Comments on "Minimum-Latency Transport Protocols with Modulo-N Incarnation Numbers"

András L. Oláh and Sonia M. Heemstra de Groot, *Member, IEEE*

***Abstract*—It has been shown that a class of minimum-latency transport protocols are analyzed. The protocols use unique incarnation numbers and caching schemes to reduce the latency of connection setup whenever possible. Three modifications to the protocol are discussed in this short paper: 1) A modification to the opening procedure which eliminates some constraints for the correctness of the protocol. 2) A modification which allows data messages in the opening state of the client to be sent. This reduces the latency in some situations for the price of stricter constraints for correctness. 3) An alternate way of closing connections. Apart from these modifications, we also show that the proofs can be refined to get somewhat less restrictive constraints for the correctness of the protocol.**

I. Introduction

In the June 1995 issue of this Transactions,[1] Shankar and Lee presented and analyzed a class of connection management protocols. The protocols use unique incarnation numbers (from a modulo-N space) and a caching scheme to achieve the minimum latency of connection setup whenever possible. In the lack of cached information, the protocol falls back to the three-way handshake which imposes longer latency. The analysis has practical importance because there are protocols already in use based on similar mechanisms, such as TCP for Transactions (T/TCP) [2] in the Internet. Throughout in this paper, it is assumed that the reader is familiar with the above paper.[1]

The purpose of this short paper is twofold: we propose some modifications to the protocol, and we show how some proofs in the above paper[1] can be refined. Three different modifications are proposed.

1) An improvement to the connection setup procedure which eliminates the timing constraints $w_C > \mathtt{W_S}$ and $r_c > \mathtt{W_S}$ in (**T1**) in the above paper[1]. In the original protocol, when an opening server b receives a $(\mathtt{CR}, a, b, sin)$ with $sin > Din_b(a)$, the server updates $Din_b(a)$ to sin. Our version generates a new local incarnation for the server as well.
2) We propose to allow the sending of DATA (data) and even DR (disconnect request) messages while the client is in the opening state. By doing so, the latency can be decreased in cases when the full request of the client does not fit into a single CR message. The disadvantage of this modification is the replacement of I with 2I in the constraint for the minimum safe value of N. Since the value of I (maximum incarnation duration) is likely to be the largest of the constants in the constraint, this modification increases the minimal safe value of N.
3) Finally, we propose a modification to the handling of some closing anomalies. This modification cannot be clearly identified as an improvement, but it makes the closing procedure of the protocol equivalent to the widely accepted method used in TCP [3] and other transport protocols.

While discussing our modifications, we also compare the behavior of the protocol[1] to TCP [3] and T/TCP [2].

Lemma 1[1] provides bounds on the incarnation numbers carried in the different messages with respect to the corresponding state variables of the client and the server. These bounds are then used to calculate the minimum safe value of N (see **T2**[1]). Tightening these bounds means a smaller safe value of N at a fixed incarnation generation rate α, or a higher safe value of α at a fixed value of N.

The effects of our modifications on the bounds of Lemma 1 are discussed below. Furthermore, we propose some refinements to the analysis in the above paper[1]. More specifically, we show that some of the bounds can be tightened allowing for a smaller safe value of N. Detailed proofs are included in the Appendix.

II. Reuse of Server Incarnations in the opening State

In the specification of the server (Fig. 2),[1] the *Receive*$(\mathtt{CR}, a, b, sin)$ event has the following code fragment to process CR packets in the opening state:

$$\{\text{previous } Din_b(a) \text{ value was from some old CR}\}$$
$$\Box Status_b(a) = \mathtt{opening} \wedge sin > Din_b(a) \longrightarrow$$
$$Din_b(a) := sin$$

When server b receives a $(\mathtt{CR}, a, b, sin)$ message with a sin newer than its $Din_b(a)$ while in the opening state, it updates $Din_b(a)$ to the sin. From the reception of the new CR, the server knows that it will not be able to connect to the old client incarnation because the client has already aborted that incarnation.

Manuscript received January 2, 1996; revised February 19, 1996; approved by IEEE/ACM Transactions on Networking Editor U. Shankar.

The authors are with the Tele-Informatics and Open Systems Group, Department of Electrical Engineering, University of Twente, Enschede, The Netherlands (e-mail: olah@cs.utwente.nl; heemstra@cs.utwente.nl).

Publisher Item Identifier S 1063-6692(96)06085-2.

[1] A. U. Shankar and D. Lee, *IEEE/ACM Trans. Networking*, vol. 3, no. 3, pp. 255–268, June 1995.

