Failsafe End-to-End Protocols in Computer Networks with Changing Topology

ISRAEL CIDON AND RAPHAEL ROM

Abstract—**End-to-end protocols in computer networks in which the topology changes with time are investigated. A protocol that delivers all packets ordered, without duplication, and which uses a window is presented. Using a precise model of the network correctness of the protocol is proven. The use of the window for flow control is also addressed.**

I. Introduction

END-TO-END communication is clearly the object of any communication network. Naturally, therefore, end-to-end protocols have been the subject of research for a long time. While many protocols achieve their goals by using timeouts [1], [2]. Finn, in an original paper [3], demonstrated the existence of end-to-end protocols without making use of timeouts.

Essentially, a failsafe end-to-end protocol is one that should deliver *all* packets in the correct *order* and without *duplication*. In addition, flow control and error recovery must also be addressed. In his paper, Finn presented a protocol which properly handled ordering and duplication avoidance in a network in which a resynch procedure operates.

In this paper, we extend that protocol to accommodate windows, i.e., allow more than one packet to be in transit between source and destination nodes, allow to control the flow, and handle erroneous packets. All this is done while using bounded counters.

The paper is structured in the following way. We first introduce a model for a network with changing topology, the concept of resynch procedures, and the assumptions underlying the operation of end-to-end protocols. These are based on previous work [4], [3]. A windowed end-to-end protocol is then presented and its correctness proven. The use of windows for flow control is also included, as is the special case of circuit-based networks.

II. The Model

We consider a network composed of autonomous processors referred to as *nodes* interconnected by bidirectional *links*. The processors exchange packetized information over these links. We distinguish between two types of networks—fixed topology networks and changing topology networks. A detailed definition of the model of such networks is given in [4], we give here a brief summary only.

In a fixed topology network all packets sent from a node to its neighbor are assumed to arrive correctly in the order sent and within an arbitrary, but finite, delay. Nodes can distinguish among packets received on different links and process packets in the order of arrival.

The major difference between fixed and changing topology networks is the states of the links. In a changing topology network links can be either *active* in which case packets sent over it arrive properly, or *inactive* in which case packets do not arrive. Nodes do not necessarily know the state of the link, but rather mark it as *operative* or *inoperative* according to whether the node assumes the link to be active or inactive, respectively. It is also assumed that nodes become aware of a change in a link state within a finite time of the change.

We assume that a communication resynch procedure (CRP) operates in the network. A CRP is a mechanism to clean the network from all packets that entered the network before some topological change took place. The CRP described here is a simplification of the resynch procedure defined in [3]. Every node maintains a cycle number (CN) that identifies the most recent resynch cycle in which the node participated. Nodes increment their CN when they become aware of a topological change in an adjacent link or when special messages originating at neighbors whose CN is higher are received. Messages belonging to previous cycles are not *accepted*. The CRP assures that a packet making it to the destination has seen the same CN in all nodes it traversed.

Our starting point is the end-to-end protocol presented in [3]. This protocol, which we refer to as the ETE protocol, proceeds as follows. The source node i sends sequenced data packets $P(i, j, n)$ to destination node j (with n being the sequence number), and the destination node j responds with acknowledgments ACK(j, i, n). Node i will not send the $n+1$st message until the nth message has been acknowledged in the current resynch cycle. Each recipient immediately acknowledges every data packet received as well as sends an ACK after every resynch cycle to indicate the last data packet received.

Two additional assumptions must be made with regard to this protocol. First, we assume that nodes do not fail with complete loss of memory—the last packet sent from the node must be saved. The second assumption regards routing. We assume that a packet sent from node s towards destination d traverses the route $s = k_0, k_1, \cdots, k_n$ in which either $k_n = d$, or the packet is rejected at k_n, or sent from it over an inactive link. This means that the packet either arrived at its intended destination or is lost due to topological changes. No other constraints are imposed on the selection of nodes in the route.

III. A Windowed ETE Protocol

We describe here a failsafe end-to-end protocol which allows more than one packet to be "in the pipe" between every pair of communicating nodes. We refer to this as the windowed end-to-end protocol (WETE). In the description that follows, we refer to a single pair of source destination nodes.

