Scalable QoS Guaranteed Communication Services for Real-Time Applications

Byung-Kyu Choi, Dong Xuan, Chengzhi Li, Riccardo Bettati, Wei Zhao
Department of Computer Science
Texas A&M University
College Station, TX 77843-3112, USA
{choib, dxuan, chengzhi, bettati, zhao}@cs.tamu.edu
Tel:1-409-845-5469, Fax:1-409-847-8578

Abstract

In this paper, we propose a system of admission control and packet forwarding scheme to improve scalability of networks while guaranteeing end-to-end delay 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. 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. Our results therefore support the claim from the DS architecture literature that scalablity can be achieved through flow aggregation without sacrificing resource utilization and with significant reduction in run time overhead.

Key Words: admission control, end-to-end deadline, real-time services, QoS guarantees, and Differentiated Services.

Contents

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* service 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. It is not a real-time service. 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.

A systematic approach based on the connection admission control and packet scheduling was proposed within the Integrated Services (IS) architecture of the IETF [1]. In Integrated Services, each connection is strictly controlled both by admission control at connection establishment time and packet scheduling during the lifetime of the connection. At establishment time, the necessary resources for the new connection are allocated, and during the lifetime, the connection is policed to ensure that abnormal behavior of a connection does not affect other connections. Therefore, it can support real-time services. This necessitates that information about every connection is kept by each node along the path for admission control and packet forwarding purposes.

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. The IS architecture therefore does not provide scalable QoS guaranteed services, and the lack of scalability is due to overhead both at connection establishment and during the connection lifetime.

This lack of scalability is, to a large extent, being addressed within the Differentiated Services architecture [27, 28, 29]. 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 network scalable regardless of the number of flows. A number of details about how to realize and deploy Differentiated Services are still being discussed.

The objective of this work is to further improve the scalability of network for QoS guaranteed services by considering both admission control and packet forwarding together within a DS architecture. We make admission control scalable by using off-line end-to-end delay computation. In data forwarding, we adopt an efficient class-based static priority packet scheduling. With these two schemes, we show in Section 6 that our system provides admission probabilities and resource utilizations that are very similar to those of other existing approaches. This is particularly interesting, since the run-time overhead of our approach is very low, compared to the other previous approaches, regardless of whether they rely on each flow's state or not.

[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.

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 provide our system including off-line end-to-end delay calculation, admission control, and class-based static priority scheduling scheme. We also present the end-to-end delay formula with detailed proof in Appendix A. 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 anything on end-to-end delay since the delay purely depends on the dynamic load of network links. A number of schemes proposed are the ways to deploy network services that provide QoS guarantees. 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 [27, 28, 29] takes scalability issues into account.

Among the QoS guarantee components, our main interests are admission control and packet scheduling, in terms of end-to-end delay guarantees for real-time services. As far as these two technical aspects are concerned, much research has been proposed in the last decade. First, with some traffic models, a number of 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, 30]. Finally admission control has been investigated as well [12, 14, 18]. Much of this work addresses admission control and packet scheduling. Some deal with meeting end-to-end delay requirements.

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 extentions of Cruz's methodology for delay analysis [4, 5]. Essentially, NetEx provides connection-oriented real-time communication service by relying on a strict admission control [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 connections increases, the admission control needs proportionally more time to make admission decision. Because they rely on each connection's information. Therefore, they provide a real-time service, but are not scalable to a large scale internetwork due to their run-time overhead in admission control and packet scheduling.

Recently, a number of efforts have focused on reducing the negative effect on scalability caused by connection awareness in signaling protocols. The work in [8] is a good example of such efforts. 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 for admission control and packet scheduling, 2) dynamic packet status information is carried over by each packet. With this information, the core router estimates each flow's dynamic information such as end-to-end delay. So it achieves that the core router doesn't need to maintain per flow information anymore but infers it based on information carried with the packet. 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. We believe that to be scalable, the core router should serve packets with class-based information only, thus rely on flow aggregation.

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 [28]. By having off-line end-to-end delay computation, our admission control complexity is definitely lower than those of [2, 3, 4, 5, 6, 8, 22], while guaranteeing end-to-end delay deadlines for all flows. Also, we argue that our system's runtime overhead should be lower than that of [8], because we use class-based static priority scheduling. To the best of our knowledge, this is the first proposal addressing end-to-end delay guarantee and scalable run time overhead within a DS model.

3 Our Models

In this section, we describe the model and define the terminology (network, link server, 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. 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.

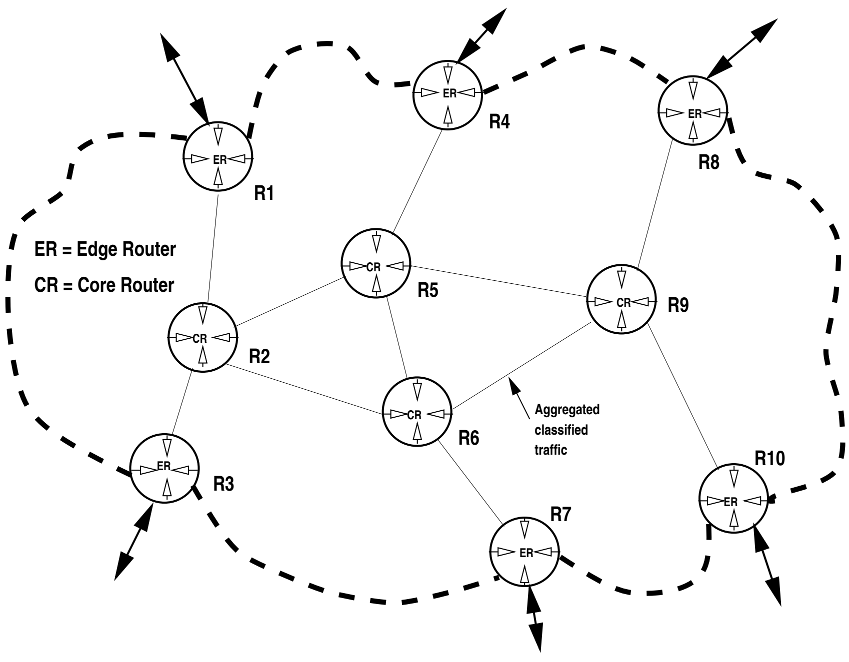

Figure 1: A simple illustration of our domain

Routers and Links There are two kinds of routers in a domain: edge router and core router as shown in Figure 1. 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 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. In the following discussion, we treat output links as link servers, or servers, with a given capacity and scheduling policy. The network is then modeled as a network of link servers. We denote link server i as S_i. There are total M servers.

