Master's Thesis

EXTENDING DIFFSERV ARCHITECTURE: INTEGRATION OF *IDCC* AND *RMD* FRAMEWORK

Costas Djouvas

UNIVERSITY OF CYPRUS

DEPARTMENT OF COMPUTER SCIENCE

May 2003

UNIVERSITY OF CYPRUS
DEPARTMENT OF COMPUTER SCIENCE

Extending DiffServ Architecture: Integration of IDCC and RMD Framework

Costas Djouvas

Supervising Professor
Andreas Pitsillides

This thesis is submitted to the Graduate Faculty of University of Cyprus in partial fulfillment of the requirements for the Degree of Master of Science at Computer Science Department

May 2003

Acknowledgements

I would like to express my sincere appreciation to Professor Andreas Pitsillides of University of Cyprus, who introduced me to the area of computer networks including the subjects of this thesis. All the work conducted in the context of this thesis would not have been possible without his support and advice. Thanks are also due to PhD Candidate, Chrysostomos Chrysostomou, who gave me insightful assistance and helpful comments and feedback. Finally, I am deeply grateful to my fiancé, Stavroula, for her understanding and support during the elaboration of this thesis.

Abstract

The rapid growth of demands that is being observed during the last few years shows the unavailability of current solutions to provide QoS. Since it is expected that these demands will continue to increase rapidly, much research is conducted so that more effective QoS schemes will be proposed. Current research is focusing on two architectures, the IntServ [6] and the DiffServ [5]. Researchers also understand the need to control the congestion level in the network, since congestion is one of the reasons that current Internet provides poor QoS.

Recently, Resource Management in DiffServ (RMD) and Intergraded Dynamic Congestion Control (IDCC), two schemes, which aim at providing QoS, were proposed. Each one is focusing on a different issue that must be solved, in the context of QoS. RMD proposed in [1], extends the DiffServ principles with new ones, necessary to provide dynamic resource management and admission control in DiffServ domain. IDCC is controlling the traffic using information on the status of each queue in the network. It is based on a nonlinear model of the network that is generated using fluid flow consideration [16, 26, 27, 28]. Both schemes, RMD and IDCC, are using the basic principles of DiffServ architecture to aggregate the flow to classes.

The aim of this Master Thesis is to integrate RMD and IDCC, in a more effective scheme, which will use the same framework and will be able to handle both real time and elastic applications. The basic goal is to secure high utilization and bounded delay and loss. Furthermore, the network is to comply with the demands each traffic class sets.

In order to evaluate the proposed scheme, NS-2 simulator was used. The simulation results demonstrate the effectiveness of the integrated RMD - IDCC scheme, since it secures all the desired network behavior and especially the expected high utilization, and bounded delay and loss.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 IP Quality of Service

In the last few years, Internet users as well as Internet applications have dramatically increased. Consequently, the existing infrastructure of the Internet cannot afford the huge amount of demands, which have been raised. In addition, new applications require strict guarantees on delivery, delay and loss [16, 26, 27, 28]. It is also obvious, that in the near future, IP-based Radio Access Network (RAN) will be widely used [20]. These networks will mostly be used by real – time applications.

All the above led the researchers to the investigation and proposal of more effective mechanisms in order to provide QoS. New techniques, which will secure loss and delay bounds must be developed and employed.

Current research is focusing on two IP QoS architectures:

a. Integrated Services (IntServ) [6] and
b. Differentiated Services (DiffServ) [4].

The IntServ Architecture is based on the concepts of admission control and resource reservation. As such, it provides good granularity by signaling QoS requirements per individual flow to network elements [8, 16]. These elements, depending on the available resources, implement the required services. Due to its poor scalability, IntServ is effective only in the case of accessing networks where the flows are limited. RSVP [4] is one of the most well-known resource reservation protocols of IntServ architectures.

The DiffServ Architecture [5, 25] originated by the need for confronting the disadvantages of IntServ. It divides the network into different domains. When a packet enters a domain the Service Level Agreement (SLA) classifies and assigns it to one of the different aggregate classes; that is to a DiffServ Behavior Agreement (BA) [5]. This classification is achieved by the use of DS field of IP header. Six bits of DS are used as the Differentiated Service Code Point (DSCP). The SLA marks the packets according to their DCSP setting and assigns the appropriate PHB to each of them [5]. PHB refers to the expected behavior, concerning scheduling, queuing, policing or shaping, that each node should perform towards a packet belonging to a BA.

There are four available standards of PHB:

a. Default PHB: the packet receives the best-effort service [23].
b. Class-Selector PHB: is used to preserve compatibilities with previous IP schemes which are still used on the network [23].
c. Assured Forwarding PHB (AF PHB): It guaranties a minimum bandwidth and gives drop probability of a packet assigned to that PHB [15].
d. Expedited Forwarding PHB (EF PHB): It provides a guaranteed bandwidth service [18].

At the previous paragraphs two methods that are used in order to provide QoS were presented. The first was based on resource reservation and the second on classification of the traffic. Congestion is another issue that must be solved in order to ensure QoS. Since bandwidth is limited and the needs are increasing, available bandwidth will not be enough to correspond to these demands. As a result, routers buffers will be overflowed and congestion will occur more and more often.

In order to control the congestion several proposal have been made. When Congestion Control was introduced [19], end-to-end approaches were used. Such methodology is not considered effective any more. One of their main disadvantages is that these approaches can only detect congestion, after the congestion occurs, and are not able to avoid the congestion.

1.2 New Protocols for IP QoS

New proposal have been made to solve this problem. Routers are using Active Queue Management (AQM) [13] schemes. These schemes are using intelligent methods in order to prevent the congestion. When they realize that congestion may occur, they send a feedback message to the sources to cut down their rate, by marking the packets.

In the last few years Explicit Congestion Notification (ECN) [12] is adopted and used. ECN receives information from the AQM of the routers and sends feedback to the sources. This is achieved by using the Congestion Experience filed of the IP header [12].

Yet, Congestion Control and especially Congestion Avoidance are very difficult issues to be solved. This is because there must be tradeoff between low loss and high utilization (i.e. we can mark every packet in order to have no packet loss, but we will have very low utilization).

Recently, two proposals have been made to the direction of QoS, the Resource Management in DiffServ (RMD) [1] and the Integrated Dynamic Congestion Control (IDCC) [16]. Both protocols are based on DiffServ principles regarding traffic differentiation, but each one approaches the problem in a different way. RMD will provide resource reservation and IDCC will ensure congestion control.

RMD extends the DiffServ principles with new ones necessary to provide dynamic resource management and admission control in DiffServ domain [8, 14, 17]. In order to avoid the complexity of resource reservation schemes and to adopt the DiffServ principles, it was based on two design goals: The first was that some nodes should be stateless; the second referred to the need for providing certain QoS guarantees for each flow, even stateless at some nodes. These goals where met by proposing two types of nodes some of them stateless and some others stateful. Between the stateful nodes, which will support per flow state, there will be nodes using an aggregate reservation per traffic class.

With these characteristics, RMD is a simple and scalable protocol and can be used to handle real time applications.

IDCC is a dynamic congestion control scheme. Since it is based on DiffServ principles, it separates the traffic into three aggregate classes: the Premium, the Ordinary and the Best Effort. The Premium traffic requires strict guarantees on delay and loss, the Ordinary traffic can tolerate delay but cannot tolerate loss, and Best Effort offers no guarantee either on delay or on loss [28].

Premium traffic will be used for real time applications. Its rate can only be regulated at the connection phase. When the connection is established, the network has to offer services in accordance with the given guarantees. Ordinary traffic, on the other hand, will be used for elastic applications. It allows the network to regulate its flow. When Premium traffic needs more recourses or severe congestion is detected, IDCC reduces the Ordinary traffic by cutting the rate that the sources sent data. When there are available resources, it increases the Ordinary traffic by increasing the rate. Finally, Best Effort traffic uses resources, which are instantaneously unused.

1.3 Proposed Solutions

This Master thesis' aim is to integrate those two schemes, RMD and IDCC. The majority of the proposed solutions in the QoS concept are based on one of the concepts described above (IntServ, DiffServ and Congestion Control). The main objective is to use two schemes which are based on DiffServ principles, despite their different concept, and propose a new scheme which will be able to handle both real time and elastic applications effectively and efficiently. RMD will be responsible for resource reservation and IDCC for congestion control and queues management.

This thesis is organized as follows: In Chapter 2 and Chapter 3 a more detailed description of the two schemes used as background is given. The RMD framework is analyzed in Chapter 2 and IDCC scheme is described in Chapter 3. In Chapter 4 the reasons driven to the proposed scheme are argued, and the way the integration was implemented, in order to maximize the effectiveness of the proposed scheme, is presented. The analysis and the performance evaluation of RMD-IDCC are described in Chapter 5, where many numerical results supported by graphs are presented. Concluding remarks as well as open issues are presented in Chapter 6.

Chapter 2

RMD Framework Fundamental Concepts

2.1 Introduction

The Resource Management in DiffServ (RMD) represents a QoS framework that is based on standardized DiffServ principles for traffic management. It was developed to satisfy requirements regarding support for large amount of real time traffic, frequent use of costly leased transmission lines, support of frequent and low – delay mobility operations [14]. It is therefore a much more dynamic resource – management scheme that extends DiffServ Architecture with new principles, necessary to provide resource provisioning and admission control [8, 17].

The basic idea behind RMD was that it should (a) be stateless on some nodes and (b) meet certain QoS guarantee requirements for each flow, by associating each reservation with each flow. This was based on the assumptions that some nodes, denoted as “edge nodes”, are stateful, supporting “per – flow states”, whereas the nodes between the edge nodes, denoted as “interior nodes”, can only support one reservation state pre traffic class [39]. In practice, RMD framework introduces a measurement – based algorithm that uses the requested and the available resources as input, in order to update the aggregated reservation state per traffic class in the interior nodes [16]. As a result, it distinguishes the problem of a complex reservation within a domain from a simple reservation within a node [24]. This leads to the conclusion that RMD is an open, edge – to – edge framework, which interoperates with other resource management mechanisms. It is also applicable to a large number of different types of DiffServ networks and being a simple scheme, it provides good scaling properties and low cost of implementation [14].

2.2 Protocols

As mentioned above, the RMD framework, even though developed on the basis of DiffServ principles, maintains its scalability due to the use of a mechanism that separates the problem of complex reservation within a domain from the simple reservation within a node. In order for this to be achieved two types of protocols are defined:

a. Per Domain Reservation (PDR), which is implemented only at the edge nodes of the DiffServ domain and is responsible for resource management in the entire domain, and

b. Per Hop Reservation (PHR) protocols that are implemented at the interior nodes and are used for regulating resources on per hop basis.

The PDR protocol is either a newly defined protocol or one of the already existing protocols, such as RSVP [11], RSVP Aggregation [38], Simple Network Management Protocol (SNMP) [24], Common Open Policy Service (CORS) [7], etc. On the other hand, PHR is a newly defined protocol [14, 34]. It is possible that in a DiffServ domain using RMD framework, different protocols of either type are being used simultaneously. Further information for each protocol type is presented below:

2.2.1 Per Domain Reservation – PDR protocol:

Per Domain Reservation (PDR) is closely related with the DiffServ Per Domain Behavior (PDB), since it extends PDB's functioning (i.e. the description of the behavior experienced by a set of packets through their crossing of a DiffServ domain [24]), with dynamic resource management. As it is used for resource management in the entire domain, PDR deals with the management of reservation requests; that is it decides whether to admit or reject them.

External end – to – end reservation or "QoS requests" arrive at the edge nodes of a DiffServ domain, since PDR is only part of their functionality and the interior nodes cannot be PDR aware. At the edge nodes various types of protocols, such as RSVP, RSVP Aggregation, COPS, etc, are used to notify the requests. The PDR aware node interprets the parameters of a "QoS request" and maps it into the appropriate DSCPs or PHBs that are used at the specific DiffServ domain [14]. The decision for the admission or rejection of a QoS request is mostly based on the results of the Per Hop Reservation (PHR) in the domain. So, it can be conclusively claimed that PDR establishes a linkage between the external resource reservation scheme and the PHR protocol [34, 39]. As a result, the external QoS reservation request flow identifier (ID), which determines a state that will be maintained at the edge nodes, is related to the internal PHR resource reservation request, thus a bi-directional resource reservation request is being supported. The flow ID can take different formats depending on the PDR type [16].

No matter if the PDR protocol is a newly developed or an already existing one, it has a number of functions which implements partly or as a whole, depending on the properties of the network where the application of RMD framework takes place. The functions listed below are explicitly analyzed in [40]:

1. Mapping of external QoS request to a DiffServ Code Point (DSCP).
2. Admission control and/or resource reservation within a domain.
3. Maintenance of flow identifier (ID) and reservation state per flow, e.g. by using soft state refresh.
4. Notification of the ingress node IP address to the egress node.
5. Notification of lost signaling messages (PHR and PDR) occurred in the communication path from the ingress to the egress nodes.
6. Notification of resource availability in all the nodes located in the communication path from the ingress to the egress nodes.
7. Initiation of a bi-directional resource reservation request.

2.2.2 Per Hop Reservation –PHR Protocol

The Per Hop Reservation (PHR) protocols focus on the management of resources in each node within the DiffServ domain, i.e. per DiffServ Code Point (DSCP). It is greatly related to the DiffServ Per Hop Behavior (PHB), that is “the externally observable behavior applied at any set of packets with the same DSCP crossing a link in a particular direction” [14]. PHR extends the PHB with dynamic resource management. Yet, any node using PHR protocols only adds one single reservation state to each traffic class, whereas no per – flow information is stored and no per – flow packet scheduling takes place. As a result, the differentiation between individual traffic packets is not possible. This reduction of requirements on the functionality of the nodes, on which this protocol type is implemented, leads to high scalability, especially in large DiffServ domains.

Two different PHR groups are defined in the context of RMD framework [8, 14, 16, 34, 38]:

a. The reservation – based PHR group
b. The measurement – based PHR group.

The *Reservation – based PHR group* consists of protocols that support the resource reservation per PHB in each node within the communication edge – to – edge path [8]. The nodes using this group of protocols combine the reservation soft state with the explicit release principles, so that one reservation state per PHB will be maintained. As a result, any reserved resources that are not regularly refreshed can be released. In addition, reserved resources can be released in the case they receive a predefined release message [34].

Resource units, which may be based in a single parameter such as bandwidth or on more complex parameters, are being used for signaling the reservation requests. In either case, a statistically and dynamically configurable threshold of maximum available resource units will be defined for each PHB.

Only one reservation – based PHR protocol is currently defined, denoted as Resource Management in DiffServ (RMD) on DemAnd (RODA) PHR protocol [35]. It is a unicast, edge – to – edge protocol designed for a single domain [16]. More information for RODA PHR protocol is given in upcoming section.

The *Measurement – based Admission Control (MBAC) PHR group* consists of protocols, which check and signal the availability of resources in the communication path edge – to – edge before any QoS requests are admitted. The check is done by the use of measurement based on the average real time traffic load. Yet, no reservation state is maintained in the nodes.

Compared to Reservation – based PHR protocols, the MBAC PHR group is optimized in the sense that it can be more efficient as far as resource utilization is concerned. Also, state maintenance is greatly limited, since only information regarding the measured traffic load associated to the PHB and the maximum allowable traffic load per PHB is maintained [1].

Currently, only one MBAC PHR protocol is specified. This is RMD measurement – based admission control (RIMA) PHR protocol, which operates on per hop basis on both edge and interior nodes in the context of an edge – to – edge DiffServ domain [16]. Further information regarding RIMA PHR protocol is presented below.

Both reservation – based and measurement – based PHR protocols aim at extreme simplicity, low cost of implementation and good scalability. As in the case of PDR protocols, there is a set of functions PHR protocols execute, either partly or as a whole, depending on the demands of the network where the RMD framework is used [40, 42]:

1. Admission control and / or resource reservation within a node.
2. Management of a reservation state per PHB by using a combination of reservation soft state and explicit release principles.
3. Storing of a pre – configured threshold value on maximum allowable resource units per PHB.
4. Adaptation to load sharing, which allows interior nodes to take advantage of multiple routes to the same destination by sending via some or all of these available routes.
5. Severe congestion notification, as a result of route changes or a link failure. The PHR has to notify the edges when this occurs.
6. Transport of transparent messages. The PHR protocol may encapsulate and transport PDR messages from an ingress node to an egress node.

2.2.3 RODA and RIMA

As mentioned above, only two PHR protocols are defined thus far. This is RODA, a reservation – based PHR protocol, and RIMA, a measurement – based PHR protocol. These two are briefly presented below.

RODA PHR protocol is used on a per hop basis on all nodes located in a DiffServ domain. It is applicable to domains, which may use either IPv4 [9] or IPv6 [10]. RODA is responsible for the measurement of traffic load [42]. It also performs the entire set of functions a PHR protocol may implement. When RODA PHR protocol is being used a resource reservation state is being established, maintained or released at each node by the means of an efficient algorithm. Furthermore, it informs all the nodes in a domain of severe congestion situations, which may derive from a route change, a link failure or a long period of congestion. Nodes also store a predefined threshold, i.e. a maximum number of allowable resource units per PHB. Finally, RODA PHR protocol supports the transportation of PDR messages sent from an ingress to an egress node [38].

There are two protocol messages RODA PHR protocol extensively uses:

a. "PHR_Resource_Request", which enables the initiation or update of the PHB reservation state in all nodes in the communication path edge – to – edge after an external QoS request. Yet, this message cannot refresh an already existing resource reservation state.
b. "PHR_Refresh_Update", which enables refreshing the PHB reservation state in all nodes in the communication path between the ingress and egress nodes, after a successfully proceeded resource reservation request [14].

A "PHR_Refresh_Update" message is to be processed in higher priority than a "PHR_Resource_Request" message. Also, a "PHR_Refresh_Update" is always processed, whereas a "PHR_Resource_Request" message can be processed only if its DSCP is identified and the requested resource units are added to the total amount of reserved units related to the DSCP. In other case, the message is marked and forwarded untouched.

RMD measurement-based admission control (RIMA) PHR protocol extends the functionality of Per Hop Behavior (PHB) by means of MBAC features. Aiming at simplicity, low cost of implementation and high scalability, it implements the following functions:

a. RIMA supports the specification of the number of resource units allocated to different DSCP classes that are in use, at each node at per hop basis, and
b. It provides the means to inform all nodes in the communication path edge-to-edge of the existence of severe congestion situation on a per hop basis [16].

In RIMA only one PHR message is used, that is “PHR_Resource_Request”. The message follows the same path as the traffic data and is used for determination of current traffic load status for each PHR. A “PHR_Resource_Request” message is being generated for each incoming flow and it signals only the specifically requested resource units [37]. RIMA PHR protocol is to be used so as the request will be admitted or rejected.

2.3 RMD Signaling Messages

The RMD signaling messages are divided into two categories: PHR and PDR protocol messages. The two categories are not fully independent, since specific PDR messages can be encapsulated into a PHR message. These messages are the “PDR_Reservation_Request”, “PDR_Refresh_Request” and “PDR_Release_Request”. Of course, these messages can alternatively be sent as separate messages [37].

