flow/session is used.

are assigned one of three preemption priorities: high, medium or low. The priority mechanism was already available in the existing base simulation model.
In ns2, the *traffic sink* can be either a Null object or a Loss monitor [ns manual, www, section 10.8]. Null objects simply discard the data packet after the header has been processed while a Loss Monitor object collects information usually, for monitoring purposes. In this assignment the traffic sink is modeled with a Loss monitor object, which allows, if required, later use of the collected information.
For unidirectional connections the sender has a traffic generator and the receiver a traffic sink, while for bi-directional reservations each end user has an attached traffic generator and traffic sink to support the bi-directional data flow.

A node module is present in end users and network nodes. The node module used in this simulation model has a layered structure, which corresponds to the IP protocol stack (Figure 6.2) in use. Not all layers of the IP protocol stack have been modeled. In this assignment the focus lies on the evaluation of the RMD-QOSM, therefore, layers that do not affect this evaluation are not included in the modeling process. These layers, physical layer, data link layer and the GIST layer, are short cut. However, the transmission speed and propagation time of the transmission link/media are included in the simulation model
From the IP layer only the functionalities: forwarding of user data packets and signaling messages, routing and DiffServ scheduling and queuing are included in the simulation model. Other functionalities are not used by RMD-QOSM and therefore are not required. Different options for routing can be used [ns manual, www, section 28.1], e.g. static, session, and dynamic. In the simulation model dynamic routing is chosen because it allows, in case a link breaks within the network topology, another route to be taken to the same destination [ns manual, www, section 28.3]. The DiffServ scheduling and queuing will be explained when the transmission links/media are discussed.
As can be seen in Figure 6.2 a node module includes UDP layer and QoS-NSLP/RMD-QOSM functionalities. Since the focus of the evaluation is the RMD-QOSM, the GIST layer and the underlying UDP or TCP layers are not included in the simulation mode. The RMD-QOSM module makes use of UDP functionality only for the identification of port (agent) address. Note however, that this layer is not supported by an interior node.
The RMD-QOSM module implements the QoS NSLP and RMD-QOSM protocol functionalities. It also processes the data messages because they are used in the sever congestion notification. This module is described in section 6.4.1.
The *network nodes* (Figure 6.1) are of two types – edge nodes (ingress and egress) and interior nodes. In the simulation model the edge nodes modules use most of the functionality described above. Interior nodes combine RMD-QOSM functionality, IP routing and forwarding functionality and Diffserv scheduling and queuing. The RMD-QOSM module functionality is responsible for performing admission control by processing signaling messages and for detecting severe congestion by using data packets. These two responsibilities are assigned to two design software classes and discussed in section 6.4.1. The interior node functionality is an independent node module that should be on top of the ns2 link functionality. For efficiency reasons the functionalities of an interior node are modeled as additional part of the transmission link/media behavior.

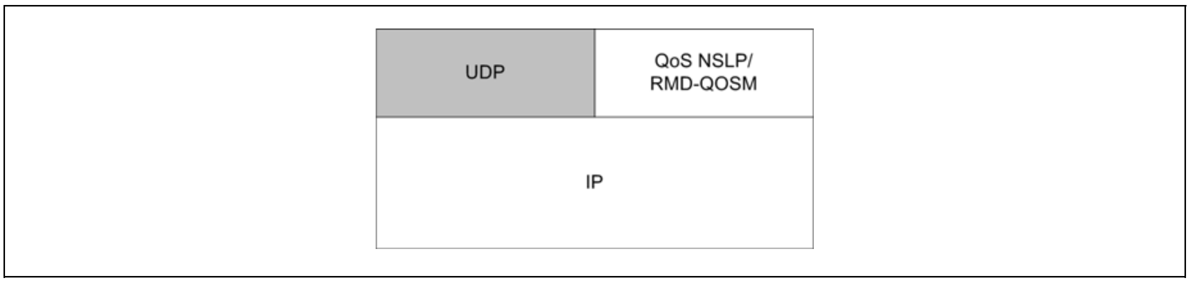


Figure 6.2 Simulation model protocol stack

As *transmission link/media* (Figure 6.1) is chosen a wired link [ns manual, www, section 6]. Other links – wireless, satellite are also part of the ns2 [ns manual, www, section 15, 16] but are not of interest in this assignment.
In ns2, duplex links are formed by two simplex links, where for each link bandwidth, delay, scheduling and queuing discipline can be specified. The link delay and capacity are managed by a separate object from the DelayLink class [ns manual, www, section 8]. Another class, the Queue class, describes the queue object (scheduling and queuing discipline) and its interface [ns manual, section 7].
The simplest type of a queuing discipline is a Drop Tail queue where the scheduling is First In First Out (FIFO). Other more complex queueing disciplines such as Random Early Drop (RED) queues or Class Bases Queuing (CBQ) can also be used [ns manual, www, section 7.3]. A special type of queue, designed for DiffServ, is used in the simulation model, the so called dsRED queue. The dsRED queue [ns manual, www, section 9.2.1] can have multiple physical queues which are served according to scheduling policy such as sliding window, token bucket, priority, null policer [ns manual, www, section 9.2.3]. Each physical queue can have a number of virtual queues which are all served as one queue. However each virtual queue has independent drop profile for example Weighted RED and Drop Tail. As result packets from one virtual queue may be dropped earlier than packets from another virtual queue but all packets are served in the order of their arrival.
In the simulation model a dsRED queue is chosen with two physical queues and priority scheduling policy. Signaling messages are put into the physical queue with higher priority and data packets into the physical queue with lower priority. The physical queue for data packets has two virtual queues – one for “encoded DSCP” data packets, and one for unmarked and “affected DSCP” data packets. Both queues have Drop Tail profile but different queue sizes. By manipulating the size of the first queue the effect of “encoded DSCP” data packets on the severe congestion behavior can be examined, which is one of the, earlier mentioned, goals of this research.
Each combination of physical and virtual queue is identified by a DS code point. Each packet is assigned also a DS code point (DSCP), which is used to classify it in the corresponding queue.
Further, the design phase is concentrated on the RMD-QOSM module.

6.4 RMD-QOSM module design

The RMD-QOSM module design is accomplished for both QoS NSLP and the RMD-QOSM. As first step in the design phase a use case diagram is created. The *use case diagram* tries to bring some classification and separation in the duties performed at the RMD-QOSM layer (Figure 6.2) of the network nodes. The RMD layer node functionality is divided in edge and interior node functionality. The reason for this is the different processing of data packets and signaling messages in these nodes. With dashed lines, the connection to the other design modules from section 6.3 is presented.

The protocol behavior is described by the RMD-QOSM message sequence diagrams, which present the participating parties, the exchanged messages and their processing. In the design of the RMD-QOSM module the message sequence diagrams for unidirectional scenarios are taken from section 3.5 and in case of bi-directional reservations from section 3.6.

The use case diagram together with the message sequence diagrams is used to create a class diagram for the simulation model. In the class diagram the RMD-QOSM tasks are assigned to software classes. Some classes have dedicated purpose, while others cover broader behavior. In the design process the already implemented simulation model is used as a starting base. Therefore some of the design decisions are based on the available base simulation model design.

6.4.1 RMD-QOSM use case diagram

As first step in the creation of the *use case* diagram the common RMD-QOSM activities were considered. These are connection management activities (establishment, maintenance, termination) and data traffic support. The connection management activities are collection of many procedures supporting different aspects of one connection:

- Procedures for signaling message exchange (QoS NSLP functionality), which make sure that the generated signaling messages have the correct header fields and that they are processed as in the specification. There is a variety of procedures due to the multiplicity of connection procedures supported by QoS NSLP.
- Procedures for maintenance of the operational and reservation states (QoS NSLP functionality).
- Procedures for authorization and authentication of the requested connection (QoS NSLP functionality), which check the initiator network rights.
- Procedures for admission control (RMD functionality), which perform check-up whether the requested connection can be supported by the network.
- Procedures for severe congestion situations (RMD functionality), which can be separated into two sub-groups according to the discussion of section 3.4.3:
    - Procedures for overload detection (interior nodes), which can discover severe congestion situations.
    - Procedures for severe congestion handling (edge nodes), which can take measures to eliminate the overload.
- Procedures for classification of flows entering the RMD domain (RMD functionality), which assign to each flow/session a DSCP value. According to the DSCP, sessions receive different treatment.

- Procedures for session maintenance (RMD functionality), which are related to the use of sessions within an RMD domain with RMD-QOSM.

All of the above activities should be performed by the RMD-QOSM simulation model. For comprehensible simulation model and for readable source code each one of the list elements is assigned one use case in Figure 6.3 and is the responsibility of a separate software class in the class diagram (Figure 6.4).
The “Data transfer” use case (Figure 6.3) represents the forwarding of data packets and their classification to particular flow/session via the flow/session ID. A level of service is assigned to each session during the initial message signaling phase. Subsequently, all packets from that session will receive the same quality of service level from the node.
A quality of service signaling protocol should be capable of differentiating between levels of service and of performing quality of service management activities. The former requires the protocol entities to classify each new connection request to a particular level of service and the latter to have mechanisms to deliver the assigned level of service and to keep that level during the connection lifetime. These tasks are represented in Figure 6.3 by the “Flow classification” and “”QoS management” use cases. The “Flow classification” use case (Figure 6.3) is responsible for applying a classification criteria and assigning to each flow/session a predefined level of service. “QoS management” use case (Figure 6.3) includes mechanisms that deliver a negotiated level of service and keep it for the lifetime of the flow/session. The “Admission control” sub-task is responsible for the QoS delivery by checking for free network resources, reserving, refreshing and releasing them. The combination of required network resources and the desired priority defines a level of service within RMD-QOSM. Currently, the admission control can only check-up for free bandwidth, without considering the priority. Another ongoing research activity is dedicated to include a priority principle in the admission control module.
The maintenance of the negotiated level of service is done by the “Overload detection”, the second sub-task of “QoS management”, and “Congestion handling” modules (Figure 6.3). “Overload detection” has the task to discover overload in the network and to apply the notification procedure from section 3.4.3.2. The “Congestion handling” task is to handle the overload according to the mechanism described in section 3.4.3.3.
The “Admission control” and “Overload detection” are functionalities of the RMD-QOSM interior nodes and “Congestion handling” is part of the RMD-QOSM edge nodes (Figure 6.3). This separation was also discussed at the beginning of section 3.4.
The two sub-cases of “State management” (Figure 6.3) provide activities of installation, refresh and deletion of the RMD-QOSM reservation or operational states. Note that only stateful QoS NSLP nodes, edge nodes in RMD-QOSM, perform all these activities. A reduced state or stateless nodes, interior nodes in RMD-QOSM, keep only aggregated reservation state or no state at all correspondingly (section 3.4).

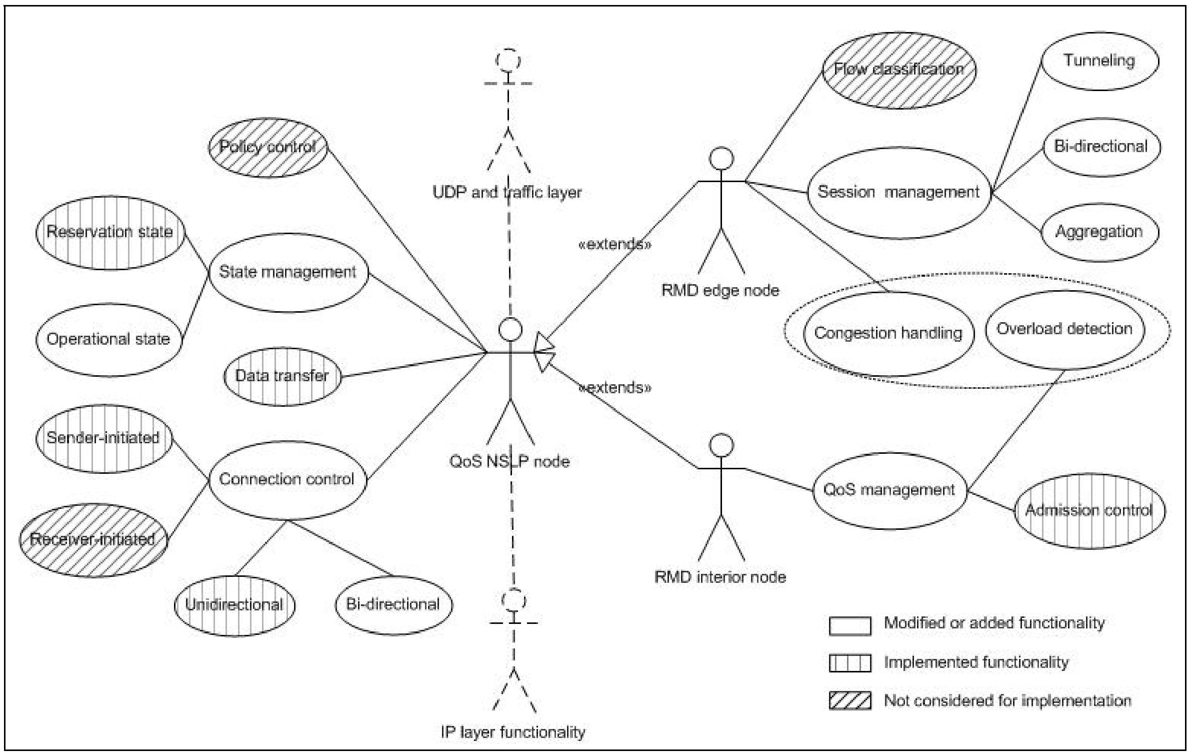


Figure 6.3 RMD-QOSM use case diagram

All signaling message exchange and the rules of the message processing are represented by the “Connection control” task from Figure 6.3. Four sub-tasks correspond to the possible scenarios defined for the QoS NSLP protocol. A reservation/session can be either “Sender-initiated” (Figure 6.3), where the RESERVE message is initiated by the data sender or “Receiver-initiated”, where the RESERVE message is initiated by the data receiver. The RMD-QOSM supports only the sender-initiated approach therefore the “Receiver-initiated” sub-task is not considered in the design. The reservations can be “Unidirectional” (Figure 6.3), where the reservation is initiated and maintained in one direction and “Bi-directional”, where two bound reservations/sessions are initiated and maintained, one in the forward direction and the other in the reverse direction.

Clearly a session management module has to be used. As it was discussed in section 3.3.1 the SESSION ID can be used to identify and bind sessions together, by using the BOUND_SESSION_ID object. Three types of binding scenarios can be identified (see Figure 6.4):

- Tunneling: per flow end to end sessions and per flow local sessions;
- Aggregation: per flow end to end sessions and aggregated reservation session
- Bi-directional: two sessions used for bi-directional reservations, one in forward and the other in the reverse direction.

Therefore the “Session management” use case (Figure 6.3) has sub-tasks for each of these binding scenarios.

The “Policy control” is also given as a case, but due to the fact that its specification is not finalized [MaKa06] it will not be implemented.

6.4.2 RMD-QOSM class diagram

The RMD-QOSM class diagram represents the protocol functionalities described in section 6.4.1, organized in software classes. Some software classes, model one of the use cases from Figure 6.3 and others multiple use cases.

The conclusions from section 5.2.3 on the required modifications to convert the RMD simulation model to a RMD-QOSM simulation model are also considered for the creation of the class diagram of Figure 6.4.

The class diagram presents two big groups of software classes – classes modeling the interior node functionality and classes modeling the edge node functionality. The RMD-QOSM interior node functionality is included for efficiency reasons in the behavior of the link. An umbrella class, "Link Monitor", see Figure 6.4, provides interfaces from the link towards the two software classes – "Admission control" and "Overload detection", which model the interior node operation. Each signaling or data packet passing the link is forwarded by the "Link monitor" to both classes. Signaling messages are processed by "Admission control" and data packets by "Overload detection". An additional class, "Measurer class" (Figure 6.4), is modeled to perform supporting functions of collection of data packets and calculation of traffic load statistics. This approach isolates the statistical administration from the message processing functionality.

The class diagram of the interior node looks almost identical to the link hierarchy of section 4.3.2. The reason is that this part of the RMD simulation model can be reused with minor changes.

The software class "Overload detection" has the responsibility to discover severe congestion situations on the link it is attached to. The description of the mechanism used by RMD-QOSM was provided in section 4.2.2 (see also section 3.4.3.2). As reminder to the reader the RMD-QOSM uses marking of data packets with "encoded DSCP" and "affected DSCP", where the former announces the level of severe congestion to the egress node. The mechanism can be followed in details in [CsTa04, WeBa06].

"Admission control" is a software class that models the admission control in an interior node with the RODA – Estimation based class from Figure 4.1 (section 4.3.4). In other words, interior nodes use reservation based admission control with estimation of the re-routed bandwidth (section 3.4.2, [CsTa05]).

The RMD-QOSM edge node is modeled as an ns2 agent, see Figure 6.4. Since the interior nodes use the reservation based approach, the edge nodes also should apply it. Therefore the RMD-QOSM edge node is modeled with the RODA agent from section 4.3.1. The best approach to represent the edge node would be by using two agents – RMD agent on top of QoS NSLP agent. The reason for this is very simple. RMD-QOSM consists of two functionalities – the QoS NSLP general signaling functionality and the RMD model for quality of service definition and provisioning. If two agents are to be used each agent can process the signaling messages in turn and to manage only the important to him information. The latter is important when other QOSM might have to be used on top of QoS NSLP. Nevertheless such approach is time consuming and requires restructuring of the available implementation – two reasons why this is not done in the current research.

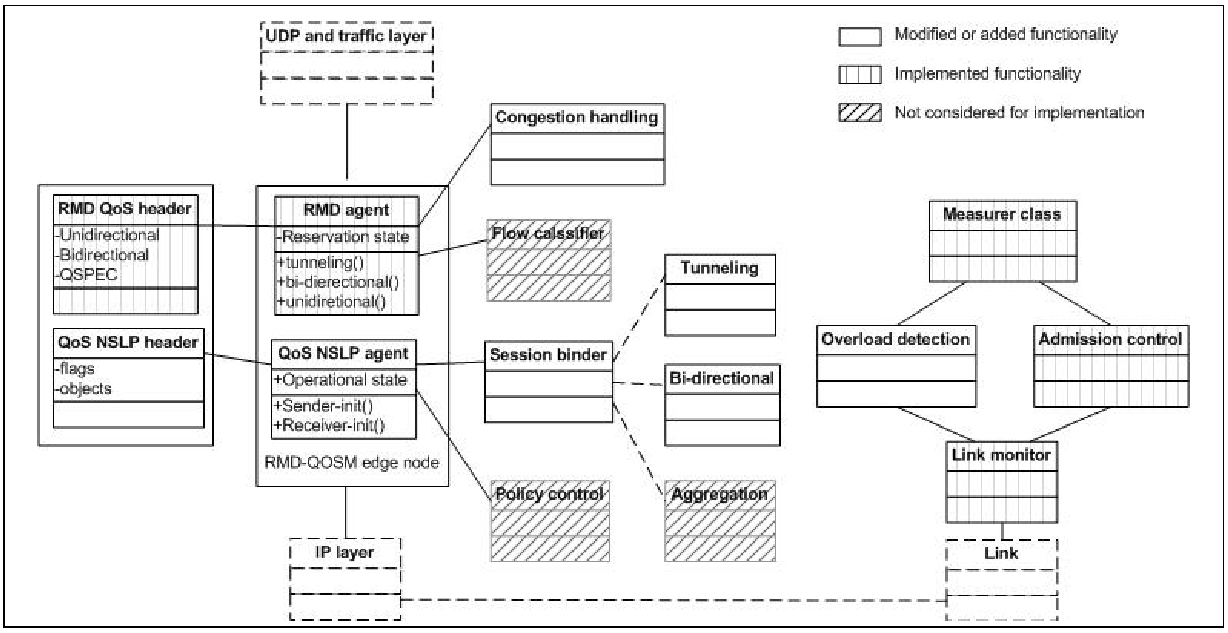


Figure 6.4 RMD-QOSM class diagram

In the developed RMD-QOSM simulation model the edge node is modeled with one ns2 agent. This agent is responsible for the processing of the signaling and data messages, according to the rules from [BaWe06] and [MaKa06]. In the class diagram (Figure 6.4) the RMD and QoS NSLP functionalities are separated to make the reader understand better the relation between the use case diagram (Figure 6.3) and the implementation.
An RMD-QOSM edge node has a complex structure and is responsible for multiple tasks. Note that ingress and egress nodes are both edge nodes but they differ in functionalities. Still some similarities can be found such as policy control and session identification.
The "Policy control" software class is carrying out the tasks of the "Policy control" depicted in Figure 6.3. All authorization and administrative procedures are to be the responsibility of a separate module that has little in common with the protocol functionality modules.
The RMD-QOSM protocol recognizes connections via their SESSION ID, see section 3.3. To be able to use the binding of sessions a new module "Session binder" is added to the class diagram (Figure 6.4) of the new RMD-QOSM simulation model. It corresponds to the "Session management" use case of Figure 6.3. The "Session binder" software class keeps SESSION ID values and binds two sessions together when such relation exists (examples are given in section 3.5 and 3.6). The Session binder has three sub-modules allowing processing for each of the binding scenarios that were previously described. In order to support the aggregation binding scenario, the "Session binder" software class has to be extended with "Aggregation" sub-class (Figure 6.4). Aggregation is not a topic of research in this paper.
Another similarity between the ingress and egress edge nodes, as stateful nodes, is the state processing. All procedures from "State management" (Figure 6.3) are modeled as part of the RMD-QOSM edge agent behavior (Figure 6.4). The operation state, newly added functionality, is modeled with the QoS NSLP header objects – RSN, RII, SESSION ID and BOUND_SESSION_ID (section 3.3.1). Note that in the RMD-QOSM simulation model

each flow/session has its own ingress-egress pair. In real network one edge node keeps information on all flows/sessions that pass trough it thus it would keep many operation states. The same it true for the reservation state, which is represented by the requested resources and the current functional state of the agent (see section 4.3.3).

In the simulation model there is only one edge agent, which models the behavior of ingress and egress nodes. To distinguish between them the functional state is used. Since no two functional states are the same and the combinations signaling message – functional state are unique no duality exists. The functional state transitions, discussed in section 4.3.3, are slightly changed to include the signaling message exchange during bi-directional reservations.

All described above edge node functionalities make use of the RMD-QOSM header. The header itself is modeled by two headers in the RMD-QOSM simulation model. The first header represents the fields of the QoS NSLP header except the QSPEC object (section 3.3.2) and it is modeled by the "QoS NSLP header" software class (Figure 6.4). The second header, "RMD header" software class (Figure 6.4), models the RMD QSPEC object of the RMD-QOSM header and includes all fields as specified in 3.4.1.

Both edge nodes, ingress and egress, are modeled by their message generation and processing of arriving signaling messages as the header processing in each edge node differs. All messages formats and header values and how they should be processed are modeled as in the specifications [MaKa06, BaWe06]. The message generation is extended with generation of local messages to support the tunneling in RMD-QOSM. The generation and processing of bi-directional reservation messages is also added (Figure 6.4). The latter resulting in specifying new combinations singling message – functional state.

One function in the edge agent manages the processing of all arriving messages and it had to be re-organized to include the new QoS NSLP functionality. Currently upon arrival signaling message is classified according to its QoS NSLP type. Subsequently the message header is processed according to the rules from the specifications. As it was mentioned the receiver-based procedure of QoS NSLP is not designed (section 6.4.1).

The software class "Flow classifier" (Figure 6.4) corresponds to the "Flow classification" use case of Figure 6.3. The class has the task to assign to each flow/session, entering the RMD domain, a DSCP value, which indicates the level of quality of service. That should be done by the ingress node. This RMD-QOSM module is not considered for implementation since only connections within RMD domain are simulated**.**

An egress node specific task is the handling of severe congestion, which task has a dedicated class, namely "Congestion handling" (Figure 6.4). It includes the procedures of the use case with the same name (Figure 6.3) and the mechanism described in sections 3.4.3.3 and 4.3.3. Since in the simulation model each flow has its own ingress-egress pair, no single egress knows the level of severe congestion. Therefore an instance of the "Congestion handling" software class is attached to one physical node and gathers information on all flows that have the same node as destination. To model bi-directional reservations modifications are done such that ingress nodes also can communicate to a "Congestion handling" instance. A change is made such that when the "Congestion handler" instance works with ingress nodes the chosen flows are stopped and when it works with egress nodes NOTIFY messages are sent.

Before the discussion on the RMD-QOSM design is ended one peculiar characteristic is explained. First, in the simulation model data packets are used in the detection and notification of severe congestion and are processed by the egress edge node and the "Congestion handling" software class. Second, edge nodes classify messages according to their QoS NSLP type. Due to these two facts data messages are also assigned special "DATA" QoS NSLP type, which is for use only in the RMD-QOSM simulation model.

6.4.3 Support software classes

All design modules described in section 6.3 and in sections 6.4.1 and 6.4.2 are necessary to model the behavior of the evaluated model and of a real network. Nevertheless without proper means of data collection this simulation model cannot be meaningfully used. Two software classes are used to collect statistics on the work of the simulation model. These are the already discussed in section 4.3.6 flow monitor and the new Scalability software class.

The "Scalability" software class works together with the "Session binder" class and keeps information on the number of flows/sessions and the number of agents that are currently installed in the network. In order that the information is up-to-date, each new entry or deletion of an old one, in the "Session binder" class, leads to change of corresponding information in the "Scalability" software class. The gathered information is preserved in a file each small period of time to allow detailed observation of the network state.

6.5 Re-designing issues

After the initial simulation model implementation when all basic specifications functionality was included, several issues have been discovered at the test phase of the simulation model. Some of them originated in this research, while others have been considered already at the time the research have started. All issues have not been included in the specifications used in this research but are available in the new versions.

The scope of the issues differs – some of them required optimizations of available solutions while others did not had solutions so far. Each of the issues is first theoretically researched and a solution is proposed. The resulting solutions are implemented and tested whether they function correctly. Some of the solutions perform well while others not as good as expected.

Subsequently, simulation experiments are performed to investigate the advantage of using the improved solution. The end goal is the improvement of the existing protocol specification. Eventually it can be said that the goal is achieved.

The issues can be classified in three groups:

- severe congestion detection and notification issues,
- flow termination issues,
- severe congestion situation in bi-directional reservations issues.

The new severe congestion algorithms discussed in this report, see also [WeBa06], are designed in strict cooperation between Georgios Karagiannis and the author of this report.

6.5.1 Severe congestion detection and notification issues

Two issues are address in the sub-section. The first is the used CBQ queues that can support priority scheduling and it can allow classes to borrow bandwidth from other classes that have unused bandwidth. With CBQ queuing all types of data packets are classified in one queue. If the queue is full potentially marked packets are dropped while there are still unmarked packets in the queue. Since marked data packets are used in severe congestion notification their transportation is crucial.

One alternative solution proposes the number of incoming marked bytes during one measurement period to be counted and to be ensured that the same number of marked bytes leaves the link. If marked packets are dropped then unmarked data packets are marked at leaving the link, such that the above number is kept. This solution nevertheless does not perform well in all situations. Imagine that two flows enter the node – flow 1 from overloaded link and having marked packets and flow 2 from a not overloaded link without marked packets. If the marked packet from flow 1 is dropped it is possible that an unmarked packet from flow 2 is marked to compensate for the drop. In this case flow 2 might be stopped without solving the congestion.

Another alternative, chosen for implementation, eliminates as much as possible, the chance that marked packets are dropped by using different dropping rules for marked and unmarked data packets. This is accomplished by using DiffServ Random Early Detection (dsRED) queuing disciplines [EtPi00]**.** The concept of dsRED queues, along with their modeling, is described in section 6.3.

The second issue is that level of severe congestion higher than 100% cannot be signaled correctly to the edge nodes. The old implementation requires marking of more data packets than can pass the link. A remarking factor N is introduced, where N is the proportion between bytes above the severe congestion restoration threshold and the actual number of bytes that is marked. For example with factor N = 2 and congestion of 180% only 90% of the passing bytes will be marked and with factor 3 only 60%. What remark proportion factor is to be used depends on what levels of severe congestion are expected on the RMD-QOSM network. Note that the same value of N has to be used by all nodes in the RMD-QOSM network domain.

The use of N requires the “Overload detection” and “Congestion handling” software classes to be informed of the used values of N and to include its processing in the used mechanisms.

6.5.2 Flow termination issues

A proposal was made such that the edge nodes are notified about the level of congestion contributed by each source. It was expected that these innovation would improve the mechanism for solving severe congestion situations by allowing more bandwidth to be kept on link where this would be possible. To implement the new proposal the existing mechanism was modified.

In the “Congestion handling” software class (Figure 6.4) a flow is chosen for termination when it has marked data packets no matter from which source the flow comes. With the modifications of the mechanism the marked data packets are collected per source. In this case the “Congestion handling” software class knows the severe congestion level coming

from each source. Since each egress node keeps information from each corresponding ingress node it is being talked about ingress-egress pair aggregate. When flows, coming from one source, are to be terminated they are chosen such as to solve the level of congestion of the same source.
Major factor when choosing flows is the flow priority – first low priority flows are stopped before other priorities are considered, see section 3.4.3.3. In the new proposal the priority principle is kept for each source. It was discovered that due to this fact the priority principle may not be kept for the whole network even if kept for each source. The performance of both mechanisms is evaluated in section 8.3.
The above described variation requires modifications in the "Congestion handling" software class such that the severe congestion level per source to be calculated and the process choosing flows to be done for each source.

6.5.3 Severe congestion situation in bi-directional reservations issues

The mechanism used in the base simulation model for unidirectional reservations chooses flows to stop looking at the reservation size on the severe congested path. When bi-directional reservations are used reservations on two data paths have to be initiated and maintained. If congestion happens on the one data path, flows can be chosen using the reservation size on the same path or the reservation size on the opposite path. It is desired that the most possible bandwidth in bi-directional reservations is preserved in the network. A modification is proposed where in unidirectional reservations the reservation on the forward, and only path, is used and in bi-directional reservations the smallest reservations on the path opposite to the severe congested, are processed.
To test the proposal the "Congestion handling" software module is re-modeled to distinguish between unidirectional and bi-directional reservations. After the type of reservation is detected the module applies different criteria to choose flows to stop.
Another issue comes from the fact that in bi-directional operation severe congestion can occur simultaneously in both directions. If both edge nodes choose flows to terminate it might cause more reservations to be terminated than necessary. The reason is that bi-directional reservations chosen for termination on the one path partly solve the congestion on the other path. To avoid such undershoot the ingress agent behavior is modeled to keep information on the stopped bandwidth on the forward path. When a NOTIFY message arrives the ingress node checks if the message would cause undershoot. If this is the case the message is not processed otherwise it is. Below, a brief example is given.
Let us imagine bidirectional reservation, where each of the generated flows has the same reservation in forward and reverse direction - 1 resource unit. A severe congestion on both paths occurs with overload of 5 resource units on the forward path and 3 resource units on the reverse. The egress sends NOTIFY messages for 5 flows (for the forward path) and the ingress stops 3 flows (for the reverse path). The three stopped flows by the ingress correspond to 3 reservation units stopped overload on the forward path. This leaves 2 resource units overload on the forward path. The first three NOTIFY messages are not processed, when they arrive at the ingress, because 3 resource units overload was already stopped on the forward path. The fourth and fifth NOTIFY are processed because the

resource units they want to stop are not yet terminated by the ingress node. These two messages solve the left 2 resource units overload on the reverse path.
If the scenario is turned such that the forward path has 3 resource units overload and the reverse path has 5 recourse units overload none of the NOTIFY messages is processed. The reason is that the ingress node has stopped more bandwidth that the egress wants to stop.
The described mechanism is an optimization of the existing one of section 4.2.3 and the performance of both mechanisms are compared in section 8.6.

7 Model implementation in ns2 simulator

This chapter describes the implementation of the simulation model based on the design described in Chapter 6. In particular, the software classes of section 6.4.2 are implemented as classes or hierarchy of classes in C++ or OTcl code. In Chapter 4 the base simulation model with its existing classes were presented. The design of Chapter 6 extended the classes functionality and added new classes. This current Chapter 7 gives general description on the already implemented classes and gives more attention to the new implementation. When it is appropriate, pieces of code are given in the text or appendix. All classes can be found in the source code and references to that are given.
In section 7.1 the implementation of the simulation model functionality available in ns is discussed. Subsequently section 7.2 presents in several subsections the implementation of the RMD-QOSM functionality. Some software classes have dedicated sections (7.2.4 and 7.2.5) while others are united in one section, i.e. section 7.2.1.
Finally the implementation of some support classes for monitoring is presented.

7.1 Simulation model implementation

The simulation model implementation is written in C++ and in OTcl because these are the programming languages used by the simulation environment – the network simulator (ns) [ns, www]. The latest available version, ns2, at the time of the research is used – version 2.29, which includes a C++ compiler. Additionally in the installation package Tcl/Tk release 8.4.11 [ns, www] and OTcl release 1.11 [ns, www] are included. The choice of simulation environment and programming languages was limited because the existing simulation RMD model was already implemented in ns. ns2 was installed in Cygwin [cygwin, www] to simulate Linux environment on a Microsoft Windows® platform.

