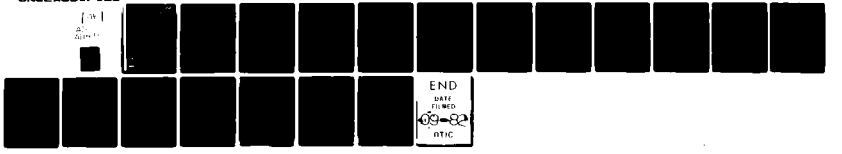
AD-A118 111 SRI INTERNATIONAL MENLO PARK CA F/G 12/1
THE IMPLICATIONS OF INTERACTIVE CONSISTENCY FOR FAULT-TOLERANT --ETC(U)
JUL 82 M C PEASE DAAG29-79-C-0102
UNCLASSIFIED ARO-16313.2-EL NL



END
DATE FILMED
DTIC

THE IMPLICATIONS OF INTERACTIVE CONSISTENCY FOR FAULT-TOLERANT COMPUTATIONS

Final Report

July 1982

By: Marshall Pease, Staff Scientist

Computer Science Laboratory
Computer Science and Technology Division

Prepared for:

U.S. Army Research Office
P. O. Box 12211
Research Triangle Park, North Carolina 27709

Attention: Dr. Jimmie Suttle, Director
Electronics Division

Contract No. DAAG29-79-C-0102
SRI Project 8492

SRI International
333 Ravenswood Avenue
Menlo Park, California 94025
(415) 326-6200
Cable: SRI INTL MPK
TWX: 910-373-2046

The views, opinions, and/or findings contained in this report are those of the author and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: Final Report	2. GOVT ACCESSION NO.: AD A118111
	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): THE IMPLICATIONS OF INTERACTIVE CONSISTENCY FOR FAULT-TOLERANT COMPUTATIONS	5. TYPE OF REPORT & PERIOD COVERED: Final Report 5/1/82 through 4/30/82
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Marshall C. Pease	8. CONTRACT OR GRANT NUMBER(s): DAAG29-79-C-0102
9. PERFORMING ORGANIZATION NAME AND ADDRESS: SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: ARO Project P-16313-EL
11. CONTROLLING OFFICE NAME AND ADDRESS: U. S. Army Research Office, Post Office Box 12211, Research Triangle Park, NC 27709	12. REPORT DATE: July 1982
	13. NUMBER OF PAGES: 17
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

NA

18. SUPPLEMENTARY NOTES

The view, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Interactive consistency, Byzantine General problem, connectivity, network connectivity, distributed algorithms, decentralized algorithms, authenticators, safe paths

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The program examined the significance of interactive consistency for battlefield communications. This environment is different from those considered in the original work on interactive consistency in three ways: the network is apt to be very large; it is incomplete in the sense that a particular node will not be able to communicate directly with every other one; and it is variable in unpredictable ways as links fade out or become useful. Recent work on interactive consistency is reviewed for its relevance to an environment with these characteristics.

(over)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. cont.

Recent work has demonstrated that, in incomplete networks, the network connectivity is critical. Specifically, in a network with N nodes of which not more than N may be faulty, interactive consistency can be assured if N is greater than 3M and the network connectivity is greater than 2M. Given the variability of the networks being considered, this makes it vital that we have a way for determining network connectivity.

Two distributed algorithms for determining network connectivity have been developed. One is a version of Dinits's max-flow algorithm. It is attractive in being reasonably efficient, but is vulnerable to a particular kind of malfunction that may lead a node to generate false information. The other is based on the Ford-Fulkerson max-flow algorithm, but using cryptographic methods to generate authenticators that limit what errors can be introduced by a faulty node (or by enemy interference) without detection. For the former, a reference is given to a technical report that describes the algorithm in detail and documents an event simulation of it. For the latter, a reference is given to a paper being presented.

Table of Contents

I PROBLEM STATEMENT

A. Interactive Consistency

The initial goal of this program was to assess the applicability of interactive consistency to battlefield communications. Interactive consistency is defined as follows:

* Suppose a node in a network has a datum that is unique to it. It is desired to transmit that datum to every other node in such a way that every node either knows the originator's correct datum, or knows that inconsistent values were transmitted by the originator to different nodes. The object is to ensure that every node reaches a conclusion about the originator's datum that is consistent with the conclusions reached by the other nodes--hence the term.