Protocol	Signaling Message	Signaling Message Description
RODA/RIMA PHR	**"PHR_Resource_Request"**	Initiate the PHB reservation state on all nodes located on the communication path between the ingress and egress nodes according to the external reservation request.
	"PHR_Refresh_Update"	Refreshes the PHB reservation soft state on all nodes located on the communication path between the ingress and egress nodes according to the resource reservation request that was successfully processed by the PHR functionality during a previous refresh period. If this reservation state does not receive a "PHR_Refresh_Update" message within a refresh period, reserved resources associated to this PHR message will be released automatically. Note that this message is used only for the RODA protocol.
	"PHR_Release_Request"	Explicitly releases the reserved resources for a particular flow from a PHB reservation state. Any node that receives this message will release the requested resources associated with it, by subtracting the amount of PHR requested resources from the total reserved amount of resources stored in the PHB reservation state. Note that this message is used only for the RODA protocol.
PDR	"**PDR_Reservation_Request**"	Initiates or updates the PDR state in the egress. It is generated by ingress node.
	"**PDR_Refresh_Request**"	Refreshes the PDR states located in the egress. It is generated by the ingress node.
	"**PDR_Release_Request**"	Explicitly release the PDR state. It is generated by the ingress node. Applied only when the PDR state does not use a reservation soft state principle.
	"**PDR_Reservation_Report"**	Reports that a "PHR_Resource_Request"/"PDR_Reservation_Request" has been received and that the request has been admitted or rejected. ". It is sent by the egress to the ingress node.
	"**PDR_Refresh_Report**"	Reports that a "PHR_Refresh_Update"/"PDR_Refresh_Request" message has been received and has been processed. It is sent by the egress to the ingress node.
	"**PDR_Congestion_Report**"	Used for severe congestion notification and is sent by egress to ingress. These PDR report messages are only used when either the "greedy marking" or "proportional marking" severe congestion notification procedures are used.
	"**PDR_Request_Info**"	Contains the information that is required by the egress node to associate the PHR signaling message that encapsulated this PDR message to for example the PDR flow ID and/or the IP address of the ingress node. It is generated by the ingress node.

Table 2.1 RODA PHR and PDR signalling messages [34]

2.4 Functional Operation

An RMD – enabled DiffServ domain, whose operation is based on the interoperation of PHR and PDR protocol principles, may be engaged in:

a. Normal operation, which refers to the successful reservation through the communication path in a DiffServ domain, and
b. Fault handling, which refers to the effort to deal with severe congestion, caused by a link failure or a route change.

2.4.1 Normal Operation

The normal operation of a DiffServ domain initiates on the arrival of a QoS request at the ingress node. The node, using PDR protocol, classifies it into the appropriate PHB, calculates the requested resources units and creates a PDR state, which will be associated with a flow specification ID. The request may be satisfied locally. In such situation, a “PHR_Resource_Request” signaling message will be created, which will encapsulate a “PDR_Reservation_Request” signaling message containing information, that the ingress node will proceed [39].

The interior nodes must recognize the DSCP type of the PHR signaling messages and, if possible, add the requested amount to the total amount of reserved resource units, thus reserve the requested resource units. After a successful reservation the egress node receives the “PDR_Reservation_Report”. The egress node is also responsible for informing the external source of the successful reservation, so that traffic will start being sent [16].

In the case of refreshing or updating of already reserved resource units, a “PDR_Refresh_Request” will be encapsulated. There is also the case of releasing reserved resources in any node. The “PHR_Release_Request” message will be generated. Furthermore, a lack of available resources is possible. In such case the “PHR_Resource_Request” is marked and the reservation request cannot be admitted.

2.4.2 Fault handling

Undesirable situations, such as route changes or link failures, are often phenomena during the operation of a network. Such situations may lead to the change of the route of the traffic resulting in overloaded nodes and severe congestion, which must be controlled very fast. Therefore, the ingress node must implement predefined policy.

The interior nodes will be the first to detect severe congestion and activate the severe congestion state, when and if one of the following occurs:

- A link failure.
- The volume of the traffic increases suddenly, which might mean that a route change and a severe congestion occurred.
- The number of dropped data packets is higher than the predefined maximum number.
- The number of resource units requested by PHR refresh signaling message is higher than the number of units previously refreshed [8].

Since the interior nodes do not maintain any flow information, the ingress node should be notified through the egress node. This will be achieved by marking all or a proportion of the packets [39] passing through a congested interior node or even by marking the PHR signaling messages [41]. The egress node processes the severe congestion information and forwards it to the ingress node, which in turn, processes the information and undertakes certain actions. The ingress nodes might be predefined to block new traffic and either re-initiate all on-going flows or terminate some of them [8].

Chapter 3

IDCC

3.1 Introduction

One of the most important issues, concerning network functionality, which must be solved so that the QoS requirements would be satisfied, is the Congestion Control. Several attempts have been made to this direction and a lot of proposed solutions have been announced [3, 21, 22, 29, 30, 33]. Seeing the advantages of non-linear approaches and the need to adapt the differentiated services architecture, a generic scheme called Integrated Dynamic Congestion Control (IDCC) was proposed [26, 27, 28].

3.2 The model

The IDCC scheme handles three different classes of services, each one with different characteristics, concerning guarantees on delivery, delay and loss. The reference model is assumed to be a generic K input, K output port, output buffered router (Fig 3.1). Each output port has a number of physical or logical queues, one for each traffic [26, 27, 28]. Each output port has its dedicated congestion control controller and buffer space, where the packets that arrived are forwarded, depending on the class they belong to.

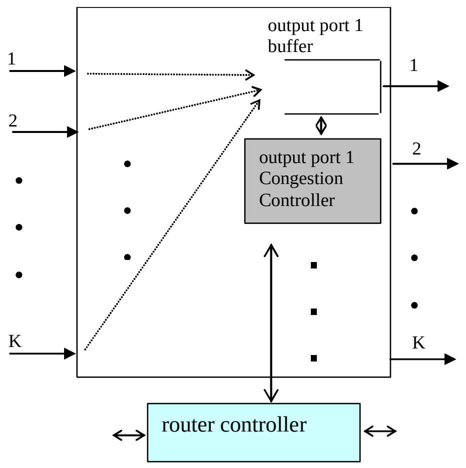
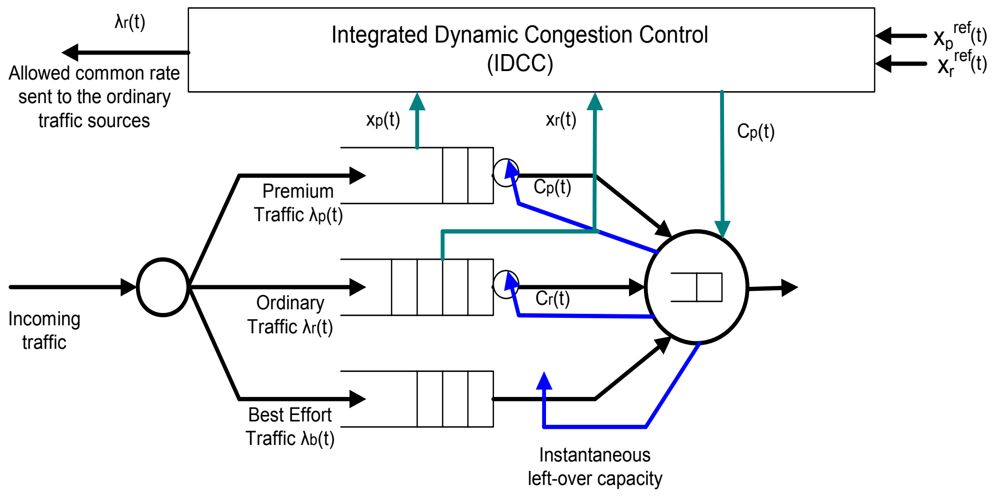


Figure 3.1 Generic output buffered K input-output router

IDCC, as mentioned above, has three different classes and consequently three output ports, each one with its dedicated buffer space. Each packet that arrives to the router may be classified as Premium Service, Ordinary Service, or Best Effort Service [Fig 3.2].

Premium service is the service that requires strict guarantees on delivery, and delay and loss bound. Real Time applications are classified to this service since it offers guarantees necessary for that kind of applications. Before the admission to the network, it is possible for the rate, by which the real – time application will send data, to be regulated. After the admission, the network must offer the decided rate until the end of the transition. IDCC cannot guarantee that there will be available bandwidth, since its role is not to negotiate and reserve bandwidth in order to reserve the requested resource units. This can be done by a protocol that reserve bandwidth, like RMD, which offers a framework to fill this gap. What IDCC can do for Premium service is to keep the Premium traffic queue to a desired value close to the reference value, which is set by the designer. Keeping the queue to that value, IDCC can indirectly guarantee delay and loss bound. This is done by giving capacity to Premium traffic up to the physical server limit, which is achieved by regulating the recourses used by Ordinary traffic.



λr(t)
Allowed common rate sent to the ordinary traffic sources
Integrated Dynamic Congestion Control (IDCC)
$x_p^{ref}(t)$
$x_r^{ref}(t)$
xp(t)
xr(t)
Cp(t)
Premium Traffic λp(t)
Cp(t)
Incoming traffic
Ordinary Traffic λr(t)
Cr(t)
Best Effort Traffic λb(t)
Instantaneous left-over capacity

Figure 3.2 Implementation of control strategy at each router output port

On the other hand, Ordinary traffic cannot tolerate loss of packets, but can tolerate queue delays. Since it can tolerate queue delays, we can decrease or increase the flow by changing the rate each Ordinary queue sends packets to the network, depending on the traffic load of the moment. The calculation of the rate is done by the Ordinary Traffic Controllers, which are installed to every router. Using only local information, like Premium and Ordinary queues length, the calculation is done. The new rate calculated by each node attempts to stabilize the queue length at each node. When the packet leaves the router, a field is updated in order to send feedback to each source. When the sources receive feedback, they change their rate. By regulating the rate, the queues length is kept to the desired value and so congestion is avoided as well as queue space for Premium traffic packets is secured.

Finally, Best Effort Service offers no guarantees on delivery, delay and loss. If there are available resources being used neither by Premium nor by Ordinary traffic, then they can be used instantaneous for Best Effort traffic.

The original algorithm has suffered a number of enhancements. Some more enhancements of the IDCC are under investigation [16].

The Premium control strategy and Ordinary Traffic control strategy are presented in [27] and in Appendix A.

Chapter 4

The proposed model: RMD-IDCC

4.1 Introduction

The limitations of the Integrated Services Architectures led to the need for developing new Differentiated Services Architectures, which will provide Quality of Service (QoS) on the internet network. Two of the proposed protocols, RMD and IDCC, which are based on different approaches, have been described in the previous chapters. A detailed analysis of these two models, can lead to the conclusion that they can be integrated, each solving the problems the other causes. The objective of this thesis is to integrate RMD and IDCC protocols into a more efficient and effective protocol, RMD-IDCC.

RMD protocol manages Real Time Traffic. It is very effective when the rate of the sources is firm and steady. This is achieved by extending the Differentiated Services Architecture with new admission control and resource reservation concepts in a scalable way [39]. Nevertheless, it is not adaptive since it deals with Constant Bit Rate (CBR) traffic, so it cannot adjust the rate of any source that would result in high utilization at any time. Another problem that comes up is the lack of a definite way of dealing with congestion problems. RMD only uses a bit, which is activated when the interior nodes detect severe congestion. Yet, none of the proposed solutions is satisfactorily evaluated.

On the other hand, IDCC satisfies Quality of Service (QoS) requirements through high utilization of the available resources and congestion control. This is achieved by regulating the rate Ordinary packets are being sent. The major problem of IDCC is its incapability to cope, when the aggregated Premium traffic rate exceeds the link capacity.

Conclusively, RMD is weak at achieving high utilization and unable to handle elastic applications effectively. IDCC cannot manage real time applications, since it does not guarantee the availability of the necessary bandwidth. The scheme proposed in this thesis can eliminate the weaknesses of both protocols and deal with both real time and elastic applications.

4.2 Integration of RMD and IDCC

Taking the two protocols and their strengths and weaknesses into consideration, my aim was to define how the integration would be implemented, in order to be mostly effective and robust. RMD offers the infrastructure for resource reservation, whereas IDCC classifies service as Premium, Ordinary and Best Effort, based on the kind of the application and the priority that should be given.

Before doing so, two issues should be investigated:

- Which of the protocols should be used as the base for the integration, and

- For which traffics should there be reservation.

As the first issue is concerned, two were the choices: (a) IDCC would be the base and each service would use RMD, and (b) RMD would be the base and each node would use IDCC for the regulation of traffic. As for the second issue, we had the choice of reserving bandwidth for the Premium service or for both Premium and Ordinary service.

My decision was to use the second approach, i.e. RMD would be the base and queue management at each node would be performed by IDCC. There were a number of reasons that led to this decision:

a. Proposed PHR protocols (RODA, RIMA) header would be used, since there are sixteen (16) available bits in IPv4 and thirty two (32) in IPv6. These fields will be used for sending feedback to the sources of Ordinary traffic.
b. Admission control procedures must precede resource management, because there must be available resources, so that the admission request will be satisfied.

It was also decided that bandwidth reservation would only occur in the case of Premium traffic, based on the fact that Premium sources send packets using Constant Bit Rate (CBR). Therefore, the amount of the resources each source needs can be estimated.

Conclusively, the integration of RMD and IDCC will result in securing resources of Premium class and, at the same time, guaranteeing high utilization. Reservation for Ordinary traffic is not feasible, since the demand of resources varies as the rate changes. Yet, RMD can limit the number of Ordinary traffic sources, in order to guarantee a minimum rate for each source.

4.3 The model

In this section the RMD-IDCC model will be presented. Firstly, it must be mentioned that each packet uses RODA PHR header in order to traverse each RMD domain. When a packet arrives to the ingress node it is classified to the one of the three classes, proposed by IDCC: Premium, Ordinary or Best Effort. If the packet is classified as Premium, it must be checked whether it contains data or is a “QoS Request” packet. In the first case, the packet is forwarded to IDCC Premium queue, which is responsible for forwarding it to the next node with bounded delay and without being dropped. In the second case, RODA PHR protocol is activated in order to request admission.

In the case the packet is classified as Ordinary, it is forwarded to IDCC Ordinary queue. IDCC is responsible for calculating the rate by which the source must send data. After doing so, the RODA PHR header, in which the rate is inscribed, is updated and the packet is forwarded to the next node.

In the last case, where the packet is classified as best effort, it is forwarded to best effort queue. If there is any bandwidth that is not used by Premium or/and Ordinary traffic, IDCC forwards best effort packets to the next node. The packet finally reaches the egress node, which is responsible for informing ingress node. In turn, the ingress node sends feedback to the external source.

Figure 4.1 shows the proposed RMD – IDCC model:

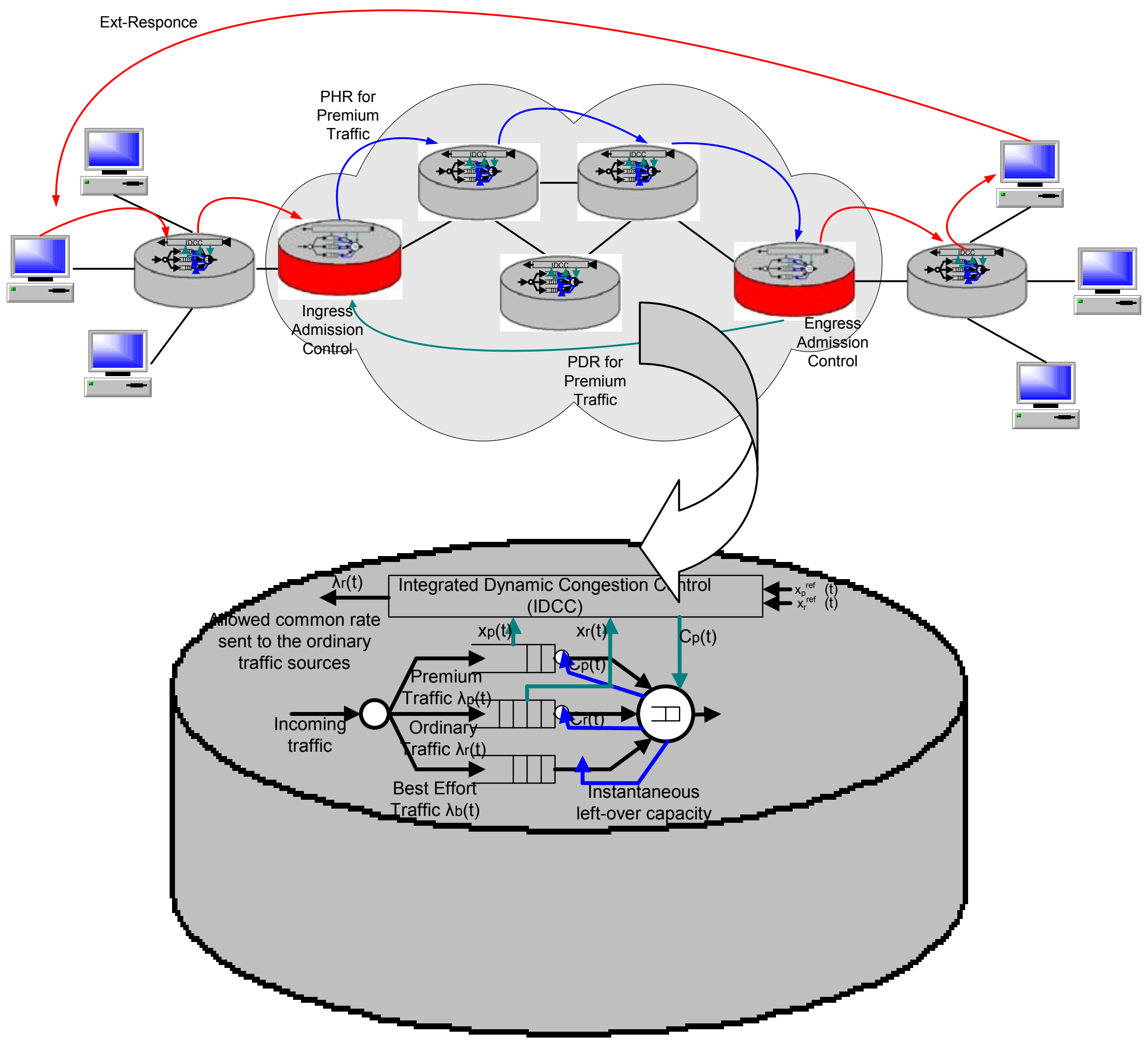

Ext-Responce
PHR for Premium Traffic
Ingress Admission Control
PDR for Premium Traffic
Engress Admission Control
IDCC
λr(t)
Integrated Dynamic Congestion Control (IDCC)
Allowed common rate sent to the ordinary traffic sources
xp(t)
xr(t)
Cp(t)
Premium Traffic λp(t)
Ordinary Traffic λr(t)
Best Effort Traffic λb(t)
Incoming traffic
Cr(t)
Instantaneous left-over capacity
$x_p^{ref}(t)$
$x_r^{ref}(t)$

Figure 4.1 RMD – IDCC model

4.4. Advantages

Most of the recent attempts to develop Quality of Service using DiffServ Architectures focus either on congestion control techniques or resource reservation. IDCC belongs to the first category, whereas RMD to the second. RMD-IDCC aims to propose a model encapsulating both approaches. It is obvious that real time and elastic applications will operate effectively, since real time applications will be handled by RMD and elastic applications by IDCC.

