Utilization-Based Admission Control for Real-Time Applications

Dong Xuan*, Chengzhi Li+, Riccardo Bettati*, Jianer Chen*, and Wei Zhao*
*Department of Computer Science
Texas A&M University
College Station TX 77843-3112, USA
dxuan, bettati, chen, zhao@cs.tamu.edu

+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 present a methodology to apply utilization-based admission control to provide hard real-time communication in a scalable fashion. We make admission control scalable by using a configuration-time test to determine a safe utilization level of servers. Admission control at run-time then is reduced to simple utilization tests on the servers along the path of the new flow. Furthermore, we discuss how appropriate route selection improve utilization levels, design a safe route selection heuristic algorithm to achieve high utilization of resources, and derive two bounds on the maximum utilization level for given traffic in a network. We compare the results of our route selection heuristics with that of a shortest-path based algorithm, and find that our heuristics can achieve a much higher maximum utilization level than that of the shortest-path based algorithm.

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

Contents

1 Introduction

This paper presents a methodology to apply utilization-based admission control to provide hard real-time communication in a scalable fashion. By *hard real-time*, we mean that packets are delivered from their sources to the destinations within their end-to-end deadlines. Packets delivered beyond their end-to-end deadlines are considered useless. By *scalable*, we mean that we want to support a large number (say, hundreds of thousands) of flows over a network, while at the same time allowing for high resource utilization.

1.1 Integrated and Differentiated Services

The Integrated Services (`intserv`) architecture of the IETF [1] is a good example to illustrate the types of problems encountered in admission control for large systems. In Integrated Services, each flow is strictly managed first by admission control at flow establishment time and then by packet scheduling during the lifetime of the flow. At establishment time, the necessary resources for the new flow are allocated, and during the lifetime, the flow is policed to ensure that abnormal behavior of a flow does not affect other flows. This necessitates that information about every flow 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 do not scale well, for two reasons: first, routers are required to maintain status information of flows and schedule packets for large numbers of flows. Second, as the number of flows increases, the run-time overhead incurred for flow establishment and tear-down increases as well. The `intserv` architecture, therefore, does not provide scalable QoS guaranteed services, due to overhead both at flow establishment and during the flow lifetime.

To make communication service scalable, the overhead of both admission control during flow establishment and packet scheduling during the flow lifetime must be largely independent of the number of flows in the system. One way to achieve this is to *aggregate* user-level flows: for example, in the Differentiated Services (`diffserv`) Architecture [31, 32, 33], individual user-level flows are assigned to only a small number of predefined *classes.* Packets in each class are served by a *class-based* scheduling policy, such as class-based static priority, or class-based weighted-fair-queuing [4, 5]. As a result, nodes are aware only of *aggregations* of flows. Each edge router is supposed to aggregate the individual flows into a small number of such aggregate flows.

While the benefits of flow aggregation in terms of reduced overhead for packet scheduling are clear, the question of how to take advantage of flow aggregation for scaling of admission control remains to be answered. In this paper, we address this question in the content of `diffserv` architecture.

1.2 Utilization-Based Admission Control

Utilization-based methods do implicitly, but naturally rely on workload aggregation to provide scalable and robust schedulability testing and admission control mechanisms. All these techniques define a utilization level below which all the workload using the resource is guaranteed to meet its deadline. This utilization level is sometimes called the *worst-case achievable utilization* (WCAU) for that resource [3]. The WCAU can then be compared against the resource utilization during schedulability testing at design time or during admission control at run time to determine whether the workload can be guaranteed to meet its deadlines. A variety of WCAU's for different settings have been found, e.g., 69% and 100% for preemptive scheduling of periodic tasks on a single server using rate-monotonic and earliest-deadline-first scheduling, respectively [2], or 33% bandwidth utilization for scheduling synchronous traffic over FDDI networks [3].

Unfortunately, previous work on utilization-based methods to provide end-to-end delay guarantees in distributed systems has been limited. WCAU's have been derived for single servers only, or for some shared servers (like token rings), and can therefore be used only indirectly for systems with multiple servers. This is typically done by deriving the local delay of a flow on each single server from that server's utilization, and then adding the local delays on all the servers used by the flow.

Most previous studies on utilization-based methods also assumed that either the traffic arrival for each flow is well behaved at the server, or that the traffic for each flow is controlled by some sort of *workload regulation*. Workload regulation in networks can be done explicitly, with the use of *traffic shapers*, or is inherently part of servers with *guaranteed-rate schedulers*, such as weighted-fair-queuing (WFQ) [23, 24] ,virtual clock (VC) [11], or others. Unfortunately, workload regulation in the network relies on identifying individual flows to regulate, thus eliminating flow aggregation for packet forwarding. While the assumption on workloads being well behaved at the first server may hold, it does not so in systems where flows traverse multiple servers and therefore become more and more jittered as they do so.

In this paper, we derive a highly scalable utilization-based mechanism to perform admission control in computer networks that allow for workload aggregation.

1.3 Main Results of This Paper

One of the main results in this paper will be to complement the existing WCAU for single server systems: We will describe a simple algorithm to test whether, for a given utilization level, a set of workloads with end-to-end delay requirements can be guaranteed to meet its deadlines over a network of servers. In particular, we make admission control scalable by using a configuration-time test to determine a safe utilization level of servers. Admission control at run-time then is reduced to simple utilization tests on the servers along the path of the new flow.

The rest of the paper is organized as follows. In Section 2, we describe previous work and our motivation. The underlying network and traffic models for this paper are introduced in Section 3. In Section 4, we give a short overview of the components needed to enable our admission control mechanism. We describe the various configuration steps and the underlying delay computation formulas in Section 5. In Section 6, we illustrate with an experiment that the utilization bounds on assignments are high. A summary of this paper and motivation of future work are given in Section 7.

2 Previous Work

A number of schemes have been proposed to deploy network services that provide QoS guarantees. As described earlier, the `intserv` architecture is one of such efforts. One step further, the `difserv` architecture [31, 32, 33] 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 [13, 14, 19, 20] for a large variety of packet scheduling mechanisms [11, 15, 17, 21, 23, 24, 34]. Finally admission control has been investigated as well [16, 18, 22]. Much of this work addresses admission control and packet scheduling. Some of it deals with meeting end-to-end delay requirements.

