

in the probabilities can be computed in polynomial time.)

- The value of x_e^{uv} for other edges f may also be determined, e.g., if $x_e^{uv} = 1$, then for all edges f adjacent to u , $x_f^{uv} = 0$.

A major stumbling block in applying the method of conditional probabilities is always the computation of the conditional probabilities. In our case, we do not compute the exact probability that there exists an overloaded edge (even initially), but rather only estimate it. Consequently, if the estimator is not chosen judiciously, it may happen that when a variable is considered, according to the estimator, no value assigned to it can lead to a good solution. To overcome this difficulty, following Raghavan [16], the notion of a pessimistic estimator is introduced. We call \hat{P}_j a pessimistic estimator of the conditional probability P_j if it satisfies the following conditions:

1. $\hat{P}_0 < 1$.
2. For any partial assignment of the first j variables, $P_j \leq \hat{P}_j$.
3. $\min\{\hat{P}_j^0, \hat{P}_j^1\} \leq \hat{P}_{j-1}$ where \hat{P}_j^i is the estimator of P_j^i for $i = 0, 1$.
4. The pessimistic estimators can be computed in polynomial time.

It is not very hard to see that such a pessimistic estimator can equally well be used in the method of conditional probabilities instead of the exact conditional probabilities which are hard to compute in general. We now show that the pessimistic estimator that we will choose indeed satisfies the above conditions. We have earlier proved that initially,

$$\begin{aligned} \text{Prob}[\text{set is bad}] &\leq \sum_{f \in E} \text{Prob}\{l(f) > (1 + \gamma_f) \bar{l}(f)\} \\ &\leq \sum_{f \in E} \frac{E[e^{\lambda_f l(f)}]}{e^{(1+\gamma_f)\lambda_f \bar{l}(f)}} < 1 \end{aligned}$$

Notice that λ_f and γ_f depend on the edge f . We define

$$P_0 = \sum_{f \in E} \frac{E[e^{\lambda_f l(f)}]}{e^{(1+\gamma_f)\lambda_f \bar{l}(f)}}$$

The estimator at Step j is defined to be

$$\hat{P}_j = \sum_{f \in E} \frac{E[e^{\lambda_f l_j(f)}]}{e^{(1+\gamma_f)\bar{l}(f)\lambda_f}}$$

where $l_j(f)$ is a random variable denoting the load on edge f at the end of Step j . For example, suppose that $l(f) = x_1 + x_2 + x_3 + x_4$ and at the end of Step j , $x_2 = 0$ and

$x_4 = 1$. Then, $l_j(f) = 1 + x_1 + x_3$. ($\bar{l}(f)$, γ_f and λ_f retain their original values).

Condition (4) holds since the changes in the probabilities at each step can be computed in polynomial time as mentioned earlier. (Notice that the random variable $l_j(f)$ is the sum of independent random variables). Condition (2) holds since

$$\begin{aligned} P_j &\leq \sum_{f \in E} \text{Prob}\{l_j(f) > (1 + \gamma_f)\bar{l}(f)\} \\ &\leq \sum_{f \in E} \frac{E[e^{\lambda_f l_j(f)}]}{e^{(1+\gamma_f)\lambda_f \bar{l}(f)}} = \hat{P}_j. \end{aligned}$$

Let us show that condition (3) holds as well. Suppose that at Step $j+1$ variable x_e^{uv} is being considered. By definition,

$$\begin{aligned} \sum_{f \in E} E[e^{\lambda_f l_j(f)}] &= P_e^{uv} \cdot \sum_{f \in E} E[e^{\lambda_f l_j(f)} | x_e^{uv} = 1] \\ &\quad + (1 - P_e^{uv}) \cdot \sum_{f \in E} E[e^{\lambda_f l_j(f)} | x_e^{uv} = 0] \end{aligned}$$

where the probability of choosing edge e as part of the path from u to v is P_e^{uv} (given the assignments of the previous j steps). Now,

$$\begin{aligned} \hat{P}_{j+1}^1 &= \sum_{f \in E} \frac{E[e^{\lambda_f l_j(f)} | x_e^{uv} = 1]}{e^{(1+\gamma_f)\bar{l}(f)\lambda_f}} \\ \hat{P}_{j+1}^0 &= \sum_{f \in E} \frac{E[e^{\lambda_f l_j(f)} | x_e^{uv} = 0]}{e^{(1+\gamma_f)\bar{l}(f)\lambda_f}} \end{aligned}$$

Hence,

$$\hat{P}_j = P_e^{uv} \cdot \hat{P}_{j+1}^1 + (1 - P_e^{uv}) \cdot \hat{P}_{j+1}^0$$

and clearly, $\min\{\hat{P}_{j+1}^0, \hat{P}_{j+1}^1\} \leq \hat{P}_j$. The value of x_e^{uv} is set to the value for which \hat{P}_{j+1}^i is minimized, for $i = 0, 1$.

4 Assigning Weights for Controlled Flooding

In this section we consider a more dynamic approach of routing—that of controlled flooding. Flooding is a routing strategy that guarantees fast arrivals with minimal enroute computation at the expense of excessive bandwidth use. To limit the extent of flooding we adopt the controlled flooding scheme first proposed in [9]. Consider a network in which each link is assigned a *weight* (sometimes referred to as *cost*) for traversing it and every message carries with it a *wealth*. A message arriving at an intermediate node will be duplicated and forwarded along all outgoing links (except the one it came from) whose cost is lower than the message wealth. The cost of the link is then deducted from the duplicated-message wealth. Consider for example the network in figure 1 depicting a message with a wealth of 10 arriving at node 2.

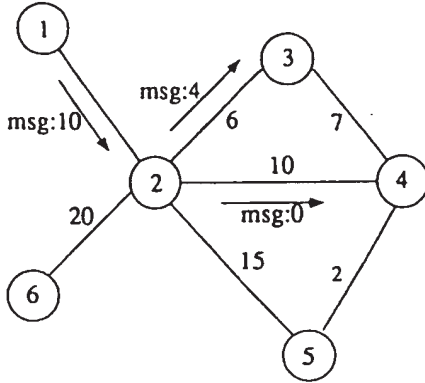


Figure 1: Example of controlled flooding

The link to node 3 has a cost of 6 associated with it resulting in a copy of the message with wealth 4 to be transmitted along that link. Similarly, a copy of the message with a wealth of 0 will arrive at node 4. Nodes 5 and 6 will not receive a copy of the message.

Since the controlled flooding scheme is a derivative of a flooding algorithm, it is impossible to assure that a message always arrives only at the nodes it is intended to. In particular, when used for point-to-point routing it is evident that more nodes than necessary might receive a message. In the above example, if the original message had arrived at node 2 with a wealth of 13 node 4 would have received two copies. Note also that there is no way for node 1 to send a message to node 4 without node 3 also receiving it. Clearly, different weight assignments may change the pattern of flooding.

The problem is to assign the link costs so as to achieve best performance. To that end a figure of merit is defined which is proportional to the (average) number of nodes that will receive every message. An optimal weight assignment is one that minimizes the figure of merit. To formalize our discussion let the network be represented by the graph $G(V, E)$ with $|V| = n$ and $|E| = m$, let the length of a path in the network be defined as the sum of the weights of the edges of the path, and let the shortest path between two nodes be the path with minimal length. Then, it is shown in [9] that for an assignment to be optimal, the following requirements (referred to as *optimality requirements*) must hold for every vertex (node) r :

- For every vertex $v \in V$, the shortest path from r to v is unique.
- For any two vertices $u, v \in V$, the length of the shortest path from r to u is different from the length of the

shortest path from r to v .

Assignments that satisfy the above requirements are called *good*. An assignment is good with respect to r if all shortest paths from r satisfy the above requirements. Let us assume without loss of generality that the weights assigned are all positive integers.

Let $[1 \dots R]$ denote the range of numbers from which weights are drawn and let n denote the number of nodes in the network. If $R = 2^{|E|}$, it is easy to find a good assignment [9]. For example, assigning 2^i as the weight of edge e_i assures that any two different paths will have different lengths. However, because the length of the path is carried by every message it is desirable to reduce R as much as possible.

We present two methods for constructing good assignments such that R is polynomial in n . In the first method the communication is restricted to a spanning tree T of the graph. This is done by assigning infinite weight to edges that are not in the tree. Denoting the tree edges by e_1, \dots, e_i, \dots , the algorithm is recursively defined as follows. Let v_i be a leaf of T , let u_i be its neighbor in the tree, and let e_i be the edge connecting u_i and v_i .

1. Compute (recursively) a good assignment for the tree $T - v_i$.
2. Extend the good assignment from $T - v_i$ to T .

We assume inductively that a good assignment was computed in Step 1. Step 2 can be implemented by checking all the values in the range $1 \dots R$ and finding one that satisfies the requirements for a good assignment. Obviously, a good value for e_i exists if R is large enough. The next lemma bounds the value of R .

Lemma 4.1 *If $R \geq n^2$, then there exists a good assignment.*

Proof: Since a good assignment was computed for $T - v_i$ at Step 1, any value assigned to e_i will complete a good assignment with respect to v_i . The number of distinct values that e_i cannot assume is at most $(n-1)(n-2)$: for each vertex $r \in T - v_i$, the distance from r to v_i should be different from the distance from r to any other vertex, and thus, there can be at most $n-2$ forbidden values (with respect to r), and the claim follows. \square

The complexity of the weight assignment algorithm is $O(n^3)$ since each step can be implemented in $O(n^2)$ time. For each vertex $v_i \in V$, a table of all its distances to the other vertices is maintained and for each node all the forbidden values in the range $[1 \dots n^2]$ are marked. One of the unmarked numbers is chosen arbitrarily for e_i . Then, the tables of all other nodes are updated.

2A.4.7

0176

The above assignment, being tree based, makes no use of many of the network links. The second assignment, which we present next, has the property that the whole network participates in the communication. We present two algorithms; the first is a randomized one that lends itself to distributed computation because the weight for each edge is chosen independently of the other edges. This algorithm generates a good assignment with high probability. The second algorithm is deterministic, and the weights are chosen from a smaller range than in the randomized algorithm.

Our main tool in the randomized case is the *Isolating Lemma* of Mulmuley, Vazirani and Vazirani [10]. A set system (S, F) consists of a finite set S of elements, $S = \{x_1, \dots, x_n\}$, and a family F of subsets of S , $F = \{S_1, \dots, S_k\}$. Let a weight w_i be assigned to each element of S . The weight of a subset is defined to be the sum of the weights of its elements.

Lemma 4.2 (Isolating Lemma) *Let $R \geq n$ and let (S, F) be a set system whose elements are assigned integer weights chosen uniformly and independently from the range $[1 \dots R]$. Then, $\text{Prob}[\text{There is a unique minimum (maximum) weight set in } F] \geq 1 - \frac{n}{R}$.*

(Note: the lemma in its original form in [10] was proven for $R = 2n$ but actually holds for all $R \geq n$). \square

We start by proving that the following randomized process will generate a good assignment with high probability. Let a weight for each edge be chosen randomly and uniformly from the range $[1 \dots R]$.

Lemma 4.3 *For $R \geq n^4$ the probability that an assignment is good is at least $\frac{1}{2}$.*

Proof: Let A_{ij} be the event the shortest path between nodes v_i and v_j is not unique. Then $A = \cup_{i,j} A_{ij}$ is the event indicating the existence of at least one pair of nodes with non-unique shortest path between them. For each pair of nodes v_i and v_j let the set system F be the set of all paths connecting them. From the isolating lemma we have that the shortest path between them will be unique with probability at least $1 - \frac{n}{R}$, or, $\text{Prob}[A_{ij}] \leq \frac{n}{R}$. Hence, $\text{Prob}[A] \leq \sum_{i,j} \text{Prob}[A_{ij}] \leq \binom{n}{2} \cdot \frac{n}{R}$.

Let B_{ijk} represent the event that nodes v_i , v_j , and v_k form a bad triplet, namely that the length of the shortest path between v_i and v_k equals that between v_j and v_k . $B = \cup_{i,j,k} B_{ijk}$ then represents the existence of at least one bad triplet in the network. In a way similar to the above we get $\text{Prob}[B] \leq \binom{n}{3} \cdot \frac{n}{R}$.

Finally, $A \cup B$ is the event indicating that the requirements are not met, and thus

$$\text{Prob}[\text{good assignment}] \geq 1 - \text{Prob}[A] - \text{Prob}[B]$$

$$\geq 1 - \frac{n^2(n-1)}{2R} - \frac{n^2(n-1)(n-2)}{6R}$$

For $R \geq n^4$, the right handside exceeds $\frac{1}{2}$. \square

The last lemma provides us with a randomized distributed algorithm for constructing a good assignment. The probability of failure can be made arbitrarily small by increasing the value of R .

Notice that this method does not ensure that every edge participates in at least one shortest path. This can be fixed by forcing the weight assignment so that the BFS tree resulting from the weight assignment is also a BFS tree in the underlying graph without weights. To that end assign weights to the edges according to any of the above described algorithms and then add the value $n \cdot R$ to each weight. Now every edge takes part in at least one shortest path.

Next we show how a good assignment can be constructed deterministically. One way would be to derandomize the above randomized process. Notice that the proof of Lemma 4.1 actually implies that every partial assignment that does not violate the optimality requirements can be completed to a good assignment. We can thus assign weights to the edges one-by-one ensuring at every step that none of the requirements is violated.

A better way of doing this is by the following algorithm that constructs a good assignment with $R = n^3$ (compared with n^4). Initially, every edge e_i is assigned weight $n^4 \cdot 2^i$. The weights of the edges are then changed one-by-one to fit into the range $[1 \dots R]$ while maintaining the goodness of the assignment. At each step, the weight of the heaviest edge is changed.

Lemma 4.4 *If $R \geq n^3$, a good assignment can be constructed.*

Proof: The invariant which is maintained at the end of each step is that the assignment remains good. This is true initially. Let w_i be the new weight assigned to edge e_i at step i , where e_i connects vertices x and y . We prove that w_i can be fitted into the range $[1 \dots R]$ by bounding the number of forbidden values for w_i and showing that at least one permitted number exists. Let l_{uv} denote the value of the shortest distance between vertex u and vertex v when edge e_i is removed from the graph (l_{uv} might be infinite).

To maintain goodness we must accommodate both optimality requirement. We first show how to maintain the uniqueness of the shortest path between every pair of vertices. Let r and v be a pair of vertices, and assume without loss of generality that $l_{rx} < l_{ry}$. (They cannot be equal by the invariant). If the removal of edge e_i from the graph leaves

vertices r and v in different connected components, then any value can be chosen for w_i with respect to r and v . Assume this is not the case. Since edge e_i had the largest weight in the graph (i.e., $n^4 \cdot 2^i$), the shortest path from r to v cannot contain edge e_i and l_{rv} is the value of the shortest distance from r to v . Hence, to maintain the uniqueness of the shortest path requirement, it is enough that

$$l_{rv} \neq l_{rx} + w_i + l_{yv}.$$

(Notice that the shortest path will remain unique even if it contains edge e_i , because of the uniqueness of the shortest paths from r to x and from y to v). This condition generates at most $n - 1$ forbidden values for w_i with respect to every vertex r in the graph, or $n(n-1)$ forbidden values altogether.

Let us now show how the second requirement of optimality is maintained. Let r , u and v be a triplet of vertices. Again, notice that if the removal of edge e_i from the graph leaves vertex r in one connected component, and vertices u and v in a different connected component, then any value can be chosen for w_i with respect to r , u and v . The same holds if the removal of e_i leaves y separated from r , u , and v . Assume this is not the case. It follows from the above discussion that the shortest distance from r to u is either l_{ru} , or $l_{rx} + w_i + l_{yu}$. Similarly, the shortest distance from r to v is either l_{rv} , or $l_{rx} + w_i + l_{yv}$.

By the invariant,

$$l_{ru} \neq l_{rx} + w_i + l_{yu} \quad \text{and} \quad l_{rx} + w_i + l_{yv} \neq l_{rx} + w_i + l_{yv}.$$

Hence, to maintain the second requirement of optimality, it is enough that

$$l_{rv} \neq l_{rx} + w_i + l_{yv}$$

and

$$l_{ru} \neq l_{rx} + w_i + l_{yu}.$$

These two conditions add at most $2 \cdot \binom{n-1}{2}$ forbidden values for w_i with respect to every vertex r in the graph, for a total of $2n \cdot \binom{n-1}{2}$.

Altogether, the number of forbidden values for w_i is $n(n-1)(n+1) < n^3$, and the lemma follows. \square

Note that the initial assignment ($e_i = n^4 \cdot 2^i$) is chosen to ensure that every edge is treated exactly once, and when it is treated it does not participate in any shortest path unless it is a bridge.

The complexity of the algorithm is $O(n^3m)$ since each step can be implemented in $O(n^3)$ time. Every vertex $v_i \in V$ maintains a table with all its shortest distances to the other vertices; it then marks all the forbidden values in the range $[1 \dots n^3]$. One of the unmarked numbers is chosen arbitrarily for e_i . Then, the tables of all other vertices are updated.

The reason why the range can be made smaller in the deterministic case is that it is enough to ensure at each step that

there is one good value, whereas in the randomized case, one has to ensure success with high probability.

A desirable property of a routing scheme is having the traffic be evenly distributed among the edges. Unfortunately, this is the drawback of routing with random weights. The following example shows that with high probability this scheme does not yield a balanced load.

Let the load on an edge be defined as the number of shortest paths that contain it, and consider a graph made of two cliques of size k that are interconnected by two edges, e_1 and e_2 . The weight for each edge is chosen uniformly and independently from the range $[1 \dots R]$. In each clique, the distribution of the weights is uniform and thus, if the weights of e_1 and e_2 are not close to one another, most of the traffic between the two cliques would go through the edge with smaller weight. Since this event will happen with high probability, the communication would not be balanced with high probability.

5 Conclusion

In this paper we examined several routing strategies for fast modern packet switching networks. The relevant characteristic of these networks is the inability to make elaborate routing decisions while packets are being switched. At the switching speeds being considered, looking up a table whose size is proportional to the number of network nodes is considered too costly.

These requirements limit the number of applicable routing strategies. The simplest and most natural strategy is to use fixed routing schemes in which the route between every pair of source-destination nodes is fixed in advance. The problem would then be to find a set of routes so that network resources are utilized as evenly as possible. Two such strategies are analyzed in this paper: routing along trees and routing along paths. For both cases polynomial algorithms are devised. We show that in both cases no network link remains unused but that routing along paths is likely to be a better strategy from load balancing standpoint.

Deviating from the fixed routing scheme we analyze a controlled flooding scheme in which every message essentially floods the networks but the extent of its flooding can be controlled by link weights. We provide a polynomial algorithm to compute these weights but show that the scheme cannot guarantee a good balance of load.

2A.4.9

Acknowledgement

We would like to thank Noga Alon for many helpful discussions on this paper and in particular for his help in analyzing the algorithm of Section 2.1.

References

- [1] A. Ephremides, "The routing problem in computer networks," in *Communications and Networks* (I. Blake and H. Poor, eds.), pp. 299-324, New York: Springer Verlag, 1986.
- [2] M. Schwartz and T. Stern, "Routing techniques used in computer communication networks," *IEEE Trans. on Communications*, vol. COM-28, pp. 539-555, April 1980.
- [3] P. Green, "Computer communications: Milestones and prophecies," *IEEE Communications*, pp. 49-63, 1984.
- [4] I. Cidon and I. Gopal, "Paris: An approach to integrated high-speed private networks," *International Journal of Digital and Analog Cabled Systems*, vol. 1, pp. 77-86, April-June 1988.
- [5] J. Turner, "Design of a broadcast packet switching network," *IEEE Trans. on Communications*, vol. COM-36, pp. 734-743, June 1988.
- [6] H. Siegel, *Interconnection Networks for Large-Scale Parallel Processing: Theory and Case Studies*. Lexington, MA: Lexington Books, 1984.
- [7] L. Fratta, M. Gerla, and L. Kleinrock, "The flow deviation method: An approach to store and forward communication network design," *Networks*, vol. 3, no. 2, pp. 97-133, 1973.
- [8] R. Gallager, "A minimum delay routing algorithm using distributed computation," *IEEE Trans. on Communications*, vol. COM-25, pp. 73-85, January 1977.
- [9] O. Lesser and R. Rom, "Routing by controlled flooding in communication networks," in *Proceedings of IEEE Infocom '90*, (San Francisco, California), pp. 910-917, IEEE, June 1990.
- [10] K. Mulmuley, U. Vazirani, and V. Vazirani, "Matching is as easy as matrix inversion," *Combinatorica*, vol. 7, no. 1, pp. 105-113, 1987.
- [11] J. Hopcroft and R. Karp, "An $n^{5/2}$ algorithm for maximum matching in bipartite graphs," *Siam J. Computing*, vol. 2, pp. 225-231, 1973.
- [12] S. Even, *Graph Algorithms*. New York: Computer Science Press, 1979.
- [13] M. Garey and D. Johnson, *Computers and Intractability*. San Francisco: W.H. Freeman and Company, 1979.
- [14] D. Angluin and L. G. Valiant, "Fast probabilistic algorithms for hamiltonian circuits and matchings," *Journal of Computer and System Sciences*, vol. 18, pp. 155-193, 1979.
- [15] J. Spencer, *Ten Lectures on the Probabilistic Method*. Philadelphia, Pennsylvania: SIAM, 1987.
- [16] P. Raghavan, "Probabilistic construction of deterministic algorithms: Approximating packing integer programs," *Journal of Computer and System Sciences*, vol. 37, pp. 130-143, October 1988.



Boeing and Panthesis Complete SWAN Transaction

Business Wire; New York; Jul 22, 2002; Business Editors & Aerospace Writers;

NAICS:336411 NAICS:336413 NAICS:336414 Duns:00-925-6819

Start Page: 1

Companies: Boeing Co Ticker:BA Duns:00-925-6819 NAICS:336411 NAICS:336413
NAICS:336414

Abstract:

IRVINE, Calif.--(BUSINESS WIRE)--July 22, 2002--The Boeing Co. and Panthesis Inc., today announced that they have completed a transaction that gives Boeing an equity stake in Panthesis and provides Panthesis with an exclusive right to commercialize Boeing's Small-world Wide Area Networking (SWAN) technology.

Based in Bellevue, Wash., Panthesis, was established in 2001 to develop and commercialize innovative software technology. Its co-founders, current Chief Development Officer Dr. Fred Holt and Chief Technology Officer Virgil Bourassa, are both former employees of The Boeing Co., where they co-invented SWAN technology while working in the Mathematics and Computing Technology unit of the Boeing Phantom Works R&D division.

Full Text:

Copyright Business Wire Jul 22, 2002

IRVINE, Calif.--(BUSINESS WIRE)--July 22, 2002--The Boeing Co. and Panthesis Inc., today announced that they have completed a transaction that gives Boeing an equity stake in Panthesis and provides Panthesis with an exclusive right to commercialize Boeing's Small-world Wide Area Networking (SWAN) technology.

SWAN technology was originally developed by Boeing to allow multiple geographically dispersed people to conduct collaborative meetings and engineering design reviews in real time.

"SWAN is a revolutionary technology that can be used to enhance numerous computing, networking and communications functions," said Linda Magnotti, CEO of Panthesis. "The sophisticated mathematics and software architecture underlying SWAN technology can provide reliable server-less communication for communities anywhere in the world."

Magnotti added that Panthesis is currently focusing its development efforts on providing the bandwidth multiplication needed for use in massive multi-player online games, real-time online auctions, content distribution and other large-scale, unlimited online collaborations.

Based in Bellevue, Wash., Panthesis, was established in 2001 to develop and commercialize innovative software technology. Its co-founders, current Chief Development Officer Dr. Fred Holt and Chief Technology Officer Virgil Bourassa, are both former employees of The Boeing Co., where they co-invented SWAN technology while working in the Mathematics and Computing Technology unit of the Boeing Phantom Works R&D division.

"Because Panthesis clearly has the expertise for adapting SWAN technology to a broad range of potential applications, we were confident in giving them the exclusive right to commercialize this technology in the global marketplace," explained Gene Partlow, vice president of Boeing's Intellectual Property Business.

The potential for this agreement was created through Boeing's Chairman's Innovation Initiative, which promotes the development of new business ventures based on entrepreneurial ideas from employees. While some ideas are developed into spin-off companies, others are spun into Boeing business units for further development or, like SWAN, into the Intellectual Property Business for other types of business transactions.

Panthesis is currently seeking investment capital to support company expansion and market penetration, and is engaged in developing relationships with key customers in the online auction and gaming markets.

The Boeing Co., with headquarters in Chicago, is the world's leading aerospace company and the No. 1 U.S. exporter. It is the largest manufacturer of satellites, commercial jetliners and military aircraft, and it provides a full range of lifecycle support for these and other products. The company is also a global market leader in missile defense, human space flight and launch services. Boeing capabilities also include financial services and advanced information and communications systems.

Reproduced with permission of the copyright owner. Further reproduction or distribution is prohibited without permission.

Microsoft Boosts Accessibility to Internet Gaming Zone With Latest Release

PR Newswire; New York; Apr 27, 1998;

Start Page: 1

Dateline: Washington

Companies: Microsoft Corp

Abstract:

REDMOND, Wash., April 27 /PRNewswire/ -- Microsoft Corp. (Nasdaq: MSFT) today released its latest update for the Microsoft(R) Internet Gaming Zone (<http://www.zone.com/>), featuring support for Netscape 4.0 and the latest versions of Microsoft Internet Explorer. The new version makes the Zone accessible to the majority of Internet users. With this new version, the Zone also introduced the new Zone Rating System, which allows game players to determine how they fare against other players. Chess and Age of Empires(R) will be the first games with the Zone Rating System, and new games are scheduled to be added to the system in the coming weeks.

The Zone is a collective place for gamers to play today's best games against others for free. Players have a wide variety of games to choose from -- including parlor games like Hearts and Chess, and action and strategy games like Jedi Knight: Dark Forces II, Age of Empires and the Fighter Ace(TM) online multiplayer game, the site's first premium game designed specifically for massive multiplayer gaming via the Internet. Furthermore, visitors can navigate through the site before downloading the Zone software required for game play.

Full Text:

Copyright PR Newswire - NY Apr 27, 1998

Industry: COMPUTER/ELECTRONICS; INTERNET MULTIMEDIA ONLINE

Netscape Support and Player Rating System Featured in Newest Version

Of the Leading Internet Gaming Site

REDMOND, Wash., April 27 /PRNewswire/ -- Microsoft Corp. (Nasdaq: MSFT) today released its latest update for the Microsoft(R) Internet Gaming Zone (<http://www.zone.com/>), featuring support for Netscape 4.0 and the latest versions of Microsoft Internet Explorer. The new version makes the Zone accessible to the majority of Internet users. With this new version, the Zone also introduced the new Zone Rating System, which allows game players to determine how they fare against other players. Chess and Age of Empires(R) will be the first games with the Zone Rating System, and new games are scheduled to be added to the system in the coming weeks.