Paths A packet is transmitted along a path. The path is a list of nodes. Equivalently, a path can also be denoted as a list of link servers. For examle, 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 which the path traverses. In the rest of this paper, the path means a list of link servers rather than nodes unless otherwise mentioned. 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 to 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 $\min\{C * I, T_i + \rho_i * I\}$.*

Here, the notations are:

- T_i is the burst size of source traffic of the flow in Class i.
- ρ_i is the average rate of source traffic of the flow 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 use *traffic functions* and their time independent counterpart *traffic contraint functions* to characterize traffic of any server in the network.

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-independant 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 $I > 0$,*

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

The QoS requirement of flows is specified on a class-by-class basis as well. To our purpose, we define the deadline requirement of traffic in Class i to be D_i. As no distinction is made between flows belonging to the same class, all flows in the same class receive the same level of delay guarantees. In the following, we will use $d_{k,i}$ to denote the queueing delay suffered by Class i traffic at Sever k.

Following the DS model, at each link server, certain percentage of bandwidth is reserved for individual traffic classes. α_i denotes the percentage of bandwidth reserved for Class i. It is the responsibility of the admission control to ensure that the bandwidth usage of individual classes does not go beyond the reserved portion. This is necessary to provide isolation among classes and hence to guarantee end-to-end delays to the flows in each class.

4 Scalable QoS Guaranteed System

4.1 Overview

While our declared objective is to provide scalability for QoS guaranteed system, other aspects must be considered as well. In this section we propose a methodology to provide QoS guaranteed real-time communication over a Differentiated Services network with the following main objectives in mind:

1. 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.
2. 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. The result is a system with high probability of admitting a new flow if resources are available.
3. Compatibility: For practical purposes, our system is compatible with current industrial practice. We avoid modifications to communication infrastructure, such as the modifications to the architecture of routers, signaling protocols, or to the packet format, etc.

While all these objectives are important and critical, they may conflict with each other. For example, in order to improve scalability and reduce overhead, the admission probability may be compromised. On the other hand, using the existing link scheduling algorithms may limit our capability to improve the scalability.

In our system, we adopt the class-based static priority algorithm to ensure the scalability in packet forwarding. We utilize off-line delay computation technology to reduce the overhead at admission control while keeping resources utilization and admission probability high.

Our system consists of three major modules:

1. Off-line delay computation and verification
   Here, we use an ingenious delay calculation method to estimate the delay upper bound for 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 works during initial network configuration, or during modification of service level agreement among the communication service providers and their customers.

2. 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.

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

Before we describe the above modules in detail, we would like to introduce a delay computation formula. This formula will be used in the module for off-line delay computation and verification.

4.2 Recursive Formula for Delay Computation

In a computer network there may exist a very large numbers, say millions, of traffic flows simultaneously. To provide QoS for communication at such a scale within a DS architecture, all these traffic flows are categorized and aggregated into a small number of classes. 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 in the network 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*

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

and

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

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

For a proof of theorem 1, see Appendix A.

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:

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

We would like to make some observations on Theorem 1.

- After observing Equation (2), we note 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 $\vec{d_l}, l = 1, \cdots, i-1$, we can denote

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

  and

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

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

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

  Since the unknown vector $\vec{d_i}$ appears on both sides of Equation (7), we need an iterative procedure, which will be discussed in next section, to find the solution.

- 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 (2) is independent from this kind of information. (2) does depend on T, ρ, α, and N. However, the values of these parameters are available at the time when the system is (re-)configured.

  This observation suggests that an off-line delay computation and verification is possible, as we are going to discuss in the next sub-section. In Appendix A, we will present the details of deriving (2) where we are able to deliberately eliminate the run-time status information. In Section 5, we are going to demonstrate that our new delay computation method does not compromise at all on admissibility of flows, in comparison with other systems where the delay computation has to take explicitly the runtime status into account and has to be performed during flow admission, rather than off-line as we do here.

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, as long as the portion of bandwidth allocated to the individual classes are not exceeded. 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. An actual delay analysis during flow establishement is then not necessary anymore.

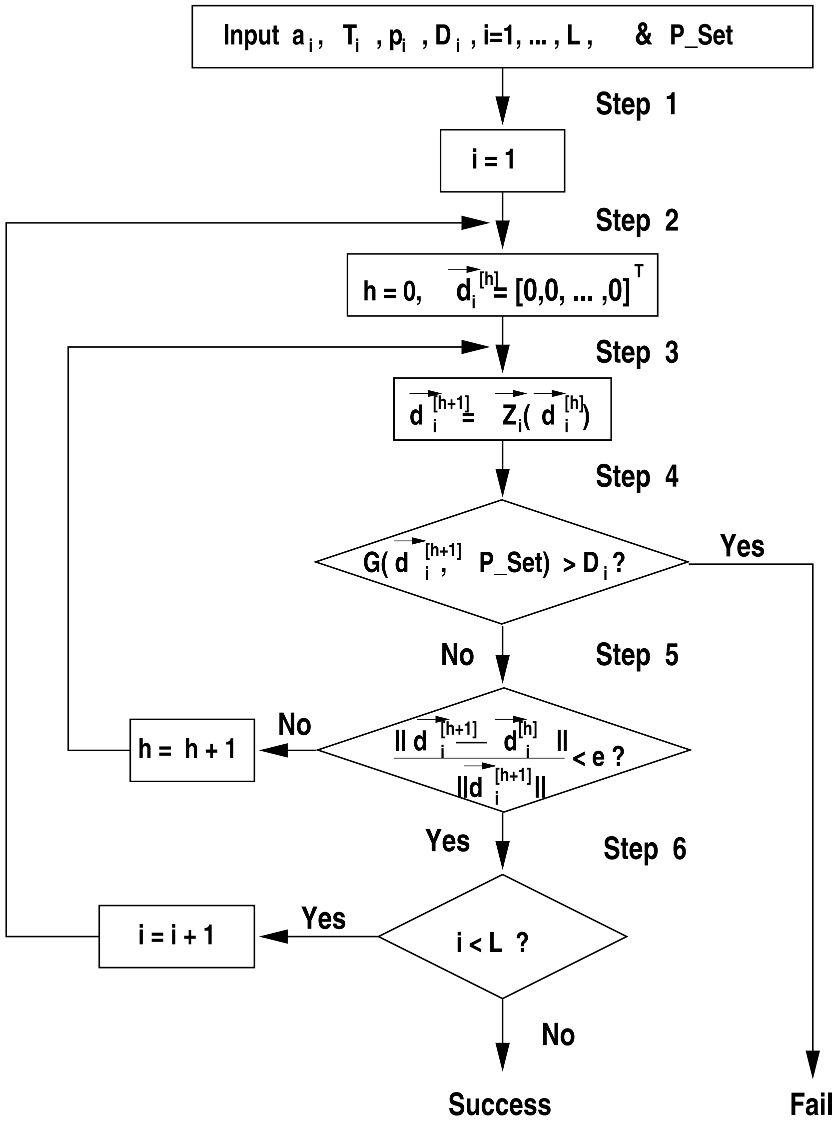

Figure 2: Algorithm for Off-line Delay Computation and Verification

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 in the order of decreasing priority. 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. In this case the network needs to be reconfigured, for example, by changing the routing policy, or by re-allocating portion of link bandwidths to classes.

