VERIFICATION OF BALANCED LINK-LEVEL PROTOCOLS.

Anneke A. Schoone and Jan van Leeuwen

RUU-CS-85-12
March 1985

Rijksuniversiteit Utrecht

Vakgroep informatica

Budapestlaan 6 3584 CD Utrecht
Corr. adres: Postbus 80.012 3508 TA Utrecht
Telefoon 030-53 1454
The Netherlands

VERIFICATION OF BALANCED LINK-LEVEL PROTOCOLS.

Anneke A. Schoone and Jan van Leeuwen

Technical Report RUU-CS-85-12
March 1985

Department of Computer Science
University of Utrecht
P.O. Box 80.012
3508 TA Utrecht
the Netherlands

VERIFICATION OF BALANCED LINK-LEVEL PROTOCOLS.

Anneke A. Schoone and Jan van Leeuwen

Department of Computer Science, University of Utrecht,
P.O.Box 80.012, 3508 TA Utrecht, the Netherlands.

Abstract. We show that the alternating bit protocol and the "balanced" two-way sliding window protocol are instances of one general protocol skeleton, that contains several further parameters to tune the simultaneous transmission of data between two senders over a full-duplex link. The partial correctness of the protocol skeleton is proved by the Krogdahl-Knuth technique of system-wide invariants. We discuss the dependence of the optimal choice of parameters in the protocol skeleton on the propagation delay of the link and the processing speed of the senders.

1. Introduction. Consider two computing stations S_A and S_B (e.g. computers) that exchange information over a full-duplex link. Several link-level protocols exist to control the exchange of packets and to guard against the loss of information when the transmission medium is unreliable (cf. Tanenbaum [9], Ch.4). The alternating bit protocol (Lynch [7], Bartlett et al. [1]) is a classic example of a protocol that works on a packet-by-packet basis. The sliding window protocol (Cerf and Kahn [4]) controls the transfer of groups (windows) of packets from S_A to S_B, with acknowledgements being transmitted from S_B to S_A. An extended (symmetric or "balanced") version of the sliding window protocol for the two-way simultaneous transfer of data between two stations was discussed by e.g. Bochmann [2] and Carlson [3].

In this paper we show that the alternating bit protocol and the "balanced" two-way sliding window protocol are instances of one general protocol skeleton, that facilitates a mathematical analysis of the correctness and the performance of the protocols. The result substantiates and generalizes the common belief that the alternating bit

protocol is "a (balanced) sliding window protocol with window size = 1". The partial correctness of the general protocol skeleton is proved by a technique due to Krogdahl [6] and Knuth [5] using system-wide invariants. Several parameters are used in the protocol skeleton that enable one to tune the performance of the protocol, depending on the propagation delay of the link and the processing speed of the senders. In the remainder of this section we present the general protocol, following the style of [5].

Suppose S_A wants to send a sequence of messages M_0, M_1, M_2, ... to S_B, while simultaneously S_B wants to send messages N_0, N_1, N_2, ... to S_A. Since we allow that messages are not strictly sent in order, we will assume for the time being that every message M_j or N_j carries its index j as a (unique) sequence number. No separate acknowledgements will be exchanged, not even by "piggy-backing" acks on M- or N-messages. Instead, when a message M_j is sent, its sequence number (j) is used as an implicit acknowledgement for an N-message, say, N_{j-g}. Likewise, if S_A receives a message N_j it interprets it as an implicit acknowledgement of an M-message, say, M_{j-f}. The parameters f and g are global constants known to both S_A and S_B. In addition to the global constants f and g, S_A needs two local variables :

A = the number of consecutive messages that S_A is sure S_B has received and stored,

a = the number of consecutive messages that S_A has received and stored.

Likewise, S_B needs the similar corresponding variables B and b. Initially a=b=0, A=-f and B=-g. As in [5], we do not consider problems of termination. S_A can do the following two types of operations :

A1 : send message M_j, where $\min(A,a+g-1) \leq j < a+g$.

A2 : receive message N_i and optionally store it;

<u>if</u> i-f+1 > A <u>then</u> A := i-f+1;

a := new value such that N_0, N_1, ... N_{a-1} have been received and stored.

Likewise, S_B can do the following two types of operations :

B1 : send message N_i, where $\min(B,b+f-1) \leq i < b+f$.