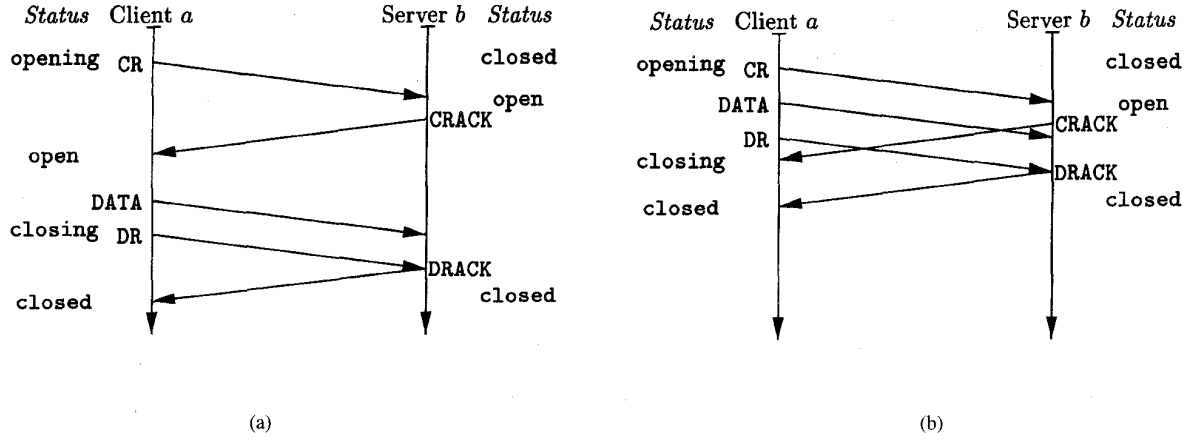


Fig. 1. The advantages of sending data-transfer messages in the opening state: (a) data messages allowed only in the open state and (b) data messages allowed in the opening state already.

The proposed action in this situation is to start a new server incarnation:

{previous $Din_b(a)$ value was from some old CR,
start a new incarnation}
□ $Status_b(a) = \texttt{opening} \wedge sin > Din_b(a) \longrightarrow$
$Din_b(a) := sin;$
$Lin_b(a) := LinGen_b$

A server incarnation is reported to the user only when it reaches the open state, so this modification is not visible to the user. The advantage of this modification is that it allows to loosen the conditions (**T1** and **T2**)[1] for the correctness of the protocol.

The above modification eliminates the need for $w_C > \mathtt{W_S}$ and $r_C > \mathtt{W_S}$ in **T1**. These two constraints protect against the scenario depicted in Fig. 8,[1] where incarnations x and y of client a both become open to incarnation u of server b. The proposed modification prevents exactly this scenario because the server always starts a new incarnation when it receives a message from client a which is from a newer incarnation than the one stored in $Din_b(a)$. Therefore, it cannot happen that the different client incarnations x and y both receive messages from the same server incarnation u. The necessary modifications to the proof of Theorem 1[1] are explained in Appendix A.

This modification also affects the bounds in Case h) of Lemma 1, which states that every *rin* in a DR received at server b when open satisfies

$$rin \in \left[Lin_b(a) - \frac{\mathtt{L} + \mathtt{I} + \mathtt{W_S}}{\alpha}, Lin_b(a)\right].$$

We show in Appendix B6 that this can be tightened to

$$rin \in \left[Lin_b(a) - \frac{\mathtt{L} + \mathtt{I}}{\alpha}, Lin_b(a)\right]$$

because of this modification.

It is interesting to see how this situation is handled in TCP [3], the transport protocol of the Internet. A TCP server simply responds with a CRR to an out-of-order CR message in the opening state.[2] When the client receives this CRR message which carries an old *rin* (the value of $Din_b(a)$ in the server), it responds with a REJ message which terminates the old connection in the server. A subsequent CR from the client will thus find the server in the closed state and the opening three-way handshake can start. The T/TCP proposal [2] does not affect this part of TCP.

Notice that our modification achieves the same effect, starting a new server incarnation when an out-of-order CR arrives, but without the need for the extra CRR and REJ messages. This is possible because we applied the result of Shankar and Lee[1] so that the fact of *sin* being greater than $Din_b(a)$ indicates to the server that the client incarnation $Din_b(a)$ is already discarded by the client. The TCP scheme does not use this extra information, it only checks *sin* and $Din_b(a)$ for equality and thus needs the extra messages to decide whether the out-of-order CR is new or old.

III. Restrictions on DR and DATA Messages

According to the specifications in the paper, DR messages can only be sent in the closing state. The client can only enter the closing state from the open state. Data-transfer messages, denoted by the type DATA, sent by the client are modeled as DR messages (see p. 259 in the above paper[1]). As a consequence, DATA messages are allowed only after the client entered the open state.

The advantage of the two-way handshake over the three-way handshake is the reduced latency for transaction-oriented applications. Sending DATA messages in the open state only, limits these advantages to the cases when the client's request fits into the CR message. In case the request is longer, the subsequent data-transfer messages must wait until the CRACK message arrives from the server as shown in Fig. 1(a). This means a delay of one round-trip time (RTT), exactly the same delay imposed by the three-way handshake. As the bandwidth of networks increases, the number of messages that could be sent within one RTT also increases making the effects of this drawback more significant.