"We believe online gaming is all about social interaction with a large and active community," said Ed Fries, general manager of the games group at Microsoft. "So we're very pleased that this new version of the Zone provides access for virtually everyone online."

Already home to nearly 1.5 million online gamers, the Zone has more than 7,500 simultaneous users at peak times -- and is gaining new registered members at the rate of one every 20 seconds.

The Zone is a collective place for gamers to play today's best games against others for free. Players have a wide variety of games to choose from -- including parlor games like Hearts and Chess, and action and strategy games like Jedi Knight: Dark Forces II, Age of Empires and the Fighter Ace(TM) online multiplayer game, the site's first premium game designed specifically for massive multiplayer gaming via the Internet. Furthermore, visitors can navigate through the site before downloading the Zone software required for game play.

In addition to Netscape 4.0 support and the Zone Rating System, the newest version of the Zone also features a new, streamlined interface, which reduces download times and makes getting into a game even easier. The Zone further assists its members with improved help and chat features.

Variety and Popularity of Games Drive Growth

The Zone offers a popular variety of classic card and board games such as Spades, Bridge and Backgammon. In fact, Spades has grown to become the most popular game on the Zone with peak usage of more than 2,000 players. In the past year, the Zone's lineup of CD-ROM games with free matchmaking has expanded rapidly with the addition of such popular Microsoft games as Age of Empires and Flight Simulator 98, and other top titles such as Jedi Knight: Dark Forces II from LucasArts Entertainment Co., Quake II from id Software and Scrabble from Hasbro Interactive, a unit of Hasbro Inc. These additions have brought the total number of games available for play on the Zone to 32. The Zone also recently announced support for upcoming Tom Clancy titles Rainbow Six and Dominant Species from Red Storm Entertainment.

The Internet Gaming Zone has served Internet gamers since October 1995. In May 1996, Microsoft acquired Electric Gravity Inc., the original designer of the Internet Gaming Zone. The Internet Gaming Zone offers free membership with three components: free classic card and board games, free matchmaking for retail games, and access to premium games designed exclusively for the Zone (connect-time charges may apply). Most recently, Microsoft launched Fighter Ace, a World War II aerial combat premium game designed specifically for the Internet in which more than 100 players can dogfight in a single flight arena.

Founded in 1975, Microsoft is the worldwide leader in software for personal computers. The company offers a wide range of products and services for business and personal use, each designed with the mission of making it easier and more enjoyable for people to take advantage of the full power of personal computing every day.

For online product information:

Microsoft Web site: <http://www.microsoft.com/>

Microsoft Internet Gaming Zone Web site: <http://www.zone.com/>

NOTE: Microsoft, Age of Empires and Fighter Ace are either registered trademarks or trademarks of Microsoft Corp. in the United States and/or other countries. Other product and company names herein may be trademarks of their respective owners. SOURCE Microsoft Corp.

Reproduced with permission of the copyright owner. Further reproduction or distribution is prohibited without permission.

Microsoft Announces Launch Date for UltraCorps, Its Second Premium Title For The Internet Gaming Zone

PR Newswire; New York; May 27, 1998;

Start Page: 1

Dateline: Washington

Companies: Microsoft Corp

Abstract:

REDMOND, Wash., May 27 /PRNewswire/ -- Microsoft Corp. (Nasdaq: MSFT) today announced plans to launch UltraCorps, its second premium online-only game for the Microsoft(R) Internet Gaming Zone (<http://www.zone.com/>), on June 25. The game is currently in open beta testing. Players can join the free beta by going to the Zone and proceeding to the UltraCorps link in the Strategy Games section. More than 3,500 players have participated in the beta so far. Microsoft also plans to spotlight two additional premium online-only titles for the Zone, plus the latest Fighter Ace(TM) online multiplayer game upgrade, at the Electronics Entertainment Expo (E3) trade show, May 28-30 in Atlanta (Booth 4420 in West Hall, Georgia Congress Center).

UltraCorps, developed by VR-1 Inc., is a turn-based strategy game that pits thousands of players against each other for domination of the universe. Players command one of 14 alien races, develop new technologies and weapons, dispatch fleets to colonize other planets, and manage resources to maintain their growing empires. Social interaction is a key component of the game as players form alliances, draw up treaties or taunt their enemies. As a turn-based game, it is well-suited to Internet play because it can challenge thousands of players without latency issues.

"UltraCorps is a galactic game of chess that forces gamers to outthink their opponents each day when they go online," said Adam Waalkes, product unit manager for the Zone team at Microsoft. "The Zone is the perfect platform to deliver UltraCorps to gamers because the size and scope of the game is a great match for our large community of players."

Full Text:

Copyright PR Newswire - NY May 27, 1998

Industry: COMPUTER/ELECTRONICS; INTERNET MULTIMEDIA ONLINE

'Oblivion,' Asheron's Call and Fighter Ace Upgrade Among Other Premium Titles

To Be Showcased at 1998 Electronics Entertainment Expo

REDMOND, Wash., May 27 /PRNewswire/ -- Microsoft Corp. (Nasdaq: MSFT) today announced plans to launch UltraCorps, its second premium online-only game for the Microsoft(R) Internet Gaming Zone (<http://www.zone.com/>), on June 25. The game is currently in open beta testing. Players can join the free beta by going to the Zone and proceeding to the UltraCorps link in the Strategy Games section. More than 3,500 players have participated in the beta so far. Microsoft also plans to spotlight two additional premium online-only titles for the Zone, plus the latest Fighter Ace(TM) online multiplayer game upgrade, at the Electronics Entertainment Expo (E3) trade show, May 28-30 in Atlanta (Booth 4420 in West Hall, Georgia Congress Center).

UltraCorps, developed by VR-1 Inc., is a turn-based strategy game that pits thousands of players against each other for domination of the universe. Players command one of 14 alien races, develop new technologies and weapons, dispatch fleets to colonize other planets, and manage resources to maintain their growing empires. Social interaction is a key component of the game as players form alliances, draw up treaties or taunt their enemies. As a turn-based game, it is well-suited to Internet play because it can challenge thousands of players without latency issues.

"UltraCorps is a galactic game of chess that forces gamers to outthink their opponents each day when they go online," said Adam Waalkes, product unit manager for the Zone team at Microsoft. "The Zone is the perfect platform to deliver UltraCorps to gamers because the size and scope of the game is a great match for our large community of players."

The arrival of Microsoft's second premium game on the Zone will cap its latest string of 1998 milestones, including the recent addition of support for Netscape Communicator 4.0, surpassing 1.5 million registered members, and its recent mark of more than 8,600 simultaneous users.

"Oblivion" Will Let Gamers Blow Opponents to Smithereens on the Zone

"Oblivion," current code name for a space-action premium game that is scheduled to arrive on the Zone late in 1998, combines detailed 3-D accelerated graphics, fluid motion and rich sound with the intellectual challenge of a strategy game. Players can engage hundreds of others online in territorial team wars, amid endless permutations of roles, missions and challenges. "Oblivion" is being developed by Microsoft Research.

More than 30 unique user-controlled spacecraft and space stations are modeled with lifelike textured exteriors and articulated parts. A panorama of cosmic phenomena includes planets, stars, black holes and wormholes rendered in graphic detail, accompanied by unearthly stereo sounds ranging from the din of asteroid impacts to the scream of failing force fields.

Asheron's Call: An Epic Online Adventure

Asheron's Call(TM) online multiplayer game, which is scheduled to arrive on the Zone in early 1999, draws together thousands of players within a dynamic, 3-D online world. Players can create truly unique characters, with varied combinations of visual appearance, attributes and skill sets. The setting for the game is a 24-by-24-mile island with all types of terrain, including mountain glaciers, desert wastelands, swamps and subterranean dungeons. The game immerses players in an intense fantasy role-playing environment where they must choose to compete against or cooperate with thousands of other real players. An extensive system of allegiance and influence greatly enhances social interaction. The story line in Asheron's Call evolves dynamically over time based on the decisions and actions of the Asheron's Call community. The game is being developed by Turbine Entertainment Software.

Fighter Ace Upgrade Set to Take Flight

Fighter Ace, a premium World War II aerial combat game that allows hundreds of players to dogfight simultaneously in a single arena, is scheduled to get new features later this summer. These include new terrain with greater geographic diversity; a new layout featuring airfields grouped farther apart so gamers can group and coordinate attacks; heavy bombers for flying missions against enemy installations; military, industrial and civilian ground targets; support for force-feedback joysticks; and improved anti-aircraft weapons.

Free Classic Games, Retail Matchmaking Continue

The Zone also offers free software and matchmaking for a variety of popular classic card and board games such as Spades, Bridge and Backgammon. In fact, Spades has grown to become the most popular game on the Zone, with concurrent usage at peak times of more than 2,100 players. In the past year, the Zone's lineup of CD-ROM games with free matchmaking has expanded rapidly with the addition of new

Microsoft games such as Outwars(TM) and Monster Truck Madness(R) 2 racing simulation, and other new titles such as Star Wars(R) Rebellion from LucasArts Entertainment Co., Quake II from id Software and SORRY! from Hasbro Interactive, a unit of Hasbro Inc. The lineup will continue to expand as Microsoft has recently announced relationships with Red Storm Entertainment and MicroProse Inc. to bring some of their new titles to the Zone.

Evolution of the Zone Continues

The Internet Gaming Zone has served Internet gamers since October 1995. In May 1996, Microsoft acquired Electric Gravity Inc., the original designer of the Internet Gaming Zone. The Internet Gaming Zone offers free membership with three components: free classic card and board games, free matchmaking for retail games, and access to premium games designed exclusively for the Zone (connect-time charges may apply).

Founded in 1975, Microsoft is the worldwide leader in software for personal computers. The company offers a wide range of products and services for business and personal use, each designed with the mission of making it easier and more enjoyable for people to take advantage of the full power of personal computing every day.

NOTE: Microsoft, Fighter Ace, Asheron's Call and Monster Truck Madness are either registered trademarks or trademarks of Microsoft Corp. in the United States and/or other countries. Star Wars is a registered trademark of Lucasfilm Ltd. Outwars is a trademark of Singletrac Studio, a GT Interactive Company. Other product and company names herein may be trademarks of their respective owners.

For online product information:

Microsoft Games Web site: <http://www.microsoft.com/games/> SOURCE Microsoft Corp.

Reproduced with permission of the copyright owner. Further reproduction or distribution is prohibited without permission.

DISTRIBUTED ALGORITHMS FOR SHORTEST-PATH, DEADLOCK-FREE ROUTING AND BROADCASTING IN ARBITRARILY FAULTY HYPERCUBES

Michael Peercy Prithviraj Banerjee

Center for Reliable and High-Performance Computing
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign

ABSTRACT

We present a distributed table-filling algorithm for point to point routing in a degraded hypercube system. This algorithm finds the shortest length existing path from each source to each destination in the faulty hypercube and fills the routing tables so that messages are routed along these paths. We continue with a distributed algorithm to fill tables used for broadcasting in a faulty hypercube. A novel scheme for broadcast routing with tables is proposed, and the algorithm required to fill the broadcast tables given the point to point routing tables is presented. In addition, we give the modifications necessary to make these algorithms ensure deadlock-free routing. We conclude with a quantitative and qualitative comparison of previously proposed reroute strategies with table routing, where the tables are filled with our algorithms.

1. INTRODUCTION

Message-passing multiprocessors such as hypercubes [1] consist of many processing nodes that interact by sending messages over communication channels between the nodes. However, the existence of a large number of components in such systems makes them vulnerable to failures. It is therefore extremely important to have schemes for message passing in such systems that can route messages efficiently in the presence of failures in nodes and links. This paper deals with message routing in hypercube networks.

Hypercubes today generally route messages using the *e-cube* routing algorithm [1]. This algorithm resolves the bit differences between the source s and the destination d from the lowest dimension to the highest and ensures the minimum length path. Numerous proposals and investigations have been made regarding routing and broadcasting in faulty hypercubes [2, 3, 4, 5, 6, 7]. Also, routing schemes which are designed to avoid network congestion can provide fault tolerant rerouting [8].

Previous schemes for routing in hypercubes have the following drawbacks. First, many of them are nonoptimal algorithms, i.e., they route messages through nonshortest paths, or fail to route messages even when paths exist. Also, algorithms that are close to optimal require very complicated algorithms whose hardware requirements are much greater than the *e-cube* routing hardware; really complicated algorithms might require microprogrammed control. Besides, the cost of the routing algorithm

This research was supported in part by the SDIO Innovative Science and Technology Office and managed by the Office of Naval Research under Contract N00014-88-K-0624 and in part by the Joint Services Electronics Program under Contract N00014-90-J-1270.

appears every time a message is routed.

In this paper we investigate reroute strategies based on routing tables [9, 10]. It should be noted that while routing tables have been proposed for loosely coupled distributed systems, they have not conventionally been used for hypercubes. The primary reason is that for fault-free hypercubes, the routing algorithms are so simple that messages can be routed optimally using minimal hardware. However, in the presence of faults, the routing algorithms become complex, and thus it is appropriate to reconsider table routing.

In distributed table routing, each node's communication coprocessor contains its own routing table. Let T_p be the routing table located at node p . T_p consists of N locations, where N is the number of processors ($N = 2^n$ in an n -dimensional hypercube). Location d of T_p , represented as $T_p[d]$, contains the dimension l for a message being routed to d to take from p . In this way a message moves from its source s to its destination d along a path (s, d) derived from routing tables in each intermediate node. Ideally the path (s, d) a message takes should succeed if at all possible and should be of minimum feasible length.

Note that n is the dimension of the hypercube of size $N = 2^n$. We are not suggesting table routing for massively parallel programming, so the N by $\log N$ size of the table should cause no concern. For instance, in a thousand processor hypercube, the required RAM is 1K by 12 (using one bit to indicate an unreachable or faulty node). Fast RAMs of this size are very inexpensive relative to the other hardware or microcode options provided by alternative fault-tolerant routing schemes. Also note that the time required for routing with tables is small and constant: the time to compute the outgoing link is the time of one memory read. Some serialization is possible among the input ports as they try to access the RAM, but, again, the RAM is fast compared to other transmission delay components. If this sequential access to the RAM is of concern, multiple copies of the routing table, or interleaving a single copy, are possible modifications.

The routing tables must be filled by some algorithm. Ideally, this algorithm would be designed to find the optimal possible paths in creating the routing tables. This algorithm needs to be run only when the configuration F of the system has changed. Researchers [11, 12, 13, 14] have presented algorithms which incrementally modify the routing tables in general networks when a change in the topology is recognized by nodes neighboring the change. Recently, Kim and Reed [15] investigated routing with tables produced by a central node using information delivered from local nodes. In this paper, we concentrate on globally designed distributed algorithms, specifically taking advantage of the hypercube topology, which entirely refill the tables after a fault or repair. Subsequently, the routing tables work

independently, routing messages along the shortest paths until the configuration changes again and the system needs to run the table-filling algorithm once more.

In this paper we propose a distributed table-filling algorithm (TFA) which determines the routing table for each node s at node s itself. This was developed from a centralized TFA in which the system host finds shortest paths using Dijkstra's algorithm [16]. In our distributed algorithm each node gathers information about the hypercube configuration F exclusively through communication with its nearest neighbors. After presenting the distributed table-filling algorithm, we propose a broadcasting technique which utilizes tables. In this scheme, a broadcast message would carry in its header the fact that it is a broadcast and the original source of the broadcast. Each node s along the broadcast paths would then lookup in a broadcast routing table on which links the broadcast should be routed from s . We give another distributed algorithm which fills the broadcast routing tables from the original routing tables. Next we provide a method to ensure that the paths found by our table-filling algorithms are deadlock-free. By splitting links in dependency cycles into two virtual links, we can route upon the links so that no cycles exist in the new configuration, and thus avoid deadlock. We present an algorithm to modify the tables produced by the distributed table-filling algorithm so that the routes are free of the possibility of deadlock.

2. DISTRIBUTED TABLE-FILLING ALGORITHM

2.1. Distributed Algorithm

The key to a distributed table-filling algorithm (TFA) is that the shortest path from a node s to a node d is the extension of the shortest path found by one of the neighbors of s . In our TFA, each node cycles through its n neighbors, exchanging tentative routing tables, until these tables cease to change. The distributed TFA D is given below.

ALGORITHM $D(s)$ (in parallel on all nodes s)

```

Let the current dimension  $l$  be  $n-1$ 
Repeat until table unmodified in  $n$  consecutive dimensions
  Exchange routing tables with neighbor along dimension  $l$ 
  For each destination in own table
    If path through neighbor shorter than presently recorded path
      Or dimension  $l$  lower than initial dimension of presently
        recorded path
      Place new path, identified as dimension  $l$  and length, in table
    Endif
  Endfor
  Decrement (mod  $n$ ) dimension  $l$ 
Enduntil
For next  $n$  dimensions
  Inform neighbor along dimension  $l$  that own table is done
  Decrement (mod  $n$ ) dimension  $l$ 
Endfor

```

To facilitate the proof of the operation of this algorithm, we define a *sweep* as one set of consecutive iterations from dimension $n-1$ through dimension 0. That is, a sweep consists of one iteration in each dimension.

THEOREM 2.1: Algorithm D terminates with the shortest paths.

PROOF of shortest paths: By induction.

BASE CASE: All paths of length 1 (all paths to nearest neighbors) are shortest paths and are discovered in the first sweep.

ASSUME: After k sweeps every node s has all shortest paths of length k that it sources.

THEN: Because every subpath of a shortest path is itself a shortest path, a shortest path of length $k+1$ from a node s to some destination d includes a shortest path of length k from a neighbor of s to destination d . In sweep $k+1$ every node s receives the length k path information from each of its neighbors. Therefore, every shortest path of length $k+1$ sourced by each node s is determined by appending the appropriate dimension onto the shortest path of length k sourced by a neighboring node. After $k+1$ sweeps every node s has all shortest paths of length $k+1$ that it sources.

PROOF of termination: To see that algorithm D does not terminate until all shortest paths are found, consider the path sourced by node s that would be the last discovered by algorithm D ; say that this path P is of length L . P necessarily contains L different paths, to different destinations, all sourcing at s . In fact, there is exactly one path of each length from 1 to L which is a (shortest) subpath of the (shortest) path P . By the induction step above, it is clear that each sweep advances the maximum length of the discovered shortest paths by 1. Therefore, in each sweep a new shortest path, which happens to be a subpath of P , is discovered by s , and algorithm D does not terminate until the last path P is found. The termination condition given in algorithm D follows when we recognize that the phase of the sweep does not matter.

Algorithm D can find more than one link of a path in each sweep. Thus the number of sweeps actually required to find all the shortest paths is a configuration-dependent value between 1 and $N-1$. The former value is for a fault-free cube and the latter value is an upper bound for a worst-case completely connected cube where the maximum length shortest path is $N-1$ links long. Thus the algorithm has a time complexity of $O(N^2 \log N)$. However, the possibility of so poor a performance is minimal, and most faulty configurations will give a time complexity much closer to that in a perfect cube: $O(N \log N)$.

D is constructed to use local information only, and builds its paths on the near end. By adding links of lowest possible dimension to the source end of its current paths, D ensures *e-cube*-like routing in a fault-free hypercube. In algorithm D we start with the highest dimension ($n-1$) and move down through the dimensions; after dimension 0 we move to dimension $n-1$ again. The reason we decrement through dimensions in D as opposed to any other order of taking dimensions is that in n iterations, that is, n exchanges of information, all nodes in a fault-free hypercube fill their routing tables with the *e-cube* paths. It is an interesting result that with only one table exchange in each dimension, every node in a fault-free cube fills its routing table perfectly. This makes the cost of implementing table-routing very small as far as filling tables in perfect cubes. However, in general faulty hypercube configurations, the tables are filled in few more iterations.

Figure 1 shows a cube with a failure in node 5, and Figure 2 shows the last 4 of the 6 steps required to fill the routing tables with the optimum paths. After the first three steps, every path which is an *e-cube* path has been identified; this set of *e-cube* paths is shown in Figure 2(a). We now note two points in Figure 2: how the shortest path is selected and how the increasing-in-

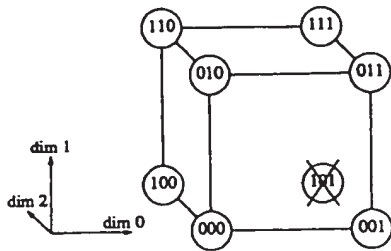


Figure 1. 3-cube with Fault in Node 5

dimension path is selected. By the third iteration, we have not yet filled $T_4[7]$ (row 4, column 7 in the routing matrix). In iteration four, TFA D places a 2 in that location. The path from node 4 to node 7 which the matrix after iteration four dictates is [4 0 1 3 7], a length 4 path. But we can see in Figure 1 that a length 2 path exists: namely [4 6 7]. This is corrected in iteration five, as we swap along dimension 1 and node 4 learns from node 6 of a shorter path to node 7. The other point to note is exemplified by $T_4[1]$, the first step on a distance 3 path. After three iterations, this location is unfilled. In each subsequent iteration, a lower first dimension of the path is found. Thus algorithm D finds the path with the lowest first dimension out of the set of shortest paths.

2.2. Extension to Partial Failures

The above description of Algorithm D handles link failures and total node failures. However, in the case of partial node failures, i.e., loss of the main processor but continuing operation of the widowed communication coprocessor, the TFA D so far does not operate ideally. In fact, as given above, D would be unable to use the routing table in a functioning, but widowed, coprocessor, and would view such a node as totally faulty when routing paths. Minor modifications to algorithm D correct this deficiency.

If a coprocessor's table is independently receivable from neighboring nodes, then another node can run Algorithm D for a widowed coprocessor. We modify Algorithm D to allow the claiming of a widowed coprocessor w by an active processor v . If there is a viable path from v to w , and w is as yet unclaimed, then v claims w . Since D forces synchronization by its very nature, at any one time w is approached only on one dimension, so the uniqueness of v is assured. After v claims w , v then executes D as though it were being executed on w (call this $D(w)$), starting with the next dimension in the iterations of the algorithm. Of course, since v must also execute its own version of D , the time it takes for v to perform each step is doubled. The claiming of widowed coprocessors can be recursive, i.e., v may need to claim w' which lies on the other side of w from v . Then, in executing $D(w')$, v must communicate through w .

The reasons the routing table must be writable from other nodes are twofold. First, messages which are sent to a claimed w must be fooled into going to v . That is, to claim w , v must write a path to itself into the location w in $T_v[w]$. Second, when the algorithm is complete and all routing tables are finalized, v must write into T_w the routing table it determined running $D(w)$.

s	d							
	0	1	2	3	4	5	6	7
0	+	0	1	0	2	.	1	0
1	0	+	0	1	0	.	0	1
2	1	0	+	0	1	.	2	0
3	0	1	0	+	0	.	0	2
4	2	.	1	.	+	.	1	.
5	+	.	.
6	1	.	2	0	1	.	+	0
7	0	.	0	2	0	.	0	+

2(a): After third iteration (after dims. 2, 1, 0)

s	d							
	0	1	2	3	4	5	6	7
0	+	0	1	0	2	.	1	0
1	0	+	0	1	0	.	0	1
2	1	0	+	0	1	.	2	0
3	0	1	0	+	0	.	0	2
4	2	2	1	2	+	.	1	2
5	+	.	.
6	1	2	2	0	1	.	+	0
7	0	2	0	2	0	.	0	+

2(b): After fourth iteration (dim. 2)

s	d							
	0	1	2	3	4	5	6	7
0	+	0	1	0	2	.	1	0
1	0	+	0	1	0	.	0	1
2	1	0	+	0	1	.	2	0
3	0	1	0	+	0	.	0	2
4	2	2	1	1	+	.	1	1
5	+	.	.
6	1	1	2	0	1	.	+	0
7	0	2	0	2	0	.	0	+

2(c): After fifth iteration (dim. 1)

s	d							
	0	1	2	3	4	5	6	7
0	+	0	1	0	2	.	1	0
1	0	+	0	1	0	.	0	1
2	1	0	+	0	1	.	2	0
3	0	1	0	+	0	.	0	2
4	2	2	1	1	+	.	1	1
5	+	.	.
6	1	0	2	0	1	.	+	0
7	0	2	0	2	0	.	0	+

2(d): Complete routing table (after dim. 0)

Figure 2. Algorithm D on 3-cube with Fault in Node 5

3. BROADCASTING WITH TABLES

We now propose table routing methods for one-to-all broadcast. To implement broadcast, our routing table requires an additional $n+1$ bits of information per word. Let us describe these additional bits per word as a separate table U_s , with location b represented by $U_s[b]$. The broadcast algorithm to use this table is as follows: a 1 in bit l of $U_s[b]$ means that, if s receives a broadcast message which originates at b , it should copy that message and send it along dimension l . Therefore, an adequate header for a broadcast message would be an indicator that it is a broadcast and the address of the original source of the broadcast. An algorithm is required to fill the table U in each node. This algorithm executes after D and determines broadcast paths from the optimal length paths found by D . We call this broadcast table-filling algorithm (BTFA) D^B . Before giving the algorithm, we introduce the concept of the link partition.

For a node s , given an original broadcast source b and a list of destination nodes $M_s(b)$, we define the *link partition* of the set of destinations $M_s(b)$. $M_s(b)$ is the union of the disjoint sets $M_s(b)_0, M_s(b)_1, \dots, M_s(b)_{n-1}, M_s(b)_n$, where $M_s(b)_l = \{d: d \in M_s(b) \text{ AND } T_s[d]=l\}$. $M_s(b)_l$ is that set of destinations in $M_s(b)$ for which the first dimension of the paths from s to those destinations is l . $M_s(b)_n$ contains the single element s , the current node. The partition $M_s(b)_l$ is determined from the routing table T_s . In fact, $M_s(s)_l$ is the inverse of $T_s[d]$; the former maps l to d , and the latter maps d to l .

For example, given as row 3 in Figure 2(d), we have the routing table for node 3 in a 3-cube with a faulty node 5: $T_3 = (0 \ 1 \ 0 \ 3 \ 0 \ . \ 0 \ 2)$, where 3= n signifies the current node and the period signifies an unreachable node. Using this routing table, $M_3(3)_0 = \{3\}$, $M_3(3)_1 = \{7\}$, $M_3(3)_2 = \{1\}$, and $M_3(3)_3 = \{0, 2, 4, 6\}$.

