AN IMPROVED ETHERNET FOR REAL-TIME APPLICATIONS

R. Hainich
Hahn-Meitner-Institut fuer Kernforschung Berlin GmbH
Germany

ABSTRACT

In the area of office communications, Ethernet is likely to become a standard because of its flexibility, reliability and ease of use, properties also desirable with real-time applications. Unfortunately, Ethernet does not guarantee an ordered scheduling of tranmissions and has no means for insuring functional security at overload.
We will propose some simple modificatons to the Ethernet protocol, providing for nearly deterministic round-robin message scheduling while maintaining interconnection compatibility with normal Ethernet interfaces.
The improvements are achieved by varying the random delays following collisions, in dependency of the individual message waiting times and the number of active nodes, which is estimated from the number of collisions taking place. Results from computer simulations involving up to 1000 nodes are shown.

ZUSAMMENFASSUNG

Ethernet wird wahrscheinlich ein Standard in der Bürokommunikation werden, wegen seiner Flexibilität, Zuverlässigkeit und einfachen Anwendung, Eigenschaften, die auch in der Prozeßdatenverarbeitung von Vorteil wären. Leider erlaubt Ethernet keinen hinreichend geordneten Nachrichtentransport; ebenso existieren keine Vorkehrungen, um eine Minimalfunktion bei Überlast sicherzustellen.
Wir wollen hier einige Protokollmodifikationen vorstellen, die eine nahezu deterministische Zugriffszuweisung erreichen und trotzdem die volle Kompatibilität zu normalen Ethernet-Interfaces bewahren.
Die Verbesserungen werden erreicht durch Variation der zufälligen Verzögerungszeiten nach Kollisionen, in Abhängigkeit von den individuellen Nachrichten-Wartezeiten und der Anzahl der aktiven Stationen, welche nach der Anzahl der auftretenden Kollisionen geschätzt wird. Wir zeigen Resultate von Computersimulationen mit bis zu 1000 Stationen.

RÈSUMÈ

Pour les applications de bureautique, Ethernet est en passe de devenir un standard grâce à sa flexibilité, sa fiabilité et sa facilité d'emploi, propeétes "egalement recherchées dans les applications temps réel. Malheureusement, Ethernet ne garantit pas d'ordonnancement particulier des transmissions et ne peut se camporta de façon prévisible en cas de surcharge. On propose des modifications simples au protocole d'Ethernet, assurant un ordonnancement quasi-détermininiste des messages tout en maintenant la compatibilité avec les interfaces standard Ethernet. Les améliorations sout obtennes en faisant varier les délais de déférence en fonction des délais d'attente et du nombre de noeuds actifs, estimé d'aprés le nombre de collisions. Des resultats de simulation pour un reseau de 1000 noeuds sent fournis.

INTRODUCTION

Ethernet has a simple, reliable protocol with no need for initialization (which is a weak point with many other networks). The absence of special protocol messages and complicated procedural dependencies guarantees for high throughput and deadlock-free operation.

In terms of office communications, Ethernet is fast. In most cases, its limited message length and high data rate result in short response times.

With real-time applications, however, a secure upper limit of response time (message delay) is needed, especially under overload conditions.

Because it is normal with such applications that several nodes are functionally linked by a common process, there is no way to exclude overloading especially in delicate situations. In this case, waiting times have extremely wide deviation because Ethernet gives all nodes arbitrary chances of getting their messages transmitted, to the point of discriminating messages with longer waiting times. This may cause delays more than ten times longer than necessary.

TRANSPARENCY

We concentrated most of our investigations on overload simulations (see also 'Simulation Model'). This 'worst case' is especially worth of consideration, because there is no way to prevent it, except for extremely oversizing the network capacity.

Overload will always cause long message queues within the nodes; however, if every node can always send its currently most important message within a short, predictable time, this essentially improves safety and simplicity; we call such a net 'transparent' to the upper system layers, meaning that it guarants for a minimum transport ability in every situation.

It should be possible to insure that of n nodes, none would have to wait for significantly more than n-1 messages of others, until it may transmit at least one message of its own. Our suggested improvements are especially aimed at this goal.

We will not consider the internal message queueing within the nodes; we assume that they promote their most important ones. We also do not use information that is not available at the Link Layer, e.g. global priorities /5/.
We will not try to make any improvements of contention time or number of collisions; however, the maximum throughput of the network does not suffer from our enhancements, while the worst-case response times at high load are reduced at least by a factor of 10.

The improved strategy was developed from a CSMA/CD strategy we call CSMA-B; after a few modifications, we were also able to obtain a result that is plug-in compatible with the Ethernet standard /2/.

ETHERNET

Although Ethernet is described thoroughly in /9/, /10/, and especially in /2/, we will highlight some points important to us. We will ignore all aspects of data representation but are interested in what we call the Ethernet 'protocol', a CSMA-scheme (Carrier Sense Multiple Access) with collision detection.

Ethernet nodes, when having a message to send, wait until they detect the channel to be idle and then start transmitting. Because of the signal propagation delay, some nodes might start transmitting simultaneously, an event called 'collision'. It might take twice the delay time from one end of the cable to the other until all of them sense this by watching for distortions of their own transmission. In /2/, the worst case round trip delay (including some safety margins) is called a 'Slot'.

In case of a collision, the nodes cease transmission for a random number of Slots and then try again as soon as the channel becomes idle. The node with the accidently smallest random delay then gets access while all others keep waiting. Using random delays minimizes the probability of repeated collisions.

Because the shortest message length with Ethernet is not much longer than one Slot, several messages of other nodes may pass during the random delay time. With our first protocol variations, we assumed that the nodes finish delaying as soon as they sense a signal (but still wait until the net becomes available) because this faciliates queueing control by the nodes themselves.

With low traffic, collisions will be rare; following the end of a transmission, however, they are to be expected more frequently, because other nodes might have got ready for another access attempt or have provided new messages meanwhile. If lots of messages are provided, queues will develop and several nodes will collide each time the net becomes available. If the nodes suspend delay counting as long as they sense a signal on the net, this 'First Collision' may be avoided /8/. As to /2/, Ethernet does not use this method.

OPTIMUM DELAYS

For our considerations, some rules on choosing optimum random delay times will be of interest:

1) It is reasonable to choose delay times that are integral multiples of one Slot (/2/, pp.13).
2) The time values should be equally distributed between 0 and a maximum value Dm equal to the number of nodes involved in the contention /10/.
If we further try to avoid the initial collision when the channel gets available, by delaying before the first transmission attempt, the total contention time (i.e. all subsequent delays and collisions until one first single node may transmit successfully) converges to e-0.5 (2.22..) Slots as node numbers go to infinity (assuming worst-case conditions about

cable length and node positioning on it) /4/. With First Collision, up to one more Slot is wasted at high load. The contention time can keep that short because among many nodes, there is probably always one which computes a short delay time.

