Resource Discovery and Fair Intelligent Admission Control over Scalable Internet

Ming Li

A Thesis presented for the degree of
Doctor of Philosophy

Department of Computer System
Faculty of Information Technology
University of Technology, Sydney
Australia

September 2004

Dedicated to

my Mum, Mrs. Ming S. Pang

CERTIFICATE OF AUTHORSHIP/ORIGINALITY

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text.

I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Signature of Candidate

____

Resource Discovery and Fair Intelligent Admission Control over Scalable Internet

Abstract

The Internet currently supports a best-effort connectivity service. There has been an increasing demand for the Internet to support Quality of Service (QoS) to satisfy stringent service requirements from many emerging networking applications and yet to utilize the network resources efficiently. However, it has been found that even with augmented QoS architecture, the Internet cannot achieve the desired QoS and furthermore, there are concerns about the scalability of any available QoS solutions. If the network is not provisioned adequately, the Internet is not capable to handle congestion condition. This is because the Internet is unaware of its internal network QoS states therefore it is not possible to provide QoS when the network state changes dynamically.

This thesis addresses the following question: Is it possible to deliver the applications with QoS in the Internet fairly and efficiently while keeping scalability?

In this dissertation we answer this question affirmatively by proposing an innovative service architecture: the Resource Discovery (RD) and Fair Intelligent Admission Control (FIAC) over scalable Internet. The main contributions of this dissertation are as follows:

1. To detect the network QoS state, we propose the Resource Discovery (RD) framework to provide network QoS state dynamically. The Resource Discovery (RD) adopts feedback loop mechanism to collect the network QoS state and reports to the Fair Intelligent Admission Control module, so that FIAC is capable to take resource control efficiently and fairly.

2. To facilitate network resource management and flow admission control, two scalable Fair Intelligent Admission Control architectures are designed and analyzed on two levels: per-class level and per-flow level. Per-class FIAC handles the aggregate admission control for certain pre-defined aggregate. Per-flow FIAC handles the flow admission control in terms of fairness within the class.

3. To further improve its scalability, the Edge-Aware Resource Discovery and Fair

Intelligent Admission Control is proposed which does not need the core routers involvement.

We devise and analyze implementation of the proposed solutions and demonstrate the effectiveness of the approach. For the Resource Discovery, two closed-loop feedback solutions are designed and investigated. The first one is a core-aware solution which is based on the direct QoS state information. To further improve its scalability, the edge-aware solution is designed where only the edges (not core) are involved in the feedback QoS state estimation. For admission control, FIAC module bridges the gap between "external" traffic requirements and the "internal" network ability. By utilizing the QoS state information from RD, FIAC intelligently allocate resources via per-class admission control and per-flow fairness control.

We study the performance and robustness of RD-FIAC through extensive simulations. Our results show that RD can obtain the internal network QoS state and FIAC can adjust resource allocation efficiently and fairly.

Publications List

All publications resulting from this thesis are listed as followed.

Referred Journal Papers

1. M. Li, and D. B. Hoang (2004), **FIAC: A Resource Discovery-Based Two-level Admission Control for Differentiated Service Networks**, In *Computer Communication Journal: Special Issue on End-to-End Quality of Service Differentiation* (In press), 2004.

2. M. Li, D. B. Hoang, and A. Simmonds (2004), **Fair Intelligent Admission Control over Resource-feedback Differentiated Service Network**, to appear In *Computer Communication Journal: the Special Issue of Quality of Service*, early 2005.

3. M. Li, D. B. Hoang, and A. Simmonds (2003), **Class-Based Fair Intelligent Admission Control over an Enhanced Differentiated Service Networks**, In *lecture Notes in Computer Science (LNCS) 2662*, Hyun-Kook Kahng (Ed.), Springer, 2003.

Referred Conference Papers

1. M. Li, and D. B. Hoang (2004), **Edge-Aware Resource Discovery and Fair Intelligent Admission Control over Multi-domain Differentiated Services Networks**, In *IEEE International Conference on Communications (ICC2004)*, Paris, France, June 22-26, 2004.

2. M. Li, and D. B. Hoang (2003), **Achieving Flow Fairness in DiffServ Class: Per-flow Fair Admission Control over Differentiated Service Network**, In *Proceedings of 4th International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD'03)*, Luebeck, Germany, October 16-18, 2003.

3. M. Li, D. B. Hoang, and A. Simmonds (2003), **Fair Intelligent Admission Control over DiffServ Network**, In *Proceedings of 2003 11th IEEE International Conference on Networks (ICON2003),* Sydney, Australia, September 28 - October 1, 2003.

4. M. Li, D. B. Hoang, and A. Simmonds (2003), **Class-Based Fair Intelligent Admission Control over an Enhanced Differentiated Service Networks**, In *IEEE International Conference on Information Networking 2003 (ICOIN2003),* Jeju Island, Korea, February 12-14, 2003.

5. D. B. Hoang, and M. Li (2003), **Fair Intelligent Congestion Control over DiffServ: A Resource Discovery and Control Scheme for DiffServ**, In *Proceedings of the 2003 International Conference on Information and Communication Technologies (ICT 2003),* Bangkok, Thailand, April 8-10, 2003.

6. D. B. Hoang, M. Li, and U. Szewcow (2002), **RTT-Aware Resource Discovery and Fair Share Mechanism for DiffServ**, In *Proceedings of the IEEE International Conference on Networking (ICN 2002),* Atlanta, USA, August 26-29, 2002.

7. Q. Yu, M. Li, D. B. Hoang, and D. Feng (2002), **Fair Intelligent Feedback Mechanism on TCP Based Network**, In *The 2002 International Conference on Internet Computing (IC 2002)*, Las Vegas, California, USA, June, 2002.

8. D. B. Hoang, Q. Yu, M. Li, and D. Feng (2002), **Fair Intelligent Congestion Control Resource Discovery Protocol on TCP Based Network**, In *Proceedings of the IFIP 6th Interworking 2002 Symposium*, Perth, Australia, October, 2002.

9. M. Li, and D. B. Hoang (2004), **Resource Discovery and Fair Intelligent Admission Control over Differentiated Services Networks for Variable-Length Packets**, In *Proceedings of the IEEE 10th Asia-Pacific Conference on Communications (APCC2004)*, Beijing, China, August, 2004.

Declaration

The work in this thesis is based on research carried out at the Advance Research in Networking Group (ARN), the Department of Computer System, Faculty of Information Technology, University of Technology, Sydney, Australia. No part of this thesis has been submitted elsewhere for any other degree or qualification and it all my own work unless referenced to the contrary in the text.

Acknowledgments

I would like to acknowledge all the help and encouragement I have received in my PhD studying. I am especially grateful to my supervisor, Professor Doan B. Hoang, for teaching me how to be a independent researcher, for his continuous support and encouragement, and for his guidance throughout the years. His invaluable insight comments and ideas have greatly improved this thesis. I hope and look forward to continued collaboration with him in the future.

I am grateful to the Mr. Ury Szewcow, my associate adviser, for his advice that helped to improve my thesis.

I am indebted to Professor David Feng for his advice and inspiration when I was in University of Sydney.

Thanks for Dr. Andrew J. Simmonds, Dr. Peter Leijdekkers, and Dr. Valerie Gay for their comments on my work that helped to improve the overall quality of this dissertation.

I am also indebted to several people in our research group, in particular Chi Nguyen, Bushar Yousef, Hanh Le, and Joe, for thoughtful comments. I am going to miss our Friday night dinners that always make the hard working week on a high and enjoyable.

I would like to express my gratitude to my wife Fiona Shen for her love and understanding through my graduate years. She was always behind me and gave her unconditional support to make my studying smooth running. Finally, I would like to thank my son David who used his own way to support me by joy, happiness, and noisy.

I dedicate this dissertation to my mum, who has been the main driving force behind my research work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

This thesis is on the subject of providing Quality of Service (QoS) guarantees and scalability in the Internet.

The Internet has been enjoying great success in the last ten years or so. IP network architecture enables Internet routers to be stateless. Routers only maintain certain highly aggregate routing information. They do not maintain any other forwarding state information. In particular, they do not maintain any per-flow state and do not perform any per-flow operations. Because routers do not need to maintain any per-flow state and do not perform any per-flow operations, the operational complexity of Internet routers does not increase linearly with the number of flows present at the routers. Consequently, Internet routers are able to handle a large number of simultaneous flows. In this sense, we say the Internet architecture is scalable. Secondly, because routers do not maintain any per-flow state, after a link or router failure, packets of the flows being affected by the failure can be automatically routed around the failure point by an upstream router. Therefore, end hosts are not even aware of such failures, and no connection re-establishments are needed. In this sense, we say the current Internet architecture is robust.

On the other hand, the stateless Internet architecture also has some shortcomings. The current Internet supports only a single "best-effort" class of service where traffic is processed as quickly as possible but there is no guarantee as to timeliness or actual delivery. Note that, because routers do not maintain any per-flow state, they cannot differentiate packets from different flows and provide performance-demanding applications with more resources or better treatment. It does not make any promise regarding the successful delivery of a packet, nor does it guarantee how long it takes for a packet to reach the destination. The best-effort service model is perhaps an adequate service model for the traditional Internet applications such as file transfers and E-mail. These applications are highly adaptive, they do not require stringent performance guarantees from the Internet. When utilizing traditional data applications, users know that at some point in time, they wait patiently, the desired web page, file or e-mail will reach their desktops. If users cannot wait that long, there is no solution, other than upgrading their connection with the access provider or opting for another one.

However, since the late eighties, the Internet is increasingly become used as a commercial infrastructure, thereby it demands more flexible service-based scheme for supporting QoS of applications. Secondly, an increasing number of different applications, such as interactive multimedia applications or streaming applications (radio or video), users are subject to frequent disruptions and delays that sometimes make it impossible to enjoy them. There is a need for differentiated services which are charged accordingly. A best-effort service, however, will remain for those customers who only need connectivity.

To address these issues, the problem of providing Quality of Service (QoS) in the Internet, the Internet Engineering Task Force (IETF) has recently recommended two QoS architectures for the Internet: the Integrated Service/Resource reSerVation Protocol model (IntServ) [53] and the Differentiated Service (DiffServ) [59] model. The goal of IntServ is to allow end-to-end QoS to be provided to individual flows. The resource requirements (computational processing and memory consumption) for running per-flow resource reservations on routers increase in direct proportion to the number of separate reservations that need to be accommodated. The use of per-flow state and per-flow processing is thus not cost effective, or even feasible across the high-speed core of a network. In contrast, DiffServ architecture proposes a scalable service discrimination model without requiring per-flow state management in the core network. DiffServ focuses primarily on aggregate flows and differentiates between service classes rather than providing absolute per-flow QoS guarantees. Depending on the amount of control state maintained at Internet routers, we can classify QoS mechanisms into two categories: *stateful and state-*

less. Stateful solutions need to maintain fine-grained per-flow state at each Internet router and perform certain per-flow operations. In contrast, stateless solutions preserve the stateless property of the current Internet architecture. Only coarse-grained aggregate state is maintained at Internet routers. The Integrated Service model (IntServ) [53] is a representative of a stateful QoS solution, while the Differentiated Service (DiffServ) model [8] is an example of a stateless QoS solution.

Stateful solutions such as IntServ provide very powerful and flexible QoS guarantees. In particular, per-flow rate and delay service guarantees can be supported in the IntServ framework. Compared with the stateful solutions, stateless solutions such as DiffServ can only provide *coarse-grained aggregate service guarantees*. Currently, it is not clear if such aggregate service guarantees are sufficient for supporting performance-demanding applications such as video streaming over the Internet. On the other hand, IntServ is a very costly solution. The operational complexity of routers in the IntServ framework increases linearly with the number of simultaneous flows at the routers. In contrast, DiffServ is much simpler and more scalable, given that no per-flow state is maintained by Internet core routers.

So, we can summarize our motivation for QoS in the Internet as follows.

- Firstly, the increasing case of Voice over IP/IP telephony and other application requiring QoS.
- Secondly, there are standardized QoS technologies, such as IntServ and DiffServ.
- Thirdly, DiffServ is being employed in IP network but it is not a complete solution.
- Fourthly, applying DiffServ across the Internet needs for QoS enhancement mechanisms which have capability to detect network state variation, and to take admission control based on the network state information.

Any proposed solution to QoS should consider the above issues as well as being practical to deploy.

1.2 Problem Definition

This dissertation attempts to answer the following question: Is it possible to provide Quality of Service without compromising scalability, flexibility and bandwidth efficiency of the current Internet in infrastructure?

We start by clarifying some important terminologies that used throughout this thesis.

- Quality of Service (QoS) is the ability of a network element (e.g. an application, a host or a router) to provide some level of assurance for consistent network data delivery in terms of throughput, end-to-end delay, fairness, delay jitter, packet loss, and service reliability.

  Some applications are more stringent about their QoS requirements than others, and for this reason we have two basic types of QoS available: Resource reservation type of QoS (e.g. integrated services) and Prioritization type of QoS (e.g. differentiated services).

  For Resource reservation type of QoS, the network resources are apportioned according to an application's QoS request, and subject to bandwidth management policy.

  For Prioritization type of QoS, the network traffic is classified and the network elements give preferential treatment to classifications identified as having more demanding requirements.

  These types of QoS can be applied to individual application "flow" or to flow aggregates. A "flow" is defined as an individual, uni-directional, data stream between two applications (senders and receiver). An aggregate is simply two or more flows. Typically the flows will have something in common (e.g. same priority number).

  Resource reservation (e.g. integrated services) type of QoS is applied on a per-flow basis. Prioritization (e.g. differentiated services) type of QoS is applied on a per-aggregate basis.

- Service is the treatment provided by routers and other network elements to packets of an application as they traverse the path between source and destination domain.

- Fairness means allocating the volume of resource of network to a flow may have in the network in a fair manner. We call this kind of fairness as “bandwidth delay product fairness”. Thus bandwidth delay product fairness takes into account the transmission rate and the round trip time as an indicator for the length of the pipe.

The differentiated service (DiffServ) [57] seems more amenable to the idea of providing QoS in the Internet. DiffServ is a simple method of providing differentiated classes of service, also known as Quality of Service, for Internet traffic. A small bit-pattern in each IP packet is used to mark a packet to receive a particular forwarding treatment, or per-hop behavior (PHB), at each network node. At the network border, service classes are identified and packets are marked as belonging to a particular service. Within the network, routers examine IP headers to determine which forwarding treatment is appropriate for each packet. In DiffServ, Assured Forwarding (AF) [60] PHB divides traffic into four classes where each AF class is guaranteed some minimum resources (capacity and buffering). Within each class, packets are further partitioned into one of three drop preference categories. Congested routers then drop/mark based on their preference values; and the Expedited Forwarding (EF) [62] PHB defines a service with low packet loss, low latency, low jitter and guaranteed bandwidth.

This thesis is focus on Prioritization type of QoS, differentiated services (DiffServ).

Indeed, as long as the network is correctly provisioned, scheduling in Assured Forwarding (AF) can be realized with a first-in-first-out (FIFO) discipline. Service differentiation can be enforced by marking packets as in-profile or out-of-profile, and using a buffer management algorithm that drops out-of-profile packets more aggressively.

RED with In and Out (RIO) [16], originally proposed to implement the Allocated Capacity framework from which the Assured Forwarding service is derived, is extension to Random Early Drops (RED) [23] which aim at class-based service differentiation.

However, as discussed in [50, 70], DiffServ, e.g. RIO-based scheme, exposes the unfair allocation of the network bandwidth among the Assured Forwarding service in case of different Round Trip Time (RTT), UDP/TCP interaction, network provision state changes, and unfair for micro flows inside the aggregate.

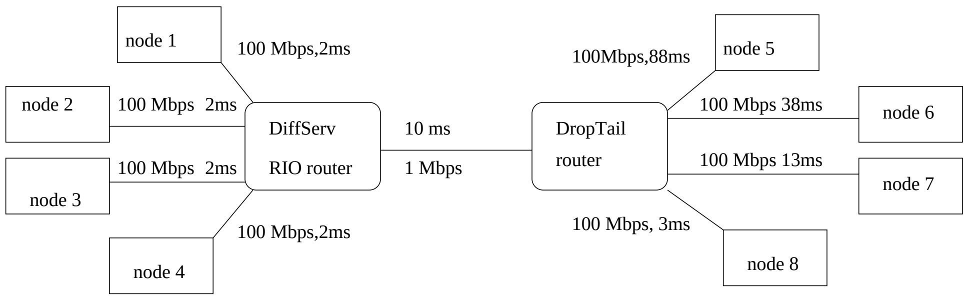


Figure 1.1: Problem Illustration by Simulation

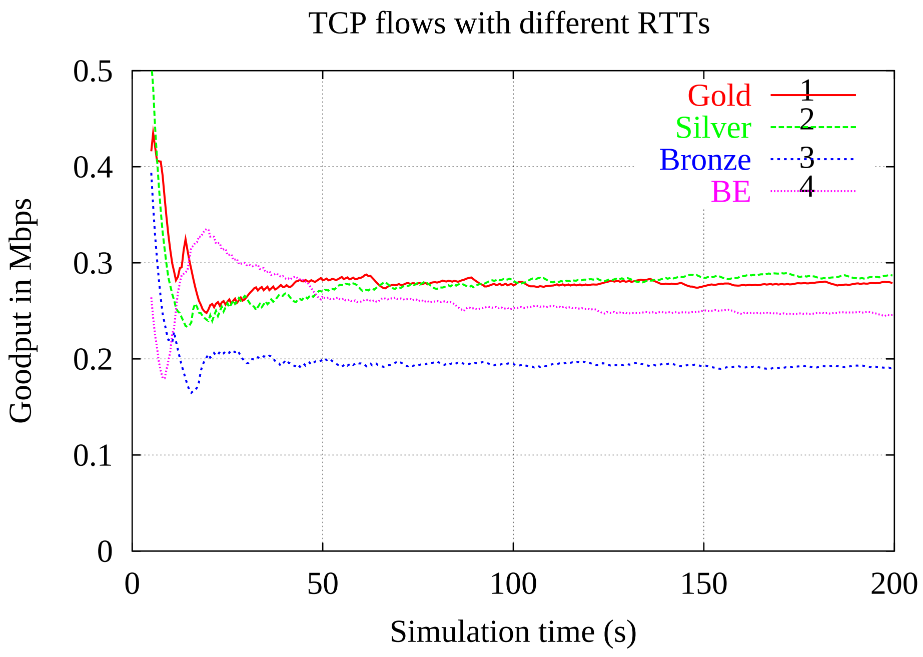


Figure 1.2: TCP flows with different RTTs

1.2.1 Problem Illustration by Simulation

The following simulations, executed with ns simulator [47], expect to demonstrate the problems of unfairness and congestive collapse in case of different Round Trip Time (RTT), UDP/TCP interaction, network provision state changes, and unfairness for micro flows inside the aggregate.

Experiment 1: TCP flows with different Round Trip Time

The conditions to evaluate the effects of heterogeneous RTTs in bandwidth sharing among TCP flows can not be easily generated, controlled and measured in a test platform. Therefore, simulation is a more obvious choice.

Using the well-known ns-2 Network Simulator [47], we built the layout depicted in Figure 1.1 to study this subject. Two routers, DiffServ RIO router and Drop-tail router, are connected through a 1 Mbps link; DiffServ RIO router with three drop-precedence levels is available at the ingress router. 4 bulk data TCP flows sent from the left to the right over the network with RTTs that go from 200 ms, 100 ms, 50 ms to 30 ms. The TCP traffic sent from node 1 to node 5 is marked as Gold class, the traffic sent from node 2 to node 6 is marked as Silver class, the traffic sent from node 3 to node 7 is marked as Bronze class, and the traffic sent from node 4 to node 8 is marked as Best effort class. The claim 40%, 30%, 20%, and 10% bandwidth of the bottleneck (DiffServ RIO node) respectively. In Best Effort class, TCP flow has short RTT, 30 ms. In other classes, TCP flows have long RTTs, 200 ms (in Gold), 100 ms (in Silver), and 50 ms (in Bronze).

DiffServ RIO parameters are set as follows: for each class queue, $min_{th} = 10$, $max_{th} = 40$, $maxp = 1/50$, respectively.

Figure 1.2 shows that the bandwidth share decreases as the RTT increases. The best effort (BE) traffic achieved more bandwidth share due to its short RTT. It can be said that Assured Forwarding (AF) does not assure an “Assured Bandwidth” service for TCP connections with different RTTs, since throughput depends on RTT. Sessions with small RTTs react faster to losses in the network, so that no big bursts are produced.

Experiment 2: UDP/TCP interaction

The layout depicted in Figure 1.1 was used. As in the previous layout, two routers are connected through a 1 Mbps link; the input are 3 TCP connections with same RTT that is 100 ms and they are marked as Gold (claim 40%), Bronze (claim 20%, and BE (claim 10% less) respectively. The UDP traffic (CBR) is marked as Silver which claim 30%. The UDP rate was 1 Mbps. Figure 1.3 shows that the UDP flow (Silver) consumes nearly 40 percent of the link-bandwidth beyond the TCP flow (Gold). Here we observe a well-known result: TCP is considerably affected by UDP traffic, which gets all the bandwidth associated to its rate. Nevertheless, the capacity not used by UDP, if any, is shared equally among TCP aggregates. When there is UDP traffic in the link, the high-rate aggregate (e.g. Gold) fails to obtain the desired rate (40%).

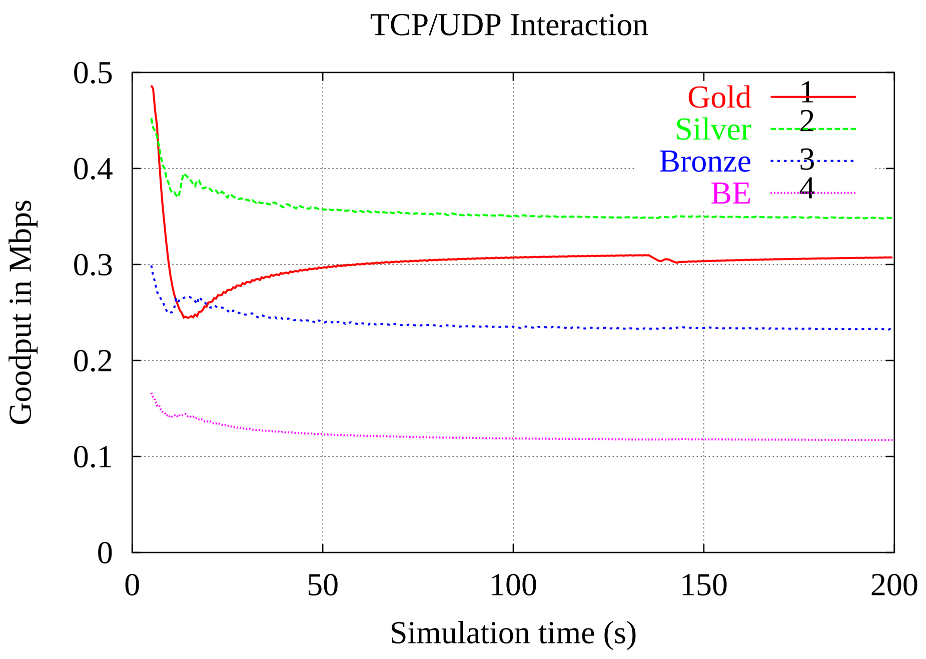


Figure 1.3: UDP/TCP Interaction

There are two initial conclusions of this test: the first one is that care should be taken into account when deciding if UDP traffic is to be aggregated with TCP traffic and processed in the AF. The second one is that bandwidth sharing can not be achieved for TCP flows in the presence of unresponsive UDP flows.

Experiment 3: Network provision state changes

In order to evaluate the effect of network provision state changing for TCP in bandwidth sharing, this experiment is designed to see if differentiation can be obtained.

The layout depicted in Figure 1.1 was used. As in the previous layout, two routers are connected through a 1 Mbps link. Four aggregates labeled Gold, Silver, Bronze, and BE. The committed rate of each aggregate is shown in Table 1.2.1.

Gold and Silver TCP traffic the same RTTs of 100 ms. Bronze and BE TCP traffic have the same RTTs of 50 ms. Figure 1.4 shows the result of the simulation when the bottleneck link state changing from 10 Mbps to 1 Mbps. It is clear that AF can not distribute bandwidth differentially in case of provision state change. This drawback might limit the flexibility of differentiation. Therefore, more research and experience in AF is needed to guarantee, at least in statistical terms, that the aggregates can get proportional

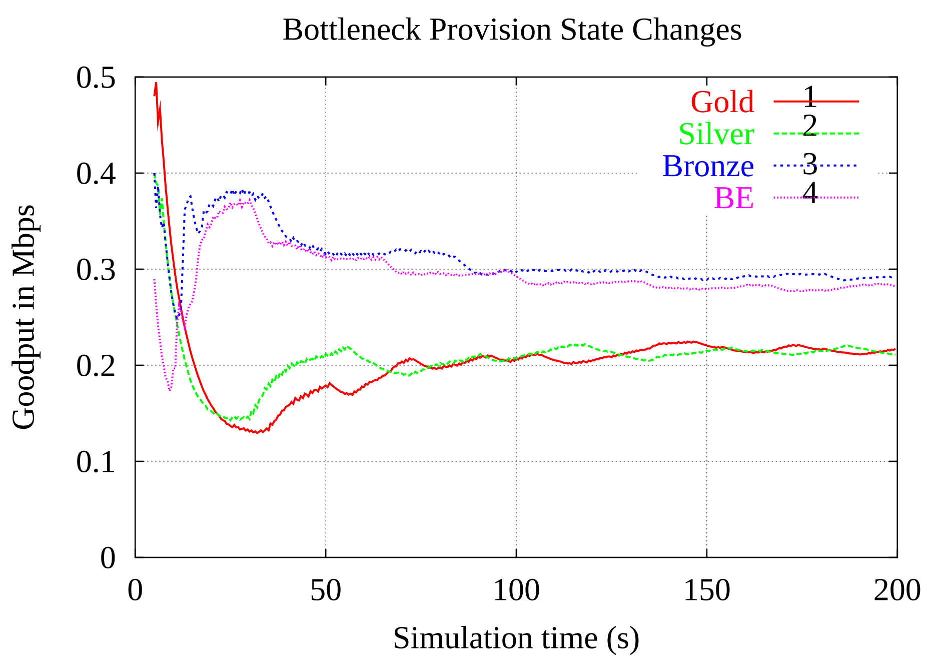


Figure 1.4: Network Provision State Changes

Aggregate	**Committed rate (kB/s)**
Gold	4000
Silver	3000
Bronze	2000
Best Effort	less than 1000

Table 1.1: Committed rates for aggregated TCP flows

bandwidth sharing regardless of provision state change.

Experiment 4: Bandwidth sharing for individual flows inside the aggregate

The layout depicted in Figure 1.1 was used. As in the previous layout, two routers are connected through a $1Mbps$ link. Four aggregates labeled Gold, Silver, Bronze and BE in Figure 1.1, each one composed of 4 individually marked flows which also have different RTTs, chosen from $200ms$, $100ms$, $50ms$ and $30ms$. The aggregates are marked with different committed rates, looking for a weighted sharing of the available bandwidth. The committed rate of each aggregate is shown in Table 1.2.1.

What happened with bandwidth distribution for micro flows inside the aggregate?

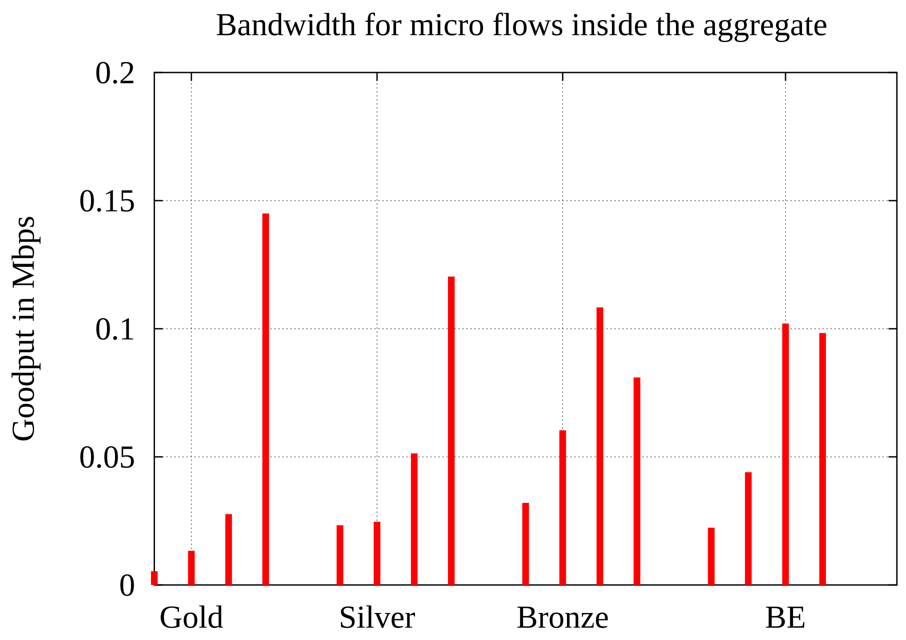


Figure 1.5: Bandwidth sharing for individual flows inside aggregate

Aggregate	**Committed rate (kB/s)**
Gold	400
Silver	300
Bronze	200
Best Effort	less than 100

Table 1.2: Committed rates for aggregated TCP flows

Figure 1.5 shows the corresponding plot. It is clear that the distribution is very unfair due to heterogeneity in RTTs.

1.3 Contributions

It has been observed from last section that DiffServ can hardly achieve the desired edge-to-edge QoS in case of different RTTs, UDP/TCP interaction, provision state change, and it can not achieve flow fairness for micro flows inside the aggregate. This is because the DiffServ and the Internet are unaware of their internal network QoS states. Even if the "inside" network is in congestion state, they still forward the overloading traffic into the network resulting in worse congestion situation.

To address the above mentioned problems, and more importantly towards providing a complete scalable QoS solution, in this dissertation we propose and investigate a *Resource Discovery and Fair Intelligent Admission Control over scalable Internet model (RD-FIAC)*. RD-FIAC addresses the QoS provisioning problem from two complementary aspects, namely the resource discovery and the fair intelligent admission control. The RD framework is core stateless. Internet core routers do not maintain any per-flow state and do not perform any per-flow operations. Moreover, both aggregate and per-flow state service guarantees can be supported within this framework. On the resource management control plane, we design and investigate fair intelligent admission control (FIAC). Research results obtained in this dissertation work have been or will be partially reported in [34–36, 38–41, 43, 44].

The fundamental contribution of this dissertation is the proposal of an architecture for providing per-class QoS and per-flow fairness inside the class that scalable to the Internet. To that effect, we consider the design and implementation of a two-plane architecture that provides assured service guarantees to classes and per-flow fairness among a class, while avoiding resource reservations.

We believe that end-to-end QoS could be obtained by analyzing end-to-edge, edge-to-edge, and edge-to-end localization QoS solutions. RD-FIAC architecture supports end-to-edge, edge-to-edge, and edge-to-end QoS solutions by its natural design.

RD-FIAC has several caveats:

- Service is the treatment provided by routers and other network elements to packets of an application as they traverse the path between source and destination domain. The RD-FIAC is able to break the limitations of the best effort service, by offering performance guarantees according to QoS metrics.

- End-to-end means that the communicating users may access the Internet from different administrative domains (network), i.e., they are not restricted to use services within their particular service provider. This distinction is important in the sense that there is a strong trend toward providing QoS guarantees for some intradomain services. One illustrative example is a virtual private network (VPN) for connecting headquarters and branch offices of the same company or creating extranets for e-business among different companies.

- An evolutionary approach is assumed, whereby the current Internet is the basis for a true converged network [18] (delivering data, voice and video). To put it more simply, this proposal does not require the current Internet to be abandoned and a brand new network to be built. The proposal aims at gradually introducing new QoS-based services in the Internet by tackling both problems and limitations of the best effort service and those from the interconnection of domains.

- The RD and FIAC architecture is proposed as an overlay (virtual) network over the scalable Internet capable of resolving the main issues related to QoS from a higher level of resource discovery, while keeping in mind practical solutions for providing the required QoS guarantees for the applications.

Most of these aspects are incorporated into the RD and FIAC architecture that was developed to provide edge-to-edge QoS and flow fairness in the Internet. It makes it possible to decouple QoS control from the normal data packet transmission. In order to provide flexibility to resource discovery and control negotiation, the control plane is introduced. The control plane generates Resource Discovery packet to detect network status and negotiation with admission control module. It plays a fundamental role in RD and FIAC. It can be seen as an overlay plane over the data plane and its operation and implementation do not cause further change to the existing infrastructure. Its main functions are resource discovery, negotiation with admission control, and resource management. In the data plane domains just function as DiffServ and its main responsibility is assuring that service is operating within the agreed quality levels.

1.3.1 Resource Discovery Protocol

We propose and develop a Resource Discovery (RD) framework to provide internal network QoS state dynamically. The concept of Resource Discovery Protocol is the closed-loop feedback mechanism. The edge routers periodically probe the network to obtain the current network QoS state. The ingress router is responsible for generating Resource

Discovery packet (RD packet) which is used to collect network QoS state information. The RD packet is referred and/or updated by core routers in terms of the available resources until it reaches the egress router. The egress router is responsible to send back the RD packet to the ingress router. Depending on the control mechanism, the ingress router may send RD packets to its connected interior routers on a per-flow or per-class basis. Moreover, the Resource Discovery Protocol is built on the DiffServ and the Internet architecture and it is backward compatible with existing DiffServ and the Internet for enabling interoperability.

We design and analyze several new resource discovery mechanisms to illustrate how both aggregate and per-flow service guarantees can be supported within this framework. Resource Discovery over Internet (RD-Internet) is applied to collect the current QoS state of the Internet domain. Resource Discovery over DiffServ (RD-DiffServ) is applied to collect the QoS state of the DiffServ domain. Moreover, we investigate the cost-performance trade-offs in supporting QoS in the Internet and DiffServ by studying the QoS provisioning power of these resources discovery algorithms.

1.3.2 Fair Intelligent Admission Control

To facilitate network resource management and flow admission control functions, two scalable Fair Intelligent Admission Control architectures are designed and investigated. Per-class FIAC handles the aggregate admission control for certain pre-defined aggregate. Per-flow FIAC handles the flow admission control for the class. They interact with Resource Discovery module for allocating and de-allocating the network resources along the path. In this way, the FIAC only needs to handle coarse time scale resource request from edge routers. Consequently, its scalability is greatly improved.

1.3.3 Edge-Aware Resource Discovery Protocol

Trying to minimize the core routers involvement, we propose an architecture called *Edge-Aware Resource Discovery*. The Edge-Aware Resource Discovery (Edge-Aware RD) measures the network QoS state indirectly at the edges, taking into account various factors such as SLA, edge queue variation, and available bandwidth variation at edge. The basic idea in Edge-Aware RD is that the available bandwidth variation at the egress shows a decreasing trend when the congestion is happening in the network. The simulations show that the Edge-Aware RD reports the available resources in a wide range of load conditions and path configurations. Also, the Edge-Aware RD does not cause significant increases in the network utilization, delays, or losses.

1.3.4 Edge-Aware Fair Intelligent Admission Control

Edge-Aware Fair Intelligent Admission Control (Edge-Aware FIAC) is based on Edge-Aware RD and applies the fair intelligent admission control algorithms to make admission decision.

Analytical models and approximate solutions are developed for both Edge-Aware Resource Discovery and Fair Intelligent Admission Control, which provide useful guidelines on how should be provisioned to construct an end-to-end QoS architecture.

1.4 Thesis Organization

This thesis is structured for introducing the concepts and proposals in a logical fashion. In this chapter, the problem of QoS in the Internet was introduced and the proposed solution was outlined. The remainder of this document is organized as follows. Chapter 2 reviews the background in the Internet QoS area. First, it analyzes the influence of the methods and structures for the interconnection of domains on the lack of QoS in the current Internet. Then, it presents some concepts involving QoS-based services in the Internet, including a proposal for a service life cycle. Finally, it introduces the main technologies that are being developed for deploying QoS in the Internet.

Chapter 3 introduces the Resource Discovery (RD) and Fair Intelligent Admission Control (FIAC) architecture, elaborating on the control and data planes and their interactions. It also presents the important concept of a RD and FIAC, which represents a group of domains willing to deploy the RD and FIAC architecture together with each other. The resource control activities for service provisioning are also described. We divide the Resource Discovery mechanisms into core-aware mechanism inside the DiffServ modes and edge-aware mechanism at the edge nodes only. Based on this network model we study core-aware Resource Discovery mechanism, in Chapter 4, edge-aware Resource Discovery mechanism, in Chapter 6, and using simulations, in Chapter 4 and Chapter 6, how the division of bandwidth between elastic and non-elastic flows depends on the DiffServ mechanisms with RD and FIAC.

Chapter 4 describes the strategy adopted the feedback loop built on the concept of the resource discovery.

Chapter 5 describes the main ideas related to Fair Intelligent Admission Control. The concepts developed for negotiation with Resource Discovery Protocol (RD). This chapter first describes the admission control model, comprised of incoming traffic and negotiations with RD. This chapter also gives some directions for the interaction of FIAC, which can use for multi-domains situation.

Chapter 6 aims at evaluating the Edge-Aware Resource Discovery and Fair Intelligent Admission Control architecture models, mostly based on a simulation study. The criteria for comparison are efficiency, fairness, scalability, reliability, and complexity and costs. The RD and FIAC model is found to be the best alternative, when comparing efficiency, fairness and scalability. It is also able to provide the right financial incentives for deploying QoS and resolves some of the current problems of the interconnection of domains in the Internet.

Chapter 7 outlines the extension and future works.

Chapter 8 concludes the thesis and discusses directions for future work.

Chapter 2

Background

Providing QoS in the Internet has been a topic of great interest for users, providers, vendors and researchers in recent years. The main motivation is turning the Internet into a true converged network, where real-time multimedia application can be deployed together with the more traditional data applications. However, the goal of achieving end-to-end or edge-to-edge QoS has to be fulfilled with the current Internet as the starting point, Therefore, this chapter aims at presenting background information on the main problems and solutions in the Internet QoS area and the challenges facing QoS deployment.

IETF has defined a *guaranteed service* [56] under its Integrated Services or IntServ architecture [15, 53], where end-to-end delay and bandwidth guarantees are provided for users on a *per-flow* basis. To support the IETF IntServ architecture, a signaling protocol, RSVP, for setting up end-to-end QoS reservation along a flow's path has also been proposed and standardized [55, 71].

Performing per-flow management inside the network, however, raises the important issue of scalability. Due to the complexities of per-flow operations both in the data plane and QoS control plane, the IntServ architecture may not scale well with the size of the Internet and the number of applications. As an alternative to per-flow based QoS provisioning, a different paradigm - the Differentiated Service or DiffServ - was later proposed and defined by the IETF [8, 57]. Under this paradigm, services are defined and implemented within individual administrative domains. To provide scalable QoS support, fine-grain user QoS control information is only maintained at the edge routers (i.e., ingress and egress routers) of an administrative domain.

In the following section, the most important issues regarding Internet QoS is exposed. Section 2.1 discusses the need for Quality of Service. In the Section 2.2, some concepts related to QoS are presented. In Section 2.3, the QoS approaches that have been seriously considered for implementing QoS in the last decade are presented. Finally, Section 2.4 summarizes the contributions of this chapter.

2.1 The Need for Quality of Service

IP QoS (Quality of Service) refers to the performance of IP packet flow through networks. There are several different definitions of QoS. Everyone who uses the Internet knows that it does not deploy QoS, but currently there is no comprehensive and unique definition of QoS. The definition adopted here characterizes QoS according to performance being measured by routers.

1. Quality of Service (QoS) is defined as the service quality that has to be guaranteed in terms of throughput, end-to-end delay, fairness, delay jitter, packet loss, and service reliability. Some applications are more stringent about their QoS requirements than others, and for this reason we have two basic types of QoS available: Resource reservation (e.g. integrated services) and Prioritization (e.g. differentiated services). For Resource reservation type of QoS, the network resources are apportioned according to an application's QoS request, and subject to bandwidth management policy. For Prioritization type of QoS, the network traffic is classified and the network elements give preferential treatment to classifications identified as having more demanding requirements. These types of QoS can be applied to individual application "flow" or to flow aggregates. A "flow" is defined as an individual, uni-directional, data stream between two applications (senders and receiver). An aggregate is simply two or more flows. Typically the flows will have something in common (e.g. same priority number). Resource reservation (e.g. integrated services) type of QoS is applied on a per-flow basis. Prioritization (e.g. differentiated services) type of QoS is applied on a per-aggregate basis.

2. The set of technologies that enable a network to make performance assurances.

This characterization is particularly useful when services are to be deployed cooperatively by a group of networks that implement different QoS technologies.

With respect to the level of guarantees a network is able to provide, QoS can be broadly classified into two types: resource reservation and prioritization. QoS based on resource reservation offers guarantees for each flow individually, for instance in IntServ. This would be ideal for end-to-end QoS, but it raises serious concerns with respect to scalability. Prioritization refers to giving a differentiated treatment to some classes of service, comprised of traffic aggregates, for example in DiffServ. With prioritization, there are no individual guarantees, but the scalability is enhanced.

Some driving forces that are leading to the QoS deployment in the Internet are users, ISPs, telecommunication operators and equipment vendors. Users want QoS-based services for use by different applications, such interactive voice and video, which face performance problems in the current Internet. With IP QoS, ISPs can achieve greater profitability through more business and more efficient use of bandwidth, enhanced service differentiation, better-than-best-effort service and customized solutions. Telecommunication companies that have been operating two different networks (PSTN and data networks) are willing to deliver both voice and data over a single converged network, in order to save the costs of maintaining a duplicate structure. Most vendors of equipments for the Internet have solutions for deploying QoS according to the current standards and want QoS solutions to be disseminated in the Internet, in order to increase their profits and market share.