* The difficulty of the problem arises from the assumption that some nodes, including the originator, may be faulty, perhaps maliciously so. The originator can send different values to its various neighbors. Other nodes can misrelay messages they have received, misrepresenting to a neighbor the message they are relaying.

* To avoid being confused or misled by faulty nodes, copies of the original message are sent by various routes. A node can receive many copies of a message via different routes. By comparing these copies, the node tries to conclude either that the originator is faulty and its message ignorable, or that some particular value should be used for the originator's datum.

* It should be carefully noted that the foregoing does not state that, if a node knows that the originator is faulty, it will always ignore the message. It can do so only if can also be certain that every other nonfaulty node can conclude the same thing. It must reach a conclusion that is consistent with that of all other nonfaulty nodes. Therefore, if some nodes are unable to determine that the originator is faulty, those that know it must ignore the fact.

* An algorithm for interactive consistency will specify what routes are to be used for the various copies of a message, and how the copies received at a node are to be evaluated. The exact conditions that must be met to obtain interactive consistency are as follows:

  (1) If the originator and node X are both nonfaulty, X must find the correct value for the originator's datum.

  (2) If nodes X and Y are nonfaulty, then whether the originator is faulty or not, X and Y must reach the same conclusion about the originator. They either both conclude that the originator is faulty and ignorable, or they must both conclude they should use identical values for the originator's datum.

> The first condition implies that the datum of any nonfaulty node is correctly identified by every nonfaulty node. The second condition implies consistency in the conclusions reached by all nonfaulty nodes. Note that we are not concerned with what conclusion may be reached by any faulty node.

Prior to the program, we had proved [1] that, in a complete network of N nodes in which as many as M nodes may be faulty, interactive consistency can be ensured if and only if $N > 3M$. The proof was constructive in the following sense:

* The sufficiency of the condition was established by specifying a particular algorithm that would be applicable when $N > 3M$ and that would ensure interactive consistency.

* The necessity was established by specifying a strategy that the faulty nodes could follow if N were not greater than 3M and that would disrupt consistency.

ARO's interest in interactive consistency arose, in part at least, within the context of battlefield communications. Of particular note was the potential application of this technique to the synchronization of spread-spectrum transmitters and receivers to eliminate or reduce the need for cross-correlation in the acquisition of a communication.

In the battlefield environment, we must guard against the possibility of the enemy's capturing one or more nodes, or creating nodes that may be accepted as nodes in the network. The enemy could then attempt to desynchronize the network. The interactive consistency requirement is an explicit statement of what is necessary to prevent the enemy from using a strategy of deliberate desynchronization.

B. Interactive Consistency in an Incomplete Network

In the battlefield environment, the communication network will usually link a great many nodes, but only incompletely. That is, we cannot expect that every node will be able to communicate directly with every other one. The original work on interactive consistency assumed that the network was complete. One of the problems of concern to us, therefore, was how the algorithm should be modified for an incomplete network.

This problem was solved by D. Dolev, then of Stanford University, subsequently at IBM Laboratory [2]. He showed that the key property of an incomplete network is its connectivity and that the main theorem should be expanded to include conditions on it.

The connectivity between nodes S and X in a network is the number of node-disjoint paths from S to X. A path from S to X is an ordered sequence of nodes such that the first node in the sequence is S, the last node is X, and the graph has an edge between every two successive nodes in the sequence. In addition, we assume that no node appears more than once in a path, so that a path does not contain any loops. Two paths from S to X are node-disjoint if they have no nodes in common except S and X[1]. A set of paths is node-disjoint if all pairs of paths in the set are node-disjoint. The connectivity of a network is the minimum connectivity that exists between any pair of discrete nodes in the network.

Dolev showed that, if a network contains N nodes of which as many as M may be faulty, and if the connectivity of the network is K--then interactive consistency can be ensured if

(1) N is greater than 3M, and

(2) K is greater than 2K.

The first condition is the same as that found before. The second is the additional condition that takes account of the network's incompleteness. Dolev showed that, under these conditions, interactive consistency could be

---

[1]If there is an edge between S and X, the path S-X is considered to be unique, and can be included only once in a set of node-disjoint paths from S to X.

obtained by using an algorithm that is very similar to the one developed in our original work.

