Scalable QoS Guaranteed Communication Services for Real-Time Applications

Byung-Kyu Choi, Dong Xuan, Riccardo Bettati, Wei Zhao
Department of Computer Science
Texas A&M University
College Station TX 77843-3112, USA
choib, dxuan, bettati, zhao@cs.tamu.edu

Chengzhi Li
Department of Electrical and Computer Engineering
Rice University
6100 S. Main Street, Houston TX 77005, USA
chengzhi@ruf.rice.edu

Abstract

In this paper, we propose an approach to flow-unaware admission control, which is combination with an aggregate packet forwarding scheme, improves scalability of networks while guaranteeing end-to-end deadlines for real-time applications. We achieve this by using an off-line delay computation and verification step, which allows to reduce the overhead at admission control while keeping admission probability and resources utilization high. Our evaluation data show our system's admission probabilities are very close to those of significantly more expensive flow-aware approaches. At the same time, admission control overhead during flow establishment is very low. Our results therefore support the claim from the DS architecture literature that scalability can be achieved through flow aggregation without sacrificing resource utilization and with significant reduction in run time overhead.

1. Introduction

This paper addresses scalable QoS guaranteed real-time communication services over computer internetworks. In this context, by *real-time* we mean that a packet is delivered from its source to the destination within a predefined end-to-end deadline. Packets delivered beyond this end-to-end deadline are considered useless. Internetworking technology is increasingly being used for applications with this kind of requirements, for example, Voice over IP, military communications, and industrial control systems.

Traditionally, *best-effort* has been the main type of service available over internetworks. While this type of service has contributed much towards the rapid growth of the Internet, it can-not support applications that have real-time requirements.

A systematic approach based on connection admission control and packet scheduling was proposed within the Integrated Services (IS) architecture of the IETF [1]. In Integrated Services, each flow is strictly controlled both by admission control at flow establishment time and packet scheduling during the lifetime of the flow. This necessitates that information about every flow is kept by each node along the path. Both admission control and packet forwarding are therefore *flow-aware*.

Integrated Services are difficult to deploy in large-scale high-speed internetworks, as they don't scale well, for two reasons: first, high-speed routers are easily forced to maintain and schedule packets for thousands of connections in the core network. Second, as the number of connections increases, the run-time overhead incurred for connection establishment and tear-down increases too.

This lack of scalability is, to a large extent, being addressed within the Differentiated Services architecture [26, 27, 28]. From the user's point of view, the DS (Differentiated Services) model partitions each user-level *flow* [1] into one of a set of predefined *classes*. Packets of each class are served inside the network according to a class-based scheduling policy. The result is that routers are aware only of *aggregations* of flows [2]. Each edge router is supposed to aggregate the individual flows into a small number of such aggregate flows. In this fashion, the DS model makes the

[1] In the following, we will use the term flow to indicate a stream of data between a source and a destination, and the term connection to indicate the virtual circuit that needs to be established to carry the flow.

[2] In this paper we use aggregated flow and class interchangeably.

network scalable regardless of the number of flows.

The objective of this work is to further improve the scalability of QoS guaranteed services by considering both admission control and packet forwarding together within a DS architecture.

The rest of the paper is organized as follows. In Section 2, we describe previous work and our motivation. Our models of network, flow, and the DS model are presented in Section 3. In Section 4, we describe our approach, including off-line end-to-end delay calculation, admission control, and class-based static priority scheduling scheme. Section 5 presents evaluation by simulation of our system and previous works. While Section 6 introduces issues involved in implementation, a summary and motivation of future work are given in Section 7 in our conclusions.

2. Previous Works

Obviously, most currently deployed packet forwarding schemes for internetworks do not guarantee end-to-end delay, since the delay purely depends on the dynamic load of network links. As described earlier, the Integrated Service architecture is a great step towards to the goal of QoS guarantees. One step further, the Differentiated Services architecture [26, 27, 28] takes scalability issues into account.

In terms of end-to-end delay guarantees for real-time services, our main interests are admission control and packet scheduling. As far as these two technical aspects are concerned, much research has been proposed. For a variety of traffic models, end-to-end delay calculation methods have been proposed [9, 10, 15, 16]. Also, many packet scheduling mechanisms have been provided [7, 11, 13, 17, 19, 20, **?**]. Finally, admission control has been investigated as well [12, 14, 18]. Some of the proposals have been implemented and deployed, for example, NetEx, RSVP, and Tenet. NetEx [2, 3] is a good working example of implemented system based on extensions of Cruz's methodology for delay analysis [4, 5, 6]. RSVP [21] has been proposed as the signaling and resource reservation in the Integrated Services architecture. Tenet Scheme II [22] uses a two-pass resource allocation scheme, with extensive functionality needed for multi-party communication. However, one common point of all these systems is their limited scalability. As the number of flows increases, the admission control needs proportionally more time to make admission decision.

