Resource Partitioning for Real-time Communication

Domenico Ferrari

Amit Gupta

Berkeley, California USA

Abstract

We investigate the network resource partitioning problem for real-time communication. Resource partitioning is useful for a number of applications, including virtual private subnetworks, advance reservation of real-time network services, fast establishment of real-time connections and support for mobile computing with real-time communication. We propose a scheme where resource partitioning requires extra computation only at connection establishment time, and where per-packet scheduling and rate control do not require any extra work for supporting resource partitioning.

1. Introduction

The increasing speed of computer networks and the improvement of workstation capabilities are enabling a new class of distributed applications, those involving multimedia data. It is widely believed that these applications should be supported in the general framework of real-time communication [5], [1], [10]. Real-time communication requires predictable performance (e.g., that the end-to-end data delivery delay be bounded); typically, the clients negotiate with the network service provider to obtain a desired Quality of Service (QoS), which the provider guarantees. A number of protocols and schemes have been proposed to provide real-time communication services. The currently proposed schemes are connection-oriented, and reserve resources (bandwidth, buffers, and so on) along the route of the connection. QoS guarantees cannot be provided if the network does not check for saturation before accepting new channels. Thus, the network must perform *admission control* to ensure that the guarantees are met.

This research was supported by the National Science Foundation and the Defense Advanced Research Projects Agency (DARPA) under Cooperative Agreement NCR-8919038 with the Corporation for National Research Initiatives, by AT&T Bell Laboratories, Hitachi, Ltd., Hitachi America, Ltd., Pacific Bell, the University of California under a MICRO grant, and the International Computer Science Institute. The views and conclusions contained in this document are those of the authors, and should not be interpreted as

For real-time communication services to achieve widespread usage, it is important that the protocols and schemes provide good capability for the network's management to control the network services. A resource partitioning capability, i.e., the ability to distribute the different resources available at any network node or link among a number of partitions, is important for network managers. Admission control and establishment computations for connections ought to be completely independent of the connections accepted outside the partition that the connection belongs to. This independence amounts to splitting the server into a number of sub-servers, where QoS guarantees can be offered to the clients by considering only the connections belonging to that partition; the guarantees are valid as long as the admission tests and rate control schemes for other partitions behave correctly, and are independent of the individual per-connection establishment decisions and computations performed in other partitions.

This independence is very useful. The different sub-servers can be used to form virtual private subnetworks. The network's management can keep a small fraction of resources for management and fault-handling traffic. A fraction of the network's resources may be kept for non-real-time traffic. Other applications include fast establishment of real-time connections and support for mobile computing.

In this paper, we present a scheme for network resource partitioning. The paper is organized as follows. Section 2 provides the background for the current work. In Section 3, we discuss the admission tests for a network server, and how these tests can be modified for resource partitioning. Section 4 discusses some of the applications of resource partitioning. We conclude this paper with a brief review of related work and suggestions for future work.

2. Background

In this paper, our approach to the partitioning of network resources is illustrated in the context of the Tenet scheme for real-time communication [5]. The next section describes the changes to be made to that scheme so that the network resources can be partitioned. The rest of this section introduces the specific Tenet scheme we refer to here, which is essentially the one called Scheme 1 to distinguish it from the more advanced one the Tenet group is developing, called Scheme 2 [5].

Scheme 1 is based on a simplex unicast network-layer abstraction, the real-time channel, characterized by traffic and performance bounds specified by the client. The traffic bounds for channel i are the following:

- $x_{min,i}$: the minimum inter-packet interval;
- $x_{ave,i}$: the minimum of the average inter-packet intervals;
- I_i: the averaging interval used in the definition of $x_{ave,i}$;
- $s_{max,i}$: the maximum packet size.

The performance bounds for the channel i are:

- $D_{max,i}$: the end-to-end delay bound;
- $Z_{min,i}$: the minimum probability that packet delay is not greater than $D_{max,i}$;
- $J_{max,i}$: the end-to-end delay jitter bound;
- $W_{min,i}$: the minimum probability that a packet is lost due to buffer overflow.