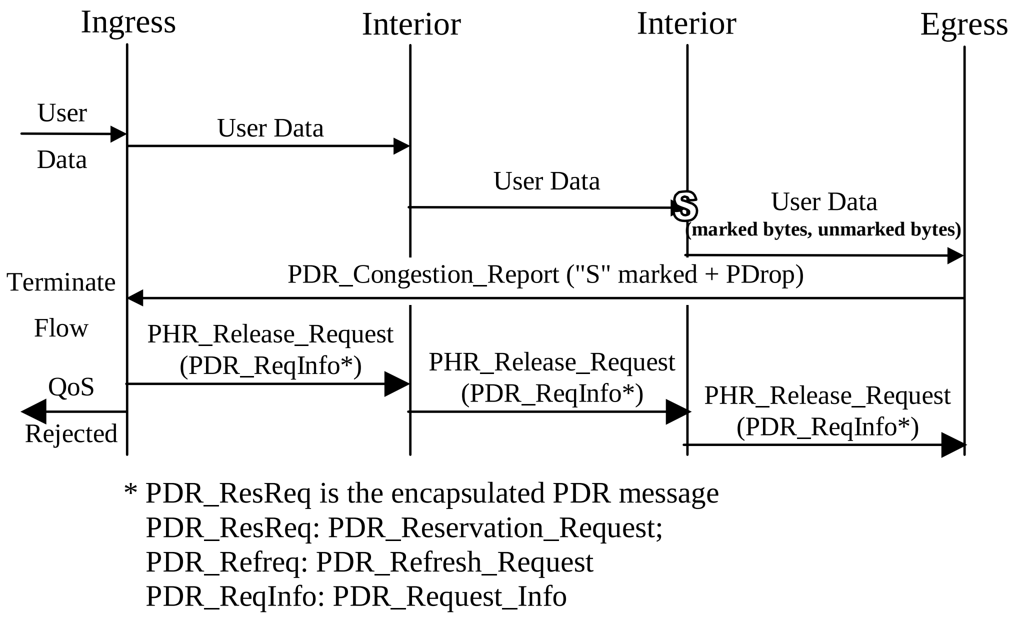

Ingress
Interior
Interior
Egress
User Data
User Data
User Data
S
User Data
(marked bytes, unmarked bytes)
Terminate Flow
PDR_Congestion_Report ("S" marked + PDrop)
PHR_Release_Request (PDR_ReqInfo*)
PHR_Release_Request (PDR_ReqInfo*)
PHR_Release_Request (PDR_ReqInfo*)
QoS Rejected
* PDR_ResReq is the encapsulated PDR message
PDR_ResReq: PDR_Reservation_Request;
PDR_Refreq: PDR_Refresh_Request
PDR_ReqInfo: PDR_Request_Info

Figure 4.2. RMD functional operation in case of severe congestion [39]

It is expected that fault handling will be improved, as well. RMD adapts to such situations by finding alternative routes and closing connections [39], possibly leading to severe congestion in the new path (Fig. 4.2), which must be controlled very fast. In this case, the proposed model will reduce the bandwidth used by Ordinary traffic. This will be done, because IDCC will be activated updating the rate Ordinary sources send data. The RMD will be activated and start closing connections, only if the severe congestion insists. The amount of time for this to be achieved is very little, since IDCC is proved to have very quick response to such situations.

Chapter 5

Performance Evaluation

5.1 Introduction

In this section, evaluation of the proposed protocol is presented. Five different simulation scenarios are presented each testing a different parameter of its performance.

Each topology has a bottleneck link. At the simulation scenarios 1, 3 and 4 the bottleneck link is set to 50Mbps and at the simulation scenario 2 to 10Mbps. The controller parameters were set as follows:

- $x_p^{ref} = 50$ (Preferred size of Premium Queue)
- $x_r^{ref} = 500$ (Preferred size of Ordinary Queue)
- $a_r = 1.5$

- $a_p = 1000$
- Sampling Interval = 0.1
- Percentage of link capacity used by Premium Traffic = 90%
- Percentage of link capacity used by Ordinary Traffic = 10%

At every simulation the following measurements are conducted, concerning the bottleneck link:

a. Common Rate

b. Throughput

c. Ordinary Queue Length

d. Premium Queue Length

e. Losses for Ordinary Traffic Packets

f. Losses for Premium Traffic Packets

5.2 Simulation Scenario 1

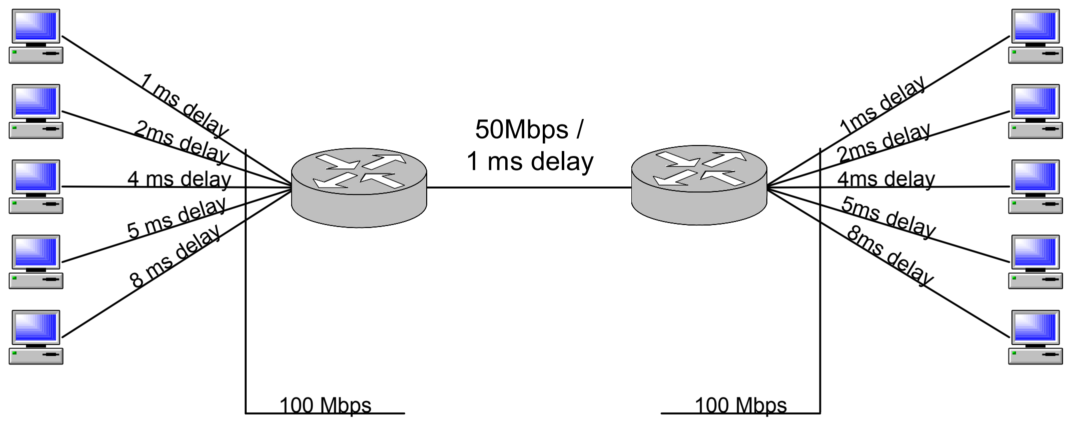


Figure 5.1. Topology of Simulation Scenario 1

The first simulation scenario was a very simple one and aimed at taking the first feedback for the integration of RMD and IDCC. As it seems by the figure 5.1

the topology is simple. There are five sources and five destinations connected by two routers. It was used for the evaluation of three different scenarios:

a. One source is sending Premium and four are sending Ordinary traffic,

b. Two sources are sending Premium and three are sending Ordinary traffic,

c. Four sources are sending Premium and one are sending Ordinary traffic,

Then, five more sources and destinations were added, which were sending TCP/FTP traffic. This topology is shown in figure 5.2. One of the existing sources is sending Premium and the rest four are sending Ordinary traffic.

Premium traffic sources send packets at a rate of 250 packets / sec. The simulation time were 100 seconds. The results of this simulation scenario are shown in figures 5.3 – 5.8.

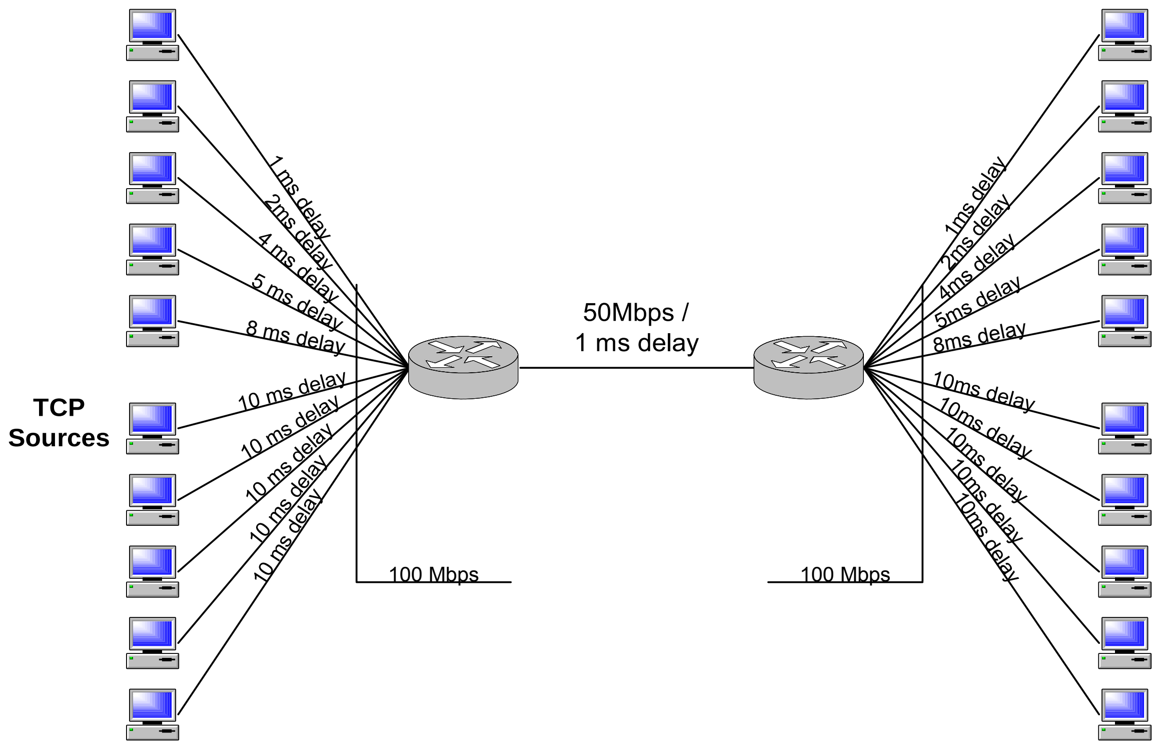
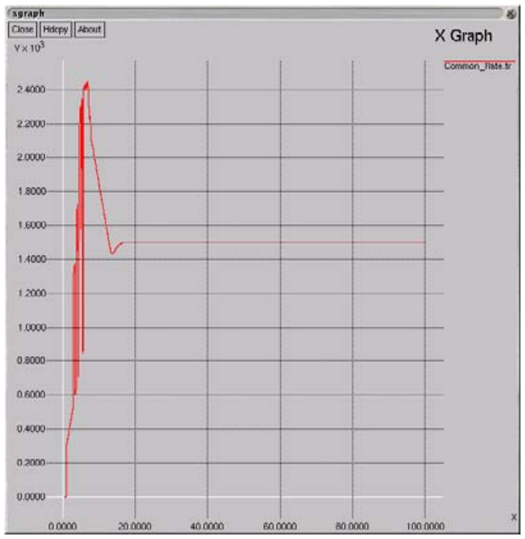
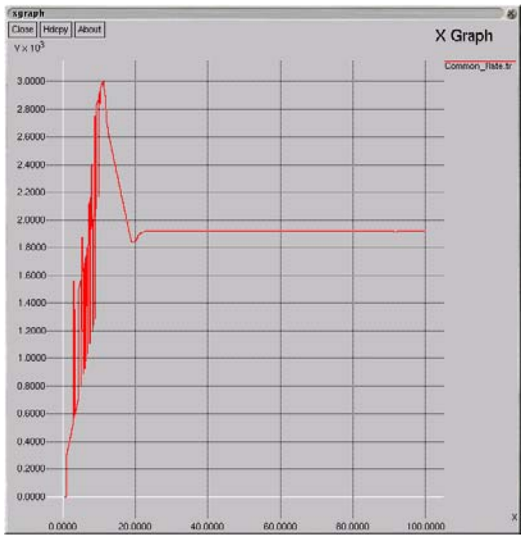
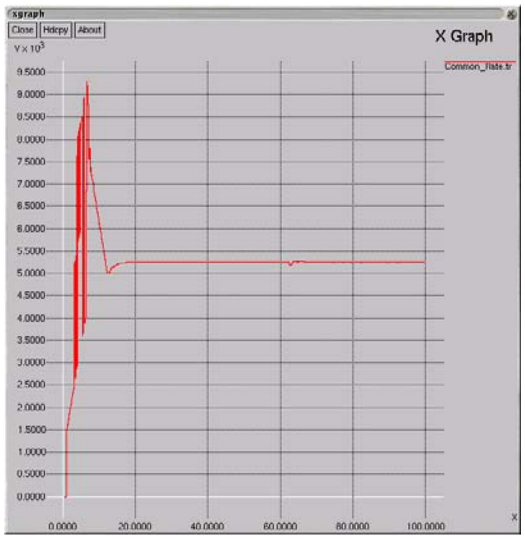


Figure 5.2. Topology of Simulation Scenario 1 with TCP/FTP sources


1 Premium, 4 Ordinary, 0 TCP


2 Premium, 3 Ordinary, 0 TCP


4 Premium, 1 Ordinary, 0 TCP

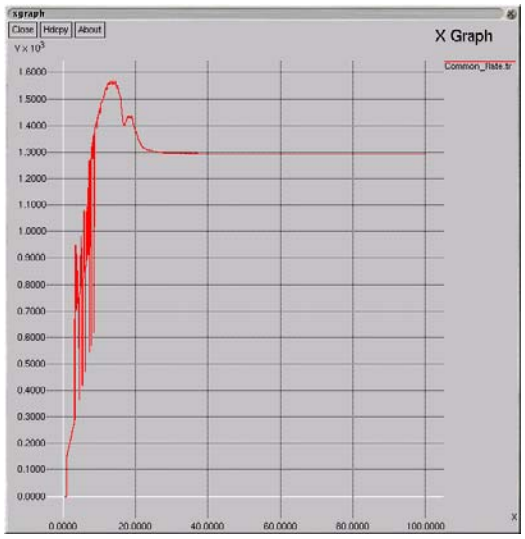
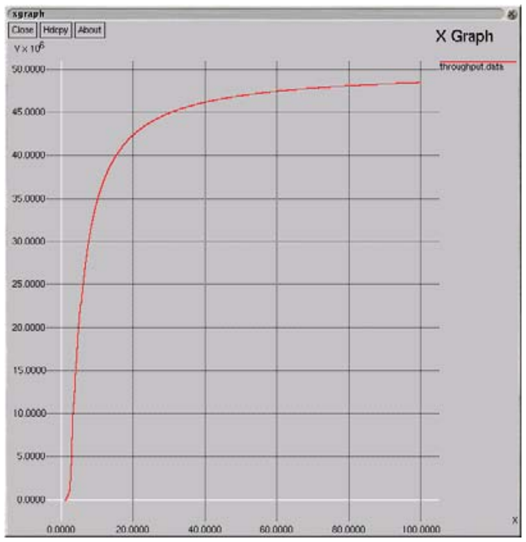
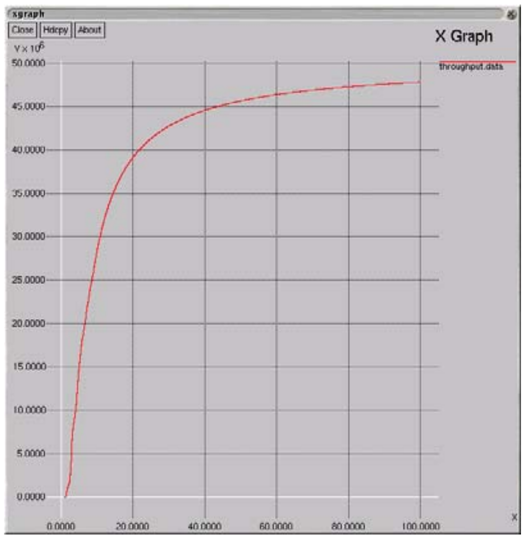

1 Premium, 4 Ordinary, 5 TCP

Figure 5.3. Common Rate

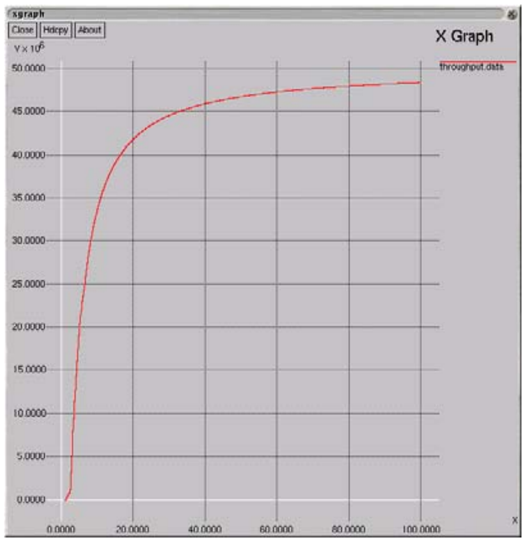
In figure 5.3 the Common Rate calculated at the bottleneck link and which will be sent back to sources, in order to change the rate that Ordinary packets are sent, is presented. The time needed for the rate to be stabilized is very little (mostly less than 20 ms) and after that is kept almost constant. It is also very important to see that the use of TCP traffic has no negative effects at the RMD-IDCC.



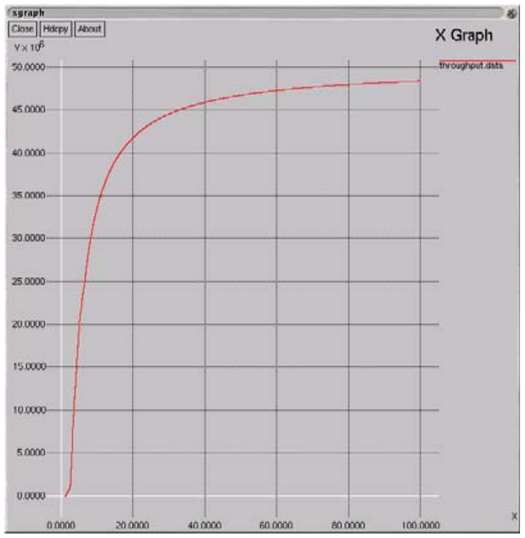
1 Premium, 4 Ordinary, 0 TCP



2 Premium, 3 Ordinary, 0 TCP



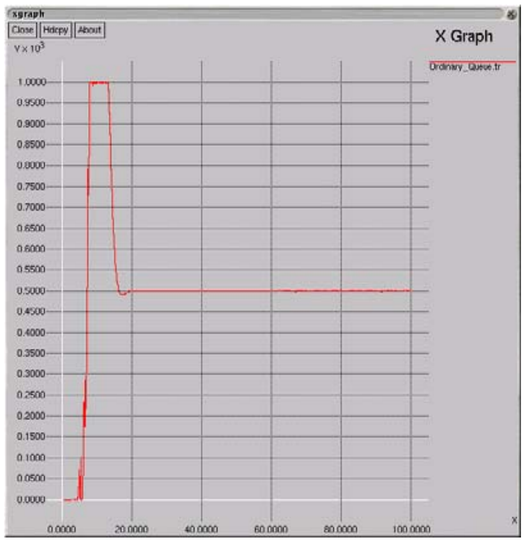
4 Premium, 1 Ordinary, 0 TCP



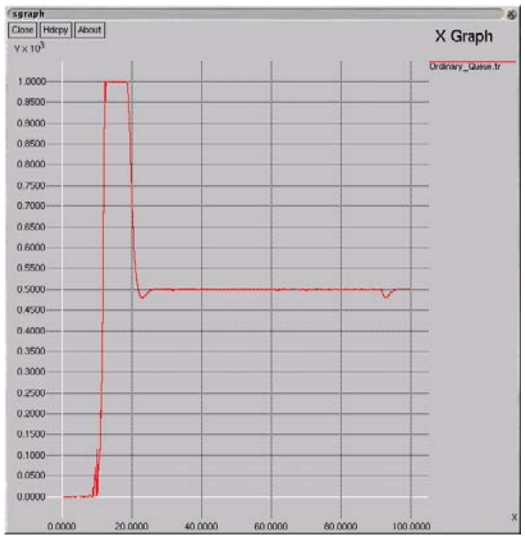
1 Premium, 4 Ordinary, 5 TCP

Figure 5.4. Throughput

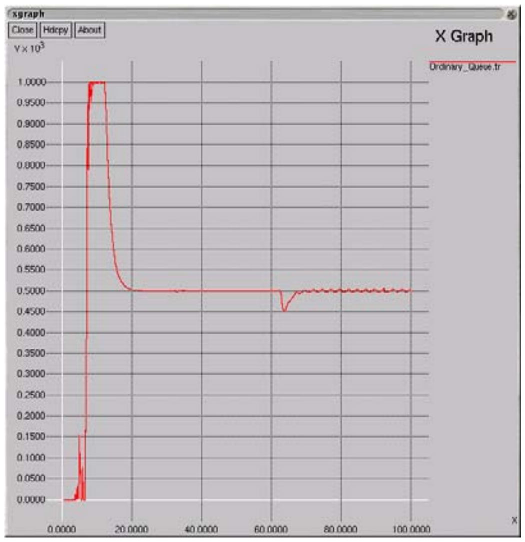
In figure 5.4 the throughput of the bottleneck link is presented. As shown, it is very high (96 – 98% link utilization) and it increases very rapidly (in the first 20 ms). Compared to the results of [39] where some simulations results for RMD are presented, it is obvious that RMD-IDCC has better results since the throughput of the link is reaching higher values.



