On Buffer-Economical Store-and-Forward Deadlock Prevention

Baruch Awerbuch, Shay Kutten, and David Peleg

Abstract— **This note deals with store-and-forward deadlock prevention in communication networks. The approach we adopt is that of establishing buffer classes in order to prevent cyclic waiting chains. This type of solutions usually tends to require many buffers. The main contribution of the current note is in showing that the number of required buffers can be reduced considerably by employing a hierarchical organization of the network. The note proposes a new hierarchical scheme for arbitrary networks, that features a tradeoff between the communication overhead and the buffer requirements of the routing. This tradeoff can be shown to be close to optimal.**

I. Introduction

Overcoming store-and-forward deadlocks is one of the major problems in the design of routing protocols for communication networks. Informally, a store-and-forward deadlock occurs at some set N of nodes when all the buffers of these nodes are full, and each of the messages occupying these buffers needs to be forwarded to some other node in the set N. Clearly, then, all of these messages are locked in a vicious circle, since the only way to forward a message is to have a free buffer in the next node on its route, and the only way to release a buffer is to forward the message currently occupying it. Avoidance or fast resolution of such store-and-forward deadlocks is essential for efficient utilization of available network resources.

In this note we concentrate on one approach to deadlock prevention that has been extensively studied in the literature. This approach involves solutions based on dividing the buffer pool into buffer classes and utilizing these classes so as to prevent cyclic waiting chains [6], [10], [13], [4]. Some of these solutions are based on restricting the family of allowed routes in order to avoid deadlocks.

The problem addressed in this note involves two basic characteristics of deadlock prevention policies, namely, their buffer requirements and the quality of the allowed routes. In order to present the problem, let us compare these characteristics as manifested in routing schemes founded on two opposing philosophies for buffer allocation proposed in the literature. The first philosophy, governing the schemes of [7], [6], [10], features complete independence of the network topology (except for knowing the network diameter D). Generally speaking, schemes based on this philosophy do not interfere with the routing itself, and thus allow sending packets along shortest paths, for instance. However, at least cn buffers per node, for some constant $c > 0$, may be required in an n-vertex network.

The opposite philosophy, taken for instance in [5], is to make full use of knowledge of the network topology. A possible scheme based on this philosophy would be to restrict the routing to a *tree* spanning the network.

This latter philosophy usually requires much fewer buffers. For instance, the tree scheme allows us to use only two buffers per vertex, one for *inbound traffic* and one for *outbound traffic*. However, this philosophy has two major disadvantages. The first is its sensitivity to topological changes. The second disadvantage is that it may require us to take routes that are sometimes far from optimal, even when the tree is carefully selected. For instance, suppose that the tree is a *shortest-path tree*, namely, a tree F rooted at some fixed vertex v with the property that for every other vertex w in the network G, the path from v to w in F is a shortest path in G. While this choice guarantees efficient routing to and from v, it may still happen that two other vertices w, w' end up using a path that is much longer than the shortest. In fact, when the edge lengths are allowed to vary arbitrarily, the ratio between the lengths of the paths connecting w to w' in the tree F and in the network G may be arbitrarily bad. Even when all edge weights are assumed to be equal, this ratio may be as bad as $n - 1$, as can be verified by considering the example of a network G whose topology is a ring.

The problem we set out to investigate in this note is the possible tradeoff between the two extreme methods described above. Essentially, our results indicate that considerable savings in buffer requirements can be achieved by making relatively modest compromises in the lengths of the routes.

We follow the hierarchical paradigm advocated in [4]. (In fact, the notion of distinguishing routing within a tree from routing between trees is already present in [10].) This approach has been proposed for networks that are originally structured in a hierarchical fashion, with communicating nodes naturally clustered into groups. The main technical novelty of our solutions is in proposing a method for covering an *arbitrary* network by a hierarchy of *tree covers*. Each level of the hierarchy provides a cover of the network, composed of a collection of (possibly overlapping) subtrees, whose union contains all the nodes of the network. Each of the covers has the additional property that a given node belongs to few

Paper approved by J. M. Jaffe, the Editor for Switching and Routing of the IEEE Communications Society. Manuscript received June 13, 1990; revised April 10, 1991 and January 27, 1993. The work of B. Awerbuch was supported by the Air Force under Contract TND GAFOSR-86-0078, ARO Contract DAAL 03-86-K-0171, NSF Contract CCR8611442, and a special grant from IBM. The work of D. Peleg was supported in part by an Allon Fellowship, a Bantrell Fellowship, and by a Walter and Elise Haas Career Development Award. This paper was presented in part at INFO COM, Bal-Harbour, FL, April 1991.