The Tenet approach is connection-oriented and reservation-based: before a real-time channel can be used by its requestor, it must be established (i.e., resources for the channel must be set aside along its route), so that the desired performance guarantees are supported. Note that these resources are not reserved for the channel's exclusive use; the channel only has higher priority for using them, but, in keeping with the principles of packet switching, they can be used by non-real-time traffic whenever no packets from the channel are present.

Channel establishment is a distributed process. A message issued by the source (though it could be issued by the destination instead) visits each node (switch, router, gateway) on the route of the channel. This message causes several admission tests and computations to be performed at each node. If the new channel passes all the tests in a node, the message is forwarded, with some state information about the current node, to the next node on the route. The final tests are performed by the destination; if they are successful, a channel-established message is sent by the destination to the source along the reverse route; when each node is revisited, the message corrects the tentative reservations made in that node by the forward message, and informs the node about the performance bounds assigned to it. Some of these bounds will be used during data transfer operations, others in tests for the admission of future channels, and some in both types of circumstances.

If any of the tests in the nodes or in the destination fails, the channel cannot be established, and a channel-rejected message is immediately sent back to the source. This message removes in each node it visits the tentative reservations made for the new channel.

A new channel's route may be computed by the source (or destination) if this host has the necessary topological information; knowledge by this host of the current real-time load (i.e., a measure of how much of each resource is currently earmarked for use by a real-time channel), and of such additional information as propagation delays and error rates of the links involved, is also quite useful to increase the probability that the chosen route will be able to support the new channel. An alternative is the construction of the route in a hop-by-hop fashion, with individual nodes usually not knowing much beyond the real-time loads of their immediate neighbors.

To complete this general description of Scheme 1, we only have to mention two special aspects of data transfers: scheduling and distributed rate-control. Most scheduling policies can be used for real-time communication under fairly liberal conditions [2]. Since some of the tests to be performed in a node during channel establishment are dependent on the scheduling discipline implemented in that node, we have chosen to restrict this discussion to the case of the Multi-class EDD (Earliest-Due-Date) discipline in its two versions: the so-called Delay-EDD (or D-EDD) [4], [12] and Jitter-EDD (or J-EDD) [3], [12]. Rate control, either at the periphery of the network or in all of its nodes, is needed to protect well-behaving channels from the misbehavior of faulty or malicious sources, and from the occasional bunching of packets on a channel due to traffic fluctuations. While the latter effect is automatically cured by the J-EDD scheduling discipline, which restores the original traffic pattern in each node implementing this policy, the former effect (and both effects in a D-EDD node) can be eliminated by suitably postponing the offending packets' deadlines in the node's scheduler [4].

3. Resource partitioning

What parts of Scheme 1 need to be modified to make network resources partitionable? Our ideal objective is to subdivide a network (or, more precisely, an internetwork) into a certain number of virtual networks (or internetworks), each one of which can be treated totally independently of the others, even though they all share the same hosts, nodes, and links. This is equivalent to saying that we would like to confine our admission tests to the resources assigned to the partition to which a new channel is requesting admission, without in any way involving the channels and the resources belonging to the other partitions. To show that this goal can be reached, we must prove that the admission tests (for instance those in Scheme 1) can be modified for partition admission. In the next subsection, we present a new, more recent version of those tests, which differs somewhat from that published in the past [4], since it is based on a different model of the internetwork. We then state and prove two results that are needed to partition one of the resource types, and show the simple changes to be made to the tests.

3.1. Admission tests for a full network

Figure 1 illustrates the model that we adopt for a simplex unicast real-time channel. Since each node visited by the channel may be represented in this model by more than one single server, the model is quite a general one. Each server is of one of two types:

(i) *J-type*, if it is scheduled by a discipline that can control jitter (in our example, the J-EDD discipline); and

(ii) *NJ-type* otherwise.

The admission tests we discuss here assume that the NJ-type servers are scheduled by the D-EDD discipline, but both D-EDD and J-EDD can easily be replaced by other disciplines, as long as the tests are modified appropriately. There is no need for all servers of either type to be homogeneous: servers may be scheduled by a variety of policies even within the same network and the same channel route, but of course we have to use the correct set of tests for each one of them.