[2] To simplify the discussion, we kept the terminology of the above paper.[1]

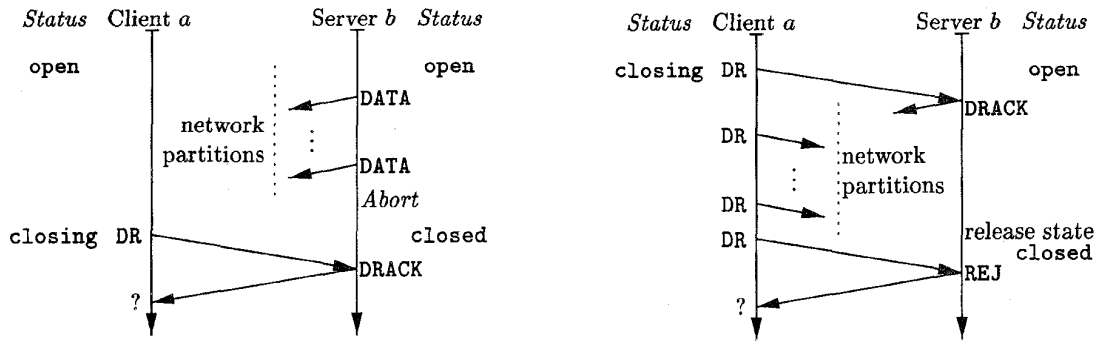


Fig. 2. Ambiguous closing scenarios: (a) client closes after server aborted (original protocol), (b) Server releases state before client gets final DRACK (modified protocol).

A solution to the problem is to allow the client to send DATA or even DR messages while it is still in the opening state. This does not require significant changes to the specification: A flag, *UserClosed* is added to the state variables of the client. This flag is set when the client receives a *DisconnectRequest* from its user in the opening state. If *UserClosed* is set, then i) the client may send DR messages in the opening state; ii) the reception of a valid CRACK or CRR moves the incarnation to the closing state instead of open. Fig. 1(b) shows the behavior of the modified protocol. Note that T/TCP [2] also allows the sending of DATA and DR messages in the opening state.

Case d) of Lemma 1, which provides bounds for sin received in DR messages at server b when b is open, is affected by this modification. The original bound in the above paper[1]

$$sin \in \left[Din_b(a) - \frac{\mathtt{L}+\mathtt{I}}{\alpha}, Din_b(a)\right]$$

is changed to

$$sin \in \left[Din_b(a) - \frac{\mathtt{L}+\mathtt{I}}{\alpha}, Din_b(a) + \frac{\mathtt{L}+\mathtt{W_C}+\mathtt{I}}{\alpha}\right]$$

as it is shown in Appendix B-3.

It is the length of the above interval which influences the safe value of N. The disadvantage of our modification is that the coefficient of I becomes two from the original one. Since I is expected to be significantly larger than the other constants in the constraint determining N, the minimum safe value of N is likely to increase. This is the price of the reduced latency for multi-packet user requests.

IV. Responding with DRACK in the closed State

The specification (see Fig. 2)[1] requires the server to send a DRACK in response to a DR message received in the closed state. This can lead to the scenario shown in Fig. 2(a) below. Incarnation x of client a is open to incarnation y of b, incarnation y is also open to x and the connection is idle, i.e., neither side expects an answer from its peer. At this point b decides to send data to a, but the network partitions for a period longer than $\mathtt{W_S}$, thus, b times out and aborts incarnation y. After the network is reconnected, the user of a decides to close the connection and a sends a $(\mathtt{DR}, a, b, x, y)$ message. Incarnation y of b does not exist any more, but b replies with a $(\mathtt{DRACK}, b, a, y, x)$ and the client closes normally despite the aborting of the connection at the server side.

We propose to change the protocol so that the server responds with a REJ message in the closed state. In order to allow for retransmissions of the DRACK message, the server should not discard its state immediately upon the receipt of a DR. By delaying the release of its state, the server can respond to retransmitted DR messages. This method of closing cannot avoid a possible ambiguity of the closing procedure either [see Fig. 2(b)]. Assume the server receives a DR message and then the network partitions for a period long enough that the server discards its state. The server receives no messages and, thus, has no means to distinguish this situation from the one when the client has normally received the DRACK and closed. The client either gives up retransmitting its DR while the network is partitioned or finally receives a REJ in response to a DR which reaches the server in the closed state. In neither of these cases can the client decide whether the server has closed orderly.

It is well known that fully reliable close cannot be achieved over unreliable communication channels [1] (see also the two-army problem in [4]). The difference in the original closing scheme and the proposed one is the handling of the ambiguous cases. The original scheme is optimistic because it responds normally to DR messages in the lack of the necessary status information. On the other hand, our proposed scheme is pessimistic because it responds with a REJ in such a case. TCP, for example, uses the pessimistic scheme.