Recently, a number of efforts have focused on reducing the negative effect on scalability caused by flow awareness in signaling protocols. For example, the basic idea in [8] is to move per-flow information to edge routers from core routers by relying on estimation. The key points of the estimation technique are: 1) each core router relies on dynamic packet status information scheduling, 2) dynamic packet status information is carried over by each packet. With this information, the core router estimates each flow's dynamic information. As such, [8] does not do flow aggregation. Only the *storage* of per-flow information has moved to the edge router from the core router. This means that, since each packet bears the flow's information, the run-time overhead of the estimation technique becomes a non-negligible factor.

For the DS architecture, much research needs to be done to provide end-to-end delay guarantees. In this paper, we propose a system of admission control and packet forwarding schemes that guarantee end-to-end delay for each flow in a DS architecture [27].

3. Our Models

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. Following the DS model, we assume that the network is partitioned into multiple *domains*. Each domain can have many routers. Figure 1 illustrates an example of one domain. The network resources are controlled by the *domain manager(s)* in centralized or distributed fashion. In this paper, we will first focus on a network with a single domain. Then, we will discuss how to extend our work to multiple domains.

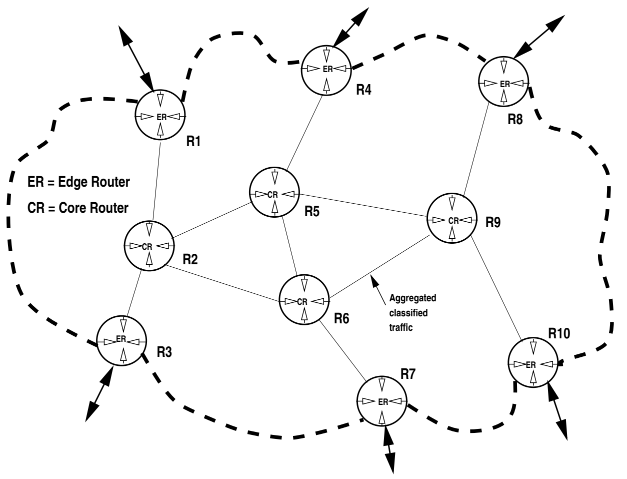


Figure 1. A simple illustration of our domain

Routers and Links There are two kinds of routers in a domain: edge router and core router as shown in Figure 1. 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.

Packets may exceed the capacity at the output link, and thus the packet may suffer some queueing delay at the output link. This delay depends on the load of the output link and the scheduling policy adopted at the output link. For delay computation purpose, the router can be regarded as a set of link *servers*, one for each output link, at which the packet can experience the queuing delay. C denotes the link capacity in bits per second. We denote link server i as S_i. There are total M servers.

Paths The path is a list of nodes. Equivalently, a path can also be denoted as a list of link servers. For example, if a path traverses link servers: S_1, S_2, S_6 and S_7, then the path can be denoted as $< 1, 2, 6, 7 >$, where the integers are the ids of link servers. In the following discussion of the admission control, we assume that the path of a new flow has been determined by a separate routing sub-system and is assumed to be given.

Flows and Classes of Service A series of continuous packets from a source to a destination router forms a *flow*. We assume that the flow is constrained by a leaky bucket at the edge router when it enters the domain. This edge router is called the *ingress* router for this flow.

Following the DS model, flows are partitioned into *classes*. QoS requirement and specifications of the traffic carried by the flows are defined on a class-by-class basis. The traffic carried by flows in the same class has the same mathematical representation at the ingress routers, which is given as follows:

DEFINITION 1 *The source traffic of a flow in Class i is controlled by a leaky bucket with parameters (T_i, ρ_i). The total amount of traffic, generated by this source during any time interval $[t, t+I)$, is bounded by $\min\{C*I, T_i+\rho_i*I\}$, where*

T_i is the burst size of source traffic of flows in Class i.
ρ_i is the average rate of source traffic of flows in Class i.
C is the link capacity.