Some of the proposals have been implemented and deployed, for example, NetEx, RSVP, and Tenet. NetEx [6, 7] is an example of an implemented system based on extensions of Cruz's methodology for delay analysis [8, 9]. Essentially, NetEx provides connection-oriented real-time communication service by relying on strict admission control [10]. Tenet Scheme II [26] uses a two-pass resource allocation scheme, with extensive functionality needed for multi-party communication. RSVP [25] has been proposed as the signaling and resource reservation protocol in the `intserv` architecture. One common point of all these systems is their limited scalability. While they provide real-time services, they 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 flow awareness in the signaling protocols. Stoca and Zhang [12] move per-flow information from the core routers to the edge of the network by relying on packets containing the flow state information which core routers rely on for admission control and packet scheduling. The result is that the core router doesn't need to maintain per flow information anymore but infers it based on the information carried with the packet. The storage of per-flow information has moved to the edge router from the core router, at the expense of additional scheduling overhead at routers and a degree of incompatibility with the traditional IP protocol. We believe that to be scalable over a wide variety technologies, the core router should serve packets with class-based information only, thus rely on flow aggregation.

For the `diffserv` architecture, much research needs to be done to provide end-to-end delay guarantees. In this paper, we propose a methodology of admission control and packet forwarding schemes that guarantee end-to-end delays for all flows within a differentiated service network [32]. By having off-line end-to-end delay computation, our admission control complexity is definitely lower than those of [6, 7, 8, 9, 10, 12, 26], while guaranteeing end-to-end delay deadlines for all flows. Also, we argue that our system's run-time overhead should be lower than that of [12], because we use class-based static priority scheduling. To the best of our knowledge, this is the first proposal addressing scalable end-to-end delay guarantees for differentiated services networks.

3 Network and Traffic Models

In this section, we describe the model and define the terminology that will be used in the following sections.

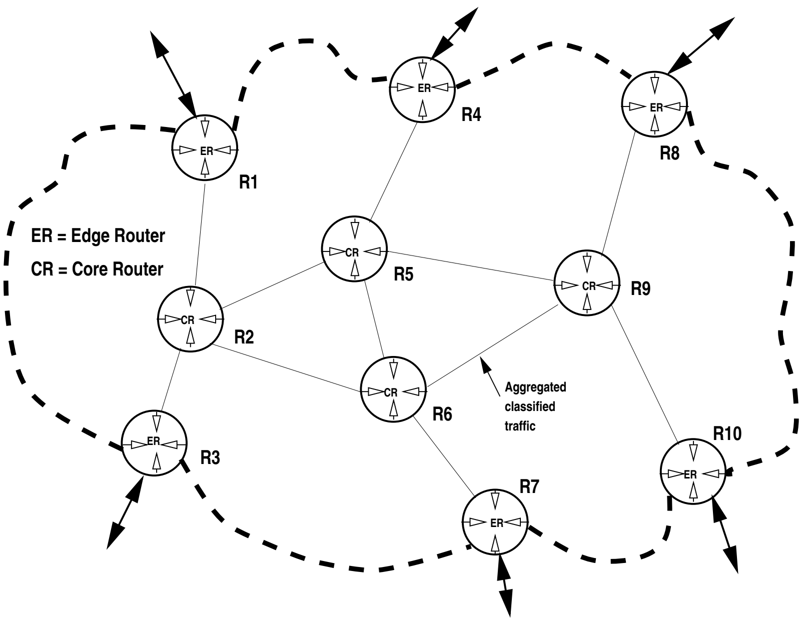

Figure 1: A Simple Illustration of A Network

Network, Routers, and Links The *diffserv* architecture distinguishes two types of routers: *Edge routers* are located at the boundary of the network, and provide support for traffic policing. *Core routers* are within the interior of the network. We assume all routers to have N input links, and all output links to be of capacity C, in bits per second. A router is connected to other routers or hosts through its input and output links. Demand of transmitting packets may temporarily exceed the capacity at an output link in a router, and thus packets may suffer some queuing delay at the output buffer. This delay depends on the load of the output link, its capacity, and the adopted scheduling policy.

For the purpose of delay computation, we follow standard practice and model a router as a set of *servers*, one for each router component, where packets can experience queuing delays. Packets are typically queued at the output buffers, where they compete for the output link. We therefore model a router as a set of output *link servers*. All other servers (input buffers, non-blocking switch fabric, wires etc.) can be eliminated by appropriately subtracting constant delays incurred on them from the deadline requirements of the traffic.

Consequently, the network can be modeled as a graph $G = (S, E)$ of connected link servers. The link servers in set S are connected through either links in the network or paths within routers, which both make up the set of edges E in the graph. We use L to denote the diameter of G, i.e. the maximum length of the shortest paths in G between any pair of hosts.

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.

It is the responsibility of the admission control module 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.

Flows and Paths We call a stream of packets between a sender and receivers application service access points a *flow*. Packets of a flow are transmitted along a single path, which we model as a list of link servers.

Classes of Service Following the `diffserv` architecture, flows are partitioned into classes. QoS requirement and traffic specifications of flows at the entrance to the network are defined on a class-by-class basis.

In order to appropriately characterize traffic both at the ingress router and within the network, we use a general traffic disciples in form of traffic functions and their independent counterpart, maximum traffic functions.

DEFINITION 1 *The traffic function $f_{i,k,j}(t)$ is defined as the amount of the traffic of Class i arriving at Server k at 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 [8]:

DEFINITION 2 *Function $F_{i,k,j}(I)$ is called the traffic constraint function of $f_{i,k,j}(t)$ if, for any $t > 0$ and $I > 0$,*

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

We assume that the source traffic of a flow in Class i is controlled by a leaky bucket with burst size T_i and average rate ρ_i. The total amount of traffic generated by this source during any time interval $[t, t + I)$ is bounded by $\min\{C * I, T_i + \rho_i * I\}$.