B. Awerbuch is with the Department of Mathematics and the Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139 USA.

S. Kutten is with IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA.

D. Peleg is with the Department of Applied Mathematics, the Weizmann Institute, Rehovot 76100, Israel.

IEEE Log Number 9404728.

	Communication Overhead	Buffers per Nodes
[Gop84,MS80]	$O(1)$	$O(n)$
Tree	$O(D)$	$O(1)$
This paper	$O(k)$	$O(kn^{1/k} \log D)$

Fig. 1. The (multiplicative) communication overhead and number of buffers per node involved in various deadlock-prevention methods. Here n is the number of nodes, $k > 0$ is a parameter, and D is the diameter of the network, i.e., the maximal *weighted* distance between any two nodes.

subtrees. The hierarchical decomposition makes no *a priori* assumptions on the structure of the network. The resulting tree cover is then used to design a deadlock-free tree-based routing scheme on each subtree, independently from all other subtrees.

Our solution exhibits a tradeoff between the quality of the routing and the buffer requirements of the scheme. Using a k level hierarchy (with $k \geq 1$ an integer parameter) yields routes that are at most $8k$ times longer than the optimal, and requires $kn^{1/k} \log D(G)$ buffers per node, where $D(G)$ denotes the diameter of the network G.

A comparison between our method and various existing ones, based on the two philosophies discussed above, is given in Fig. 1. The communication measure represents multiplicative overhead, i.e., the ratio between the amount of communication used for transmitting a message by the method at hand and the optimal communication cost.

We would like to stress that the hierarchical approach described herein seems to be rather general, and has several other useful applications in the area of distributed network algorithms. For more on the hierarchical approach, related partitioning and covering techniques and various applications, see [1], [2], [8], [11], [12].

II. Definitions

We consider the standard model of a point-to-point communication network, described by an undirected graph $G = (V, E)$. The vertices V represent the processors of the network and the edges E represent bidirectional communication channels between the vertices. A vertex may communicate directly only with its neighbors, and messages between nonadjacent vertices are sent along some path connecting them in the network.

We assume the existence of a *weight* function $w: E \rightarrow \mathcal{R}^+$, assigning an arbitrary positive weight $w(e)$ to each edge $e \in E$. For two vertices u, w in a graph G let $\text{dist}_G(u, w)$ denote the (weighted) length of a shortest path in G between those vertices, i.e., the cost of the cheapest path connecting them, where the cost of a path $(e_1, \cdots, e_s)$ is $\sum_{1 \leq i \leq s} w(e_i)$.

Let $D = D(G)$ denote the *diameter* of the network, i.e., $\max_{u, v \in V}(\text{dist}(u, v))$. For a vertex $v \in V$, let $R(v, G) = \max_{w \in V}(\text{dist}_G(v, w))$. Let $R(G)$ denote the *radius* of the network, i.e., $\min_{v \in V}(R(v, G))$. A *center* of G is any vertex v realizing the radius of G (i.e., such that $R(v, G) = R(G)$).

Given a set of vertices $S \subseteq V$, let $G(S)$ denote the subgraph induced by S in G. A *cluster* is a subset of vertices $S \subseteq V$ such that $G(S)$ is connected. A *cover* is a collection of clusters $\mathcal{S} = \{S_1, \cdots, S_m\}$ such that $\bigcup_i S_i = V$. Given a collection of clusters $\mathcal{S}$, let $R(\mathcal{S}) = \max_i R(G(Si))$.

