MINIMAL COMPLEXITY FOR THE SIMPLEST PROTOCOL

Christian Huitema *Walid Dabbous*

INRIA Centre de Sophia Antipolis, 2004 Route des Lucioles
BP-109, 06561 VALBONNE FRANCE

Abstract

Multimedia applications require multiple data streams. Real time flows require minimal control while the classical control procedures should be applied for other data flows. In this paper we discuss the design issues of a multimedia transport protocol achieving the multiplexing of both real time and normal data flows at the transport layer. The use of a single transport connection to carry both data flows is proposed. Two schemes for the support of the synchronization of these data flows are presented. Then the protocol TP5 implementing the above mechanisms is described.

1 Introduction

The development of new multimedia applications over high speed LANs, MANs and B-ISDN is a challenge for the early 90's networking research. One of the important problems to solve is to find the communication protocol stack that allow applications to benefit from the available bandwidth of these high speed networks. Classical communication protocols (TCP [1], TP4 [2]) were designed to run over low speed high error rate networks. They include complex flow and error control mechanisms that slow down the transmission below the actual raw bandwidth. With the availability of high speed low error rate networks, these protocols should be revised in order to propagate the bandwidth gain to the applications.

One solution is to "tune" these standard protocols while still keeping their general purpose aspect. In other words, the parameters of the control procedures (e.g. window size, retransmission timer) are tuned in order to enhance the throughput and to avoid congestion or to reduce delays during data transfer [3], [4], [5], [6]. In addition, adequate implementation choices (e.g. minimize context switching) will allow to obtain better performance without losing interoperability [7], [8]. With this scheme we have several full functionality protocol stacks (TCP-IP, TP4-CLNP).

The other alternative is to design new *light weight* protocols tailored to respond to specific applications needs. These light weight protocols are designed with the following concept: in order to go faster the overhead caused by the control procedures should be minimized. For example, this could be performed by replacing the flow control by transmission rate control (i.e base the packet transmission decision on local timers rather than received credits from the destination transport entity as in XTP [9]) and using selective NACKs rather than cumulative ACKs for error control as in NETBLT [10]. See [11] for a survey on these light weight protocols.

But how far can we go in reducing the protocol functionality? Could we imagine a void transport protocol over low error rate high speed networks? This solution could be tempting for multimedia applications including real time

CH2979-3/91/0000-1481 $1.00 © 1991 IEEE

transmission of voice and video because of the error tolerance of these applications. In fact, the real time traffic requires a stream like connection with a minimal control. An important issue here is to define *what* control should be performed for real time traffic and *how*.

2 Transmission control for real time data flows

Classical transport control protocols, such as TCP or TP4, use packet numbering and end to end acknowledgments to detect and correct such network errors as lost or duplicated packets, or losses of sequence ordering, and to provide end to end flow control. These techniques are not well suited for the control of high speed data flows, as they demand important memory and CPU resources, and result in an increase of the transmission delays [12]:

- End to end error correction based on acknowledgments, obliges the sender to keep a copy of all unacknowledged packets. The size of the memory necessary to keep this copy is proportional to the round-trip delay and to the throughput of the network. It would reach 5 Megabytes for a Gbps network with a modest end to end delay of 40 ms.
- The processing of an acknowledgment requires approximately the same computing time as that of a data packet, while computing power is probably a bottleneck for high speed applications.
- Real time data flows require stable transmission delays. Allowing for occasional retransmissions implies delaying all other packets by at least one retransmission delay, which will not only lower the global quality of service but will also require very large resequencing buffers – in fact, the same size as the copies kept for possible retransmission at the sender side.

In order to speed-up the transmission of the real time data, we propose to simplify drastically the protocol by accepting the occasional loss of a certain type of packets. Lost real time packets will not be retransmitted. A numbering procedure will be used to "detect" transmission errors; a resequencing buffer at the receiver side will be used to correct the sequence ordering; a data rate negotiation will replace the end to end flow control. A *partial control* will be performed on the real time flow: it includes a checkpoint mechanism serving as a "global acknowledgement" i.e. to indicate that things are *globally OK* until the precised position.

