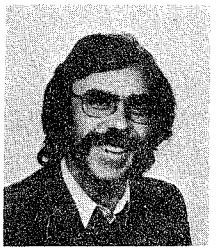
Verification and Evaluation of Communication Protocols

Gérard LE LANN and Hervé LE GOFF
Institut de Recherche en Informatique et Systèmes Aléatoires, Université de Rennes, Rennes, France

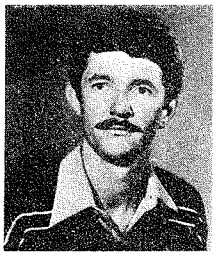
Transmission errors, failures and variable transit delays are important characteristics of computer communication networks. Hence, designing a reliable communication protocol for such an uncertain environment is a challenging task. A case study is reported which shows how helpful heuristic techniques can be in protocol verification and a theorem is given regarding synchronization of communicating entities.

Another facet of the communication protocol design issue is efficiency. Heuristic techniques have been used also to evaluate performances of various flow control mechanisms intended for internode protocols and transport protocols as well as performances as seen by users of computer networks. Finally data regarding tradeoff choices are given.

Keywords: packet-switching, simulation, flow control, communication protocols, interprocess synchronization, performance evaluation.



G. Le Lann received his Diplôme d'Ingénieur en Informatique from ENSEIHT, University of Toulouse and his Doctorat d'Etat in 1977, from the University of Rennes. Past activities include design and implementation of a real-time system for the French Navy and a seven computer star network at CERN (Switzerland). In 1972, G. Le Lann joined the Cyclades Project. Since 1974, he is leader of the Computer Network Group at IRISA, Rennes. Recent work includes formalization, design and performance evaluation of computer networks and distributed systems.



H. Le Goff received his Doctorat de Troisième Cycle in 1976 from the University of Rennes. In 1972 and 1973, H. Le Goff was with the Cyclades project at IRIA, participating in the software implementation of the first Cyclades transport stations. Since 1974, he is with the Computer Network Group at IRISA, Rennes. His main activities are related to the design and the performance evaluation and implementation of end-to-end protocols.

Computer Networks 2 (1978) 50–69

Introduction

First transport protocol specifications for the Cyclades computer network were issued in November 1972. It was felt necessary to invest some effort in assessing these specifications before embarking upon an implementation. The responsibility for this was given to the Computer Network Group of IRISA (Institut de Recherche en Informatique et Systèmes Aléatoires). Results of this work led to the definition of a new transport protocol, and are reported in part I.

For the last years, activities in protocol design have been blossoming, specially inside IFIP WG 6.1 (INWG). Part II includes detailed performance studies intended for current internode protocols, transport networks and users of transport networks.

This paper is a complete version of results partially published before [3–6], unpublished material and recent developments.

Part I. Heuristic techniques in verification of communication protocols

1. Introduction

In the Cyclades network [8], the transport protocol and the transport station (TS) are roughly equivalent to the host–host protocol and the NCP in the ARPA network. The initial design of the Cyclades transport protocol included three modes of operation: regular letters, letters on connections (liaisons) and letters on virtual channels (voies virtuelles).

Connections and VC's are opened and closed identically. Nevertheless, they provide for a different service. On VC's, sequential LT references are given by the TS. Error and flow control are performed automatically on a LT basis. A LT is not size-limited.

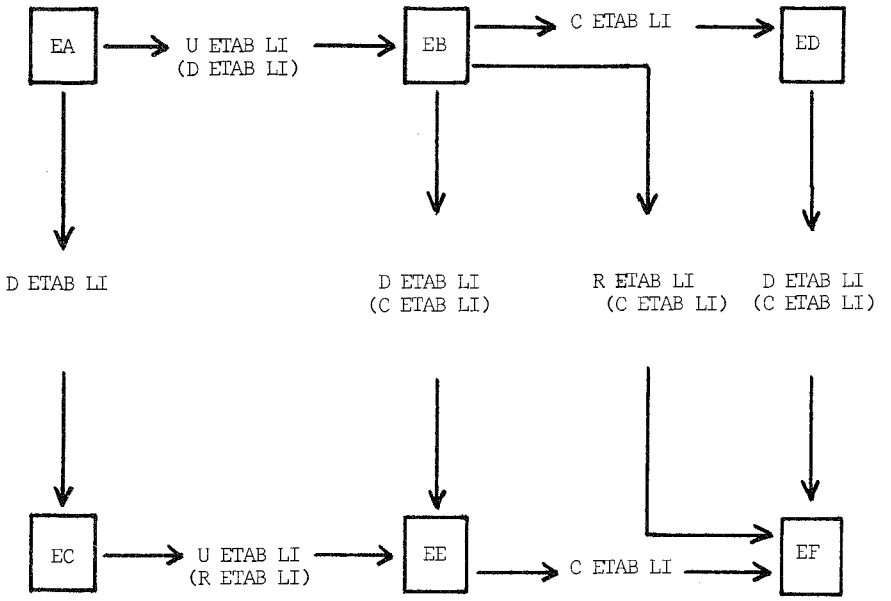
On connections, LT's are size-limited (255 octets) and are mutually independent. LT references are given by the user. LT's can be or not error-controlled. Flow control is dynamic and works as follows: a credit (CR) must be allocated to the sender for each LT to be sent. Requests must be made continuously

by the sender according to the current load and are expressed as absolute values. CR requests are sent within normal user LT's or within acks. According to local availability, the receiver goes on returning CR values to the sender.

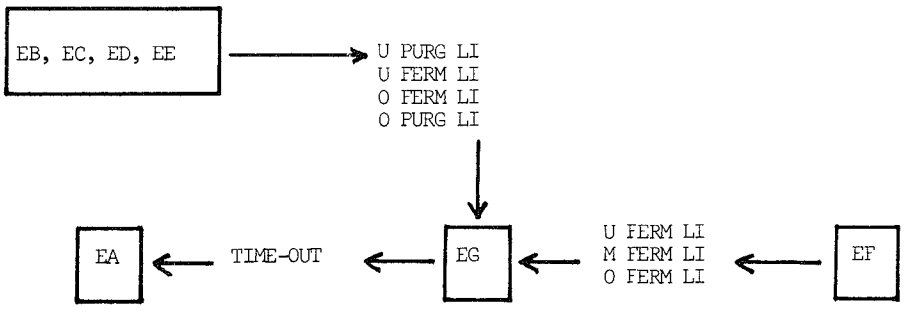
To open a link (connection or VC), a command called U-ETAB-LI must be activated by the TS user. This results in the transmission of a message D-ETAB-LI sent to distant TS. The local TS switches from internal state EA to state EB (fig. 1). A different path is followed if a distant message D-ETAB-LI is received first (EA → EC).

Setting up a connection or a VC is done according to a Demand + Response + Confirmation scheme along with negotiation of parameters.

D-ETAB-LI is a "demand" message which carries



OPENING A CONNECTION OR A VC



CLOSING A CONNECTION OR A VC

Fig. 1. Opening and closing a connection or a VC.

parameters indicating the offered local characteristics and the required distant characteristics. R-ETAB-LI is a "response" message, the parameters of which indicate the allocated local characterisics and the accepted distant characteristics. Similar parameters are included in C-ETAB-LI, which is a "confirmation" message.

A TS can go through seven internal states. EA is the idle state. EB, EC, ED and EE are intermediate states, all protected by a time-out. EF is the active state: transmission and reception of letters are allowed. EG is the resetting state during which a TS is "unavailable" for a given user, while returning to state EA. EG is also controlled by a time-out.

On a connection or a VC, two addressing formats can be used: general address: to be used during the set-up phase; short address: to be used after completion of the set-up phase (EF at both ends). This was intended to reduce transmission overhead. A short address is dynamically allocated.

Two different "reset" commands were also provided to TS users: a close command (U-FERM-LI) using short addressing, and a purge command (U-PURG-LI) using general addressing.

Our objective was to investigate how reliable that scheme was. Because of the specific characteristics of computer networks like failures, errors, race conditions, etc., the most convenient approach was the use of a simulation model [5]. Formal techniques were almost inexistent and the conclusions of a recent survey [9] were valid a fortiori at that time. However, simulation is not an easy game to play when the rule is: build a piece of software which will show you something you do not expect and you do not have the slightest idea about the way it may appear. Because of this, special attention was paid to outputs (see below). In order to discover possible inconsistencies or abnormal conditions, user commands at both ends were issued at random and the simulated subnetwork behaved like a real one, i.e., it introduced variable transmission delays and losses of packets.

.1. Outputs

Examples are given in figs. 2 and 3. Two TS' contexts are represented. From left to right, we have:

Logical references: each event is given a unique logical code which traces down origins and consequences of any event. For instance, in fig. 3, action 24 – reception of a D-ETAB-LI with foreign parameters not matching the local ones – generates two other actions referenced by code 725: transmission of a CLOSE message (0-FERM-LI) and awaking of a timer (TIMOUT-LI, initial value = 1000). An arrow is used to trace commands issued by the user and code 0 indicates that an action has no consequence.