2.2 Traffic in the Internet

2.2.1 TCP traffic in the Internet

The TCP mechanism is described in RFC 793 [52], dating back to 1981. The transport protocol is a connection oriented and end-to-end reliable protocol designed to fit into a layered hierarchy of protocols that support internetworking applications. The operative entities include a basic data transfer, reliability, flow control, multiplexing, connections, precedence and security.

During the connection establishment phase a TCP connection is formed with a three-way handshake mechanism with random initial sequence numbers sent by both sender and receiver to synchronize sequence numbers between the end-points. A TCP connection is uniquely identified by the 4-tuple of source and destination ports and addresses. The TCP header includes also the sequence number field for reliability, where the sequence number of the segment corresponds to the sequence number of the first byte.

The receiver advertised window (rwnd) and the acknowledgment (ACK) fields are needed for flow control. The sender also paces the datagram transmission according to the acknowledgment packets sent by the receiver. This is termed acknowledgment or self clocking.

To retransmit packets that are lost, adaptive retransmission is used. Segments are retransmitted if an acknowledgment packet (ACK) for the segment has not been received within a specific timeout. To this end, TCP has to have an effective and adaptive timeout as a function of both estimated round trip-time (RTT) and its variance. The RTT is sampled once rather than once per packet using a coarse 500 ms clock. Furthermore each time TCP retransmits, the next timeout is set to twice the previous timeout.

Linked with the study of TCP models and the resulting steady state throughputs is the study of how fair TCP is in dividing bandwidth between competing flows. The notion of fairness was set to assess this question. The study of pricing schemes is also strongly wrapped around the fairness concepts, mainly due to its economical heritage.

Fairness is a measure of the equal fulfillment of requirements of the network users in the same way as efficiency is a measure of the fulfillment of the technical requirements. As fairness measures the satisfaction of customer needs, it can be assessed using utility functions and social optima calculations. This has been the formulation in the work performed by Kelly et al. [31].

If a price is introduced that depends on the flow pattern x, we can formulate the measure of fairness as an optimization problem. Given a utility function $U_r(x_r)$, the optimization problem of maximizing overall utility results in charging and allocating network resources according to some fairness criteria. Besides the work by Kelly et al. different fairness concepts and TCP fairness have been studied by Massoulie et al. [46].

Ideally, assuming all customers are well behaving, the use of TCP should result in each flow receiving the same amount of service, e.g. capacity and packet loss. A natural

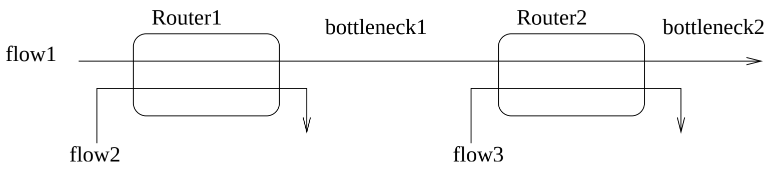


Figure 2.1: Simple Network

extension is to design a mechanism that offers the same amount of service inside a class, while between classes the service offered differs by a given proportion.

Fairness of TCP

The fairness of TCP can be evaluated in two ways. TCP uses an Additive Increase Multiplicative Decrease (AIMD) algorithm [12]; the TCP sending rate is controlled by a congestion window which is halved for every window of data containing a packet drop, and increased by one packet per window of data acknowledged. Based on this assumption, the max-min fairness principle has been proposed [51]. On the other hand the fairness of TCP can be evaluated based on how well actual TCP implementations are able to divide the capacity in a non-ideal network.

Kelly et al. [32] show that additive increase multiplicative decrease (AIMD) is proportionally fair. Based on the ideal model of TCP, which states that TCP is an AIMD mechanism, it can be thus deduced that best-effort TCP using congestion control is proportional fair.

Max-min fairness [51] intuitively means that the smallest demands for resources are satisfied first and whatever resources are left over are divided equally between the unsatisfied demands. Max-min fairness can be defined more precisely via the following procedure. Allocate each flow a rate of 0. Add all flows to the set of unsatisfied flows. Increase each unsatisfied flow's rate allocation equally until a flow becomes bottlenecked or all capacity in the network has been allocated. When all capacity has been allocated for a particular link, remove all flows from the unsatisfied set and add these flows to the set of bottlenecked flows. If the unsatisfied set is nonempty then resumes increasing the rate allocations of the remaining unsatisfied flows. The final rate allocation is the max-min fair rate allocation. Figure 2.1 shows a network containing two bottlenecks each with capacity μ. For this example, let $\mu = 1$. Flow 1 passes through two bottlenecks and sends with rate λ_1. Flows 2 and 3 each pass through only one bottleneck and send with rates λ_1 and λ_2 respectively. In this example, max-min fairness leads to the allocation $\lambda_1 = 1/2$, $\lambda_2 = 1/2$, and $\lambda_3 = 1/2$. Max-min fairness does not consider the possibility that reducing one flow might allow multiple other flows to increase their rates.

Ben Fredj et al. [26] have studied the statistics of the realized bandwidth share for elastic document transfers, including short lived flows. They conclude that with similar RTTs and one bottleneck link TCP tends to share bandwidth equally among all flows. Furthermore, they demonstrate that if the bandwidth allocated to a flow r on a link j is given by the TCP equation, where K is a constant, e.g. $\sqrt{\frac{3}{2}}$,

$$x_r = \frac{K}{RTT_r \sqrt{\sum_{j \in r} p}} \tag{2.1}$$

Several factors affect the ideal TCP model based on AIMD and affect thus the modeling of its fairness. The main factor is the bias of TCP against connections with long round trip times, which results from the linear increase of one unit per RTT.

There is a bias in the relationship of the throughput to the RTT, the current mechanisms have not done to remove the unwanted bias of the bandwidth share as a function of RTT.

2.2.2 UDP traffic in the Internet

TCP congestion control was designed as a response to the congestion collapse observed in the Internet in the late 1980s. However, not all forms of traffic want to use TCP as the transport mechanism. Namely, TCP together with its congestion control mechanism is suited for traffic that can not tolerate packet losses, i.e. one where a lost packet is always retransmitted. Therefore real time traffic rather use the other popular transport protocol of the Internet: UDP, where no flow control or congestion control is employed.

User Datagram Protocol (UDP) is a transport protocol that adds demultiplexing functionality allowing multiple application processes on each host to share the network. Processes indirectly identify each other using an abstract locater, e.g. the port. The UDP header includes the source and the destination ports. The UDP thus provides a connectionless datagram service, as opposed to TCP, which is connection oriented. UDP is also unreliable, as it does not use flow control, i.e. acknowledgments to clock its sending rate.

The effect of UDP flows is twofold. They can starve the bandwidth from the TCP flows, resulting in unfair division of capacity. The UDP flows may also, due to their unresponsive nature, bring to congestion collapse, where the network is in a state of transmitting packets that will be discarded before reaching the destination.

2.3 Approaches for QoS in the Internet

The need for service differentiation and QoS for the Internet became a topic of interest in the late 1980s and early 1990s, with the advent of networked multimedia applications. The discussion in this section focuses on most popular service architectures, and the mechanisms required to implement them.

The remainder of this section is organized as follows. In Section 2.3.1, we discuss the Best Effort service which has been using in the Internet. Then, in Section 2.3.2, we discuss the Integrated Service architecture briefly. In Section 2.3.3, we discuss in great details the Differentiated Service architecture which has been the starting point of our work. Last, in Section 2.3.4, we discuss Overlay network which has been proposed recently as future research area.

2.3.1 Best Effort Service

There is no formal service model definition for the default best effort service provided by the IP protocol. RFC 1812 [54] states that the IP protocol provides a connectionless service with no end-to-end delivery guarantees. Packets may arrive at the destination host damaged, duplicated, out of order or not arrive at all. Data delivery in IP has been designed to be undemanding, so that any underlying physical network can provide the service. The delivery is done through a connectionless datagram mode.

In the current Internet, the network resource allocation scheme is built on a set of congestion control mechanisms in both routers and end hosts. In routers, there are drop-tail queues that drop packets when the buffers are full. Lost packets in network are detected by TCP in the end hosts, and are taken as congestion signals. TCP slows down its sending rate upon detecting congestion.

At the network level, all packets are subjected to the same treatment from the network, i.e., forwarded according to the destination IP address and dropped if the routers are congested. The best-effort service model is simple and allows a great deal of statistical multiplexing, which leads to efficient use of the network resources. The pitfall of this approach is that without explicit support from the network, it is difficult to provide any kind of service assurance or guarantees to end host traffic.

2.3.2 Integrated Services

The Integrated Service (IntServ) model [53] aims at providing end-to-end QoS guarantees for individual traffic flows by means of reserving resource in every router all the way from source to destination. In addition to the basic best effort service, IntServ adds two new service bounds on delay and throughput for intolerant applications (that cannot adapt by any means to the network conditions). It was developed for providing a behavior similar to leased lines for IP users, but it has not have been deployed due to its high complexity. The controlled load service is a sort of predictive service that provides QoS guarantees similar to those experienced by a flow in a lightly loaded network. The idea behind the IntServ model is that there are no QoS guarantees without resource reservation.

RFC 1633 [53] gives an overview of Integrated Services (IntServ) as seen in 1994. The objective of IntServ, as given in [53], is to offer guaranteed and predictive services in addition to best-effort service and thus implicitly change the Internet best-effort service model. The new model uses controlled link sharing, implemented by resource reservations and admission control. The QoS requirements of the admitted flows are met through an explicit resource reservation setup mechanism, per flow state in the routers and advanced packet scheduling mechanisms. The measures used are latency and fidelity, a measure of loss, reordering or delay of packets.

IntServ proposes an implementation framework with four components: the signaling protocol, the admission control, the classifier and the packet scheduler. Applications requiring guaranteed service or controlled load service must set up paths and reserve resources before transmitting their data. The admission control component is responsible for deciding whether there are sufficient resources for granting the request. Whenever an IntServ packet arrives at a router, the classifier identifies that packet and put it on a specific queue to be forwarded. Finally, the packet scheduler will schedule the packet in

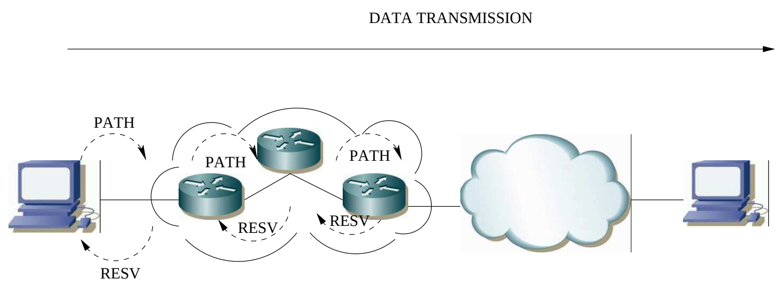


Figure 2.2: RSVP message processing

order to achieve its QoS target.

Although any signaling protocol may be used with IntServ, Resource ReSerVation Protocol (RSVP) [55] is the de facto standard. RSVP is a receiver oriented soft-state protocol, developed for applications to reserve resources in one direction in an integrated services network. It is based on the exchange of two messages by sender and receiver: the PATH message to the next hop, determined by the routing protocol. Upon receiving a PATH message, the receiver answers with a RESV message to actually request resources, if it is willing to accept the service request. Every router along the path can accept or reject the RESV message, according to the admission control routine. When a request is rejected, an error message is sent back to the sender. Otherwise, the amount of requested resources is reserved and the RESV message is passed to the next hop. Reserving resource implies installing state information for identifying every flow in their requirements. In order to cope with the routing instability of the Internet, RSVP is a soft-state protocol, that is, the reservations are only valid for a time period. Therefore, receivers must continuously send new PATH messages for keeping the reservations active.

The IntServ/RSVP architecture represented a major advance in the Internet, which was based on the concept that only end systems should maintain flow-related information. However, this model comes at some cost. Basically, two problems have been identified:

1. It generates too much state information for keeping the reservations active.

2. Too many signaling messages are exchanged for providing flow-level guarantees.

These two problems have a negative effect. It is believed that in the Internet core there are hundreds of thousands of simultaneous flows, which would impose a huge processing and storage burden on the routers. Therefore, although IntServ has not been implemented in the Internet core, it is widely accepted that it is not scalable enough for it.

The IntServ architecture requires per-flow classification in routers. Additionally, IntServ implementations require packet scheduling primitives, e.g., [10,49], which run a dynamic priority sorting algorithm. The scheduling overhead can become significant when the routers have to process a large number of packets within a short period of time. Also, the IntServ architecture relies on a signaling protocol (e.g., RSVP [55]) for reserving network resources, and on admission control for determining which flows can be admitted with the assurance that no service violation with will occur. Both of these mechanisms require that each router keep per-flow state information. Furthermore, it has been shown that using admission control mechanisms such as peak-rate

2.3.3 Differentiated Services

The Differentiated Service (DiffServ) architecture was developed within the IETF in the late nineties [57], and in several aspects it is still work-in-progress. The initial objective in DiffServ was a more scalable, manageable, and easily deployable architecture for service differentiation in IP networks. The major DiffServ premise is that individual flows, or microflows, with similar QoS requirements can be aggregated in larger traffic sets, called macroflows. All packets in a macroflow receive the same “forwarding behavior” in routers. So, a macroflow is the minimum level of granularity in a DiffServ network in terms of service differentiation. Each macroflow enters the network through an ingress edge router where it is conditioned and forwarded using a Per-Hop Behavior (PHB) which provides the service intended for the flow. A PHB is identified by a short label (currently six bits) in the IPv4 or IPv6 header, which is called Differentiated Service Code Point (DSCP).

Microflows are aggregated into macroflows at the edges of a DiffServ network. The edge can be a router that connects a microflow-aware network to a DiffServ network. The mapping from microflows to macroflows requires flow classification. This is a relatively expensive operation, but since it is only performed at the network edges, where the number of microflows is much smaller, it is expected to not cause scalability problems. Microflows are aggregated in macroflows based on rules set by the network operator. For instance, a macroflow can be all the traffic origination or destined to a certain organization, the traffic that is sent by a certain host, or all the traffic of a certain application.

The main elements of DiffServ are traffic classification and conditioning at the edge nodes and traffic forwarding through scheduling and discarding at the DiffServ interior nodes. The traffic classification and conditioning, i.e. the division of flows into per hop behaviors (PHB) and drop precedence inside PHBs, is done at the edge nodes. After the packets of the flows have been classified, the traffic is conditioned, more specifically, traffic is metered and packets marked into drop precedence levels.

Inside a DiffServ node all traffic functions are performed on aggregates based on the given PHB. Inside each PHB, a number of precedence levels exist based on the marking done at the edge node. In the scheduling unit two main elements affect the service experienced by the PHBs, the scheduling policy of the buffers and realization and relationship between the discarding thresholds of the buffers.

The DiffServ proposals considered here are Expedited Forwarding (EF) and Assured Forwarding (AF). We can categorize EF and AF to be of type assured service and relative service. Relative services are easier to realize, in the sense that in overload situations the capacity has to be divided in shares without having to give the contracted rates to the flows. With assured services and conditioning only at the edge of the network, it may happen that at a core link the total capacity is less than the sum of assured rates, and then without relative differentiation, no differentiation may result.

EF and AF have the status of an IETF RFC and give a conceptual service model of the PHB groups that could be implemented. Expedited Forwarding (EF), as explained in RFC 2598 [59], aims at providing a low loss, low latency, low jitter and assured bandwidth end-to-end service. Implementing EF requires that the nodes forward packets by at least a contracted rate. This can for example be implemented using priority queues, where EF traffic preempts other traffic. On the other hand, the EF traffic must be conditioned or policed so that the aggregate arrival rate of packets is not more than the contracted rate and so that the EF traffic does not starve the low priority traffic. Scheduling could also be implemented using Weighted Fair Queuing (WFQ) [7] with weights for EF in accordance with the contracted rate.

EF is a very strict service, in the sense that anything above the contracted rate is normally not admitted into the network. Therefore traffic of EF type do not have flexibility of sending more traffic at lower priority in times of low load. However, this is not the main aim of EF, as the use of EF is designed for a small set of flows requiring stringent delay bounds.

Assured Forwarding (AF) outlined in RFC 2597 [60] is designed to provide assured service to the traffic that is in profile, i.e. does not exceed the subscribed rate. It does not concern quantifiable delay or jitter requirements, but instead requires that a certain amount of forwarding resources are allocated to AF classes. AF is a per hop behavior (PHB) group with four forwarding classes and three drop precedence levels. Traffic may be conditioned to in profile or out of profile using a token bucket or leaky bucket mechanism. AF also requires that packets belonging to the same class are not reordered, for example, according to precedence level.

Different AF classes are used to offer different levels of assurance and each class is allocated a certain amount of buffer space and bandwidth. Inside an AF class packets are assigned a drop precedence, with the packets with the highest drop precedence discarded before the packets with lower drop precedence. Each AF class should be serviced so that at worst it achieves its contracted rate. An AF class may receive more forwarding resources if excess resources are available from other AF classes of PHBs.

Results of performance evaluation in the area of DiffServ are sometimes contradictory and several open questions remain to be answered. For instance, which end-to-end services can be implemented defining only per-hop behaviors, how to parameterize DiffServ mechanisms in order to optimize link utilization and service differentiation, how to balance the granularity of service/signalling-overhead trade-off, how to extend the DiffServ architecture for multicast support?

In addition to supporting the traditional best-effort service, the Differentiated Services architecture supports two per-hop behaviors: Expedited Forwarding (EF), and Assured Forwarding (AF).

Expedited Forwarding

The Expedited Forwarding (EF) was initially proposed by Jacobson et al. [61], and was refined in [59], as a service that provides a guaranteed peak rate service with negligible queuing delays or losses. While EF only provides local, per-hop guarantees, the objective is to use EF as a building block for network-wide services such as the Virtual Wire per-domain behavior. The goal of the Virtual Wire services is to provide each EF macroflow with a service equivalent to a virtual leased line, or a virtual circuit in ATM networks.

The authors of [59] envision that EF requires shaping at the network edge, so that EF traffic does not enter the network at a rate exceeding a peak rate R. A capacity of R is reserved in the entire network for EF traffic, so that EF macroflows do not experience delay or losses. The bandwidth reservation R is statically configured in a bandwidth broker. The bandwidth broker is a centralized agent configured with a set of policies, which determine the level of service different classes should receive. The bandwidth broker keeps track of the current allocation of traffic to different classes, and handles new requests to mark new traffic subject to the configured policy and current allocation. Routers in turn query the bandwidth broker to determine how much link capacity shall be reserved for EF traffic.

Subsequent research [9] showed that even with peak rate allocation for EF macroflows, an EF service cannot guarantee negligible losses and delays. Indeed, multiplexing EF traffic from several input ports in routers can result in bursty traffic, which, in turn, may cause delay and losses. This finding led to a change to the original definition of the Expedited Forwarding PHB. Instead of guaranteeing no losses and negligible delays, the authors of [62] propose to guarantee bounded delay variations to EF macroflows.

Assured Forwarding

The Assured Forwarding service is based on a proposal that was originally called the “Allocated Capacity framework”, introduced by Clark and Fang in [16]. In the Allocated Capacity framework, a class of traffic is provided with a certain bandwidth profile, defined by a rate R. As long as the aggregate amount of traffic from that class has a rate lower than R, traffic is marked as in-profile; otherwise it is marked as out-of-profile. In times of congestion, out-of-profile traffic is dropped more aggressively than in-profile traffic. In other words, a class is allowed to exceed its profile R when there is no congestion and the network load is low, but is restricted to sending traffic within its profile when the network is congested. The rate R is statically reserved, or provisioned, at network design time.

The AF service of the DiffServ architecture supports qualitative guarantees, but no classes are provided absolute service guarantees, and the difference in the service received by different classes is not quantified. While some have argued that Assured Forwarding provides absolute differentiation, because the profile R can be viewed as a throughput guarantee, we point out that in-profile traffic is not guaranteed a lossless service. Hence, traffic sending at rate $R^l < R$, thereby remaining in-profile, can still experience traffic losses, and obtain a service rate $R^{ll} < R^l < R$, which contradicts the notion that R is a throughput guarantee. The absence of throughput guarantee is clearly exhibited in the case of TCP traffic, as discussed in [70]: regardless of how well provisioned the network is, it may be impossible to provide throughput guarantees to TCP flows with the AF service. In fact, the only assurance that in-profile traffic gets in that, should congestion occur, it will not be dropped as aggressively as out-of-profile traffic. In other words, AF only provides isolation between different AF classes, and qualitative loss differentiation between the drop precedence levels within each class. We refer to the discussion in [20] to summarize concerns raised about the actual differentiation offered between different classes of traffic.

DiffServ Research

The AF and EF proposals having an IETF RFC status have been studied both analytically and through simulations. In reviewing the previous work, we emphasize the difference between providing assured and relative services. In the analytical work, attention is paid to the loss process characterization, especially in the studies where TCP modeling is enhanced to include differentiated services.

The papers by Sahu et al. [64] and Yeom [70] are analytical studies on the relationship between the contracted rate and the transmission rate of TCP flows. The work by Sahu et al. [64] models how the token bucket marking of flows based on their assured rate affects the received rate. As this is a DiffServ study, the marking of the flows is performed at the edge of the network while aggregates are forwarded inside the DiffServ node. The main result of the paper is that assured rate may not be achieved. If the assured rates are low, the link may be in under subscription state and the received rate is larger than the assured rate. Similarly, if the assured rates are too high, the link may be in over subscription state and less than the assured rate is received. The system model does not provide insight into how the ratio of received rates depends on the share of assured rates of different types of TCP flows. The model and the results are therefore applicable only in determining how assured differentiated services are achieved.

More specifically, the TCP process under consideration is of type Reno and is modeled as a renewal process, where the window is either reduced to half upon a triple duplicate ACK or to one upon a timeout. The sending rate of this TCP flow is then determined for two cases: under subscription and over subscription. In the under subscription case, only the lower priority traffic is discarded. In the over subscription case, the lower priority traffic is discarded with probability one, and the higher priority traffic with a positive probability.

The result for the over subscribed case illustrates the problem of dividing capacity according to assured rate. The received rate r in terms of assured rate A, bucket size B, round trip time T and loss probability p_1 is

$$r = min(A, \quad \frac{3(A + \frac{\sqrt{2B}}{T}}{4}, \quad \frac{1}{T}\sqrt{\frac{3}{2p_1}}) \tag{2.2}$$

The result shows that under an assured services discipline a TCP flow never receives more than its assured rate. Furthermore, when the received rate is below the assured rate, the received rate can be that of a TCP flow under no differentiation mechanism.

In the under subscription case, the result is not so easy to interpret, but if the assured rate is small enough, the extra capacity of the link is divided equally according to the TCP equations. Thus the TCP flow receives its assured rate plus an equal share of the link capacity.

The reason for this kind of division is in the token bucket marking scheme and in using two priorities. More priorities give the possibility of having many assured rates, thus introducing more leverage into the differentiation.

The paper by Yeom and Reddy [70] gives a better model for assured services. They model the interaction of many TCP flows and have two or three priority levels in their model, but do not study the effect of the token bucket size on the bandwidth share.

The system model is again one with token bucket marking and given loss rates for the different priorities assuming that RED is used. The division of the model into under and over subscription cases was first proposed by these authors. In cases of over subscription, the assured rate may not be reached and the capacity is divided equally, with the received rate being the rate for a TCP flow when no differentiation is present. Even in the under subscription case, those flows with small assured rates may have a received rate almost equal to a flow with a higher assured rate.

Under subscription case is defined as the case when only out of profile packets are dropped from the network. However, in the case of many TCP flows with different assured rate, the network would be over subscribed if all flows had the highest assured rate. Then those flows with the highest assured rate are only able to receive their assured rate up to the threshold $A = \frac{3}{T}\sqrt{\frac{2}{p_{out}}}$, corresponding to $\sqrt{3}$ times the best-effort share, when no reservations are used. Below this assured rate, the TCP flows receive their assured rate and some excess rate.

Some previous simulation studies have studied the effect of having both TCP and UDP flows in the network. Goyal et al. [28] study different factors related to differentiation, namely related to AF. These include: number of priority levels, percentage of highest priority traffic, buffer management and traffic types. The simulations assume many TCP flows, but a fixed number of flows. The system model is similar to the ones of assured services presented in the analytical studies. Based on assured rates, two cascaded token bucket markers mark the packets to three priorities: green, yellow or red.

As a result they show the importance of three priority levels in distinguishing between congestion sensitive and insensitive flows. The marking into priorities is done so that in the case of over subscription of the network the priorities are redundant, and thus the case of over subscription is not studied. This is always the case when assured services are used, as highest priority is given to a flow conforming to assured rate. Differentiation would occur in the over subscription case if middle priority would be given to a flow conforming to its assured rate. Then a flow would reach higher priorities by dropping the rate below the assured rate.

An interesting conclusion of the paper is to mark excess insensitive flows, e.g. UDP traffic to lowest priority, while excess congestion sensitive flows, e.g. TCP traffic receives middle priority. Thus UDP traffic is given less priority levels than TCP traffic.

Pieda et al. [50] study the claim made by Goyal et al. that TCP flows can be protected from UDP flows by marking out of profile UDP flows to lowest priority, but out of profile TCP flows to middle priority. They divide the study of priority levels into six scenarios. In each scenario three priority levels are used, but packets of a flow are only marked in or out of profile. They also compare the effect of using RED with overlapping discarding thresholds and different dropping probabilities compared to using RED with non-overlapping thresholds and different dropping probabilities.

As a result they show that when TCP and UDP are marked to the same AF class, three priorities are not enough to achieve the three targets: that in an over provisioned network both TCP and UDP target rates are achieved, that UDP and TCP out of profile packets should have a reasonable share of excess bandwidth, and that in an under provisioned network TCP and UDP flows experience degradation in proportion to the assured rate.

The scenario where TCP in profile packets have highest priority, UDP in profile packets have middle priority, and all out of profile packets have lowest priority is able to guarantee assured rates, and the relative deviation from assured rate is the smallest. However none of the scenarios is able to meet all three differentiation targets.

The simulation and analytical studies on AF showed that when assured services are used, it is difficult, even impossible, to guarantee division of excess or division of limited capacity in ratio of assured rate or weight. Furthermore, Pieda et al. concluded that the best way of protecting UDP and TCP traffic from each other is separating them into different AF classes.

The DiffServ, AF and EF specifications given in [57, 60] do not impose a particular scheduling or buffer management algorithm. EF can for instance be implemented using well-known fixed-priority scheduling algorithms, or rate-based algorithms, e.g., Class-Based Queuing (CBQ).

The Assured Forwarding service, on the other hand, can be enforced with buffer management algorithms. Indeed, as long as the network is correctly provisioned, i.e., enough link capacity has been reserved in advance for each class of traffic, scheduling in Assured Forwarding can be realized with a first-in-first-out (FIFO) discipline. Service differentiation can be enforced by marking packets as in-profile or out-of-profile, and using a buffer management algorithm that drops out-of-profile packets more aggressively.

2.4 Service Overlay Networks

An overlay network is a “virtual” network created on top of an existing network. The overlay network creates architecture of a higher level of abstraction, so that it can resolve a variety of problems that are very difficult to be dealt with at the router-level [1]. The nodes of an overlay network are usually responsible for a certain aspect of an entire network or part of it. Examples of overlay networks include application-layer multicast [13], Web content distribution networks, congestion management, peer-to-peer networks, content delivery networks, QoS, and resilient overlay networks [1]. Some overlay networks have been proposed in the past years for providing QoS in the Internet. Most of them consider DiffServ as the basic underlying QoS technology. Motivated in part by the positive results of these approaches for specific network services, some researchers are seeking to investigate if an overlay network can do for the Internet QoS.

Overlays permit designers to implement their own routing and packet management algorithms on top of the Internet. The Internet itself began as an overlay network on top of the telephone network, using long-distance telephone links (that consist of multiple physical links) to connect Internet routers. Modern overlays operate similarly, using the Internet paths between end-hosts as “links” upon which the overlay routes data, building a network on top of the network. As a result, overlays can be used to deploy new functionality instead of requiring years of upgrades to Internet routers; they also present developers with a flexible and powerful platform on which to create new services.

benefit of overlays is that they avoid burdening the underlying network with features better performed at higher layers.

The Bandwidth Broker (BB) [61] was the first attempt for building an overlay network for resource management in DiffServ network. It can be carried out manually, or semi-automatically. A BB is a logical entity playing two main roles in a DiffServ domain:

1. Intradomain resource management: related to provisioning resources for specific PHBs and configuration of traffic conditioning mechanisms, according to organizational policies.

2. Interdomain resource negotiation: BBs responsible for different DS domains negotiate with each other in order to discover and provision resources for end-to-end

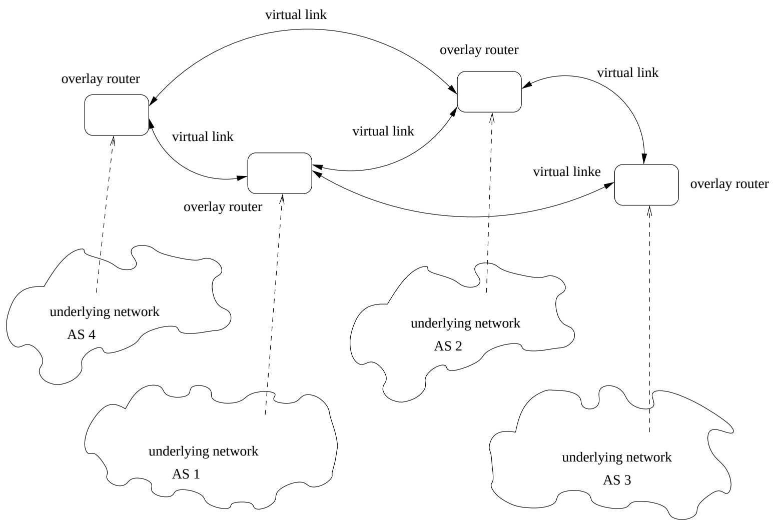


Figure 2.3: Overlay network framework

advanced services.

The SON architecture [19] studied the bandwidth provisioning problem for the service overlay networks. They took into account various factors such as service QoS, traffic demand distributions, and bandwidth costs to tackle bandwidth provisioning problem.

The OverQoS [67] proposed an architecture for providing Internet QoS using overlay networks. OverQoS empowers third-party providers to offer enhanced network services to their customers using the notion of a controlled loss virtual link (CLVL). The CLVL abstraction bounds the loss-rate experienced by the overlay traffic and is used to provide differential rate allocations, statistical bandwidth, loss assurance, and enables explicit-rate congestion control algorithms. When QoS is provided at the IP layer, an IP router has total control over its packet buffers and the output link bandwidth, and can directly schedule these resources. In contrast, an OverQoS node controls neither the bandwidth nor the losses on the underlying IP path. They show that the OverQoS can provide approximate statistical assurances when used in conjunction with adaptive admission control schemes.

Figure 2.3 describes the overlay network architecture. A virtual link is the underlying IP path connecting two overlay nodes. A virtual link is unidirectional and carries traffic from an entry overlay node to an exit overlay node. A bundle is the stream of application data packets carried across the virtual link; it typically includes packets from multiple transport-layer flows across different sources and destinations.

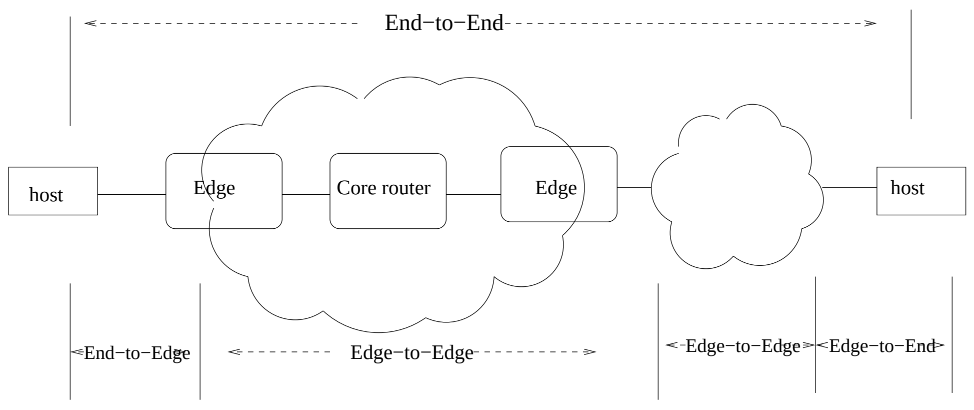


Figure 2.4: End-to-End QoS

2.5 The Challenge of End-to-End QoS

The major challenge in the Internet is to provide end-to-end QoS guarantees, as opposed to only covering part of the path packets have to follow from one communicating host to the another. As end-users see QoS through applications, the notion of QoS guarantees needs to encompass the path from one application all the way to the other application. In other words, both the end-to-end network path between communicating hosts and the top-to-bottom and bottom-to-top paths must be considered in order to guarantee end-to-end QoS.

Figure 2.4 depicts the scenario of end-to-end QoS. By the end-to-end perspective, service may be also classified according to their geographic scope. An intra-domain service is to be used only within the boundaries of a domain, which is called as edge-to-edge QoS. An end-to-end service may be used when data source and destinations are located in distinct domains, possibly with several other domains along the path between them which can be analyzed as end-to-edge QoS, edge-to-edge QoS, and edge-to-end QoS

Figure 2.5 depicts the top-to-bottom QoS. From the top-to-bottom perspective, QoS can be analyzed into application layer QoS, network layer QoS, datalink layer QoS and physical layer QoS. Each OSI layer from the application down must also support QoS to assure that high-priority send and receive requests can get high priority treatment from the host's network system. The important point is that the application itself should be QoS-enabled, in order to make use of the available guarantees. In other words, the application should be written with QoS in mind, by means of a QoS API provided by a specific QoS

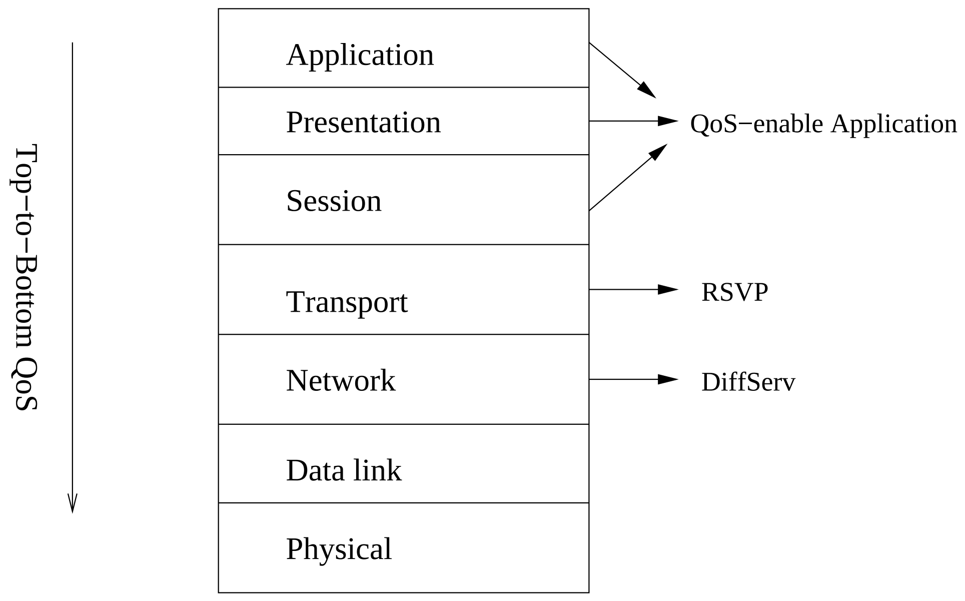


Figure 2.5: Top-to-bottom QoS

middleware. ReSerVation Protocol (RSVP) [55] provides the signaling to enable network resource reservation. Although typically used on a per-flow basis and provides feedback to application. Differentiated Service (DiffServ) [57] provides a coarse and simple way to categorize and prioritize network traffic aggregates. DiffServ can be applied at network core ingress level. Subnet Bandwidth Management (SBM) [69] is a signaling protocol that allows communication and coordination between network nodes and switches in the SBM framework and enables mapping to higher-layer QoS protocol. The problem of top-to-bottom QoS is typically tackled by QoS architectures [2].

This thesis only deals with the part of the problem of end-to-end QoS and even so, within a limited scope.

2.6 Summary

We surveyed major QoS approaches for the Internet. It is evident that Internet users are seeking QoS to be deployed in the Internet, it is not a reality today. The IntServ architecture does not scale well with the size of the Internet and the number of applications. On the other hand, the DiffServ architecture aims to provide scalable QoS support. The fine-grain user QoS control information is only maintained at the edge routers (i.e., ingress and egress routers) of an administrative domain. The user traffic is appropriately conditioned or shaped before injected into the network core. At core routers, no per-flow QoS state is maintained. It remains to be seen whether such a service model would be sufficient to meet the user QoS requirements fairly and efficiently.

Indeed, the exact QoS provisioning power that can be provided by DiffServ is still under great debate. For example, in the DiffServ framework, how to allocate the unused network resources fairly and efficiently? how to allocate the network resources fairly in case of congestion state?

We believe that one missing characteristic of the Internet with DiffServ is internal-unaware.

Service Overlay Network can be used to facilitate the creation and deployment of value-added Internet services such as VoIP, Video-on-Demand, and other emerging QoS-sensitive services. In addition to its ability to deliver QoS sensitive services, the Service Overlay Network can decouple application services from network management service, thereby reducing the complexity of network management and control.

The research community seems to have reached a consensus that per-class architectures will be a viable solution for providing service guarantees in the Internet, because class-based architectures have the advantage that they work with simpler algorithms for enforcing QoS guarantees than per-flow architectures, and can be deployed with only minor changes to the network architecture.

The next chapter presents the Resource Discovery and Fair Intelligent Admission Control Architecture, which aims at the adding some contributions to improve QoS in the Internet, based on the afore-mentioned findings of this chapter.

Chapter 3

A Framework for Resource Discovery and Fair Intelligent Admission Control over Scalable Internet

In this chapter, an architecture for enabling the deployment of QoS-based edge-to-edge services for scalable Internet is proposed and it is called the Resource Discovery and Fair Intelligent Admission Control architecture (RD-FIAC). As stated in Chapter 2, obtaining end-to-end QoS is a complex task, due to limitations of the technology, and topology. We believe that end-to-edge QoS, edge-to-edge QoS, and edge-to-end QoS are all necessary stages to obtain end-to-end QoS.

Our aim through the RD-FIAC architecture is to provide Prioritization type of QoS (refer to Section 1.2, Chapter 1) on each of the above stages. We will use the framework described in this chapter as a basis throughout this dissertation. Most work in this thesis was devoted to the resource discovery and admission control, described in more details in Chapter 4 and Chapter 5, respectively. The high level design of the RD-FIAC architecture is presented in this chapter. This dissertation will show that using algorithms based on RD-FIAC enables us to enhance QoS of class-based service differentiation without sacrificing scalability.

In the sequence of this chapter, Section 3.1 presents an overview of the RD-FIAC architecture, which puts the above-mentioned functions into two logical planes. Section 3.2 presents a more in-depth view of Resource Discovery and Fair Intelligent Admission

Control in terms of End-to-End QoS. Section 3.3 draws some comments on the main topics discussed in this chapter.

3.1 Overview of the Resource Discovery and Fair Intelligent Admission Control Architecture

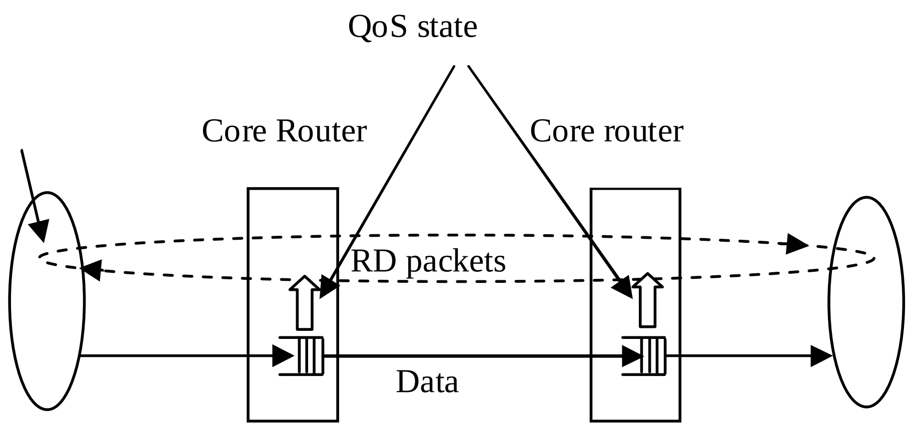


Figure 3.1: The RD-FIAC architecture–conceptual view

The basic building block of our solution is called Resource Discovery and Fair Intelligent Admission Control (RD-FIAC) over Internet while core routers maintain no per-flow state.

We will use the DiffServ architecture as the underlying scalable network. Since the normal DiffServ can hardly achieve the desired edge-to-edge QoS under traffic overloaded and network state changes, RD-FIAC architecture is designed to address these drawbacks.

The Resource Discovery protocol (RD) is a closed-loop feedback mechanism working between ingress edge router and egress edge router. On the forward path, ingress edge routers perform two functions. They (a) initiate the RD packet, mark the Differentiated Services Code Point (DSCP) (e.g. EF class) in the DSCP-field of RD packet, and (b) generate RD packet in a constant rate (or proportional to the incoming traffic rate). In turn, the core routers forward RD packets to collect the network QoS state information. On the backward path, egress edge routers terminate the RD packet and send them back to ingress edge router with updated network QoS state information along the path. Upon receiving backward RD packets, ingress edge routers report the network capability to the Admission Control Module.

Fair Intelligent Admission Control (FIAC) is a predictive control scheme which performs two functions: (a) it estimates the amount of class traffic expected to flow into the domain based on history of the traffic, and (b) it makes admission decision based on the incoming traffic and network QoS states report from the Resource Discovery mechanism.