The QoS requirements of flows (in our case, end-to-end delay requirements) are specified on a class-by-class basis as well. For our purpose, we define the end-to-end deadline requirement of Class i traffic to be D_i. As no distinction is made between flows belonging to the same class, all flows in the same class are guaranteed the same delay. In the following, we will use $d_{i,k}$ to denote the worst-case queuing delay suffered by Class i traffic at Server k.

4 Resource Management Components

The admission control methodology presented in this papers relies on three components, of which the first is invoked during system configuration and modification to service level agreements, and the remaining two are invoked at flow establishment time and during flow life time, respectively.

1. Configuration: Given the network topology and the traffic requirement, the main task of configuration module is the resource assignment for different traffic classes and route determination for the required traffic.

2. Run-time Admission Control: This module makes the admission decision for a new flow at run-time depending on whether or not there is enough resource along the path of the new flow in order to guarantee its QoS requirement.

3. Packet Forwarding: This module controls how packets are scheduled and forwarded in a link server.

Utilization-based admission control makes the admission decision during flow establishment at run-time based only on the usage of resources, that is, their current utilization. This kind of run-time admission control is very efficient and straight-ward. It needs only check if the bandwidth is available along the path of the new flow. To be able to do that, safe levels of utilization need to be determined to compare against during admission control in order to guarantee the required delays. This is done as part of a separate configuration step, which is done at system startup or after renegotiation of service level agreements in the case of `diffserv` networks. Hence, in the following sections, we will focus our discussion on configuration.

The value for safe utilization depends on the particular packet forwarding control algorithm being used, as the delay performance depends on how the packets are forwarded. In this paper, we assume that the class-based static priority policy is used in packet forwarding. With this policy, packets are transmitted according to their class priorities, and packets are served in FIFO order within a class.

5 Configuration

We described earlier how the admission control procedure works during flow establishment. Our approach consists of a simple utilization test on the link servers along the path of the new flow. This relies on having previously verified a safe level of link server utilization (or link bandwidth) for each class of service. This verification is part of a separate service configuration procedure, which is invoked either during initial network configuration, or during modifications to service level agreements.

During service configuration, the network topology is given, together with the set of sender and receiver pairs, their traffic characteristics, and their QoS requirements. We distinguish three types of configuration, depending on whether the routes of the traffic and the utilization assignments for the traffic classes α_i are given:

1. *Verification of Safe Utilization Assignment:* Both the routes of the traffic and the utilization assignment of classes are given. The objective here is to verify whether the end-to-end delay QoS requirement of all the required traffic will be guaranteed along the given routes under the given utilization.

2. *Safe Route Selection for Given Utilization:* The utilization assignment of classes are given but the routes are not. The objective here is to select a set of routes so as to make sure that the QoS requirement of all the required traffic can be guaranteed along the selected routes under the given utilization assignment.

3. *Safe Route Selection to Maximize Utilization:* Neither the routes of the traffic nor the utilization assignment of classes are given. The objective here is to do route selection to maximize the utilization of classes (via route selection) under the constraint that the QoS requirement of all the traffic should be guaranteed.

In the following subsections we will first describe these three types of configuration for a special case of two traffic classes, (1) a real-time, high priority class and (2) a non-real-time, best effort, class. Subsequently, we will shortly present how the result can be extended to systems with multiple real-time classes.

5.1 Verification of Safe Utilization Assignment

A utilization assignment is considered *safe* if the QoS requirement of all the required traffic will be guaranteed along all given routes as long as no traffic class exceeds its assigned utilization on any link. An algorithm to verify the safety of an utilization assignment must

```
Algorithm Verification of Safe Utilization Assignment:

INPUT: network topology $G = (S, E)$, a set of routes $SR$, utilization assignment $\alpha$:

Step 1: Compute the packet delay for each link server in $G$.

Step 2: Compute the end-to-end delay for every route in $SR$.

Step 3: If the delay requirement is satisfied for flows of every class on every routes, return
    $SUCCESS$ and terminate. Otherwise, return $FAILURE$ and terminate.
```

Figure 2: Verification of Safe Utilization Assignment

therefore first compute the worst-case end-to-end delay of flows of each class for every route in the network, and then compare it with the deadline requirement for each class. If all the deadline requirements are being met, the utilization assignment is safe. Once the local delays on all link servers have been determined, the end-to-end delays can be computed by adding the delays on link servers along the paths of flows.

The key step in this algorithm is the worst case delay computation for the link servers. Hence, we will focus on how to compute the class-based delay for each link server at configuration time.

5.1.1 Delay Computation

The worst-case delays on link servers in such systems depend on the number and traffic characteristics of flows competing for the server. Inside the network, the traffic characteristics of a flow at the input of a link server depends on the amount of contention experienced upstream by that flow. Intuitively, all the flows *currently* established in the network must be known in order to compute delays when no flow separation is provided. Delay formulas for this type of systems have been derived for a variety of scheduling algorithms [27].

While such formulas could be used (at quite some expense) for flow establishment at system run-time, they are not applicable for delay computation during configuration time, as they rely on run-time information about flows.

As the information is not available at configuration time, the worst case delays must be determined under the assumption of the worst-case combination of flows has been established. An obviously impractical way to do that is to exhaustively list all possible combinations of flows in the system and compute the delays on the link servers for the every possible combinations to get the worst case delays. Fortunately, we can derive an upper bound on the worst-case delay without having to exhaust all the flow combinations. In this section, we will describe such a method for the case of class-based static priority scheduling at the link servers. Recall that, initially, we limit our solution to systems with two flow classes: (1) high-priority class with deadline constraints, i.e. real-time class and (2) a low priority, best-effort class, i.e. non-real-time traffic. In this type of systems, the deadline-guaranteed traffic is not affected by low-priority traffic, and we conveniently omit the class index in the following description. Thus, unless otherwise pointed out, all the traffic parameters are referred to that of the high priority class.

We will start with a formula for delay computation that depends on flow information, which we call *the general delay formula.* We will focus on how to remove its dependency on flow information.