Each server is endowed with given amounts of the following resources: (i) buffer space, (ii) computing power, and (iii) *delay resource*; depending on the component modeled by the server, computing power may represent the processor power or the link bandwidth, and where the *delay resource* refers to the feasibility of scheduling all packets by the server's scheduler within the respective local delay bounds.

A server on the modeled channel's route is in general traversed by a number of other, already established real-time channels. Each channel's end-to-end characterization is the 8-tuple of traffic and performance parameters listed in Section 2 above. In each server *n*, an established channel *k* is characterized by the following quantities:

- $d_{n,k}$: local delay bound (used to compute the EDD deadline);
- $z_{n,k}$: local minimum probability of delay bound violation;
- $w_{n,k}$: local minimum probability of buffer overflow;
- $t_{n,k}$: maximum packet service time.

While the first three parameters of this 4-tuple are computed during the establishment procedure, $t_{n,k}$ is assumed to be known in each server before the server is visited by the establishment message during its forward trip. In a J-type server, a channel whose delay jitter bound $J_{\max}$ is specified is characterized also by the canonical delay $D^*_{n,i}$, which is the delay with respect to the departure time from the source each packet on that channel will be forced to have in that server.

Another quantity that is known beforehand (and that we assume is transmitted to the destination through the establishment message for each visited server) is the propagation delay $d_{prop,n}$ of the link preceding server *n* along the new channel's route. Note that this quantity is the only one that characterizes each link in the model in Figure 1.

During the forward trip of the establishment message, we can see whether in each server

(a) the addition of a new channel with a deterministic delay bound (i.e., $Z_{min,i}=1$) would cause the worst-case utilization of the server by deterministic channels to rise above 1 (**deterministic test**);

(b) the addition of the new channel would cause the worst-case utilization of the server by all channels to reach a point where the probability of violating the local delay bound of an established channel would be greater than its local bound for that probability (**statistical test**);

(c) the amount of buffer space that is available for the new channel in the server would allow its packets to spend there a maximum amount of time larger than the minimum local delay bound the server can offer the new channel (**buffer space test**).

The minimum local delay bound $dmin_{n,i}$ offered by the server *n* to new channel *i* is the minimum value of $d_{n,i}$ that makes the packets on channel *i* schedulable. Computing $dmin_{n,i}$ is the goal of the **delay bound computation** to be performed by the server before the buffer space test. As shown in Table A.1, which summarizes the admission tests for EDD-scheduled servers, this computation entails sorting the established channels in order of increasing local delay bound, and determining the first position in this list where the new channel can be inserted without causing any of the local delay bounds of the channels below it to violate the condition

$$d_{n,h} \geq \sum_{l=1}^{h} t_{n,l} + T_n + t_{n,c_n+1}, (h = k, k+1, ..c_n), \qquad (1)$$

where T_n is the maximum service time the server n may give to any packet (including the non-real-time packets), c_n is the number of channels established in server n, and c_n+1 is the index locally assigned to the new channel. The value of $dmin_{n,i}$ is the value of the right-hand side of (1) for $h=k$, where k is the smallest value such that inequalities (1) are all satisfied. The proof of this result can be found in [4].

The admission tests and computations to be performed by the destination are collected in Table A.2. Note that there are several different ways of organizing the tests and computations; the choices made in building Tables A.1 and A.2 have been dictated by the desire to simplify understanding and to avoid local computations during the reverse trip; they are by no means optimum in any sense of the term.

