Connection Establishment for Multi-Party Real-Time Communication

R. Bettati, D. Ferrari, A. Gupta, W. Heffner,
W. Howe, M. Moran, Q. Nguyen, R. Yavatkar

The Tenet Group
University of California, Berkeley, and
International Computer Science Institute

Abstract. There is considerable interest in the network community in supporting real-time multi-party applications, such as video conferencing. The Tenet Group at UC Berkeley and ICSI has designed and implemented protocols that provide *quality of service* (QoS) guarantees for real-time traffic on packet switching networks. Suite 2 of the Tenet protocols provides scalable, flexible and efficient network support for real-time multi-party connections. We outline our method of connection establishment and describe the design issues and alternatives, and our decisions. Preliminary measurements confirm the viability of our approach for real-time multicast connection establishment.

1 Introduction

The design goals and principles of multicast real-time channel establishment in the Tenet Suite 2 are described, along with the mechanisms that support the establishment procedure. The suite described in this paper is based on the Tenet Suite 1 [1]. Therefore, the approach taken in Suite 1 is described in Section 2. The changes made to that approach to adapt it to the multicast abstraction in Suite 2, and a step-by-step example of Suite 2 channel setup are give in Section 3. The performance measurements of a Suite 2 prototype reported in Section 4 show that establishment times are likely to be quite reasonable, even if this approach is used in a wide-area context. Section 5 discusses some other connection establishment methods proposed in the literature.

2 The Tenet Unicast Channel Establishment Procedure

The basic communication abstraction in the Tenet real-time schemes and protocol suites is the *real-time channel*, to be simply called *channel* in the rest of this paper. In the Tenet Scheme 1, which is the set of algorithms and procedures on which the Tenet Suite 1 [1] is based, a channel is a simplex *unicast* connection with QoS guarantees [4]. Since guaranteeing performance and reliability bounds requires admission control (and keeping track of how much of each resource is still available for additional channels, an operation usually called somewhat improperly "resource reservation"), channels are established prior to their use for data transport; in other words, real-time communication has the basic characteristics of a connection-oriented service.

The main design objectives of the Tenet channel establishment procedure since the beginning of our investigation in 1987 were:

- **Speed:** Many real-time clients cannot afford long waits for their transmissions to start [5].
- **Scalability:** The procedure must work also in very large networks and internetworks [6].
- **Mathematical provability of the guarantees:** Real-time clients must be offered firm contractual commitments by the service provider [4, 11].
- **Flexibility in the specification of QoS and traffic parameters:** The network-layer clients must be given as much freedom as possible in specifying their QoS requirements and traffic descriptions; service classes can be easily defined and easily added on top of a flexible interface, while extending the classes around which a channel establishment procedure is built may be very difficult.

To achieve the goal of fast establishment, we needed a relatively simple traffic model and a relatively simple set of QoS parameters, so that the admission tests could be rapidly executed. To be easily scalable, the procedure had to be fully distributed; a centralized approach was likely to be a scalability, reliability, and performance bottleneck. Thus, we decided that an establishment message would be sent out by the source, carrying the QoS and traffic parameters of the requested channel, and that it would follow a pre-computed route, (or build one in a hop-by-hop way), triggering the execution of admission tests in each node it traversed, and tentatively "reserving" resources in the node if all tests were passed therein. Any test failure would cause a *channel reject* message to be sent back to the source; this message would release all the resources tentatively set aside for the channel in each upstream node. To make channel establishment fast, the number of round trips required to complete the establishment procedure was limited to one, even though allowing for more round trips could often decrease the amounts of resources to be set aside for a channel. A consequence of this constraint was the need for the final acceptance or rejection decision to be made in the destination host. If the channel is accepted and enough information is available, this host can also decide the amounts of resources to be allocated to the new channel in each node, and can inform the nodes about these amounts through the *channel accept* message; alternatively, the adjustment (or *relaxation*) of the allocated amounts can be left to the individual nodes, which will inform the upstream nodes about the effects of these adjustments via the same message. After completing the initial design of Scheme 1, we realized that two simple modifications allowed channels to be established in the reverse direction (i.e., starting from the destination) and as duplex connections, respectively.