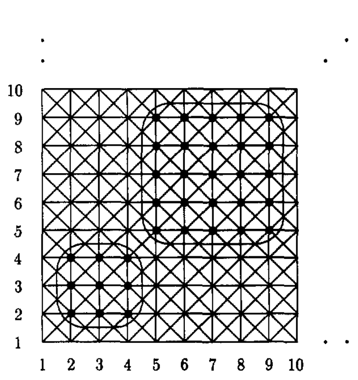

Fig. 2. The lower left corner of the 19×19 grid G_{19}, and the neighborhoods $N_1(3, 3)$ and $N_2(7, 7)$.

Given two covers $\mathcal{S} = \{S_1, \cdots, S_m\}$ and $\mathcal{T} = \{T_1, \cdots, T_l\}$, we say that $\mathcal{T}$ *coarsens* $\mathcal{S}$ if each cluster of $\mathcal{S}$ is entirely "covered" by some cluster of $\mathcal{T}$, i.e., for every $S_i \in \mathcal{S}$ there exists a $T_j \in \mathcal{T}$ such that $S_i \subseteq T_j$.

Denote the *m − neighborhood* of a vertex v by

$$N_m(v) = \{w \mid \text{dist}(w, v) \leq m\}.$$

The *m − neighborhood cover* of the graph G is the collection of neighborhood clusters

$$\mathcal{N}_m(V) = \{N_m(v) \mid v \in V\}.$$

This is clearly a cover for G since, in particular, $v \in N_m(v)$ for every $v \in V$ and $m \geq 0$. The radius of $\mathcal{N}_m(V)$ is m.

The construction relies heavily on the following lemma, proved in [2], [3].

Lemma 2.1 [2], [3]: Given a graph $G = (V, E)$, $|V| = n$, a cover $\mathcal{S}$ and an integer $k \geq 1$, it is possible to construct a cover $\mathcal{T}$ that satisfies the following properties:

1. $\mathcal{T}$ coarsens $\mathcal{S}$,
2. $R(\mathcal{T}) \leq (2k - 1)R(\mathcal{S})$, and
3. every vertex $v \in V$ occurs in at most $\lfloor k|\mathcal{S}|^{1/k} \rfloor$ clusters of $\mathcal{T}$.

Example : Consider the 19×19 grid $G_{19} = (V, E)$, $V = \{(i, j) \mid 1 \leq i, j \leq 19\}$, with each vertex connected to its (up to) eight neighbors (see Fig. 2). This network has $n = 361$ vertices. Assume all edges have unit cost. Then $N_1(v)$, respectively, $N_2(v)$, consists of the 3×3 (resp., 5×5) subgrid centered at v (with the obvious modifications for vertices on the borders of the grid). For example, Fig. 2 describes also the 1-neighborhood $N_1(3, 3)$ and the 2-neighborhood $N_2(7, 7)$.

Now consider the initial cover $\mathcal{S} = \mathcal{N}_2(V)$. Note that in this cover, each vertex occurs in at most 25 clusters. Given a parameter k, our purpose is to find a cover $\mathcal{T}$ that coarsens $\mathcal{S}$, such that the radius of clusters in $\mathcal{T}$ is at most $(2k - 1)R(\mathcal{S})$, and the number of $\mathcal{T}$ clusters in which any particular vertex occurs is at most $\lfloor k|\mathcal{S}|^{1/k} \rfloor$. For $k = 2$, $\mathcal{S}$ itself satisfies the requirements, since the desired bound on the number of occurrences is $\lfloor 2\sqrt{361} \rfloor = 38$. For $k = 3$, we need to find a coarsening cover $\mathcal{T}$ such that $R(\mathcal{T}) \leq (2 \cdot 3 - 1)R(\mathcal{S}) = 10$, and the number of $\mathcal{T}$ clusters in which any particular vertex occurs is at most $\lfloor 3 \cdot 361^{1/3} \rfloor = 21$. A possible solution,