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.

The correctness of the algorithm is given by the following two theorems.

THEOREM 2 *The algorithm for the off-line end-to-end delay computation and verification will terminate.*

This theorem guarantees the work flow will not loop infinitely. Since the number of classes (L) is limited, the outer loop will execute a finite numbers of times. Thus we need to only prove that inner loop for computing delays for classes i will stop after some finite number of iterations. The proof of this theorem is in Appendix B.

THEOREM 3 *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] $\|[d_1, \cdots, d_n]^\top\| = \max_{i=1}^n |d_i|$

Proof of Theorem 3 This theorem follows directly from the algorithm. If the theorem is untrue then there will be one path, along which the end-to-end delay for one class (say Class i) is larger than its deadline D_i under the condition that the bandwidth is available for for Class i. However, according to the condition in Step 4 of the algorithm, the algorithm must return $FAILURE$. Q.E.D

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. The run time overhead of the admission control is therefore reduced considerably, as opposed to systems at establishment time delay calculation, and hence the scablability of admission control is significantly improved.

4.4 Admission Control

During flow establishment, the admission control module must make sure that the QoS requirement can be satisfied under the current network load, and that the network resources are not over-committed. In our case this means that the delay requirements of the new and of existing flows are not violated.

If 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. We showed in Theorem 3 that no flow can miss its delay requirements, as long as no bandwidth allocation of a class is exceeded. 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.

Class-based static priority scheduling integrates well with the DS architecture, as the priority of a packet can be defined in the specific field TOS (Type of Service) of the IP packet heads by the ingress router when the packet enters a domain [28]. This field is used for packet forwarding by the core router.

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. This allows to study the effect of connection-based versus class-based admission control with the same packet forwarding scheme (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, such as class-based static priority scheduling. In these experiments, we compare admission probability and run-time overhead of the three systems.

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 to determine the paths of flows. As routing should not affect the result, we use a shortest-path routing policy throughout the experiments.

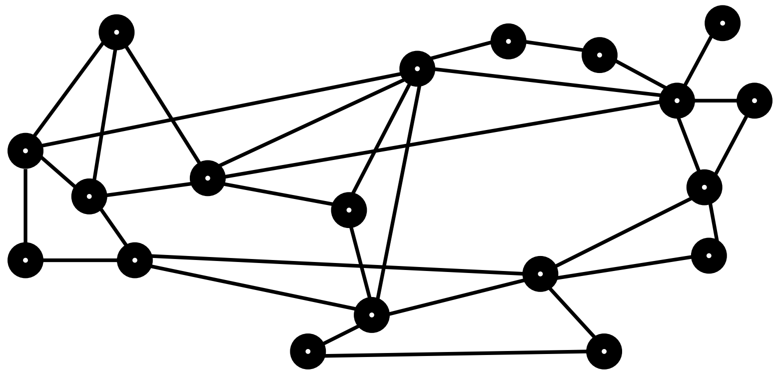

Figure 3: Simulation network topology

We simulate the admission control behavior in the system by simulating flow requests and establishments at varying rates with a constant average flow lifetime. 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 chosen randomly and the flow is set up along the shortest path.

We consider a system with two classes. One is real-time class. Another is 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) [25], 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. This kind of QoS requirements is similar to "Voice-over-IP" [24]. All links in the simulated network have the same capacity of 15.5 Mbps. The reason for the low capacity of 15.5 Mbps is to reduce simulation time. With higher link capacity (e.g., 155 Mbps, 622 Mbps and 1 Gbps etc.), we expect the similar relative performance.

The admission control for the Virtual Clock system is relatively simple. With a Virtual Clock system, the delay bound at a server is tightly coupled with the bandwith reserved. 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-tmie 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 for the real-time class traffic, and the lower one for the case of 10%. As it should be, 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 at the

[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 [25]. We use this value is just for the reference. With other values, we would see similar results.

system 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 $\lambda = 20$.

