AERONAUTICAL TELECOMMUNICATIONS NETWORK PANEL

WG2/18

Naples, Italy

May 1999

TP4 timeout and retransmission issues

Prepared by Stéphane Tamalet

(France)

SUMMARY

At its 17th meeting, the Working Group 2 agreed that it would be appropriate to enhance the ATN transport service to better cope with the loss of TPDUs. A first technique based on the use of the TP4 selective acknowledgement option was proposed in WG2/WP486. In the discussion of this document, it was observed that others mechanisms, such as the use of adaptive retransmission timers needed to be considered. This resulted in Action 17/10 for the group to submit proposals and the results of analysis from investigation into candidate enhancements to the transport service.

In response to this action, STNA performed an analysis of the mechanisms currently implemented in existing TP4 implementations (namely the Vertel, and Atos (Marben) implementations). It was observed that both implementations implement algorithms derived from studies of the IETF working groups on the enhancements to the TCP protocol.

Our analysis concluded that there are 3 different TCP mechanisms (Jacobson algorithm, Karn's algorithm, and Fast Retransmit algorithm) related to timeout and retransmission issues, which are directly applicable to the TP4 protocol and worth to be considered as candidate enhancements to the ATN transport service.

This paper provides a description of these mechanisms.

TABLE OF CONTENTS

1 Introduction

At its 17th meeting, the Working Group 2 agreed that it would be appropriate to enhance the ATN transport service to better cope with the loss of TPDUs. A first technique based on the use of the TP4 selective acknowledgement option was proposed in WG2/WP486. In the discussion of this document, it was observed that others mechanisms, such as the use of adaptive retransmission timers needed to be considered. This resulted in Action 17/10 for the group to submit proposals and the results of analysis from investigation into candidate enhancements to the transport service.

In response to this action, STNA performed an analysis of the mechanisms currently implemented in existing TP4 implementations (namely the Vertel, and Atos (Marben) implementations). It was observed that both implementations implement algorithms derived from studies of the IETF working groups on the enhancements to the TCP protocol.

Our analysis concluded that there are 3 different TCP mechanisms (Jacobson algorithm, Karn's algorithm, and Fast Retransmit algorithm) related to timeout and retransmission issues, which are directly applicable to the TP4 protocol and worth to be considered as candidate enhancements to the ATN transport service.

This paper provides a description of these mechanisms.

2 References

1	V. Jacobson, "Congestion Avoidance and Control", Computer Communication Review, vol. 18, no. 4, pp 314-329, Aug. 1988.
2	P. Karn, C. Partridge, "Improving Round Trip Time Estimates in Reliable Transport Protocols"
3	B. Braden, "Requirements for Internet Hosts – Communication Layers", RFC 1122
4	W. R. Stevens, "TCP/IP Illustrated, Volume 1: The protocols", Addison-Wesley, 1994

3 TCP timeout and retransmission

3.1 General

In a similar way to TP4, TCP provides a reliable transport layer by retransmitting data, when the data has not been acknowledged after some period of time. A critical element of any TCP (and TP4) implementation is the timeout and retransmission strategy: the main issue is the determination of an appropriate timeout interval:

The timeout interval has important and conflicting effects on individual user throughput and overall network efficiency. To achieve optimal throughput, short timeout interval should be used. Unfortunately, what is good for throughput is disastrous for efficient network utilization. If the timeout interval is too short, then a large number of packets may be retransmitted unnecessarily because the sender times out too soon.

The TCP timeout and retransmission strategy relies on the dynamic measurement of the round-trip time (RTT), i.e. the time interval between sending a packet and receiving an acknowledgement for it. In an internetwork, the RTT is expected to change over time, as routes and network traffic load might change. For this reason TCP has been designed to re-estimate the RTT and compute a new time out interval everytime a new packet is acknowledged.