Chronological references: logical codes are followed by the value of the simulated time.

State: the internal state of each TS is given for every event.

Negotiated parameters (two integer values): examples of negotiated parameters are the maximum letter length, size of the element (for VC's), minimum initial buffer space, etc. Disagreement leads to an abort of the attempt to open a connection or a VC.

Event: events are user commands, prefixed by a U or message transmission and reception. Examples are: U-ETAB-LI, user request to open a connection or a VC; D-ETAB-LI, the corresponding request is sent to the distant TS; C-ETAB-LI, confirmation for opening a connection or a VC.

Comments: in the last two columns, internal decisions or events are indicated in "clear language". For instance, 22/7: values of the simulated negotiated parameters; LOST: this message is lost by the subnetwork; INSU: negotiated parameters do not match; KLOK: waking of a timer.

2. Results

2.1. Opening and closing a communication link

Simulations put to evidence the fact that absolutely reliable techniques for opening and closing a connection or a VC do not exist. In this context, reliability means that the communicating parties are always in agreement. Let us call desynchronization the situation of a TS not being in the state as expected by the other TS. Losses, errors and variable transmission delays are the three facets of uncertainty. From such characteristics, it is easy to infer that desynchronization may occur. Two examples are given below.

Example 1. An interesting situation was shown to occur because of the existence of two different reset commands (fig. 2). Station A (left, state EF) and station B (right, state EF) are exchanging letters on a connection; at time 465 995, user B decides to close the connection (U-FERM-LI, code 57). Unfortunately, the corresponding message is lost on the subnet. At time 466 995, station B is reset to EA (normal time action). It is now impossible for station B either

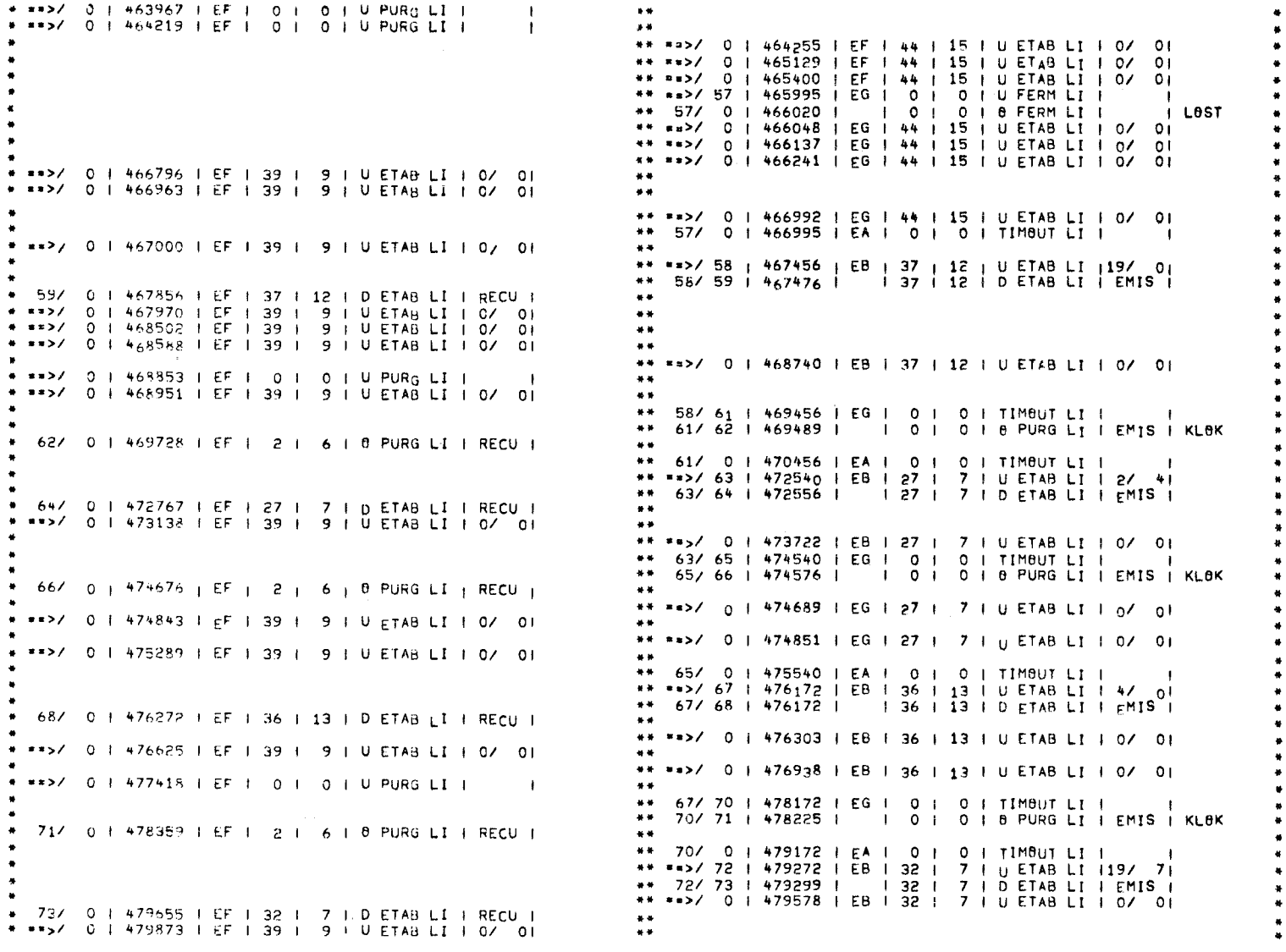


Fig. 2. Example of output.

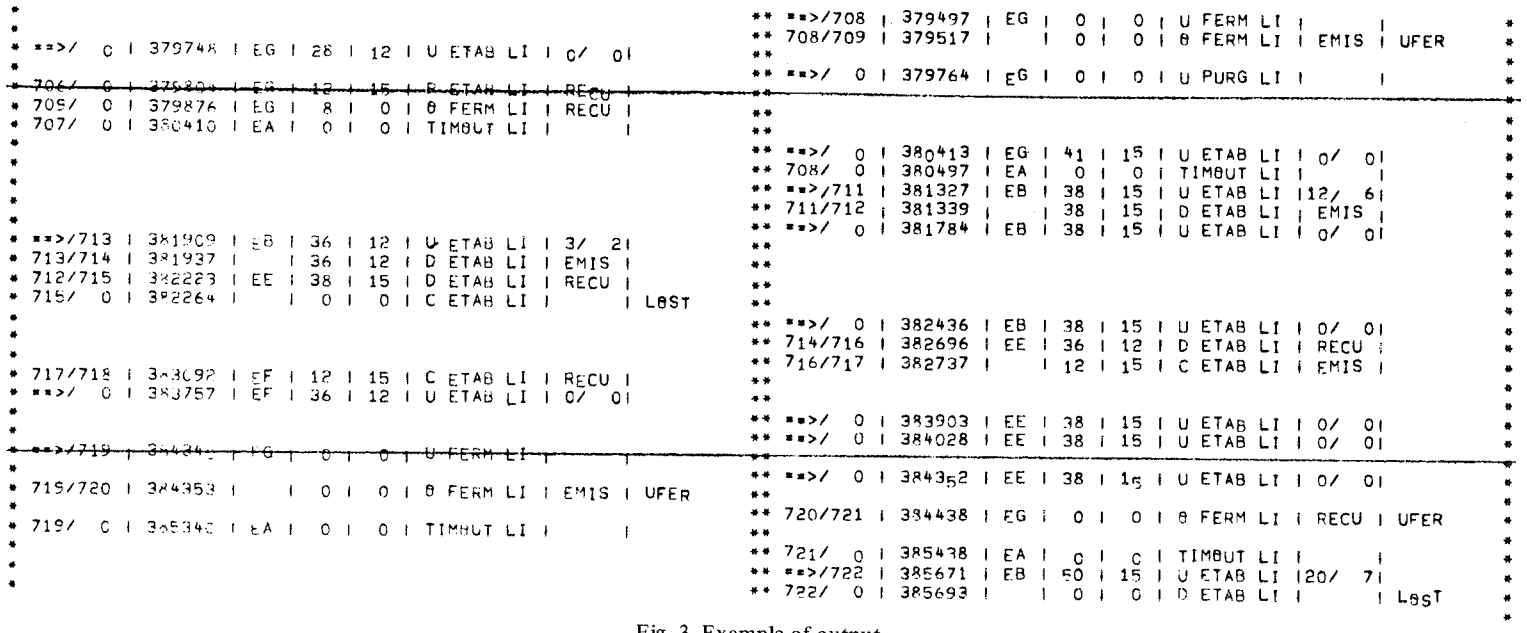


Fig. 3. Example of output.

to close the connection in station A or to open the same connection locally (which should be available).