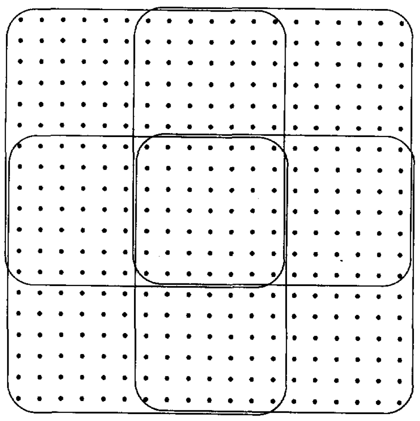

Fig. 3. The cover $\mathcal{T}$ for $\mathcal{N}_2(V)$ on G_{19}.

depicted by the four ovals in Fig. 3, is

$$\mathcal{T} = \{N_6(6i+1, 6j+1) \mid 1 \le i, j \le 2\}.$$

Note that this cover easily meets the requirement on the number of occurrences, and furthermore, its radius is only 6.

III. Tree-Cover Routing

In this section we present our routing policy and analyze its behavior. We first use Lemma 2.1 to construct our desired *tree cover.*

Lemma 3.1: For every undirected graph $G(V, E)$ and integer $k \ge 1$, it is possible to construct a *tree cover*, namely, a collection $\mathcal{F}_k$ of trees in G, that satisfies the following properties:

1. For every two nodes $u, v \in V$, there exists a tree $F \in \mathcal{F}_k$ such that $\text{dist}_F(u, v) \le 8k \cdot \text{dist}_G(u, v)$, i.e., the distance between u and v in that tree is at most $8k$ times longer than their distance in the original graph.
2. Every node belongs to at most $\lceil k \cdot n^{1/k} \cdot \log D(G) \rceil$ different trees in $\mathcal{F}_k$.

Proof: Let $\delta = \lceil \log D(G) \rceil$. The collection $\mathcal{F}_k$ consists of δ sets of trees, denoted $\mathcal{F}_k^i$, $i = 1, \cdots, \delta$. Each set $\mathcal{F}_k^i$ is constructed as follows. We start by constructing the 2^i-neighborhood cover of G, $\mathcal{S}_i = \mathcal{N}_{2^i}(V)$, and computing a coarsening cover $\mathcal{T}_i$ for $\mathcal{S}_i$ as in Lemma 2.1. Next, we select in each cluster $T \in \mathcal{T}_i$ a shortest-path tree $F(T)$ rooted at some center of the cluster and spanning T. Finally we set

$$\mathcal{F}_k^i = \{F(T) \mid T \in \mathcal{T}_i\}.$$

The entire collection of trees is defined as

$$\mathcal{F}_k = \bigcup_{i=1}^{\delta} \mathcal{F}_k^i.$$

Let us now prove the first claim of the lemma. Consider two nodes $u, v \in V$, and suppose $2^{i-1} < \text{dist}_G(u, v) \le 2^i$ for some integer $i \ge 1$. By Property (1) of Lemma 2.1, there is a cluster $T \in \mathcal{T}_i$ such that $N_i(v) \subseteq T$, and hence $u \in T$. Consequently, the tree $F(T) \in \mathcal{F}_k^i$ spans both u and v. Furthermore, since it a shortest-path tree rooted at the center of the cluster T, its depth equals $R(T)$, the radius of T. Consequently, it follows from Property (2) of Lemma 2.1 that $\text{dist}_F(u, v) \le 2(2k-1)2^i \le 8k \cdot \text{dist}_G(u, v)$.

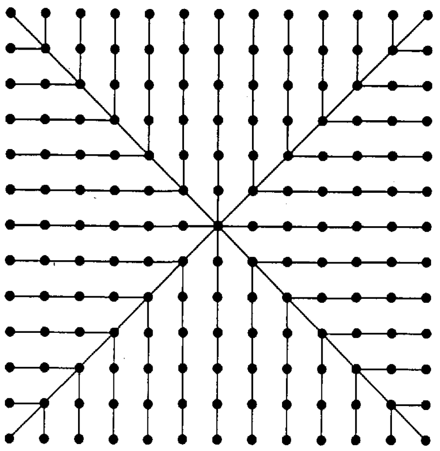

Fig. 4. A shortest-path spanning tree for a cluster of the cover $\mathcal{T}$ on the grid G_{19}.

Finally, the second claim of the lemma follows from Property (3) of Lemma 2.1, noting that $|\mathcal{S}_i| = n$.