There are deterministic Protocols that might need as few as 1 Slot if all nodes are involved /7/, but error recovery, switching nodes on and off and especially flexible addressing is still a problem with this.

B E B

In order to avoid excessive delay times as well as too many collisions, Ethernet starts up with an instant transmission attempt and then uses delay times redoubled at every collision: Dm=2↑C-1, where C is the number of collisions that a station experienced since first trying to transmit its actual message. This strategy is called 'Binary Exponential Backoff' (BEB). While collisions are very likely while Dm is short, the chance of resolving the conflict rapidly increases when it becomes longer. Therefore, this works with any number of nodes, making a deadlock impossible. In /2/, this Strategy is modified to 'Truncated BEB', where Dm is limited to a maximum value of 1024. This may help to avoid uselessly long delay times in some cases, but may also increase contention times when some hundred nodes are competing (see fig.1).

ETHERNET WAITING TIMES

Because the Collision Counter of an Ethernet node is reset after its message has been transmitted, but nowhere else, and because delay times are always completed regardless of any events on the net, some bad effects on access assignment take place.
At high load, nodes often finish delaying while some message is being transmitted. They wait until the net becomes idle and then try to transmit. If there are more nodes or the one just having transmitted still has another message, a collision occurs; the node that just had access would win this contention almost for sure, because its C was set to 0 (resulting in zero delay), while all others get their C's increased, lowering their future chances.
By this mechanism, a node once having access may send as many subsequent messages as it wants, leaving almost no chance for the others ('Blocking' the net). This might be useful when long documents are to be transmitted, splitted in many packets, but is dangerous with real-time applications.
It also has the advantage of dramatically lowering the contention times (but only if every nodes really wants to send several messages). In practice, nodes are not likely to send more than a few messages in immediate series, due to the slowness of their software and their limitation in data volumes; however, nodes newly becoming active, also have C-values of 0 and might therefore alternately block up others that have waited longer. This implies that message waiting times may at best have a distribution like with random queueing (fig.2); however, especially with load patterns that are good for queueing (i.e. that inhibit Blocking), Ethernet's contention times tend to be too long because its BEB increases the delay values too fast.

SIMULATION MODEL

The empirical studies we have undertaken could only be done by means of simulation. Simulation of very large systems requires careful modeling to keep computing times as short as possible.

We decided to use the Slot as the smallest time raster; Ethernet's message lengths are therefore approximated to be from 1 to 24 Slots. Because this is nearly accurate for the shortest message length, no finer resolution was required.

The simulation routine was of course event-driven. Message generation was allowed only during transmission time, i.e. no new demands arose during contention. This is suitable in the infinite-load situation upon which we concentrated in this research, as well as with long messages.

At high load, messages will queue up within the nodes causing a number of them to be constantly competing.

Because the number of nodes actually competing is statistically dependent on the number of connected nodes and the present system load (lower load means that fewer of them are involved at the average), considerations about strategy behaviour in this situation, with different node numbers, will also apply to limited periods of time under medium load.

In order to avoid Blocking with the Ethernet strategy, we also ran simulations where a total of n nodes out of 1000 nodes were revolvingly kept active by random message generation.

Our simulation program had a runtime of O(nodes*messages). We ran several 1000 simulations, with about 400 Million nodes*messages.

ETHERNET AT OVERLOAD

With Ethernet, overload may have diastrous consequences: If it continues for some time, the Collision Counters in the nodes approach 16 after about 4000 Slots, causing an error message and possibly suspending the node's activities, a catastrophic event with many real-time applications (if there is overload, a node that accidently experienced some collisions, will likely not get access because of its high C-value but increase it until 16 in 'First Collisions').

In order to get some reasonable results for comparison at high load, we assumed that the Ethernet nodes' Collision Counters are simply reset to 0 when approaching 16; this will then occur frequently (see also fig.3).