A natural question is why use the RD-FIAC over DiffServ that offers relative bandwidth differentiation instead of the service that uses a fixed bandwidth profile such as Service Level Agreement (SLA)? After all, the Assured Service arguably provides a more powerful and useful abstraction; ideally, a user is guaranteed a fixed bandwidth irrespective of where or when it sends traffic. In contrast, with RD-FIAC, the amount of traffic that can be controlled varies as a result of the congestion along the paths to the destination. The simple answer is that, while the fixed bandwidth profile such as SLA is arguably more powerful, it is unclear whether it can be efficiently implemented. The main problem follows directly from the service definition, as a fixed bandwidth profile service such as SLA does not put any restriction on where or when a user can send traffic. This results in a fundamental conflict between maximizing resource utilization and achieving a high service assurance. In particular, the network needs to provide enough resources to all possible destinations, but the network has no idea about the current network state. In the worst case, when the traffic traverses the congested link in the network, this will result in severe resource underutilization, which is unacceptable in practice.

In contrast, RD-FIAC can achieve better performance in terms of resource utilization. The fixed bandwidth profile such as SLA and the incoming traffic rate are considered as parameters of the “external” traffic QoS requirements. The traffic can obtain the network resources depends on the network QoS states: the more congested the network, the lower the resources the traffic can obtain.

3.1.1 Design Goals and Assumptions

In this chapter, we propose a Resource Discovery and Fair Intelligent Admission Control architecture, which we call RD-FIAC, to detect network state and then to make appropriate admission control. The RD-FIAC enhances the QoS and performance of the DiffServ network by adding some functionalities to the network routers (mainly on edge routers) and leveraging information from traffic monitoring devices.

The main goals that drive our design of the RD-FIAC architecture are:

- **QoS provisioning:** The RD-FIAC attempts to provide edge-to-edge Prioritization type of QoS (e.g. differentiated service) by detecting network state. The admission control, FIAC, is performed only at the edge routers, but it should take into consideration the "internal" network state, RD information, and "external" requirement, incoming traffic SLA.

- **Scalability:** The RD-FIAC is on the overlay control plane that can scale to support a large underlying network (i.e., large geographic regions and large volume of simultaneous calls), which might be DiffServ domain or the Internet. It inherits the scalability from DiffServ. We strive to minimize the number of states maintained in each node of the RD-FIAC and the backbone routers.

- **Dynamic Network State:** The RD-FIAC attempts to detect current network QoS state by performing Resource Discovery Mechanism and passes the network QoS state information to admission control module.

- **Fair Intelligent Admission Control:** In RD-FIAC architecture, the ingress router takes admission decision based on "internal", the network QoS states, and "external", the incoming traffic QoS requirements. In this way, the RD-FIAC is able to provide fair intelligent resource control by coordinating the communications between "internal" and "external".

- **Support for end-to-end QoS:** The RD-FIAC infrastructure could be extended to support end-to-edge, and edge-to-end stages by coordinating each domain RD-FIAC at different level. This is part of future work.

We address mainly the above four design goals in our thesis work. Specifically, we design Resource Discovery mechanisms within RD-FIAC to probe the network state.

RD framework has three key components: RD packet, edge router mechanism, and core router mechanism. In the following we briefly discuss each of them one by one.

- **RD Packet:**

  The key role of RD packet is to detect current router state. The RD packet is initialized and released at the network edge router. Depending on the type of services supported by the network, some additional classification information may be carried in the packet header, such as the class identification that the packet belong to. As we will see shortly, core routers rely on its state to update RD packets.

- **Edge Router Mechanism:**

  We classify the edge routers into the ingress router and the egress router.

  At the ingress router, it generates the RD packets to collect the "internal" network QoS states, negotiates the network QoS states with the incoming traffic QoS requirements. When receiving the backward RD packet, the ingress router extracts the network QoS state information from the RD packet and inform the admission control module.

  At the egress router, it terminates the RD packets and send them back to the ingress router with updated network QoS state information. Upon receiving backward RD packets, the ingress nodes report the network capability to the Admission Control Module. By this way, the admission control module keeps the latest the QoS states.

  Another important role of an edge router is to shape the traffic to satisfy the "external" traffic requirements and "internal" network QoS states.

- **Core Router Mechanism:**

  Depending on the specific packet algorithm implemented in network, core routers may update the network state carried in the RD packets.

- **Bandwidth and queue control functions:**

  The bandwidth and queue control functions are designed to calculate the fairshare of available bandwidth and the degree of network congestion it can tolerate.

  For further details, please refer to Chapter 4.

  In designing the RD-FIAC architecture, we make the following assumptions:

1. The networks are capable of providing different service levels through a combination of packet marking, scheduling and queue management mechanisms. We assume network edge routers can verify whether the QoS assurance agreement is met by measuring the packet loss, throughput, average queuing delay, delay variance, etc.

2. Every routing domain has the capability to monitor and collect statistics of the incoming and outgoing traffic. We assume this information is trustable, and will be used by RD-FIAC to negotiate or exchange resource discovery information with neighboring domains.

3. Control plane (e.g., RD packet and FIAC) and data plane (underlying normal DiffServ) are separated. We decouple RD-FIAC and normal DiffServ procedures to reduce the overall response time and to be transparent for underlying network.

3.1.2 The Core-Aware Resource Discovery and Fair Intelligent Admission Control Model

The RD-FIAC architecture is an overlay network aimed at providing QoS in the Internet. It is divided up into two logical planes to discover available resources, and control of proper operation of the contracted traffic. This design is inspired by the dual view of QoS as being both the resource allocation for applications and the underlying technologies for enabling the network to provide such services.

The basic centralized RD-FIAC model for the resource discovery and fair intelligent admission control of a network domain is schematically depicted in Figure 3.2. In this architectural model, the RD mechanisms detect the network state regarding the network domain. The Resource Discovery and Fair Intelligent Admission Control together provide a logical representation, the QoS control plane, of the network domain and its entire state. With this QoS control plane, the RD-FIAC performs resource discovery and QoS control functions. In this sense, the QoS control plane is decoupled from the data plane that is normal DiffServ or Internet domain. The core routers are removed from the QoS control plane: core routers do not maintain any QoS state information, whether per-flow

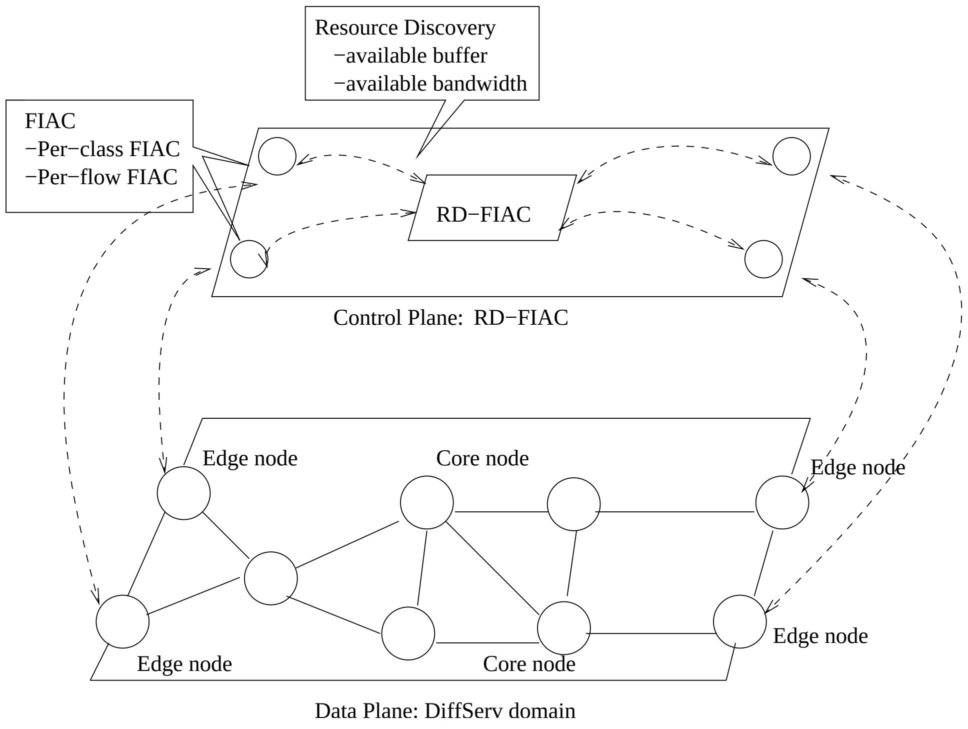


Figure 3.2: Illustration of RD-FIAC model

or aggregate, and do not perform any QoS control functions such as admission control.

We consider an individual router in the network. For the presentation of this framework, we consider that all traffic backlogged in the router is queued in the transmission queues at the output links. In other words, we assume that the router under consideration uses output queuing architecture. In Figure 3.3, we give a simplified representation of the router architecture we consider. The router consists of N input and output links, connected by a fabric interconnect. Demultiplexers are used to direct packets from the input links to the proper output links. Each output link is governed by a transmission queue. In the figure, packets flow from left to right. The packets, marked as gold, silver, bronze or BE, are in contention for the same output link. The output queuing architecture implies that the router interconnect is fast enough to move packets from all input ports to the output ports and completely avoid any contention at the input links or in the switch fabric.

By avoiding traffic backlogs at the input links or in the fabric, the output queuing assumption enables us to solely focus on the operations performed at the output links.

In addition to assuming output queuing router architecture, we take a fluid-flow in-

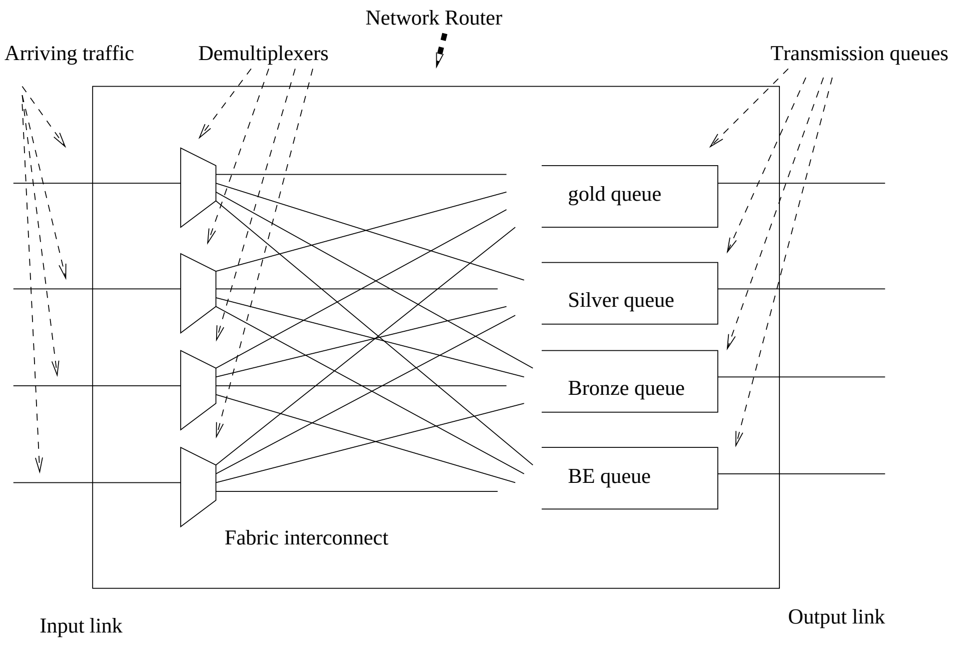


Figure 3.3: Router Architecture

terpretation of traffic. That is, the output link is viewed as simultaneously serving traffic from several classes. Since the actual traffic is sent in discrete packets, a fluid-flow interpretation of traffic is idealistic.

In our RD-FIAC model, the network QoS states and management are represented at control plane. The RD module maintains the network QoS states information regarding the QoS states of each ingress-egress pair in the network domain, such as the explicit feedback rate. The FIAC module makes the appropriate admission decision based on the network QoS state and traffic requirement (e.g., SLA). As will be demonstrated in this chapter, this two-plane representation of the network is also the key feature that leads to scalable design for dynamic QoS provisioning. Lastly, we note that the QoS states are aggregate QoS states regarding the ingress-egress pair. No per-flow QoS states are maintained in the core router. The QoS and other control state information regarding each flow is maintained by the FIAC module at edge router.

We now briefly describe the operations of RD-FIAC scheme to illustrate how dynamic QoS provisioning can be performed under the basic RD-FIAC architecture. We consider the Resource Discovery (RD) packet generation first.

For each traffic class, the ingress router introduces a Resource Discovery (RD) packet at a rate proportional to the traffic rate for that traffic class (or assign a special class, Expedited Forwarding, EF) destined for the destination egress edge router. The RD packets contain a vector of QoS parameters as well as traffic parameters for the ingress-egress pair.

The first core router exercises an intelligent bandwidth allocation algorithm to estimate the bandwidth fair share for this class based on the traffic parameters contained in the RD packets, and to determine the explicit QoS parameters it can support. The core router then consults its QoS state for that Differentiated Service Code Point (DSCP) class and modifies the parameters of the RD packets it can support accordingly. The router then forwards the RD packets to the next router along the path to the destination edge router.

When a RD packet arrives at the egress edge router, the egress router consults its QoS state and modifies the parameters accordingly. The egress router returns the RD packet containing the maximum QoS capability that can be supported from the ingress to egress. By doing so for each of the DSCP class, the ingress discovers QoS capabilities of the path.

Armed with this feedback mechanism for QoS state discovery, the Fair Intelligent Admission Control (FIAC) module will be able to perform several QoS-related functions. It is able to perform local admission control function to map per-class QoS to be appropriate to aggregate QoS in the DiffServ network region; it is able to report the QoS information to higher traffic management authority for overall traffic engineering, or it may interwork with the emerging Multi Protocol Label Switching (MPLS) technologies for building Label Switch Path (LSP) with adequate QoS attributes.

The crucial task is to invent a feasible feedback loop on network to discover available resources, and to devise an effective fair intelligent admission control with adequate QoS state information. From previous works in the area of TCP congestion control and ATM network we believe that a feasible feedback loop between edge routers can be applied. The second task is then to devise a fair intelligent admission control algorithm at the edge routers. For this task we rely partly on the FIAC we recently developed. However, the FIAC is designed for the connectionless DiffServ environment so that we need the routers that can identify the particular aggregate for RD packets.

The aim of RD-FIAC is to employ a feedback resource discovery mechanism, taking into account various constraints related to congestion bottleneck, available buffer, and available bandwidth, to inform FIAC module so that it could make appropriate admission decision.

The essential ideas of the RD-FIAC are as follows:

- The algorithm has to guarantee a minimum rate to all connections that are admitted to the network by the network admission control procedure.
- It has to keep track of the amount of bandwidth available for allocation.
- It has to keep a running average account of current share allocation for each class.
- If some classes are restricted by their peak rate and cannot take up their allocated fair share, the unused bandwidth is allocated fairly among the rest of the classes.
- It has to observe the current transmission rate of each class and make appropriate adjustment to bring the current share allocation of each class to the target fair share for that class.
- In case where a connection is throttled by a switch along its path to the destination, the RD algorithm has to monitor the situation and inform FIAC module.
- In case where congestion is a problem, the FIAC algorithm exercises its admission control mechanism to stabilize the network around a target operating point.

This architecture in two planes provides an integrated view of the network, although the actual deployment of an end-to-end service may need the cooperation of many networks that possibly implement distinct QoS technologies. Figure3.4 shows the relationship between two planes.

The greatest advantage of RD-FIAC architecture is that it is compatible with current Internet and DiffServ domain without any effect on actual data transmission. All the resource discovery and admission control functions are within the control plane.

The RD-FIAC architecture can be seen as an overlay network that extends the Internet architecture in order to deploy QoS service and available resource allocation. It follows the model of network architectures based on layers, and protocols (e.g. agents). To a large extent, the description of the architecture is dedicated to explain these aspects, their relationships and the internal components and to present suggestions for future implementations.

- **Control Plane:** creates an overlay model of the network, through this plane the Resource Discovery (RD) agents can discover the available resources among the whole network and negotiate the available resources with Fair Intelligent Admission Control (FIAC) module by means of given service contract, e.g. Service Level Agreement (SLA).
- **Data Plane:** that is normal Internet which is used to transmit the actual data.

The control plane plays a fundamental role in the RD-FIAC architecture, mainly because it is involved with available resource discovery, and management of QoS in order to support service guarantees when combining multiple domains in an edge-to-edge scope. This plane implements the horizontal interaction of the RD-FIAC.

The control plane creates an overlay network over the data plane, which is basically represented by the current Internet infrastructure or DiffServ domain, made of routers, hosts, links, and other network components. The main outcome of this feature is that the control plane can be implemented in any domain without changes to the existing network, except for introduction of QoS technologies used for providing service guarantees.

We now briefly describe the interactions between the two planes.

3.1.3 RD-FIAC Planes Interaction

In the RD-FIAC architecture, the communication among planes is done through interactions with well-defined functions. The usual notion of network architectures is extended to a broader concept that, for instance, allows the direct communication between each pair of planes. The RD-FIAC interactions can be grouped into two types, horizontal and vertical interactions, as depicted in Figure 3.4 and Figure 3.5.

Horizontal interactions refer to interaction between RD and FIAC modules. The common understanding in network architectures is that horizontal interactions are comprised of protocols. The horizontal interaction has been defined (Figure 3.5):

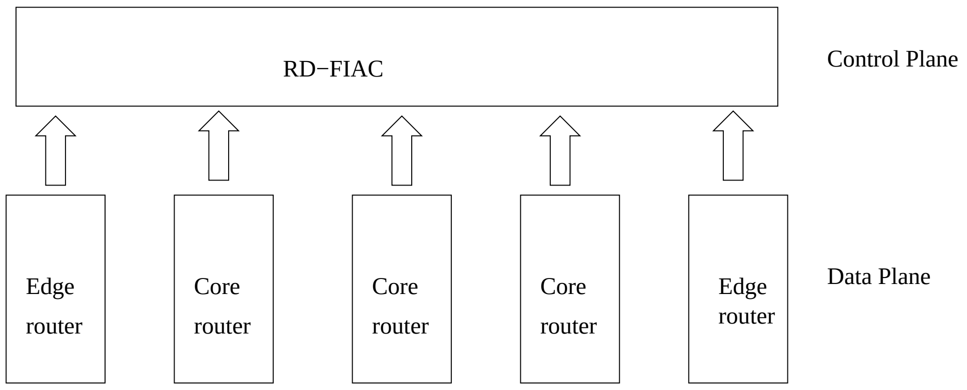


Figure 3.4: Vertical Interaction: Control Plane and Data Plane

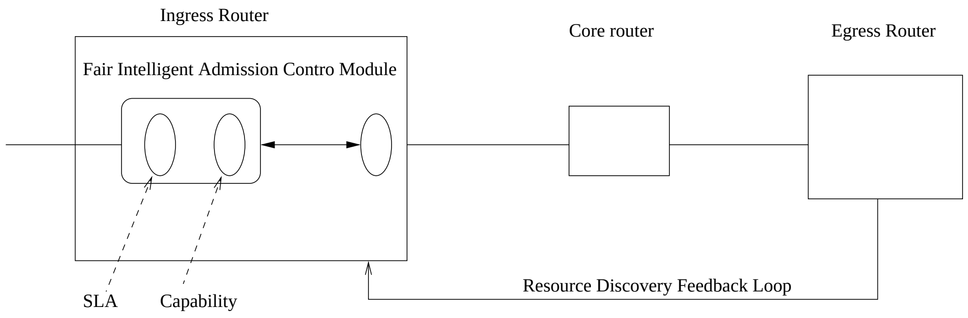

Ingress Router
Fair Intelligent Admission Contro Module
Core router
Egress Router
Resource Discovery Feedback Loop
SLA
Capability

Figure 3.5: Horizontal Interaction: RD and FIAC

- **horizontal interaction:** communications on control plane of FIAC and RD, and available resources report. Elements of interaction is presented in more details in Chapter 4.

Vertical interactions always involve two distinct planes of a given domain. In a given interaction, the functions can be activated only by control plane. Similarly, to the horizontal interaction, the definition is as follows.

- **Vertical interaction:** interaction whereby the control plane offers service or admission control to data plane. The functions belonging to the vertical interaction refer to the queries from the control plane to the data plane in order to get information for available resources and admission control functions from control plane to data plane.

In the control plane, admission control is related to the decisions involving the amount of resources to be allocated to each class in the data plane, independently of the QoS implementation strategies. It is responsible for controlling the provisioning in the data plane through the interface between data plane and control plane. After the confirmation of the RD-RD negotiation, each domain RD-FIAC must perform its internal management in order for end-to-end QoS to be achieved.

3.2 End-to-End QoS Framework

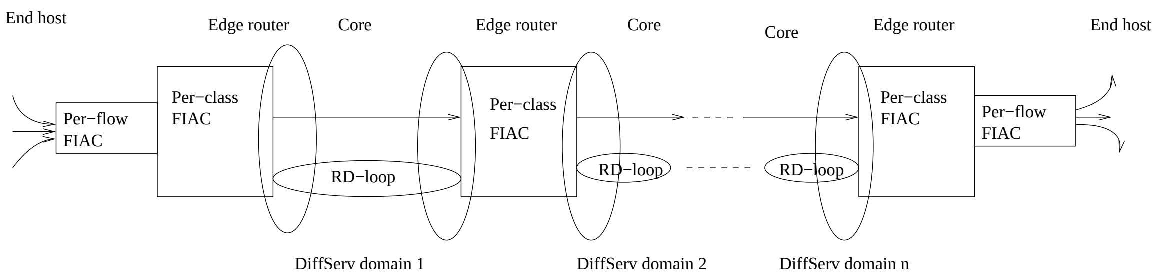


Figure 3.6: End-to-End QoS Framework

The RD-FIAC architecture is not only able to provides edge-to-edge QoS, but also could support End-to-End QoS through the concatenation of RD-FIAC of the intermediate domains the traffic can cross. From the ISP administrator point of view, the service is actually the user service, and he/she should not be aware of the existence of control plane. Figure 3.6 illustrates the user's view of an End-to-End QoS, through the combination of the intermediate domains' RD-FIAC by using negotiation mechanism in the control plane.

In this framework the RD-FIAC modules are connected through the routers and communicated on the control plane. User data is transported in the data plane. The QoS state of the domain can be passed to its previous neighboring domain until to the first domain. So, the RD-FIAC in the first domain obtains the End-to-End QoS state through the concatenation of such bilateral negotiation. After receiving the QoS state information from RD, the FIAC module compares the "external" traffic requirement (e.g., Service Level Agreement) and the "internal" network capability (QoS state from RD) and then applies Fair Intelligent Admission Control algorithms to make admission decision so that the End-to-End QoS could be approached.

3.3 Summary and Discussions

We have designed a Resource Discovery and Fair Intelligent Admission Control architecture that coordinates control plane and data plane. The scalability of the architecture is attributed to its overlay structure that is based on DiffServ infrastructure. We use the RD mechanisms to estimate available resources and pass the information to the Fair Intelligent Admission Control (FIAC) module. The FIAC module makes a decision to allow or reject the application based on the "internal" network QoS states, and "external", requirement of the application. The details of the Resource Discovery scheme and evaluation of its effectiveness is presented in the next chapter. Further analysis on how the throughput can be improved using queue control algorithm and bandwidth control algorithm is also reported.

In summary, the basic strengths of the RD-FIAC approach are:

- It improves the QoS of DiffServ without sacrificing scalability. The network state information that needs to be maintained by RD mechanism is class-based. Since every edge router maintains only the state for its own domain, this allows the entire RD-FIAC to scale better to a larger user base. If a particular router gets overloaded due to the growth of user-base, it is possible to be detected by RD mechanisms and be informed to the FIAC module to take admission control properly.
- It can be treated as overlay network based on DiffServ infrastructure. The control plane where RD-FIAC functioning on has ability to discover available resources, allocate available resources efficiently, and provide QoS among DiffServ domain. It also pointed out a way to deliver end-to-end QoS sensitive services. It decouples control plane from the underlying scalable Internet, data plane, thereby reducing the complexity of QoS management and control, especially in terms of QoS management and control. The data plane, underlying DiffServ infrastructure, is focused on normal DiffServ functions.

Although we present the RD-FIAC as general architecture, one specific example where RD-FIAC will be useful is for IT managers to manage a WAN that interconnects corporate offices, remote and mobile employees. Corporations have turned to Internet VPNs to deliver performance, security and manageability to their various sites scattered across the country. However, existing SLAs between ISPs and customers have focused on backbone performance guarantees, and do not reflect the end-to-end performance of individual applications. In addition, some fraction of the traffic may traverse multiple routing domains that belong to different ISPs. IT managers still face the challenging of provisioning the total capacity efficiently among the various types of traffic to meet application requirements such as latency and reliability characteristics. A RD-FIAC architecture can be deployed in this case to handle intra- and inter- domain resource allocation. For example, IT managers can treat each corporate site as a basic domain, and introduce a edge router to monitor the traffic flow, adapt resource allocation, and re-negotiate SLAs with the corresponding ISPs when necessary.

In the next two chapters, we will present several specific implementations within this framework to illustrate its components and properties in detail. More specifically, in Chapter 4, we study Resource Discovery mechanisms. In Chapter 5, we design a novel fair intelligent admission control algorithms to support per-class and per-flow service guarantees. In Chapter 6, we study edge-aware resource discovery and fair intelligent admission control. By studying these mechanisms, we illustrate the fundamental trade-offs in supporting QoS in the scalable Internet. Moreover, we demonstrate that both aggregate and per-flow service guarantees can be supported in the same framework.

Chapter 4

Core-Aware Resource Discovery Protocol Based on Feedback Scheme

4.1 Introduction

In this chapter, we illustrate how Resource Discovery (RD) protocol works. We attempt to address the fundamental trade-offs in the design of Resource Discovery protocol and its overhead. In particular, we study the interaction and relationships between the control plane (RD works on) and data plane (underlying Internet or DiffServ).

The key point of RD system is to obtain the accurate knowledge of current network QoS state and this can be described as a closed-loop problem, which enables us to use feedback control theory to analyze the properties of the RD algorithm. For instance, we will use feedback control analysis to derive the available bandwidth and available buffer. By Resource Discovery (RD) mechanism, the routers inside networks are equipped with the RD mechanism to detect congestion and to explicitly signal control module before congestion actually occurs. A router with RD mechanism measures the available bandwidth and available buffer as QoS state indicator, and judiciously feedback to the edge router with the objective of reducing packet loss ratio and improving link utilization. As one of the most important components in Differentiated Services architecture is the queue management mechanism used at core routers, we then explore the use of RD-DiffServ in the context of assured services architecture. We will show analytically that the widely referenced queue management mechanism, *RED with in and out (RIO)* [16], cannot achieve

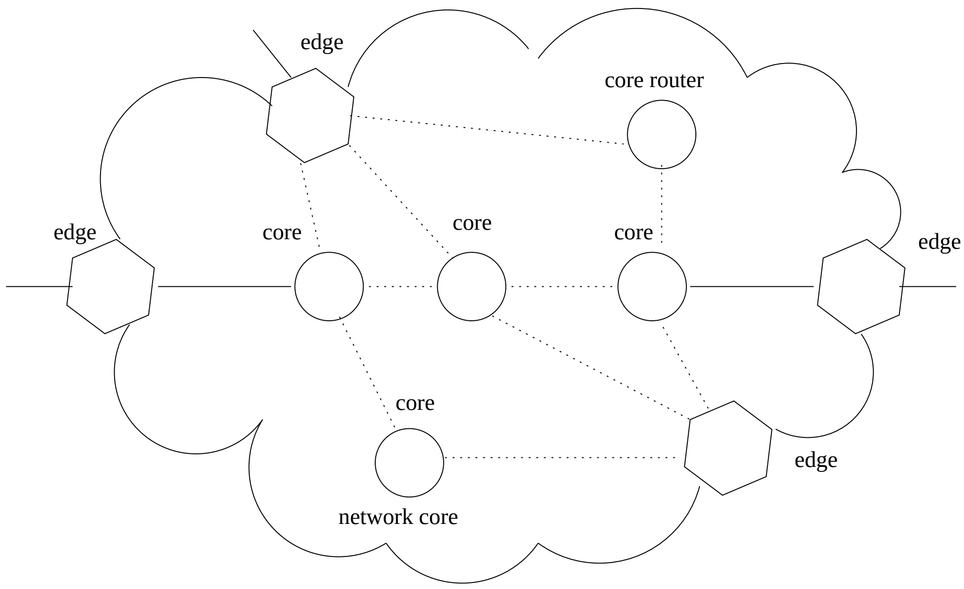


Figure 4.1: The network model

throughput assurance and proportional bandwidth sharing. We then propose the RD-DiffServ architecture to improve the assurance for assured service. Finally we validate the design of RD-DiffServ and RIO by comparing their performance.

The proposed RD protocol has three significant objectives.

- First, as a resource discovery mechanism, it must not mandate any specific scheduling algorithms or queue management to be employed in a network in order to provide quality of service. In other words, it must allow for diverse scheduling algorithms or queue managements as long as they are capable of providing QoS guarantees. In fact, we will show that our RD protocol can accommodate the traditional FIFO or Round Robin with drop-tail or RED, which is the most widely implemented and deployed mechanism in router today.

- Second, the RD protocol is transparent to the data plane where the underlying network being unaware RD protocol.

- Third, the QoS states information obtained from RD protocol is stored at ingress edge.

- Fourth, our first scheme, Core-Aware Resource Discovery mechanism, collects the QoS states of the en route routers, such as available bandwidth and available buffer.

Admittedly, it is not perfect for the connectionless Internet because the en route core routers have to be involved. In this case the solution is not optimum, however, our scheme still performs well as the Internet with its shortest path routing protocol will almost always choose the same path from a source to a destination. We believe our second scheme, Edge-Aware Resource Discovery, fits precisely into connectionless IP delivery. First, Edge-Aware Resource Discovery only needs the edge routers rather than the core routers. Second, flows can be routed throughout the network rather than going through the particular path. As a result, the second scheme can fit into the dynamically and more complicated network topologies. The RD flows going through a network is likely to be connectionless delivery so that the Edge-Aware Resource Discovery is a feasible and practical solution for the Internet.

We believe that these three objectives are important in implementing quality of service in practice. For example, the ability to employ diverse scheduling algorithms not only encourages choice and competition among equipment vendors and Internet service providers (ISPs), but also, perhaps more importantly, allows a network and its services to evolve. Similarly, by maintaining QoS states only at edge router, core routers are relieved of QoS control functions making them potentially more efficient. Furthermore, a QoS control plane which is decoupled from the data plane allows an ISP to deploy sophisticated provisioning and admission control algorithms to optimize network utilization.

We describe how the available network resources of a router (edge or core) can be characterized via the queue control function and available bandwidth function. Furthermore, we show that the notion of network state leads to the design of fair intelligent admission control algorithms. In addition, our framework does not exclude the use of existing scheduling algorithms such as stateful fair queuing algorithms to support guaranteed services.

The remainder of this chapter is organized as follows. In the next section we will briefly outline the basic architecture of the RD system. In Section 4.3 and 4.4 we describe in more details the feedback loops that are used to collect network state information in terms of available bandwidth and available buffer. In Section 4.5, we present simulation results to quantify the performance of the algorithm. We summarize this chapter in Section 4.6.

4.2 Core-Aware Resource Discovery Scheme: Basic Architecture

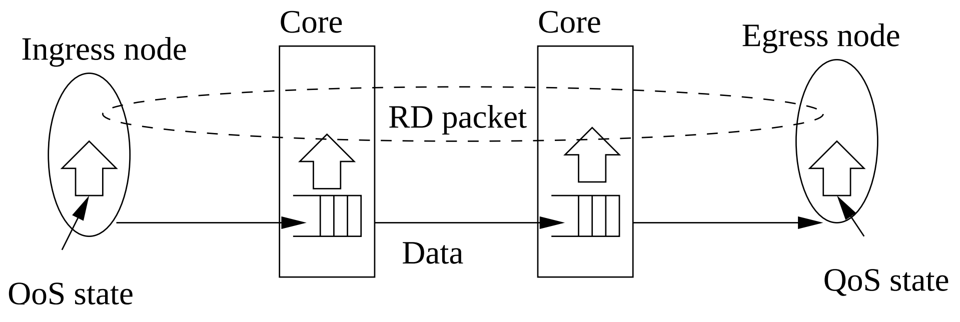


Figure 4.2: Resource Discovery: Basic Architecture

The modules described in Chapter 2 constitute the basic elements upon which an IP network with RD capabilities may by built. In this section we present the Resource Discovery (RD) model that is being defined at the control plane, as well as being transparent to the underlying network infrastructure.

A central of the current Internet architecture is that congestion control is performed mainly by TCP transport protocols at end hosts. However, as continuous media multicast applications (which usually do not deploy TCP for congestion control) become widely deployed on the Internet, it becomes difficult to perform end-to-end congestion control. It has been agreed upon that the router itself must now participate in QoS states and resource management. By Resource Discovery, we mean the routers inside the network are equipped with the capability to detect QoS states and to explicitly feedback the edge router before congestion actually occurs. If a router signals QoS states prematurely, packet loss can be avoided and at the expense of low link utilization. As high link utilization usually leads to high throughput, maintaining high link utilization while reducing packet loss is an important objective for Resource Discovery module.

4.2.1 Features of the Resource Discovery Model

The basic RD protocol model for the resource discovery of the QoS provisioning of a network domain is schematically depicted in Figure 4.2. In this architectural model, the RD protocol centrally manages and maintains a number of management information re-

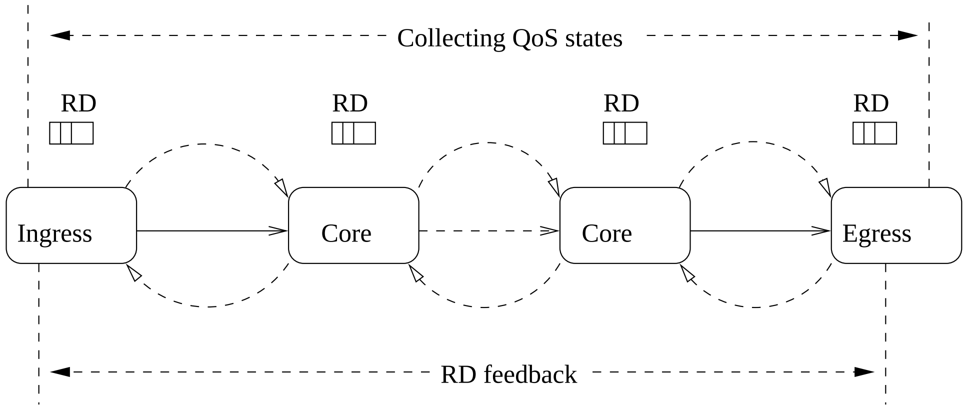


Figure 4.3: QoS states discovery: RD packets life cycle

garding the network domain. Among them, the network QoS state are most relevant to the study of this chapter. The available bandwidth and available buffer together provide a logical representation (i.e., a QoS state abstraction) of the network QoS state. With this QoS abstraction of the network domain, the RD protocol performs QoS control functions by managing and updating these databases. In this sense, the QoS control plane of the network domain is decoupled from its data plane. The core routers of the network domain are removed from the QoS control plane: core routers do not maintain any QoS state, whether per-flow or aggregate, and do not perform any QoS control functions such as admission control.

In our RD model, the network QoS states are combined with two states: available bandwidth states and available buffer state. The available bandwidth state maintains information regarding the available bandwidth of the link. The available buffer maintains the QoS state information regarding the queue state of the router.

For completing the QoS states discovery, the RD protocol requires three stages. Figure 4.3 shows the RD packets life cycle.

- Stage 1: It is a RD packet generation stage, whereby the ingress edge generates RD packet and send it to the egress routers along the path, updating the QoS states. During the generation stage, the ingress edge is responsible for negotiating with the admission control module.

  Generally speaking, RD packets should receive special service better than data packets as a delayed or lost RD packets could impact on admission control. The

Expedited Forwarding (EF) PHB [59] defines a particular level of assured bandwidth, low loss, low latency, or low jitter end-to-end service by providing higher priority. Other packets on the network are preempted to make room for EF packets when congestion arises. The EF service appears suitable for the RD packets. We find that the throughput of TCP flows is affected and the loss in TCP throughput is higher in the presence of EF traffic at short time-scale. However, the aggregate throughput of TCP flows remains roughly unaffected by the EF traffic. Further, the classification of RD packets as EF service improves the fairness of bandwidth allocation among TCP flows.

- Stage 2: It is a collecting QoS states stage, whereby the routers en route to the egress router (core routers) check their QoS states and update the fields of RD packet.

- Stage 3: It is a feedback state, whereby the egress edge feedback the RD packets with updating QoS states to the ingress edge.

RD module implements edge-to-edge resource discovery in terms of the available bandwidth and the available buffer states. RD protocol generates the resource discovery packet, RD packet, from the ingress to detect the network QoS states along the path to the egress router.

It is commonly believed that different network vendors may prefer to deploy their proprietary control mechanism according to their policy requirements. Our proposed Resource Discovery framework, therefore, should be generic and flexible enough for this purpose. Moreover, it is desirable that it is backward compatible with existing mechanisms for enabling interoperability.

In considering these requirements, we define a general RD in a way that a variety of control mechanisms can be derived from it. The concept of RD framework is that the ingress routers periodically probe the core of the network to obtain the current QoS states information. This network QoS states information is used by the Fair Intelligent Admission Control (FIAC) module which resides at ingress or egress routers such that a more precise control on the incoming traffic can be achieved.

The RD module can be built based on the Internet with or without DiffServ. We define the additional functions required to construct a closed-loop feedback resource discovery mechanism working between the ingress router and the egress router.

On the forward path, ingress routers perform two functions.

1. Initiate the RD packet, mark the DSCP (e.g. EF class) or flow identification in the DSCP-field of RD packet.

2. Generate RD packet at a constant rate (or proportional to the incoming traffic rate);

In turn, the core routers forward RD packets to collect the network QoS states information.

On the backward path, the egress routers terminate the RD packet and send them back to the ingress router with updated network QoS states information along the path. Upon receiving backward RD packets, the ingress routers report the network QoS states to the Fair Intelligent Admission Control (FIAC) module.

As a result, the RD mechanism can provide “inside” network QoS states or capability information dynamically.

It is possible that the resource discovery mechanism spans across multiple domains or within only one domain. In this Section, we consider only the intra-domain control mechanism while a brief discussion on inter-domain resource discovery is given in Section 4.5.

4.2.2 Background: Queue Management and Differentiated Services

Congestion occurs when the aggregate traffic volume at an input link is higher than the capacity of the corresponding output link. One obvious approach to congestion control is to dedicate a separate queue for each flow at a router and employ certain rate-based fair queuing algorithm to regulate packet flows. However, this approach does not scale well, because queue management on a per-flow basis usually incurs high control overhead.

Among all the queue management mechanisms that do not require per-flow buffer management, drop tail queue is the most widely deployed and simplest one. Since a drop tail queue drops packets and conveys congestion signals only at the time of congestion, there is a significantly long time period between the instant when congestion occurs and the instant when end hosts reduce their sending rates. During this time period, packets may be sent and eventually dropped. Also, because TCP connections exhibit bursty behavior, a drop tail queue is prone to drop multiple packets of one connection. This may undesirably shut down the TCP congestion control window of the connection. Moreover, drop tail may result in global synchronization (that results from signaling all TCP connections to reduce their congestion windows at the same time), which is usually followed by a sustained period of low link utilization.

To remedy the aforementioned drawbacks, several feedback or active queue management algorithms have been introduced, e.g., RED and its variation FRED. By "feedback", we mean that a router that employs such a mechanism is responsible for detecting congestion and notifying end hosts of congestion so that the latter can adapt their sending rates before congestion actually occur. These algorithms differ in the following aspects:

- the parameter used as an indicator of traffic load (and congestion);
- the policy used to detect congestion (or the likelihood of congestion); and
- the policy used to adjust the packet dropping probability in response to (an increased likelihood of) congestion.

Random Early Detection (RED)

RED starts to drop packets long before the buffer is full, providing early congestion indications before the buffer overflows. RED operates by calculating the average queue length, *avg-queue*, upon packet arrival using a low-pass filter with an exponentially weighted moving average. The parameter *avg-queue* is used to measure traffic load. The policy used to detect likelihood of congestion is characterized by two thresholds, *minth* and *maxth*. If *avg-queue* is less than *minth*, the router is considered congestion free and no arriving packets will be dropped. When *avg-queue* is greater than *maxth*, the router is likely to incur congestion, and all arriving packets will be dropped. When *avg-queue* is between the two thresholds, the probability of dropping a packet linearly increases with *avg-queue* from 0 to p_{max} is maximum packet dropping probability. As one of the first active queue management algorithms proposed, RED was shown to prevent global synchronization, accommodates bursty traffic, incurs little overheads, and coordinates well with TCP under serious congestion conditions.

The performance of RED, however, heavily depends on whether or not the two thresholds are properly selected. If the thresholds are set too small, buffer overflow is avoided but at the expense of low link utilization. On the other hand, if the thresholds are set too large, congestion occurs before end-hosts are notified to reduce their sending rates, and a large amount of bandwidth is wasted on transporting packets that will be eventually dropped. On heavily congested links, it may be difficult to keep both high link utilization and low packet loss ratio simultaneously.

Another problem associated with RED is that with the reason of accommodating bursty traffic, RED uses the average queue length as an index of traffic load, and uses the buffer space between the $minth$ and $maxth$ thresholds to accommodate the delay between early congestion characteristics of network traffic, the instantaneous queue length varies rapidly and may be very different from the value of *avg-queue*. As a result, buffer overflow may occur when *avg-queue* is still less than *maxth*, especially when the buffer size is not large enough.

Overview of Differentiated Service

The Differentiated Services architecture relies on packet tagging and lightweight router support to provide premium/assured services that extend beyond best effort. In particular, the class of assured services is intended to give the customer the assurance of a minimum throughput (called the target rate), even during periods of congestion, while allowing it to consume, in some fair manner, the remaining bandwidth when the network load is low. More specifically, the two major requirements of assured services are:

- Throughput assurance: Each flow should receive its subscribed target rate on average;
- Fairness: The surplus bandwidth is proportionally shared among assured service connections by “proportional bandwidth sharing”.