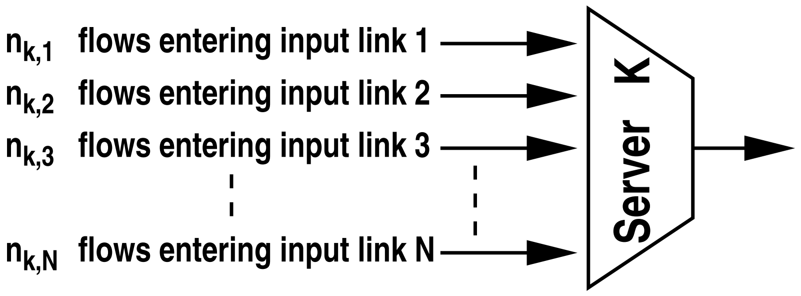
$n_{k,1}$ flows entering input link 1
$n_{k,2}$ flows entering input link 2
$n_{k,3}$ flows entering input link 3
$n_{k,N}$ flows entering input link N
Server K

Figure 3: Server k

General delay formula Consider Server k, which at some time moment has $n_{k,j}$ high-priority traffic flows through its Input Link j. Let $F_{k,j}(I)$ be the constraint function for the aggregated traffic belonging to the high-priority class on Input Link j of Server k. This constraint function can be formulated as the sum of the constraint functions of the individual flows using Input Link j, that is,

$$F_{k,j}(I) = \sum_{m=1}^{n_{k,j}} H_{k,j,m}(I), \tag{2}$$

where $H_{k,j,m}(I)$ is the constraint function for the m^{th} flow in the real-time class on Input Link j of Server k.

The worst case delay, d_k, can then easily be formulated in terms of the aggregated traffic constraint functions and the service rate C of the server as follows [27]:

$$d_k = \frac{1}{C}\max_{I>0}(\sum_{j=1}^{N} F_{k,j}(I) - C * I). \tag{3}$$

Substituting (2) into (3), we observe that the above delay formula depends on the dynamic system status, i.e., $n_{k,j}$, the numbers of flows at each input link and $H_{k,j,m}$, the traffic constraint function of the individual flows. This kind of dependency on the dynamic system status must be removed in order to perform delay computations at configuration time.

We will describe below how we first eliminate the dependency of the delay formula on the traffic constraint function $H_{k,j,m}(I)$. Then we eliminate the dependency on the numbers of flows on each link, i.e., $n_{k,j}$.

Removing the dependency on $H_{k,j,m}$, the constraint function of the individual flows: In the following theorem, we show that the aggregated traffic function $F_{k,j}(I)$ can be bounded by replacing the individual traffic constraint functions $H_{k,j,m}(I)$ by a common upper bound $H_k(I)$, where $H_k(I)$ is defined to be the traffic constraint function of the flow experiencing the longest delay upstream from Server k.

THEOREM 1 *The following inequality holds: for $j = 1, \cdots, N$*

$$F_{k,j}(I) \leq n_{k,j} * H_k(I), \tag{4}$$

where $H_k(I)$ can be formulated as follows:

$$H_k(I) = \min\{C * I, T + \rho * Y_k + \rho * I\}. \tag{5}$$

where

$$Y_k = \max_{Path \in S_k} \sum_{j \in Path} d_j, \tag{6}$$

and where S_k is the set of all paths upstream from Server k for flows that traverse Server k.

A proof of Theorem 1 is given in Appendix A.

According to Theorem 1, the general delay formula in (3) can be re-formulated as:

$$d_k \leq d_k^* = \frac{1}{C}\max_{I>0}(\sum_{j=1}^{N} n_{k,j} * H_k(I) - C * I). \tag{7}$$

Computing delay by using (7) still depends on the number of flows on each input links. Next, we describe how to remove this dependency.

Removing the dependency on $n_{k,j}$, the numbers of flows per input link: As we described earlier, admission control at run-time makes sure that the link utilization allocated to each class of flows is not exceeded. The number of flows on each input link is therefore subject to the following constraint:

$$\sum_{j=1}^{N} n_{k,j} \leq \frac{\alpha C}{\rho}. \tag{8}$$

Under this assumption, the delay bound in (7) can be maximized for all the possible distribution of $n_{k,j}$, as the following theorem shows:

THEOREM 2 *The delay in (7) is maximized when*

$$\max\{n_{k,1}, n_{k,2}, \cdots, n_{k,N}\} = \lceil \frac{\alpha C}{\rho N} \rceil. \tag{9}$$

A proof of Theorem 2 is given in Appendix B. This theorem states that when the maximum value of $n_{k,j}$ is $\lceil \frac{\alpha C}{\rho N} \rceil$, d_k^{**}, the bound of the delay is maximized. In fact, the theorem is consistent with the intuition that when the flows distribute evenly among the input links, the delay of the server will be maximized. Using the above results, we have the following theorem:

THEOREM 3 *The delay d_k is bounded as follows:*

$$d_k \leq d_k^{**} \quad = \quad (T + \rho * Y_k)\frac{\alpha}{\rho} + (\alpha - 1)\frac{\alpha(T + \rho Y_k)}{\rho(N - \alpha)}. \tag{10}$$

A proof of Theorem 3 is given in Appendix C. The right-hand side of (10) does not depend on any parameters related to dynamic system status. Hence, this formula can be used for delay computation at configuration time.

Configuration-time Delay Computation Procedure In order to simplify notations, we use vector $\vec{d}$ to denote the upper bounds of the queuing delays suffered by the traffic of the real-time class at all servers:

$$\vec{d} \quad = \quad [d_1, d_2, \cdots, d_{|S|}]^\top. \tag{11}$$

We note that, in (10), d_k depends on Y_k. Then, according to (6), the value of Y_k, in turn, depends on the delays experienced at servers other than Server k. In general, we have a circular dependency, as $\vec{d}$ depends on Y_k, and Y_k depends on $\vec{d}$. Hence, the values of d_k, $k = 1, \cdots, |S|$ depend on each other and must be computed simultaneously. Let us define:

$$z_k(\vec{d}) \quad = \quad (T + \rho * Y_k)\frac{\alpha}{\rho} + (\alpha - 1)\frac{\alpha(T + \rho Y_k)}{\rho(N - \alpha)}. \tag{12}$$

and

$$\vec{Z}(\vec{d}) \quad = \quad [z_1(\vec{d}), z_2(\vec{d}), \cdots, z_{|S|}(\vec{d})]^\top. \tag{13}$$

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

$$\vec{d} \;=\; \vec{Z}(\vec{d}). \tag{14}$$

Since the unknown vector $\vec{d}$ appears on both sides of (14), an iterative procedure is needed to compute $\vec{d}$.

