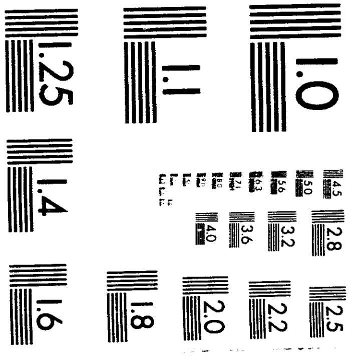
1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3

1 of 1

SAND93-2704C
Conf-940563--1

An $\Omega(\sqrt{\log\log n})$ Lower Bound for Routing in Optical Networks

Leslie Ann Goldberg, Sandia National Labs*

Mark Jerrum, University of Edinburgh†

Philip D. MacKenzie, University of Texas‡

6th November 1993

ABSTRACT Optical communication is likely to significantly speed up parallel computation because the vast bandwidth of the optical medium can be divided to produce communication networks of very high degree. However, the problem of contention in high-degree networks makes the routing problem in these networks theoretically (and practically) difficult. In this paper we examine Valiant's *h-relation routing problem*, which is a fundamental problem in the theory of parallel computing. The h-relation routing problem arises both in the direct implementation of specific parallel algorithms on distributed-memory machines and in the general simulation of shared memory models such as the PRAM on distributed-memory machines. In an h-relation routing problem each processor has up to h messages that it wishes to send to other processors and each processor is the destination of at most h messages. We present a lower bound for routing an h-relation (for any $h > 1$) on a complete optical network of size n. Our lower bound applies to any randomized distributed algorithm for this task. Specifically, we show that the expected number of communication steps required to route an arbitrary h-relation is $\Omega(h + \sqrt{\log\log n})$. This is the first known lower bound for this problem which does not restrict the class of algorithms under consideration.

* Sandia National Labs, PO Box 5800, Albuquerque, NM 87185-1110 USA. E-mail: lagoldb@cs.sandia.gov. This work was performed at Sandia National Laboratories and was supported by the U.S. Department of Energy under contract DE-AC04-~~76DP00789~~. 94AL85000

† Department of Computer Science, The University of Edinburgh, The King's Buildings, Edinburgh EH9 3JZ, United Kingdom, E-mail: mrj@dcs.ed.ac.uk. This work was supported in part by grant GR/F 90363 of the UK Science and Engineering Research Council, and Esprit Working Group "RAND;" and was partly done while the author was visiting the NEC Research Institute, Princeton NJ, USA.

‡ Department of Computer Sciences, University of Texas, Austin, TX 78712-1188, USA. E-mail: philmac@cs.utexas.edu. This work was supported by Texas Advanced Research Projects Grant 003658480.

1. Introduction

In current distributed-memory parallel computers, a number of processors equipped with private local memory communicate by sending messages via a network of communication links. Current technology restricts the network to be of low degree: each processor in the network can communicate directly with only a few others, and the remainder must be reached indirectly by routing messages along a sequence of links. The emerging technology of optical communication challenges the assumption that the network must be of low degree. In particular, the huge bandwidth of the optical medium can be divided so that each processor has its own channel for receiving messages and each processor can send on any channel. Even though such an interconnection network is a complete graph, there remains the problem of contention: no processor can receive messages simultaneously from two other processors without corruption. The problem of avoiding contention is much more difficult in high-degree networks (such as optical networks) than in traditional low-degree networks.

The problem of routing in optical networks is captured mathematically by Anderson and Miller's OCPC model. In an n-processor *completely connected Optical Communication Parallel Computer* (n-OCPC) n processors with local memory are connected by a complete network. A computation on this computer consists of a sequence of communication steps. During each communication step each processor can perform some local computation and then send one message to any other processor. If a processor is sent a single message during a communication step then it receives this message successfully, but if it is sent more than one message then the transmissions are garbled and it receives none of them.

The OCPC model was first introduced by Anderson and Miller [2], and has subsequently been studied by several authors including Valiant [17], Eshaghian [7], Geréb-Graus and Tsantilas [11], and Gerbessiotis and Valiant [10] (though not always under the name OCPC). Aside from its importance as a model for optical communication, the OCPC has the attraction of being a clean, mathematically appealing model that allows us to study a single issue, namely the resolution of contention between independent processors, in isolation from other factors. It has recently been observed that the n-processor OCPC is equivalent to an ERCW PRAM with n global memory cells. Thus our results carry over to this model. For details, see [15].

In this paper we study a fundamental communication problem for multiprocessor computers: that of routing h-relations. This problem arises both in the direct implementation of specific parallel algorithms [2], and in the simulation of shared-memory models, such as the PRAM, on more realistic distributed-memory models [17]. An *h-relation* [17] is a communication problem in which each processor has up to h messages that it wishes to send to other processors. The destinations of these messages can be arbitrary except that each processor is the destination of at most h messages. The goal is to design a fast n-OCPC algorithm that can route an arbitrary h-relation.

Anderson and Miller [2] have observed that an h-relation can easily be routed in h communication steps if all of the processors are given *total* information about the h-relation to be routed. A more interesting (and more realistic) situation arises if we assume that each processor initially knows only about the messages that it wants to send, and that processors learn about the the rest of the h-relation only through receiving messages from other processors. This is the usual assumption, and the one that will be made here.

Valiant [17], building on work of Anderson and Miller [2], developed a randomized algorithm that routes an arbitrary h-relation in $O(h + \log n)$ steps, on average. Subsequently, Goldberg, Jerrum, Leighton, and Rao [12] presented a more complex randomized algorithm for the same task that runs in $O(h + \log\log n)$ steps and has failure probability $n^{-\alpha}$ for any constant α. The latter algorithm is asymptotically the fastest known, and it would be interesting to discover whether it is the best possible. Our attention therefore turns to lower bounds.

Goldberg et al. [12] proved a lower bound for a restricted class of algorithms known as *direct*, in which a processor may only send messages directly to their final destination. (Thus the only freedom a processor has is in its choice of *when* to attempt to send its messages.) They proved that for any (randomized) direct algorithm there is a 2-relation that takes $\Omega(\log n)$ steps to route with success probability $\frac{1}{2}$, thus showing that even in a completely connected network it is advantageous to route messages indirectly.