This result showed what is needed for applying interactive consistency to the types of networks that occur under battlefield conditions. In addition, it demonstrated the importance of the connectivity property.

C. Distributed Connectivity Algorithms

In considering the connectivity of a communication network in the battlefield environment, we must take account of several conditions. One is that the network will change as vehicles move and as other events occur. Furthermore, the changes cannot be accurately predicted or immediately identified. The network is likely to change, however, at a rate that is sufficiently slow to make it useful to invest some effort into finding the characteristics of the network. Although the available connectivity between a given source and sink changes with time, information about the existing connectivity and corresponding node-disjoint paths can be expected to be valid for a sufficient period to be useful.

Given, then, that we wish to determine the connectivity between nodes S and X (where, for example, S may be a command post and X a subordinate one), there are some special requirements on an algorithm to find the connectivity:

* The algorithm should be distributed so that its operations are executed in the network itself rather than in some central unit. This is necessary to avoid having to transmit the required information to the central unit, an action that would itself depend on the availability of reliable communications.

* The algorithm should lead to at least a reasonably efficient set of paths--i.e., to paths of the minimum lengths consistent with the connectivity requirement. In addition, execution of the algorithm should not require excessive message passing that would itself be an inordinate load on the network's resources.

* The algorithm should be protected to the maximum extent possible from malicious interference. We cannot prevent a faulty node from being incorporated into one of the paths. A faulty node can hide its faultiness during execution of the algorithm, misbehaving only when the path is used for communication purposes. Because we are using node-disjoint paths, however, the effect of a hidden fault is limited to at most a single path (assuming the faulty node is not S or X, the terminal nodes on the path). Therefore, the interference of greatest concern occurs when a faulty node can affect execution of the algorithm, preventing the discovery of potential paths that are actually fault-free.

The issue of efficiency is not as straightforward as it may seem. To make maximum use of the available connectivity, we may be required to use paths all of which are of greater than minimum length. This is illustrated in the network of Figure 1.

In the network of Figure 1, the shortest path from S to T is S-E-C-T.

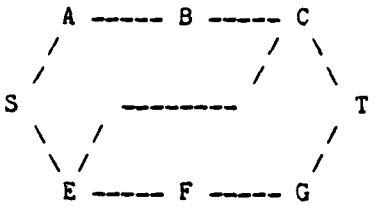


Figure 1

A Network Illustrating the Efficiency Issue

However, the use of this path makes it impossible to find any other. To obtain the full connectivity of the graph, we must use the paths S-A-B-C-T and S-E-F-G-T, both of which are longer than the minimum path.

We might also be tempted to depend on the use of a broadcast mode, e at the cost of some additional message traffic. The simplification of control problem might justify the traffic penalty. However, the network Figure 1 can also serve to illustrate the fact that using a broadcast mode not a suitable answer.

Consider what happens in Figure 1 with a broadcast mode. Broadcasting requires that we limit the propagation of messages some way so that they will not ring indefinitely. A common way of doing this is to let a node accept a message only once. Any copies it receives after accepting the first one are ignored. Suppose we use this rule in the network of Figure 1. Suppose, also, that the only delays are transmission ones, and that all links exhibit the same delay. Under these conditions, S, seeking to send a message to T, broadcasts copies to its immediate neighbors, A and E. These two nodes rebroadcast it: A sends a copy to B and E sends copies to C and F. B tries to rebroadcast the message but is rebuffed by C, which has already seen a copy. F does manage to send a copy to G, and C and G send copies to T. Thus, two copies are received at T, but they did not come via node-disjoint paths. One message was transmitted along the path S-E-C-T, the other along S-E-F-G-T. If node E is in fact faulty, it has the opportunity to alter both messages identically. T, consequently, cannot discover that E is faulty by comparing the two copies it received.

In examining the literature, we found that much work had been done on algorithms that could be adapted to determining the connectivity of a network.

However, the known algorithms were neither distributed nor protected against malicious or accidental interference. We therefore recognized the problem as critical for the maintenance of reliable battlefield communications.

II RESULTS

A. Interactive Consistency

A new paper on interactive consistency entitled "The Byzantine Generals Problem," by L. Lamport, R. Shostak, and M. Pease has been accepted for publication in TOPLAS. This paper, due to appear shortly, acknowledges that the work on which it reports was supported in part by the present contract.