In section 6.3 it was mentioned that some of the design modules are available in ns2 and that they have standardized interfaces. A choice for each module was made. This section only describes how these can be configured for use in the current simulation model. References are given to the sections of the ns2 manual where more detailed description of the modules can be found. The actual values to be set are specified in the chapter on experiments, Chapter 8.
The design modules already implemented in ns are the traffic generators, traffic sinks, transportation protocol module, link and queuing functionality. The chosen traffic generator is a CBR source [ns manual, www, section 37.3] attached on top of the RMD-QOSM, which is created and set by the commands [utils_new.tcl]:

```
set gen [new Application/Traffic/CBR];
$gen set packetSize_ $pktSize_;
$gen set rate_ [expr $pktSize_ * 8 / $interval_];
$ingress_agent attach $udp $gen
```

The second two lines show how the packet size and the rate can be set. The values `$pkSize` and `$interval` are chosen by the tester [main.tcl]. The last line attaches the traffic generator and the UDP transport agent to the RMD-QOSM agent, where the `attach` procedure is writen in [rmd-lib.tcl]. The traffic sink object, Loss monitor, is created and attached to the agent with the commands:

```
            set sink [new Agent/LossMonitor]
            $egress_agent attach $sink
```

This procedure is done for one direction of data transfer, which means that in bi-directional reservations the same procedure is repeated for the reverse direction [utils_new.tcl]. Then a CBR is also attached to the egress node and a Loss Monitor to the ingress node.
The choice of transportation module is UDP protocol. Therefore the UDP agent [ns manual, www, section 32.1] is created and set with the commands:

```
            set udp [new Agent/UDP]
            $udp set fid_ 1
            $udp set prio_ 1
            $udp set packetSize_ $pktSize_;
```

The IP header field `fid_` is used such that a link monitor, see section 7.3, can distinguish between signaling and data packets. The `prio_` field carries the code point used in dsRED queuing. All data packets upon generation have an unmarked code point (see dsRED queue configuration).
The used IP layer functionality dynamic routing is set by the command:

```
            $ns rtproto DV
```

The links in the simulation model are duplex links, modeled as two simplex links each with attached RMD-QOSM admission control and severe congestion detection modules (section 6.4.2). The command to create such duplex link is [utils_new.tcl]:

```
      $ns duplex-newrmdlink $src $dst $bw $del dsRED RMDADC RMDCH
      simplex-rmd-link-with-dsRED $src $dst //configuration
      simplex-rmd-link-with-dsRED $dst $src //configuration
```

The source (src), destination (dst), bandwidth (bw) and delay (del) of the link can be set by the user. The default values [rmd-lib.tcl] of the admission control object (RMDADC) and the sever congestion detection object (RMDCH) are automatically attached. The command `duplex-newrmdlink` calls a procedure in [rmd-lib.tcl], where a standard ns2 duplex link is created, with the first five parameters. To it the RMD-QOSM objects are attached. Before the dsRED queuing can be used the queues have to be configured. This is done by a special procedure for each simplex link:

```
                  proc simplex-rmd-link-with-dsRED {src dst} {
                  global ns

                  //Group 1
                  set dsredq [[$ns link $src $dst] queue]
                  $dsredq meanPktSize 40;
                  $dsredq set numQueues_ 2;
                  $dsredq setSchedularMode PRI;

                  //Group 2
                  $dsredq setNumPrec 2;
                  //     addPHBEntry     <codePoint>     <physicalQueue>
                  <virtualQueue>
                  $dsredq addPHBEntry 0 0 0; #signaling packets
                  $dsredq addPHBEntry 1 1 0; #marked packets
                  $dsredq addPHBEntry 2 1 1; #data packets

                  //Group 3
                  $dsredq setMREDMode DROP
                  // configQ <physicalQueue> <virtualQueue> <queueSize>
                  $dsredq configQ 0 0 1100
                  $dsredq configQ 1 0 1638
                  $dsredq configQ 1 1 1450
                  }
```

Group 1 (see the above code) creates a dsRED queue object, implemented in ns2, and sets the number of physical queues (numQueues_) to two. A priority scheduling between these queues is chosen (setSchedularMode). In Group 2 for each physical queue two virtual queues (setNumPrec) are defined and a code point is assigned to each pair physical-virtual queue. Finally, in Group 3, the queue length of each pair is configured and the scheduling is set to be FIFO (DROP).
As last, the uniform distribution of flow arrivals is discussed. The time at which a flow is started is taken from a random number generator (RNG). The RNG generates values with random distribution. The RNG is seeded and used as follows:

```
                  $defaultRNG seed 5858
                  $defaultRNG uniform lowerBoundry upperBoundry
```

7.2 RMD-QOSM module implementation

7.2.1 RMD-QOSM link objects

The link class in the ns2 simulator is characterized with bandwidth, delay, direction and type of queuing. The existing RMD simulation model attaches additionally to the link, four new software classes (Figure 6.4), which were discussed in section 6.4.2. The first software class, “Link monitor” (Figure 6.4), is implemented with the class RMDMon [rmd-mon.h/cc]. RMDMon manages the access to the software classes “Admission control” and “Overload detection” (Figure 6.4), which model the behavior of RMD-QOSM interior node.
“Admission control” is implemented by the RODAADC C++ class [rmd-adc.h/cc] from the class hierarchy in section 4.3.4, Figure 4.4. The implementation of “Overload detection” is the DampenedRateProportionalMarkingBWMeasuredRMDCH class [rmd-ch.h/cc], from the class hierarchy in section 4.3.5, Figure 4.5.

The fourth software class attached to the link is the "Measurer class" (Figure 6.4), which is implemented by the CSATAMeasure C++ class [csata-measure.h/cc] and has supportive internal function.
An instance of the RMDMon class is attached to each simplex link with the command:

```
            [$self link $n1 $n2] init-rmd-mon $adctype $sevcontype
```

The whole procedure `init-rmd-mon` can be found in [rmd-lib.tcl]. First an RMDMon instance is attached to the ns2 link. Second, the source, destination and link bandwidth values that have supportive role, are set. Third, instances of the "Admission control" software class (Figure 6.4), and the "Overload detection" software class (Figure 6.4) are created. Finally, these instances are attached to the ns2 link via the RMDMon class.
As consequence each packet that passes the link is forwarded to the "Admission control" and "Overload detection" implementation classes. This is done via the three methods `in(Packet* p)` for arriving at the link packets, `out(Packet* p)`for leaving the link packets, after packet drop, and `drop(Packet* p)` for packets that are dropped.
Each of the methods has its corresponding method with the same name in the "Admission control" and "Overload detection" implementation classes. Other methods that have supportive role are part of the RMDMon class and can be found in [rmd-mon.h/cc].

7.2.1.1 "Measurer class"

The class CSATAMeasure has the supporting role to collect information on the traffic load and to calculate the average bandwidth on the link. This value is used to detect whether the traffic load is above the allowed threshold and, as result, if severe congestion has occurred.
The number of bytes that pass the link is collected by the method `virtual inline void recv_bytes(int newbytes, int classid).` At the end of each measurement period the method `virtual void measure(void)` is called. It calculates the bandwidth on the link, for the measurement period, which bandwidth value can be retrieved by calling the method `virtual double avg_bw(int classid).`A methods ca be found in [csata-measure.h].

7.2.1.2 "Admission control" implementation

The RODAADC class implements functionalities necessary to perform reservation based admission control, see section 3.4.2. In the simulation model the estimation approach based on refresh messages [CsTa04] is chosen. As a reminder, this mechanism allows discovery of re-routing flows and estimation of the re-routed load, which are used for the processing of signaling messages.
Due to the multiple choices of admission control mechanisms, before simulation is started, the type of admission control algorithm is set [main.tcl] by:

```
            set CAC_solution 1; # 1 : RODA admission control.
```

In the estimation based admission algorithm two thresholds are used, one to process new reservation requests and another for refresh messages to discover re-routed flows. Both thresholds are specified [main.tcl] by:

```
                  set CAC_admission_threshold(0) 100;
                  # CAC admission threshold for first RMD class (in %
                  of total link BW).
                  set CAC_refresh_threshold(0) 100;
# CAC refresh threshold for first RMD class (in % of total link BW).
```

Indexing of the threshold is used because each PHB class can have different thresholds.
The RODAADC class is a child class of the RMDADC class. As such the RODAADC class inherits methods that provide information, which is used for debugging or to manage internal class variables. These are given in [rmd-adc.h].
A special timer, RODAADCCellTimer class [rmd-adc.h/cc], is attached to the RODAADC to delimit the end of a cell in the sliding window algorithm, see section 3.4.2. Furthermore a signaling message that arrives on the link is processed in the `out{}` method of the RODAADC class [rmd-adc.h/cc]. The method contains instructions to process three types of RESERVE messages – for reservation request, for refresh and for release. In the processing of the first two RESERVE messages the estimated re-routed load is used. When a (release) RESERVE message arrives the correct cell is calculated, see section 4.2.1, and then the resources are released.
In the processing of a RESERVE message for reservation request modifications are implemented to include the processing of the new RMD-QOSM header fields Admitted hops and Hop U (see section 3.4.1). In Appendix B, Table B. 1, part of the `out{}` method is presented and the new processing is highlighted.

7.2.1.3 “Overload detection” implementation

The mechanism for discovery of severe congestion and notification that is used in the “Overload detection” software class, was described in section 4.2.2. In short severe congestion is discovered when the total link load of data packets goes above the detection threshold. To notify the level of congestion the interior node marks data packets with “encoded DSCP”, which amount is calculated as the number of packets above the restoration threshold. All other passing data packets have “affected DSCP” (see section 4.2.2).
In the class hierarchy in Figure 4.5 “Overload detection” is implemented by the DampenedRateProportionalMarkingBWMweasured class [rmd-ch.h/cc], called here for short Dampened class. The type of mechanism to use is chosen before simulation with:

```
            set sev_con_detection_solution 2; # 2 : Dampened bandwidth
            measured rate proportional marking.
```

The detection and restoration thresholds can also be different for each PHB class and indexing is used. Both thresholds were explained in section 4.2.2 and are to be included in the new RMD-QOSM specification [WeBa06].The thresholds are set at the beginning of the experiments with the command:

```
                  set BW2_CT_CRT(0) { 103 100 };
```

The first value is the severe congestion detection threshold and the second is the severe congestion restoration threshold.
The Dampened class processes data packets when they enter the node and when they leave it. At arrival the data packets are just count to discover overload and at departure, after packet drop has happened, they are marked if necessary. The `in(Packet* p)` method of BWMeasuredRMDCH class [rmd-ch.h/cc] implements the processing at arrival. The method functions together with the CASATAMeasurer class (see section 7.2.1.1). Addition to the `in(Packet* p)`method is made to count the incoming marked packets that are used when the blocking probability is calculated:

```
            void BWMeasuredRMDCH::in(Packet* p) {
                  ………………………………………………
// Marked packet arrives, IP header field prio_ is 1
                  if (iph->prio() == 1)
                        marked_in_ += hdr_cmn::access(p)->size();
                  ………………………………………………
            }
```

At departure the data packets are processed by the `out(Packet* p)` method of the Dampened class. If severe congestion is detected unmarked data packets are remarked with “encoded DSCP” to notify the level of severe congestion. When all marking has finished the rest of the unmarked packets are re-marked with “affected DSCP”, see section 4.2.2.
An addition was made such that the prio_ field of the IP header for “encoded DSCP” data packets is set to 1. That is necessary because of the use of dsRED queues, see section 7.1.
Before any re-marking can happen the interior node has to detect the overload and to calculate how many data packets should be marked. The algorithm, described in section 4.2.2 for the Dampened class, is implemented by the `update(void)` method of the Dampened class. This method is called at the end of each measurement period, see section 4.2.2, which is delimited by the expiration of a timer, `BWUpdateTimer` class. In the calculation the remark proportion N is implemented to support notification of more than 100% severe congestion. This is shown in Appendix B, Table B. 2.
Additional small mechanism is implemented to calculate the duration of the severe congestion in terms of measurement periods.
Note that the number of cells in the sliding window, which is part of the used algorithm, has major influence. If the cells are too few undershoot happens but if they are too many, the time to solve the congestion increases. This was proved experimentally. The number of cells is set in [rmd-lib.tcl]:

```
            RMDCH/BWMeasured/RateProportional/Dampened              set
            reset_marked_bytes_cells_ 6
```

7.2.2 RMD-QOSM headers

“RMD header” and “QoS NSLP header” software classes (Figure 6.4) together model the RMD-QOSM header. The “QoS NSLP header”, see section 3.3.1, is implemented in [nslp-header.h]. Of all QoS NSLP header fields described in [MaKa06] only the field for the protocol version is not included since there is only one version. All other implemented fields are:

- QoS NSLP messages type is represented by nslp_type. There are five values for RESERVE, QUERY, RESPONSE, NOTIFY and DATA message (see end section 6.4.2).
- The type of quality of service model, used in combination with QoS NSLP, is represented by qos_model. In the current simulation model this is Resource Management in DiffServ (RMD). Options to add other QOSM are left open.
- All QoS NSLP flags (section 3.3.1), acknowledge flag, scope flag, replace flag, tear flag, and for QUERY message reverse flag, are represented by variables.
- The objects of the QoS NSLP header (section 3.3.1), REFRESH PERIOD, RSN number, RII number, BOUND SESSION ID object and INFO SPEC object, are also implemented by variables.
  The implementation of the INFO SPEC is limited to values meaningful for the RMD-QoSM – for result of a reservation or refresh procedure and for severe congestion notification. Additionally general occurrence of protocol, transient, permanent or QOSM specific errors can be signaled.
- the RMD-QoSM QSPEC is represented by additional header that is discussed next.

The “RMD header” is implemented in the file [rmd-header.h] with new variables added to it. A separate header is chosen for implementation of the QSPEC object because this object is QOSM specific. Such approach allows easy adding of different QOSM on top of the QoS NSLP simulation header. The QSPEC is implemented as follow:

- The type of the PHR container (section 3.4.1.2) is represented by phr_type variable.
- The type of the PDR container (section 3.4.1.3) is represented by pdr_type variable.
- The SESSION ID filed is introduces for session identification.
- The header M flag and S flag, described in section 3.4.1.2, are implemented, along with the preemption priority, a flow can have within one PHBclass.
- The fields used in the partial release procedure (section 4.2.7) are kept, base simulation model, along with the new implementation of the fields, Hop_U flag and Admitted hops, used in the RMD-QOSM (section 3.4.1.2).
- The QoS Desired object (section 3.4.1.1) is represented by requested_ and dscp_ variables to indicate the reservation in forward direction and a DSCP value. Bi-directional reservations (section 3.4.1.2) are supported by the introduction of two new fields B flag and Reverse requested resources.

7.2.3 Node implementation

An edge node that uses reservation based admission control is implemented in the RODAEdge class, which is a child class of the RMDEdge, section 4.3.1. Therefore methods from RMDEdge are inherited. For reservation based admission control a reservation state, along with operational state is kept. They both represent the “State management” (Figure 6.3) use case and they are implemented by internal class variables, see Appendix B, Table B. 3. The “RMD-QOSM edge node” software class (Figure 6.4) is implemented in the RODAEdge class.
The functional states of the edge nodes are the same as in the base simulation model (see section 4.3.3 and they are used in the sending and receiving of signaling messages.
The implementation of sending signaling messages is combination of Otcl commands and RODAEdge methods. A RODAEdge agent generates three types of message, besides the refresh explained later, (request) RESERVE, (release) RESERVE and NOTIFY.
The generation of flows is implemented in OTcl code but the exchange of signaling message is in the C++ code. When a flow is started in the OTcl code the command “QoSRequest” is given to the ingress agent and as results in calling two functions. The first, `send_reservation_message_e2e(void)`, sends the end-to-end RESERVE message, and the second, `send_reservation_message(void)`, sends the local RESERVE message. Note that when bi-directional reservations are used only the local message is sent.
An example how the new end-to-end RESERVE message is created in presented in Appendix B, Table B. 4. In Table B. 4 it can also be seen how the new QoS NSLP header fields are set. The implementation of each message, generated in the RMD-QOSM simulation model, is changed to include the processing of the new header. The generation of all other messages can be found in the source code.
The severe congestion handling functionality is also implemented in the OTcl code. When a flow is chosen for termination by an egress agent, the command “terminate” calls the C++ method `send_congestion_report_message(void)` and a NOTIFY message is sent. As result the ingress agent stops the flow in the OTcl code and starts a release procedure, see section 3.5.4.4, by the command “QoS Release”. This command calls the `send_release_message(void)` method and (release) RESERVE message is sent.

The timeout and refresh procedures are implemented by the use of two timers, `ReportTimeoutTimer` and `RODARefreshTimer`, respectively. The timeout procedure is used to make sure that a session initiator does not wait infinitely a session establishment to happen. When the timeout timer expires the `timeout(void)` method is called, which stops the request procedure.
A refresh procedure is necessary to implement because RMD-QOSM is a soft state protocol and the reservation states have to be updated. At the expiration of the refresh timer a refresh message is sent by the method `send_refresh_message(void)`.

The receiving functionality is implemented by one method with complex behavior, `recv(Packet*, Handler*) [rmd-edge.h/cc]`. Upon message arrival the NSLP message type is used to choose the appropriated processing. Four big processing scenarios are

implemented, for DATA, RESERVE, RESPONSE and NOTIFY messages. QUERY messages are not use by the RDM-QOSM and their implementation is missing.
Processing of three types of RESERVE messages is implemented and the implementation closely follows the specifications [MaKa06, BaWe06]. The first type is for reservation request. In unidirectional reservations two RESERVE messages are processed, end-to-end and local RESERVE. In the implementation of bi-directional reservations only local message are used. If the reservation is successful new local RESERVE for the reverse direction is generated. This processing behavior can be followed in the RODAEdge `recv(Packet*, Handler*)` method in the switch `case PDR_RESERVATION_REQUEST.`
The second type of RESERVE comes from the refresh procedure. If it is used in unidirectional reservations a RESPONSE is generated and if it is for bi-directional reservations new (refresh) RESERVE for the reverse direction is sent. The same processing is true for the last (release) RESERVE message. The processing of these two messages is implemented in `case PDR_REFRESH_REQUEST` and `case PDR_REQUEST_INFO` correspondingly.
For RESPONSE messages two choices are implemented, `case PDR_RESERVATION_REPORT` and `case PDR_REFRESH_REPORT`. The first is to process a response from reservation request and the second – a response from refresh procedure. The detailed processing can be found in the specifications [MaKa06, BaWe06].
Only one type of NOTIFY message exists. The implementation of its processing is extended to include the mechanism from section 6.5.3. An additional check-up is performed whether the RMDEdge agent that received the NOTIFY, has already stopped flows. Depending on the result the NOTIFY message is processed or not. This is done in co-operation with the `RMDEdgeSevereCongestionHandler` class.
All above messages are signaling messages. Since edge nodes handle severe congestion they also should process data packets. That is implemented as data packets have special value of the NSLP message type – DATA. This choice is made only out of convenience. A data packet can arrive form the application, then it is just forwarded, or it can arrive from the sender node, then it is processed. The processing is implemented in `case DATA` [rmd-edge.h/cc] and it is a simple check-up whether the data packet is marked. If this is the case the class `RMDEdgeSevereCongestionHandler` is accessed.
To process data packets a RODAEdge agent has to be attached to the traffic generator and the traffic sink. This is done with the commands `"CPP-attach-agent"` and `"CPP-attach-reverse-agent".` The latter is implemented for bi-directional reservations, see section 7.1, to support data transfer in both directions. At the end of the reservation the commands `"CPP-detach-agent"` and `"CPP-detach-reverse-agent"` are called.

Two classes work together with the RODAEdge class to implement the behavior of RMD-QOSM edge node. These are the `RMDEdgeSevereCongestionHandler` and `RMDEdgeSessionBinder` classes. The `handle_sessions(int option, int session_id_)` method provides means the methods of the `RMDEdgeSessionBinder` to be accessed. The `RMDEdgeSessionBinder` is discussed in section 7.2.5.

The two methods `recv_marked_packet(int pktsize)` and `recv_affected_packet(int pktsize)` are used to pass information on "encoded DSCP" and "affected DSCP" data packets from the RODAEdge to the `RMDEdgeSevereCongestionHandler` class that is presented in section 7.2.4.

7.2.4 "Congestion handling" software class

The solving of a severe congestion situation is assigned to independent software class, "Congestion handling" (see section 6.4.2, Figure 6.4). The used mechanism is as explained in section 3.4.3.3 with the modifications of section 6.5. "Congestion handling" is implemented in the C++ class RMDEdgeSevereCongestionHandler, addressed here as CongestionHandler. Major functionality of the CongestionHandler is in the OTcl code with C++ method to communicate with the RODAEdge class.

The Congestion handler is responsible for the collection of data packets with "encoded DSCP" and "affected DSCP" and for choosing flows to terminate to solve the severe congestion.

Each physical node in the network has one CongestionHandler. Many flows may have as receiver the same physical node. The RODAEdge agents of these flows inform the CongestionHandler for marked or affected data packets. Thus the CongestioHandler collects information for all flows that arrive at the node. The collection of marked packets is managed by the procedure `recv_marked_packet {agent bytes disconnected}` [rmd-lib.tcl]. In the calculation of the total amount of marked bytes the remark proportion N is implemented (see highlighted code in Appendix B, Table B. 5). Another procedure, `recv_affected_packet {agent bytes disconnected}`, collects the affected data packets. If the ingress-egress pair aggregate (see section 6.5.2) is implemented indexing is used to collect the same information but for each source node (Appendix B, Table B. 5, Variant 2).

Each RODAEdge agent that informs the CongestionHandler for marked or affected data packet is recorded in `agent_list [rmd-lib.tcl]`. This information is used later to choose which flows should be stopped and to find the RODAEdge that should be informed to do it.

The collection of data packets is done during one measurement period, see section 4.2.3. A C++ timer, `RMDEdgeSevereCongestionHandlerMeasurementTimer`, keeps the measurement running and when it expires the procedure `measurement_ended {}` is called. The timer is started at the receiving of the first marked or affected data packet for the current measurement period.

`measurement_ended {}` implements the algorithm that was explained in section 4.2.3. In unidirectional operations the flows with biggest reservation that is close to the severe congestion level are chosen first. This is existing implementation and is done by the procedure `get_max { agent_list_name maximum }`. A new approach, see section 6.5.3, for bi-directional reservations is taken where the flow with the smallest reservation, on the path opposite to the overloaded path, is picked first. To achieve this, the implementation can distinguish between both types of reservations and to call the correct procedure (Appendix B, Table B. 6). The procedure choosing the smallest flow is `get_min_reverse` and is presented in Appendix B, Table B. 7.

Note that in bi-directional reservations ingress and egress agents can receive marked packets and to use the procedure. A special separation is made such that each agent uses the appropriate resource reservation sizes (Appendix B, Table B. 8).
The ingress – egress pair aggregate experiments require anther modifications – the loop for priorities to be embedded into a loop per source node. That implies that the flow selection happens for each source node based on the same rules as described above.
When a flow is chosen for termination the procedure `terminate { agent_list }` is called. To implement bi-directional operation when an egress agent wants to stop the flow a NOTIFY message is sent, and when this is requested by ingress agent the flow is stopped directly (Appendix B, Table B. 8). Additionally for the optimization of section 6.5.3 the total stopped bandwidth in the ingress is kept (Appendix B, Table B. 8, highlighted code).
Along with the above modification two new methods are added to the C++ class `RMDEdgeSevereCongestionHandler` for the optimization from section 6.5.3 [rmd-edge.h/cc]. Both methods are called when a NOTIFY message is processed by the ingress node. The first one, `checkBW()`, only checks how many flows the ingress has stopped. The second method, `changeBW(int resources)`, is used only if the NOTIFY message to keep the information on stopped bandwidth up-to-date.

7.2.5 Session binder software class

The "Session binder" software class (Figure 6.4), implemented in the RMDEdgeSessionBinder class [rmd-edge.h/cc], keeps the connection between SESSION IDs when tunneling or bi-directional reservations are used. An instance of the class is attached to each physical edge node.
To understand the class operation the reader should remember that in the simulation model each flow has an ingress-egress pair RODAEdge agents. The ingress agent is attached to one physical node and the egress agent to another. Each agent has an unique agent ID [rmd-edge.h].
The RMDEdgeSessionBinder class uses a hash table with key element the agent ID. Each entry keeps information on two SESSION IDs. In the case of tunneling these are the end-to-end and local SESSION ID and in the case of bi-directional reservations the forward and reverse SESSION ID.

```
struct struct_session_group {
      int sid;
      int local_sid;
} session_group_member;
map<int,struct_session_group>session_map;
```

Each physical node has attached an RMDEdgeSessionBinder instance that is created when the first entry is registered. To add entries the methods `setSID(int& agent, int& session)` and `setLocalSID(int& agent, int& session) are used` [rmd-edge.h/cc]. The first method records an end-to-end (or reverse) SESSION ID in the sid element of the hash table. The second method records the local (or forward) SESSION ID in the local_sid element of the hash table, see above.

When a flow is terminated and its agents are destroyed the method `removeAgent(int& agent)` is called to remove the corresponding entry from the hash table.
Each RODAEdge agent, ingress or egress, knows the identity of its RMDEdgeSessionBinder instance. Each time the agent want to perform SESSION ID check, it uses its agent ID and the method `getSID(int& agent)` to access the recorded entry.

7.3 Support software classes

As mentioned in section 6.4.3 two additional classes are part of the simulation model. Their purpose is to provide information, which is used in the evaluation of the RMD-QOSM performance.
The first class, new to the model, is the ScalabilityHandler class [rmd-edge.h]. By keeping a local counter on the number of installed RODAEdge agents, the class provides statistical information on the number of agents and number of flows in the network. The number of flows can be derived by the number of agents since each flow has a pair RODAEdge agents. The collected information is written to a file – record_file. The file is updated by the method `scal_record(void)`each time a timer, ScalabilityHandlerTimer, expires.
There is only one ScalabiliyHandler instance that is created by calling the OTcl procedure `get-ScalabilityHandler {}` [rmd-lib.tcl]. This is done by the first instance of the SessoinBinder class. Both classes work in close co-operation because the SessionBinder class registers RODAEdge agents and the ScalabilityHandler counts them. When a new entry is registered in a SessionBinder instance the ScalabilityHandler method `addEntry()` increases the local counter. If an entry is removed the method `removeEntry()` is called to decrease the counter.

The second support class is a flow monitor, see section 4.3.6, implemented with the EnhancedFlowMon class [enhanced-flow-mon.h/cc]. This functionality already existed in the base simulation model. An instance of the monitor is created in the main OTcl script [mani.tcl] by the command:

```
$ns at TIME "setup_and_start_link_monitor beginLinkNode endLinkNode"
```

The procedure, see Appendix B, Table B. 9, first creates a monitor on the link that connects beginLinkNode and endLinkNode, second it starts a timer and last it opens new file. The flow monitor collects load statistics. Each time the timer expires the collected data is written to the file.
All messages that pass the link are processed by the flow monitor. By using its own classifiers, PreemptionPriorityHashClassifier [enhance-flow-mon.h], and a build-in ns2 classifier, the flow monitor can distinguish between signaling messages and data packets and between different priority data packets. For the former the fid_ field of the IP header is used and in for the latter the preemption priority from the RMD header. Unlimited number of flow monitors can be created to monitor all links of interest.

8 Simulation Experiments

This chapter presents the simulation experiments for RMD-QOSM that have been performed using the simulation model described in Chapter 6 (simulation model design) and 7 (simulation model implementation). The main goal of this chapter is to provide an insight on the performance behavior of the RMD-QOSM. To achieve this sets of simulation experiments are run where two possible protocol mechanisms are used, the existing one and a proposed optimization. Due to lack of time only severe congestion simulation experiments and scalability experiments have been performed.
The performed severe congestion experiments are:

- **Experiment 1: Dropping of marked packets**: This experiment observes the impact of dropping marked packets on the severe congestion detection and handling solution. This experiment is presented in section 8.2.
- **Experiment 2: Using ingress-egress pair aggregates**: this experiment observes the impact on using the ingress-egress pair aggregates (see Section 6.5.2) on the severe congestion performance. The description of the experiment is given in section 8.3.
- **Experiment 3: Sizes of bi-directional reservations in forward and reverse direction**: this experiment observes the effect of the bi-directional reservation sizes, in forward and reverse directions, on the performance of the severe congestion solutions in both directions. Section 8.4 describes this experiment.
- **Experiment 4: Flow termination based on size**: this experiment observes the impact of flow selection and termination, based on reservation sizes, on the performance of the severe congestion solutions in the forward and in the reverse direction. This experiment is described in section 8.5.
- **Experiment 5: Optimization of the severe congestion mechanism**: this experiment observes the impact of the optimized severe congestion solving mechanism, see section 6.5.3, on the performance of the severe congestion solutions in the forward and in the reverse direction. This experiment is discussed in section 8.6.

The scalability experiment is:

- **Experiment 6: State Scalability**: The final experiment concentrates on the state scalability of the RMD-QOSM protocol when compared to the pure QoS NSLP protocol. Since scalability is broad term, only the number of generated states is used in the comparison. This experiment is described in section 8.7.

Each experiment uses a simulated network topology, presented in the corresponding section, and the implementation of the simulation model from Chapter 7. In each simulated network topology the data flow is presented with line starting at the sender node and ending at the receiver as the data flow direction is indicated by an arrow. In bi-directional scenarios there are two arrows for the two directions. Between each two nodes on the data flow path there is a functional link. Currently unused links do not have an arrow. Two variants of each simulated network topology are presented – before and after a link failure. The link break/failure is introduced to emulate a severe congestion situation. Note that all

simulated network topologies are kept as simple as possible. The goal is to limit the factors that affect the protocol behavior, so that their influence can be easily detected and isolated.
Along with the simulated network topologies, the necessary performance parameters of the simulation model, described in Chapter 7, have to be assigned values. These parameters are common for all simulated scenarios and network topologies and remain the same in all experiments. The parameters are assigned values in section 8.1. Parameters that change are described in the section where the experiment is introduced.
At the end of the first section 8.1 the performance measures used to evaluate the RMD-QOSM behavior are defined.

8.1 Common settings

This section describes the common settings of the performance parameters, section 8.1.1 and the definition of the performance measures, see section 8.1.2.

8.1.1 Performance parameters

This section describes the common settings of the performance parameters.
The common settings can be divided into two groups: setting of the RMD-QOSM module of the design (section 7.2) and settings of the other simulation modules (section 7.1). The settings of the latter are:

- *Link* capacity is 10Mbps with propagation delay of 2ms.
- *dsRED queue*s are used in all simulations just as discussed in section 7.2. Physical queue 1 (signaling messages) has priority 0 (i.e., highest priority) with a size of 44 Kbytes. Physical queue 2 (data packets) receives a lower priority 1 and the default size of virtual queue 1 (marked) is 65 Kbytes and the size of virtual queue 2 (unmarked) is 58 Kbytes. The queue size of the virtual queue used by the marked packets is experimentally found such that no marked packets are dropped during severe congestion. In Experiment 1, the queue size of the marked data packets is varied and the values are specified in the experiment settings.
- In most experiments the CBR flows have data packet size of 40 bytes and a rate of 16 Kbps (50 packets per second). These values are taken to represent closely Voice over IP (VoIP) traffic. In bi-directional scenarios other data rates are also used – 8 Kbps (25 packets/sec), 32 Kbps (100 packets/sec) and 64 Kbps (200 packets/sec).
- *Flow generation and holding time*: Due to the fact that the performance of RMD QOSM when severe congestion occurs as result of link failure, it is assumed that at the time of link failure all flows have been generated. Therefore, a uniform distribution for the flow generation is used, which ensures that all flows are generated during the period from 5 seconds to 35 seconds. The holding time of all flows is considered to be higher than the simulation duration.
- *Simulation duration* is chosen to be 120 sec where the link failure is scheduled to occur at the 100sec. This value makes sure that a network stable state is reached after flow generation. At the link failure the routing founds another route to the destination.

The settings of the RMD-QOSM are:

- *Admission threshold* is 100% (section 7.2.1.2), that is all link capacity can be occupied. This threshold allows worst case scenarios to be observed for severe congestion situations.
- As *severe congestion detection threshold* is chosen 103% of the link capacity and as *restoration threshold* – 100% (section 7.2.1.3).
- The measurement periods used for severe congestion detection and for collection of re-marked packets (section 7.2.4) are 50ms.
- Number of cells in the severe congestion marking – 8 cells, which is experimentally found for the tested scenarios (section 7.2.1.3).
- Remark proportion of N = 2 (section 6.5.1) is used because in simulation the link overload is 100%.
- Bi-directional operations are activated by setting the corresponding "B" flag in the RMD-QOSM PHR and PDR containers to 1 (section 7.2.3).

8.1.2 Performance measures

This section describes the definition of the used performance measures [Jain91].
The used performance measures to evaluate the performance of the RMD-QOSM protocol are:

- Detection and handling time is the time it takes to solve the severe congestion. In other words the time from the link failure (100 sec) until the link utilization drops back to the severe congestion restoration threshold (10 Mbps).
- Dropping probability is the ratio between the number of dropped data packets and the total number of marked packets arriving on the link.
- Average dropping probability has the same definitions as above only the calculations are span and averaged over the first 10 measurement periods (50 msec * 10) after the severe congestion occurrence (at time: 100sec).
- Link load: the traffic load on a link
- Link load after stabilization: this performance measure indicates the load on a link after the severe congestion detection and handling time.

8.2 Experiment 1: Dropping marked packets

The *goal* of this experiment is to observe the impact of dropping marked bytes on the severe congestion detection and handling solution. In this type of experiments it is considered that the reservations are unidirectional and that multiple severe congestion points are occurring on the downstream path from ingresses to an egress. Furthermore it is considered that three types of flow priorities are used: high, medium and low.