Example (continued): The grid G_{19} described earlier (Fig. 2) has diameter 18. Consequently, the hierarchy requires five levels. Fix the parameter $k = 3$. Then in fact, all four highest levels of the hierarchy coincide, since the cover produced by our construction for all of these levels is identical, and consists of a single cluster covering the entire graph. Hence, any shortest-path spanning tree for the graph is appropriate as the single tree in $\mathcal{F}_3^i$ for $i = 2, 3, 4, 5$. Now consider the first level ($i = 1$). In this level, the cover selected for $\mathcal{N}_1(V)$ may be $\mathcal{T}$ of the above example, and the tree collection $\mathcal{F}_3^1$ consists of four shortest-path trees, each spanning one of the four clusters depicted in Fig. 3. A possible tree for such a cluster is given in Fig. 4

The construction of the above lemma is now used in order to set up our routing scheme, as follows. Given a parameter k, we first construct the tree cover $\mathcal{F}_k$ as in Lemma 3.1. At each node v we allocate two buffers for every tree F passing through the node. These buffers are called the *inbound* and *outbound* buffers of F, and are denoted by $B_I(v, F)$ and $B_O(v, F)$, respectively. The first buffer is dedicated to message traffic towards the root of F, and the other to message traffic going away from the root.

For every two nodes u, v in the network, we select the route connecting them in the tree F mutual to both of them, whose existence is asserted in Lemma 3.1. (In case there are several such trees, choose the one yielding the shortest route.) This route is composed of two segments: an inbound segment going from u towards the root of F, until hitting $l = l(u, v)$, the lowest common ancestor of u and v, and an outbound segment going from l towards v. A message traveling from u to v on this route uses the inbound buffers $B_I(w, F)$ at the nodes w it passes in the inbound segment. At l, it switches from the inbound buffer $B_I(l, F)$ to the outbound buffers $B_O(l, F)$. Finally, in the outbound segment, the message uses

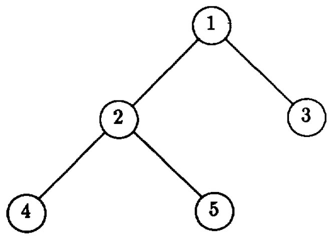

Fig. 5. The tree F.

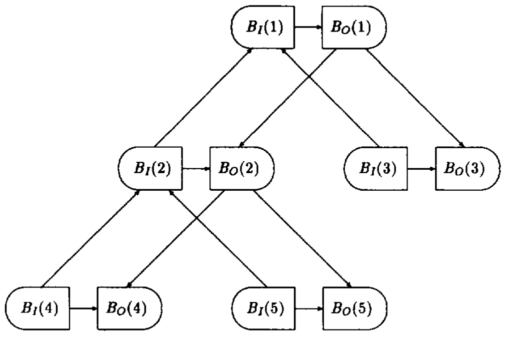

Fig. 6. The associated component $\mathcal{B}_F$ of the buffer graph $\mathcal{B}$.

the outbound buffers $B_O(w, F)$. Collisions between messages occupying the same buffers are resolved by maintaining queues and delivering messages in a FIFO order.

Lemma 3.2 : The resulting routing scheme is starvation-free, i.e., every message eventually arrives its destination.

Proof: Consider the *buffer graph* $\mathcal{B} = (V_\mathcal{B}, E_\mathcal{B})$ induced by the construction. This is a directed graph that is defined as follows. Its nodes are the buffers in the network,

$$V_\mathcal{B} = \{B_I(w, F), B_O(w, F) \mid w \in V, F \in \mathcal{F}_k\}.$$

The edge set $E_\mathcal{B}$ contains a directed edge from B to B' iff there is a route in which a message is moved from B to B' in one of the steps along the route.

Note that the buffer graph $\mathcal{B}$ is partitioned into connected components induced by the collection of trees. Namely, for each tree F in the collection $\mathcal{F}_k$, the buffers

$$\{B_I(w, F), B_O(w, F) \mid w \in F\}$$

form a connected component $\mathcal{B}_F$ of $\mathcal{B}$, with no directed edges between any two components $\mathcal{B}_F$ and $\mathcal{B}_{F'}$. Moreover, by definition of the routing scheme, the component $\mathcal{B}_F$ is acyclic. Fig. 5 and 6 depict a tree F and the associated component $\mathcal{B}_F$ of the buffer graph, respectively.