This modification affects Case f) of Lemma 1, which states that every *sin* in a DRACK received at client a when closing satisfies

$$sin \in \left[Din_a(b) - \frac{2\mathtt{L}+\mathtt{I}+\mathtt{W_S}}{\alpha}, Din_a(b) + \frac{2\mathtt{L}+\mathtt{W_S}}{\alpha}\right].$$

We show in Appendix B-5 that this can be tightened to

$$sin \in \left[Din_a(b) - \frac{\mathtt{L}+\mathtt{I}}{\alpha}, Din_a(b)\right]$$

due to the above modification.

V. Refinements of the Analysis

Some of the bounds in Lemma 1[1] can be made tighter which results in less stringent constraints on N. We list these bounds with a short comment in this section. The details of the proofs can be found in the Appendix.

Case a) of Lemma 1 states that every *sin* in a CR received at server b when open or closed with $Cache_b(a) \neq \{\texttt{nil}, \texttt{old}\}$ satisfies

$$sin \in \left[Cache_b(a) - \frac{\mathtt{L} + \mathtt{W_C}}{\alpha}, Cache_b(a) + \frac{\mathtt{L} + \mathtt{W_C} + \mathtt{C_S} + \mathtt{W_S}}{\alpha}\right].$$

We show in Appendix B-1 that this can be tightened to

$$sin \in \left[Cache_b(a) - \frac{\mathtt{L} + \mathtt{W_C}}{\alpha}, Cache_b(a) + \frac{\max(\mathtt{L}, \mathtt{W_S}) + \mathtt{W_C} + \mathtt{C_S}}{\alpha}\right].$$

Case c) of Lemma 1 states that every *sin* in a CRRACK received at server b when opening satisfies

$$sin \in \left[Din_b(a) - \frac{3\mathtt{L} + \mathtt{W_C} + \mathtt{W_S}}{\alpha}, Din_b(a) + \frac{\mathtt{L} + \mathtt{W_C} + \mathtt{W_S}}{\alpha}\right].$$

This can be tightened to

$$sin \in \left[Din_b(a) - \frac{2\mathtt{L} + \mathtt{W_C} + \mathtt{W_S}}{\alpha}, Din_b(a) + \frac{\mathtt{L} + \mathtt{W_C}}{\alpha}\right]$$

as shown in Appendix B-2. In the above cases, we exploited the fact that CRRACK is a secondary message which is only sent upon the reception of a valid CRR when the client is in the opening or open state.

Case e) of Lemma 1 states that every *sin* in a CRR received at client a when open satisfies

$$sin \in \left[Din_a(b) - \frac{\mathtt{L} + \mathtt{W_S}}{\alpha}, Din_a(b) + \frac{2\mathtt{L} + \mathtt{W_C} + \mathtt{W_S}}{\alpha}\right].$$

We show in Appendix B-4 that this can be tightened to

$$sin \in \left[Din_a(b) - \frac{\mathtt{L} + \mathtt{W_S}}{\alpha}, Din_a(b) + \frac{\mathtt{L} + \mathtt{W_C}}{\alpha}\right].$$

During the analysis of this case, the authors in the above paper[1] used the current value of $LinGen_b$ as an upper bound on the *sin* in a just received CRR message. Note however that a tighter upper bound is the value of $LinGen_b$ at the time when the server last received a CR message.

VI. Modified T2

The net effect of our modifications to the protocol and our refinements to Lemma 1 is that **T2**[1] changes from

$$N \times \alpha \geq 2\mathtt{L} + \mathtt{W_S} + \max(2\mathtt{W_C} + \mathtt{C_S}, 2\mathtt{L} + 2\mathtt{W_C} + \mathtt{W_S}, 2\mathtt{L} + \mathtt{W_S} + \mathtt{I})$$

to the constraint below

$$N \times \alpha \geq \mathtt{L} + \mathtt{W_C} + \max(\mathtt{L} + \mathtt{W_C} + \mathtt{C_S}, \mathtt{W_C} + \mathtt{W_S} + \mathtt{C_S}, 2\mathtt{L} + \mathtt{W_C} + \mathtt{W_S}, \mathtt{L} + 2\mathtt{I}).$$

The first three arguments of the $\max(\cdots)$ function are lower than the corresponding arguments in the original constraint **T2**, but the last argument has increased due to our modification in Section III.

VII. Conclusion

A class of connection management protocols are analyzed in the above paper,[1] which use a caching scheme to achieve minimum-latency connection setup whenever possible. We have proposed three modifications to the specification and some refinements to the analysis.

1) An improvement to the opening procedure which eliminates some constraints for the correctness of the protocol. In particular, it allows a client to start a new connection attempt immediately after a crash or after the lack of response from the server by removing the requirements $r_C > \mathtt{W_S}$ and $w_C > \mathtt{W_S}$.
2) Reduction of the connection-setup latency when the request of the client does not fit into a single message. The price of this reduced latency is an approximately twofold increase in the minimal safe value of N. We believe, however, that the extra bit in N required by this increase is a good compromise for the achieved latency improvement.
3) Alternatives to achieving graceful close has been discussed and we have compared the method proposed in the above paper[1] to the one used in TCP [3].
4) We have identified a few points in the original analysis where the constraints for the minimal safe value of N can be made less strict.