While the leaky bucket appropriately characterizes the source traffic at the ingress router, this traffic characterization is not valid within the networks, as the traffic is perturbed by queueing delays at servers as it traverses the network. We re-characterize traffic at any servers in the network using *traffic functions* and their time independent counterpart, *traffic constraint functions*.

DEFINITION 2 *The traffic function $f_{i,k,j}(t)$ is defined as the amount of the traffic of Class i arriving at Server k by the input link j during time interval $[0, t)$.*

Traffic functions are cumbersome to handle and not of much help in admission control, as they are time dependent. A time-independent traffic characterization is given by the traffic constraint function, which is defined as follows:

DEFINITION 3 *The function $F_{i,k,j}(I)$ is called the traffic constraint function of $f_{i,k,j}(t)$ if for any $t > 0$ and $I > 0$,*

$$f_{i,k,j}(t+I) - f_{i,k,j}(t) \leq F_{i,k,j}(I). \qquad (1)$$

In addition to the traffic characterizations, the QoS requirement of flows is specified on a class-by-class basis as well. For our purpose, we define the deadline requirement of traffic in Class i to be D_i. All flows in the same class receive the same level of delay guarantees. In the following, we will use $d_{i,k}$ to denote the queueing delay suffered by Class i traffic at Sever k. Following the DS model, we assume that at each link server, a certain percentage of bandwidth is reserved for individual traffic classes. Let α_i denote the percentage of bandwidth reserved for Class i at all link servers.

4. Scalable QoS Guaranteed System

4.1 Overview

While our objective is to provide scalability for QoS guaranteed system, other aspects must be considered as well.

Scalability : In our system, admission control and packet forwarding scheme are scalable regardless of the number of flows in the system. This means that the overhead of admission control and packet forwarding is independent of the number of flows in the system.

Effectiveness : Our admission control maximizes the bandwidth utilization to the extent possible. This means that it is highly accurate even though it does not rely on per-flow information.

Compatibility : For practical purposes, our system is compatible with current industrial practice.

Our system consists of three major modules:

Off-line delay computation and verification : Here, we use a novel delay calculation method to estimate upper bounds on delays for flows of every class at each router. This module verifies whether the end-to-end delay bound in each feasible path of the network satisfies the deadline requirement as long as the bandwidth usage on the path is within a pre-defined limit. The module comes into play during initial network configuration, or during modification of service level agreement.

Efficient admission control : As the delay has been verified off-line, this module checks only if the bandwidth is available along the path of the new flow.

Packet forwarding : In a router, packets are transmitted according to their class priorities. Within a class, packets are served in FIFO order.

4.2. Recursive Formula for Delay Computation

Within a DS architecture, each router forwards arriving packets only based on their class identification, without regard to which flow they belong to. In order to determine whether delay requirements can be met, queuing delays experienced by traffic must be computed. Given the information about the amount of bandwidth reserved for the aggregated traffic of each class on the links and the source flow model of that traffic, we derive the following formula to estimate the worst case queuing delay suffered by packets belonging to a particular class.

THEOREM 1 *The worst case queuing delay $d_{i,k}$ suffered by the packets of Class i at Server k is*

$$d_{i,k} = \frac{1}{1-\sum_{l=1}^{i-1}\alpha_l}\left[\sum_{l=1}^{i}(T_l+\rho_l * Y_{l,k})\frac{\alpha_l}{\rho_l} + \left(\sum_{l=1}^{i}\alpha_l - 1\right)\frac{\alpha_i(T_i+\rho_i Y_{i,k})}{\rho_i(N-\alpha_i)}\right] \qquad (2)$$

where

$$Y_{l,k} = \max_{Path\in S_{l,k}} \sum_{j\in Path} d_{l,j}, \qquad (3)$$

and where $S_{l,k}$ is the set of all paths traversed by the packets of Class l before arriving at Server k.

In order to simplify notations, we use vector $\vec{d_i}$ to denote the bounds of the queuing delays suffered by the traffic of Class i at all servers:

$$\vec{d_i} \quad = \quad [d_{i,1}, d_{i,2}, \cdots, d_{i,M}]^{\top}. \qquad (4)$$

We note in Equation (2) that $d_{i,k}$ depends on $Y_{l,k}, l = 1, \cdots, i$, that is, the worst case queuing delays experienced before arriving at Server k by traffic of classes with i or higher. On the other hand, according to Equation (3), the value of $Y_{l,k}$ in turn depends on the delays experienced by Class l traffic at servers other than Server k. In general, we have a circular dependency, as $\vec{d_l}$ depends on $Y_{l,k}$, and $Y_{l,k}$ in turn on $\vec{d_l}$.