Studies on the RMD-QOSM performance for unidirectional reservations with one point severe congestion have been done previously. In reality it is possible that more than one link can become overloaded. This situation is denoted as multiple points severe congestion. If it occurs, queues may fill up and possibly unmarked and marked data packets can be dropped, which may impact the performance of the severe congestion solution. How big the influence is can be answered by performing experiments on the performance of severe congestion solution when the drop of the marked packets is controlled and varied. For these experiments two scenarios are simulated with exchanged level of severe congestion on the two consecutive links.

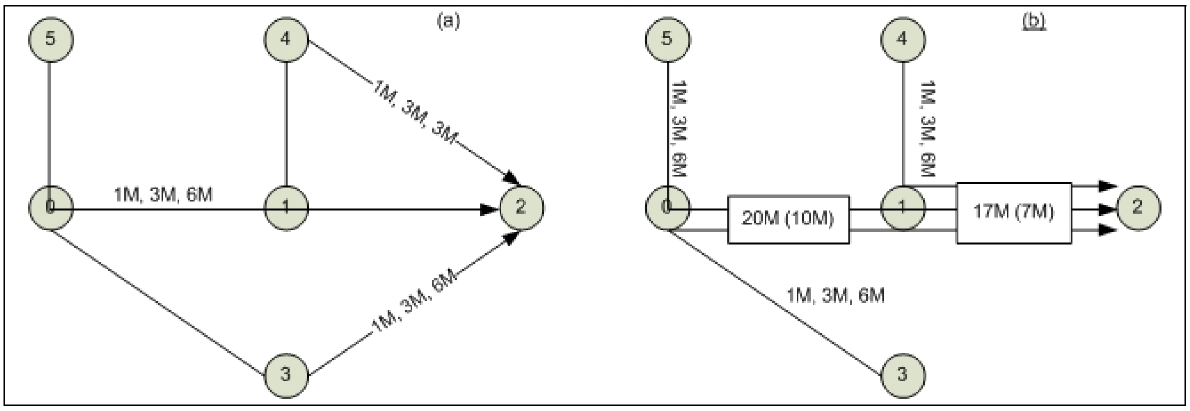


Figure 8.1 Network topology 1

The performed set of experiments is based on the network topology shown in Figure 8.1. This network topology is chosen such that two points severe congestion can be emulated. Figure 8.1 (a) depicts the network topology before the severe congestion event and Figure 8.1 (b) depicts the network topology after the event. The topology has six nodes – two intermediate nodes (0 and 1), three ingresses, emulating the data source nodes (3, 4 and 5), and one egress, emulating the data receiver node (2). All common settings, besides queue size of marked data packets, from section 8.1 are used this set of experiments.
Note that in this set of experiments no ingress/egress pair aggregates are maintained by the edges.
Two simulation scenarios are defined, see Figure 8.1, the one with a severe congestion level higher on the first link (link 0-1) and the other with severe congestion level higher on the second link (link 1-2). For each scenario, a set of two experiments is performed. In the first experiment, denoted as *Not dropped marked,* the queue sizes, given section 8.1, are used, such that no marked packets are dropped. Note however, that unmarked packets may be dropped. This experiment is followed by a second experiment, denoted as *Dropped marked.* For it the queue size for marked packets is reduced to 20 Kbytes such that marked packets are dropped.

The shown figures, Figure 8.2 up to Figure 8.9, depict the traffic loads generated by high, medium and low priority flows that are passing through a link, versus the simulation time. Figure 8.10 shows the average dropping probability versus the queue size of the marked packets. Figure 8.11 shows the detection and handling time performance measure versus the average dropping probability values, depicted in Figure 8.10.

8.2.1 Higher severe congestion level on the first link

To achieve a higher level of severe congestion on the first link (Figure 8.1 (a)) the total traffic generated by each data source is as presented in Table 8-1.

Table 8-1 Flow generation, higher congestion on first link

Source – Destination	Total load	High priority flows	Medium priority flows	Low priority flows
3 – 2	10 Mbps (9.984[4] Mbps)	1 Mbps (0.992 Mbps)	3 Mbps (2.992 Mbps)	6 Mbps
4 – 2	7 Mbps (6,976 Mbps)	1 Mbps	3 Mbps	3 Mbps
5 – 2	10 Mbps	1 Mbps	3 Mbps	6 Mbps

At the time of the link failure (at 100sec) link 3-2 and link 4-2 are considered to be dropped, which causes re-routing of flows, see Figure 8.1 (b). Flows from source 3 are re-routed via path 3-0-1-2. Link 0-1 gets total a load of 20 Mbps, corresponding to 100% severe congestion. With link capacity of 10 Mbps half of the 20 Mbps are dropped. The 10 Mbps that passed through link 0-1 arrive at node 1. The re-routed traffic from source 4, i.e., 7 Mbps, arrives also at node 1, which results in total load on link 1-2 of 17 Mbps, thus a 70% severe congestion occurs.
Each severe congested link (Figure 8.1) calculates its own level of severe congestion and marks the appropriate number of data packets to notify the egress node (algorithm from section 3.4.3.2). The egress node 2 (Figure 8.1), based on the collected data packets (section 3.4.3.3), starts to stop flows until the link utilization drops to the desired value (restoration threshold = 100%).
This is a normal operation when no marked data packets are dropped. On Figure 8.2 the load on the second severe congested link (link 1-2) is shown. As it can be seen the aggregate link load is 10 Mbps after the severe congestion is solved. The detection and handling time, equal to 0.55 sec, is acceptably small. Furthermore, all high priority flows are passed, and the rest of the link capacity is used for medium priority flows. All low priority flows are terminated because there is no free link capacity to transport them.
In Figure 8.3, the graph of the first congested link (0-1) shows that the link utilization drops below 100%. The reason of this is that data packets from sources 3 and 5 contribute to the severe congestion on link 0-1 and on link 1-2. As result their data packets are marked by both links. Eventually the flows stopped from 3 and 5 are enough to solve the congestion on both links but it also means the utilization on link 0-1 is below 100%. That would not happen if only flows from source 4 are terminated to solve the severe congestion on the second link. Note however that it would cause high priority flows to be stopped and

[4] The number in braces is the exact rate. It is not possible to generate all rates as exact since these are aggregated total rates consisting of many 16kbps flows. For example, 1 Mbps can be achieved by generating 62.5 flows. The problem is that only an integer number of flows can be simulated. As result 62 flows are generated with a total aggregated rate of 992 kbps. Approximated aggregate values (1Mbps) are used because they are simple and more comprehensible in discussion. Further each time an aggregate rate is chosen then two values are shown, is the approximated value and the other one that is the real value given between brackets. When the approximated value and the real value are equal then only one value is given.

is not acceptable. The graphs on the aggregate load versus simulation time of all other links, of the network topology in Figure 8.1, can be found in Appendix C.1.
In the second experiment, associated with the same simulation scenario, small queue sizes are used, i.e., Dropped marked. In this experiment drop of the marked packets occurs in the second link. This experiment is performed to investigate if the performance of the severe congestion solution is affected when marked packets are dropped. Even if marked packets are dropped the egress receives part of them. Due to this fact the egress node terminates an amount of flows but not all flows necessary to solve the congestion. Therefore, an overload situation remains in the network and the whole severe congestion detection and handling is repeated. The severe congestion is solved but after a longer detection and handling time, i.e., 1.25 sec, see Figure 8.4. The drop of marked data packets leads to a delay in the detection and handling time of about 0.7 sec and more fluctuations during the severe congestion solving process. Nevertheless the flow priority principle of terminating first low priority flows before higher priorities, is maintained. Additional graphs associated with this experiment are given in Appendix C.1.

8.2.2 Higher severe congestion level on the second link

In the second simulation scenario the second link, i.e., link 1-2 (Figure 8.1) is the one that has a higher level of congestion (100%), while the first is less congested (70%). The initial simulation settings for this scenario are given in Table 8-2.
These load values lead to severe congestion levels of 70% at link 0-1 and 100% at link 1-2 at link, which levels correspond to total load arriving at link 0-1 17 Mbps (10 Mbps from source 3 and 7 Mbps from source 5) and at link 1-2 - 20Mbps (10 Mbps from source 4 and 10 Mbps from link 0-1).
The initial utilization of links 0-1 and 1-2 is 7 Mbps corresponding to the aggregate traffic load from source 5 (Figure 8.6). The peak going to 10 Mbps after link failure (at 100 sec.), see Figure 8.7, is due to re-routed flows coming from source 3. The reason for the load drop under the link capacity after the detection and handling time is as described in section 8.1.1 – data packets from 3 and 5 participate in both severe congestions.
In the first experiment, i.e., *Not dropped marked*, see Figure 8.6, the detection and handling time on link 1-2 is again 0.55 sec and the flow priority principle is kept.

In the second experiment, i.e., *Dropped marked*, the drop of marked packets prologues the *detection and handling time* to approx. 1.5 seconds. Along with this the *link load after stabilization* in the *Dropped marked* experiment is approx. 1 Mbps lower than the value of the same performance measure calculated in the *Not dropped marked* experiment, see Figure 8.8 and Figure 8.9. The undershoot in the *Dropped marked* experiment, affects flows sent from sources 3 and 5 (Figure 8.9) when compared with the *Not dropped marked* case (Figure 8.7). Similar observations can be derived from the graphs shown in Appendix C.1.

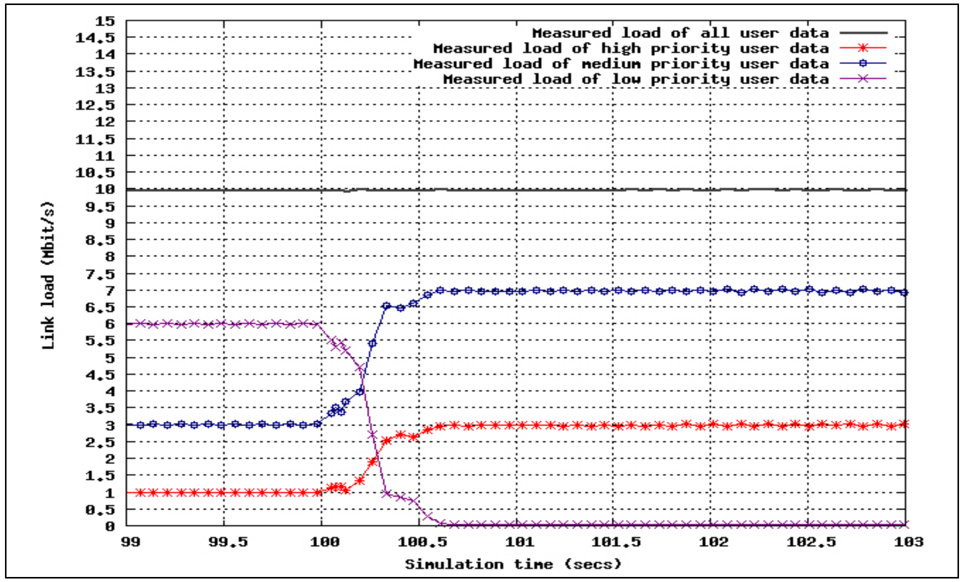


Figure 8.2 Link 1-2: Higher congestion on first link (0-1), Not dropped marked

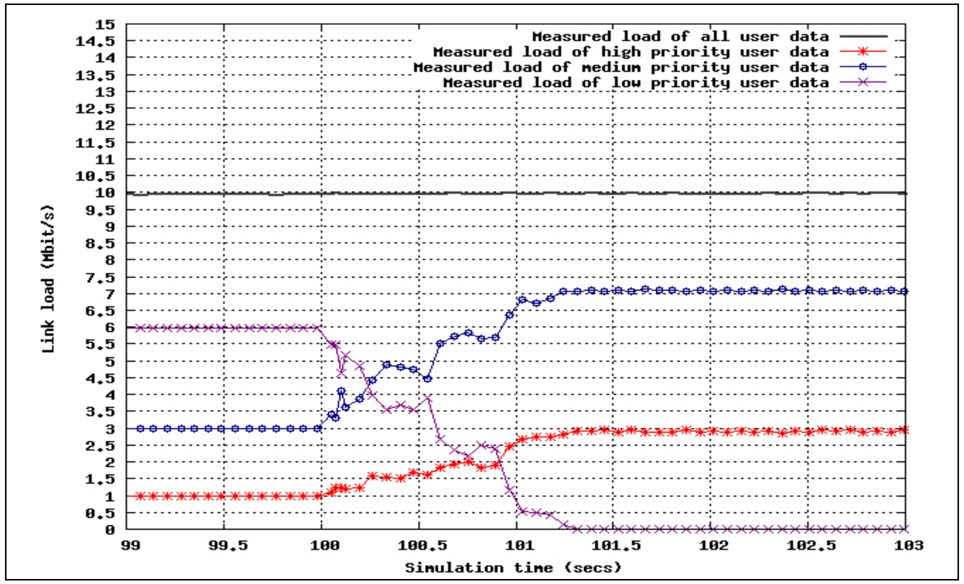


Figure 8.4 Link 1-2, Higher congestion on first link (0-1), Dropped marked

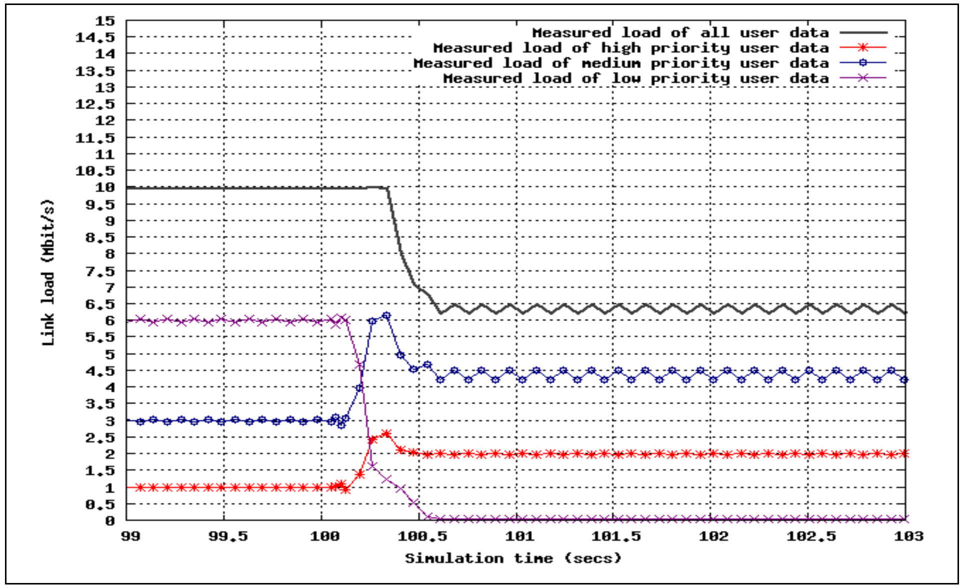


Figure 8.3 Link 0-1, Higher congestion on first link (0-1), Not dropped marked

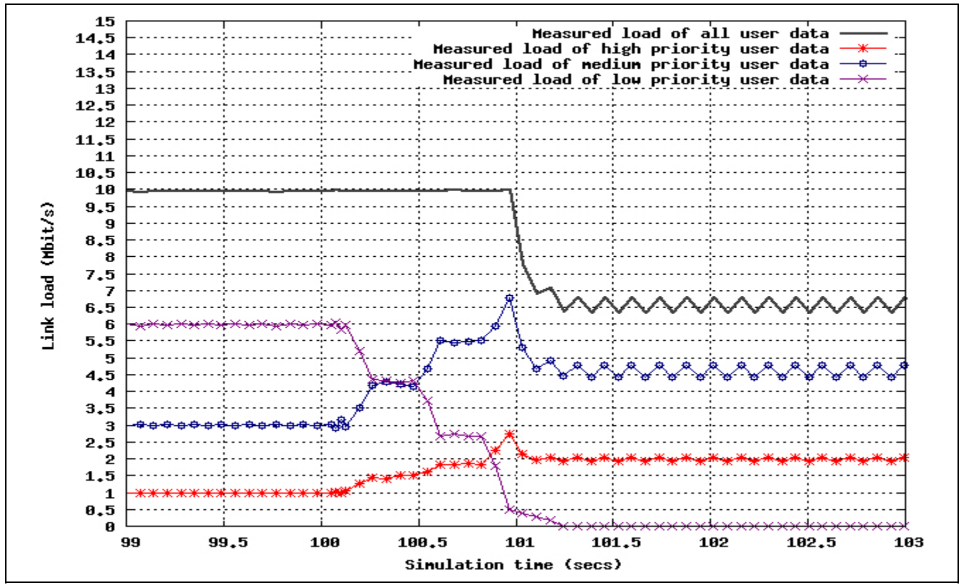


Figure 8.5 Link 0-1, Higher congestion on first link (0-1), Dropped marked

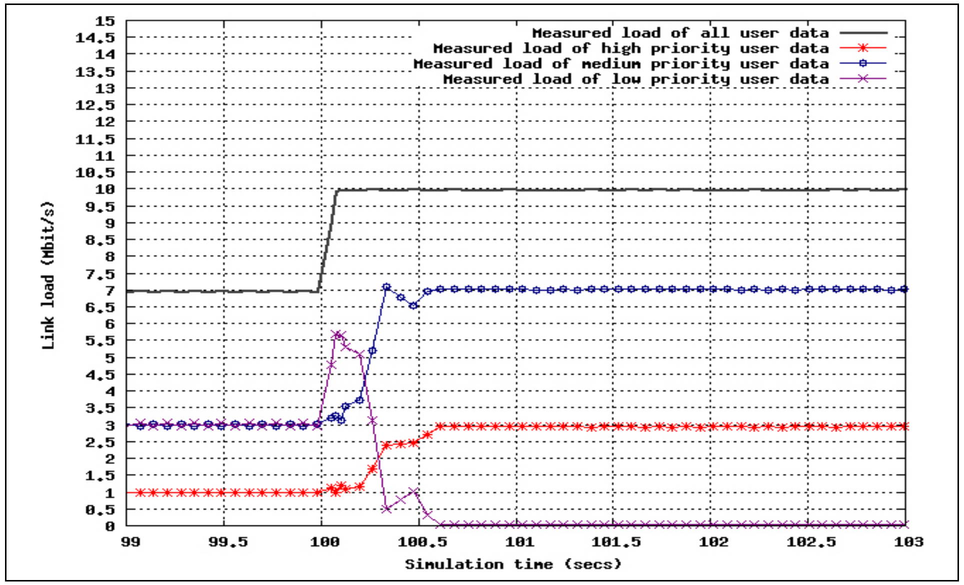


Figure 8.6 Link 1-2: Higher congestion on second link (1-2), Not dropped marked

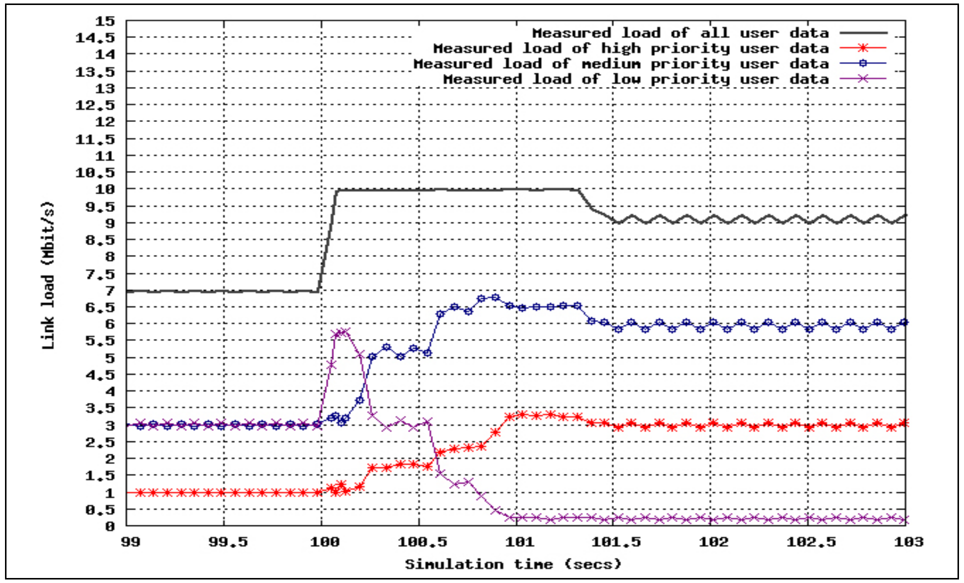


Figure 8.8 Link 1-2, Higher congestion on second link (1-2), Dropped marked

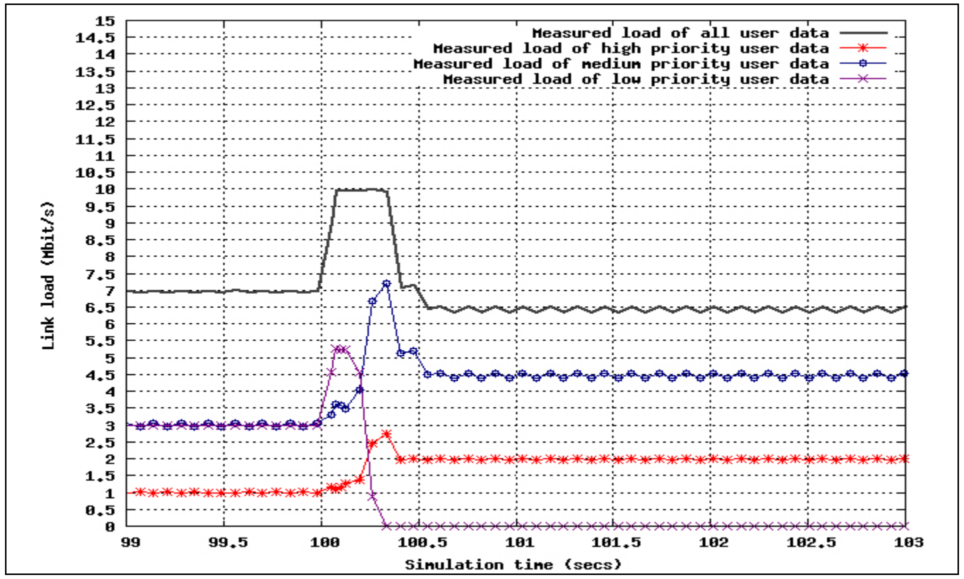


Figure 8.7 Link 0-1: Higher congestion on second link (1-2), Not Dropped marked

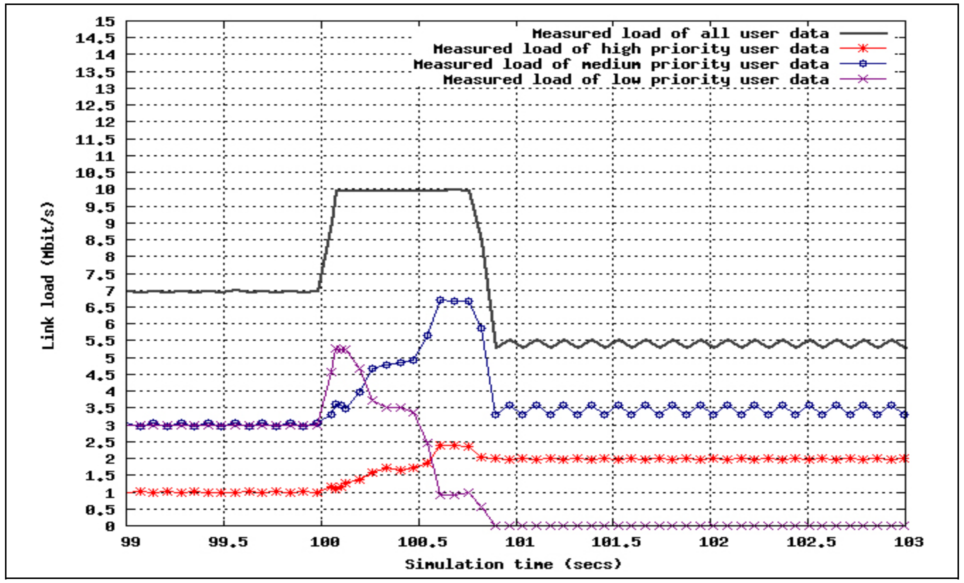


Figure 8.9 Link 0-1, Higher congestion on the second link (1-2), Dropped marked

Table 8-2 Flow generation, higher congestion on second link

Source – Destination	Total load	High priority flows	Medium priority flows	Low priority flows
3 – 2	10 Mbps	1 Mbps	3 Mbps	6 Mbps
4 – 2	10 Mbps	1 Mbps	3 Mbps	6 Mbps
5 – 2	7 Mbps	1 Mbps	3 Mbps	3 Mbps

8.2.3 Average dropping probability calculated during “Higher congestion on the first link” scenario

The “higher congestion on the first link” scenario described in section 8.1.2 is used to evaluate the average dropping probability (ADP) of marked data packets. For a relatively trustworthy value of the average dropping probability a confidence interval is calculated by using the method of the independent replicas. In this method the same experiment is repeated multiple times but with different RNG seeds. The resulting samples can be used to calculate a mean value of the ADP and the standard deviation. They will allow a confidence interval to be calculated. The confidence interval is an area around the mean value where an ADP of random sample can be found with some level of certainty, a confidence level [Jain91]. In other words Confidence interval of 95% means that the ADP of a sample will be with 95% percent probability found in the confidence interval area.

In the simulation experiments the queue size is varied from 38 Kbytes to 60 Kbytes as twelve different values are taken. Each queue size corresponds to an ADP value. Due to time constrains a confidence interval is found only with ten different seeds for three queue sizes. In the other nine cases experiments with only four seeds are done. The used queue sizes followed by the number of seeds used with them are given in Table 8-3.

For each performed experiment the ADP and the detection and handling time are recorded. The collected information allows two dependencies to be observed - between the queue size for marked data packets and the average dropping probability, and between the average dropping probability and the detection and handling time. The first dependency is depicted in the graph shown in Figure 8.10. In the graph the three calculated confidence intervals are presented with the mean value and the lower and upper boundary.

A two sided 95% confidence intervals are found using the central limit theorem [Jain91]. A special case of the theorem has to be used when the number of samples is small. In such situations the boundaries of the confidence interval have student distribution, which affects the value of a coefficient that is used in the theorem.

What is more the theorem can be applied only if the samples are normally distributed [Jain91]. It is proved that the sample means of independent observations have such distribution. Since the method of the independent replicas is used it can be assumed that the sample means from the experiments are distributed normally and the theorem can be applied.

Table 8-3 Used queue size for marked data packets in Kbytes

38 (10)	40 (4)	42 (4)	44 (4)	46 (4)	**48 (10)**
50 (4)	52 (4)	54 (4)	**56 (10)**	58 (4)	60 (4)

The confidence interval is calculated by the formula:

$$(\bar{x} \pm t_{1-a/2,\, n-1} s/\sqrt{n})$$

$\bar{x}$ is the mean of the samples, s is the standard deviation, n is the number of the samples and t is the coefficient coming from the student distribution, as a = 0.05 corresponds to confidence level of 95%.

For nine queue sizes (Table 8-3) simulation experiments with only four seeds were run. The average dropping probability of each sample is plotted in Figure 8.10. Even if the confidence interval is not calculated it can be seen that the sample values for one queue size lay close to each other. It can be therefore expected that the confidence interval in these cases will resemble the already calculated ones. Note that the biggest queue size causes no drop and it is plotted on the horizontal axis.

The general conclusion of the observations is the when the queue size is decreasing that leads to increase in the average dropping probability. For the calculated confidence intervals it can be concluded that the values are acceptably good. To achieve even higher level of authenticity more experiments should be performed.

The detection and handling time increases when the queues size is decreased. Based on this conclusion and on the above made observations it can be expected that higher dropping probability causes longer detection and handling time. This expectation is confirmed by the results shown in Figure 8.11. The upper group of graphs are for the second, more congested link, and the lower group – for the first, less congested link. It should be mentioned that the mechanism used to observe the detection and handling time is not very precise, which causes the sharp drops in the graphs. These observations have more or less supportive, illustrative role but a better mechanism should be implemented if a precise knowledge on the detection and handling time is required.

Conclusions:

- The drop of marked data packets causes longer detection and handling time, which means that the severe congestion situation, remains for a longer time in the network.
- The drop of marked data packets in some situations can cause unexpected drop in the link load, which is not desired since more flows are terminated than necessary.
- The flows are in all cases terminated taking into account their flow priorities.
- The smaller the queue size for marked data packets the bigger the average dropping probability is. The calculated confidence intervals area acceptably small.
- The detection and handling time increases with the increase of the average dropping probability and the decrease of the queue size for marked data packets.

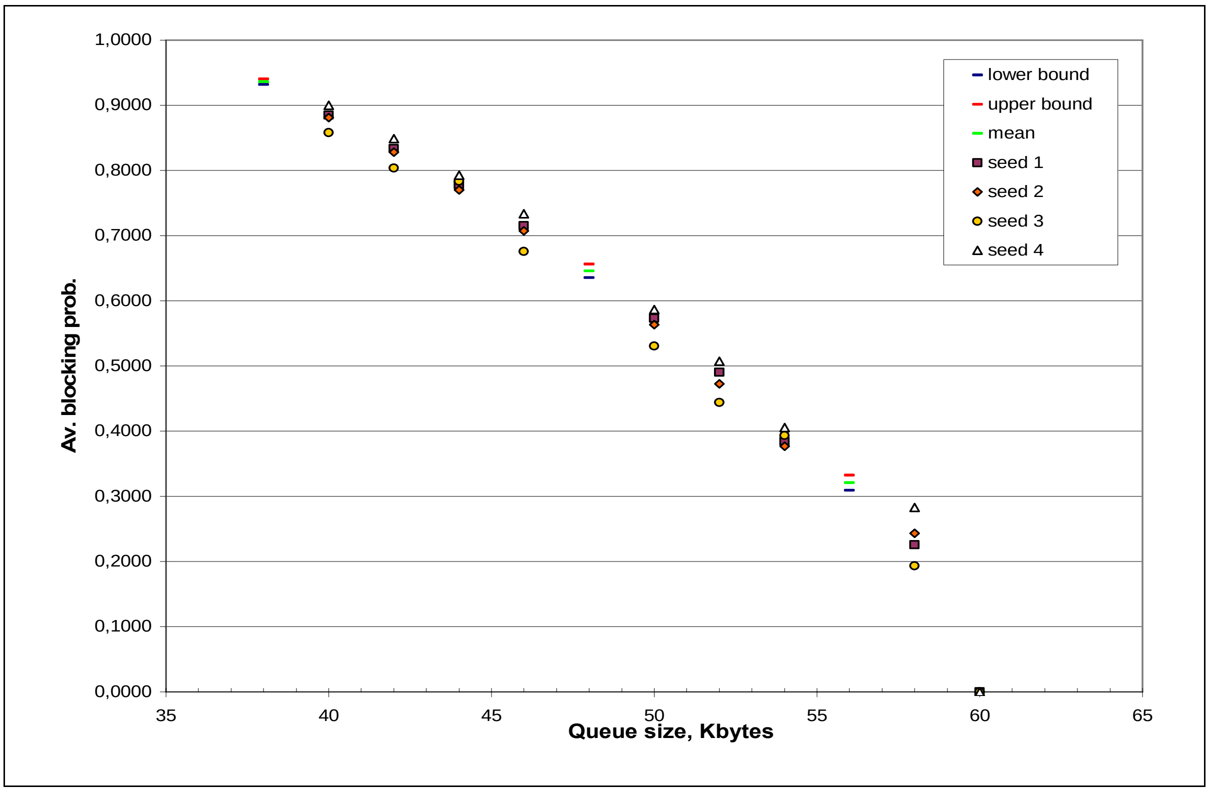


Figure 8.10 Average dropping probability vs. queue size

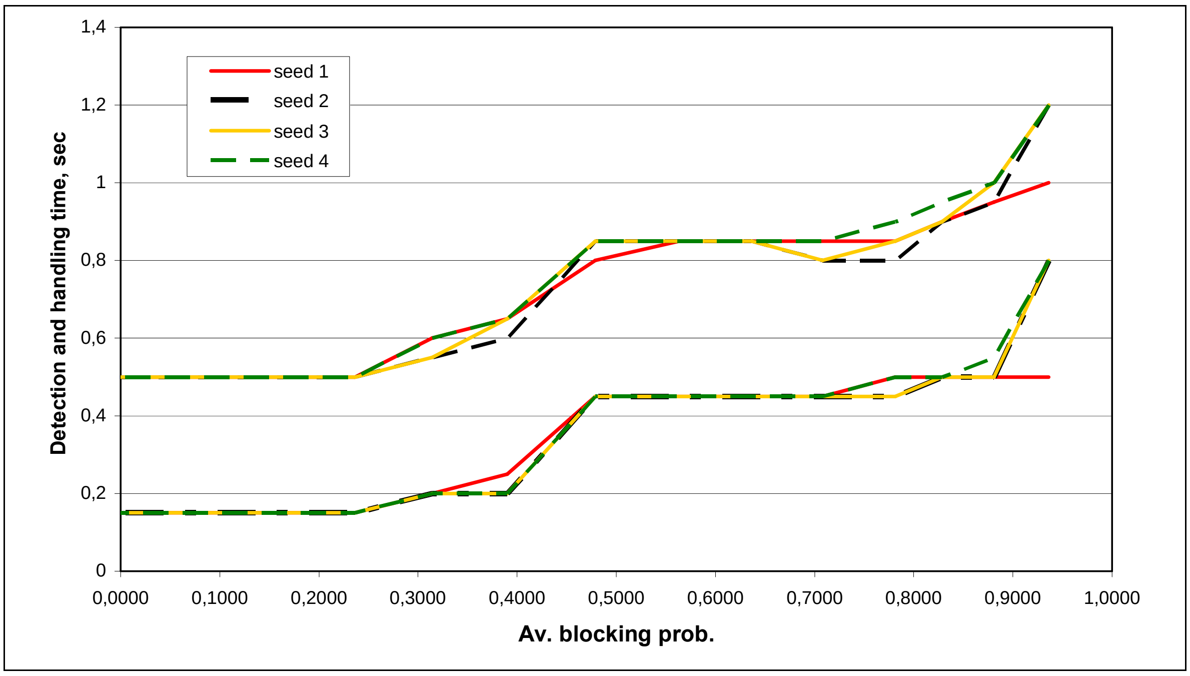


Figure 8.11Detection and handling time vs. average dropping probability

8.3 Experiment 2: Using ingress-egress pair aggregates

The *goal* of this experiment is to observe the impact of using the ingress-egress pair aggregates (see Section 6.5.2) on the severe congestion performance. It is considered that flows with different priorities, high, medium and low pass through different ingresses and through the same egress. In this experiment only unidirectional reservations are considered. Two types of experiments are performed. In one type the ingress-egress pair aggregate is used while in the other set this pair is omitted.