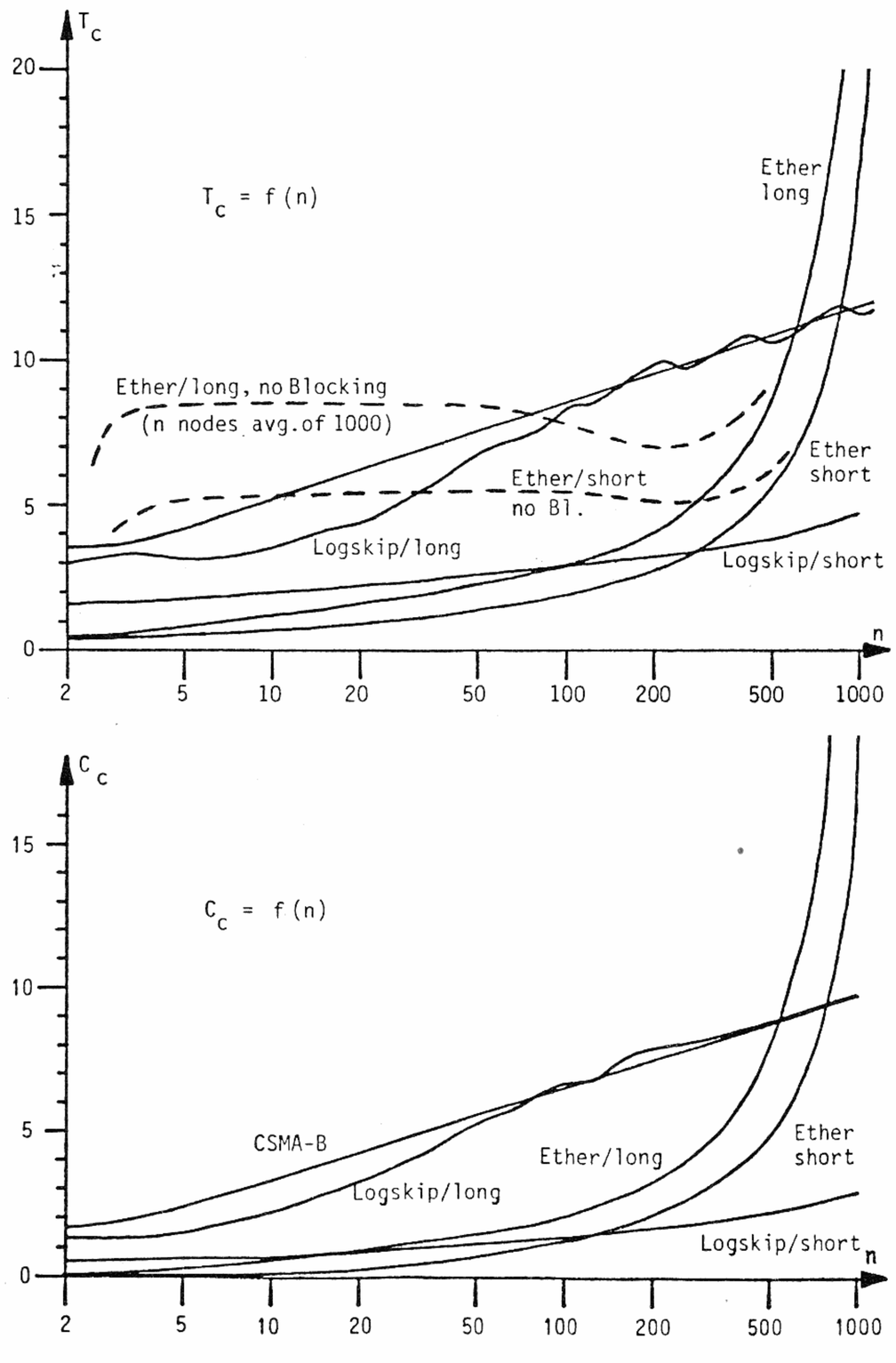


Fig.1 shows the mean contention time (Slots), Tc, and the mean number of collisions per contention, Cc, for various strategies at infinite load (n nodes constantly competing). Ethernet and the compatible Logskip strategy are shown at short and long messages (1 resp. 24 Slots). Because Ethernet's BEB is truncated at 1024, contention times with Blocking explode above 200 nodes. If Blocking is prevented (see also 'Simulation Model'), contention times are always higher than necessary.

WAITING TIMES WITH RANDOM ACCESS

The term 'Delay' often used in communication protocol analysis does not sufficiently characterize the performance with real-time applications. First, it is a mean value, while here we are interested in maxima; secondly, it normally includes queueing within nodes, which we cannot consider to be part of a protocol defining the two lowest ISO-layers only. Of primary interest to us are the waiting times of messages already offered to the net until their transmission, especially their deviation and expected maxima.

A simple random access scheme without any influences on queueing may illustrate this. Because each message has the same chance at every contention phase, the waiting time distribution (no. of messages transmitted above waiting time, expressed in message lengths), is given by the Poisson distribution:

$$r = \frac{mO}{N} e^{-\frac{m}{N}}$$

Where N : number of active nodes,
m : waiting time in avg. message-lengths,
mO: total messages processed,
r : no. of messages that were waiting for m others.

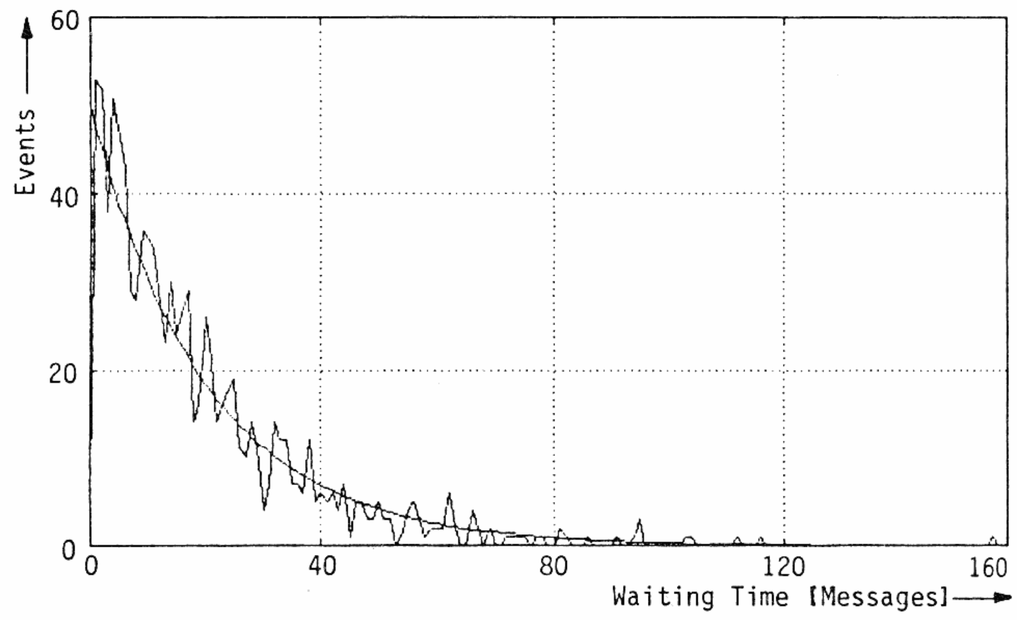


Fig.2: The theoretical r-curve for 20 nodes after 1000 messages, together with a simulation result.

EXPECTED MAXIMUM

r gives the distribution of mO discrete events; we could split the area under the curve into areas of the content 1, each representing a single event. We further might think of those events like being placed exactly in the middle of the areas, between two adjacent areas of the size 1/2. With this, the area right of the longest waiting time to be expected, mx, should also be 1/2:

$$\int_{mx}^{c\infty} r \, dm = \frac{1}{2}$$

Leading to

$$mx = N \ln 2\, mO$$

This is very close to our simulation results. For a system with accurate round-robin queueing, the corresponding formula would write like:

$$mx' = N - 1$$

Although the mean transmission waiting time of any protocol always has to be N-1, simple CSMA will easily produce maxima 10 times longer. Ethernet with Blocking shows maxima more than 100 times above the average; however, we have to admit that few messages will wait for longer than the duration of the longest congestion phase occuring (during an idle interval, they have a good chance of being tranmsmitted).

STRATEGY DESCRIPTION

The most important points of the strategy variants to be presented will be shown in tables like the following:

```
S T R A T E G Y  : ETHER
Collis./Involved : C:=C+1, Dm:= min(2↑C-1,1024)
       Actives   : -
Success/Winner   : C:=0,   Dm:=0
       Losers    : -
```

This describes the Ethernet strategy by the action taken on strategy variables (C, the collision counter, and Dm, the maximum delay time) at certain events: We distinguish between Collision and Success (one node gets access).
At Collision, there are Involved (colliding) and Active nodes (having messages pending, but delaying).
At Success, the Winner node transmits, while the Losers keep waiting.
Nodes having no message pending (Passives) are not subject to any of the strategy variants.
The strategy formulas are ordinary assignments, and the leftmost expression is to be computed first.
A random delay D between 0 and Dm is computed with all strategies; this is therefore not mentioned.
We can now easily identify the BEB at the Collis./Involved line, where C is incremented and exponentiated.