We can simplify the computation by taking into account that high-priority traffic is not delayed by low-priority traffic. Therefore, if we first compute $\vec{d_1}$, then $\vec{d_2}$, and so on, we can simply assume $Y_{l,k}, l = 1, \cdots, i-1$, are constants when computing $d_{i,k}$. Nevertheless, for given Class i, the values of $d_{i,k}, k = 1, \cdots, M$ depend on each other and must be computed simultaneously. A vector equation to do this can be derived as follows: Having obtained $\vec{d_l}$, for $l = 1, \cdots, i-1$, we can denote

$$z_{i,k}(\vec{d_i}) = \frac{1}{1-\sum_{l=1}^{i-1}\alpha_l}\left[\sum_{l=1}^{i}(T_l+\rho_l * Y_{l,k})\frac{\alpha_l}{\rho_l} + \left(\sum_{l=1}^{i}\alpha_l - 1\right)\frac{\alpha_i(T_i+\rho_i Y_{i,k})}{\rho_i(N-\alpha_i)}\right] \qquad (5)$$

and

$$\vec{Z_i}(\vec{d_i}) \quad = \quad [z_{1,i}(\vec{d_i}), z_{2,i}(\vec{d_i}), \cdots, z_{M,i}(\vec{d_i})]^{\top}. \qquad (6)$$

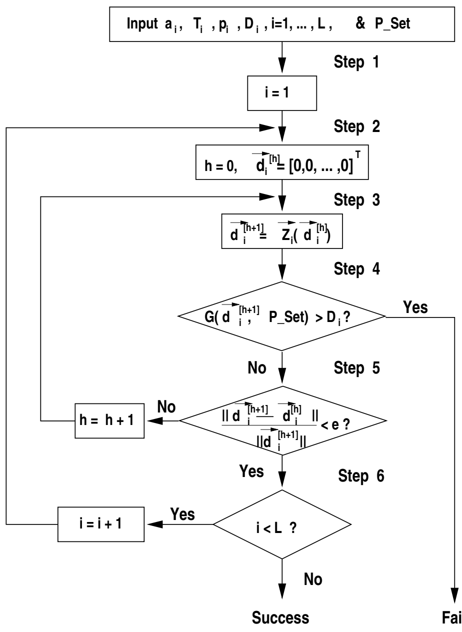
The queuing delay bound vector $\vec{d_i}$ can then be determined by the following vector equation:

$$\vec{d_i} \quad = \quad \vec{Z_i}(\vec{d_i}). \qquad (7)$$

Usually a delay computation formula for a server would depend on the state of the server, i.e., the number of flows that are admitted and pass through the server. We note that our new delay formula in Equation (2) depends on T, ρ, α, and N. The values of these parameters are available at the time when the system is (re-)configured.

4.3. Off-line End-to-end Delay Computation and Verification

Equation (7) provides a means to compute the worst-case delay experienced by any flow in a given network. We can directly make use of this to verify off-line whether the network with the given bandwidth allocation of service classes satisfies all the delay requirement of the classes. If this is the case, admission control during flow establishment is limited to simply check whether enough bandwidth is available on the links along the path of the flow.



Input a_i, T_i, p_i, D_i, i=1, ... , L, & P_Set
Step 1
i = 1
Step 2
h = 0, $\vec{d_i}^{[h]} = [0,0, \ldots ,0]^T$
Step 3
$\vec{d_i}^{[h+1]} = \vec{Z_i}(\vec{d_i}^{[h]})$
Step 4
$G(\vec{d_i}^{[h+1]}, P\_Set) > D_i$?
Yes
No
Step 5
$\frac{\|\vec{d_i}^{[h+1]} - \vec{d_i}^{[h]}\|}{\|\vec{d_i}^{[h+1]}\|} < e$?
No
h = h + 1
Yes
Step 6
i < L ?
Yes
i = i + 1
No
Success
Fail

Figure 2. Algorithm for off-line delay computation and verification

A work-flow representation of the algorithm for the off-line delay computation and verification is given in Figure 2. The input of the algorithm are defined as follows:

α_i: the portion of link bandwidth assigned to the traffic of Class i.
T_i: The burstiness of the traffic of a flow of Class i at the ingress routers.
ρ_i: The average rate of traffic of a flow of Class i.
D_i: The delay requirement of flows in Class i.
P_Set: The set of all possible paths that can be taken by flows. This is determined by the network topology and the routing policy.
L: number of classes.