Let us assume that N packets are allowed to be on their way at any time. To maintain fail safety we require both the source and the destination nodes to be able to store N packets. In the source this buffer is required because following a topological change all these packets may have to be retransmitted. In the

Paper approved by the Editor for Computer Communications Theory of the IEEE Communications Society. Manuscript received August 10, 1984; revised July 10, 1986.

I. Cidon was with the Department of Electrical Engineering, Technion, Israel Institute of Technology. He is now with the IBM T. J. Watson Research Center, Yorktown Heights, NY 10598.

R. Rom is with the Department of Electrical Engineering, Technion, Israel Institute of Technology, Haifa 32000, Israel.

IEEE Log Number 8613144.

destination it is required because packets may arrive unordered, but have to be delivered in the correct order. Since more than a single packet may be in transit, packets must carry a sequence number. To maintain a finite field for these numbers, cyclic numbers are used.

We number the buffer cells from 1 to N so that at the receiver the packet in cell number 1 is the next one to be delivered, whereas at the sender cell number 1 contains the earliest unacknowledged packet. Nodes maintain cyclic counters to relate messages to cells. The cyclical counters are updated upon delivery of packets at the receiver or receipt of acknowledgments at the sender.

In a more precise way, let the source and destination nodes maintain cyclical counters n_s (mod $N+1$) and n_d (mod $N+1$), respectively. The source s sends to destination d data packets of the form $P(s, d, n)$ where n is a cyclical sequence number (CSN). The destination responds with acknowledgment packets of the form ACK(d, s, n).

Data packets generated at the source are put into the buffer in the next available cell. This packet is assigned a CSN equaling $n_s \oplus K$ and sent immediately towards the destination (K is the cell number into which the packet is put, and $\oplus$ refers to addition mod $N+1$). Upon receipt of ACK(d, s, n) the source marks the packet in cell number $n \ominus n_s$ as acknowledged. The packet in cell number 1 is now examined. Should it be marked as acknowledged, it is discarded, n_s is incremented by 1, and all packets from higher numbered cells are advanced one cell. This is repeated until the packet in cell number 1 is unmarked.

At the receiver, an arriving packet $P(s, d, n)$ is deposited in cell number $n \ominus n_d$ and an ACK(d, s, n) is sent. Cell number 1 is now examined. Should it be full, its contents is delivered, n_d is incremented by 1, and all packets in higher numbered cells are advanced one cell. This process is repeated until cell number 1 is empty.

To adapt the protocol to a changing topology environment, we make use of the Cycle Number changes of the CRP operating in the network. Every time a CN is incremented the receiver discards all the packets in his buffer and sends an ACK(d, s, n_d). The sender waits for this acknowledgment and upon its arrival treats it as an ACK for all packets in the $n_d \ominus n_s$ lowest cells (note that if $n_s = n_d$ no packet is actually acknowledged). All unacknowledged packets are then retransmitted.

Note that the protocol can be made slightly more efficient by using a selective repeat retransmission scheme rather than the go-back-N scheme which is used, i.e., at the beginning of a new cycle the receiver does not discard any packet and sends an ACK carrying not just the value of his counter but also an indicator (N bits long) for all packets in his buffer. The sender, of course, retransmits only the unacknowledged packets.

A. Formal Specification

Following are the variables at the source node s with respect to destination d:

- $MEM(n)$ The buffer that contains for each $n = 1, 2, \cdots, N$: $DP(n)$—a place for a data packet and, $AP(n)$—a place for an acknowledgment.
- n_s A cyclical counter (cyclically points to the bottom of the buffer)
- a_d A Boolean variable indicating receipt of an ACK in the current resynch cycle.

Variables at the destination node d with respect to source s:

- $MEM(n)$ This is the buffer that contains for each $n = 1, 2, \cdots, N$ a place for a data packet.
- n_d A cyclical counter (cyclically points to the bottom of the buffer).

Packets sent and received:

- $P(s, d, n)$ A data packet sent from source node s to destination d carrying a (cyclic) sequence number n.
- ACK(d, s, n) An acknowledgment packet sent from destination node d to source s carrying a (cyclic) sequence number n.

In addition, we make use of the notation

$$K = \min \{l \mid 1 \le l \le N;\ DP(l) = \varnothing\}$$

and assume that K is always updated.