To emphasize the practical consequences of these protocol design decisions, we have compared the protocol in the above paper[1] to TCP [3] and T/TCP [2]. T/TCP is a variant of TCP which uses many of the techniques discussed in the above paper.[1] We believe our paper helps to better understand the design of reliable transport protocols.

Appendix A
Modifications to the Proof of Theorem 1

Theorem 1[1] states that the protocol satisfies the correctness properties if the following inequalities hold

$$c_S > \mathtt{W_C},\ r_S > \mathtt{W_C},\ w_C > \mathtt{W_S},\ r_C > \mathtt{W_S}.$$

Our modification in Section II makes the requirements $w_C > \mathtt{W_S}$ and $r_C > \mathtt{W_S}$ unnecessary.

Only Part A of the proof is affected by our modification which proves the consistency of connections. The consistency of connections is defined[1] as follows: If incarnation x becomes open to incarnation y, then there is no incarnation z $(\neq y)$ that was previously open to x.

The only use of the requirements in question is in case 2.2 of the proof, where it is proven that two client incarnations, say u and y, cannot both become open to the same server incarnation x. Reception of a $(\texttt{CRR/CRACK}, b, a, x, u)$ message is a precondition for a client incarnation u to become open to the server incarnation x. With our modification, however,

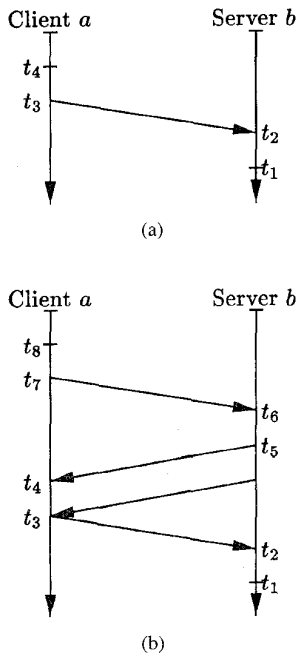


Fig. 3. Upper bound on *sin* in a CR received at server b when open or closed: (a) server opened to CR, (b) server opened to CRRACK.

the following invariant can be proven. If the two messages $(\texttt{CRR/CRACK}, b, a, x, z_1)$ and $(\texttt{CRR/CRACK}, b, a, x, z_2)$ are in the channel from b to a (or were in the channel before, not necessarily simultaneously), then $z_1 = z_2$ must hold. This invariant excludes the possibility of two client incarnation becoming open to the same server incarnation. The truth of the invariant follows directly from the modified specification of the *Receive*$(\texttt{CR}, a, b, sin)$ event.

APPENDIX B
PROOF OF THE MODIFIED LEMMA 1:

1) Bounds on sin in a CR *Received at Server b when* open *or* closed*:* Every *sin* in a CR received at server b when open or closed with $Cache_b(a) \neq \{\texttt{nil}, \texttt{old}\}$ satisfies

$$sin \in \left[Cache_b(a) - \frac{\texttt{L} + \texttt{W}_\texttt{C}}{\alpha}, \; Cache_b(a) + \frac{\max(\texttt{L}, \texttt{W}_\texttt{S}) + \texttt{W}_\texttt{C} + \texttt{C}_\texttt{S}}{\alpha} \right].$$

To determine the upper bound on *sin* in a CR message, let us consider the case study outlined in Fig. 3. Assume the server receives $(\texttt{CR}, a, b, u)$ at time t_1, when $Cache_b(a) = x$. Let x be assigned to $Cache_b(a)$ at some time t_2 upon the reception of a CR or CRRACK message. Because $Cache_b(a) \neq \texttt{old}$ at t_1, we have $t_2 \geq t_1 - \texttt{C}_\texttt{S}$.

Suppose that the message received at t_2 is a $(\texttt{CR}, a, b, x)$ [see Fig. 3(a)]. The message was sent at some time $t_3 (\geq t_2 - \texttt{L})$ by incarnation x of client a. Incarnation x started at some time $t_4 (\geq t_3 - \texttt{W}_\texttt{C})$. Combining the above inequalities, we get $t_4 \geq t_1 - (\texttt{L} + \texttt{W}_\texttt{C} + \texttt{C}_\texttt{S})$. Let l_i denote the value of $LinGen_a$ at time t_i. Since $x = l_4$ and we know that $u \leq l_1$, we get $u \leq x + \frac{\texttt{L}+\texttt{W}_\texttt{C}+\texttt{C}_\texttt{S}}{\alpha}$ in this case.

