AD-A099 140 STANFORD UNIV CA DEPT OF COMPUTER SCIENCE F/G 12/1
VERIFICATION OF LINK-LEVEL PROTOCOLS,(U)
JAN 81 D E KNUTH N00014-76-C-0330
UNCLASSIFIED STAN-CS-81-840 NL

END
DATE FILMED
8-81
DTIC

(11) Jan[illegible]1981

(12) 7

(12)

(14) Report. No. STAN-CS-81-840

LEVEL II

(6) Verification of Link-Level Protocols

by

(10) Donald E. Knuth

Contract N00014-76-C-0330

(15) NSF-MCS77-23731

Department of Computer Science

Stanford University
Stanford, CA 94305

Verification of Link-Level Protocols

by Donald E. Knuth

Abstract. Stein Krogdahl [1] has given an interesting demonstration of the partial correctness of a "protocol skeleton", by which the validity of the essential aspects of a large variety of data transmission schemes can be demonstrated. The purpose of this note is to present a simpler way to obtain the same results, by first establishing the validity of a less efficient skeleton and then "optimizing" the algorithms. The present approach, which was introduced for a particular protocol by N. V. Stenning [2], also solves a wider class of problems that do not require first-in-first-out transmissions.

1. Introduction.

Alice wants to send messages $M_0\, M_1\, M_2 \ldots$ to Bill over noisy transmission lines. They decide to handle the problem in the following way: Alice keeps a local variable

$A =$ the number of consecutive messages that Alice is sure Bill has received and stored;

Bill keeps a local variable

$B =$ the number of consecutive messages that Bill is sure he has received and stored.

Initially $A = B = 0$; we ignore problems of termination, since this can be dealt with as in [1]. Alice does two types of operations:

A1. Send message M_j, where j is an integer in the range $A \le j < A + k$.

A2. Receive an acknowledgment 'b' and set $A \leftarrow b$.

Bill also does two types of operations:

B1. Send an acknowledgment 'B'.

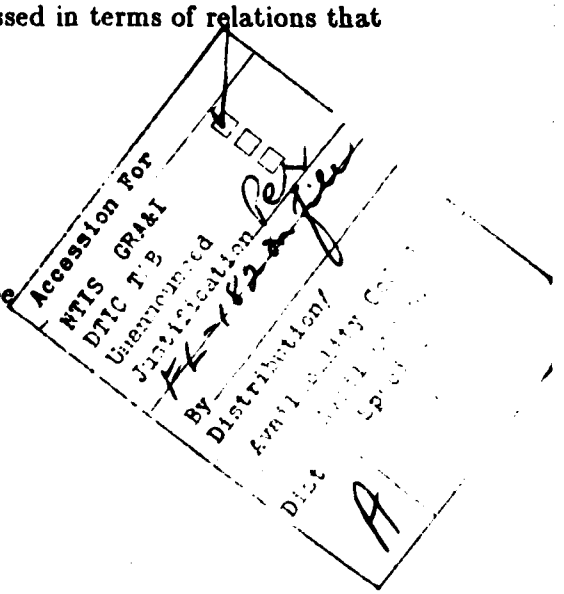
B2. Receive message M_j, and optionally store it; then set B to any value $b \ge B$ such that messages $M_0\, M_1 \ldots M_{b-1}$ have been received and stored.

Here k is a constant representing the size of some internal buffer storage maintained by Alice. We shall assume as in [1] that the "send" operation either inserts an item at the rear of a queue, or it causes nothing at all to happen; the latter event accounts for transmission errors, since a garbled message or a garbled acknowledgment will be treated as if it has not arrived at all. According to this convention, the sender does not know whether the sent item has been put into the queue or not. The "receive" operation is performed only when the queue is nonempty; in such a case the receiver reads and deletes the item at the front of the queue.

The preparation of this paper was supported in part by National Science Foundation grants MCS-77-23738 and IST-79-21977, and by Office of Naval Research contract N00014-76-C-0330.

Thus there are two queues, one containing messages and the other containing acknowledgments. The only essential difference between the above conventions and those of [1] is that we assume as in [2] that each message M_j in the message queue specifies its own integer index j, and each acknowledgment in the acknowledgment queue specifies an integer b, where j and b can be arbitrarily large. After this simple but unrealistic model has been examined, it will be clear that only a limited amount of information about j and b need actually be sent.