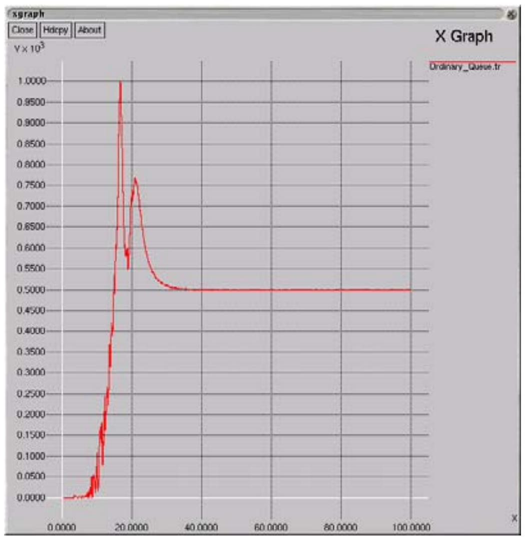
1 Premium, 4 Ordinary, 0 TCP



2 Premium, 3 Ordinary, 0 TCP



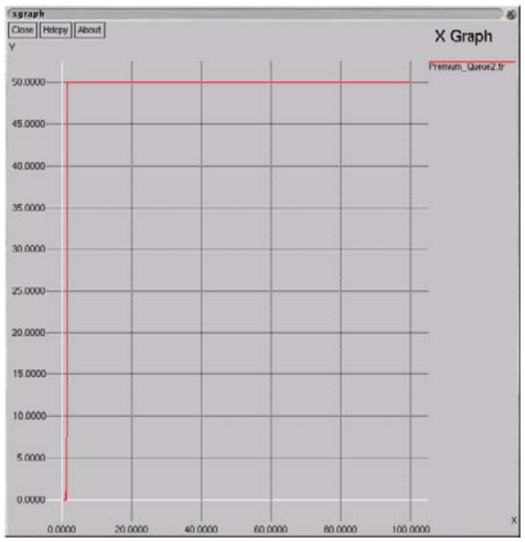
4 Premium, 1 Ordinary, 0 TCP



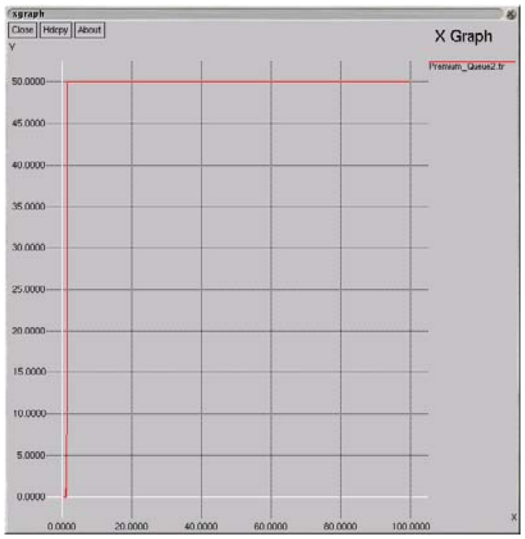
1 Premium, 4 Ordinary, 5 TCP

Figure 5.5. Ordinary Queue Length

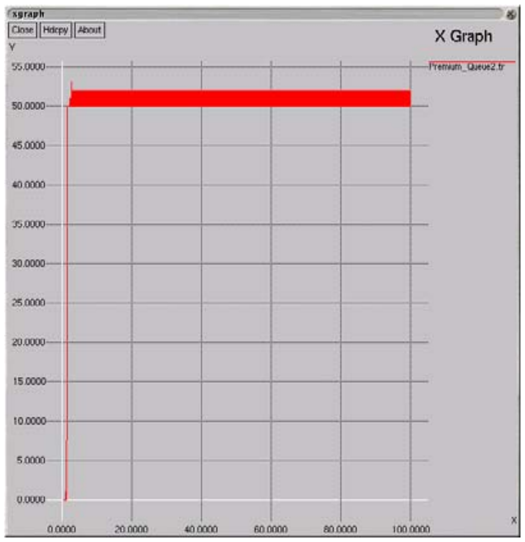
Figure 5.5 presents the Ordinary Queue Length. It behaves in the same way as Common Rate does. When the Common Rate is stabilized, the Ordinary Queue Length takes the reference value, which was set to 500 packets. The queue length remains constant for the time left.



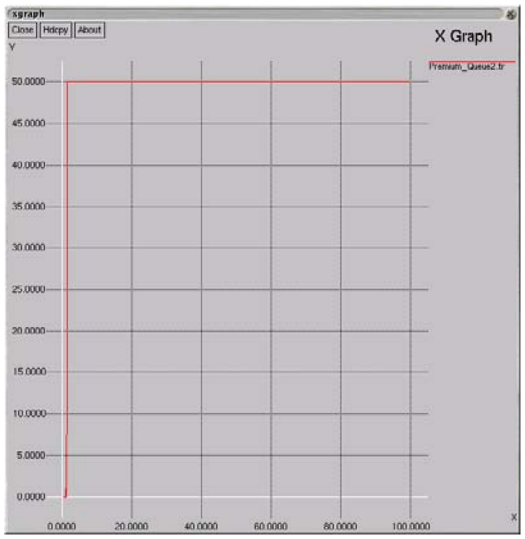
1 Premium, 4 Ordinary, 0 TCP



2 Premium, 3 Ordinary, 0 TCP



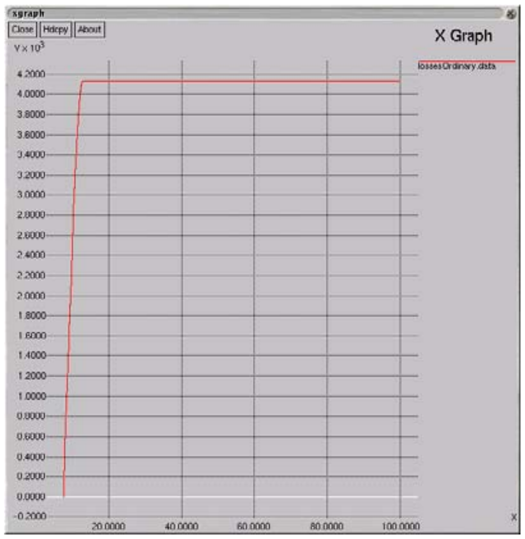
4 Premium, 1 Ordinary, 0 TCP



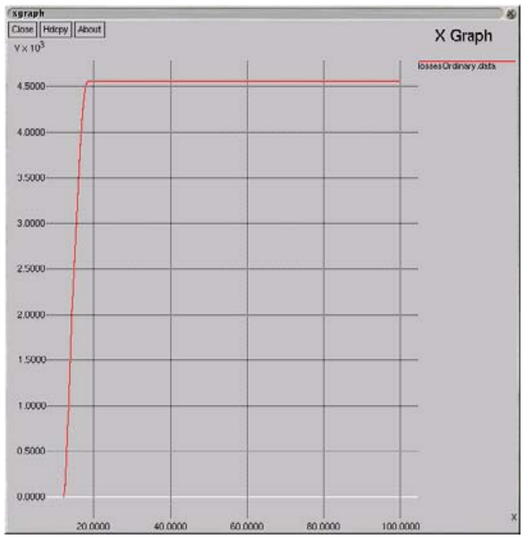
1 Premium, 4 Ordinary, 5 TCP

Figure 5.6. Premium Queue Length

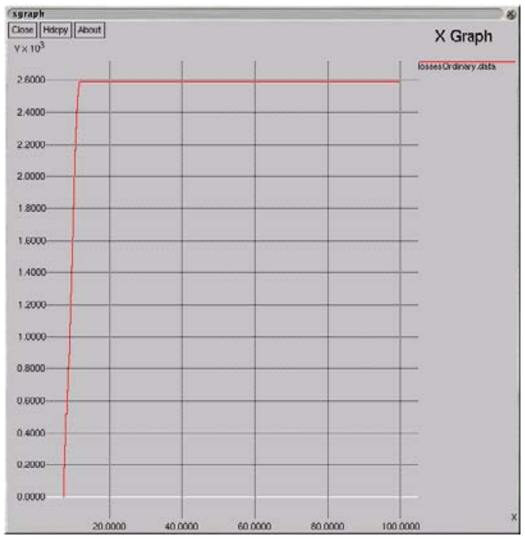
Premium Queue Length is shown in Figure 5.6. What is observed is that the Premium Queue Length is well controlled. The only case when the length takes greater value than the reference value is when the majority of the traffic is Premium, having 4 nodes sending Premium traffic and 1 node sending Ordinary traffic. Nevertheless, it did not exceed 52 packets, when the reference value was 50 packets.



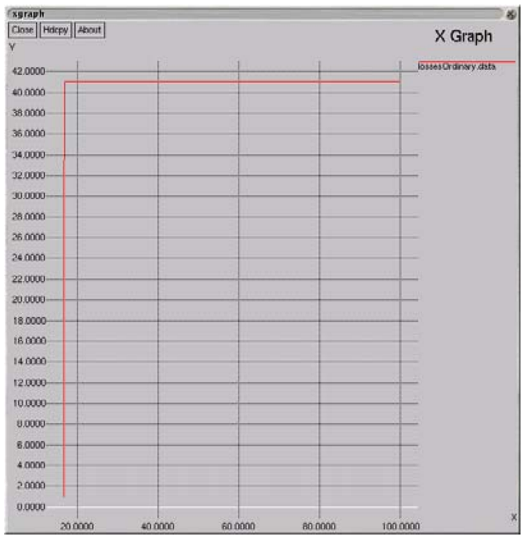
1 Premium, 4 Ordinary, 0 TCP



2 Premium, 3 Ordinary, 0 TCP



4 Premium, 1 Ordinary, 0 TCP



1 Premium, 4 Ordinary, 5 TCP

Figure 5.7. Losses for Ordinary Traffic Packets

Losses of Ordinary Packets are presented only at the beginning of the simulation (Fig. 5.7). When the rate takes a constant value, there are no losses in none of the cases. This proves that the implementation of RMD – IDCC model leads to bounded loss of Ordinary Traffic packets.

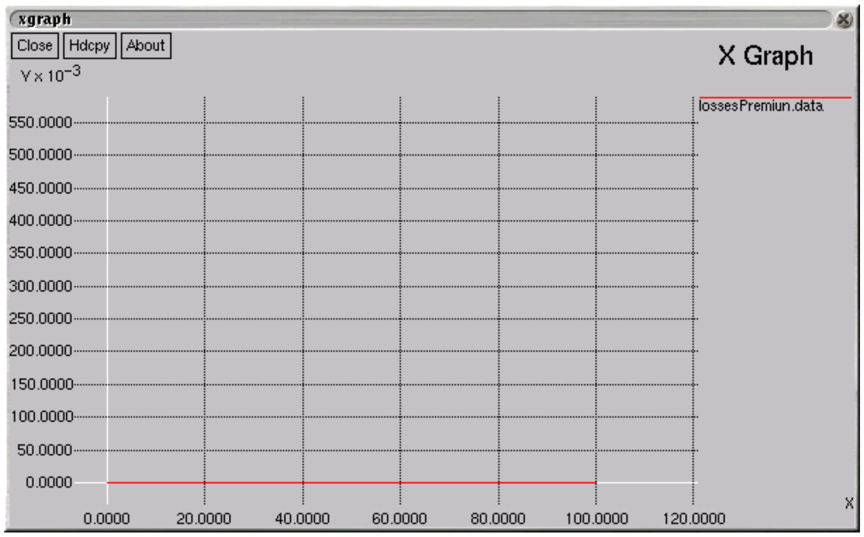


Figure 5.8. Losses for Premium Traffic Packets

What is very important and encouraging is that no losses of Premium Traffic packets occurred during the whole simulation. This can be observed in figure 5.8, where the loss of packets is 0.

5.3 Simulation Scenario 2

This simulation scenario was used to observe the reaction of RMD-IDCC when the link delays change. The topology of that simulation scenario is presented in figure 5.9. It was used for three different scenarios:

- The links delay before the bottleneck link is *equal* to the links delay after the bottleneck link and is set to 1ms.
- The links delay before the bottleneck lick is *greater* than the links delay after the bottleneck lick and is set to 20ms and 1ms respectively.
- The links delay before the bottleneck lick is *less* than the links delay after the bottleneck lick and is set to 1ms and 20ms respectively.

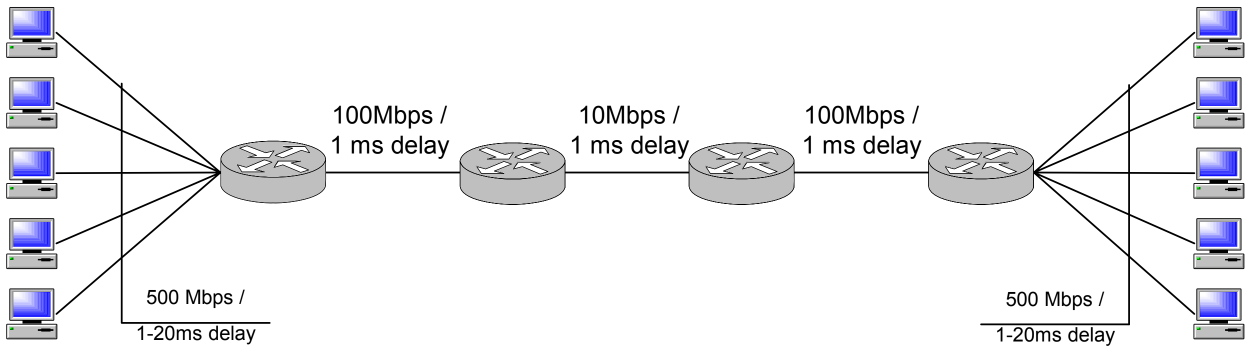
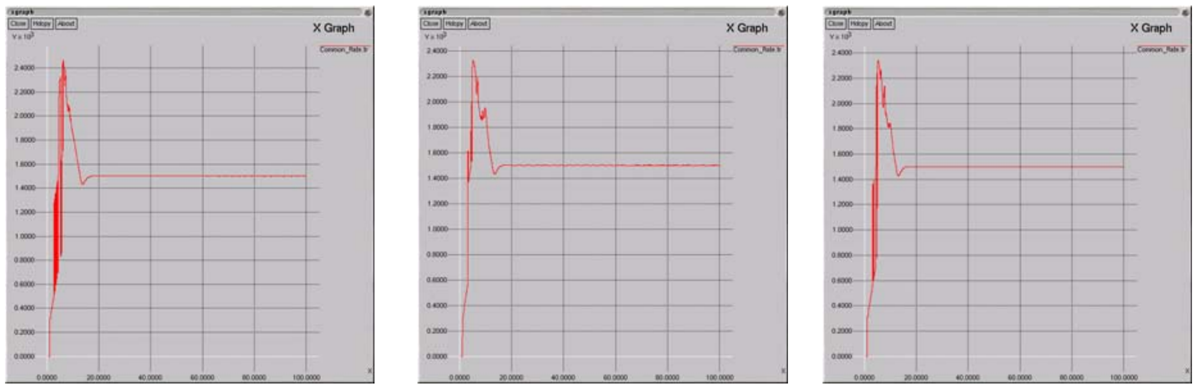


Figure 5.9. Topology of Simulation Scenario 2

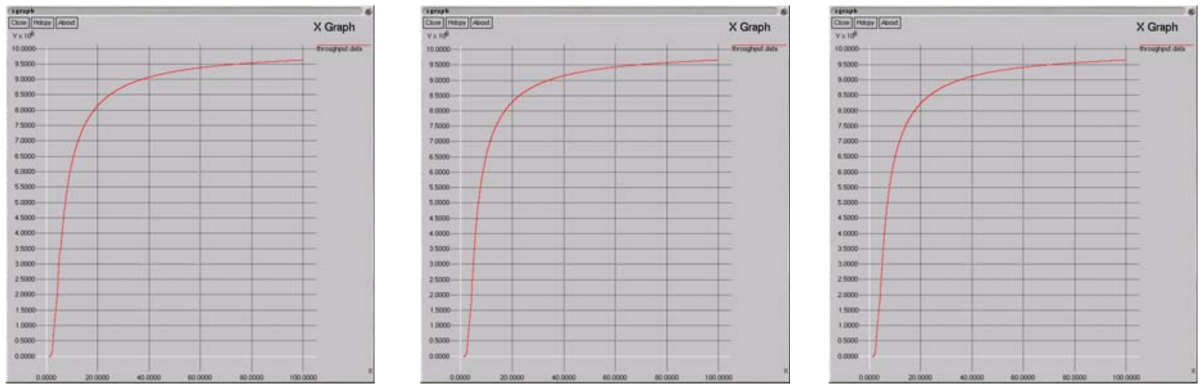
The results are shown in figures 5.10 - 5.15. They are very similar to the results of simulation scenario 1. The Common Rate is stabilized at the value of 1450 packets/second in all cases (Fig. 5.10). When the delay before the bottleneck link is less than the delay after, the throughput increases a little bit faster, but eventually it takes the same value as in the other cases (fig. 5.11). The Ordinary Queue Length reaches the reference value at the same time in all three cases (fig. 5.12), whereas Premium Queue Length presents exactly the same behavior in all cases (fig. 5.13).

As far as losses for Ordinary Traffic packets are concerned, the greatest value is observed when the delay before and after the bottleneck link is equal. When the delay before is greater than the delay after the bottleneck link, the value of losses is smaller. It gets even smaller in the case of greater delay after the bottleneck link (Fig. 5.14). In the case of Premium Traffic packets, there are no losses in none of the cases (Fig. 5.15). Conclusively, it can be deducted that link delay has no serious effect on the performance of the RMD – IDCC model.



