Client-Network Interaction in a Real-Time Communication Environment*

Jean Ramaekers† Giorgio Ventre

The Tenet Group
International Computer Science Institute
and
Computer Science Division
University of California, Berkeley
1947 Center Street, Suite 600
Berkeley, CA, 94704

ABSTRACT In the recent years new protocols and algorithms have been proposed in order to guarantee performance and reliability in high-speed computer networks, and new services have been presented to allow cooperative office work, distributed conferencing, etc. Less attention has been paid to how applications and, more generally, clients of real-time communication services can interact with the network in order to specify and negotiate the quality of service (QOS) of a connection. We believe that this problem is going to become a key issue for the success of future distributed systems, since it affects both client and network performances. In this paper we present a model for the client-network interaction developed for the Tenet real-time protocol suite; the model includes new mechanisms for the establishment and runtime management of real-time connections. By improving the information exchanged between the network and the clients, the model allows to reduce the complexity and the time required to establish a real-time connection, and increases the network utilization. Additionally, we introduced a new class of real-time communication service to support adaptive quality of service, in order to enhance the chances of the network to face congestion situations.

I. Introduction

The number of applications that require a distributed computing system is increasing. In the case of large numerical computations, for example, the need for accessing higher computing power requires that a large number of computers can communicate and cooperate through a high-bandwidth communication system [2]. In the same way, the success of multimedia services such as remote collaborative work and teleconferencing essentially relies on the availability of an efficient network [9]. Efficiency is generally measured in terms of communication bandwidth, reliability, and delay. In particular, limitation and control of delay seem to be the main issues for all the problems requiring real-time data communication.

Several protocols have been presented in literature that allow a client to attain from a distributed system a service with guaranteed performance for what concerns communication [8, 14, 5, 1]. From a general point of view, the solutions proposed are characterized by a model in which communication is realized by means of a channel abstraction. The establishment of a channel and the traffic to be transmitted through it are controlled by the network by imposing limitation on the throughput and/or introducing an admission policy. If the communication support provided by the network is not a best-effort service but it is able to assure quality of service, the general form of the interaction between the client and the network necessitates an establishment phase, in which the client requests the network to have a connection characterized by some performance requirements and the network verifies whether this request can be accommodated [6]. In the Tenet real-time protocol suite [8, 7], this phase can be summarized in the following steps:

1. The client determines the values of the communication parameters characterizing a connection to be established with one or more destination nodes. The client is presumed not to have, in general, any previous information on the network state.
2. The network receives the request and evaluates it. This evaluation is not centralized in a particular node, but it is performed by submitting the request to each node on the path selected between the source and the destination. If in a node are available sufficient processing power and resources to accept the new connection without compromising the performance of the already existing channels, an adequate amount of resources is conditionally reserved.
3. The request proceeds toward the destination until one of the following cases happens: (*i*) one of the nodes along the path cannot accommodate the new channel, (*ii*) the destination is reached. In the first case an answer is elaborated by that node, stating the rejection of the request. In the second case, some final tests are executed at the destination.
4. Starting from the last node reached, an answer message proceeds backward to the requesting host, following the

*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 representing official policies, either expressed or implied, of the U.S. Government or any of the sponsoring organizations.

†On sabbatical leave from University of Namur, Belgium

same path of the request message. If the answer contains a rejection message, the resources tentatively reserved are released, otherwise they are confirmed and the channel is established.

In general, it is assumed that the client should proceed following a converging algorithm. First the client sends a request for a channel which will best fit his traffic and performance requirements. Then the network replies with a *channel accepted* or a *channel rejected* message. If the request has been refused, then the client has two possibilities; the first one is to repeat the same request until it will be accepted. In addition to generate a flood of establishment related traffic on the network, this solution causes a waste of time and computing resources for the client, which periodically has to submit the request, and for the network, which has to accomodate it. Since rejections of channel establishments are most likely to happen during periods of network congestion, this procedure seems to be particularly inappropriate. The second possibility is that the client modifies the request, if this is feasible, by changing the value of some communication requirements and submits it again to the network. This solution appears to be more logical than the first one; however, in the existing real-time protocols a client receives no or little information from a rejection message. Consequently, the preparation of a second "guess" cannot take advantage of it.