The basic idea behind the differentiated service architecture is to tag packets based on the Service Level Agreement (SLA) at the edge routers, indicating the preferential treatment that the packets expect. Core routers inside a network make packet forwarding or dropping decisions according to the tags. In the assured service architecture, there are thus two major components:

- the packet marking mechanism at edge routers to classify packets as in-profile packets (IN packets) or out-of-profile packets (OUT packets) prior to their entering the network.
- the queue management mechanism used at core routers to differentiate and forward packets based on their service class and the marking made by the edge router.

Succinctly, RIO [16] can be viewed as example of assured service architecture. The basic idea in the RIO is quite simple. In the RIO, a class of traffic is provided with a certain bandwidth profile, defined by a rate R. As long as the aggregate amount of traffic from that class has a rate lower than R, traffic is marked as "IN-profile"; otherwise, it is marked as "OUT-of-profile". In times of congestion, out-of-profile traffic is dropped more aggressively than in-profile traffic. In other words, a class is allowed to exceed its profile R when there is no congestion and the network load is low, but is restricted to sending traffic within its profile when the network is congested. The rate R is statically reserved, or provisioned, at network design time.

As reported in [29], the throughput assurance and fairness requirements of assured services cannot be met under many cases. In particular, RIO tends to favor connections with small target rates and small round trip times (RTTs). This is because TCP reacts to packet loss by halving its congestion window of a connection with a large target rate (or RTT) is usually larger than that of a connection with a small target rate (RTT), it takes more time for a large RTT connection to regain its original rate after packet loss. Moreover, since a TCP connection always attempts to utilize surplus bandwidth, the connection with small RTT will continue to increase its congestion window after its congestion window restores, and hence in the case that multiple TCP connections experience packet loss, the small RTT connection may capture the bandwidth that is originally subscribed by the other large RTT connections. The absence of throughput guarantee is clearly exhibited in the case of TCP traffic, as discussed in [70]: regardless of how well provisioned the network is, it may be impossible to provide throughput guarantees to TCP flows with the AF service. In fact, the only assurance that in-profile traffic gets is that, should congestion

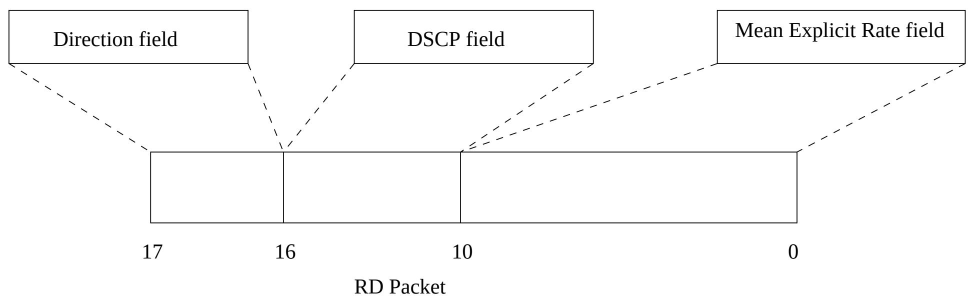


Figure 4.4: RD packet format

occur, it will not be dropped as aggressively as out-of-profile traffic.

4.2.3 RD Packet Format and Generation

RD Packet Format

In order to collect the QoS states of network along the path, our RD mechanisms require packets to carry state in their headers. The main challenge to implementing these algorithms is to use minimum space for storing QoS states variables and at the same time remain fully compatible with current standards and protocols. In particular, we want the network domain to be transparent to end-to-end protocols, i.e., the egress router should feedback the RD packet to the ingress router. As described in Chapter 3, there are three pieces of state that need to be encoded in a RD packet:

1. the direction of RD packet which specifies the RD packet in forward or backward path,
2. the DSCP field contains the aggregate classification assigned by the ingress edge router,
3. the Mean Explicit Rate (MER) field provides Mean Explicit Rate per class-unit.

Figure 4.4 shows the RD packet format. The bits of the RD packet are allocated to the following fields:

- a 1-bit flag, called *DIRECTION*, that specifies the RD packet in forward or backward path

- a 6-bit field *DSCP* that contains the aggregate classification assigned by the ingress router
- a 12-bit field *MER* that encodes QoS states.

RD Packet Generation

The RD packet is generated by an ingress router upon receiving the incoming traffic. The ingress forwards the RD packet to the corresponding egress router in order to collect the QoS states along the path. Upon receiving this packet, the en route router checks its QoS states, and updates the states carried by the RD packet accordingly. When an egress router receives a RD packet, it checks its QoS states, updates the states of the RD packet, and sends it back to the ingress. Upon receiving this packet, the ingress router reports the QoS states carried by the RD packet to the admission control module, FIAC.

Basically, the RD packet generation is determined by two parameters: generating period and granularity.

Generating period refers to how frequently a RD packet is generated or the temporal resolution of the RD protocol. It can be specified in term of a time interval or packet count. The choice of a generating period is related to the dynamics of the DiffServ domain under control as well as the variation of the incoming traffic. To obtain a higher control precision, the ingress router may choose a shorter detecting period, i.e. generate RD packets more frequently. However, this generating frequency should be balanced with the amount of bandwidth cost which can be tolerated by the network routers. It's natural to generate RD packets in a constant rate or proportional to the incoming traffic rate. For example, the domain administrator can control the overhead within the tolerable range (e.g. 5%). Please refer to Section 6.4.4 in details.

Generating granularity, however, refers to the resolution of the RD mechanism in spatial domain. The following lists some possible examples:

- Per-flow or per-microflow basis, in which one RD packet is generated per contracted incoming flow. It implies a flow based control mechanism, which can generally give the finest grain control precision.

- Per-BA basis, in which one probe is generated per behavioral aggregate (PHB). If an ingress node has access to more than one PHBs, multiple probe packets will be generated in each probing interval.

- Per-egress-node or per-boundary-node basis, in which one RD packet is generated per boundary node. Notice that the notion of ingress-egress- pair is defined only when there is a flow. Therefore, this generating scheme can be regarded as a topology based mechanism in which each boundary (ingress) node keeps the statistics of all possible paths having other boundary nodes as egress points.

A combination of above, depending on their control algorithm and, particularly, their required control precision, network providers may choose to have a variant or a combination of the above mentioned schemes.

In general, in choosing a generating granularity, one may consider (1) the required control precision, (2) the processing capability of the routers, and (3) the amount of tolerable control overhead.

Since RD packets are sent from the ingress edge to the egress edge router, an interior/egress node needs a mechanism to distinguish the RD packets from other data packets. RD mechanism creates a new packet with a special DSCP or flow id, in which QoS state information is carried in the data area of the packet.

After identifying a RD packet, a node can handle it using same level of forwarding treatment as other data packets, thereby it is subjected to being delayed or even dropped when the node is congested. This approach simplifies the design of the interior node. By examining the arrival of the RD packets, one can also obtain a sample of the current congestion level of the forwarding path.

Another approach is to offer the RD packets a special service, usually better than data packets, such as EF [59] class, at an interior node. It requires a special arrangement within the forwarding module of an interior node.

RD Packet Overhead

We express the overhead calculation O as in (4.1).

$$O = \frac{R_{rd}}{R_{rd} + \sum_{i=1}^{n} R_i} \tag{4.1}$$

Where R_{rd} is the RD traffic rate, R_i is for other traffic rate, and n is the number of classes except RD traffic.

As we can see, Resource Discovery mechanism introduces overheads at the routers. However, the RD traffic configuration can be adjusted according to the network states to minimize the overheads. For example, the RD packets could be assigned as low priority class (e.g. Bronze service) and low sending rate when the traffic behaves well; the RD packets could be assigned as high priority (e.g. Gold or Premium Service) and high sending rate when the network states vary dramatically. Resource Discovery mechanism is designed in a manner that allows RD traffic configuration accordingly.

4.2.4 RD Ingress node

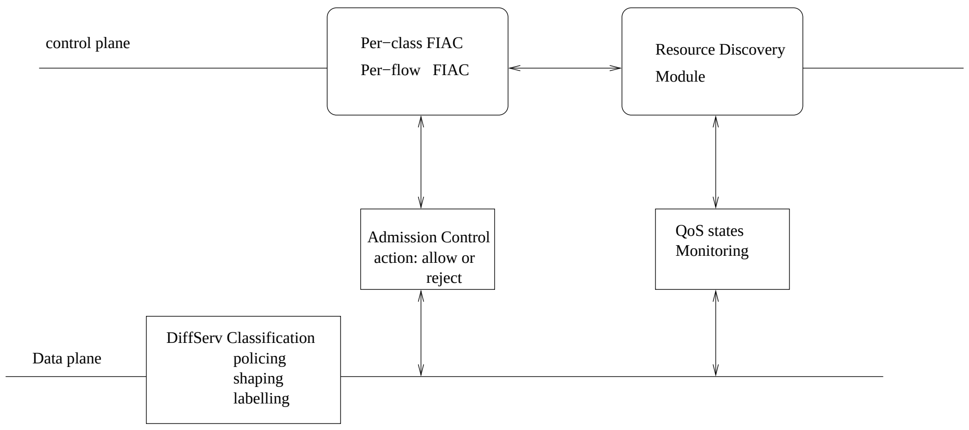

control plane
Per−class FIAC
Per−flow FIAC
Resource Discovery Module
Admission Control action: allow or reject
QoS states Monitoring
Data plane
DiffServ Classification policing shaping labelling

Figure 4.5: Building blocks of an ingress router

In the Resource Discovery mechanism, the ingress router is responsible for generating the Resource Discovery (RD) packets destined for the egress router and assigns RD packets a special class (e.g. EF class). The RD packets are used to collect network QoS states information which containing a vector of QoS parameters for all classes along the path. Upon receiving a RD packet, the core router consults its QoS states in terms of class and modifies the fields of the RD packets accordingly and then forwards to other routers

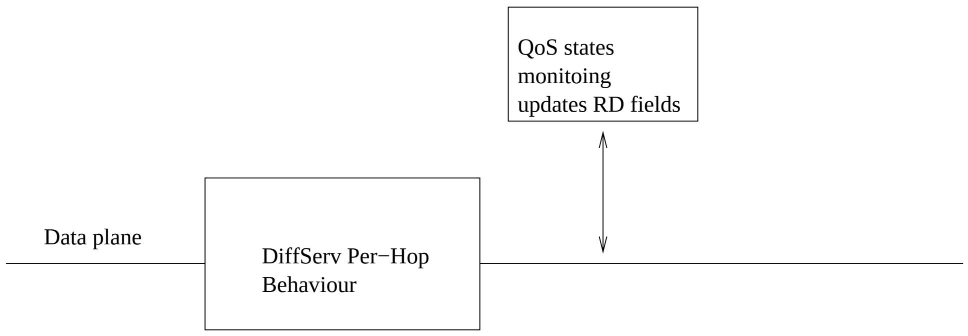


Figure 4.6: Building blocks of an interior router

along the path to the egress router. The egress router is responsible for sending back the RD packet to the ingress edge router.

The structure of an ingress router can be decomposed into two blocks, as illustrated in Figure 4.5. Most of the complexity of the RD-Internet or RD-DiffServ is concentrated in this kind of equipment. Ingress routers must perform normal DiffServ functions, such as classification, policing, and labeling. Besides the normal Internet/DiffServ router functions, ingress router performs some additional RD functions, generating an RD traffic for per TCP or UDP connection, negotiation with RD module and FIAC module, QoS states monitoring, and admission control action.

4.2.5 RD Interior node

In addition to the basic packet forwarding function, an interior node is extended to include a QoS states monitoring function. Upon receiving a RD packet, it updates the information carried in the RD packet with its current state information and then forwards the RD packet to other connected node(s). Figure 4.6 illustrates the structure of an interior node.

4.2.6 RD Egress node

For the case of intra-domain resource discovery, a RD egress router is where the RD packets are terminated. The egress router is responsible to return the RD packet to the ingress node.

For the case of inter-domain resource discovery, the RD egress router negotiates with the next domain's ingress router to decide inter-domain's QoS states that will be feedback

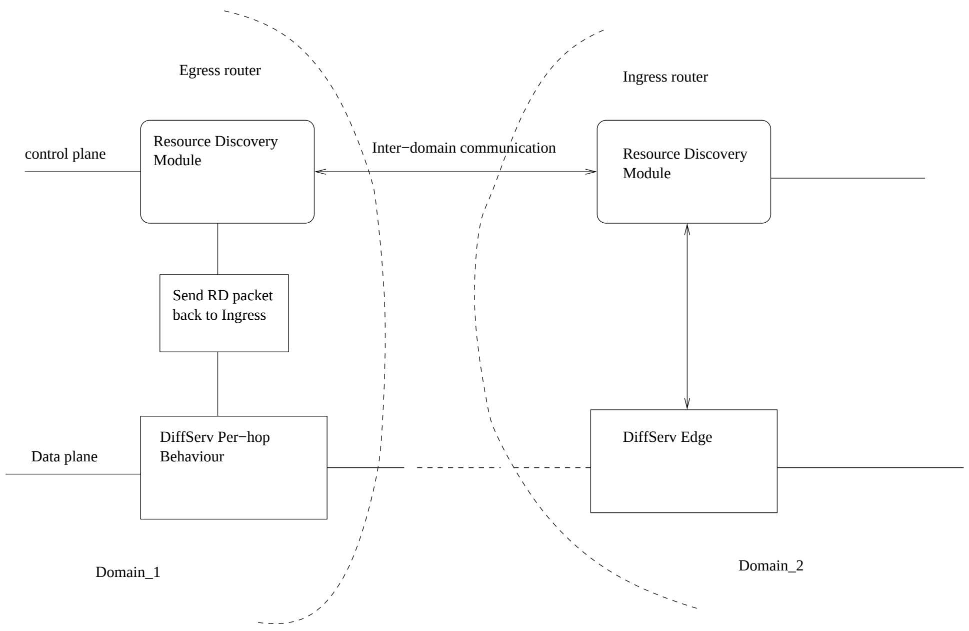

Egress router
Ingress router
control plane
Resource Discovery Module
Inter−domain communication
Resource Discovery Module
Send RD packet back to Ingress
DiffServ Per−hop Behaviour
Data plane
DiffServ Edge
Domain_1
Domain_2

Figure 4.7: Building blocks of an egress router

to the ingress router. Figure 4.7 illustrates the components of an egress node.

4.3 QoS states functions

We have described a general framework to model RD-DiffServ architecture from resources in a network, and the edge router react to the dynamic resource discovery. In the following sections, we will explain the functions in the Resource Discovery to calculate QoS states information and conduct numerical studies to illustrate the analytical results we obtained from this section.

4.3.1 Computation of Flow Arrival Rate

Recall that in our architecture, the rates $r_i(t)$ are estimated at the edge routers. At each edge router, we use exponential averaging formula as in Core-Stateless Fair Queuing (CSFQ) [66] to estimate rate of a flow. Using an exponential weight gives more reliable estimation for the burst traffic, even when the packet inter-arrival time of the aggregate has significant variance.

Let t_i^k and l_i^k be the arrival time and length of the k^{th} packet of flow i. The estimated rate of flow i, r_i, is updated every time a new packet is received:

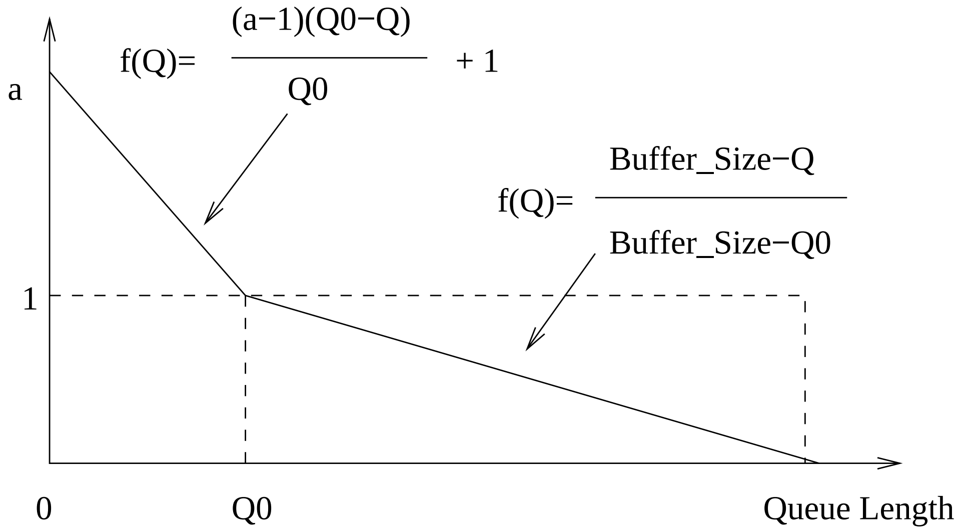


Figure 4.8: Queue Control Function f(Q)

$$r_i^{new} = (1 - e^{-T_i^k/K})\frac{l_i^k}{T_i^k} + e^{-T_i^k/K} r_i^{old} \tag{4.2}$$

where T_i^k represents the k^{th} sample of the inter-arrival time of traffic i, i.e., $T_i^k = t_i^k - t_i^{k-1}$ and K is a constant.

The choice of K in the above expression with several tradeoffs. First, while a smaller K increases the system responsiveness to rapid rate fluctuations, a larger K better filters the noise and avoids potential system instability. Second, K should be large enough such that the estimated rate, calculated at the edge of the network, remains reasonably accurate after a packet traverses multiple links. This is because the delay-jitter changes the packetsinter-arrival pattern, which may result in an increased discrepancy between the estimated rate (received in the packetslabels) and the real rate. To counteract this effect, as a rule of thumb, K should be one order of magnitude larger that the delay-jitter experienced by a flow over a time interval of the same size, K. Third, K should be no larger than the average duration of a flow. Based on this constraints, an appropriate value for K would be between 100 and 500 ms.

4.3.2 Fair Available Bandwidth Control Function

The Available Bandwidth Fair Control Function is designed to calculate the fairshare of available bandwidth. It needs two components: one is to estimate the incoming traffic rate (refer to Equation 6.1) and another one is to calculate Mean Available Rate (MAR).

It should be noted that the RD mechanism uses the linear function $f(Q)$ to calculate the available bandwidth (see the Figure 4.8). The idea behind the queue control function is that it differentiates the congestion status into two phases at the threshold, Q_0. Suppose we start with a queue length at the router less than Q_0, which means that the router is not congested, and as long as the router does not detect congestion, the evolution of the available bandwidth corresponds to Phase 1. If congestion is detected, which means the queue length exceeds Q_0, Phase 2 is entered, the available bandwidth is decreased until the queue length drops below Q_0 and Phase 2 terminates. The next period starts again from Phase 1.

The queue control function, $f(Q)$, is a linear function with values between 1 and 0 for queue length in the range of $[Q_0, Buffer_{size}]$ and over 1 for queue length in the range of $[0, Q_0]$. The two lines intersect at Q_0, where the value of $f(Q)$ is 1. Various exponential forms for $f(Q)$ have been explored, but it is simplest to use this linear function with negligible performance penalty. The function can be defined as:

$$f(Q) = \frac{Buffer_{Size} - Q}{Buffer_{Size} - Q_0}, \text{ if } Q_0 \leq Q \tag{4.3}$$

and

$$f(Q) = \frac{(\alpha - 1) * (Q_0 - Q)}{Q_0} + 1, \text{ if } Q \leq Q_0 \tag{4.4}$$

To calculate Mean Available Rate (MAR), we designed the function as follows.

$$MAR_{new} = \begin{cases} \text{if } (Q \leq Q_0) \text{ or} \\ (Q > Q_0 \text{ and } R_i \leq MAR_{old}) \\ MAR_{old} + \beta * (R_i - MAR_{old}) \\ \text{else} \\ MAR_{old} \end{cases} \tag{4.5}$$

where β is the exponential average factor and R_i is the traffic arriving rate. In available bandwidth control function,the value of MAR is a running average of the current load when the network operates below the target operating point. When the queue length exceeds the target operating point, the MAR function does not allow MAR to increase further.

Instead of using queue occupancy as a congestion index, RD uses the average packet enqueue rate as the index. Since

$$\frac{d(ql)}{dt} = enqueue_{rate} - dequeue_{rate} \tag{4.6}$$

where ql is the queue length, and the packet dequeue rate is the link capacity. RD controls the *rate of queue occupancy change* and intends to keep the queue stabilized at an target point at which the aggregate packet enqueue rate is approximately equal to or slightly below the packet dequeue rate (link capacity).

4.3.3 Queue Control Function

The queue control function expresses the degree of network congestion it can tolerate. Since the queue builds up and drains out continuously, the queue control function can regulate the queue fluctuation smoothly. The target point, Q_0, represents the optimal buffer utilization point we expected. The linear function is defined as in the following equation.

$$f(Q) = \begin{cases} \frac{Buffer\_size - Q}{Buffer\_size - Q_0} & \text{, if } Q_0 \leq Q \\ 1 + \frac{(\alpha-1)*(Q_0-Q)}{Q_0} & \text{, if } 0 \leq Q \leq Q_0 \\ 1 \text{ , if } Q = 0 \end{cases} \tag{4.7}$$

The parameter, α, was chosen within the range of 1.01 to 1.9 can be considered as an oversell the network when it is in under-utilized. Q_0 is configured as in-profile threshold. The Q_0 is the target point at which the queue control function expects to maintain. When the queue length is beyond the target ($Q \geq Q_0$), that means the network becomes highly loaded, risking long delay and buffer overflow. The queue control function will discourage the incoming traffic accordingly ($f(Q) \leq 1$). When the queue length is below the target, it is an indication of network being underutilized. The queue control function is designed to encourage to take more incoming traffic ($f(Q) \geq 1$). When the queue length is close to 0, that means the congestion does not occur at the egress node, the congestion might

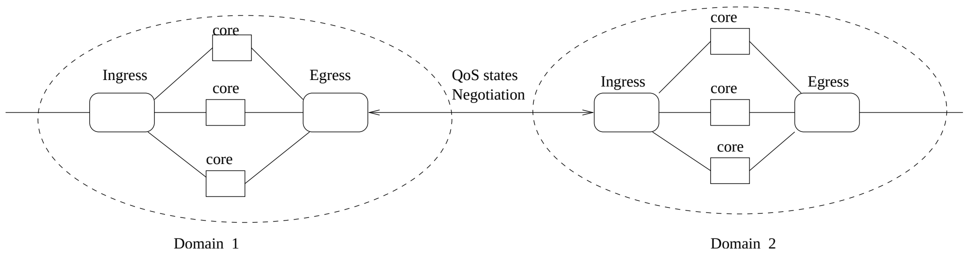


Figure 4.9: QoS states negotiation for inter-domain Resource Discovery

occur at core node. In this case, congestion happens at core, the Edge-Aware Resource Discovery Protocol will estimate the congestion degree based on the measurement of available bandwidth without considering the queue state ($f(Q) = 1$).

4.3.4 Network QoS state

The Resource Discovery Protocol determines the network QoS states, Explicit Rate (ER), by calculating the product of Queue Control Function $f(Q)$ and Mean Available Rate, MAR.

$$ER = f(Q) * MAR \tag{4.8}$$

When the network operates above the target operating point, Explicit Rate (ER) is reduced due to $f(Q) \leq 1$ (refers to equation (5.1)) and throttling is performed fairly. However, when the network operates below the target operating point, Explicit Rate (ER) is allowed to increase its rate by a factor greater than 1, which enables the incoming traffic capable of using the available bandwidth to take advantage of it.

4.3.5 Inter-domain RD-DiffServ

To obtain inter-domain QoS states, RD module was designed to be compatible among multiple domains. It needs domain-to-domain negotiation module to pass the individual domain's QoS states. Figure 4.9 illustrates the inter-domain QoS framework. In Figure 4.9, the egress node of domain 1, Egress1, negotiates with the ingress node of domain 2 via the negotiation module. Therefore, Domain 2 QoS states can be passed to Domain 1 via Egress2 $\rightarrow$ Ingress2 $\rightarrow$ Egress1 $\rightarrow$ Ingress1.

Generally, in order to obtain a consistent domain QoS states, all boundary nodes (ingress and egress) must be upgraded to RD feedback capable nodes. The operational procedures of inter-domain QoS states passing are as follow:

1. At the domain boundary, the ingress node generates and delivers a RD packet along with the data packets per aggregated flow. This RD packet carries the QoS states information along the path, but it is marked with special DSCP.

2. At any node inside a RD-DiffServ or RD-Internet domain, upon receiving a RD packet, the core router first calculates its QoS states using the available bandwidth control function and available queue control function. If its QoS states, explicit rate, is smaller than the one carried at the fields of the received RD packet, the explicit rate field, MER, of the RD packet will be replaced. The updated RD packet is then forwarded to the next node.

3. When a RD packet is received by an egress router, the RD packet is terminated and returned to the ingress router. The RD packet contains the domain's QoS states accordingly.

4. When the RD packet reaches the ingress router, the ingress router will negotiate with the former domain egress router to pass its QoS states to the former domain egress router.

5. Finally, when the preceding domain egress router receives the QoS states from its following domain ingress router, it will replace the explicit rate field of its RD packet if the QoS states is smaller than the one carried at the fields of the RD packet.

4.4 Discussion 1: Resource Discovery over Internet

4.4.1 Resource Discovery Algorithm for Internet

The Resource Discovery Protocol determines the domain capability or QoS states, Explicit Rate (ER), by calculating the product of Queue Control Function $f(Q)$ and Mean Available Rate, MAR.

$$ER = f(Q) * MAR \tag{4.9}$$

When the network operates above the target operating point, Explicit Rate (ER) is reduced due to $f(Q) \leq 1$ (refers to equation (5.1)) and throttling is performed fairly. However, when the network operates below the target operating point, Explicit Rate (ER) is allowed to increase its rate by a factor greater than 1, which enables the incoming traffic capable of using the available bandwidth to take advantage of it.

Pseudo code reflecting this algorithm is described in the Algorithm (1).

where Q_0 specifies the queue length to which the RD queue asymptotically converges and hence $Buffer\_size - Q_0$ is the maximum burst an RD router can accommodate. We use Q_0 to separate two stages: no congestion and congestion alert. When the queue length is larger than Q_0, we treat it as congestion alert stage. When queue length is less than Q_0, we treat it as no congestion stage. When an RD router is under the congestion alert stage, it attempts to keep the packet enqueue rate around an operating point by decreasing $\beta * (R - MACR)$. When an RD router is under the no congestion stage, it attempts to keep the packet enqueue rate around an operating point by increasing $\beta * (R - MACR)$.

4.4.2 Simulation setting

In this section, we evaluate Resource Discovery module over Internet (RD-Internet) by comparing it against drop-tail, and RED under a wide range of traffic conditions. Drop-tail serves as a baseline case in which no active queue management mechanism is employed. All the simulations were performed in ns-2 which provides accurate packet-level implementation for various network protocols, such as TCP/UDP, and various buffer management and scheduling algorithms, such as drop-tail and RED. All algorithms used in the simulation were part of the standard ns-2 distribution.

4.4.3 Simulation 1: bandwidth and buffer utilization

In the first experiment, we consider a simple topology (shown in Figure 4.10) with two TCP connections connecting to the same destination. Each TCP connection has a window size of 200 packets that is roughly equal to the delay bandwidth product. The buffer size

Algorithm 1 Resource Discovery Algorithm

```
Ensure: $ER$: The Explicit Rate
Require: The incoming traffic rate $R$
    $\beta$ : {The average ratio}
    $\alpha$ : {Congestion function parameter}
Require: Per-class Variable
    $MACR$ : {Mean Allowed Class Rate}
    $Q_0$ : {Target Class Queue Length}
    $ER$ : {Explicit Rate}
    $DPF$ : {Queue Control Factor}
    $Q_0 \Leftarrow$ 0.3 * BufferSize
    $MACR \Leftarrow$ 100Mb {Assign a large value}
    $\beta \Leftarrow 0.08$
    $\alpha \Leftarrow 1.23$
    Bandwidth Fair Control Function {on forward path at Ingress/core/Egress}
    if QueueLength $\geq Q_0$ then
      if $R \leq MACR$ then
        $MACR_{new} = MACR + \beta * (R - MACR)$
      else
        $MACR_{new} = MACR$
      end if
    else
      $MACR_{new} = MACR + \beta * (R - MACR)$
    end if
    Queue Control Function {on the backward path}
    if QueueLength $\geq Q_0$ then
      $DPF = (Buffer\_Size - QueueLength)/(Buffer\_Size - Q_0)$
    else
      $DPF = 1 + (\alpha - 1) * (Q_0 - QueueLength)/Q_0$
    end if
    The Explicit Rate
    if QueueLength then
      $ER = MACR_{new} * DPF$
    else
      $ER = MACR_{new}$
    end if
```

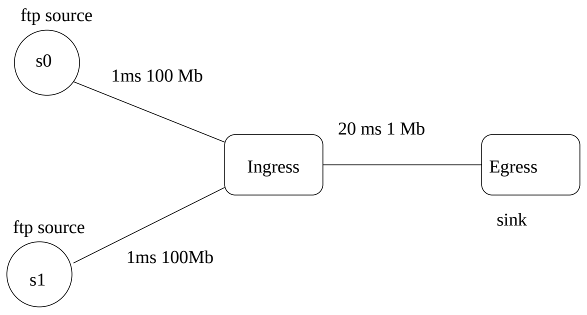


Figure 4.10: The RD over Internet simulation topology

is fixed at 100 packets in both RED and drop-tail. The threshold $minth$ ranges from 3 to 40 packets and the threshold $maxth$ is set to 3 times of the $minth$ value in RED. The parameter Q_0 is set to 30 packets.

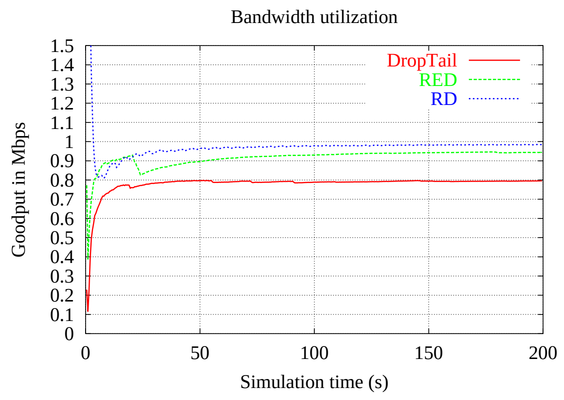
As shown in Figure 4.11, RD-Internet outperforms RED and drop-tail in terms of bandwidth utilization and buffer utilization, while drop-tail performs worst among all the mechanisms. This suggests that from the perspective of network management, RD-Internet achieves high throughput with efficient buffer utilization. RD-Internet managed the buffer utilization around the operating point which had been set to 40%.

4.4.4 Simulation 2: performance under TCP flows with different RTTs

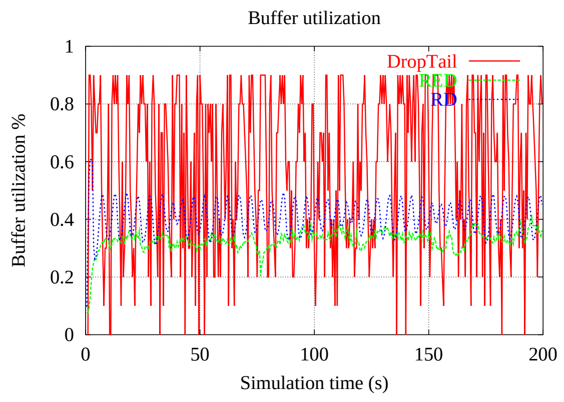
In the second experiment, we consider a single bottleneck topology (shown in Figure 4.12) with 16 TCP connections to their respective destinations via a common bottleneck (with link capacity set to 1 Mb). The buffer size is set to 100 packets at each router. All TCP connections are bulk data transfer (FTP). The round trip time (RTT) varies from 70 ms to 670 ms, and are different for the 16 TCP connections. The target point, Q_0, is set to $1/3$ of the buffer size.

Figure 4.13(a) gives the average throughput of each connection over a 200 seconds simulation. RD-Internet is more effective in evenly sharing link bandwidth. RED incurs the bad performance in terms of bandwidth sharing, while DropTail incurs the worst performance.

To further investigate why RD-Internet outperform the RED and DropTail, we study the Figure 4.13(b) of queue occupancy under the various mechanisms. As shown in Figure 4.13(b), the RD-Internet keeps the queue length approximately at its desired operating



(a) Bandwidth utilization



(b) Buffer utilization

Figure 4.11: Bandwidth and buffer utilization

point, Q_0, so that the RD-Internet keeps the enqueue rate more stable than RED and Drop-Tail.

Based on this set of plots, we can make following observation. The Internet can not allocate bandwidth fairly for the TCP flows that have different round trip time. The Internet allocates more bandwidth to the TCP flows that have short round trip time. It is observed that some TCP applications do not achieve their fair-share bandwidths, while others grossly exceed their bandwidth shares. The main reason is that the Internet routers don't know the current states of the network, which leads to less than fair resource allocation. With the Resource Discovery feedback control mechanism, this effect can be removed. Our proposed model allocates TCP flows quite fairly regardless of different

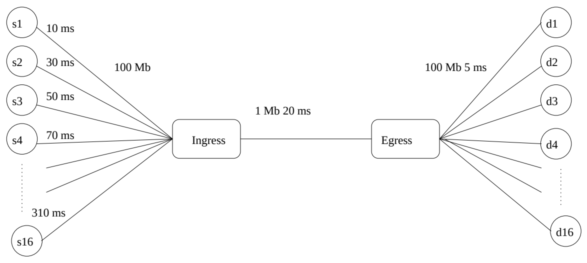


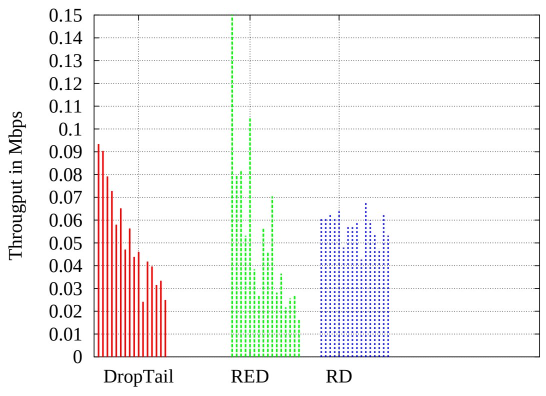
Figure 4.12: TCP flows with different RTTs

round trip time.

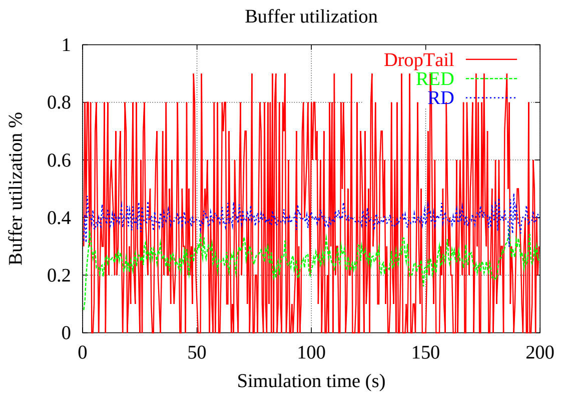
4.4.5 Simulation 3: performance under TCP and UDP traffic

In the third experiment, we study how the various mechanisms accommodate TCP traffic and UDP traffic. In the simulation, there are 4 TCP connections and one UDP connection competing for the bandwidth of a single bottleneck link. We use the experiment 1 topology (see the Figure 4.10). The bandwidth of the bottleneck link is set to 1 Mbps. The round trip time of all TCP connections is set to 30 ms. The UDP traffic is set as Constant Bit Rate (CBR) with constant rate as 1 Mbps. The threshold $minth$ ranges is set to 10 packets, the threshold $maxth$ is set to 3 times of the $minth$ value, and the buffer size is set to 4 times of the $minth$ value in RED. The parameter, Q_0, is set to 40 packets. The simulation is run for 200 seconds, and the throughput of connection is calculated in each 0.5 second period.

Figure 4.14 gives the average throughput of each connection. X-axis is time, and y-axis is the ratio of the throughput of connection to the bottleneck bandwidth. A DropTail mechanism is highly biased against TCP connection. This is because DropTail drops the packets regardless of the traffic. The RED mechanism performs better than DropTail but still is biased against TCP connections. RD-Internet is not biased against bursty TCP connection and performs better that of DropTail and RED, because it calculates the fair share dynamically.



(a) Bandwidth utilization



(b) Buffer utilization

Figure 4.13: Bandwidth and buffer utilization

4.5 Discussion 2: Resource Discovery over Differentiated Service Network

We have shown in last section that RD-Internet outperforms RED and DropTail in

1. providing better service discrimination and fair bandwidth sharing;
2. achieving high link utilization

We now explore the use of RD mechanism in the context of Assured Service of Differentiated Service architecture, RD-DiffServ. In the following subsections, the pseudo codes of the RD-DiffServ with In and Out mechanism and several sets of simulations will be

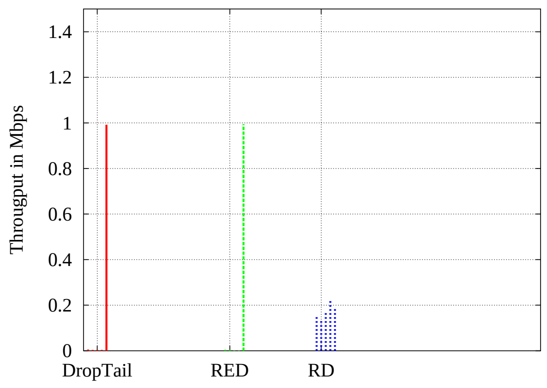


Figure 4.14: TCP flows and UDP flow

presented.

4.5.1 Resource Discovery Algorithm for DiffServ

Succinctly, RD-DiffServ can be considered as a combination of two RD-Internet instances, one for *in-profile* (IN) packets, and the other one for all the packets. RD-DiffServ differs from RIO in the following aspects. First, RD-DiffServ detects the available bandwidth and the available buffer as the control variables. This control variable is directly related to the bandwidth requirement of assured service, and more importantly, it expresses the "internal" network QoS states. Second, RD-DiffServ takes into account of both "external" requirement, Service Level Agreement (SLA), and "internal" state, Explicit Rate, to determine the admission of the arrival packet.

The Resource Discovery Protocol determines the domain capability or QoS states, Explicit Rate (ER), by calculating the product of Queue Control Function $f(Q)$ and Mean Available Rate, MAR.

$$ER = f(Q) * MAR \tag{4.10}$$

When the network operates above the target operating point, Explicit Rate (ER) is reduced due to $f(Q) \leq 1$ (refers to equation (5.1)) and throttling is performed fairly. However, when the network operates below the target operating point, Explicit Rate (ER) is allowed to increase its rate by a factor greater than 1, which enables the incoming traffic capable of using the available bandwidth to take advantage of it.

Pseudo code reflecting this algorithm is described in the Algorithm (3).

Algorithm 2 RD-DiffServ Parameter Settings

```
Ensure: $ER_i$: The ith class Explicit Rate
Require: The incoming traffic rate $R$
    $\beta$ : {The average ratio}
    $\alpha$ : {Congestion function parameter}
Require: Per-class Variable
    $MACR$ : {Mean Allowed Class Rate}
    $Q_0$ : {Target Class Queue Length}
    $Q0_{in}$ : {Target point for IN packet queue}
    $qlen_{in}$ : {Length of IN packet queue}
    $qlen$ : {Packet queue length}
    $ER$ : {Explicit Rate}
    $DPF$ : {Queue Control Factor}
    $weight_i$ : {The ith class weight}
    $Q_0 \Leftarrow$ 0.3 * BufferSize
    $Q0_{in} \Leftarrow$ 0.3 * IN BufferSize
    $MACR \Leftarrow$ 100Mb {Assign a large value}
    $\beta \Leftarrow 0.08$
    $\alpha \Leftarrow 1.23$
```

where Q_0 specifies the queue length to which the RD queue asymptotically converges (in our experiments, we set it to $1/3$ of the buffer size), and hence $Buffer\_size - Q_0$ is the maximum burst an RD router can accommodate. We use Q_0 to separate two stages: no congestion and congestion alert. When the queue length is larger than Q_0, we treat it as congestion alert stage. When queue length is less than Q_0, we treat it as no congestion stage. When an RD router is under the congestion alert stage, it attempts to keep the packet enqueue rate around an operating point by decreasing $\beta * (R/weight_i - MACR)$. When an RD router is under the no congestion stage, it attempts to keep the packet enqueue rate around an operating point by increasing $\beta * (R/weight_i - MACR)$

Algorithm 3 Resource Discovery Algorithm for DiffServ

```
Bandwidth Fair Control Function {on forward path at Ingress/core/Egress}
if RD.tag ==IN then
  if $qlen_{in} \geq Q0_{in}$ then
    if $R/weight_i \leq MACR$ then
      $MACR_{new} = MACR + \beta * (R/weight_i - MACR)$
    else
      $MACR_{new} = MACR$
    end if
  else
    $MACR_{new} = MACR + \beta * (R/weight_i - MACR)$
  end if
else
  if $qlen \geq Q_0$ then
    if $R/weight_i \leq MACR$ then
      $MACR_{new} = MACR + \beta * (R/weight_i - MACR)$
    else
      $MACR_{new} = MACR$
    end if
  else
    $MACR_{new} = MACR + \beta * (R/weight_i - MACR)$
  end if
end if
Queue Control Function {on the backward path}
if qlen $\geq Q_0$ then
  $DPF = (Buffer\_Size - qlen)/(Buffer\_Size - Q_0)$
else
  $DPF = 1 + (\alpha - 1) * (Q_0 - qlen)/Q_0$
end if
The ith class Explicit Rate
if qlen then
  $ER_i = MACR_{new} * DPF * weight_i$
else
  $ER_i = MACR_{new}$
end if
```

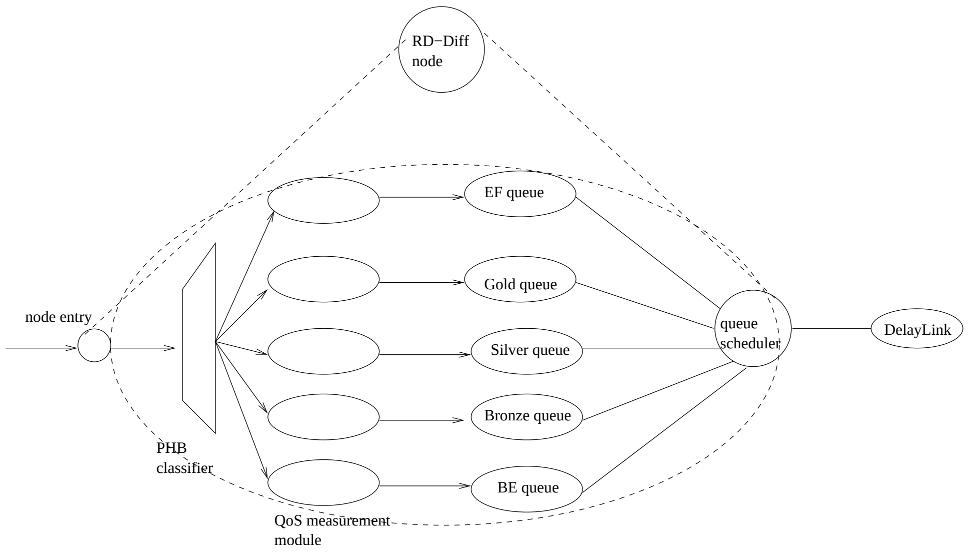


Figure 4.15: NS2 implementation model of a RD-DiffServ Interior node

4.5.2 Implementation Issues

Figure 4.15 depicts an implementation of a RD-DiffServ capable interior node. Note that the components of PHB Classifier, packet queues with various types of queue management schemes and output scheduler are commonly found in most DiffServ nodes. For a RD-DiffServ enabled node, a QoS states module, which is tightly coupled with a incoming traffic estimator and a per-queue measurement modules, is included. In our design of a packet queue, the *Queue/RIO* implements an AF PHB class with drop preferences. The four queues can be ranked according to their weight as Gold class (e.g., claim 40%), Silver class (e.g., claim 30%), Bronze class (e.g., claim 20%), and Best Effort class (e.g., claim less than 10%). In addition, the outputs of packet queues are controlled by Round Robin scheduler. Furthermore, for the edge routers, an additional admission control module is included in an ingress router and egress router, respectively.

