M.I.T. Laboratory for Computer Science

Local Network Note No. 25
November 5, 1980

Communication ring initialization without central control

by Jerome H. Saltzer

Introduction

One of the more subtle problems of design in a digital communication ring is how to do ring initialization without assigning a special station to do housekeeping. Initialization is required both at startup and also following a failure that causes the access token to be lost. It is easy enough to insist that every station be prepared to reinitialize the ring (and to detect the need for reinitialization) but this insistence introduces the danger that two or more stations will independently attempt reinitialization. These contending reinitializers can interfere with one another in such a way that none is successful. In a ring of 100 stations, one can imagine (in nightmares) an avalanche of contending attempts to reinitialize, none successful, going on indefinitely.

Earlier solutions to this problem have not been systematic or even very satisfactory. Primenet, for example, offers some suggestions to software implementors concerning token management, but apparently does not provide a foolproof procedure [1]. The L.C.S. version one ring network relies on the software at each node testing for ring format correctness either periodically or before each message origination [2]. The only protection against the reinitialization avalanche effect is that its probability is low with the small number of stations (less than ten) in the net. This approach ignores the coordination problem rather than solving it. The original design of the L.C.S. version two ring network envisioned an automatic scheme based on placing reinitialization responsibility only on stations that discover the need for reinitialization while attempting to originate a message [3]. This scheme does not solve the contention problem, it merely attempts to reduce the typical number of contending stations to a level where a contention-backoff-retry algorithm has a greater chance of working.

To do ring reinitialization systematically yet without central control, we here propose a novel, straightforward approach, and then describe in detail how this approach could be implemented in the version two ring network. The approach has three coordinated elements, jamming, a virtual token, and a try-at-most-once rule.

1) Jamming is a technique borrowed from the Ethernet, where it is used to insure that all contending station s agree that there was a collision [4]. In the Ethernet, whenever an originating station detects a collision by analysis of the signal levels, that station impresses an easily recognizable jamming signal on the net for a time long enough to propagate to every other network station. This jamming guarantees that all contenders agree on the need to backoff and retry. It also guarantees that all agree (to within a couple of propagation times) on when to begin the backoff timeout. It is this last property that is of interest in the ring. Therefore, the first step in systematic ring initialization is that whenever any station detects a need for ring reinitialization (generally by noticing that no format flags have passed by for one ring transit time) that station jams the ring by transmitting a string of zeros for T milliseconds, where T is chosen to be a little longer

than one ring transit time. The string of zeros contains no format flags, so every other station will, within one ring transit time, similarly detect that the ring needs reinitialization and emit T milliseconds worth of zeros. Thus a little more than 2*T milliseconds later all stations will have completed jamming and be in agreement, within about 2*T milliseconds, on when the entire procedure started.

2) Orderly, contention-free initialization can now be accomplished simply by having exactly one station place a correctly formatted message on the ring. The trick is to find a distributed algorithm that chooses exactly one station from a collection of stations that cannot currently communicate and that are not even certain which other stations are participating in the exercise. The ring normally avoids contention for message origination by circulating a token, and requiring that a station not originate a message unless it possesses the token. A similar technique can be used for ring initialization, with the exception that since the ring is not operating yet, the circulation of the initialization token must be simulated. This simulated initialization token is called a <u>virtual token</u>. The virtual token technique is borrowed from the CHAOSNET, which uses it to reduce contention in retries of message origination [5].

In the ring, the virtual token works as follows: each station sets a timer to a value consisting of its station number times 2*T. When this timer finally goes off, it is this station's turn to initialize the ring. Of course, if some other station initializes the ring first, the flag presence detector in each station will terminate that station's interest in ring reinitialization, and operation will return to normal.

Thus the virtual token in effect visits each station in the ring in the order of its station number. The lowest-numbered active station will first decide that it has possession of the virtual token, and will reinitialize the ring. If that station is successful, the rest, waiting for their turn, will notice the success and will return to their usual activities.

One interesting difference between this technique and that of the CHAOSNET is the time scale involved. In the CHAOSNET, multiples of the network propagation time, which is measured in microseconds, are used. In the ring, the transit time is measured in milliseconds. If, for example, a value of T = 0.5 ms. is chosen, and there are 200 stations on a network, one might wonder if it will often require the better part of a second for the initialization to complete. This concern is not really important, however, for two reasons. First, since reinitialization should occur relatively rarely, promptness is not so important a design criterion as is inevitability and accuracy of the automatic procedure. Second, it is very likely that some low-numbered station will be active (one might intentionally assign bridges, gateways, and other high-availability servers low network numbers) so that reinitialization normally will occur very rapidly.