Obtaining a lower bound for *unrestricted* algorithms has proved a much greater challenge, owing, no doubt, to the rich variety of strategies that are available to a non-direct algorithm. (Some of the possibilities will be glimpsed in Section 2.) Indeed, no lower bound beyond the trivial $\Omega(h)$ was previously known. The new result in this paper is a lower bound on the number of steps required to route 2-relations on an n-OCPC. We prove that for any randomized algorithm there is a 2-relation such that the expected number of steps required to route the relation is

$\Omega(\sqrt{\log\log n})$. (See Theorem 1 for a precise statement of the result.) Our result implies that for any $h > 1$ the number of steps required to route an arbitrary h-relation is $\Omega(h + \sqrt{\log\log n})$. We note that our lower bound also holds for routing $c+1$-relations in the c-collision OCPC model studied by Dietzfelbinger and Meyer auf der Heide [4].

The proof presented here has elements in common with that of Beame and Hastad's lower bound for computing parity on a CRCW PRAM [3]. However, our proof technique (which is based on one used by MacKenzie in [14]) is stronger, in that it applies to random inputs rather than worst-case inputs. (As a side note, we mention that by modifying Beame and Hastad's proof in accordance with our proof technique, it is possible to show that their lower bound holds for probabilistic CRCW PRAMs. It was known that their lower bound could be extended to the probabilistic case, but the extension was *indirect* and rested on simulating randomized algorithms by deterministic ones, using a technique of Ajtai and Ben-Or [1]. Our technique, however, provides the first known *direct* proof of a lower bound for parity on probabilistic CRCW PRAMs.)

The gap between the current upper and lower bounds on routing h-relations deserves comment. Section 4 indicates why a lower bound of the form $\Omega(\sqrt{\log\log n})$ is the limit of the current technique. An example presented in that section points to an issue that must be faced in any attempt to improve the current lower bound. It appears that some new idea is necessary to make further progress on this front.

2. Some Preliminary Observations

Imagine that two processors p and q wish to deliver a single message each to a common destination processor within $O(\log\log n)$ steps. Assume that p and q do not know each other's identity. A simple strategy is for p and q each to flip a coin and attempt to transmit its packet to the destination processor if the coin comes up "heads." After $O(\log\log n)$ steps, the probability that p and q have failed to transmit their packets is at least $(\log n)^{-O(1)}$. If $n^{\Omega(1)}$ pairs of processors simultaneously employ this strategy to deliver their messages to separate destinations, the probability that they all succeed is negligible. Some more subtle approach is required.

One possibility, suggested by Rao, is the following. Suppose the processors are assigned binary sequence numbers, and that the numbers assigned to p and q are $p_1p_2\ldots p_r$ and $q_1q_2\ldots q_r$, where $r \sim \log n$. By simultaneously sending messages to processors $p_1p_2\ldots p_{r/2}0\ldots0$ and $q_1q_2\ldots q_{r/2}0\ldots0$, respectively, processors p

and q may discover whether their sequence numbers differ in the first $r/2$ bits. After about $\log\log n$ experiments of this general form, and using binary search, p and q can agree on a bit position at which their sequence numbers differ; this bit can then be used to determine a priority for the processors, and hence resolve the conflict. Note that this method (with slight modification) could be used by $n^{\Omega(1)}$ pairs of processors simultaneously. Observe that p and q are not sending messages in order to get the content of the message to another processor, but to learn some information about the competing processor.

A second strategy is replication of messages. In $O(\log\log n)$ binary replication steps, p and q can each prime a set of $\Theta(\log n)$ processors with the message they are required to transmit. These two sets of processors then use the naive coin-flipping strategy to attempt to send their cloned messages to a common target set of size $\Theta(\log n)$. In just a constant number of attempts, the probability that either a p-message or a q-message fails to get through is reduced to $n^{-\Omega(1)}$ where the implicit constant is arbitrary. Finally, the messages in the target set can be funneled into the destination processor by a procedure which is an inverse to the cloning phase. Note that the failure probability is much smaller here than for the naive strategy, and can be expected to remain small when many pairs of processors simultaneously attempt to send to distinct targets.

These two examples indicate the subtle strategies that are available to indirect algorithms. With these in mind, it is possible to give a little of the flavor of the lower bound argument. After t-steps, some set of processors (of size at most exponential in t) will be aware that processor p or q has a packet to send. Viewing the situation crudely, these "agents" for p and q can act in one of two modes, or possibly a mixture: (a) they can send messages to some narrow set of destinations that is only weakly dependent on the identity of the source processor, or (b) they can send to a wide destination set, or one that is strongly dependent on the identity of the source processor.

The first strategy sketched above operates purely in mode (a), while the second strategy relies on mode (b) to recruit the processors that are required in the replication phase. The key point is that the effectiveness of mode (a) is limited by the collisions that inevitably occur, while mode (b) is limited in its ability to "advance messages towards their destination." The lower bound proof to be described in Section 3 analyses the tradeoff between these modes. That both strategies described above are effective suggests that the whole range of the tradeoff must be examined, and explains some of the technical complexity of the proof.

3. The Lower Bound Argument

Our goal is to establish the following.

Theorem 1. *Let A be a randomized algorithm that routes 2-relations on an n-OCPC. Then there is a 2-relation on which the expected number of communication steps used by A is at least $\sqrt{\log\log n}/4$.*

Due to space constraints, we will defer the full proof of this result to the appendix, and simply present some highlights of the proof in this extended abstract.

The first step is to reduce to the case of deterministic A. A certain restricted class of 2-relations (to be defined presently) will be termed "relevant." According to a theorem of Yao (see [9]), proving Theorem 1 reduces to proving the following.

Theorem 2. *Let A be a deterministic algorithm that allegedly routes 2-relations in $T = \sqrt{\log\log n/2}$ steps. Let the input to A be drawn u.a.r. from the set of relevant 2-relations. Then the probability that A successfully routes the input is at most $\frac{1}{2}$.*

We partition the n-OCPC into $n^{4/5}$ *ranges* containing $n^{1/5}$ processors each. For the purposes of the proof, a ***partial** h**-relation*** is a function from the set $\{1,\ldots,n\}$ of processors to $\{0,1,*\}$. A partial h-relation is called an *h-relation* iff no processor is mapped to '$*$'. Intuitively, a '1' indicates that the processor has a message to send and a '0' indicates that it does not. In all cases the messages will be destined for the first processor in the range containing the sending processor. Let f_* denote the partial h-relation which maps every processor to '$*$'. A partial h-relation f is called a *refinement* of a partial h-relation f' if $f'(p) = 1$ implies $f(p) = 1$, and $f'(p) = 0$ implies $f(p) = 0$. (We denote this by $f \leq f'$.) We will say that an h-relation is a ***relevant 2-relation*** if it has exactly two processors in each range mapped to '1'. We will say that a partial h-relation is a ***partial relevant 2-relation*** if it has a refinement which is a relevant 2-relation.