The simulated network topology is given in Figure 8.12. Nodes 1 and 3 are interior nodes, nodes number 0 and 2 are ingresses that emulate also the sources of the data traffic, and node number 4 is the egress that emulates the data receiver. In this scenario it is considered that only one link on the ingress-egress path can be severe congested. Note that the conclusions of this set of experiments hold also for the situation that multiple point severe congestion occurs because the mechanism affects only the work of the egress node.
The aggregate traffic rate for the experiment is presented in Table 8-4.
At link failure time (at 100 sec.) link 0-1 is dropped and data flows from source 0 re-routed via 3 such that at link 3-4 a total load of 17 Mbps arrives, corresponding to 70% level of severe congestion.
Two simulation experiments are performed. In the first one the ingress-egress pair aggregates, see section 6.5.2, are not used, while in the second they are used. An estimation of the expected terminated bandwidth in each case is made. The severe congestion is 70% corresponding to 17 Mbps is arriving on 10 Mbps link, which means that a total rate of 7 Mbps have to be stopped.
Expected aggregate load termination in the case when no ingress-egress pair aggregates are used:

- Low priority flows – 2 Mbps from source 0 and 2 Mbps from source 2 give aggregate rate of 4 Mbps. This is not enough and the termination mechanism moves to:
- Medium priority flows – 2 Mbps from source 0 and 1 Mbps from source 2 generate an aggregate rate of 3 Mbps, which summed with the terminated low priority flows results in 7 Mbps. The severe congestion is solved and no high priority flows have to be terminated.

The expected aggregate load termination for ingress-egress pair aggregates is:

- Source 0 (7 Mbps) generates 0.7 of the rate of source 2 (10 Mbps) and therefore source 0 contributes to the congestion with 2.88[5] Mbps, while source 2 contributes with 4.12 Mbps. The same aggregated load has to be stopped from each source.
- Source 0 stops 2 Mbps low priority flows and 0.88 Mbps medium priority flows, which result in total rate of 2.88 Mbps. 3 Mbps high and 1.12 Mbps medium priority flows are left from source 0.

[5] The proportion of the overload part source 0: source 2 is 1:0.7 and the sum of the sum of the parts is the total overload of 7 Mbps. The exact values of the overload per source are calculated from this proportion.

Table 8-4 Flow sizes, experiment 2

Source – Destination	Total load	High priority flows	Medium priority flows	Low priority flows
0 – 4	7 Mbps	3 Mbps	2 Mbps	2 Mbps
2 – 4	10 Mbps	7 Mbps	1 Mbps	2 Mbps

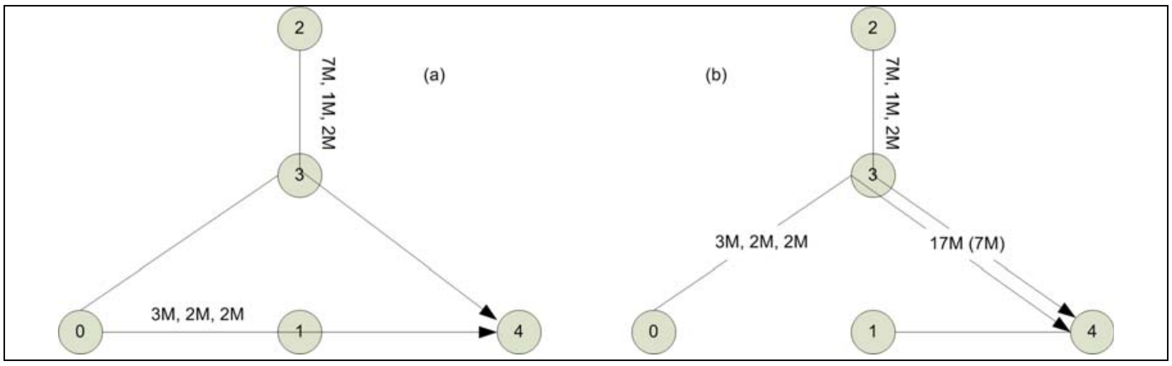


Figure 8.12 Network topology 2

- Source 2 terminates 2 Mbps low priority flows, 1 Mbps medium priority flows and 1.12 Mbps high priority flows, the sum of which is 4.12 Mbps. 5.88 Mbps high priority flows are kept.

From the analytical calculations seems that the use of ingress-egress pair aggregate causes disturbance in the priority principle for flow termination. High priority flows from one source are stopped, while lower priority flows from another source are still passed. On the other hand the mechanism without the ingress-egress pair aggregate always keeps the flow priority.

The described scenarios are also simulated. Figure 8.14 shows the link load vs. simulation time for the situation that the ingress-egress pair aggregate is not used, i.e., *No ingress – egress pair*. The graph in Figure 8.15 shows the link load vs. simulation time when an ingress-egress pair is used, i.e., *ingress-egress pair*. Both graphs are for the severe congested link (3-4). The graphs for all other links are shown in Appendix C.2.

Both graphs, see Figure 8.14 and Figure 8.15, show no difference between the expected results and the simulation results. No high priority flows are terminated in the case of *no ingress –egress pair* and for *ingress-egress pair* high priorities more than 1 Mbps are stopped. Due to the slight fluctuations in the load of high priority flows it is hard to say whether this would be 1.12 Mbps but the author trusts this to be the case.

The detection and handling time in both situations is 0.25 sec.

<u>**Conclusions:**</u>

- The mechanism without ingress-egress pair aggregate supports good handling of the flow priority principle, i.e., lower priority flows are terminated before terminating higher priority flows.
- The mechanism with ingress-egress pair aggregate does not strictly maintain the flow priority principle. Thus it is less accurate in terminating lower priority flows before terminating higher priority flows.
- Both mechanisms have the same detection and handling times.

8.4 Experiment 3: Sizes of bi-directional reservations in forward and reverse direction

The *goal* of this experiment is to observe the impact of the bi-directional reservation sizes on the forward and reverse directions, on severe congestion solutions in the forward and in the reverse direction. In this type of experiments it is considered that the reservations are bi-directional and that one severe congestion point is occurring on either the forward direction or on the reverse direction. Furthermore three types of flow priorities are used: high, medium and low.
In this type of experiments no ingress-egress pair aggregates are used. The used bi-directional flow termination mechanism is described in section 6.5.3. In the experiments the link load is observed when different combinations of reservation sizes (forward and reverse direction) are applied.

All experiments for bi-directional reservations use the same network topology given in Figure 8.13. All simulation settings specified in section 8.1 are kept only the traffic load generation is modified. The effect of the used new termination mechanism, which stops flows/sessions with the smallest reverse reservation size, can be shown only when the reservation sizes in forward and reverse directions are different. Therefore a mix of flows sizes is used, which particular values are specified for each experiment.
In Figure 8.13 nodes number 1 and 2 are interior nodes. Nodes 0 and 3 are ingress edges and emulate the sources for the forward direction, and at the same are data receivers for the reverse direction. The nodes 4 and 5 are the egresses that are emulating the data receivers for the forward direction and data sources for the data sent on the reverse path. The combination of sources and receivers makes possible to be monitored the load coming from each data traffic source during the simulation time.
In the forward direction source 0 sends traffic to node 5 and source 3 sends traffic to node 4. In the reverse direction source 5 sends traffic to node 0 and source 4 sends traffic to node 3. At link failure time (at 100 sec.) link 2-3 breaks, see Figure 8.13 (b) and flows from source 3 are re-routed via path 1-2 and flows from source 4 via path 2-1. As result severe congestion situation occurs on link 1-2. The direction of the severe congestion depends on the used flow sizes.
In experiments 3 and 4 mix of three different flow rates is used where the rates can be 8, 16, 32 and 64 Kbps. In the direction, where severe congestion is to be simulated, combination of 16, 32 and 64 Kbps rates is used. When no severe congestion is to happen on the path rates of 8, 16 and 32 Kbps are chosen. Such rates are chosen because severe congestion only in one direction is of interest. Since each flow has forward and reverse reservation their sizes should allow after the flows are re-routed the one path to be overloaded but the other not.
Different rates mean also that different combinations between the forward and reverse reservation can be used. When the flows with highest forward reservation have also the highest reverse reservation this is referred to as *big-big flows* and when the highest forward reservation have the smallest reverse – as *big-small flows*.

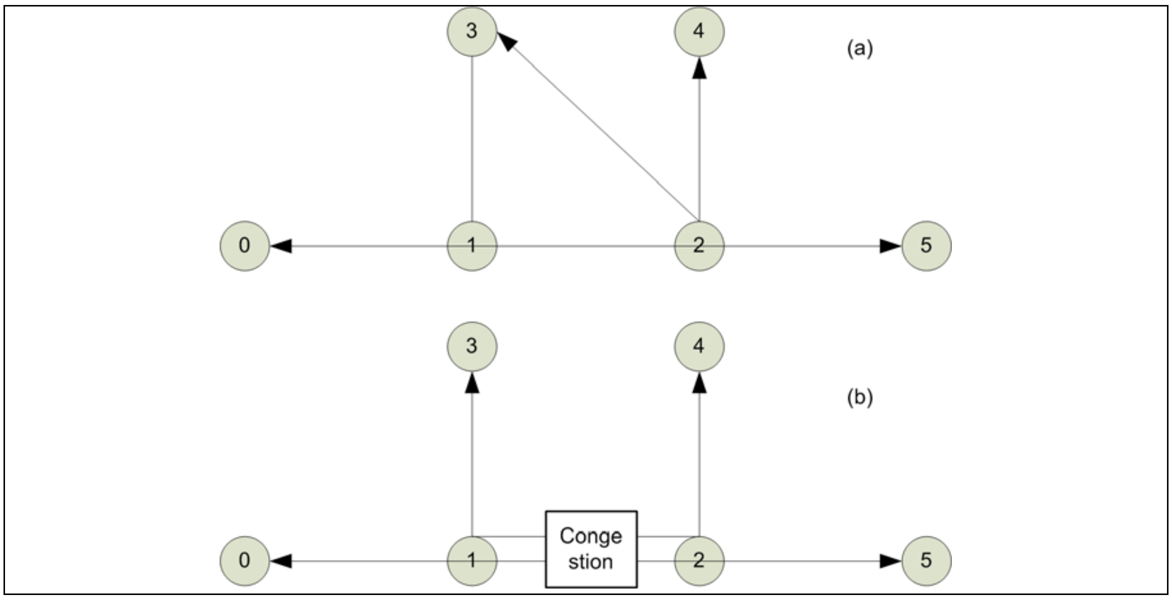


Figure 8.13 Network topology 3

Table 8-5 Flow sizes, experiments 3, 4 and 5

Source – Destination	Total load	High priority flows	Medium priority flows	Low priority flows	Used rates, Kbps	Used rate combination
Experiment 3 and 4 , severe congestion in forward direction						
0 – 5, 3 – 4	10 Mbps	1 Mbps	3 Mbps	6 Mbps	16,32,64	big-big, big-small
5 – 0, 4 – 3	5 Mbps	0.5 Mbps	1.5 Mbps	3 Mbps	8,16,32	
Experiment 3, severe congestion in reverse direction						
0 – 5, 3 – 4	5 Mbps	0.5 Mbps	1.5 Mbps	3 Mbps	8,16,32	big-big, big-small
5 – 0, 4 – 3	10 Mbps	1 Mbps	3 Mbps	6 Mbps	16,32,64	
Experiment 5, severe congestion in both directions						
0 – 5, 3 – 4 5 – 0, 4 – 3	10 Mbps	1 Mbps	3 Mbps	6 Mbps	16	none

For experiment 3 a severe congestion either on the forward or on the reverse path is simulated and experiment 4 observed only severe congestion on the forward path. In all these cases two sets of simulation experiments are discussed, one with *big-big flows* and one with *big-small flows*. In experiment 5 only 16 Kbps flows are used with enough aggregated rate to cause overload in both directions of the data transfer. The aggregate rates for each experiment are given in Table 8-5.

8.4.1 Severe congestion on the forward path

Two simulation experiments are run, one with *big-big flows* and one with *big-small flows* as the rates are as defined in Table 8-5.

The link load on the forward path of the overloaded link 1-2 in both experiments (*big-big flows* and *big-small flows*) is the same and the load for *big-small flows* case is presented in Figure 8.16. After the congestion is solved the link utilization is still 100% but the

proportion of different priority groups is re-arranged to pass all high priority flows and as much as possible the medium priority flows. The detection and handling time is 0.25 sec.
The detection and handling time is the same – 0.25 sec and the flow priority principle is kept. However the reverse path link load after stabilization for both simulation experiments differs. In both graphs after the link failure (at 100 sec) the link load initially raises to 10 Mbps due to the re-routed flows from the reverse path source 4, see Figure 8.18. For the *big-big flows* experiment, see Figure 8.18, termination of the half of the bandwidth on the forward path results in a termination of also the half of the bandwidth on the reverse direction. The reason is the forward – reverse reservations proportion equal to the ratio of 2:1. To solve the congestion 10 Mbps have to be kept on the forward path, which are associated with a reservation of 5 Mbps on the reverse path. All the other graphs associated with the *big-big flows* experiment are presented in Appendix C.3.
In the big-*small flows* experiment the link load on the reverse path does not drop to 50% but it stays above it, see Figure 8.19, because the proportion of forward – reverse bandwidth is different for each of the flow size combinations. The flow termination starts with the smallest reverse bandwidth, which is in this case the biggest forward bandwidth. As result, the congestion is solved by stopping fewer flows than in the *big-big flows* experiment and after the detection and handling time, the flows with the biggest reverse reservation are still kept in the network.
The reader attention might be drawn to one peculiar drop in the total load that is more visible in Figure 8.19. If the graphs of the signaling load are consulted (Figure 8.22, Figure 8.23) the explanation is clear. RMD-QOSM uses in-band signaling and the signaling packets have the highest priority. When the NOTIFY messages are sent they use part of the link capacity and only the left over capacity is used for data transfer. If the values of the drops[6] are compared with the size of the signaling it can be seen that they are the same. Additionally, the size of signaling for the case *big-big flows* is bigger (Figure 8.23), which proves that more flows have to be stopped than in comparison with the case of *big-small flows*.

8.4.2 Severe congestion on the reverse path

Again two simulation experiments are performed, one for *big-big flows* and another one for *big-small flows*. The rates are given in Table 8-5. All observations made in section 8.4.1 are valid with the exception that the reverse link has now the total link utilization of 100% (Figure 8.17). The forward link utilization is less than 100% and the same influence of the combination of flow sizes is observed. The forward link occupancy drops to 50% for *big-big flows* generation (Figure 8.20). Again the link utilization is considerably higher if the *big-small flows* are used (Figure 8.21), where the explanation is identical to the one given in section 8.4.1.
Difference with severe congestion on the forward path is the missing drop in link load due to signaling messages. The ingress node, when it receives marked data packets, stops the flows without sending NOTIFY messages, which explains the lack of drop.

[6] That is from the 10 Mbps value until the drop on the total data load graph.

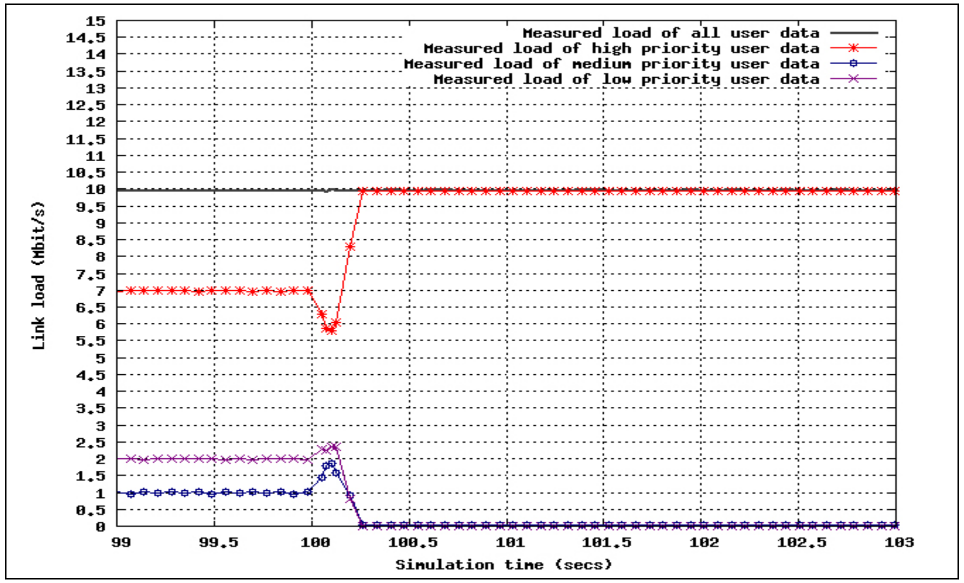


Figure 8.14 Link 3-4, No ingress –egress pair

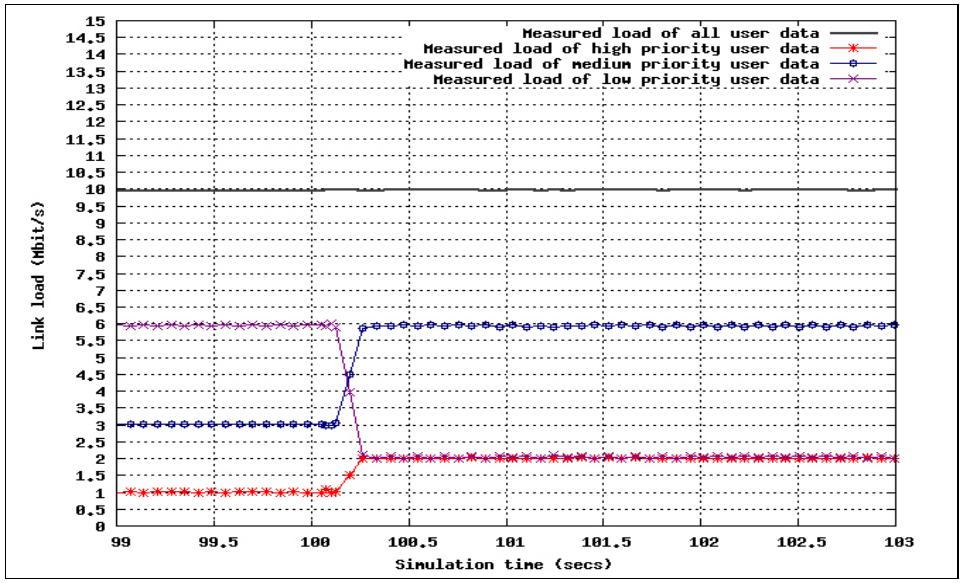


Figure 8.16 Link 1-2, forward path, big-small flows

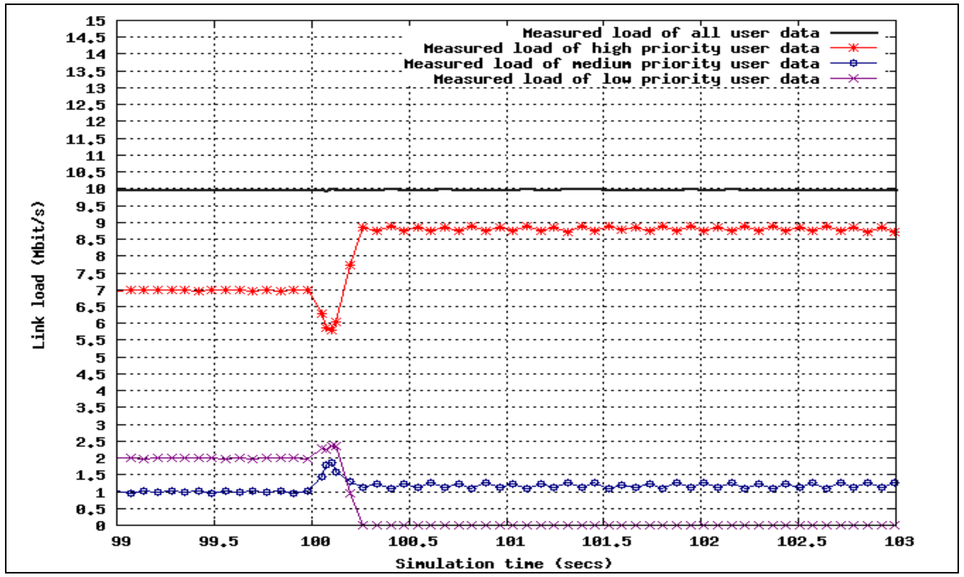


Figure 8.15 Link 3-4, ingress-egress pair

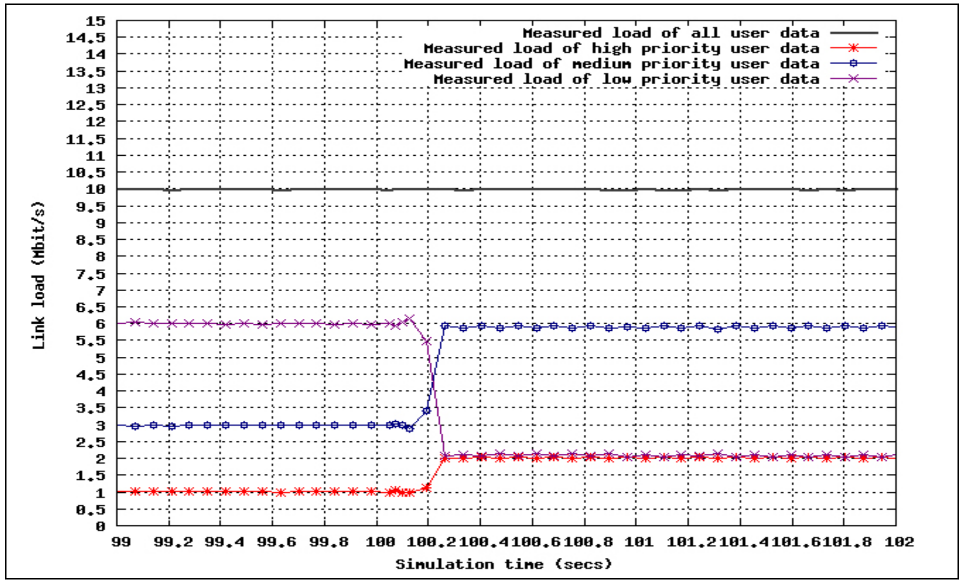


Figure 8.17 Link 2-1- reverse path, big-small flows

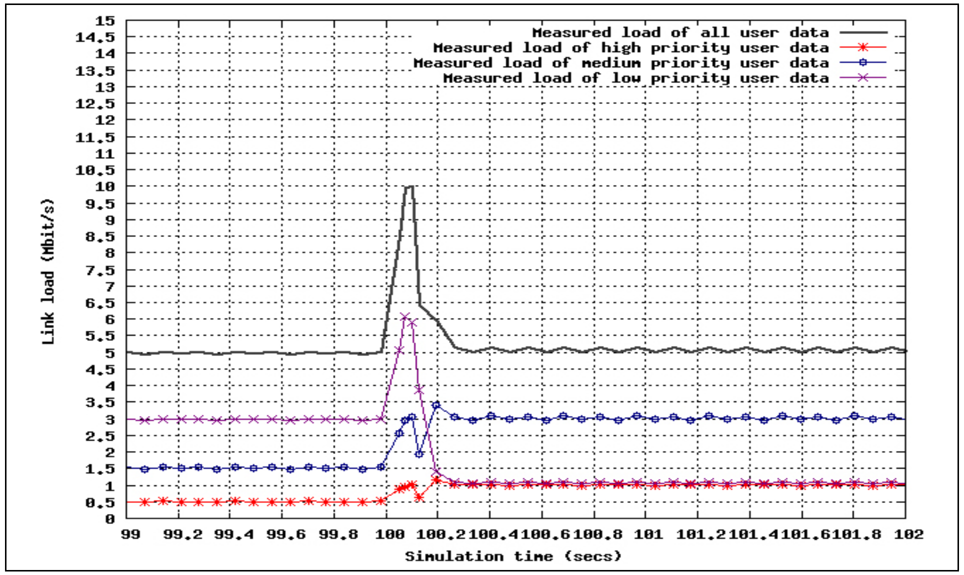


Figure 8.18 Link 2-1, Reverse path, big-big flows

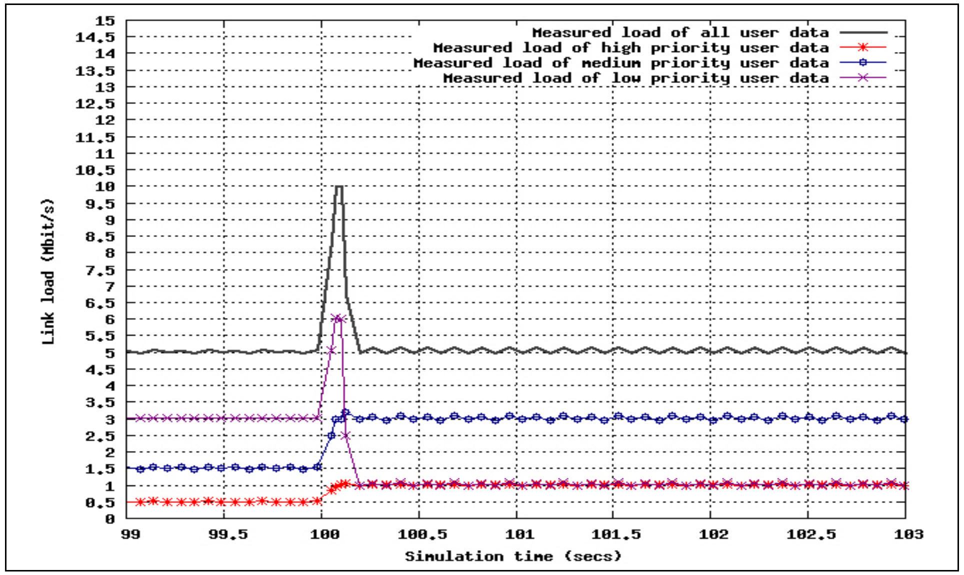


Figure 8.20 Link 1-2: Forward path, big-big flows

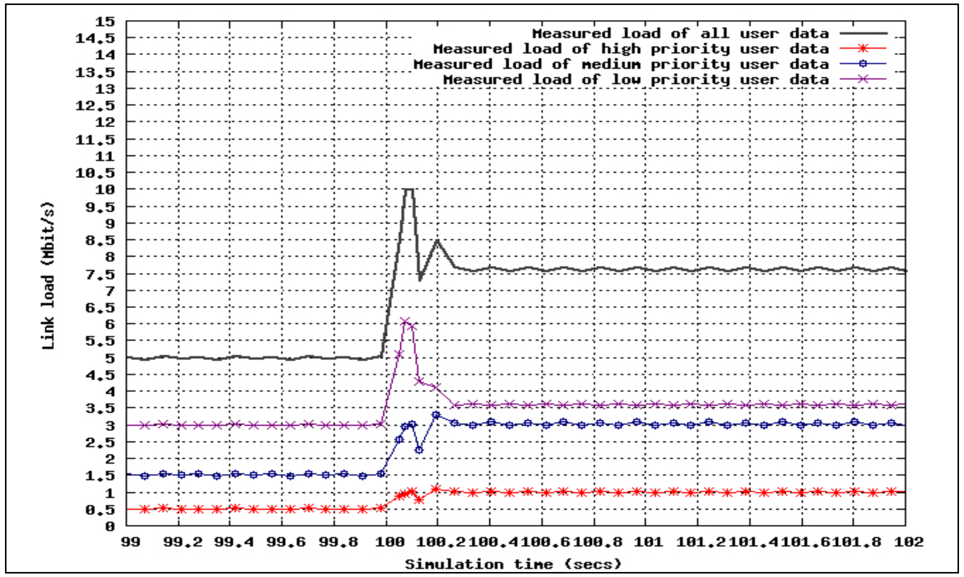


Figure 8.19 Link 2-1, Reverse path, big-small flows

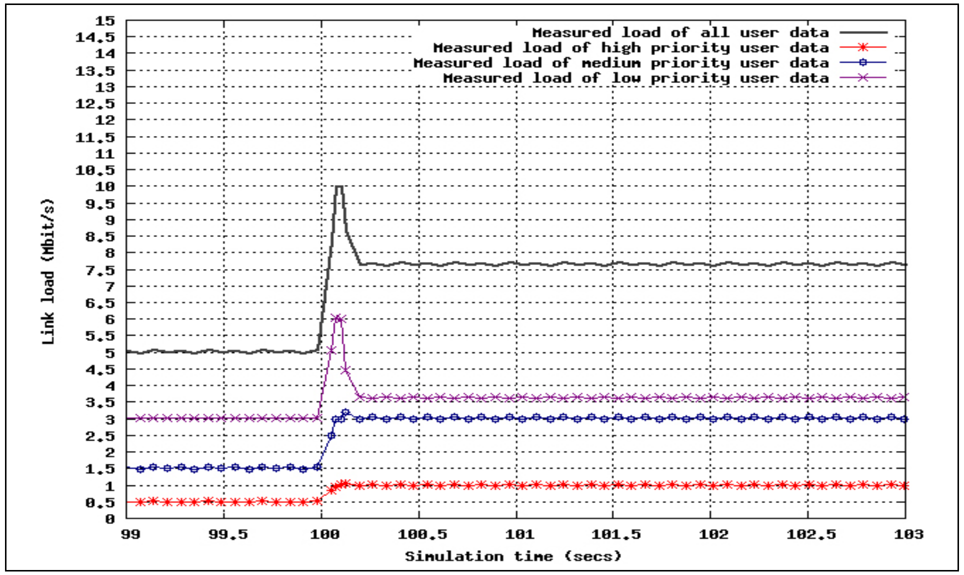


Figure 8.21 Link 1-2: Forward path, big-small flows

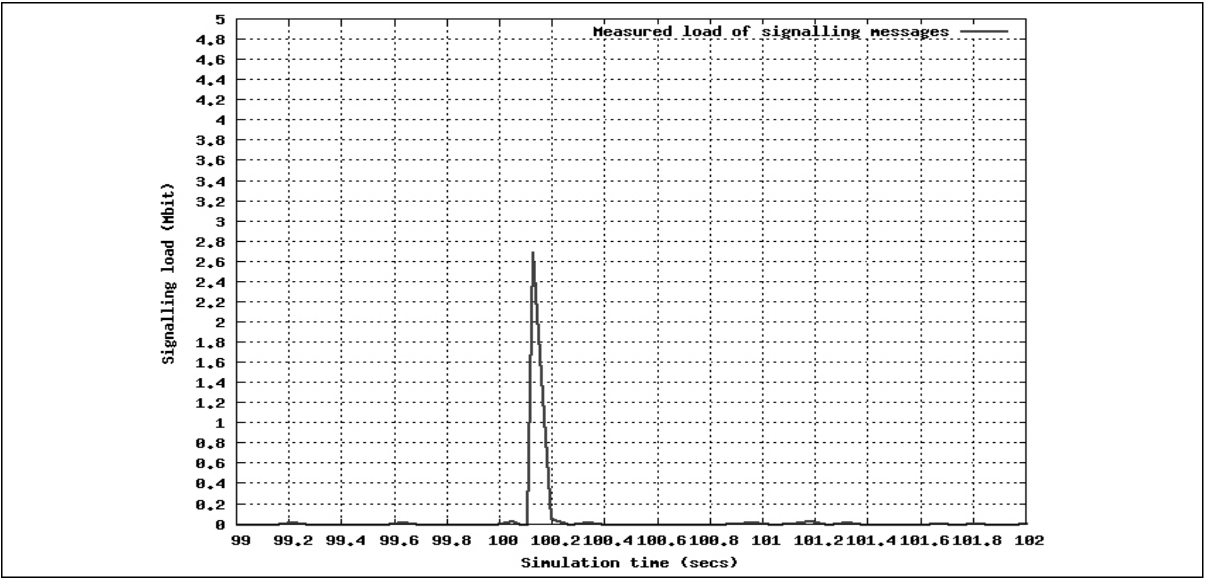


Figure 8.22 Signaling load of NOTIFY - big-small flows

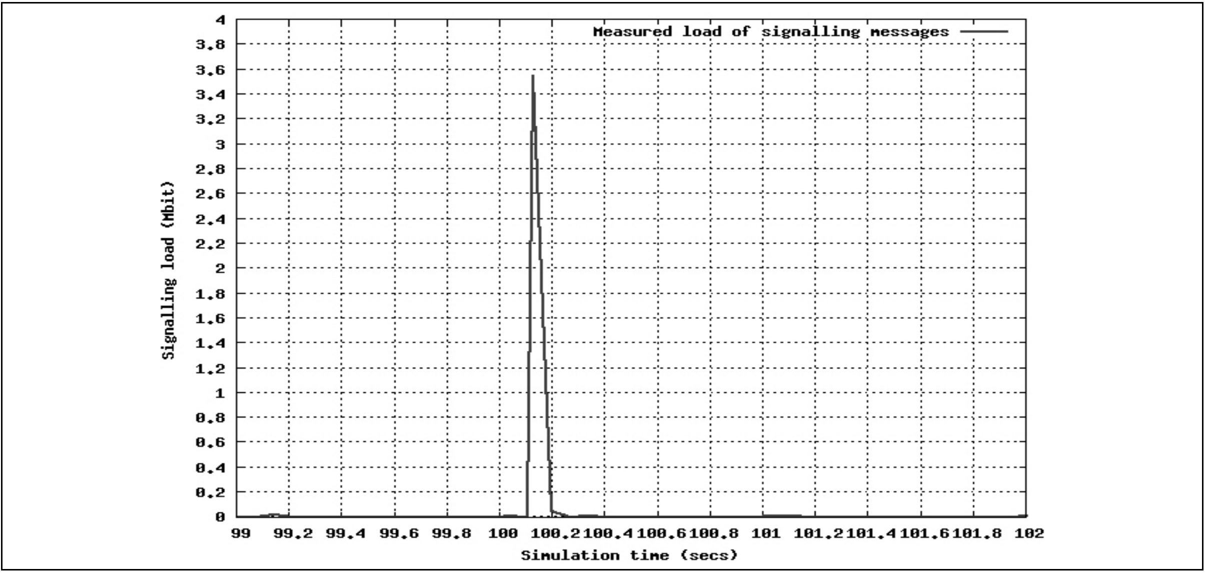


Figure 8.23 Signal load of NOTIFY - big-big flow

<u>**Conclusions:**</u>

- The bi-directional flow termination mechanism described in the beginning of section 6.5.3 does not affect the flow priority principle handling.
- A bigger difference in the size of the reservations in both direction results in a higher link utilization.
- The transfer of NOTIFY messages causes decrease in the link utilization on the reverse path when the severe congestion occurs only on the forward path.
- The *detection and handling time* is the same on both paths for both scenarios of severe congestion – on the forward or on the reverse path.