B. Distributed Connectivity Algorithm

The first attempt to handle the connectivity problem led to a distributed algorithm that is reasonably efficient, but vulnerable to certain types of malfunction. In particular, under especially adverse conditions, a single fault might possibly prevent the algorithm from finding any fault-free paths, regardless of the actual connectivity of the network. The algorithm is a distributed version of Dinits's max-flow algorithm [3, 4].

This algorithm has been developed in detail in Technical Report CSL-127, entitled "A Decentralized Algorithm for Network Connectivity," by M. Pease, J. Meseguer, and D. Dolev, distributed in July 1981. The report discusses the background of the problem, and surveys the related work described in the literature. It presents a detailed exposition of the algorithm and proves it. It also documents an event simulation program that allows the user to apply the algorithm to an arbitrary network.

The vulnerability of the algorithm arises as follows. An process that is executed periodically during application of the algorithm develops an acyclic subnetwork rooted in S that is consistent with the link and node capacities and with any known flows. During this process, each node finds what it believes is its minimum distance from S and informs its neighbors of the fact. When each node has established a distance that is consistent with the distances reported to it by its neighbors, the desired acyclic subnetwork has been identified. The algorithm searches for a new flow in this subnetwork that can be used to augment any known flows. Dinits proved that this process would lead to the maximum possible flow after some finite number of iterations.

The algorithm is vulnerable to a fault that may cause a node to misstate its distance from S. If node X asserts that its distance is smaller than it

actually is, X's neighbors may be led to believe that their shortest distance from S is through X, and that, consequently, their minimum distances from S are less than they actually are. False information can be propagated. Under worst-case conditions, the algorithm can fail to find any path from S to T that does not pass through X, regardless of the actual connectivity of the network.

The algorithm has a number of desirable properties, but the vulnerability identified above does limit its usefulness for battlefield networks that are potentially subject to enemy interference.

B. Safe Paths in a Faulty Environment

To avoid the vulnerability of the algorithm cited above, we have developed a second algorithm. A paper describing the latter has been accepted for the ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, to be held in Ottawa, Canada, August 18-20, 1982. This algorithm can be described as a distributed form of the Ford-Fulkerson max-flow algorithm [5] using authenticators [6, 7] to ensure that a message cannot be altered undetectably before relay.

This algorithm, using messages that record the routes each has followed, develops paths back from the terminal towards the sink. These messages are protected by authenticators--public-key cryptosystems that are presumably unforgeable. Node X, for example, may receive a message from node Y that claims a route from the terminal T to Y. This message includes both the message Y that claims it received and Y's authenticator. By using Y's public key, X can strip Y's authenticator and obtain the message that Y claims to have received. If this message was presumably received from Z, it will contain Z's authenticator and the message claimed by Z. And so on, back to T. Node X, by using the public keys for the various nodes, can verify the messages at all intermediate nodes, and therefore--up to a point--the route from T to itself.

A faulty node can create trouble. Node Y may claim that the message it is relaying came from Z. Perhaps it actually received the message from U which received it from Z. During the decoding process, Y recovered an exact copy of the message U received. It can forward this message to X, claiming that message to be the one it received. Hence, while a faulty node cannot falsify the route of a message, it can excise one or more nodes at the end of the claimed route from T. This is the only way a faulty node can introduce an undetectable error into the information being disseminated.

The effect, proved in the paper, is that M faulty nodes can, at worst, prevent the algorithm from finding 2M fault-free paths from S to T--i.e., two for each faulty node. In particular, if Y is faulty, it can cause S to believe there is a path to T through Y symbolized as S-(p1)-Y-(p2)-T where (p1) and (p2) are subpaths. The acceptance of this path can prevent S from finding one fault-free path that uses a node in (p1), and another that uses a node in (p2). We have proved that this is the worst that can happen; the

algorithm sharply limits the damage a faulty node can do.

To illustrate the worst-case effect of a faulty node, let us consider the simple network shown in Figure 2.

