Fast Establishment of Real-Time Channels

Spiridon Damaskos

Gesellschaft fur Mathematik und Datenverarbeitung mbH
Berlin, West Germany

Dinesh C.Verma

Computer Science Division
EECS Department
University of California
and International Computer Science Institute
Berkeley, California

Abstract

A real-time channel[Fer89a] is a simplex connection between two nodes characterized by parameters representing the performance requirements of the client. In this paper, we consider fast establishment of real-time channels, i.e. data can be sent on a real-time channel without waiting for a connection establishment to be confirmed by the destination.

This research has been supported in part by the Defense Advanced Research Projects Agency (DoD), ARPA Order No. 4871, monitored by Naval Electronic Systems Command under Contract No. N00039-84-C-0089, the University of California with a MICRO Program grant, Cray Research, Inc., Hitachi, Ltd., IBM Corporation, Ing. C. Olivetti & C., S.p.A., 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 any of the sponsoring organizations or of the U.S. Government.

1. Introduction

A real-time channel is a simplex connection with guaranteed performance parameters. Such a channel can be established via the scheme in [Fer89a]. A scheme to multiplex a number of real-time channels can be found in [Dam89]. In this paper, we consider the issue of fast establishment of real time channels. Fast establishment is defined as the ability to deliver packets on a channel while its establishment is not yet complete.

In the later part of this section we look at the conditions necessary for fast establishment to be possible in a general network. The next section describes the environment we assume for providing a real-time service. An obvious definition for fast establishment will be to send some data along with the establishment request. Since channel establishment tests may take substantially longer time than normal packet processing, the first packet may experience a large delay. The next section discusses an alternative scheme in which the data and establishment packets are separated. In order to avoid the arbitrary choice of a parameter required for the second scheme, we propose the third and final scheme in section 5. Finally, we attempt to analyze the situations where fast establishment might be a win over a normal establishment.

We now try to examine the properties of the connections which are amenable for fast establishment. Some connections negotiate critical parameters at connection establishment time e.g. the sequence number of the first packet to be sent. A packet can not be sent on such a connection until its sequence number is known. Obviously, such a connection is not suitable for fast establishment.

The general establishment scheme can be represented as following. On receiving a connection request, each intermediate node needs to prepare the context for the connection. The context will keep the information about connection numbers, any resources reserved by the channel etc. depending on the properties of the connection. Let us call these as context variables. On the successful establishment of the connection, the final values of the context variables are determined. While the connection is pending, the value of a context variable may be different from the final value. A context variable may be of the following types:

Exact critical: The exact value of the context variable is critical for correct transmission of data. An example of such a variable is the initial sequence number[Dal78] of the first packet on the channel.

Lower value critical: Correct transmission of data is possible if the context variable has a value lower than the agreed-upon value, but not otherwise. An example of such a variable is local delay bounds in real-time channel establishment scheme[Fer89a].

Higher value critical: Correct transmission of data is possible if the context variable has a value higher than the agreed-upon value, but not otherwise. An example of such a variable is the buffer allocated for real-time channels with guaranteed loss-rates.

Not critical: An agreement for the context variable is not essential for the correct operation of the protocol. The node may make a local choice for the context variable. An example of such a context variable is some kind of index used by the local node to find the context entry in a fast manner.

A connection is amenable for fast establishment only when there are no *exact critical* context variables to be negotiated.

The final value of the context variables may be established in a number of passes. Current connection establishment requests generally consist of 2 or 3 passes, (a 2 or 3 way hand-shake).

2. Environment

Connection management (i.e. establishing the context) is a difficult problem and a number of problems arise in practise due to lost and/or duplicated connection. Three different mechanisms are used to ensure correct operation, 3-way hand-shakes, unique connections identifiers and timer based protocols [Wat89].

A 3-way hand-shake is not appropriate for fast connection establishment since it is required only when negotiating a crucial context variable. We propose to use unique connection identifiers and timers to resolve the problems of correct connection management when opening a connection. A fast establishment is not possible if we need a three way hand-shake. For the generation of unique connection-ids, we adopt a very simple scheme.

The connection-id of a real-time channel consists of 3 parts, specifying its source, destination and the time at the source when the connection request was generated.