5.2 Safe Route Selection

Although the delay bounds derived above are independent of the dynamic configuration of flows, they still depend on the route selection. As an appropriate selection of routes results in lower end-to-end delay bounds for a given bandwidth assignment, better effective utilization of resource can be achieved. Any route selection should satisfy the QoS requirements for all classes of flows in a given network.

By finding a *safe* route selection for a given network and utilization assignment, we mean determining a set of routes, one for each source and destination pair, such that, for each class and each route, the end-to-end delay does not exceed the deadline requirement for that class.

Unfortunately, the safe route selection problem is $\mathcal{NP}$ hard, even for the case of two classes. This can be shown by reduction from the `Maximum Fixed − Length Disjoint Paths` problem [36].

While it is unlikely that a polynomial-time algorithm exists for the safe route selection problem, simple heuristic-supported search has proven to work acceptably well. In the simplest case, a no-backtrack search algorithm repeatedly selects a pair of source and destination, and then chooses the most promising route among the routes for that pair that satisfy the delay requirements of all flows. If, at any point in time, no more such route can be found, the algorithm declares failure.

The performance of this simple search procedure can be improved by means of appropriate selection heuristics when choosing the next source/destination pair or when choosing the most promising routes for a given source/destination pair.

In our realization of the algorithm, we use these heuristics as follows: (1) We select the next source/destination pair in decreasing order of distance between source and destination. (2) We select a group of candidate routes for the new pair so that whenever possible, each of them forms a noncyclic graph with existing routes. By minimizing the number of cycles in the routes, the feedback in the queuing of packets is reduced, and so is the delay [27]. (3) Among the candidate routes, we select a route which leads to the minimum route delay.

The result is a polynomial-time algorithm. In Section 6, we will discuss performance of this algorithm in terms of effectiveness of finding a safe route selection for a given utilization assignment.

5.3 Maximizing Utilization by Safe Route Selection

5.3.1 Basic Ideas

From the view point of a user of the real-time communication service, the assigned utilization for the real-time class should be as high as possible in order to maximize the chance that a real-time flow is admitted. The issue is then how high this utilization can be? The objective of this subsection is therefore to develop an algorithm that selects routes so as to maximize the utilization of the real-time class that can be assigned.

Let α^* be the maximum utilization that can be assigned. The following theorem provides the lower bound and the upper bound of α^*. These bounds will be useful when we develop our algorithm.

THEOREM 4 *For a two-class network of diameter L with N input links at each router and real-time traffic with parameters T and ρ, and deadline D, α^* is bounded as follows:*

$$\frac{N}{(\frac{LT}{\rho D} + (L-1))(N-1) + 1} \le \alpha^* \le \frac{N((\frac{D\rho}{T} + 1)^{\frac{1}{L}} - 1)}{N + (\frac{D\rho}{T} + 1)^{\frac{1}{L}} - 2} \tag{15}$$

While the lower and upper bounds in (15) are functions of various system parameters, they do not depend on other information about network topology, except L, the network diameter. That is, the theorem holds for any network topology. In practice, it is particularly useful to have a lower bound of α^* that is topology independent. A network manager is then safe to assign this lower bound for the utilization regardless what the topology is and how it may change, as long as the diameter is bounded by L.

Given these two bounds, we can easily develop a search algorithm that identifies the maximum utilization bound and its associated set of routes. The algorithm simply applies a binary search on the assigned utilization: The search space is initialized by the lower bound and upper bound given in Theorem 4. The middle point of the search space is taken as the value of assigned utilization. Then, the safe route selection algorithm described in the last subsection is called to choose a set of safe routers, if they exist. The search space reduces to the upper half or lower half depending on whether or not the route selection algorithm can successfully find the safe routes. The search continues until the size of the search space reduces to zero.

5.3.2 Derivation of the Utilization Bounds

A complete proof of Theorem 4 can be found in [37]. It is not given here due to the space limitation. Here, we discuss the physical meanings of these bounds and show how these intuitions help the derivations.

The lower bound of the maximum utilization means that for a given network with parameters N, ρ, T, D, and L, there exists at least one safe route selection as long as the link utilization does not exceed the lower bound value. Thus, this bound can be derived by 1) developing an upper bound of the delay (at the longest path) with a particular route selection algorithm (say, Shortest Path based algorithm (SP)) and (2) seeking a condition under which the deadline constraint can always be met.

We assume that d is the maximum worst case queuing delay suffered by the traffic of the real-time class at any servers. L is the diameter of the network. Let SP be used for the route selection. Thus, the length of the longest path is bounded by L. Recall that Y_k is the worst case queuing delay suffered by the traffic of the high class before arriving at Server k.

$$Y_k \leq (L-1) * d. \tag{16}$$

Substituting (16) into (10), we can have

$$d = (T + \rho * (L-1) * d)\frac{\alpha}{\rho} + (\alpha - 1)\frac{\alpha(T + \rho(L-1) * d)}{\rho(N-\alpha)}. \tag{17}$$

Note that (17) gives the upper worst delay a server can have regardless what topology is. In order to let the packet on the longest path to satisfy the deadline constraint, we have

$$d * L \leq D \tag{18}$$

Substituting (17) into (18), we have

$$\frac{N}{(\frac{LT}{\rho D} + (L-1))(N-1) + 1} \leq \alpha^*. \tag{19}$$

Now, let's deal with the upper bound of the maximum utilization (15). This upper bound implies that for a given network, no safe route selection can be found if the utilization exceeds this value. We derive this bound by (1) developing a lower bound on the end-to-end delay, (2) seeking a condition under which even the lower bounded delay will exceed the deadline.