B2 : receive message M_j and optionally store it;
if $j-g+1 > B$ then $B := j-g+1$;
b := new value such that M_0, M_1, ... M_{b-1} have been received and stored.

We assume that both S_A and S_B buffer incoming messages in a FIFO-queue. As in [5] we shall assume that "sending" means either inserting a message at the rear of a queue, or loosing it. "Receiving" is done only when the queue is nonempty, in which case the message at the front of the queue is read and deleted from the queue. Thus there are two queues : one with messages on their way from S_A to S_B, the S_A to S_B queue, and one with messages on their way from S_B to S_A, the S_B to S_A queue. For the moment we will let the indices i and j grow arbitrarily large, just for the sake of argument. For a practical implementation of a protocol according to this skeleton, we need a way to decide when to perform operations A1 and A2, and B1 and B2, respectively, and to decide which message to send. In particular, we need a timeout mechanism to ensure retransmission of possible lost or damaged messages.

As the constants f and g are related to the implicit acknowledgements, we require that $f,g \geq 0$. In addition to this, we also want $f+g \geq 1$. The case f=g=0 corresponds to the deadlocking protocol where S_A waits for S_B to send and S_B waits for S_A to send, as is easily seen. We will see later that the alternating bit protocol is obtained by setting f=0 and g=1 (or f=1 and g=0).

The remainder of the paper is organized as follows. In section 2 we will derive the system-wide invariants that hold for the protocol skeleton. Section 3 contains the main theorems on the partial correctness of the protocol skeleton and on the special case of the alternating bit protocol. Section 4 discusses the selection of the parameters f and g.

2. Invariants. As the relations we will now derive remain invariant under all four operations A1, A2, B1 and B2, we are not interested in the actual order in which the operations are performed. Since the protocol is symmetric (balanced), all relations we prove for S_A will have a symmetric counterpart for S_B. Thus we will only state but not prove the symmetric relations, as the proofs are symmetric too.

Lemma 1. (i) A and a are nondecreasing.
(ii) B and b are nondecreasing.

Proof. Operations A1, B1 and B2 change neither A nor a. If A is changed in A2, it is increased. Similarly, if a receives a new value in A2, it must increase. Q.E.D.

Lemma 2. (i) $A \geq a-f$
(ii) $B \geq b-g$.

Proof. Initially $A = -f$ and $a = 0$, and the relation holds. Operations A1, B1 and B2 do not change A or a. In operation A2, if only A is changed it is increased, thus keeping the relation valid. When a is changed, N_0, N_1, ... N_{a-1} must have been received and stored. In particular N_{a-1} has been received. At that moment either $A \geq (a-1)-f+1 = a-f$, or we had $A < a-f$ and A became a-f. Since A is nondecreasing, $A \geq a-f$ still holds. Q.E.D.

Lemma 3.
(i) In A1, the number of possible values for j is at least 1 and at most f+g.
(ii) In B1, the number of possible values for i is at least 1 and at most f+g.

Proof. The value of a+g-1 is always a possibility for j. If $A \leq a+g-1$, then the number of possible values for j is $a+g-A \leq a+g-(a-f) = f+g$. Q.E.D.

Lemma 3 implies that the buffer space S_A needs for messages that might have to be retransmitted is bounded by a constant : f+g. Observe that by choosing for the lower bound of j in A1 the minimum of A and a+g-1, we keep the possibility for S_A to send at least one message open. Note that although the value a+g-1 is always a possibility for j, this value might be equal to -1, which does not correspond to the index of a message to send. This happens if a=0 and g=0. However, since $f+g \geq 1$, we must have $f \geq 1$ and $b \geq 0$, and it follows that there is a non-negative possibility for i. S_A will have to wait with sending until it receives N_0 from S_B.

Lemma 4. (i) $A \leq b$,
(ii) $B \leq a$.

Proof. Initially, $-f = A \leq b = 0$. A1 and B1 do not change A or b. B2 might increase b, but this does not destroy the inequality. A2 might increase A to A = i-f+1 if a message N_i was received. However, when N_i was sent, the relation $i < b+f$ was valid. Since b is nondecreasing, it still holds. Hence $A = i-f+1 < b+f-f+1 = b+1$, so $A \leq b$. Q.E.D.