The algorithm consists of two nested loops. The outer loop traverses all the classes. For each Class i, the algorithm computes the delays on all servers, assuming that lower priority of classes do not affect the transmission for flows in Class i, and that delays of higher priority classes have already been computed. The inner loop computes the delays for traffic in Class i by iterating in Step 4 (h is the iteration counter) until either the delay values of the delays in $\vec{d_i}$ converge or it can be safely concluded that the delay requirement for Class i can not be met.

In Step 3, the recursive formula of Equation (7) for delay computation is called to calculate $\vec{d_i}^{[h+1]}$ based on $\vec{d_i}^{[h]}$, until the maximum end-to-end delay as computed by Function $G(\cdot)$ is larger than the delay requirement D_i of flows in Class i (Step 4), or until the computation of delays has converged (Step 5)[3].

Function $G(\cdot)$ computes the worst-case end-to-end delay that can be experienced by any flow of Class i in the system, given $\vec{d_i}^{[h+1]}$ and the route information. This can be done, for example, by summing up and comparing the local delays in $\vec{d_i}^{[h+1]}$ for the link servers along any possible paths used by flows in Class i. If Function $G(\cdot)$ returns a delay larger than D_i, there is at least one route along which the delay requirement for Class i traffic can-not be guaranteed. In that case, the algorithm returns $FAIL$ to report that the verification of the end-to-end delay did not succeed.

If the delay computation converges, the delay computation for Class i is finished, and the algorithm proceeds to compute the delays for the class with the next lower priority. If the delay of all the classes traffic is calculated, the outer loop terminates, and the algorithm returns $SUCCESS$, which indicates that the algorithm was successful in verifying that the delay requirements for all classes can be met.

THEOREM 2 *If the algorithm returns $SUCCESS$, the end-to-end delay will be guaranteed for any new request as long as enough bandwidth is available along the path of the flow.*

[3]We let $\|\cdot\|$ denote the maximum norm operator, that is, $\|[d_1,\cdots,d_n]^\top\| = \max_{i=1}^{n}|d_i|$

In [30], we prove the correctness of this algorithm and prove that of completeness as well.

Once the algorithm has determined that end-to-end deadlines are met for all classes, the delay requirement need not to be checked anymore during the establishment of new flows. Note that the algorithm runs at network (re-)configuration, not during network operation.

4.4. Admission Control

Once the network configuration has been previously verified off-line to meet the delay requirements of all classes for the given bandwidth allocation to classes, the admission control needs to check only that the bandwidth allocation along the path of the new flow is not violated. Thus, the admission control at flow establishment time reduces to keeping track of the amount of bandwidth used on each link by flows of each class and making sure that no class exceeds its allocated portion of bandwidth.

4.5. Packet Forwarding Scheme

The delay formula assumes the usage of a class-based static priority scheduling policy at the link servers, in which each class has its own FIFO queue. We do not rely on any form of traffic shaping in the link server. Note that while this significantly complicates delay calculation, it keeps the packet forwarding mechanism extremely simple.

5. Experimental Evaluation

In order to evaluate the performance (in terms of admission probability and of admission control overhead) of our class-based approach, we ran a suite of simulation experiments to compare it with two forms of connection oriented admission control systems: (1) NetEx [2, 3] has been selected as a generic connection oriented admission control with static priority system packet forwarding. (2) Virtual Clock [7, 9] has been selected in order to study whether there is any difference in using a guaranteed rate scheduling policy such as Virtual Clock as opposed to a simple class-based policy. Figure 3 shows the topology of the MCI ISP backbone network, which we use throughout the experiments. In the experiments, all routers can act as edge routers. Following common practice, we rely on a separate routing subsystem (SPF: Shortest Path First) to determine the paths of flows.

In the simulation, requests for flow establishment form a Poisson process with rate λ, while flow lifetimes are exponentially distributed with an average lifetime of 180 seconds for each flow. Source and destination edge routers are