Now suppose that the message received at t_2 is a $(\texttt{CRRACK}, a, b, x, y)$ [Fig. 3(b)]. Because CRRACK is a secondary message, we know that it must have been sent at some time $t_3 (\geq t_2 - \texttt{L})$ in response to a $(\texttt{CRR}, b, a, y, x)$ while the client was in the opening or the open state. Suppose incarnation x of a became open to b at some time $t_4 (\leq t_3)$ upon the reception of a $(\texttt{CRR}, b, a, y, x)$ message.[3] This CRR was sent at some time $t_5 (\geq t_4 - \texttt{L})$ by incarnation y of server b. Incarnation y started at some time t_6 upon the reception of a $(\texttt{CR}, a, b, x)$. We know that $t_6 \geq t_2 - \texttt{W}_S$, otherwise the server could not become open to x at t_2. The CR was sent at some time t_7 while incarnation x of a was in the opening state. Incarnation x started at some time $t_8 (\geq t_4 - \texttt{W}_\texttt{C})$ because the client became open at t_4. Combining the above inequalities with the obvious $t_4 \geq t_6$, yields $t_8 \geq t_1 - (\texttt{W}_\texttt{C} + \texttt{W}_\texttt{S} + \texttt{C}_\texttt{S})$. In this case $x = l_8$, thus, we get $u \leq x + \frac{\texttt{W}_\texttt{C}+\texttt{W}_\texttt{S}+\texttt{C}_\texttt{S}}{\alpha}$.

2) Bounds on sin in a CRRACK *Received at Server b When* opening*:* Every *sin* in a CRRACK received at server b when opening satisfies

$$sin \in \left[Din_b(a) - \frac{2\texttt{L} + \texttt{W}_\texttt{C} + \texttt{W}_\texttt{S}}{\alpha}, Din_b(a) + \frac{\texttt{L} + \texttt{W}_\texttt{C}}{\alpha} \right].$$

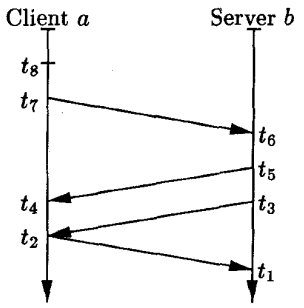
Let us consider the lower bound first [Fig. 4(a)]. Suppose at t_1 the server receives a $(\texttt{CRRACK}, a, b, x, y)$ message. It was sent at some time $t_2 (\geq t_1 - \texttt{L})$ in response to a $(\texttt{CRR}, b, a, y, x)$ which was sent at some time $t_3 (\geq t_2 - \texttt{L})$. Assume that the client incarnation x became open at time t_4 upon reception of a $(\texttt{CRR}, b, a, y, x)$ which was sent at some time t_5 by incarnation y of server b. Incarnation y started at some time $t_6 (\geq t_3 - \texttt{W}_\texttt{S})$ upon the reception of a $(\texttt{CR}, a, b, x)$ sent at some time t_7 by incarnation x of client a which started at some time $t_8 (\geq t_4 - \texttt{W}_\texttt{C})$. Combining these inequalities with the obvious $t_4 \geq t_6$, we obtain $t_8 \geq t_1 - (2\texttt{L} + \texttt{W}_\texttt{C} + \texttt{W}_\texttt{S})$. The upper bound on $Din_b(a)$ at t_1 is l_1 and $x = l_8$, where l_i denotes $LinGen_a$ at time t_i. This proves the lower bound on the *sin*.

As for the upper bound on the *sin* received in CRRACK messages, the following has to be taken into account. CRRACK is a secondary message which is only sent in response to a valid CRR. For the server to receive a CRRACK message with a *sin* that is newer than $Din_b(a)$, the CRR which triggered the CRRACK and the CR from which the current value of $Din_b(a)$ is taken must have been crossed in the network. It is very unlikely in practice, but still possible as described below [Fig. 4(b)].

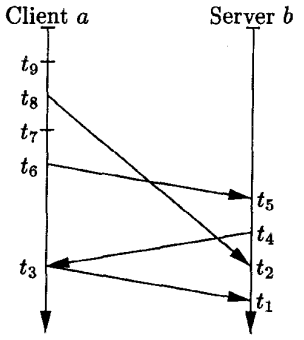
Suppose $Din_b(a) = u$ at some time t_1, when incarnation v of server b receives a $(\texttt{CRRACK}, a, b, x, y)$ message. Let incarnation v become opening at t_2 upon the reception of a $(\texttt{CR}, a, b, u)$ message. Assume the CR was sent at some time $t_6 (\geq t_2 - \texttt{L})$ by incarnation u which was started at some time $t_9 (\geq t_8 - \texttt{W}_\texttt{C})$.