With Ethernet, C and Dm are reset at Success and not changed at Losing. This is exactly the place for improvements.

INITIAL MODIFICATIONS

The most natural way to avoid exessive waiting times for network access is, to establish some mechanisms of round-robin access assignment among the competing nodes. Because Ethernet nodes will only react on their own collisions, some changes are required before we can think of introducing any form of distributed queueing control; the following strategy we used as a base for improvements:

```
S T R A T E G Y  : CSMA-B
Collis./Involved : C:= C+1, Dm:= 2↑C-1
       Actives   : C:= C+1, Dm:= 2↑C-1
Success/Winner   : C:= 0,   Dm:= 0
       Losers    : C:= 0,   Dm:= 0
```

Here, all active nodes sense every collision and all successful transmissions. They may sense collisions they are not involved in, because the resulting Collision Fragments are always shorter than valid messages (/2/,pp.14). All delays are ended and all collision counters reset at Success. In addition all C's are incremented at every collision on the net (but only in nodes competing for access); we call this behaviour 'Global Consensus' (GC). The BEB starts over again at each contention phase, which of course increases the number of overall collisions; however, there are no overflowing C's, and every node has at least an equal chance at each contention phase. The overhead resulting from contention times can be kept sufficiently low.

QUEUEING CONTROL

Most interesting with CSMA-B is, that the average number of collisions during one contention phase, Cc, is logarithmically dependent on the number of competing nodes, with a standard deviation of about 1 (fig.1), /4/:

$$Cc \simeq \log_2(Na) \quad \text{(binary logarithm)}$$

Therefore, we can estimate Na from Cc and use this knowledge in order to vary the maximum delay times Dm of the nodes, giving them different chances for access according to their different waiting times.

At ideal round-robin access assignment, each node should wait for exactly one transmission of all other competing ones before it claims access itself; its waiting time then equals the number of competitors minus one (itself):

$$Qopt = Na - 1$$

In order to know the right time for an access attempt, the node neeeds to know its own waiting time Q as well as Na.

A simple counter for successful transmissions by others, started when an own message gets pending, will deliver the right value for Q.

Knowing Q and, approximately, Na, is sufficient for variating Dm appropriately. We want some algorithm that tends to decrease Dm until zero in the longest waiting node and lets it remain higher in the others. If more nodes happen to calculate short delay times because of the variability of the Na-estimation, this is no matter of concern because the conflict may easily be solved by maintaining the BEB.

We will maintain the strategy formula $Dm=2^{C}-1$. Then, we must vary C in order to get the right Dm-values for queueing control. We will still count collisions with C, so that it increases by Cc at every contention phase; however, after the contention, the loser nodes shall decrease their C by a value derived from the waiting time Q. Because now C does more than simply counting collisions, we will re-christen it 'Collision Weight'.

Idealizing, we can define a minimal condition that C should have the value 0 in the longest waiting node and the value 1 in the second longest waiting one following to a contention.

This would cause the C of the former second longest waiting node to increase to Cc+1 (averagely) at the next contention. The longest waiting node we assume to transmit; its C rises to Cc. Our new longest-waiting node should get its C down to zero by subtracting some function of Q:

$$Cc + 1 - f(Q) = 0$$

With

$$Cc \simeq \log_2(Na)$$

we get

$$f(Q) \simeq \log_2(Na) + 1$$

We assume round-robin queueing

$$Q \simeq Na - 1$$

and get

$$f(Q) \simeq \log2(Q+1) + 1$$

INTEGER LOGARITHM

With regard to simple implementation, we did not consider the use of floating-point logarithms. Instead, we use an integer logarithm that can simply be derived from a priority encoder:

$$f(Q) = ld'(Q)$$

where
ld'(0)=0, ld'(1)=1, ld'(2,3)=2, ld'(4,5,6,7)=3
etc..

Simulations showed that more 'accurate' implementations of f(Q) would produce no better results.

We may summarize: The Collision Weight is incremented at every collision (as was), but then decreased by ld'(Q) after the contention. We will see that this simple approach really works.

THE WINNER NODE

We still need an appropriate strategy for the Winner node. We know, that the shortest contention times may be expected when Dm=Na. Therefore,
with

$$Dm := 2\uparrow C - 1$$

we get

$$C \simeq \log2(Dm+1) \simeq \log2(Na)$$

Because we said that in the winner node, C should have approached

$$C \simeq Cc \simeq \log2(Na),$$

it should already have the most reasonable value; however, this worked out poorly with low node numbers. We therefore prefered to set C explicitly:

Assuming

$$Q \simeq Na$$

we may define:

$$C := ld'(Q)$$

This leads to lower contention times with few nodes; however, one may doubt if the additional effort is worthwhile, because few nodes simply can't produce high load for long, due to the slowness of their software.

LOGLOG

The strategy is now complete:

```
S T R A T E G Y  : Loglog
Collis./Involved : C:= C+1, Dm:= 2↑C-1
       Actives   : C:= C+1, Dm:= 2↑C-1
Success/Winner   : C:= ld'(Q),  Dm:= 0,  Q:=  0
       Losers    : Q:= Q+1, Dm:=0, C:= C-ld'(Q)
```

The difficulties in the theoretical analysis of this strategy led us to the use of simulation in order to investigate its performance and to test the effects of changes in the formulas. The conclusion of this investigation was that performance was good and no changes were recommendable except for some minor additions.

C-LIMITATION

A theoretical analysis of the development of the C-values within single nodes under equilibrium conditions, as well as simulations, showed it to be reasonable to limit C to a maximum of about 16. We also found that the C-values would react very fast if we changed the number of active nodes; the strategies' dynamic behaviour at load changes is excellent.

C-RANGE

An implementor should know that with Loglog, C-values down to -8 are to be supported, although they are to be interpreted as 0 when it comes to calculating Dm (see fig.5)

DELAY TIMEOUT

Given the strategy formula Dm=2↑C-1 with a maximum C of 16, it becomes obvious that Dm could theoretically become very large. Although this could only result from an instantaneous load decrease by violently discarding nodes or messages, it could sometimes become a problem.

A simple modification will solve this: D (not Dm !) should be limited to about 16 Slots.
If the net has been idle for this time, C should be lowered or set to 0 (we preferred the latter), and D should be recomputed (we call this 'Delay Timeout'). If more nodes are still competing, all are doing this simultaneously; the instant access attempt (D=0 !) results in a contention phase that serves for regaining reasonable C-values. An arbitrary node gets access, but the Losers then decrement their C's according to their waiting times, preparing the next messages to be sent in the right sequence.