In this paper we present a new model for the interaction between a network providing quality of service communication and a client who wants to implement distributed applications with guaranteed performance. We propose three major mechanisms to enhance the utilization of the network resources, increase the number of channel successfully established in the network, and reduce the time required for the establishment. The mechanisms are: *a)* improve the information given to clients by the network when a channel request is rejected; *b)* modify the channel establishment algorithm; *c)* introduce a new class of guaranteed communication service.

II. The Real-Time Network Model

The model of network we assume is the one presented in [7]. In this model, a client asks the network for the establishment of a connection between the source and the destination nodes on which data will be routed in form of packets. The communication service offered to the client is a real-time service, in the sense that the performance of each connection should conform to the requirements expressed by the client when the service establishment is negotiated. These requirements are expressed with two sets of parameters.

The first set can be defined as *traffic requirements.* It includes the minimum interarrival times for the transmitted packets, x_{min}, the minimum value of the average packet interarrival time, x_{ave}, and the interval I on which these value have been computed. By giving these parameters to the network, the client communicates the characteristics of his traffic and his pledge to conform to them for all the duration of the communication.

The second set contains the *performance requirements.* These requirements are expressed in terms of transmission delay and reliability. The first type of requirement can consist of an absolute bound on the source-destination transmission delay, D_{max}. In this case the channel is called *deterministic*; if a deterministic channel j established, in each node n the packets on the channel will be subjected to a maximum delay $d_{j,n}$ computed at the end of the establishment phase. Alternatively, a requirement on the expected delay can be expressed in terms of a maximum value D_{max} to be satisfied by the packets with a probability greater than or equal to a value Z_{min} (*statistical* channel). Or can be expressed in terms of a limit J_{max} on the variance, or jitter, of a delay with a maximum value of D_{max} (*bounded jitter* channel).

The reliability of a channel is specified in probabilistic terms. In particular it is given as a bound W_{min} on the losses of packets in the network due to buffer overflows in the nodes[1].

The establishment of a channel with guaranteed performances is assured only if the network is able to respect the expected figures. This is verified by executing a group of tests, a detailed description of which can be found in [8, 7]. The main goal of these tests is to verify that each node on the path has enough resources to allow the respect of the expected performances without jeopardizing the already existing channels. Two of these tests are performed in the intermediate nodes along the channel path, and one is executed at the destination.

The first test is called *deterministic*; it consists in verifying that the node has enough processing power to accomodate the new channel. This control is executed in the worst possible load condition, i.e. when all the channels are sending packets at the maximum rate. If t_j is the service time required to process packets from deterministic channel j, the condition is that in node n

$$\sum_{j=1}^{C} \frac{t_j}{x_{min,j}} \leq 1 \tag{1}$$

where the sum is extended to all the C channels passing through that node, including the new one.

The second test to be performed in the nodes is the *delay bound* test. It is required to evaluate the minimum delay to be assigned to each channel in order to have enough time to serve all the channels passing for that node. Channels already established are divided into two sets, U and V; a channel j is included in set U if its local delay bound $d_{j,n}$ is lower than the sum of the service time of all the channels. A lower bound $d^l_{j,n}$ for the minimum delay is given by

$$d^l_{j,n} = \sum_{i=1}^{j} t_{i,n} + T \tag{2}$$

where T is the largest service time for the channels in the set V. Condition 2 must be verified for all the channels included in the set U. This value is included in the establishment message and will be used by the destination in the final part of the establishment phase to evaluate the delay $d_{j,n}$ to be assigned to that node for the new channel.

When the establishment message reaches the destination node, it must be verified that the sum of the lower bounds $d^l_{j,n}$ computed in each node for the delay is compatible with the value $D_{max,j}$ requested by the client for channel j:

$$D_{max,j} \geq \sum_{n=1}^{N} d^l_{j,n} \tag{3}$$

where the sum is extended to all the N nodes traversed by the channel.