Suppose the CRRACK message received by the server at t_1 was sent at some time t_3 by incarnation x of client a. The client sent this CRRACK in response to a $(\texttt{CRR}, b, a, y, x)$ which was sent by incarnation y of server b at some time t_4. Apparently, $t_4 \leq t_2$ because the server starts incarnation v at t_2 and, thus, would not send the CRR with its $sin = y$. Incarnation y started at some time t_5 upon the reception of a $(\texttt{CR}, a, b, x)$ which

[3] The CRR received at t_3 may be a duplicate or a retransmission.



(a)



(b)

Fig. 4. Bounds on *sin* in a CRRACK received at server b when opening: (a) lower bound, (b) upper bound.

was sent at some time t_6. Incarnation x of client a started at some time t_7. Obviously, $t_7 \leq t_4 \leq t_2$, therefore we get $t_9 \geq t_7 - (\mathtt{L} + \mathtt{W_C})$. Denoting the value of $LinGen_a$ at time t_i, we get $u = l_9$ and $x = t_7$, thus, $x \leq u + \frac{\mathtt{L}+\mathtt{W_C}}{\alpha}$.

3) Bounds on sin in a DR *(or* DATA*) Received at Server b when* open*:* Every *sin* in a DR received at server b when open satisfies

$$sin \in \left[Din_b(a) - \frac{\mathtt{L}+\mathtt{I}}{\alpha}, Din_b(a) + \frac{\mathtt{L}+\mathtt{W_C}+\mathtt{I}}{\alpha}\right].$$

Suppose incarnation v of server b is open and $Din_b(a) = u$ at t_1 when it receives a $(\mathtt{DR}, a, b, x, y)$ message. We know that $x \leq l_1$, where l_i denotes the value of $LinGen_a$ at t_i. Let incarnation v started at some time $t_2(\geq t_1 - \mathtt{I})$ upon the reception of a $(\mathtt{CR}, a, b, u)$ message from the client. The CR message was sent at some time $t_3(\geq t_2 - \mathtt{L})$ by incarnation u of client a which started at some time $t_4(\geq t_3 - \mathtt{W_C})$. Therefore, $t_4 \geq t_1 - (\mathtt{L} + \mathtt{I} + \mathtt{W_C})$ and $u = l_4$ which yields our upper bound $x \leq u + \frac{\mathtt{L}+\mathtt{I}+\mathtt{W_C}}{\alpha}$.

4) Bounds on sin in a CRR *Received at Client a when* open*:* Every *sin* in a CRR received at client a when open satisfies

$$sin \in \left[Din_a(b) - \frac{\mathtt{L}+\mathtt{W_S}}{\alpha}, Din_a(b) + \frac{\mathtt{L}+\mathtt{W_C}}{\alpha}\right].$$

Assume incarnation x of client a receives a $(\mathtt{CRR}, b, a, z, x)$ message at some time t_1 when a is open and $Din_a(b) = y$ (Fig. 5). The CRR was sent at some time t_2 by incarnation z of server b. Incarnation z started at some time t_3 upon the reception of a $(\mathtt{CR}, a, b, x)$ message. This CR message was sent

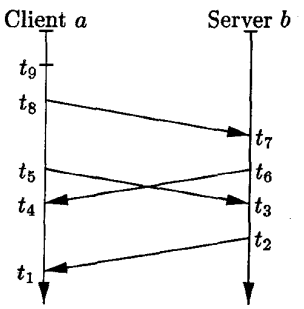


Fig. 5. Upper bound on *sin* in a CRR received at client a when open.

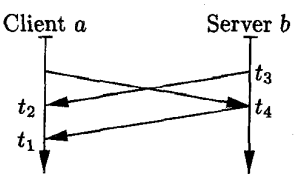


Fig. 6. Upper bound on *sin* in a DRACK received at client a when closing.

at time some $t_5(\geq t_3 - \mathtt{L})$ while a was in the opening state. Suppose a entered the open state at time t_4 when it received a $(\mathtt{CRR}, b, a, y, x)$ message. We obviously have $t_4 \geq t_5$ because at t_5 the client sends a CRR message which must happen before the client changes its state to open at t_4. The CRR received at t_4 was sent at some time t_6 by incarnation y of server b. Apparently, the server must have aborted y sometimes between t_6 and t_3 because it started a new incarnation z at t_3. Incarnation y of server b started at some time t_7 upon the reception of a $(\mathtt{CR}, a, b, x)$ message. This CR message was sent at some time t_8 by incarnation x of client a. Incarnation x started at time $t_9(\geq t_4 - \mathtt{W_C})$, otherwise the client could not become open at t_4.

Combining these inequalities with the obvious $t_7 \geq t_8 \geq t_9$, we get $t_7 \geq t_3 - (\mathtt{L}+\mathtt{W_C})$. We also know that $y = l_7$ and $z = l_3$, where l_i denotes $LinGen_b$ at time t_i. This yields $z \leq y + \frac{\mathtt{L}+\mathtt{W_C}}{\alpha}$ which is our upper bound. For a to receive a CRR with a *sin* newer than z requires the reception of a CR by b later than t_3. But this is impossible, because only incarnation x of a can send a CR message after t_9 and the latest possible time for x to send a CR message is at t_5. Therefore, our upper bound is indeed tight.