Let A be any deterministic algorithm for an n-OCPC and let f be a partial 2-relation. For the purposes of this proof sketch, we informally define the (t,f)-*knowledge set* of a processor p as the subset of processors mapped to '$*$' by f which could affect the state of p within the first t steps of A when it is run on an input 2-relation g that refines f.

Now define the following constants and functions of n:

$$k_i = 3^i, \quad s_0 = n^{1/5}, \quad w_i = s_i^{1/k_i}/21k_i^2, \quad r_i = s_i^4, \text{ and } s_i = w_{i-1}^{1/7} \text{ (for } i \geq 1\text{)}.$$

A partial h-relation f is called t-*good* if the following three conditions are satisfied.

1. r_t ranges have s_t processors that are mapped to '$*$' by f, and no processors that are mapped to '1' by f, while the remaining ranges have no processors mapped to '$*$' by f, and two processors that are mapped to '1' by f.
2. The (t, f)-knowledge set of each processor p has size at most one.
3. Each processor q is in the (t, f)-knowledge set of at most k_t processors.

Condition (2) captures a crucial idea, which can be traced to Fich et al. [8], and may be expressed informally as follows. Suppose that A is run on input g, where g is a 2-relation that refines f. Then the entire state of the n-OCPC at time t depends in a particularly simple way on the restriction of g to the processors p with $f(p) =$ '$*$'.

At the heart of the proof is a randomized procedure CONSTRUCT(t, f) that takes a time t and a partial 2-relation f and returns a new partial 2-relation f' that is a refinement of f. Aside from the parameters t and f, CONSTRUCT depends implicitly on the algorithm A, in particular on the action of A at time step $t + 1$. (The approach here is similar to that used by MacKenzie in the context of lower bounds for load balancing [14].) The procedure CONSTRUCT has two important properties, the first of which is concerned with invariance. Let $T = \sqrt{\log\log n/2}$.

Lemma 3. *If $t < T$ and* CONSTRUCT *is called with parameters (t, f), where f is t-good, then with probability at least $1 - n^{-2}$* CONSTRUCT *will return a partial relation f' that is $(t + 1)$-good.*

The second property is that CONSTRUCT is unbiased. Specifically, suppose that GENERATE is a procedure that starts with the relation $f_0 = f_*$, and applies CONSTRUCT T times to generate a sequence of partial relevant 2-relations $f_0 = f_* \geq f_1 \geq \cdots \geq f_T \geq f$ in which each $f_t =$ CONSTRUCT(t, f_{t-1}) is a refinement of f_{t-1}, and f is a relevant 2-relation generated u.a.r. from the set of refinements of f_T.

Lemma 4. *The relevant 2-relation f produced by* GENERATE *is uniformly distributed.*

Note that from Lemma 3 we also have.

Lemma 5. *With probability at least $1 - n^{-1}$, the partial relation f_T is T-good.*

It is not possible to give a complete description here of the procedure CONSTRUCT, still less a proof of correctness. However, some of the main ideas may be sketched. Suppose that CONSTRUCT is called with parameters (t, f) where the partial 2-relation f is t-good. Let the jth range be denoted R_j and let S_j denote the set of processors in R_j that are mapped to '$*$' by f. Let J be the set of indices j such that $|S_j| > 0$. We say that a processor p *zero-affects* a processor q if there is a processor p' such that p is in the (t, f)-knowledge set of p', and for any relevant 2-relation g which refines f and has $g(p) = 0$: when A is run with input g, processor p' sends to q on step $t+1$. The notion of p *one-affecting* processor q is defined analogously. Whenever it is the case that a processor p is zero-affected or one-affected by a processor q there is a risk that the $(t+1, f)$-knowledge set of p will grow to size greater than one. Recall that the aim of CONSTRUCT is to produce a refinement f' of f that is $(t+1)$-good; in particular this entails arranging that the $(t+1, f')$-knowledge set of p has at most one. CONSTRUCT's strategy is to nominate, for each range R_j with $j \in J$, a certain subset S'_j of S_j, and then randomly select a refinement f' of f such that undetermined part of f' lies precisely over the union of the S'_j. Observe that there is a (unique) probability distribution on refinements that is consistent with GENERATE being unbiased.

To form S'_j, CONSTRUCT starts with $S'_j = S_j$ and removes processors from S'_j in four stages. To a certain degree of approximation, these are as follows.

1. For each $j \in J$, shrink S'_j so that for every processor p, either (i) p is zero-affected by at most one processor in S'_j, or (ii) p is the site of a collision of messages at step $t+1$.
2. For each $j \in J$, further shrink S'_j so that for every processor p either (i) p is one-affected by at most one processor in S'_j, or (ii) p is the site of a collision of messages at step $t+1$. This stage uses a theorem of Erdős and Rado [6] concerning the existence of large "sunflowers" in a set system. A sunflower is a collection of sets such that if an element is in two of the sets then it is in all of them. The set system in question comprises sets of the form "all processors that are one-affected by some processor p," where p ranges over S'_j. The "corolla" of the sunflower guaranteed by Erdős-Rado contains the collision sites, while the "petals" contain processors that are one-affected by a single processor. A similar construction was used by Grolmusz and Ragde [13].
3. At the start of this stage, a processor p may be zero-affected by one processor in S'_j, one-affected by another, and have a third in its (t, f)-knowledge set. Now we further shrink S'_j so that at most one of these possibilities occur. This is accomplished by constructing a dependency graph on processors, and choosing a large independent set as guaranteed by Turán's theorem. A similar

construction was used by Fich, Meyer auf der Heide, and Wigderson [8].

4. At this point each processor p may still be affected by several processors, though at most one of these will come from any given S'_j. In the final stage, the ranges $\{R_j : j \in J\}$ are divided into groups; these groups are considered in turn, and the sets S'_j shrunk further, so that no processor is affected by more than one processor from any of the groups processed so far. Also at this stage, the restriction of f to the group being processed is refined — by making random assignments to the various sets $S_j - S'_j$ within the group — to yield the restriction of f' to that group. The satisfactory processing of a group depends on decisions made in earlier groups; however, it turns out that the probability that CONSTRUCT fails while processing a particular group, conditioned on the choices made in all previous groups, is very small. A probabilistic lemma assures us that with high probability all groups will be processed successfully.