$M_s(b)$ in each node s is the set of destinations to which s is expected to forward a broadcast message from b , and the link partition gives the set of destinations each neighbor of s is expected to forward. The table $U_s[b]$ will have a 1 in every bit l for which $M_s(b)_l$ is nonempty. Node s would then forward a broadcast by (1) recognizing the original source of the broadcast b , and (2) forwarding it along each and every link l for which $U_s[b]_l = 1$.

ALGORITHM $D^B(s)$ (for every node s)

```

 $M_s[s] \leftarrow$  all viable destinations
 $l \leftarrow s$ 
Determine link partition of  $M_s[s] =$ 
     $M_s[s]_0, M_s[s]_1, \dots, M_s[s]_{n-1}, M_s[s]_n$ 
 $l \leftarrow 0$ 
While  $l \neq \emptyset$  or there are viable sources  $s$  has not heard from
    Send  $(i, M_s[i]_l)$ , for all  $i \in l$  and  $M_s[i]_l \neq \emptyset$ , along link  $l$ 
    For all  $i \in l$  and  $M_s[i]_l \neq \emptyset$ 
         $U_s[i]_l \leftarrow 1$ 
         $M_s[i] \leftarrow M_s[i] - M_s[i]_l$ 
        if  $M_s[i] = M_s[i]_n$  then  $U_s[i]_n \leftarrow 1; l \leftarrow l - \{i\}$ 
    Endfor
    Receive  $(j, M_s[j])$ , for all  $j \in J$  (for some set  $J$ ), along link  $l$ 
    For all  $j \in J$ 
        Determine link partition of  $M_s[j] =$ 
             $M_s[j]_0, M_s[j]_1, \dots, M_s[j]_{n-1}, M_s[j]_n$ 
        If  $M_s[j] = M_s[j]_n$  then  $U_s[j]_n \leftarrow 1; J \leftarrow J - \{j\}$ 
    Endfor
 $l \leftarrow l \cup J$ 

```

$l \leftarrow l + \text{mod } n \ 1$
Endwhile

THEOREM 3.1: The tables filled by algorithm D^B will broadcast using shortest paths.

PROOF: Broadcast paths are exactly those shortest paths found by algorithm D .

For simplicity of presentation, our BTFA D^B shows the broadcast table bits $U_s[b]_l$ modified with each iteration l . We could write all of $U_s[b]$ once the partition of $M_s(b)$ is done. $U_s[b]_l = 1$ if and only if $M_s(b)_l \neq \emptyset$. We use the n^{th} partition set and the n^{th} bit of the broadcast table as a convenience to imply that any broadcast message forwarded by node s should be received and absorbed by node s as well. Note also that, in subsequent steps of the algorithm, the determination of the link partition of newly received sets can be computed for each l from the initial link partition as $M_s[j]_l = M_s[j] \cap M_s[s]_l$.

BTFA $D^B(s)$ is very efficient. The amount of work involved in the send, receive, and partition steps is proportional to the length of the lists. The algorithm's complexity is $O(N^2 \log N)$, but, as with algorithm D , this order is reached only at degenerate worst cases. On a perfect cube the algorithm runs in time $O(N \log N)$, only executing one iteration for each dimension.

Figure 3 shows an example of the operation of this algorithm on node 0 of a fault-free 3-cube. Each horizontal block in Figure 3(a) is one iteration of BTFA D^B . The first block is the initial state, with all destinations reachable from node 0 partitioned by the first links in their respective paths. In each iteration k , node 0 sends along link $l = k \text{ mod } n$ the lists of all destinations the paths to which node 0 routes on link l . Then the current partitions are modified to show the removal of the just-sent lists, and newly received lists are partitioned and included as current. When the list of nodes in a current partition b includes only node 0, signifying that all other nodes have been taken care of, then the partition is removed from further consideration.

We also give an example of D^B executing on a faulty cube. Recall the single-fault hypercube of Figure 1; in this 3-cube, node 5 is faulty. Row 3 of Figure 2(d) is T_3 , the table used in computing the initial link partition. Figure 4 shows the operation on D^B from the viewpoint of node 3 and the broadcasting table at node 3 which results. To illustrate the basic rule behind the operation of D^B , we describe what happens when node 3 receives $(6, \{1, 3\})$. This information tells D^B that node 6 expects its broadcasts to reach nodes 1 and 3 through node 3. Node 3 then determines how it reaches nodes 1 and 3. It sends to node 1, according to the routing table, using dimension 1. Thus the algorithm waits until dimension 1 is dictated by execution, and sends $(6, \{1\})$, telling the neighbor along dimension 1 that node 6 expects to communicate with node 1 through that neighbor. Node 6 expects to communicate also with node 3 through node 3, but that path is trivial and no further computation is necessary.

Executing an all-to-all broadcast, in which each node sends the same message to every other node, could be accomplished in one of two ways. The messages could be broadcast independently and asynchronously, mutually contending for limited link resources, or the messages could be broadcast synchronously. Specifically for the synchronous case, when an all-to-all broadcast is required, the nodes could execute a variant of algorithm D^B . Every node would thus communicate along the same

k	send		receive		current partitions $M_0(i)_l$				
	i	$M_0(i)_k$	j	$M_0(j)_l$	i	l=3	l=2	l=1	l=0
-					0	0	4	2,6	1,3,5,7
0	0	{1,3,5,7}	1	{0,2,4,6}	0	0	4	2,6	.
1	0	{2,6}			1	0	4	.	.
	1	{2,6}	2	{0,4}	2	0	4	.	.
			3	{0,4}	3	0	4	.	.
2	0	{4}			0	0	.	.	.
	1	{4}			1	0	.	.	.
	2	{4}			2	0	.	.	.
	3	{4}			3	0	.	.	.
			4	{0}	4	0	.	.	.
			5	{0}	5	0	.	.	.
			6	{0}	6	0	.	.	.
		7	{0}	7	0	.	.	.	

3(a): Operation of D^B at node 0 in 3-cube

broadcast routing table $U_0(b)_l$				
b	l=3	l=2	l=1	l=0
0	1	1	1	1
1	1	1	1	0
2	1	1	0	0
3	1	1	0	0
4	1	0	0	0
5	1	0	0	0
6	1	0	0	0
7	1	0	0	0

3(b): Broadcast routing table for node 0

Figure 3. Example of Algorithm D^B on Fault-Free 3-cube

dimension at the same time a composite message of individual broadcasts. Since in fact algorithm D^B is an all-to-all broadcast of dynamic messages (the destination lists), a synchronous all-to-all broadcast would take exactly as many steps as D^B . The broadcast routing tables filled by D^B would serve either the synchronous or asynchronous all-to-all broadcast.

4. DEADLOCK AVOIDANCE

The standard routing algorithm *e-cube* is the primary algorithm for routing messages in hypercubes today. Three principal reasons explain this preference for *e-cube*: (1) it is easy to implement, (2) it spreads messages evenly throughout the network, and (3) it prevents deadlock. The prevention of deadlock can be assured if and only if there are no cycles in the channel dependency graph [17]. The reason that no cycles exist in the *e-cube* algorithm is that every channel is dependent only on channels of higher dimension. No dependency can go backwards in dimension. Thus deadlock is impossible in *e-cube*.

However, in a hypercube containing faulty links (channels), extra precautions must be taken to ensure deadlock-free routing. We adapt the method given in [17] to avoid deadlock. Essentially this method consists of defining virtual channels along the physical links. Each virtual channel is distinguished

k	mod n	send		receive		current partitions $M_3(i)_l$				
		i	$M_0(i)_k$	j	$M_3(j)_l$	i	l=3	l=2	l=1	l=0
-						3	3	7	1	0,2,4,6
0	3	{0,2,4,6}		2	{1,3,7}	2	3	7	1	.
1				0	{3,7}	0	3	7	.	.
	2	{1}		1	{3,7}	1	3	7	.	.
	3	{1}				2	3	7	.	.
2						3	3	7	.	.
	0	{7}				0	3	.	.	.
	1	{7}				1	3	.	.	.
	2	{7}				2	3	.	.	.
0				6	{1,3}	6	3	.	1	.
				7	{1,3}	7	3	.	1	.
1	6	{1}				6	3	.	.	.
2	7	{1}				7	3	.	.	.
				4	{3}	4	3	.	.	.

4(a): Operation of D^B at node 3 in 3-cube with faulty node 5

broadcast routing table $U_3(b)_l$				
b	l=3	l=2	l=1	l=0
0	1	1	0	0
1	1	1	0	0
2	1	1	1	0
3	1	1	1	1
4	1	0	0	0
5	0	0	0	0
6	1	0	1	0
7	1	0	1	0

4(b): Broadcast routing table for node 3

Figure 4. Example of Algorithm D^B on Single-Fault 3-cube

from the others on one link by a unique address and its own queue. The virtual channels can be time multiplexed on the physical links with the use of these queues. By maintaining a strict ordering of these virtual channels, we can show that the new channel dependency graph is free of cycles, and thus the network is free of deadlock.

As an example of the configuring of virtual channels to avoid deadlock, we show Figures 5 and 6. Figure 5 gives a configuration of a hypercube with directed links (one each way between processors) which has a possible deadlock configuration. The links which may cause deadlock are extracted from Figure 4 and given with explicit unidirectionality in Figure 6(a). We call such a set of links in a hypercube a *loop*. A loop contains two cycles, one in each direction on the loop.

We represent the possibility of deadlock with the channel dependency graph of Figure 6(b). The vertices of this graph are the links from Figure 6(a); the edges represent the (nontransitive) dependencies. The vertices are labeled with a unique link label. Each link is identified by an ordered pair (s,l) , where s is the

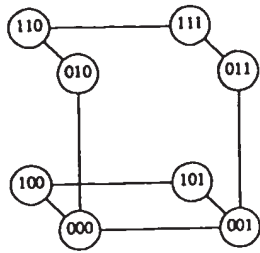


Figure 5. Faulty Configuration of 3-cube Inducing Cycles

node sourcing the link and l is the dimension of the link. For example, the link from node 0 to node 2 is represented in the right cycle of Figure 6(b) by the vertex $(0,1)$.

To prevent deadlock, we split each of the links in a cycle into two virtual links which share the same physical communication line but have different queues (Figure 6(c)). Then we address the virtual links in the cycle with labels of the form xyz , where $x \in \{0,1\}$, y increases around the cycle, and z is a unique cycle identifier. Thus we can break the cycle by permitting dependencies only in increasing order of the virtual link addresses (Figure 6(d)). To force the dependencies to be acyclic, we give each processor node in each cycle the label yz , where the node sources virtual links $0yz$ and $1yz$, and enforce the following message routing rule at each source or intermediate node: if the current node label is less than the destination node label, route along the higher addressed link; if the current node label is greater than the destination node label, route along the lower addressed link. Note that, in our example, links 15a and 15b are not used.

In general, after running a TFA we have routing tables for which the dependency graph contains cycles. The problem facing us is that these routing tables are distributed among the nodes, and it would be very inefficient to detect cycles from the local tables. However, we can globally broadcast all the routing tables so that each processor has the complete routing matrix. This could be a large amount of communication, but the following theorem and corollary allows us to reduce it.

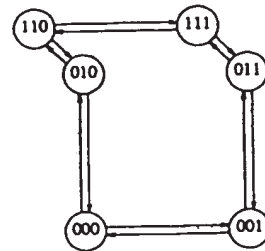
THEOREM 4.1: For any path found by TFA D for configuration F , all subpaths of that path are themselves paths found by TFA D .

PROOF: Follows from the way paths are determined during routing.

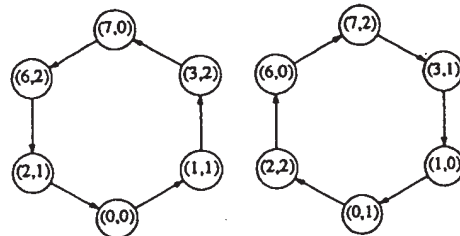
THEOREM 4.2: The dependence graph for a routing matrix found by TFA D can be constructed with information on paths of length 2 only.

PROOF: Since every path (i.e., every string of channel dependencies) is composed of paths of length 2, the paths of length 2 capture all the consecutive dependencies. The transitive dependencies can be ignored in finding cycles.

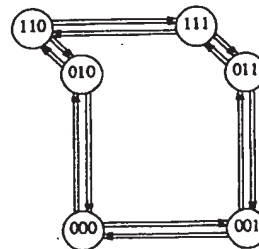
We only need to communicate the paths of length 2 throughout the network to provide full channel dependency information. Paths of length 2 can be derived from an abridged routing matrix which contains source-destination pairs no more than distance 2 from each other. Thus each node need only communicate its table for distance 1 and distance 2 destinations. There are



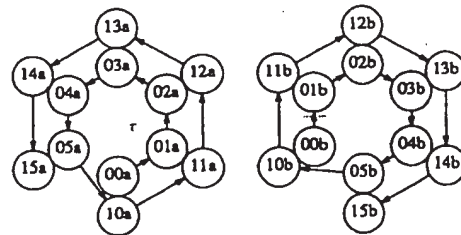
6(a): Loop extracted from faulty cube



6(b): Channel dependency graph of cycles



6(c): Virtual links to break cycles



6(d): Dependency graph of virtual links

Figure 6. Example of Breaking Cycles in Faulty 3-cube

$C_f + C_g$ of these in each node, where C_f denotes the number of ways to choose k items from n items.

Once each node has complete information of the total routing matrix, it can construct the channel dependency graph and find all cycles. An algorithm such as that given in [16] is used to find the cycles. Every node has the same information and, if each runs the same algorithm, each finds the same cycles. Then, by splitting each cycle into a spiral of virtual channels, the nodes remove dependencies from the graph. The nodes then modify their routing tables so that, given a destination, they indicate the correct virtual link to reach that destination without the possibility of deadlock.

We have not yet considered the impact of our cycle removal schemes on paths which deviate from the cycles. For example, in Figure 5 node 2 routes to node 5 along two links of the counterclockwise cycle included in the path [2 0 1 5]. We need to correctly identify which link (higher or lower) we should take to reach each destination. The correct link will be dependent on the last intermediate node in the cycle that the path routes through to reach its destination. From information of paths of length 2, we cannot construct each longer path, we cannot determine the last node for each path in the intersection of path and cycle, and we therefore can not tell from paths of length 2 whether to route each path along the higher or the lower virtual link in a cycle. We can correct this problem by passing complete routing tables along cycles, so that every source knows exactly how far along the cycle every path goes. The cycle address of the last node in the path-cycle intersection determines whether the higher or lower link is taken along the cycle.

Below is Algorithm DEADLOCK_FREE, which modifies the routing tables found by D to ensure the avoidance of deadlock in path selection. The hardware and encoding in the routing architecture at each node s must be altered to permit the addressing of multiple virtual links per physical link. In the presentation of DEADLOCK_FREE below, we simply show the routing table getting the virtual link address, i.e., $T_s[d] \leftarrow xz$. (We suppress the y from our notation xyz because the y implicitly refers to the current node s .)

ALGORITHM DEADLOCK_FREE {in each node s }

```

Run algorithm  $D$ 
Run algorithm  $D^B$ 
Do all-to-all broadcast of routing table contents for distance
  1 and 2
Construct channel dependency graph
Find all cycles using cycle detection algorithm
For each cycle  $z$  found which includes an outgoing link of  $s$ 
  Create two virtual channels  $0z$  and  $1z$  to replace instance of
  link in  $z$ 
Allocate and address one queue for each outgoing virtual channel
Exchange complete routing tables around each cycle to determine
  complete paths along each cycle
For each destination  $d$  with path  $(s d)$ 
  If  $T_s[d]$  is in some cycles
    Choose a cycle  $z$  which intersects  $(s d)$  along the greatest
    length
    Let  $p$  be the last node in the intersection of the path and  $z$ 
    If the label of  $p$  in cycle  $z$  is less than that of  $s$ 
      (denoted  $y$  in text)
         $T_s[d] \leftarrow 0z$ 
      Else

```

```

   $T_s[d] \leftarrow 1z$ 
Endif
Endif
Endfor

```

Two questions which arise are the following. Do we need to alter the broadcast routing tables? Will any part of algorithm DEADLOCK_FREE induce deadlock before the avoidance techniques are in place? The answer to both these questions is found in the single statement: deadlock cannot involve a single-link path. Deadlock involves a path acquiring one link and holding it while it awaits another. In single-link paths, once the first link is acquired, no waiting need be done: the path is complete, the message is sent, and the link is freed. Both algorithms D and D^B operate synchronously on single-link paths. Broadcast, also, generally occurs in single-link paths. If we wished to allow broadcasts to operate on multiple-length paths, we could simply rerun D^B after DEADLOCK_FREE, this time partitioning the destinations, and the broadcast table, among the virtual links; the rest of the algorithm D^B is unchanged.

5. PERFORMANCE OF TABLE ROUTING

We now compare the performance of table-routing under TFA D with another proposed reroute scheme, the adaptive scheme of Chen and Shin [2]. Their reroute method, which we will here refer to as *adaptive*, finds a path from source to destination by starting an *e-cube* path, then altering it as necessary when it is blocked by a fault. A tag is used to mark blocked and extra dimensions to prevent oscillation.

We compare these with two measures of performance applicable to reconfigured networks [18]. The reconfiguration strategy we use is that of process adoption [19]: an adjacent processor adopts the task running on a processor after it fails. The

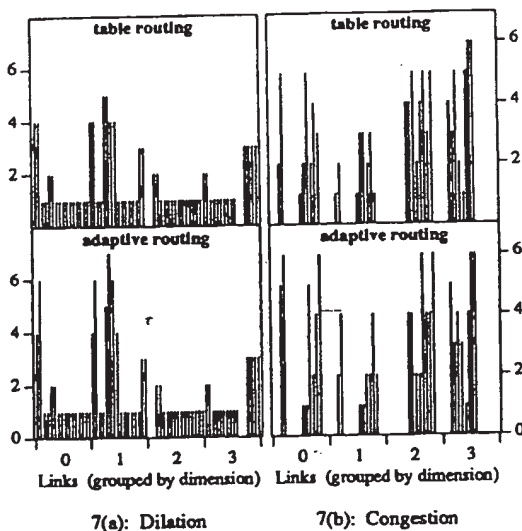


Figure 7. Comparison of Dilation and Congestion

first measure, called *dilation*, gives length in links of the logical replacement of a previously physical link. That is, a path between two adjacent processes in a fault-free cube may be mapped to a multiple-link path due to fault and subsequent reconfiguration. The second measure is called *congestion*. This measures the number of logical links which use each fault-free physical link in the faulty configuration.

Our example configuration F is one with four faulty nodes: node 0, node 5, node 6, and node 15. The process from node 0 is mapped to node 4, process 5 is mapped to node 13, process 6 to node 3, and process 15 to node 10. The results are shown in Figure 7.

Both the dilation and congestion measurements are a constant 1 across all links in a fault-free hypercube; every logical link is on exactly one physical link, and every physical link carries exactly one logical link. However, in our faulty hypercube example, with node 4 very far from other nodes in the network, the dilation and congestion measurements are quite high. The areas of the dilation and congestion histograms are equal; this area is essentially the number of physical links all the logical links use. The area for this four-fault hypercube is 104 with the routes determined by algorithm *D* and 112 with the *adaptive* routing scheme, demonstrating the reduced system communication load due to the shorter paths of table routing.

6. CONCLUSIONS

We have introduced table routing in faulty hypercubes, demonstrating the power and ease of such a routing method. Our distributed algorithms have shown table routing to be not only possible, but preferable in faulty hypercubes. Our distributed table-filling algorithm *D* executes in $O(N^3 \log N)$ time in the very rare worst case. Generally, performance of *D* is of the order of $N^2 \log N$. We have also proposed the use of tables for broadcast in faulty hypercubes. The broadcast table-filling algorithm runs in $O(N^2 \log N)$ worst case time, with a general performance around $N \log N$.

We have shown the superior dilation and congestion measures of the shortest paths generated by *D* in faulty hypercubes and the minimal extra hardware and communication delay of table routing. Also we have presented a deadlock prevention scheme applied to distributed routing tables. To our knowledge, this is the first routing scheme that has been proposed for faulty hypercubes that is shortest-path and deadlock-free.

REFERENCES

- [1] H. Sullivan and T. R. Bashkow, "A large scale homogeneous fully distributed parallel machine," *Proc. 4th Symp. Computer Architecture*, pp. 105-117, Mar. 1977.
- [2] M. S. Chen and K. G. Shin, "Message routing in an injured hypercube," *Proc. 3rd Conf. Hypercube Concurrent Computers and Applications*, pp. 312-317, Jan. 1988.
- [3] T. C. Lee and J. P. Hayes, "Routing and broadcasting in faulty hypercube computers," *Proc. 3rd Conf. Hypercube Concurrent Computers and Applications*, pp. 346-354, Jan. 1988.
- [4] J. M. Gordon and Q. F. Stout, "Hypercube message routing in the presence of faults," *Proc. 3rd Conf. Hypercube Concurrent Computers and Applications*, pp. 318-327, Jan. 1988.
- [5] M. S. Chen and K. G. Shin, "Routing in the presence of an arbitrary number of faults in hypercube multicomputers," *4th Conf. Hypercube Concurrent Computers and Applications*, Mar. 1989.
- [6] Kasho, et al., "Distributed fault tolerant routing in hypercubes," *4th Conf. Hypercube Concurrent Computers and Applications*, Mar. 1989.
- [7] Al-Dhelaan and Bose, "Efficient fault-tolerant broadcasting algorithm for the hypercube," *4th Conf. Hypercube Concurrent Computers and Applications*, Mar. 1989.
- [8] E. Chow, H. Madan, J. Peterson, D. Grunwald, and D. Reed, "Hyperswitch network for the hypercube computer," *Proc. 15th Int. Symp. Computer Architecture*, pp. 90-99, May 1988.
- [9] D. A. Reed and R. M. Fujimoto, *Multicomputer Networks: Message-Passing Parallel Processing*. Cambridge, MA: MIT Press, 1987.
- [10] A. S. Tanenbaum, *Computer Networks*. Englewoods Cliffs, NJ: Prentice-Hall, Inc., 1981.
- [11] W. D. Tajibnapis, "A correctness proof of a topology information maintenance protocol for a distributed computer network," *Commun. of the ACM*, vol. 20, pp. 477-485, July 1977.
- [12] Gallager R. G., "A minimum delay routing algorithm using distributed computation," *IEEE Transactions on Communications*, vol. COM-25, pp. 73-85, Jan. 1977.
- [13] Segall A., "Advances in verifiable fail-safe routing procedures," *IEEE Transactions on Communications*, vol. COM-29, pp. 491-497, Apr. 1981.
- [14] E. M. Gafni and D. P. Bertsekas, "Distributed algorithms for generating loop-free routes in networks with frequently changing topology," *IEEE Transactions on Communications*, vol. COM-29, pp. 11-18, Jan. 1981.
- [15] C. Kim and D. A. Reed, "Adaptive packet routing in a hypercube," *Proc. 3rd Conf. Hypercube Concurrent Computers and Applications*, pp. 625-629, Jan. 1988.
- [16] N. Deo, *Graph Theory with Applications to Engineering and Computer Science*. Englewoods Cliffs, NJ: Prentice-Hall, Inc., 1974.
- [17] W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Transactions on Computers*, vol. C-36, pp. 547-553, May 1987.
- [18] J. Hastad, T. Leighton, and M. Newman, "Reconfiguring a hypercube in the presence of faults," *Proc. 19th ACM Symp. Theory of Computing*, pp. 274-284, 1987.
- [19] P. Banerjee, "Reconfiguring a hypercube in the presence of faults," *Proc. 4th Conf. Hypercube Concurrent Computers and Applications*, Mar. 1989.



Welcome to IEEE Xplore

- Home
- What Can I Access?
- Log-out

Your search matched 46 of 972916 documents.

A maximum of 46 results are displayed, 25 to a page, sorted by Relevance in descending order. You may refine your search by editing the current search expression or entering a new one the text box. Then click Search Again.

(((flood* or broadcast*) <near/3> routing and ((hypercub* or hypercube*))) Search Again

Tables of Contents

- Journals & Magazines
- Conference Proceedings
- Standards

Results: Journal or Magazine = JNL Conference = CNF Standard = STD

Search

- By Author
- Basic
- Advanced

1 **Distributed algorithms for shortest-path, deadlock-free routing and broadcasting in arbitrarily faulty hypercubes**
Peercy, M.; Banerjee, P.;
 Fault-Tolerant Computing, 1990. FTCS-20. Digest of Papers., 20th International Symposium , 26-28 June 1990
 Page(s): 218 -225

[Abstract] [PDF Full-Text (652 KB)] IEEE CNF

Member Services

- Join IEEE
- Establish IEEE Web Account
- Access the IEEE Member Digital Library
- Print Format

2 **Multi-level hypercube network**
Aboelaze, M.A.;
 Parallel Processing Symposium, 1991. Proceedings., Fifth International , 30 April-2 May 1991
 Page(s): 475 -480

[Abstract] [PDF Full-Text (428 KB)] IEEE CNF

3 **Cross-cube: a new fault tolerant hypercube-based network**
Haq, E.;
 Parallel Processing Symposium, 1991. Proceedings., Fifth International , 30 April-2 May 1991
 Page(s): 471 -474

[Abstract] [PDF Full-Text (280 KB)] IEEE CNF

4 **Distributed algorithms for shortest-path, deadlock-free routing and broadcasting in Fibonacci cubes**

A DISTRIBUTED RESTORATION ALGORITHM FOR MULTIPLE-LINK AND NODE FAILURES OF TRANSPORT NETWORKS

Hiroaki Komine, Takafumi Chujo, Takao Ogura, Keiji Miyazaki, and Tetsuo Soejima

Fujitsu Laboratories, Ltd.

1015 Kamikodanaka, Nakahara-ku, Kawasaki, 211, Japan

Abstract

Broadband optical fiber networks will require fast restoration from multiple-link and node failures as well as single-link failures. This paper describes a new distributed restoration algorithm based on message flooding. The algorithm is an extension of our previously proposed algorithm for single-link failure. It restores the network from multiple-link and node failures, using multi-destination flooding and path route monitoring. We evaluated the algorithm by computer simulation, and verified that it can find alternate paths within 0.5s whenever the message processing delay at a node is 5ms.

1. Introduction

There is an increasing dependency on today's communication networks to implement strategic corporate functions. User demands for high-speed and economical communications services lead to the rapid deployment of high-capacity optical fibers in the transport networks. At the same time, the demands for high-reliability services raise a network survivability problem. For example, if the network is disabled for one hour, up to \$6,000,000 loss of revenue can occur in the trading and investment banking industries [1]. As the capacity of the transmission link grows, a link cut results in more loss of services. Therefore, rapid restoration from failures is becoming more critical for network operations and management.