The gain of the described control scheme is to simplify the protocol design by changing the provided transport service (see figure 1). However, the following design issues should be taken into consideration:

- In order to avoid resequencing confusions, a high sequence modulus should be used. In fact, as the receiver buffers the incoming packets for resequencing, an old packet received late may be taken as a new packet of a later frame. The value of the sequence modulus F should be sufficiently high to prevent this confusion. The checkpoint mechanism allow to avoid such situation if it is performed before a complete rotation of the sequence numbers.
- There is no limitation on the size of transport service data units. However, the small cell size on ATM networks will result in a very large number of segments, thus increasing the packet loss as the loss of a segment means that the entire packet will be discarded.
- The size of the resequencing buffers at the receiver depends on the variance and not on the mean value of the total transmission delay. This will result in

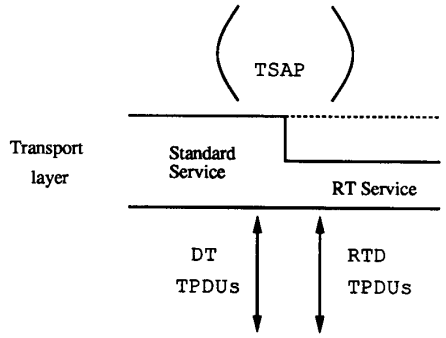

Figure 1: **Modified transport service for multimedia applications**

smaller buffers as the receiver does not have to wait for a retransmission of a lost packet.

3 Multimedia applications requirements

Multimedia applications require multiple data streams with different characteristics to be "integrated" on the same application association. The data and urgent data streams should be securized by retransmissions, while only the partial checkpoint control shall be performed to the real time traffic.

Another question raised here is how to tailor the transport system to respond to the requirements of multimedia applications. Should we have different protocol stacks for the various traffic types or have one protocol to provide the required "multimedia transport service". In the first alternative, the transport system will provide the required functionality by using an appropriate transport protocol i.e. use of a general purpose protocol for normal data transmission and of an enhanced special purpose protocol for file transfer, for transaction based communication or for real time communication. The transport system should in this case be a customizable set of different protocol stacks implementation. The other solution is to avoid this proliferation of protocols by using an "integrated" approach: a super transport protocol where service primitives can be configured on a connection basis, i.e. the transport service user can negotiate the use of the service primitives during the transport connection establishment.

The use of a single transport protocol for multiple streams is conceptually justified by the following reason: the partial control of the real time traffic could be performed at the transport level, and more specifically this control will be ensured by other traffic streams. In fact, as a "real time" transport connection should any how support securized data transfer for checkpoint purpose, the same connection can be used to carry also the normal data flow which will also serve to provide this checkpoint procedure.

A very important characteristic to be studied is how the mutual synchronization of normal and real time data flows can be performed and how this synchronization contribute to the control of real time traffic. One could wonder whether it is suitable to perform transport level synchronization while this function is provided by the session layer primitives. In other words, should we have separate data flows at the transport level and use session level synchronization primitives or multiplex all traffic types on the same transport connection?

The first idea that comes in mind is to establish "typed" transport connections i.e. to have a separate transport connection for each traffic type. Therefore a multimedia application which requires the transmission of both text and video will request two transport connections. One is a controlled connection (normal error and flow control) to carry the text traffic on DaTa Transport Protocol Data Units

(DT TPDUs). The video stream will be transmitted on an other "uncontrolled" connection in Real Time Transport Protocol Data Units (RT TPDUs). This approach requires a synchronization procedure at the session level. In fact, the delays are different and unpredictable on the two separate connections, and session level primitives should be used to synchronize each connection separately. But the session level protocol supposes a securized transport of transport service data units, while data may be lost on the uncontrolled flow without any transport level retransmission. It means that a checkpoint packet on the real time flow may never arrive to its destination. As there is a need to securize the checkpoint packets, they could be sent on the controlled flow. However, as there is no mutual synchronization information between the two streams the checkpoint procedure may also fail (e.g. a transport service user waiting indefinitely for a frame to arrive).