8.5 Experiment 4: Flow termination based on size:

The *goal* of this experiment is to observe the impact of flow selection and termination, based on reservation sizes, on the performance of the severe congestion solutions on the forward and on the reverse direction. In this type of experiments it is considered that the reservations are bi-directional and that one point severe congestion occurs on either the forward direction or on the reverse direction. Furthermore it is considered that three types of flow priorities are used: high, medium and low.
Note that in this set of experiments no ingress/egress pair aggregates are maintained by the edges.

In bi-directional reservations, the termination of flows leads to the decrease of bandwidth usage in both directions. If congestion occurs on the one path one approach is to terminate the least possible bandwidth on the other path. A mechanism that attempts to do that, addressed as *mechanism 1*, was described in section 6.5.2 and the results of it are shown in the previous section. A comparison is made with the mechanism for unidirectional reservations, addressed as *mechanism 2* (section 4.2.3).
The experiment is performed for the situation where the severe congestion occurs on the forward path. Mix of *big-big flows* and *big-small flows* is used with the rates that are specified in Table 8-5.
Beginning with the *big-big flows* experiment, when *mechanism 1* is used, the flow termination starts with flows with the smallest reverse size, which corresponds to the smallest forward size. To solve the severe congestion level of 100% the mechanism has to terminate, say X flows. *Mechanism 2* starts picking the flows with biggest forward size, which is also the biggest reverse size. To solve the severe congestion level of 100% *mechanism 2* stops, say Y flows, where Y is smaller than X. Figure 8.23 and Figure 8.26 depict the signaling load for *mechanism 1* and *mechanism 2*, respectively. The result can be also seen on the reverse path of the severe congested link 2-1. When *mechanism 1* is used, the drop in the total load of data packets (Figure 8.18) is bigger than for *mechanism 2* (Figure 8.24). The forward link 1-2 utilization is the same as in Figure 8.16. All other links have the identical load and are presented in Appendix C.3.
The second scenario uses the *big-small flows* mix. No differences in link utilization and signaling load is expected. The reason is very simple. *Mechanism 1* stops as first the flows with smallest reverse reservation. These are the flows with biggest forward reservation. *Mechanism 2*, on other hand, begins with the highest forward reservation flows, which corresponds to the smallest reverse size. It can be concluded that both mechanisms terminate the same number of flows. Therefore the same number of NOTIFY messages is generated and the signaling load is the same. These conclusions are confirmed by the experiment results (Figure 8.25). No difference is observed with Figure 8.19. All other links are shown in Appendix C.3.

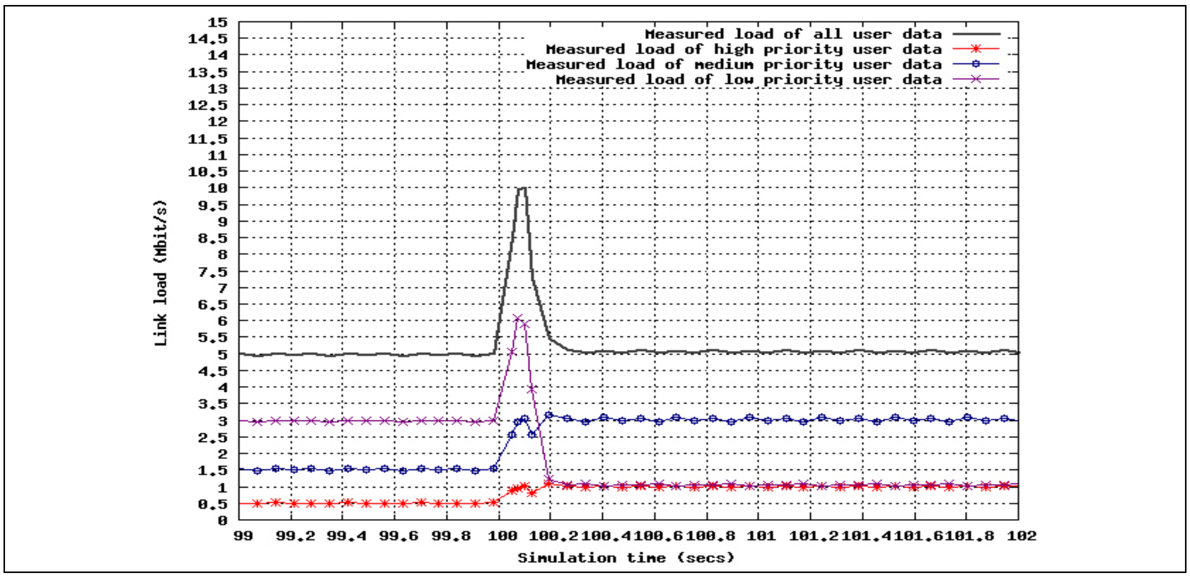


Figure 8.24 Link 2-1: *Mechanism 2* - big-big flows

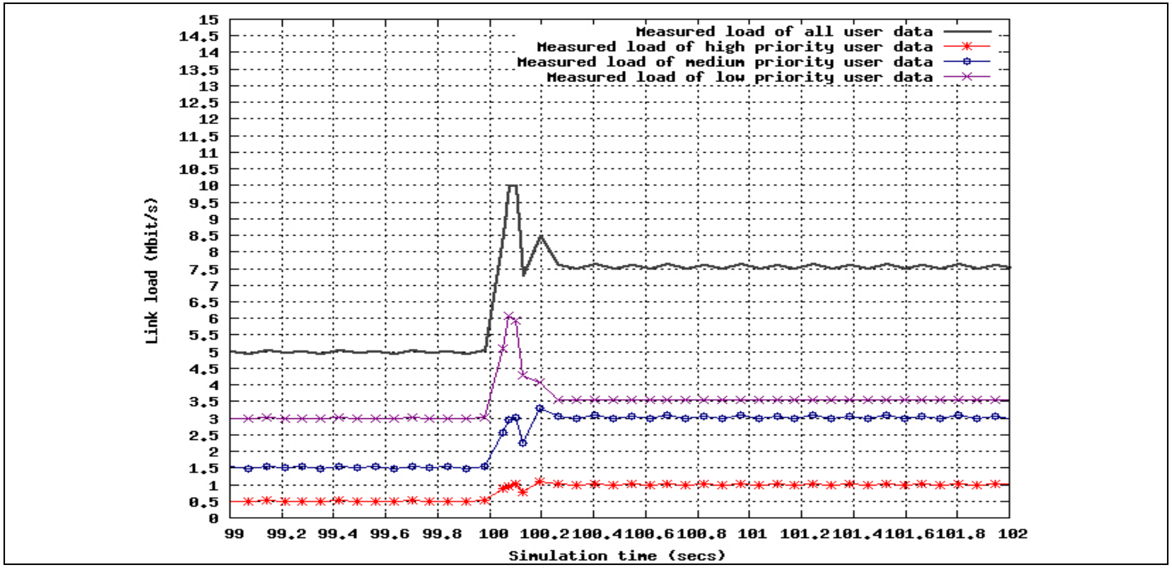


Figure 8.25 Link 2-1: Mechanism 2 - big-small flows

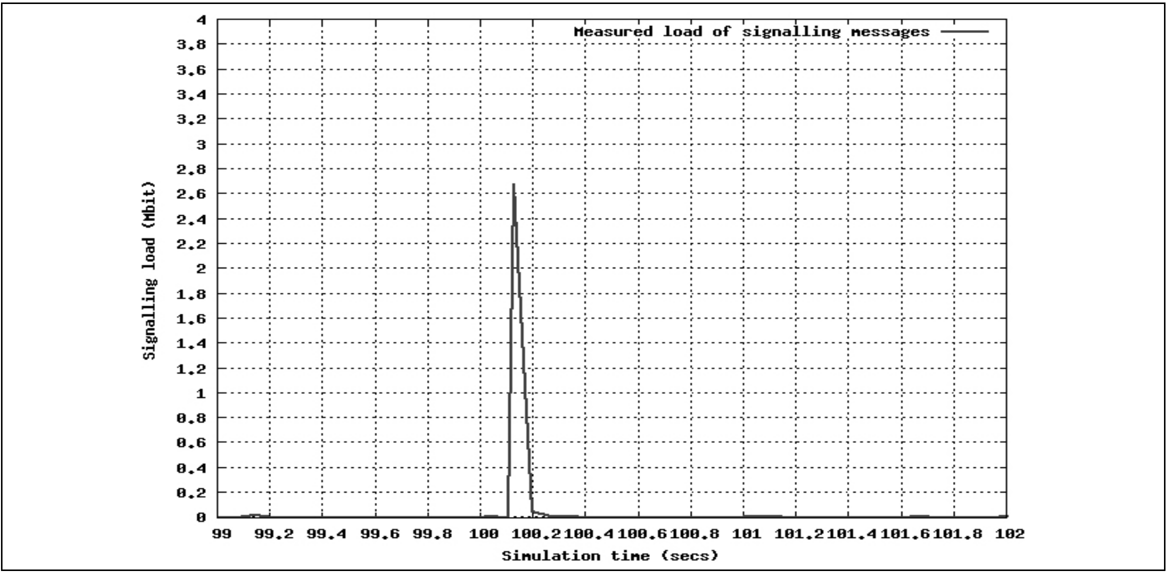


Figure 8.26 *Mechanism 2*: Signaling load

Conclusions:

- Both mechanisms, i.e., *mechanism 1* and *mechanism 2*, solve a severe congestion situation by terminating the same amount of bandwidth, i.e., 10 Mbps on the forward path and 5 Mbps on the reverse.
- *Mechanism 1* stops more flows but preserves the flows with biggest reservation in the direction opposite to the overloaded. It has higher signaling load than *mechanism 2*.
- *Mechanism 2* terminates a smaller number of flows but these are flows with the biggest reservations. It has lower signaling load than *mechanism 1*.
- The *detection and handling time* for both mechanisms is the same.
- Both mechanisms keep the flow priority principle in severe congestions.

8.6 Experiment 5: Optimization of the severe congestion mechanism

The *goal* of this experiment is to observe the impact of the optimized severe congestion solving mechanism, see section 6.5.3, on the performance of the severe congestion solutions in the forward and in the reverse direction. In this type of experiments it is considered that the reservations are bi-directional and that one point severe congestion occurs on the forward direction and another one, almost simultaneously, occurs on the reverse direction. Furthermore it is considered that three types of flow priorities are used: high, medium and low.
Note that in this set of experiments no ingress/egress pair aggregates are maintained by the edges.

In bi-directional operations when both directions are overloaded the ingress and the egress nodes choose flows to terminate. Each flow has forward and reverse reservation sizes. As result after the severe congestion is solved more flows might be terminated than it is necessary to solve the congestion. To observe that behavior and to test the algorithm that is proposed for optimization (section 6.5.3), the same simulation experiment is run, once without, i.e., *without optimization*, and once with the new algorithm, i.e., *with_optimization*.
The same network topology depicted in Figure 8.13 is used and the traffic aggregate rate from each source, per forward and per reverse direction is 10 Mbps (9,984 Mbps) load of only 16kbps flows. In other words all flows have the same size.
The results from the simulation experiments *without_optimization* show that the existing mechanism leads to undershoot on both directions (Figure 8.29, Figure 8.30). The reason is the lack of communication between the egress and ingress node and each of them terminates bandwidth proportional to the collected marked data packets. Nevertheless when speaking about bi-directional reservations, both severe congestions are related and when a flow is stopped actually resources in both directions are released. All source links can be seen in Appendix C.4.
Note that it might be expected that the drop in link load will be 50% if no optimization is used. This does not happen because a flow that has marked data packets on the forward path can also have marked data packets on the reverse path. As result double marking can

happen and the same flow can be chosen for termination by the ingress and the egress. Therefore the link utilization drop is not as big as expected.
The *detection and handling time* is about 0.55 sec and the flow priority principle is not affected by other factors besides the undershoot.
The performance of the *with_optimization* mechanism described in section 6.5.3 is tested. It is expected to solve the overshoot since the ingress node has information on the flows to be terminated in both directions and can compensate for the undesired flow termination drop. The same scenario, described in the beginning of the section is run with the optimization mechanism, *with_optimization*, see section 6.5.3. From the graphs about the overloaded links 1-2 (Figure 8.31) and 2-1 (Figure 8.32), it can be concluded that the optimization successfully solves the undershoot problem. The flow priority principle is kept and the *detection and handling time* is actually decreased to the value of 0.25 sec on both paths – forward and reverse. Again the temporal drop in the data flow, due to NOTIFY messages, is observed. The size of the drop is big because when only 16 Kbps flow sizes are used, a large number of flows have to be terminated to solve the congestion.

Conclusions:

- If no measures are taken such that no double termination of flows in the edge nodes happens there is undershoot on the reverse and on the forward direction.
- The use of the optimizations solves the overshoot on the forward and on the reverse link.
- The *detection and handling time* is faster when optimization is used.
- The priority of the flows is not affected regardless whether the optimization is used or not.
- All above observations imply that the existing mechanism does not perform as bad as expected, not too many flows are stopped unnecessary, but an optimization is possible.

8.7 State scalability comparison RMD-QOSM vs. QoS NSLP

The *goal* of performing these experiments is to study the scalability of the RMD-QOSM protocol when compared to the pure QoS NSLP protocol. The advantages of using RMD-QOSM protocol are shown in discussion on the scalability concerning the number of maintained states. The QoS NSLP states and their creation were discussed in section 3.3. To summarize stateful QoS NSLP nodes (edges) have two states – operational and reservation, while reduced state nodes (interior) have only a reservation state.
The comparison is organized completely as an analytical discussion. The drawback of such approach is that there is always the question whether the results are realistic and whether the analytical model of the protocol behavior corresponds to the real one. Therefore the simulation model is used to test whether the experimental results and the analytical expectations for the RMD-QOSM agree.
For the scalability research the simple topology of Figure 8.27 is used.

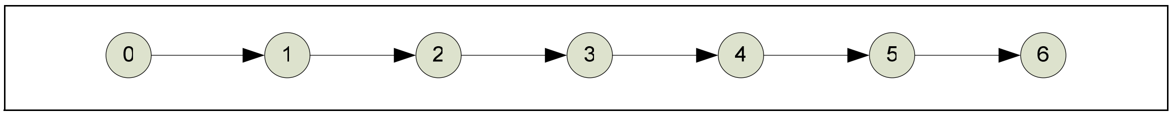


Figure 8.27 Simulated topology for scalability test

When RMD-QOSM is the protocol for evaluation, node 0 is the ingress node and node 6 is the egress (Figure 8.27). They, as edge nodes, are stateful nodes, all other nodes are reduced state nodes. The data path for all hundred flows is the same. Each flow passes through two stateful nodes, each with two states. The number of intermediate nodes on the date path is five each of them with one aggregated state (reservation). This state is common for all flows because all flows are taken to be from the same PHB class.

Based on the above relations the number of states is calculated for twenty different numbers of flows between 1 and 100. The taken values can be seen on the x- axis in Figure 8.28. On the other side hundred flows are started in simulation using the same topology from Figure 8.27. The implemented scalability monitor periodically collects information on the current the number of flows and number of installed states in the network. The step of collection is taken small enough so that the small flow changes can be detected. The collected and calculated number of states is the same. That implies that the analytical calculations are correct and they can be further used to evaluate the scalability of the RMD-QOSM protocol.

The next step is the calculation of the number of states when pure QoS NSLP protocol is used. If the same topology from Figure 8.27 is taken there are seven peer QoS NSLP nodes. Each of these nodes has one operational and one reservation state per flow. That makes in total 14 states only for one flow. The total number of states is calculated for the same number of flows as for RMD-QOSM. The comparison of the numerical results shows that RMD-QOSM installs much fewer states in the network. This is easily seen if the results are presented graphically (Figure 8.28). As conclusion, if possible RMD QoS NSLP should be used in the network core and the use of pure QoS NSLP should be limited to the peripheries of the network at the end users. That implies that the network core should be a RMD domain with peripheral QoS NSLP access and user nodes.

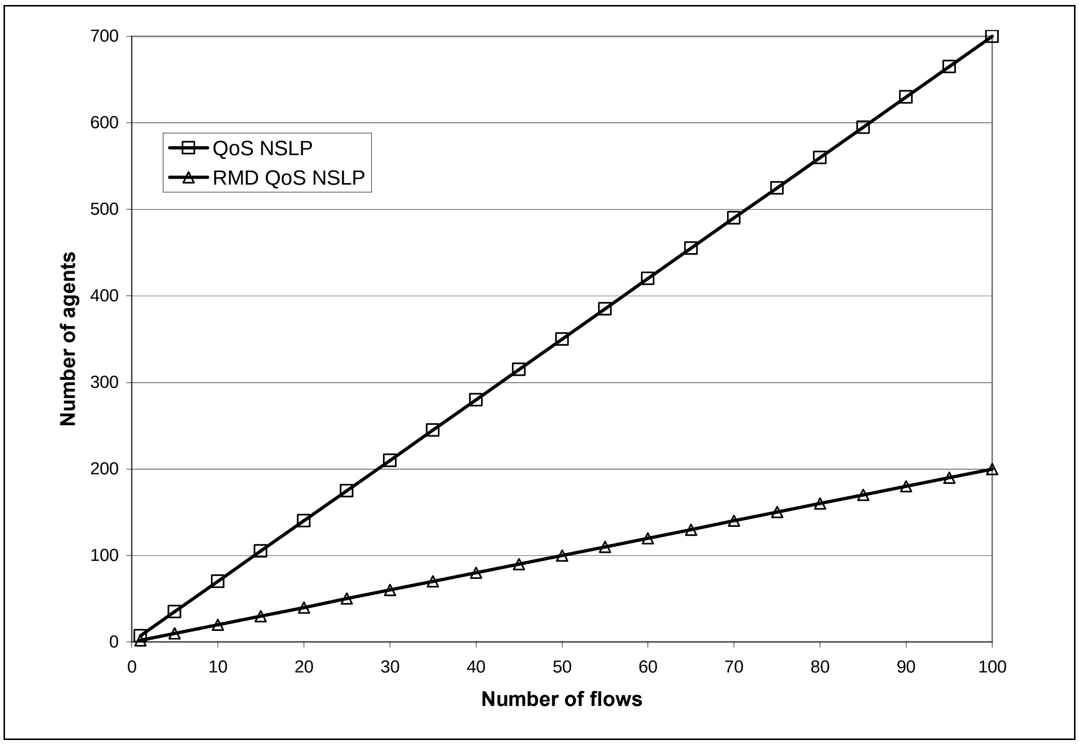


Figure 8.28 Scalability of state generation

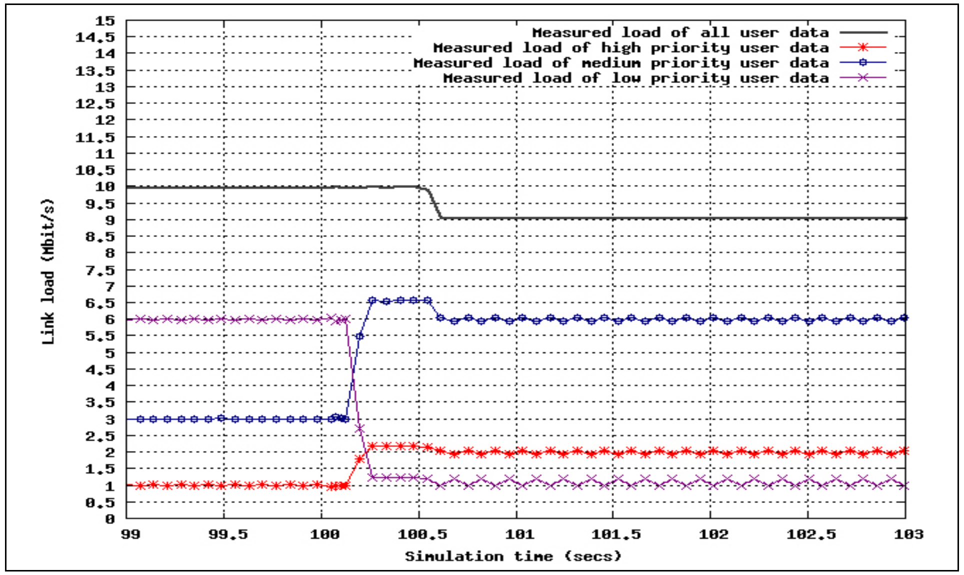


Figure 8.29 Link 1-2: Without_optimization - both paths congested

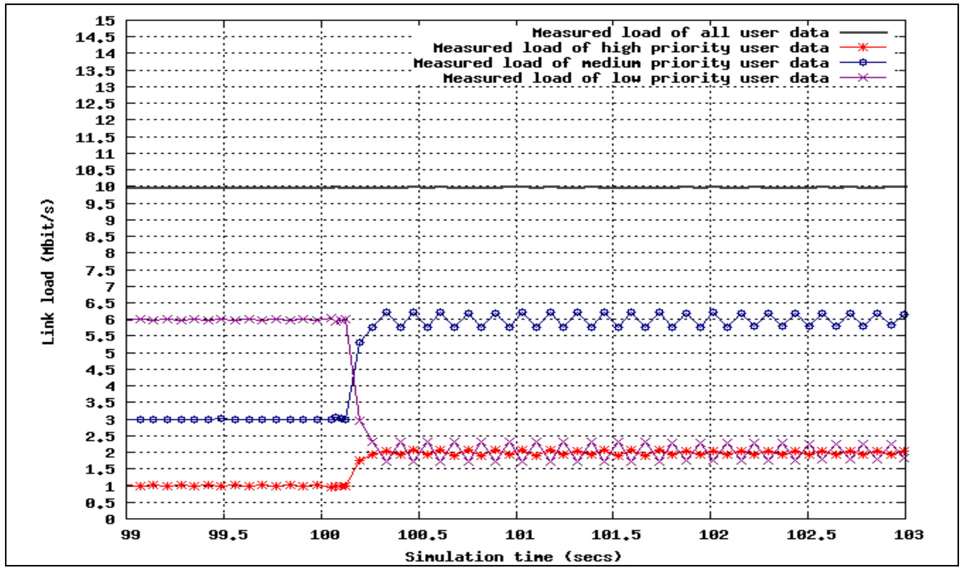


Figure 8.31 Link 1-2: With_optimization - both path congestion

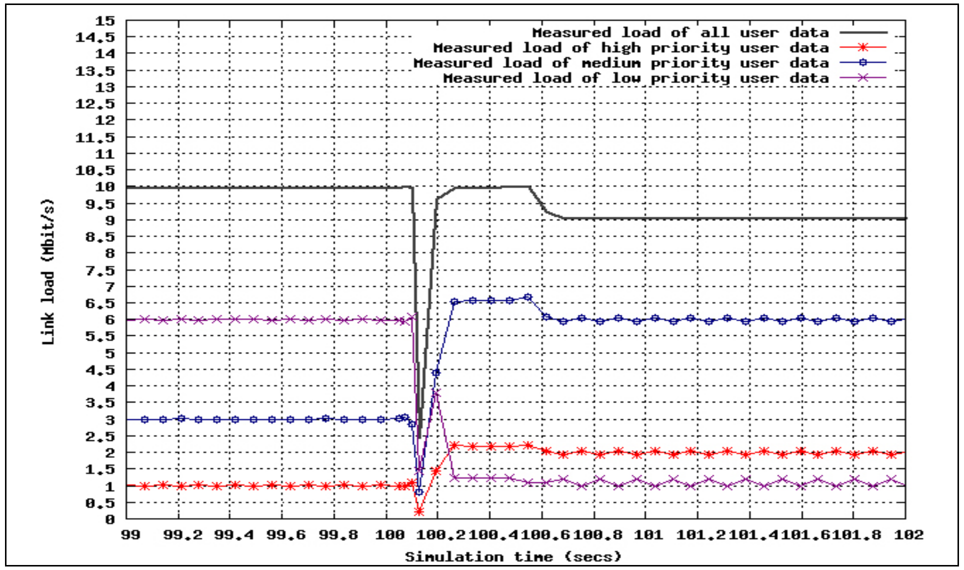


Figure 8.30 Link 2-1: Without_optimization - both paths congested

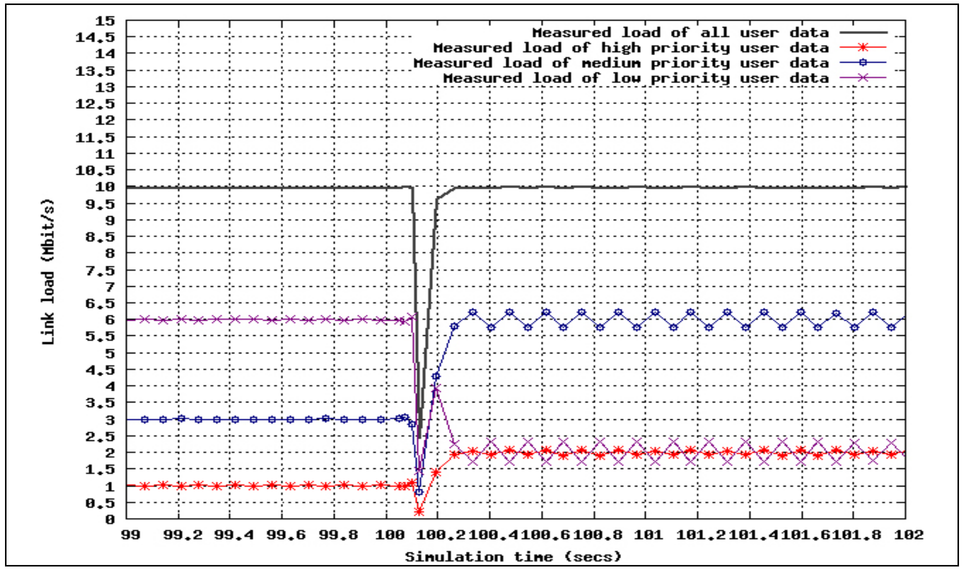


Figure 8.32 Link 2-1: With_optimization - both link congestion

9 Discussion

This chapter begins with presentation of the conclusions of the performed research and how these have influenced the solutions on the open issues of the RMD-QOSM protocol specification. Afterwards a short evaluation of the research in the terms of achieved goals is done, followed by the contributions of the author. Furthermore, proposals for future research topics are presented.

9.1 Conclusions

A new signaling framework NSIS was described in Chapter 3 and in particular one of the protocols defined within it. The later being RMD-QOSM that is used to deliver quality of service when the DiffServ concepts are applied. Before the protocol can be standardized its behavior and performance should be tested. This research aims to cover only some of the aspects of the protocol behavior. The whole RMD-QOSM is too complex to be of one independent study. For the performance evaluation the accent is put on the mechanisms for severe congestion detection and handling. A simulation model for the protocol was developed and implemented in a simulator environment. Simulation experiments were performed to test the RMD-QOSM behavior and to provide insight on its operation. Based on the experiment results important guidelines for optimization of the RMD-QOSM functionality can be given. First the existing open issues are listed and afterwards the achieved improvements are given.

In the first set of experiments the impact of dropping marked bytes on the severe congestion detection and handling solution is observed. Unidirectional reservations are considered and that multiple severe congestion points are occurring on the downstream path from ingresses to an egress. Marked data packets are used, among others, in the severe congestion detection and handling procedures. Therefore these packets are crucial for the functioning of RMD-QOSM. The existing queuing discipline, CBR queues, used in the existing RMD simulation model, cannot provide marked data forwarding in 100% of the situations. If queues get filled marked data packets may be dropped, which might degrade the performance of the severe congestion solution. It leads to longer time to solve the congestion and even decreased link utilization. To solve this problem a special type of queuing discipline might be used, i.e., dsRED queue. It has been shown that this results in certain scenarios in considerably better performance.

In the second set of experiments the impact of using the ingress-egress pair aggregates on the severe congestion performance is observed, when the flows are passing through different ingresses and through the same egress and when unidirectional reservations are used. Under ingress – egress pair aggregate is understood information on flows having the same ingress and egress node.

As it can be concluded by these experiments the use of ingress-egress pair aggregate causes disturbance in the flow priority principle used during the flow termination.

In the third and fourth sets of experiments the impact of the bi-directional reservation sizes on the forward and reverse directions on the performance of the severe congestion solutions on the forward and on the reverse direction is observed.

Different flow sizes were used as for the forward, as for the reverse direction. It was observed that the reservation size (on the path opposite to the path where the severe congestion occurs) plays an important role for the link utilization. Along with that, an optimization of the severe congestion mechanism was proposed, which is used to chose which flows should be stopped to solve the severe congestion. The results show that the proposal does improve the performance of the severe congestion mechanism.
In the fifth set of experiments the impact of flow selection and termination, based on the severe congestion state of the ingress, on the performance of the severe congestion solutions on the forward and on the reverse direction is observed.
An issue arises when bi-directional flows are terminated and the bandwidth reserved in both (forward and reverse) paths is released. In the old mechanism for severe congestion solving the nodes, ingress and egress, do not have information about how much bandwidth its corresponding node has stopped. Thus they do not posses knowledge of how much bandwidth was already terminated at the moment they have to choose flows for termination. An optimization to the mechanism is introduced where such knowledge is used, which results in an improved link utilization without an utilization undershoot.
At last an analysis on the scalability performance of RMD-QOSM was made. The analysis was supported by simulation experiments to confirm the analytical expectations. It was shown that QoS NSLP scales better when applied in combination with the RMD-QOSM. The use of RMD-QOSM results in considerable fewer number of states that have to be supported, especially for a large number of flows.

9.2 Achieved goals

The main goal of this research, performance evaluation of the RMD-QOSM protocol, was defined in chapter 1. Subsequently the goal was divided in sub-goals in chapter 2 and each of them was used to determine what simulation experiments should be performed.
Eventually a comparison between what was expected to be achieved at the beginning of this research and what was realized should be done. In the subsequent section it is shown which of the sub-goals were achieved and in which experiments.
The performance evaluation of RMD-QOSM in severe congestion situations for unidirectional reservations is achieved by using the simulation experiments of sections 8.2 and 8.3. A requirement towards the used network is discovered according to which large size of the queue for marked data packets is recommended for use. Furthermore bi-directional reservations were tested with a variety of different loads and several mechanisms for flow termination were applied to assess their effect on the link utilization and the number of flows in the network. These tests in combination with the experiment on the protocol operation when both paths are overloaded, sections 8.4, 8.5 and 8.6, fulfill the goal of performance evaluation of RMD-QOSM in bi-directional reservations. As result of the experiment on two overloaded directions, section 8.6, a new optimized algorithm for solving of severe congestion is proposed. No experiments were made on evaluation of the admission control mechanism used by RMD-QOSM. Finally, section 8.7, where a scalability comparison between RMD-QOSM and QoS NSLP is made, satisfies the last sub-goal on the scalability evaluation of the RMD-QOSM. A comparison with the goals set

in Chapter 2 shows that all goals except one were achieved – the admission control performance evaluation was not tested.