There have been many algorithms developed to restore networks, including centralized control [1] and distributed algorithms [2-4]. In centralized control, the network is controlled and managed from a central office. In distributed control, the processing load is distributed among the nodes and restoration is thus faster. However, more computation capability and high speed control data channels are required. Recently it has been possible to provide high performance microprocessors for digital cross-connect system (DCS). High capacity optical fibers enable high speed data transmission for OAM through overhead bytes, which is under study by CCITT.

The distributed algorithms proposed so far [2-4] are based on simple flooding [5]. When a node detects failure, it broadcasts a restoration message to adjacent nodes to find an alternate route. In the algorithm [2], a restoration message requests a spare DS-3 or STS-1 path and is sent through the path overhead of each spare path. To avoid congestion of the messages in this algorithm, a message in both the algorithms [3,4] requests a bundle of spare

paths and is sent through the section overhead of each link. Algorithm [3] finds the maximum capacity along an alternate route, and our algorithm [4] finds the shortest alternate route. As described in [4], our algorithm was faster. However these algorithms are designed to handle single-link failures, they cannot handle multiple-link or node failures.

In this paper, we first discuss the major issues that must be addressed in order to handle multiple-link and node failures in Section 2. Based on these consideration, we propose a new restoration algorithm using multi-destination flooding and path route monitoring. These are described in Section 3. For a node failure, the node which detected the failure sends a restoration message to the last N-consecutive nodes each logical path passed through. An alternate path is made between the message sender node and one of the multiple nodes specified in the message. Each node collects the identifier of these nodes, using a path route monitoring technique. The algorithm was evaluated by computer simulation for multiple-link failure as well as for node failure. The results will be described in Section 4.

2. Limitations of simple flooding

In this section, we review simple flooding and discuss its limitations to handle multiple-link and node failures. In principle, the distributed algorithms [2-4] based on simple flooding work as follows. When a link fails, the two nodes connected to the link detect the failure and try to restore the path. One node becomes the sender and the other becomes the chooser (Fig. 1). The sender broadcasts restoration messages to all links with spare capacity. Every node except the sender and the chooser respond by re-broadcasting the message. When the restoration message reaches the chooser, the chooser returns an acknowledgement to the sender. In this way, alternate paths are found. Message congestion caused by routing messages far away is avoided by limiting the number of hops.

These algorithms based on simple flooding [2-4] usually assume a single-link failure, but in reality, some links which go different nodes may be in the same conduit. Therefore, if the conduit is cut, many links fail at the same time [3]. This is the case of multiple-link failure. Fire or earthquakes can also damage a large number of nodes, so the restoration algorithm must be able to handle these situations.

Simple flooding can not handle multiple-link or node failures because of following problems.

403.4.1

CH2827-4/90/0000-0459 \$1.00 © 1990 IEEE

0459

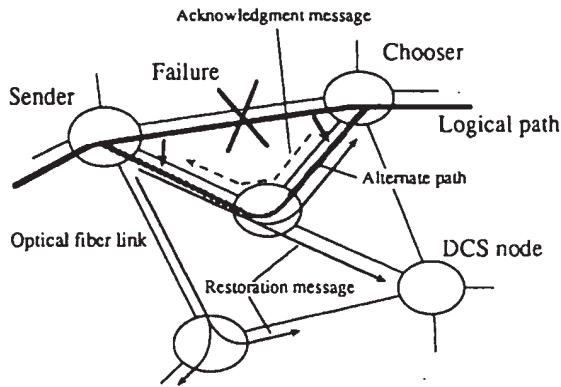


Fig. 1 Distributed restoration based on simple flooding

- Contention of spare capacity

In case of multiple-link failure, restoration messages coming from different nodes might contend for spare capacity on the same link. For example, if capacity is assigned to arriving messages in turn, the first message reserves the capacity. Whether or not the reserved capacity is later used for an alternate path, the reserved capacity is not released and therefore can not be assigned to another restoration message. Thus, the restoration ratio decreases.

- Fault location

Because the algorithms assume link failure, one of the two nodes connected to the failed link becomes the sender and the other becomes the chooser. However, for a node failure, there is a chooser and sender for each affected path. They are neighbors of the failed node and depend on the route of the paths. Each node detects failure by the loss of the signal on the link, and cannot distinguish between link or node failure.

The first problem could be alleviated by simple message cancelling. Spare capacity is assigned to restoration messages on a first-come, first-served basis. Assignment is cancelled when the message can not go forward due to hop limits or lack of capacity. During message flooding, cancel messages are sent to inform a node that a restoration message, which reserves spare capacity on a specific link, did not reach its destination and the served capacity of this link can be released for other restoration messages. Restoration messages are canceled immediately after reception if they are identical to messages already received, if the hop limit is reached, or if there is no more capacity at the node. In these cases, the unused capacity can be assigned to another restoration message.

Solving the second problem requires more sophisticated techniques and we propose a new distributed restoration algorithm in the following section.

3. Multi-destination flooding

To solve the fault location problem described above, we propose a new multi-destination flooding technique. We also propose path route monitoring which is essential to achieve multi-destination flooding.

3.1 Principle of multi-destination flooding

Simple flooding methods assume just one chooser. We extended this to allow multiple choosers as message destinations. When a node detects the loss of a signal from a link, the node can not tell whether the link or the node at the other end has failed. It sends a restoration message directed to the node which is the chooser in a link failure as well those that are choosers in a node failure. In Fig.2, for example, the link between nodes B and C fails, node B is the chooser for all affected paths, and nodes A and D are possible choosers for paths P1 and P2. If node B fails, nodes A and D become choosers for paths P1 and P2. The restoration message contains all choosers and the required capacity for each sender-chooser pair. The node which received the restoration message checks the destination field of the message, and if it is a chooser candidate, it returns an acknowledgment to the sender.

Thus, by extending simple flooding into multi-destination flooding, link or node failures do not have to be distinguished because there is always at least one chooser. Different messages are sent to the chooser candidates, but the same restoration message listing all candidates is sent towards all candidates. The number of restoration messages decreases and congestion is reduced.

Restoration processing consists of a broadcast phase, an acknowledgment phase, and a confirmation phase. To handle multiple failures, cancel processing is performed during the broadcast and acknowledgment phases.

The node states are sender, chooser, reserved tandem, and fixed tandem. The sender is the node which detected the failure. The chooser is the destination node of a restoration message. Chooser candidates set by the sender become choosers when they receive

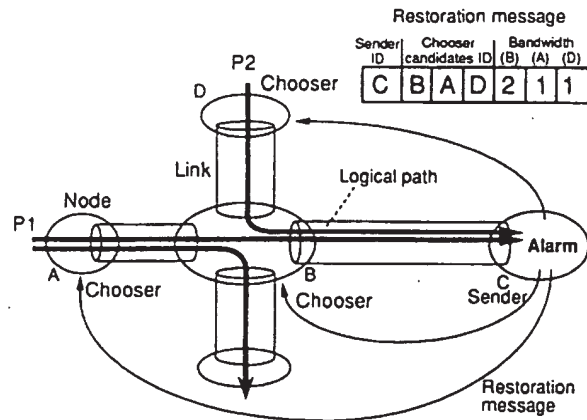


Fig. 2 Multi-destination flooding

a restoration message. The reserved tandem is a candidate node for alternate paths reserved by the restoration message. A received confirmation message of the sender turns a reserved tandem node into a fixed tandem node.

a) Broadcast phase

In the broadcast phase, the sender broadcasts restoration messages which reserve spare capacity in the network toward chooser candidates. A failure occurring on a link or node is detected by the next node on the path below the failure. This node becomes the sender. The sender looks up the chooser candidates and their capacities for the failed paths which were determined before by the path route monitoring described in the following section. The restoration message is then broadcast.

The restoration message contains the following information.

- 1) Message type : restoration, acknowledgment, confirmation, cancel
- 2) Message index
- 3) Sender ID
- 4) Chooser IDs (Multiple destination)
- 5) Required capacity of each sender-chooser pair
- 6) Reserved capacity
- 7) Hop count

The message index is set by the sender. It represents the number of flooding waves broadcast. The combination of the message index, the sender ID and chooser IDs is the Message ID. The required capacity is the capacity required between the sender and the various choosers. The reserved capacity is the capacity of the route taken by the restoration message.

The sender broadcasts the restoration message to all connected links except failed links and then waits for an acknowledgment from one of the choosers. Each node in the network except the sender and chooser receives a restoration message, and examines the hop count and the Message ID. If the hop count reaches the limit set by the sender, or a message with the same ID has arrived before, the node returns a cancel message to the link originating

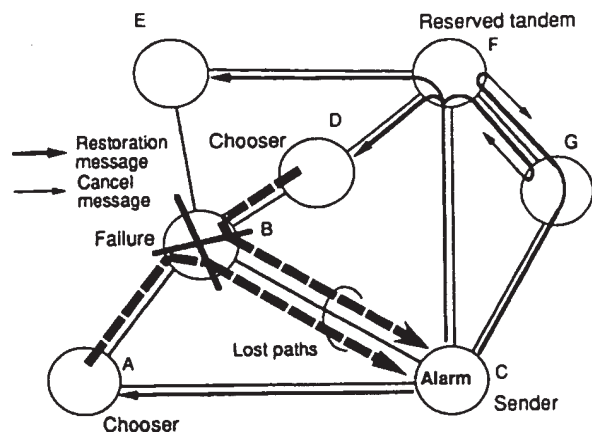


Fig. 3 Broadcast phase

the restoration message. Otherwise, the state of the node is set to reserved tandem. If spare capacity is available, a restoration message is broadcast. If the spare capacity of a link is insufficient, the reserved capacity is set to the spare capacity of the link. A node that finds its own node ID among the chooser IDs in the restoration message becomes the chooser. Figure 3 shows the broadcast phase when a failure has occurred at node B.

b) Acknowledgment phase

In the acknowledgment phase, the chooser sends an acknowledgment message to the sender. By the entries in the acknowledgment message, the sender is informed which chooser the acknowledgement message is from. If another restoration message with the same message ID arrives at the chooser, it is canceled.

A reserved tandem node which receives an acknowledgment message passes it back to the source of the corresponding restoration message. All other reserved spare capacity of this restoration message is canceled. Message flow during an acknowledgment phase is shown in Fig. 4.

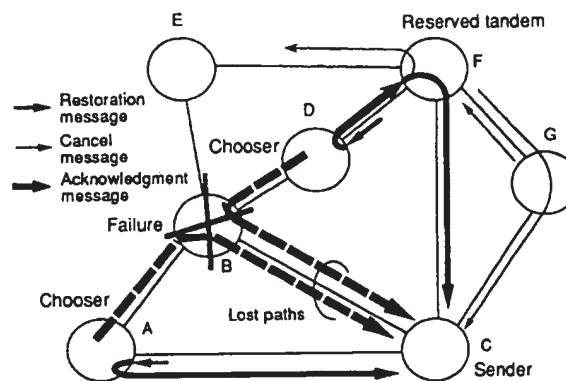


Fig. 4 Acknowledgment phase

c) Confirmation phase

When the acknowledgment message reaches the sender, a confirmation message is sent to the chooser. The reserved spares are switched over to alternate paths. If the sender received acknowledgment or canceled messages from all links it sent restoration messages to, and if the restoration of the failure is not completed, the sender increments the message index and attempts restoration from the broadcast phase again.

The reserved tandem node which received a confirmation message changes its status to fixed tandem and connects the reserved spares. In Fig. 5, node F has become fixed tandem, and the failed path between node D and node C is rerouted through the nodes D, F, and C. The other path which failed between node A and node C are also rerouted.

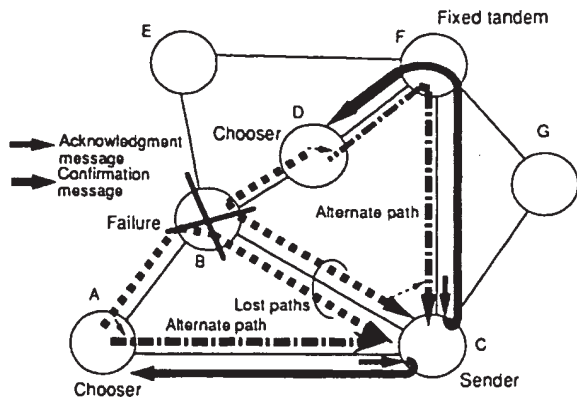


Fig. 5 Confirmation phase

3.2 Path route monitoring

For multi-destination flooding, each node must have route information on the paths passing through the node. One approach is to have the central office distribute such route information to all nodes. However, the routes are changing dynamically under customer control and nodes might receive inconsistent route information because updating route data takes time. We propose a path route monitoring method in which each node collects route information in real time.

The route information required at every node are the ID's of the last two consecutive nodes in every path before the node. This information is collected as follows. Node ID's are sent through assigned space in the path overhead. For every path going through a node, the data in the ID area is shifted and the ID of the node it is going through is written in. In this way, every node receives continuous and real-time route information.

4. Simulation

4.1 Simulation tool and conditions

We evaluated the ability of the algorithm to restore multiple-link and node failures using an event-driven network simulator [4,6] which works on the SUN3 workstation. We used the mesh network model shown in Fig. 6. This network consists of 25 nodes and 40 links. Each link length was generated at random, and the average link length is 184 km. Every link has 35 working paths.

We assumed a transmission speed of 64 kb/s. Messages were 16 bytes long, and the hop limit was 9. In a SONET frame structure, 64 kb/s for transmission speed means that one byte of overhead is used for message communications between nodes. The processing delay time from the arrival of a message to the end of the processing depends on the architecture of the DCS hardware. We assumed a 5 ms delay. This simulation does not include failure detection or crossconnection times.

4.2 Simulation results

Figure 7 shows a cumulative restoration ratio of node failure. The restoration ratio of the network is the ratio of restored to lost paths. For node failure, paths terminating at the failed node are not counted as lost paths because it is impossible to restore them.

We also simulated the algorithm for single-link failure. The result is shown in Fig. 7.

Figure 8 shows the cumulative restoration ratio in a multiple-link failure. There are many link combinations, but only one is shown. Failures between node N8 and N13, and one of the other links, occurred simultaneously on two links. The results indicate

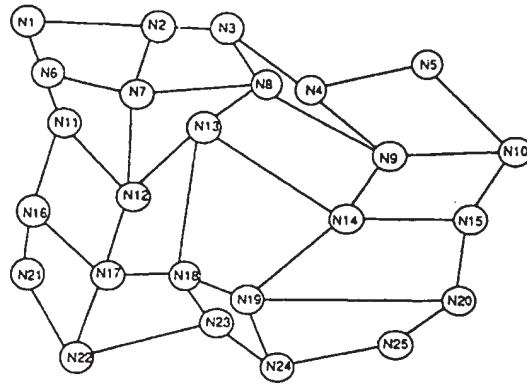


Fig. 6 Network model

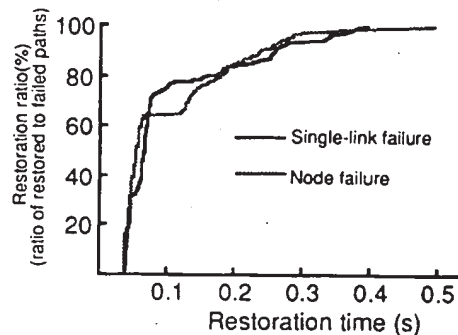


Fig. 7 Simulation results on single-link and node failure

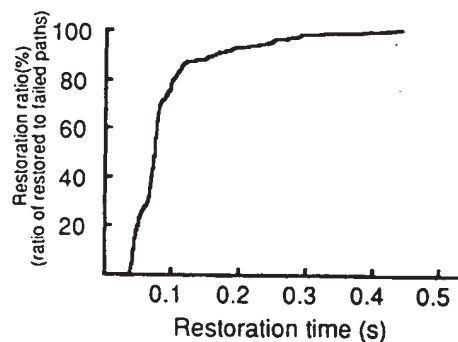


Fig. 8 Simulation result on multiple-link failure

that the proposed algorithm can handle multiple-link and node failure as well as single-link failure. All restorations are completed within 0.5s with message processing delay at the nodes being 5ms.

5. Conclusion

We pointed out problems associated with adapting a restoration algorithm based on flooding to recover from multiple-link and node failures. The main problem is to position the chooser nodes correctly. We proposed multi-destination flooding and path route monitoring. We simulated the algorithm with a mesh network and verified that the algorithm can handle multiple-link and node failures as well as single-link failures.

The message delay within a node depends on the architecture of the DCS and the processing load. The next step will be to analyze these delays and to include restoration time.

Acknowledgment

The authors thank Dr. Takanashi, Dr. Murano, and Mr. Yamaguchi of Fujitsu Laboratories Ltd., and Mr. Tokimasa of Fujitsu Ltd. for their encouragement and advice.

References

- [1] W. Falconer, "Services Assurance in Modern Telecommunications Networks," IEEE Communications Magazine, Vol. 28, No. 6, pp. 32-39, June 1990.
- [2] W. D. Grover, "The Selfhealing Network: A FAST DISTRIBUTED RESTORATION TECHNIQUE FOR NETWORKS USING DIGITAL CROSSCONNECT MACHINES", Globecom'87, pp. 28.2.1-28.2.6, Nov. 1987.
- [3] C. H. Yang and S. Hasegawa, "FITNESS: Failure Immunization Technology for Network Service Survivability", Globecom'88, pp. 47.3.1-47.3.6, Dec. 1988.
- [4] T. Chujo, T. Soejima, H. Komine, K. Miyazaki, and T. Ogura, "The Design and Simulation of an Intelligent Transport Network with Distributed Control", NOMS'90, pp. 11.4-1 - 11.4-12, Feb. 1990.
- [5] A. S. Tanenbaum, "Computer Networks", pp. 298-299, Prentice-Hall International, 1988.
- [6] T. Chujo, T. Soejima, H. Komine, K. Miyazaki, and T. Ogura, "The Modeling and Simulation of an Intelligent Transport Network with Distributed Control", ITU-COM'89, VII.1, pp. 343 - 347, Oct. 1989.



Welcome
United States Patent and Trademark Office

Help [FAQ](#) [Terms](#) [IEEE Peer Review](#)

[Quick Links](#)

[» Search Abst](#)

Welcome to IEEE Xplore®

- Home
- What Can I Access?
- Log-out

Tables of Contents

- Journals & Magazines
- Conference Proceedings
- Standards

Search

- By Author
- Basic
- Advanced

Member Services

- Join IEEE
- Establish IEEE Web Account
- Access the IEEE Member Digital Library

[Search Results](#) [[PDF FULL-TEXT 364 KB](#)] [PREV](#) [NEXT](#) [DOWNLOAD CITATION](#)

[Order Reuse Permissions](#)
[RIGHTS LINK](#)

A distributed restoration algorithm for multiple-link and node failures of transport networks

[Komine, H.](#) [Chujo, T.](#) [Ogura, T.](#) [Miyazaki, K.](#) [Soejima, T.](#)

Fujitsu Lab. Ltd., Kawasaki, Japan ;

This paper appears in: Global Telecommunications Conference, 1990, and Exhibition. 'Communications: Connecting the Future', GLOBECOM '90., IEEE

Meeting Date: 12/02/1990 - 12/05/1990

Publication Date: 2-5 Dec. 1990

Location: San Diego, CA USA

On page(s): 459 - 463 vol.1

Reference Cited: 6

Inspec Accession Number: 3976310

Abstract:

Fast restoration of broadband optical fiber networks from multiple-link and node failures as well as single-link failures, is addressed. A **distributed restoration algorithm** based on message flooding is described. The algorithm is an extension of a previously proposed algorithm for single-link failure. It restores the network from multiple-link and node failures, using multidestination flooding and path route monitoring. Computer simulation of the algorithm verified that it can find alternate paths within 0.5 s, whenever the message processing delay at a node is 5 ms

Index Terms:

[broadband networks](#) [optical links](#) [broadband optical fiber networks](#) [distributed restoration algorithm](#) [message flooding](#) [message processing delay](#) [multidestination flooding](#) [multiple-link failures](#) [node failures](#) [path route monitoring](#) [single-link failures](#) [transport networks](#)

Documents that cite this document

Select link to view other documents in the database that cite this one.

[Search Results](#) [[PDF FULL-TEXT 364 KB](#)] [PREV](#) [NEXT](#) [DOWNLOAD CITATION](#)

Copyright © 2004 IEEE — All rights reserved

On Four-Connecting a Triconnected Graph[†] (Extended Abstract)

Tsan-sheng Hsu
Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712-1188
tshsu@cs.utexas.edu

Abstract

We consider the problem of finding a smallest set of edges whose addition four-connects a triconnected graph. This is a fundamental graph-theoretic problem that has applications in designing reliable networks.

We present an $O(n\alpha(m, n) + m)$ time sequential algorithm for four-connecting an undirected graph G that is triconnected by adding the smallest number of edges, where n and m are the number of vertices and edges in G , respectively, and $\alpha(m, n)$ is the inverse Ackermann's function.

In deriving our algorithm, we present a new lower bound for the number of edges needed to four-connect a triconnected graph. The form of this lower bound is different from the form of the lower bound known for biconnectivity augmentation and triconnectivity augmentation. Our new lower bound applies for arbitrary k , and gives a tighter lower bound than the one known earlier for the number of edges needed to k -connect a $(k-1)$ -connected graph. For $k=4$, we show that this lower bound is tight by giving an efficient algorithm for finding a set of edges with the required size whose addition four-connects a triconnected graph.

1 Introduction

The problem of augmenting a graph to reach a certain connectivity requirement by adding edges has important applications in network reliability [6, 14, 28] and fault-tolerant computing. One version of the augmentation problem is to augment the input graph to reach a given connectivity requirement by adding a smallest set of edges. We refer to this problem as the

[†]This work was supported in part by NSF Grant CCR-80-23059.

smallest augmentation problem.

Vertex-Connectivity Augmentations

The following results are known for solving the smallest augmentation problem on an undirected graph to satisfy a vertex-connectivity requirement.

For finding a smallest biconnectivity augmentation, Eswaran & Tarjan [3] gave a lower bound on the smallest number of edges for biconnectivity augmentation and proved that the lower bound can be achieved. Rosenthal & Goldner [26] developed a linear time sequential algorithm for finding a smallest augmentation to biconnect a graph; however, the algorithm in [26] contains an error. Hsu & Ramachandran [11] gave a corrected linear time sequential algorithm. An $O(\log^2 n)$ time parallel algorithm on an EREW PRAM using a linear number of processors for finding a smallest augmentation to biconnect an undirected graph was also given in Hsu & Ramachandran [11], where n is the number of vertices in the input graph. (For more on the PRAM model and PRAM algorithms, see [21].)

For finding a smallest triconnectivity augmentation, Watanabe & Nakamura [33, 35] gave an $O(n(n+m)^2)$ time sequential algorithm for a graph with n vertices and m edges. Hsu & Ramachandran [10, 12] developed a linear time algorithm and an $O(\log^2 n)$ time EREW parallel algorithm using a linear number of processors for this problem. We have been informed that independently, Jordan [15] gave a linear time algorithm for optimally triconnecting a biconnected graph.

For finding a smallest k -connectivity augmentation, for an arbitrary k , there is no polynomial time algorithm known for finding a smallest augmentation to k -connect a graph, for $k > 3$. There is also no efficient parallel algorithm known for finding a smallest augmentation to k -connect any nontrivial graph, for $k > 3$.

The above results are for augmenting undirected graphs. For augmenting directed graphs, Masuzawa, Hagihara & Tokura [23] gave an optimal-time sequential algorithm for finding a smallest augmentation to k -connect a rooted directed tree, for an arbitrary k . We are unaware of any results for finding a smallest augmentation to k -connect any nontrivial directed graph other than a rooted directed tree, for $k > 1$.

Other related results on finding smallest vertex-connectivity augmentations are stated in [4, 19].

Edge-Connectivity Augmentations

For the problem of finding a smallest augmentation for a graph to reach a given edge connectivity property, several polynomial time algorithms and efficient parallel algorithms are known. These results can be found in [1, 3, 4, 5, 8, 9, 13, 16, 19, 24, 27, 30, 31, 34, 37].

Augmenting a Weighted Graph

Another version of the problem is to augment a graph, with a weight assigned to each edge, to meet a connectivity requirement using a set of edges with a minimum total cost. Several related problems have been proved to be NP-complete. These results can be found in [3, 5, 7, 20, 22, 32, 33, 36].

Our Result

In this paper, we describe a sequential algorithm for optimally four-connecting a triconnected graph. We first present a lower bound for the number of edges that must be added in order to reach four-connectivity. Note that lower bounds different from the one we give here are known for the number of edges needed to bi-connect a connected graph [3] and to triconnect a bi-connected graph [10]. It turns out that in both these cases, we can always augment the graph using exactly the number of edges specified in this above lower bound [3, 10]. However, an extension of this type of lower bound for four-connecting a triconnected graph does not always give us the exact number of edges needed [15, 17]. (For details and examples, see Section 3.)

We present a new type of lower bound that equals the exact number of edges needed to four-connect a triconnected graph. By using our new lower bound, we derive an $O(n\alpha(m, n) + m)$ time sequential algorithm for finding a smallest set of edges whose addition four-connects a triconnected graph with n vertices and m edges, where $\alpha(m, n)$ is the inverse Ackermann's function. Our new lower bound applies for arbitrary k , and gives a tighter lower bound than the one known earlier for the number of edges needed to k -connect a $(k - 1)$ -connected graph. The new lower bound and the algorithm described here may lead to a better un-

derstanding of the problem of optimally k -connecting a $(k - 1)$ -connected graph, for an arbitrary k .

2 Definitions

We give definitions used in this paper.

Vertex-Connectivity

A graph¹ G with at least $k + 1$ vertices is k -connected, $k \geq 2$, if and only if G is a complete graph with $k + 1$ vertices or the removal of any set of vertices of cardinality less than k does not disconnect G . The vertex-connectivity of G is k if G is k -connected, but not $(k + 1)$ -connected. Let U be a minimal set of vertices such that the resulting graph obtained from G by removing U is not connected. The set of vertices U is a separating k -set. If $|U| = 3$, it is a separating triplet. The degree of a separating k -set S , $d(S)$, in a k -connected graph G is the number of connected components in the graph obtained from G by removing S . Note that the degree of any separating k -set is ≥ 2 .