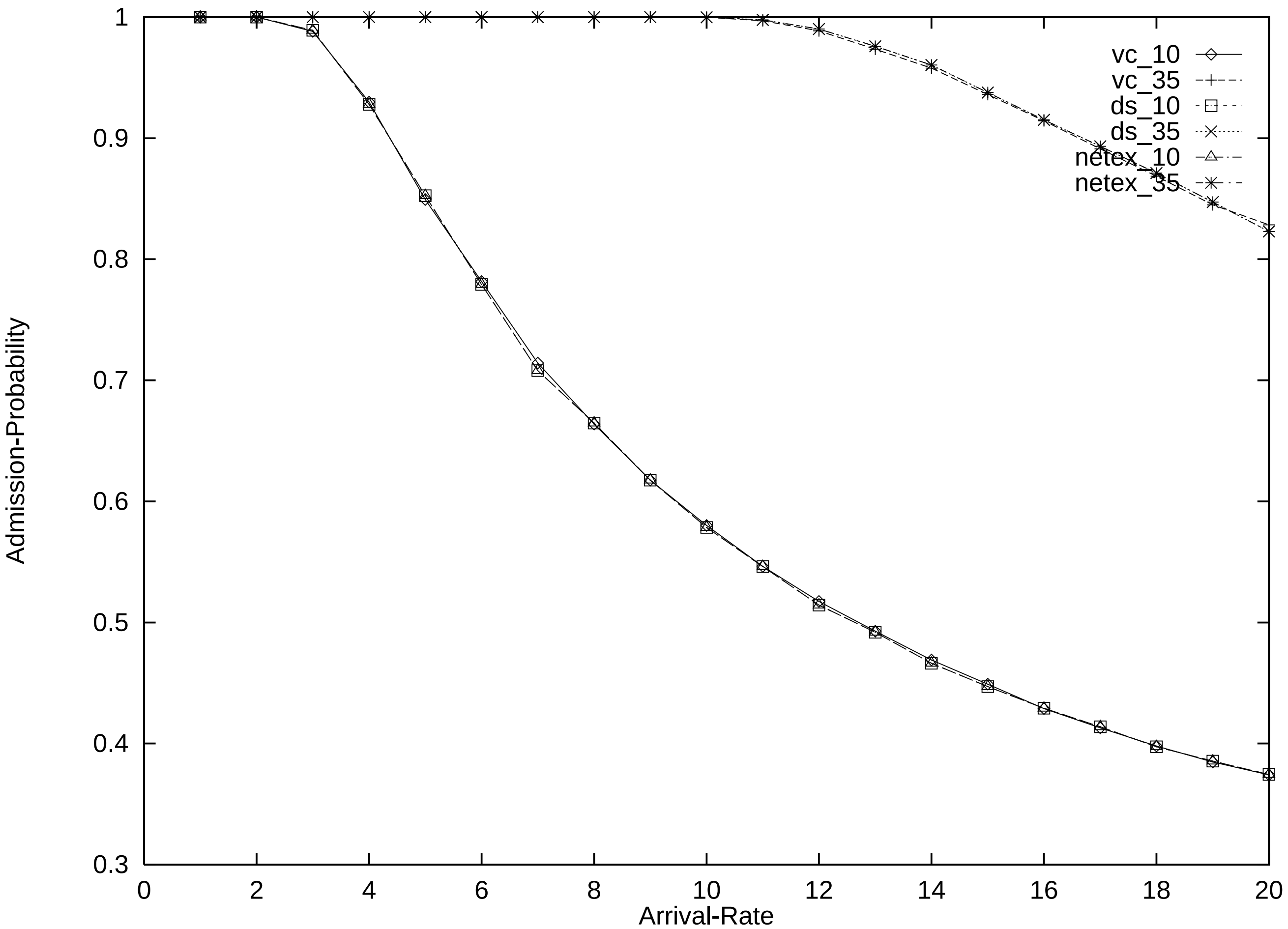

Figure 4: Admission Probability Comparison of DS, NetEx, and Virtual Clock Systems

As can be seen, the admission probabilities of all three systems are almost identical. The reason for the similarity of the results 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 of the results 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. Because the establishment of some flows using the Virtual Clock system requires more bandwidth than the class-based system, as the arrival rate increases, the class-based system scheduling outperforms the Virtual Clock. For a link with 8 hops for example, the Virtual Clock system needs roughly 1.6 times more bandwidth than the class-based system. In this way allocated bandwidth gets used up quite quickly, and admission probability suffers.

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)

These experimental results show that a simple class-based system provides admission probabilities that are very similar to those of significantly more expensive approaches. 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.

We measured the run time overhead in admission control for comparison. 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. While the admission overhead for the class-based and the Virtual Clock system remain constant with increasing system load, overhead of the NetEx system grows linearly with the number of connections in the system. This data illustrates that the class-based system outperforms both the Virtual Clock and NetEx systems in terms of run time overhead of admission control.

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. In addition, they indicate that admission overhead during flow establishment is very low. Given this encouraging results, we proceed to design and implement the proposed system as part of a DS architecture. Recall that our system consists of three modules: off-line delay computation and verification, admission control, and packet forwarding. These modules have two basic requirements: first, the incoming traffic is constrained by a leaky bucket at the edge router; second, both the edge routers and the core routers must provide a class-based static priority scheduling police at output links.

6.2 PowerRail Routing Switch

We use PowerRail Routing Switches produced by Packet Engines in our implementation. The PowerRail Switch is a gigabit switch and provides many capabilities including multiple Quality of Service queues, TOS-based routing etc. 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. The OMF operates based on the policies set by the administrator. These policies can apply offset mask filtering to a specific application.

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. Hence, the switches are well suited to act as the edge routers and core routers.

6.3 Architecture

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

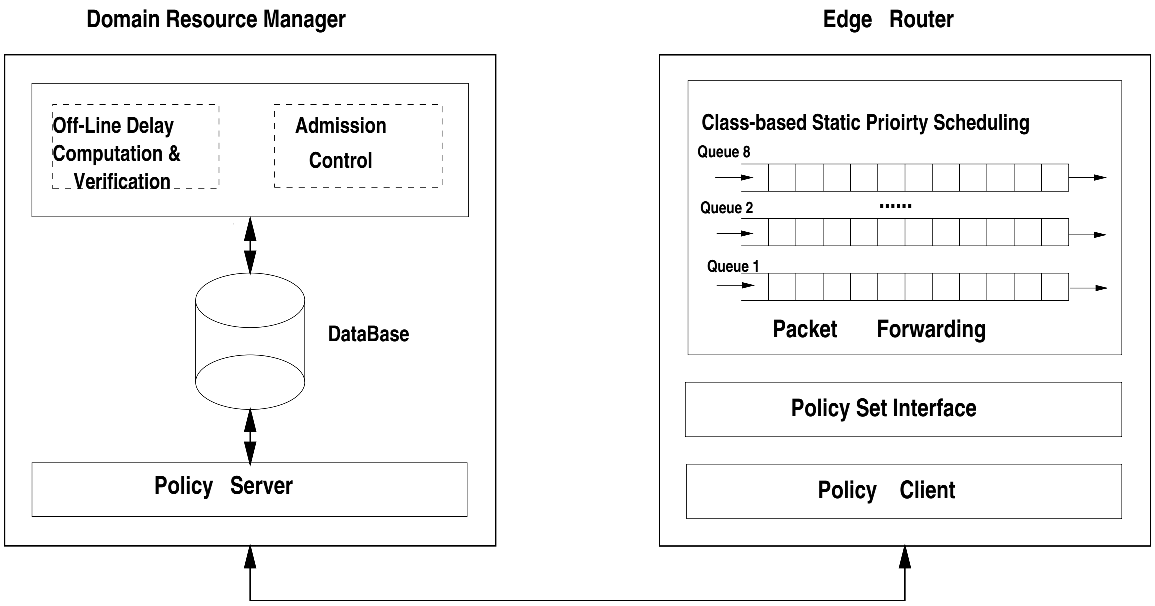

Figure 5: The Architecture of Proposed System

For each domain, we designate a Domain Resource Manager (DRM), which is performed after the Bandwidth Broker in [27]. 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 off-line delay computation and verification is performed during initialization the network, or during the modification of SLAs between the service provider and its customers.

In our implementation, both the edge and core routers are PowerRail Routing Switches. In addition to forwarding packets, 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 [26].

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 just set. 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 schelduling.

The information maintained in the DRM is limited to one counter for each class per link server in the domain, requiring very little memory storage. Also, in addmission control the DRM need only check the bandwidth usage along the path of the flow in its local database, making the run time overhead very low. Based on this consideration, we believe that the DRM will not be a bottleneck for the performance of network.

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 ovehead during flow establishment is very low. Our results therefore support the claim from the DS architecture literature that scalablity can be achieved through flow aggregation without sacrificing resource utilization and with significant reduction in run time overhead.

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. Depending on the details of the inter-domain service level agreement, a variety of approaches could be used. For example, we could partition the end-to-end delay deadlines into sub-deadlines, one for each domain, and use traffic shapers in the edge routers at the intermediate domains. The methodology we develop in this paper can then be applied to each of individual domain.

For many applications, deterministic quarantees are not necessary [31]. 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 quarantees.