It follows that the entire buffer graph $\mathcal{B}$ is acyclic. This implies that the scheme is deadlock-free. The FIFO policy for handling arriving messages guarantees progress, which completes the proof of the lemma.

Lemma 3.3 : The resulting routing scheme requires $2k \cdot \lceil n^{1/k} \cdot \log D(G) \rceil$ buffers per vertex and the routes are at most $8k$ longer than optimal.

Proof: Follows from the construction and Lemma 3.1.

Let us finally make a brief comment on the issue of handling faults. Our solution, as described, assumes a static network. In most existing networks, link failures are detected via time-outs of low-level data link protocols. This allows a "semi-static" approach to failures. Whenever the network topology changes, this fact is detected by the involved nodes, who then trigger the initiation of an adaptation stage, replacing the present (static) routing scheme with a new scheme consists with the current topology. In this sense, the situation is analogous to that of most common algorithms for routing and other topology sensitive tasks (cf. [9]), where the topological databases and delay estimates have to be recomputed periodically in an adaptive fashion.

The covers required for setting up the tree collection $\mathcal{F}_k$ can be constructed using the distributed clustering techniques of [1], in communication cost $O(E \cdot \log^4 n)$. Since any topological change can completely change the distance metric, there is apparently no way to avoid this cost in the worst case. Further research is needed in order to develop efficient cluster updating methods, attempting to take advantage of the probable resemblance between the previous and current topology.

References

[1] B. Awerbuch and D. Peleg, "Efficient distributed construction of sparse covers," Weizmann Institute, Tech. Rep. CS90-17, July 1990.
[2] ——, "Sparse partitions," in *Proc. 31st IEEE Symp. Foundations Comput. Sci.*, 1990, pp. 503–513.
[3] ——, "Routing with polynomial communication—Space tradeoff," *SIAM J. Discrete Math.*, vol. 5, pp. 151–162, 1992.
[4] B. J. Brachman and S. T. Chanson, "A hierarchical solution for application level store-and-forward deadlock prevention," in *SIGCOM89*, ACM, Sept. 1989, pp. 25–32.
[5] B. Gavish, P. Merlin, and A. Segall, "Minimal buffer requirements for avoiding store-and-forward deadlock," IBM Yorktown, Res. Rep. RC-6672, Aug. 1977.
[6] I. S. Gopal, "Prevention of store-and-forward deadlock in computer networks," IBM Yorktown, Res. Rep. RC-10677, Aug. 1984.
[7] K. D. Gunther, "Prevention of deadlocks in packet-switched data transport systems," *IEEE Trans. Commun., Special Issue on Control in Computer Networks*, pp. 512–524, June 1981.
[8] N. Linial and M. Saks, "Decomposing graphs into regions of small diameter," in *Proc. 2nd ACM-SIAM Symp. Discrete Algorithms*, ACM/SIAM, Jan. 1991, pp. 320–330.
[9] J. McQuillan, I. Richer, and E. Rosen, "The new routing algorithm for the ARPANET," *IEEE Trans. Commun.*, vol. COM-28, pp. 711–719, May 1980.
[10] P. M. Merlin and P. J. Schweitzer, "Deadlock avoidance in store-and-forward networks—I: Store and forward deadlock," *IEEE Trans. Commun.*, vol. COM-28, pp. 345–352, Mar. 1980.
[11] D. Peleg, "Sparse graph partitions," The Weizmann Institute, Tech. Rep. CS89-01, Feb. 1989.
[12] ——, "Distance-dependent distributed directories," *Inform. and Computation*, vol. 103, pp. 270–298, 1993; also in The Weizmann Institute, Tech. Rep. CS89-10, May 1989.
[13] S. Toueg and J. D. Ullman, "Deadlock-free packet switching networks," *SIAM J. Comput.*, vol. 10, pp. 594–611, Aug. 1981.