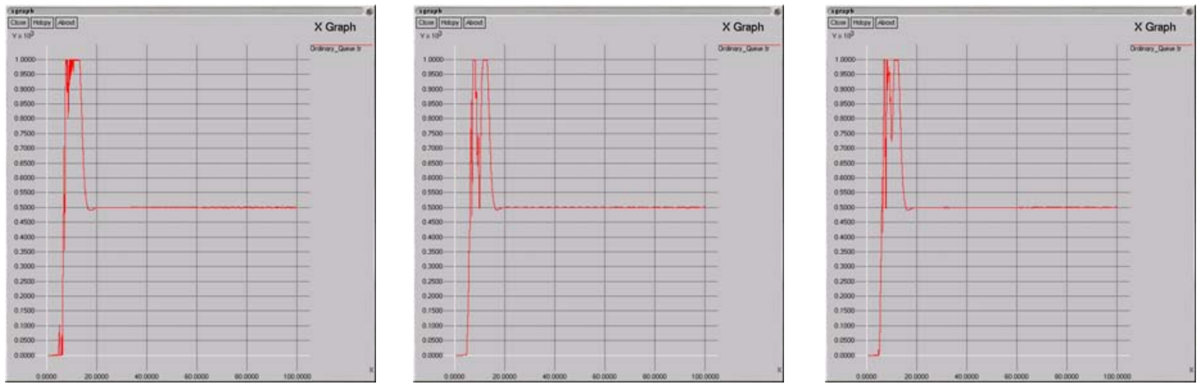
1ms, 1ms 20ms, 1ms 1ms, 20ms

Figure 5.10. Common Rate



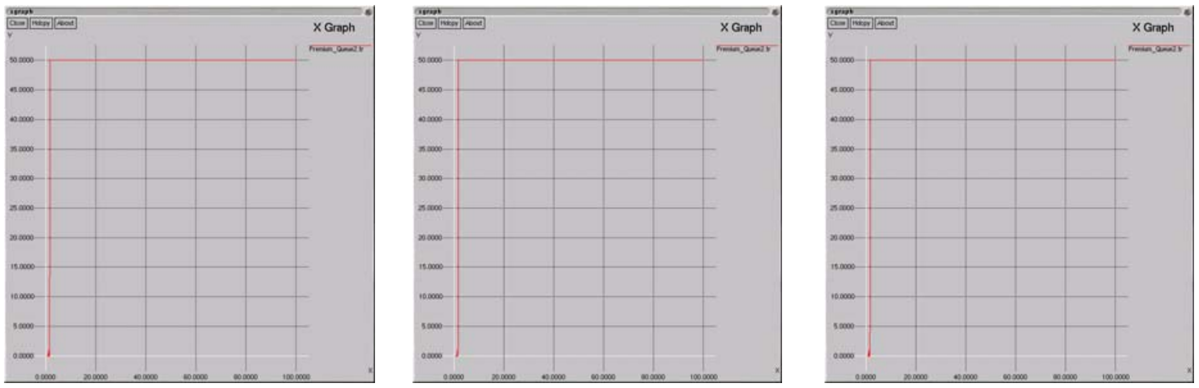
1ms, 1ms 20ms, 1ms 1ms, 20ms

Figure 5.11. Throughput



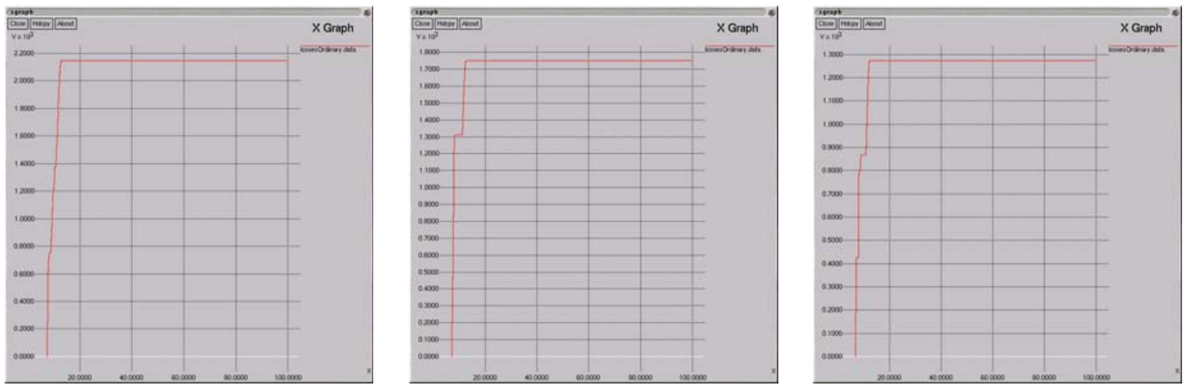
1ms, 1ms 20ms, 1ms 1ms, 20ms

Figure 5.12. Ordinary Queue



1ms, 1ms 20ms, 1ms 1ms, 20ms

Figure 5.13. Premium Queue Length



1ms, 1ms 20ms, 1ms 1ms, 20ms

Figure 5.14. Losses for Ordinary Traffic Packets

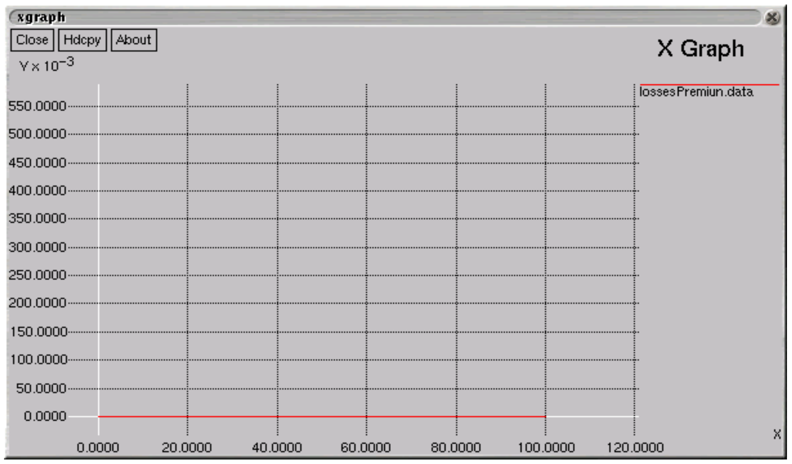


Figure 5.15. Losses for Premium Traffic Packets

5.4 Simulation Scenario 3

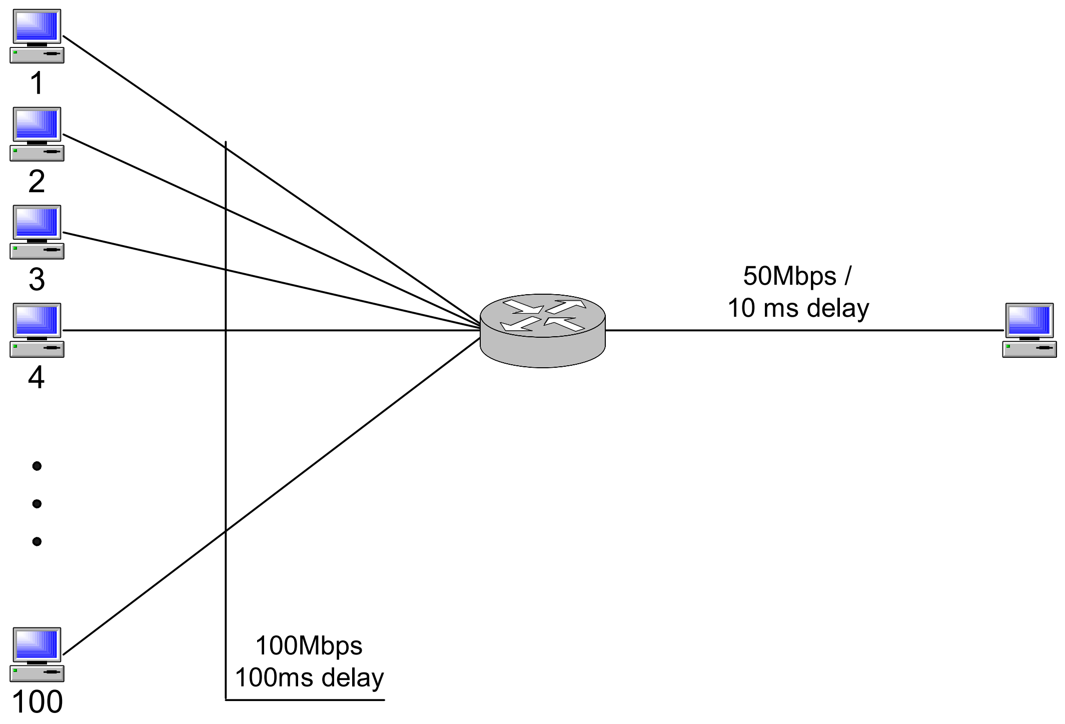


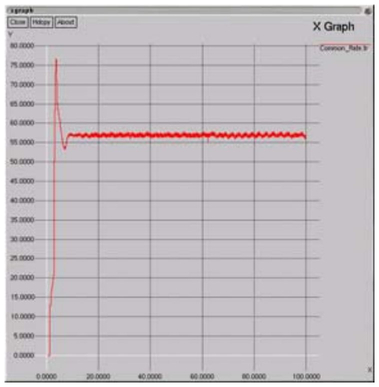
Figure 5.16. Topology of Simulation Scenario 3

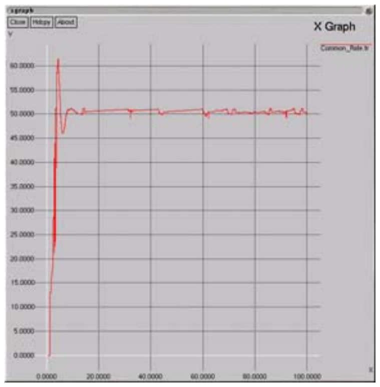
The third simulation scenario aims at observing the performance of RMD – IDCC in a more realistic situation. This simulation scenario involves 100 sources, some of them sending Premium traffic and the others sending Ordinary traffic. All the Premium sources are sending packets at the rate of 250 packets/second. The bandwidth and the link delays are shown in Figure 5.16. The maximum bit rate that the bottleneck link can afford is 6300 packets/second. This topology was run for six different scenarios:

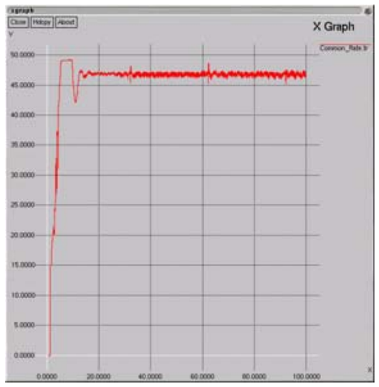
- 4 sources are sending Premium and 96 are sending Ordinary traffic.
- 8 sources are sending Premium and 92 are sending Ordinary traffic.
- 10 sources are sending Premium and 90 are sending Ordinary traffic.
- 25 sources are sending Premium and 75 are sending Ordinary traffic.

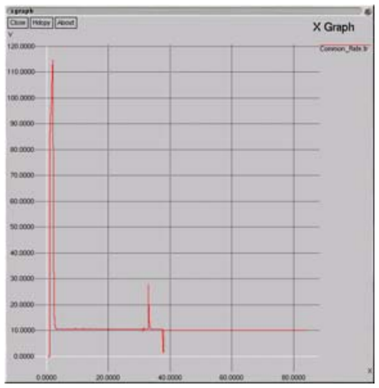
- 30 sources are sending Premium and 70 are sending Ordinary traffic *without using RMD.*
- 30 sources sending Premium traffic and 70 Ordinary with the use of RMD-IDCC.

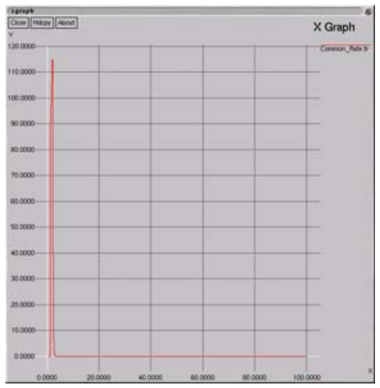
Only the last two scenarios are being discussed, since these scenarios can be used to prove the effectiveness of RMD-IDCC as compared to IDCC. The results of the first four scenarios were expected, since the aggregated Premium Traffic was less than the bottleneck link capacity.


4 Premium, 96 Ordinary


8 Premium, 92 Ordinary


10 Premium, 90 Ordinary


25 Premium, 75 Ordinary


30 Premium, 70 Ordinary (IDCC)

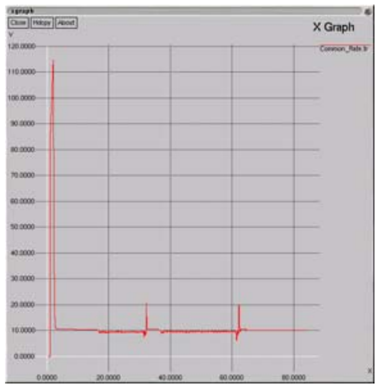
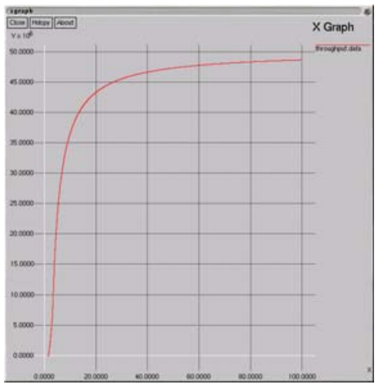
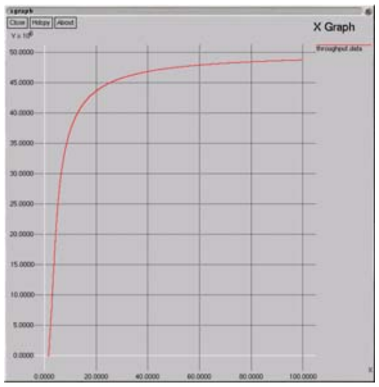

30 Premium, 70 Ordinary (RMD-IDCC)

Figure 5.17. Common Rate

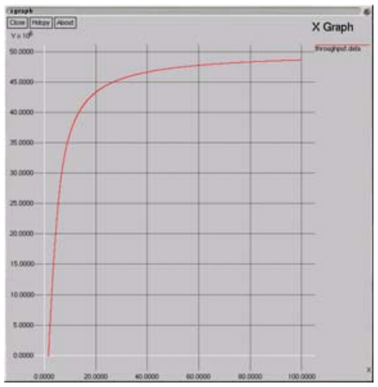
When simple IDCC was used, the Common Rate was set to 0, almost from the beginning. This happened because all of the bandwidth was used by Premium Traffic (30 sources x 250 packets/sec = 7500 packet/sec which is greater than the link capacity). When RMD-IDCC was used the rate remained constant at the value of 10 packet/sec (Fig. 5.17).



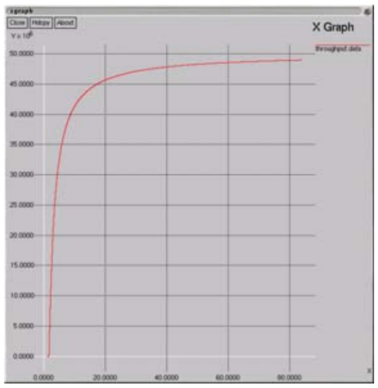
4 Premium, 96 Ordinary



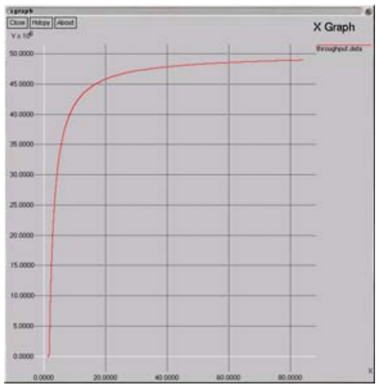
8 Premium, 92 Ordinary



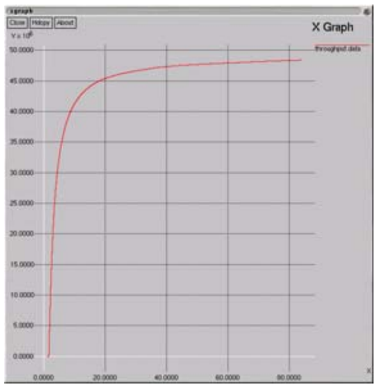
10 Premium, 90 Ordinary



25 Premium, 75 Ordinary



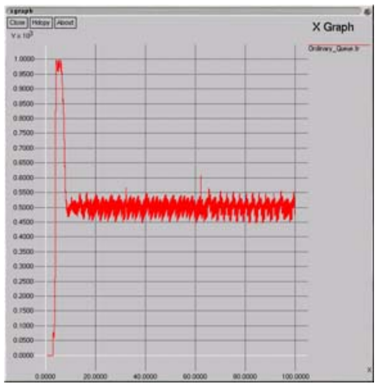
30 Premium, 70 Ordinary (IDCC)



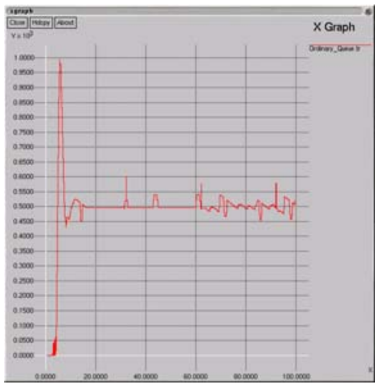
30 Premium, 70 Ordinary (RMD-IDCC)

Figure 5.18. Throughput

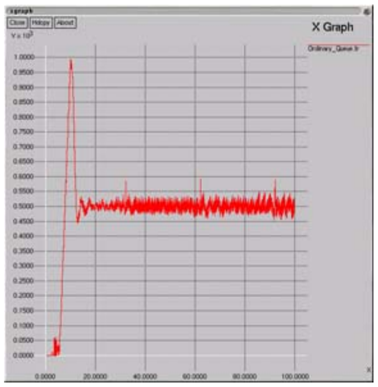
The throughput, shown in Figure 5.18, is very high for all scenarios, something that was expected. For the simple IDCC scenario many packets drops were observed as will be shown in upcoming figures. Even though the throughput in IDCC seems to be better than in RMD – IDCC scenario, the goodput of RMD – IDCC will be much better.