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 VirtualClock Service Discipline," *IEEE/ACM Transactions on Networking*, Vol. 3. Aug. 1995.

[10] A. Banerjea and S. Keshav, "Queueing delays in rate controlled ATM networks," *Proceedinds of Inforcom'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 Seleted 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 aApproach 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 aApproach 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] C. Li, R. Bettati, and W. Zhao, "Static priority scheduling for ATM networks," *Proceedings of the 18th IEEE Real-Time Systems Symposium*, 1997.

[24] 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.

[25] 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.

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

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

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

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

[30] Zhang, J.-L., Liu, Z., and Towsley, D., "Closed-form deterministic end-to-end performance bounds for the generalized processor sharing scheduling discipline," *Journal of Combinatorial Optimization*, 1996.

[31] 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.

A Proof of Theorem 1

We will first present a model for an arbitrary server and establish several lemmas. Then we formally prove the Theorem 1.

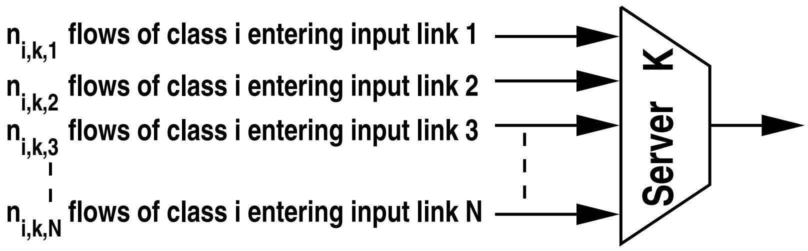

Figure 6: Server k

Consider Server k as shown in Figure 6. Assume that at an arbitrary time moment, $n_{i,k,j}$ is the total number of traffic flows of Class i passing through input link j on Server k. Recall that $Y_{i,k}$ is the worst case queuing delay suffered by the traffic of Class i before arriving at Server k. $F_{i,k,j}(I)$ is the constraint function for the aggregated traffic of class i on input link j of Server k.

LEMMA 1 *The aggregated traffic of Class i on input link j of Server k is constrained by*

$$F_{i,k,j}(I) = \begin{cases} C * I, & I \leq \tau_{i,k,j}, \\ n_{i,k,j} * T_i + n_{i,k,j} * \rho_i * Y_{i,k} + n_{i,k,j} * \rho_i * I, & \tau_{i,k,j} \leq I, \end{cases} \tag{8}$$

where

$$\tau_{i,k,j} = \frac{n_{i,k,j} * T_i + n_{i,k,j} * \rho_i * Y_{i,k}}{C - n_{i,k,j} * \rho_i}. \tag{9}$$

Proof of Lemma 1 Consider the x^{th} flow that arrives at input link j of Server k and belongs to Class i. Let $H_x(I)$ be its source traffic constraint function. By Definition 1,

$$H_x(I) = \min\{C * I, T_i, +\rho_i * I\}. \tag{10}$$

Let Y_x be the total worst case queuing delay experienced by the packets from this flow before arriving at Server k. We note

$$Y_x \leq Y_{i,k}. \tag{11}$$

Let $H_x^*(I)$ be the traffic constraint function of the x^{th} flow at input link j of Server k. According to Theorem 2.1 in [4],

$$H_x^*(I) \leq H_x(I + Y_x). \tag{12}$$

Let t be an arbitrary time moment and consider time interval $[t, t+I)$. According to Definition 2, the total amount of traffic from the x^{th} flow arriving during this time interval is given by

$$\begin{aligned} f_{i,k,j}(t+I) - f_{i,k,j}(t) &\leq \sum_{x=1}^{n_{i,k,j}} H_x^*(I) \\ &\leq \sum_{x=1}^{n_{i,k,j}} H_x(I + Y_x) \\ &\leq \sum_{x=1}^{n_{i,k,j}} H_x(I + Y_{i,k}). \end{aligned} \tag{13}$$

Substituting (10) into (13), we have

$$f_{i,k,j}(t+I) - f_{i,k,j}(t) \leq \min\{n_{i,k,j} * C, \quad n_{i,k,j} * T_i + n_{i,k,j} * \rho_i Y_{i,k} + n_{i,k,j} * \rho_i * I\}. \tag{14}$$

On the other hand, the total amount of traffic that can be transmitted over input link j of Server k during any time interval of length I is constrained by the link capacity C. Therefore, we have

$$f_{i,k,j}(t+I) - f_{i,k,j}(t) \leq C * I. \tag{15}$$

Synthesizing (14) and (15), we have

$$\begin{aligned} f_{i,k,j}(t+I) - f_{i,k,j}(t) &\leq \min\{C * I, n_{i,k,j} * T_i + n_{i,k,j} * \rho_i Y_{i,k} + n_{i,k,j} * \rho_i * I\} \\ &= \begin{cases} C * I, & I \leq \tau_{i,k,j}, \\ n_{i,k,j} * T_i + n_{i,k,j} * \rho_i * Y_{i,k} + n_{i,k,j} * \rho_i * I, & \tau_{i,k,j} \leq I, \end{cases} \end{aligned} \tag{16}$$

where

$$\tau_{i,k,j} = \frac{n_{i,k,j} * T_i + n_{i,k,j} * \rho_i * Y_{i,k}}{C - n_{i,k,j} * \rho_i}. \tag{17}$$

Due to the arbitrariness of the time moment t, Lemma 1 then follows from Definition 3. Q.E.D

Note that $F_{i,k,j}$ is a two-piece linear function of I. The value $\tau_{i,k,j}$ identifies the intersection of the two linear segments, and is called the *flex point* of $F_{i,k,j}(I)$. Let

$$\mathcal{F}_{i,k}(I) = \sum_{j=1}^{N} F_{i,k,j}(I). \tag{18}$$

In general, $\mathcal{F}_{i,k}(I)$ will have N flex points. As we will see, the maximum flex point of $\mathcal{F}_{i,k}(I)$ plays an important role when we compute the queuing delay. The next lemma provides a delay formula for the traffic of Class i at Server k once the input traffic constraint functions are given.

LEMMA 2 *If*

$$F_{l,k,j}(I) = \begin{cases} C * I, & I \leq \tau_{l,k,j}, \\ A_{l,k,j} + B_{l,k,j} * I, & \tau_{l,k,j} \leq I. \end{cases} \tag{19}$$

for $l = 1, \cdots, i$, *where*

$$\tau_{l,k,j} = \frac{A_{l,k,j}}{C - B_{l,k,j}} \tag{20}$$

then

$$d_{i,k} = \frac{\sum_{l=1}^{i}\sum_{j=1}^{N} A_{l,k,j} + (\sum_{l=1}^{i}\sum_{j=1}^{N} B_{l,k,j} - C)\max_{j=1}^{N}\{\tau_{i,k,j}\}}{C - \sum_{l=1}^{i-1}\sum_{j=1}^{N} B_{l,k,j}}. \quad (21)$$

Sketch of the proof of Lemma 2 Following [23], we have the following formula that indicates how long an newly arriving packet of Class i can be delayed at Server k, assuming a class-based static priority scheduling discipline is used:

$$d_{i,k} = \frac{1}{C}\max_{I>0}(\sum_{l=1}^{i-1}\mathcal{F}_{l,k}(I + d_{i,k}) + \mathcal{F}_{i,k}(I) - C * I). \quad (22)$$

Substituting (18) into (22) with certain algebra manipulations [23], we can then derive (21).