Indeed, any attempt (U-ETAB-LI) will be rejected after a while because of lack of answer from station A; see events 58, 63, 67, 72. The reason for this is that station A will drop any incoming PURG-message because, when in state EF, the only reset message to be obeyed is a FERM-type message (events 62, 66, 71, 75). D-ETAB-LI requests are also discarded because of state EF (events 59, 64, 68, 73). Meanwhile, nothing prohibits station A to keep sending and retransmitting letters to station B.

Example 2. A confirmation message (code 715) is lost on the subnet (fig. 3). Station B (right half) will return to state EA whereas station A (left half) switches to state EF and starts transmitting letters. Requiring all service messages like C-ETAB-LI to be acknowledged does not solve the problem because acknowledgments may be lost. For example, let us assume that the C-ETAB-Li message (code 715) is not lost. Station B moves to state EF while acknowledging this message. Assuming now that the ack is lost, then the reverse situation holds: station A will move from state EE to state EG and then EA. Note: at time 384 340, a local user request to close the connection has been generated but this is fortuitous.

Even simpler protocols suffer desynchronizations. The current Cyclades protocol [10] has been simulated, mainly for the purpose of performance evaluation (see Part II). This protocol is based upon a Demand + Response scheme. Unfortunately, simplicity does not prohibit undesirable situations to occur as shown in figs. 4a and b.

For local reasons (malfunctioning, user mistake, user decision, ...) a CLOSE operation is initiated from station A shortly after the OPEN operation. If the messages are received reversed and if the OPEN is granted in station B, a symmetrical OPEN message will be sent to station A. When such a message occurs during the "closing state" phase, it is rejected, thus leading to a situation where the liaison is closed for A and open for B.

If acknowledgment of messages is to be used, then one may suggest that non-acknowledged messages should be retransmitted several times in order to avoid such desynchronizations. Actually, it is impossible to devise a mechanism, based on timers and retransmissions, which is absolutely reliable.

2.2. Theorem on interprocess synchronization

Let us analyse the following problem: process A can set up a communication link with process B by sending a *message* which has to be acknowledged by process B. A timer (initial value TM) is used by process A and m successive message transmissions are allowed. Upon receiving the message, process B starts transmitting *letters* to process A. A timer (initial value TL) is used by process B and l successive transmissions of a letter are allowed.

Once process B has initiated transmission of letters, two choices are possible:

(1) B does not process repeated messages which are sent by A (A keeps on sending as long as a message acknowledgment has not been received); in that case, desynchronization is obvious.

(2) B sends back one acknowledgment per message received from A.

Solution (2) is assumed to be chosen. Furthermore, the link is open in A upon receiving a message acknowledgment; letters reaching A before the link is open are rejected.

Let us consider now two different hypotheses:

- Transmission delays *between processes* are variable, finite, but no upper limit is known a priori (H1);
- transmission delays *between processes* are variable and may be infinite. For finite delays, an upper limit is known a priori. Let D be the value of this limit (H2).

Theorem. *Under H1, it is impossible to find a priori a 4-tuplet (m, TM, l, TL) such that synchronization is always achieved. Under H2, the highest probabilities of synchronization completion are achieved when* $TM \geqslant 2D$ *and* $lTL \geqslant (m-1)\,TM + 3D$.

Proof of the theorem can be found in Appendix A. It should be observed that delays considered here are not pure physical transmission delays but interprocess transmission delays.

Conclusion. Simulation showed that it is impossible to devise a mechanism, based on timers and retransmission, which is absolutely reliable when opening or closing a link. It also indicated that only one "reset" command should be used in order to avoid desynchronizations as described in example 1. This was adopted for the current Cyclades transport protocol.

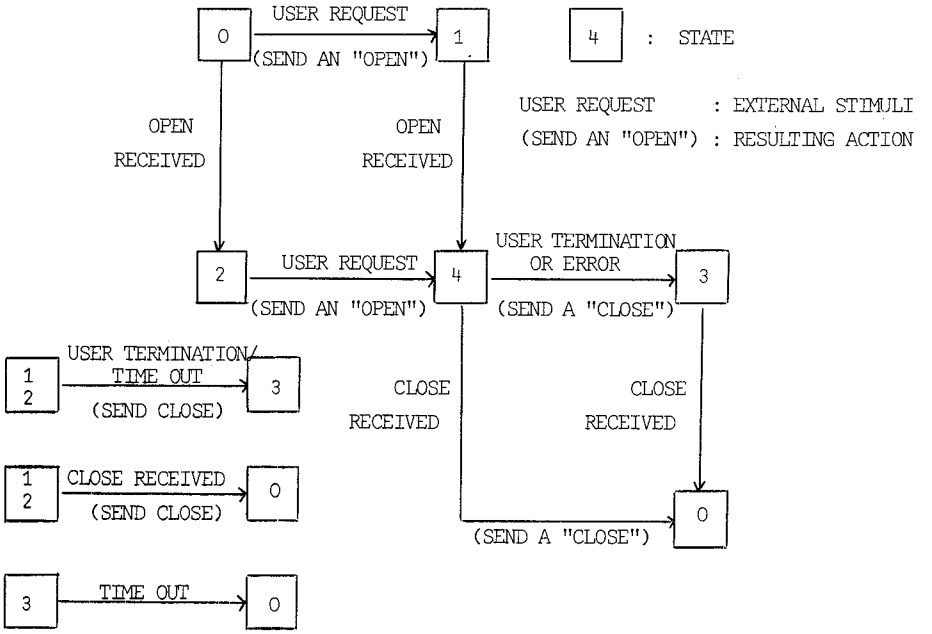


Fig. 4a. Undesirable situation.

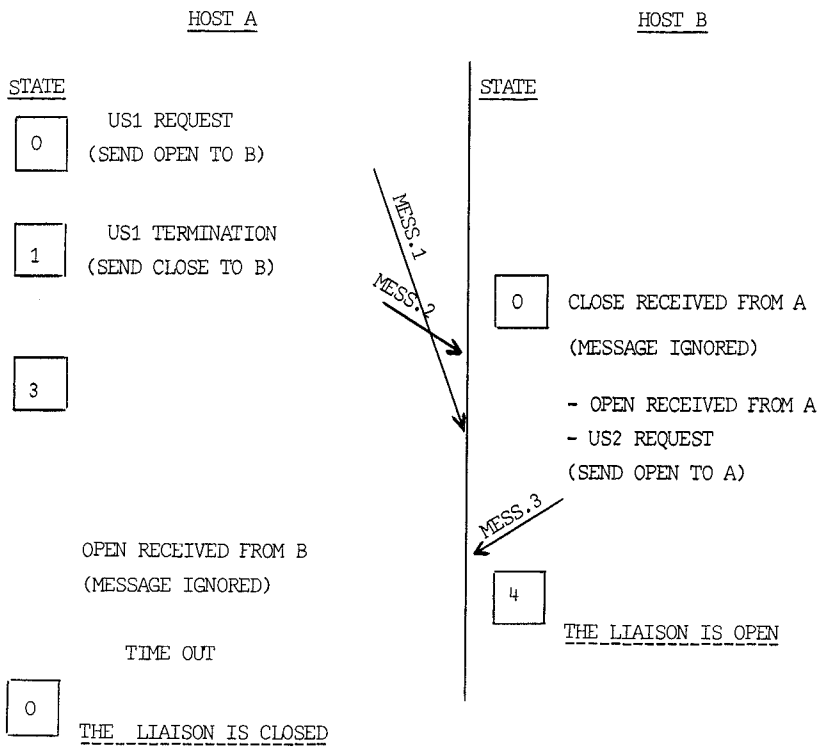


Fig. 4b. Undesirable situation.

2.3. Flow control

The flow control scheme on connections has been described in 1. Simulation has put to evidence several cases of desynchronization.

2.3.1. Desynchronization on connections

a) If error control is off, it is then impossible to know whether or not the requested amount of CR has been received by the receiver. This applies to acks also as they are not error-controlled.

b) If error control is on, then over-CR-allocation is possible when LT' acks are lost. Indeed, in that case, the LT + CR copy is sent again to the receiver.

Symmetrically, allocated CR-values are transmitted in the reverse direction, either within user LT's or within acks.

c) If error control is off, it is then possible that CR-allocation may be lost which is over-allocation again. This applies to acks also.

d) If error control is on, then artificial extra-allocation is possible at the sender's when LT' acks are lost because in that case, the LT + CR copy is sent again to the sender.