At the end of these four stages, the $(t+1, f')$-knowledge sets of the processors are all of size at most one. The other claims about f' implicit in Lemma 3 come from the bounds provided by the Erdős-Rado and Turán theorems, and from the probabilistic argument mentioned in connection with stage (4) of CONSTRUCT.

The proof of Theorem 2 follows quickly from Lemmas 4 and 5, provided we are prepared to set aside a minor technical complication, which is dealt with in the Appendix. With high probability, the partial 2-relation f_T produced by GENERATE has many ranges with no processors mapped to '1' by f_T. In these ranges the target processor has a (T, f_T)-knowledge set of size at most one; thus the target processor can have received at most one of the messages destined for it.

4. The prospect for tightening the bound

Recall the situation in which two processors p and q each have a single message to transmit to a common destination. Consider the following OCPC "algorithm" which is a parallel version of a strategy consider in Section 2. In $\Theta(\sqrt{\log\log n})$ steps, p and q recruit $k = \Theta(\exp(\sqrt{\log\log n}))$ "agents" to help discover a bit position at which the binary sequence numbers for p and q differ. This is done using the method of Section 2, but with k-way search in place of binary search: a p-agent and a q-agent simultaneously attempt to transmit a message to processors with sequence numbers of the form $0\ldots0p_{i+1}\ldots p_{i+r/k}0\ldots0$ and $0\ldots0q_{i+1}\ldots q_{i+r/k}0\ldots0$, respectively, and hence discover whether the sequence numbers of p and q differ on a particular block of r/k bits. This would seem to give a $O(\sqrt{\log\log n})$ algorithm for delivering the messages.

Of course, the catch is that a p-agent that finds a block on which the sequence numbers of p and q differ is unable to alert the other p-agents to the discovery, at least, not sufficiently quickly to obtain an improvement over the original binary search strategy. Unfortunately, the lower bound argument presented here is oblivious to a cheating "algorithm" in which an agent that finds an appropriate block broadcasts its discovery to the other agents in one step. The problem is that in the lower bound argument, the behavior of a processor is considered to be a function of a partial 2-relation f that provides far more information than a processor could in reality know.

Acknowledgments

The authors thank Satish Rao and Mike Sipser for useful discussions.

References

[1] MIKLOS AJTAI and MICHAEL BEN-OR, A theorem on probabilistic constant depth computations, *Proceedings of the ACM Symposium On Theory of Computing* **16** (1984) 471–474.

[2] RICHARD J. ANDERSON and GARY L. MILLER, Optical Communication for Pointer Based Algorithms, Technical Report CRI 88-14, Computer Science Department, University of Southern California, Los Angeles, CA 90089-0782 USA, 1988.

[3] PAUL BEAME and JOHAN HASTAD, Optimal bounds for decision problems on the CRCW PRAM, *Journal of the ACM* **36(3)** (1989) 643–670.

[4] MARTIN DIETZFELBINGER and FRIEDHELM MEYER AUF DER HEIDE, Simple Efficient Shared Memory Simulations, *Proceedings of the ACM Symposium On Parallel Algorithms and Architectures* **5** (1993) 110–119.

[5] PATRICK W. DOWD, High performance interprocessor communication through optical wavelength division multiple access channels, *Symp. on Computer Architectures* **18** (1991) 96–105.

[6] P. ERDŐS and R. RADO, Intersection theorems for systems of sets, *Journal of the London Mathematical Society* **35** (1960) 85–90.

[7] Mary Mehrnoosh Eshaghian, Parallel Algorithms for Image Processing on OMC, *IEEE Transactions on Computers* 40(7) (1991) 827–833.

[8] Faith E. Fich, Friedhelm Meyer auf der Heide and Avi Wigderson, Lower bounds for parallel random-access machines with unbounded shared memory, *Advances in Computing Research* 4 (F. Preparata, ed.), JAI Press 1987, 1–15.

[9] Faith E. Fich, Prabhakar Ragde and Avi Wigderson, Relations between concurrent-write models of parallel computation, *SIAM Journal of Computing* **17**(3) (1988) 606–627.

[10] Alexandros V. Gerbessiotis and Leslie G. Valiant, Direct bulk-synchronous parallel algorithms, *Scandinavian Workshop on Algorithm Theory* **3**, (Springer-Verlag, 1992).

[11] Mihály Geréb-Graus and Thanasis Tsantilas, Efficient optical communication in parallel computers, *Proceedings of the ACM Symposium On Parallel Algorithms and Architectures* **4** (1992) 41–48.

[12] Leslie Ann Goldberg, Mark Jerrum, Tom Leighton and Satish Rao, A Doubly Logarithmic Communication Algorithm for the Completely Connected Optical Communication Parallel Computer, *Proceedings of the ACM Symposium On Parallel Algorithms and Architectures* **5** (1993) 300–309.

[13] Vince Grolmusz and Prabhakar Ragde, Incomparability in parallel computation, *Proceedings of the IEEE Symposium on Foundations of Computer Science* **28** (1987) 89–98.

[14] Philip D. MacKenzie, Load balancing requires $\Omega(\log^* n)$ expected time, *Proceedings of the ACM-SIAM Symposium On Discrete Algorithms* **3** (1992) 94–99.

[15] Philip D. MacKenzie and Vijaya Ramachandran, Optical Communication and ERCW PRAMs, submitted to *Proceedings of the ACM Symposium On Parallel Algorithms and Architectures* **6** (1994).

[16] Satish B. Rao, *Properties of an interconnection architecture based on wavelength division multiplexing*, Technical Report TR-92-009-3-0054-2, NEC Research Institute, 4 Independence Way, Princeton, NJ 08540 USA, 1992.

[17] L.G. Valiant, General purpose parallel architectures, Chapter 18 of *Handbook of Theoretical Computer Science*, (J. van Leeuwen ed.) (Elsevier 1990) (See especially p. 967)

Appendix - Proof of the Lower Bound

An $\Omega(\sqrt{\log\log n})$ Lower Bound for Routing in Optical Networks

Leslie Ann Goldberg, Mark Jerrum, and Philip D. MacKenzie

First we state the following facts about the functions which were defined in the abstract.