The upper bound can be derived by determining the worst-case end-to-end delay on the longest shortest route (say R with length L), assuming that all flows are routed so that the delays among that route are minimized.

According to (10), we see that d_k depends on Y_k. Let R_k be the the sub-path of R from the source to the server k. In the best situation, Y_k is the sum of the link server delay along the route R_k itself. In other words, routes are selected so as to cause no feedback due to circulation routing for delays on Route R. In this kind of best case, we have:

$$d_k = (T + \rho * \sum_{j \in R_k} d_j)\frac{\alpha}{\rho} + (\alpha - 1)\frac{\alpha(T + \rho \sum_{j \in R_k} d_j)}{\rho(N-\alpha)}. \tag{20}$$

On the other hand,

$$\sum_{j \in R} d_j \leq D. \tag{21}$$

After a few algebraic manipulations, we have

$$\alpha^* \leq \frac{N((\frac{D\rho}{T} + 1)^{\frac{1}{L}} - 1)}{N + (\frac{D\rho}{T} + 1)^{\frac{1}{L}} - 2}. \tag{22}$$

.

5.4 Extension to Multiple Classes

In the previous sections, we have limited our discussion to systems with one real-time class and one best-effort class. This allowed us to clearly describe how to derive a delay upper bound that does not relay on run-time information about established flows. In this section we will briefly discuss how to extend this to multiple (≥ 3) traffic classes.

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. Let $H_{i,k,j,m}(I)$ is the constraint function for the m^{th} flow in Class i on Input Link j of Server k. Let $F_{i,k,j}(I)$ be the constraint function for the aggregate traffic of Class i on Input Link j of Server k. Then

$$F_{i,k,j}(I) = \sum_{m=1}^{n_{i,k,j}} H_{i,k,j,m}(I). \tag{23}$$

In order to take into account the impact of higher-priority traffic, the delay formula in (3) must be appropriately modified [27]:

$$d_{i,k} = \frac{1}{C} \max_{I>0} \left(\sum_{l=1}^{i-1} \sum_{j=1}^{N} F_{l,k,j}(I + d_{i,k}) + \sum_{j=1}^{N} F_{i,k,j}(I) - C * I \right). \tag{24}$$

Similar to the two-class delay formula in (10), the above delay formula depends on the run-time information about established flows. Following an approach similar to that in Section 5.1, we obtain the following delay bound for a system with multiple class. We present the result as the following theorem. Its proof can be found in [37].

THEOREM 5 *The worst case queuing delay $d_{i,k}$ suffered by the traffic of Class i at Server k is given as*

$$d_{i,k} = \frac{\sum_{l=1}^{i}(T_l + \rho_l * Y_{l,k})\frac{\alpha_l}{\rho_l} + (\sum_{l=1}^{i} \alpha_l - 1)\frac{\alpha_i(T_i+\rho_i Y_{i,k})}{\rho_i(N-\alpha_i)}}{1 - \sum_{l=1}^{i-1} \alpha_l}. \tag{25}$$

where T_i and ρ_i are the burst size, the average rate and the percentage of bandwidth reserved for Class i, respectively.

$$Y_{l,k} = \max_{Path \in S_{l,k}} \sum_{j \in Path} d_{l,j}, \tag{26}$$

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

We note that (25) does not rely on run-time information of established flows. We can use the iterative procedure described in Section 5.1, and, thus, have a verification procedure for systems with multiple classes.

Variations of the algorithms derived in Sections 5.2 and 5.3 can then be used to select safe routes and to either maximize utilization assignments or trade-off utilization assignments of classes against each other.

6 Experimental Evaluation

An effective configuration-time delay computation and route selection method allows for assignment of high utilization. The latter in turn is closely related to the degree at which high-priority traffic is allowed to utilize the network resource without violating deadline requirements. The higher the utilization assignment for a class, the more flows the admission control mechanism allows to be established at run-time over a given link.

In this section, we compare our heuristic search algorithm described in Section 5.2 with the Shortest Path based routing algorithm (SP). The measurement is the achieved maximum utilization assignment level of the compared algorithms.

Figure 4 shows the topology used throughout these experiments. We selected the MCI ISP backbone network[1] Note that L, the diameter of the topology is 4, N, the maximum number of links for a router is 6. In the experiments, all routers can act as edge routers, and flows can be established between any two routers. All links in the simulated network have the same capacity of 100 Mbps.

We use a voice-over-IP as scenario for the experiments. We distinguish two types of traffic in this network, namely guaranteed-delay voice traffic, and best-effort data traffic. The voice traffic is assigned to the high-priority, real-time class, while the data traffic is assigned to the low-priority class. All voice flows are policed at the entrance of the network using a leaky bucket with the maximum bust size of 640bits and a rate of 32kbps. The end-to-end deadline requirement of all real-time flows is fixed at 100ms. This kind of QoS requirements

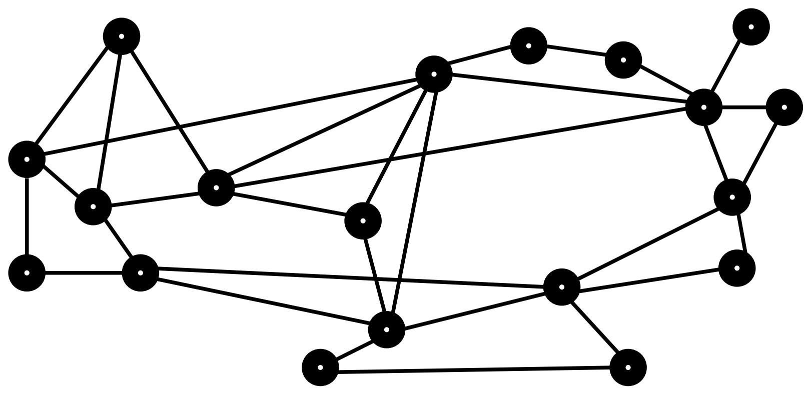

Figure 4: Simulation Network Topology

is similar to "Voice-over-IP" applications [28].