The origin of the problem with this mechanism was the lack of control on references given to letters by the user. An example of CR over-allocation is shown in fig. 5. $A(d) - R(d)$ is plotted as curve 1, with $A(d)$ = accumulated amount of buffers allocated in distant station (d is the receiver), and $R(d)$ = accumulated amount of LT's received in distant station. $A(l) - S(l)$ is plotted as curve 2, with $A(l)$ = accumulated amount of allocation CR received in local station (l is the sender), and $S(l)$ = accumulated amount of LT's sent to distant station. Obviously:

$$A(d) - R(d) \geqslant A(l) - S(l).$$

For this particular example, $A(d) = 22$, $R(d) = 12$, $A(l) = 19$, $S(l) = 19$. A total of 10 buffers (10×255 octets) are over-allocated at the receiver's and will never be used.

Similar desynchronizations are known to be possible with the Allocate mechanism on the Arpanet connections.

3. Conclusion

These results were obtained before connection or VC implementation had begun and contributed to the decision of revamping the transport protocol. A description of the current Cyclades transport protocol is given in [10].

Obviously, the lack of a precisely defined reference system or using time references lead definitely to unrecoverable desynchronizations. The other reference which can be used is space and this is what has been adopted for VC's. A common measuring unit is agreed upon by the two stations, thus allowing error and flow control to be achieved by referring to a specific location in the data stream. VC's were the first transport mechanism to include the so called "window scheme". Simulations showed us that such a scheme was highly reliable.

The demonstration that it is impossible to design a mechanism which is absolutely reliable in an uncertain environment is neither surprising nor alarming. End-to-end communication in a computer network is just a particular example of an operating function in so-called multireference systems. A multireference system is a system where processing units do not have the same time reference and possibly do not share the same physical space. Furthermore, transmission delays between processing units are variable and are not known with absolute accuracy. Such systems can be shown to have specific properties [7], one of them being the non-determinacy property stating that no place exists in the system where it is possible to know the internal state value of a unit and the value of the time when that state value was true. It is only possible to know exactly one of these two variables and to associate possible values weighted with probabilities for the other one.

Conventional techniques cannot be used to control such systems. Oppositely, new distributed control schemes may be proposed to allow for operation of self-organizing multireference systems.

For the specific case of transport protocols, absolute safety is possible for flow control assuming that uncertainty is reduced at the subnetwork level: if a limit on lifetime of packets exists, then the window mechanism, which uses a cyclic numbering scheme for letters, is absolutely reliable. Although desynchronizations have been shown to be possible when opening a communication link, hierarchical control provides for an improved reliability at higher levels in the system, the highest one being the user level. Abnormal silence of a Transport Station may be noticed or systematically controlled by one of the higher system levels and clearing of the unsafe situation is then feasible. Theoretically, desynchronization at higher levels is still possible but, assuming that additionnal

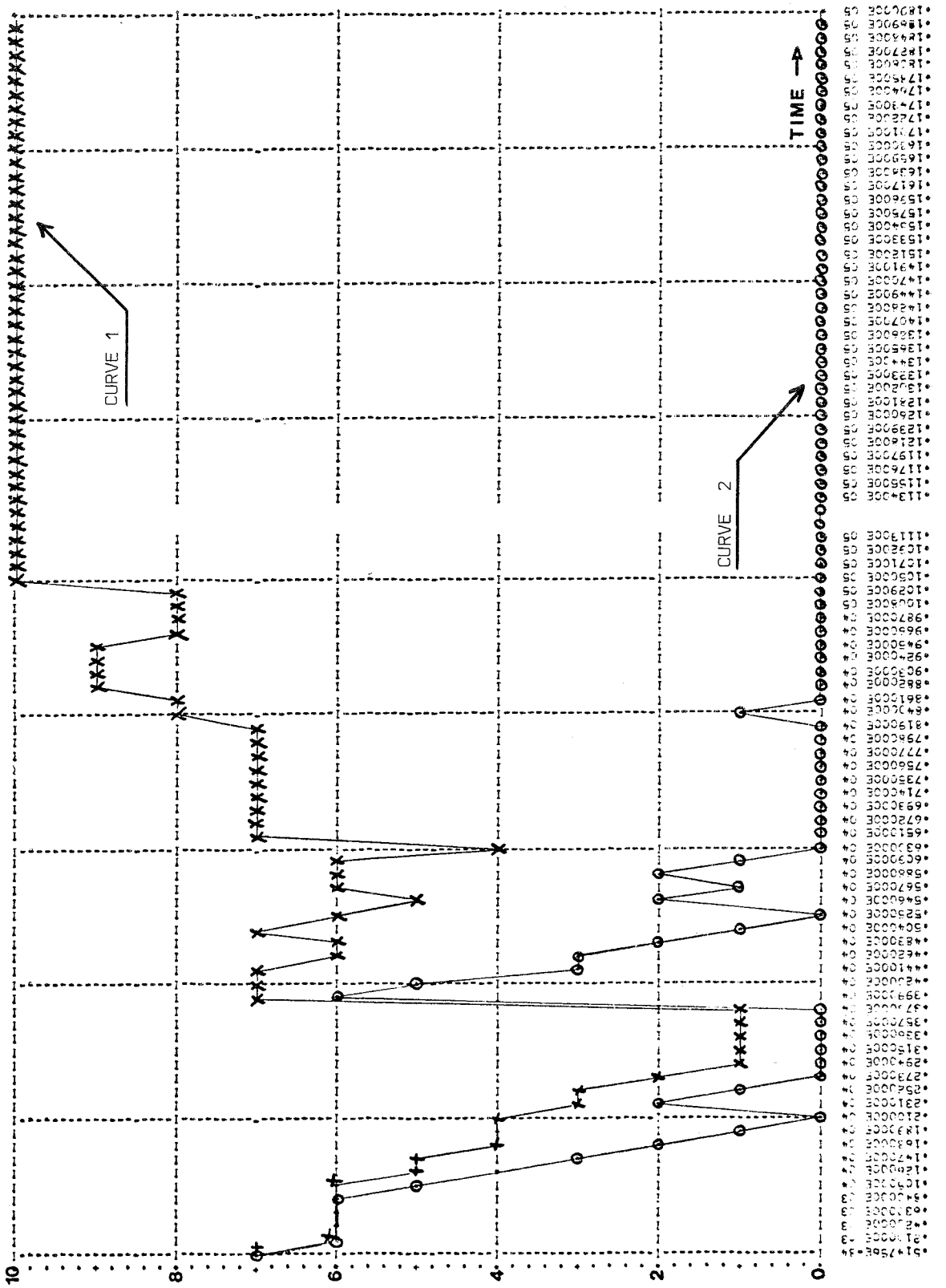


Fig. 5. Example of CR over-allocation.

error control is performed at these lvels, the probability of a desynchronization to occur is so small that absolute reliability is practically achieved.

Part II. Heuristic techniques in performance evaluation of communication protocols

1. Introduction

As well as any system, a computer network may be viewed logically as a layered architecture including a number of layers. Four layers will be considered here and they are shown in fig. 6.

A Text is the logical piece of data as it is handled at the process level. Exchange of Texts between a sender and a receiver is considered as a virtual communication being monitored by an Interprocess Protocol. The actual physical transmission is performed through a cascade of several independent real communications, each of which being controlled by an External Protocol, sometimes called an Interface. The same scheme applies to the other layers in the hierarchy.

A physical realization of a layer is called a station. A station (except for the switching-station) has three interfaces: two real interfaces with the adjacent layers and a virtual one with the station it is communicating with.

External protocol between levels i and $i+1$ includes specifications of: commands and formats of data transmitted from $(i+1)$ to (i), status reports and data formats delivered from (i) to $(i+1)$, error control performed by (i), and congestion control performed by (i).

Internal protocol between two stations of a given layer specifies: orders and formats of data exchanged between the stations, error control on the virtual communication path, and flow control on the virtual communication path.

We will now focus on the problems of flow control as they arise at the transport level and at the switching level.

2. Results

2.1. Flow control for internode protocols

At the switching level, an important goal to achieve is good utilization of the offered transmission facility. Most of network designers have agreed upon a packet size ranging around 2000 bits. The fixed overhead paid per packet transmission seems acceptable for such a size with the current technology.