Fact 6. *For* $t \leq T$, $k_t \leq 3^{\sqrt{\log\log n}}$

Fact 7. *For large enough* n *and* $t \leq T$, $s_t \geq 2^{\log^{1/3} n}$.

Fact 8. *For large enough* n *and* $t < T$, $3k_t \leq w_t^{1/7}$.

Fact 9. $r_t/w_t^{4/7} > s_t^3$.

1. Generating a random 2-relation

Algorithm RANDOMSET will be used to randomly generate a relevant 2-relation one processor at a time. It is called with a partial relevant 2-relation f and a set P of processors which are mapped to '$*$' by f. The processors in P are randomly mapped to '0' or '1' in such a way that the resulting function f' is a partial relevant 2-relation and Claim 10 holds.

```
Function RANDOMSET($f, P$)
    Let $f' := f$
    For each $p \in P$
        Let $s = |\{ q \mid q$ is in the range of $p$ and $f(q) =$ '$*$'$\}|$
        If no processors in the same range as p are mapped to '1' by $f$
            With probability $2/s$ set $f'(p) = 1$
            With probability $1 - 2/s$ set $f'(p) = 0$
        If one processor in the same range as p is mapped to '1' by $f$
            With probability $1/s$ set $f'(p) = 1$
            With probability $1 - 1/s$ set $f'(p) = 0$
        Otherwise set $f(p) = 0$
    Return $f'$
End RANDOMSET
```

Claim 10. *An* h*-relation* f *generated solely by calls to RANDOMSET is a relevant 2-relation generated uniformly at random (u.a.r.) from the set of relevant 2-relations.*

Proof: Straightforward. □

2. Defining the knowledge set and t-good partial h-relations

Now we make some definitions that deal with the running of a deterministic algorithm on an n-OCPC.

Let A be any deterministic algorithm for an n-OCPC and let f be an h-relation. The $(0, f)$-trace of processor p is defined to be the tuple $< p, f(p) >$. The (t, f)-trace of processor p (for $t \geq 0$) is defined to be the tuple $< p, f(p), \lambda_1, \ldots, \lambda_t >$ in which λ_j is the message that processor p receives at step j if such a message exists and λ_j is the null symbol otherwise.

Note that we lose no generality by assuming that if p sends a message on step t then it sends its entire $(t-1)$-trace. (Since each processor is allowed to know the algorithms that the other processors run we can simulate an algorithm which sends different messages by an algorithm which sends traces using the same pattern of communications.)

We will say that processor p is a direct (t, f)-receiver of processor q if either $p = q$ or when A is run with input f, p receives a message from q in the first t steps. We will say that p is an indirect (t, f)-receiver of q if either p is a direct (t, f)-receiver of q, or when A is run with input f, there is some processor k and some time-step $t' < t$ such that k is an indirect (t', f)-receiver of q and p receives a message from k during steps $t' + 1, \ldots, t$.

Let g be any partial h-relation. We will say that a set S of processors is a (t, g)-dependency set of a processor p if it is the case that for any relevant 2-relations f_1 and f_2 which refine g and have $f_1(q) = f_2(q)$ for every processor $q \in S$, the (t, f_1)-trace of p is the same as the (t, f_2)-trace of p. (Intuitively, p is not dependent on processors outside S, since these could not affect its trace.) Note that if S' and S'' are (t, g)-dependency sets of a processor p then so is $S' \cap S''$, so p has a unique (t, g)-dependency set of minimum size, which we will call p's (t, g)-knowledge set.

Suppose that g is a partial h-relation and that f is a relevant 2-relation which refines g. Note that if $g(p) =$ '$*$' and q has a (t, g)-dependency set which excludes p then q cannot be an indirect (t, f)-receiver of p. Also note that if $g(p) \neq$ '$*$' then p is not in the (t, g)-knowledge set of any processor.

We make use of the definition of t-*good* given in the abstract.

3. Refining partial 2-relations with CONSTRUCT

Below we give an algorithm CONSTRUCT which is called with a time t and a partial 2-relation f, and which randomly refines f based on the action of algorithm A at step $t+1$.

```
Function CONSTRUCT$(t, f)$
    For each $i \in \{1, \ldots, \ell\}$
        Let $V_i = \emptyset$
        For each $j \in J_i$
            Let $S = \emptyset$
            Let $S' = \emptyset$
            While $|S| < w_t^{1/7}$ and $|W_j - S - S'| > 0$
                Let $p$ be the lowest numbered processor in $W_j - S - S'$
                If there is no $p' \in V_1 \cup \cdots \cup V_{i-1}$ such that
                    AFFECTS$(p) \cap$ AFFECTS$(p') \neq \emptyset$ Then
                    Let $S = S \cup \{p\}$
                Else
                    Let $S' = S' \cup \{p\}$
            Let $f$ = RANDOMSET$(S_j - S, f)$
            If $f$ maps any processor in $S_j - S$ to '1' Then
                Let $f$ = RANDOMSET$(S, f)$
                Next $j$
            Else
                Let $V_i = S$
                For each remaining $j' \in J_i$
                    Let $f$ = RANDOMSET$(S_{j'}, f)$
                Next $i$
    Let $f' = f$
    Return $f'$
End CONSTRUCT
```

To explain algorithm CONSTRUCT, we use the definitions given in the extended abstract and add the following definitions.

Let W_j' be a subset of S_j which is as large as possible and has the property that if two processors p_1 and p_2 are in W_j' and zero-affect the same processor q, then two processors in $S_j - W_j'$ also zero-affect processor q. Let W_j'' be a subset of W_j' which is as large as possible and has the property that if two processors p_1 and p_2 are in W_j'' and one-affect the same processor q, then all processors in W_j'' one-affect processor q.

For each processor p in range R_j we define the set AFFECTS(p) as follows.

1. If p is in the (t, f)-knowledge set of any processor q then put q in AFFECTS(p).
2. If p zero-affects any processor q and there are not two processors in $S_j - W_j'$ which zero-affect q then put q in AFFECTS(p).

   (The intuition here is that if there are two processors in $S_j - W_j'$ which zero-affect q and all of the processors in $S_j - W_j$ are mapped to '0' there will be a collision at processor q at step $t + 1$ so q will not be affected by p.)
3. If p one-affects any processor q and there is some processor in W_j'' which does not one-affect q then put q in AFFECTS(p).

   (The intuition here is that if every processor in W_j'' one-affects q and all of the processors in $S_j - W_j''$ are mapped to '0' there will be a collision at processor q at step $t + 1$ so q will not be affected by p.)