The results are listed in Table 1. According to (15), we also obtain the lower bound and upper bound of utilization for the above mode respectively.

We observe that the maximum utilization of SP routing is 33%, close to the lower bound which is 30%. This means that using SP selection routing results in very low achievable utilization. On the contrary, our heuristic route selection algorithm can achieve the maximum utilization up to 45%. This is an increase of about 50% compared with the SP routing.

[1] The nodes in figure 4 are routers, not link servers.

Lower Bound	SP	Our Heuristics	Upper Bound
0.30	0.33	0.45	0.61

Table 1: Maximum Utilization

7 Conclusions

We have proposed a scalable methodology for providing QoS-guaranteed services within a `diffserv` 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 make admission control scalable by using a configuration-time test to determine a safe utilization level of servers. Admission control at run-time then is reduced to simple utilization tests on the servers along the path of the new flow. Furthermore, we design a safe route selection heuristic algorithm to achieve high utilization of resources, and derive two bounds on the maximum utilization level for given traffic in a network. We compare the results of our route selection heuristics with that of a shortest-path based algorithm, and find that our heuristics can achieve a much higher maximum utilization level than that of the shortest-path based algorithm.

For many applications, deterministic guarantees are not necessary [35]. 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 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] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard real time environment," *J. ACM*, Vol. 20, No. 1, 1973, pp.46-61.

[3] G. Agrawal, B. Chen, W. Zhao, and S. Davari, "Guaranteeing Synchronous Message Deadlines with the Timed Token Protocol," *IEEE International Conference on Distributed Computing Systems*, 1992, pp.468-475.

[4] P. Ferguson and G. Houston, "Quality of services - Delivering QoS on the Internet and in Corporate Networks," *Wiler Computer Publishing*, 1998

[5] S. Floyd and V. Jacobson, "Link-sharing and Resource Management Models for Packet Networks," *IEEE/ACM Transactions on Networking*, Vol. 3 No. 4, pp. 365-386, August 1995.

[6] 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.

[7] 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.

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

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

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

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

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

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

[14] A. Banerjea and S. Keshav, "Queueing delays in rate controlled ATM networks," *Proceedinds of Inforcom'93*, 1993.

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

[16] 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.

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

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

[19] 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.

[20] 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.

[21] 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.

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

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] C. Li, R. Bettati, and W. Zhao, "Static priority scheduling for ATM networks," *Proceedings of the 18th IEEE Real-Time Systems Symposium*, 1997.

[28] 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.

[29] 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.

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

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

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

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

[34] 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.

[35] 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.

[36] M. R. Garey and D. S. Johnson, "Computers and Intractability: A Guide of to the Theory of NP-Completeness," *W.H. Freeman and Company, New York*, 1979

[37] D. Xuan, C. Li, R. Bettati, J. Chen and W. Zhao, " Configuration in the Scalabe Real-time Systems," Texas A&M University, Department of Computer Science. Feb. 2000.

Appendix A: Proof of Theorem 1

Theorem 1 *The following inequality holds: for $j = 1, \cdots, N$*

$$F_{k,j}(I) \leq n_{k,j} * H_k(I), \tag{27}$$

where $H_k(I)$ can be formulated as follows:

$$H_k(I) = \min\{C * I, T + \rho * Y_k + \rho * I\}. \tag{28}$$

where

$$Y_k = \max_{Path \in S_k} \sum_{j \in Path} d_j, \tag{29}$$

and where S_k is the set of all paths upstream from Server k for flows that traverse Server k.

Proof of Theorem 1 Recall that $H_{k,j,m}(I)$ is the traffic constraint function of flow m of the real-time class at Input Link j of Server k. As we assume that at their sources, flows in the same class have the same constraint functions $H(I)$, i.e.

$$H(I) = \min\{C * I, T, +\rho * I\}. \tag{30}$$

Let $Y_{k,j,m}$ be the total worst case queuing delay experienced by the packets from this flow before arriving at Server k. Recall that Y_k is the maximum of the worst case queuing delay suffered by any flow in real-time class before arriving at Server k. That is,

$$Y_{k,j,m} \leq Y_k. \tag{31}$$

According to Theorem 2.1 in [8], and using (28), (30) and (31), we have

$$H_{k,j,m}(I) \leq H(I + Y_{k,j,m}) \leq H(I + Y_k) = H_k(I). \tag{32}$$

That is, the traffic constraint functions of individual flows are bounded by $H_k(I)$ that is the traffic constraint function of the flow which experience the largest delay before arriving at Server k. By substituting (32) into (2), we have

$$F_{k,j}(I) \leq \sum_{m=1}^{n_{k,j}} H_k(I) = n_{k,j} * H_k(I) \tag{33}$$

Thus, (27) holds. Q.E.D

Appendix B: Proof of Theorem 2

Theorem 2 *The delay in (7) is maximized when*

$$\max\{n_{k,1}, n_{k,2}, \cdots, n_{k,N}\} = \lceil \frac{\alpha C}{\rho N} \rceil. \tag{34}$$

Before we formally prove Theorem 2, we need some lemmas.

LEMMA 1 *The aggregated traffic of the real-time class on Input Link j of Server k*

$$F_{k,j}(I) = n_{k,j} * H_k(I) \tag{35}$$

can be expressed in the following format:

$$F_{k,j}(I) = \begin{cases} C * I, & I \leq \tau_{k,j}, \\ n_{k,j} * T + n_{k,j} * \rho * Y_k + n_{k,j} * \rho * I, & \tau_{k,j} \leq I, \end{cases} \tag{36}$$

where

$$\tau_{k,j} = \frac{n_{k,j} * T + n_{k,j} * \rho * Y_k}{C - n_{k,j} * \rho}. \tag{37}$$

Sketch of Proof of Lemma 1 This lemma can be proved by substituting (27) into (35), and then rewriting $F_{k,j}(I)$ in the segmented form.