In the original TCP specification (RFC793), TCP had to update a smoothed RTT estimator (called R) using the formula:

$$R_{new} = \alpha R_{prev} + (1-\alpha)M$$

Where α is a smoothing factor with a recommended value of 0.9, and M is the last round trip time that has been measured.

This smoothed RTT is updated every time a new measurment is made. Ninety percent of each new estimate is from the previous estimate and ten percent is from the new measurement.

Given this smoothed estimator, which changes as the RTT changes, RFC793 recommended the retransmission timeout value (RTO) be set to:

$$RTO_{new} = \beta R_{new}$$

Where β is a delay variance factor with a recommended value of 2.

There were known problems with the RTO calculations specified in RFC 793. First, the accurate measurement of RTTs is difficult when there are retransmissions. Second, the algorithm to compute the smoothed roundtrip time is inadequate, because it incorrectly assumed that the variance in RTT values would be small and constant. These problems where solved by Karn's and Jacobson's algorithm, respectively. These two algorithms are described in the next two sections.

The performance increase resulting from the use of these improvements varies from noticeable to dramatic. Jacobson's algorithm is especially important on a low speed link, where the natural variation of packet sizes causes a large variation in RTT.

3.2 Jacobson's algorithm

Van Jacobson has developed a refinement of the TCP algorithm that dynamically computes the variance instead of using a fixed β. The proposal has been to calculate the RTO based on both the mean and variance of the RTT rather than just calculating the RTO as a constant multiple of the mean RTT). This led to propose the following new equations that are applied to each RTT measurement M.

$$Err = M - R_{prev}$$

$$R_{new} = R_{prev} + gErr$$

$$D_{new} = h(|Err| - D_{prev})$$

$$RTO_{new} = R_{new} + 4D_{new}$$

Where *R* is the smoothed RTT (the estimator of the average RTT) and *D* is the smoothed mean deviation. *Err* is the difference between the measured value just obtained and the current RTT estimator. Both R_{new} and D_{new} are used to calculate the next retransmission timeout (RTO_{new}). The gain *g* is for the average and is set to 1/8 (0.125). The gain for the deviation is *h* and is set to 0.25. The larger gain for the deviation makes the RTO go up faster when the RTT changes.

Jacobson specifies a way to do all these calculations using integer arithmetic, and this is the implementation typically used. That is one reason *g*, *h* and the multiplier 4 are all powers of 2, so the operations can be done using shifts instead of multiplies and divides.

Comparing the original method with Jacobson's, we see that the calculations of the smoothed average are similar (α is one minus the gain *g*) but a different gain is used. Also, Jacobson's calculation of the RTO depends on both the smoothed RTT and the smoothed deviation, whereas the original method used a multiple of the smoothed RTT.

This important improvement has been shown to produce dramatically improved roundtrip time estimates and is now in widespread use

3.3 Karn's algorithm

3.3.1 The back-off mechanism

Whenever a timeout occurs, virtually every TCP implementation increases the RTO by some factor before retransmitting the unacknowledged data. Should the new, larger RTO expire yet again before retransmission is acknowledged, the RTO is increased still further. This technique is known as *back-off*. (Back-off is performed independently of the mean RTT calculation, since without an acknowledgement there is no new timing information to be fed into the calculation). A variety of back-off algorithms are used since the TCP specification does not prescribe one. Most often, the implementations simply double the RTO (i.e. perform binary exponential back-off) for each consecutive attempt. Whatever the algorithm, TCP back-off is essential in keeping the network stable when sudeen overloads cause datagram to be dropped. When the overload condition disappears, datagram loss stops and the TCPs reduce their RTO to their normal "mean RTT"-based values.