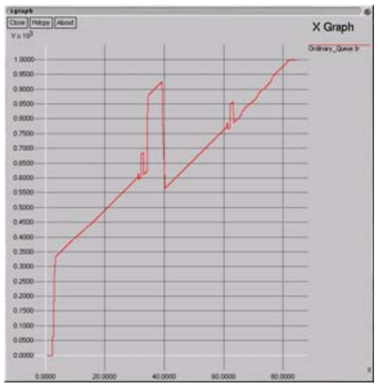
4 Premium, 96 Ordinary



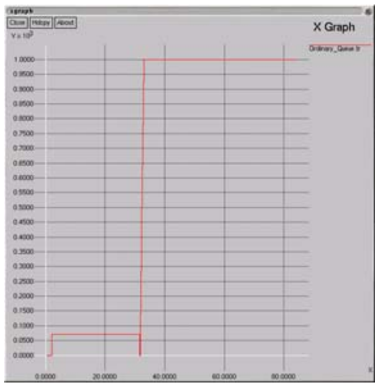
8 Premium, 92 Ordinary



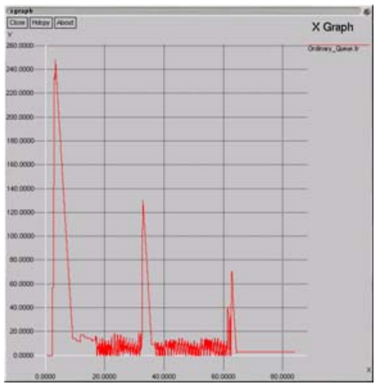
10 Premium, 90 Ordinary



25 Premium, 75 Ordinary



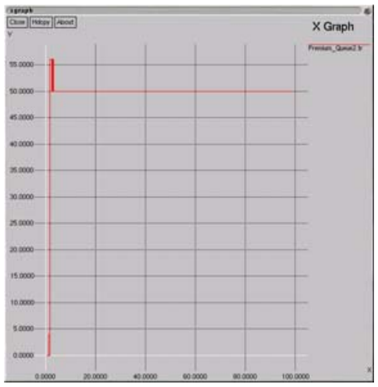
30 Premium, 70 Ordinary (IDCC)



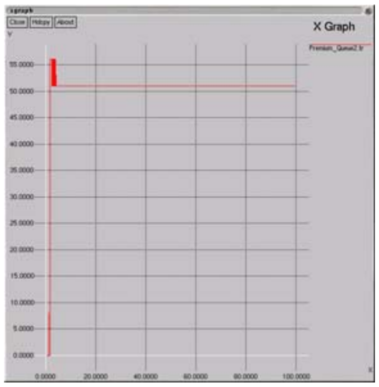
30 Premium, 70 Ordinary (RMD-IDCC)

Figure 5.19. Ordinary Queue Length

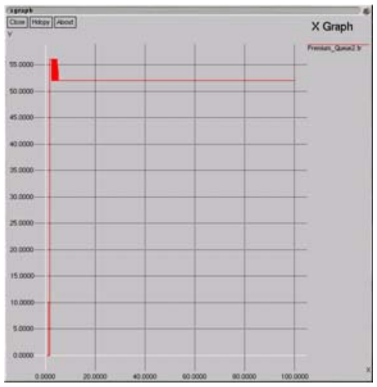
In Figure 5.19 the Ordinary Queue Length is presented. The first three cases were expected. The fourth graph shows the Ordinary Queue Length when there are 25 Premium nodes, using the 90% of the link capacity. That is why the queue length raises so much, but after the 80th ms the value is stabilized. When IDCC is used, the queue length reaches the routers limit and consequently there are losses. On the other hand, when RMD-IDCC is used, that does not happen. Also, the Ordinary Queue Length takes a constant value after the 60th second.



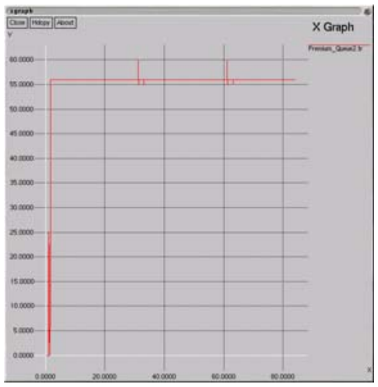
4 Premium, 96 Ordinary



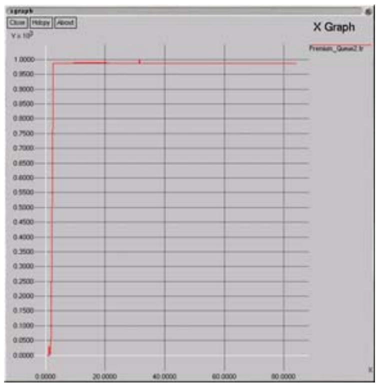
8 Premium, 92 Ordinary



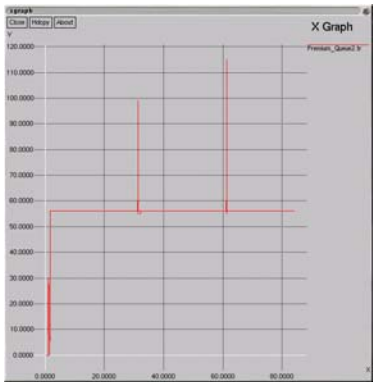
10 Premium, 90 Ordinary



25 Premium, 75 Ordinary



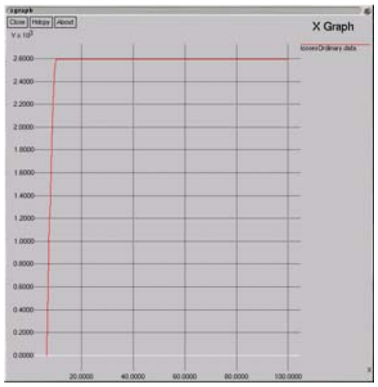
30 Premium, 70 Ordinary (IDCC)



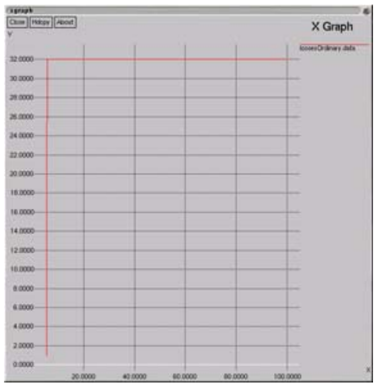
30 Premium, 70 Ordinary (RMD-IDCC)

Figure 5.20. Premium Queue Length

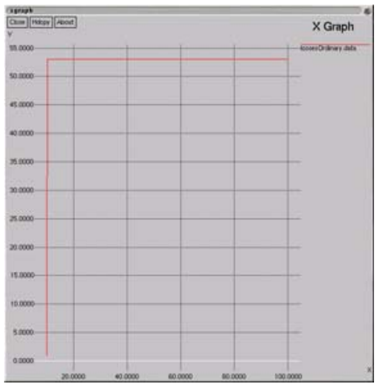
Figure 5.20 shows the unavailability of IDCC to handle real time applications, since, when using simple IDCC, the Premium Traffic Length rises up to the Queue Limit, which means that losses occur. When the proposed RMD-IDCC is used this does not happen.



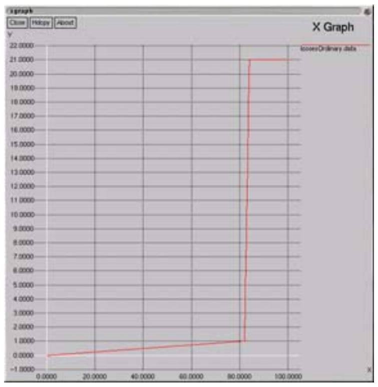
4 Premium, 96 Ordinary



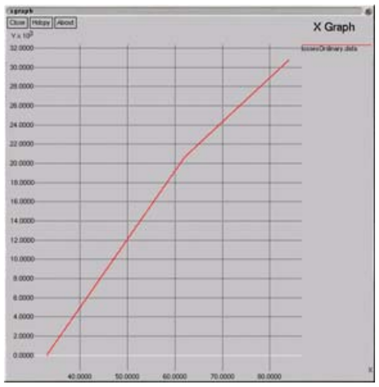
8 Premium, 92 Ordinary



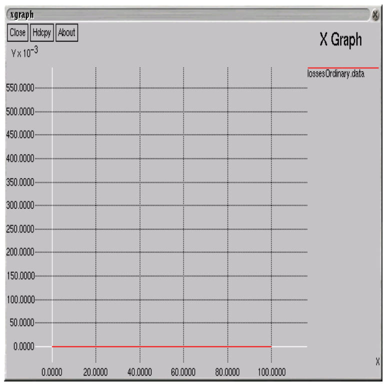
10 Premium, 90 Ordinary



25 Premium, 75 Ordinary



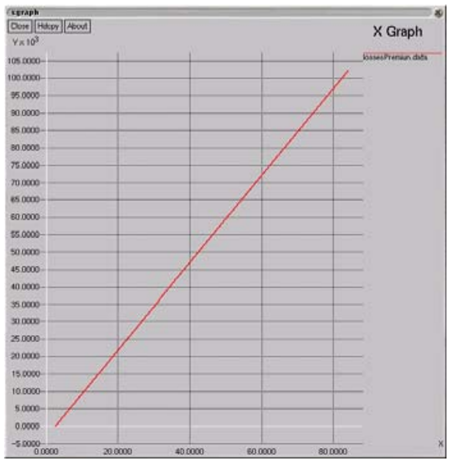
30 Premium, 70 Ordinary (IDCC)



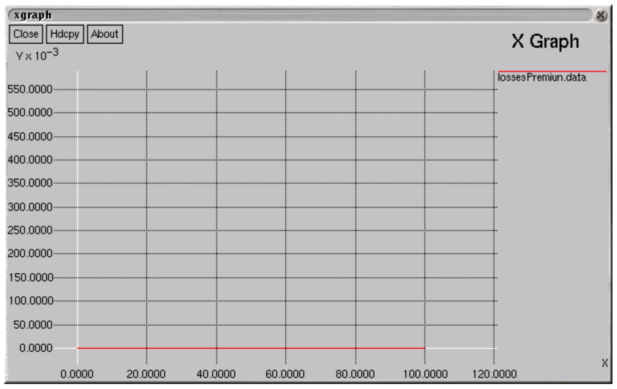
30 Premium, 70 Ordinary (RMD-IDCC)

Figure 5.21. Losses for Ordinary Traffic Packets

Figures 5.21 and 5.22 show that simple IDCC cannot handle situations, where the aggregated Premium Traffic is greater than the link capacity. This is deducted by the observation of the simple IDCC graphs where is shown that IDCC has bounded losses neither for Ordinary nor for Premium Traffic. On the contrary, when RMD – IDCC is used no losses were observed.



30 Premium, 70 Ordinary (IDCC)



All the others

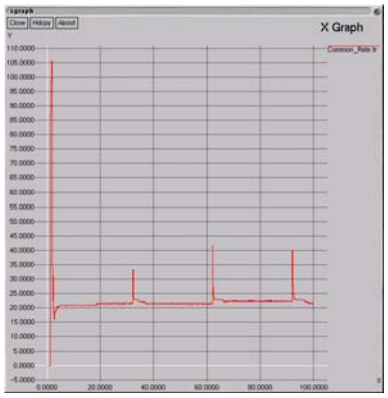
Figure 5.22. Losses for Premium Traffic Packets

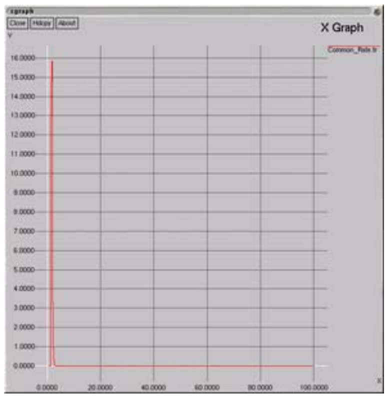
5.5 Simulation Scenario 4

In this simulation scenario, the same topology as in simulation scenario 3 was used. The only difference was that the Premium sources send packets with the rate of 2000 packet/sec. The bottleneck link can handle only three Premium Traffic packets, since the link capacity is 6300 packets/sec. This was done so that the performance of RMD-IDCC would be evaluated, in the case that Premium Traffic sources are sending packets at so high rate. This topology was used for the following three scenarios:

- 3 sources are sending Premium and 97 are sending Ordinary Traffic.
- 4 sources are sending Premium and 96 are sending Ordinary Traffic *without using RMD*.
- 4 sources are sending Premium and 96 are sending Ordinary Traffic with the use of RMD-IDCC.

The simulation results are shown in figures 5.24 – 5.29. There is great similarity between these and the results in simulation scenario 3. Again the effectiveness of the RMD – IDCC proposed model is proved.


3 Premium, 97 Ordinary


4 Premium, 96 Ordinary (IDCC)

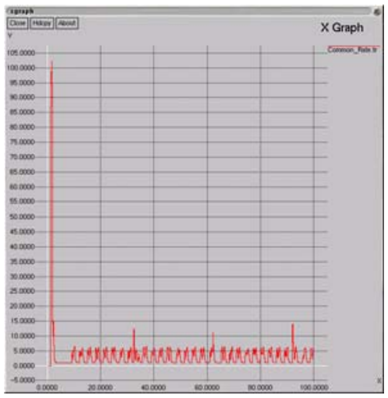
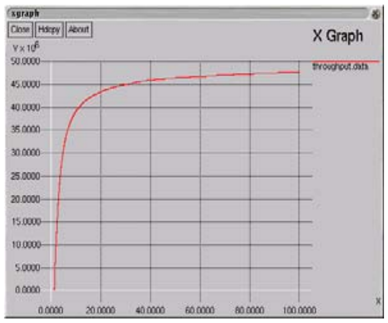
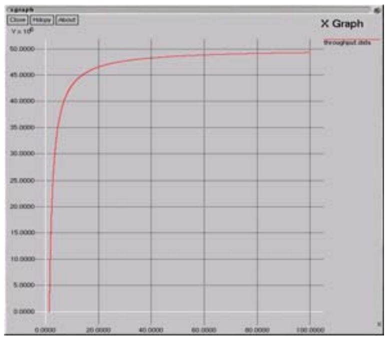

4 Premium, 96 Ordinary (RMD-IDCC)

Figure 5.23. Common Rate


3 Premium, 97 Ordinary


4 Premium, 96 Ordinary (IDCC)

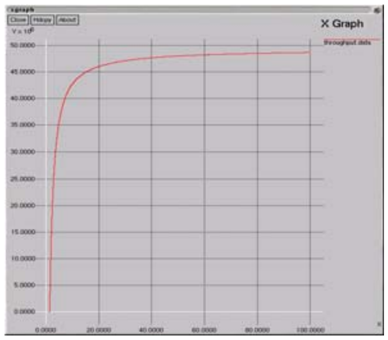
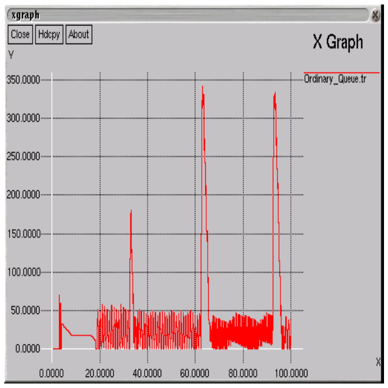
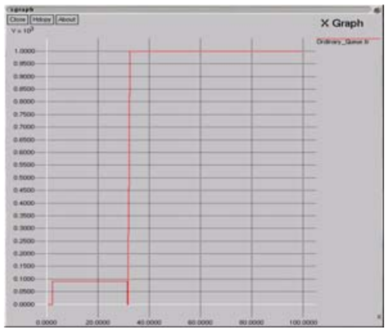

4 Premium, 96 Ordinary (RMD-IDCC)

Figure 5.24. Throughput


3 Premium, 97 Ordinary


4 Premium, 96 Ordinary (IDCC)

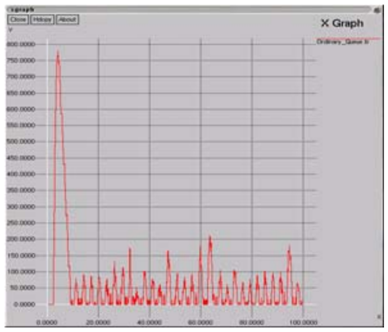
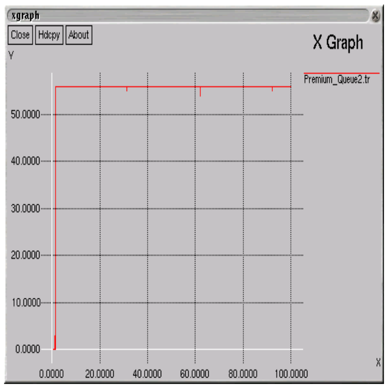
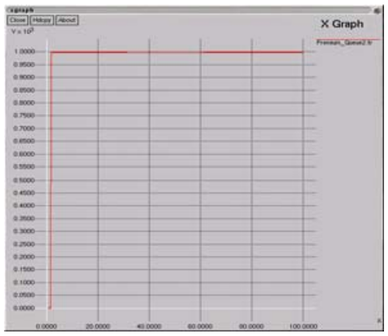

4 Premium, 96 Ordinary (RMD-IDCC)

Figure 5.25. Ordinary Queue Length


3 Premium, 97 Ordinary


4 Premium, 96 Ordinary (IDCC)

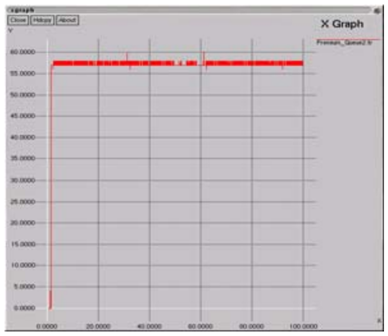
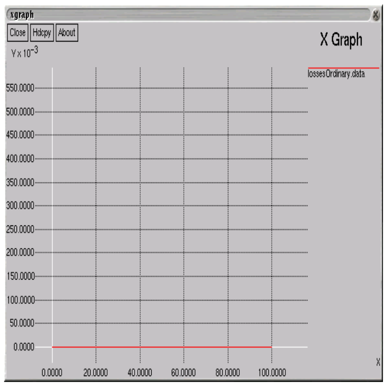
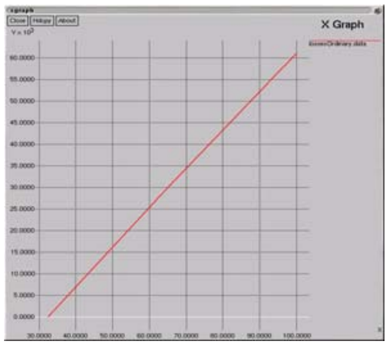

4 Premium, 96 Ordinary (RMD-IDCC)

Figure 5.26. Premium Queue Length


3 Premium, 97 Ordinary


4 Premium, 96 Ordinary (IDCC)

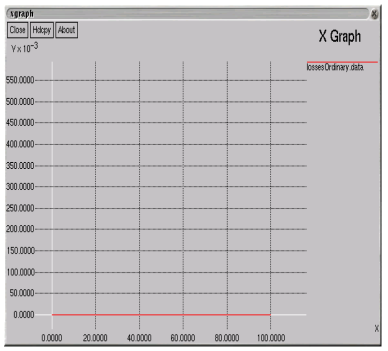
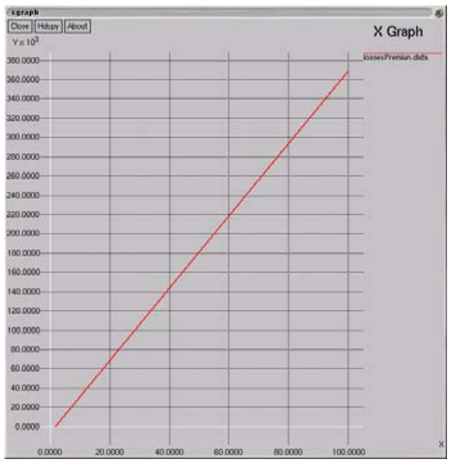
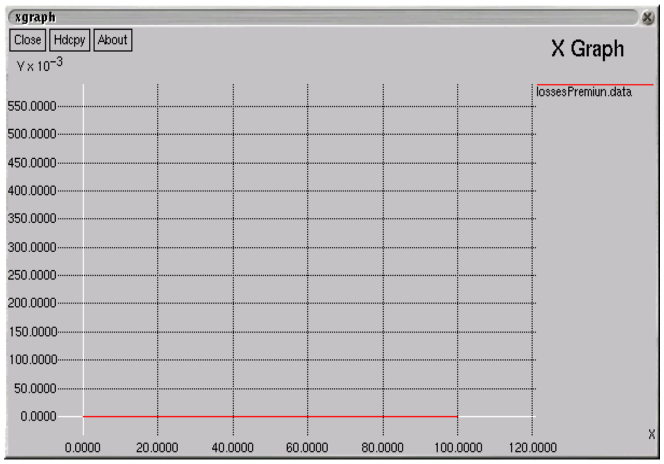

4 Premium, 96 Ordinary (RMD-IDCC)

Figure 5.27. Losses for Ordinary Traffic Packets



4 Premium, 96 Ordinary
(IDCC)

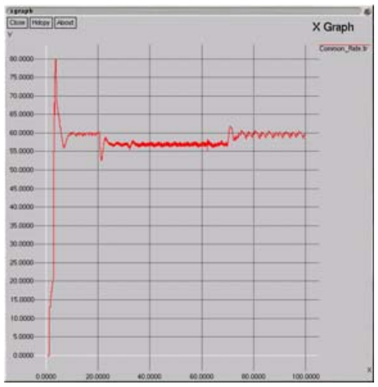
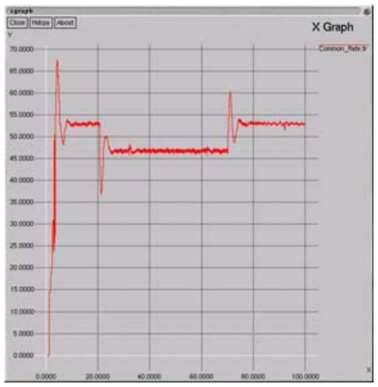
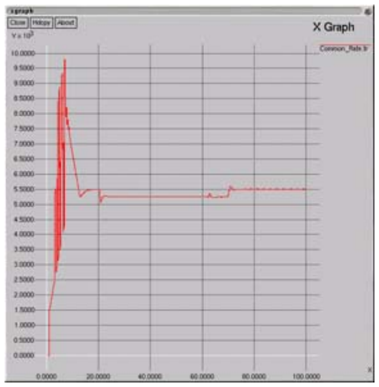


All the other

Figure 5.28. Losses for Premium Traffic Packets

5.6 Simulation Scenario 5

The simulation scenario 5 aimed at observing the RMD-IDCC behavior when all the sources did not start sending packets simultaneously. The topology of the simulation scenario 3 was used for the first two simulations. The topology of the simulation scenario 1 was used for the third simulation. What is observed from the results of these simulations is the ability of RMD-IDCC to pace with such situations. When new sources start sending packets the rate of Ordinary sources decrease and as soon as some other sources stop sending the rate of Ordinary sources increases again (Fig. 5.29).