5) Bounds on sin in a DRACK *Received at Client a when* closing*:* Every *sin* in a DRACK received at client a when closing satisfies

$$sin \in \left[Din_a(b) - \frac{\mathtt{L}+\mathtt{I}}{\alpha}, Din_a(b)\right].$$

Let the client receive a $(\mathtt{DRACK}, b, a, x, y)$ message at time t_1 when the client is closing. Let $Din_a(b) = u$ and $LinGen_b = l_1$ at t_1. The DRACK message was sent at some time $t_2(\geq t_1 - \mathtt{L})$ by incarnation x of server b which started at some time $t_3(\geq t_2 - \mathtt{I})$. Therefore, $x \geq l_1 - \frac{\mathtt{L}+\mathtt{I}}{\alpha}$. Because, $u \leq l_1$, we have $x \geq u - \frac{\mathtt{L}+\mathtt{I}}{\alpha}$.

For the upper bound on *sin*, we examine first how incarnation u of client a can reach the closing state (Fig. 6). Suppose incarnation u is closing at time t_1. The closing state can

only be entered from the open state. Assume u became open to incarnation v of b at some time t_2 upon the reception of a $(\texttt{CRACK/CRR}, b, a, v, u)$ message which was sent at some time $t_3(\geq t_2 - \texttt{L})$.

Now suppose that the client receives a $(\texttt{DRACK}, b, a, y, x)$ message at t_1. Notice that $Din_a(b) = v$ at t_1. For y to be greater than v, the DRACK message must have been sent at some time $t_4(\geq t_3)$ upon the reception of a $(\texttt{DR}, a, b, x, y)$ message while the server was open. If the DR message is sent by the current client incarnation u, i.e., $x = u$, then $y = v$ would also hold because the server cannot become open twice to the same client incarnation. If $x < u$, then the server could not become open to x after being open to u. If u became open to v upon the reception of a CRACK message, then the server cached u at some time before t_3 and it would not connect to x after t_3. If u became open upon the reception of a CRR message, then $Cache_b(a)$ was nil at t_3. After t_3, it may become u or remain nil depending on the outcome of the three-way handshake. Both cases prevent the server from becoming open to x after t_3.

6) Bounds on rin in a DR *Received at Server b when* open*:* Every *rin* in a DR received at server b when open satisfies

$$rin \in \left[Lin_b(a) - \frac{\texttt{L} + \texttt{I}}{\alpha}, Lin_b(a)\right].$$

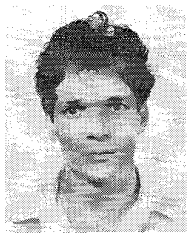
For this case, our argument is similar to the one in the above paper[1] (p. 267). The only difference is that if incarnation x of client a, which is started at some time t_1, becomes open to incarnation y of server b, which is started at some time t_2, then we know that $t_2 \geq t_1$. This is so because the server starts a new incarnation even if it receives the CR of the client while in the opening state to an earlier client incarnation. For the original specification, one could only say that $t_2 \geq t_1 - \texttt{W}_\texttt{S}$ in this case.

ACKNOWLEDGMENT

The authors are grateful for the insightful comments of the referee.

REFERENCES

[1] D. Belsnes, "Single-message communication," *IEEE Trans. Commun.*, vol. COM-24, no. 2, pp. 190–194, Feb. 1976.
[2] R. Braden, "T/TCP: TCP extensions for transactions, functional specification," *RFC 1644*, ISI, July 1994.
[3] J. Postel, Ed., "Transmission control protocol," *RFC-793*, Sept. 1981.
[4] A. S. Tanenbaum, *Computer Networks*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1989.



András L. Oláh received the M.Sc. degree in electrical engineering from the Technical University of Budapest, Hungary, in 1991. Since 1992, he has been a Ph.D. student in the group of Tele-Informatics and Open Systems at the University of Twente, The Netherlands.

His research interests are in the design, analysis, and implementation of reliable transport protocols. He participates in the FreeBSD project. His main contribution is the porting of T/TCP to FreeBSD.

Sonia M. Heemstra de Groot (S'85–M'86) received the degree in electrical engineering from Mar del Plata National University, Argentina, in 1983. She received a second Masters degree in electrical engineering from the NUFFIC in 1986 and the Ph.D. degree in electrical engineering from the University of Twente, The Netherlands, in 1990.

From 1986 to 1990 she worked as an Assistant Researcher in the group for Network Theory and VLSI Design of the Department of Electrical Engineering of the University of Twente, concentrating her attention on topics related to real-time processing of DSP algorithms and efficient multiprocessor scheduling. In 1991 she became a lecturer at the group Tele-Informatics and Open Systems of the same department. Her research interests include different aspects of protocol design and implementation, in particular, time-critical protocol functions and resource allocation and control.