Lemma 5.

(i) Let the contents of the S_A to S_B queue be $M_{j_1} \ldots M_{j_r}$ 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 S_A to S_B queue. (If nothing has ever been removed, let $j_{max} = -1$.) Let $j_0 = j_{max}$ and $j_{r+1} = a-f$, then
$$j_k < j_{k'}+f+g \quad \text{for} \quad 0 \leq k < k' \leq r+1.$$

(ii) Let the contents of the S_B to S_A queue be $N_{i_1} \ldots N_{i_s}$ from the front to the rear, where $s \geq 0$, and let i_{max} be the maximum index of any message that has ever been removed from the S_B to S_A queue. (If nothing has ever been removed, let $i_{max} = -1$.) Let $i_0 = i_{max}$ and $i_{s+1} = b-g$, then
$$i_k < i_{k'}+f+g \quad \text{for} \quad 0 \leq k < k' \leq s+1.$$

Proof. Initially the S_A to S_B queue is empty, $j_0 = -1$, and $j_{r+1} = a-f = -f$. Hence the relation $-1 = j_0 < j_{r+1}+g+f = g$ holds, since $g \geq 0$. A2 and B1 do not change the contents of the S_A to S_B queue. A2 might increase j_{r+1}, which preserves the inequalities too. A1 adds a $M_{j_{r'}}$ at the end of the queue, with necessarily $\min(A,a+g-1) \leq j_{r'} < a+g$. Distinguish two cases :

case 1. $\min(A,a+g-1) = A$. It follows that $j_{r'} \geq A \geq a-f$.

case 2. $\min(A,a+g-1) = a+g-1$. It follows that $j_{r'} \geq a+g-1 \geq a-f$ since $f+g \geq 1$.

$j_k \leq j_{r+1}+f+g = a-f+f+g \leq j_{r'}+f+g$ and $j_{r'} < a+g = a-f+f+g = j_{r+1}+f+g$. Hence the invariant remains valid under A1. B2 removes M_{j_1} from the front of the queue, and $j'_{max} = \max(j_{max}, j_1)$. Since the invariant was valid for both j_{max} and j_1, it remains valid for the maximum of both. Q.E.D.

3. Consequences. The lemmas shown in section 2 lead to the following results that characterize the effect of the protocol skeleton.

Theorem 1. (i) If M_j is in the S_A to S_B queue, we have

$$b-f-g \leq j < b+f+g.$$

(ii) If N_i is in the S_B to S_A queue, we have

$$a-f-g \leq i < a+f+g.$$

Proof. Let M_j be in the S_A to S_B queue. By lemma 5, $j < j_{r+1}+f+g = a-f+f+g = a+g$. From lemmas 2 and 4 follows $a+g \leq A+f+g \leq b+f+g$. It follows that $j < b+f+g$. Next, let us look closer at the relationship between B and j_{max}. Initially, $j_{max} = -1$ and $B = -g$. Later,

$j_{max} = \max(j;\ M_j$ removed from S_A to S_B queue) and

$B = \max(j-g+1;\ M_j$ removed from S_A to S_B queue)

Hence, $B = j_{max}-g+1$. By lemma 5 $j_{max} < j+f+g$, hence $j > j_{max} -f-g = B-f-1$. By lemma 2 $B-f \geq b-g-f$, and it follows that $j \geq b-f-g$. Q.E.D.

It follows from theorem 1 that it is not necessary for S_A and S_B to

use and send the full indices i and j as sequence numbers, but that it is sufficient to send their remainder mod n, where n is any fixed integer with $n \geq 2(f+g)$.

Theorem 2. The number of integers in [A, a+g) is smaller than k iff the number of integers in [B, b+f) is larger than f+g*k.

Proof. a+g*A < k implies A*a*g > *k, hence A+f*a > f+g*k. Lemma 4 gives b+f*B $\geq$ A+f*a > f+g*k. Conversely, b+f*B > f+g*k implies b*B*g > *k implies B+g*b < k. Lemma 4 gives a+g*A $\leq$ B+g*b < k. Q.E.D.

Hence, when one sender is restricted in its choice of messages to send, the other sender has more messages to choose from, and vice versa.

Theorem 3. The alternating bit protocol is obtained as a special case of the general protocol skeleton by setting f and g such that f+g=1.