A crucially important aspect of the procedure was the derivation of local (node) performance bounds from the end-to-end bounds. Admission tests must verify that a new channel will not endanger the guarantees given to the channels already established in each node. For speed and scalability, these tests must be performed locally, i.e., without requiring additional messages (to, for example, the sources of all those channels). Thus, local bounds contain all the information admission tests need to know about each channel, allowing the establishment procedure to be much faster and less expensive; the price that is paid for these advantages is the reduction in the real-time capacity of the network due to the worst-case assumptions to be made in subdividing some end-to-end bounds. For instance, the minimum probability of timely delivery is broken into the product of the local probabilities, which is equivalent to assuming that a packet's timely delivery to the destination requires that the packet be transmitted by each node before the local deadline expires, a clearly pessimistic assumption. The derivation of local bounds, besides localizing the admission tests, is also extremely useful for interoperability: the Tenet establishment procedure works in networks in which different nodes use different packet scheduling policies, as long as it uses in each node the admission tests suitable for that node.

For guarantees to be mathematically provable, the worst-case philosophy must be applied to all admission tests and to the traffic model. Any traffic model to be used for admission control must be a *bounding* model, i.e., a worst-case description of the traffic on the requested channel. However, *worst-case* does not necessarily mean *deterministic*: from the outset, the Tenet Scheme 1 included *statistical* bounds for delays and packet losses, which were based on statistical worst-case techniques. Mathematical provability was also responsible for the decision to allocate buffers statically to each channel in each node [8].

3 Multi-Party Connection Establishment

We first discuss our goals for multi-party communication and their implications for channel establishment. We then illustrate these concepts with a brief example.

3.1 Changes in Establishment for Multi-Party Communication

Implementation of the first-generation real-time protocol suite enabled us to demonstrate the feasibility of providing guaranteed performance communication with real-time connections and admission control. To support multi-party communication, we have extended our goals as follows:

- **Scalability and Efficiency:** In moving to multi-party communication, we have extended the notion of scalability to include the number of *participants* in a given multi-party interaction; achieving such scalability requires efficient allocation of network resources.
- **Flexibility:** The notion of flexibility is extended to include support for dynamic changes in the set of destinations receiving data on a channel, and support for the use of heterogeneous traffic specifications and scheduling algorithms.

- **Apportionment of resources:** The network should *be able* to apportion network resources between user groups according to the "share" that may be specified for each group by the network manager(s).
- **Ease of use:** Facilities must be provided to assist clients in creating and managing real-time channels; for example, clients must be able to book channels in advance of their use.

In keeping with the connection-oriented design of the first Tenet scheme, most of the changes for multi-party communication have affected channel establishment, so that data forwarding remains relatively simple and fast. However, two changes are required in the data forwarding system: to support multicast channels, packets must be copied at branching nodes, and, to support resource sharing, the packet scheduling mechanism must be modified [9].

Implementing multi-party communication with unicast channels would be cumbersome and inefficient in the use of network resources. On the other hand, a single $M \Rightarrow N$ channel abstraction either would be very complex or would fail to match the requirements of all types of multi-party communication, and might introduce inefficiencies for $1 \Rightarrow N$ or unicast communication. Therefore, we obtain an efficient equivalent of an $M \Rightarrow N$ channel by combining $1 \Rightarrow N$ multicast channels (hereafter called simply *multicast channels*) and *resource sharing* among them. The $1 \Rightarrow N$ multicast channel provides a natural abstraction for the delivery of data from a single source to all intended receivers. Performance bounds are defined as in the case of the unicast service. Utilization of network resources is reduced by using a multicast tree, so that multiple copies of the same data are not forwarded across a link.

Naive use of multicast channels would result in over-allocation of resources, since in most multi-party communication scenarios, only a small subset of potential senders are active at a given time. For example, in a typical multimedia conference, only one person will speak at a time. We have introduced a *sharing group* abstraction to allow clients to describe such behavior to the network, allowing the network to share resource allocations between such related channels. The client specifies the channels in the sharing group, and a *group traffic specification*, which indicates the maximum combined traffic entering the network from *all* channels in the group. Admission control tests and packet scheduling mechanisms use the group traffic specification rather than the traffic specifications for the individual channels. More details on resource sharing and results of simulation experiments can be found in [9]. Those experiments indicate that resource sharing significantly improves the scalability of multi-party communication.