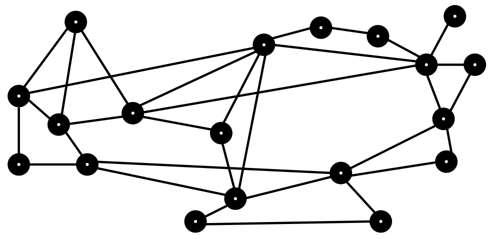


Figure 3. Simulation network topology

chosen randomly. We consider a system with two classes, real-time and non-real-time class. We assume that all flows in the real-time class have a fixed packet length of 640 bits (RTP, UDP, IP headers and 2 voice frames) [24], and a flow rate of 32 Kbps. The end-to-end delay requirement of all flows is fixed at 100ms.[4] Thus, the input traffic of each flow is constrained by a leaky bucket with parameters $T = 640$ bits and $\rho = 32$ Kbps like "Voice-over-IP" [23]. All links in the simulated network have the same capacity of 15.5 Mbps. We assume that the admission control in the Virtual Clock system allocates sufficient but not excessive bandwidth to a flow so that both delay and bandwidth requirements of the flow are met. Figure 4 shows the admission probabilities for the real-time class in the three cases as a function of arrival rates. The graphs show the results of two experiment runs with different amounts of bandwidth allocated to the real-time class traffic. In the figure, "vc_10" means the Virtual Clock admission probabilities with $\alpha = 10$, so does "vc_35" with $\alpha = 35$. By the same rule, "ds_10" stands for Differentiated Services with $\alpha = 10$, and so on. So, the upper curve is the case of 35% bandwidth allocation, and the lower one for the case of 10%. With larger bandwidth allocated to the class, we get higher admission probabilities. The practical meaning of $\lambda = 20$ is that an average of 20 flows arrive per second. Since the average life time of the flow is 180 seconds, we have 3,600 flows on average in the simulation system with $\lambda = 20$.

As can be seen, the admission probabilities of all three systems are almost identical. The reason for the similarity for the class-based system and the NetEx system lies in the fact that we limit the amount of bandwidth allocated to the real-time class, while NetEx can use connection information to compute tighter delay bounds, mostly this does not come into play, as the flow is rejected earlier due to lack of available (allocated) bandwidth. The reason for the similarity for the Virtual Clock system and the class-based system is again because the admission decision reduces to checking the availability of allocated bandwidth for the class. For flows with longer paths, the admission probability in the Virtual Clock system is actually worse. This is due to the delay-bandwidth coupling of Virtual Clock scheduling: in order to meet tighter deadlines Virtual Clock requires more bandwidth. As the arrival rate increases, the class-based system scheduling outperforms the Virtual Clock. The comparison with NetEx as a system with a similar packet scheduling policy shows that a connection-oriented admission control scheme brings little to no benefit over a class-based scheme. In addition, the comparison with the Virtual Clock system illustrates how the use of a connection oriented scheme and a sophisticated guaranteed-rate scheduling policy brings no benefit over a simple class-based scheme as well.

[4] In a real situation this value is not correct, as we only consider queueing delay for end-to-end delay deadline. There are many other factors like propagation delay, Codec (Coder and Decoder) delay or others we do not know now. More accurate values can be found in [24].

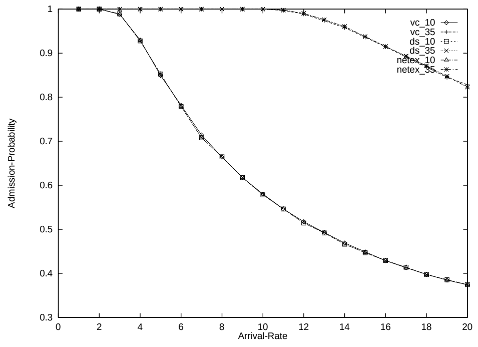


Figure 4. Admission probability comparison of DS, NetEx, and Virtual Clock Systems

Arrival Rate	Our System	Virtual Clock	NetEx
λ=2	3.95	4.58	61,752
λ=5	3.86	4.46	121,327
λ=10	4.09	4.58	312,142

Table 1. Run time overhead for three cases with varying system load (μsec)

Table 1 shows average run time overhead values in μsec with three different λ values for the three systems. As can be seen, the class-based system is the smallest overhead. Virtual Clock's overhead is close to that of the class-based system, because the admission control of the class-based system checks only if there is enough available bandwidth for the new request, while Virtual Clock's admission control computes the required bandwidth to meet the predefined end-to-end delay. So the difference is just for new bandwidth calculation in Virtual Clock. However, since both systems don't check the state of each flow, their overhead is much smaller than that of NetEx. This is because NetEx calculates the end-to-end delay bounds of all existing flows whenever a new request arrives.

6. Implementation

6.1 Motivation

The experimental results demonstrate that the proposed class based scheme is competitive with connection oriented schemes when it comes to network utilization and admission overhead Given this encouraging results, we proceed to design and implement the proposed system as part of a DS architecture.