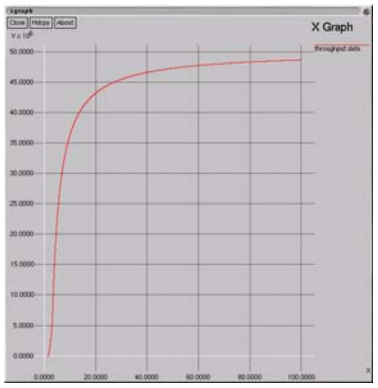
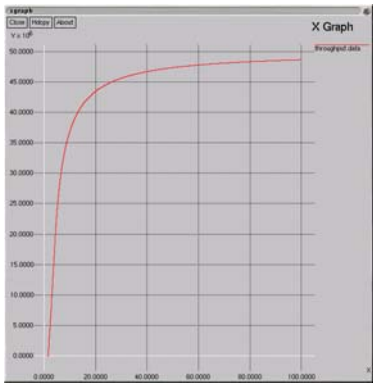
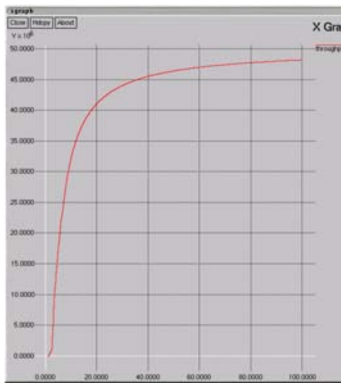




100 sources
4 Premium, 96 Ordinary

100 sources
10 Premium, 90 Ordinary

5 sources
4 Premium, 1 Ordinary

Figure 5.29. Common Rate

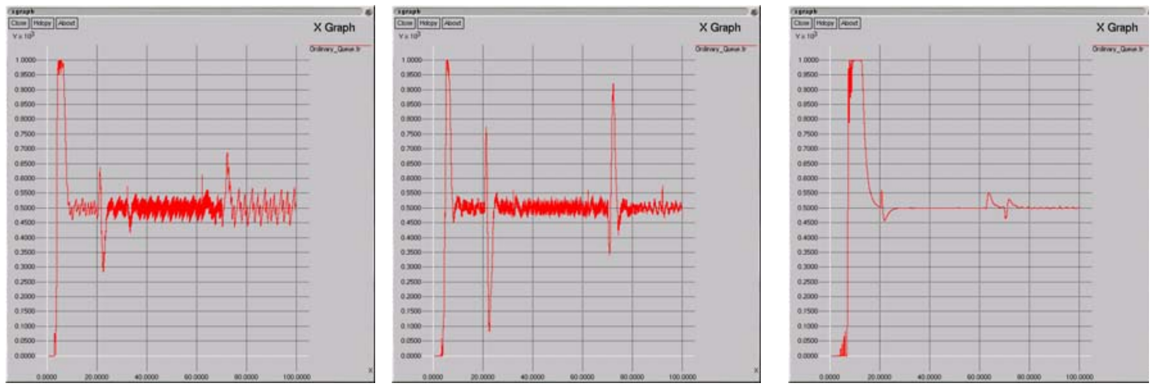
100 sources
4 Premium, 96 Ordinary

100 sources
10 Premium, 90 Ordinary

5 sources
4 Premium, 1 Ordinary

Figure 5.30. Throughput

Although, some sources were sending packets only for a few seconds the throughput is very high and without having any undershoot (Fig. 5.30).

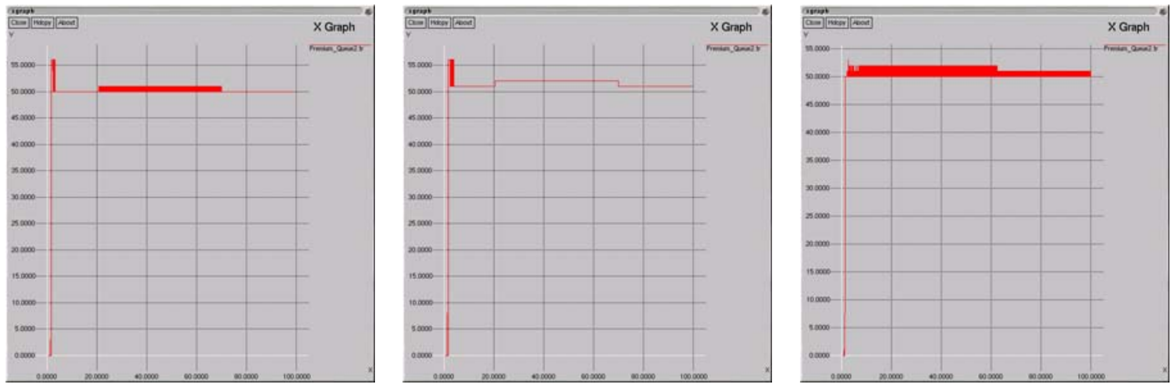


100 sources 100 sources 5 sources

4 Premium, 96 Ordinary 10 Premium, 90 Ordinary 4 Premium, 1 Ordinary

Figure 5.31. Ordinary Queue Length

The overshoot and undershoot is observed when the rate adapts. When it takes a constant value the Ordinary queue length takes the reference value (Fig. 5.31).



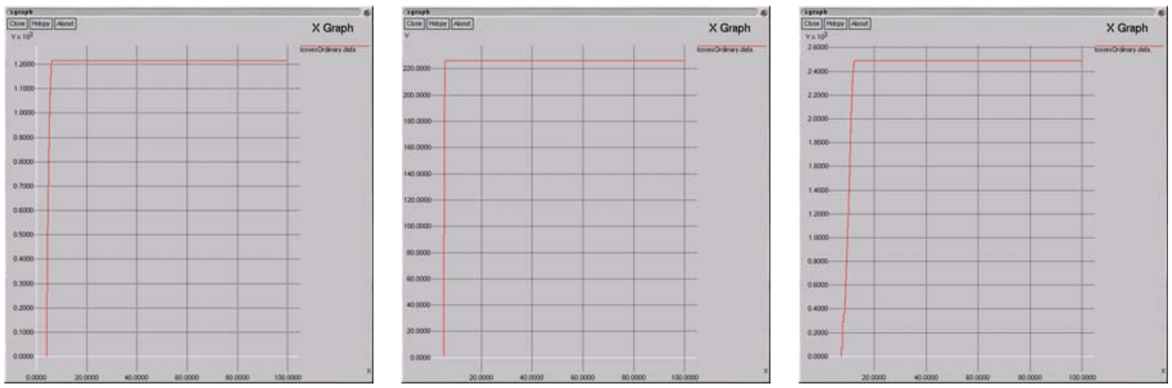
100 sources 100 sources 5 sources

4 Premium, 96 Ordinary 10 Premium, 90 Ordinary 4 Premium, 1 Ordinary

Figure 5.32. Premium Queue Length

The Premium Queue Length has the same behavior as Ordinary Queue. The length is increased when the sources are increased and decreased again when they stop sending packets (Fig. 5.32).

In figures 5.33 and 5.34 the losses of Ordinary and Premium packets are presented respectively. They show that the number of losses is not affected by the changes mentioned above. The loss of the last Ordinary packet happened before the 20th second, after which the number of the sources increased or decreased (Fig. 5.33). As for the Premium traffic no losses occurred during the whole simulation (Fig. 5.34).



100 sources	100 sources	5 sources
4 Premium, 96 Ordinary	10 Premium, 90 Ordinary	4 Premium, 1 Ordinary

Figure 5.33. Losses for Ordinary Traffic Packets

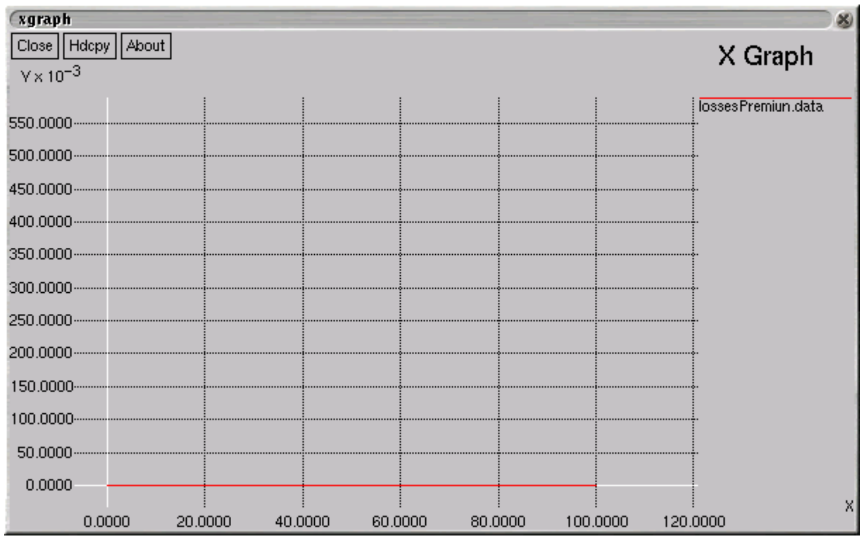


Figure 5.34. Losses for Premium Traffic Packets

Chapter 6

CONCLUSIONS

One of the most important issues still existing on the Internet is that of QoS. Since now there are many schemes proposed to this direction, but no one solves the problem in an effective way.

In this thesis, a new model was proposed, which aimed at providing QoS in IP networks. This model is based on two proposed schemes, RMD and IDCC. The traffic was divided into three classes, the Premium, Ordinary and Best Effort class. The Premium traffic was used for real time applications and, in order to secure the bandwidth needed, the Resource reservation approach was applied. This was done by RMD. The Ordinary Class was used for elastic applications, where the regulation of their rate is possible. IDCC was responsible for the avoidance of congestion and for improving utilization by decreasing or increasing the Ordinary Traffic in the network. The Best Effort traffic was used for applications that can afford instantaneous bandwidth.

The results shown in chapter 5 are very encouraging, and prove the effectiveness of the proposed scheme. The high utilization of the link is presented. In addition, no loss at all for Premium traffic and bounded losses for Ordinary traffic were observed. The queue length for both Premium and Ordinary Traffic was taking the reference value. Since the Premium traffic sends with CBR and the queue length is constant, a guaranteed delay is expected. As for Ordinary traffic, delay is bounded, since RMD guarantees that at least 10% of the link capacity is being used by Ordinary traffic.

Comparing the new scheme RMD – IDCC to the generic RMD and IDCC schemes one can easily understand the enhancement made. IDCC can guarantee high utilization, whereas RMD can guarantee the availability of resources.

Yet, there are some open issues that should be studied. The first issue is the implementation of Best Effort queue in IDCC. This implementation may help to improve throughput but at the same time it may degrade the goodput. The second issue is the possibility of resource reservation for Ordinary traffic, as well. Of course, explicit bandwidth cannot be reserved, since rate is changing but a kind of reservation can be used to limit the number of Ordinary traffic sources and guarantee a minimum available bandwidth for each source. The last issue is the possibility of integrating ECN [32], as well. The proposed model uses full feedback in order to send the calculated rate to the sources. The regulation of the rate may be possible by the use of a single bit feedback provided by ECN.

References

[1] El Allali, H., Heijenk, G., "Resource management in IPbased Radio Access Networks", Proceedings CTIT Workshop on Mobile Communications, ISBN 90-3651- 546-7, February 2001.

[2] Y. Bernet et al., "A Framework for Differentiated Services," Internet Draft, draft-ietf-diffserv-framework-02.txt, Feb. 1999.

[3] L. Benmohamed, Y.T. Yang, “A Control-Theoretic ABR Explicit Rate Algorithm for ATM Switches with Per-VC Queuing”, Infocom 98, 1998.

[4] S. Blake, D. Black, M. Carlson, E. Davies, Z. Whang. W. Weiss, “An Architecture for Differentiated Services”, IETF RFC-2475, 1998

[5] S. Black et al., "An Architecture for Differentiated Services," RFC2475, Dec. 1998.

[6] Braden, R., Clark, D., Shenker, S., “Integrated Services in the Internet Architecture: An Overview”, IETF RFC 1633, 1994.

[7] Case, J., McCloghrie, K., Rose, M. and S. Waldbusser, "Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC1905, 1996.

[8] Csaszar, A., Takacs, A., Szabo, R., Rexhepi, V., Karagiannis, G., "Severe Congestion Handling with Resource Management in Diffserv On Demand", Networking 2002 proceedings, May 19-24 2002, Pisa - Italy.

[9] DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION, "Internet Protocol", IETF RFC 791, September 1981.

[10]Deering, S., Hinden, R., "Internet Protocol, Version 6 (IPv6) Specification", IETF RFC 2460, December 1998.

[11]A. Eriksson, “Resource reservation in a connectionless network”, Proc. of Performance Information and Communication Systems (PICS’98), 1998.

[12]Floyd, S. TCP and explicit congestion notification, ACM Computer Communication, 1994

[13]V. Firoiu, M. Borden, "A Study of Active Queue Management for Congestion Control," IEEE INFOCOM'00.

[14]Heijenk, G., Karagiannis, G., Rexhepi, Westberg, L., "DiffServ Resource Management in IP-based Radio Access Networks", Wireless Personal Multimedia Communications (WPMC'01), Aalborg, Denmark, September 2001.

[15]Heinanen, J., Baker, F., Weiss, W., Wroclawski, J., “Assured Forwarding PHB group”, IETF RFC 2597, 1999.

[16]Hommad el Allali and Vlora Rexhepi, Network and Traffic Model for Enhanced UMTS Network, August 2002

[17]Jacobsson, M., Oosthoek, S., Karagiannis, G., “Resource Management in Differentiated Services: A Prototype Implementation”, to be presented in ISCC 2002 proceedings.

[18]Jacobson, V., Nichols, K., Poduri, K., “An Expedited Forwarding PHB”, IETF RFC 2598, 1999.

[19]Jacobson, V. (1988). “Congestion avoidance and control”, ACM Computer Communication Review

[20]Karagiannis, G., Rexhepi, V., Heijenk G. “A Framework for QoS & Mobility in the Internet Next Generation”, Sep 2000

[21]S. Keshav, “A control theoretic approach to flow control”, ACM SIGCOMM’91, Zurich, Switzerland

[22]A. Kolarov, G. Ramamurthy, “A control theoretic approach to the design of an explicit rate controller for ABR service”, IEEE/ACM Transactions on Networking, October 1999.

[23]K. Nichols, S. Blake, F. Baker, D. Black, "Definition of the Defferentiated Services Field (DS Field) in the IPv4 and IPv6 Headers," RFC2474, Dec. 1998.

[24]Nichols, K., Carpenter, B., "Definition of Differentiated Services Per Domain Behaviors and Rules for their Specification", RFC3086, April 2001.

[25]Nichols, K., Jacobson, V., Zhang, L., “A two-bit Differentiated Services Architecture for the Internet”, IETF RFC 2638, 1999.

[26]Pitsillides A., Chrysostomou, C., Ioannou, P., Lestas, M., “Non-linear Controllers for Congestion Control in Differentiated Services Networks”, Submitted, July 2002.

[27]A. Pitsillides, Petros Ioannou, Marios Lestas, “Congestion Control in Differentiated Services Networks: A Non-linear Control Theory Approach”, Submitted, April 2002.

[28]A. Pitsillides, P. Ioannou, L. Rossides, “Congestion Control for Differentiated-Services using Non-Linear Control Theory”, in Proceedings of the Sixth IEEE Symposium on Computers & Communications, ISCC 2001, Hammamet, Tynisia, 3-5 July 2001, pp. 726-733.

[29]A. Pitsillides, P. Ioannou, D. Tipper, “Integrated control of connection admission, flow rate, and bandwidth for ATM based networks”, IEEE INFOCOM'96, 15th Conference on Computer Communications, San Francisco, USA, March 1996, pp. 785-793.

[30]A. Pitsillides, J. Lambert, "Adaptive congestion control in ATM based networks: quality of service with high utilization", Journal of Computer Communications, 20, 1997, pp. 1239-1258.

[31]Ramakrishnan, K., “A Proposal to add Explicit Congestion Notification (ECN) to IP”, RFC 2481, January 1999

[32]Ramakrishnan, K.K., Floyd, S., and Black, D, “The Addition of Explicit Congestion Notification (ECN) to IP”. RFC 3168, Proposed Standard, September 2001

[33]C.E. Rohrs and R.A. Berry and S.J. O’Halek, “A Control Engineering’s Look at ATM Congestion Avoidance”, IEEE GLOBECOM’95, Singapore, 1995.

[34]Szabó, R., Tamás, Henk, Rexhepi,V., Karagiannis, G., "Resource Management in Differentiated Services (rmd) IP Networks" ICETA 2001 proceedings.

[35]L. Westberg et al, “Resource Management in Diffserv (RMD) Framework”, IETF’s I-D: draft-westberg-rmdframework-00.txt, April, 2001 (work in progress).

[36]L. Westberg et al, “Resource Management in Diffserv (RMD) Framework”, IETF’s I-D: draft-westberg-rmd-framework-01.txt, February 2002

[37]L. Westberg et al, “Resource Management in Diffserv Measurement-based Admission Control PHR”, IETF’s I-D: draft-westberg-rmd-mbac-phr-00.txt, October 2002.