Wheel and Flower

A set of separating triplets with one common vertex c is called a wheel in [18]. A wheel can be represented by the set of vertices $\{c\} \cup \{s_0, s_1, \dots, s_{q-1}\}$ which satisfies the following conditions: (i) $q > 2$; (ii) $\forall i \neq j$, $\{c, s_i, s_j\}$ is a separating triplet except in the case that $j = ((i + 1) \bmod q)$ and (s_i, s_j) is an edge in G ; (iii) c is adjacent to a vertex in each of the connected components created by removing any of the separating triplets in the wheel; (iv) $\forall j \neq (i + 1) \bmod q$, $\{c, s_i, s_j\}$ is a degree-2 separating triplet. The vertex c is the center of the wheel [18]. For more details, see [18].

The degree of a wheel $W = \{c\} \cup \{s_0, s_1, \dots, s_{q-1}\}$, $d(W)$, is the number of connected components in $G - \{c, s_0, \dots, s_{q-1}\}$ plus the number of degree-3 vertices in $\{s_0, s_1, \dots, s_{q-1}\}$ that are adjacent to c . The degree of a wheel must be at least 3. Note that the number of degree-3 vertices in $\{s_0, s_1, \dots, s_{q-1}\}$ that are adjacent to c is equal to the number of separating triplets in $\{(c, s_i, s_{(i+2) \bmod q}) \mid 0 \leq i < q, \text{ such that } s_{(i+1) \bmod q} \text{ is degree 3 in } G\}$. An example is shown in Figure 1.

A separating triplet with degree > 2 or not in a wheel is called a flower in [18]. Note that it is possible that two flowers of degree-2 $f_1 = \{a_{1,i} \mid 1 \leq i \leq 3\}$ and $f_2 = \{a_{2,i} \mid 1 \leq i \leq 3\}$ have the property that $\forall i$, $1 \leq i \leq 3$, either $a_{1,i} = a_{2,i}$ or $(a_{1,i}, a_{2,i})$ is an edge in G . We denote $f_1 \mathcal{R} f_2$ if f_1 and f_2 satisfy the above

¹Graphs refer to undirected graphs throughout this paper unless specified otherwise.

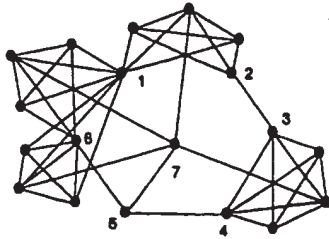


Figure 1: Illustrating a wheel $\{7\} \cup \{1, 2, 3, 4, 5, 6\}$. The degree of this wheel is 5, i.e. the number of components we got after removing the wheel is 4 and there is one vertex (vertex 5) in the wheel with degree 3.

condition. For each flower f , the *flower cluster* \mathcal{F}_f for f is the set of flowers $\{f_1, \dots, f_x\}$ (including f) such that $fRf_i, \forall i, 1 \leq i \leq x$.

Each of the separating triplets in a triconnected graph G is either represented by a flower or is in a wheel. We can construct an $O(n)$ -space representation for all separating triplets (i.e. flowers and wheels) in a triconnected graph with n vertices and m edges in $O(n\alpha(m, n) + m)$ time [18].

K-Block

Let $G = (V, E)$ be a graph with vertex-connectivity $k - 1$. A k -block in G is either (i) a minimal set of vertices B in a separating $(k - 1)$ -set with exactly $k - 1$ neighbors in $V \setminus B$ (these are *special k -blocks*) or (ii) a maximal set of vertices B such that there are at least k vertex-disjoint paths in G between any two vertices in B (these are *non-special k -blocks*). Note that a set consisting of a single vertex of degree $k - 1$ in G is a k -block. A k -block leaf in G is a k -block B_i with exactly $k - 1$ neighbors in $V \setminus B_i$. Note also that every special k -block is a k -block leaf. If there is any special 4-block in a separating triplet S , $d(S) \leq 3$. Given a non-special k -block B leaf, the vertices in B that are not in the flower cluster that separates B are *demanding vertices*. We let every vertex in a special 4-block leaf be a demanding vertex.

Claim 1 Every non-special k -block leaf contains at least one demanding vertex. \square

Using procedures in [18], we can find all of the 4-block leaves in a triconnected graph with n vertices and m edges in $O(n\alpha(m, n) + m)$ time.

Four-Block Tree

From [18] we know that we can decompose vertices in a triconnected graph into the following 3 types: (i) 4-blocks; (ii) wheels; (iii) separating triplets that are

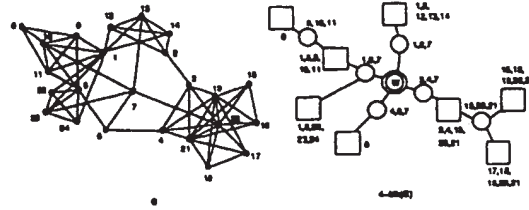


Figure 2: Illustrating a triconnected graph and its $4\text{-blk}(G)$. We use rectangles, circles and two concentric circles to represent R -vertices, F -vertices and W -vertices, respectively. The vertex-numbers beside each vertex in $4\text{-blk}(G)$ represent the set of vertices corresponding to this vertex.

not in a wheel. We modify the decomposition tree in [18] to derive the *four-block tree* $4\text{-blk}(G)$ for a triconnected graph G as follows. We create an R -vertex for each 4-block that is not special (i.e. not in a separating set or in the center of a wheel), an F -vertex for each separating triplet that is not in a wheel, and a W -vertex for each wheel. For each wheel $W = \{c\} \cup \{s_0, s_1, \dots, s_{q-1}\}$, we also create the following vertices. An F -vertex is created for each separating triplet of the form $\{c, s_i, s_{(i+1) \bmod q}\}$ in W . An R -vertex is created for every degree-3 vertex s in $\{s_0, s_1, \dots, s_{q-1}\}$ that is adjacent to c and an F -vertex is created for the three vertices that are adjacent to s . There is an edge between an F -vertex f and an R -vertex r if each vertex in the separating triplet corresponding to f is either in the 4-block H_r corresponding to r or adjacent to a vertex in H_r . There is an edge between an F -vertex f and a W -vertex w if the wheel corresponding to w contains the separating triplet corresponding to f . A *dummy R -vertex* is created and adjacent to each pair of flowers f_1 and f_2 with the properties that f_1 and f_2 are not already connected and either $f_1 \in \mathcal{F}_{f_2}$, $f_2 \in \mathcal{F}_{f_1}$ (i.e. their flower clusters contain each other) or their corresponding separating triplets are overlapped. An example of a 4-block tree is shown in Figure 2.

Note that a degree-1 R -vertex in $4\text{-blk}(G)$ corresponds to a 4-block leaf, but the reverse is not necessarily true, since we do not represent some special 4-block leaves and all degree-3 vertices that are centers of wheels in $4\text{-blk}(G)$. A special 4-block leaf $\{v\}$, where v is a vertex, is represented by an R -vertex in $4\text{-blk}(G)$ if v is not the center of a wheel w and it is in one of separating triplets of w . The degree of a flower F in G is the degree of its corresponding vertex in $4\text{-blk}(G)$. Note also that the degree of a wheel W in

G is equal to the number of components in $4\text{-blk}(G)$ by removing its corresponding W -vertex w and all F -vertices that are adjacent to w . A wheel W in G is a *star wheel* if $d(W)$ equals the number of leaves in $4\text{-blk}(G)$ and every special 4-block leaf in W is either adjacent to or equal to the center. A star wheel W with the center c has the property that every 4-block leaf in G (not including $\{c\}$ if it is a 4-block leaf) can be separated from G by a separating triplet containing the center c . If G contains a star wheel W , then W is the only wheel in G . Note also that the degree of a wheel is less than or equal to the degree of its center in G .

K -connectivity Augmentation Number

The k -connectivity augmentation number for a graph G is the smallest number of edges that must be added to G in order to k -connect G .

3 A Lower Bound for the Four-Connectivity Augmentation Number

In this section, we first give a simple lower bound for the four-connectivity augmentation number that is similar to the ones for biconnectivity augmentation [3] and triconnectivity augmentation [10]. We show that this above lower bound is not always equal to the four-connectivity augmentation number [15, 17]. We then give a modified lower bound. This new lower bound turns out to be the exact number of edges that we must add to reach four-connectivity (see proofs in Section 4). Finally, we show relations between the two lower bounds.

3.1 A Simple Lower Bound

Given a graph G with vertex-connectivity $k - 1$, it is well known that $\max\{\lceil \frac{l_k}{2} \rceil, d - 1\}$ is a lower bound for the k -connectivity augmentation number where l_k is the number of k -block leaves in G and d is the maximum degree among all separating $(k - 1)$ -sets in G [3]. It is also well known that for $k = 2$ and 3, this lower bound equals the k -connectivity augmentation number [3, 10]. For $k = 4$, however, several researchers [15, 17] have observed that this value is not always equal to the four-connectivity augmentation number. Examples are given in Figure 3. Figure 3.(1) is from [15] and Figure 3.(2) is from [17]. Note that if we apply the above lower bound in each of the three graphs in Figure 3, the values we obtain for Figures 3.(1),

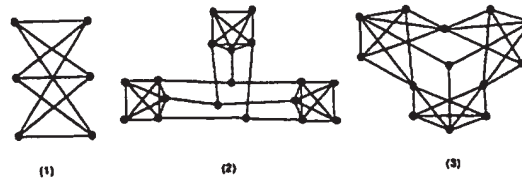


Figure 3: Illustrating three graphs where in each case the value derived by applying a simple lower bound does not equal its four-connectivity augmentation number.

3.(2) and 3.(3) are 3, 3 and 2, respectively, while we need one more edge in each graph to four-connect it.

3.2 A Better Lower Bound

Notice that in the previous lower bound, for every separating triplet S in the triconnected graph $G = \{V, E\}$, we must add at least $d(S) - 1$ edges between vertices in $V \setminus S$ to four-connect G , where $d(S)$ is the degree of S (i.e. the number of connected components in $G - S$); otherwise, S remains a separating triplet. Let the set of edges added be $A_{1,S}$. We also notice that we must add at least one edge into every 4-block leaf B to four-connect G ; otherwise, B remains a 4-block leaf. Since it is possible that S contains some 4-block leaves, we need to know the minimum number of edges needed to eliminate all 4-block leaves inside S . Let the set of edges added be $A_{2,S}$. We know that $A_{1,S} \cap A_{2,S} = \emptyset$. The previous lower bound gives a bound on the cardinality of $A_{1,S}$, but not that of $A_{2,S}$. In the following paragraph, we define a quantity to measure the cardinality of $A_{2,S}$.

Let Q_S be the set of special 4-block leaves that are in the separating triplet S of a triconnected graph G . Two 4-block leaves B_1 and B_2 are *adjacent* if there is an edge in G between every demanding vertex in B_1 and every demanding vertex in B_2 . We create an *augmenting graph for S* , $\mathcal{G}(S)$, as follows. For each special 4-block leaf in Q_S , we create a vertex in $\mathcal{G}(S)$. There is an edge between two vertices v_1 and v_2 in $\mathcal{G}(S)$ if their corresponding 4-blocks are adjacent. Let $\overline{\mathcal{G}(S)}$ be the complement graph of $\mathcal{G}(S)$. The seven types of augmenting graphs and their complement graphs are illustrated in Figure 4.

Definition 1 The *augmenting number* $a(S)$ for a separating triplet S in a triconnected graph is the number of edges in a maximum matching \mathcal{M} of $\overline{\mathcal{G}(S)}$ plus the number of vertices that have no edges in \mathcal{M} incident on them.

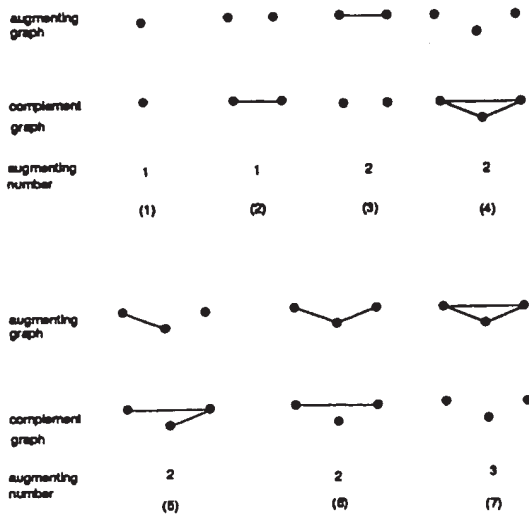


Figure 4: Illustrating the seven types of augmenting graphs, their complement graphs and augmenting numbers that one can get for a separating triplet in a triconnected graph.

The augmenting numbers for the seven types of augmenting graphs are shown in Figure 4. Note that in a triconnected graph, each special 4-block leaf must receive at least one new incoming edge in order to four-connect the input graph. The augmenting number $a(S)$ is exactly the minimum number of edges needed in the separating triplet S in order to four-connect the input graph. The augmenting number of a separating set that does not contain any special 4-block leaf is 0. Note also that we can define the *augmenting number* $a(C)$ for a set C that consists of the center of a wheel using a similar approach. Note that $a(C) \leq 1$.

We need the following definition.

Definition 2 Let G be a triconnected graph with l 4-block leaves. The leaf constraint of G , $lc(G)$, is $\lceil \frac{l}{2} \rceil$. The degree constraint of a separating triplet S in G , $dc(S)$, is $d(S) - 1 + a(S)$, where $d(S)$ is the degree of S and $a(S)$ is the augmenting number of S . The degree constraint of G , $dc(G)$, is the maximum degree constraint among all separating triplets in G . The wheel constraint of a star wheel W with center c in G , $wc(W)$, is $\lceil \frac{d(W)}{2} \rceil + a(\{c\})$, where $d(W)$ is the degree of W and $a(\{c\})$ is the augmenting number of $\{c\}$. The wheel constraint of G , $wc(G)$, is 0 if there is no star wheel in G ; otherwise it is the wheel constraint of the star wheel in G .

We now give a better lower bound on the 4-connectivity augmentation number for a triconnected graph.

Lemma 1 We need at least $\max\{lc(G), dc(G), wc(G)\}$ edges to four-connect a triconnected graph G .

Proof: Let A be a set of edges such that $G' = G \cup A$ is four-connected. For each 4-block leaf B in G , we need one new incoming edge to a vertex in B ; otherwise B is still a 4-block leaf in G' . This gives the first component of the lower bound.

For each separating triplet S in G , $G - S$ contains $d(S)$ connected components. We need to add at least $d(S) - 1$ edges between vertices in $G - S$, otherwise S is still a separating triplet in G' . In addition to that, we need to add at least $a(S)$ edges such that at least one of the two end points of each new edge is in S ; otherwise S contains a special 4-block leaf. This gives the second term of the lower bound.

Given the star wheel W with the center c , $4-blk(G)$ contains exactly $d(W)$ degree-1 R -vertices. Thus we need to add at least $\lceil \frac{d(W)}{2} \rceil$ edges between vertices in $G - \{c\}$; otherwise, G' contains some 4-block leaves. In addition to that, we need to add $a(\{c\})$ non-self-loop edges such that at least one of the two end points of each new edge is in $\{c\}$; otherwise $\{c\}$ is still a special 4-block leaf. This gives the third term of the lower bound. \square

3.3 A Comparison of the Two Lower Bounds

We first observe the following relation between the wheel constraint and the leaf constraint. Note that if there exists a star wheel W with degree $d(W)$, there are exactly $d(W)$ 4-block leaves in G if the center is not degree-3. If the center of the star wheel is degree-3, then there are exactly $d(W) + 1$ 4-block leaves in G . Thus the wheel constraint is greater than the leaf constraint if and only if the star wheel has a degree-3 center. We know that the degree of any wheel is less than or equal to the degree of its center. Thus the value of the above lower bound equals 3.

We state the following claims for the relations between the degree constraint of a separating triplet and the leaf constraint.

Claim 2 Let S be a separating triplet with degree $d(S)$ and h special 4-block leaves. Then there are at least $h + d(S)$ 4-block leaves in G . \square

Claim 3 Let $\{a_1, a_2, a_3\}$ be a separating triplet in a triconnected graph G . Then $a_i, 1 \leq i \leq 3$, is incident on a vertex in every connected component in $G - \{a_1, a_2, a_3\}$. \square

Corollary 1 *The degree of a separating triplet S is no more than the largest degree among all vertices in S .* \square

From Corollary 1, we know that it is not possible that a triconnected graph has type (6) or type (7) of the augmenting graphs as shown in Figure 4, since the degree of their underlying separating triplet is 1. We also know that the degree of a separating triplet with a special 4-block leaf is at most 3 and at least 2. Thus $dc(S)$ is greater than $d(S) - 1$ if $dc(S)$ equals either 3 or 4. Thus we have the following lemma.

Lemma 2 *Let $low_1(G)$ be the lower bound given in Section 3.1 for a triconnected graph G and let $low_2(G)$ be the lower bound given in Lemma 1 in Section 3.2. (i) $low_1(G) = low_2(G)$ if $low_2(G) \notin \{3, 4\}$. (ii) $low_2(G) - low_1(G) \in \{0, 1\}$.* \square

Thus the simple lower bound extended from biconnectivity and triconnectivity is in fact a good approximation for the four-connectivity augmentation number.

4 Finding a Smallest Four-Connectivity Augmentation for a Triconnected Graph

We first explore properties of the 4-block tree that we will use in this section to develop an algorithm for finding a smallest 4-connectivity augmentation. Then we describe our algorithm. Graphs discussed in this section are triconnected unless specified otherwise.

4.1 Properties of the Four-Block Tree

Massive Vertex, Critical Vertex and Balanced Graph

A separating triplet S in a graph G is *massive* if $dc(S) > lc(G)$. A separating triplet S in a graph G is *critical* if $dc(S) = lc(G)$. A graph G is *balanced* if there is no massive separating triplet in G . If G is balanced, then its $4-blk(G)$ is also *balanced*. The following lemma and corollary state the number of massive and critical vertices in $4-blk(G)$.

Lemma 3 *Let S_1, S_2 and S_3 be any three separating triplets in G such that there is no special 4-block in $S_i \cap S_j$, $1 \leq i < j \leq 3$. $\sum_{i=1}^3 dc(S_i) \leq l + 1$, where l is the number of 4-block leaves in G .*

Proof: G is triconnected. We can modify $4-blk(G)$ in the following way such that the number of leaves in the resulting tree equals l and the degree of an F -node f equals its degree constraint plus 1 if f corresponds

to S_i , $1 \leq i \leq 3$. For each W -vertex w with a degree-3 center c , we create an R -vertex r_c for c , an F -vertex f_c for the three vertices that are adjacent to c in G . We add edges (w, f_c) and (f_c, r_c) . Thus r_c is a leaf. For each F -vertex whose corresponding separating triplet S contains h special 4-block leaves, we attach $a(S)$ subtrees with a total number of h leaves with the constraint that any special 4-block that is in more than one separating triplet will be added only once (to the F -node corresponding to S_i , $1 \leq i \leq 3$, if possible). From Figure 4 we know that the number of special 4-block leaves in any separating triplet is greater than or equal to its augmenting number. Thus the above addition of subtrees can be done. Let $4-blk(G)'$ be the resulting graph. Thus the number of leaves in $4-blk(G)'$ is l . Let f be an F -node in $4-blk(G)'$ whose corresponding separating triplet is S . We know that the degree of f equals $dc(S) + 1$ if $S \in \{S_i \mid 1 \leq i \leq 3\}$. It is easy to verify that the sum of degrees of any three internal vertices in a tree is less than or equal to 4 plus the number of leaves in a tree. \square

Corollary 2 *Let G be a graph with more than two non-special 4-block leaves. (i) There is at most one massive F -vertex in $4-blk(G)$. (ii) If there is a massive F -vertex, there is no critical F -vertex. (iii) There are at most two critical F -vertices in $4-blk(G)$.* \square

Updating the Four-Block Tree

Let v_i be a demanding vertex or a vertex in a special 4-block leaf, $i \in \{1, 2\}$. Let B_i be the 4-block leaf that contains v_i , $i \in \{1, 2\}$. Let b_i , $i \in \{1, 2\}$, be the vertex in $4-blk(G)$ such that if v_i is a demanding vertex, then b_i is an R -vertex whose corresponding 4-block contains v_i ; if v_i is in a special 4-block leaf in a flower, then b_i is the F -vertex whose corresponding separating triplet contains v_i ; if v_i is the center of a wheel w , b_i is the F -vertex that is closest to $b_{(i \bmod 2) + 1}$ and is adjacent to w . The vertex b_i is the *implied vertex* for B_i , $i \in \{1, 2\}$. The *implied path P between B_1 and B_2* is the path in $4-blk(G)$ between b_1 and b_2 . Given $4-blk(G)$ and an edge (v_1, v_2) not in G , we can obtain $4-blk(G \cup \{(v_1, v_2)\})$ by performing local updating operations on P . For details, see [18].

In summary, all 4-blocks corresponding to R -vertices in P are collapsed into a single 4-block. Edges in P are deleted. F -vertices in P are connected to the new R -vertex created. We *crack wheels* in a way that is similar to the cracking of a polygon for updating 3-block graphs (see [2, 10] for details). We say that P is *non-adjacent* on a wheel W , if the cracking of W creates two new wheels. Note that it is possible that a separating triplet S in the original graph is no

longer a separating triplet in the resulting graph by adding an edge. Thus some special leaves in the original graph are no longer special, in which case they must be added to $4\text{-blk}(G)$.

Reducing the Degree Constraint of a Separating Triplet

We know that the degree constraint of a separating triplet can be reduced by at most 1 by adding a new edge. From results in [18], we know that we can reduce the degree constraint of a separating triplet S by adding an edge between two non-special 4-block leaves B_1 and B_2 such that the path in $4\text{-blk}(G)$ between the two vertices corresponding to B_1 and B_2 passes through the vertex corresponding to S . We also notice the following corollary from the definitions of $4\text{-blk}(G)$ and the degree constraint.

Corollary 3 *Let S be a separating triplet that contains a special 4-block leaf. (i) We can reduce $dc(S)$ by 1 by adding an edge between two special 4-block leaves B_1 and B_2 in S such that B_1 and B_2 are not adjacent. (ii) If we add an edge between a special 4-block leaf in S and a 4-block leaf B not in S , the degree constraint of every separating triplet corresponding to an internal vertex in the path of $4\text{-blk}(G)$ between vertices corresponding to S and B is reduced by 1. \square*

Reducing the Number of Four-Block Leaves

We now consider the conditions under which the adding of an edge reduces the leaf constraint $lc(G)$ by 1. Let *real degree* of an F -node in $4\text{-blk}(G)$ be 1 plus the degree constraint of its corresponding separating triplet. The real degree of a W -node with a degree-3 center in G is 1 plus its degree in $4\text{-blk}(G)$. The real degree of any other node is equal to its degree in $4\text{-blk}(G)$.

Definition 3 (The Leaf-Connecting Condition)

Let B_1 and B_2 be two non-adjacent 4-block leaves in G . Let P be the implied path between B_1 and B_2 in $4\text{-blk}(G)$. Two 4-block leaves B_1 and B_2 satisfy the leaf-connecting condition if at least one of the following conditions is true. (i) There are at least two vertices of real degree at least 3 in P . (ii) There is at least one R -vertex of degree at least 4 in P . (iii) The path P is non-adjacent on a W -vertex in P . (iv) There is an internal vertex of real degree at least 3 in P and at least one of the 4-block leaves in $\{B_1, B_2\}$ is special. (v) B_1 and B_2 are both special and they do not share the same set of neighbors.

Lemma 4 *Let B_1 and B_2 be two 4-block leaves in G that satisfy the leaf-connecting condition. We can find vertices v_i in B_i , $i \in \{1, 2\}$, such that $lc(G \cup \{(v_1, v_2)\}) = lc(G) - 1$, if $lc(G) \geq 2$. \square*

4.2 The Algorithm

We now describe an algorithm for finding a smallest augmentation to four-connect a triconnected graph. Let $\delta = dc(G) - lc(G)$. The algorithm first adds 2δ edges to the graph such that the resulting graph is balanced and the lower bound is reduced by 2δ . If $lc(G) \neq 2$ or $wc(G) \neq 3$, there is no star wheel with a degree-3 center. We add an edge such that the degree constraint $dc(G)$ is reduced by 1 and the number of 4-block leaves is reduced by 2. Since there is no star wheel with a degree-3 center, $wc(G)$ is also reduced by 1 if $wc(G) = lc(G)$. The resulting graph stays balanced each time we add an edge and the lower bound given in Lemma 1 is reduced by 1. If $lc(G) = 2$ and $wc(G) = 3$, then there exists a star wheel with a degree-3 center. We reduce $wc(G)$ by 1 by adding an edge between the degree-3 center and a demanding vertex of a 4-block leaf. Since $lc(G) = 2$ and $wc(G) = 3$, $dc(G)$ is at most 2. Thus the lower bound can be reduced by 1 by adding an edge. We keep adding an edge at a time such that the lower bound given in Lemma 1 is reduced by 1. Thus we can find a smallest augmentation to four-connect a triconnected graph. We now describe our algorithm.

The Input Graph is not Balanced

We use an approach that is similar to the one used in biconnectivity and triconnectivity augmentations to balance the input graph [10, 11, 26]. Given a tree T and a vertex v in T , a v -chain [26] is a component in $T - \{v\}$ without any vertex of degree more than 2. The leaf of T in each v -chain is a v -chain leaf [26]. Let $\delta = dc(G) - lc(G)$ for an unbalanced graph G and let $4\text{-blk}(G)'$ be the modified 4-block tree given in the proof of Lemma 3. Let f be a massive F -vertex. We can show that either there are at least $2\delta + 2$ f -chains in $4\text{-blk}(G)'$ (i.e. f is the only massive F -vertex) or we can eliminate all massive F -vertices by adding an edge. Let λ_i be a demanding vertex in the i th f -chain leaf. We add the set of edges $\{(\lambda_i, \lambda_{i+1}) \mid 1 \leq i \leq 2\delta\}$. It is also easy to show that the lower bound given in Lemma 1 is reduced by 2δ and the graph is balanced.

The Input Graph is Balanced

We first describe the algorithm. Then we give its proof of correctness. In the description, we need the following definition. Let B be a 4-block leaf whose implied vertex in $4\text{-blk}(G)$ is b and let B' be a 4-block leaf whose implied vertex in $4\text{-blk}(G)$ is b' . B' is a nearest 4-block leaf of B if there is no other 4-block leaf whose implied vertex has a distance to b that is shorter than the distance between b and b' .