Since the connection-id can only be generated by the source, the scheme requires that a failed real-time channel request has to be returned back to the source. [1] If the establishment request has succeeded at an intermediate node B, but rejected by a subsequent node C, B can not try to establish this channel through a node other than C. This request has to be returned to the source who may retry it with a different connection identifier. This is necessary because it is not possible to clean-up the effects of a rejected channel establishment request in the presence of lost messages.

We assume an environment where the processing of an establishment request does not interfere with the transmission of data messages. Thus, each node on our hypothetical network has two modules, a manager and an operator. The manager is responsible for accepting/rejecting connections through the node. The operator is responsible for calculating the deadline of packets and transmitting them over the network. An establishment packet has a flag associated with it, which causes it to be delivered to the manager instead of the operator. Packets can only be transmitted by the operator, which means that the channel manager must hand over any establishment requests to the channel operator if it needs to be forwarded to another node. The scheme is similar to datakit where the connection establishment is handled by a controller computer, and does not affect the transmission of data on the established channels.

The transmission of establishment requests can still interfere with the delay guarantees made to each connection. In order to eliminate this effect, we assume that the establishment message is accorded the same priority as the packet on the channel it requests. It means that a request for a deterministic channel is accorded the priority of a deterministic packet. If the node calculates that the packets on a channel can be guaranteed a delay of 5 time units, and the establishment tests are completed successfully at time instant 20, the establishment packet will be put in the deterministic queue with a deadline of 25. A statistical (or best-effort) channel establishment request is treated as a statistical (or best-effort) packet with a deadline computed similarly. Since an establishment request is transmitted to the operator only when the operator is able to meet the deadlines of all the other packets in the presence of packets from the new channel, transmission of successful establishment messages does not interfere with the transmission of data packets, provided the first packet on the channel arrives after an interval of $x_{\min}$ or more after the establishment tests are completed.

A rejected channel request can not be safely transmitted back to the source at the same priority. It can be treated as a best-effort packet. The disadvantage is that the resources reserved at the nodes already covered successfully by the establishment message may be tied up for a longer period.

For the moment, we also assume that there is a common table describing the channels established through the node shared between the manager and the operator. This table can be used by the operator to find out the local delay bounds for calculating packet deadlines and determining the route. This means that we are not considering self-routing packets for the moment.

---

[1] We assume that the source is both the node requesting the channel and sending the data. In the other case, when a node establishes a channel to receive data, fast open is not an issue.

3. Scheme I

The establishment of a real-time channel is done in two stages. During the forward movement of the establishment request, each node specifies a lower limit on the local delay bounds of packets on the channel. It may also reject the channel if no such bound can be assigned. The destination node does the final allocation of the delays for the intermediate nodes, using the delay requirements of the channel to increase, but not decrease, the bound on the local delays of the packets. On the reverse path, each node knows the final value of the local delay bound.

Two problems need to be solved when we try to establish a fast connection. The first one is to ensure that the packets that are transmitted before the channel is committed satisfy the performance requirements of the channels. The other problem is to make sure that the packets from an earlier unsuccessful attempt to establish the connection do not interfere with the final connection. The second problem can be solved by the use of unique connection identifiers, and we concentrate on the first one.

In the context of real-time channels, the per channel final local delay bounds, which are used to calculate the deadlines of packets on the channel are only known at the commitment time. However, for the correct operation of the channels, we only need to ensure that the local delay bound used before the channel is committed is lower (or equal) than the final delay bounds.

Since delays are lower-value critical and buffer-space is higher-value critical, we can use the lower bounds on delay and the upper bound on the buffer space instead of the final agreed-upon values for the delays. Correctness is guaranteed by this method. However, the time taken by the establishment tests has to be accounted for as well. If the establishment tests take too much time then we can not meet the local delay bounds for the first packet.

We consider fast establishment for deterministic channels first. We ignore for the time being the problems of packet loss due to buffer overruns and concern ourselves only with delay guarantees. All the packets that arrive at an intermediate node before the channel is finally established have to be delivered before the channel's deadline. Since the local delay bounds for the channel are more conservative than the final delay bounds, these packets will reach the destination in the proper time. But we must also ensure that the packet which established the connection arrived within the required deadline. Equivalently, if the establishment packet does not arrive at the destination before the required delay bound, the establishment request is rejected.