Let W_j be a subset of W_j'' which is as large as possible and has the property that for any two processors p_1 and p_2 in W_j, AFFECTS(p_1) $\cap$ AFFECTS(p_2) is empty. (Intuitively, at this point, we would like each processor to be affected by at most one processor in each W_j)

In CONSTRUCT, we split J into groups $J_1, J_2, \ldots, J_\ell$ each of size $r_t/w_t^{4/7}$, with the last group possibly smaller. For each group J_i CONSTRUCT will construct a set V_i containing some of the processors from up to one of the ranges in J_i. The sets will have the property that if two processors p and p' are in $\bigcup_i V_i$, then AFFECTS(p) $\cap$ AFFECTS(p') is empty. Intuitively, this means that no processor could be affected by two processors in $\bigcup_i V_i$. We will let V denote $\bigcup_i V_i$. We will say that algorithm CONSTRUCT is *successful* if each set V_i has size $w_t^{1/7}$.

4. Analysis of CONSTRUCT

Claim 11. *If f is t-good then $|AFFECTS(p)| \leq 3k_t$ for each p.*

Proof: Since f is t-good, each p is in the (t, f) knowledge set of at most k_t processors. Each of these k_t processors can cause p to zero-affect at most one other processor and to one-affect at most one other processor. □

Claim 12. *If f is t-good then each processor q is in at most 3 sets AFFECTS(p) with $p \in W_j''$.*

Proof: Since f is t-good, the (t, f)-knowledge set of q has size at most one. Therefore, q is added to at most one set AFFECTS(p_1) using the first part of the definition of AFFECTS(p). By the construction of W'_j, q is added to at most one set AFFECTS(p_2) using the second part of the definition of AFFECTS(p). Finally, by the construction of W''_j, q is added to at most one set AFFECTS(p_3) using the third part of the definition of AFFECTS(p). □

Claim 13. *If f is t-good then for each $j \in J$ we have $|W'_j| \geq |S_j|/(2k_t + 1)$.*

Proof: We use the following procedure, which we call Procedure A:

```
Procedure A
    For each $j \in J$
        Let $S' = \emptyset$
        Let $S = S_j$
        While $|S| > 0$
        Select a processor $p \in S$
            Let $S = S - p$
            Let $S' = S' \cup \{p\}$
            For each processor $q$ which $p$ zero-affects
                Let $Z = \{v | v$ zero-affects $q$ and $v \in S\}$
                If $Z > 1$ Then
                    Let $p_1, p_2$ be two processors in $Z$
                    Let $S = S - \{p_1, p_2\}$
                Else
                    If $Z = 1$ Then
                        Let $p_1$ be the processor in $Z$
                        Let $S = S - \{p_1\}$
End A
```

Using procedure A we can construct a set $S' \subseteq S_j$ such that if two processors p_1 and p_2 are in S' and zero-affect the same processor q, then two processors in $S_j - S'$ also zero-affect processor q. Procedure A starts by setting $S = S_j$. Since f is t-good each processor $p \in S$ zero-affects at most k_t processors. So for each iteration of the while loop at most $2k_t + 1$ processors are removed from S with exactly one of them placed in S'. Thus $|S'| \geq |S_j|/(2k_t + 1)$. By the definition of W'_j, $|W'_j| \geq |S'| \geq |S_j|/(2k_t + 1)$. □

Claim 14. *If f is t-good then for each $j \in J$ we have $|W''_j| \geq |W'_j|^{1/(k_t)}/k_t$.*

Proof: For $p \in W'_j$, let $D(p)$ be the set of processors which p one-affects. Then $|D(p)| \leq k_t$. A *sunflower* is defined as a collection of sets such that if an element is in two of the sets, then it is contained in all of the sets. The Erdös-Rado Theorem says: Let t and m be positive integers and let F be a family of sets such that every element of F has size at most t and $|F| > t!(m-1)^t$. Then F contains a sunflower of size m. If we let F be the family of sets $D(p)$ for $p \in W_j'$, then F contains a sunflower of size $(|W_j'|/k_t!)^{1/k_t} \geq |W_j'|^{1/k_t}/k_t$. If two processors p_1 and p_2 correspond to two sets in this sunflower and they one-affect the same processor q, then (by the definition of D(p) and sunflower) all p corresponding to sets in this sunflower one-affect q, and since W_j'' is the largest set of processors which satisfy this property, $|W_j''| \geq |W_j'|^{1/k_t}/k_t$. □

Claim 15. *If f is t-good then for each $j \in J$ we have $|W_j| \geq |W_j''|/7k_t$.*

Proof: Construct a graph $G = (W_j'', E)$ where $(p,q) \in E$ if AFFECTS(p) ∩ AFFECTS(q) is non-empty. Then an independent set S in this graph has the property that for p_1, p_2 in S, AFFECTS(p_1) ∩ AFFECTS(p_2) is empty. Then W_j is simply the largest independent set in this graph. By Turán's Theorem, $|W_j| \geq |W_j''|^2/|W_j''| + 2|E|$. By Claim 11 and Claim 12, for each $p \in W_j''$, $|\text{AFFECTS}(p)| \leq 3k_t$, and each q is in at most 3 sets AFFECTS(p). Thus each $p \in W_j''$ is an end-point of at most $6k_t$ edges in E and therefore $|E| \leq 3k_t|W_j''|$. We conclude that $|W_j| \geq |W_j''|/7k_t$. □

Corollary 16. *If f is t-good then for each $j \in J$ we have $|W_j| \geq w_t$.*

Proof: Since f is t-good $|S_j| = s_t$. Then the corollary follows from Claim 13, Claim 14, and Claim 15. □

Claim 17. *If f is t-good then the number of groups used by algorithm CONSTRUCT is $w_t^{4/7}$.*

Proof: This follows from the definition of t-good and from the fact that the size of the groups is $r_t/w_t^{4/7}$. □

Claim 18. *If f is t-good and $t < T$ then the while loop in algorithm CONSTRUCT always terminates with $|S| = w_t^{1/7}$.*