Proof. Let f=0 and g=1. This corresponds to the case where S_A begins sending. If we want S_B to begin, take f=1 and g=0. As we have seen in lemma 3, the number of possible values for i and j is bounded by f+g=1. Hence S_A and S_B each have exactly one message to send, as in the alternating bit protocol. This one message could be a retransmission of the last one, if no acknowledgement for it was received, or the next one, if an acknowledgement (i.e. a next message from the other) was received and a and A were increased by one accordingly. The consequence of theorem 1 was, that it is sufficient to send i and j mod n=2(f+g)=2. Hence one bit set to 0 or 1 is sufficient. It follows that the alternating bit protocol is a special case of the protocol skeleton with f=1 and g=0, if we impose the ordering of operations as A1, A2, A1, A2, ... on S_A and B2, B1, B2, B1, ... on S_B. Q.E.D.

As illustration we reformulate the alternating bit protocol, as it can now be expressed.

```
Protocol for S_A :
A:=0; M:=M_0; (*A=a always*)
while no termination do
begin send message <M,A mod 2>;
      receive message <N,bit>;
      if bit=A mod 2
      then begin store N; A:=A+1; M:=M_A end
end.
```

```
Protocol for S_B :
B:=1; (*B=b+1 always*)
while no termination do
begin receive message <M,bit>;
      if bit≠B mod 2
      then begin store M; B:=B+1; N:=N_B end;
      send message <N,B mod 2>
end.
```

4. The choice of f and g. It is clear that a successful exchange of messages not only depends on the choice of the parameters f and g and on the error rate of the link over which the messages are sent, but also on a sensible choice and timing of A1 and A2 for S_A, and of B1 and B2 for S_B. For example, if both S_A and S_B only send but refuse to receive, then neither party will get anywhere. On the other hand, the choice of f and g will depend on things like: the rate at which messages can be sent or received, and the propagation delay on the link. For a more precise analysis we define the following quantities :

d_{AB} = propagation delay from S_A to S_B,
d_{BA} = propagation delay from S_B to S_A,
s_A = minimum time between two operations A1,
s_B = minimum time between two operations B1,
r_A = minimum time between two operations A2,

r_B = minimum time between two operations B2,

$m = \max(s_A, s_B, r_A, r_B)$.

We allow the minimum time between A1 and A2 and between B1 and B2, respectively, to be zero. Hence we neglect the time to perform the operations, or rather, the execution times are assumed to be contained in s_A, s_B, r_A and r_B. To make matters simple we assume for the moment that these quantities are constant. To make matters simpler still, we begin with assuming that there are no transmission errors and that no messages are lost.

Definition. Strategy I is the order of operations determined by the following algorithm (formulated in terms of S_A) :
do A2 as soon as possible depending on r_A and the contents of the S_B to S_A queue, as well as
do A1 as soon as possible for the next message (beginning with the sending of M_0) depending on s_A and the values of A and a.

Hence in strategy I there are no retransmissions, nor are messages sent out of order. We say that S_A and S_B communicate with speed x if the time between consecutive A1's, A2's, B1's and B2's respectively is x.

Lemma 6. If no messages are lost and there are no transmission errors, and S_A and S_B operate under strategy I, then eventually S_A and S_B communicate with speed m if

$$f+g \geq (d_{AB}+d_{BA})/m.$$

Proof. Say S_A wants to send M_j at time t. S_A is only allowed to do so if $j < a+g$. The value of a is determined by the indices of the incoming N messages. Since S_A and S_B use strategy I, and there are no losses or errors, a equals i+1 if N_i was the last message received so far. Hence $j < i+1+g$ and $i \geq j-g$, so at least N_{j-g} must have been received by S_A. That means it must have been sent by S_B before time $t-d_{BA}$. But for S_B to be able to send N_{j-g}, one must have $j-g < b+f$ and (hence) $b > j-f-g$. This

means that some $M_{j'}$ must have been received by S_B before time $t-d_{BA}$, with $j' \geq j-f-g$. So M_{j-f-g} must have been sent by S_A before time $t-d_{BA}-d_{AB}$. Since S_A sends one message in m time, M_{j-f-g} was sent at time $t-(f+g)m$. $t-(f+g)m \leq t-d_{BA}-d_{AB}$ implies $f+g \geq (d_{AB}+d_{BA})/m$. Q.E.D.