On the receipt of a channel request, the node must do the following

1) record t_a, the time of arrival of the request.

2) calculate d, the lower bound on the local delay bound of the channel as given by the scheme in [Fer89a].

3) note the time when the tests are completed i.e. the current time t_c

4) increment d by $t_c - t_a$. Thus we use $d' = d + t_c - t_a$ as the new proposed local delay bound.

5) Assign to the packet a deadline of $t_a + d'$

For example, suppose we are establishing a deterministic channel with an $x_{\min}$ of 5. The first packet arrives at the intermediate node at time 0. The node makes all the required calculations and determine that a delay bound of 6 is the minimum guarantee it can provide to a packet on this channel. The time taken in making these tests is 2 units, so the current time is 2. The first packet will have to get a deadline of d' i.e. (2 + 6) or 8. The total delay of the first packet can be as high as 8 time units.

If the second packet arrives at time 5, it can not be granted a deadline of $t_a + d$ i.e. (5 + 6) or 11. In this case, the deadline of the first packet is only 3 units less than the second one, inconsistent with the value of this interval assumed during the test, namely the $x_{\min}$ value of 5. This packet must get a deadline of $t_a + d'$, i.e. 5 + 8 or 13.

If at time instance 21, the channel acceptance message arrives from the destination with a final value of 10 for the local delay bound, the final value can be used for all subsequent packets. Thus if packets 2, 3, 4 and 5 had arrived at time instants 5, 10, 15 and 20 respectively, they would have been assigned a deadlines of 13, 18, 23 and 28 respectively. Packet 6, which can arrive at time instant 25 will get a deadline of 35.

Packets 4 and 5 might still be with the operator when the request was committed. It is debatable whether their deadline should be increased. It will increase the complexity of the operator, and the advantages gained may not be of much importance. We believe that it will be better if their deadlines are left as they are.

It may seem at the first glance that proposing a delay bound of 8 is being too conservative when the computed delay bound is merely 6. However, if the first packet contains data and establishment request, the manager can not hand it over to the operator without performing the establishment tests. The first packet will have to face this extra delay at every intermediate node. Hence, it is a cost which can not be avoided.

Moreover, if the time to perform the tests exceeds the delay bound that is computed, e.g. suppose that the tests took 10 time units instead of 2, a delay bound of 6 is clearly infeasible. So the first packet must be assigned the higher deadline as required in step 5.

For the correct operation of the delay-bound algorithm, we require that the deadlines of any two packets differ by at least $x_{\min}$ value specified by the channel client. For this reason, it is not safe to use the value of d to compute the deadline for the second packet on the channel and by induction on any packet that arrives at the node before the channel is committed. We have to use the value of d' to compute the local deadline.

At the time of commitment, the delay bound can only be increased and not decreased. For this reason, the destination node must allow a local deadline greater or equal than $t_c - t_a + d$ to every intermediate node. It suffices to increase the local deadline by that amount.

For the scheme to work correctly, each node must be able to record the time of arrival of a packet. Since this has to be done to determine the deadline of the packet anyway, it seems that fast establishment is feasible on any node where the real-time channel establishment scheme of [Fer89a] is feasible.

For a statistical channel, increasing the computed value d of the local delay bound to d' may seem too conservative. It is necessary however, since the time taken for establishment tests may be larger than the computed delay bounds. This delay can ripple down to later packets because of the requirement that the deadlines of any two packets differ by $x_{\min}$. This effect will persist until the channel becomes inactive, i.e. it stops sending packets at the peak rate for some time. If more information about the traffic pattern of the statistical channel is available, e.g. a distribution of the burst lengths, it may be possible to use some value between d and d' as the suggested local delay bound for the channel.

4. Scheme II

The scheme mentioned in the previous section works well if the tests are fast, i.e. the time taken to perform the tests is of the same order as the service time of a packet. In general, this is not a safe assumption because the time taken for the tests may run up to several milliseconds[Fer89b]. The service time of a packet on a fast network like an ATM network will be in the order of microseconds. In this case the delay bounds will be too large and thus rather meaningless.