Proof: We will show that if f is t-good then $|S| < w_t^{1/7}$ implies $|W_j - S - S'| > 0$. Suppose that some vertex p in W_j cannot be added to S. Then for some $p' \in V_1 \cup \cdots \cup V_{i-1}$ we have AFFECTS(p) ∩ AFFECTS(p') $\neq \emptyset$. But the size of each set V_α is at most $w_t^{1/7}$ and i is at most the number of groups, which is equal to $w_t^{4/7}$ by Claim 17. Furthermore, for each $p' \in V_1 \cup \cdots \cup V_{i-1}$, $|\text{AFFECTS}(p')| \leq 3k_t$. So at most $3k_t w_t^{5/7}$ members of R_j will be put in S'. By

Fact 8, $3k_t w_t^{5/7} < w_t - w_t^{1/7}$ for $t < T$ and big enough n. We conclude that if $|S| < w_t^{1/7}$ then $|W_j - S - S'| > w_t - (w_t^{1/7}) - (w_t - w_t^{1/7}) = 0$. □

Claim 19. *If f is t-good and $t < T$ then the probability that CONSTRUCT is successful is at least $1 - n^{-2}$.*

Proof: We have already shown in the proof of Claim 18 that if f is t-good then the while loop in algorithm CONSTRUCT always terminates with $|S| = w_t^{1/7}$. It remains to show that with probability at least $1 - n^{-2}$ each group i has a range j such that the function f returned by the call "Let f = RANDOMSET($S_j - S, f$)" does not map any processor in $S_j - S$ to '1'. Assume that this is true for groups 1 to $i - 1$. For $1 \leq v \leq i - 1$, let X_v be the random variable equal to the index of the first such range in group v. For $1 \leq j \leq r_t/w_t^{4/7}$, let $Y_{i,j}$ be a binary random variable which is 1 when range j is such a range for group i. Let $Z_i = \sum_{i=1}^{r_t/w_t^{4/7}} Y_{i,j}$. Note that Z_i is zero if and only if group i does not have such a range. Note that for $j \neq j'$, $Y_{i,j}$ and $Y_{i,j'}$ are independent. By construction, for any $b_1, \ldots, b_{i-1} \in [1, r_t/w_t^{4/7}]$, using the facts that $s_t \geq 2^{\log^{1/3} n}$ (from Fact 7, and $r_t/w_t^{4/7} > s_t^3$ (from Fact 9), and assuming n is large,

$$\begin{aligned}
\Pr(Z_i = 0 | X_{i-1} = b_{i-1}, \ldots, X_1 = b_1) \\
&= \Pr\left(\sum_{j=1}^{r_t/w_t^{4/7}} Y_{i,j} = 0 | X_{i-1} = b_{i-1}, \ldots, X_1 = b_1\right) \\
&= \Pr\left(\bigcap_{j=1}^{r_t/w_t^{4/7}} (Y_{i,j} = 0) | X_{i-1} = b_{i-1}, \ldots, X_1 = b_1\right) \\
&= \prod_{j=1}^{r_t/w_t^{4/7}} \Pr(Y_{i,j} = 0 | X_{i-1} = b_{i-1}, \ldots, X_1 = b_1) \\
&= \left(1 - \frac{\binom{w_t^{1/7}}{2}}{\binom{s_t}{2}}\right)^{r_t/w_t^{4/7}} \\
&\leq \left(1 - \frac{1}{s_t^2}\right)^{s_t^3} \\
&\leq e^{-s_t} \\
&\leq n^{-3}.
\end{aligned}$$

The probability of failing in any group can then be bounded by

$$\begin{aligned}
\textstyle\sum_{i=1}^{w_t^{4/7}} & \Pr(Z_i = 0|Z_{i-1} = 1,\ldots,Z_1 = 1) \\
&= \textstyle\sum_{i=1}^{w_t^{4/7}} \sum_{b_1,\ldots,b_{i-1}\in[1,r_t/w_t^{4/7}]} \Pr(Z_i = 0|X_{i-1} = b_{i-1},\ldots,X_1 = b_1) \\
&\qquad\qquad\qquad\qquad\qquad\qquad \Pr(X_{i-1} = b_{i-1},\ldots,X_1 = b_1) \\
&\le n^{-3} \textstyle\sum_{i=1}^{w_t^{4/7}} \sum_{b_1,\ldots,b_{i-1}\in[1,r_t/w_t^{4/7}]} \Pr(X_{i-1} = b_{i-1},\ldots,X_1 = b_1) \\
&= w_t^{4/7} n^{-3} \\
&\le n^{-2}. \ \square
\end{aligned}$$

Corollary 20. *If f is t-good and algorithm CONSTRUCT is successful then after CONSTRUCT is executed r_{t+1} ranges have s_{t+1} processors that are mapped to '*' by f', and no processors that are mapped to '1' by f', while the remaining ranges have no processors mapped to '*' by f', and two processors that are mapped to '1' by f'*

Proof: Immediate from the definition of successful and from Claim 17.□

Claim 21. *If f is t-good then after CONSTRUCT is executed every processor q that is in the $(t+1, f')$-knowledge set of a processor p has $p \in$ AFFECTS(q).*

Proof: By the definition of dependency sets, we can form a $(t+1, f')$ dependency set D of p by taking the union of the (t, f)-knowledge set of p and the (t, f)-knowledge sets of all processors p' satisfying the following: there is some refinement g of f which is a relevant 2-relation and on which p' sends to p on step $t+1$. Note that D is the union of the (t, f)-knowledge set of p and the set of processors that zero-affect p and the set of processors that one-affect p. If q is in the (t, f)-knowledge set of p then p is in AFFECTS(q) by the first part of the definition of AFFECTS. Suppose that q_1 is a processor in some range j which zero-affects p and that $p \notin$ AFFECTS(q). By the second part of the definition of AFFECTS we know that there are two processors in $S_j - W_j'$ which zero-affect q. If both of these are mapped to '0' by f' then for any refinement of f' processor p has a conflict at step $t+1$ so $D - q_1$ is a $(t+1, f')$-dependency set of p. If, on the other hand, one of these is mapped to '1' by f' then algorithm CONSTRUCT maps every member of the range of q_1 to '0' or '1' so $D - q_1$ is a $(t+1, f')$-dependency set of p. (Recall that if $f'(q_1) \neq$ '*' then q_1 cannot be in the $(t+1, f')$-knowledge set of any processor.) Similarly, suppose that q_2 is a processor in some range j which one-affects p and that $p \notin$ AFFECTS(q). By the third part of the definition of AFFECTS we know that every processor in W_j'' one-affects q. If all of the processors in $S_j - W_j''$ are mapped to '0' by f' then for any refinement of f' that is a relevant 2-relation processor p has a conflict at step