All output queues in the RD-DiffServ architecture have the same architecture, which is outlined in Figure 4.15. Each class of traffic is associated with a FIFO per-class queue. When a packet is passed to a network interface a classifier looks up which class the packet belongs to, and places the packet in the appropriate per-class FIFO buffer. The per-class buffers have a finite size configured by the domain operators. After the incoming packet has been placed in a per-class buffer, the QoS states measurement module measure the QoS states in terms of the available bandwidth and available buffer. Our implementation is

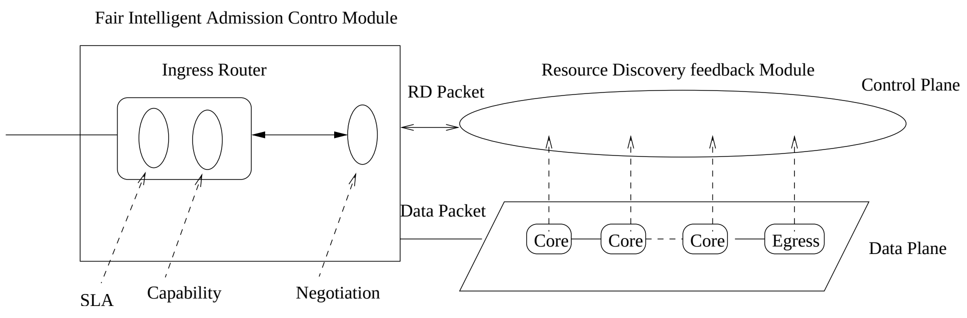


Figure 4.16: Admission control module in an ingress router

modular in the sense that it requires few changes to the current Internet router. The major change is to replace the queue manipulations. In other words, we first check whether the router is configured as a RD-DiffServ router or not, and if yes, we call *RDEnqueue* to enqueue the packet in a special data structure maintained by RD mechanism. If not, we simply use the default queue mechanism.

Figure 4.16 illustrates an admission control module. The admission control module can be considered as a token bucket model. The token bucket generation rate is determined by the explicit rate from Resource Discovery mechanism. The buffer at ingress router is used to store the packets that cannot be delivered right away until enough tokens get accumulated in the bucket.

4.5.3 Simulation Settings

We have implemented the Resource Discovery algorithm in the ns simulator, version 2. Simulations include scenarios investigating different parameter settings, topologies, multiple congested gateways and different mixes of TCP, CBR and ON/OFF flows. Additionally, scenarios with different bottleneck link capacities, different RTTs and number of TCP flows are investigated. Further simulations show that the max-min fair share is approximated sufficiently accurately if the maximum number of iterations of the algorithm computing the fair share is set.

Figure 4.17 shows the network simulated in this chapter.

Access links between sources to the ingress router have a capacity of 100 Mbps. Also the links between the egress router to the sinks have a capacity of 100 Mbps. The link delay varies with the simulations. For simulations in this section the RD-DiffServ with

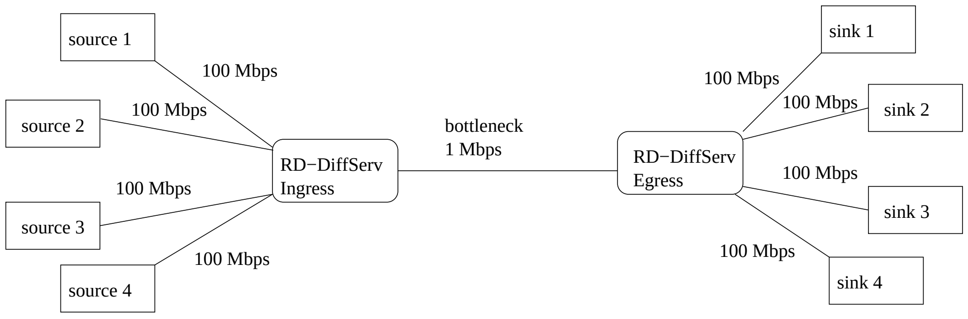


Figure 4.17: Simulated network

Aggregate	Committed rate (kB/s)
Gold	400
Silver	300
Bronze	200
Best Effort	less than 100

Table 4.1: Committed rates for aggregated flows

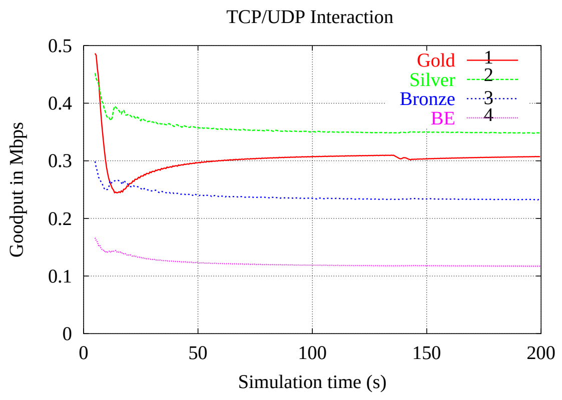
adoption of operating point has been implemented. We have set the operating point factor equal 70% in all simulations. The aggregates are marked with different committed rates, looking for a weighted sharing of the available bandwidth. The committed rate of each aggregate is shown in Table 4.5.3.

Common preferences for all simulations, except otherwise noted:

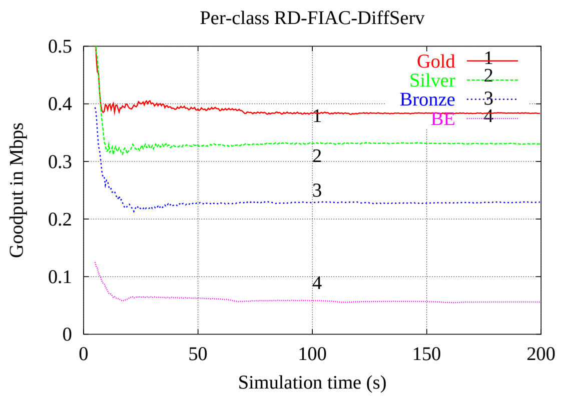
- All output-ports are capable of storing 500 packets.
- The simulation-time equals 100 seconds
- All output-ports except the output-port at the ingress/egress router served by the bottleneck link use RIO queue management.
- RIO parameters: ECN is disabled, *mean-pktsize=500 bytes*.

The simulation results are illustrated in several different kind of figures. Figures entitled

- "per-class throughput over time" show the per-class traffic bandwidth consumption of the bottleneck link as a percentage of the link capacity at the bottleneck link.



(a) DiffServ: CBR and TCP flows



(b) RD-DiffServ: CBR and TCP flows

Figure 4.18: Goodput of Classes (CBR and TCP flows)

- “queue-size over time” show the instantaneous and average queue size at the output port of the ingress/egress router. Unit for the x axis is seconds.
- “end-to-end delay” show the per-flow packet average delay from the source to the sink in seconds.

4.5.4 Simulation 1: RD-DiffServ with unresponsive CBR and TCP flows

RD-DiffServ ingress/egress router parameters: $w_q = 0.002$, $min_l = 20$, $max_l = 40$, $min_h = 50$, $max_h = 150$ packets, $maxp_h = 0.1$.

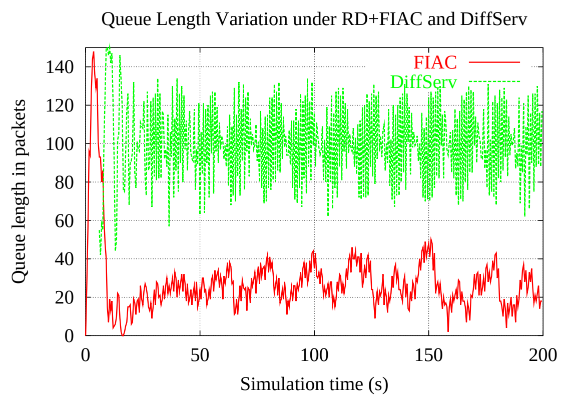
The bottleneck link is 1 Mbps, and delay is 5 ms.

The layout depicted in Figure 4.17 was used. As in the previous layout, two routers are connected through a 1 Mbps link; the input are 3 TCP connections with same RTT that is 100 ms and they are marked as Gold (claim 40%), Bronze (claim 20%, and BE (claim 10% less) respectively. The UDP traffic (CBR) is marked as Silver which claim 30%. The UDP rate was 1 Mbps. Figure 4.18(a) shows that under DiffServ the UDP flow (Silver) consumes nearly 40 percent of the link-bandwidth beyond the TCP flow (Gold). Packet sizes of TCP flows are uniformly distributed with a mean of 500 bytes. None of the flows terminates prior to the simulation. Here we observe a well-known result: TCP is considerably affected by UDP traffic, which gets all the bandwidth associated to its rate, see Figure 4.18(a). CBR flows consume a significant portion of the bandwidth between the ingress and the egress routers even they are marked as lower classes compared to the TCP flows which are marked as Gold class. This behavior causes the unfairness for the TCP flows. When there is UDP traffic in the link, the high-rate aggregate (e.g. Gold) fails to obtain the desired rate (40%).

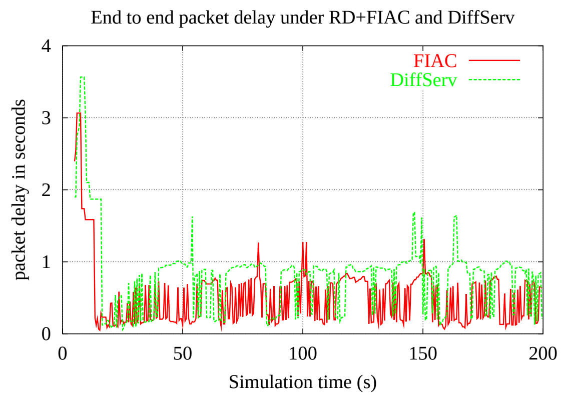
There are two initial conclusions of this test: the first one is that care should be taken into account when deciding if UDP traffic is to be aggregated with TCP traffic and processed in the AF. The second one is that bandwidth sharing can not be achieved for TCP flows in the presence of unresponsive UDP flows.

The simulation, Figure 4.18(b), shows that RD-DiffServ is able to penalize the aggressive flows (e.g., CBR flows) in case their bandwidth consumption is larger than their fair share. As soon as CBR flow being penalized its share closes to its fair share rate, $300Kbps$.

Figure 4.19(a) and Figure 4.19(b) show the reason why CBR flows being penalized: the average queue size from CBR flow are detected as too much aggressive and more than its fair share. Due to their unresponsiveness some of their packets are dropped by admission control module to keep the fair share state. This causes shorter end-to-end packet delay over time curves for RD-DiffServ, see Figure 4.19(b). This simulation lets us conclude that using RD-DiffServ for unfriendly flows, CBR flow, seems attractive, as it detects the available resources dynamically and the admission control module makes admission for flows based on the discovery from RD-DiffServ thereby keeps the queue



(a) queue length variation



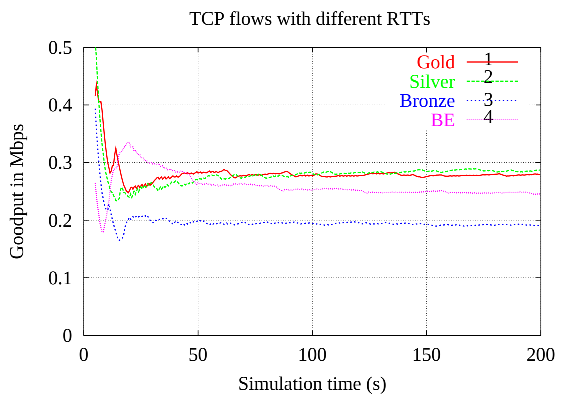
(b) end-to-end packet delay

Figure 4.19: DiffServ and RD-DiffServ: queue length variation and packet delay

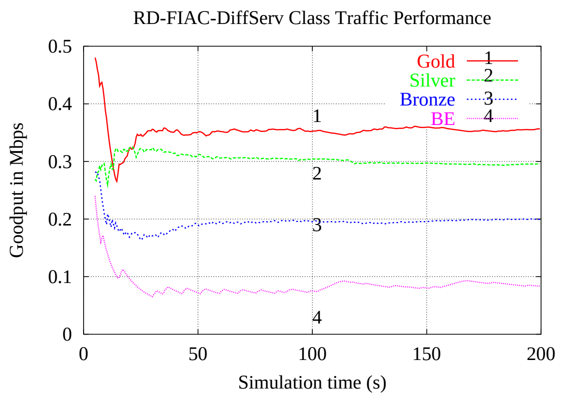
length around the target point and reduces the end-to-end packet delay.

4.5.5 Simulation 2: TCP flows with heterogeneous RTTs

We used the layout depicted in Figure 4.17 to study this subject. Two routers, DiffServ RIO router and Drop-tail router, are connected through a 1 Mbps link; DiffServ RIO router with three drop-precedence levels is available at the ingress router. 4 bulk data TCP flows send from the left to the right over the network with RTTs that go from 200 ms, 100 ms, 50 ms to 30 ms. The TCP traffic sent from node 1 to node 5 is marked as Gold class, the traffic sent from node 2 to node 6 is marked as Silver class, the traffic sent from node 3 to node 7 is marked as Bronze class, and the traffic sent from node 4 to node 8 is marked



(a) DiffServ: TCP flows with different RTTs



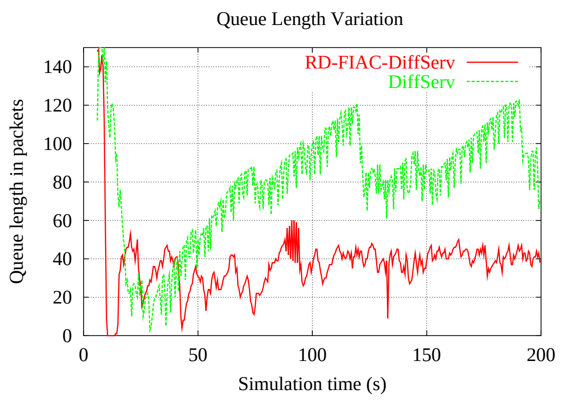
(b) RD-DiffServ: TCP flows with different RTTs

Figure 4.20: Goodput of Classes: TCP flows with different RTTs

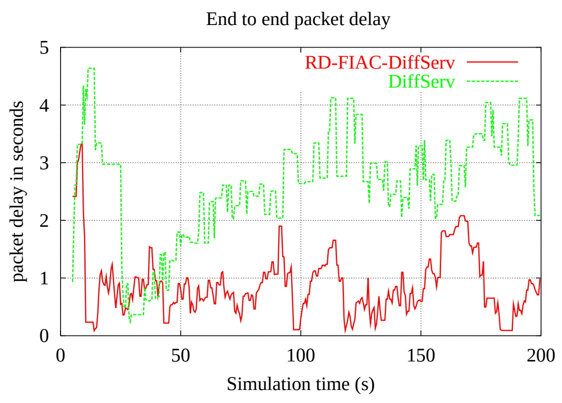
as Best effort class. The claim 40%, 30%, 20%, and 10% bandwidth of the bottleneck (DiffServ RIO node) respectively. In Best Effort class, TCP flow has short RTT, 30 ms. In other classes, TCP flows have long RTTs, 200 ms (in Gold), 100 ms (in Silver), and 50 ms (in Bronze).

DiffServ RIO parameters are set as follows: for each class queue, $min_{th} = 10$, $max_{th} = 40$, $maxp = 1/50$, respectively.

Figure 4.20(a) shows that the bandwidth share decreases as the RTT increases. The best effort (BE) traffic achieved more bandwidth share due to its short RTT. It can be said that Assured Forwarding (AF) does not assure an “Assured Bandwidth” service for TCP connections with different RTTs, since throughput depends on RTT. Sessions with small



(a) queue length variation



(b) end-to-end packet delay

Figure 4.21: DiffServ and RD-DiffServ: queue length variation and packet delay

RTTs react faster to losses in the network, so that no big bursts are produced.

Based on this set of plots, we can make following observation. The DiffServ can not allocate bandwidth fairly for the classes which have different round trip time. The traditional DiffServ allocates the Silver class (short RTT) more bandwidth, around 300 Kbps, which is beyond the Gold class. Our proposed model allocates each class quite fairly according to their service level agreement.

The end to end packet delay (flow 1 in Gold) and queue length of bottle neck router are shown in Figure 4.21. From the results, the proposed model achieves better performance compared to DiffServ in terms of packet delay and queue length. The average queue size stays around the target point, 40, as on the ratio of short TCP flows arrival rate to the link capacity, RD-DiffServ determines its fair share and thereby takes the admission control early. If a TCP flow with short RTT is detected too aggressive in terms of buffer and bandwidth, RD-DiffServ scheme work with admission control module will drop its packets and causes significant reduction of its incoming rate. Consequently, it will reduce the queue length and end-to-end packet delay.

4.5.6 Related Research on Resource Discovery Modeling and Parameter Setting

As a rule of thumb, it recommends to set target point from ten packets to 40 packets. The exact value of target point has to depend on the burstiness of the arrival process, link capacity, propagation delay and maximum buffer size. Additionally, it is remarked that the optimum setting of target point balances a trade-off between high queuing delay and poor link utilization. Without giving quantitative guidelines, the latter two statements imply that the target point setting should be directly proportional to the bandwidth*delay product of the scenario.

In measurements of TCP with RED have shown that link utilization decreases in case of DiffServ. Some researchers recommend to set the buffer size greater or equal to the bandwidth*RTT product of the path, which seems impossible for very high speed WANs. Although the buffering requirements of RED are significant, setting the buffer size equal to the bandwidth*RTT product would be overly conservative in case of TCP flows in steady state.

Based on the investigation of DiffServ RIO behavior in the presence of a small number of TCP flows, we proposes simple some guidelines for Resource Discovery (RD) parameter setting. The results can be summarized as follows.

- **target point:** As a rule of thumb, it is recommended to set the target point, Q_0, to $1/3$ of the buffer size. This recommendation is not supported by a model or performance analysis. It determines when to start restricting/encouraging the incoming traffic rate, it also determines the slope of the queue control function, or in other words, the rate at which $f(Q)$ increases/decreases as a function of queue length. Large target point allows the queue to build up to a large value before control is activated, and reduces the buffer capacity allocated to drain the queue. Hence tight queue control is required to make the drain capacity sufficient. Setting target point to a large value allows more packets to be injected into the network therefore avoids link underutilization. It must be set to allow the drain of the queue before the buffer limit is reached, and not to take the risk of buffer overflow when overloaded. A small value of target point is desired for smooth operation and shorten the packet delay time. Meanwhile fully link utilization must be guaranteed to achieve high throughput.

- **Parameter α:** Parameter α is used to for the queue control function during no congested period. It determines how much excess capacity would be allocated when the queue length is zero. It also determines the slope of the line, or how quickly $f(Q)$ increases as a function of queue length when the router is non-congested. A large value of α increases the amount of over-allocation. High value of α can potentially result in long queues thus prolong the packet delay time. The abrupt variation makes the scheme sensitive to both the queue length and the value of α. Small variation in delay may cause large change in queue length and the control module hence leads to instability. In the RD scheme, the incoming traffic is controlled to be around the optimal $MACR$ during all time. Network remains at a stable status around target operating point. A mild control by the linear function is sufficient to bring the network back to the target point. Therefore small value for α is more desirable by the queue control function. Specially, α should be a little larger than 1 to allow a small over-allocation above 100% and bring the network back to the target point. Note that the factor must be larger than 1 to encourage load increase and avoid link under-utilization.

- **Parameter insensitivity to the traffic control:**

  Resource Discovery control mechanisms take the buffer utilization and available bandwidth into account as a measure of the severity of congestion. But the parameters setting is also affected by the traffic characteristics. It is hard to find the general parameters setting for the all of the traffic. It needs more investigation in future.

- **Impact of parameter settings:** In the RD scheme, linear queue control functions are employed. The two curves of linear functions intersects in target point and change smoothly with queue change. The bandwidth allocation is consistent and resulting admission rate fluctuates consistently and smoothly around the optimal point. Minor change on target point or α has minor impact on the curves. Therefore minor parameter mistuning won't degrade the control effect and the network performance. The RD scheme is robust and relatively insensitive to parameter setting.

4.6 Fairness Analysis of RD-DiffServ Model

In this section, we analyze the stationary behavior of a bulk data transfer TCP connection under the RIO architecture and show that RIO cannot achieve throughput assurance and proportional fairness. Based on the derived analytic results, we analyze RD-DiffServ and compared to RIO.

4.6.1 A TCP Performance Model under the Differentiated Service

Several analytic models have been proposed to characterize the stationary behavior of a bulk data transfer TCP flow as a function of packet loss ratio, round trip time, and other parameters. The most well-known result (refer to [25]) is that the steady state throughput of a long-lived TCP connection is upper bounded by

$$throughput \lesssim \frac{MSS}{t_r} * \frac{C}{\sqrt{p}} \tag{4.11}$$

where MSS is the segment size, t_r is the round trip time, C is a constant, and p is the packet loss probability. Padhye [48] proposed a model that captures the impact of not only TCP's fast retransmit mechanism but also TCP's timeout mechanism on the throughput. These models are usually derived under the following assumptions:

- The sender always has data to send if it is permitted by the congestion window;
- The lifetime of the connection under consideration is at least in the order of several round trip times;

- The receiver advertisement window is always large enough so that the congestion window is not constrained by the advertisement window;
- Packet losses are independent events with the same, small probability.

The only last assumption that hinders the results from being directly applied to the RIO, because the packet loss probability becomes dependent on the buffer occupancy in RED.

4.6.2 RD-DiffServ Model Fairness Analysis

We follow the stochastic model of TCP congestion control model, and focus on TCP's congestion avoidance mechanism in which the congestion window size, W, is increased by $1/W$ each time an ACK is received, and is halved each time packet loss is detected. Since we do not intend to derive an accurate throughput formula, we do not consider timeout effects.

We model the TCP congestion avoidance behavior in terms of "round". A round begins with the back off transmission of W packets, where W is the current TCP congestion window size. Let b be the number of packets acknowledged by an ACK. Many TCP implementations send one cumulative ACK for two consecutive packets received, and b is usually equal to 2. If W packets are sent in the first round and are all correctly received and acknowledged, then W/b ACKs will be received. Since each ACK increase the window size by $1/W$, the window size is $W + 1/b$. That is, under the congestion avoidance mechanism and in the absence of packet loss, the window size increases linearly in time, with a slope of $1/b$ packets per round trip time. Once all packets in a congestion window have been sent in back off manner, no other packets are sent until the first ACK is received for these W packets. The duration of a round is equal to the round trip time and is assumed to be independent to the congestion window size. We assume that if a packet is lost in a round, all remaining packets transmitted until the end of the round are also lost. We define a "cycle" to be a period between two packets losses. Suppose a TCP sender starts to send data at time $t = 0$. For any given time $t > 0$, we define N_t to be the number of packets transmitted and acknowledged in the interval $[0, t]$, and $B_t = N_t/t$ to be the goodput on that interval. We define the long term steady-state TCP goodput, B, as

$$B = \lim_{t \to \infty} B_t = \lim_{t \to \infty} \frac{N_t}{t} \tag{4.12}$$

To facilitate the analysis, we also define the following terms:

- N_i: the number of packets sent in the ith cycle and eventually received.
- D_i: the duration of the ith cycle.
- W_i: the congestion window size at the end of the ith cycle.
- W_r: the ideal window size of IN packets for a flow with target rate r.
- p_d^i: the average packet dropping probability for IN packets.
- p_d^o: the average packet dropping probability for OUT packets.
- p_d: the average packet dropping probability for IN and OUT packets.

We are interested in establishing the relationship between the goodput, $B(r)$, of a TCP connection with target rate r, and the average packet dropping probability of OUT packets p_d^o. We model W_i as a Markov regenerative process with rewards N_i, and define the long term goodput B as

$$B = \frac{E(N)}{E(D)} \tag{4.13}$$

In the ith cycle, the duration $D_i = \sum t_{ij}$. Under the assumption that the round trip time t_{ij} is a random variable that is independent of the congestion window size, we have

$$E(D) = (E(X) + 1) * E(t_r) \tag{4.14}$$

where $E(t_r)$ is the average round trip time. We first observe the following relationship between W_i and X_i:

$$W_i = \frac{W_{i-1}}{2} + \frac{X_i}{b} \tag{4.15}$$

It then follows that:

$$E(X) = \frac{b * E(W)}{2} \tag{4.16}$$

From above, we have

$$E(N) = \frac{E(X)}{2} * (\frac{3 * E(W)}{2} - 1) + \frac{E(W)}{2} \tag{4.17}$$

Assume $p_d^i \cong 0$ (which holds if the network is well-provisioned), we then have

$$E(W) = \sqrt{\frac{8}{3b} * \frac{1}{p_d^o}} + o(\frac{1}{\sqrt{p_d^o}}) \tag{4.18}$$

In the ith cycle, there are X_i round (approximately). Suppose the target rate is r. The window size designated for IN packets is then $W_r = r * t_r$, and the ratio of IN packets to all packets in a cycle can then be expressed as $W_r * X_i/N_i$, where N_i can be expressed as

$$N_i = \sum_{k=0}^{X_i/b-1} (\frac{W_{i-1}}{2} + k) * b + \beta_i \tag{4.19}$$

$$= \frac{W_{i-1}}{X_i} + \frac{X_i}{2} * (\frac{X_i}{b} - 1) + \beta_i \tag{4.20}$$

$$= \frac{X_i}{2} * (\frac{W_{i-1}}{2} + W_i - 1) + \beta_i \tag{4.21}$$

Here β_i is the number of packets sent in the last round, and is assumed to be uniformly distributed between 1 and W_i, i.e., $E(\beta) = E(W)/2$, and hence,

$$\frac{W_r X_i}{N_i} = \frac{W_r}{\frac{W_{i-1}}{4} + \frac{W_i}{2} + \frac{\beta_i}{X_i} - \frac{1}{2}} \cong \frac{W_r}{\frac{W_{i-1}}{4} + \frac{W_i}{2}} \tag{4.22}$$

Consider two TCP connections with target rate $r1$ and $r2$ in the under-subscribed case (in which $W_r << E(W)$). The ratio of the long term goodputs of the two connections can be approximated as

$$\frac{B_1}{B_2} \approx \frac{E(t_{r2})\sqrt{p_2^o}}{E(t_{r1})\sqrt{p_1^o}} \tag{4.23}$$

The implication of Equation 4.23 is two-fold: first, the current Differentiated Service architecture, RIO, cannot provide throughput assurance and proportional share of surplus bandwidth, if the TCP connections are of different target rates and RTTs. Second, if we set the packet dropping probability for packets as

$$\frac{p_1^o}{p_2^o} = \frac{r_2^2 E(t_{r2})^2}{r_1^2 E(t_{r1})^2} \tag{4.24}$$

then the ratio of the long-term goodput is approximately

$$\frac{B_1}{B_2} \approx \frac{r_1}{r_2} \tag{4.25}$$

so by adjusting the packet dropping for packets according to the target rate and RTT, it is possible to achieve throughput assurance and proportional bandwidth sharing. Our admission control module based on RD-DiffServ model calculates the packet dropping probability in terms of target rates and explicit rate which relates to RTTs accordingly.

4.7 Feedback Loop Analysis Background

The RD framework enables a resource discovery approach which leads to distributed resource sharing to QoS through the feedback signals. We now explain how the feedback signals can be derived, and how they link to the notions of "fairness", which is based of the work on Kelly and co-workers [32].

A network is a set of linked resources, and a path through the network will use some subset of these resources in a specific order. The description is general, but it may help to think of the current Internet where typical resources are bandwidth or buffer capacity in a router output port. We shall assume fixed routing, so a route determines a specific subset of the resources. Let J be a finite set of resource indexed by j, and R a finite of traffic (aggregate flow or individual flow), indexed by r, where the $0-1$ incidence matrix $A = A_{jr}$ indicates whether traffic r uses resource j or not. Suppose further that we can characterize a flow's preference for bandwidth by a concave, non-decreasing utility function, $U_r(x)$, which is appropriate for 'elastic' traffic. For convenience assume U_r is everywhere differentiable. It is natural for traffic to attempt to pursue their own ends, and seek to maximize $U_r(x_r)$ over $x_r \geq 0$. But resources are finite, so this behavior will tend to overload certain resources. To counteract this, suppose that when the load on resource j is y_j, the resource incurs a cost at rate $C_j(y_j)$. The a social planner would seek to maximize

$$\sum_r U_r(x_r) - \sum_j C_j(\sum_r A_{jr}x_r) \tag{4.26}$$

At the optimum,

$$U'_r(x_r) = \sum_{j \in r} p_j(y_j) \tag{4.27}$$

where p_j is the derivative or shadow price $p_j(y_j) = C'_j(y_j)$ with a corresponding load on resource j given by

$$y_j = \sum_r A_{jr}x_r \tag{4.28}$$

Suppose now we send feedback closed-loop to the edge router whereas we do admission control. We want RD packets to be additive, which means RD packets need to be proportional to the traffic loading, suggesting a feedback RD packet of the form $t_r x_r$. If these reflect a 'charge' to the traffic, the traffic wants to maximize the net return, which is to maximize

$$U_r(x_r) - t_r x_r \tag{4.29}$$

hence if these 'price' are set correctly, that is if

$$t_r = \sum_{j \in r} p_j(y_j) \tag{4.30}$$

the traffic acting independently will drive the system toward the social optimum. Note that the congestion price t_r internalizes the congestion costs, and is sum of the resource prices along the route.

We now show that is it possible for the network to mandate all traffic to use Fair Intelligent Admission Control (FIAC) which will impose a particular form of fairness in terms of class. Suppose that each traffic (aggregate flow or flow) has to use the 'willingness-to-pay' after FIAC adjusting

$$\frac{d}{dt}x_r(t) = \kappa_r(w_r - x_r(t)\sum_{j\in r} p_j(y_j(t))) \tag{4.31}$$

This is equivalent to using a utility function of the form

$$U_r(x) = w_r \log x_r \tag{4.32}$$

and hence the Network is implicitly maximizing $\sum_r w_r \log x_r - \sum_j C_j(y_j)$. It can be shown that such an allocation is weighted Proportionally Fair [32]. At the optimum,

$$w_r = p_r x_r \tag{4.33}$$

hence w_r is the amount of the traffic r is prepared to pay per unit time. If the edge router adapts the parameter w_r over time according to

$$w_r = x_r U_r'(x_r) \tag{4.34}$$

then the traffic optimum and social optimum again coincide, based upon an underlying proportional fairness model. Updating w_r according to the resource discovery is equivalent to the traffic seeking to maximize

$$U_r(\frac{w_r}{p_r}) - w_r \tag{4.35}$$

Much of the literature has concentrated on the 'willingness to pay' algorithm, which is equivalent to $F_r(x_r) = w_r \log x_r$. This is appealing since firstly, w_r has a ready interpretation, secondly the behavior of aggregates is linear with respect to w_r and thirdly the underlying fairness model, weighted proportional fairness.

Suppose FIAC module at edge routers adjust the traffic rate according to the Gibbens and Kelly ‘willingness-to-pay’ strategy [27],

$$\frac{d}{dt}x_r(t) = \kappa_r(w_r - p_r(t)x_r(t)) \tag{4.36}$$

where $p_r(t)$ is the feedback RD packets along the route, the sum of the resource prices $p_r(t) = \sum_{j\in r} p_j(t)$. Then w_r is the amount the traffic r is prepared to pay per unit time, and κ_r is a gain parameter. If the traffic increase rate at rate x_r, it is prepared to pay w_r and the network feedback charge $p_r x_r$. In the steady state, traffic has a throughput proportional to w_r, i.e. $x_r = K * \frac{w_r}{p_r}$. Recall that this algorithm is equivalent to using a logarithmic utility function, $U_r(x_r) = w_r \log x_r$, hence the resulting allocations are weighted proportionally fair. The parameter κ_r affects the rate of convergence. Equation 4.36 always converges to the equilibrium point, however this assumes instantaneous feedback. Note that the multiplier only depends on quantities along route. The larger the κ_r, the faster the convergence, but if κ exceeds the bound the system may be unstable. When a connection first enters, it can take some time to reach this point.

4.8 Summary

In this chapter we investigated the Resource Discovery (RD) module for collecting the QoS states. Based on a novel feedback approach that focuses on network-wide performance issues, we studied the relationships between the RD module and the normal router infrastructure. Using the RD algorithm as well as the simple scheduling algorithms, FIFO or Round Robin, we demonstrated that with additional QoS states information carried in the RD packet header and added reasonable complexity at network edge, both the network link utilization level and the fairness can be significantly improved. We further investigated the Internet router equipped with RD module (RD-Internet) and DiffServ router equipped with RD module (RD-DiffServ) on the performance trade-offs as well as the provisioning of the algorithms. In particular, we showed that with relatively RD overhead, the RD algorithms can attain fairly good performance. These results illustrate the fundamental trade-offs in the design of Resource Discovery algorithm, and shed light on the provisioning in support of edge-to-edge quality of service.

However, it is worth noting that Resource Discovery control mechanism may not work well over all circumstances. In case of TCP connections with same round trip times, Drop-tail or RED mechanisms have achieved better performance than RD control mechanism in terms of throughput. This is primarily because our Resource Discovery mechanism needs Resource Discovery traffic to feedback the current network QoS states that results in bandwidth and processing. Actually, Resource Discovery control mechanism is an edge-to-edge control mechanism that can be combined with the Internet and DiffServ infrastructures. This new approach improves fairness in TCP/UDP interactions, and RTTs disparities.

Chapter 5

Fair Intelligent Admission Control

In this chapter we discuss Fair Intelligent Admission Control at the per-class level (Per-class FIAC) and per-flow level (Per-flow FIAC) that are used to solve per-class service guarantee and per-flow fairness within a class. We also discuss the interaction between Resource Discovery (RD) and Fair Intelligent Admission Control (FIAC).

1. Per-class FIAC is designed to ensure per-class edge-to-edge Prioritization type of QoS. It incorporates at edge router to admit the application based on “internal” network QoS states.

2. Per-flow FIAC is designed to ensure individual flow fairness within DiffServ class. It does not operate at congested output ports of routers, but uses per-flow fairness algorithm at edge router to achieve per-flow fairness.

We use the FIAC scheme in all simulations in this chapter. We have made a comparison with DiffServ scheme and the results are comparable. Before showing some computational results, we discuss linear stability solution for the problem. Details are explained in subsequent sections.

5.1 Introduction

Traditionally, best-effort service has been the main type of service available over networks. While this type of service has contributed much toward the rapid growth of the

Internet, it cannot support applications that have real-time requirements. One way to provide such a service is to strictly control input traffic by admission control and schedule packets based on the requirements of the connection they belong to.

In this chapter we propose and develop a novel Fair Intelligent Admission Control (FIAC) as a unifying admission control framework to provide scalable support for guaranteed services. More specifically, this FIAC provides a unifying framework to characterize, in terms of their abilities to provide per-class edge-to-edge QoS, per-flow fairness within a class, and the end-to-end QoS by their concatenation. The key construct in the proposed FIAC is the interaction between FIAC and Resource Discovery which references and updates the "internal" QoS states as RD packet traverse each core router. A crucial property of the FIAC algorithm is that they can be incorporated just at edge router. In this sense, the FIAC scheme is core stateless, as no per-flow state is needed at core routers for computing fairness states.

In this chapter we lay the theoretical foundation for the definition and construction of the FIAC scheme. We describe how interact with Resource Discovery module, and based on "internal" QoS states, establish edge-to-edge per-class service and per-flow fairness guarantee. Consequently, we demonstrate that in terms of support for assured services, the proposed FIAC scheme preserves the scalability of the DiffServ model. Furthermore, we show that the notion of FIAC with RD leads to the design of the new state-less QoS router.

The objectives of the proposed FIAC are two-fold. First, as a admission control mechanism, it must not mandate any specific scheduling algorithms to be employed in a network in order to provide guaranteed services. In other words, it must allow for diverse scheduling algorithms as long as they are capable of providing QoS guarantees. In fact, our FIAC can accommodate both core stateless scheduling algorithms and stateful scheduling algorithms. Second, the FIAC provides a QoS abstraction for admission control mechanisms that decouples that data plane from the QoS control plane. This abstraction facilitates the design of a resource control architecture, where QoS states are maintained only at edge router, while still being capable of providing QoS guarantees with similar granularity and flexibility of the DiffServ architecture. We believe that these two objectives are important in implementing quality of service in practice. For example, the ability to employ diverse scheduling algorithms not only encourages choice and competition among equipment vendors and ISPs, but also, perhaps more importantly, allows a network and its services to evolve. Similarly, by maintaining QoS states only at edge router, core routers are relieved of QoS control functions making them potentially more scalable, reliable, and efficient. Furthermore, a QoS control plane which is decoupled from the data plane allows an ISP to deploy sophisticated provisioning and admission control algorithms to optimize network utilization without incurring software upgrades at core routers.

The rest of this chapter is organized as follows. In the next section we will briefly outline the basic architecture of FIAC. In Section 5.2 we outline a FIAC architecture in the context of an ideal edge-to-edge operational region. This FIAC is extended in Section 5.3 to account for the per-class quality of service. In Section 5.4 we introduce per-flow FIAC to address flow fairness within a class. In Section 5.5, we apply the FIAC to address the fairness problem of DiffServ under variable packet size situation. In Section 5.6, we summarize the contributions of the chapter.

5.2 Fair Intelligent Admission Control: Basic Architecture

In this section we outline the basic architecture of the proposed unifying admission control framework- the Fair Intelligent Admission Control (FIAC). The FIAC algorithm is based on the incoming traffic measurements and the QoS states report from the Resource Discovery module to make admission decision.

Conceptually, FIAC has the following characteristics:

- Scalability: The admission control schemes are scalable regardless of the number of flows in the system. This means that the overhead of admission control is independent of the number of flows in the system.

- Effectiveness: The admission control schemes try to maximize the bandwidth utilization to the extent possible. The result is a system with high probability of admitting a new flow if resources are available.

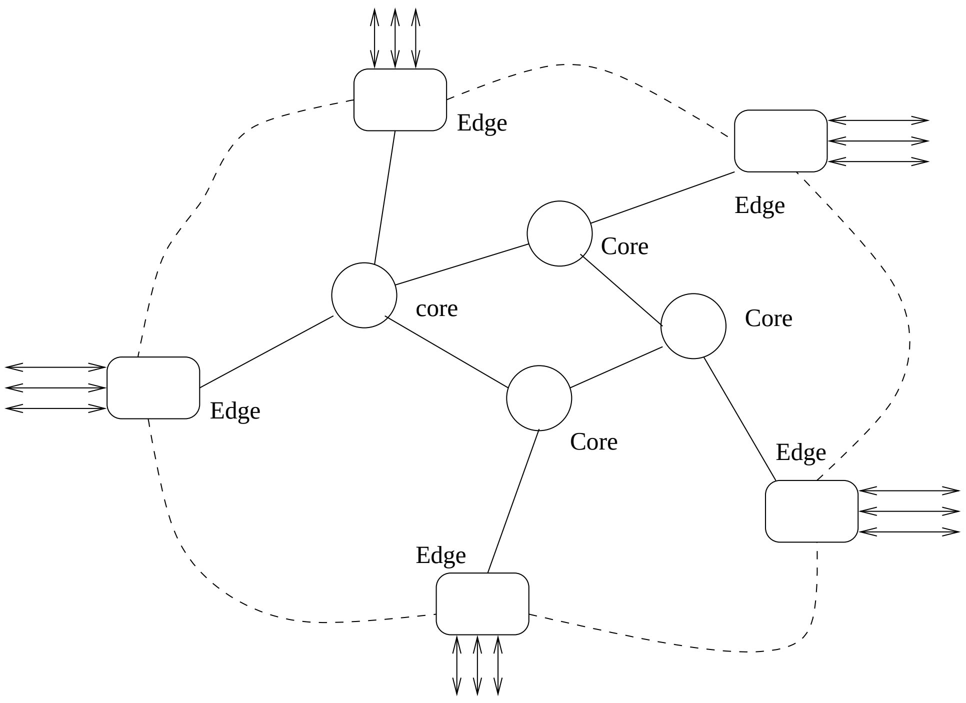


Figure 5.1: The network model

- Compatibility: The admission control methods presented are compatible with current industrial practice. Minor modifications to communication infrastructure of routers, signaling protocols, or packet format, etc are required.