{* G is triconnected with ≥ 5 vertices; the algorithm finds a smallest four-connectivity augmentation. *}
graph function aug3to4(graph G);
{* The algorithmic notation used is from Tarjan [29]. *}
 $T := 4\text{-blk}(G)$; root T at an arbitrary vertex;
let \bar{l} be the number of degree-1 R -vertices in T ;
do \exists a 4-block leaf in $G \rightarrow$
if \exists a degree-3 center $c \rightarrow$
1. if $lc(G) = 2$ and $wc(G) = 3 \rightarrow$
{* Vertex c is the center of the star wheel w . *}
 $u_1 :=$ the 4-block leaf $\{c\}$;
let u_2 be a non-special 4-block leaf
| \exists another degree-3 center c' non-adjacent to $c \rightarrow$
let u_2 be the 4-block leaf $\{c'\}$
| \exists a special 4-block leaf b non-adjacent to $u_1 \rightarrow$
let $u_2 := b$
| \exists (degree-3 center or special 4-block leaf)
non-adjacent to $u_1 \rightarrow$
let u_2 be a 4-block leaf such that \exists an internal
vertex with real degree ≥ 3 in their implied path
fi
| $lc(G) \neq 2$ or $wc(G) \neq 3 \rightarrow$
if $\bar{l} > 2$ and \exists 2 critical F -vertices f_1 and $f_2 \rightarrow$
2. find two non-special 4-block leaves u_1 and u_2 such
that the implied path between them passes through
 f_1 and f_2
| $\bar{l} > 2$ and \exists only one critical F -vertex $f_1 \rightarrow$
if \exists two non-adjacent special 4-block leaves in the
separating triplet S_1 corresponding to $f_1 \rightarrow$
3. let u_1 and u_2 be two non-adjacent 4-block leaves
in S_1
| \exists two non-adjacent special 4-block leaves in the
separating triplet S_1 corresponding to $f_1 \rightarrow$
4. let v be a vertex with the largest real degree
among all vertices in T besides f_1 ;
if real degree of v in $T \geq 3 \rightarrow$
find two non-special 4-block leaves u_1 and u_2
such that the implied path between them
passes through f_1 and v
fi
{* The case when the degree of v in $T < 3$ will
be handled in step 8. *}
fi
| \exists two vertices v_1 and v_2 with real degree $\geq 3 \rightarrow$
5. find two non-special 4-block leaves u_1 and u_2 such
that the implied path between them passes
through v_1 and v_2
| \exists an R -vertex v of degree $\geq 4 \rightarrow$
6. find two non-special 4-block leaves u_1 and u_2 such
that the implied path between them passes
through v
| \exists a W -vertex v of degree $\geq 4 \rightarrow$

7. let u_1 and u_2 be two non-special 4-block leaves such
that the implied path between them is
non-adjacent on v
| \exists only one vertex v in T with real degree $\geq 3 \rightarrow$
{* T is a star with the center v . *}
8. find a nearest vertex w of v that contains a 4-block
leaf v_1 ;
let w' be a nearest vertex of w containing a 4-block
leaf non-adjacent to v_1 ;
find two 4-block leaves u_1 and u_2 whose implied
path passes through w , w' and v
{* The above step can always be done, since T is a
star. *}
{* Note that T is path for all the cases below. *}
| \exists two non-adjacent special 4-block leaves in one
separating triplet $S \rightarrow$
9. let u_1 and u_2 be two non-adjacent special 4-block
leaves in S
| \exists a special 4-block leaf $u_1 \rightarrow$
10. find a nearest non-adjacent 4-block leaf u_2
| $\bar{l} = 2 \rightarrow$
let u_1 and u_2 be the two 4-block leaves
corresponding to the two degree-1 R -vertices in T
fi
fi;
let $y_i, i \in \{1, 2\}$, be a demanding vertex in u_i such that
 (y_1, y_2) is not an edge in the current G ;
 $G := G \cup \{(y_1, y_2)\}$;
update $T, \bar{l}, lc(G), wc(G)$ and $dc(G)$
od;
return G
end aug3to4;

Before we show the correctness of algorithm
aug3to4, we need the following claim and corollaries.

Claim 4 [26] *If $4\text{-blk}(G)$ contains two critical
vertices f_1 and f_2 , then every leaf is either in an f_1 -chain
or in an f_2 -chain and the degree of any other vertex
in $4\text{-blk}(G)$ is at most 2.* \square

Corollary 4 *If $4\text{-blk}(G)$ contains two critical vertices
 f_1 and f_2 and the corresponding separating triplet S_i ,
 $i \in \{1, 2\}$, of f_i contains a special 4-block leaf, then
its augmenting number equals the number of special
4-block leaves in it.* \square

Corollary 5 *Let f_1 and f_2 be two critical F -vertices
in $4\text{-blk}(G)$. If the number of degree-1 R -vertices in
 $4\text{-blk}(G) > 2$ and the corresponding separating triplet
of $f_i, i \in \{1, 2\}$, contains a 4-block leaf B_i , we can add
an edge between a vertex in B_1 and a vertex in B_2 to
reduce the lower bound given in Lemma 1 by 1.* \square

Theorem 1 Algorithm *aug3to4* adds the smallest number of edges to four-connect a triconnected graph. \square

We now describe an efficient way of implementing algorithm *aug3to4*. The 4-block tree can be computed in $O(n\alpha(m, n) + m)$ time for a graph with n vertices and m edges [18]. We know that the leaf constraint, the degree constraint of any separating triplet and the wheel constraint of any wheel in G can only be decreased by adding an edge. We also know that $lc(G)$, the sum of degree constraints of all separating triplets and the sum of wheel constraints of all wheels are all $O(n)$. Thus we can use the technique in [26] to maintain the current leaf constraint, the degree constraint for any separating triplet and the wheel constraint for any wheel in $O(n)$ time for the entire execution of the algorithm. We also visit each vertex and each edge in the 4-block tree a constant number of times before deciding to collapse them. There are $O(n)$ 4-block leaves and $O(n)$ vertices and edges in $4\text{-blk}(G)$. In each vertex, we need to use a set-union-find algorithm to maintain the identities of vertices after collapsing. Hence the overall time for updating the 4-block tree is $O(n\alpha(n, n))$. We have the following claim.

Claim 5 Algorithm *aug3to4* can be implemented in $O(n\alpha(m, n) + m)$ time where n and m are the number of vertices and edges in the input graph, respectively and $\alpha(m, n)$ is the inverse Ackermann's function. \square

5 Conclusion

We have given a sequential algorithm for finding a smallest set of edges whose addition four-connects a triconnected graph. The algorithm runs in $O(n\alpha(m, n) + m)$ time using $O(n + m)$ space. The following approach was used in developing our algorithm. We first gave a 4-block tree data structure for a triconnected graph that is similar to the one given in [18]. We then described a lower bound on the smallest number of edges that must be added based on the 4-block tree of the input graph. We further showed that it is possible to decrease this lower bound by 1 by adding an appropriate edge.

The lower bound that we gave here is different from the ones that we have for biconnecting a connected graph [3] and for triconnecting a biconnected graph [10]. We also showed relations between these two lower bounds. This new lower bound applies for arbitrary k , and gives a tighter lower bound than the one known earlier for the number of edges needed to k -connect a $(k - 1)$ -connected graph. It is likely that

techniques presented in this paper may be used in finding the k -connectivity augmentation number of a $(k - 1)$ -connected graph, for an arbitrary k .

Acknowledgment

We would like to thank Vijaya Ramachandran for helpful discussions and comments. We also thank Tibor Jordan, Arkady Kanevsky and Roberto Tamassia for useful information.

References

- [1] G.-R. Cai and Y.-G. Sun. The minimum augmentation of any graph to a k -edge-connected graph. *Networks*, 19:151-172, 1989.
- [2] G. Di Battista and R. Tamassia. On-line graph algorithms with spqr-trees. In *Proc. 17th Int'l Conf. on Automata, Language and Programming*, volume LNCS # 443, pages 598-611. Springer-Verlag, 1990.
- [3] K. P. Eswaran and R. E. Tarjan. Augmentation problems. *SIAM J. Comput.*, 5(4):653-665, 1976.
- [4] D. Fernández-Baca and M. A. Williams. Augmentation problems on hierarchically defined graphs. In *1989 Workshop on Algorithms and Data Structures*, volume LNCS # 382, pages 563-576. Springer-Verlag, 1989.
- [5] A. Frank. Augmenting graphs to meet edge-connectivity requirements. In *Proc. 31th Annual IEEE Symp. on Foundations of Comp. Sci.*, pages 708-718, 1990.
- [6] H. Frank and W. Chou. Connectivity considerations in the design of survivable networks. *IEEE Trans. on Circuit Theory*, CT-17(4):486-490, December 1970.
- [7] G. N. Frederickson and J. Ja'Ja'. Approximation algorithms for several graph augmentation problems. *SIAM J. Comput.*, 10(2):270-283, May 1981.
- [8] H. N. Gabow. Applications of a poset representation to edge connectivity and graph rigidity. In *Proc. 32th Annual IEEE Symp. on Foundations of Comp. Sci.*, pages 812-821, 1991.
- [9] D. Gusfield. Optimal mixed graph augmentation. *SIAM J. Comput.*, 16(4):599-612, August 1987.
- [10] T.-s. Hsu and V. Ramachandran. A linear time algorithm for triconnectivity augmentation. In *Proc. 32th Annual IEEE Symp. on Foundations of Comp. Sci.*, pages 548-559, 1991.

- [11] T.-s. Hsu and V. Ramachandran. On finding a smallest augmentation to biconnect a graph. In *Proceedings of the Second Annual Int'l Symp. on Algorithms*, volume LNCS #557, pages 326–335. Springer-Verlag, 1991. *SIAM J. Comput.*, to appear.
- [12] T.-s. Hsu and V. Ramachandran. An efficient parallel algorithm for triconnectivity augmentation. Manuscript, 1992.
- [13] T.-s. Hsu and V. Ramachandran. Three-edge connectivity augmentations. Manuscript, 1992.
- [14] S. P. Jain and K. Gopal. On network augmentation. *IEEE Trans. on Reliability*, R-35(5):541–543, 1986.
- [15] T. Jordan, February 1992. Private communications.
- [16] Y. Kajitani and S. Ueno. The minimum augmentation of a directed tree to a k -edge-connected directed graph. *Networks*, 16:181–197, 1986.
- [17] A. Kanevsky and R. Tamassia, October 1991. Private communications.
- [18] A. Kanevsky, R. Tamassia, G. Di Battista, and J. Chen. On-line maintenance of the four-connected components of a graph. In *Proc. 32th Annual IEEE Symp. on Foundations of Comp. Sci.*, pages 793–801, 1991.
- [19] G. Kant. Linear planar augmentation algorithms for outerplanar graphs. Tech. Rep. RUU-CS-91-47, Dept. of Computer Science, Utrecht University, the Netherlands, 1991.
- [20] G. Kant and H. L. Bodlaender. Planar graph augmentation problems. In *Proc. 2nd Workshop on Data Structures and Algorithms*, volume LNCS #519, pages 286–298. Springer-Verlag, 1991.
- [21] R. M. Karp and V. Ramachandran. Parallel algorithms for shared-memory machines. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 869–941. North Holland, 1990.
- [22] S. Khuller and R. Thurimella. Approximation algorithms for graph augmentation. In *Proc. 19th Int'l Conf. on Automata, Language and Programming*, 1992, to appear.
- [23] T. Masuzawa, K. Hagihara, and N. Tokura. An optimal time algorithm for the k -vertex-connectivity unweighted augmentation problem for rooted directed trees. *Discrete Applied Mathematics*, pages 67–105, 1987.
- [24] D. Naor, D. Gusfield, and C. Martel. A fast algorithm for optimally increasing the edge-connectivity. In *Proc. 31th Annual IEEE Symp. on Foundations of Comp. Sci.*, pages 698–707, 1990.
- [25] V. Ramachandran. Parallel open ear decomposition with applications to graph biconnectivity and triconnectivity. In J. H. Reif, editor, *Synthesis of Parallel Algorithms*. Morgan-Kaufmann, 1992, to appear.
- [26] A. Rosenthal and A. Goldner. Smallest augmentations to biconnect a graph. *SIAM J. Comput.*, 6(1):55–66, March 1977.
- [27] D. Soroker. Fast parallel strong orientation of mixed graphs and related augmentation problems. *Journal of Algorithms*, 9:205–223, 1988.
- [28] K. Steiglitz, P. Weiner, and D. J. Kleitman. The design of minimum-cost survivable networks. *IEEE Trans. on Circuit Theory*, CT-16(4):455–460, 1969.
- [29] R. E. Tarjan. *Data Structures and Network Algorithms*. SIAM Press, Philadelphia, PA, 1983.
- [30] S. Ueno, Y. Kajitani, and H. Wada. Minimum augmentation of a tree to a k -edge-connected graph. *Networks*, 18:19–25, 1988.
- [31] T. Watanabe. An efficient way for edge-connectivity augmentation. Tech. Rep. ACT-76-UULU-ENG-87-2221, Coordinated Science lab., University of Illinois, Urbana, IL, 1987.
- [32] T. Watanabe, Y. Higashi, and A. Nakamura. Graph augmentation problems for a specified set of vertices. In *Proceedings of the first Annual Int'l Symp. on Algorithms*, volume LNCS #450, pages 378–387. Springer-Verlag, 1990. Earlier version in *Proc. 1990 Int'l Symp. on Circuits and Systems*, pages 2861–2864.
- [33] T. Watanabe and A. Nakamura. On a smallest augmentation to triconnect a graph. Tech. Rep. C-18, Department of Applied Mathematics, faculty of Engineering, Hiroshima University, Higashi-Hiroshima, 724, Japan, 1983. revised 1987.
- [34] T. Watanabe and A. Nakamura. Edge-connectivity augmentation problems. *J. Comp. System Sci.*, 35:96–144, 1987.
- [35] T. Watanabe and A. Nakamura. 3-connectivity augmentation problems. In *Proc. of 1988 IEEE Int'l Symp. on Circuits and Systems*, pages 1847–1850, 1988.
- [36] T. Watanabe, T. Narita, and A. Nakamura. 3-edge-connectivity augmentation problems. In *Proc. of 1989 IEEE Int'l Symp. on Circuits and Systems*, pages 335–338, 1989.
- [37] T. Watanabe, M. Yamakado, and K. Onaga. A linear time augmenting algorithm for 3-edge-connectivity augmentation problems. In *Proc. of 1991 IEEE Int'l Symp. on Circuits and Systems*, pages 1168–1171, 1991.


IEEE Xplore[®]
 RELEASE 1.6

 Welcome
 United States Patent and Trademark Office

[Help](#) | [FAQ](#) | [Terms](#) | [IEEE Peer Review](#)
[Quick Links](#)
[» Search Absl](#)
Welcome to IEEE Xplore[®]

- Home
- What Can I Access?
- Log-out

Tables of Contents

- Journals & Magazines
- Conference Proceedings
- Standards

Search

- By Author
- Basic
- Advanced

Member Services

- Join IEEE
- Establish IEEE Web Account
- Access the IEEE Member Digital Library

[Search Results](#) | [\[PDF FULL-TEXT 776 KB\]](#) | [PREV](#) | [NEXT](#) | [DOWNLOAD CITATION](#)
[Order Reuse Permissions](#)
RIGHTS LINK

On four-connecting a triconnected graph

Hsu, T.

Dept. of Comput. Sci., Texas Univ., Austin, TX, USA;

This paper appears in: Foundations of Computer Science, 1992. Proceedings., Annual Symposium on

Meeting Date: 10/24/1992 - 10/27/1992

Publication Date: 24-27 Oct. 1992

Location: Pittsburgh, PA USA

On page(s): 70 - 79

Reference Cited: 37

Inspec Accession Number: 4488295

Abstract:

The author considers the problem of finding a smallest set of edges whose addition f connects a triconnected graph. This is a fundamental graph-theoretic problem that has applications in designing reliable **networks**. He presents an $O(n\alpha(m,n)+m)$ time sequential algorithm for four-connecting an undirected graph G that is triconnected by adding the smallest number of edges, where n and m are the number of vertices and edges in G , respectively, and $\alpha(m, n)$ is the inverse Ackermann function. He presents a new lower bound for the number of edges needed to four-connect a triconnected graph. The form of this lower bound is different from the form of the lower bound known for biconnectivity augmentation and triconnectivity augmentation. The new lower bound applies for arbitrary k , and gives a tighter lower bound than the one known earlier for the number of edges needed to **k-connect** a $(k-1)$ -connect graph. For $k=4$, he shows that this lower bound is tight by giving an efficient algorithm for finding a set of edges with required size whose addition four-connects a triconnected graph.

Index Terms:

[computational complexity](#) | [computational geometry](#) | [four-connecting](#) | [graph theory](#) | [graph-theoretic problem](#) | [inverse Ackermann function](#) | [reliable networks](#) | [triconnected graph](#) | [computational complexity](#) | [computational geometry](#) | [four-connecting](#) | [graph theory](#) | [graph-theoretic problem](#) | [inverse Ackermann function](#) | [reliable networks](#) | [triconnected graph](#)

Documents that cite this document

There are no citing documents available in IEEE Xplore at this time.

[Search Results](#) [[PDF FULL-TEXT 776 KB](#)] [PREV](#) [NEXT](#) [DOWNLOAD CITATION](#)

[Home](#) | [Log-out](#) | [Journals](#) | [Conference Proceedings](#) | [Standards](#) | [Search by Author](#) | [Basic Search](#) | [Advanced Search](#) | [Join IEEE](#) | [Web Account](#) | [New this week](#) | [OPAC Linking Information](#) | [Your Feedback](#) | [Technical Support](#) | [Email Alerting](#) | [No Robots Please](#) | [Release Notes](#) | [IEEE Online Publications](#) | [Help](#) | [FAQ](#) | [Terms](#) | [Back to Top](#)

Copyright © 2004 IEEE — All rights reserved

A Flexible Architecture for Multi-Hop Optical Networks

A. Jaekel, S. Bandyopadhyay

and

A. Sengupta

School of Computer Science,
University of Windsor,

Windsor, Ontario N9B 3P4, CANADA

Department of Computer Science
University of South Carolina

Columbia, SC 29208

Abstract

It is desirable to have low diameter logical topologies for multihop lightwave networks. Researchers have investigated regular topologies for such networks. Only a few of these (e.g., GEMNET [8]) are scalable to allow the addition of new nodes to an existing network. Adding new nodes to such networks requires a major change in routing scheme. For example, in a multistar implementation, a large number of retuning of transmitters and receivers and/or renumbering nodes are needed for [8]. In this paper, we present a scalable logical topology which is not regular but it has a low diameter. This topology is interesting since it allows the network to be expanded indefinitely and new nodes can be added with a relatively small change to the network. In this paper we have presented the new topology, an algorithm to add nodes to the network and two routing schemes.

Keywords: *Optical networks, multihop networks, scalable logical topology, low diameter networks.*

1. Introduction

Optical networks [1] are interconnections of high-speed broadband fibers using *lightpaths*. Each lightpath provides traverses one or more fibers and uses one wavelength division multiplexed (WDM) channel per fiber. In a multihop network, each node has a small number of lightpaths to a few other nodes in the network. The physical topology of the network determines how the lightpaths get defined. For a multistar implementation of the physical topology, a lightpath $u \rightarrow v$ is established when node u broadcasts to a passive optical coupler at a particular wavelength and the node v picks up the optical signal by tuning its receiver to the same wavelength. For a wavelength routed network, a lightpath $u \rightarrow v$ might be established through one or several fibers interconnected by router nodes. The lightpath definition between the nodes in an optical network is usually represented by a directed graph (or digraph) $G = (V, E)$ (where V is the set of nodes and E is the set of the edges) with each node of G representing a

node of the network and each edge (denoted by $u \rightarrow v$) representing a lightpath from u to v . G is usually called the logical topology of the network. When the lightpath $u \rightarrow v$ does not exist, the communication from a node u to a node v occurs by using a (graph-theoretic) path (denoted by $u \rightarrow x_1 \rightarrow x_2 \rightarrow \dots \rightarrow x_{k-1} \rightarrow v$) in G using k hops through the intermediate nodes x_1, x_2, \dots, x_{k-1} . The information is buffered at intermediate nodes and, to reduce the communication delay, the number of hops should be small. If a shortest graph-theoretic path is used to establish a communication from u to v , the maximum hop distance is the *diameter* of G . Clearly, the lightpaths need to be defined such that G has a small diameter and low average hop distance. The indegree and outdegree of each node should be low to reduce the network cost. However, a reduction of the degree usually implies an increase in the diameter of the digraph, that is, larger communication delays. The design of the logical topology of a network turns out to be a difficult problem in view of these contradictory requirements. Several different logical topologies have been proposed in the literature. An excellent review of multihop networks is presented in [1].

Both regular and irregular structures have been studied for multihop structures [2], [3], [4], [5], [6], [7]. All the proposed regular topologies (e.g., shuffle nets, de Bruijn graphs, torus, hypercubes) enjoy the property of simple routing algorithms, thereby avoiding the need of complex routing tables. Since the diameter of a digraph with n nodes and maximum outdegree d is of $O(\log_d n)$, most of the topologies attempt to reduce the diameter to $O(\log_d n)$. One common property of these network topologies is the number of nodes in the network must be given by some well-defined formula involving network parameters. This makes the topology non-scalable. In short, addition of a node to an existing network is virtually impossible. In [8], the principle of shuffle interconnection between nodes in a shuffle net [4] is generalized (the generalized version can have any number of nodes in each column) to obtain a scalable network topology called GEMNET. A similar idea of generalizing

the Kautz graph has been studied in [9] showing a better diameter and network throughput than GEMNET. Both these scalable topologies are given by regular digraphs.

One topology that has been studied for optical networks is the bidirectional ring network. In such networks, each node has two incoming lightpaths and two outgoing lightpaths. In terms of the graph model, each node has one outgoing edge to and one incoming edge from the preceding and the following node in the network. Adding a new node to such a ring network involves redefining a fixed number of edges and can be repeated indefinitely.

Our motivation was to develop a topology which has the advantages of a ring network with respect to scalability and the advantages of a regular topology with respect to low diameter. In other words, our topology has to satisfy the following characteristics:

- The diameter should be small
- The routing strategy should be simple
- It should be possible to add new nodes to the network indefinitely with the least possible perturbation of the network.
- Each node in the network should have a predefined upper limit on the number of incoming and outgoing edges.

In this paper we introduce a new scalable topology for multihop networks where the graph is not, in general, regular. Given integers n and d , our proposed topology can be defined for n nodes with a fixed number of incoming and outgoing edges in the network. The major advantage of our scheme is that, as a new node is added to the network, most of the existing edges of the logical topology are not changed, implying that the routing schemes between the existing nodes need little modification. The edges to and from the new added node can be implemented by defining new lightpaths which is small in number, namely, $O(d)$. For multistar implementation, for example, this can be accomplished by retuning $O(d)$ transmitters and receivers.

The paper is organized as follows. In section 2, we describe the proposed topology and derive its pertinent properties. Section 3 presents two routing schemes for the proposed topology and establishes that the diameter is $O(\log_d n)$. Our experiments in section 4 show that, for a network with n nodes and having an indegree of at most $d+1$, an outdegree of d and the average hop distance is approximately $\log_d n$. We have concluded with a critical summary in section 4.

2. Scalable topology for multihop networks

2.1 Proposed interconnection topology

Given two integers n and d , $d \leq n$, we define the interconnection topology of the network as a digraph G in the following. As mentioned earlier, the digraph is not

regular - the indegree and outdegree of a node varies from l to $d+1$. We will assume that there is no k , such that

$n = d^k$; if $n = d^k$ for some k , our proposed topology is the same as given by [2]. Let k be the integer such that $d^k < n < d^{k+1}$. Let Z_k be the set of all $(k+1)$ -digit strings

choosing digits from $Z = \{0, 1, 2, \dots, d-1\}$ and let any string of Z_k be denoted by $x_0 x_1 \dots x_k$. We divide Z_k into $k+2$ sets S_0, S_1, \dots, S_{k+1} such that all strings in Z_k having x_j as the left most occurrence of 0 is included in S_j , $0 \leq j \leq k$ and all strings with no occurrence of 0 (i.e. $x_j \neq 0, 0 \leq j \leq k$) is included in S_{k+1} . We note that

$$|S_{k+1}| = (d-1)^{k+1} \quad \text{and} \quad |S_j| = (d-1)^j d^{k-j},$$

$0 \leq j \leq k$. We define an ordering relation between every pair of strings in Z_k . Each string in S_i is smaller than each string in S_j if $i < j$. For two strings $\sigma_1, \sigma_2 \in S_j$, $0 \leq j \leq k+1$, if $\sigma_1 = x_0 x_1 \dots x_k$ and $\sigma_2 = y_0 y_1 \dots y_k$ and r is the largest integer such that $x_i \neq y_i$ then $\sigma_1 < \sigma_2$ if $x_r < y_r$.

Definition: For any string $\sigma_1 = x_0 x_1 \dots x_i \dots x_j \dots x_k$, the string $\sigma_2 = x_0 x_1 \dots x_j \dots x_i \dots x_k$ obtained by interchanging the digits in the i^{th} and the j^{th} position in σ_1 , will be called the i - j -image of σ_1 .

Clearly, if σ_2 is the i - j -image of σ_1 then σ_1 is the i - j -image of σ_2 and if $x_i = x_j$, σ_1 and σ_2 represent the same node.

We will represent each node of the interconnection topology by a distinct string $x_0 x_1 \dots x_k$ of Z_k . As $d^k < n < d^{k+1}$, all strings of Z_k will not be used to represent the nodes in G . We will use n smallest strings from Z_k to represent the nodes of G . Suppose the largest string representing a node is in S_M . We will use a node and its string representation interchangeably. We will use the term *used* string to denote a string of Z_k which has been already used to represent some node in G . All other strings of Z_k will be called *unused* strings.

Property 1: all strings of S_0 are used strings.

Property 2: if $\sigma \in S_j$ is an used string, then all strings

of S_0, S_1, \dots, S_{j-1} are also used strings.

Property 3: If $\sigma_1 = 0x_1 \dots x_k$, σ_2 is the 0-1-image of σ_1 and $x_1 \neq 0$, then $\sigma_2 \in S_1$.

Property 4: If $\sigma_1 = 0x_1 \dots x_k$, $x_1 \neq 0$ and σ_2 , the 0-1-image of σ_1 , is an unused string, then all strings of the form $x_1 x_2 \dots x_k j$, $0 \leq j \leq d-1$ are unused strings.

The proofs for Properties 1 - 4 are trivial and are omitted.