In Scheme 2, the real-time channel is a multicast connection. Therefore, the channel establishment procedure attempts to produce a multicast connection from the source to all destinations. *Partial establishment* is allowed, whereby a channel is successfully established even if one or more destinations could not be reached by the channel. Scheme 2 supports the dynamic addition and deletion of destinations to a multicast channel by an *incremental establishment and teardown* mechanism. This mechanism can be exploited in Suite 2 to achieve *receiver-initiated establishment*. In receiver-initiated establishment, the receiver first establishes (in the reverse direction) a channel from the sender to itself. Once this channel is established, other receivers can join it dynamically using the incremental establishment capability. We expect that receiver-initiated establishment will prove to be more scalable in the number of receivers than sender-initiated establishment. However, we should note that in the current prototype, receiver-initiated establishment would not fully realize this scalability gain, since, for ease of implementation, incremental join and leave are implemented in terms of sender-initiated mechanisms. To achieve the expected scalability gains for receiver-initiated establishment, we must first implement receiver-to-source establishment and distributed target set, channel and routing objects.

Local establishment tasks are divided between a component that handles signalling (e.g., parsing, updating, and forwarding establishment messages) and one or more components that manage a single resource (i.e., by performing admission control and resource allocation), with a well-defined interface between the two. This architecture supports the use of heterogeneous scheduling algorithms, since the algorithms and the state information are encapsulated in the resource managers.

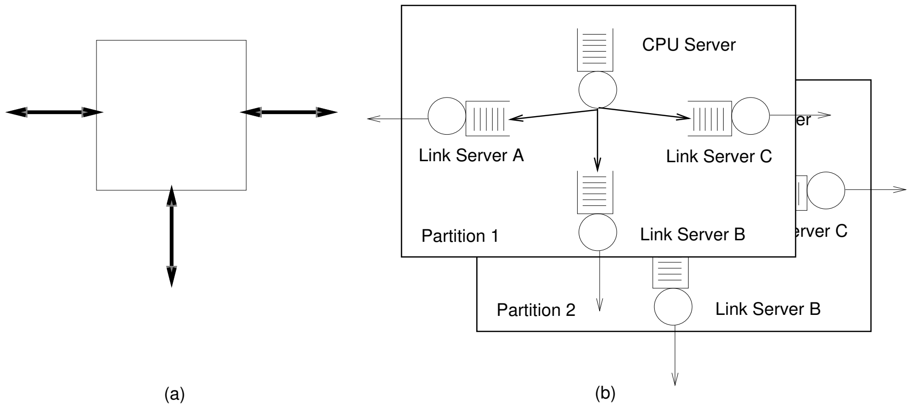

Fig. 1. Node model for node with three interfaces and two partitions

Resource partitioning allows network managers to specify how resources should be divided among user groups. In particular, network resources are apportioned into independent, disjoint partitions, so that channels can be established in one partition without affecting (or being affected by) channels in any other partition. A node model including resource partitioning is shown in Figure 1. We define a *server* as a schedulable network resource (e.g., CPU, link). A node (e.g., a router) with a CPU and three network interfaces, then, consists of four servers. Figure 1(b) shows the node model in the case where the node supports two partitions. Resource partitioning is completely supported through changes in channel establishment, and therefore requires no change in the data forwarding mechanisms. One or more resource partition(s) can be dedicated for channels that reserve resources in advance of their use. Admission control is somewhat more complicated for these "advance-booked channels"; [7] describes these admission control mechanisms.

Several long-lived objects have been introduced to ease the creation and management of multi-party communication in Scheme 2. *Channel objects* maintain channel state. *Target set objects* maintain information related to data receivers, and provide a rendezvous point between senders and receivers. A target set is similar to a host group in IP multicast [2], except that a target set specifies performance requirements for each destination.

When a destination joins (leaves) a target set, the target set requests the channel objects associated with it to add (remove) the destination to (from) their respective multicast trees.

3.2 Example of Channel Establishment

Figure 2 illustrates the steps taken to establish a typical channel. For the sake of