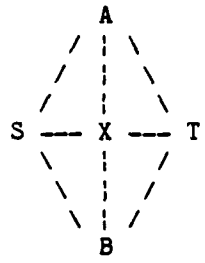


FIGURE 2

Network Illustrating the
Worst-Case Effect

Consider what can happen, first, if X is not faulty--or, at least, does *not misbehave* during execution of the algorithm. On the first pass through the algorithm, S might find the path is S-A-X-B-T. (Although this is not the shortest path, it might still be the first one found.) On the next execution of the inner loop of the algorithm, we could find any one of S-X-A-T, S-B-X-T, and S-B-X-A-T. (The path S-X-T, which might appear to be possible, violates the node-capacity limit by creating two flows through X.) The first of the possible paths would cancel the flow from A to X in the initial path, leaving the two paths S-A-T and S-X-B-T. The second would cancel the flow in X-B, leaving the two paths S-A-X-T and S-B-T. The third would cancel the flows in both A-X and X-B, leaving the paths S-A-T and S-B-T. In any case, we could then proceed to find the third path in a third pass through the algorithm's inner loop. The algorithm will find the three paths, S-A-T, S-X-T and S-B-T.

On the other hand, if X is faulty, it can prevent any one of the three new paths from being found during the second execution of the inner loop. It can do this simply by refusing to relay any messages. The algorithm stops on the second pass after having found only the S-A-X-B-T path, which is not fault-free. It has failed to find the two fault-free paths that are available. Of the two fault-free paths that were not found, S-A-T uses A, which is in (p1), S-B-T uses B in (p2).

It is proved that this is the worst that a faulty node can do to disrupt the algorithm.

The paper discusses the details of this algorithm, its formal proof, and its complexity.

III PARTICIPATING SCIENTIFIC PERSONNEL AND DEGREES AWARDED

The following personnel were supported by, and contributed to, the research effort:

Marshall C. Pease (part time)
Jose Meseguer (part time)

No degrees were awarded.

REFERENCES

1. M. Pease, R. Shostak and L. Lamport, "Reaching Agreement in the Presence of Faults." Journal of the ACM, Vol. 27, No. 2, pp. 228-234 (April 1980).

2. D. Dolev, "Unanimity in an Unknown and Unreliable Environment," 22nd IEEE Annual Symposium on the Foundations of Computer Science, pp. 159-168 (1981).

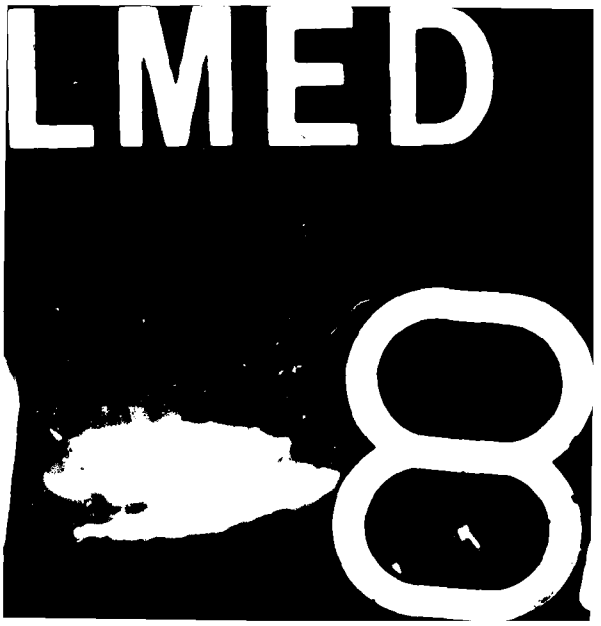
3. E. A. Dinits, "Algorithm for Solution of a Problem of Maximal Flow in a Network with Power Estimation, " Soviet Mathematics, Report No. 11, pp. 1277-1280 (1970).

4. S. Even, "The Max Flow Algorithm of Dinic and Karzanov: An Exposition," Technical Memorandum MIT/LCS/TM-80, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts (December 1976).

5. L. R. Ford and D. R. Fulkerson, Flows in Networks (Princeton University Press, Princeton, New Jersey, 1962).

6. W. Diffie and M. E. Hellman, "New Directions in Cryptography," IEEE Transactions in Information Theory, IT-22, pp. 644-654 (November 1976)

7. R. L. Rivest, A. Shamir and L. Adlemen, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Communications of the ACM, Vol. 21, No. 2, pp. 120-126 (February 1978).


LMED
8