We now define the edge set of the digraph G . Let any node u in G be represented by $x_0 x_1 \dots x_k$. The outgoing edges from node u are defined as follows:

- There is an edge $x_0 x_1 x_2 \dots x_k \rightarrow x_1 x_2 \dots x_k j$ whenever $x_1 x_2 \dots x_k j$ is an used string, for some $j \in Z$,
- There is an edge $0x_1 x_2 \dots x_k \rightarrow x_1 0x_2 \dots x_k$ whenever the following conditions hold:
 - a) $x_1 x_2 \dots x_k j$ is an unused string for at least one $j \in Z$ and
 - b) $x_1 0 \dots x_k$, the 0-1-image of u , is an used string
- There is an edge $0x_1 x_2 \dots x_k \rightarrow 0x_2 \dots x_k j$ for all $j \in Z$ whenever the following conditions hold:
 - a) $x_1 \neq 0$ and
 - b) $x_1 0x_2 \dots x_k$, the 0-1-image of u , is an unused string

We note that if $u \in S_j$, $j > 0$, node $v = x_1 x_2 \dots x_k j$ always exists (from property 2, since $v \in S_{j-1}$). As an example, we show a network with 5 nodes for $d=2, k=2$ in figure 1. We have used a solid line for an edge of the type $x_0 x_1 x_2 \dots x_k \rightarrow x_1 x_2 \dots x_k j$, a line of dots for and a line of dashes and dots for an edge of the type $0x_1 x_2 \dots x_k \rightarrow 0x_2 \dots x_k j$. We note that the edge from 010 to 100 satisfies the condition for both an edge of the type $x_0 x_1 x_2 \dots x_k \rightarrow x_1 x_2 \dots x_k j$ and an edge of the type $0x_1 x_2 \dots x_k \rightarrow x_1 0x_2 \dots x_k$.

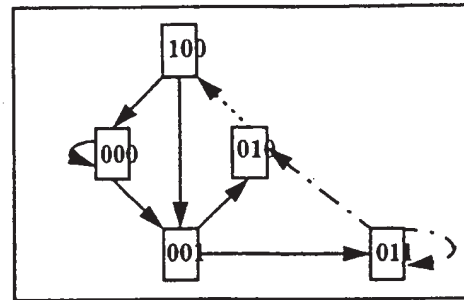


Figure 1: Interconnection topology with $d=2, k=2$ for $n=5$ nodes.

2.2 Limits on Nodal Degree

In this section, we derive the upper limits for the indegree and the outdegree of each node in the network. We will show that, by not enforcing the regularity, we can easily achieve scalability. As we add new nodes to the network, minor modifications of the edges in the logical topology suffice, in contrast to large number of changes in the edge-set as required by other proposed methods.

Theorem 1: In the proposed topology, each node has an outdegree of up to d .

Proof: Let u be a node in the network given by $x_0 x_1 \dots x_k \in S_j$. We consider the following three cases:

- i) $0 < j \leq k$: For every v given by $x_1 x_2 \dots x_k t$ for all t , $0 \leq t \leq d-1$ is an used string since $v \in S_{j-1}$. Therefore the edge $u \rightarrow v$ exists in the network. If $u \in S_j$, $j > 0$, these are the only edges from u . Hence, u has outdegree d .
- ii) $j = 0$: According to our topology defined above, u will have an edge to $x_1 x_2 \dots x_k j$ whenever $x_1 x_2 \dots x_k j$ is an used string for some $j \in Z$. We have three sub-cases to consider:
 - If $x_1 x_2 \dots x_k j$ is an used string for all j , $0 \leq j < d$ then u has outdegree d .
 - Otherwise, if p of the strings $x_1 x_2 \dots x_k j$ are used strings, for some j , $0 \leq j < d$ and the 0-1-image of u is also an used string, then u has edges to all the p nodes with used strings of the form $x_1 x_2 \dots x_k j$ and to the 0-1-image of u . Hence u has outdegree $p + 1$. Here u has an outdegree of at least 1 and at most d .
 - Otherwise, if the 0-1-image of u is an unused string, then all strings of the form $x_1 x_2 \dots x_k j$ are unused

strings (Property 4) and u has d outgoing edges to nodes of the form $0x_2x_3\dots x_kj$, $0 \leq j < d$. Hence u has outdegree d .

iii) $j = k + 1$: If p of the strings $x_1x_2\dots x_kj$ are used strings, for some j , $0 \leq j < d$, then u has outdegree of p . We note that $x_1x_2\dots x_k0 \in S_k$ is an used string. Therefore $1 \leq p \leq d$, and u has an outdegree of at least 1 and at most d .

Theorem 2: In the proposed topology, each node has an indegree of up to $d+1$.

Proof: Let us consider the indegree of any node v given by $y_0y_1\dots y_k \in S_j$. As described in 2.1, there may be three type of edges to node v as follows:

- An edge $ty_0y_1\dots y_{k-1} \rightarrow y_0y_1\dots y_k$ whenever $ty_0y_1\dots y_{k-1}$ is an used string, for some $t \in Z$. There may be at most d edges of this type to v .
- If $y_1 = 0$, $y_0 \neq 0$ there may be an edge $0y_0y_2\dots y_k \rightarrow y_0y_1\dots y_k$
- If $y_0 = 0$ and $ty_0y_1\dots y_{k-1}$ is an unused string for some $t \in Z$, there is an edge $0ty_1\dots y_{k-1} \rightarrow y_0y_1\dots y_k$. There may be at most d edges of this type to v .

We have to consider 3 cases, $j = 0$, $j = 1$ and $j > 1$. If $j > 1$, the only edges are of the type $ty_0y_1\dots y_{k-1} \rightarrow y_0y_1\dots y_k$ and there can be up to d such edges. If $j = 1$, in addition to the edges are of the type $ty_0y_1\dots y_{k-1} \rightarrow y_0y_1\dots y_k$, there can be only one edge of the type $0y_0y_2\dots y_k \rightarrow y_0y_1\dots y_k$. Thus the total number of edges cannot exceed $d + 1$, in this case. If $j = 0$, an edge of the type $0ty_1\dots y_{k-1} \rightarrow y_0y_1\dots y_k$ exists if and only if the corresponding edge of type $ty_0y_1\dots y_{k-1} \rightarrow y_0y_1\dots y_k$ does not exist in the network. Therefore, there are always exactly d incoming edges to v in this case.

2.3 Node Addition to an Existing Network

In this section we consider the changes in the logical topology that should occur when a new node is added to the network. We show that at most $O(d)$ edge changes in G would suffice when a new node is added to the network. When a multistar implementation is considered, this means

$O(d)$ retuning of transmitters and receivers, whereas for a wavelength routed network, this means redefinition of $O(d)$ lightpaths. In contrast, for other proposed topologies [8], [9] the number of edge modifications needed was $O(nd)$. As discussed in the previous section, the nodes are assigned the smallest strings defined earlier. Addition of a new node u implies that we will assign the smallest unused string to the newly added node. Let the string be $x_0x_1\dots x_k \in S_j$. We consider the following three cases:

- i) $1 < j \leq k$: For every v given by $x_1x_2\dots x_kt$, $0 \leq t \leq d-1$, $v \in S_{j-1}$. Therefore v is an used string and we have to add a new edge $u \rightarrow v$ to the network. The node given by $w_0 = 0x_0x_1\dots x_{k-1}$ is guaranteed to be an used string, since $w_0 \in S_0$ and we have to add a new edge $w_0 \rightarrow u$ to the network. If $x_k = d-1$, we have to delete the edge from w_0 to its 0-1-image at this time. For every w given by $tx_0x_1\dots x_{k-1}$, $1 \leq t \leq d-1$, $w \in S_{j+1}$, and is an unused string. Therefore w_0 is the only predecessor of u .
- ii) $j = k + 1$: If $v = x_1x_2\dots x_kt$, $0 \leq t \leq p-1$ is an used string, we add a new edge $u \rightarrow v$ to the network. We note that $x_1x_2\dots x_k0 \in S_k$ is an used string. Therefore, there is at least one v such that $u \rightarrow v$ exists. Similarly, if $w = tx_0x_1\dots x_{k-1}$, $0 \leq t \leq p-1$ is an used string, we add a new edge $w \rightarrow u$ to the network. We note that $w_0 = 0x_0x_1\dots x_{k-1} \in S_0$ is an used string. Therefore, there is at least one w such that $w \rightarrow u$ exists. If $x_k = d-1$, we delete the edge from w_0 to its 0-1-image at this time.
- iii) $j = 1$: Let $w_c = 0x_0x_2\dots x_k$ be the 0-1-image of u . Before inserting u , the node $0x_0x_2\dots x_k$ was connected to all nodes $v = 0x_2\dots x_kt$, $0 \leq t \leq d-1$ (case iii in our topology given in 2.1). We have to
 - delete the edge $w_c \rightarrow v$ for each node $v = 0x_2\dots x_kt$ in the network.
 - add an edge $u \rightarrow v$ for each node $v = 0x_2\dots x_kt$ in the network.
 - add a new edge $w_0 = 0x_0x_1\dots x_{k-1} \rightarrow u$ to the network

- If $w_c \neq w_0$, add an edge $w_c \rightarrow u$ to the network.
- If $x_k = d - 1$, and $w_0 \neq 0x_0000\dots0$ delete the edge from w_0 to its 0-1-image.

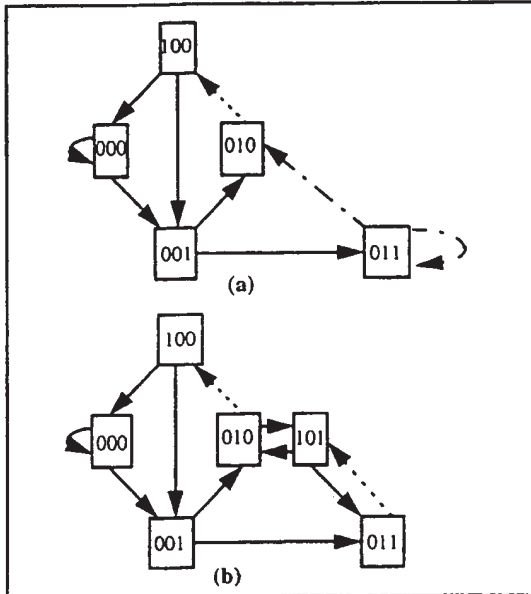


Figure 2: Expanding a topology with $d = 2, k = 2$ from (a) $n = 5$ to (b) $n = 6$ nodes.

Figure 2(a) shows again the network with 5 nodes given in Figure 1. We choose the smallest unused string $u = 101$ to represent the new node being inserted. The node u will have outgoing edges (shown by solid lines) to all nodes of the form $01j$, to nodes 010 and 011 . The 0-1 image of u is node 011 . Hence all edges from 011 to nodes 010 and 011 are deleted and a new edge from 101 to 011 is inserted (shown by a dashed line). Also a new edge is inserted from node 010 to 101 . The final network is shown in Figure 2(b)

3. Routing strategy

In this section, we present two routing schemes in the proposed topology from any source node S to any destination node D . Let S be given by the string $x_0x_1\dots x_k \in S_j$ and D be given by the string $y_0y_1\dots y_k \in S_l$.

3.1 Routing scheme

Let l be the length of the longest suffix of the string $x_0x_1\dots x_k$ that is also a prefix of $y_0y_1\dots y_k$ and let

$\sigma(S, D)$ denote the string $x_0x_1\dots x_ky_ly_{l+1}y_{l+2}\dots y_k$ of length $2(k+1)-l$. Since $\sigma(S, D)$ is of length $2(k+1)-l$, it has $(k+1)-l+1$ substrings, each of length $(k+1)$. Two of these substrings represent S and D . Since S and D are nodes in the network, these two substrings are used strings. If all the remaining $k-l$ substrings of $\sigma(S, D)$ having length $k+1$ are also used strings, then a routing path from S to D of length $k+1-l$ exists as given by the sequence of nodes given in (1) below.

$$S = x_0x_1\dots x_k \rightarrow x_1x_2\dots x_ky_l \rightarrow x_2\dots x_{2k-1}x_ky_ly_{l+1} \rightarrow \dots \rightarrow x_ky_ly_{l+1}y_{l+2} \rightarrow y_0y_1\dots y_k = D \quad (1)$$

In other words, if all the $k-l+2$ substrings of $\sigma(S, D)$ are used strings, we can use $\sigma(S, D)$ to represent the path from S to D in (1).

Property 5: If all the $k-l+2$ substrings of $\sigma(S, D)$ are used strings, $\sigma(S, D)$ represents the shortest path from S to D .

However, if some of the substrings of $\sigma(S, D)$ are not used strings, then some of the corresponding nodes do not currently appear in the network and hence this path does not exist. We note that any two consecutive strings in $\sigma(S, D)$ is given by $\alpha\beta$, where $\alpha = x_ix_{i+1}\dots x_ky_ly_{l+1}\dots y_{l+i}$, $0 \leq i \leq k-l-1$, and

$\beta = x_{i+1}x_{i+2}\dots x_ky_ly_{l+1}\dots y_{l+i}y_{l+i+1}$. Let β be the first unused string in (1). According to our topology, either $\alpha \in S_0$ or $\alpha \in S_{k+1}$.

Property 6: If $\alpha \in S_0$ and

$\gamma = x_{i+1}0x_{i+2}\dots x_ky_ly_{l+1}\dots y_{l+i}$, the 0-1-image of α is an unused string, then

- $\sigma(S, \alpha)$ represents a path from S to α of length i ,
- there exists a path $\alpha \rightarrow \gamma \rightarrow \delta = 0x_{i+2}\dots x_ky_ly_{l+1}\dots y_{l+i}y_{l+i+1}$
- $\sigma(\delta, D)$ is a string of length $k+2-l-i$

Property 7: If $\alpha \in S_0$ and

$\gamma = x_{i+1}0x_{i+2}\dots x_ky_ly_{l+1}\dots y_{l+i}$ the 0-1-image of α is an unused string, then

- $\sigma(S, \alpha)$ represents a path from S to α of length i ,
- there exists a path

$$\alpha \rightarrow \delta = 0x_{i+2} \dots x_k y_l y_{l+1} \dots y_{l+i} y_{l+i+1}$$

- $\sigma(\delta, D)$ is a string of length $k+2-l-i$

Properties 6 and 7 follow directly from our topology defined in 2.1.

Property 8: If a network contains all nodes in S_0, S_1, \dots, S_k then

- there exists an edge $S \rightarrow \gamma = x_1 x_2 \dots x_k 0$ and
- $\sigma(\gamma, D)$ represents a path from α to D of length that cannot exceed $k+1$.

Proof of Property 8: Since the network contains all nodes in S_0, S_1, \dots, S_k , $\gamma \in S_j$ for some j , $j \leq k$ and must exist. Our topology (section 2.1) ensures that the edge $S \rightarrow \gamma$ exists. The path given below consists only strings belonging to groups S_i , $0 \leq i \leq k$ and hence are used strings:

$\gamma \rightarrow x_2 \dots x_k 0 y_0 \rightarrow x_3 \dots x_k 0 y_0 \rightarrow \dots \rightarrow y_0 y_1 \dots y_k$. The number of edges in the path is $k+1$, hence the proof.

Theorem 3: The diameter of a network using the proposed topology cannot exceed $2(k+1)$.

Proof: We consider any source-destination pair (S, D) . If all the $k-l+2$ substrings of $\sigma(S, D)$ are used strings, $\sigma(S, D)$ represents the shortest path from S to D and cannot exceed $k+1$. If β is the first unused string in (1), and α is the preceding string then we have to consider two cases:

Case 1) $\alpha \in S_0$: In this situation we can apply property 6 if 0-1-image of α is an used string. Otherwise we can use property 7. If we can use property 6, it means we need two edges to insert the digit y_{l+i+1} . Alternatively, if we can use property 7, it means we need one edge to insert the digit y_{l+i+1} .

Case 2) $\alpha \in S_{k+1}$: In this situation we discard the partial path from S to α . The first edge in our new path will be $S = x_0 x_1 \dots x_k \rightarrow x_1 x_2 \dots x_k 0$. Property 8 guarantees that once we have this situation, we can always start all over again inserting digits y_0, y_1, \dots, y_k without ever encountering an unused string and requires a

maximum of $k+1$ edges. This represents the worst case since there may exist a shorter path by finding the longest suffix of $x_1 x_2 \dots x_k 0$ that matches the corresponding prefix of D . In this case the path cannot exceed $k+2$.

Case 1 can appear repeatedly. The worst situation is when we have to apply it to insert every digit of D . In other words, the path in this case can be as long as $2(k+1)$.

3.2 Example of routing

Let us consider the network of Figure 2(b). Suppose, $S = 011$ and $D = 001$. Since the only outgoing edge from 011 is to its 0-1-image 101, the first edge in the path is $011 \rightarrow 101$. From 101, we shift in the successive digits of the destination. So, the final path is given by $S = 011 \rightarrow 101 \rightarrow 010 \rightarrow 100 \rightarrow 001 = D$. In this particular example, there are no nodes belonging to group $k+1$. So, case 2 is not used.

4. Experiments to determine the average hop distance

We carried out some experiments to determine the average hop distance \bar{h} . In each of these experiments, we have started with a given value of d , the minimum indegree (or outdegree) and a specified value of an integer k . The network with d^k nodes is identical to that given in [8]. We have calculated the average hop distance \bar{h} of this network from the hop distances of every source/destinations pairs using the routing scheme described in the previous section. Then we have added a node to the network and calculated \bar{h} for the new network in the same way. We continued the process of adding nodes until the network contained d^{k+1} nodes. The results of the experiments are shown in Table 1 and reveal the following:

- The average hop distance is approximately $k+1$.
- The average hop distance starts at approximately k and increases to approximately $k+1$ as we start adding nodes to the network.

We interpret these results as follows. Even though the diameter is $2(k+1)$, the number of lightpaths through paths involving 0-1 images, which increase the number of hops, is relatively small. Our network is identical to that in [2] when the number of nodes in the network is d^k or d^{k+1} and, for these values, it is known that the network has a diameter of

k and k+1 respectively.

Table 1: Variation of average hop distance with number of nodes

Number of nodes	d	k	average hop \bar{h}
10	3	2	2.4333
13	3	2	2.6154
16	3	2	2.6618
19	3	2	2.4954
22	3	2	2.5974
25	3	2	2.5148
10	2	3	2.7000
12	2	3	2.9470
14	2	3	2.8022
16	2	3	2.8333
65	4	3	3.5954
75	4	3	3.8366
85	4	3	4.1077
95	4	3	4.2215
105	4	3	4.5172
115	4	3	4.5506
18	2	4	3.5915
20	2	4	3.67630
22	2	4	3.8636
24	2	4	4.30181
26	2	4	3.7908
28	2	4	3.7169

5. Conclusions

In this paper we have introduced a new graph as a logical network for multihop networks. We have shown that our network has an attractive average hop distance compared to existing networks. The main advantage of our

approach is the fact that we can very easily add new nodes to the network. This means that the perturbation of the network in terms of redefining edges in the network is very small in our architecture. The routing scheme in our network is very simple and avoids the use of routing tables.

Acknowledgments: The work of A. Jaekel and S. Bandyopadhyay has been supported by research grants from the Natural Science and Engineering Research Council of Canada. The work of A. Sengupta has been partially supported by Office of Naval Research grant # N00014-97-1-0806.

REFERENCES

- [1] B. Mukherjee, "WDM-based local lightwave networks part II: Multihop systems," *IEEE Network*, vol. 6, pp. 20-32, July 1992.
- [2] K. Sivarajan and R. Ramaswami, "Lightwave Networks Based on de Bruijn Graphs," *IEEE/ACM Transactions on Networking*, Vol. 2, No. 1, pp. 70-79, Feb 1994.
- [3] K. Sivarajan and R. Ramaswami, "Multihop Networks Based on de Bruijn Graphs," *IEEE INFOCOM '91*, pp. 1001-1011, Apr. 1991.
- [4] M. Hluchyj and M. Karol, "ShuffleNet: An application of generalized perfect shuffles to multihop lightwave networks," *IEEE/OSA Journal of Lightwave Technology*, vol. 9, pp.1386-1397, Oct. 1991.
- [5] B. Li and A. Ganz, "Virtual topologies for WDM star LANs: The regular structure approach," *IEEE INFOCOM '92*, pp.2134-2143, May 1992.
- [6] N. Maxemchuk, "Routing in the Manhattan street network," *IEEE Trans. on Communications*, vol. 35, pp. 503-512, May 1987.
- [7] P. Dowd, "Wavelength division multiple access channel hypercube processor interconnection," *IEEE Trans. on Computers*, 1992.
- [8] J. Innes, S. Banerjee and B. Mukherjee, "GEMNET: A generalized shuffle exchange based regular, scalable and modular multihop network based on WDM lightwave technology", *IEEE/ACM Trans. Networking*, Vol 3, No 4, Aug 1995.
- [9] A. Venkateswaran and A. Sengupta, "On a scalable topology for Lightwave networks", *Proc IEEE INFOCOM'96*, 1996.



Welcome
United States Patent and Trademark Office

[Help](#) [FAQ](#) [Terms](#) [IEEE Peer Review](#)

[Quick Links](#)

» Search Abst

Welcome to IEEE Xplore®

- Home
- What Can I Access?
- Log-out

Tables of Contents

- Journals & Magazines
- Conference Proceedings
- Standards

Search

- By Author
- Basic
- Advanced

Member Services

- Join IEEE
- Establish IEEE Web Account
- Access the IEEE Member Digital Library

Search Results [PDF FULL-TEXT 580 KB] [NEXT](#) [DOWNLOAD CITATION](#)

Order Reuse Permissions
RIGHTS LINK

A flexible architecture for multihop optical networks

[Jaekel, A.](#) [Bandyopadhyay, S.](#) [Sengupta, A.](#)
Sch. of Comput. Sci., Windsor Univ., Ont., Canada;
This paper appears in: Computer Communications and Networks, 1998. Proceedings. 7th International Conference on

Meeting Date: 10/12/1998 - 10/15/1998

Publication Date: 12-15 Oct. 1998

Location: Lafayette, LA USA

On page(s): 472 - 478

Reference Cited: 9

Number of Pages: xxii+929

Inspec Accession Number: 6226042

Abstract:

It is desirable to have low diameter logical topologies for multihop lightwave network. Researchers have investigated regular topologies for such networks. Only a few of them (e.g., GEMNET) are scalable to allow the addition of new nodes to an existing network. Adding new nodes to such networks requires a major change in routing scheme. For example, in a multistar implementation a large number of retuning of transmitters at receivers anti/or renumbering nodes are needed for GEMNET. We present a scalable logical topology which is not regular but it has a low diameter. This topology is interesting since it allows the network to be expanded indefinitely and new nodes can be added with a relatively small change to the network. We present the new topology, an algorithm to add nodes to the network and two routing schemes.

Index Terms:

[network topology](#) [optical fibre networks](#) [optical receivers](#) [optical transmitters](#) [telecommunication network routing](#) [wavelength division multiplexing](#) [GEMNET](#) [WDM](#) [algorithm](#) [flexible architecture](#) [low diameter logical topologies](#) [multihop lightwave networks](#) [multihop optical networks](#) [multistar implementation](#) [network nodes](#) [receivers](#) [regular topologies](#) [retuning routing scheme](#) [scalable logical topology](#) [transmitters](#)

Documents that cite this document

There are no citing documents available in IEEE Xplore at this time.

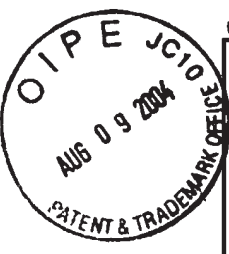
Search Results [PDF FULL-TEXT 580 KB] [NEXT](#) [DOWNLOAD CITATION](#)

[Home](#) | [Log-out](#) | [Journals](#) | [Conference Proceedings](#) | [Standards](#) | [Search by Author](#) | [Basic Search](#) | [Advanced Search](#) | [Join IEEE](#) | [Web Account](#) | [New this week](#) | [OPAC Linking Information](#) | [Your Feedback](#) | [Technical Support](#) | [Email Alerting](#) | [No Robots Please](#) | [Release Notes](#) | [IEEE Online Publications](#) | [Help](#) | [FAQ](#) | [Terms](#) | [Back to Top](#)

Copyright © 2004 IEEE — All rights reserved

08-11-04

IFW
2/53
B



PTO/SB/21 (02-04)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

TRANSMITTAL FORM <i>(to be used for all correspondence after initial filing)</i>	Application Number	09/629,570-Conf. #5411	
	Filing Date	July 31, 2000	
	First Named Inventor	Fred B. Holt	
	Art Unit	2153	
	Examiner Name	B. E. Edelman	
Total Number of Pages in This Submission	1	Attorney Docket Number	030048002US

ENCLOSURES (Check all that apply)

<input type="checkbox"/> Fee Transmittal Form	<input type="checkbox"/> Drawing(s)	<input type="checkbox"/> After Allowance communication to Technology Center (TC)
<input type="checkbox"/> Fee Attached	<input type="checkbox"/> Licensing-related Papers	<input type="checkbox"/> Appeal Communication to Board of Appeals and Interferences
<input type="checkbox"/> Amendment/Reply	<input type="checkbox"/> Petition	<input type="checkbox"/> Appeal Communication to TC (Appeal Notice, Brief, Reply Brief)
<input type="checkbox"/> After Final	<input type="checkbox"/> Petition to Convert to a Provisional Application	<input type="checkbox"/> Proprietary Information
<input type="checkbox"/> Affidavits/declaration(s)	<input type="checkbox"/> Power of Attorney, Revocation Change of Correspondence Address	<input type="checkbox"/> Status Letter
<input type="checkbox"/> Extension of Time Request	<input type="checkbox"/> Terminal Disclaimer	<input checked="" type="checkbox"/> Other Enclosure(s) (please identify below):
<input type="checkbox"/> Express Abandonment Request	<input type="checkbox"/> Request for Refund	Return Postcard
<input checked="" type="checkbox"/> Information Disclosure Statement	<input type="checkbox"/> CD, Number of CD(s) _____	
<input type="checkbox"/> Certified Copy of Priority Document(s)		
<input type="checkbox"/> Response to Missing Parts/ Incomplete Application		
<input type="checkbox"/> Response to Missing Parts under 37 CFR 1.52 or 1.53		
Remarks		

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm or Individual name	PERKINS COIE LLP Chun M. Ng - 36,878
Signature	
Date	8/6/04

I hereby certify that this correspondence is being deposited with the U.S. Postal Service as Express Mail, Airbill No. EV336668894US, in an envelope addressed to: MS Amendment, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on the date shown below.

Dated: 8/9/04 Signature: Melody J. Almborg (Melody J. Almborg)

I hereby certify that this correspondence is being deposited with the U.S. Postal Service as Express Mail, Airbill No. EV336668894US, in an envelope addressed to: MS Amendment, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on the date shown below.

Dated: 8/9/04

Signature:

Melody J. Almbert
(Melody J. Almbert)

Docket No.: 030048002US
Client Ref No. 99-481A

(PATENT)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:
Fred B. Holt et al.