3) Some final, minor problems must be accounted for. Unless high-precision components are used the timers in different stations may be different enough to cause trouble. For example, if station 100 has a timer that is 1% slow, it may attempt reinitialization at the same time as station 101. This problem will arise only if there are no low-numbered stations, and the high-numbered stations have closely-spaced numbers. Similarly, a station may happen to join the ring in the middle of an ongoing reinitialization sequence, notice no flags, and try to initiate yet another reinitialization

sequence. Both these problems are eliminated by providing one more degree of backoff. A node should try exactly once to do reinitialization with the virtual token. If that attempt fails, it should get out of the way to let some other node try. If, after a few seconds, no station has successfully reinitialized the ring, automatic reinitialization is probably a hopeless activity anyway, and manual intervention should be called for. Assuming the ring is not actually damaged and thus the only problems are new participants and off-beat clocks, this try-at-most-once rule provides a very high probability of eventual success. Every active station will get to try, while collision-type interference becomes less and less likely as more stations exhaust their try and back off. (This observation suggests that one could even replace the systematic timeouts of the virtual token with random timeouts, and still expect a high probability of eventual reinitialization success. That approach would probably work, though with 250 stations it might be the usual case that many collisions occur at every reinitialization attempt.)

An Implication for interfaces

One of the attractions of this method of automatic ring reinitialization is that it can be implemented entirely in hardware as part of the ring controller, without involving host-specific hardware or software. This isolation of implementation between the ring controller and the station allows the interface between the ring controller and the station to be more technology-independent than it would otherwise be.

Application to the version 2 local network interface

Implementation of jamming, the virtual token, and the try-at-most-once rule for automatic ring reinitialization would involve several changes and some substantial simplification to the previous version 2 local network interface design [3]. The simplifications arise because in the previous design, ring reinitialization involved the ring controller card, the host-specific card, software in the host and four different timers. With this new approach automatic ring reinitialization can be carried out entirely by the ring controller card, and only two timers are required. Following is a list of mechanisms and procedures that would be implemented in each station interface.

1. The four timers that provide flag detect timeout (1.2 ms), token detect timeout (300 ms), Originate timeout (300 ms), and lost message timeout (1.2 ms), are replaced by two timers, all part of the control card. The first provides token detect timeout, and is set to a little more than the maximum possible token circulation time. The second provides flag detect timeout, and is set to a little more than the maximum ring transmit time. The principle of operation of the two timers is as follows: in a normally operating ring, the access token will periodically circulate by each repeater. Lack of appearance of the token is a certain indication that the ring requires reinitialization, so the token detect timeout is set to elapse if the maximum token circulation time is exceeded; whenever the repeater notices the token passing it resets this timer. The token detect timeout by itself would be sufficient to trigger reinitialization, but the maximum token circulation time can be quite large. The token can be captured by each station in sequence for one maximum-length message

transmission time, 1 ms. in the 10 Mb ring, so with 256 stations the token could take as long as 256 ms. to circulate. To detect ring failures more quickly, and to insure rapid agreement among all stations as to the starting time of the reinitialization procedure, a second timer is used. In a correctly operating ring, a flag sequence will appear at the beginning and end of every message and at the beginning of every token. Therefore, failure to see a flag within one ring transit time (determined by the repeater delay timer, the maximum number of repeaters, and the maximum length of wire connecting the repeaters--about 0.5 ms in the present design) is another certain indication that the ring requires reinitialization. The flag timeout detector therefore is set to elapse if more than one ring transit time is exceeded; whenever the repeater notices a flag passing it resets this timer. (The flag timeout detector by itself is _not_ sufficient to detect all need for ring initialization, since a failure could in principle leave the ring with a circulating flag and all stations in repeat mode.)

2. Whenever a station detects a token it loads a virtual token counter with the station address. A zero value in the virtual token counter inhibits entry to reinitialization mode. This zero-inhibit is the mechanism that implements the try-at-most-once rule.

3. Elapsing of either timer causes the control card to enter a reinitialization sequence unless reinitialization is inhibited by a zero in the virtual token counter. The ring control card performs the following sequence:

a. Immediately abandon any ongoing copy or originate operation returning error status to the host-specific board ("Message lost" on originate, "Bad format" on copy).

b. Continue as pending any pending originate operation.

c. Force the transmitter to idle (with PLL modem, send zeros) for one flag detect timeout time. (This idle ensures that every other flag detect timeout in the ring elapses.) Then return the transmitter and receiver to repeat mode.

d. Inhibit the token detect timeout.

e. Reset the flag detect timeout and allow it to run until it lapses or a flag is detected. If a flag is detected, the control card resets both detector timeouts, leaves reinitialization mode, and returns to normal operation.

f. If the flag detect timeout lapses, lower the virtual token counter by one, and if greater than zero, repeat step e.

g. If the virtual token counter reaches zero, reset both detector timeouts, originate a broadcast packet with no data, and after removing this packet return to normal operation.