3.3.2 Karn's algorithm

When finally, after one or several retransmissions, an acknowledgement of the packet reception is received, the TCP wishes to update its RTT estimator. The TCP is however faced to the following dilemma: the measurement of the last RTT is normally done by computing the interval of time between the packet transmission and the acknowledgement reception; however, when retransmissions occurred, the TCP cannot know whether the received acknowledgement was issued on receipt of the first packet or on receipt of one of the subsequent retransmitted packets. This is called the *retransmission ambiguity problem.*

If the TCP chooses to measure the new RTT, as the interval of time from the first transmission to the receipt of the acknowledgement, it runs the risk to inflate unnecessarily the value of the RTT estimator. Inflated RTT estimates may not be a problem if the original reason for the retransmission was network congestion, because congestion tends to increase RTT anyway. However, if the path is lossy, the RTT estimator grows and throughput unnecessarily decrease to low level.

If the TCP chooses to measure the new RTT, as the interval of time from the most recent retransmission to the receipt of the acknowledgement, it runs the risk to decrease inappropriately the value of the RTT estimator. The result may be dramatic: unnecessary retransmission occur constantly, the RTT estimator stabilises at an unreasonable low estimate, useful throughput drops sharply and network bandwidth is wasted.

Another strategy was to simply ignore round trip times for packets that have been retransmitted. In that case, the RTT estimator and the derived RTO are not updated when retransmission occurs. However this method does not work if the causes of the retransmissions is a sudden increase in network roud trip time (e.g. failure of a primary path that causes datagrams to be sent over a slower secondary path). This stategy may also lead to have numerous unnecessary retransmissions, and to waste the network capacity.

Karn solved this problem by proposing the following simple rules:

1. When an acknowledgement arrives for a packet that has been sent more than once, ignore any round-trip measurement based on this packet (this is the third strategy described above)

2. In addition, don’t calculate a new RTO, but reuse instead the last backed-off RTO for the next transmission. Only when a packet is acknowledged without an intervening retransmission will the RTO be recalculated from the RTT estimator.

3.4 .Fast retransmit

A TCP enhancement has been proposed with the aim to speed-up the retransmission in case of TPDU loss. This enhancement is known under the name “Fast retransmit”.

The first part of this enhancement is to require that TCP generate an immediate acknowledgement (i.e. without delay) when an out-of-order packet is received. The consequence of acknowledging out-of-order packets is that the sender receives duplicate ACKs PDU. For instance if the sender sends packets 7, 8, 9 and 10 and the receiver receives and acknowledges packets 7, 9, and 10 (packet 8 is lost), the sender will receive 3 identical ACK PDUs signalling that the next expected is the packet 8 (ACK(8)).

The purpose of this (these) duplicate ACK(s) is to let the other end know that a segment has been received out of order, and to tell it what sequence number is expected.

The Fast retransmit algorithm uses the redundant “ ACKs to deduce that a packet has been lost before the retransmission timer has expired. It works as follows: since TCP does not know whether a duplicate ACK is caused by a lost segment or just a reordering of segments, it waits for a small number of duplicate ACKs to be received. It is assumed that if there is just a reordering of the segments, there will be only one or two duplicate ACKs before the reordered segment is processed, which will then generate a new ACK. If three or more duplicate ACKs are received in a row, it is a strong indication that a segment has been lost. TCP then performs an immediate (fast) retransmission of what appears to be the missing packet, without waiting for a retransmission timer to expire.

4 Discussion

The 3 firsts above mechanisms (Jacobson’s, back-off, Karn’s) cope with intrinsic problems (RTT variation) of internetworks. The TP4 implementations in the ATN internet will have to face these problems, and must therefore support adaptive retransmission mechanisms. A minimal TP4 implementation that uses fixed retransmission timers will not work properly in the ATN. Indeed, whatever the configured value of its fixed RTO, such a minimal TP4 implementation would either load the network with unnecessary retransmissions or unnecessarily limit the end user traffic throughput each time variations in network transit delays are experienced.

The Jacobson's, back-off, and Karn's algorithms are applicable to a TP4 implementation. . In the absence of further studies on timeout and retransmissions issues in the specific case of the ATN, these 3 mechanisms appears to be natural candidates for an enhancement of the ATN TP4 specification. Indeed, these 3 mechanisms have already been proved efficient in the Internet, and they have already supported by different TP4 implementations (namely the Vertel's and Marben-Atos's implementations).