[1] For the sake of brevity, in this paper we will consider only deterministic channels and the related admission control procedure. This limitation, however, does not affect the validity of our proposal for the complete Tenet protocol suite.

```
test = OK
node = origin
request_ = parameter_list
while (node != destination) && (test == OK)
do
    test = compute_local_tests(request_)
    node = succ(node)
done
if test != OK   /* cannot allocate the new channel */
    while node != origin
    do
          node = prev(node)
          release_resources(node)
    done
else            /* request arrived to destination */
    test = compute_final_test(request_)
    while node != origin
    do
          node = prev(node)
          if test != OK  /* destin. tests failed */
                  release_resources(node)
          else
                  confirm_resources(node)
    done
[]
```

Fig. 1: The RCAP real-time channel establishment algorithm. `request_` is initialized with the traffic and performance requirements `parameter_list` specified by the client. If the local tests are passed, the establishment request is passed to the next node on the path.

Finally, if test 3 is passed, the values to be assigned in each node to the delay and the delay probability are computed and communicated to the nodes in the message sent back to the source to confirm the establishment of the channel.

III. The Client-Network Interaction

A complete suite of real-time communication protocols has been developed on the model we presented above [10, 11]. In particular, the procedures for the establishment and the management of the channels are included in the Real-Time Channel Administration Protocol (RCAP) [3]. The algorithm implemented by RCAP to establish a channel is briefly presented in Fig. 1.

Our idea is to exploit the information produced by RCAP during the establishment phase. When a client requests the establishment of a channel, in the case of a negative answer, he can attain from the network not only a brief answer stating the rejection of the request, but a set of very detailed information on the local state of the network at that moment. The information is expressed in terms of resources available in the node; in this way the client can use this information to elaborate a new request with different characteristics.

Since the parameters given by the client specify the characteristics of the traffic to be sent on that channel and the quality of the service to be provided by the network, during the establishment of the channel the client can act on both sets of parameters in order to adapt them to the current network load. If he acts on the traffic set, he accepts to modify the characteristics of the produced traffic, trading this adaptation with the guarantee for a QOS exactly conforming to the performance specifications. If the client modifies the performance requirements, the characteristics of the traffic will be maintained, but a certain level of degradation in the communication service is accepted. It is up to the client to evaluate which solution is best suited to his application, or to follow both approaches.

Let us consider the request for a deterministic channel. In this case in each node the request must first satisfy equation 1. If the node is not able to accomodate the incoming request, then a negative answer must be sent back to the client. However, if in this answer an information about what kind of test failed and the amount of the still available resources is included, the client can have a more precise hint about how to have his needs satisfied. In the case of a fail in the deterministic test, the rejecting node could send back to the client a precise figure about the processing power currently available. From equation 1, if there are $n-1$ channels already passing for that node, we have that the test is passed if

$$1-\sum_{j=1}^{n-1}\frac{t_j}{x_{min,j}} \geq \frac{t_n}{x_{min,n}} \tag{4}$$

The value of the left term in 4 could be included in the rejection message prepared by the node, so that the network client can accordingly modify the original request. In this case, the client has to reduce the $\frac{t_n}{x_{min,n}}$ ratio he requested for the new channel. To do so, he can increase the minimum interarrival time of his packets; for a multimedia application such as video conferencing, we can obtain this effect for example by reducing the number of video frames transmitted in a second. In this case the modification will decrease the quality of the transmitted data, but it should ensure that the request will be accepted by the network.

In a similar way, if the channel establishment is rejected at the destination due to a fail in test 3, in the rejection message should be included the value $\sum_{n=1}^{N} d^l_{j,n}$. This value is the delay that the network can guarantee for that channel in the current situation, and can be used by the client to correspondingly modify the channel request.

However, the information received by the client can be dramatically increased if we change the channel establishment algorithm. A problem of the original establishment procedure is that it is interrupted when a node without enough computational resources is encountered. Even with the modification proposed above, the client can have a detailed information about the exact amount of resources available in that node, but he still does not have data about the successive part of the channel path. This means that a channel request modified accordingly to a negative answer of a node is not assured to be accepted. In addition, the resources available in the nodes during the previous establishment attempts could have been allocated to other incoming requests, and could not be further available.