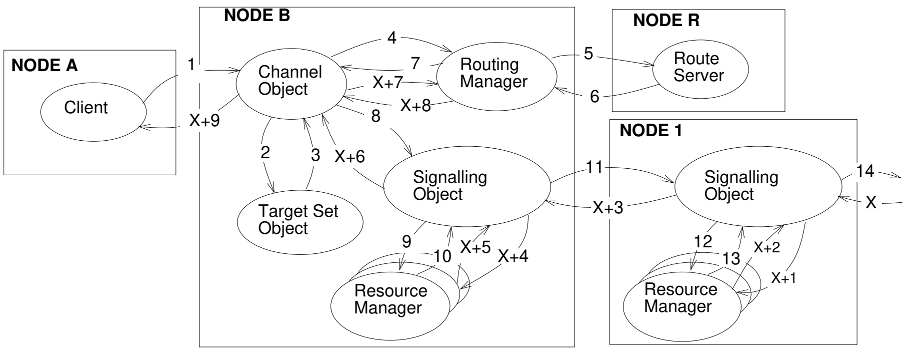

Fig. 2. Steps taken in typical channel establishment

brevity, we describe here only a source-initiated channel establishment. When a client requests the channel object to establish its channel (1), the channel object gets the list of destinations from the target set object (2, 3). The channel object sends this list to the routing manager, which gets a multicast route from the route server (4 – 7). This route is included in the establishment message, which is then sent to the (local) signalling object (8). The signalling object parses the establishment message; it requests the necessary resource manager(s) to run the admission control tests, and (if successful) to make the reservations (9, 10). If the tests fail, a *channel reject* message is sent back along the path. Otherwise, the signalling object updates and forwards the establishment message to the signalling object(s) of the next node(s) in the tree (11, ...). The establishment continues hop-by-hop until no other destinations exist along that branch. At each destination, end-to-end admission control tests are performed to ensure that the destination's performance requirements are met. Depending on the results, either a *channel accept* or a *channel reject* message is sent back to the originator of the establishment request $(\ldots, X, \ldots, X+9)$. At each node along the return path, resources are committed for each branch along which at least one destination was reached, and released otherwise (e.g., $X+1, X+2$). When replies have been received from all downstream branches (or a timeout has occurred), a *channel accept* or *channel reject* message is sent back along the path. The first signalling object then reports the result to the channel object $(X+6)$, which reports the results to the routing manager and to the client $(X+7,\ X+9)$.

4 Performance Measurements

One concern often expressed about admission control is that the admission tests might add considerably to the cost of connection establishment. To address this issue, we measured round-trip delays, per-node latencies, and the times spent in running admission tests on a Suite 2 prototype. Round-trip delay was defined in this context as the duration of the interval between the time the channel object received the request to set up a channel, and the time the results of that establishment attempt were returned by the channel object. In Figure 2, this would correspond to starting measurement at the completion of Step 1 and stopping immediately at the beginning of Step $X+9$.

The measurements were collected using the logical topology shown in Figure 3, which contains one branch node and three destination nodes; the maximum number of hops is four. All six machines are variants of `Sun Sparc 1` class workstations, and are connected to an Ethernet. A total of 35 channels were established successively. The numbers presented in Table 1 are typical ranges of the measured establishment delays.

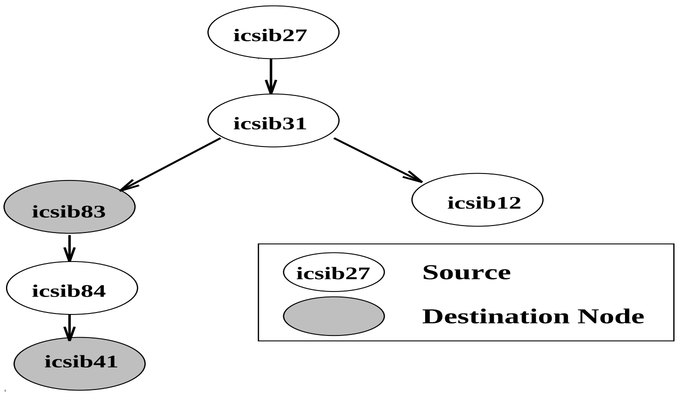

Fig. 3. Network topology of the experiment

Round-trip times had a maximum of 180 milliseconds and a minimum time of 103 milliseconds. On the average, one channel setup took 114 milliseconds. The one-time cost of Step 2 through Step 7 in Figure 2 was 7 ms. For advance-reserved channels, the measurements were similar (total channel setup time was 112 ms, Step 2 through Step 7 was 8 ms). Note that in the prototype that was measured we had messages to and from the routing server, but the route had been pre-computed; thus, route calculation times are not included in our measurements. We are currently integrating the routing server with our prototype, and anticipate that route calculation will generally add a small overhead to the measured 7 milliseconds. The remaining time (107 ms) was spent in the forward and reverse passes of channel establishment. This time will grow with the number of nodes on the longest of the paths from the source to each destination. In our experiments, the 5-node path from icsib27 to icsib41 was the longest.