The reading and writing from/to the buffer is handled by MEMPOP and MEMPUSH which are defined in the following way:

- MEMPUSH(X, m) $DP(m) \leftarrow P$ or $AP(m) \leftarrow 1$ depending on whether X is a data or acknowledgment packet, respectively.
- MEMPOP For $t := 1$ to $N-1$ $MEM(t) \leftarrow MEM(t+1)$; $MEM(N) \leftarrow \varnothing$; $n_s \leftarrow n_s \oplus 1$

Algorithm for Source Node s:

1) *event:* New packet P, $a_d = 0$, and $K \le N$
   1.a MEMPUSH(P,K)
   1.b send $P(s,d,n_s \oplus K)$
2) *event:* $A(d,s,n)$ and $a_i = 0$
   2.a MEMPUSH(A, $n \ominus n_s$)
   2.b *while* $AP(1) = 1$ *MEMPOP*
3) *event:* $A(d,s,n)$ and $a_d = 1$
   3.a *for* $t := 1$ *to* $n \ominus n_s$ *MEMPUSH*(A,t)
   3.b *while* $AP(1) = 1$ *MEMPOP*
   3.c *for* $t := 1$ *to* K send $P(s,d,n_s \oplus t)$ from $DP(t)$
   3.d $a_d := 0$
4) *event* Cycle Number (CN) changes:
   4.a *for* $t := 1$ *to* K $AP(t) := 0$
   4.b $a_d := 1$

Algorithm for Destination Node d:

1) *event:* $P(s,d,n)$
   1.a *MEMPUSH*(P,$n \ominus n_d$)
   1.b send ACK(d,s,n)
   1.c *while* $MEM(1) \neq \varnothing$ *do MEMPOP*
2) *event* Cycle Number (CN) changes:
   2.a *for* $t := 1$ *to* N $MEM(t) := \varnothing$
   2.b send ACK(d,s,n_d)

B. Correctness of the ETE Protocol

To prove the correctness of the above protocol we make use of properties of the CRP that underlies the operation of the ETE protocol. Two lemmas with regard to the CRP are quoted; their proof follows directly from lemmas proved in [4] and will not be repeated here.

Lemma 1: Given a network and a finite sequence of topological changes terminating at time t^*. A finite time after t^* for every connected pair of nodes k and j, $R_k = R_j$ (where R_i is the CN of node i).

Lemma 2: If a packet sent at time t from node i along the path $i = i_0, i_1, \cdots, i_n$ where i_n is the first node in which it is not accepted, then within a finite time of t i increments its CN.

In the following lemmas we assume a WETE protocol that operates with unbounded sequence numbers rather than with cyclic numbers and an infinite number of buffer cells (meaning that all arithmetic is regular and not done mod $N+1$). Three different numbers are distinguished:

- Sequence numbers (SN)—absolute, nondecreasing numbers for ordering data packets.
- Cyclical sequence numbers (CSN) which were previously defined and which obey CSN = SN mod $N+1$

- Cycle numbers (CN) which are the numbers associated with every CRP cycle.

Lemma 3: For every CN no two identical packets are sent.

Proof: By the definition of the protocol the sender does not send any packet more than once for any given CRP cycle, and since every packet receives a different SN they are all distinguishable.

The same applies to the receiver since acknowledgments are sent only once for each received packet and are distinguishable by their SN. The first acknowledgment sent carries an SN for a packet that will not be resent and therefore will not be reacknowledged. The CRP property assures that packets of the wrong CN are rejected. □

Lemma 4: Let $\bar{n}_d$ be the (noncyclic) counter at the destination. For every packet $P(s, d, \bar{n})$

$$\bar{n}_d + N \geq \bar{n} > \bar{n}_d$$

Proof: Since the receiver never acknowledged packets number $\bar{n}_d + 1$, by the rules of the protocol the sender could not have sent any packet whose SN is greater than $\bar{n}_d + N$.

It remains to be shown that no packet can arrive in the current cycle with an SN less then or equal to $\bar{n}_d$. Assume $\bar{n}_d \geq \bar{n}$, meaning that packet $P(s, d, \bar{n})$ has been previously delivered—either with the current CRP cycle or in a previous one. The first case contradicts Lemma 3. The second case implies that at the beginning of the current cycle $\bar{n}_d \geq \bar{n}$, implying that in the current CRP cycle all data packets carry SN's greater than $\bar{n}$, contradicting our assumption. □

Lemma 5: Let $\bar{n}_s$ be the (noncyclic) counter at the source. For every arriving acknowledgment ACK$(d, s, \bar{n})$

$$\bar{n}_s + N \geq \bar{n} \geq \bar{n}_s$$

Proof: Clearly, $\bar{n} \leq \bar{n}_s + N$ since no more than N packets may be sent without being acknowledged.