The particular order in which Alice and Bill decide to perform operations A1, A2, B1, and B2 is immaterial to us, and so are the particular choices of optional actions in steps A1 and B2. Our goal is to derive facts about any scheme that is based on these four operations; it is in this sense that we are st··· ing a "protocol skeleton" for a large class of conceivable protocols. The facts we shall ··· ve are expressed in terms of relations that remain *invariant* under all four operations A1, A2, B1, B2.



2. Invariants.

The first invariant relation we shall prove is

Lemma 1. *Let the contents of the acknowledgment queue be*

$$b_1 \dots b_r$$

from the front to the rear, where $r \geq 0$. *Then*

$$A \leq b_1 \leq \cdots \leq b_r \leq B.$$

Proof. This condition holds initially, when $A = B = r = 0$, and it is unaffected by operation A1. Operation A2 is performed only when $r > 0$, and it replaces $(A, b_1 \dots b_r)$ by $(b_1, b_2 \dots b_r)$; operation B1 either does nothing or replaces $b_1 \dots b_r, B)$ by $(b_1 \dots b_r B, B)$; and operation B2 has no effect except possibly to increase B. Thus the stated relation remains invariant. ∎

As a corollary of Lemma 1, we conclude that variable A never decreases during the course of a computation, since it changes only during A2. Notice that the invariant in Lemma 1 expresses a joint property of the entire communication system; although Alice does not know the value of B and Bill does not know the value of A, and although neither knows the contents of the queue, they can be sure that the unknown quantities satisfy the invariant relation. The introduction of system-wide invariants like this is one of the main features of Krogdahl's treatment.

Lemma 2. *Let the contents of the message queue be*

$$M_{j_1} \dots M_{j_r}$$

from the front to the rear, where $r \geq 0$, *and let* $j_{\max}$ *be the maximum index of any message that has ever been removed from the message queue. (If nothing has ever been removed, let* $j_{\max} = 0$.) *Let* $j_0 = j_{\max}$ *and* $j_{r+1} = A$; *then*

$$j_i < j_{i'} + k \quad \text{for } 0 \leq i < i' \leq r+1 .$$

Proof. Initially $r = 0$, so there is nothing to prove. Operation A1 either does nothing or replaces $j_1 \dots j_r$ by $j_1 \dots j_r j$ for some $A \leq j < A + k$; this leaves the stated relation invariant (we must consider two new cases, namely $j_i = j$ and $j_{i'} = j$). Operation A2 does not decrease A, as we have already observed, and operation B1 changes nothing. Operation B2 is performed only when $r > 0$, and it replaces $(j_{\max}, j_1 \dots j_r)$ by $(\max(j_{\max}, j_1), j_2 \dots j_r)$; again the relation remains invariant. ∎

3. Consequences.

The comparatively simple invariants proved in Lemmas 1 and 2 lead immediately to our main result:

Theorem. *If* M_j *is in the message queue, we have*

$$B - k \leq j < B + k .$$

If b *is in the acknowledgment queue, we have*

$$A \leq b \leq A + k .$$

Proof. We know from Lemma 2 that $j < A + k$ and from Lemma 1 that $A \leq B$, hence $j < B + k$. Furthermore $B - 1 \leq j_{\max}$, where $j_{\max}$ is defined in Lemma 2, because messages $M_0 M_1 \dots M_{B-1}$ have all been removed from the message queue; hence $B - 1 < j + k$ and $B - 1 < A + k$ by Lemma 2. This completes the proof, since $b \leq B$ by Lemma 1. ∎

The theorem tells us that only a limited amount of information about j needs to appear in the message queue, and only a limited amount about b needs to appear in the acknowledgment queue. Let us consider b first: If m_1 is any fixed integer $> k$, it suffices to send the remainder $B \bmod m_1$ instead of the arbitrarily large integer B in step B1, since Alice will be able to construct the full acknowledgment b from the remainder $b \bmod m_1$

received in step A2, given the fact that $A \le b \le A + k$. Indeed, the operation $A \leftarrow b$ is simply replaced by

$$A \leftarrow A + (b' - A) \bmod m_1$$

where $b' = b \bmod m_1$ is the acknowledgment that was received.