Q.E.D

We denote

$$T(n_{i,k,1}, n_{i,k,2}, \cdots, n_{i,k,N}) = \max_{j=1,\cdots,N}\{\tau_{i,k,j}(n_{i,k,j})\}, \quad (23)$$

where $\tau_{i,k,j}$ is defined in (18). Recall that α_i is the portion of bandwidth of each server reserved for traffic of Class i. Let M_i be the total number of the flows of Class i supported by Server k. That is,

$$M_i = \sum_{j=1}^{N} n_{i,k,j}. \quad (24)$$

It is obvious that $M_i \leq \frac{\alpha_i C}{\rho_i}$. For a given M_i, we have the following lemma which will be useful in deriving our queuing delay formula.

LEMMA 3

$$\min_{\sum_{j=1}^{N} n_{i,k,j} = M_i} T(n_{i,k,1}, n_{i,k,2}, \cdots, n_{i,k,N}) = \frac{\lceil \frac{M_i}{N} \rceil (T_i + \rho_i * Y_{i,k})}{C - \lceil \frac{M_i}{N} \rceil \rho_i}. \quad (25)$$

Proof of Lemma 3 Substituting (17) into (23), we have

$$\begin{aligned} T(n_{i,k,1}, n_{i,k,2}, \cdots, n_{i,k,N}) &= \max_{j=1,\cdots,N}\{\tau_{i,k,j}(n_{i,k,j})\} \\ &= \max_{j=1,\cdots,N}\{\frac{n_{i,k,j}(T_i + \rho_i Y_{i,k})}{C - n_{i,k,j} * \rho_i}\}. \quad (26) \end{aligned}$$

Because $f(x) = \frac{x(T_i + \rho_i * Y_{i,k})}{C - x\rho_i}$ is a monotonically increasing function of variable x, we have

$$T(n_{i,k,1}, n_{i,k,2}, \cdots, n_{i,k,N}) = \frac{n^*_{i,k}(T_i + \rho_i * Y_{i,k})}{C - n^*_{i,k}\rho_i}, \quad (27)$$

where $n^*_{i,k} = \max\{n_{i,k,1}, n_{i,k,2}, \cdots, n_{i,k,N}\}$. Without loss of generality, in the rest of this proof, we assume that $n_{i,k,1} \leq n_{i,k,2} \leq \cdots \leq n_{i,k,N} = n^*_{i,k}$.

What this lemma says is that if a sequence of integers $n_{i,k,1}, n_{i,k,2}, \cdots, n_{i,k,N}$, satisfies

$$\sum_{j=1}^{N} n_{i,k,j} = M_i, \tag{28}$$

and minimizes $T(n_{i,k,1}, n_{i,k,2}, \cdots, n_{i,k,N})$, then

$$n^*_{i,k} = \lceil \frac{M_i}{N} \rceil. \tag{29}$$

Let us assume that this is not true. That is, (29) does not hold. We have three cases to deal with.

- *Case 1:* $n^*_{i,k} < \lceil \frac{M_i}{N} \rceil = \lfloor \frac{M_i}{N} \rfloor$. We have

$$\sum_{j=1}^{N} n_{i,k,j} \leq N n^*_{i,k} < N \lfloor \frac{M_i}{N} \rfloor < M_i. \tag{30}$$

  This is impossible by (28).

- *Case 2:* $n^*_{i,k} < \lceil \frac{M_i}{N} \rceil$ and $\lceil \frac{M_i}{N} \rceil > \lfloor \frac{M_i}{N} \rfloor$. That is $n^*_{i,k} \leq \lfloor \frac{M_i}{N} \rfloor$. We have

$$\sum_{j=1}^{N} n_{i,k,j} \leq N n^*_{i,k} \leq N \lfloor \frac{M_i}{N} \rfloor < M_i. \tag{31}$$

  This is impossible either in accordance with (28).

- *Case 3:* $n^*_{i,k} > \lceil \frac{M_i}{N} \rceil$. Now let us consider another sequence of integers $\hat{n}_{i,k,1}, \hat{n}_{i,k,2}, \cdots, \hat{n}_{i,k,N}$, where

$$\hat{n}_{i,k,j} = \begin{cases} \lfloor \frac{M_i}{N} \rfloor, & 1 \leq j \leq K, \\ \lceil \frac{M_i}{N} \rceil, & K < j \leq N, \end{cases} \tag{32}$$

  where $K = N \lceil \frac{M_i}{N} \rceil - M_i$. It is easy to verify that $\sum_{j=1}^{N} \hat{n}_{i,k,j} = M_i$ and

$$\frac{\hat{n}_{i,k,N}(T_i + \rho_i * Y_{i,k})}{C - n_{i,k,N}\rho_i} < \frac{n^*_{i,k}(T_i + \rho_i * Y_{i,k})}{C - n^*_{i,k}\rho_i}. \tag{33}$$

  This is a contradiction because the sequence $n_{i,k,1}, \cdots, n_{i,k,N}$ minimizes $T(n_{i,k,1}, \cdots, n_{i,k,N})$.

Hence,

$$n^*_{i,k} = \lceil \frac{M_i}{N} \rceil. \tag{34}$$

Q.E.D

Following Lemma 3, we have, if $M_i = \frac{\alpha_i C}{\rho_i}$ and $\sum_{j=1}^{N} n_{i,k,j} = M_i$,

$$\begin{aligned} T(n_{i,k,1}, n_{i,k,2}, \cdots, n_{i,k,N}) &\geq \frac{\lceil \frac{M_i}{N} \rceil (T_i + \rho_i * Y_{i,k})}{C - \lceil \frac{M_i}{N} \rceil \rho_i} \\ &= \frac{\lceil \frac{1}{N} \frac{\alpha_i C}{\rho_i} \rceil (T_i + \rho_i * Y_{i,k})}{C - \lceil \frac{1}{N} \frac{\alpha_i C}{\rho_i} \rceil \rho_i} \\ &\geq \frac{\frac{1}{N} \frac{\alpha_i C}{\rho_i} (T_i + \rho_i * Y_{i,k})}{C - \frac{1}{N} \frac{\alpha_i C}{\rho_i} \rho_i} \\ &= \frac{\alpha_i}{\rho_i} \frac{(T_i + \rho_i * Y_{i,k})}{N - \alpha_i}. \end{aligned} \tag{35}$$