See figure 1 for what happens for a specific choice of all parameters if S_A and S_B use strategy I over an error-free link. It is clear that we get queuing delays if $r_A < s_B$ and/or $r_B < s_A$.

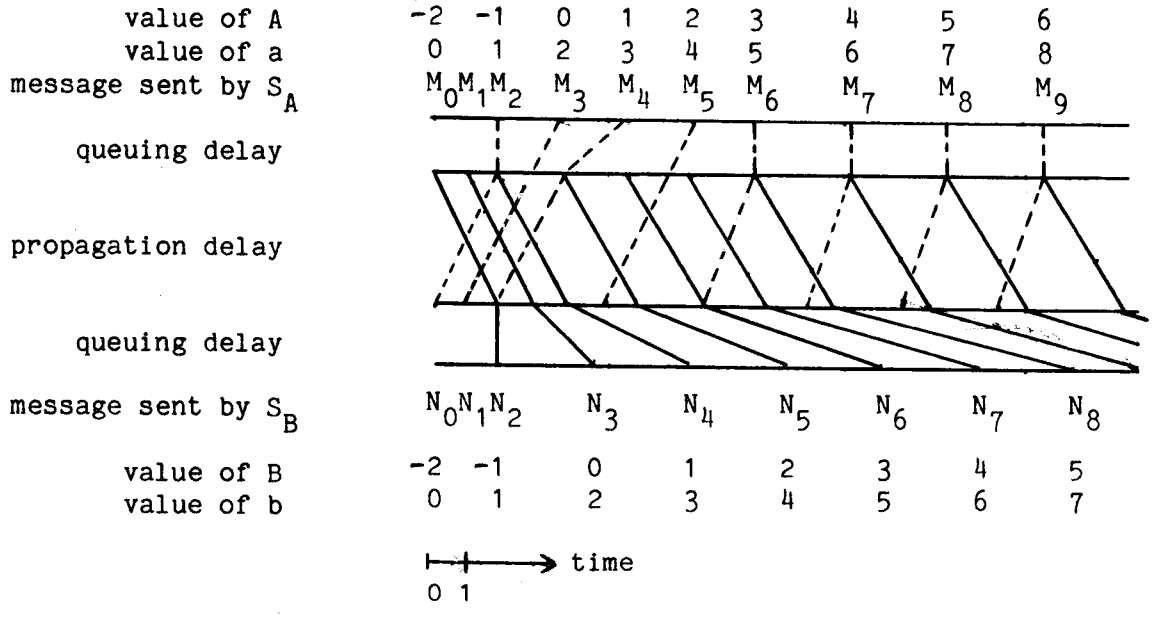

Figure 1.

Communication between S_A(top) and S_B(bottom) for strategy I with $s_A=s_B=1$, $r_A=2$, $r_B=3$, $d_{AB}=d_{BA}=2$ and $f=g=2$. —— M messages from S_A to S_B, — — — N messages from S_A to S_B.

Lemma 7. Let $f+g \geq (d_{AB}+d_{BA})/m$, and suppose S_A and S_B operate under strategy I over an error-free link. If eventually there is a non-empty queue of messages, then the total queuing delay (i.e., the queuing delay for one message from S_A to S_B, plus the queuing delay for one message from S_B to S_A) is $(f+g)m-(d_{AB}+d_{BA})$.

Proof. As we saw in the proof of lemma 6, the time between sending a message and receiving its acknowledgement is $(f+g)m$. Since only $d_{AB}+d_{BA}$ accounts for the propagation delay, the rest must be queuing delay. Q.E.D.

Note that if there are no queues, we still have the extra delay, except that it is spent as idle waiting time for the next message to arrive, by S_A and/or S_B.

Lemma 8. If S_A and S_B never retransmit a message, then the total number of messages on their way from S_A to S_B or from S_B to S_A is never more than $f+g$ at one moment.