3.2. Admission tests for a network partition

Network resources can be partitioned if, given a number of partitions and suitable admission tests to each partition, we can prove that all channels established in all partitions always satisfy the admission tests to the full network. Since in our approach most of the admission testing is distributed, and the final tests performed by the destination are only concerned with the end-to-end performance bounds of the new channel (see Table A.2) rather than directly with the server's resources, we can reduce the problem to the one of partitioning the resources in each server, and only investigate the changes to be made to the tests and computations in each server. We observe that there is no need for the partitions to be characterized by the same fractions of each resource in all servers. Thus, a partition P_j can be defined as a set of server partitions P_{nj}, where $P_{nj}=(\beta_{nj},\gamma_{nj},\delta_{nj})$ is the triple of the fractions of server-n resources (buffer space, computing power, delay resource) assigned to P_{nj}, and $0 \le (\beta_{nj}, \gamma_{nj}, \delta_{nj}) \le 1$, $\sum_j \beta_{nj} = \sum_j \gamma_{nj} = \sum_j \delta_{nj} = 1$.

It is easy to prove that a valid deterministic test for partition P_{nj} can be obtained by replacing the utilization $U_{det,n}$ in Table A.1 with $U_{det,nj}$, and the test with $U_{det,nj} < \gamma_{nj}$. Also, the buffer space test can be performed as specified in Table A.1 except for the following modification to the constraint on the maximum amount of space the server can give a new channel:

$$bmax_{n,i} \leftarrow B \le \beta_{nj} B_n - \sum_{k=1}^{c_{nj}} b_{n,k}, \$ \qquad (2)$$

where c_{nj} is the number of established channels in P_{nj}, and these channels are assumed to be numbered $1,2,..c_{nj}$.

The statistical test is performed only when the worst-case utilization of server n, U_n, is greater than or equal

to 1. One would be tempted to replace 1 with γ_{nj} in this condition and state that the test can be done without modifications in each partition. However, this would be incorrect, since the values of $z_{n,k}$ for all the channels with a statistical delay bound established in server n are to be tested against $1-P_{do,n}$, where $P_{do,n}$ is the probability of deadline overflow in server n [4]. Thus, neither the value of $P_{do,n}$ for a partition can be used, nor can one restrict the tests to the statistical channels within a partition; perhaps this would be possible if packets from different partitions were treated in different ways by the scheduler in each server, but our approach excludes this possibility.

The solution we propose for this problem is that a suitable value of $P_{do,n}$, to be denoted by $P^*_{do,n}$, be chosen for each server n, by experience, or by simulation, or by successive adjustments, and that the probability of deadline overflow $P_{do,nj}$ of each partition not be allowed to exceed $P^*_{do,n}$. The value of $zmax_{n,i}$ for a statistical channel will be set equal to $1-P^*_{do,n}$, and the statistical test will become $P_{do,nj} \le P^*_{do,n}$, as testing the values of the z's will no longer be necessary. Note that the server-wide value of $P_{do,n}$ is guaranteed to be smaller than $P^*_{do,n}$ in any situation in which $U_n \ge 1$; hence, the server-wide statistical tests in Table A.1 are guaranteed to be successful in all cases in which the per-partition statistical tests have been successful.

To determine the changes required by the delay bound computation, we need to introduce a few definitions and theorems.

Definition 1: A *schedule* S in an EDD server n is a set of channels $C_1, C_2, ..C_m$ with local delay bounds in n such that $d_{n,1} \le d_{n,2} \le .. \le d_{n,m}$.

Definition 2: Schedule S is *acceptable* for δ_{nj} in EDD server n if

$$d_{n,k} \ge \frac{1}{\delta_{nj}} \sum_{l=1}^{k} t_{n,l} + T_n \quad (k=1,2,..m), \qquad (3)$$

where $0 \le \delta_{nj} \le 1$, $t_{n,l}$ is the maximum service time for a packet on channel C_l in server n, and T_n is the maximum service time that any packet can have in n.

Definition 3: Two or more schedules are said to be *disjoint* if their pairwise intersections are all empty.

Theorem 1: If schedule S is acceptable for δ_{nj} in server n, then S is acceptable for all λ such that $\delta_{nj} \le \lambda \le 1$.

Proof: Follows directly from (3).

Theorem 2: If disjoint schedules $S_1, S_2, ..S_r$ are acceptable for $\delta_{n1}, \delta_{n2}, ..\delta_{nr}$, respectively, in server n, so that $\sum_{h=1}^{r} \delta_{nh} \le 1$, then the combined schedule