Because Delay Timeouts are extremely rare, other strategies than setting C to 0 in this case will yield no practical advantage.

With the Ethernet strategy, Delay Timeout could replace the present Dm-limit of 1024.

QUEUEING PROPERTIES

Message waiting times, in our terms, begin when the message is at the head of the node's output queue and last until transmission. We are not concerned with the waiting times within the node's internal queues. 'Queueing' therefore takes place between the foremost messages or, so to say, between the nodes themselves.

We will show queueing behaviour by the waiting-time distribution: The number of messages having waited for some other messages until transmission, are plotted above those times, that are denoted in (average) message-lengths. The sample plots shown are representative.

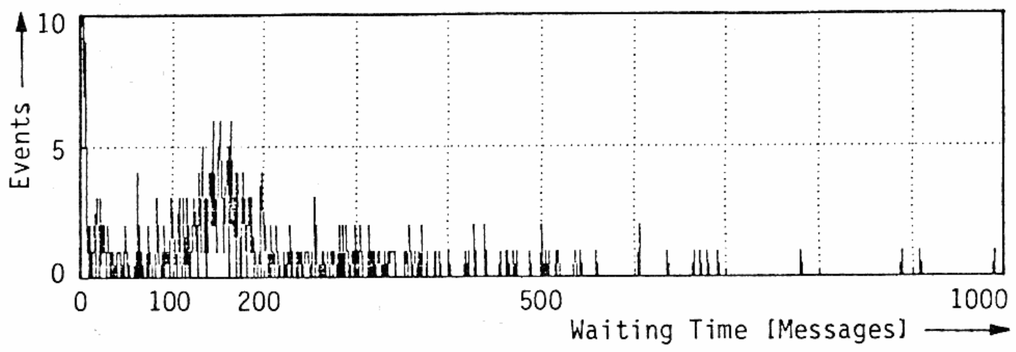


Fig.3 shows the distribution of the waiting times (in message lengths) with an Ethernet of 100 nodes after 1000 messages of maximum length have been transmitted (with Blocking). Note that one message had waited for almost all others. The maximum around 150 messages ((24+3)*150≃4000 Slots including contention time) is caused by the Collision-counters approaching 16 and being reset after about 4000 Slots at the average; this temporarily increases the node's chances at contention. If Blocking was inhibited, the distribution would more resemble to fig.2.

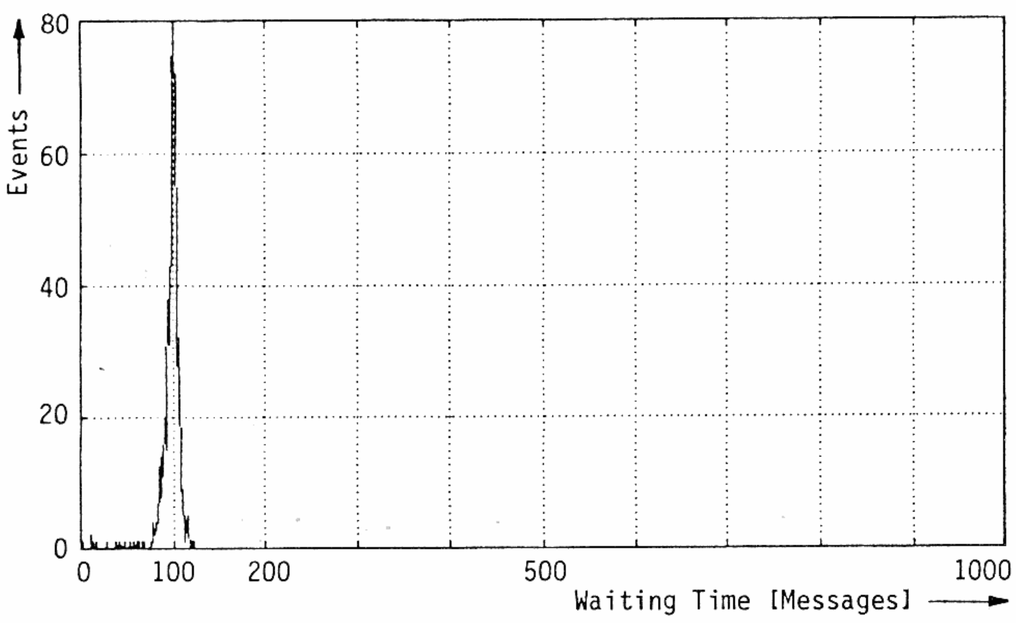


Fig.4 is drawn from a Loglog-simulation with the same assumptions like in fig.3 in order to show the difference in queueing behaviour.

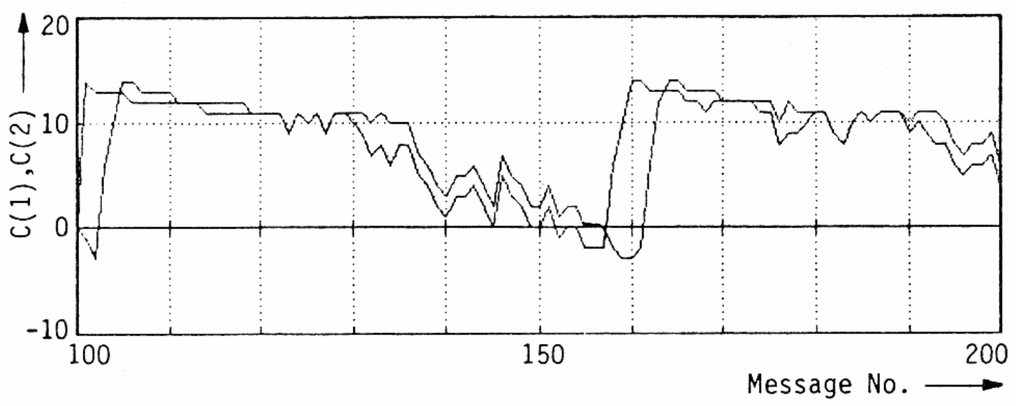
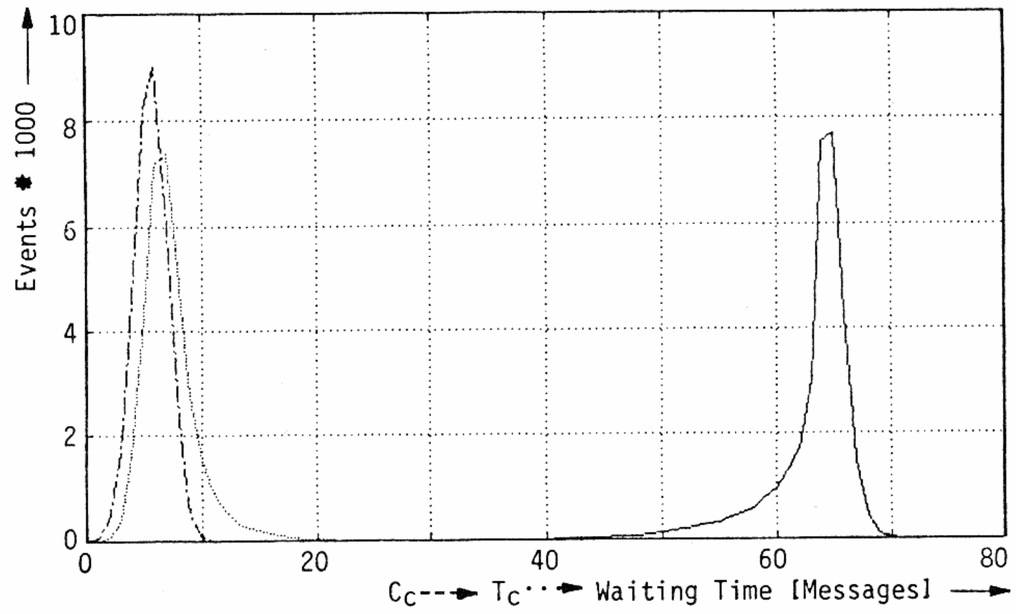