The second way to transmit the two streams is to share a single "multimedia" connection. DT and RT will be considered as two TPDU types, and service primitives performed to a TPDU will depend on its type. DT TPDUs may either contain control data for RT TPDUs or standard data to be synchronized with the RT traffic. This approach minimizes context switching as both streams are multiplexed on the same connection. It also allows for better synchronization between the DT and RT flows as the mutual sequencing of DT and RT data will be used for this purpose. Still to define how this synchronization can be achieved. In section 4 and 5 we present two schemes allowing to perform this synchronization. The first scheme is based on a modification of the normal data packet in order to carry information about real time data packets and is called the "relative sequence" scheme. The other alternative is to add also the synchronization information in the real time data packets, resulting in an "absolute sequence" scheme.

4 Synchronization scheme

One simple way to achieve transport level synchronization is to have a "precedence rule", i.e. establish a *relative sequence* scheme in the transmission of the two flows. The rule can be stated as follows: *Real Time data is "faster" than standard data.* In other words, if a RT TPDU is transmitted before a DT TPDU it will not be received by the peer transport service user after this DT TPDU. It will either be received before the DT TPDU or discarded. The justification of this rule is derived from the time constraint for the RT traffic: RT TPDUs are not allowed to be late, therefore the transport service user should not wait for them if they don't arrive "in time" (before deadline or before last DT TPDU transmitted after them).

In order to implement this scheme, standard DT TPDUs should carry the sequence number of the *next* RT TPDU to send on the transport connection. A new parameter $N(RTS)$ is added in DT TPDUs to designate this sequencing information. At the receiver side the transport entity saves this parameter in a state variable and updates it whenever a DT TPDU is received. This will allow DT TPDUs to perform partial control on RT TPDUs i.e. without stopping the transmission of the RT flow. The scenario of a data transfer of a real time application (e.g. video image transfer) will be as described below:

<u>At the sender side</u>

The transport service user initiates a RT-DATA request when a frame is to be sent. The frame is transmitted as several RT TPDUs followed by a DT TPDU (for partial control purpose). Any of the transmitted TPDUs may not reach the destination. RT TPDUs loss will degrade the quality of the display, it is up to the application to decide whether the degradation is too grave to stop transmission. DT TPDUs are error controlled i.e. retransmitted upon time out if the acknowledgment is not received. If a DT

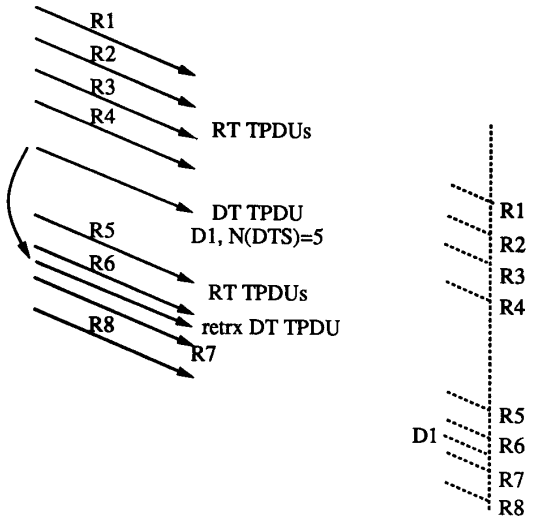

Figure 2: **DT and RT TPDU transfer in the relative sequence scheme**

TPDU containing "normal" data is lost, RT TPDU will continue to be transmitted in the normal sequence. The retransmitted DT TPDU will carry the sequence number of an "old" RT TPDU. Nevertheless, the RT TPDUs transmitted in between will not be discarded as the precedence rule does not prevent that RT TPDUs pass DT TPDUs.