Proof. Initially, if S_A and S_B have not received any messages yet, they can send at most $f+g$ messages (g for S_A and f for S_B). Let some station, say S_B, have received a message, and suppose S_B is now sending N_i. Let $N_{i'}$ be the message with the smallest index which S_A has not received yet. Hence $a \leq i'$. Let the current value of b be b'. This implies that S_B has received $M_0, M_1, \ldots M_{b'-1}$. The message with the highest index which S_B can send is $N_{b'+f-1}$. Thus there are at most $b'+f-1-i'+1$ messages on their way from S_B to S_A. Since the message with the highest index which S_A can send is M_{a+g-1}, there are at most $a+g-1-b'+1$ messages on their way from S_A to S_B. Thus the total number of messages "on their way" (i.e. in the link and in the queues) is at most $b'+f-i'+a+g-b' \leq b'+f-i'+i'+g-b' = f+g$. Q.E.D.

Lemma 8 tells us something about the amount of buffer space needed by S_A and S_B for the incoming messages.

Theorem 4. The optimal values for communication between S_A and S_B are $f=\lceil d_{AB}/m \rceil$ and $g=\lceil d_{BA}/m \rceil$.

Proof. It should be clear that optimality means that no unnecessary idle waiting time or unnecessary queuing delays are incurred. Hence we will have to choose f+g around $(d_{AB}+d_{BA})/m$ (lemmas 6 and 7). This also minimizes the delay between the occurrence of a message loss or error and the suspicion thereof by the sender. Moreover, we want that each station has enough messages to send before it can do the first receive. This means for S_A that it must have received N_0 by the time it wants to send M_g (otherwise it will not be allowed to send it). Hence $g.m \geq d_{BA}$. Similarly for S_B $f.m \geq d_{AB}$. Q.E.D.

It is clear from the above that it only makes sense to choose the timeout time at S_A and S_B larger than (f+g)m. One of our assumptions was that r_A, r_B, s_A and s_B are constants. This is not very realistic if S_A and S_B are not the only stations to communicate over the link(s). As long as the variance is small, this will not influence the results too much, although we will need a larger buffer space for the queues, or take extra losses because of buffer overflow for granted. But if the values vary wildly, the analysis does not help us much. One would like to change the value of f and g while S_A and S_B are sending, to adapt to the changed circumstances. Recall that we did not send the actual value of the indices, but their remainder modulo some $n \geq 2(f+g)$. Increasing f and g will lead to great problems if this means that this inequality does not hold any more for n. Moreover, if f and g are decreased we still have the problem that we might get the indices and their implicit meaning as acknowledgements mixed up. Messages may have been sent out of order or got lost. Hence it seems necessary to completely clear the link and all queues before we can change f and g. A better strategy seems to be the following in case we would want to decrease f and g. We want to decrease f and g if the queuing delays have become too big. Say there is a large queue at S_B. Then it would help S_B if S_A stopped sending for a

while to give it time to empty its queue, and resumed sending with a speed adapted to S_B's. Then the total delay is decreased, without actual decreasing f and g. In figure 1, if S_A waits with sending M_7 until t=19, the time at which it now sends M_9 , and from then on keeps sending with s_A=3, then the time a message from S_A to S_B is on its way decreases from 10 to 4. However, the time between sending a message and receiving its acknowledgement of course stays the same .

References.

(Reference [8] is not cited in the text.)

[1] Bartlett, K.A., R.A.Scantlebury, and P.T.Wilkinson, A note on reliable full-duplex transmission over half-duplex links, C.ACM 12(1969) 260-261.

[2] Bochmann, G.V., Logical verification and implementation of protocols, Proc. 4th Data Communications Symp., Quebec, Canada, 1975, pp. 8-5-8-20.

[3] Carlson, D.E., Bit-oriented data link control, in: P.E.Green Jr (Ed.), Computer Network Architectures and Protocols, Plenum Press, New York, NY, 1982, pp. 111-143.

[4] Cerf, V.E., and R.E.Kahn, A protocol for packet network intercommunication, IEEE Trans. Communications COM-22 (1974) 637-648.

[5] Knuth, D.E., Verification of link-level protocols, BIT 21 (1981) 31-36.

[6] Krogdahl, S., Verification of a class of link-level protocols, BIT 18(1978) 436-448.

[7] Lynch, W.C., Reliable full-duplex file transmission over half-duplex telephone lines, C.ACM 11(1968) 407-410.

[8] Stenning, N.V., A data transfer protocol, Comp. Netw. 1(1976) 99-110.

[9] Tanenbaum, A.S., Computer Networks, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1981.