Let us suppose that Bill will store a message A_j that he receives in operation B2 only if $B \le j < B + l$, where l represents a fixed amount of buffer storage. There is of course no point in storing M_j when $j < B$, since all such message have already been stored. We might as well assume that $l \le k$, because j will always be less than $B + k$. In this case it suffices to send only the remainder $j \bmod m_2$ as an identification number for M_j, instead of the full integer j, provided that $m_2 \ge k + l$. For we know that the index j received by Bill in B2 must satisfy $B - k \le j < B + k$; the values of $j \bmod m_2$ in the range $B \le j < B + l$ are distinct, and they are disjoint from the values of $j \bmod m_2$ in the range $B - k \le j < B$ or $B + l \le j < B + k$. The fact that $(B + l) \bmod m_2$ might coincide with $(B - k) \bmod m_2$ does not matter; Bill would not store such a message in either case, and he doesn't care about the precise value of j when the message isn't being stored since such messages might as well have been dropped.

Krogdahl's paper [1] essentially discusses the case $l = 1$ and $m_1 = m_2 = k + 1$ in detail; he also gives a sketch of the case $l = k$, $m_1 = m_2 = 2k$ without proof. The argument above is not only simpler and more general, it shows that the modulo $m_1 = k+1$ and $m_2 = 2k$ are sufficient when $l = k$.

4. Generalization.

Krogdahl conjectured that the theory can be extended to the case where the queues do not quite operate in a first-in-first-out manner. It is clear that we cannot avoid sending the full integer j or b when the queuing discipline allows the deletion of items in arbitrary order, since small values might remain in the queue until they coexist with large ones. Let us suppose, however, that if entries are inserted in the order $x_1 x_2 x_3 \ldots$ and deleted in the order $x_{p(1)} x_{p(2)} x_{p(3)} \ldots$, then $p(1) p(2) p(3) \ldots$ is a permutation of the positive integers such that we have $|p(i) - i| \le q$ for all i. Furthermore the $p(i)$ must be consistent with the actual sequence of insertions and deletions made to the queue, in the sense that at least $p(i)$ elements must have been inserted at the time of the ith deletion. How does our previous analysis of the case $q = 0$ extend to this more general situation?

In the first place it is clear that the assignment at the end of operation A2 should be replaced by

$$A \leftarrow \max(A, b)$$

in this more general setting, otherwise the monotone growth of A would be destroyed.

Before considering the general protocol problem in detail, it is useful to study the general queuing discipline more carefully. If $i < i'$ and $p(i) > p(i')$, let us say that element $x_{p(i)}$ "passes" element $x_{p(i')}$, since it was inserted later but deleted earlier.

Lemma 3. *A permutation $p(1)p(2)p(3)\ldots$ of the positive integers satisfies the condition $p(i) \geq i - q$ for all i if and only if no element of the corresponding queuing discipline is passed by more than q other elements. It satisfies the condition $p(i) \leq i + q$ for all i if and only if no element of the corresponding queuing discipline passes more than q other elements.*

Proof. If $p(i) \geq i-q$ for all i, then $p(i') > i-q$ for all $i' > i$; but p is a permutation, so at least $i - q$ of the indices $i' \leq i$ have $p(i') \leq i - q$. This leaves at most q indices $i' < i$ that could have $p(i') > p(i)$; so $x_{p(i)}$ is not passed by more than q other elements. Conversely, if $p(i) < i-q$ for some i, then at most $p(i)-1$ indices $i' < i$ have $p(i') < p(i)$, so at least $i - p(i)$ indices $i' < i$ have $p(i') > p(i)$; in other words, at least $q + 1$ elements pass $x_{p(i)}$. The second half of the lemma follows from the first half, if we replace p by the inverse permutation. ∎

As long as we are generalizing the case $q = 0$, we might as well generalize further by supposing that the queuing discipline satisfies

$$i - q \leq p(i) \leq i + q'$$

for all i. Here $q = 0$ if and only if $q' = 0$, but each pair of positive integers (q, q') defines a different queuing discipline. We shall assume that the acknowledgment queue satisfies such a discipline with parameters q_1 and q_1', while the message queue satisfies such a discipline with parameters q_2 and q_2'.

Let $b_1\, b_2\, b_3\ldots$ be the entries that are inserted into the acknowledgment queue, and let $j_1\, j_2\, j_3\ldots$ be the indices of the messages inserted into the message queue. We can prove as before that $b_1 \leq b_2 \leq \cdots \leq b_n \leq B$, after n acknowledgments have been inserted; that $j_i < A + k$ for $1 \leq i \leq n$, after n messages have been inserted; and that

$$j_i < \quad \cdot k \quad \text{for } i \leq i < i'.$$

It follows that $A \leq B < A + k$.