6.2 PowerRail Routing Switch

We use PowerRail Routing Switches, a gigabit switch, produced by Packet Engines in our implementation. The switch has 8 different classes of service queues per output port, and three different ways of servicing and scheduling packets from these 8 queues: FIFO, STARVE (static priority) and WFQ (Weighted Fair Queuing). The PowerRail switch provides a series of offset mask filters (OMF) on each port which the switch uses to filter, count, prioritize, or redirect frames. Of interest to us is that the PowerRail switch provides class-based (up to eight classes) static priority scheduling. Also, it provides a series of filters, which can be used to constrain incoming traffic.

6.3 Architecture

Figure 5 illustrates the architecture of proposed system based on the PowerRail switches.

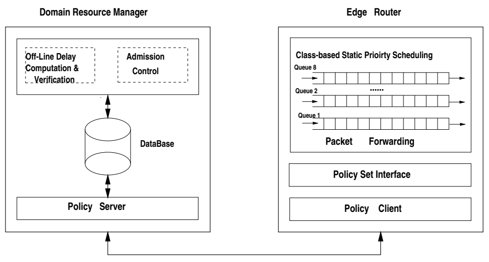


Figure 5. The architecture of proposed system

For each domain, we designate a Domain Resource Manager (DRM), which is performed after the Bandwidth Broker in [26]. The DRM has access to the whole domain topology and link capacity information. It performs the off-line delay computation and verification as well as admission control. The edge routers participate in the flow establishment. They are responsible for communicating with the DRM. For communication between edge routers and the DRM, we use the policy client-server protocol, such as COPS [25]. Upon receiving a flow admission request, the ingress router forwards it requests to the DRM. The DRM invoke its admission control function, and sends a policy (for example, the admission decision and traffic shaping policing parameters) to the edge router. The edge router will set the policies on the appropriate port via the policy-set interface. Once the flow is admitted, the edge routers will appropriately filter the incoming traffic according to the policies. For each packet passing through the filter, the priority is marked in the TOS field in the packet header and the packet is forwarded to the appropriate output links. Core routers then honor the priority in their packet forwarding scheduling.

7. Conclusions

We have proposed a scalable methodology for providing QoS-guaranteed services within a DS architecture. To the best of our knowledge, this is the first approach to guarantee real-time services with flow aggregation and no connection awareness in core routers thus achieving much lower complexities in admission control and packet forwarding.

We achieve this by using an off-line delay computation and verification step which allows to reduce the overhead at admission control while keeping admission probability and resources utilization high. Indeed, our evaluation data show our system's admission probabilities are very close to those of significantly more expensive approaches. At the same time, admission control overhead during flow establishment is very low.

We paid particular attention to the compatibility with the existing DS architecture. We are currently proceeding to design, implement, and deploy the proposed methodology with a Differentiated Service capable IP network to further illustrate its easy deployability.

We are also extending this approach to networks with multiple domains. The key issue is how to perform off-line delay computation and verification. For many applications, deterministic guarantees are not necessary [29]. The quality of IP telephony, for example, would not suffer from the underlying system providing high-quality statistical guarantees instead of deterministic guarantees. We are therefore investigating how to extend our off-line delay computation methodology to take into account statistical guarantees.

References

[1] R. Braden, D. Clark and S. Shenker, "Integrated Services in the Internet Architecture," *RFC 1633*, Jun. 1994.

[2] B. Devalla, C. Li, A. Sahoo, and W. Zhao, "Connection-oriented real-time communication for mission critical applications-an introduction to NetEx: a portable and efficient tool kit," *Aerospace and Electronics Conference*, NAECON 1997., on Pages: 698 - 707 vol.2 1997.

[3] A. Sahoo, B. Devalla, Y. Guan, R. Bettati, and W. Zhao, "Adaptive connection management for mission critical applications over ATM Networks," *International Journal of Parallel and Distributed Systems and Networks*, Special Issue On Network Architectures for End-to-end Quality-of-Service Support, to appear.

[4] Rene L. Cruz, "A Calculus for Network Delay, Part I: Network Elements in Isolation," *IEEE Transactions on Information Theory*, Vol. 37. Jan. 1991.

[5] Rene L. Cruz, "A Calculus for Network Delay, Part II: Network Analysis," *IEEE Transactions on Information Theory*, Vol. 37. Jan. 1991.