Fig.5 shows the Collision Weights of 2 Loglog-nodes out of 60, at long messages. The Collision-Weights leap upwards after transmission and then stick to their limit until they go down again when the time for the next transmission has come. Negative values are interpreted as zero; This causes some extra collisions, but it is also good for proper queueing. Throughput is not seriously affected, because with the Ethernet-compatible strategy Logskip (see 'Compatibility'), we can save contention time with short messages (fig.1).



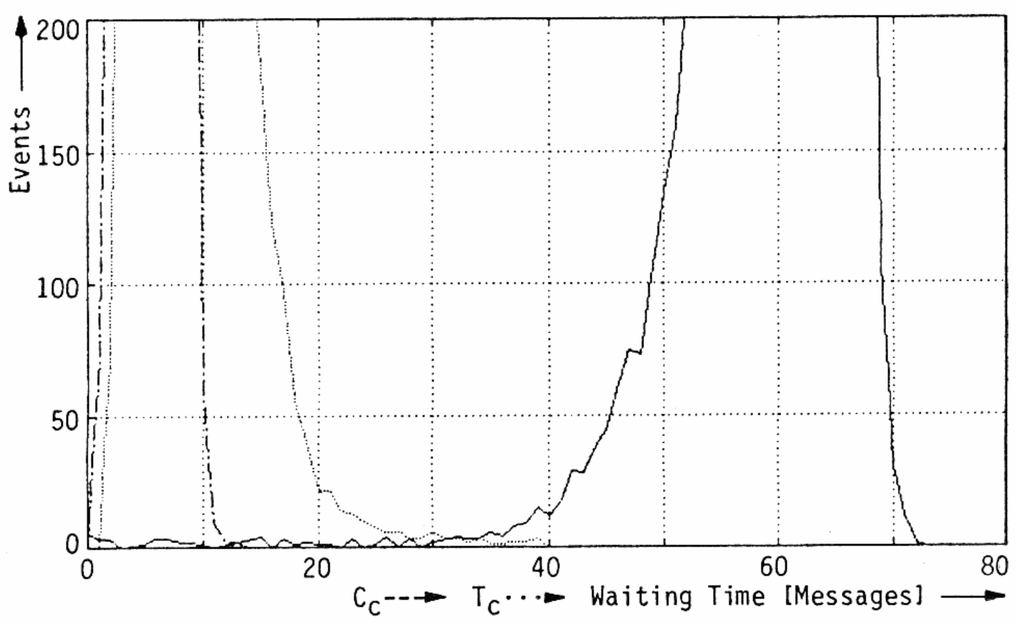


Fig.6 was drawn from a long-term simulation with 64 Loglog nodes over 30000 messages. In the second drawing, we have enlarged the vertical axis, in order to make single events visible. The waiting time distribution (solid line) is well limited; no single message had to wait for more than 72 others. The conclusion out of several very long simulations was, that maximum waiting times do not increase by more than one single message (at any number of nodes) if the number of processed messages is multiplied by a factor of ten. Also shown in the above diagram is the number of collisions per contention (dashed line) and the contention time in Slots (dotted line).