At the receiver side

A RT TPDU will be discarded if it is overtaken by DT TPDU. As DT TPDU loss is noticed at the receiver side only after successful retransmission, RT TPDUs will continue to be accepted in the resequencing buffer then delivered to the application in between. The receipt of the "late" DT TPDU will not cause any RT TPDU loss: the information in the N(RTS) parameter will be considered as an "out of date" indication of status. The protocol providing this service will perform a 3-way handshake procedure, in order to avoid that the "phase jitter" between the two flows causes a complete out of synchronization state. However, if the retransmitted DT TPDU contains an application level synchronization request, the delay will affect the quality of service of the real time display application.

It is clear that the correct behavior of a transport protocol providing the synchronization scheme described above requires an adequate numbering mechanism. Two constraints should be satisfied to ensure the correct reception of RT TPDUs: the first is to have a sufficiently high modulus in the RT TPDUs sequence numbers space to avoid confusion states, and the second is a condition on the DT TPDU rate in order to keep the "full alignment" status.

In order to explain the problem of the circular sequence numbers consider the following scenario: a DT TPDU that was retransmitted N times is received at the destination transport entity. The $N(RTS)$ parameter should be considered as referencing an "old" RT TPDU (i.e. the DT TPDU should be considered as a late TPDU) in order to prevent the discard of all subsequent RT TPDUs. In fact, if the DT TPDU carries a "very old" $N(RTS)$, it may be considered, because of the cyclic numbering mechanism as referencing a "newer" RT TPDU. This means that all subsequent RT TPDUs with sequence numbers lower than the value of the received $N(RTS)$ will be considered as late and therefore will be discarded.

Let N be the maximum number of retransmissions, τ the interval between the transmission of the two successive RT TPDUs (constant for a given transmission rate), T the retransmission time out. When a DT TPDU with $N(RTS) = I$ is retransmitted N times, the maximum number M of RT TPDUs that may be transmitted before successful reception of the DT TPDU at the destination should verify: $I + M > I$. If the modulus of the circular sequence numbers space is F the previous equation translates to $M < F/2$. Replacing M by NT/τ we obtain:

$$F > 2NT/\tau \qquad (1)$$

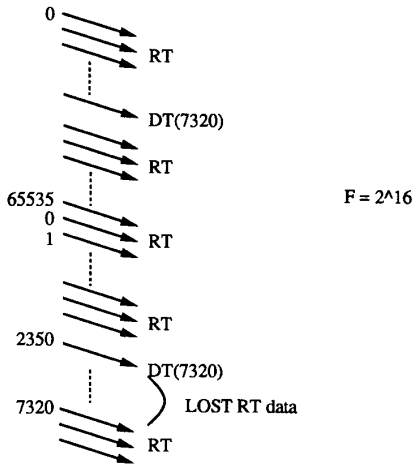

Figure 3: **Real time data loss in the relative sequence scheme**

For a 1 Gbps transmission rate, a TPDU size of 1K, with $N = 10$ and $T = 2$s we have $F > 4.10^7$.

Another constraint should be verified to avoid a sequencing problem: the sender should not transmit more than $F/2$ RT TPDU without informing the destination of the current value of the $N(RTS)$ parameter. This could be performed by sending ACK TPDUs carrying the incremented value of $N(RTS)$. However, as ACK TPDUs may be lost, the sender should transmit an ACK TPDU after $F/2N$ RT TPDU. In this case the sequencing error rate is reduced below the value that causes the release of the transport connection. Let W be the window time (the maximum time the transport entity will wait before retransmitting up-to-date window information). This parameter should verify:

$$W < F\tau/2N \qquad (2)$$

If the two conditions above (on F and W) are verified RT TPDUs can be transmitted with a partial control performed by DT TPDUs. The only RT TPDU loss caused by this synchronization scheme is due to the reception of RT TPDU out of sequence with respect DT TPDUs.