Note that trying to initialize the ring inhibits any future attempts to reinitialize until such time as a token is detected. Thus for a single ring failure each station makes no more than one attempt to reinitialize.

4. If token detect timeout occurs while reinitialization is inhibited, this coincidence can be interpreted as failure of the ring reinitialization

procedure by all parties involved. This event should be reported as an error status to the host specific board. The line "ring not OK" is used for this purpose. Once asserted, the status "ring not OK" remains asserted until the next time a token is detected. (Note that this reuse of the token detect timeout is not strictly legitimate since the time normally required to complete the reinitialization procedure is only accidentally smaller than the the maximum token circulation time. However, it is very likely that if the ring is operating normally, some station will successfully reinitialize within that time. Further, the software response to this status is supposed only to log a status report and invoke external intervention. In the case that the ring later succeeds with reinitialization after this status is reported, the station can discover this fact by inspecting the status line.)

5. Whenever a station powers up, or enables or disables its modem, it should also reset the two timers and enable initialization by loading the virtual token counter. With this provision, startup of an LNI proceeds as follows:

   a) Power is applied to the LNI, at which time it comes up in digital loopback mode with both timeout detectors reset but active, the virtual token counter loaded, and idle (zeros) circulating by digital loopback.

   b) When the flag timeout detect elapses, reinitialization becomes active, should succeed, and digital loopback from then on circulates a token. The station can now test the LNI.

c) The station enables the LNI modem, cutting in the analog cable. A moment later the flag timer will elapse, and the analog loop will be initialized with a token. The station can now test the cable, and because a token is circulating reinitialization is not inhibited.

d) Now the station joins the ring, probably destroying the currently circulating ring token. Ring reinitialization automatically occurs, this time in concert with other ring participants.

If at any stage the station decides to abort the startup sequence, disabling the modem will reload the virtual token counter, thereby allowing the sequence to be tried again without need to power down.

6. Whenever a broadcast packet of zero length is received, it can be taken as a signal that the ring was just reinitialized. Individual stations may ignore this notice, log it, or investigate the status of their current connections, as appropriate.

7. Following a ring failure all LNI's will be inhibiting reinitialization and awaiting further instructions. After the ring is thought to be repaired, some LNI should be run through its startup sequence. This sequence, if successful, will end with a circulating token, which will notify every station that the ring is again operating. If, while the ring awaits repair, some new station attaches itself to the ring, it will attempt reinitialization and (presumably) fail. The token at the end of the broadcast message it launches may cause some set of stations to believe the ring is operating, time out, and reattempt initialization. These stations will soon reinhibit reinitialization, so such transients are harmless.

8. If joining the ring is accomplished by closing a relay, the jamming time T should be set to the larger of the ring transit time and the relay bounce time, so as to insure that reinitialization is not attempted until there is a chance it will work.

Plans

The initial implementation of the version two local network interface does not include automatic reinitialization, it uses instead the earlier design involving software. As an experiment, two wirewrapped control cards will be modified to implement automatic reinitialization, and it will be tried out by running them in a ring also containing unmodified control cards, with initialization software disabled. If successful, this design will be incorporated in all later control card assemblies.

Acknowledgements

David Clark, David Reed, J. Noel Chiappa, and Kenneth Pogran participated in several rounds of intense discussions that laid the groundwork for the ideas suggested here.

References

1. Farr, W., "Primenet Node Controller Specification," PE-T-307, August 24, 1978 (working paper).

2. Clark, D.D., Pogran, K.T., and Reed, D.P., "An Introduction to Local Area Networks," Proc. IEEE 66, 11 (November 1978) pp. 1497-1517.

3. Saltzer, J.H., "Version Two Local Net Interface Design Considerations," Massachusetts Institute of Technology Laboratory for Computer Science, Network Implementation Note 18, July 1979, (working paper).

4. Metcalfe, R.M., and Boggs, D.R., "Ethernet: Distributed Packet Switching for Local Computer Networks," Comm. ACM 19, 7 (July, 1976) pp. 395-404.

5. The CHAOSNET is currently undocumented.