Now we describe the model and define the terminology (network, link, path, flow and service classes) that will be used in the following sections.

Network

A network consists of a number of nodes (e.g., routers and hosts). Nodes are connected by physical links on which packets are forwarded. We assume that the network can be partitioned into multiple domains. Each domain can have many routers. Figure 5.1 illustrates an example of one domain. The network resources are controlled by the FIAC module at edge router.

Routers and Links

There are two kinds of routers in a domain: edge router and core router as shown in Figure 5.1. As the name implies, the edge router is at the edge of domain, and the core router is in the interior of the domain. A router has multiple input and output links. We assume that there are N input links at a router. For the incoming packet, the router determines the output link by looking up its routing table, and transports the packet to the proper output link, which, in turn, connects to the next hop. It is possible that a packet may have to be queued at the output link because of the limitation of output link capacity. We denote C as the output link capacity, in bits per second. Packets may exceed the capacity at the output link, and thus the packet may suffer some queuing delay at the output link.

Paths

A packet is transmitted along a path. In a network, a path is a route between any two nodes.

Flows and Classes of Service

An IP traffic flow is defined as a set of packets being sent from the application in the network during a certain time interval. All packets belonging to a particular flow have a set of common properties. A flow can be constrained by a leaky bucket at the edge router when it enters the domain. This edge router is called the ingress router to this flow. Following the domain, flows are aggregated into classes. QoS requirement and specifications of the traffic carried by the flows are defined on a class-by-class basis.

Fair Intelligent Admission Control (FIAC) was designed to solve two problems of the Internet and traditional DiffServ: (a) per-class (under DiffServ) or flow (under the Internet) QoS under the traffic overloading or internal network provisioning changes situation; (b) per-flow fairness within a class (under DiffServ) or flow fairness (under the Internet).

We believe the cause of the first problem is due to the domain (DiffServ domain or Internet domain) being unaware of "internal" network capability or QoS states and hence it is unable to ensure traffic conditions at which Per-Hop-Behavior can operate satisfactorily. Regardless of the "internal" network QoS states, the domain (DiffServ domain or the Internet domain) still admits overloading traffic into the network and this creates a situation similar to that of a "under-provisioned" network, over which the domain does not perform well. It needs a responsive admission scheme.

The second problem is caused by the fact that DiffServ is not interested in the composition of a DiffServ class. Packets labeled with the same code point are treated exactly the same according to the class' PHB. The amount of bandwidth received by a flow within a class depends heavily on the way the flow is controlled externally. For example, a UDP flow can always grab more than its fair share of the bandwidth available to the class com-

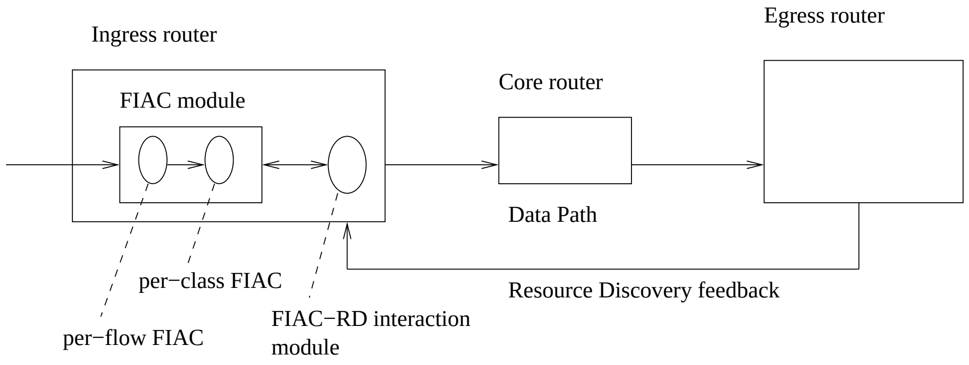


Figure 5.2: FIAC Components

pared to a TCP flow with the same DiffServ class. In the same way, a short round trip time TCP flows can gain more bandwidth share than a long round trip time TCP flow. We address this problem using a per-flow level admission control which resolves flow's fairness before they are aggregated into a DiffServ class. The similar situation can be found under the Internet domain.

The philosophy of FIAC is to prevent overloading traffic as early as possible (at edge node) so that it will alleviate network congestion situation and guarantee per-class service. Furthermore, FIAC per-flow mechanism can guarantee per-flow fairness among flows which belong to the same class.

The idea behind FIAC is to setup a communication channel between "internal" (network capability report) and "external" (QoS requirements) and to make admission decision based on "internal" state (network capability report) and "external" requirement (QoS requirements).

To achieve the above two goals, we designed three modules in FIAC: (a) per-class FIAC; (b) per-flow FIAC; and (c) FIAC-RD interaction module with the Resource Discovery. Figure 5.2 illustrates the relationship between the three components.

Traffic arriving at the ingress router of the domain is differentiated by its QoS requirements. All arriving traffic with the same QoS requirements are treated as the same aggregate class. In DiffServ, a typical arrangement would be to categorize traffic into premium (EF), gold, silver, bronze, and best-effort classes [60]. We also follow this arrangement to design five physical queues to map above categories in DiffServ domain.

FIAC is a predictive control scheme which performs two functions: (a) it estimates the

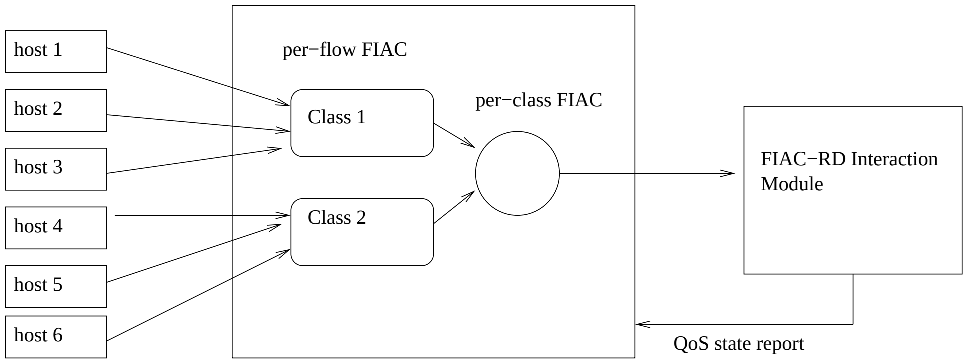


Figure 5.3: Fair Intelligent Admission Control Module

amount of aggregated traffic of a DiffServ domain based on history of the traffic, and (b) it makes admission decision based on the feedback QoS states information from Resource Discovery module.

Figure 5.3 illustrates the relationship between FIAC module and Resource Discovery feedback module.

The Fair Intelligent Admission Control module (FIAC) is an entity that implements the admission control. The FIAC is comprised of the following components:

1. **FIAC-RD interaction module:** In the FIAC-RD interaction module, it takes the network QoS state information from the Resource Discovery module to the FIAC and passes the admission rate to the RD module.

2. **Per-class FIAC:** Based on feedback DiffServ QoS state report, it adjusts the incoming traffic rate to satisfy per-class requirement.

3. **Per-flow FIAC:** it is responsible for ensuring fairness among flows within a class.

‘

FIAC works on edge router and considers the buffer’s status and internal network capability to make admission decision. Figure 5.4 illustrates the buffer consideration in FIAC module. The key steps of FIAC are as follows:

- First, FIAC tries to maintain the queue length close to a target value Q_0 so that the queue does not go empty and the output link is always keeping busy.

- Second, FIAC adjusts the per-class incoming traffic rate to minimize variations in buffer queue length and in packet delay.

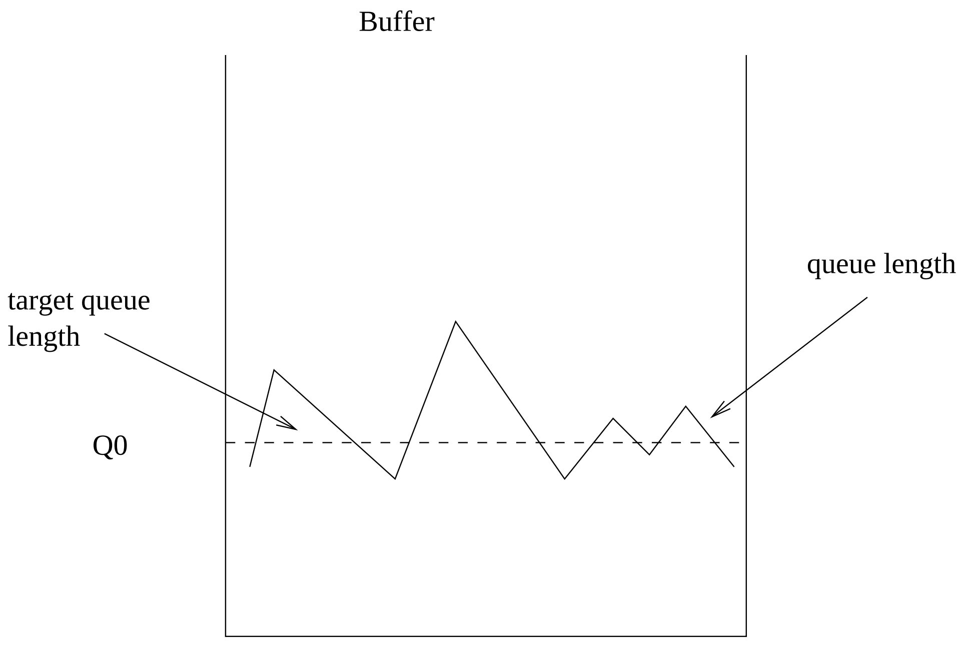


Figure 5.4: buffer consideration

- Third, FIAC tries to allocate bandwidth proportional to the class service requirements and to distribute the unused bandwidth proportional fairly for per-class traffic that can use an additional share. To achieve this objective, FIAC oversells bandwidth when the network operates below the target point.

- Finally, FIAC is able to prevent congestion as early as possible so that it can provide stronger service compared to DiffServ even in traffic overload or internal network capacity shrinking conditions.

Figure 5.2 illustrates the per-class and per-flow framework in FIAC module. The FIAC provides a unifying framework to formalize the per-hop behavior of a core router and to quantify its ability to provide quality of service. This formalism is independent of the current mechanism employed by the core routers, be it stateful or stateless.

In summary, while based on the Internet or DiffServ paradigm, the FIAC renders the same expressive power and generality, in terms of the ability to provide relative assured service. Furthermore, FIAC architecture provides a unifying framework upon which a scalable QoS provisioning and admission control framework can be built, where all QoS reservation states for guaranteed services are eliminated from the network core.

The remainder of this chapter is devoted to laying formal foundation for FIAC system.

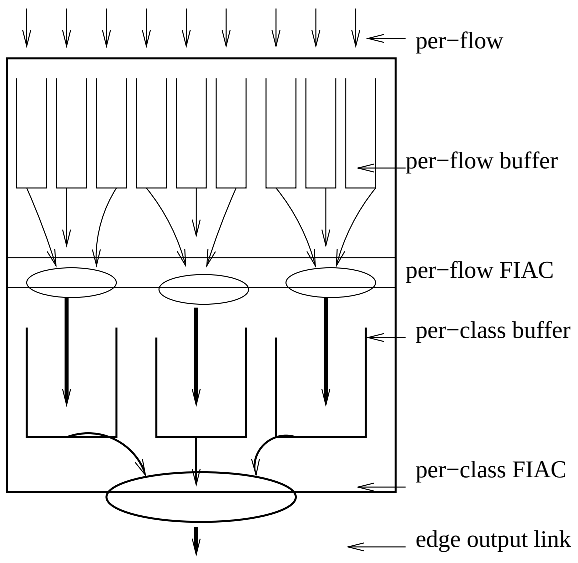


Figure 5.5: Per-flow FIAC and Per-class FIAC

We also illustrate how per-class FIAC and per-flow FIAC algorithms fit into the unifying framework. Issues of implementation and admission control will also be briefly discussed.

5.3 Per-class Fair Intelligent Admission Control

Per-class FIAC determines the Explicit Admission Rate (EAR) of a class based on the internal network capability rate (reported by the Resource Discovery Mechanism), the current available router resources (e.g. available buffer, available bandwidth). To calculate EAR, FIAC performs two functions, queue control function and bandwidth fair control function which are explained in the following sections.

Queue Control Function

FIAC employs "queue control function" to express the degree of network congestion it can tolerate. Since the queue builds up and drains out continuously, the queue control function can regulate the queue fluctuation smoothly. The target point, Q_0, represents the optimal buffer utilization point we want to maintain. The linear function is defined as in following equation.

$$f(Q) = \begin{cases} \frac{Buffer\_Size - Q}{Buffer\_Size - Q_0} & \text{, if } Q > Q_0 \\ 1 + \frac{(\alpha-1)*(Q_0 - Q)}{Q_0} & \text{, if } Q \leq Q_0 \end{cases} \tag{5.1}$$

The parameter, α, can be considered as an oversell factor of FIAC when the network in under-utilized.

Bandwidth fair control function

To calculate the available bandwidth for per-class traffic, it needs two components: one is the estimation of the incoming per-class traffic rate and another one is the feedback internal network capability rate.

We use exponential averaging formula as in Core-Stateless Fair Queuing (CSFQ) [66] to estimate the incoming per-class traffic rate. Using an exponential weight gives more reliable estimation for burst traffic, even when the packet inter-arrival time of the aggregate has significant variance.

If we indicate the arrival time of the k^{th} packet of class i as T_i^k and its length as $l_i^k(t)$, the new estimation of arriving rate $R_i(t)$ can be computed as follows:

$$R_i^{new}(t) = (1 - e^{-T_i^k/K}) * \frac{l_i^k}{T_i^k} + e^{-T_i^k/K} * R_i^{old} \tag{5.2}$$

Where T_i^k represents the k^{th} sample of the inter-arrival time of class traffic i, i.e., $T_i^k = t_i^k - t_i^{k-1}$ and k is a constant, 400 ms.

Upon receiving the feedback explicit rate (ER) from the Resource Discovery mechanism, the FIAC bandwidth fair control function will calculate the Mean Available Rate, MAR_i, for per-class traffic. This function is defined as follows:

$$MAR_i = \begin{cases} \text{if } (Q \leq Q_0) \text{ or } (Q > Q_0 \text{ and } R_i \leq ER) \\ \ ER + \beta * (R_i - ER) \\ \text{else} \\ \ ER \end{cases} \tag{5.3}$$

where β is the exponential average factor. In FIAC bandwidth fair control function the value of MAR_i is an exponential running average of the current load from all connections only when the network operates below the target operating point. When the network exceeds the target operating point, FIAC bandwidth fair control function does not allow MAR_i to increase further. That means that new MAR_i does not track the arriving rate R_i value larger than the current internal network capability rate, ER, when the queue is congested. This rule prevents all incoming per-class traffic whose R_i is equal or larger than the current ER to increase further, thereby preventing further overloading of the network and helping prevent excessive congestion.

Explicit Admission Rate Control Function

Edge router determines Explicit Admission Rate for per-class traffic (EAR_i) by considering its available buffer resource and available bandwidth resource, the product of function (5.1) and function (5.3). Explicit Admission Rate Control function is defined as follows:

$$EAR_i = f(Q) * MAR_i \tag{5.4}$$

When the network operates above the target operating point, Explicit Admission Rate (EAR) is reduced due to $f(Q) \leq 1$ (refer to function (5.1)) and the throttling is performed fairly. However, when the network operates below the target operating point, Explicit Admission Rate is allowed to increase its rate by a factor greater than 1, which enables the per-class incoming traffic capable of using the available bandwidth to take advantage of it.

5.3.1 Per-class FIAC Algorithm

Pseudo code reflecting this algorithm is described in the Algorithm (5).

where Q_0 specifies the queue length to which the queue asymptotically converges and hence $Buffer\_size - Q_0$ is the maximum burst an edge router can accommodate. We use Q_0 to separate two stages: tolerable stage and intolerable alert stage. When the queue length is larger than Q_0, we treat it as intolerable alert stage. When queue length is less than Q_0, we treat it as non-congestion stage. When an edge router is under the intolerable alert stage, it attempts to keep the packet enqueue rate around an operating

Algorithm 4 Per-class-FIAC Parameter Settings

```
Ensure: Per-class AF Service
Require: Per-class ER
    $ER$ : {Explicit Rate, from Resource Discovery Mechanism}
Require: Parameters
    $\beta$ : {The average ratio}
    $\alpha$ : {Congestion function parameter}
    $weight$ : {The traffic priority constant}
Require: Per-class Variable
    $MAR$ : {Mean Available Class Rate}
    $Q_0$ : {Target Class Queue Length}
    $Q_0^{in}$ : {Target point for IN packet queue}
    $qlen_{in}$ : {Length of IN packet queue}
    $EAR$ : {Explicit Admission Rate}
    $DPF$ : {Queue Control Factor}

    Initialization
    $Q_0 \Leftarrow$ k * BufferSize
    $Q_0^{in} \Leftarrow$ k * IN-profile BufferSize
    $MAR \Leftarrow$ {Assign a large value}
```

point by decreasing $\beta * (R/weight_i - MACR)$. When an RD router is under the tolerable stage, it attempts to keep the packet enqueue rate around an operating point by increasing $\beta * (R/weight_i - MACR)$

5.3.2 Simulation Investigation: Per-class Fair Intelligent Admission Control

We have implemented the Per-class FIAC algorithm into the ns simulator [47]. Simulations are performed on a network topology, the parking-lot topology, as illustrated in Figure 5.6.

Some traffic traverses Ingress1→B→C→Egress1, and other traffic traverses Ingress2→B→C→Egress2. The inter-router link (B→C) is the bottleneck link which is $1Mbps$. All

Algorithm 5 Per-Class FIAC Algorithm at Edge Router

```
At Edge Router
Bandwidth Fair Control Function
if RD.tag ==IN then
  if $qlen_{in} \geq Q_0^{in}$ then
    if $R/weight \leq MAR$ then
      $MAR_{new} = ER + \beta * (R/weight - ER)$
    else
      $MAR_{new} = ER$
    end if
  else
    $MAR_{new} = ER + \beta * (R/weight - ER)$
  end if
else
  if $qlen \geq Q_0$ then
    if $R/weight \leq MAR$ then
      $MAR_{new} = ER + \beta * (R/weight - ER)$
    else
      $MAR_{new} = ER$
    end if
  else
    $MAR_{new} = ER + \beta * (R/weight - ER)$
  end if
end if

Queue Control Function
if QueueLength $\geq Q_0$ then
  $DPF = (Buffer\_Size - QueueLength)/(Buffer\_Size - Q_0)$
else
  $DPF = 1 + (\alpha - 1) * (Q_0 - QueueLength)/Q_0$
end if

Per-class Explicit Admission Rate
$EAR = MAR_{new} * DPF$
```

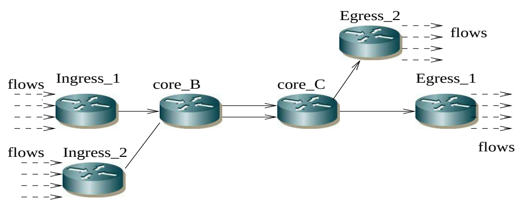


Figure 5.6: Parking-lot Experiment

other links are configured as $100Mbps$.

Experiment 1 : per-class traffic behavior under Per-class FIAC

In Experiment 1, we focus on relative service differentiation and compare the performance of the following four cases. In case 1, we consider four classes TCP traffic with different round trip time (RTT). In case 2, we study the performance of the four classes of TCP traffic with the same round trip time. In case 3, we study the performance of the four classes of UDP traffic. In case 4, we study the performance of the one class of TCP traffic and the three classes of UDP traffic. The Table 5.3.2 describes the four cases in terms of RTT, traffic type, and class policy.

For services, we define the relative QoS guarantees as Gold service (40%), Silver service (30%), Bronze service (20%), and Best Effort (10%). In experiment 1, each class has multiple flows (on Ingress1→Egress1, and Ingress2→Egress2). The results are shown in Figure 5.7.

In Case 1, refer to Figure 5.7, we plot the goodput for all classes. Based on this set of plots, we can make following observations. The DiffServ can not properly differentiate among Gold traffic, Silver traffic, Bronze traffic, and Best Effort traffic under the different round trip time situation. Even the BE traffic with low priority (10%) also gains more bandwidth due to having short round trip time. The Per-class FIAC behaves quite well after the start interval [0s, 5s]. Gold traffic gained around $0.38Mbps$, Silver traffic gained about $0.27Mbps$, Bronze traffic gained about $0.19Mbps$, and BE traffic gained about $0.1Mbps$ regardless of they have different round trip time.

Another observation is in Case 2, refer to Figure 5.8, all TCP having the same round trip time, the DiffServ and Per-class FIAC all are able to provide relative differentiated service according to the policy. A similar observation was found in Case 3, all UDP

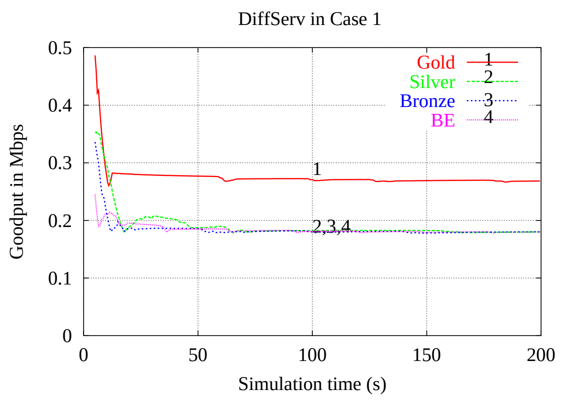
Table 5.1: Experiment 1 - Four Cases Description

Case 1	Gold	Silver	Bronze	BE
Traffic	TCP	TCP	TCP	TCP
RTT	140ms	80ms	50ms	32ms
CBR	–	–	–	–
Case 2	Gold	Silver	Bronze	BE
Traffic	TCP	TCP	TCP	TCP
RTT	50ms	50ms	50ms	50ms
CBR	-	–	–	–
Case 3	Gold	Silver	Bronze	BE
Traffic	UDP	UDP	UDP	UDP
RTT	–	–	–	–
CBR	1Mbps	1Mbps	1Mbps	1Mbps
Case 4	Gold	Silver	Bronze	BE
Traffic	TCP	UDP	UDP	UDP
RTT	50ms	–	–	–
CBR	–	1Mbps	1Mbps	1Mbps

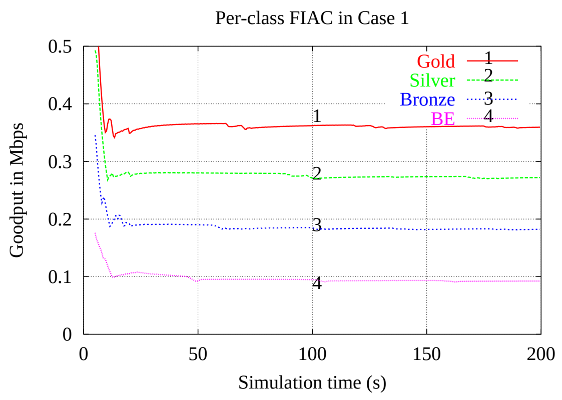
traffic, DiffServ and Per-class FIAC all provide proper differentiated service.

When the Gold traffic is TCP and others are UDP traffic, Case 4 (Figure 5.8), the DiffServ failed to provide the Gold traffic (TCP flow) the best service. In DiffServ, Gold traffic takes up around 31% bandwidth (310 Kbps) but Silver traffic takes up around 35% bandwidth (350 Kbps). Per-class FIAC differentiates traffic classes fairly by allocating 43% bandwidth (430 Kbps) to Gold traffic, 28% bandwidth (28 Kbps) to Silver traffic, and 19% bandwidth (19 Kbps) to Bronze traffic, and 10% bandwidth (10 Kbps) to BE traffic.

The average packet delay behavior is shown in Figure 5.9. In mixed traffic type case (TCP and UDP), Per-class FIAC maintains its operation around the threshold with little variations, resulting in lower packet delay compared to DiffServ.



(a) DiffServ in Case 1



(b) Per-Class FIAC in Case 1

Figure 5.7: Experiment 1

5.4 Per-flow Fair Intelligent Admission Control

The aim of Per-flow FIAC is to guarantee per-flow fairness within a class. It determines the per-flow Explicit Admission Rate (EAR_f), based on the per-flow available buffer, and per-flow available bandwidth. Per-flow FIAC also has two functions, queue control function and bandwidth control function, to calculate per-flow Explicit Admission Rate. In the following sections, we will explain the two functions in details.

Per-flow queue control function

The per-flow Queue Control Function is similar to the per-class and defined as follows.

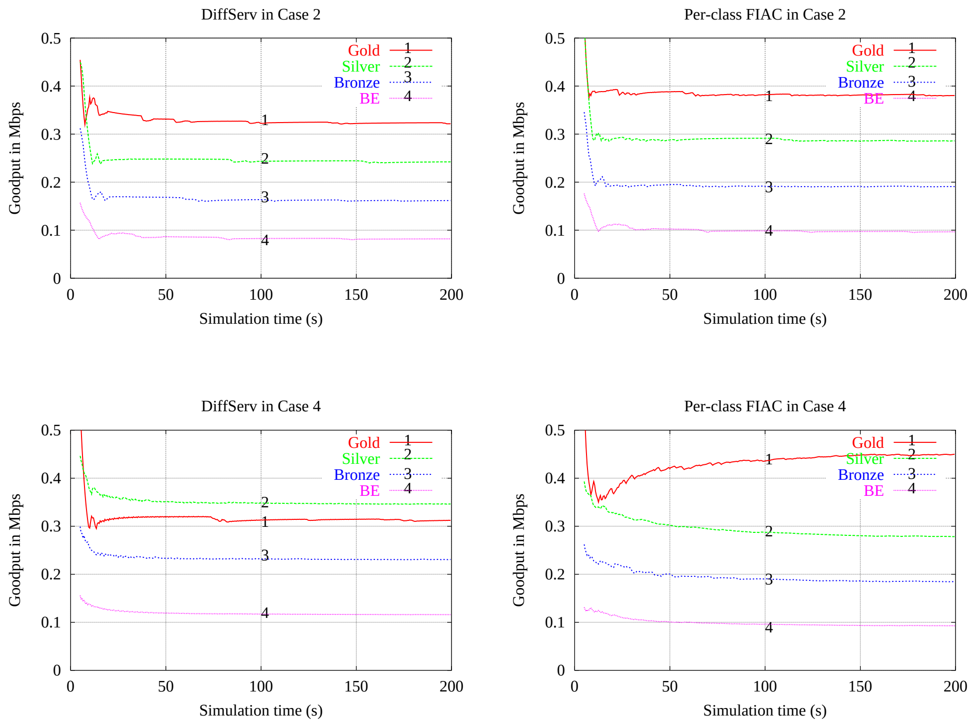


Figure 5.8: Case 2 and 4 in Experiment 1

$$f(Q_f) = \begin{cases} \frac{Buffer\_Size_f - Q_f}{Buffer\_Size_f - Q_0} & \text{if } Q_f > Q_0 \\ 1 + \frac{(\alpha-1)*(Q_0-Q_f)}{Q_0} & \text{if } Q_f \leq Q_0 \end{cases} \tag{5.5}$$

Per-flow bandwidth control function

The per-flow Bandwidth Control Function is designed to calculate the fair share of available bandwidth for per-flow. To estimate the incoming per-flow rate, we also use function (6.1) but to measure per-flow traffic. To calculate Flow Mean Available Rate (FMAR), we designed the similar function to formula (5.3).

$$FMAR = \begin{cases} \text{If} \quad Q_f \leq Q_0, \\ \text{If} \quad Q_f > Q_0 \text{ and } R \leq FMAR_{old} \\ \quad FMAR_{old} + \beta(R - FMAR_{old}) \\ \text{Otherwise} \\ \quad FMAR_{old} \end{cases} \tag{5.6}$$

Where the value of $FMAR$ is a running average of the current load from all flows

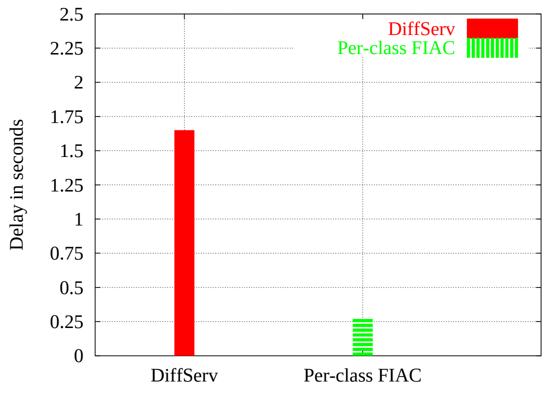


Figure 5.9: Average Packet Delay: 1 TCP flow and 3 UDP flows

for the same class and R is the flow arriving rate. When the queue length, Q_f, exceeds the target operating point (Q_0), the per-flow FIAC Bandwidth Control Function does not allow $FMAR$ to increase further. This mechanism aims to allocate available bandwidth fairly among all flows within a class.

Per-flow explicit admission rate

The per-flow FIAC determines the Per-flow Explicit Admission Rate ($FEAR$) by calculating the product of per-flow Queue Control Function $f(Q_f)$ and per-flow Bandwidth Control Function $FMAR$. The per-flow Explicit Admission Rate Function is defined as follows.

$$EAR_f^i = f(Q_f) * FMAR \tag{5.7}$$

It is not surprised that the per-flow and per-class admission controls are similar in structure. They both try to estimates bandwidth fair share for a class or flow taking into account network conditions, buffer available, and bandwidth usage in their decisions to allocate bandwidth accordingly.

5.4.1 Per-flow Fair Intelligent Admission Control Algorithm

After receiving the QoS report (Explicit Rate, ER) from the Resource Discovery Mechanism, FIAC applies FIAC algorithm to calculate per-class Explicit Admission Rate and

```
Algorithm 6 Per-flow-FIAC Parameter Settings
Ensure: $EAR^i_f$: per-flow Explicit Admission Rate within the class i
Require: $EAR^i$: Explicit Admission Rate of class i
Require: Parameters
    $\beta$ : {The average ratio}
    $\alpha$ : {Congestion function parameter}
Require: Per-flow Variable
    $FMAR$ : {Flow Mean Available Rate}
    $Q_0$ : {Target Class Queue Length}
    $EAR^i_f$ : {Explicit Admission Rate flow of class i}
    $DPF$ : {Queue Control Factor}
    $R^i_f$ : {the incoming rate of flow which belongs to class i}

    Initialization
    $Q_0 \Leftarrow$ k * BufferSize
    $FMAR \Leftarrow EAR^i$ from per-class FIAC
```

per-flow Explicit Admission Rate. The Per-flow FIAC algorithm is described in Algorithm 7.

5.4.2 Simulation Investigation: Per-flow Fair Intelligent Admission Control

We use the parking-lot network topology, as illustrated in Figure 5.10, to investigate the per-flow fair intelligent admission control.

Figure 5.11 shows the fairness among flows within a class. It is clearly shown that DiffServ is not concerned about fairness among flows within the same class.

Figure 5.12 shows the fairness among flows within a class. It is clearly shown that conventional DiffServ is not concerned about fairness among flows within the same class (Figure 5.12(a) and Figure 5.12(c)).

The situation is not much different when the per-flow FIAC is disabled. As discussed before, the unfairness between flows is caused by factors outside DiffServ domain and hence does not fall within DiffServ's handling capability. It is clear from Figure 5.12(b)

Algorithm 7 Per-flow FIAC Algorithm at Edge

```
At Edge Router

Bandwidth Fair Control Function
if QueueLength $\geq Q_0$ then
  if $R_f^i \leq FMAR$ then
    $FMAR_{new} = FMAR + \beta * (R_f^i - FMAR)$
  else
    $FMAR_{new} = FMAR$
  end if
else
  $FMAR_{new} = FMAR + \beta * (R_f^i - FMAR)$
end if

Queue Control Function
if QueueLength $\geq Q_0$ then
  $DPF = (Buffer\_Size - QueueLength)/(Buffer\_Size - Q_0)$
else
  $DPF = 1 + (\alpha - 1) * (Q_0 - QueueLength)/Q_0$
end if

Per-flow Explicit Admission Rate
$EAR_f^i = FMAR_{new} * DPF$
```

and Figure 5.12(d) that FIAC handles fairness among DiffServ classes when its per-class level is enabled, but it does not affect the composition of the flows within a class. Unfairness among flows is caused by their own flow control schemes.

The next experiment demonstrates the capability of per-flow level FIAC.

Experiment 2: Per-flow FIAC and Per-class FIAC both enabled vs. DiffServ

Experiment 2 was designed to study per-flow fairness among flows within a class, the Gold service. We have designed three cases: Case 1 is used to study per-flow fairness among 4 TCP flows which have different round trip time; Case 2 is designed to study per-flow fairness among 4 TCP flows which have the same round trip time; Case 3 is

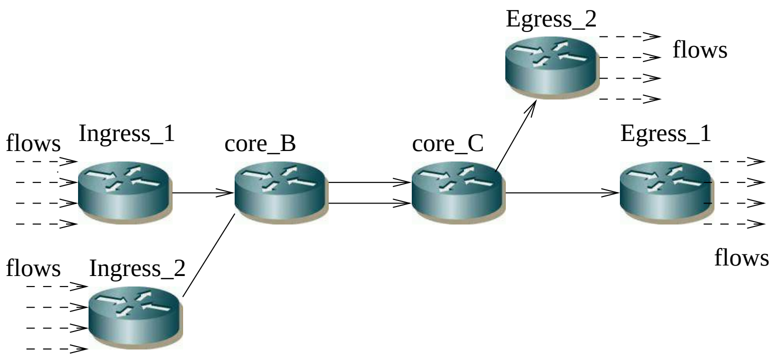


Figure 5.10: Parking-lot Experiment

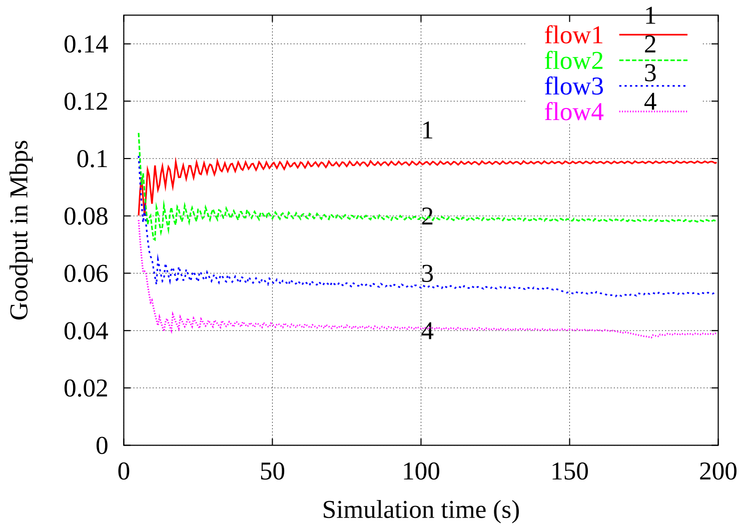
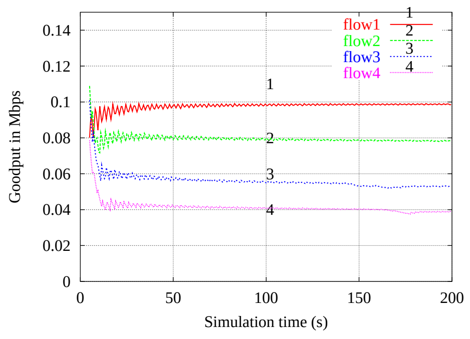


Figure 5.11: Per-flow fairness: in Gold class

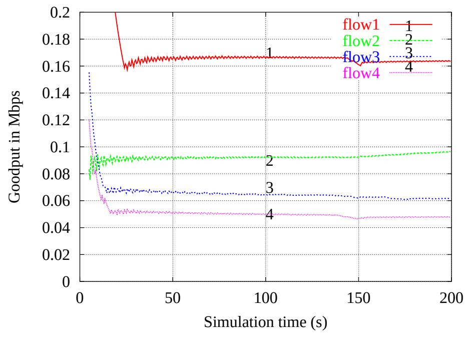
used to study per-flow fairness among 1 TCP flow and 3 UDP flows. The Experiment 2 configuration is described in Table 5.2.

In Experiment 2, Gold traffic (which has 4 flows) traverses from Ingress1→Egress1 and others (Silver, Bronze, and BE) traverse Ingress2→Egress2 which are all TCP traffic (with RTT 50ms).

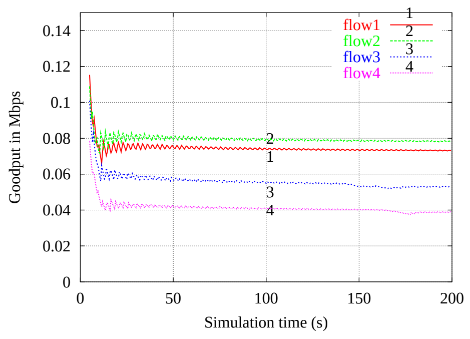
DiffServ and FIAC enabled results are shown in Figure 5.13. Case 1 demonstrates that the normal DiffServ is not fair for the flows which have different round trip time. The fair share for each flow is around 100 Kbps of the available bandwidth. The normal DiffServ allocates the flow4 (shortest RTT) 180Kbps bandwidth and allocates the flow1 (longest RTT) 80 Kbps bandwidth. The FIAC (Per-class and Per-flow enabled) allocates each flow 93 Kbps bandwidth compared to the ideal fair share bandwidth, 100 Kbps.



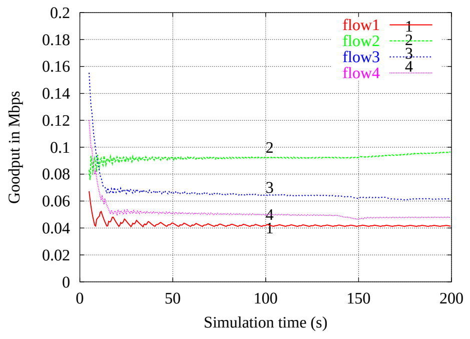
(a) Per-flow fairness in DiffServ Gold service



(b) Per-flow fairness in Per-class FIAC Gold service



(c) Per-flow fairness in DiffServ BE



(d) Per-flow fairness in Per-class FIAC BE

Figure 5.12: Per-flow fairness in Experiment 1: only Per-class FIAC without per-flow FIAC

The normal DiffServ and FIAC show the similar behavior when the all flows have the same round trip time. They all guarantee per-flow fairness.

In Case 3, the normal DiffServ allocates less bandwidth to TCP flow (only 8 Kbps) compared to other UDP flow (105 Kbps) despite they have same priority (they are in same class). FIAC (per-flow and per-class enabled) demonstrates it can allocate bandwidth fairly (95 Kbps) among the flows which are in same class.

Table 5.2: Experiment 2 - Three Cases Description

Case 1	flow1	flow2	flow3	flow4
Traffic	TCP	TCP	TCP	TCP
RTT	230ms	140ms	50ms	32ms
CBR	–	–	–	–
Case 2	**flow1**	**flow2**	**flow3**	**flow4**
Traffic	TCP	TCP	TCP	TCP
RTT	50ms	50ms	50ms	50ms
CBR	–	–	–	–
Case 3	**flow1**	**flow2**	**flow3**	**flow4**
Traffic	TCP	UDP	UDP	UDP
RTT	50ms	–	–	–
CBR	–	1Mbps	1Mbps	1Mbps

5.5 Fair Intelligent Admission Control with Variable Packet Size

Differentiated Service Network (DiffServ) provides an architecture that is scalable and capable of differentiating applications' quality of service (QoS). However, if the flows use packets of different sizes, resource sharing is no longer fair. One problem with the basic DiffServ algorithm, such as RIO [16], is that it treats all packets the same regardless of their size. As a result, flows with larger packet sizes get more bandwidth than flows with smaller packet sizes, and hence basic DiffServ cannot ensure fairness among classes with variable-length packets.

In this section we study the impact of variations in packet size on DiffServ and apply FIAC mechanism to remove the throughput bias resulting from the use of different packet sizes. We suggest a solution that involves Fair Intelligent Admission Control (FIAC) scheme for each DiffServ domain. We evaluate these designs through simulation and conclude with some concrete findings that FIAC scheme is robust enough independent of packet size and also can be used to improve fairness among flows within the class.