5 Synchronization requirements revision

The relative sequence approach is based on the rule "RT is faster than DT" in order to allow for the correct receipt of RT TPDUs even when DT TPDUs are retransmitted. However, this is not the only scheme to ensure mutual synchronization. One other possible alternative is to allow the reception of RT TPDUs only when they fit in place i.e. *if a RT TPDU is transmitted before a DT TPDU it should be either received before this DT TPDU or discarded* **and** *if a TPDU is transmitted after a DT TPDU it should be either received after this DT TPDU or discarded.*

This scheme can be implemented by carrying the synchronization information in RT TPDUs: each RT TPDU will contain a parameter representing the sequence number of the next DT TPDU the transport entity will send, namely a $N(DTS)$ parameter. This scheme ensure mutual synchronization according to the following scenario:

At the sender side

The transport entity transmit the video frame in a sequence of RT TPDUs followed by a DT TPDU. After the emission of DT TPDUs, the application may continue to send RT TPDUs belonging to another frame, carrying an incremented $N(DTS)$. This scheme can be described as being an "absolute sequence" of DT TPDUs with RT TPDUs inserted in between. Overflowing RT TPDUs in both sides will be discarded at the reception. In addition, if a DT TPDU is lost all subsequent RT TPDUs will be discarded until successful reception of the DT TPDU.

At the receiver side

When a RT TPDU is received the $N(DTS)$ parameter is compared to the state variable representing the next DT TPDU to be received $N(DTR)$,

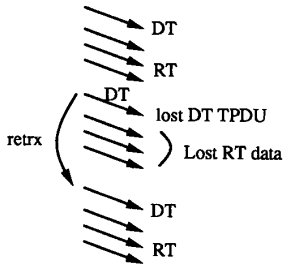

Figure 4: **The absolute sequence scheme**

- if $N(DTS) < N(DTR)$, the received RT TPDU belong to an "old frame", it was transmitted before the last received DT TPDU, it shall be discarded.
- if $N(DTS) = N(DTR)$ the RT TPDU is received in "absolute sequence" and is copied in the resequencing buffer. If the N(RTR) state variable does not correspond to the next RT TPDU number a RT TPDU loss is detected.
- if $N(DTS) > N(DTR)$ a DT TPDU was lost; two alternative are possible: copy the RT TPDU in the resequencing buffer or discard it. If the DT TPDU can be retransmitted rapidly, the buffered RT TPDUs can be delivered to the application. Otherwise, if the delay exceeds the RT time constraint the RT TPDU will be discarded.

The absolute sequence approach described above allow for a "more strict" control of the RT flow by accepting RT TPDUs only when arriving "in sequence" (i.e. in between the delimiting DT TPDUs). RT TPDUs should "synchronize themselves" to the DT flow. However, the price to pay is a higher RT TPDU loss: in the absence of DT TPDU loss both late and early received RT TPDUs are discarded. On the other hand, if a DT TPDU is lost all subsequent RT TPDUs are also discarded.

When the mutual synchrony of DT and RT information is not mandatory e.g. in the case of simultaneous transmission of text and video images on the same transport connection, the relative sequence scheme is superior: it allows to synchronize the two flows upon an application level request. It also allows partial control of the RT flow using the $N(RTS)$ parameter of independent DT TPDUs. However, the required checkpoint procedures jeopardize the protocol correctness if special numbering requirements are not met.

If the normal and real time data flows should be in complete synchrony the absolute sequence scheme may be useful. Each DT TPDU is automatically a synchronization request. A modeling study is underway to derive a quantitative estimation of the loss percentage in both cases [13].

6 TP5

TP5 is a multimedia transport protocol inspired from TP4. We designed this protocol by modifying the TP4 protocol in order to enhance the provided transport service. A new TPDU type the RT TPDU is introduced to carry the real time traffic. No explicit flow or error control will be performed to RT TPDUs. A lost RT TPDU will degrade the quality of the transmission, whether to stop or not is application dependent. The transport entity will only count lost RT TPDUs and will initiate a connection release procedure if this value exceeds a negotiated value at the connection establishment.