If the first packet contains both the data and the establishment request, it is obvious that this packet can not be forwarded from one node to the next without performing the tests. In this case, the delay bound for deterministic channels has to include the time taken for the tests at each node. Furthermore, since the test-times at all the nodes accumulate, the end-to-end delay bound is likely to be much larger than the actual delays of the later packets.

As an example, consider a channel with a large service time, say 100 microseconds. On a fast network (1 Gbps) it will correspond to a packet with size in the order of a hundred Kilobits.

The time for establishment tests on a VAX-8600 is about 4 milliseconds[Fer89b]. If the channel to be established spans 4 nodes, with a VAX-8600 for the manager module, the first packet has to wait for at least 4 milliseconds per node while the establishment tests are being made. If each node calculates the delay bound d to be 1 millisecond, and the propagation delay on the network is 20 milliseconds, the first packet experiences a delay of 20 + 5 * 4 = 40 milliseconds. If the channel ever becomes inactive after its establishment, the packets on the next burst will not feel the effect of the establishment tests. Their delay is bounded by 20 + 5 * 1 i.e. 25 milliseconds.

Fast open is a win only when the propagation delays are relatively high, typically much higher than the time taken by the establishment tests. In this case, adding the establishment test time may be acceptable. However, if the propagation delay is small, then the bounds including the time for establishment may not be acceptable to some clients.

One solution is to force an artificial period of inactivity after the arrival of the establishment packet. This period of inactivity is the round trip establishment time for a normal open. The client need not wait for the connection establishment to be committed. Data packets can be sent x_f time units after the connection establishment request. The case where the first packet contains both the establishment request and data can be looked upon as a special case with x_f as zero.

With this scheme, the traffic on the channel is characterized by 3 values, $x_{\min}$, x_{ave} and x_f. The value of x_f has to be chosen so as to allow the intermediate nodes to have sufficient time to perform the establishment tests.

The establishment packet processing is slower than that of a data packet by $t_e - t_s$, where t_e is the establishment test time and the maximum service time of a channel on the packet is t_s. For the moment, we neglect the case of rapid load fluctuations at the nodes. That is, we do not consider the case where the establishment packet is delayed for a long time by the operator and the first data packet has very small delay. In this case, if no data packet is to overrun the establishment request, which will cause the intermediate node to drop it, we must have that

$$x_f \geq \sum_i t_{e,i} ,$$

with the summation ranging over all the intermediate nodes of the channel.

In the example given above, fast open in this fashion will imply that the client will have to wait only for 16 milliseconds (the time taken for the establishment tests at all the node, of course the source does not know the exact value of the establishment time) after the establishment request to send the packets on the channel rather than the round trip time of 60 milliseconds (20 + 20 for round trip propagation time and 20 for the tests made in the forward passage). Fast open by scheme II is able to save 44 seconds of waiting time and provide a end-to-end delay bound of 24 milliseconds. The first approach saves 60 milliseconds of waiting time but provides a delay bound of 40 milliseconds for every packet on the channel.

With this method, the scheme for fast establishment can be described by the following algorithm. We assume that the establishment request carries the sum of the establishment test times at all the nodes covered so far. This time is referred to as t_g and initialized to zero when the establishment request is made by the channel client.

1) record t_a, the time of arrival of the request.

2) calculate d, the lower bound on the local delay bound of the channel by the scheme in [Fer89a].

3) note the time when the tests are completed i.e the current time t_c

4) Increment t_g in the establishment message by $t_c - t_a$.

5) If $t_g > x_f$, reject channel request.

In the case of the client not adhering to his promise of x_f, the packet will arrive at an intermediate node before the packet establishment tests are completed. In that case, the node will declare this packet to be from an unknown channel and discard it.

A minor modification to improve the chances of acceptance can be made if the channel operator can notify the channel manager if packets on an unknown channel were received. Since we use unique connection identifiers, the node can check in step 5 if a packet on the new channel had arrived before the establishment tests were completed. If so, he rejects the channel, else accepts it. [2]

The problem in scheme II is to determine the appropriate value of x_f. Since the sending node may not be aware of the establishment time taken by the tests, he may not be able to select a *good* value. However, he can choose any random value between the establishment test time at his node and the round trip latency. If the value chosen is not large enough the channel establishment will fail. The arbitrary nature of x_f has to be overcome.