Internode traffic may be monitored by using a general line control procedure. However, as we deal with computer networks, it is possible to go into more details as far as the traffic is concerned. For example, packets handled at the switching level can be classified into three categories: short packets (0–15 octets), corresponding to service packets and interactive traffic (C-packets), long packets (128–255 octets) corresponding to file transfers (A-packets) and middle-size packets which are neither short nor long (B-packets). Reduced transmission delays are required for the short packets. This can be achieved by the means of a priority scheme working on the queues of packets waiting for transmission; C-packets are

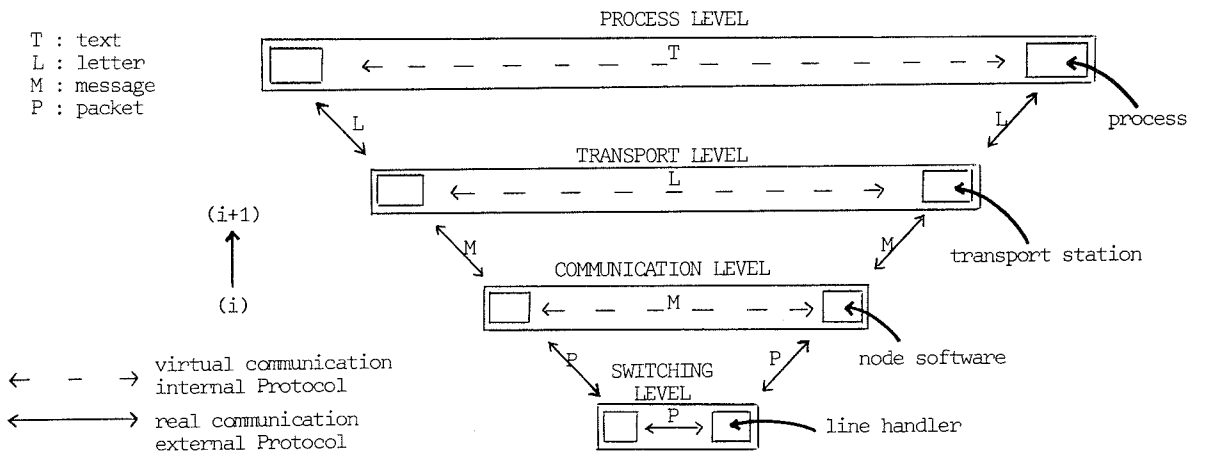


Fig. 6. Four-layered network architecture model.

selected first. Then, the problem of sharing the bandwidth between categories A and B has to be solved.

With the MV8 Protocol Implemented in Cyclades, a transmission line is considered as made up of 8 virtual channels. The alternate scheme on each channel is similar to the one used in the Arpa subnetwork. Sending a packet is allowed only on free channels. Each packet header carries an acknowledgment octet. A channel is freed upon receiving an acknowledgment for the packet sent on that channel. The MV8 channel allocation scheme is the following one: channel 0 is reserved for C-packets; C-packets are sent prior to any other packet on all channels 1 through 7. Channels 1, 3, 5 and 7 are B-reserved, and channels 2, 4 and 6 are A-reserved. This allocation scheme is intended to provide a fair sharing of the bandwidth between categories A and B.

It should not be forgotten that the target is to design a good internode protocol to handle heavy traffic. Naturally, in the case of light traffic, there will be no contention to access the line and the channel management technique will have no influence on the performances. For heavy traffic, the problem is that no assumption can be made on the nature of that traffic. What percentages of A-packets and B-packets have to be transmitted? Furthermore, that "traffic cocktail" is dynamically evolving. Then the need for a dynamic channel allocation scheme, that allocation being permanently updated according to the "traffic cocktail" itself.

The MV8/Channel Stealing Protocol includes such a mechanism: the same preallocation scheme as in MV8 is used but free channels can be stolen from one category for another one if that latter category is predominating. Stolen channels are immediately returned to their "owner" when a matching packet has to be transmitted. Therefore, the MV8/CS Protocol should provide a better throughput than MV8 for any "traffic cocktail".

Simulations have been performed in order to get a better insight into the behaviour of these Protocols. A third one has been evaluated too, which is similar to HDLC and which handles packets on a FIFO basis, sending any packet on any channel (no preallocation).

A unique traffic generator has been used for the various simulations and the internode traffic is assumed to be symmetrical. A triplet $\langle c, b, a \rangle$ is used to represent the traffic cocktail (respective ratios of C-packets, B-packets and A-packets). A 48 Kbits/s line has been simulated. Packet headers are 12 octets long and the transmission overhead is 12 octets per packet.

Results are shown in tables 1 and 2. The average delay is given for one complete hop. Waiting time indicates the time elapsed from the routing decision until the actual transmission on the line. Buffer occupancy indicates the time needed for a busy buffer to be freed. The overhead includes packet headers, transmission octets and acknowledgment packets. The number U indicates what percentage of the available bandwidth is provided to the user traffic.

2.1.1. Results

For $U < 0.5$, the three procedures have equal performances. Above that ratio, priority rules exhibit some effect.

Compared to FIFO, MV8 and MV8/CS provide a constant transmission delay for C-packets. For $U = 0.6$, the gain ranges between 0 and 5 ms according to the traffic pattern. One interesting case is triplet 181: MV8 induces some kind of a node saturation by B-packets. This is due to the artificial limitation of up to 4 simultaneous B-channels in use. With MV8/CS, that saturation disappears. The price to pay is 0.9 ms as extra-delay for A-packets transmission. This has to be compared to the fantastic reduction of buffer occupancy delay (94.9 ms instead of 194 ms). This saturation case has been reproduced with A-packets by using triplet 111 and a 33.5 Kbits/s user traffic ($U = 0.7$). MV8/CS equalizes the line saturation effect between the three categories, thus delaying node saturation: 3 buffers instead of 4 are occupied at the end of the simulation run and savings on buffer occupancy for A-packets are noticeable (67 ms). C-packets experience a 0.7 ms extra-delay with MV8/CS. No other benefit can be expected from the protocol itself because in that case, the line reaches saturation.

Considering the results, it may seem attractive to recommand a simple protocol for internode traffic handling. Such a protocol could deal with two packet categories only, each of which being served FIFO: the C-packets category and the non-C-packets category, the first one being scanned first. Obviously, this would perform efficiently as far as the subnetwork itself is concerned. But, as we know, consequences of such a protocol for the higher levels in the hierarchy must be evaluated. In particular, it may be necessary to provide the transport network level with the same priority scheme because short Letters or ack Letters are very important for host resources utilization. Such Letters may be classified as B-packets at the

Table 1

Triplet	111						181								
Category	C	B	A	C	B	A	C	B	A	C	B	A	C	B	A
Interarrival time (MS)	75	75	75	75	75	75	214	27	214	214	27	214	214	27	214
Average delay (MS)	44.7	52.7	63.7	40	53.8	64.1	42.5	52.4	68.8	37.7	147	69.7	37.3	52.2	70.6
Waiting time (MS)	18.1	15.3	6.58	13.4	16.5	6.98	15.9	15.1	12.1	11.1	110	13.1	10.71	14.9	14
Buffer occupancy (MS)	89.9	96.8	107	83.7	97.3	107	88.1	95.4	115	89.1	194	120	82.2	94.9	116
User throughput (Kb/s)	0.84	7.66	20.3	0.84	7.66	20.3	0.38	21.4	7.02	0.37	21.2	7.03	0.38	21.4	7.02
Length of files (end of simulation)		0		0	0	0		0		0	9	0	0	0	0
Total user throughput (Kb/s)		28.8			28.8			28.8			28.6			28.8	
Overhead (Kb/s)		9.78			9.83			11.3			9.82			11.3	
Extra bandwidth available (Kb/s)		9.42			9.37			7.9			9.58			7.9	
$U = 0.6$		FIFO			MV8-MV8/CS			FIFO			MV8			MV8/CS	

Table 2

Triplet	111						118					
Category	C	B	A	C	B	A	C	B	A	C	B	A
Interarrival time (MS)	64	64	64	64	64	64	384	384	48	384	384	48
Average delay (MS)	43.3	100	243	44	106	178	43.7	54.1	68.9	44.4	56.9	62.1
Waiting time (MS)	16.7	63.1	186	17.4	68.9	121	17.1	16.9	11.73	17.8	19.8	4.99
Buffer occupancy (MS)	95.2	152	294	95.4	157	229	97.7	116	121	89.9	114	110
User throughput (Kb/s)	0.97	8.93	23.6	0.87	8.93	23.7	0.23	1.47	31.8	0.13	1.47	31.09
Length of files (end of simulation)	0	0	4	1	1	1		0			0	
Total user throughput (Kb/s)		33.5			33.5			33.5			33.5	
Overhead (Kb/s)		9.73			9.77			6.63			7.04	
Extra bandwidth available (Kb/s)		4.77			4.73			7.87			7.46	
$U = 0.7$		MV8			MV8/CS			MV8			MV8/CS	

subnetwork level and what is then really needed is not a FIFO scheme mixing A and B-packets but an MV8/CS like scheme.

On the other hand, it should be noticed that MV8 and MV8/CS protocols introduce more packet disturbance (desequencing) than FIFO-like protocols. If sequenced delivery is required at the user level, then the tradeoff is between less efficient utilization of buffer space at the receiving TS and higher potential switching capacity (or the reverse).