Measure	Round-trip delay	Start-up latency	Per-node latency	Overhead	Communication	TCP/IP connect
Single Node	–	–	6.8	–	9	Tot: 4-5 Send: 0.4
Aggregate	114	7	34	73	72	Tot: 32-40 Send: 3-4

Table 1. Average channel establishment times (ms)

We can further break down our measurements into per-node latencies. These costs reflect the time spent in the signalling object and resource manager at each node on the longest path. On the forward pass, the resource manager sets up state information for the new channel and performs admission control (4.8 ms per node). During the reverse pass, the resource manager commits the resources for that channel (2 ms). Admission control consumes approximately 0.7 of the 4.8 milliseconds measured for the forward pass. Combining times for both passes, we obtain a total average per-node latency of 6.8 ms.

We can perform a rough calculation of message overhead by summing up the per-node latencies for each node on the longest path (34 ms) and subtracting this from the forward and reverse pass time (107 ms). This difference (73 ms) reflects the the costs of sending messages from one node's signalling object to the next. There are 8 inter-node messages on our longest path. To gain insights into inter-node message costs, we measured the time from when the signalling object requests that a message be sent to the time when that send completes (9 ms). Given this measurement, the aggregate cost along the longest path for the 8 messages is 72 ms, which is close to the difference calculated above.

In our initial prototype, we are sending control messages reliably via TCP/IP by opening a TCP connection, sending the message, and then closing it. To gauge the TCP overhead, we measured several successive attempts to open a TCP connection, send a small message and close that connection, and we found that these operations cost between 4 and 5 ms. We can reduce the communication overhead by caching TCP connections (i.e., keeping connections open between adjacent nodes). This was done in the implementation of Suite 1. The TCP/IP component of the message overhead would be reduced from 4-5 ms to 0.4 ms (per hop) by caching connections, potentially reducing the aggregate overhead from 73 ms to 36-44 ms.

Considering that we do not cache TCP connections, these results are not too different from the connection establishment latencies measured for the unicast Suite 1 [1]. For the latter, the time for channel establishment was measured to be between 80 and 90 ms for unicast channels with six hops between Berkeley and San Diego, California. Also, simulations show that the computational overhead associated with admission control does not increase appreciably with the addition of resource sharing [9] and advance reservations mechanisms [7].

As a final note, we performed the same experiment on a similar topology of DECstations and measured an average round-trip time of 100 ms. The average start-up cost (i.e., Step 2 through Step 7) was 3.5 ms with an average per-node cost of 4 ms at each signalling object. The relative breakdown of time is quite similar to that seen in the Sparc experments, except that the DECstations are newer and faster machines, thus resulting in faster computation times.

5 Related Work

Connection establishment in the Tenet Suite 2 differs in a number of points from the approaches taken by other multi-party resource reservation protocols, such as RSVP or ST-II. [1] RSVP is strictly a reservation protocol; it does not include routing or data transport functions, and has been designed to be used as a companion to IP. ST-II is a network-layer protocol that includes data transport, resource reservation, and routing.

The establishment process in RSVP is strictly receiver-oriented. Receivers establish a connection by sending a *Reservation* request message to the source. The path of the resource reservation message ends when it reaches either the source or a node of the existing distribution tree. The Tenet Suite 2 allows for both sender-initiated and receiver-initiated connection establishment, by either having channels be established to existing target sets or by having destinations join or leave target sets with established channels.

The channel establishment process in RSVP is done in a single pass from the receiver to the first node encountered on the distribution tree. Suite 2 establishes connection in two passes: the first pass tentatively reserves resources on the path from the source to the destinations (in the case of channel establishment) or from the destination to a node on the channel's multicast tree (in the case of a reverse-direction, incremental establishment); the second pass goes back to the origin of the establishment request and either confirms the resource allocation (in case of

[1] Although other significant differences exist, in this paper we will focus only on differences in the establishment procedures.

a successful establishment) or frees the temporarily allocated resources (in case of a failure to establish the connection with the required QoS parameters). This return pass informs the originator of the request about the QoS guarantees given to the established connection.

Unlike those made by Suite 2, RSVP reservations are considered "soft-state" and must be periodically refreshed. This approach allows RSVP connections to adapt to network load and router failures, but allows for disruption in real-time service due to underlying route changes.