The modified admission control mechanism to support flow priority when resources are reserved is a topic of another ongoing research. The mechanism should be tested first independently before it could be included as a part of the RMD QOMS. Otherwise potential errors might be discovered during the evaluation of the more complex RMD-QOSM, when they can be more difficult to solve. This research has completed before the ongoing research on the modified admission control mechanism. Therefore the goal to test the behavior of the modified admission control mechanism within RMD-QOSM could not be completed.

9.3 Contribution

As result of the performed experiments existing optimization ideas such as notification of higher than 100% severe congestion are tested. Furthermore new optimizations are proposed, i.e. solving of utilization undershoot for bi-directional reservations and severe congestion on both paths. Based on the experiment results the functionality of the RMD-QOSM is modified to include these solutions that deliver better results. The modifications will be included in the new RMD-QOSM specification. Not as last it was concluded that the used protocol mechanism should be described in more details in the specifications, which is currently fulfilled.

9.4 Future work

Metaphorically speaking the battle was won but the war is not over. Aspects of the RMD-QOSM protocol behavior were examined and improved in performance, but still other questions are left open for further discussion and research. Some important topics for future work, in the author's opinion, are presented.

Probably the most obvious choice is the performance evaluation of the modified admission control mechanism but within the RMD-QOSM simulation model. Another one could be the test of the bi-directional operation with bigger diversity of flow sizes in both directions of the data transfer. As the conclusions show the flow size influences a lot the link utilization. It is true that the flow sizes in real network cannot be chosen but the RMD-QOSM can be evaluated for a wide range of reservation sizes such that the unexpected behavior is diminished. Also in the field of bi-directional reservations research can be done on the signaling and interoperation at the edge of the domain.

Next step in the simulation model of RMD-QOSM can be making the implementation of QoS NSLP independent from the implementation of RMD-QOSM. This would open a lot of new possibilities. One of them is the implementation of flow classification at the edge of the RMD domain. Another is the use of other QoS mechanism different than DiffServ.

More unsolved questions are probably left unmentioned especially what concerns the QoS NSLP broad functionality. They are left as future work of the researchers to come.

References

Literature

[AsBa06] J. Ash, A. Bader and C. Kappler. 2006. QoS-NSLP QSPEC Template. Version 8 http://www.watersprings.org/pub/id/draft-ietf-nsis-qspec-08.txt (IETF draft)

[BaKa05] A. Bader, G. Karagiannis, L. Westberg et al. 2005. QoS Signaling Across Heterogeneous Wired/Wireless Netwroks: Resource Management in Diffserv Using the NSIS Protocol Suite. Proceedings of the 2nd International Conference on Quality of Service in Heterogeneous Wired/Wireless Netwroks.

[BaWe06] A. Bader, L. Westberg, G. Kragiannis et al. 2006. RMD-QOSM – The Resource Management in Diffserv QOS Model. Version 5 http://www.watersprings.org/pub/id/draft-ietf-nsis-rmd-05.txt (IETF draft)

[CsTa04] A. Csaszar, A. Takacs, R. Szabo and T.Henk. 2004. State Correction After Re-Routing with Reduced State Resource Reservation Protocols. IEEE Global Telecommunications Conference.

[CsTa05] A. Csaszae, A. Takacs and A. Bader. 2005. A Practical Methid for the Efficient Resolution of Congestion in an On-path Reduced-State Signaling Environment. 2005. 13th International Workshop on Quality of Service (in proceedings), LNCS 3552, p. 286 – 297.

[EtPi00] J. Ethridge, P. Pieda, M. Baines and F. Shallwani. 2000. A network Simulator Differentiated Services Implementation. http://www-sop.inria.fr/mistral/personnel/Eitan.Altman/COURS-NS/DOC/DSnortel.pdf . Online publication.

[FuBa05] X. Fu, A. Bader, C. Kappler and H. Tschofenig. 2005. NSIS: A New Extensible IP Signaling Protocol Suite. IEEE Communications Magazine. Internet Technology Series (to be published).

[Jain91] R. Jain. 1991. The art of computer system performance analysis. John Willey and Sons Inc.

[KaBa04] G. Karagiannis, A. Bader, G. Pongracz et al. 2004. RMD – a lightweight application of NSIS. Proceedings on the 11th International Telecommunications Network Strategy and Planning Symposium.

[LeLa01] T. Lethbridge and R. Laganiere. 2001. Object – Oriented Software Engineering. McGraw Hill.

[MaKa06] J. Manner, G. Karagiannis and A. McDonald. 2006. NSLP for Quality-of-Service Signaling. Version 9 http://www.watersprings.org/pub/id/draft-ietf-nsis-qspec-08.txt (IETF draft)

[RFC1633] R. Braden, D. Clark and S. Shenker. 1994. Integrated service in the internet architecture. RFC 1633. http://www.ietf.org/rfc/rfc1633.txt (IETF draft)

[RFC2205] R. Braden, L. Zhang, S. Berson et al. 1997. Resource Reservation Protocol. RFC 2205. http://www.ietf.org/rfc/rfc2205.txt (IETF draft)

[RFC2475] S. Blake, D. Black, M. Carlson et al. 1998. An Architecture for Differentiated Services. RFC 2475. http://rfc.net/rfc2475.html (IETF draft)

[ScHa06] H. Schulzrinne and R. Hancock. 2006. GIST: General Internet Signaling Transport. http://ietf.org/internet-drafts/draft-ietf-nsis-ntlp-09.txt (IETF draft)

[WeBa06] A. Bader, L. Westberg, G. Kragiannis et al. 2006. RMD-QOSM – The Resource Management in Diffserv QOS Model. Latest version 7 http://ietf.org/internet-drafts/draft-ietf-nsis-rmd-07.txt (IETF draft)

[WeCs02] L. Westberg, A. Csaszar, G. Karagiannis et al. 2002 Resource Management in Diffserv (RMD): A Functionality and Performance Behavior Overview. Proceedings of 7th International Workshop on Protocols for High Speed Netwroks.

[WeJa03] L. Westberg, M. Jacobsson, S. Oosthoek et al. 2003. Resource Management in Diffserv (RMD) Framework.http://www.watersprings.org/pub/id/draft-westberg-rmd-framework-04.txt (IETF expired draft)

[WeKo03] L. Westberg. M. Jacobsson, M. Kogel et al. 20003 Resource Management in Diffserv on Demand (RODA) PHR. http://www.watersprings.org/pub/id/draft-westberg-rmd-od-phr-04.txt (IETF dreaft)

WWW

[cygwin, www] http://www.cygwin.com/ Main page of Cygwin Linux-like environment for Windows platform.

[Ericsson, www] http://www.ericsson.com/ericsson/worldwide/hungary.shtml The main page of Ericsson Hungary.

[ns, www] http://www.isi.edu/nsnam/ns/ Main page of the network simulator version 2.

[ns1, www] http://nile.wpi.edu/NS/ NS by Example, on-line tutorial on ns2 [ns2, www] http://www.isi.edu/nsnam/ns/tutorial/index.html Marc Greis Tutorial, on-line tutorial on ns2

[ns manual, www] http://www.isi.edu/nsnam/ns/ns-documentation.html The main page of the ns Manual, formerly known as ns Notes and Documentation.

[NSIS, www] http://ietf.org/html.charters/nsis-charter.html The NSIS working group of IETF.

[wiki, www] http://nl.wikipedia.org/wiki/Hoofdpagina On line encyclopedia, where definitions can be found.

Appendices

Appendix A.1: RMD simulation model states

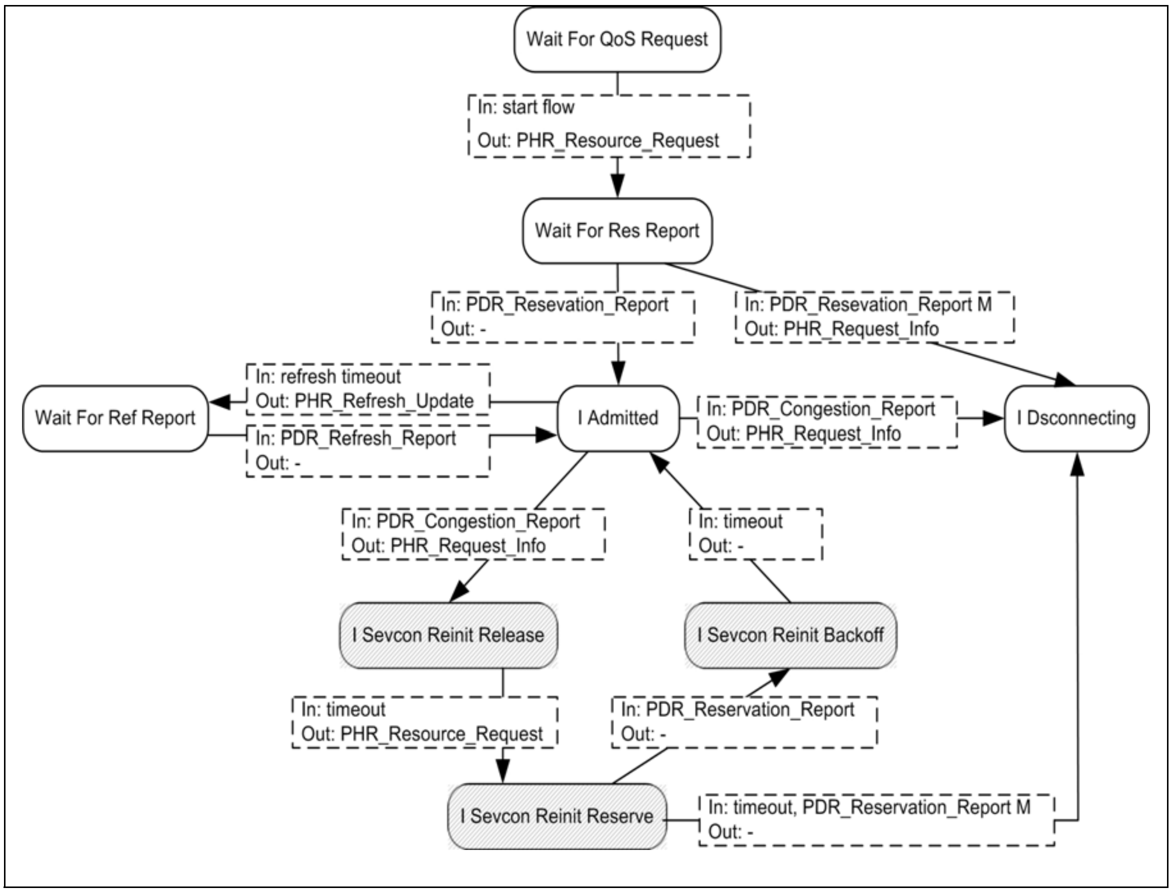


Figure A. 1 Functional state diagram, ingress node

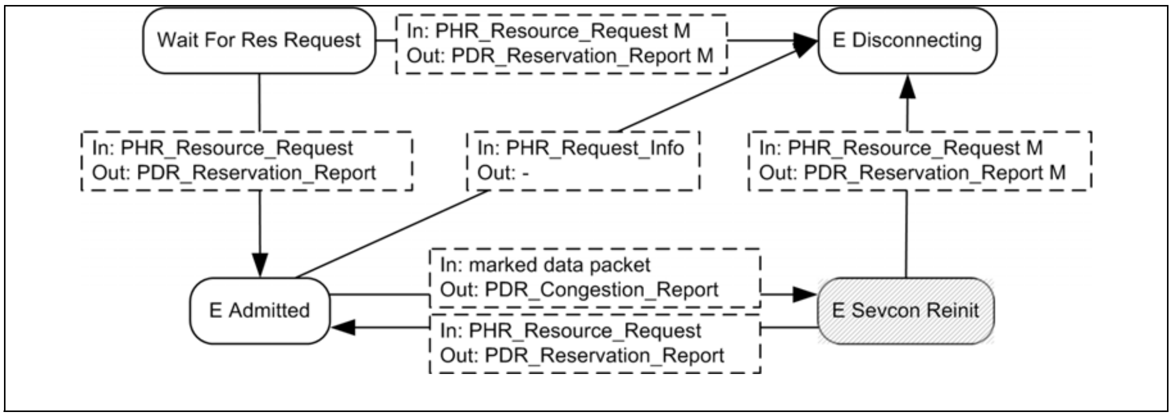

Wait For Res Request
In: PHR_Resource_Request M
Out: PDR_Reservation_Report M
E Disconnecting
In: PHR_Resource_Request
Out: PDR_Reservation_Report
In: PHR_Request_Info
Out: -
In: PHR_Resource_Request M
Out: PDR_Reservation_Report M
In: marked data packet
Out: PDR_Congestion_Report
E Admitted
E Sevcon Reinit
In: PHR_Resource_Request
Out: PDR_Reservation_Report

Figure A. 2 Functional state diagram, egress node

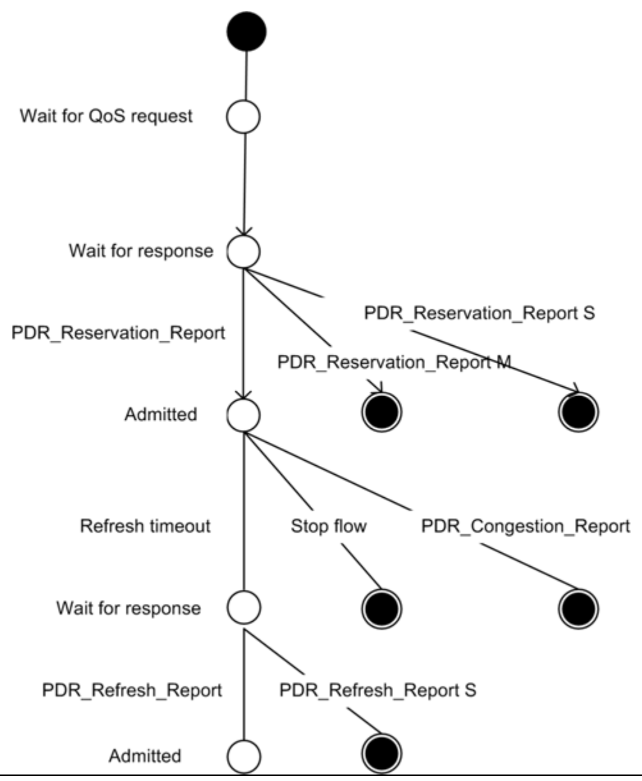
Appendix A.2: State machines

Table A. 1 Ingress node test suite

Case	Begin State	Input	Output	End state
1	Wait for QoS request	QoS request	PHR_Resource_Request	Wait for response
2	Wait for QoS request	PDR_Reservation_Report PDR_Reservation_Report, M=1 PDR_Reservation_Report, S=1 PDR_Refresh_Report PDR_Refresh_Report, S=1 PDR_Congestion_Report Refresh timeout Stop traffic Data packet	ERROR	Abort
3	Wait for response	PDR_Reservation_Report	None	Admitted
4	Wait for response	PDR_Reservation_Report, M=1	PHR_Resource_Release	-
5	Wait for response	PDR_Reservation_Report, S=1	PHR_Resource_Release	-
6	Wait for response	PDR_Refresh_Report	None	Admitted
7	Wait for response	PDR_Refresh_Report, S=1	None	-
8	Wait for response	QoS request PDR_Congestion_Report Refresh timeout Stop traffic Data packet	ERROR	Abort
9	Admitted	Refresh timeout	PHR_Refresh_Update	Wait for response
10	Admitted	Stop traffic	PHR_Reource_Release	-
11	Admitted	PDR_Congestion_Report	PHR_Resource_Rlease	-
12	Admitted	Data packet	Data packet	Admitted
13	Admitted	QoS request PDR_Reservation_Report PDR_Reservation_Report, M=1 PDR_Reservation_Report, S=1 PDR_Refresh_Report PDR_Refresh_Report, S=1	ERROR	Abort

Table A. 2 Egress node test suite

Case	State	Input	Output function	Transfer function
1	Wait for request	PHR_Resource_Request	PDR_Reservation_Report	Admitted
2	Wait for request	PHR_Resource_Request, M=1	PDR_Reservation_Report, M=1	-
3	Wait for request	PHR_Resource_Request, S=1	PDR_Reservation_Report, S=1	-
4	Wait for request	PHR_Refresh_Update PHR_Refresh_Update, S=1 PHR_Resource_Release Data packet	ERROR	Abort
5	Admitted	PHR_Refresh_Update	PDR_Refresh_Report	Admitted
6	Admitted	PHR_Refresh_Update, S=1	PDR_Refresh_Report, S=1	-
7	Admitted	PHR_Resource_Release	-	-
8	Admitted	Data packet, S=1	PDR_Congestion_Report	-
9	Admitted	PHR_Resource_Request PHR_Resource_Request, M=1 PHR_Resource_Request, S=1	ERROR	Abort



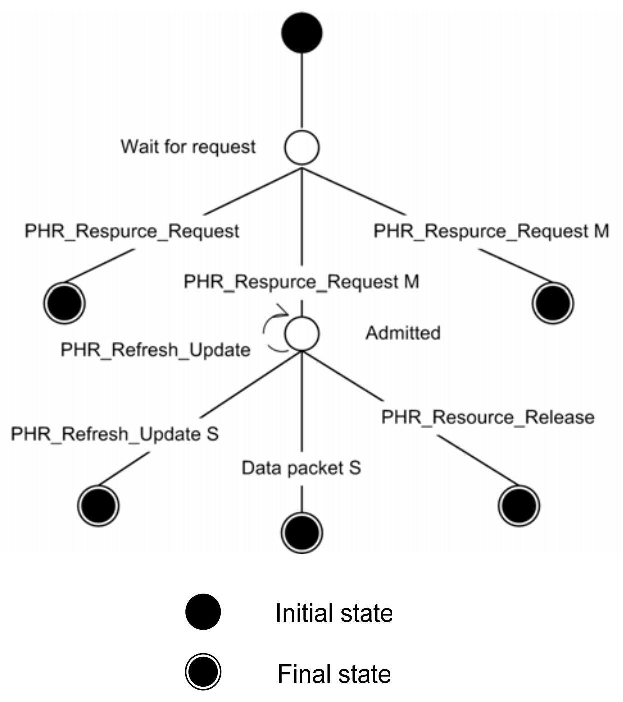


Figure A. 3 Egress node test suite **Figure A. 4 Ingress node test suite**

Appendix A.3: Source file documentation

Table A. 3 Class - source file correspondence

Source file	Classes from section 4.2	Classes in the model
rmd-ch.h rmd-ch.cc	Congestion detection	RMDCH
	BWUpdate	BWUpdateTimer
	BWMeasured	BWMeasuredRMDCH
	RateProportionalMarking	RateProportionalMarkingBWMeasuredRMDCH
	DampenedRateProportional Marking	DampenedRateProportionalMarkingBWMeasured RMDCH
	ThresholdBasedMarking	ThresholdBasedMarkingBWMeasuredRMDCH
	TBUpdate	TBUpdateTimer
	TBMeasured	TBMeasuredRMDCH
	ThresholdBasedMarking	ThresholdBasedMarkingTBMeasuredRMDCH
rmd-adc.h rmd-adc.cc	Admission control	RMDADC
	Measurement based	MBACADC
	RODA	RODAADC
	CellEnd	RODAADCCellEndTimer
rmd-edge.h rmd-edge.cc	Edge agent	RMDEdge
	Timeout	ReportTimeoutTimer
	MBAC agent	MBACEdge
	RODA agent	RODAEdge
	Refresh	RODARefreshTimer
	Congestion handling	RMDEdgeSevereCongestionHandler
	Congestion handling timer	RMDEdgeSevereCongestionHandler
enhanced-flow-mon.h/cc	EnhancedFlowMon	EnhancedFlowMon

Appendix B: Implementation functions

Table B. 1 RODAADC out {} method

```
void RODAADC::out(Packet* p) {
…………………………………………………………………………

      switch (rmdh->phr_type) {
            case PHR_RESERVED:
            break;

// RESERVE for reservation request
            case PHR_RESOURCE_REQUEST:
            {
                  ……………………………………………………
// Rejected request due to link overload, set Hop U flag to true
                  bool s = false;

                  if ((s = check_sevcon(class_id))) {
                        rmdh->s = s;
                        rmdh->m = true;

                        if (!rmdh->t) {
                              rmdh->t = true;
                              rmdh->pdr_ttl = iph->ttl();
                              rmdh->hopU_flag = true; //Desi
                        }
                        ………………………………………
// Not M marked RESERVE
                  } else if (rmdh->m == false) {
                        ………………………………………………………
// Admitted RESERVE due to enough resources, increase Admitted hops field
                        if (admitted) {
                              lastsum_[class_id] += rmdh->requested;
                              rscount_[class_id] += rmdh->requested;
                              newsum_[class_id] += rmdh->requested;
                              rsmesgcount_[class_id]++;
                              rmdh->admitted_hops++; //Desi
                              auto_dump();
                              ……………………………………………………………………
// Rejected RESERVE due to not sufficient resources, set Hop U flag to true
                        } else {
                              rmdh->m = true;
                              rmdh->t = true;
                              rmdh->pdr_ttl = iph->ttl();
                              rmdh->hopU_flag = true; //Desi

                              ……………………………………………………………………
                        }
                  }
            }
            break;
…………………………………………………………………………
}
```

Table B. 2 Remark proportion N

```
void DampenedRateProportionalMarkingBWMeasuredRMDCH::update(void) {
      BWMeasuredRMDCH::calc_bw();
……………………………………………………………………
//Detection threshold is passed
            if (avg_measure_[i][1] >
second_level_congestion_thresholds_[i]) {
                  perclass_bytesToMark_[i] = (int)
ceil((avg_measure_[i][1] -
second_level_congestion_restoration_thresholds_[i]) *
bw_update_period_);
                  perclass_bytesToMark_[i] = (int)
ceil(perclass_bytesToMark_[i]/remark_proportion_ - bytesMarked);
………………………………………………………………………
      bw_update_timer_.resched(bw_update_period_);
}
```

Table B. 3 RMD-QOSM implemented states

```
// Operational state variables
      int ingress_rsn_/egress_rsn_; //Reservation sequence number
      int ingress_rii_/egress_rii_; //Request identifier information
      int ingress_SID_/egress_SID_; //session id end-to-end
      int ingress_local_SID_/egress_local_SID_; //local session id
// Reservation state variables
      int requested_resources_, egress_requested_resources_;
      int requested_reverse_, egress_requested_reverse_;
```

Table B. 4 end-to-end RESERVE message

```
void RMDEdge::send_reservation_message_e2e(void) {
      ingress_rsn_++;
      Packet* pkt = allocpkt();
      hdr_cmn::access(pkt)->size() = resource_request_pkt_size_;
      hdr_rmd* rmdh = hdr_rmd::access(pkt);
      hdr_nslp* nslph = hdr_nslp::access(pkt);
      nslph->nslp_type = RESERVE;
      nslph->rsn = ingress_rsn_;
      nslph->scope_flag = false;
      nslph->replace_flag = false;
      Tcl& tcl = Tcl::instance();
      tcl.evalf("Simulator set soft_state_refresh_period_");
      nslph->refresh_period = atof(tcl.result());

      rmdh->phr_type = PHR_RESERVED;
      rmdh->pdr_type = PDR_RESERVATION_REQUEST;
      rmdh->session_id = ingress_SID_; //Desi e2e session id
      rmdh->send_time = NOW;
      send(pkt, 0);

      if(desidebug_ > 1)
            printf("(Ingress)[%d] e2e resource request SID %d, RSN
%d.\n",agent_id_, rmdh->session_id,nslph->rsn);
}
```

Table B. 5 CongestionHandler, recv_marked_packet method

```
RMDEdgeSevereCongestionHandler instproc recv_marked_packet {agent bytes
disconnected} {
      $self instvar total_marked_bytes agent_status_matrix
priority_list agent_list

// Variant 1: no pair
      incr total_marked_bytes [expr $bytes*[RMDCH set
remark_proportion_]]

// Variant 2: ingress-egress pair aggregate
/*set source [$agent set dst_addr_]

      if { [lsearch $source_list $source] == -1 } {
            lappend source_list $source
            set total_marked_bytes($source) 0
            set total_affected_bytes($source) 0
            puts "new source $source was added.
$total_marked_bytes($source)"
            }

      incr total_marked_bytes($source) $income_bytes_
*/

      if {$disconnected} {
            return;
      }

      set priority [$agent set preemption_priority_]

      if { [lsearch $priority_list $priority] == -1 } {
            lappend priority_list $priority
      }

      set agent_list_index [lsearch $agent_list $agent]

      if { $agent_list_index == -1 } {
            lappend agent_list $agent
            set agent_list_index [expr [llength $agent_list] - 1]
      }
// Variant 1
      set agent_status_matrix($priority,$agent_list_index) "marked"
// Variant 2
/*set agent_status_matrix($priority,$source,$agent_list_index) "marked"
*/
}
```

Table B. 6 CongestionHandler measurement_ended method

```
RMDEdgeSevereCongestionHandler instproc measurement_ended {} {
    ........................................................................
// Variant 2: Loop for source node and priority
      #     foreach k $source_list {
      #           set handled_bytes $previous_lag
// Variant 1: Loop for priority
            for {set i 0} {$i < $size} {incr i} {
                  if { $handled_bytes >= $total_marked_bytes($k) } {
                        break
                  }
                  ...................................................................................
                        if  { $bytes_marked_agents > 0 } {
                              if { [expr $handled_bytes +
$bytes_marked_agents] > $total_marked_bytes($k) } {
                                    set next_agent_to_terminate [$self
get_max marked_agents [expr $total_marked_bytes($k) - $handled_bytes]]
// Indexing, $k, is used by Variant 2
```

Table B. 7 CongestionHandler get_min_reverse method

```
RMDEdgeSevereCongestionHandler instproc get_min_reverse {
agent_list_name } {
      ..........................................................................

      for {set i 0} {$i < $size} {incr i} {
            set temp_agent [lindex $agent_list $i]
            set traffic_agent [$temp_agent set traffic_sender_]

// Calculation for egress agent
      if {!$traffic_agent} {
                  set temp_bw [$temp_agent set
egress_requested_reverse_]
// Calculation for ingress agent
            } else {
                  set temp_bw [$temp_agent set requested_resources_]
            }
// Calculation of smallest flow
            if { $temp_minimum == -1} {
                  set temp_minimum $temp_bw
                  set index $i
            } elseif { $temp_bw < $temp_minimum } {
                  set index $i
                  set temp_minimum $temp_bw
            }
      }

      if { $index != -1 } {
            set result_agent [lindex $agent_list $index]
            set agent_list [lreplace $agent_list $index $index]
      }
      return $result_agent
}
```

Table B. 8 Congestion handler terminate procedure

```
RMDEdgeSevereCongestionHandler instproc terminate { agent_list } {
      $self instvar sev_con_handler_measurement_period_ terminated_bw
      set size [llength $agent_list]

      for {set i 0} {$i < $size} {incr i} {
            set agent [lindex $agent_list $i]
            set traffic_agent [$agent set traffic_sender_]
            if {$traffic_agent == 0} {

//RDMEdge egress agent
                  [lindex $agent_list $i] terminate
            } else {

//RMDEdge ingress agent
                  set temp_bw [expr double([$agent set 
requested_resources_]) * double([$agent set RMDBWUnit_]) * 
$sev_con_handler_measurement_period_]
                  set terminated_bw [expr $terminated_bw + $temp_bw]
                  [$agent set myflow] stop
            }
      }
}
```

Table B. 9 Flow monitor support class

```
proc setup_and_start_link_monitor { i j } {
    global ns n link_monitor_files link_monitor_measurement_interval
    set l [$ns link $n($i) $n($j)]
    set fmon [$ns make_enhanced_flow_mon]
    $ns attach-fmon $l $fmon
    set link_monitor_files($fmon-out) [open $i-$j-out.dump w]
    puts  $link_monitor_files($fmon-out)  "#  time  |  total  load  | 
aggregate user data load | aggregate signaling load | low priority user 
data  load  |  medium  priority  user  data  load  |  high  priority  user  data 
load  |  low  priority  signaling  load  |  medium  priority  signaling  load  | 
high priority signaling load"
    $fmon set my_src_id_ $i
    $fmon set my_dst_id_ $j
    $ns  after  $link_monitor_measurement_interval  "measure_load  $fmon 
$link_monitor_measurement_interval"
}
```