To solve this problem we have designed a new establishment algorithm that reduces the call-blocking probability for a new channel. In this algorithm the establishment message proceeds through the network until the destination is reached, even though rejections have been received from some nodes along the path. The information about the rejecting nodes is collected and passed to the client with the rejection message.

If a node is not able to accommodate the new channel, the reason for the rejection and the results of the local tests are included in the establishment message. When the message reaches the destination, the final tests are always performed; if these tests and the ones executed previously in the intermediate nodes give all positive answers, a *request accepted*

```
node = origin
request_[origin] = parameter_list
while (node != destination)
do
    request_[node] = compute_local_tests(request_)
    node = succ(node)
done
request_[destination] = compute_final_test(request_)
return_=prepare_(request_)
while node != origin
do
    node = prev(node)
    if return_ != OK   /* at least one test failed */
           release_resources(node)
    else
           confirm_resources(node)
done
[]
```

Fig. 2: The proposed RCAP establishment algorithm. `request_` initially contains only the parameters specifying the quality of service requested by the client. During the establishment phase it accepts the results of the tests performed in the nodes. At the destination, the answer of the network is prepared. If the channel request is rejected the answer is included in `return_` together with the data concerning the nodes where the request has been rejected.

message is prepared. Otherwise the information about the rejecting nodes is collected and sent back to the client inside a rejection message. A brief description of the proposed algorithm is presented in Fig. 2.

To avoid the problem of competing again for resources that originally were available, the resources reserved during a rejected establishment message can be not released immediately, and be kept available to the same client for a limited amount of time. This time has to be evaluated carefully, in order to avoid that these resources can be unavailable to other clients too long.

IV. A New Class of Real-Time Communication Service

If we analyze the establishment phase of a real-time connection, we see that the chances a new channel has to be established rely on the resources left by the channels previously accepted. To increase the probability to have a channel accepted by the network, we must increase the flexibility of our model, in order to allow the network to adjust the distribution of its resources to both the characteristics of the existing traffic and the requests of new clients.

Flexibility can be introduced by exploiting the characteristics of real-time traffic. An idea emerging from the research community is that future high-speed networks will have two different classes of clients for real-time communication services [4]. Clients with firm QOS requirements for their connections, who will not accept any degradation, albeit predictable, of the service provided by the network, and clients whose applications have less restrictive requirements.

Consider now an application that needs to establish a deterministic real-time channel c_α. During the establishment phase of this channel, characterized by the $x_{min,\alpha}$, $x_{ave,\alpha}$, t_α set of parameters, and a maximum delay bound of $D_{max,\alpha}$, the deterministic test on node j fails. This is due to the fact that M real time channels are already passing through that node, and the resources available in it are saturated.

Our proposal constists in introducing in the Tenet protocol suite a new type of real-time communication service. This type, that we called flexible, admits a certain flexibility in the quality of service the network commits to assure. This flexibility is expressed by the client by specifying ranges of acceptable performance guarantees instead of single values for the parameters in the establishment request. For example, a delay requirement can be given as a range of acceptable delay bounds. If the client accepts that its traffic characteristics can vary in a given range from the desired value, the network can use this range to let the new channel fit in the available resources. A new establishment algorithm has been designed that allows the network to exploit the flexibility of this new class of clients.

Coming back to the previous example, flexibility could be used to allow the satisfaction of the deterministic test, and should interest the packet processing time t_α, and/or the minimum packet interarrival time $x_{min,\alpha}$ of the new channel c_α. From equation 4, the network can attain the new value for the $\frac{t_\alpha}{x_{min,\alpha}}$ ratio to be adopted by the new channel. The new values for t_α and $x_{min,\alpha}$ are computed and, if they are comprised in the ranges specified by the client, they are included in the request message to substitute the original ones, otherwise a rejection message is sent back to the source. Of course the ranges originally expressed by the client are also modified, to reflect the modification already made. If along the path the deterministic test fails in another node, the procedure is repeated. If the destination is successfully reached, a message is sent back to the channel source, containing the final values of the parameters. These values are used by the nodes along the channel path to modify accordingly the resources reserved for that channel, and by the source to adapt its traffic. The same procedure can be applied for the delay requirement.