The “fast retransmit” algorithm is a candidate mechanism to mitigate the problem raised in WG2/WP486 (potential long service interruptions due to a change in subnetwork connectivity). The “fast retransmit” mechanism is very similar to the one proposed in WG2/WP486. It is however simpler, since it does not require the use of the selective acknowledgement option. (The selective acknowledgement option is not supported by many TP4 implementations (e.g. the Vertel's and Marben-Atos's ones) and its addition represents a major upgrade to the code).

In conclusion, the 4 mechanisms presented in this paper are good candidates for an enhancement of the ATN TP4 specification. However it must be noted that studies on timeout and retransmission issues applied to the particular case of ATN air/ground communications could lead to discover more appropriate mechanisms (and notably more appropriate RTO calculation rules).

As an example of a potential ATN-specific TP4 feature allowing to mitigate the problem raised in WG2/WP486, a new mechanism is presented in the next chapter.

5 The “anticipated retransmit” algorithm

One of the particularities of the ATN Air/ground communications is that an “unnecessary retransmission” may not be “dramatic”. This can be true when the deflate compression is in use over the mobile subnetworks. Indeed, the deflate mechanism allows to compress an exact copy of a previously transmitted packet, into a very small number of octets (typically less than ten octets). An unjustified retransmission has therefore some chance to result only in a few numbers of octets sent over the air/ground link.

Taking into account this particularity, an acceleration of the rhythm of retransmissions on air/ground critical ATSC TP4 connections can be considered acceptable. It could then be proposed to retransmit the packets before the Jacobson's RTO expiration. For instance the RTO could be set to the mean RTT value.

However, it may be argued that even if retransmitted packets are greatly compressed, the transmission of the resulting short packets over mobile subnetworks would represent a non-negligible overhead. This is because these short packets would compete with other packets for the access to the mobile subnetwork (assuming a CDMA strategy is used); furthermore the transmission of a packet, even short, may trigger the transmission of several data link and ISO 8208 PDU over the mobile subnetwork. Due to this overhead, a simple acceleration of the rhythm of retransmissions may not be acceptable.

To overcome this problem, the proposed mechanism is to perform an anticipated data packet retransmission only when the local TP4 entity sends an ACK TPDU to the remote TP4 entity. (i.e. the retransmitted data TPDUs ride piggyback on the ACK TPDUs which are used to acknowledge the reverse data flow).

With this mechanism, the retransmission of DT TPDUs can be anticipated without adding more than a few octets overhead on the mobile subnetworks. (Note: a retransmitted DT TPDU concatenated to an ACK TPDU will be much more compressed than a standalone retransmitted DT TPDU (gain of the size of the compressed CLNP PDU header)). Only the ground subnetworks are impacted by the overhead caused by the anticipation of the retransmissions. But overhead on the ground is assumed to be no cause of concern.

For air/ground time critical ATSC traffic (class A, B, C, D?…?), the anticipated retransmit mechanism could be triggered on a TP4 connection, as soon as there is weak likelihood of TPDU loss. For instance, the mechanism could be started at the time interval (*R – D)* where *R* is the current RTT estimator and *D* is the current mean deviation. Once triggered the anticipated mechanism must be waiting for an opportunity to mount the DT TPDU to be retransmitted piggyback on an ACK TPDU. If no ACK TPDU is sent, then the DT TPDU is only retransmitted at expiration of the RTO.

This mechanism should only be selected (at transport service level) for air/ground time critical ATSC traffic. It must not be used for ground/ground communications.

The use of this mechanism would also require to mandate the use of the deflate compression mechanism over mobile subnetworks.

6 Recommendations

The Working Group 2 is invited to consider whether the above mechanisms are valid candidates for an enhancement of the ATN transport service, and whether it is worth developing associated guidance materials or recommendations or SARPs requirements.