5. Scheme III

We overcome the problem involved in the choice of x_f by using the value of x_f to determine the value of the local delay bounds. If a channel wishes to use an x_f equal to $x_{\min}$, the first packet must tolerate the delay of establishment tests.[3] In this case, the proposed delay bounds for the data packets (d') is greater than the minimum the node can offer (d as calculated by the tests in [Fer89a]) by an amount t_e (the time taken for the establishment tests). However, if the value of x_f is large enough, we can have d' to be the same as d. The present scheme tries not to reject a channel if the source has specified a value of x_f lower than that required by scheme II, but to choose a value of d' between d and $d+t_e$.

We also try to incorporate the case of rapid load fluctuations in the node. The maximum time an establishment request has to wait in node i is bounded by $t_{e,i} + d_i$. The first data packet on the channel may not have any queueing delay and thus pass right through. The latest the establishment packet can arrive at node n on the path is $\sum_i (t_{e,i} + d_i)$. The first packet must not overtake the establishment request by this time. The requirement thus is: [4]

$$\sum_i (t_{e,i} + d_i) \le x_f$$

If the node can not store the packets that arrive before the channel has arrived, it must reject a channel if the previous condition is not met. Assuming that these packets can be stored somewhere, perhaps in a special queue designated for this purpose with the channel manager, the new local delay bound d'_n for the packets on this channel can be obtained by the constraint

$$\sum_i (t_{e,i} + d_i) + d_n + x_{\min} \le x_f + d'_n$$

The formula can be derived if we consider the condition that the deadlines of the establishment packet and the first data packet must differ by at least $x_{\min}$. d'_n is added to the instance of arrival of the first data packet on the channel to obtain its deadline. The deadline of the establishment packet is given by adding d (i.e. the value of delay bound given by the test in [Fer89a]) to the time of arrival of the establishment request at the node. [5]

Thus the local delay bounds for the new channel can be computed by the formula

$$d'_n = d_n + \max[0, \sum_i (t_{e,i} + d_i) + x_{\min} - x_f] \,.$$

---

[2] The channel need not be rejected if the packets that arrive too early can be stored somewhere. In this case, the node can increment the delay bound to the channel by the time taken for the establishment tests.

[3] This value of x_f must be greater than $x_{\min}$ because of the scheme we use to prevent establishment packets from interfering with the delay guarantees of the data packets.

[4] This method to incorporate the effect of load fluctuations can also be used for scheme II.

[5] Since the deadlines of the establishment request are obtained by adding a different value than that of the normal data packets, the operator must handle establishment requests specially, i.e. allow the manager to specify the deadline of the establishment packet for the channel.

This scheme can be looked upon as a generalization of the previous two schemes. However, there must be a special buffer meant only for the packets that arrive before the establishment tests are completed. Otherwise, the channel client must be prepared to face the loss of any packets that he sends before the channel is committed.

6. Cost Analysis

In this section, we consider a brief analysis of the costs involved in a fast open rather than a normal open. We assume a very simple accounting scheme. A client is charged at the rate of c_t units for every unit of time he keeps the connection open after initiating the connection request and at the rate of c_b units for every bit of information transferred on the channel during this period. Suppose the time taken for establishment of the channel (i.e. the time between the source sends a establishment request and receives a accepted/rejected reply) is t_e. If b bits of data are to be transferred on the channel, and the probability of a successful connection establishment is p, the number of attempts required to establish a successful connection is $1/p$. At an average rate of s bits per time unit, an amount of data st_e is wasted for each unsuccessful establishment attempt.

The cost of a fast open is

$$C_f = (1/p)(c_b t_e s + c_t t_e) + c_b(b - st_e) + c_t(b/s - t_e)$$

when the amount of data to be transferred is relatively large. If all the data can be transferred within t_e, then

$$C_f = (1/p)(c_b b + c_t t_e) .$$

The cost of a slow open is simply

$$C_s = c_t t_e / p + c_b b + c_t b / s$$

A fast open is better if $C_f < C_s$. This would imply that the probability of success should satisfy

$$p > \frac{1}{1 + c_t / sc_b} .$$

This implies that for channels that want to transmit at an higher bit-rate, fast open is better only if the probability of acceptance is higher. However, channels of higher bit-rate are more difficult to obtain than channels of lower bit-rate. It appears that fast open will be a win only for channels that want to send at slow bit-rates and not for channels that want to send at high bit-rates. If we know the value of p, then we can determine the rate below which a channel can win by a fast open.