2.2. *Flow control for transport protocols*

Two INWG proposals have been evaluated by simulation. These protocols are the Cyclades protocol (CY) [10], and TCP [1]. They both use a positive acknowledgment + retransmission scheme and the window technique: credits are transmitted to the sender to control the flow and these credits are incremental values relating to the last acknowledged data block. The main differences are:

Error control. When fragmentation is on (Letters not fitting into one packet), any acknowledgment scheme is acceptable with TCP. However, buffer utilization is better when each fragment is promptly acknowledged and this is what has been assumed for our simulations. With CY protocol, only one acknowledgment per whole Letter is required.

Flow control. Credit values are indicated in octets with TCP. The CY protocol requires an agreement to be reached by the processes on a common Letter size at the communication set-up time. Then, the credit values are indicated in Letters.

Evaluation results (figs. 7 and 8). Identical environments have been simulated for the CY protocol and TCP. Letter lengths of 120 octets and 480 octets have been chosen to make a comparison when fragmentation is on and when it is not. The virtual full-duplex link used by the communicating processes is symmetrical, i.e., processes behave identically. Fragment headers are 8 octets long for CY and 10 octets long for TCP. With regard to the subnetwork, an accurate simulation was felt necessary. A three hop path has been chosen with links operating at 19.2 Kbits/s. The packet header is 12 octets long and line control procedures introduce at 13 octet overhead per text packet and a 6 octet overhead per ack packet. The maximum text capacity for a packet is 255 octets.

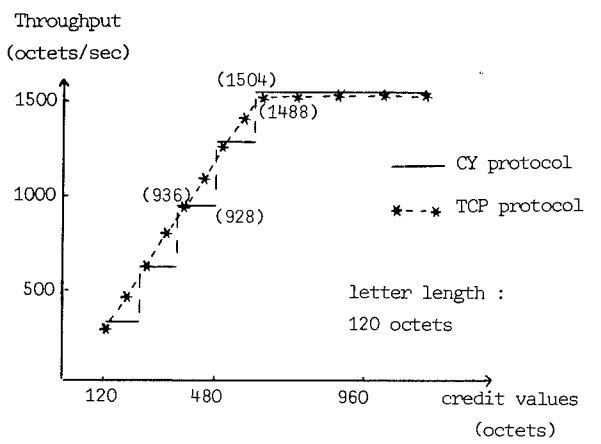


Fig. 7. Evaluation results.

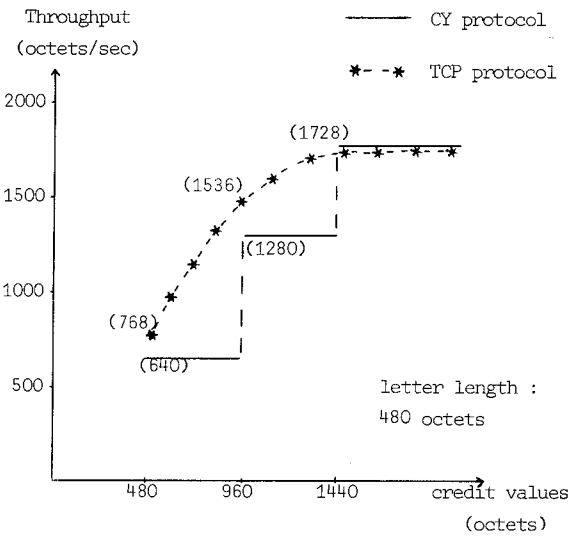


Fig. 8. Evaluation results.

For medium-size letters (120 octets), both protocols have nearly identical performances. TCP achieves a better throughput than CY when credit values made available by the receiver are small and not exact multiple of the letter length. This is due to the octet based flow control policy. For all other cases, CY allows for a slightly increased throughput because of a smaller overhead incurred per packet.

For long letters (480 octets), throughput with TCP is higher than with CY as long as credit values are kept small. This is due to the acknowledgment per fragment policy. Otherwise, CY takes advantage again of a smaller overhead per packet.

Simulation results for transmission delays indicate quite similar performances for the CY protocol and TCP.

2.2.1. Remarks

What has been compared is the highest achievable throughput when using different flow and error control strategies. No assumption has been made about the semantics associated with data streams (TCP) and fragments or letters (CY).

Also, more sophisticated buffer management software achieves a better throughput. This is true up to the point where buffer space is not a scarce resource any more. Something which cannot be evaluated through simulation is software complexity.

Current trends indicate that transmission costs are likely to decrease much slower than memory costs and that software costs keep increasing. Then, for achieving a given throughput, the real tradeoff is between more memory space and more sophisticated software. This is a choice to be left to system designers.

2.3. Performances as seen by transport network users

It may be questioned how costly it is to transmit a short letter (a few octets) into one packet. This is the case for many applications, like interactive ones. This mode of operation induces probably a high overhead, thus reducing the available bandwidth for the user. It has been usually advocated that relative wasting of the useful throughput is the price to pay in order to achieve fast transmission, which is precisely the important criterion in interactive applications. Here, the confusion comes from the fact that it is attempted to perform a unique optimization for two heterogeneous parameters. If it is right to say that a short transmission delay is important to the user then, the right question to ask is how much of the throughput is wasted *at the user level* too and not at the switching network level.

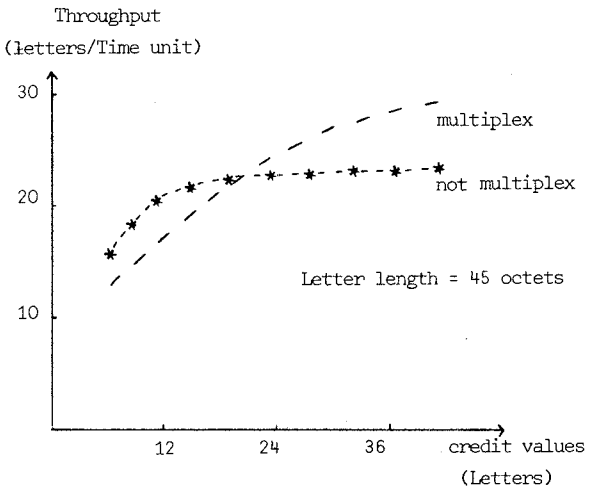


Fig. 9. Results of simulation study.

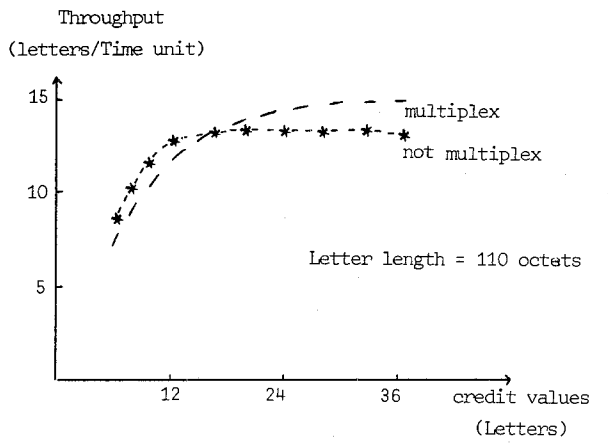


Fig. 10. Results of simulation study.

User letters are handled by Transport Stations which are in charge of both error control and flow control. Then, the maximum throughput one can hope to achieve at the user level is the one allowed by the TS flow control mechanism.

Results of a simulation study are shown in figs. 9 and 10. The CY protocol was chosen for that simulation and the same model as the one described earlier applies here. Multiplexing several interactive letters into packets has been tried against the one letter per packet case. In order to make fair comparisons, the amount of short letters waiting for transmission is always high enough to allow for the maximum multiplexing ratio.

2.2.2. Results

The results indicate very clearly that the highest throughput for the user is reached when not multiplexing short letters, for the most realistic credit values at the receiving TS. Differences may rank 25%. This is due to the fact that multiplexing letters into packets leads to longer packets which are much slower than the one letter packets to travel across the various network layers. For the same reason, acknowledgments being transmitted with the reverse traffic are slower to come back. New credit values are carried with these acknowledgments. It may happen then that the sending TS having spent all its credits has to wait for a future acknowledgment carrying a non-zero credit value before resuming the letter trans-

mission. This leads to a non-optimal throughput.

Multiplexing is efficient only if a large number of credits is made available at the receiving station. As stations are supposed to handle many conversations simultaneously, such required values do not seem realistic. Thus, in standard cases, a better delay and a better throughput will be achieved at the user level by not multiplexing short letters into packets.

The threshold value is a function of the subnetwork load. It can be seen in fig. 11 that multiplexing is more likely to be efficient as subnetwork traffic increases. The reason for this is that contention delays inside nodes are longer when the traffic increases and, for a given amount of letters, multiplexing reduces the number of packets to be transmitted compared to non-multiplexing.

When multiplexing is not used because it provides the user with better performance, some useful bandwidth is wasted at the switching level. This is a typical example of a conflicting optimization case in hierarchical systems.