Both ST-II and the Tenet Suite 2 are connection-oriented. This allows for routing to be performed at connection establishment time. Moreover, since the routing function is a part of, or interacts with, the respective setup protocol, resource availability is known at routing time. This allows for QoS guarantees to be given whenever a connection has successfully been established.

Originally, connection establishment in ST-II was rigorously sender-oriented [10]. In order for receivers to join, they had to inform the sender out-of-band, prompting it to generate a connection request for the receiver wishing to join. The limitations of the sender-initiated scheme have been identified, and two extensions for receiver-initiated connection establishment have been proposed in the IBM ENC ST-II implementation [3]. The first extension ("Join Stream at Router") moves the join request in-band and allows routers that are part of the multicast tree to act as senders and to issue a *connect* request to the new receiver. The second extension ("Create Path Backwards") allows receivers to initiate reverse connection establishments to either the origin or routers in the multicast tree. The second approach is similar to the approach used in Suite 2, where a join by a new receiver causes the routing server to be contacted, which returns a route that is used by the new receiver to establish a connection to the most convenient router in the multicast tree.

6 Summary

When designing the connection establishment component of a multi-party real-time communication protocol, the following issues must be addressed: scalability and efficiency, to provide service to a potentially large number of participants; flexibility, to cope with dynamic changes in the underlying media or in the set of participants; and manageability, to control resource usage. In this paper we described a number of design decisions made when addressing these issues in the design of the Tenet Suite 2, which provides multicast connections, resource sharing, partial and incremental establishment, resource partitioning, and advance reservation.

The preliminary results we have obtained by measuring a prototype implementation of Suite 2 show that Tenet-style multicast channels can be established with delays that are comparable to WAN propagation delays. Even when including relatively long propagation latencies and a relatively large number of hops, multicast channel establishment should appear nearly instantaneous to human users.

References

1. Anindo Banerjea, Domenico Ferrari, Bruce Mah, Mark Moran, Dinesh Verma, and Hui Zhang. The Tenet real-time protocol suite: Design, implementation, and experiences. Technical Report TR-94-059, International Computer Science Institute, Berkeley, California, November 1994. To appear in *IEEE/ACM Transactions on Networking.*
2. S. Deering. Host extensions for IP multicasting. RFC-1112, August 1989.
3. Luca Delgrossi, Ralf Guido Herrtwich, Frank Oliver Hoffmann, Sibylle Schaller. Receiver-Initiated Communication with ST-II, Technical Report 43.9314, IBM European Networking Center, Heidelberg, 1993.
4. Domenico Ferrari. Real-time communication in packet switching wide-area networks. Technical Report TR-89-022, International Computer Science Institute, Berkeley, California, May 1989.
5. Domenico Ferrari. Client requirements for real-time communication services. *IEEE Communications Magazine*, Vol. 28, No. 11, pp. 65–72, November 1990.
6. Domenico Ferrari, Anindo Banerjea, and Hui Zhang. Network support for multimedia: a discussion of the Tenet approach. *Computer Networks and ISDN Systems*, Vol. 20, pp. 1267–1280, 1994.
7. Domenico Ferrari, Amit Gupta, and Giorgio Ventre. Distributed advance reservations of real-time connections. *Proceedings of NOSSDAV 95*, Durham, NH, April 1995.
8. Domenico Ferrari and Dinesh Verma. Buffer allocation for realtime channels in packet switching networks. Technical Report TR-90-022, International Computer Science Institute, Berkeley, California, June 1990.
9. Amit Gupta, Wingwai Howe, Mark Moran, and Quyen Nguyen. Resource sharing in multi-party realtime communication. *Proceedings of INFOCOM 95*, Boston, MA, April 1995.
10. Claudio Topolcic. Experimental Internet Stream Protocol, Version 2 (ST-II). RFC 1190, October 1990.
11. Dinesh Verma. Guaranteed performance communication in high speed networks. Ph.D. Dissertation, Technical Report UCB/CSD 91/663, University of California, Berkeley, December 1991.
12. Lixia Zhang, Steve Deering, Deborah Estrin, Scott Shenker, and Daniel Zappala. RSVP: A New Resource Reservation Protocol. *IEEE Networks Magazine*, Vol. 30, No. 9, pp. 8–18, September 1993.

This article was processed using the LaTeX macro package with LLNCS style