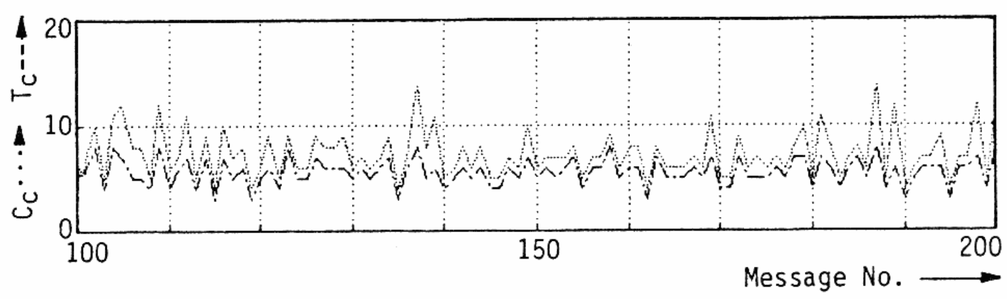
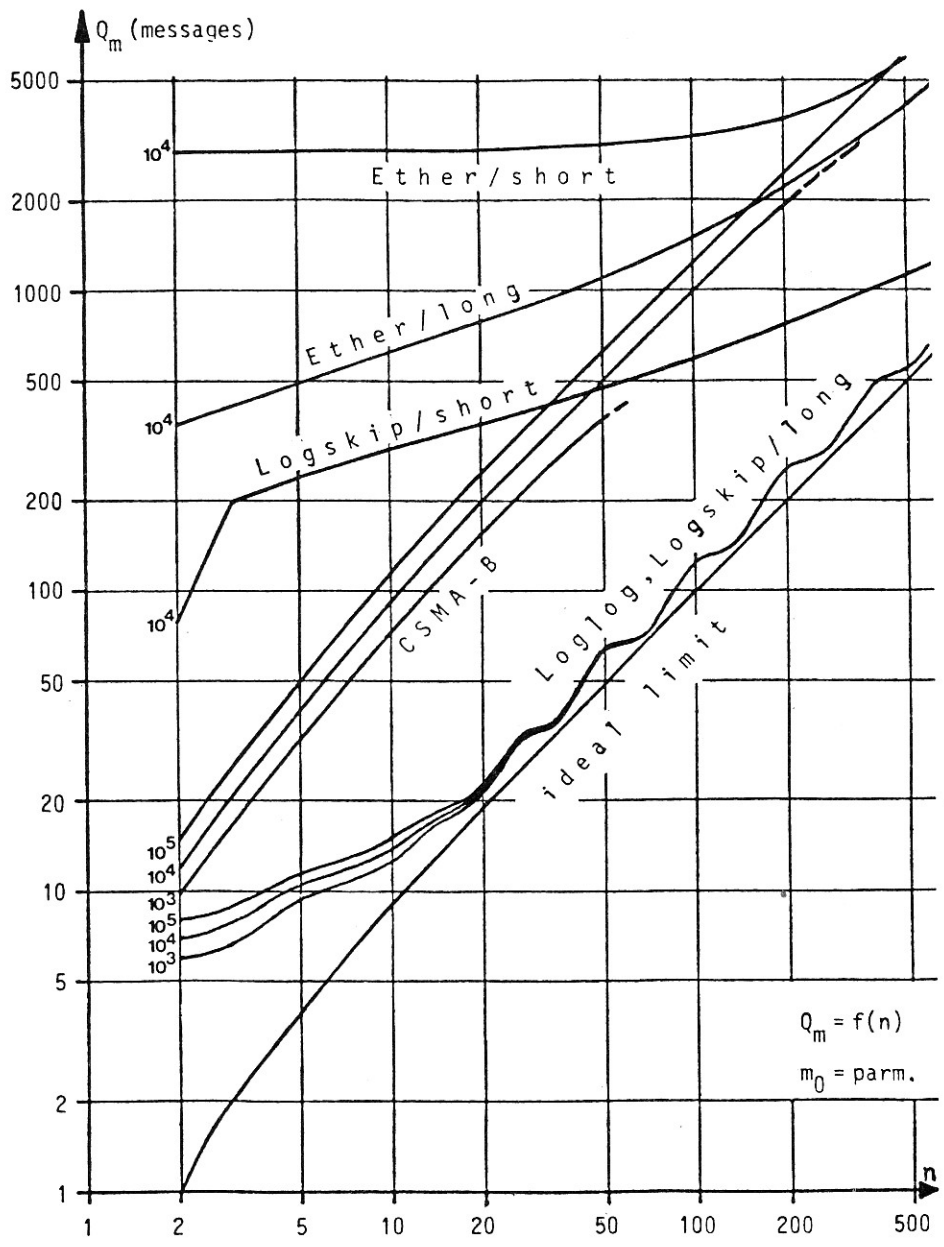


Fig.7 shows the contention times and collisions associated with each of 100 subsequent messages in a net of 60 Loglog nodes (like fig.5). The Collision numbers (dash-dotted line) have only a narrow deviation; an extra Collision Counter could be used for error detection, with a threshold of 16 like with Ethernet. The Contention Times (dotted line) show some minor spikes; this is similar to Ethernet.



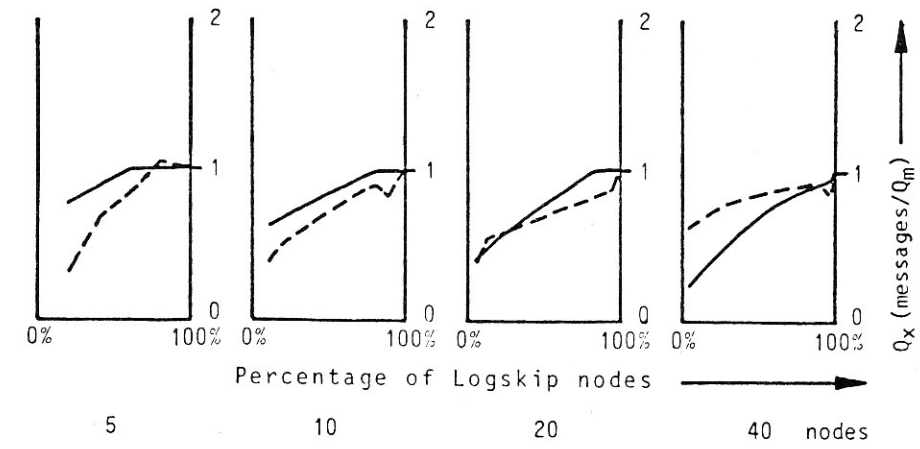
In fig.8, we have collected the information about waiting times. The maximum message waiting times Qm (in message lengths) to be expected with the different strategies are shown as a function of the number of active nodes N (load=1), for message lengths of 1 (short) and 24 Slots (long) and for different observation times (total messages processed, m0). Numerous simulation runs were necessary to derive these curves. The vacillation of the Loglog curves is due to the use of ld' (integer logarithm) in the strategy formulas, which makes performance somewhat dependent on the number of nodes with respect to the next power of 2. If Blocking is inhibited, Ethernet's maximum waiting times may be lower.

COMPATIBILITY

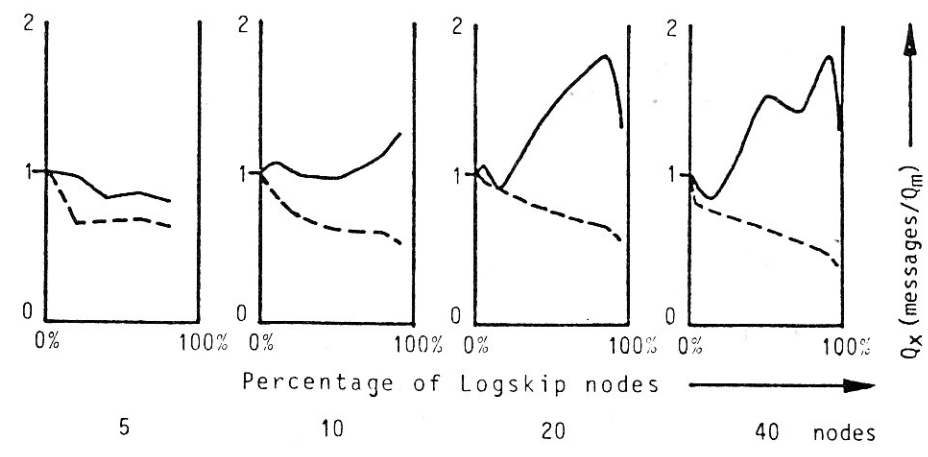
During our investigations, we found that interfaces using different CSMA/CD schemes could join the same net, provided that their retransmission attempt probabilities during contention are approximately alike. Loglog and Ethernet nodes are not compatible; however, little modifications of the Loglog strategy are sufficient to achieve this. Because we also desired to improve Loglog's throughput at short messages (i.e. to achieve less collisions), we tried the following change:

A node finishes delaying only if its delay time ends naturally (still limited to 16) or if it senses a collision. The node only consideres itself to have been 'Losing', if its delay time ends during the transmission by another one. It therefore does not increment its Q and does not take part in a First Collision if its delay time happens to 'skip' over the duration of a short message by others. This even allows for some 'Blocking'.