2.4. Influence of priority routing for acknowledgment packets

Considering that round-trip time is the limiting factor for the user throughput when multiplexing is on, it may seem attractive to send back acknowledgments (and credit values) on special service packets only, provided that short packets are routed prior to any other packets inside the subnetwork. Internode protocols using such schemes have been analysed in Part II, section 2.1.

Three categories have been simulated:

(i) Conventional strategy: ack packets are issued by a TS only if it is not possible to send a user letter (credit value is null). Priority is given to ack packets inside switching nodes.

(ii) For each letter received by a TS, an ack packet is generated. Priority is given to ack packets inside nodes and at TS level.

(iii) Only one ack packet is generated for all letters multiplexed within one packet. Again, priority is

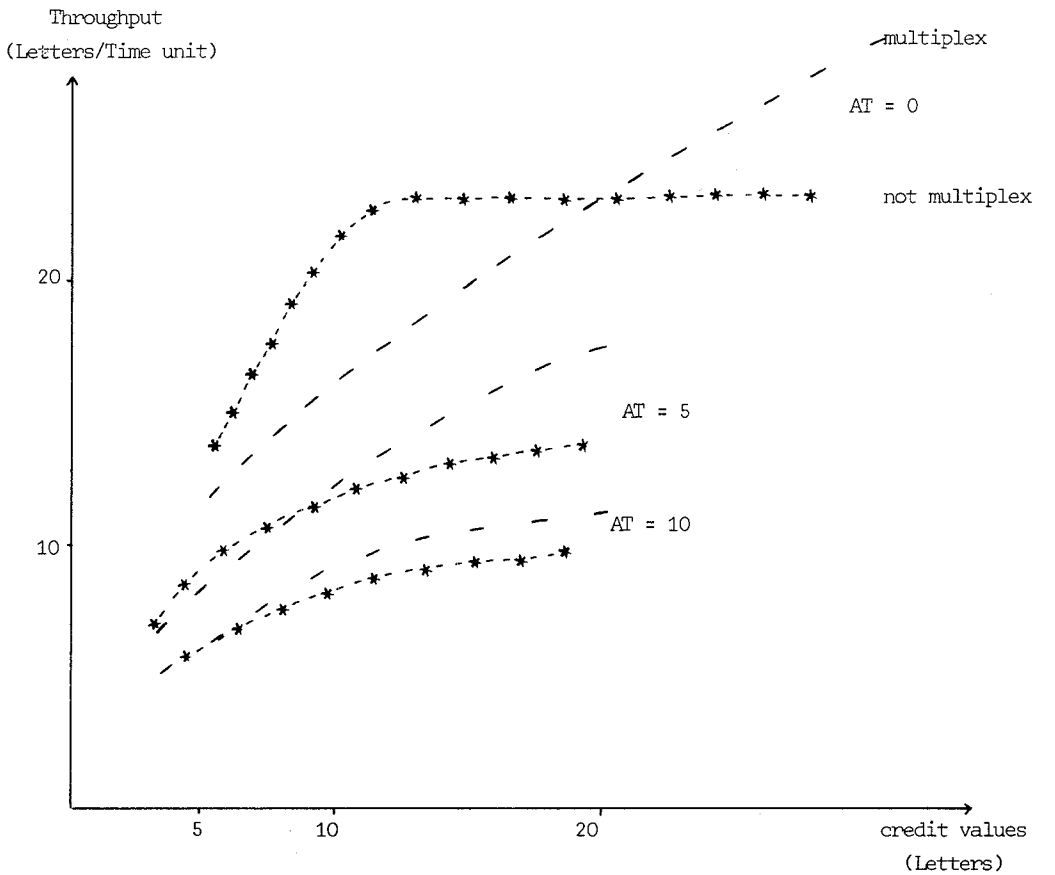


Fig. 11. Threshold value as a function of the subnetwork load. AT = artificial outgoing traffic per node (packets/time unit).

given to ack packets inside nodes and at TS level.

Acknowledgment and credit values are stored into the "text" field of packets. It is then necessary to guarantee that the text length is less than 16 octets if such packets are to be processed as short ones. For this simulation, the average letter length is 45 octets.

A reference model is used for the purpose of comparing performances in identical environments and which is similar to strategy (i) except that short priority routing is not exercised. TS user traffic is symmetrical and is always high enough to achieve the maximum throughput. Interfering traffic has been simulated within the subnet; the artificial output load for each node and for every line along the network path amounts to 2 packets/sec.

2.4.1. Results

Quantitative results are given in table 3.

Multiplexing option. Throughput achieved with strategies (i) and (iii) is higher than with the reference model (see comment 1) whereas strategy (ii) is less efficient than this model (see comment 2).

Non-multiplexing option. Identical comparison between strategies (i), (ii) and the reference model can be made.

Performances (throughput, transmission overhead) allowed with non-multiplexing option for (i) and (ii) are better than when using multiplexing option in the reference model when credit values are small enough (see comment 1). When credit values get large enough, multiplexing is more efficient as explained in section 2.3.

Strategies (i) and (iii) are generally more efficient than strategy (ii) as credit values increase. The reason is that conflicting cases occur more often with (ii) because the number of ack packets is always higher than with strategies (i) and (iii) (see comment 2 also).

Comment 1. Most of returned credits are sent within ack packets. When priority routing is exerciced on these packets, the resulting throughput is improved (as expected).

Comment 2. TS behaviour is symmetrical. Priority is given to ack packets in a TS to optimize useful incoming traffic but this is bought at the expense of reducing the useful outgoing traffic. Resulting loss in throughput takes over potential improvement with strategy (ii) for two reasons: user letters experience contention delays not only in TS' but also within nodes and contention probability is highest with (ii).

In any case, differences in performances are of the same order of magnitude as the accuracy of the simulation model. Furthermore, smaller differences correspond to smaller credit values which are realistic situations. Thus, the conclusion must be that using the special ack packets + priority routing mechanism to return credits brings practically no advantage to the user.

2.4.2. Conclusion

Computer networks are operated through a hierarchy of communication protocols, each of which being designed to achieve a specific optimization. It has been shown how useful heuristic techniques can be to evaluate performances of such protocols. Absolute and comparative results have been given. Another important result has been to put to evidence the fact that global optimization of the hierarchy is hard to achieve; not only conflicting cases have been reported

Table 3

Credit	Reference model		Strategy (i)		Strategy (ii)		Strategy (iii)
	[1]	[2]	[1]	[2]	[1]	[2]	[2]
5	142 120	138 127	149 126	148 135	145 (19) 143	143 (7) 136	142 (8) 118
8	168 135	169 134	182 120	179 135	169 (47) 163	174 (20) 168	189 (27) 135
11	193 126	198 125	209 123	207 136	185 (59) 177	200 (32) 165	210 (24 109
20	218 113	257 122	231 104	260 136	196 (82) 183	233 (40) 170	270 (40) 116

$\frac{x}{y}$ (z) x = total number of letters received by station 2, y = total number of ack packets received by station 1, (z) = total number of conflicts in station 1 (priority given to acks), [1] = non multiplexing, [2] = multiplexing.

but data has been given indicating what choice to make or what the tradeoffs are.

This paper is a contribution to the general effort needed to build efficient computer networks. Global optimization is still an open-ended way.

Acknowledgments

Special thanks are due to R. Pedrono who participated in some of the simulation studies. Part of the work reported here has been supported by D.R.M.E., Ministère de la Défense, under contract 74/540.

All simulation programs have been written in Simula 67.

Appendix A. Proof of the theorem on interprocess synchronization

(i) minimum value for TM (fig. 12)

T_0 will be the time of the first message transmission (A → B); T_1 is the time of the first successful message reception in B, T_6 is the corresponding time in A. At T_1, transmission of letters begins. (T'_0, T'_1, T_2) is the triplet of the times corresponding to the first completely successful message transmission (message and ack are error-free and are not lost). TM must be large enough for allowing reception of a message-ack in A before reaching the limit of m attempts. Let us write:

$$T'_0 - T_0 = x\mathrm{TM}\,, \quad 0 \leqslant x \leqslant m-1\,, \quad x \text{ integer}\,,$$

$$T_2 - T'_0 = d_1 + d_2\,.$$

Constraint:

$$T_2 - T_0 \leqslant m\mathrm{TM}\,,$$

or

$$\mathrm{TM} \geqslant (d_1 + d_2)/(m - x)\,. \tag{1}$$

(ii) minimum value for TL (fig. 13)

TL must be large enough so that A has open the link (T_2), one copy of letter No. 1 reaches A and a letter-ack is received in B before reaching the limit of l attempts.