Assume $\bar{n} < \bar{n}_s$ and consider the data packet $P(s, d, \bar{n})$. If this packet was sent in the current CRP cycle, it was already acknowledged in contradiction to Lemma 3.

Consider the case in which $P(s, d, \bar{n})$ was not sent in the current CRP cycle. Two subcases must be treated depending on whether this is or is not the first ACK received in the current cycle. The first subcase means that the ACK comes in response to a data packet $P(s, d, \bar{n})$ which contradicts the assumption. The second subcase means that the receiver has previously acknowledged *all* packets up to $\bar{n}_s$ which implies $\bar{n} > \bar{n}_s$. □

Lemma 6: If a packet $P(s, d, \bar{n})$ arrives at the destination at which time its counter is $\bar{n}_d$ then all packets with SN between $\bar{n}_d$ and $\bar{n}$ have been transmitted in the current CRP cycle.

Proof: At the beginning of the cycle an ACK$(s, d, \bar{m})$ with $\bar{m} \leq \bar{n}_d$ has been received at the source. The lemma holds since the source sent all the packets in their sequential order starting at $\bar{m}$. □

Theorem 1: Given a network with a finite sequence of topological changes in which a CRP operates. Let the WETE protocol operate in conjunction with the CRP. Under these conditions all packets are delivered, in the correct order, and without duplicates.

Proof: Lemmas 4 and 5 prove the equivalence between regular and modulo $N + 1$ arithmetic.

We prove the rest by induction on the CRP cycles and on the sequence numbers within every cycle. Lemma 1 assures that for every packet that is not accepted a new CRP cycle is started causing an ACK to be sent which when arriving at the source causes all as yet undelivered packets to be retransmitted. By Lemma 2, when the sequence of changes terminates, these packets will be accepted.

Lemma 6 means that a packet that arrives is eventually delivered. Lemma 4 assures of no duplicates (due to the strict inequality there).

Since packets are delivered to the user only from cell number 1 they are delivered in the correct order. □

C. A WETE with FIFO Routing

The WETE presented and proven in the previous section applies to an environment with general routing mechanisms. In many cases knowledge about the actual routing may result in processing and memory savings. We take as example FIFO routing which is typical of circuit-switched networks as well as virtual circuit-switching networks.

We refer to FIFO routing as a routing mechanism that assures, for every pair of nodes s and d and within a single CRP cycle, that:

1) The routing property (see Section II) is obeyed.

2) Packets are not accepted out of order, i.e., if two packets accepted at times $t_{d_1} < t_{d_2}$ they had been sent at times $t_{s_1} < t_{s_2}$.

3) If a packet sent at time t in a given CRP cycle is not accepted, then no packets sent after t will be accepted at that cycle.

Within a FIFO routing environment the WETE can be implemented in a simplified way. At the receiver, all packets are immediately delivered, the counter n_d is incremented, but the ACK sent does not carry a CSN. A special ACK that does carry a CSN (equaling n_d) is sent only whenever the CN is incremented.

At the source, receipt of an unnumbered ACK causes discarding of the packet in cell number 1 and incrementing n_s. Receipt of a numbered acknowledgment ACK(d, s, n) causes deletion of the $n \ominus n_s$ first packets and retransmission of the rest in sequential order.

The saving is manifested in the lack of buffer at the receiver, shorter packets, simpler computation, and simpler memory structure (the buffer at the source is a regular FIFO buffer).

Furthermore, if the network provides virtual circuit routing then it is possible to replace the global CRP by an appropriate route set-up/take-down procedure. The properties required of such a procedure are: 1) in the absence of topological changes route set-up always succeeds, 2) if a link belonging to a route fails, then within finite time the end nodes are notified and the circuit is taken down and considered canceled, 3) following a circuit cancellation a new circuit is established, and 4) no packets sent in the past over a currently canceled circuit may ever be received by a node after it has been notified of the cancellation.

Given such a routing mechanism, the nodes execute the same ETE algorithm interpreting a circuit setup as a CN change. No messages should be sent between the cancellation of a circuit and the establishment of its replacement. An example of a route management procedure that fulfills the above is given in [5].