Appendix C.1: Two point severe congestion

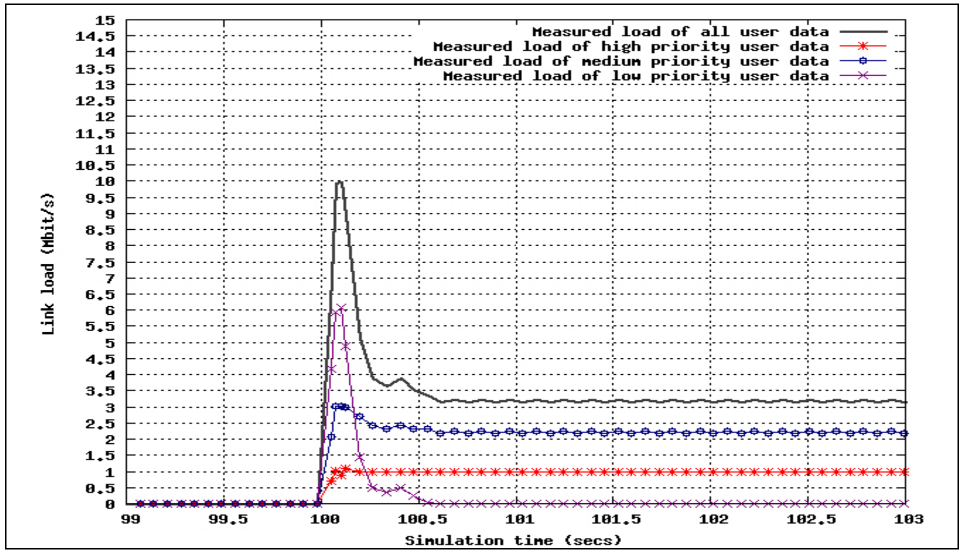


Figure C. 1 Link 3-0: Higher congestion on first link (0-1) – Not dropped marked

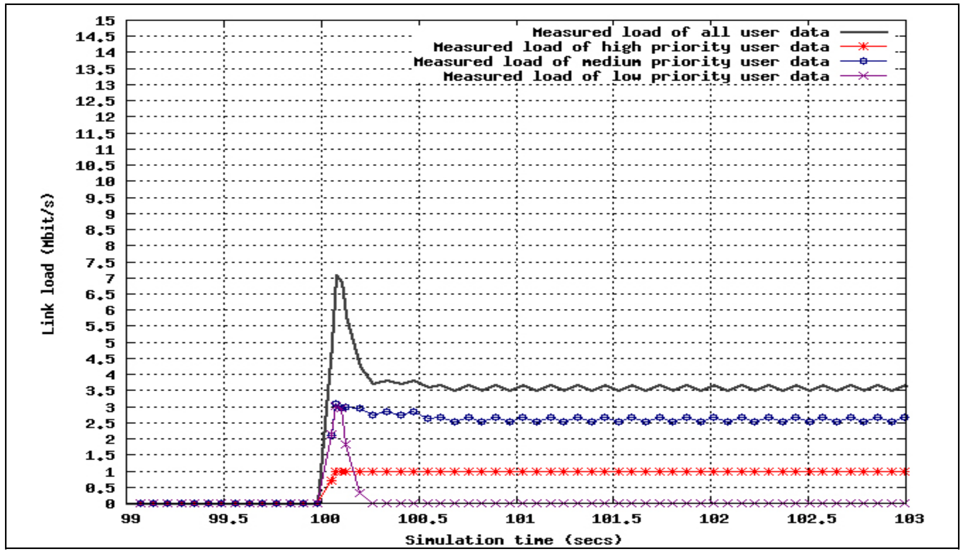


Figure C. 3 Link 4-1: Higher congestion on first link (0-1) – Not dropped marked

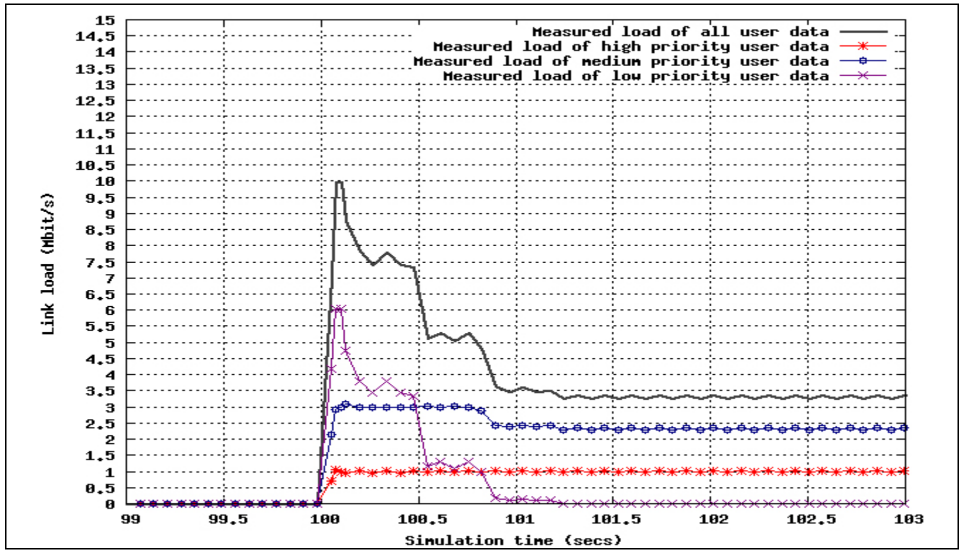


Figure C. 2 Link 3-0: Higher congestion on first link (0-1) – Dropped marked

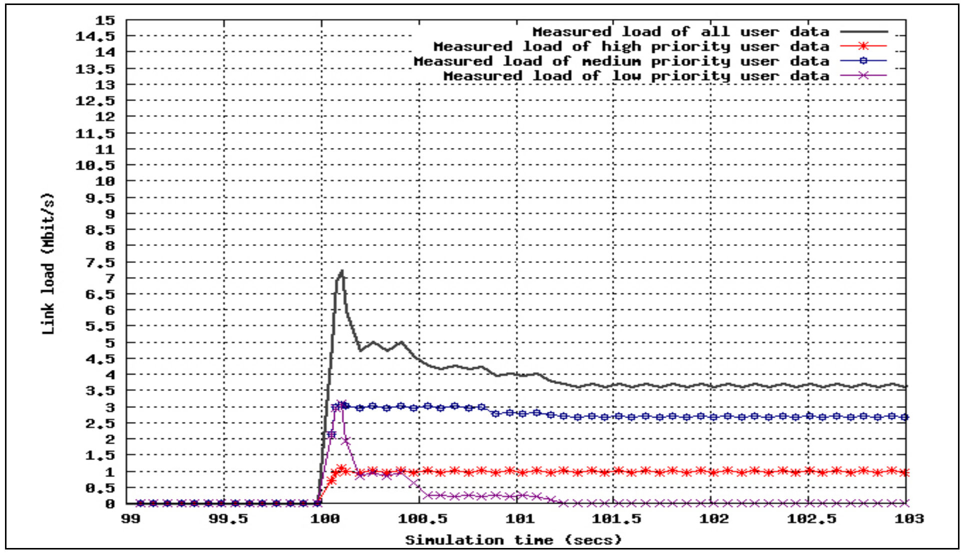


Figure C. 4 Link 4-1: Higher congestion on first link (0-1) – Dropped marked

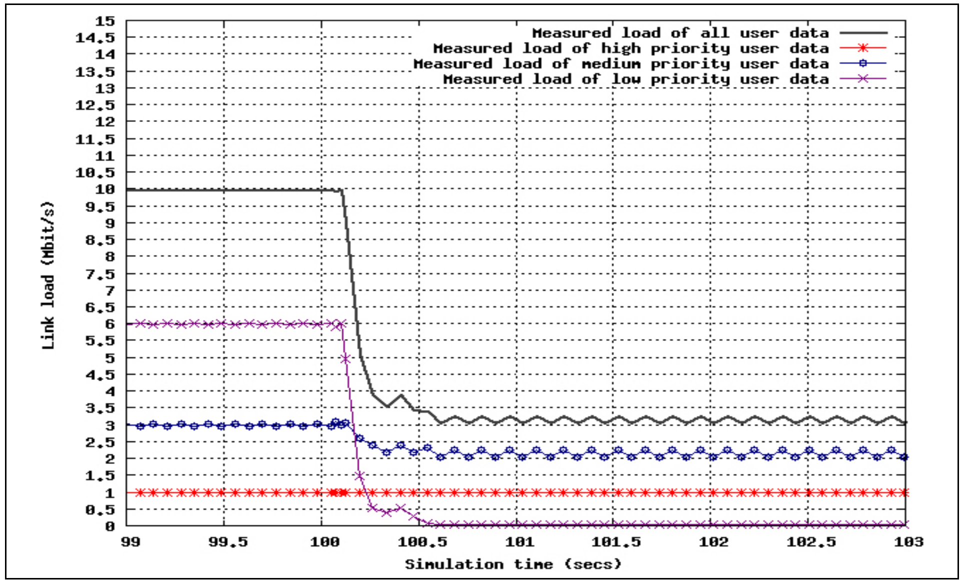


Figure C. 5 Link 5-0: Higher congestion on first link (0-1) – Not dropped marked

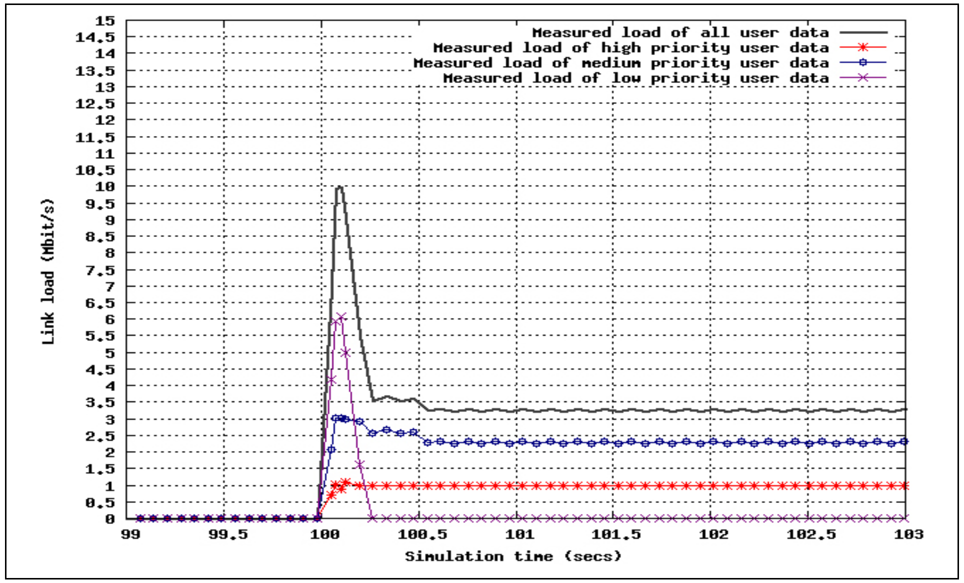


Figure C. 7 Link 3-0: Higher congestion on second link (1-2) – Not dropped marked

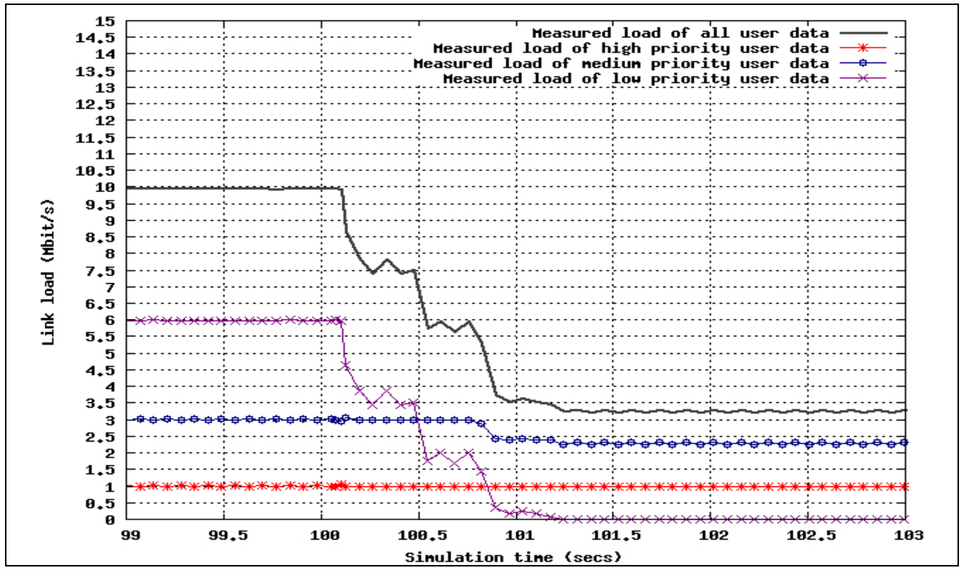


Figure C. 6 Link 5-0: Higher congestion on first link (0-1) – Dropped marked

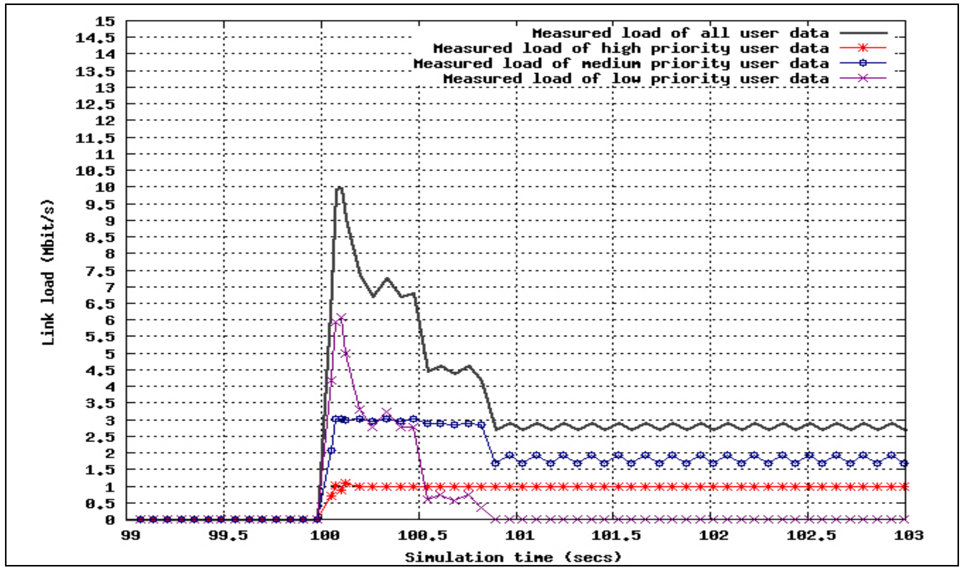


Figure C. 8 Link 3-0: Higher congestion on second link (1-2) – Dropped marked

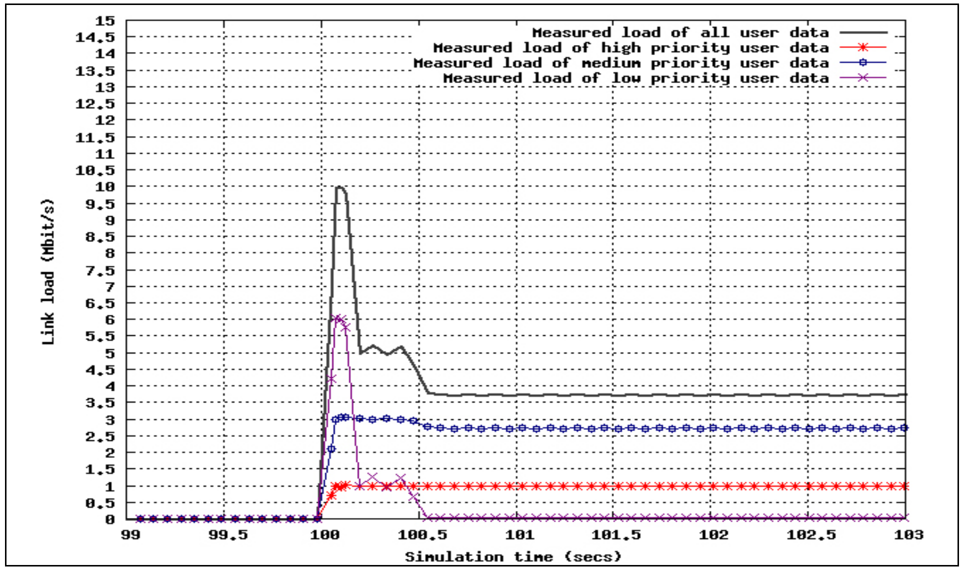


Figure C. 9 Link 4-1: Higher congestion on second link (1-2) – Not dropped marked

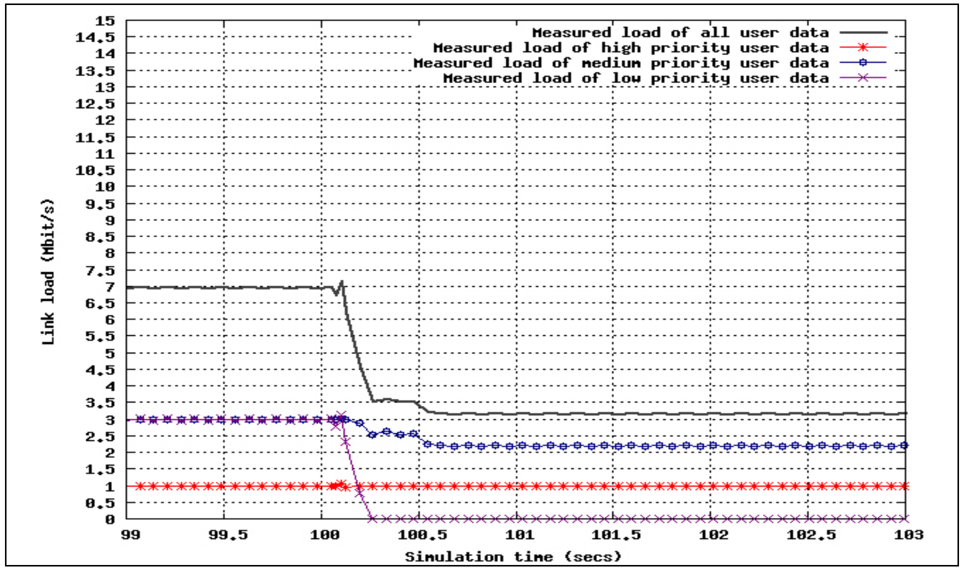


Figure C. 11 Link 5-0: Higher congestion on second link (1-2) – Not dropped marked