[6] Amitava Raha, Sanjay Kamat, Wei Zhao, "Guaranteeing End-to-end Deadlines in ATM Networks," *The 15th International Conference on Distributed Computing Systems*, 1995

[7] L. Zhang, "Virtual Clock: A new traffic control algorithm for packet switching networks," *ACM Transactions on Comput. Syst.*, Vol. 9. May. 1991.

[8] I. Stoica, H. Zhang, "Providing Guaranteed Services Without Per Flow Management," *SIGCOMM*, to appear in 1999.

[9] Norival R. Figueira, Joseph Pasquale, "An Upper Bound on Delay for the Virtual Clock Service Discipline," *IEEE/ACM Transactions on Networking*, Vol. 3. Aug. 1995.

[10] A. Banerjea and S. Keshav, "Queueing delays in rate controlled ATM networks," *Proceedings of Infocom'93*, 1993.

[11] J. Bennett and H. Zhang, "WF^2Q: worst-case fair weighted fair queueing," *Proc. of IEEE INFOCOM'96*, 1996.

[12] A. Dailianas and A. Bovopoulis, "Real-time admission control algorithms with delay and loss guarantees in ATM networks," In *Proceedings of INFOCOM'94*, pages 1065–1072, 1994.

[13] N. R. Figueira and J. Pasquale, "Rate-function scheduling," *Proceedings of Inforcom'97*, 1997.

[14] V. Firoiu, J. Kurose, and D. Towsley, "Efficient admission control for EDF schedulers," *Proceedings of Inforcom'97*, 1997.

[15] S. J. Golestani, "Network delay analysis of a class of fair queueing algorithms," *IEEE J. on Selected Areas in Communications*, vol. 13, no. 6, 1995.

[16] P. Goyal, S. S. Lam, and H. Vin, "Determining end-to-end delay bounds in heterogeneous networks," In *5th Int. Workshop on network and Op. Sys. support for Digital Audio and Video*, 1995.

[17] J. M. Hyman, A. A. Lazar, and G. Pacifici, "Real-time scheduling with quality of service constraints," *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 7, 1991.

[18] J. Liebeherr, D.E. Wrege, and D. Ferrari, "Exact admission control in networks with bounded delay services," *IEEE/ACM Transactions on Networking*, 1996.

[19] A. K. J. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single-node case," *IEEE/ACM Trans. Networking*, vol. 1, no. 3, 1993.

[20] A. K. J. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the multiple-node case," *IEEE/ACM Trans. Networking*, vol. 2, no. 2, 1994.

[21] L. Zhang, S. Deering, D. Estrin, S. Shenker and D. Zappala, "RSVP: a new resource reservation protocol," *IEEE Networks Magazine*, vol. 31, No. 9, pp. 8-18, September 1993.

[22] R. Bettati, D. Ferrari, A. Gupta, W. Heffner, W. Howe, M. Moran, Q. Nguyen, and R. Yavatkar, "Connection establishment for multiparty real-time communication," *Proceedings of the 5th International Workshop on Network and Operating System Support for Digital Audio and Video*, Durham, New Hampshire, Apr. 1995.

[23] F. Borgonovo, A. Capone, L. Fratta, C. Petrioli, "End-to-End Call Admission Control for Telephony Services over IP Networks," *IEEE Network and IEEE Internet Computing joint issue*, submitted.

[24] Thomas J. Kostas, Michael S. Borella, Ikhlaq Sidhu, Guido M. Schuster, Jacek Grabiec, and Jerry Mahler, "Real-Time Voice Over Packet-Switched Networks," *IEEE Network Mag*, Jan./Feb. 1998.

[25] J. Boyle, R. Cohen, D. Durham, S. Herzog, R. Rajan, A. Sastry, "The COPS (Common Open Policy Service) Protocol," *Internet-Draft*, Feb. 1999.

[26] K. Nicols, V. Jacobson, L. Zhang, "A Two-bit Differentiated Services Architecture for the Internet," *Internet-Draft*, Nov. 1997.

[27] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, "An Architecture for Differentiated Service," *RFC 2474*, Dec. 1998.

[28] Y. Bernet et al., "A Framework for Differentiated Services," *Internet-Draft*, IETF, Feb. 1999.

[29] Zhang, Z.-L., Towsley, D., and Kurose, J., "Statistical analysis of the generalized processor sharing scheduling discipline," *IEEE Journal of Selected Areas in Communications*, 13(6):1071-1080, Aug. 1995.

[30] B. Choi, D. Xuan, C. Li, R. Bettati and W. Zhao, "Technical Report TR00-007," Texas A&M University, Department of Computer Science. Jan. 2000.