A deterministic channel is usually more difficult to obtain than a statistical or best-effort channel, i.e. between a deterministic channel and a statistical channel with the same traffic characteristics and delay requirements, the deterministic one is less likely to be accepted. Thus fast open does not seem suitable for deterministic channels of high bandwidth. On the other hand, a best-effort channel is almost always accepted. Thus, they are the best bet for fast establishment.

One situation where channel establishment always succeeds is that of multiplexing real-time channels [Dam89]. Since the source only multiplexes channels that the destination will accept, the value of p is 1, hence a fast open is always a win. Of course, this holds only when multiplexing over an already existing real-time channel and not when a new real-time channel has to be requested from the network.

7. References

[Dal78] Yogen Dalal, "Connection Management in Transport Protocols",

[Dam89] S. Damaskos and D. C. Verma, "Multiplexing Real Time Channels", 1989.

[Fer89a] D. Ferrari, "Real Time Communication in Packet Switching Wide Area Networks", Tech. Rept. TR-89-022, International Computer Science Institute, Berkeley, May 1989.

[Fer89b] D. Ferrari and D. C. Verma, "A scheme for Real-Time Channel Establishment in Wide Area Networks", Tech. Rept. TR-89-036, International Computer Science Institute, Berkeley, June 1989.

[Wat89] Richard T Watson, "The Delta-t Protocol", Proceedings of the International Conference on High Performance Networks", Zurich Mar 1989.

Appendix I
Duplex Connections

A real-time channel is a simplex connection between two nodes. The abstraction is useful for many kinds of applications where there is a distinct direction of information flow. However, there are many applications which require a duplex connection, an example being telephone conversations where both the parties should be able to talk to each other. In this section, we try to establish duplex connections between two nodes. using a scheme to establish simplex connections e.g. one given in [Fer89a]. In order to establish duplex connections, we can establish two simplex connections in opposite directions. It seems that this set-up is sufficient to provide a duplex connection with guaranteed performance in both directions.

The major problem in this scheme is that the traffic requirements and characteristics of both directions of communication need to be known. While the entity that requests a connection can specify the traffic pattern of the data he will be sending, he may not be aware of the characteristics of the data he will be receiving, i.e. the data that the other end-point of the connection will be sending.

The most obvious solution is for the initiator (or the caller) of the connection to specify his traffic characteristics and performance requirements. On the successful establishment of the simplex channel in the forward direction, the destination of the call tries to establish a connection in the reverse direction, with its traffic requirements. When the two calls are established successfully, the initiator can send packets on the channel. It may however be required in some cases that both the simplex channels follow the same route.

The scheme takes 2 round trips to establish the connection. Apart from the high latencies, problem arise if the connection in the reverse direction is rejected by some intermediate node. In that case, the forward direction channel will also have to be rejected. The same problem exists even if the establishment acceptance message and the request for the reverse direction channel are combined together.

In the other scheme, the source will specify both his requirements as well as the requirements of the channel in the reverse direction. The destination can relax the specifications made by the source. With the traffic characterization selected in [Fer89a], this would mean that the value of $x_{\min}$ and x_{ave} can be increased, but not decreased. Similarly, the delay guarantees can be increased but not decreased. The destination, which presumably can determine the traffic requirements of the channel in the reverse direction can decide on the characteristics and the final local delay bounds for both the channels at each intermediate node. This scheme takes only one exchange of messages and also keeps the same path for both the channels automatically.

This scheme will work well in the cases where the channels in both directions are symmetric, with almost the same requirements. The scheme will also work if the traffic on the reverse direction is slower than the traffic in the forward direction. In these cases, the node can specify the same attributes for both the channels. Otherwise, the node has to make a guess about the traffic patterns of the reverse direction, which may not be good enough in some cases.