We can now show that all entries b in the acknowledgment queue satisfy

$$A - q_1 k \leq b \leq A + k.$$

The upper bound is obvious, because $b \leq B$. To prove the lower bound, we may suppose that $b < A$. When b was first placed into the queue, we had $b = B \geq A$, so A must have

increased since then, by being set to other entries read from the queue. Suppose that n of these other entries have "passed" b, i.e., were inserted after b; only the entries inserted after b can have a value $> b$. Before the first such entry was read by Alice, we had $b \geq A$; afterwards we had $b \geq A-k$, because A cannot increase by more than k during operation A2. (All entries in the queue at that time are $\leq B$.) By induction we have $b \geq A-nk$ if n entries have passed b, but Lemma 3 tells us that $n \leq q_1$.

Finally, we can prove that all indices j in the message queue satisfy

$$B-k-q_2 \leq j < B+k.$$

Again the upper bound is obvious, since $j < A+k$. To prove the lower bound, suppose that n message indices have "passed" j in the queue; all other indices j' read by Bill satisfy $j' \leq j+k-1$. Therefore if Bill has received and stored messages $M_0 \ldots M_{B-1}$, we have $B-1 \leq j+k-1+n$, with equality only if the n messages that passed M_j were distinct messages whose index lies in the interval $[j+k, j+k-1+n]$. By Lemma 3, we have $n \leq q_2$.

It is not difficult to verify that the above inequalities on b and j are best possible, by constructing scenarios in which the extreme values occur. As before, we can conclude that it suffices to transmit only the residues $b \bmod m_1$ in the acknowledgment queue and $j \bmod m_2$ in the message queue, where m_1 and m_2 are any integers satisfying

$$m_1 > (q_1+1)k,$$
$$m_2 \geq k+l+q_2;$$

we assume that Bill has a buffer for receiving up to $l \leq k$ messages whose indices lie in $\{B, B+1, \ldots, B+l-1\}$. It is curious that q_1' and q_2' do not enter into these formulas.

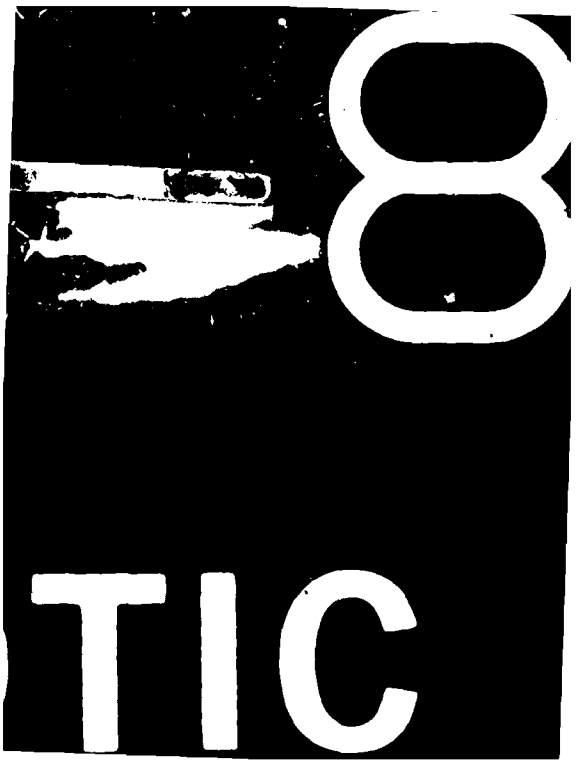
The protocol of Stenning [2] requires that at least one acknowledgment be transmitted per message received; in this special case the bound $m_1 \geq k+l+q_1$ is necessary and sufficient, where q_1 is the maximum number of other acknowledgments that can be sent and received between the transmission and receipt of any particular acknowledgment.

In practice, Alice is a system program that receives messages sequentially from some user, and Bill is a system program that delivers messages sequentially to another user. Therefore, as Krogdahl has observed, the variables A and B need not be explicitly maintained; only their values modulo a common multiple of m_1 and m_2 are needed.

[1] Stein Krogdahl, *Verification of a class of link-level protocols*, BIT 18 (1978), 436–448.

[2] N. V. Stenning, *A data transfer protocol*, Computer Networks 1 (1976), 99–110.

DEPARTMENT OF COMPUTER SCIENCE
STANFORD UNIVERSITY
STANFORD, CALIFORNIA 94305 USA


8
TIC