LEMMA 4 *The worst case queuing delay is experienced by the traffic of Class i at Server k when M_i reaches its upper bound $\frac{\alpha_i C}{\rho_i}$.*[5]

Proof of Lemma 4 Since

$$d_{i,k} \quad = \quad \frac{1}{C}\max_{I>0}(\sum_{l=1}^{i-1}\mathcal{F}_{l,k}(I+d_{i,k})+\mathcal{F}_{i,k}(I)-C*I), \tag{36}$$

we know that the larger $\mathcal{F}_{i,k}(I)$, the larger $d_{i,k}$. Furthermore, since $\mathcal{F}_{i,k}(I)$ is the aggregated traffic of Class i at Server k, we know that the larger M_i, the larger $\mathcal{F}_{i,k}(I)$. Therefore, when $M_i = \frac{\alpha_i C}{\rho_i}$, the traffic of Class i will experience the worst case queuing delay at Server k. Q.E.D

Now we can prove the main result in this paper.

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

$$d_{i,k} \quad = \quad \frac{\sum_{l=1}^{i}(T_l+\rho_l*Y_{l,k})\frac{\alpha_l}{\rho_l}+(\sum_{l=1}^{i}\alpha_l-1)\frac{\alpha_i(T_i+\rho_i Y_{i,k})}{\rho_i(N-\alpha_i)}}{1-\sum_{l=1}^{i-1}\alpha_l}. \tag{37}$$

and

$$Y_{l,k} = \max_{Path\in S_{l,k}}\sum_{j\in Path} d_{l,j}, \tag{38}$$

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

Proof of Theorem 1 Recall that α_l is the portion of bandwidth at Server k that is reserved for Class l traffic and $M_l = \sum_{j=1}^{N} n_{l,k,j}$ is the total number of traffic flows in Class l traversing Server k. Now consider (21). According to Lemma 1, parameters $A_{l,k,j}$ and $B_{l,k,j}$, $l = 1,\cdots,L$, in (21) are given as follows:

$$A_{l,k,j} \quad = \quad n_{l,k,j}*(T_l+\rho_l*Y_{l,k}), \tag{39}$$

$$B_{l,k,j} \quad = \quad n_{l,k,j}*\rho_l. \tag{40}$$

Therefore, we have

$$\begin{aligned}\sum_{l=1}^{i}\sum_{j=1}^{N}A_{l,k,j} \quad &= \quad \sum_{l=1}^{i}\sum_{j=1}^{N}n_{l,k,j}*(T_l+\rho_l*Y_{l,k}) \\ &= \quad \sum_{l=1}^{i}(T_l+\rho_l*Y_{l,k})\sum_{j=1}^{N}n_{l,k,j} \\ &= \quad \sum_{l=1}^{i}(T_l+\rho_l*Y_{l,k})M_l \\ &\leq \quad \sum_{l=1}^{i}(T_l+\rho_l*Y_{l,k})\frac{\alpha_l C}{\rho_l}, \end{aligned} \tag{41}$$

[5] In general, $\frac{\alpha_i C}{\rho_i}$ is not necessarily an integer. However, in a modern practical system $\alpha_i * C$ is very large in comparison with ρ_i. For example, if we consider a gigabit router, $C = 10^9 bits/sec$. For voice traffic $\rho_i = 3.2\times 10^4 bits/sec$, if $\alpha_i = 15\%$, $\frac{\alpha_i C}{\rho_i} = 4687.5$. Therefore, in order to simplify notation, we assume that $\lfloor\frac{\alpha_i C}{\rho_i}\rfloor \approx \frac{\alpha_i C}{\rho_i}$.

and

$$
\begin{aligned}
\sum_{l=1}^{i}\sum_{j=1}^{N} B_{l,k,j} - C &= \sum_{l=1}^{i}\sum_{j=1}^{N} n_{l,k,j} * \rho_l - C \\
&= \sum_{l=1}^{i} \rho_l \sum_{j=1}^{N} n_{l,k,j} - C \\
&= \sum_{l=1}^{i} \rho_l M_l - C \\
&\leq \sum_{l=1}^{i} \rho_l \frac{\alpha_l C}{\rho_l} - C \\
&= (\sum_{l=1}^{i} \alpha_l - 1)C. \qquad (42)
\end{aligned}
$$

Since $\sum_{l=1}^{i} \alpha_l < 1$,

$$
\sum_{l=1}^{i}\sum_{j=1}^{N} B_{l,k,j} - C < 0. \qquad (43)
$$

Furthermore, we have

$$
\begin{aligned}
C - \sum_{l=1}^{i-1}\sum_{j=1}^{N} B_{l,k,j} &= C - \sum_{l=1}^{i-1}\sum_{j=1}^{N} n_{l,k,j} * \rho_l \\
&= C - \sum_{l=1}^{i-1} \rho_l \sum_{j=1}^{N} n_{l,k,j} \\
&= C - \sum_{l=1}^{i-1} \rho_l M_l \\
&\geq C - \sum_{l=1}^{i-1} \rho_l \frac{\alpha_l C}{\rho_l} \\
&= (1 - \sum_{l=1}^{i-1} \alpha_l)C. \qquad (44)
\end{aligned}
$$

Substituting (41), (42), and (44) into (21), we have

$$
\begin{aligned}
d_{i,k} &= \frac{\sum_{l=1}^{i}\sum_{j=1}^{N} A_{l,k,j} + (\sum_{l=1}^{i}\sum_{j=1}^{N} B_{l,k,j} - C)\max_{j=1}^{N}\{\tau_{i,k,j}\}}{C - \sum_{l=1}^{i-1}\sum_{j=1}^{N} B_{l,k,j}} \\
&\leq \frac{\sum_{l=1}^{i}(T_l + \rho_l * Y_{l,k})\frac{\alpha_l C}{\rho_l} + (\sum_{l=1}^{i}\alpha_l - 1)C\max_{j=1}^{N}\{\tau_{i,k,j}\}}{(1 - \sum_{l=1}^{i-1}\alpha_l)C} \qquad (45)
\end{aligned}
$$

Considering $\sum_{l=1}^{i} \alpha_l - 1 < 0$, and substituting (35) into (45), we have

$$
d_{i,k} \leq \frac{\sum_{l=1}^{i}(T_l + \rho_l * Y_{l,k})\frac{\alpha_l}{\rho_l} + (\sum_{l=1}^{i}\alpha_l - 1)\frac{\alpha_i(T_i + \rho_i Y_{i,k})}{\rho_i(N - \alpha_i)}}{1 - \sum_{l=1}^{i-1}\alpha_l}. \qquad (46)
$$

Q.E.D

B Proof of Theorem 2

We need to establish a lemma before proving the theorem.