A similar idea of allowing flexible requirements has been proposed earlier [5, 4]. However, the major difference is that in our approach the network is still committed to guarantee a reliable and predictable service strictly within the performance ranges specified by the client.

V. Conclusions

In a performance-guaranteed network a client-network interaction is desirable from both the client and the network point of view. A client must specify the characteristics of his traffic and the performance he requires for the communication in order to have a commitment from the network. In this way the network can evaluate the amount of resources required to accommodate the new channel. However, if this scheme is kept rigid, the possibilities for a client to interact effectively with the network and to take advantage from the knowledge of its current state are minimal and so are the chances to fully exploit the network resources.

Different solutions have been proposed for the definition of an interface between a real-time network and its clients [13, 1, 3]. Problems related to the negotiation of the communication service between the network and the clients have been only partially addressed [12].

In this paper we have shown a new model for the client-network interaction developed for the Tenet real-time protocol suite. The solution we propose is based on three ideas. The first idea is to increase the amount of data that a client obtain from the network when the request for a communication is rejected. The second is an improved procedure for channel establishment. The last is the introduction of a more flexible class of real-time communication service. We believe that the introduction of these mechanisms will reduce the probability of service denial due to "call blocking" and the time required for the establishment of a channel, allowing a better utilization of the network resources. We plan to use simulation to verify these effects.

Acknowledgments

The authors are grateful to all members of the Tenet Group, and in particular to Domenico Ferrari, Amit Gupta, Jorg Liebeherr, and Hui Zhang for their comments and suggestions on this paper.

References

[1] D. Anderson, R. Herrtwich, and C. Shaefer. SRP: A Resource Reservation Protocol for Guaranteed-Performance Communication in the Internet. Technical Report TR-90-006, International Computer Science Institute, February 1990.

[2] E.H. Bal, J.G. Steiner, and A.S. Tanenbaum. Programming Languages for Distributed Computing Systems. *ACM Computing Surveys*, 21(3), September 1989.

[3] A. Banerjea and B. A. Mah. The Design of a Real-Time Channel Administration Protocol. In *Proceedings of Second International Workshop on Network and Operating System Support for Digital Audio and Video*, Heidelberg, November 1991.

[4] D. D. Clark, S. Schenker, and L. Zhang. Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism. In *Proceedings of SIGCOMM 92*, 1992.

[5] C. Topolcic ed. Experimental Internet Stream Protocol, Version 2 (ST-II). *Networking Working Group, Request for Comments*, N. 1190, October 1990.

[6] D. Ferrari. Client Requirements for Real-Time Communication Services. *IEEE Communication Magazine*, 11(11), November 1988.

[7] D. Ferrari. Real-Time Communication in an Internetwork. Technical Report TR-92-001, International Computer Science Institute, January 1992.

[8] D. Ferrari and D. Verma. A Scheme for Real-Time Channel Establishment in Wide-Area Networks. *IEEE Journal on Selected Areas in Communications*, 8(3), April 1990.

[9] T. D. C. Little and A. Ghafoor. Network Considerations for Distributed Multimedia Object Composition and Communication. *IEEE Network Magazine*, November 1990.

[10] C. L. Lowery. Protocols for Providing Performance Guarantess in a Packet Switching Internet. Technical Report TR-91-002, International Computer Science Institute, January 1991.

[11] M. Moran and B. Wolfinger. A Continuous Media Data Transport Service for Real-Time Communication in High Speed Networks. In *Proceedings of Second International Workshop on Network and Operating System Support for Digital Audio and Video*, Heidelberg, November 1991.

[12] C. Partridge and S. Pink. An Implementation of the Revised Internet Stream Protocol (ST-2). In *Proceedings of Second International Workshop on Network and Operating System Support for Digital Audio and Video*, Heidelberg, November 1991.

[13] G. M. Parulkar and J. S. Turner. Toward a Framework for High-Speed Communication in a Heterogeneous Networking Environment. *IEEE Network Magazine*, March 1990.

[14] L. Zhang. VirtualClock: A New Traffic Control Algorithm for Packet-Switched Networks. *ACM Trans. on Computer Systems*, 9(2), May 1991.