Let (T_3, T_4, T_5) be the triplet of the times corresponding to the first completely successful letter reception ($T_4 \geqslant T_2$; letter and ack are not lost and are error-free). Let us write:

$$T_3 - T_1 = y\mathrm{TL}\,, \text{ with } 0 \leqslant y \leqslant l-1\,, \quad y \text{ integer}\,,$$

$$T_5 - T_3 = d_3 + d_4\,.$$

Constraint:

$$T_5 - T_1 \leqslant l\mathrm{TL}$$

or

$$(l - y) \geqslant d_3 + d_4 \tag{2}$$

Eq. (2) gives

$$y\mathrm{TL} \leqslant l\mathrm{TL} - (d_3 + d_4)\,. \tag{2bis}$$

Eqs. (2bis) and (3) are compatible if and only if:

$$T_2 - T_1 - d_3 \leqslant l\mathrm{TL} - (d_3 + d_4)\,.$$

$$T_1 = T_6 + d_5\,.$$

Then the constraint

$$l\mathrm{TL} \geqslant T_2 - T_6 + d_4 - d_5\,. \tag{4}$$

(iii) maximum value for TM (fig. 14)

TM must be small enough to provide for reception of a new message in B in the case of no reception of ack in A and this should occur before the limit of l attempts is reached in B (otherwise, it is useless to provide A with a timer + retransmission mechanism).

T_6 and T_1 have previously been defined. T_7 is the time of the second successful message reception in B and T_8 is the corresponding time in A. We have

$$T_0 \leqslant T_6 \leqslant T_8 \leqslant T'_0\,.$$

Let us write:

$$T_7 = T_6 + w\mathrm{TM} + d_6\,, \quad \text{with } 0 < w \leqslant x\,.$$

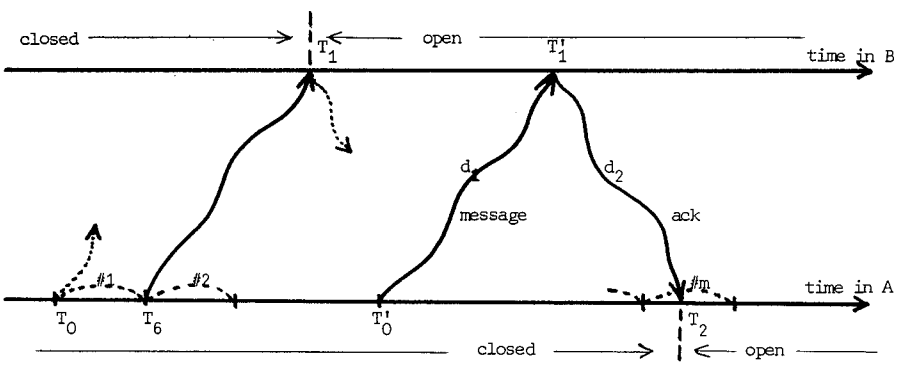


Fig. 12. Minimum value for TM.

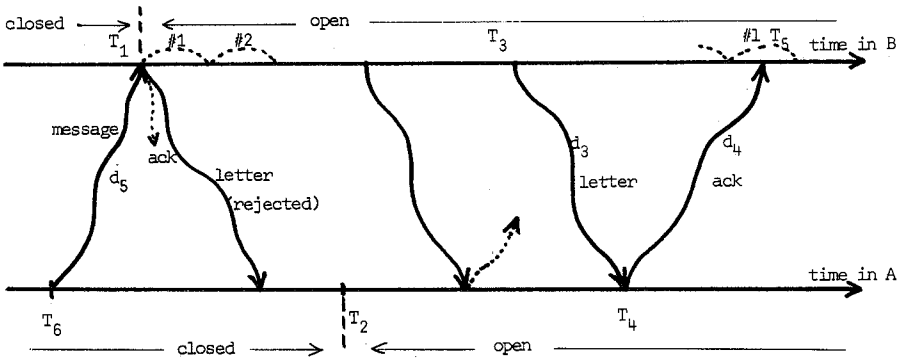


Fig. 13. Minimum value for TL.

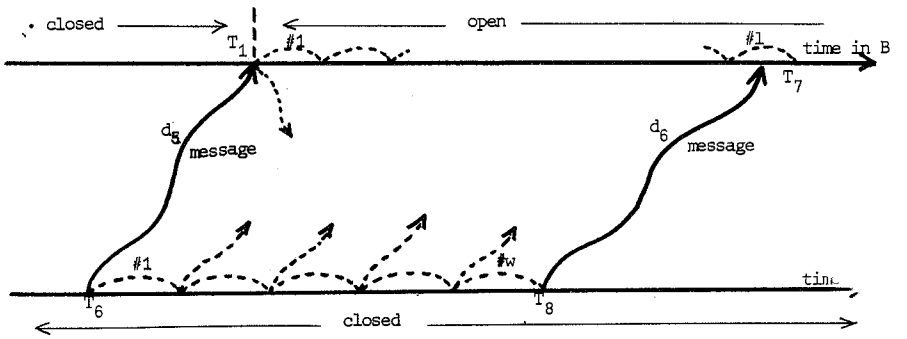


Fig. 14. Maximum value for TM.

Constraint:

$T_7 - T_1 \leqslant l\mathrm{TL}$

or

$$\mathrm{TM} \leqslant \frac{l\mathrm{TL} + d_5 - d_6}{w} . \qquad (5)$$

Are constraints (1) and (5) always compatible? According to these inequalities:

$1 \leqslant m - x \leqslant m$ and $0 < w \leqslant x \leqslant m - 1$

compatibility is always possible if $(m \neq 1)$:

$$l\mathrm{TL} \geqslant (m - 1)(d_1 + d_2) - d_5 + d_6 . \qquad (6)$$

On the other hand, constraint (2) is always satisfied provided that:

$l\mathrm{TL} \geqslant \max(T_2 - T_6) + d_4 - d_5$,

$\max T_2 = T_0 + (m - 1)\,\mathrm{TM} + d_1 + d_2$,

$\min T_6 = T_0$.

Thus:

$$l\mathrm{TL} \geqslant (m - 1)\,\mathrm{TM} + d_1 + d_2 + d_4 - d_5 . \qquad (4')$$

Provided that lTL satisfies the strongest of the constraints (6) and (4') and TM satisfies constraints (1) and (5), synchronization is possible. Clearly, if no upper limit is known for transmission delays, it is impossible to prove that, by picking any value a priori for lTL and TM, these constraints will be always satisfied (hypothesis H1).

Under hypothesis H2, it is possible to show that constraint (4') is the strongest one. D being the upper limit, the upper bound value for expression (6) is $(2m - 1)D$. The upper bound value for expression (4') is $(m - 1)\mathrm{TM} + 3D$. It is easy to show that: $(2m - 1)D < (m - 1)\mathrm{TM} + 3D$ holds in any case as it is obvously satisfied when replacing TM by min (TM), which is equal to $2D$ according to (1).

Thus, for such systems, the highest probabilities of synchronization completion are achieved when:

$\mathrm{TM} \geqslant 2D$ and $l\mathrm{TL} \geqslant (m - 1)\mathrm{TM} + 3D$.

Knowing D, it is possible to choose a 4-tuplet $(m, \mathrm{TM}, l, \mathrm{TL})$ whatever the probability of infinite delays occurence.

References

[1] V.G. Cerf and R.E. Kahn, A protocol for packet network intercommunication, IEEE Trans. on Com., vol. COM-22, 5, (May 1974) 637–648.

[2] J.F. Chambon et al., Spécifications fonctionnelles des Stations de Transport du réseau Cyclades, Technical Publication IRIA-SCH 502.3 (May 1973) 97 p.

[3] H. Le Goff, Etude générale et évaluation de protocoles de transport dans les réseaux informatiques, Doctorat de 3è cycle Thesis, University of Rennes, France (April 1976) 173 p.

[4] H. Le Goff and G. Le Lann, Communication and synchronization tools in a distributed environment, First ECI Conference, in Springer-Verlag, ed., Lecture Notes in Computer Science n° 44 (Amsterdam, 1976) 50–61.

[5] G. Le Lann, La Simulation et le projet Cyclades, AFCET National Conference on Computers and Telecommunications (Rennes, France, Nov. 1973) 297–306.

[6] G. Le Lann and H. Le Goff, Advances in performance evaluation of communication protocols, Third ICCC (Toronto, Canada, Aug. 1976) 361–366.

[7] G. Le Lann, Distributed systems-towards a formal approach, IFIP Congress (Toronto, Canada, Aug. 1977) 155–160.

[8] L. Pouzin, Presentation and major design aspects of the Cyclades computer network, Third ACM/IEEE Data Communication Symp. (Tampa, Nov. 1973) 80–87.

[9] C.A. Sunshine, Survey of communication protocol verification techniques, IEEE/NBS Symp. on Computer Networks (Gaithersburg, U.S.A., Nov. 1976) 24–26.

[10] H. Zimmermann, The Cyclades end-to-end protocol, Fourth ACM/IEEE Data Communication Symp. (Québec, Canada, Oct. 1975) 7/21–7/26.