Application No.: 09/629,570

Confirmation No.: 5411

Filed: July 31, 2000

Art Unit: 2153

For: JOINING A BROADCAST CHANNEL

Examiner: B. E. Edelman

INFORMATION DISCLOSURE STATEMENT (IDS)

MS Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

Pursuant to 37 CFR 1.56, 1.97 and 1.98, the attention of the Patent and Trademark Office is hereby directed to the references listed on the attached PTO/SB/08. It is respectfully requested that the information be expressly considered during the prosecution of this application, and that the references be made of record therein and appear among the "References Cited" on any patent to issue therefrom.

This Information Disclosure Statement is filed more than three months after the U.S. filing date, OR more than three months after the date of entry of the national stage of a PCT application, AND after the mailing date of the first Office Action on the merits, whichever occurs first, but before the mailing date of a Final Office Action or Notice of Allowance (37 CFR 1.97(c)).

Copies of the references have been provided.

08/11/2004 SSANDARA 00000010 09629570

01 FC:1806

180.00 OP

Application No.: 09/629,570

Docket No.: 030048002US

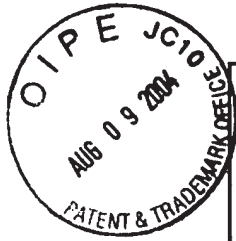
Our check in the amount of \$180.00 covering the fee set forth in 37 CFR 1.17(p) is enclosed. The Director is hereby authorized to charge any deficiency in the fees filed, asserted to be filed or which should have been filed herewith (or with any paper hereafter filed in this application by this firm) to our Deposit Account No. 50-0665, under Order No. 030048002US.

Dated: 8/6/04

Respectfully submitted,

By 
Chun M. Ng

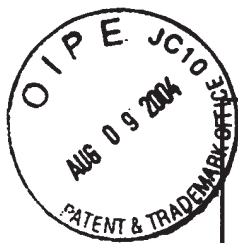
Registration No.: 36,878
PERKINS COIE LLP
P.O. Box 1247
Seattle, Washington 98111-1247
(206) 359-8000
(206) 359-7198 (Fax)
Attorneys for Applicant



Substitute for form 1449A/B/PTO				Complete if Known	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT <i>(Use as many sheets as necessary)</i>				Application Number	09/629,570-Conf. #5411
				Filing Date	July 31, 2000
				First Named Inventor	Fred B. Holt
				Art Unit	2153
				Examiner Name	B. E. Edelman
				Attorney Docket Number	030048002US
Sheet	1	of	2		

U.S. PATENT DOCUMENTS					
Examiner Initials*	Cite No. ¹	Document Number	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		Number-Kind Code ² (if known)			
		US-2002-0027896	03/2002	Hughes et al.	
		US-5,058,105	10/1991	Mansour et al.	
		US-5,079,767	01/1992	Perlman	
		US-5,345,558	09-06-1994	Opher	
		US-5,511,168	04-23-1996	Perlman	
		US-5,459,725	10/1995	Bodner et al.	
		US-5,568,487	10-22-1996	Sitbon	
		US-5,644,714	07/1997	Kikinis	
		US-5,757,795	05-26-1998	Schnell	
		US-5,850,592	12/1998	Ramanathan	
		US-5,925,097	07/1999	Gopinath et al.	
		US-5,946,316	08-31-1999	Chen et al.	
		US-5,953,318	09/1999	Nattkemper et al.	
		US-5,970,232	10/1999	Passint et al.	
		US-6,073,177	06-06-2000	Hebel et al.	
		US-6,115,580	09/2000	Chuprun et al.	
		US-6,151,633	11-21-2000	Hurst	
		US-6,167,432	12/2000	Jiang	
		US-6,173,314	01/2001	Kurashima et al.	
		US-6,195,366	02-27-2001	Kayashima	
		US-6,252,884	06-26-2001	Hunter	
		US-6,269,080	07-31-2001	Kumar	
		US-6,272,548	08/2001	Cotter et al.	
		US-6,321,270	11/2001	Crawley	

Examiner Signature		Date Considered	
--------------------	--	-----------------	--



Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/B/PTO			Complete if Known	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT (Use as many sheets as necessary)			Application Number	09/629,570-Conf. #5411
			Filing Date	July 31, 2000
			First Named Inventor	Fred B. Holt
			Art Unit	2153
			Examiner Name	B. E. Edelman
			Attorney Docket Number	030048002US
Sheet	2	of	2	

		US-6,353,599	03-05-2002	Bi et al.	
		US-6,415,270	07-02-2002	Rackson	
		US-6,434,622	08-13-2002	Monteiro	
		US-6,463,078	10/2002	Engstrom et al.	
		US-6,499,251	09-19-2002	Weder	
		US-6,524,189	02/2003	Rautila	
		US-6,611,872	08/2003	McCanne	
		US-6,618,752	09-09-2003	Moore et al.	
		US-6,701,344	03-02-2004	Holt	


FOREIGN PATENT DOCUMENTS							
Examiner Initials*	Cite No. ¹	Foreign Patent Document		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T ⁶
		Country Code ³	Number ⁴ -Kind Code ⁵ (if known)				

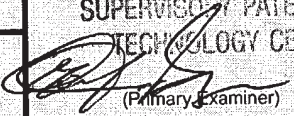
*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. ¹ Applicant's unique citation designation number (optional). ² See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. ³ Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). ⁴ For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. ⁵ Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. ⁶ Applicant is to place a check mark here if English language Translation is attached.

NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	Cite No. ¹	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T ²
		YAVATKAR et al., "A reliable Dissemination Protocol for Interactive Collaborative Applications," Proc. ACM Multimedia, 1995, p. 333-344; http://citeseer.nj.nec.com/article/yavatkar95reliable.html	
		Business Wire, "Boeing Panthesis Complete SWAN Transaction," July 22, 2002, pp 1ff	
		PR Newswire, "Microsoft Annouces Launch Date for UltraCorps, Its Second Premium Title for the Internet Gaming Zone," March 27, 1998, pp 1 ff	
		PR Newswire, "Microsoft Boosts Accessibility to Internet Gaming Zone with Latest Release," April 27, 1998, pp 1ff	
		PEERCY et al., "Distributed Algorithms for Shortest-Path, Deadlock-Free Routing and Broadcasting in Arbitrarily Faulty Hypercubes," June 1990, 20th International Symposium on Fault-Tolerant Computing, 1990, pp-218-225	
		AZAR et al., "Routing Strategies for Fast Networks," May 1992, INFOCOM '92 Eleventh Annual Joint Conference of the IEEE Computer Communications Societies, vol. 1, 170-179###	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

Examiner Signature		Date Considered	
--------------------	--	-----------------	--

Issue Classification 	Application No.	Applicant(s)	
	09/629,570	HOLT ET AL.	
	Examiner	Art Unit	
	Bradley Edelman	2153	

ISSUE CLASSIFICATION										
ORIGINAL			CROSS REFERENCE(S)							
CLASS	SUBCLASS		CLASS	SUBCLASS (ONE SUBCLASS PER BLOCK)						
709	221		709	252	243	227				
INTERNATIONAL CLASSIFICATION										
		/								
		/								
		/								
		/								
		/								
<i>Bradley Edelman</i> 8/13/04 (Assistant Examiner) (Date)			GLENTON B. BURGESS SUPERVISORY PATENT EXAMINER TECHNOLOGY CENTER 2100  (Primary Examiner) 8/19/04 (Date)				Total Claims Allowed: 17			
(Legal Instruments Examiner) (Date)							O.G. Print Claim(s) 1		O.G. Print Fig. 11	

<input checked="" type="checkbox"/> Claims renumbered in the same order as presented by applicant		<input type="checkbox"/> CPA		<input type="checkbox"/> T.D.		<input type="checkbox"/> R.1.47	
Final	Original	Final	Original	Final	Original	Final	Original
	1		31		61		91
	2		32		62		92
	3		33		63		93
	4		34		64		94
	5		35		65		95
	6		36		66		96
	7		37		67		97
	8		38		68		98
	9		39		69		99
	10		40		70		100
	11		41		71		101
	12		42		72		102
	13		43		73		103
	14		44		74		104
	15		45		75		105
	16		46		76		106
	17		47		77		107
	18		48		78		108
	19		49		79		109
	20		50		80		110
	21		51		81		111
	22		52		82		112
	23		53		83		113
	24		54		84		114
	25		55		85		115
	26		56		86		116
	27		57		87		117
	28		58		88		118
	29		59		89		119
	30		60		90		120
							121
							122
							123
							124
							125
							126
							127
							128
							129
							130
							131
							132
							133
							134
							135
							136
							137
							138
							139
							140
							141
							142
							143
							144
							145
							146
							147
							148
							149
							150
							151
							152
							153
							154
							155
							156
							157
							158
							159
							160
							161
							162
							163
							164
							165
							166
							167
							168
							169
							170
							171
							172
							173
							174
							175
							176
							177
							178
							179
							180
							181
							182
							183
							184
							185
							186
							187
							188
							189
							190
							191
							192
							193
							194
							195
							196
							197
							198
							199
							200
							201
							202
							203
							204
							205
							206
							207
							208
							209
							210

Index of Claims



Application No.

09/629,570

Examiner

Bradley Edelman

Applicant(s)

HOLT ET AL.

Art Unit

2153

√	Rejected
=	Allowed

-	(Through numeral) Cancelled
+	Restricted

N	Non-Elected
I	Interference

A	Appeal
O	Objected

Claim		Date		Claim		Date		Claim		Date	
Final	Original			Final	Original			Final	Original		
1	(1)			51				101			
2	2			52				102			
3	3			53				103			
4	4			54				104			
5	5			55				105			
6	6			56				106			
7	7			57				107			
8	8			58				108			
9	9			59				109			
10	10			60				110			
11	11			61				111			
12	12			62				112			
13	13			63				113			
14	(14)			64				114			
15	15			65				115			
16	16			66				116			
17	17			67				117			
18				68				118			
19				69				119			
20				70				120			
21				71				121			
22				72				122			
23				73				123			
24				74				124			
25				75				125			
26				76				126			
27				77				127			
28				78				128			
29				79				129			
30				80				130			
31				81				131			
32				82				132			
33				83				133			
34				84				134			
35				85				135			
36				86				136			
37				87				137			
38				88				138			
39				89				139			
40				90				140			
41				91				141			
42				92				142			
43				93				143			
44				94				144			
45				95				145			
46				96				146			
47				97				147			
48				98				148			
49				99				149			
50				100				150			

Search Notes



Application No.

09/629,570

Examiner

Bradley Edelman

Applicant(s)

HOLT ET AL.

Art Unit

2153

SEARCHED

Class	Subclass	Date	Examiner
709	221, 220 252, 243 227	8/13/04	BE

SEARCH NOTES
(INCLUDING SEARCH STRATEGY)

	DATE	EXMR
Updated EAST Search	8/13/04	BE

INTERFERENCE SEARCHED

Class	Subclass	Date	Examiner
709	221, 252	8/13/04	BE
709	243, 227	8/13/04	BE



UNITED STATES PATENT AND TRADEMARK OFFICE

3

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

NOTICE OF ALLOWANCE AND FEE(S) DUE

25096 7590 08/26/2004

PERKINS COIE LLP
PATENT-SEA
P.O. BOX 1247
SEATTLE, WA 98111-1247

EXAMINER

EDELMAN, BRADLEY E

ART UNIT PAPER NUMBER

2153

DATE MAILED: 08/26/2004

Table with 5 columns: APPLICATION NO., FILING DATE, FIRST NAMED INVENTOR, ATTORNEY DOCKET NO., CONFIRMATION NO.
Values: 09/629,570, 07/31/2000, Fred B. Holt, 030048002US, 5411

TITLE OF INVENTION: JOINING A BROADCAST CHANNEL

Table with 6 columns: APPLN. TYPE, SMALL ENTITY, ISSUE FEE, PUBLICATION FEE, TOTAL FEE(S) DUE, DATE DUE
Values: nonprovisional, NO, \$1330, \$0, \$1330, 11/26/2004

THE APPLICATION IDENTIFIED ABOVE HAS BEEN EXAMINED AND IS ALLOWED FOR ISSUANCE AS A PATENT. PROSECUTION ON THE MERITS IS CLOSED. THIS NOTICE OF ALLOWANCE IS NOT A GRANT OF PATENT RIGHTS. THIS APPLICATION IS SUBJECT TO WITHDRAWAL FROM ISSUE AT THE INITIATIVE OF THE OFFICE OR UPON PETITION BY THE APPLICANT. SEE 37 CFR 1.313 AND MPEP 1308.

THE ISSUE FEE AND PUBLICATION FEE (IF REQUIRED) MUST BE PAID WITHIN THREE MONTHS FROM THE MAILING DATE OF THIS NOTICE OR THIS APPLICATION SHALL BE REGARDED AS ABANDONED. THIS STATUTORY PERIOD CANNOT BE EXTENDED. SEE 35 U.S.C. 151. THE ISSUE FEE DUE INDICATED ABOVE REFLECTS A CREDIT FOR ANY PREVIOUSLY PAID ISSUE FEE APPLIED IN THIS APPLICATION. THE PTOL-85B (OR AN EQUIVALENT) MUST BE RETURNED WITHIN THIS PERIOD EVEN IF NO FEE IS DUE OR THE APPLICATION WILL BE REGARDED AS ABANDONED.

HOW TO REPLY TO THIS NOTICE:

I. Review the SMALL ENTITY status shown above.

If the SMALL ENTITY is shown as YES, verify your current SMALL ENTITY status:

- A. If the status is the same, pay the TOTAL FEE(S) DUE shown above.
B. If the status above is to be removed, check box 5b on Part B - Fee(s) Transmittal and pay the PUBLICATION FEE (if required) and twice the amount of the ISSUE FEE shown above, or

If the SMALL ENTITY is shown as NO:

- A. Pay TOTAL FEE(S) DUE shown above, or
B. If applicant claimed SMALL ENTITY status before, or is now claiming SMALL ENTITY status, check box 5a on Part B - Fee(s) Transmittal and pay the PUBLICATION FEE (if required) and 1/2 the ISSUE FEE shown above.

II. PART B - FEE(S) TRANSMITTAL should be completed and returned to the United States Patent and Trademark Office (USPTO) with your ISSUE FEE and PUBLICATION FEE (if required). Even if the fee(s) have already been paid, Part B - Fee(s) Transmittal should be completed and returned. If you are charging the fee(s) to your deposit account, section "4b" of Part B - Fee(s) Transmittal should be completed and an extra copy of the form should be submitted.

III. All communications regarding this application must give the application number. Please direct all communications prior to issuance to Mail Stop ISSUE FEE unless advised to the contrary.

IMPORTANT REMINDER: Utility patents issuing on applications filed on or after Dec. 12, 1980 may require payment of maintenance fees. It is patentee's responsibility to ensure timely payment of maintenance fees when due.

Handwritten mark

PART B - FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), to: **Mail** **Mail Stop ISSUE FEE**
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450
or Fax (703) 746-4000

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 5 should be completed where appropriate. All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Use Block 1 for any change of address)

25096 7590 08/26/2004

PERKINS COIE LLP
 PATENT-SEA
 P.O. BOX 1247
 SEATTLE, WA 98111-1247

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

Certificate of Mailing or Transmission

I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Mail Stop ISSUE FEE address above, or being facsimile transmitted to the USPTO (703) 746-4000, on the date indicated below.

_____	(Depositor's name)
_____	(Signature)
_____	(Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/629,570	07/31/2000	Fred B. Holt	030048002US	5411

TITLE OF INVENTION: JOINING A BROADCAST CHANNEL

APPLN. TYPE	SMALL ENTITY	ISSUE FEE	PUBLICATION FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	NO	\$1330	\$0	\$1330	11/26/2004

EXAMINER	ART UNIT	CLASS-SUBCLASS
EDELMAN, BRADLEY E	2153	709-221000

<p>1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).</p> <p><input type="checkbox"/> Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.</p> <p><input type="checkbox"/> "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. Use of a Customer Number is required.</p>	<p>2. For printing on the patent front page, list</p> <p>(1) the names of up to 3 registered patent attorneys or agents OR, alternatively, _____ 1</p> <p>(2) the name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed. _____ 2</p> <p>_____ 3</p>
--	---

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. If an assignee is identified below, the document has been filed for recordation as set forth in 37 CFR 3.111. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE _____ (B) RESIDENCE: (CITY and STATE OR COUNTRY) _____

Please check the appropriate assignee category or categories (will not be printed on the patent): Individual Corporation or other private group entity Government

<p>4a. The following fee(s) are enclosed:</p> <p><input type="checkbox"/> Issue Fee</p> <p><input type="checkbox"/> Publication Fee (No small entity discount permitted)</p> <p><input type="checkbox"/> Advance Order - # of Copies _____</p>	<p>4b. Payment of Fee(s):</p> <p><input type="checkbox"/> A check in the amount of the fee(s) is enclosed.</p> <p><input type="checkbox"/> Payment by credit card. Form PTO-2038 is attached.</p> <p><input type="checkbox"/> The Director is hereby authorized by charge the required fee(s), or credit any overpayment, to Deposit Account Number _____ (enclose an extra copy of this form).</p>
--	---

5. Change in Entity Status (from status indicated above)

a. Applicant claims SMALL ENTITY status. See 37 CFR 1.27. b. Applicant is no longer claiming SMALL ENTITY status. See 37 CFR 1.27(g)(2).

The Director of the USPTO is requested to apply the Issue Fee and Publication Fee (if any) or to re-apply any previously paid issue fee to the application identified above. NOTE: The Issue Fee and Publication Fee (if required) will not be accepted from anyone other than the applicant, a registered attorney or agent; or the assignee or other party in interest as shown by the records of the United States Patent and Trademark Office.

Authorized Signature _____ Date _____

Typed or printed name _____ Registration No. _____

This collection of information is required by 37 CFR 1.311. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, Virginia 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

Table with 5 columns: APPLICATION NO., FILING DATE, FIRST NAMED INVENTOR, ATTORNEY DOCKET NO., CONFIRMATION NO.
Row 1: 09/629,570, 07/31/2000, Fred B. Holt, 030048002US, 5411
Row 2: 25096, 7590, 08/26/2004, [EXAMINER: EDELMAN, BRADLEY E], [ART UNIT: 2153, PAPER NUMBER]

PERKINS COIE LLP
PATENT-SEA
P.O. BOX 1247
SEATTLE, WA 98111-1247

DATE MAILED: 08/26/2004

Determination of Patent Term Adjustment under 35 U.S.C. 154 (b)
(application filed on or after May 29, 2000)

The Patent Term Adjustment to date is 719 day(s). If the issue fee is paid on the date that is three months after the mailing date of this notice and the patent issues on the Tuesday before the date that is 28 weeks (six and a half months) after the mailing date of this notice, the Patent Term Adjustment will be 719 day(s).

If a Continued Prosecution Application (CPA) was filed in the above-identified application, the filing date that determines Patent Term Adjustment is the filing date of the most recent CPA.

Applicant will be able to obtain more detailed information by accessing the Patent Application Information Retrieval (PAIR) WEB site (http://pair.uspto.gov).

Any questions regarding the Patent Term Extension or Adjustment determination should be directed to the Office of Patent Legal Administration at (703) 305-1383. Questions relating to issue and publication fee payments should be directed to the Customer Service Center of the Office of Patent Publication at (703) 305-8283.



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

Table with columns: APPLICATION NO., FILING DATE, FIRST NAMED INVENTOR, ATTORNEY DOCKET NO., CONFIRMATION NO., EXAMINER, ART UNIT, PAPER NUMBER. Includes details for Fred B. Holt and Perkins Coie LLP.

Notice of Fee Increase on October 1, 2004

If a reply to a "Notice of Allowance and Fee(s) Due" is filed in the Office on or after October 1, 2004, then the amount due will be higher than that set forth in the "Notice of Allowance and Fee(s) Due" because an increase in fees effective on October 1, 2004 is anticipated.

The current fee schedule is accessible from WEB site (http://www.uspto.gov/main/howtofees.htm).

If the fee paid is the amount shown on the "Notice of Allowance and Fee(s) Due" but not the correct amount in view of the fee increase, a "Notice of Pay Balance of Issue Fee" will be mailed to applicant.

Effective October 1, 2004, 37 CFR 1.18 is proposed to be amended by revising paragraphs (a) through (c) to read as set forth below.

Section 1.18 Patent post allowance (including issue) fees.

- (a) Issue fee for issuing each original or reissue patent, except a design or plant patent: By a small entity (Sec. 1.27(a))... \$670.00
(b) Issue fee for issuing a design patent: By a small entity (Sec. 1.27(a))... \$245.00
(c) Issue fee for issuing a plant patent: By a small entity (Sec. 1.27(a))... \$325.00

Questions relating to issue and publication fee payments should be directed to the Customer Service Center of the Office of Patent Publication at (703) 305-8283.

PL

Notice of Allowability

Application No.	Applicant(s)	
09/629,570	HOLT ET AL.	
Examiner	Art Unit	
Bradley Edelman	2153	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address--

All claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice of Allowance (PTOL-85) or other appropriate communication will be mailed in due course. **THIS NOTICE OF ALLOWABILITY IS NOT A GRANT OF PATENT RIGHTS.** This application is subject to withdrawal from issue at the initiative of the Office or upon petition by the applicant. See 37 CFR 1.313 and MPEP 1308.

1. This communication is responsive to the amendment filed on May 10, 2004.
2. The allowed claim(s) is/are 1-17.
3. The drawings filed on 31 July 2000 are accepted by the Examiner.
4. Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
 - a) All b) Some* c) None of the:
 1. Certified copies of the priority documents have been received.
 2. Certified copies of the priority documents have been received in Application No. _____.
 3. Copies of the certified copies of the priority documents have been received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

* Certified copies not received: _____.

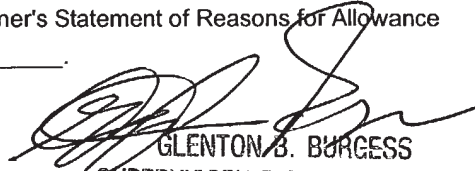
Applicant has THREE MONTHS FROM THE "MAILING DATE" of this communication to file a reply complying with the requirements noted below. Failure to timely comply will result in ABANDONMENT of this application. **THIS THREE-MONTH PERIOD IS NOT EXTENDABLE.**

5. A SUBSTITUTE OATH OR DECLARATION must be submitted. Note the attached EXAMINER'S AMENDMENT or NOTICE OF INFORMAL PATENT APPLICATION (PTO-152) which gives reason(s) why the oath or declaration is deficient.
6. CORRECTED DRAWINGS (as "replacement sheets") must be submitted.
 - (a) including changes required by the Notice of Draftsperson's Patent Drawing Review (PTO-948) attached
 - 1) hereto or 2) to Paper No./Mail Date _____.
 - (b) including changes required by the attached Examiner's Amendment / Comment or in the Office action of Paper No./Mail Date _____.

Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the drawings in the front (not the back) of each sheet. Replacement sheet(s) should be labeled as such in the header according to 37 CFR 1.121(d).
7. DEPOSIT OF and/or INFORMATION about the deposit of BIOLOGICAL MATERIAL must be submitted. Note the attached Examiner's comment regarding REQUIREMENT FOR THE DEPOSIT OF BIOLOGICAL MATERIAL.

Attachment(s)

- | | |
|---|---|
| 1. <input type="checkbox"/> Notice of References Cited (PTO-892) | 5. <input type="checkbox"/> Notice of Informal Patent Application (PTO-152) |
| 2. <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948) | 6. <input checked="" type="checkbox"/> Interview Summary (PTO-413),
Paper No./Mail Date _____. |
| 3. <input type="checkbox"/> Information Disclosure Statements (PTO-1449 or PTO/SB/08),
Paper No./Mail Date _____ | 7. <input checked="" type="checkbox"/> Examiner's Amendment/Comment |
| 4. <input type="checkbox"/> Examiner's Comment Regarding Requirement for Deposit
of Biological Material | 8. <input checked="" type="checkbox"/> Examiner's Statement of Reasons for Allowance |
| | 9. <input type="checkbox"/> Other _____. |


GLENTON B. BURGESS
 SUPERVISORY PATENT EXAMINER
 TECHNOLOGY CENTER 2100

Art Unit: 2153

EXAMINER'S AMENDMENT

An examiner's amendment to the record appears below. Should the changes and/or additions be unacceptable to applicant, an amendment may be filed as provided by 37 CFR 1.312. To ensure consideration of such an amendment, it MUST be submitted no later than the payment of the issue fee.

Authorization for the claim cancellation and re-writing of the abstract in this examiner's amendment was given in a telephone interview with Chun Ng on August 13, 2004.

The application has been amended as follows:

IN THE CLAIMS:

- a. Cancel claims 32-40.

IN THE SPECIFICATION:

- a. In the "Cross-Reference to Related Applications" section of the Amendment filed on May 10, 2004, delete all parenthetical references to Attorney Docket Numbers.
- b. In the "Cross-Reference to Related Applications" section of the Amendment filed on May 10, 2004, on line 12, after the phrase "No. 09/629,043, entitled 'AN INFORMATION DELIVERY SERVICE,' filed on July 31, 2000," insert the phrase --, now U.S. Patent No. 6,714,966--.

IN THE ABSTRACT:

Replace the abstract with the abstract that appears on the following page:

Art Unit: 2153

Abstract:

A technique for adding a participant to a network is provided. This technique allows for the simultaneous sharing of information among many participants in a network without the placement of a high overhead on the underlying communication network. To connect to the broadcast channel, a seeking computer first locates a computer that is fully connected to the broadcast channel. The seeking computer then establishes a connection with a number of the computers that are already connected to the broadcast channel. The technique for adding a participant to a network includes identifying a pair of participants that are connected to the network, disconnecting the participants of the identified pair from each other, and connecting each participant of the identified pair of participants to the added participant.

Allowable Subject Matter

Claims 1-17 are allowed.

The following is an examiner's statement of reasons for allowance: the claims are allowed for the reasons set forth by Applicant in Applicant's response filed on May 10, 2004.

Any comments considered necessary by applicant must be submitted no later than the payment of the issue fee and, to avoid processing delays, should preferably accompany the issue fee. Such submissions should be clearly labeled "Comments on Statement of Reasons for Allowance."

Conclusion

The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Bradley Edelman whose telephone number is 703-306-3041. The examiner can normally be reached from 9 a.m. to 5 p.m.

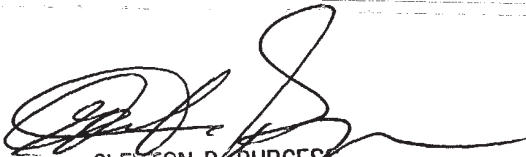
If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Glen Burgess can be reached on 703-305-4792. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Art Unit: 2153

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

BE

August 13, 2004



CLENTON B. BURGESS
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100