The following lemma gives the new format of d_k originally in (7):

LEMMA 2

$$d_k = (T + \rho * Y_k)\frac{\alpha}{\rho} + (\alpha - 1)\max_{j=1}^{N}\{\tau_{k,j}\} \tag{38}$$

Sketch of the proof of Lemma 2 With some algebraic manipulations on (7), we can have [27]

$$d_k = \frac{(T + \rho * Y_k)\sum_{j=1}^{N} n_{k,j} + \rho(\sum_{j=1}^{N} n_{k,j} - C)\max_{j=1}^{N}\{\tau_{k,j}\}}{C}. \tag{39}$$

As we know

$$\sum_{j=1}^{N} n_{k,j} \leq \frac{\alpha C}{\rho} \tag{40}$$

Substituting (40) into (39), we have

$$\begin{aligned} d_k &\leq \frac{(T + \rho * Y_k)\frac{\alpha C}{\rho} + \rho(\frac{\alpha C}{\rho} - C)\max_{j=1}^{N}\{\tau_{k,j}\}}{C} \\ &= (T + \rho * Y_k)\frac{\alpha}{\rho} + (\alpha - 1)\max_{j=1}^{N}\{\tau_{k,j}\} \end{aligned} \tag{41}$$

Q.E.D

Now, we are ready to prove theorem 2.

Proof of Theorem 2 Observing (2), we notice that the delay only depend on $\tau_{k,j}$ (as we know, $\tau_{k,j}$ is a function of $n_{k,j}$). Since $(\alpha - 1)$ is negative, to maximize the worst case delay, we need to get the minimum value of $\max_{j=1}^{N}\{\tau_{k,j}\}$. For this purpose, we denote

$$T(n_{k,1}, n_{k,2}, \cdots, n_{k,N}) = \max_{j=1,\cdots,N}\{\tau_{k,j}(n_{k,j})\}, \tag{42}$$

where $\tau_{k,j}$ is defined in (37). Recall that α is the portion of bandwidth of each server reserved for traffic of the real-time class. Let M be the total number of the flows of real-time class supported by Server k. That is,

$$M = \sum_{j=1}^{N} n_{k,j} \leq \frac{\alpha C}{\rho}. \tag{43}$$

Substituting (37) into (42), we have

$$\begin{aligned} T(n_{k,1}, n_{k,2}, \cdots, n_{k,N}) &= \max_{j=1,\cdots,N}\{\tau_{k,j}(n_{k,j})\} \\ &= \max_{j=1,\cdots,N}\{\frac{n_{k,j}(T + \rho Y_k)}{C - n_{k,j} * \rho}\}. \end{aligned} \tag{44}$$

Because $f(x) = \frac{x(T+\rho * Y_k)}{C - x\rho}$ is a monotonically increasing function of variable x, we have

$$T(n_{k,1}, n_{k,2}, \cdots, n_{k,N}) = \frac{n_k^*(T + \rho * Y_k)}{C - n_k^* \rho}, \tag{45}$$

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

Now, we would like to show that if a sequence of integers $n_{k,1}, n_{k,2}, \cdots, n_{k,N}$, satisfies

$$\sum_{j=1}^{N} n_{k,j} = M, \tag{46}$$

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

$$n_k^* = \lceil \frac{M}{N} \rceil. \tag{47}$$

Let us assume that this is not true. That is, (47) does not hold. We have three cases to deal with.

- *Case 1:* $n_k^* < \lceil \frac{M}{N} \rceil$. That is $n_k^* \leq \lfloor \frac{M}{N} \rfloor$. We have

$$\sum_{j=1}^{N} n_{k,j} \leq N n_k^* \leq N \lfloor \frac{M}{N} \rfloor < M. \tag{48}$$

  This is impossible either in accordance with (46).

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

$$\hat{n}_{k,j} = \begin{cases} \lfloor \frac{M}{N} \rfloor, & 1 \le j \le K, \\ \lceil \frac{M}{N} \rceil, & K < j \le N, \end{cases} \tag{49}$$

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

$$\frac{\hat{n}_{k,N}(T + \rho * Y_k)}{C - n_{k,N}\rho} < \frac{n_k^*(T + \rho * Y_k)}{C - n_k^*\rho}. \tag{50}$$

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

Hence, (47) holds. Recall that to maximize the worst-case delay (38), we need to maximized n_k^*. n_k^* is maximized when M is maximized, which gives

$$M = \frac{\alpha C}{\rho}. \tag{51}$$

The theorem then follows[2]. Q.E.D

Appendix C: Proof of Theorem 3

Theorem 3 *The delay d_k is bounded as follows:*

$$d_k \le d_k^{**} = (T + \rho * Y_k)\frac{\alpha}{\rho} + (\alpha - 1)\frac{\alpha(T + \rho Y_k)}{\rho(N - \alpha)}. \tag{52}$$

Proof of Theorem 3 Following Theorem 2, we have, if $\sum_{j=1}^{N} n_{k,j} = M$,

$$\begin{aligned} T(n_{k,1}, n_{k,2}, \cdots, n_{k,N}) &\ge \frac{\lceil \frac{M}{N} \rceil (T + \rho * Y_k)}{C - \lceil \frac{M}{N} \rceil \rho} \\ &= \frac{\lceil \frac{1}{N}\frac{\alpha C}{\rho} \rceil (T + \rho * Y_k)}{C - \lceil \frac{1}{N}\frac{\alpha C}{\rho} \rceil \rho} \\ &\ge \frac{\frac{1}{N}\frac{\alpha C}{\rho}(T + \rho * Y_k)}{C - \frac{1}{N}\frac{\alpha C}{\rho}\rho} \\ &= \frac{\alpha}{\rho}\frac{(T + \rho * Y_k)}{N - \alpha}. \end{aligned} \tag{53}$$

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

Substituting (53) and (42) to (38), we have:

$$\begin{aligned} d_k &= (T + \rho * Y_k)\frac{\alpha}{\rho} + (\alpha - 1)\max_{j=1}^{N}\{\tau_{k,j}\} \\ &\leq (T + \rho * Y_k)\frac{\alpha}{\rho} + (\alpha - 1)\frac{\alpha(T + \rho Y_k)}{\rho(N - \alpha)}. \qquad (54) \end{aligned}$$

Q.E.D