Everything else remains the same. During Contention, Global Consensus is maintained. With messages of 16 or more Slots in length, delay ends during the transmission, but this is not considered to be a Timeout, because the net has not been idle;
therefore, at message lengths of 16 up, Logskip is absolutely identical to Loglog.



Maximum expected message delays for the Logskip nodes
(normalized, message length ---1, —— 24 Minislots)



Maximum expected message delays for the Ethernet nodes
(normalized, message length ---1, —— 24 Minislots)

Fig.9 contains detailed information about the performance of mixed systems: it shows the relative message delays to be expected with different percentages of Logskip nodes in Ethernet systems of 5,10,20 and 40 nodes.
The general conclusion is, that there are no substantial disadvantages, if especially the more demanding nodes are equipped with Logskip. We have found that the Ethernet nodes may get a smaller relative share of the channel capacity if they represent less than 1/4 of all nodes. This is, however, no serious problem, because the Ethernet nodes, as said, would already show undesirable behaviour under extreme load situations. We have indications that throughput and compatibility might be further improved with some compromises in queueing behaviour; however, queueing showed to be the most important key to short waiting times. We therefore decided first to concentrate on thoroughly investigating the Logskip strategy, which, in its present form, is adequate for upgrading exitsting netorks without problems.

Because with short messages Q is not always incremented, the nodes cause less collisions on the cost of queueing.

The worst-case waiting time for access (most important with real-time applications) remains unchanged because it depends mostly on the behaviour with long messages. Even with short messages, queueing is still better than that of Ethernet with long messages (see fig.8).

An important point is, that Logskip is indeed fairly compatible with Ethernet (fig.9). Channel utilization (given by the contention time / message length relation) with Logskip is, all in all, no worse than with Ethernet.

PRIORITY CONTROL

Sometimes there is a demand for processing certain messages faster than others. This might be achieved by presetting the Q-values within certain node interface to values other than zero at message generation.
Care must be taken to keep all waiting time demands within the net consistent. Some of the Q-values therefore should begin below zero, interpreted as zero in the strategy formulas.

This type of priority control is very interesting for its security, because simulations showed that Loglog and Logskip nodes would recover almost instantly if we were setting any of their strategy variables to inconsistent values.

RESPONSE TIMES

We get the worst-case message waiting time by taking the maximum message length of 24 Slots, adding the contention time (fig.2) and multiplying this with the maximum waiting time (in messages) from fig.8.
Given 50 microseconds Slot length and 1024 nodes, we get 40 seconds for Ethernet with Blocking (unrealistic, because the nodes can't generate messages fast enough), about 30 seconds for Ethernet without Blocking and 1.8 seconds for Logskip.

Because the absolute minimum (no contention time, perfect queueing) would be about 1.3 sec., Logskip is already near optimum. If some application requires time limits below this, one would have to reduce the maximum message length, the maximum node number, or even the Slot time.

CONCLUSION

We have shown a CSMA/CD strategy with controlled queueing behaviour, maintaining the security advantages of a decentralized, probabilistic protocol.
Priority control may be achieved in a very fault-tolerant way. Interfaces using the proposed strategy may work together with Ethernet interfaces on the same cable, allowing for easy and fully compatible upgrading of existing networks.

ACKNOWLEDGEMENTS

This work originated from a Diploma Thesis /3/ done at the Institut fuer Prozessdatenverarbeitung of the Technical University of Berlin. The author also wishes to thank the Process Control Center (Prozessrechnerverbundzentrale-FSP/PV) of the Technical University of Berlin for providing computing time for the simulations.

R E F E R E N C E S

/1/ Abramson, N., 1970. The ALOHA System-Another Alternative for Computer Communications. AFIPS Conf.Proc. vol.37, (1970)

/2/ BLUE BOOK, 1980. XEROX Corporation, OPD Systems Development, 3450 Hillview Avenue, Palo Alto, CA 94304

/3/ Hainich, R., 1980. Problemanalyse und Entwurf eines dezentralen, lokalen Kommunikations- Systems fuer die Realzeit-Datenverarbeitung. Diplomarbeit am Inst. f. Techn. Informatik der TU Berlin

/4/ Hainich, R., 1983. Acess Mechanisms for CSMA/CD with Real-Time Applications (to be published)

/5/ Iida, Ishizuka, Yasuda, Onoe, 1980. Random Access Packet Switched Local Computer Network w. Priority Function. NTC'80, IEEE 1980 Nat. Telecomm. Conf., Houston,TX, USA, 30Nov-4Dec

/6/ Johnson, D.H., O'Leary, G.C., 1979. A Local Access Network for Packetized Digital Voice Communication. NTC'79, IEEE 1979 Nat. Telecomm. Conf., Washington, DC, USA, 27-29 Nov 1979

/7/ Kleinrock, L., Scholl, M.O., 1980. Packet Switching Radio: New Conflict-Free Multiple Access Schemes. IEEE Trans. Comm. 7/80

/8/ Labetoulle, J., 1980. Etude Analytique du Reseau DANUBE. (Communication no.204). INRIA Rocquencourt B.P.105, 78150 Le Chesnay, France

/9/ Metcalfe et al., 1977. Multipoint Data Communication System w. Collision Dtection. U.S. Patent No. 4.063.220, 13.12.77

/10/ Metcalfe, R.M., Boggs, D.R., 1976. Ethernet: Distributed Packet Switching for Local Computer Networks. Comm. ACM, 7/76

/11/ Shoch, J.F., Hupp, J.A., 1979. Performance of an Ethernet Local Network. Proc. Local Area Netw. Symp., Boston, May 79

/12/ Tobagi, F.A., Hunt, V.B. 1979. Performance Analysis of Carrier Sense Multiple Access with Collision Detection. Proc. of the LACN Symposium, May 1979

/13/ Yemini, Y., Kleinrock, L., 1979. On a General Rule for Access Control or Silence is Golden. Flow Control in Comp. Netw., ed. J.L.Granse & M.Gien, North Holland 1979