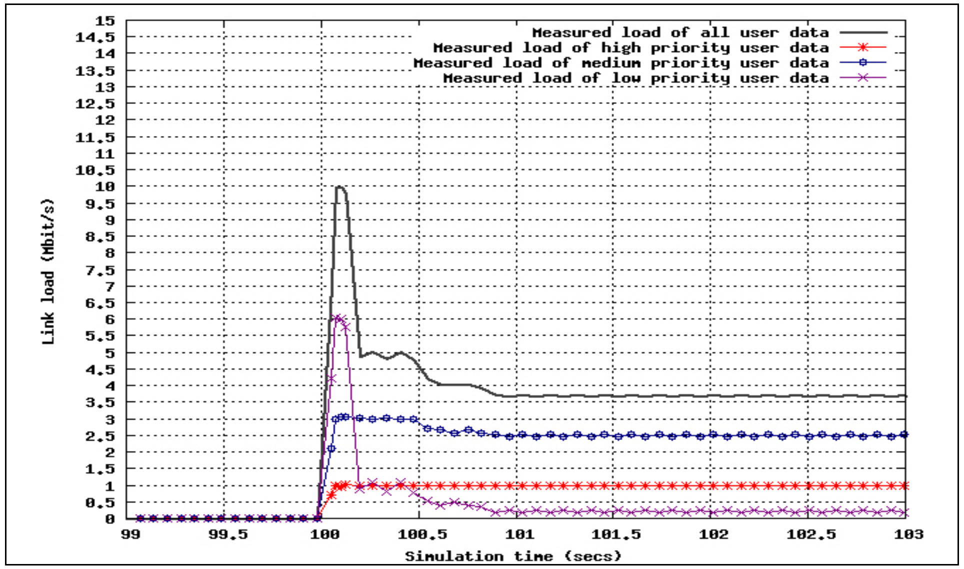


Figure C. 10 Link 4-1: Higher congestion on second link (1-2) – Dropped marked

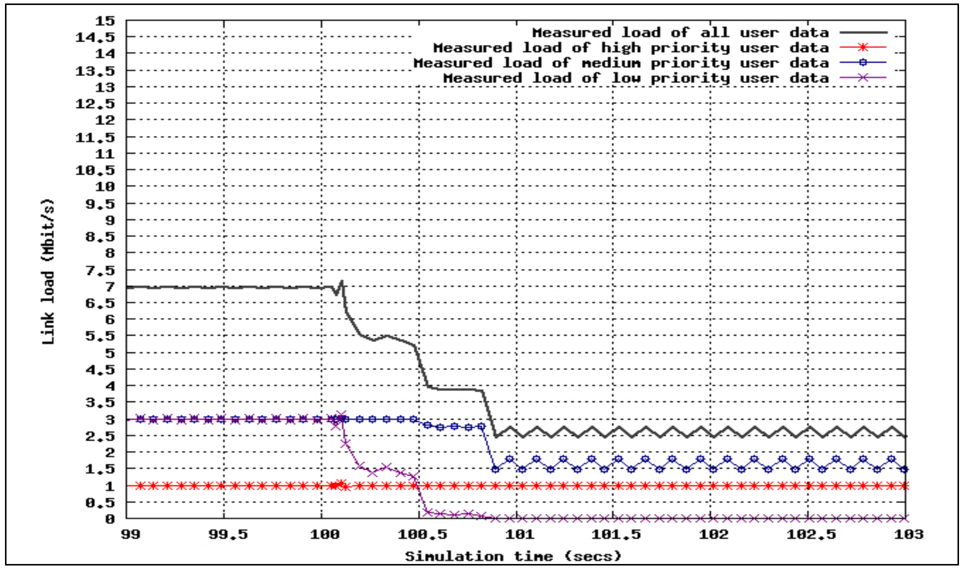


Figure C. 12 Link 5-0: Higher congestion on second link (1-2) – Dropped marked

Appendix C.2: Pair vs. no-pair aggregate

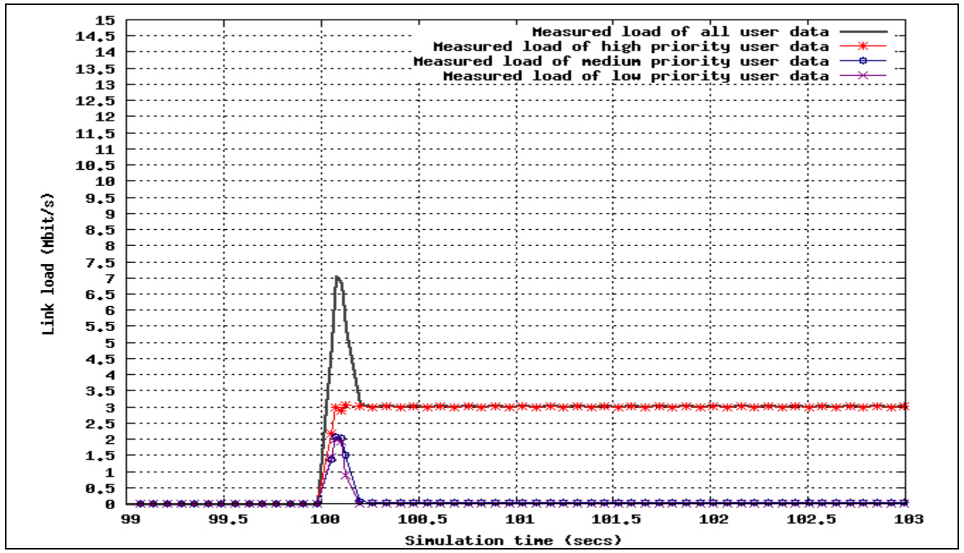


Figure C. 13 Link 0-3: No ingress-egress pair

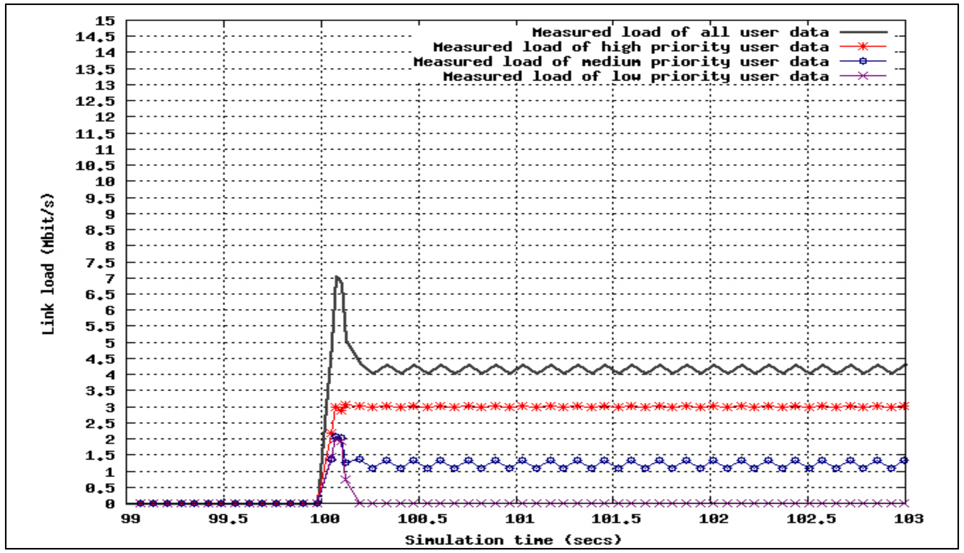


Figure C. 15 Link 0-3: Ingress-egress pair

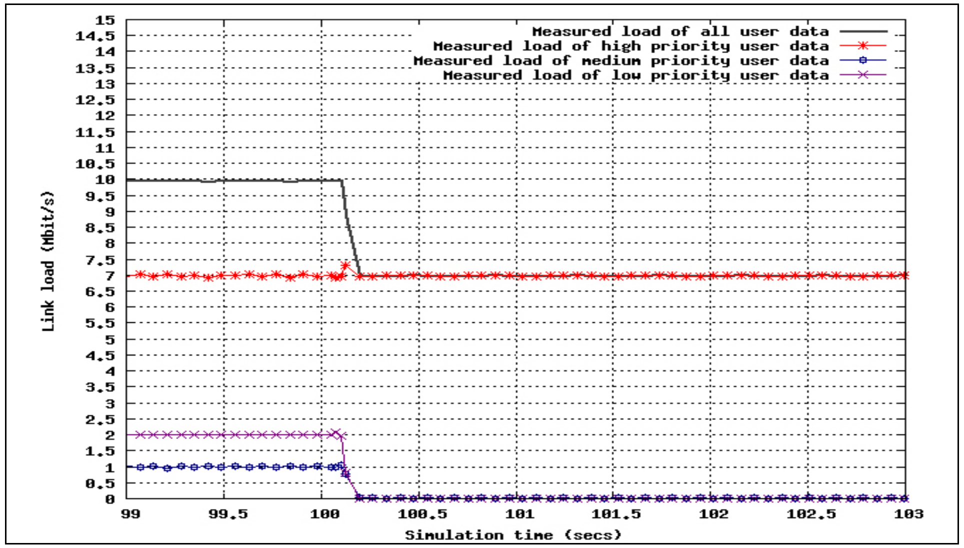


Figure C. 14 Link 2-3: No ingress-egress pair

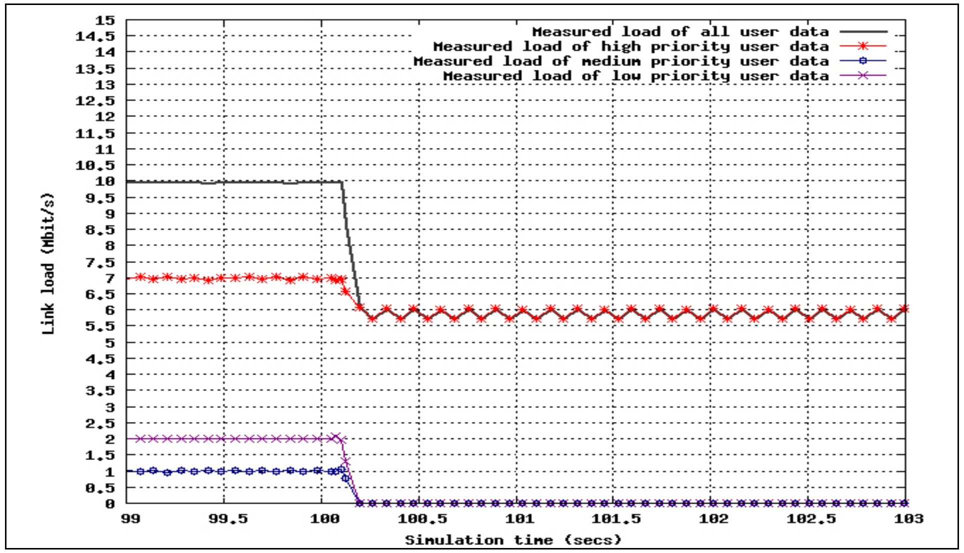


Figure C. 16 Link 2-3: Ingress-egress pair

Appendix C.3: One path severe congestion

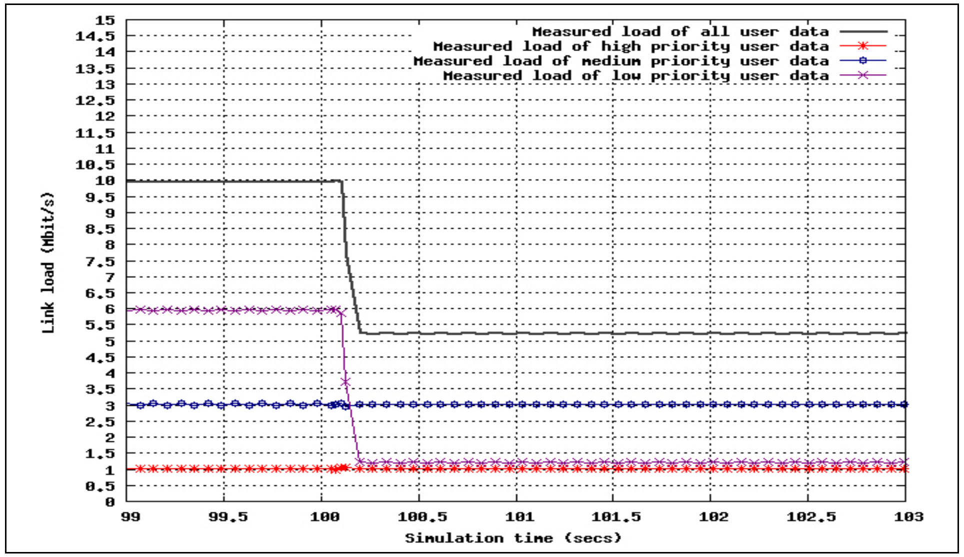


Figure C. 17 Link 0-1: Forward path, big-big flows

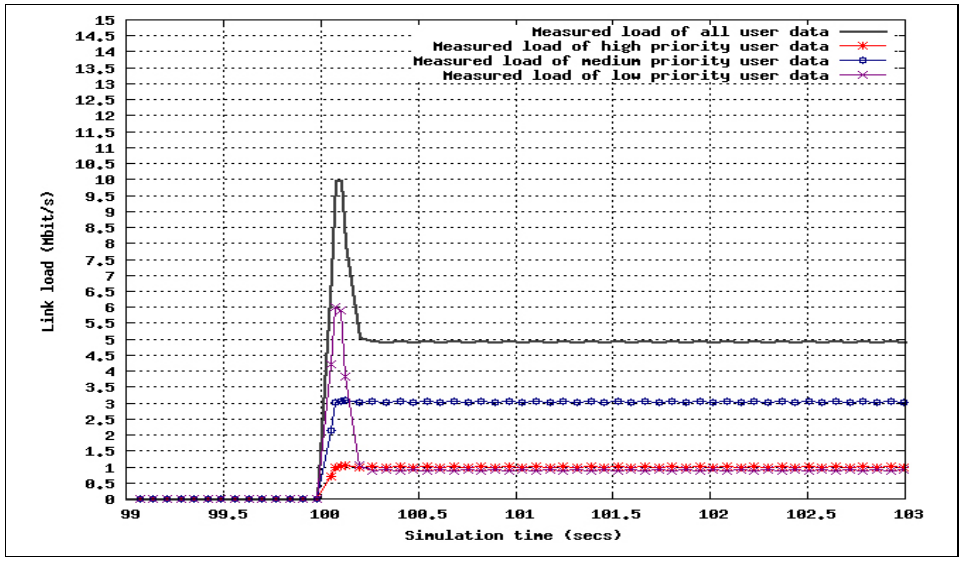


Figure C. 19 Link 3-1: Forward path, big-big flows

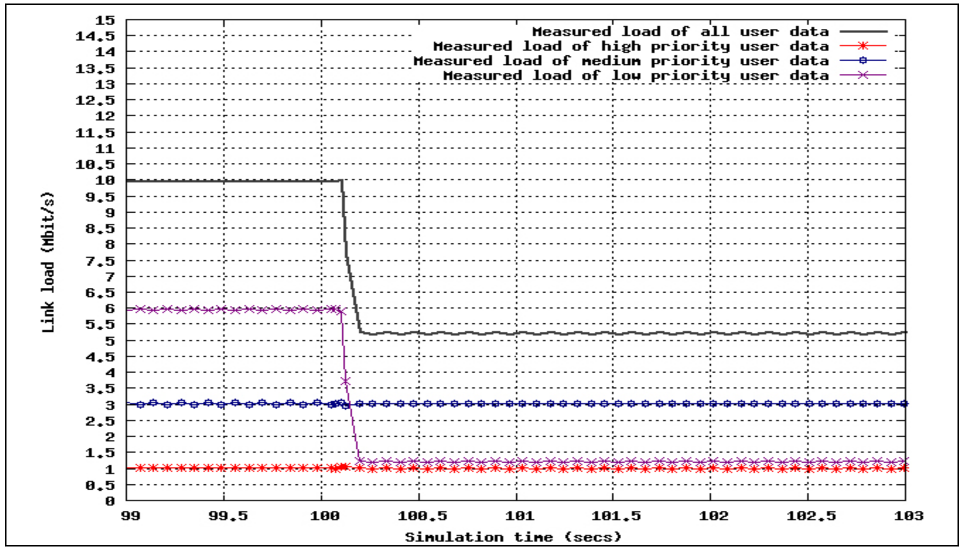


Figure C. 18 Link 0-1: Forward path, big-big flows, mechanism 2

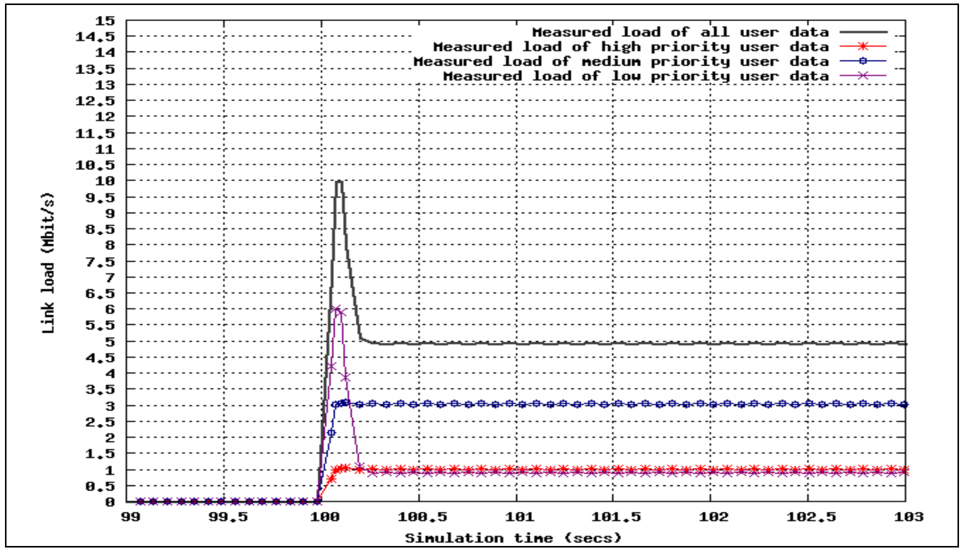


Figure C. 20 Link 3-1: Forward path, big-big flows, mechanism 2

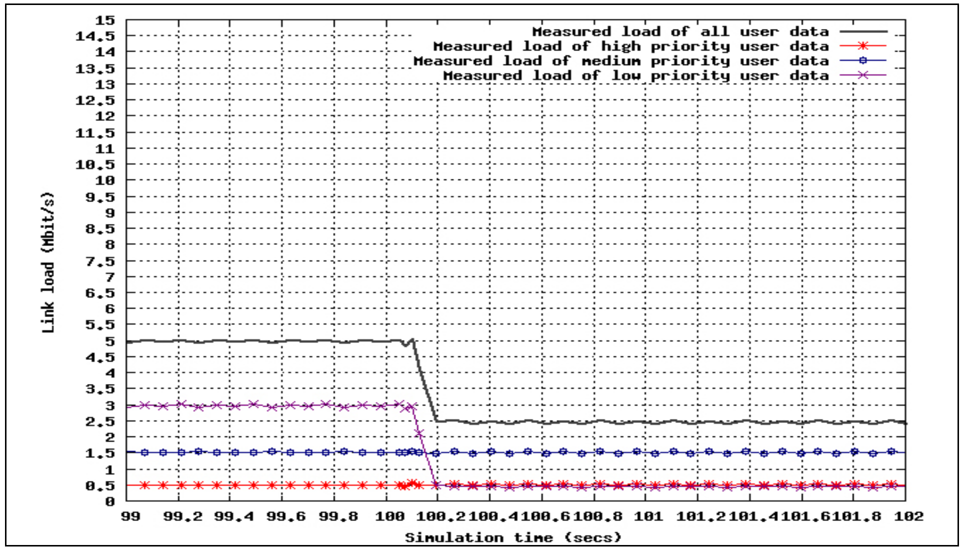


Figure C. 21 Link 4-2: Forward path, big-big flows

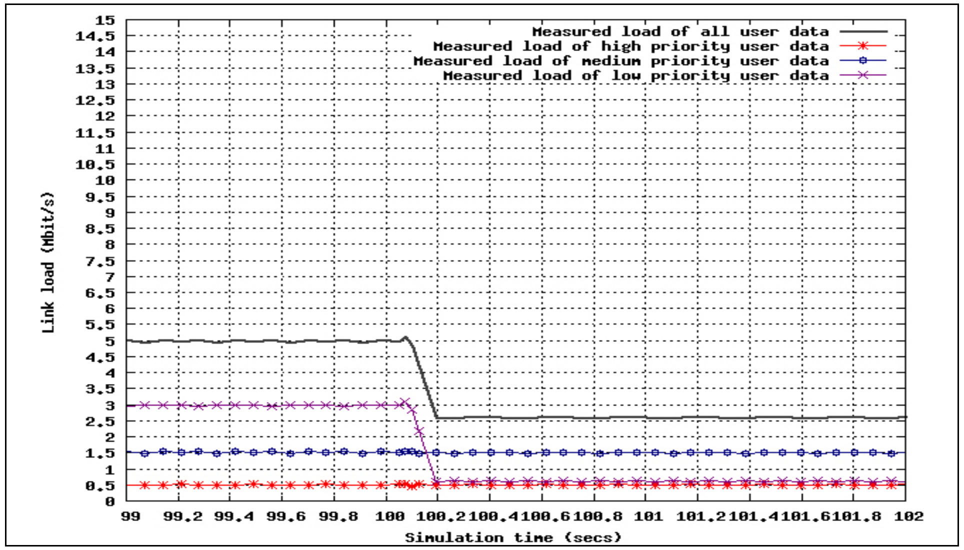


Figure C. 23 Link 5-2: Forward path, big-big flows

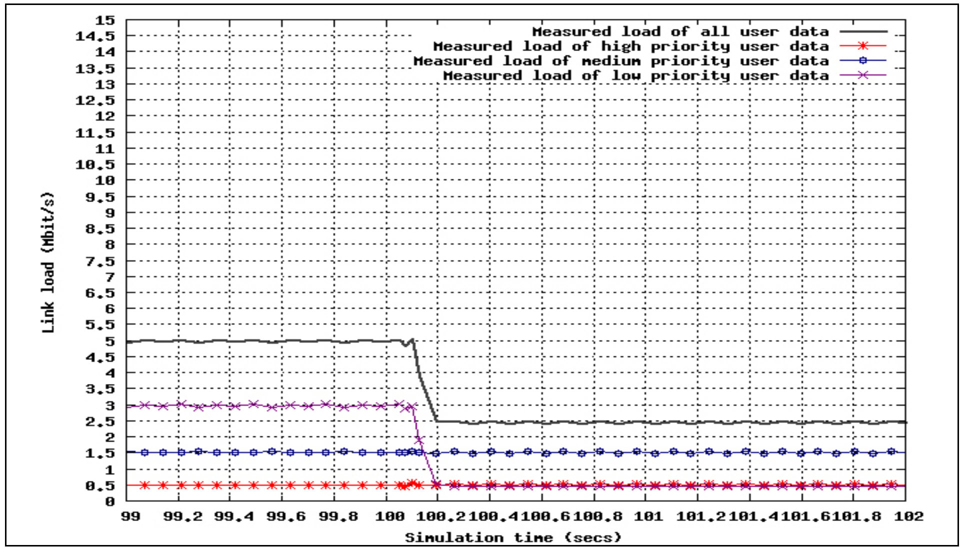


Figure C. 22 Link 4-2: Forward path, big-big flows, mechanism 2

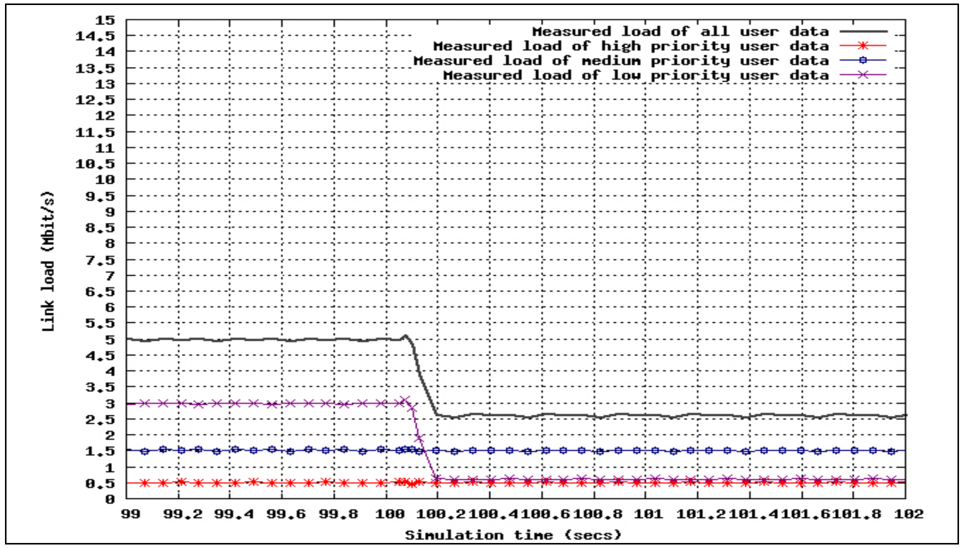


Figure C. 24 Link 5-2: Forward path, big-big flows, mechanism 2

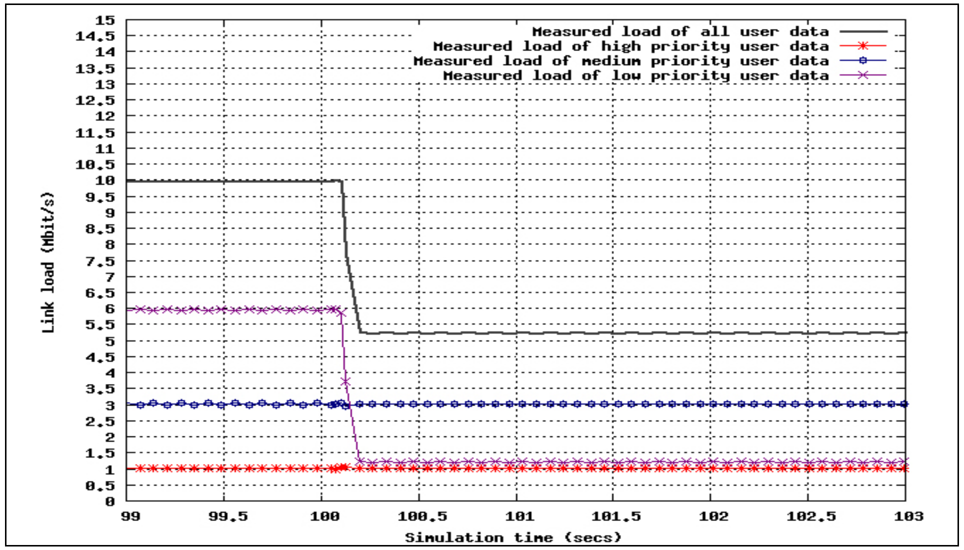


Figure C. 25 Link 0-1: Forward path, big-small flows

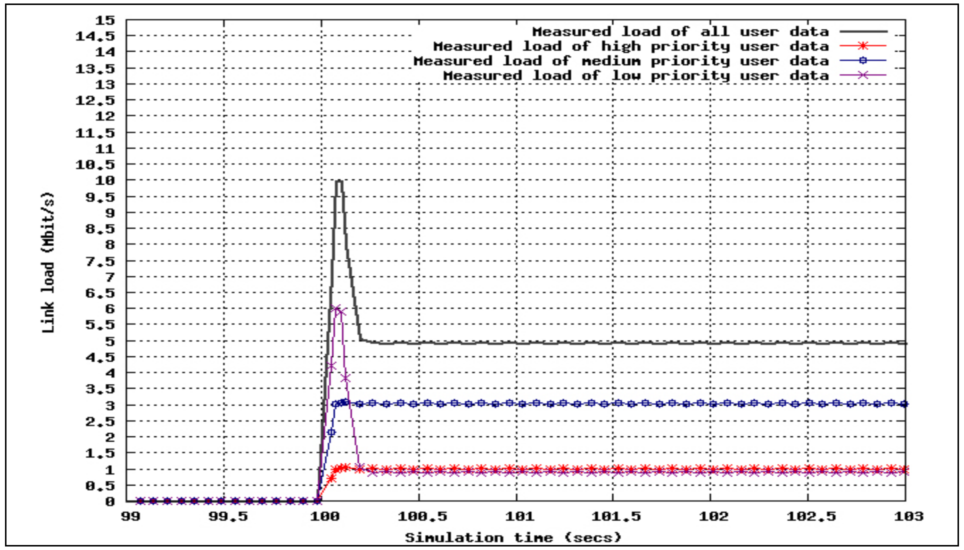


Figure C. 27 Link 3-1: Forward path, big-small flows

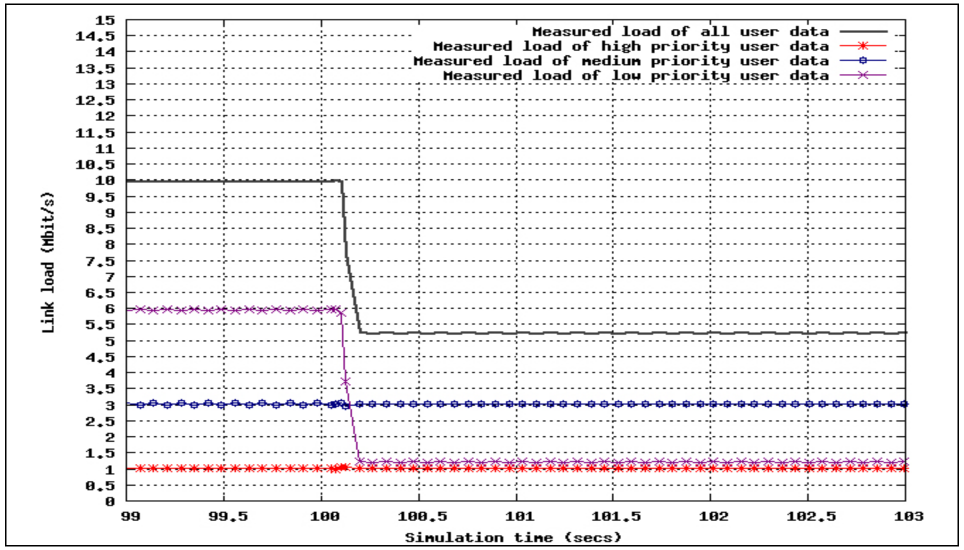


Figure C. 26 Link 0-1: Forward path, big-small flows, mechanism 2

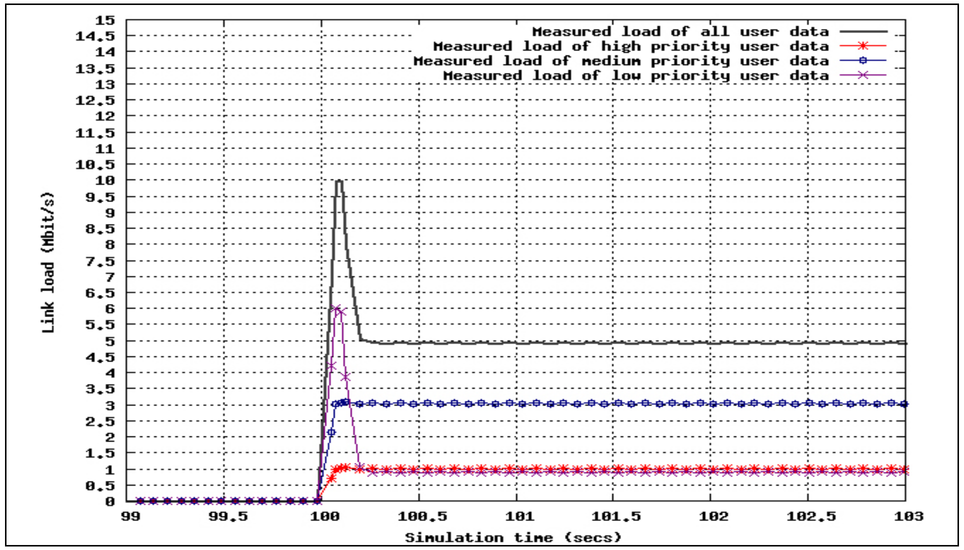


Figure C. 28 Link 3-1: Forward path, big-small flows, mechanism 2

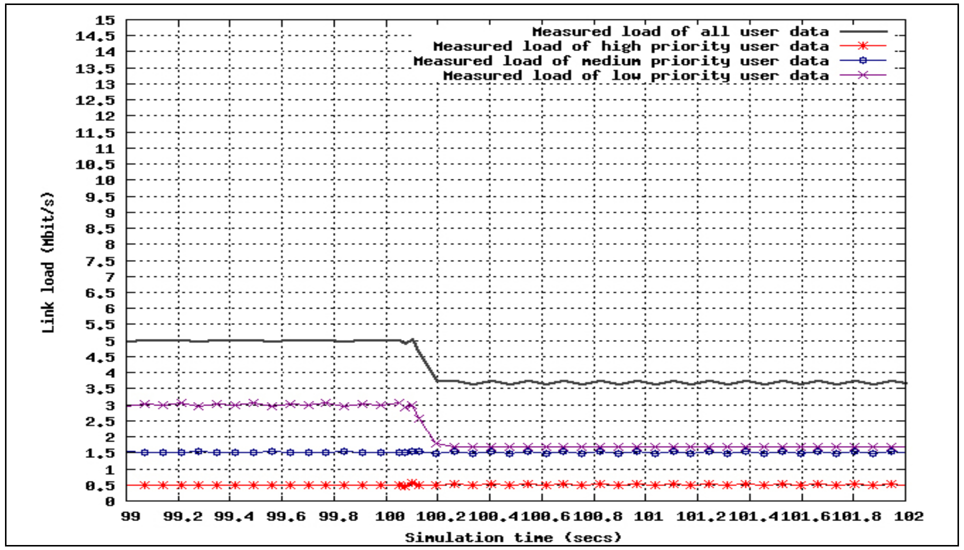


Figure C. 29 Link 4-2: Reverse path, big-small flows

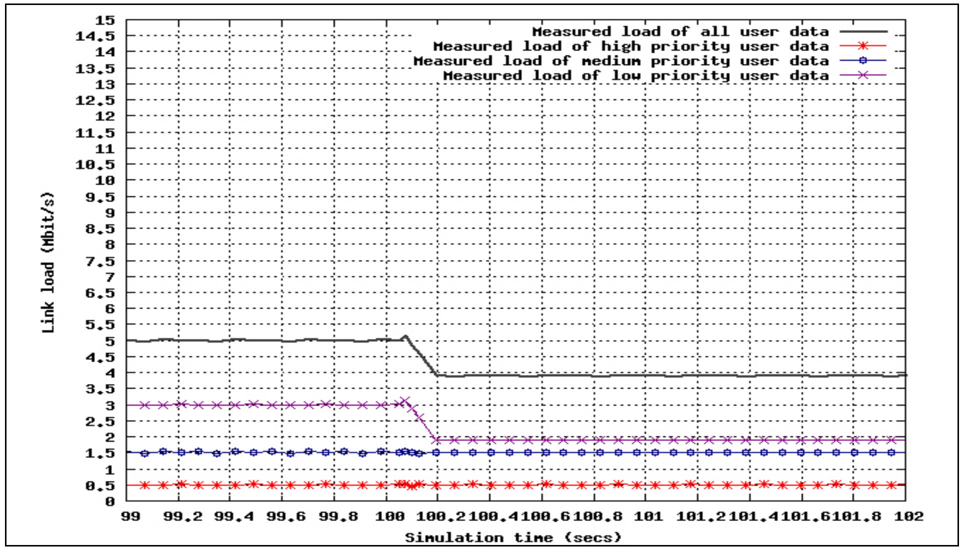


Figure C. 31 Link 5-2: Reverse path, big-small flows

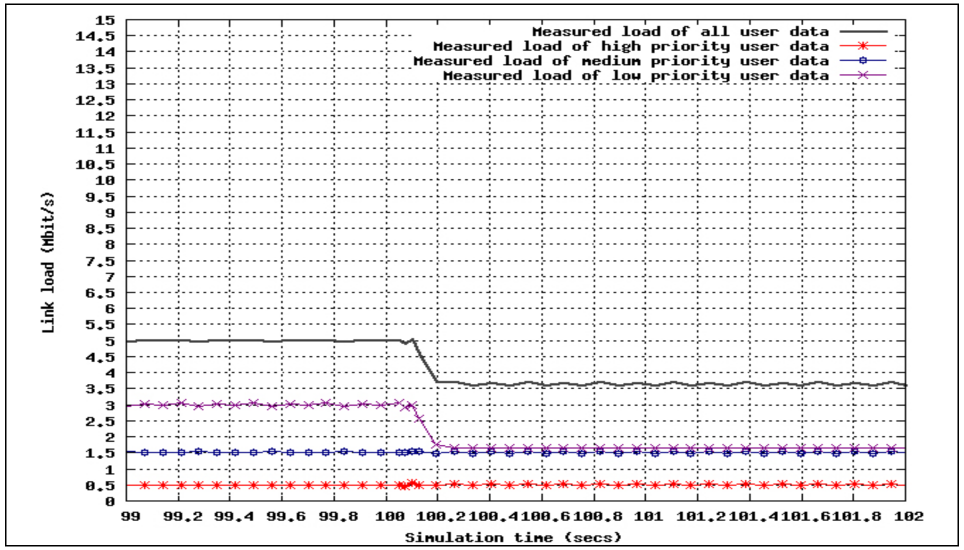


Figure C. 30 Link 4-2: Reverse path, big-small flows, mechanism 2

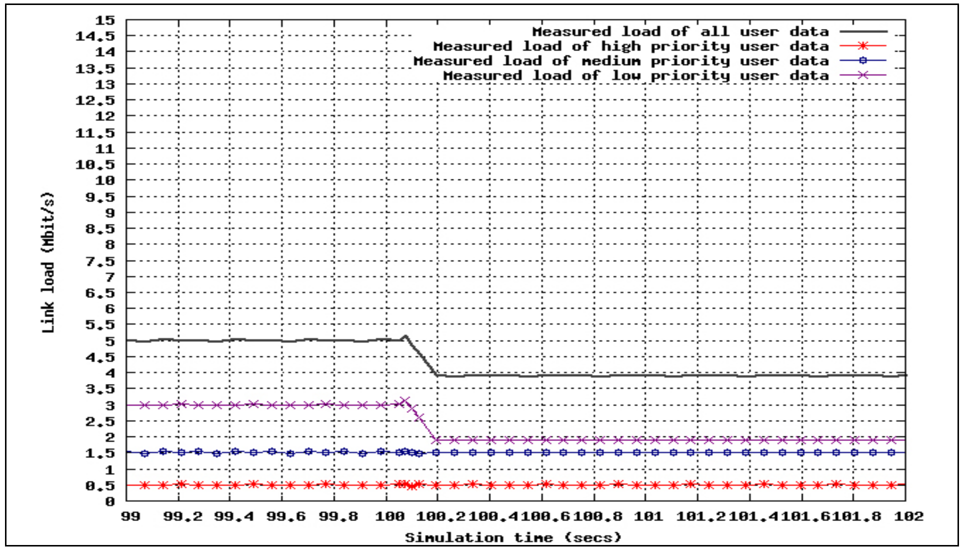


Figure C. 32 Link 5-2: Reverse path, big-small flows, mechanism 2

Appendix C.4: Two paths severe congestion

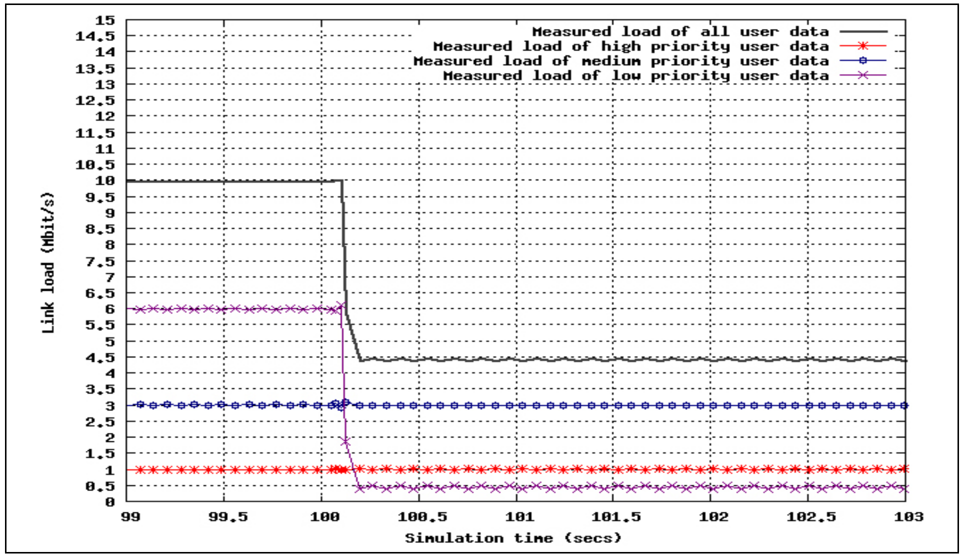


Figure C. 33 Link 0-1: Without_optimization – both paths congested

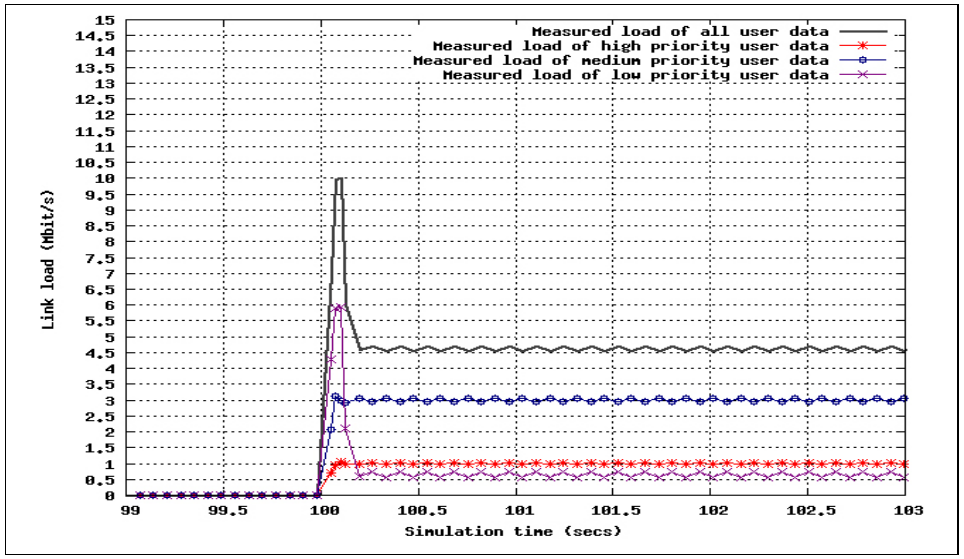


Figure C. 35 Link 3-1: Without_optimization – both paths congested

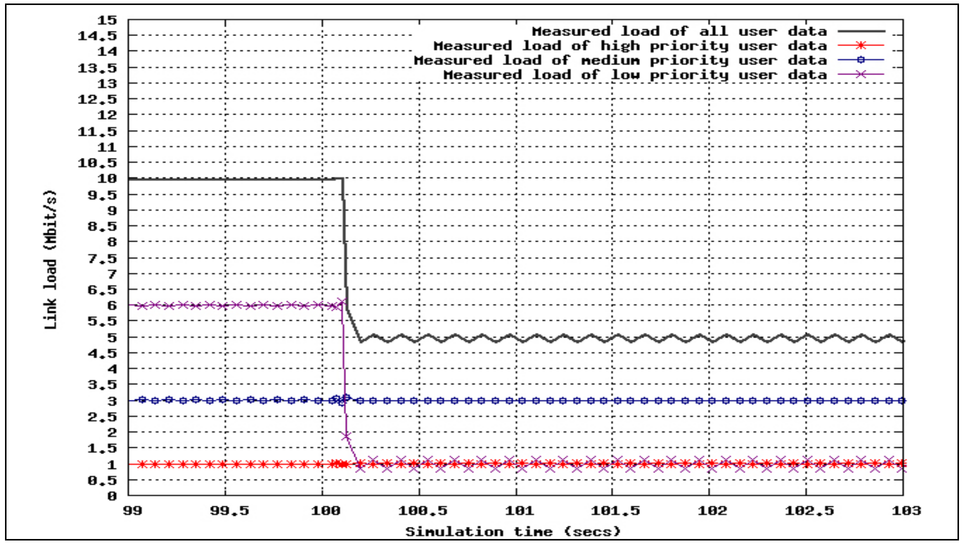


Figure C. 34 Link 0-1: With_optimization – both paths congested

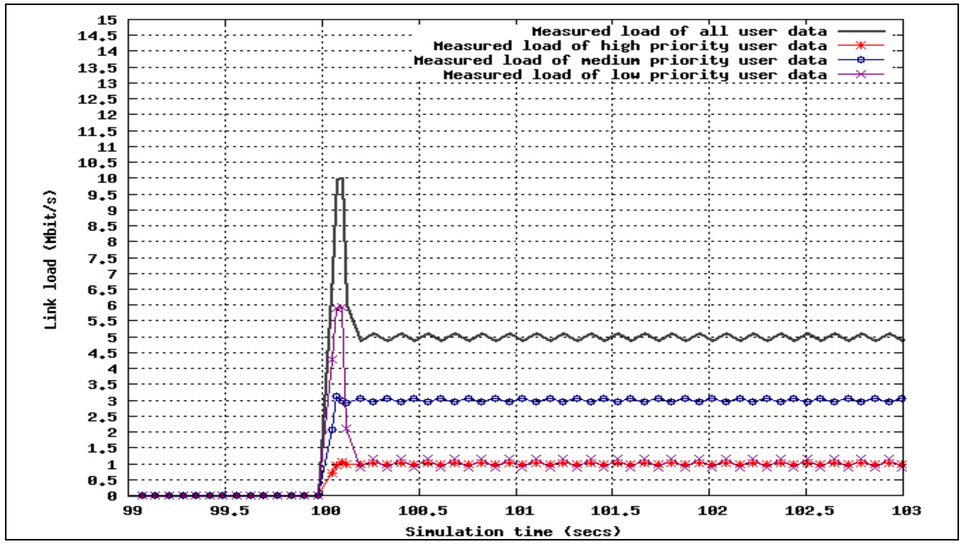


Figure C. 36 Link 3-1: With_optimization – both paths congested

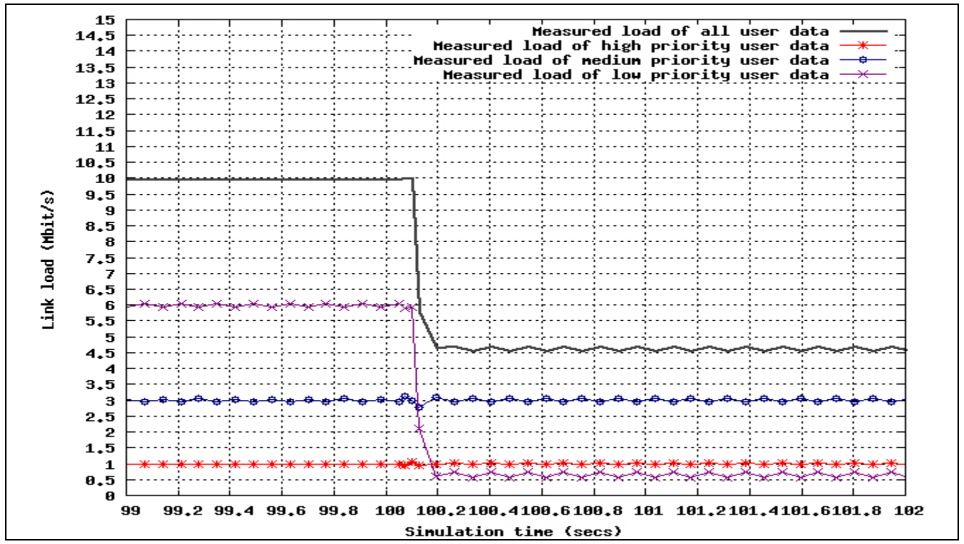


Figure C. 37 Link 4-2: Without_optimization – both paths congested

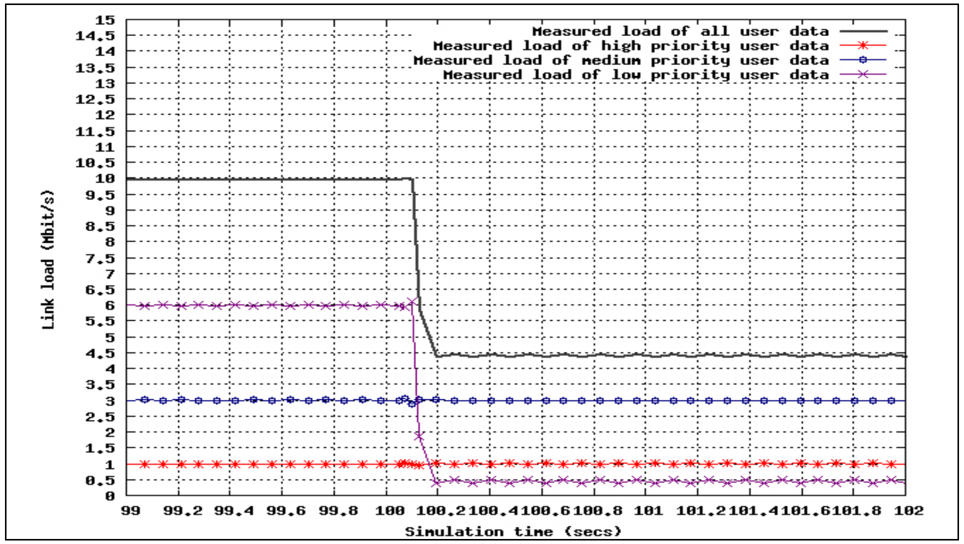


Figure C. 39 Link 5-2: Without_optimization – both paths congested

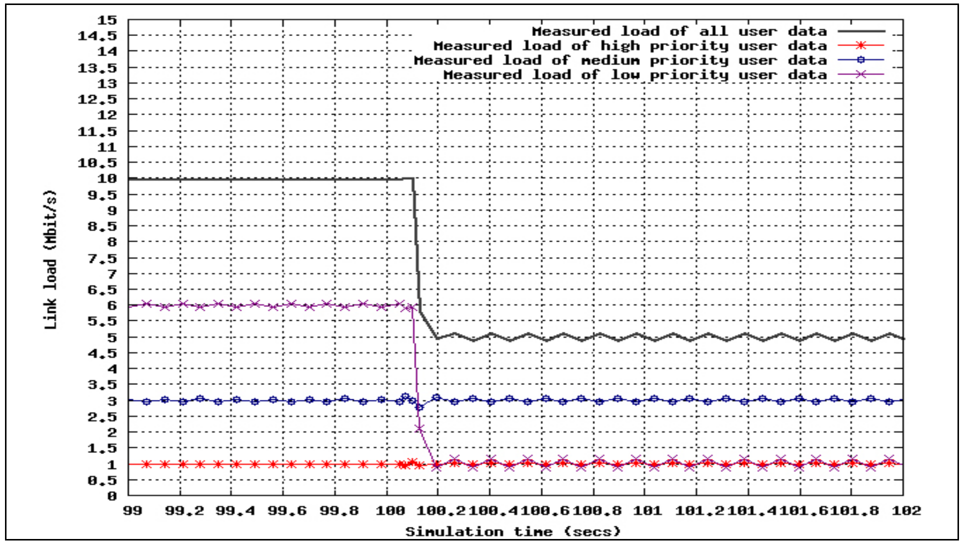


Figure C. 38 Link 4-2: With_optimization – both paths congested

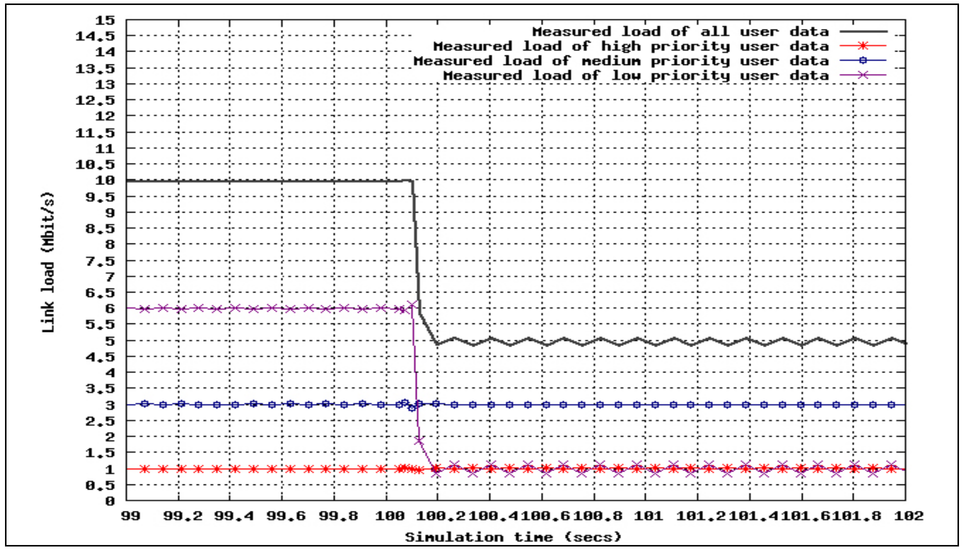


Figure C. 40 Link 5-2: With_optimization – both paths congested