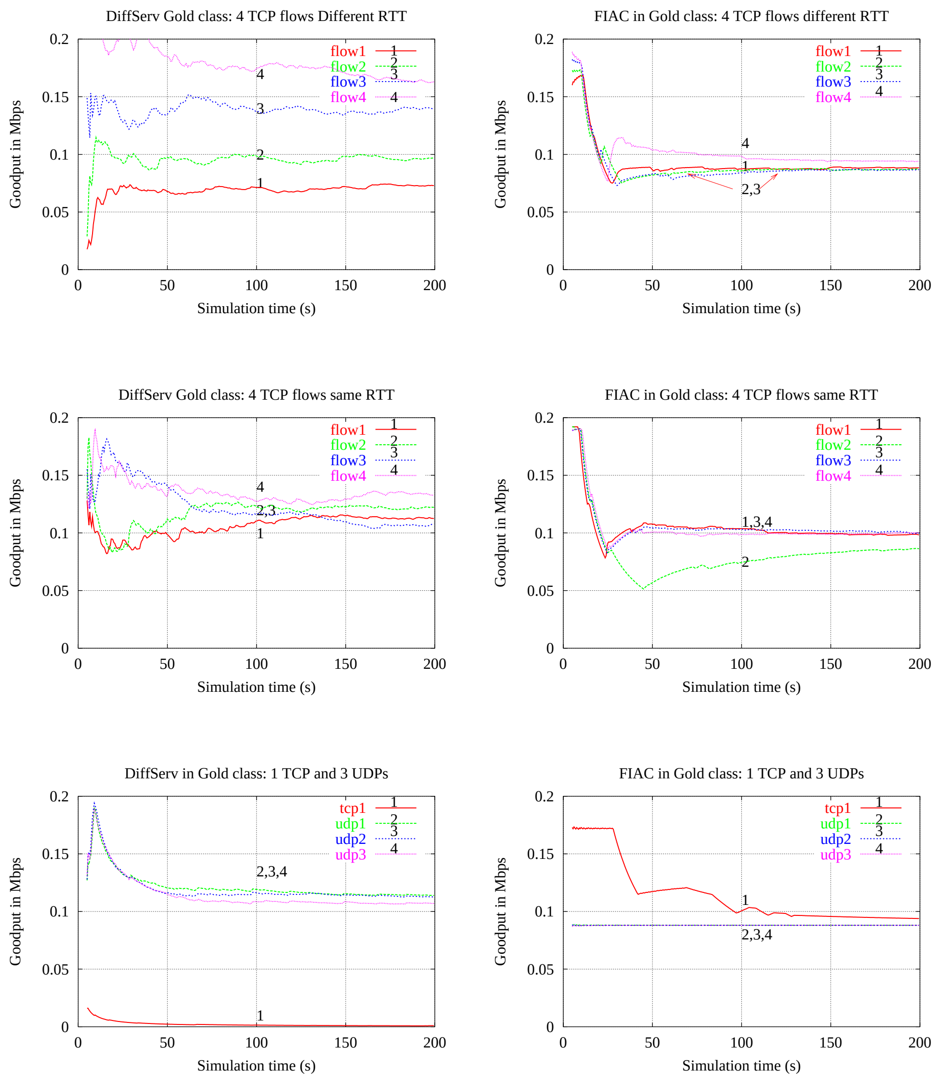


Figure 5.13: Per-flow fairness in Experiment 2: Per-class FIAC and Per-flow FIAC both enabled vs. DiffServ

5.5.1 Virtual Class-unit

The FIAC mechanism introduces the class-unit concept to represent basic weight unit to measure. The idea behind virtual class-unit is the fact that individual class traffic with different weight (or priority) can be subdivided into smaller unit, which can be manipulated effectively. The smaller the class-unit is, the more sensitive the RD mechanism responses to the network fairness state variation. So if the class traffic with large packet size, it would take more bandwidth under normal DiffServ mechanism. But with RD-FIAC-DiffServ, RD mechanisms feedback the fairness state to FIAC module dynamically, so

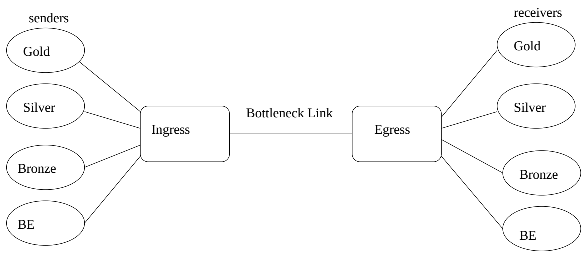


Figure 5.14: Simulation Topology

that the FIAC module will take admission control to achieve fairness for classes dynamically. In other words, FIAC will limit the class traffic with large packet size to maintain the fairness state. In our experiment, we choose 10% of the total bandwidth as a class-unit.

5.5.2 Evaluation: FIAC over DiffServ for Variable Packet Size

Evaluation Setting

In order to evaluate our solutions, we implemented them in the ns-2 network simulator [47] and measured their performance under various conditions. This allows us to check the feasibility and investigate the behavior under more realistic settings.

For the simulation topology, we use the well-known dumbbell topology, with senders and receivers on either side of a single bottleneck link, as depicted in Figure 5.14. The access links to the routers are provisioned with 100 Mbps, while the bottleneck link between the routers has a lower capacity so that the total bandwidth consumed by all flows is well below 100 Mbps, we configured as 1 Mbps. When discussing the scenarios, we will denote TCP flows with a different packet size. The standard ns-2 implementation of TCP Reno is used. We set up four classes: Gold, Silver, Bronze, and Best Effort.

In Experiment 1, we evaluate TCP traffic (in Gold, Silver, Bronze, and Best effort classes) performance with variable packet sizes in terms of goodput.

In Experiment 2, we study performance of TCP flows in the same class with variable packet sizes in terms of throughput.

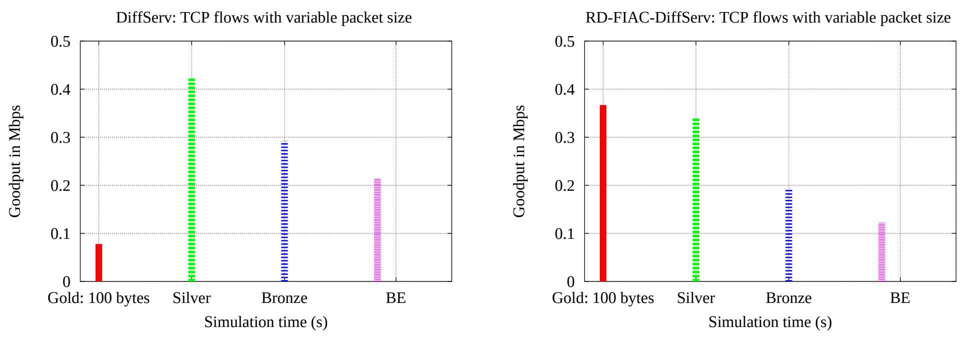


Figure 5.15: Goodput of Classes (Gold flow with short packet size)

Experiment 1: TCP traffic with variable packet sizes in different classes

In Experiment 1, we consider four classes, which are Gold, Silver, Bronze, and Best effort classes. They claim 40%, 30%, 20%, and 10% respectively. Each class has four TCP flows with same round trip time (RTT). In Gold class, TCP flow has short packet size, 100 bytes. In other classes, TCP flows have large packet size, 1024 bytes.

In Figure 5.15, we plot the goodput results for all classes under FIAC-DiffServ.

Based on this set of plots, we can make following observation. The DiffServ can not allocate bandwidth fairly for the classes which have variable packet sizes. The DiffServ allocates the Gold class, (short packet size) less bandwidth, around 90 Kbps, which is lower than other classes. A TCP flow that chooses a large packet size, can have an advantage over other TCP flows. Our proposed model allocates each class quite fairly according to their service level agreement.

Experiment 2: TCP flows in same class with variable packet sizes

Experiment 2 was designed to study the performance of TCP flows within the same class with variable packet sizes. In the simulation, one TCP flow whose packet size is 1000 bytes and other TCP traffic flows have same packet size, 100 bytes.

We investigate the simulation results in terms of bandwidth. The result is shown in Figure 5.16. From the results, the TCP flows with short packet size suffer acute degradation compared to TCP flow with large packet size, 1000 bytes.

However, in Figure 5.16(b), the distribution of bandwidth is more fair than traditional DiffServ. Our proposed model is to preferentially drop the packets from the TCP flow

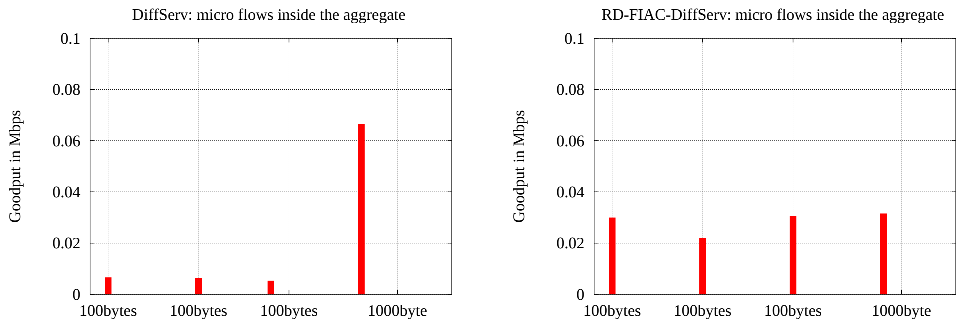


Figure 5.16: TCP flows within the class: one flow with large packet size

which has large packet size and its incoming rate is beyond the fair share rate.

5.6 Conclusion

In this section we described the Fair Intelligent Admission Control mechanisms in which the admission control is used to control the incoming traffic between the edges based on “internal” network QoS states and “external” traffic requirements, Service Level Agreements. We then introduced two Fair Intelligent Admission Control mechanisms: per-class FIAC and per-flow FIAC.

At per-class level, FIAC admits traffic classes according their fair share and usage while preventing possible congestion within the DiffServ core.

At per-flow level, FIAC estimates and allocates the fair share for each flow relying only on the interaction with its per-class admission control.

Last we demonstrated that it is also applicable for variable packet size traffic, we can cause the admission control mechanism to adjust its control according to dynamic QoS states report from Resource Discovery module so that to achieve fairness objective.

We recognize that our work only touches the surface. Future work will have to address outstanding issues such as performing a complete sensitivity analysis of all of the parameters in the system, better studying the optimal target point of the scheme, and extending the network domain to larger and multiple domains with multiple admission control mechanisms involved.

Together our advances lay the ground work for developing new, scalable, practical control plane architectures to achieve the Internet QoS.

Chapter 6

Edge-Aware Resource Discovery and Fair Intelligent Admission Control scheme

6.1 Introduction

Basically, Edge-Aware Resource Discovery Mechanism is used to generate Resource Discovery (RD) packets for collecting QoS states of the network domain. It involves the ingress node and egress node only. The RD packets are generated and marked (e.g. EF class) at the ingress node. Upon receiving a RD packet, the egress node consults its QoS states and estimates "internal" domain QoS state (congestion state) and send back to the ingress node.

The philosophy behind the "Edge-Aware" feature is that the egress router can feel the "pain" of internal congestion without knowing where the accurate congestion node is. It is based on the following observations. First, the egress router is the only exit gateway to other domains. Second, the traffic rate at egress router can be seen dramatically decreased if the congestion occurring at the internal core router. Therefore, the egress router is able to feel the congestion state when it found the traffic rate dramatically low compared to its flushing rate at the ingress router. The edge routers learn the network QoS state from indirect information, such as the traffic flushing rate variation, the traffic arriving rate variation, and estimate the network congestion intelligently by using the Edge-Aware

Resource Discovery algorithms.

We propose and advocate the notion of Edge-Aware Resource Discovery and Fair Intelligent Admission Control as an effective means for improving QoS without involving core router mechanisms. The Edge-Aware Resource Discovery architecture relies on well-defined Edge-Aware bandwidth control algorithm. The Edge-Aware Fair Intelligent Admission Control mechanism makes admission or rejection decision based on the network QoS states reported from the Edge-Aware Resource Discovery mechanism and the incoming traffic rate.

In addition to its ability to deliver edge-to-edge QoS states, the Edge-Aware Resource Discovery and admission control architecture has a number of other important advantages.

- It decouples control plane from data plane, thereby reducing the complexity of network management and control. The edge routers of the network domain are now concerned primarily with associated edge-aware resource discovery and edge-aware admission control mechanism on a much coarser granularity.

- Because of its edge awareness, Edge-Aware Resource Discovery and admission control (Edge-Aware RD-FIAC) can deploy resource management and QoS control mechanisms just at edge routers. Hence the Edge-Aware RD-FIAC architecture not only simplifies the network QoS management, but also makes it more scalable.

We study the Edge-Aware RD-FIAC mechanisms in terms of various factors on bandwidth variation, queue length variation, Service Level Agreement, and variation between the flushing rate at the Ingress and the arriving rate at the Egress router. In Section 6.2 we outline the Edge-Aware Resource Discovery architecture. In Section 6.3 , we present the Edge-Aware Fair Intelligent Admission Control mechanism. We also demonstrated the simulation results in the Section 6.4. We summarize the Edge-Aware Resource Discovery and Edge-Aware Fair Intelligent Admission Control schemes in the Section 6.5.

6.2 Edge-Aware Resource Discovery Scheme

The Figure 6.1 illustrates the Edge-Aware Resource Discovery structure and relationship with other DiffServ components.

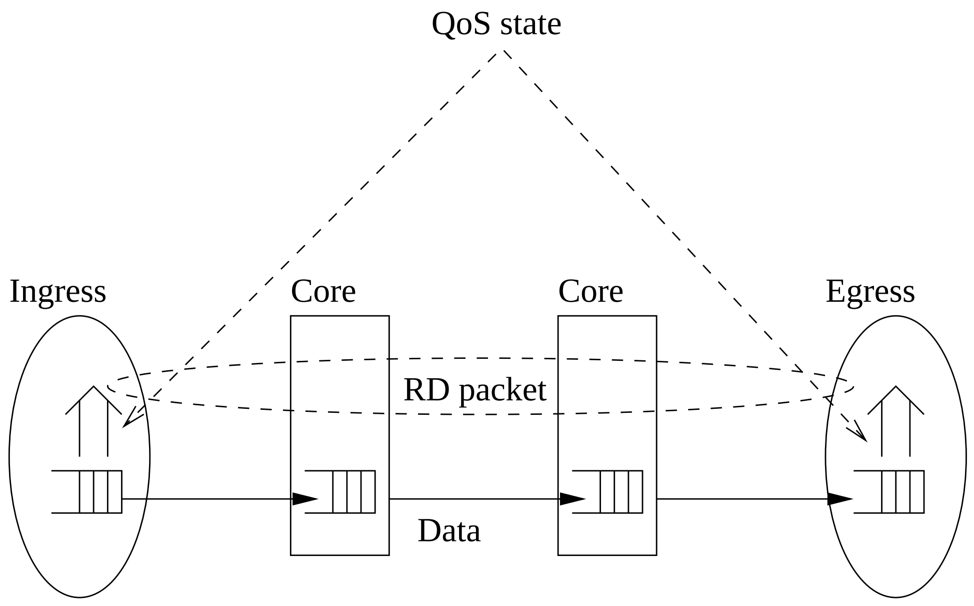


Figure 6.1: Edge-Aware Resource Discovery scheme

The Edge-Aware Resource Discovery mechanism is a closed-loop feedback mechanism working between the ingress router and the egress router.

On the forward path, the ingress nodes perform three functions. They (a) initiate the RD packet, mark the DSCP (e.g. EF class) in the DSCP-field of RD packet, (b) generate RD packet in a constant rate (or proportional to the incoming traffic rate). and (c) calculate the ingress explicit rate which is the flushing rate into the network domain.

On the backward path, egress routers terminate the RD packet and send them back to ingress node with updated network QoS state information which is measured at the egress node. The egress nodes perform three functions to measure the QoS states of the domain. They (a) measure the available bandwidth state of the egress router, (b) measure the available buffer state of the egress router, and (c) compare with the ingress explicit rate and calculate the QoS states of the domain.

As a result, the Edge-Aware Resource Discovery mechanism can provide “inside” network QoS states dynamically.

To achieve the above the functions, we designed the three components to measure the available bandwidth, available buffer, and QoS states calculation.

Figure 6.1 illustrates the Edge-Aware Resource Discovery architecture. The Edge-Aware Resource Discovery mechanism is pieced together via the edge routers which per-

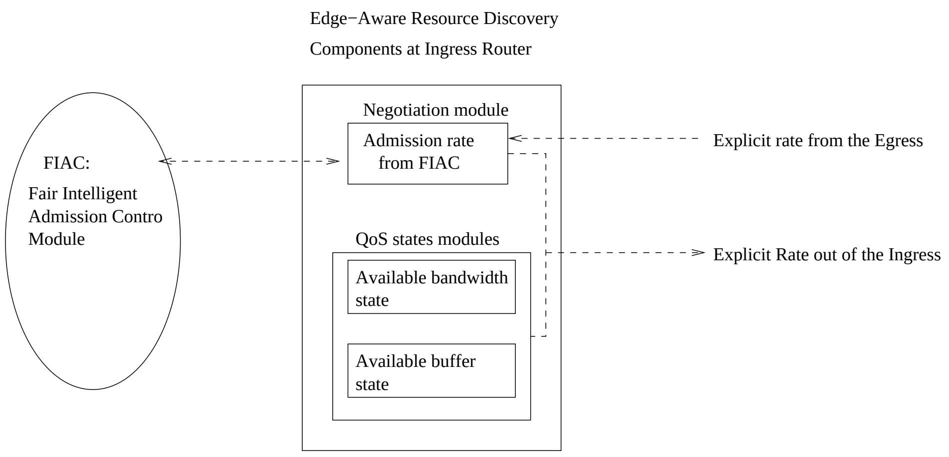


Figure 6.2: Edge-Aware Resource Discovery Components at Ingress Router

form RD generation, data forwarding and QoS states measurement functions. The logical connection of RD feedback loop between the ingress and egress routers is provided by the underlying network domain.

At the egress router, the Edge-Aware Resource Discovery mechanism measures the available bandwidth variation, the available queue length variation, and calculates the QoS states by comparing with the admitted rate.

6.2.1 The Ingress Router Algorithm

The Edge-Aware Resource Discovery mechanism at the Ingress router has three components:

- **Negotiation module:** It's used to negotiate with admission control module. When the ingress router generates the RD packet, it gets the admission rate from admission control module. When the ingress router receives the backward RD packets, it reports the explicit rate to the admission control module to update the latest QoS states of the domain.

- **The incoming traffic rate measurement module:** It's used to estimate the incoming traffic rate by using the equation 6.1.

- **The flushing rate measurement module:** It's used to calculate the flushing rate out of the ingress router.

The Edge-Aware Resource Discovery mechanism at the Ingress focuses on the flushing rate measurement rather than the local QoS state. It negotiates with the admission control module. When the RD packet being generated, the Edge-Aware Resource Discovery mechanism gets the admission rate from the admission module, estimates the traffic incoming rate, and chooses the smaller one as the flushing rate to put into the field of RD packet which destines to the egress router. When receiving the backward RD packet, the Edge-Aware Resource Discovery mechanism at the ingress router extracts the network QoS state (calculated at the egress router) from the RD packet and informs to the admission control module. By this way, the admission control module keeps the latest the QoS states of the domain.

The Edge-Aware Resource Discovery algorithm at the Ingress router is described in the Algorithm (9).

Upon arrival of a packet, the Edge-Aware Resource Discovery at ingress still use 6.1 equation to estimate the incoming traffic rate.

$$r_i^{new} = (1 - e^{-T_i^k/K})\frac{l_i^k}{T_i^k} + e^{-T_i^k/K} r_i^{old} \tag{6.1}$$

where T is the packet interarrival time, K is a constant (that is usually set to a large value), and l is the current packet length.

In Edge-Aware Resource Discovery algorithm, the optimal point, $Q0$, is set to $1/3$ of the buffer size which specifies the queue length to which the queue asymptotically converges, and hence $Buffer_{size} - Q0$ is the maximum burst the router can accommodate. When the instantaneous queue length is larger than the $Q0$, it attempts to slow down the packet enqueue rate.

The pseudo code, Algorithm(9), that highlights the modification made in Chapter 4 to give the Edge-Aware Resource Discovery algorithm at the ingress router. The basic modification behind the Edge-Aware RD algorithms at ingress is to measure the exact flushing rate out of the ingress router. The flushing rate is obtained by the following four distinct steps:

1. **The ingress router estimates the incoming traffic rate by using the equation 6.1**.

2. **The ingress router calculates the local QoS states in terms of the available bandwidth and the available buffer**.

	RD at ingress	**Edge-Aware RD at ingress**
Outcome	flushing rate	local QoS state
RD packet	Generate RD packet	Generate RD packet
the incoming traffic	Need to measure	need to measure
the queue state	Need to measure	not necessary
the available bandwidth	Need to measure	not necessary
the admission rate	Not necessary	Obtained from FIAC

Table 6.1: Difference between the normal RD algorithm and the Edge-Aware RD algorithm at ingress

3. **The ingress router obtains the admission rate from the Fair Intelligent Admission Control module**.

4. **At the end, the ingress router chooses the smaller one as the traffic flushing rate for the network**.

The implication of the Algorithm 9 is two fold: first, the Edge-Aware RD algorithm at ingress is designed to measure the traffic flushing rate rather than the local QoS state and which will be carried by RD packet for the egress router. Second, the incoming traffic rate might not be the flushing rate if it is larger than the admission rate. The decision for choosing the flushing rate depends on the comparison between the admission rate and the traffic incoming rate. Table 6.2.1 summarizes the difference between the Edge-Aware RD algorithm at ingress and the normal RD algorithm at ingress. Table 1.2.1.

6.2.2 The Egress Router Algorithm

The Edge-Aware Resource Discovery algorithm at egress (Edge-Aware RD at egress) differs from the Resource Discovery algorithm at egress (RD at egress) in the following aspects.

1. Unlike the normal RD at egress, which needs to consult all QoS states of the local routers, the Edge-Aware RD at egress is not tied to the all the local QoS states. Edge-Aware RD at egress uses the indirect information to detect the network QoS

Algorithm 8 Edge-Aware Resource Discovery Algorithm Parameter Settings

```
Ensure: $R_{ingress}$: The flushing rate at ingress
Ensure: $ER$: The local Explicit Rate
Require: The incoming traffic rate $R_{in}$
Require: The admitted rate from admission control module $R_{admit}$
    $\beta$ : {The average ratio}
    $\alpha$ : {Congestion function parameter}
Require: Per-class Variable
    $MACR$ : {Mean Allowed Class Rate}
    $Q_0$ : {Target Class Queue Length}
    $Q0_{in}$ : {Target point for IN packet queue}
    $qlen_{in}$ : {Length of IN packet queue}
    $qlen$ : {Packet queue length}
    $ER^i_{ingress}$ : {ingress Explicit Rate}
    $DPF$ : {Queue Control Factor}
    $w_i$ : {The ith class weight}
    $Q_0 \Leftarrow$ 0.3 * BufferSize
    $Q0_{in} \Leftarrow$ 0.3 * IN BufferSize
    $MACR \Leftarrow$ 100Mb {Assign a large value}
    $\beta \Leftarrow 0.08$
    $\alpha \Leftarrow 1.23$
```

Algorithm 9 Edge-Aware Resource Discovery Algorithm (for DiffServ) at Ingress

```
Upon receiving packet p
Estimate the incoming traffic rate
$R_{in} = estimate(p)$
Bandwidth Fair Control Function
* The parts are the same as in Algorithm 2, Chapter 4
Queue Control Function
* The parts are the same as in Algorithm 2, Chapter 4
The ith class ingress Explicit Rate
* The parts are the same as in Algorithm 2, Chapter 4
if $R_{admit} \geq R_{in}$ then
    $R_{ingress} = R_{in}$
end if
if $R_{ingress} \geq ER$ then
    $R_{ingress} = ER$
end if
Put the $R_{ingress}$ into the RD field
```

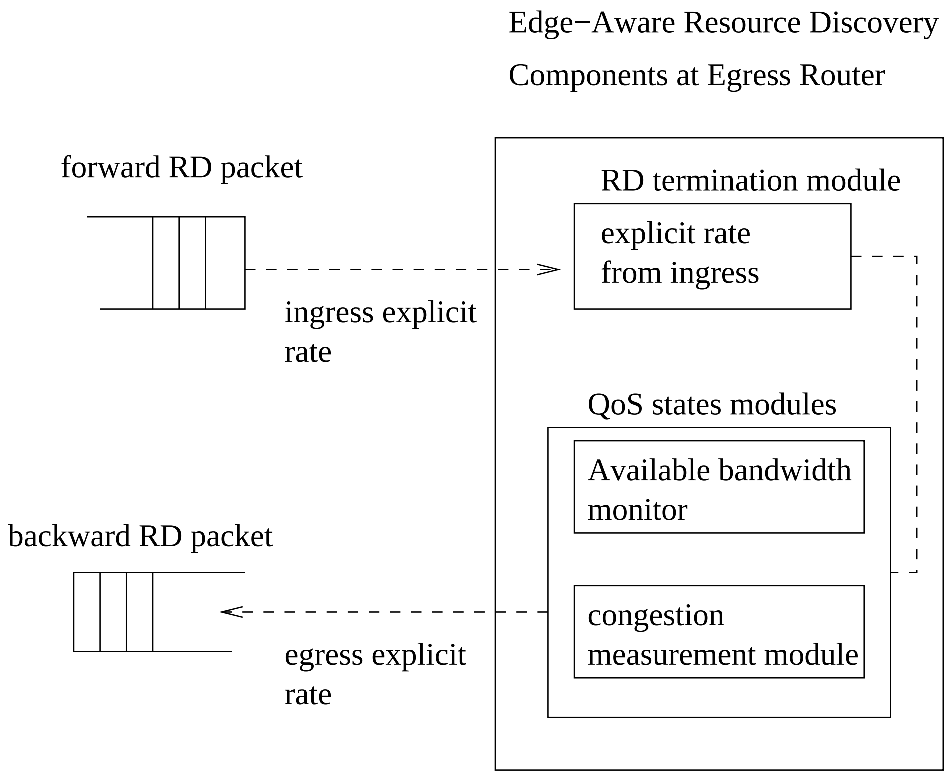

Edge−Aware Resource Discovery
Components at Egress Router
forward RD packet
RD termination module
explicit rate from ingress
ingress explicit rate
QoS states modules
Available bandwidth monitor
backward RD packet
congestion measurement module
egress explicit rate

Figure 6.3: Edge-Aware Resource Discovery Components at Egress Router

states. On the other hand, the normal RD at egress uses the direct and complete local router state information to obtain the network QoS states.

2. Information carried by RD packet is different: for the Edge-Aware Resource Discovery, the RD packet carries the flushing rate out of the ingress. On the contrary, the RD packets in the normal Resource Discovery mechanism collect the local router QoS states along the path.

3. QoS states Measurement: The Edge-Aware RD at egress measures the network QoS states by using the difference between the flushing rate out of the ingress and the arriving rate at the egress to detect the network QoS states. On the contrary, the normal RD at egress only needs to consult the RD packet which carries the QoS state along the path to update the latest network QoS states.

The Edge-Aware Resource Discovery mechanism at Egress router has three components:

- **RD termination module:** It's used to terminate the RD packet and extract the ingress explicit rate from RD packet.
- **Arriving traffic rate monitor:** It's used to calculate the arriving traffic rate at the egress router.
- **Internal congestion measurement:** It's used to calculate the internal congestion and calculate the egress explicit rate.

Once the egress router receives the RD packet, the Edge-Aware Resource Discovery starts to calculate the network QoS states. The calculation is composed of four distinct steps:

1. **Step 1:** the Edge-Aware Resource Discovery mechanism terminates the RD packet, extract the flushing rate from the RD packet.
2. **Step 2:** the Edge-Aware Resource Discovery mechanism measures the packet arriving rate at egress.
3. **Step 3:** calculates the network QoS states by comparing the difference between the flushing rate out of the ingress router and the arriving rate at egress router.
4. **Step 4:** puts the Network QoS states into the backward RD packet which destines to the ingress router.

The Edge-Aware Resource Discovery algorithm at the Egress router is described in the Algorithm (10).

The Edge-Aware Resource Discovery at the egress router adds the functionality of estimating the network QoS states with indirect information: the difference between the flushing rate out of the ingress router and the arriving rate at the egress router. The subsequent paragraphs discuss the new features of the Edge-Aware Resource Discovery in terms of parameter settings:

1. We illustrate the significance of the *difference* between the flushing rate out of the ingress and the arriving rate at the egress router.

Algorithm 10 Edge-Aware Resource Discovery Algorithm (for DiffServ) at Egress

```
Ensure: $ER^i_{egress}$: The ith class Explicit Rate of Egress
Require: The ith class traffic admission explicit rate at Ingress $ER^i_{ingress}$
    $\alpha$ : {Congestion function parameter}
Require: Per-class Variable
    $ER^i_{egress}$ : {Explicit Rate of Egress}
    $R^i$ : {The ith class traffic arriving rate at Egress}
    $weight_i$ : {The ith class weight}
    $\alpha \Leftarrow 1.2$ {congestion parameter}
    Bandwidth Fair Control Function
    if $R^i \leq 1/\alpha * ER^i_{ingress}$ then
        $ER^i_{egress} = ER^i_{ingress} + (\alpha * R^i - ER^i_{ingress})$
    else
        $ER^i_{egress} = ER_{ingress} + (\alpha * R^i - ER^i_{ingress})$
    end if
```

2. The distinction between the Resource Discovery and Edge-Aware Resource Discovery algorithm is that the interior routers have involvement or not. The Edge-Aware Resource Discovery algorithm estimates the network QoS states without the interior routers QoS states information.

3. As the class traffic rate at egress router can be seen dramatically decreased if the congestion occurring at the internal core router. This simple observation has important consequences on the design of the Edge-Aware Resource Discovery algorithms: First, the parameters needed for computation of the network QoS state (ER) are the flushing rates at the ingress router ($ER_{ingress}$), the arriving rate at the egress router (R), and the congestion function parameter (α). The congestion function parameter, α, represents the congestion threshold. We assume that there is no congestion when holding the condition, $R_{egress} \geq \frac{1}{\alpha} * ER_{ingress}$. We assume that the congestion happened when $R_{egress} \leq \frac{1}{\alpha} * ER_{ingress}$ is true. The second consequence of our observation is that recent occurrences should not have a stronger influence on the estimation of the network QoS state, ER_{egress}, than occurrences further in the past, and not for instance as an exponentially weighted giving recent occurrences more

	Normal RD	**Edge-Aware RD**
RD packet	collects the en route QoS states	carries the flushing rate out of Ingress
measurement	need to measure local QoS state	measure the arriving rate
information	direct and complete	indirect and incomplete. Only has knowledge of flushing rate out of Ingress and arriving rate at Egress
core router involvement	Yes	No need.

Table 6.2: Differences at Egress: normal RD vs. Edge-Aware RD

weight.

4. There are two considerations in Algorithm (10). First, Edge-Aware Resource Discovery mechanism tries to increase the egress explicit rate when there is no congestion. Second, Edge-Aware Resource Discovery mechanism tries to decrease the egress explicit rate when congestion happened.

We summarize the difference between the normal Resource Discovery algorithm at egress and the Edge-Aware Resource Discovery algorithm at egress in Table 6.2.2.

6.3 Edge-Aware Fair Intelligent Admission Control

After receiving QoS report from Edge-Aware Resource Discovery mechanism via the negotiation module at Ingress router, the Admission Control Module compares the incoming class traffic rate and applies the fair intelligent admission control algorithm to make admission decision. The Edge-Aware FIAC components are illustrated in Figure 6.4.

In this model, the Edge-Aware FIAC has three components:

- **Traffic rate estimator:** It's used to estimate the incoming traffic rate from "external".

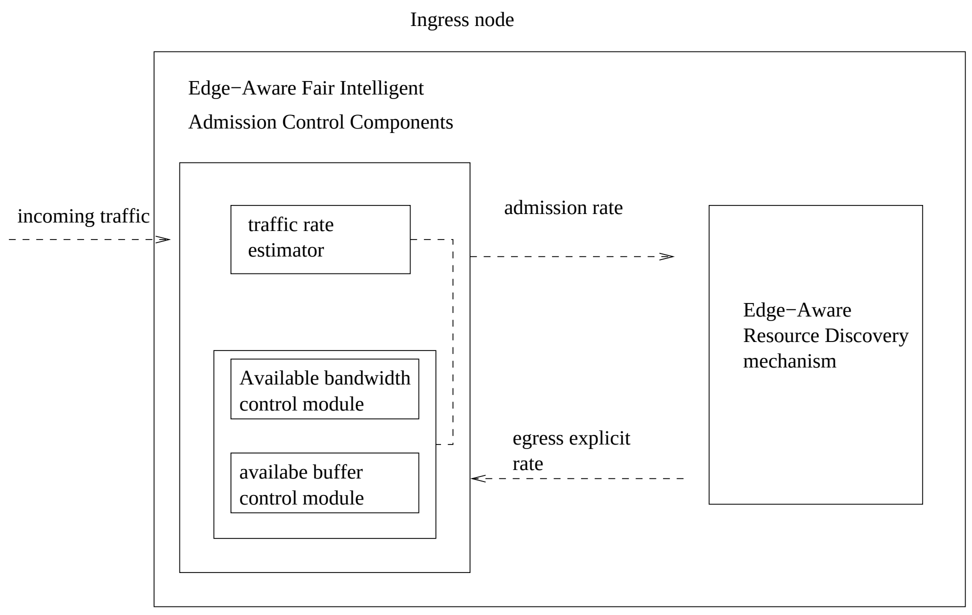


Figure 6.4: Edge-Aware Fair Intelligent Admission Control Components at Ingress

- **Available bandwidth control module:** It's used to measure the available bandwidth variation state.
- **Available buffer control module:** It's used to measure the available buffer state.

The traffic rate estimator module interacts with "external" to estimate the incoming traffic rate. The bandwidth control function and buffer control function take into account the available bandwidth variation, available buffer variation, and "internal" network capability (from Edge-Aware Resource Discovery) to calculate the explicit admission rate. The Edge-Aware FIAC makes negotiation with "external" states (incoming traffic rate, and service level agreement) and "internal" states (the egress explicit rate from Edge-Aware Resource Discovery mechanism). This leads to a optimum target point for both "external" requirement and "internal" capability.

The philosophy behind Edge-Aware FIAC is to prevent overloading traffic as early as possible (at edge node) so that it will alleviate network congestion situation and guarantee per-class level of service.

The Edge-Aware FIAC determines Explicit Admission Rate (EAR) based on the domain capability rate, Explicit Rate (reported by Edge-Aware Resource Discovery Mechanism), and the incoming ith class traffic rate, R^i. The Edge-Aware Fair Intelligent Admission Control algorithm is described as in Algorithm (12).

Algorithm 11 Edge-Aware Fair Intelligent Admission Control Algorithm Parameter Settings

```
**Ensure:** $EAR^i$: The ith class Explicit Admission Rate at Ingress
**Require:** The incoming traffic rate $R$
**Require:** The admitted rate from admission control module $R_{admit}$
    $\beta$ : {The average ratio}
    $\alpha$ : {Congestion function parameter}
**Require:** Per-class Variable
    $Q_0$ : {Target Class Queue Length}
    $Q0_{in}$ : {Target point for IN packet queue}
    $qlen_{in}$ : {Length of IN packet queue}
    $qlen$ : {Packet queue length}
    $ER$ : {Explicit Rate}
    $DPF$ : {Queue Control Factor}
    $weight_i$ : {The ith class weight}
    $Q_0 \Leftarrow$ 0.3 * BufferSize
    $Q0_{in} \Leftarrow$ 0.3 * IN BufferSize
    $MACR \Leftarrow$ 100Mb {Assign a large value}
    $\beta \Leftarrow 0.08$
    $\alpha \Leftarrow 1.23$
```

Algorithm 12 Edge-Aware FIAC Algorithm (for DiffServ) at Ingress

```
Bandwidth Fair Control Function
if RD.tag ==IN then
  if $qlen_{in} \geq Q0_{in}$ then
    if $R^i \leq ER^i_{egress}$ then
      $EAR^i = ER^i_{egress} + \beta * (R^i - ER^i_{egress})$
    else
      $EAR^i = ER^i_{egress}$
    end if
  else
    $EAR^i = ER^i_{egress} + \beta * (ER^i_{egress} - R^i)$
  end if
else
  if $qlen \geq Q_0$ then
    if $R^i \leq ER^i_{egress}$ then
      $EAR^i = ER^i_{egress} + \beta * (R^i - ER^i_{egress})$
    else
      $EAR^i = ER^i_{egress}$
    end if
  else
    $EAR^i = ER^i_{egress} + \beta * (ER^i_{egress} - R^i)$
  end if
end if
Queue Control Function
if qlen $\geq Q_0$ then
  $DPF = (Buffer\_Size - qlen)/(Buffer\_Size - Q_0)$
else
  $DPF = 1 + (\alpha - 1) * (Q_0 - qlen)/Q_0$
end if
The ith class Explicit Admission Rate
if qlen then
  $EAR^i = EAR^i * DPF$
else
  $EAR^i = EAR^i$
end if
```

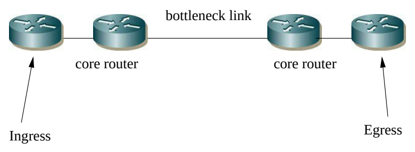


Figure 6.5: Simulation Topology: bottle neck is between two core routers

There are two considerations in Algorithm (12). First, The Edge-Aware FIAC tries to decrease the Explicit Admission Rate when the incoming traffic rate is greater than the network capability rate, ER_{egress}, and queue length is beyond the target point. Second, Edge-Aware FIAC tries to increase Explicit Admission Rate when the incoming traffic rate is less than the network capability rate and queue length is far from the target point.

6.4 Simulation Investigation

We evaluate and study the results of our Edge-Aware Resource Discovery DiffServ with Network Simulator version 2 (NS-2) [47].

The simulation topology is shown in Figure 6.5 with two DiffServ domains in tandem. Bandwidth of bottle neck link is 1 Mbps and others link is 100 Mbps. Multiple flows are aggregated to the class. Link delay is varied in order to aggregate TCP flows with different RTTs. We set up four classes: Gold, Silver, Bronze, and Best Effort.

In Experiment 1, we evaluate TCP traffic (in Gold, Silver, Bronze, and Best effort classes) performance in terms of goodput, end to end packet delay, and queue length.

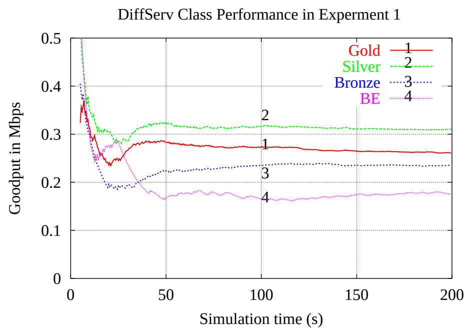
In Experiment 2, we study performance of mixing TCP and UDP traffic of classes in the point of throughput, end to end packet delay, and queue length.

In Experiment 3, we evaluate the bottleneck link efficiency when TCP flows traverse or TCP and UDP mixed flows traverse.

In Experiment 4, we study the impact of overhead on the performance.

6.4.1 Experiment 1: TCP traffic with different RTTs in different classes

In Experiment 1, we consider four classes, which are Gold, Silver, Bronze, and Best effort classes. They claim 40%, 30%, 20%, and 10% respectively. Each class has four TCP flows



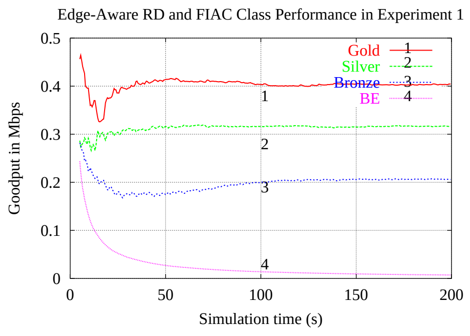
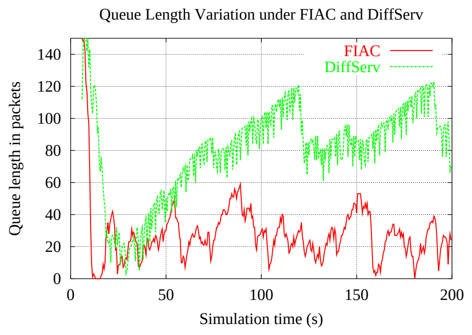


Figure 6.6: Goodput of Classes (TCP flows with different RTTs)



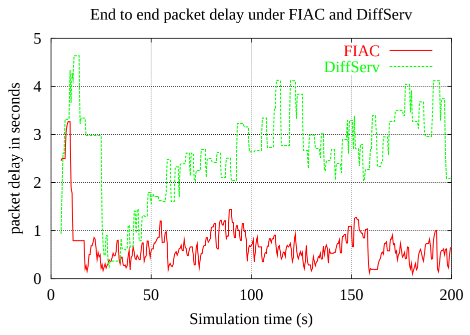


Figure 6.7: Queue Length and Packet Delay

with same round trip time (RTT). In Silver class, TCP flow has short RTT, 32 ms. In other classes, TCP flows have long RTTs, 150 ms (in Gold), 120 ms (in Bronze), and 50 ms (in Best effort).

In Figure 6.6, we plot the goodput results for all classes under traditional DiffServ and under our Edge Aware Resource Discovery and FIAC over DiffServ.

Based on this set of plots, we can make following observation. The DiffServ can not allocate bandwidth fairly for the classes which have different round trip time TCP flows. The traditional DiffServ allocates the Silver class (short RTT) more bandwidth, around 300 Kbps, which is beyond the Gold class. Our proposed model allocates each class quite fairly according to their service level agreement.

The end to end packet delay (flow 1 in Gold) and queue length of bottle neck router are shown in Figure 6.7. From the results, the proposed model achieves better performance compared to DiffServ in terms of packet delay and queue length.

6.4.2 Experiment 2: mixing TCP and UDP classes

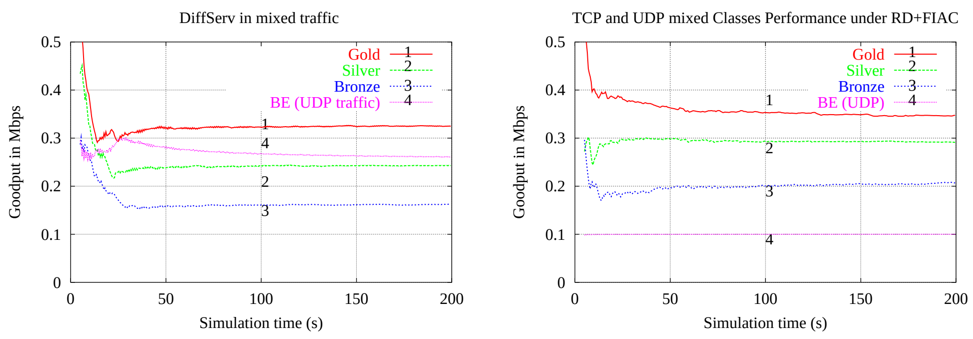


Figure 6.8: Goodput of TCP and UDP classes

Experiment 2 was designed to study the performance of mixing TCP and UDP classes in which one class has UDP traffic (in Best Effort) and other three classes only have TCP traffic (in Gold, Silver, and Bronze). In the simulation, UDP traffic whose constant bit rate is 1 Mbps so that the bottleneck is oversubscribed. The other TCP traffic flows (in Gold, Silver, and Bronze) have same RTT, 50 ms.