D. Dynamic Flow Control

The WETE protocol uses a fixed window of size N. In this section, we suggest several ways in which the receiver can effectively change the window size based on the congestion measured at the receiver.

The problem with changing the window size is maintaining the accountability of all the messages underway. Thus, a simple way is to fix the window size at the beginning of a CRP cycle when no packets are in transit. To allow window size changes within a cycle (which is typically a long period) one can artificially initiate a CRP cycle whenever the window needs to be changed. In these solutions, large inefficiencies and waste are introduced since they involve discarding packets which would have been accepted.

An improvement along these lines is to use a selective resynch mechanism which works exactly like the regular CRP

operating in the network, except that it causes discarding only packets belonging to the specified pair. The selective resynch is initiated only when the window needs to be changed and a topological change does not occur. When a topological change does occur, a regular (global) CRP is performed superseding the selective resynchs. As a result, the selective resynch always operates within the same global CRP cycle and therefore needs no internal cycle numbers.

The main drawback of the selective resynch is the fact that its messages flood the network. To have only the effected endpoints involved the receiver can withhold the ACK's until it is certain that no data packets are in transit (i.e., when a windowful of packets have not been acknowledged) and then send a special window changing packet.

A yet more flexible scheme can be devised using a window of size $N - W$ by withholding the W most recent ACK's. ACK(d, s, n) is sent only when packet $P(s, d, n \oplus W)$ is accepted. This achieves our goal because when $P(s, d, n \oplus W)$ is in transit $P(s, d, n)$ is as yet unacknowledged so there can be at most $N - W$ under way. In this case no special coordination is required between the sender and the receiver—all is managed by the receiver alone. This scheme allows the freeing of W cells from the buffer, window size may be decreased to zero (i.e., $W = N$), and increased to any size up to N at any time. Note, however, that the last W packets must be identified in order to be acknowledged and avoid deadlocks. Note also that the sender must always keep the W unacknowledged packets in case a new cycle starts when they are retransmitted.

References

[1] V. G. Cerf and R. E. Kahn, "A protocol for packet network interconnection," *IEEE Trans. Commun.*, vol. COM-22, no. 5, pp. 637–648, May 1974.

[2] M. C. Easton, "Design choices for selective-repeat retransmission protocols," *IEEE Trans. Commun.*, vol. COM-29, no. 7, pp. 944–953, July 1981.

[3] S. G. Finn, "Resynch procedures and fail-safe network protocol," *IEEE Trans. Commun.*, vol. COM-27, no. 6, pp. 840–845, June 1979.

[4] Part I: Protocol extension," EE Publication 485, Faculty of Electrical Engineering, Technion, Haifa, Israel, Feb. 1984.
Electrical Engineering, Technion, Haifa, Israel, Feb. 1984.

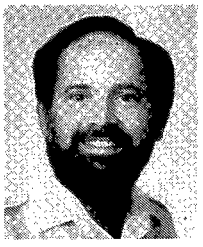
[5] A. Segall and J. M. Jaffe, "Route setup with local identifiers," *IEEE Trans. Commun.*, vol. COM-34, no. 1, pp. 45–53, Jan. 1986.

★

Israel Cidon received the B.Sc. (summa cum laude) and the D.Sc. degrees from the Technion, Israel Institute of Technology, Haifa, Israel, in 1980 and 1984, respectively, both in electrical engineering.

From 1977 to 1980, he was a consulting research and development engineer involved in the design of microprocessor-based equipment. From 1980 to 1984 he was a teaching assistant and an instructor at the Technion. From 1984 to 1985, he was a faculty member with the Faculty of Electrical Engineering at the Technion. Since 1985, he has been with IBM T. J. Watson Research Center. His current research interests are in distributed algorithms and voice/data communication networks.

★



Raphael Rom received the B.Sc. and M.Sc. degrees from the Technion, Haifa Israel, and the Ph.D. degree in electrical engineering and computer science from the University of Utah, Salt Lake City, UT.

In 1975, he joined the research staff of SRI International in California. During 1986, he was on the teaching staff of Stanford University. He is currently with the Faculty of Electrical Engineering in the Technion, Israel Institute of Technology. His areas of interest are algorithms and performance analysis of data communication networks and the design of survivable data communication systems.