The two described synchronization schemes may be used. Both schemes require a modification of the DT and ACK

TPDUs to carry the $N(RTS)$ parameter. Another parameter may also be introduced in ACK TPDUs the $N(TRR)$ parameter denoting the next real time TPDU the transport entity generating the TPDU expect to receive. This parameter can be used to perform "on demand" Acking of RT TPDUs (using the bit "demand ACK" in RT TPDUs) in order to minimize the out of synchronization state in the relative sequence scheme.

TP5 is compatible with ISO TP4 i.e. the CR initiated by a TP5 entity shall propose the class 4 as a reply class. If the destination does not implement TP5 a TP4 connection will be established and all the RT data will be transmitted as normal DT TPDUs.

7 Conclusion

In this paper we presented two transport level synchronization schemes of multimedia flows. With the use of the described synchronization schemes, the TP5 protocol minimizes the control of the real time traffic while still controlling the normal data flow. As RT TPDUs rate is expected to be much higher than the DT TPDU rate, the proposed schemes lead to a minimal degradation of the quality of the real time traffic due to the control performed.

The synchronization function may indeed be done by session level procedures. However, the systematic need for synchronizing for multimedia applications requires a transport level synchronization: even if synchronizing the two flows is conceptually simpler if there were a separate connection for each flow, the use of a multimedia connection with one of the synchronization schemes described above allows for a more efficient synchronization.

References

[1] Jon Postel. *Transmission Control Protocol Specification.* DDN NIC, SRI International Menlo Park, CA, September 1981. RFC-793.

[2] Recommendation X.224, *Transport Protocol Definition For Open Systems Interconnection (OSI).* Red Book, Volume VIII, Fascicle VIII.5.

[3] Journées Bilan du Projet-Pilote NADIR (par l'INRIA et le CNET). Novembre 1985, Palais des Congrès, Versailles.

[4] Van Jacobson. *Congestion avoidance and control.* Computer Communication Review, Volume 18, Number 4, 1988.

[5] C. Huitema *A high speed network connection between NSF and INRIA.* Research note, INRIA Sophia Antipolis, October 1987.

[6] W. D. Brodd and R.A. Donnan. IBM corporation. *Data Link Control Improvements For Satellite Transmission.* International Symposium on Satellite and Computer Communications. Versailles, France, April 1983. pp. 201-213.

[7] David D. Clark, Van Jacobson, John Romkey, Howard Salwen. *An analysis of TCP processing overhead.* IEEE Communications Magazine, June 1989, pp. 23-29.

[8] Richard W. Watson and Sandy A. Mamrak. *Gaining efficiency in transport services by appropriate design and implementation choices.* ACM transactions on computer systems, Vol 5, No. 2, May 1987, pp 97-120.

[9] Greg Chesson. *Protocol Engine Design.* Usenix Conference Proceedings, June 1987. pp. 209-215.

[10] David Clark, Mark Lambert, and Lixia Zhang. *NETBLT: A High Throughput Transport Protocol.* Proceedings of the ACM SIGCOMM '87 Workshop. Computer Communication Review, Volume 17, Number 5, Special Issue. pp. 353-359.

[11] Walid S. Dabbous. *On high speed transport protocols.* Proceedings of the IFIP Workshop on Protocols for high speed networks, Zurich, Switzerland, 9-11 May, 1989.

[12] Christian Huitema, Walid Dabbous. *End to end transmission control on ATM networks.* Nato advanced research workshop on Architecture and performance issues of high-capacity local and metropolitan area networks, INRIA Sophia Antipolis, June 1990.

[13] Walid Dabbous. *Etude des protocoles de contrôle de transmission dans les réseaux à haut débit.* Phd Thesis, In preparation.