LEMMA 5 *In an execution of the off-line delay computation and verification algorithm shown in Figure 2,*

$$\vec{d}_i^{[h+1]} \geq \vec{d}_i^{[h]}, \tag{47}$$

where $h \geq 0$ and $i = 1, \cdots, L$.

Proof of Lemma 5: We prove this lemma by induction.

According to the algorithm

$$\vec{d}_i^{[0]} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix} \tag{48}$$

and

$$\vec{d}_i^{[1]} = \vec{Z}_i(\vec{d}_i^{[0]}) = \begin{bmatrix} z_{i,1}(\vec{d}_i^{[0]}) \\ z_{i,2}(\vec{d}_i^{[0]}) \\ \vdots \\ \vdots \\ z_{i,M}(\vec{d}_i^{[0]}) \end{bmatrix} = \begin{bmatrix} \frac{\sum_{l=1}^{i} T_l \frac{\alpha_l}{\rho_l} + (\sum_{l=1}^{i} \alpha_l - 1)\frac{\alpha_i T_i}{\rho_i (N-\alpha_i)}}{1-\sum_{l=1}^{i-1} \alpha_l} \\ \frac{\sum_{l=1}^{i} T_l \frac{\alpha_l}{\rho_l} + (\sum_{l=1}^{i} \alpha_l - 1)\frac{\alpha_i T_i}{\rho_i (N-\alpha_i)}}{1-\sum_{l=1}^{i-1} \alpha_l} \\ \vdots \\ \vdots \\ \frac{\sum_{l=1}^{i} T_l \frac{\alpha_l}{\rho_l} + (\sum_{l=1}^{i} \alpha_l - 1)\frac{\alpha_i T_i}{\rho_i (N-\alpha_i)}}{1-\sum_{l=1}^{i-1} \alpha_l} \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix} = \vec{d}_i^{[0]}, \tag{49}$$

where $z_{i,k}$ *and* $\vec{Z}_i$ are defined in (5) and (6), respectively.

Assuming that $\vec{d}_i^{[0]} \leq \vec{d}_i^{[1]} \leq \cdots \leq \vec{d}_i^{[h]}$, we need to show $\vec{d}_i^{[h]} \leq \vec{d}_i^{[h+1]}$. Note that, by its definition, $Y_{i,k}$ is an increasing function of $\vec{d}_i$. That is, if $\vec{d}_i^{[h]} \geq \vec{d}_i^{[h-1]}$, $Y_{i,k}(\vec{d}_i^{[h]}) \geq Y_{i,k}(\vec{d}_i^{[h-1]})$. Furthermore, we observe

$$\frac{\partial z_{i,k}}{\partial Y_{i,k}} = \frac{\alpha_i + (\sum_{l=1}^{i} \alpha_l - 1)\frac{\alpha_i}{N-\alpha_i}}{1 - \sum_{l=1}^{i-1} \alpha_l} \geq 0. \tag{50}$$

Thus, $z_{i,k}$ is an increasing function of $Y_{i,k}$.

Synthesizing the monotonic properties of $z_{i,k}$ and $Y_{i,k}$, which are observed above, we conclude that $z_{i,k}(\vec{d}_i)$ is an increasing function of $\vec{d}_i$. Thus, if $\vec{d}_i^{[h]} \geq \vec{d}_i^{[h-1]}$,

$$z_{i,k}(\vec{d}_i^{[h]}) \geq z_{i,k}(\vec{d}_i^{[h-1]}). \tag{51}$$

Now, following the algorithm, we have

$$\vec{d}_i^{[h+1]} = \vec{Z}_i(\vec{d}_i^{[h]}) = \begin{bmatrix} z_{i,1}(\vec{d}_i^{[h]}) \\ z_{i,2}(\vec{d}_i^{[h]}) \\ \vdots \\ \vdots \\ z_{i,M}(\vec{d}_i^{[h]}) \end{bmatrix} \geq \begin{bmatrix} z_{i,1}(\vec{d}_i^{[h-1]}) \\ z_{i,2}(\vec{d}_i^{[h-1]}) \\ \vdots \\ \vdots \\ z_{i,M}(\vec{d}_i^{[h-1]}) \end{bmatrix} = \vec{Z}_i(\vec{d}_i^{[h-1]}) = \vec{d}_i^{[h]}. \tag{52}$$

Q.E.D

THEOREM 2 *The off-line delay computation and verification algorithm shown in Figure 2 terminates.*

Proof of Theorem 2: We prove this theorem by a contradiction. Assume that the inner loop of the algorithm iterates infinitely. If this is the case, from Lemma 5, we know

$$\vec{d}_i^{[0]} \leq \vec{d}_i^{[1]} \leq \vec{d}_i^{[2]} \leq \cdots \leq \vec{d}_i^{[h]} \leq \cdots. \tag{53}$$

Since the algorithm iterates infinitely, we know, for $h > 1$,

$$\epsilon < \frac{||\vec{d}_i^{[h+1]} - \vec{d}_i^{[h]}||}{||\vec{d}_i^{[h+1]}||} \leq \frac{||\vec{d}_i^{[h+1]} - \vec{d}_i^{[h]}||}{||\vec{d}_i^{[1]}||}. \tag{54}$$

That is,

$$||\vec{d}_i^{[h+1]} - \vec{d}_i^{[h]}|| \geq \epsilon ||\vec{d}_i^{[1]}|| > 0. \tag{55}$$

Therefore, we can find $k_h, h = 2, 3, \cdots$, such that

$$d_{i,k_h}^{[h+1]} - d_{i,k_h}^{[h]} = ||\vec{d}_i^{[h+1]} - \vec{d}_i^{[h]}|| \geq \epsilon ||\vec{d}_i^{[1]}||, \tag{56}$$

where $1 \leq k_h \leq M$.

Then we can find a server k and a sequence of integers

$$h_1 < h_2 < \cdots < h_n < \cdots, \tag{57}$$

such that

$$\epsilon ||\vec{d}_i^{[1]}|| \leq d_{i,k}^{[h_n+1]} - d_{i,k}^{[h_n]}. \tag{58}$$

Hence, we have

$$\lim_{n\to\infty} d_{i,k}^{[h_n]} = \infty. \tag{59}$$

(59) implies, for a given D_i, there is an $X > 0$ so that when $n > X$

$$d_{i,k}^{[h_n]} > D_i. \tag{60}$$

However, by the algorithm,

$$d_{i,k}^{[h_n]} \leq G(\vec{d}_i^{[h_n]}, P\_Set) \leq D_i, \tag{61}$$

where $G(\vec{d}_i^{[h_n]}, P\_Set)$ is defined in Section 4.3. This is a contradiction. Q.E.D