We investigate the simulation results in terms of bandwidth. The result is shown in Figure 6.8. From the results, the TCP flows in Gold class suffer acute degradation compared to UDP flow which is in Best effort class.

However, in Figure 6.8(b), the distribution of bandwidth is more fair than traditional DiffServ. That is our proposed model is to preferentially drop UDP packets because it has higher incoming rate, 1 Mbps, more than Explicit Admission Rate.

6.4.3 Experiment 3: Bottleneck link efficiency

Firstly we show the performance of Edge-Aware RD-FIAC when TCP flows traverse the bottleneck. We use the same topology with congested link depicted in Figure 6.9. The TCP flows are configured as same as Experiment 1. The bottleneck efficiency for Edge-Aware RD-FIAC is more than 90% as shown in Figure 6.9(a), comparable to DiffServ performance in the Figure 6.9(b). From this simulation, we see that without Edge-Aware RD-FIAC, DiffServ shows a degraded behavior under TCP flows traverse, which is around 60% bottleneck link efficiency. This result is derived from TCP behavior. If a packet loss

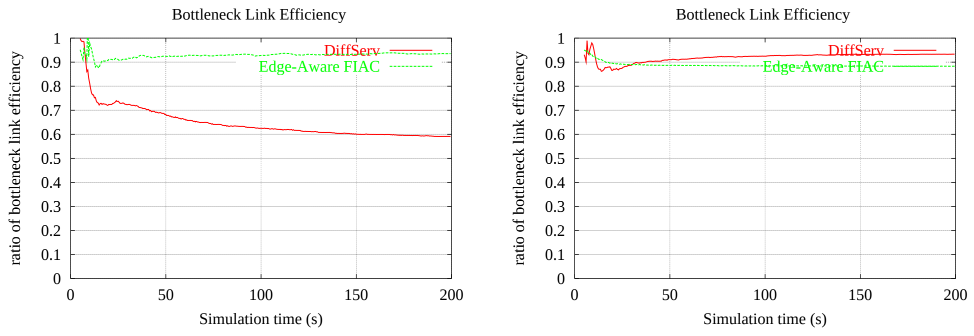


Figure 6.9: Bottleneck Link Efficiency

happens, a TCP data-sender halves its congestion window. So, TCP flows with DiffServ are not able to use the bottleneck link efficiently. For such the TCP flows, DiffServ won't work efficiently and high packet loss at the bottleneck node will cause more Slowstart. However, Edge-Aware RD-FIAC over DiffServ considers the available network resource and control TCP flows admission according to the network status. This causes less packet loss and thereby high bottleneck link efficiency.

In the second simulation, Figure 6.9(b) shows a scenario with one CBR flow and TCP flows which are configure as same as in Experiment 2. Although less admission control, performance of DiffServ is similar to Edge-Aware RD-FIAC over DiffServ. This is due to the CBR flow grabs the major part of the bottleneck link capacity.

6.4.4 Experiment 4: Overhead vs. Performance

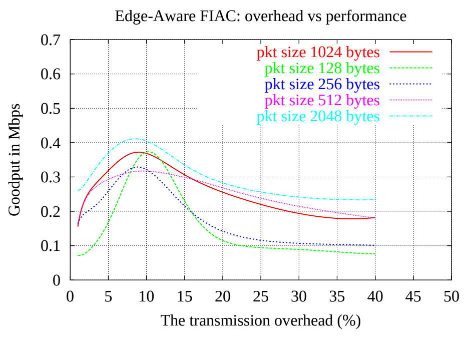


Figure 6.10: Overhead vs. Performance under different packet size

This experiment investigates the behavior of the Edge-Aware RD-FIAC if the scenario-parameters (overhead and packet size) are varied. Simulations in this experiment use the topology shown in Figure 6.10.

Figure 6.10 illustrates that overhead is an important parameter to be set properly. The overhead (RD packet) should be sufficiently high to discover the network status variation so that FIAC is able to response as soon as possible. However, increasing overhead may degrade the performance. We are aware of the fact that it is difficult to determine the optimal operating point for low overhead and high performance. Due to the lack of a tractable and sufficiently accurate analytical model we use an empirical approach to determine the setting of overhead as a function of the bottleneck bandwidth, packet size, RTT and number of classes. For the moment, we only consider the case of TCP flows with different packet size. The idea behind this empirical model is that we may find out the proper setting of overhead for a specific scenario by trial. By performing many simulations, exploiting different packet size, varied overhead, we get some rough optimal points which are around 10%.

6.5 Discussion

The proposed Edge-Aware Resource Discovery scheme intelligently resolves the overloading problem of DiffServ and preserves scalability. We have tested the Edge-Aware Resource Discovery scheme under a wide range of conditions; conditions purposely designed to stress its ability to achieve fair allocation without core router involvement. Certainly Edge-Aware Resource Discovery scheme is far superior in this regard to the normal DiffServ. Moreover, in all situations the Edge-Aware Resource Discovery scheme is roughly comparable with the Resource Discovery scheme. Recall that the normal Resource Discovery scheme requires core router to calculate its QoS states while the Edge-Aware Resource Discovery does not, so we are achieving these levels of fairness in a more scalable manner. However, there is clearly room for improvement in the Edge-Aware Resource Discovery scheme.

6.6 Summary

In this chapter, we studied the Edge-Aware Resource Discovery scheme. The key technique used in the Edge-Aware Resource Discovery scheme is Egress explicit rate calculation which detects the internal congestion. We considered the admission rate at Ingress router and the QoS states at Egress router to calculate the Egress explicit rate. By presenting a design and the simulation of the proposed algorithms, we have demonstrated that it is indeed to apply the Edge-Aware Resource Discovery scheme with existing routers.

Chapter 7

Future and Extension Work

There are a few interesting future research directions that are either direct extension of our work, or are motivated by more general problem of designing next-generation Internet capable of supporting QoS.

In the sequence of this chapter, Section 7.1 present directions for future work. Finally, Section 7.2 concludes our works.

7.1 Future Work

In the following we briefly discuss possible future research directions that we plan to explore.

7.1.1 Resource Discovery Protocol

So far, we have primarily focused on providing Edge-to-Edge QoS guarantees in the domain. The QoS states reported in this dissertation are the considerations of the available bandwidth and the available buffer states. On the other hands, many real-time multimedia applications may be more interested in average delay metrics. Therefore, it is of great interests and importance to study and provide QoS states in terms of the end-to-end delay and queue delay guarantees in the Internet. Consequently, an end-to-end delay scheme may be desirable, be it on a per-flow basis or on a per organization basis.

QoS states representation

The QoS states were studied in this dissertation. However, many issues regarding the QoS states representation of the Resource Discovery architecture remain unanswered. For example, how many local QoS states are needed to construct the exact local available resources, while still being able to provide certain degrees of QoS guarantees? What's the necessary QoS-state presentation in terms of the bandwidth requirement or delay bound or loss bound? We have found that it is difficult to find the general solution to satisfy the bandwidth assurance and delay bound. All of these questions are important for the Resource Discovery mechanism to express the exact QoS states. We plan to investigate these issues in the future work.

Multicast traffic

In the Resource Discovery scheme we have studied, the RD packets need to traverse all paths to collect the QoS states information. So, in order to control multicast traffic in the context of RD-DiffServ, one fundamental requirement is to duplicate the RD packet at the point of divergence. At the ingress router, when multiple QoS states reports are returned from the leaf nodes of the multicast tree, an algorithm is required to consolidate the reports and derive a suitable set of the QoS states parameters. Details of these issues regarding to multicast traffic need further study.

Granularity

The RD granularity refers to the resolution of the resource discovery mechanism in spatial domain. The following lists some possible examples:

- *Per-aggregated flow or per-microflow basis,* in which one RD packet is generated per contracted incoming flow. It implies a flow based resource discovery mechanism, which can generally give the finest grain precision.

- *Per-BA basis,* in which one RD packet is generated per behavioral aggregate (PHA). If an ingress node has access to more than one PHAs, multiple RD packets will be generated in each resource discovery interval.

- *Per-ingress-egress pair basis,* in which one RD packet is generated per ingress-egress-pair. Notice that the notion of ingress-egress-pair is defined only when there is a flow.

- A combination of above. Depending on their required control precision, network providers may choose to have a variant or a combination of the above mentioned schemes.

Details of these strategies need further study.

Overhead

The RD packet overhead is determined by two parameters: resource discovery period and packet size. The resource discovery period refers to how frequently a RD packet is generated. Typically, it can be specified in term of a time interval or packet count. The choice of a resource discovery period is related to the dynamics of the domain under control as well as the variation of the incoming traffic. To obtain a higher control precision, the ingress router may choose a shorter probing period, i.e. generate RD packets more frequently. However, this resource discovery frequency should be balanced with the cost required at the domain routers. The RD packet size should be designed to have enough space to accommodate the enough QoS states information can be carried in the RD packet. However, how frequently a RD packet should be generated is still a further research topic in terms of the variation of the traffic.

The main challenge is to balance the freshness of the QoS states information maintained at ingress routers with the overhead of the feedback mechanism. This is hard because the precision of the QoS states depends directly on the complexity of the resource discovery mechanism and high overhead.

The RD packet marking

Since the RD packets are sent on the same link, an interior/egress node needs a mechanism to distinguish the RD packets from other data packets. The RD packet can be created with a special DSCP (e.g. EF class) or normal DSCP (e.g. AF class), in which the QoS states information is carried in the fields of the RD packet.

For the normal DSCP, such as AF class, The RD packet clings together with data packets and receives the same level of forwarding treatment as other data packets, thereby it is subjected to being delayed or even dropped when the router is congested. This approach simplifies the design of the interior routers. By examining the arrival of the RD packets, the domain administrator can also obtain a sample of the current congestion level of the forwarding path.

For the special DSCP, such as EF class, the RD packets receive special service, usually better than data packets, at an interior node. It requires a special arrangement within the forwarding module of an interior node.

We will study different the RD marking schemes in the future..

Inter-Domain Resource Discovery Negotiation

So far, we have focused on the intra-domain resource discovery mechanism. In some cases, an inter-domain resource discovery negotiation is needed. Usually, domains are operated by different network providers. To enforce a global control mechanism across multiple domains, several problems need to be resolved.

- SLA conflicts: different providers usually maintain their own SLA policies in terms of management objectives. To resolve inter-domain resource discovery negotiation, it needs additional mechanism to resolve the SLA conflicts between two domains.

- Compatibility: the RD packet and information models used in different domains need to be compatible. Otherwise, a domain-to-domain resource discovery negotiation is not possible.

- Overall adaptation: since the RD packets need to traverse more than one domain, a longer round-trip control delay is unavoidable. The overall adaptation mechanism may be needed to consider.

- Negotiation model: hierarchical negotiation model needs to be investigated in the future.

Specifically, we plan to investigate the characteristics of RD packets observed at multiple vantage points, and understand the causes of and relationships among the domains.

We should also like to study the optimal approximation of the RD packets overhead associated with the network efficiency, or more generally, to optimize the network utilization and stability.

7.1.2 Fair Intelligent Admission Control

In the current admission control architecture design, we focused on the bandwidth allocation and management problem within a single network domain. However, the bandwidth allocation problem can be different in a multiple domain environment, where different network domains may belong to different administrations. Consequently, an end-to-end bandwidth negotiation scheme may be desirable, be it on a per-flow basis or on a per-aggregate basis.

Delay control

Another question that can be worth pursuing regards the delay control. Note that the end-to-end packet delay is highly dependent on the per-hop queuing delay, which requires cooperation between the different domains, so that different network domains agree on some common semantics for the service offered.

Inter-domain Fair Intelligent Admission Control Negotiation

We note that the techniques used in this dissertation can be further improved to achieve end-to-end QoS by devising appropriate inter-domain FIAC negotiation model. To find global solution, it may need inter-domain negotiation model to determine the local domain admission control.

Relationship between admission control and traffic characteristic

Our current approach assumes TCP Reno; extending it to other flavors of TCP such as SACK or Vegas [3] or other UDP traffic could be interest. Other applications such as video streaming would prefer a more flexible service in which they can opportunistically take advantage of the unused bandwidth to achieve better quality. In this context, it would be interesting to explore admission control scheme in terms of traffic characteristics.

Decoupling Bandwidth Allocation and Delay Allocation

Our solution can provide guaranteed services associates a single parameter to the traffic flow: the admission rate. While we can provide both the delay and bandwidth guarantees by appropriately setting the parameters, the fact that we are restricted to only one parameter may lead to inefficient resource utilization.

A future direction would be to decouple bandwidth allocation and delay allocations and uses at least two parameters to specify the traffic requirements. This significantly complicates the control plane implementations. The challenge is doing this without compromising the flexibility offered by decoupling the bandwidth allocation and delay allocation.

7.1.3 QoS Overlay

Overlay network is regarded as a promising solution for providing high-level network services over today's Internet. An overlay network is a virtual network built over an existing network as substrate. It brings more flexibility to add higher layer functionalities for today's Internet to support QoS. Our solutions described in this dissertation require changes to routers within a network domain. This is serious limitation that may delay or even preclude the deployment in the Internet. It would be much natural to apply our solution on the Overlay network.

7.1.4 Programmable Router

The programmable router architecture is capable to provide service accommodation with secure user separation in an extensible control plane and user based verified access to the real time forwarding plane. We envisage, with this architecture, a new class of network elements enriched with programmable functionality to be deployed as fundamental building blocks for a service infrastructure whereby end-to-end services can be introduced on-demand, composed across network domains, technologies, and administration boundaries.

The programmable router provides differentiated per-class resource allocation and fair allocation for users within each class. Due to the unpredictable nature of service resource consumption, the differentiated allocation is dynamic- allocating resource from lower classes to an upper class at its load increase.

7.2 Conclusion

In this dissertation, we have demonstrated by examples that it is possible to provide network service with resource discovery and admission control mechanisms in a scalable manner. To illustrate the power and the generality of our solution, we have applied the resource discovery mechanism and admission control mechanism on the DiffServ and the Internet infrastructure. While it is hard to predict the exact course of research in this area, we believe that many new and challenging problems of great practical importance and theoretical interest.

Chapter 8

Summary and Conclusions

From our design experience and the evaluation results discussed throughout this dissertation, we conclude that the resource discovery and fair intelligent admission control can be employed to utilize the network resources efficiently, through building the communication channel between the "internal" network QoS states capability and the "external" traffic requirements, without per-flow management at every router. We illustrate how these techniques work within a scalable architecture. The main strengths of our approach are scalability, fairness, support for different traffic, and robustness against the QoS states fluctuations.

Our work establishes a foundation for further studies on inter-domain resource and control negotiation issues. For example, some of our findings provide insights into making better resource negotiation and control negotiation.

In the sequence of this chapter, Section 8.1 summarizes the contributions of this thesis. Section 8.2 present directions for future work. Finally, Section 8.3 presents the final thoughts and conclusions.

8.1 Summary of the contributions

This thesis contributes with some steps toward a scenario where the Internet is able to deploy QoS in a scalable way.

This dissertation addressed the scalability issues in supporting Prioritization type of QoS from two complementary aspects, namely the resource discovery and the admission control. In the Resource Discovery framework, the core routers do not need to maintain any per-flow state and do not perform any per-flow operations. Consequently, they are able to handle a large number of simultaneous flows. The key notion in the resource discovery framework is to measure the available bandwidth state and the available buffer state by using the RD packet which traverses the en route path. The RD packet is initialized at network edge routers and referred and/or updated by network core routers, depending on the QoS states supported by the network. Several new resource discovery algorithms were designed and analyzed to illustrate how both the available bandwidth state and the available buffer state can be measured within this same framework. Moreover, we investigated the cost-performance trade-offs in supporting QoS in the Internet by studying the resource discovery mechanism.

On the admission control scheme, two scalable admission control schemes were designed and investigated. The first one was a per-class admission control scheme, which is built upon the fair intelligent admission control algorithms we designed. By conducting admission controls on a ingress-to-egress basis instead of on “hop-by-hop” basis, this admission control architecture significantly reduces the complexity; therefore, it improves the scalability of existing architectures. To further improve its scalability, a overlay admission control architecture was proposed for future direction. In this architecture, multiple admission control mechanisms are deployed at edge router in a overlay network, along with the conventional underlying network infrastructure. Edge routers handle the the admission control and interact with the other domain’s edge router for allocating and de-allocating resources control along the paths. In this way, the overlay admission control only needs to handle coarser time scale the resources control at the edge. Consequently, its scalability is greatly improved. The second scheme was a per-flow fairness control scheme, which was designed to achieve the flow fairness within a class.

Finally, to provide QoS support with more scalability and to facilitate the creation and deployment of value-added services over the Internet, an architecture called Edge-Aware Resource Discovery and Fair Intelligent Admission Control scheme was proposed which didn’t involve the core routers at all. The resource discovery mechanisms are deployed at the edge router to estimate the QoS states of the network. The Edge-Aware admission control was investigated, taking into account various factors such as SLA, QoS states,

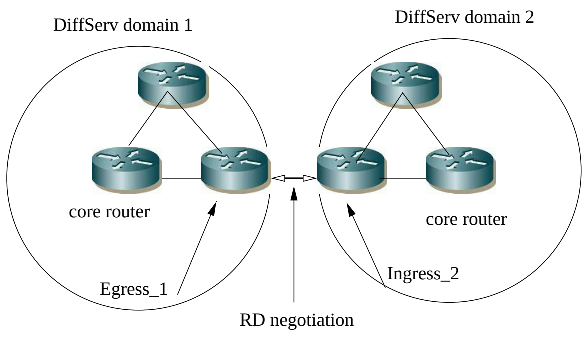


Figure 8.1: Inter-domain Resource Discovery Negotiation Module

bandwidth costs, and local buffer tolerable range.

In this dissertation we describe complete solutions including architecture, algorithms and implementations which address some of the important issues in today's Internet:

- Providing QoS. The RD-FIAC is able to provide per-class Prioritization type of QoS and per-flow flow fairness among a class.

- Scalability. RD-FIAC architecture eliminates the most complex operations on both data and control planes in the network in which the core routers do not maintain any per flow state.

- Edge-to-Edge resource control. By combining the Resource Discovery with Fair Intelligent Admission Control, we demonstrated that the RD-FIAC is possible to differentiate service among the aggregate traffic. It requires the network QoS states which is reported from the Resource Discovery mechanism and takes admission control at the edge routers based on the network QoS states.

- Edge-Aware Resource Discovery and Fair Intelligent Admission Control. We demonstrated that by modifying the Resource Discovery algorithms and Fair Intelligent Admission Control algorithms employed by the edge routers, we can make the Resource Discovery mechanism to estimate the "internal" network QoS states. We then introduced new fair intelligent admission control mechanisms to make admission decision based on the edge-aware resource discovery.

- Inter-domain Resource Discovery. Inter-domain resource discovery negotiation refers to the process whereby domains communicate with each other in order to make admission control on the right place. This negotiation is performed in a dynamic manner, to ensure the contracted QoS is sustained when traffic traverse multiple domains. Each Domain measures its resource availability by using the Resource Discovery mechanism and exchanges its resource availability with its neighboring domains. Upon receiving the next domain's state information, the egress router compares its available resource with the next domain's available resource and updates with the smaller one, then the QoS states information is feedback to the last domain. In other words, the ingress router of the first domain keeps the available resource which is the edge-to-edge path (multiple domains) can support. Figure 8.1 shows how the resource discovery works on multiple domains.

- QoS mapping. The RD-FIAC architecture layouts a helpful framework for implementing QoS mapping for different local QoS control.
- End-to-End QoS. The end-to-end QoS could be approached by concatenating the multiple domains edge-to-edge QoS with bilateral resource negotiation agreements and admission control.

8.1.1 The Design of the Resource Discovery and Fair Intelligent Admission Control Architecture

The high level design of the RD-FIAC architecture was presented in Chapter 3. RD-FIAC gives a step forward from the current proposal by clearly separating the functions particular to abstract services to their implementation with QoS technologies. The control and data planes are logically isolated from each other, and the communication between them happens through well-defined interfaces, following the usual approach of isolating layers in network architectures. This separation makes it possible to a domain to choose any approach for obtaining QoS, as long as it is able to provide services with the required performance guarantees.

In this dissertation, we presented a novel architecture for the Internet, which control the network resources efficiently based on the dynamic network QoS states report with the need for low computational overhead in network routers. We devised and analyzed mechanisms that implement the architecture, and demonstrated the effectiveness of the approach through analysis, simulation and measurement experiments in the simulations.

8.1.2 Resource Discovery Protocol

Chapter 4 presented the Resource Discovery Protocol, which is based on feedback mechanism. The RD architecture combines some very useful features that are only partially covered (or not at all) by other approaches found in the literature:

- The RD feedback mechanism is not tied to a particular QoS technology. Packets can be easily identified as belonging to a particular class and given the right treatment in order to preserve the semantics.
- It supports communication between “internal” network capability and “external” traffic requirements.
- It is aimed at finding the network QoS states which take into account the available bandwidth state and the available buffer states.
- The RD mechanism provides scalability and flexibility in collecting QoS states without per-flow information maintained at routers.

In the RD feedback loop for collecting the network QoS states, we used linear control techniques in the design of the algorithm, and derive stability around the operating point. While the stability condition derived does not ensure that the non-linear control converges, the stability condition gives useful guidelines for selecting the the control parameters.

Simulation results indicate that the proposed closed-loop resource discovery algorithm is an effective approximation of the optimization-based algorithm, and that the feedback QoS states can be updated dynamically.

To reduce the computational overhead of the RD feedback mechanism, we described the overhead analysis and algorithm that can be applied to implement our proposed mechanism at higher link speeds.

8.1.3 Fair Intelligent Admission Control

The Fair Intelligent Admission Control mechanism has enough information about the network capability and traffic requirements. Therefore, it is able to perform near optimal choices and use optimized algorithms for resource control. The model has therefore the potential scalability for dealing with resource and service negotiation in the whole Internet, as the DNS system does for the current Internet.

Based on the network QoS states report, we first presented the Fair Intelligent Admission Control (FIAC) algorithm that dynamically solves the network resources allocation efficiently and fairly which requires collaboration between the “internal” network QoS states and “external” traffic requirements. Using this admission control architecture, we demonstrated how admission control can be done on an entire path basis, instead of on a “hop-by-hop” basis. Such an approach may significantly reduce the complexity of the admission control algorithms without involving core router.

In designing per-class admission control algorithms, we investigated the problem of dynamic flow aggregation in allocating services and resources efficiently and fairly.

In designing per-flow fairness admission control algorithms, we illustrated how flow-level fairness can be achieved within a class at the ingress router. We consider the average flow fair share. When a new flow arrives at an ingress router, the average flow fair share will be recalculated and per-flow admission control applies the admissibility to determine admission decision which leads to the fairness. Clearly, using per-class or per-flow admission control schemes presented above, processing either the class-based resources allocation or per-flow fair share allocation requires the network QoS states information.

The issue of scalability is an important consideration in the design of the Fair Intelligent Admission Control scheme. To achieve scalability we push the complexity all the way to the edge routers.

8.1.4 Edge-Aware Resource Discovery and Fair Intelligent Admission Control

This dissertation suggests an Edge-Aware Resource Discovery (RD) feedback loop and Fair Intelligent Admission Control (FIAC) scheme for improving scalability. The scheme is characterized by two fundamental features. First, the RD loop infers the availability of network resources, yet for scalability, it does not involve core routers. Second, the admission control module understands incoming traffic's requirements, reconciles with available resources via the RD loop, and admits traffic intelligently according to the FIAC algorithm. Multiple-domain control is obtained by concatenation of the RD loop and FIAC admission for each domain along the path from a source to a destination.

An evaluation of the Edge-Aware Resource Discovery and Fair Intelligent Admission Control models was carried out and the results were reported in Chapter 6. The efficiency and fairness criteria were evaluated by a simulation study. The scalability criterion considered multiple domains, number of core routers, and multiple edge routers. The negotiation model was compared according to a simple analytical modeling and to the simulation results. Some findings of the comparisons are:

- The Edge-Aware Resource Discovery and Admission Control model is by far the best alternative, when comparing efficiency, fairness and scalability. It is also able to provide the right financial incentives for deploying QoS and to resolve some of the current problems of the interconnection of domains in the Internet. On the other hand, the Edge-Aware RD-FIAC model is simple due to the implementation and maintenance of the existing Internet and DiffServ infrastructure.
- They are robust in the presence of network losses and partial failures.
- Unlike existing reservation protocols, this Edge-Aware Resource Discovery and Admission Control is implemented without core routers involving at all.

8.2 Final Remarks

In this dissertation, we have presented the solution that can discover the network QoS states and take resources control in efficient and fair way. To illustrate the power and generality of our solution, we have implemented the Resource Discovery mechanism and Fair Intelligent Admission Control mechanism proposed in the context of the Internet and the DiffServ infrastructure. We believe that the door has been opened to many new and challenging problems of great practical importance and theoretical interest.

8.3 Conclusions

As shown in the first two parts of this thesis, it is feasible to implement mechanisms in the Internet that are able to discover the QoS states dynamically and admission control algorithms.

The resource discovery can be considered as a foundation to allocate resources efficiently and fairly to approximate end-to-end QoS. We propose two different mechanisms for resource discovery: core router involved resource discovery mechanism and edge-aware resource discovery mechanism. In Chapter 5, we investigate the Resource Discovery algorithm (RD). The RD implements functionality of detecting the network QoS states which is a function of the available bandwidth and the available buffer state. As a result, the RD mechanism can provide "inside" network QoS states or capability information dynamically.

We examine the design and functionality of the Resource Discovery mechanism in detail and present plausibility explanations how the RD packet detects and calculates the network QoS states. Additionally, we show that the Resource Discovery has low computational overhead and is scalable as it does not require storing per flow-information when the RD packet traverses the routers.

The second part of this thesis proposes the Fair Intelligent Admission Control algorithm (FIAC). The operational principle of FIAC is to take appropriate traffic admission control at the ingress router based on the network QoS states report from the Resource Discovery mechanism. If the incoming traffic rate is beyond the admission rate, its packet will be rejected or reshaped. In order to calculate the admission rate, the Fair Intelligent Admission Control needs to communicate the "internal" network QoS states and "external" traffic QoS requirements.

The RD-FIAC algorithm fits into the Differentiated Service Architecture and the Internet and can be considered scalable as it only requires per-flow state at ingress and egress routers. We show by simulation that RD-FIAC is able to detect the network QoS states dynamically and control the resources effectively and fairly. Additionally, parameter settings are discussed and the RD overhead caused by the information exchange between edge routers is analyzed. As outlined in Section 6.4, right parameters setting of the RD-FIAC algorithm can significantly reduce this overhead without significant sacrificing performance.

The third part of this thesis aims at investigating the Edge-Aware Resource Discovery and admission control mechanisms without the core routers involving. It depends on the observation that the difference between the admission rate out of the ingress and the arriving rate at the egress can be used to measure the network QoS states.

For the assumed traffic-pattern, stability can be achieved assuming the exact QoS states obtained. We show that the queue oscillates around the target point if it is set correctly. This model is suitable for heterogeneous RTTs, and mixed traffic. Finally, this system of equation is solved numerically and we may obtain the total model how to set the RD parameters as a function of the available bandwidth, the available buffer state, and target point settings. The model is evaluated by measurements and simulations with TCP traffic and UDP traffic.

However, it is still questionable whether appropriate setting of RD-FIAC parameters is possible in a realistic Internet environment. Although the quantitative models proposed in this thesis make an important step ahead, it is questionable whether ISPs will be able to retrieve sufficiently accurate QoS states information of the routers. The complexity of the model for TCP and UDP traffic suggest that the resource discovery and admission control algorithms need different parameters to accommodate the different situations.

Although the simplicity of the IP protocol is the main reason for the scalability and robustness of the Internet, some extensions to the Internet infrastructure are needed in order to make it possible to resolve both challenges of deploying QoS and giving the right incentives to users and domains. Sometimes over-provisioning is presented as the solution, but this approach has not been successful in solving the afore-mentioned problems. At most, over-provisioning provides a means for large domains having a network nearly free of packet loss.

Bibliography

[1] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris (2001), **Resilient Overlay Networks**, 18th ACM Symposium on Operating Systems Principles, October, 2001.

[2] C. Aurrecoechea, A. T. Campbell, and L. Hauw (1998), **A Survey of QoS Architectures**, ACM Multimedia Systems Journal, Special Issue on QoS Architectures, May 1998.

[3] L. Brakmo, and L. Peterson (1995), **TCP Vegas: End to End Congestion Avoidance on a global Internet**, IEEE Journal on Selected Areas in Communications, 13(8):1465-1480, October 1995.

[4] J. W. Barrett and J. F. Blowey (1995), *An error bound for the finite element approximation of the Cahn-Hilliard equation with logarithmic free energy*, Numerische Mathematics, **72**, pp 1–20.

[5] J. W. Barrett and J. F. Blowey (1997), *Finite element approximation of a model for phase separation of a multi-component alloy with non-smooth free energy*, Numerische Mathematics, **77**, pp 1–34.

[6] J. W. Barrett and J. F. Blowey (1999a), *An improve error bound for finite element approximation of a model for phase separation of a multi-component alloy*, IMA J. Numer. Anal. **19**, pp 147-168.

[7] J. Bennett, and H. Zhang (1997), **Hierarchical packet fair queueing algorithms**, In *IEEE/ACM Transactions on Networking*, Volume 5(5), pages 675-689, Oct 1997.

[8] Y. Bernet, J. Binder, S. Blake, M. Carlson, B. E. Carpenter, S. Keshav, E. Davies, B. Ohlman, D. Verma, Z. Wang, and W. Weiss (1999), **A framework for differentiated service**, Internet Draft, February, 1999.

[9] J. Bennett, K. Benson, A. Charny, W. Courtney, and J. -Y. Le Boudec (2001), **Delay jitter bounds and packet scale rate guarantee for expedited forwarding**, In *Proceedings of IEEE INFOCOM 2001*, volume 3, pages 1502-1509, Anchorage, AK, April 2001.

[10] J. Bernett, H. Zhang (1996), WF^2Q**: worst-case weighted fair queueing**, In *Proceedings of IEEE INFOCOM 1996*, volume 1, pages 120-128, San Franscisco, CA, March 1996.

[11] R. Callon, P. Doolan, N. Feldman, A. Fredette, G. Swallow, and A. Viswanathan (1999), **A framework for multiprotocol label switching**, Internet Draft, September 1999.

[12] D. M. Chiu, and R. Jain (1989), **Analysis of Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks**, Computer Networks and ISDN System 17, pp 1-14, 1989.

[13] Y. Chu, S. G. Rao, S. Seshan, and H. Zhang (2001), **Enabling Conferencing applications on the Internet using an Overlay Multicast Architecture**, In *Proceedings of ACM SIGCOMM 2001*, San Diego, CA, USA, August 2001.

[14] P. G. Ciarlet (1978), **The Finite Element Method for Elliptic Problems**, North-Holland.

[15] D. Clark, S. Shenker, and L. Zhang (1992), **Supporting real-time applications in an integrated services packet network: architecture and mechanism**, In *Proceedings ACM SIGCOMM*, August 1992.

[16] D. Clark, and W. Fang (1998), **Explicit allocation of best-effort packet delivery service**, In *IEEE/ACM Transactions on Networking*, 6(4):362-373, August 1998.

[17] R. Cruz (1991), **A calculus for netowrk delay, Part I: Network elements in isolation**, *IEEE Transactions on Information Theory*, 37(1):114-131, January 1991.

[18] Y. De Serres, L. Hegarty (2001), **Value-Added Services in the Converged Network**, *IEEE Communications Magazine*, September 2001.

[19] Z. Duan, Z. L. Zhang, and Y. T. Hou (2002), **Service overlay networks: SLAs, QoS and bandwidth provisioning**, In *Proceedings of IEEE International Conference on Network Protocols (ICNP)*, Paris, France, November, 2002.

[20] V. Firoiu, and A. Casati (1998), **Amendments to the Assured Forwarding PHB group**, IETF Internet Draft, November 1998.

[21] V. Firoiu, X. Zhang, and E. Gunduzhan (1998), **Feedback output queueing: a novel architecture for efficient switching systems**, In *Proceedings of Hot Interconnects X*, Stanford, CA, August 2002.

[22] S. Floyd, M. Handley, J. Padhye, and J. Widmer (2000), **Equation-based congestion control for unicast applications**, In Proc. ACM SIGCOMM, pp 43-56, Stockholm, Sweden, August 2000.

[23] S. Floyd, and V. Jacobson (1993), **Random early detection gateways for congestion avoidance**, IEEE/ACM Transactions on Networking, 1(4): 397-413, August 1993.

[24] S. Floyd, and K. Fall (1998), **Promoting the Use of End-to-End Congestion Control**, IEEE/ACM Transactions on Networking, February 1998.

[25] S. Floyd, and K. Fall (1998), **Promoting the Use of End-to-End Congestion Control in the Internet**, IEEE/ACM Transactions on Networking, August 1999.

[26] S. Ben Fredj, T. Bonald, G. Regnie A Proutiere, and J. W. Roberts (2001), **Statistical bandwidth sharing: A study of congestion at flow level**, In Proceedings of ACM SIGCOMM, pp 111-122, August 2001.

[27] R. J. Gibbens, and F. P. Kelly (1999), **Resource pricing and the evolution of congestion control**, Automatica, 35: 1969-1985, 1999.

[28] M. Goyal, A. Durresi, R. Jain, and C. Liu (1999), **Performance analysis of Assured Forwarding**, IETF Draft, October 1999.

[29] J. Ibanez, and K. Nichols (1998), **Preliminary simulation evaluation of an assured service**, IETF Draft, August 1998.

[30] V. Jacobson (1988), **Congestion Avoidance and Control**, In Proceedings of ACM SIGCOMM, August 1988.

[31] F. Kelly (1997), **Charging and Rate Control for Elastic Traffic**, Europe Trans. Telecommun, 8:33-37, 1997.

[32] F. Kelly, A. Maulloo, and D. Tan (1998), **Rate control in communication networks: Shadow prices, proportional fairness and stability**, Journal of the Operational Research Society, 15(49): 237-255, 1998.

[33] J. L. Lions (1969), **Quelques Móthodes de Résolution des Problémes aux Limites**, Dunod.

[34] M. Li, and D. B. Hoang (2004), **Edge-Aware Resource Discovery and Fair Intelligent Admission Control over Multi-domain Differentiated Services Networks**, In *IEEE International Conference on Commmunications (ICC2004)*, Paris, France, June 22-26, 2004.

[35] M. Li, and D. B. Hoang (2004), **FIAC: A Resource Discovery-Based Two-level Admission Control for Differentiated Service Networks**, In *Computer Communication Journal: Special Issue on End-to-End Quality of Service Differentiation*(In press), 2004.

[36] M. Li, D. B. Hoang, and A. Simmonds (2004), **Fair Intelligent Admission Control over Resource-feedback Differentiated Service Network**, to appear In *Computer Communication Journal: the Special Issue of Quality of Service,* early 2005.

[37] M. Li, and D. B. Hoang (2003), **Achieving Flow Fairness in DiffServ Class: Per-flow Fair Admission Control over Differentiated Service Network**, In *Proceedings of 4th International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD'03)*, Luebeck, Germany, October 16-18, 2003.

[38] M. Li, D. B. Hoang, and A. Simmonds (2003), **Fair Intelligent Admission Control over DiffServ Network**, In *Proceedings of 2003 11th IEEE International Conference on Networks (ICON2003)*, Sydney, Australia, September 28 - October 1, 2003.

[39] M. Li, D. B. Hoang, and A. Simmonds (2003), **Class-Based Fair Intelligent Admission Control over an Enhanced Differentiated Service Networks**, In *IEEE International Conference on Information Networking 2003 (ICOIN2003)*, Jeju Island, Korea, February 12-14, 2003.

[40] M. Li, D. B. Hoang, and A. Simmonds (2003), **Class-Based Fair Intelligent Admission Control over an Enhanced Differentiated Service Networks**, In *lecture Notes in Computer Science (LNCS) 2662*, Hyun-Kook Kahng (Ed.), Springer, 2003.

[41] D. B. Hoang, and M. Li (2003), **Fair Intelligent Congestion Control over DiffServ: A Resource Discovery and Control Scheme for DiffServ**, In *Proceedings of the 2003 International Conference on Information and Communication Technologies (ICT 2003)*, Bangkok, Thailand, April 8-10, 2003.

[42] D. B. Hoang, M. Li, and U. Szewcow (2002), **RTT-Aware Resource Discovery and Fair Share Mechanism for DiffServ**, In *Proceedings of the IEEE International Conference on Networking (ICN 2002)*, Atlanta, USA, August 26-29, 2002.

[43] Q. Yu, M. Li, D. B. Hoang, and D. Feng (2002), **Fair Intelligent Feedback Mechanism on TCP Based Network**, In *The 2002 International Conference on Internet Computing (IC 2002)*, Las Vegas, California, USA, June, 2002.

[44] D. B. Hoang, Q. Yu, M. Li, and D. Feng (2002), **Fair Intelligent Congestion Control Resource Discovery Protocol on TCP Based Network**, In *Proceedings of the IFIP 6th Interworking 2002 Symposium*, Perth, Australia, October, 2002.

[45] M. Li, and D. Feng (2004), **Resource Discovery and Fair Intelligent Admission Control over Differentiated Services Networks for Variable-Length Packets**, In *Proceedings of the IEEE 10th Asia-Pacific Conference on Communications (APCC2004)*, Beijing, China, August, 2004.

[46] L. Massoulie, and J. Roberts (1999), **Objectives and algorithms**, In Proceedings of IEEE INFOCOM, pp 1395-1403, 1999.

[47] NS (2000), **NS Simulator Homepage**, http://www-mash.cs.berkeley.edu/ns/, 2000.

[48] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose (1998), **Modeling TCP throughput: a simple model and its empirical validation**, In Proceedings of ACM SIGCOMM'98, September 1998.

[49] A. K. Parekh, and R. G. Gallagher (1993), **A gereralized processor sharing approach to flow control in Integrated Service networks: The single-node case**, IEEE/ACM Transactions on Networking, 1(3): 344-357, June 1993.

[50] P. Pieda, N. Seddigh, and B. Nandy (1999), **The dynamics of TCP and UDP interaction in IP-QoS Differentiation Service networks**, In 3rd Canadian Conference on Broadband Research (CCBR), November 1999.

[51] , K. K. Ramakrishnan, R. Jain, and D. M. Chiu (1987), **Congestion avoidance in computer networks with a connectionless network layer. Part iv: A selective binary feedback scheme for general topologies methodology**, Technical Report DEC-TR-510, Digital Equipment Corporation, 1987.

[52] J. Postel (1981), **Transmission Control Protocol**, IETF RFC 793, September 1981.

[53] R. Braden, D. Clark, and S. Shenker (1994), **Integrated Service in the Internet Architecture: an overview**, IETF RFC1633.

[54] F. Baker (1995), **Requirements for IP Version 4 Routers**, IETF RFC 1812, June 1995.

[55] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin (1997), **Resource reservation protocol (RSVP) - version 1 functional specification**, IETF RFC 2205, September 1997.

[56] S. Shenker, C. Partridge, and R. Guerin (1997), **Specification of guaranteed quality of service**, IETF RFC 2212, September 1997.

[57] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss (1998), **An Architecutre for Differentiated Service**, IETF RFC2475.

[58] G. Huston (2000), *Analysis of Existing QoS Solutions*, IETF RFC2990.

[59] V. Jacobson, K. Nichols, and K. Poduri (1999), *An Expedited Forwarding PHB group*, IETF RFC2598.

[60] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski (1999), *Assured Forwarding PHB group*, IETF RFC2597.

[61] K. Nichols, V. Jacobson, and L. Zhang (1999), **A Two-bit Differentiated Services Architecture for the Internet**, IETF RFC 2638, July, 1999.

[62] B. Davie, A. Charny, J. Bennett, K. Benson, J.-Y. Le Boudec, W. Courtney, S. Davari, V. Firoiu, and D. Stiliadis (2002), **An expedited forwarding PHB**, IETF RFC 3246, March 1999.

[63] E. C. Rosen, A. Viswanathan, and R. Callon (1999), **Multiprotocol label switching architecture**, Internet Draft, August 1999.

[64] S. Sahu, P. Nain, D. Towsley, C. Diot, and V. Firoiu (2000), **On achievableservice differentiation with token bucket marking for TCP**, In Proceedings ACM SIGMETRICS'00, pp 23-33, June 2000.

[65] V. Srinivasan, S. Suri, and G. Varghese (2001), **Packet classification using tuple space serach**, In *Proceedings of ACM SIGCOMM'99*, pages 135-146, Boston, MA, August 2001.

[66] I. Stocia, and H. Zhang (1999), **Providing guaranteed services without per flow management**, In Proceeding ACM SIGCOMM, Boston, MA, USA, September 1999.

[67] L. Subramanian, I. Stocia, H. Balakrishnan, and H. Katz (2002), **OverQoS: Offering Internet QoS Using Overlays**, In *Proceedings of HotNet-I Workshop 2002*, Princeton, NJ, USA, October 2002.

[68] B. Teitelbaum, and T. Hanss (1998), **QoS Requirements for Internet2**, Internet2 QoS Working Group Draft, April 1998.

[69] R. Yavatkar, D. Hoffman, Y. Bernet, and F. Baker (1999), **SBM (Subnet Bandwidth Manager): A Protocol for RSVP-based Admission Control over IEEE 802-style networks**, *IETF draft ¡draft-ietf-issll-is802-sbm-08.txt¿*, May 1999.

[70] Ikjun Yeom, and A. L. Narasimha Reddy (2001), **Modeling TCP behaviour in a differentiated services network**, IEEE/ACM Transactions on Networking, 9(1): 31-46, 2001.

[71] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala (1993), **A new resource reservation protocol**, *IEEE Network*, pages 8-18, September 1993.