[38]L. Westberg et al, “Resource Management in Diffserv On DemAnd (RODA) PHR”, IETF’s I-D: draftwestberg-rmd-od-phr-00.txt, April, 2001 (work in progress).

[39]Westberg et.al., "Resource Management in Diffserv (RMD):A functionality and performance behavior", "Seventh International Workshop on Protocols For High-Speed Networks (PfHSN'2002)" proceedings

[40]Westberg, L., Jacobsson, M., Karagiannis, G., Oosthoek, S., Partain, D., Rexhepi, V., Szabo, R., Wallentin, P., “Resource Management in Diffserv Framework”, Internet Draft, Work in Progress, 2001.

[41]Lars Westberg, Martin Jacobsson, Georgios Karagiannis, Simon Oosthoek, David Partain, Vlora Rexhepi, and Pontus Wallentin, “Resource management

in diffserv on demand (RODA) PHR," Internet Draft, Internet Engineering Task Force, Apr. 2001, Work in progress.

[42]Westberg, L., Karagiannis, G., Partain, D., Oosthoek, S., Jacobsson, M., Rexhepi, V., "Resource Management in Diffserv On DemAnd (RODA) PHR", Internet Draft, Work in progress.

APPENDIX A

A.1 Premium Traffic control strategy:

The algorithms developed are based on a non-linear dynamic model of the behavior of a queue. The model has been verified by a number of researchers and is of the form shown below:

$$\dot{x}(t) = -\frac{x(t)}{1+x(t)}C(t) + \lambda(t)\,,\; x(0) = x_o$$

where $x(t)$ is the state of the queue, given by the ensemble average of the number of cells $N(t)$ in the system (i.e. queue + server) at time t; $\lambda(t)$ is the rate packets arrive at the queue, and $C(t)$ is the capacity of queue server. Note that this equation is valid for $0 \le x(t) \le x_{buffer\ size}$ and $0 \le C(t) \le C_{server}$ where $x_{buffer\ size}$ is the maximum possible queue size and C_{server} the maximum possible server rate. For Premium Traffic Service, the proposed approach is to tightly control the length of the Premium Traffic queue to be always close to a reference value, chosen by the designer (or the network operator), so as to indirectly guarantee acceptable bounds for the maximum delay and loss. The capacity for the Premium Traffic is dynamically allocated, up to the physical server limit, or a given maximum. In this way, the Premium Traffic is always given resources, up to the allocated maximum (C_{max}: maximum available or assigned capacity, and x_{max} : maximum buffer size) to ensure the provision of Premium Traffic Service with known bounds. Whenever this service does not require the use of maximum capacity to maintain its QoS at the prescribed levels it offers the excess capacity to the Ordinary Traffic Service. The control objective is to choose the capacity $C_p(t)$ to be allocated to the Premium Traffic under the constraint that the incoming traffic rate $\lambda_p(t)$ is unknown but bounded by $\hat{k}_p$ so that the averaged buffer size $x_p(t)$ is as close to the desired value x_p^{ref} (chosen by the operator or designer) as possible. In mathematical terms we need to choose Cp(t) so that $\overline{x}(t) \to 0$ ($\overline{x}_p(t) = x_p(t) - x_p^{ref}$) under the constraints that $C_p(t) \leq C_{server}$ and $\lambda_p(t) \leq \hat{k}_p < C_{server}$. Based on the fluid flow equation (1), feedback linearization, and robust adaptive control ideas:

$$C_p(t) = \max\left[0,\ \ \min\left\{C_{server},\ \ \rho_p(t)\frac{1+x_p(t)}{x_p(t)}\left[\alpha_p\overline{x}_p(t) + k_p(t)\right]\right\}\right]$$

where:

$$\rho_p(t) = \begin{cases} 0 & if \quad x_p(t) \leq 0.01 \\ 1.01x_p(t) - 0.01 & if \quad 0.01 < x_p(t) \leq 1 \\ 1 & if \quad x_p(t) > 1 \end{cases}$$

and

$$\dot{k}_p(t) = \Pr\left[\delta_p\overline{x}_p(t)\right], \qquad 0 \leq k_p(0) \leq \hat{k}_p$$

where $\Pr[\cdot]$ is a projection operator defined as:

$$\Pr\left[\delta_p\overline{x}_p(t)\right] = \begin{cases} & if\,(0 < k_p(t) < \hat{k}_p\ \ ) \\ \delta_p\overline{x}_p(t) & or\,(k_p(t) = \hat{k}_p\ and\ \overline{x}_p(t) \leq 0) \\ & or\,(k_p(t) = 0\ and\ \overline{x}_p(t) \geq 0) \\ 0 & else \end{cases}$$

where $\hat{k}_p$ is a constant indicating the maximum rate that could be allocated to incoming Premium Traffic (e.g. through a connection admission policy), and $\alpha_p > 0$, and $\delta_p > 0$, are design constants that affect the convergence rate and performance.

A.2 Ordinary Traffic control strategy:

The Ordinary Traffic Service Controller regulates the flow of Ordinary Traffic into the network, by monitoring the length of the Ordinary Traffic queue and the available capacity. The length of the Ordinary Traffic queue is compared with the reference value and using a non-linear control strategy it informs the sources of the maximum allowed rate they can transmit over the next control interval. This algorithm takes into account the leftover capacity $C_r(t) = \max\left[0,\ \ C_{server} - C_p(t)\right]$, uses the error between the queue length $x_r(t)$ of the Ordinary Traffic queue and the reference queue length $x_r^{ref}(t)$ as the feedback information ($\overline{x}_r(t) = x_r(t) - x_r^{ref}$), and calculates the common rate $\lambda_{rd}(t)$ to be allocated to the Ordinary Traffic users once every control interval Ts. Based on the fluid flow equation (1) and feedback linearization the controlled traffic input rate is

$$\lambda_{rd}(t) = \max\left[0,\quad \min\left\{C_r(t),\quad C_r(t)\frac{x_r(t)}{1+x_r(t)} - \alpha_r \overline{x}_r(t)\right\}\right]$$

where $\alpha_r > 0$ is a design constant. Once the common rate is calculated it is sent to all upstream sources. The way the latter is done is critical, as feedback delays and different feedback implementation schemes can degrade the system performance considerably and may lead to poor stability margin. In an IP network it is assumed that a new field has been introduced in the TCP header, which enables explicit transfer of feedback information. The introduction of such a field (basically the extension of ECN from 1 bit of information to several bits) has already been brought to the attention of the research community so we hope that in a RAN network this will be made possible. The source does not allow its transmission rate over the next control interval Ts ms to exceed the allowed calculated common rate received. Note that any excess source demand (above calculated common rate) is queued at the source queues, rather than be allowed to enter the network, and thus lead to congestion.

APPENDIX B

B.1 Installation of NS-2 and update with RMD-IDCC files

In order to evaluate the proposed model some simulations had to be ran. These simulations were built in NS-2 (Version 2.1b9a). The source code of NS-2 is found in the enclosed CD, in the directory NS-2. A full documentation of NS-2, which will help install, will be found in the same directory.

After installing the NS-2, RMD and IDCC files must be copied to some directories of NS-2, installed on your computer. All these files will be found in the directory RMD-IDCC. These files must be copied to specific NS-2 directories. Table B.1 shows all the source and destination directories.

Files	**CD Directory**	**NS-2 Directory**
rmd-edge.cc, rmd-edge.h, rmd-header.h, roda-mon.cc, roda-mon.h, sarima.cc, sarima.h	RMD-IDCC/adc/rmd/	ns-allinone-2.1b9a/ns-2.1b9a/adc/rmd
idccq.cc, idccq.h	RMD-IDCC/queue/	ns-allinone-2.1b9a/ns-2.1b9a/queue
idcc.cc, idcc.h, idcc-sink.cc, idcc-sink.h	RMD-IDCC/tcp	ns-allinone-2.1b9a/ns-2.1b9a/tcp
packet.cc, pachet.h	RMD-IDCC/common	ns-allinone-2.1b9a/ns-2.1b9a/common
ns-default.tcl, ns-lib.tcl, ns-	RMD-IDCC/tcl/lib	ns-allinone-2.1b9a/ns-

packet.tcl, rmd-lib.tcl		2.1b9a/tcl/lib
queue-monitor.cc, queue-monitor.h	RMD-IDCC/tools	ns-allinone-2.1b9a/ns-2.1b9a/tools
Makefile	RMD-IDCC/	ns-allinone-2.1b9a/ns-2.1b9a/

Table B.1 Source and Destination Directories

The files ns-default.tcl, ns-lib.tcl, ns-packet.tcl and Makefile already exist in NS-2. So you can either overwrite or update them, making the changes necessary for RMD-IDCC. It is strongly recommended, especially for Makefile where the path is used, not to overwrite these files but to add the changes to the existing files.

B.2 Script Example

Here an example of a script written in tcl is given. On the CD you can find all the scripts used for the simulations, in the directory "simulations", accompanied by the graphs taken by each simulation.

```
#Create a simulator object
set ns [new Simulator]

#PerSimulation variables -- To make sure, let's make an instance variable copy from these
class variables!
$ns set numRMDFIDs_ 2              ;#How many flowIDs will be used for RMD
signaling messages
$ns set RMDFIDOffset_ 0            ;#Where these flowIDs (DSCPs) start
$ns set RODARefreshPeriod_ 30      ;#in case of RODA PHR group is used

RODAMonitor set RMDBWUnit_ 2000              ;#BytesPerSec - one resource unit
corresponds to this BW value

RODAMonitor set dump_enabled_ 0          ;#Enable dumping in user readable format
RODAMonitor set auto_dump_enabled_ 0    ;#Non-user readable format for processing
scripts
RODAMonitor set debuglevel_ 0                   ;#Debug info

#PerEdge variables
Agent/RMDEdge set dump_enabled_ 0
Agent/RMDEdge set auto_dump_enabled_ 0
Agent/RMDEdge set debuglevel_ 0

#IDCC variables
Queue set limit_ 1000
Queue/IDCCQ set bandw_ 6300
Queue/IDCCQ set alphar_ 1.5
Queue/IDCCQ set alphap_ 1000
Queue/IDCCQ set xrref_  500
Queue/IDCCQ set xpref_ 50
Queue/IDCCQ set noflows_ 4
Queue/IDCCQ set cq 0
Queue/IDCCQ set lambda_rd 0
Queue/IDCCQ set len 0
Queue/IDCCQ set sampling_time 0.1
Queue/IDCCQ set before_rate_ 0
Queue/IDCCQ set Cpt 0
Queue/IDCCQ set cstat 0
Queue/IDCCQ set intcon 1
Queue/IDCCQ set gamma 1
Queue/IDCCQ set alpha 0.1

Agent/IDCC set packetSize_ 1000
Agent/IDCC set InitRate_ 10
Agent/IDCC set filterpar_ 3
Agent/IdccSink set packetSize_ 30

#Define different colors for data flows (for NAM)
$ns color 0 Blue
$ns color 1 Red
$ns color 2 Green
$ns color 3 Cyan

#Open the NAM trace file
#set nf [open out.nam w]
#$ns namtrace-all $nf
set f [open outidcc.tr w]
set f4 [open idccrate.tr w]
set f5 [open Common_Rate.tr w]
set f6 [open premiumq.tr w]
set f7 [open ordq.tr w]
set f8 [open Ordinary_Queue.tr w]
set f9 [open Premium_Queue2.tr w]
set f10 [open cpt.tr w]
set nofn_ 12
set sources [expr ($nofn_-2)/2]
set destinations [expr $sources +2]
set router1 [expr $sources]
set router2 [expr $sources+1]

#Define a 'finish' procedure
proc finish {} {
    global ns nf f f5 f4 f6 f7 f8 f9 f10 old_data
    $ns flush-trace
    close $f5
    close $f4
    close $f6
    close $f7
    close $f8
    close $f9
    close $f10
    close $f
    #Close the NAM trace file
    #close $nf
    #Execute NAM on the trace file
    #exec nam out.nam &
    exec awk {
        {
        if($1 == "-")
        {
            old_data = old_data + $6
            print $2, old_data*8.0/$2
        }
        }
    } outidcc.tr > throughput.data

    exec awk {
        {
        if($1 == "d" && $8 == "1")
        {
            old_data = old_data + 1
            print $2, old_data
        }
        }
    } outidcc.tr > lossesPremiun.data
    exec awk {
        {
        if($1 == "d" && $8 == "0")
        {
            old_data = old_data + 1
            print $2, old_data
        }
        }
    } outidcc.tr > lossesOrdinary.data

     exit 0
}

#Create nodes
for {set i 0} {$i < $nofn_} {incr i} {
            set n($i) [$ns node]
}

#trace the bottleneck link
$ns at 0.0 "$ns trace-queue $n(5) $n(6) $f"

#For each link set the link capacity the propagation delay and the buffer scheduling
algorithm.

            $ns duplex-rodalink $n(0) $n(5) 100Mb 1ms IDCCQ
            $ns duplex-rodalink $n(1) $n(5) 100Mb 2ms IDCCQ
            $ns duplex-rodalink $n(2) $n(5) 100Mb 4ms IDCCQ
            $ns duplex-rodalink $n(3) $n(5) 100Mb 5ms IDCCQ
            $ns duplex-rodalink $n(4) $n(5) 100Mb 8ms IDCCQ

    $ns duplex-rodalink $n(6) $n(7) 100Mb 1ms IDCCQ
    $ns duplex-rodalink $n(6) $n(8) 100Mb 2ms IDCCQ
    $ns duplex-rodalink $n(6) $n(9) 100Mb 4ms IDCCQ
    $ns duplex-rodalink $n(6) $n(10) 100Mb 5ms IDCCQ
    $ns duplex-rodalink $n(6) $n(11) 100Mb 8ms IDCCQ

    $ns duplex-rodalink $n(5) $n(6) 50Mb 1ms IDCCQ

#Create the roda monitor giving 10% to the Ordinary Traffic and 90% to the Premium
traffic

for {set i 0} {$i < $sources} {incr i} {
            [[$ns link $n($i) $n($router1)] get-roda-mon] init-thresholds-by-percent 10 90
    [[$ns link $n($router1) $n($i)] get-roda-mon] init-thresholds-by-percent 10 90
}
```

```
   [[$ns link $n($router1) $n($router2)] get-roda-mon] init-thresholds-by-percent 10 90
   [[$ns link $n($router2) $n($router1)] get-roda-mon] init-thresholds-by-percent 10 90

for {set i $destinations} {$i < $nofn_} {incr i} {
   [[$ns link $n($i) $n($router2)] get-roda-mon] init-thresholds-by-percent 10 90
   [[$ns link $n($router2) $n($i)] get-roda-mon] init-thresholds-by-percent 10 90
}

set qmon [$ns monitor-queue $n(5) $n(6) 0]
set q [[$ns link $n(5) $n(6)] queue]


set FINISH_TIME 100.0

#Create the IDCC and RMD agents

for {set i 0} {$i < $sources} {incr i} {
         set s [expr $i+$destinations]
         set ingress_agent($i) [new Agent/RMDEdge/RODA]
         set egress_agent($i) [new Agent/RMDEdge/RODA]
         $ns attach-agent $n($i) $ingress_agent($i)
         $ns attach-agent $n($s) $egress_agent($i)
         $ns connect $ingress_agent($i) $egress_agent($i)
         set idcc($i) [new Agent/IDCC]
         set idccsink($i) [new Agent/IdccSink]
         $ingress_agent($i) attach $idcc($i)
         $egress_agent($i) attach $idccsink($i)
}

$ingress_agent(0) set fid_ 1
$idcc(0) set fid_ 1
$ingress_agent(1) set fid_ 0
$idcc(1) set fid_ 0
$ingress_agent(2) set fid_ 0
$idcc(2) set fid_ 0
$ingress_agent(3) set fid_ 0
$idcc(3) set fid_ 0
$ingress_agent(4) set fid_ 0
$idcc(4) set fid_ 0

$idcc(0) set InitRate_ 250

   #This automatically calls "Admitted $ingress_agent" upon admission
   #Otherwise, auto calls "Refused $ingress_agent"

         # Request resources for Premium Traffic
   $ns at 1.0 "$ingress_agent(0) QoSRequest 125" ;# in units!!!
          # Request resources for Ordinary Traffic (That why “QoSRequest 0”)
   $ns at 2.1 "$ingress_agent(1) QoSRequest 0"
   $ns at 2.2 "$ingress_agent(2) QoSRequest 0"
   $ns at 2.3 "$ingress_agent(3) QoSRequest 0"
   $ns at 2.4 "$ingress_agent(4) QoSRequest 0"

#I can self define what needs to be done upon admittance, maybe I don't want to start
traffic at all because I just
#care about signaling traffic...
proc Admitted {ingress_agent} {
          global ns
          [$ingress_agent get-agent] start

          $ns at 100.0 "[$ingress_agent get-agent] stop"
          $ns at 100.0 "$ingress_agent QoSRelease"
          #May send explicit release request message, and will definetly call "Released
$ingress_agent"
}

#Self define what happens when the connection is refused
#Maybe in that case I can change my DSCP to a best-effort DSCP and attach my regular
agent directly on the node...
proc Refused {ingress_agent} {
          global ns n
          set egress_agent [$n([$ingress_agent set dst_addr_]) agent [$ingress_agent set
dst_port_]]
          delete [$ingress_agent get-agent]
          $ns detach-agent [$ingress_agent set node_] $ingress_agent
          $ns detach-agent [$egress_agent set node_] $egress_agent
          $ns at [expr [$ns now] + 0.0000000001] "delete $ingress_agent"
          $ns at [expr [$ns now] + 0.0000000001] "delete $egress_agent"
}

proc Released {ingress_agent} {
          global ns n
          set egress_agent [$n([$ingress_agent dst_addr_]) agent [$ingress_agent dst_port_]]
          delete [$ingress_agent get-traffic-app]
          delete [$ingress_agent get-agent]

          #The following need to be manually done just like in case of any regular agents,
like TCP or UDP...
          $ns at-now  "$ns detach-agent [$ingress_agent set node_] $ingress_agent"
          $ns at-now  "$ns detach-agent [$egress_agent set node_] $egress_agent"
          $ns at-now  "delete $ingress_agent"
          $ns at-now  "delete $egress_agent"
}

proc record2 {qs} {
   global f5 f6 f7 f8 f9 f10 f11 ns q
   set ns [Simulator instance]
   set rate [$q set lambda_rd]
   set length [$q set len]
   set prs [$q set cstat]
   set ords [$q set Mhat]
   set time 0.01
   set now [$ns now]
   set curqueue [$q set cq]
   set cpt [$q set Cpt]
   puts $f9 "$now $length"
   puts $f8 "$now $curqueue"
   puts $f5 "$now $rate"
   puts $f6 "$now $prs"
   puts $f7 "$now $ords"
   puts $f10 "$now $cpt"
 $ns at [expr $now+$time] "record2 $qs"
}

proc record1 {so} {
   global f4 ns
   set ns [Simulator instance]
   set qrate [$so set rate_]
   set time 0.01
   set now [$ns now]
   puts $f4 "$now $qrate"

   $ns at [expr $now+$time] "record1 $so"
}

$ns at 0.5 "record2 $qmon"
$ns at 0.5 "record1 $idcc(1)"

#Call the finish procedure after x seconds of simulation time
$ns at $FINISH_TIME "finish"

#Run the simulation
$ns run
```