$t+1$ so $D - q_2$ is a $(t+1, f')$-dependency set of p. If, on the other hand, one of these is mapped to '1' by f' then algorithm CONSTRUCT maps every member of the range of q_2 to '0' or '1' so $D - q_2$ is a $(t+1, f')$-dependency set of p. □

Claim 22. *If f is t-good and algorithm CONSTRUCT is successful then after CONSTRUCT is executed the $(t+1, f')$-knowledge set of every processor p has size at most one.*

Proof: We know from Claim 21 that every processor p has a $(t+1, f')$-dependency set D which contains only those processors q such that $p \in$ AFFECTS(q). Suppose that two processors q and q' have $f'(q) = f'(q') =$ '$*$'. (If a processor q is not mapped to '$*$' by f' then it is not in the $(t+1, f')$-knowledge set of any processor so it is not in the $(t+1, f')$-knowledge set of p.) Then q must be in some $W_j \subset W''_j \subset W'_j$ and q' must be in some $W_{j'} \subset W''_{j'} \subset W'_{j'}$ and both q and q' are in the set V constructed by algorithm CONSTRUCT. If $j = j'$, then the definition of W_j guarantees that AFFECTS(q) $\cap$ AFFECTS(q') $= \emptyset$, implying that p is in just one of these sets, and thus either q or q' is not in D. If, on the other hand, $j \neq j'$ by the construction of V, AFFECTS(q) $\cap$ AFFECTS(q') $= \emptyset$, implying p is in just one of these sets, and thus either q or q' is not in D. Thus $|D| \leq 1$. □

Claim 23. *If f is t-good then after CONSTRUCT is executed each processor q is in the $(t+1, f')$-knowledge set of at most k_{t+1} processors.*

Proof: Let q be a processor which is in the $(t+1, f')$-knowledge set of a processor p. By Claim 21, $p \in$ AFFECTS(q). But by Claim 11, |AFFECTS(q)| $\leq 3k_t = k_{t+1}$. The claim follows. □

Lemma 24. *If $t < T$ and CONSTRUCT is called with (t,f), where f is t-good, then with probability at least $1 - n^{-2}$ CONSTRUCT will return a function f' which is $(t+1)$-good.*

Proof: This follows from Claim 19, Corollary 20, Claim 22, and Claim 23. □

5. Proof of the Theorem

We use the following function, which calls CONSTRUCT to generate a sequence of partial relevant 2-relations $f_0 = f_* \geq f_1 \geq \cdots \geq f_T \geq f$ in which each f_t is a refinement of f_{t-1}, f is a refinement of f_T, and f is a relevant 2-relation generated u.a.r.

```
Function GENERATE
    Let $f_0 = f_*$
    Let $f = f_0$
    Let $t = 0$
    While $t \leq T$ Do
        If for some $p$, $f(p) =$ '$*$' Then
            Let $f_t =$ CONSTRUCT$(t, f)$
        Else
            Let $f_t = f$
        $t = t + 1$
        $f = f_t$
    Let $P = \{p | f(p) = \text{'}*\text{'}\}$
    Return RANDOMSET$(f, P)$
End GENERATE
```

Lemma 25. *With probability at least* $1 - n^{-1}$ f_T *is* T*-good.*

Proof: Let Z_t be a random variable which is equal to 1 when CONSTRUCT succeeds at step t. Then by Lemma 24,

$$\Pr(Z_t = 0 | Z_{t-1} = 1, \ldots, Z_1 = 1) = \Pr(Z_t = 0 | f_t \text{ is } t\text{-good}) \leq n^{-2}.$$

The probability of failing at any step $t \leq T$ can then be bounded by

$$\sum_{t=1}^{T} \Pr(Z_t = 0 | Z_{t-1} = 1, \ldots, Z_1 = 1) \leq Tn^{-2} \leq n^{-1} \quad \Box$$

Theorem 26. *Let* A *be a deterministic algorithm that allegedly routes 2-relations in* $T = \sqrt{\log\log n/2}$ *steps. Let the input to* A *be drawn u.a.r. from the set of relevant 2-relations. Then the probability that* A *successfully routes the input is at most* $1/2$.

Proof: We will generate a relevant 2-relation by running algorithm GENERATE. By Claim 10 algorithm GENERATE generates relevant 2-relations u.a.r. GENERATE also produces a sequence $f_0 \geq \cdots f_T \geq \cdots f$ in which f is the final relevant 2-relation. By Lemma 25, f_T will be T-good with probability at least $1 - 1/n$.

Suppose that f_T is T-good. Then there is a range R that has a set S of s_T processors which are mapped to '$*$' by f_T. R has no processors which are mapped to '1' by f_T. Let d denote the first processor in range R. (d is the destination of the messages in range R.) The (T, f_T)-knowledge set of d contains at most one processor. There are three cases which must be examined concerning f_T:

CASE 1: The (T, f_T)-knowledge set of d contains a processor q which is a member of S:

We wish to bound the probability that A succeeds, given that f_T is in case 1. Let $\mathcal{F}_1$ denote the set of relevant 2-relations which refine f_T and map q to '1' and let $\mathcal{F}_0$ denote the set of relevant 2-relations which refine f_T and map q to '0' One can see by examining algorithm RANDOMSET that the probability that f is in $\mathcal{F}_1$ is $2/s_T$ and the probability that f is in $\mathcal{F}_0$ is $1 - 2/s_T$. We now examine the following sub-cases concerning f.

CASE 1A: f is in $\mathcal{F}_1$:

We wish to bound the probability that A succeeds, given that f is in $\mathcal{F}_1$. There is a particular trace τ which is the (T, f')-trace of d for every input h-relation $f' \in \mathcal{F}_1$. Since A runs in T steps processor d uses this trace τ to deduce the pair of messages that were destined for d in every input h-relation that is in $\mathcal{F}_1$. But there are $s_T - 1$ such pairs of messages, each of which is equally likely to come up in a randomly chosen member of $\mathcal{F}_1$. So the probability that A is successful given that f is in $\mathcal{F}_1$ is at most $1/(s_T - 1)$.

CASE 1B: f is in $\mathcal{F}_0$:

We wish to bound the probability that A succeeds, given that f is in $\mathcal{F}_0$. There is a particular trace τ which is the (T, f')-trace of d for every input h-relation $f' \in \mathcal{F}_0$. Since A runs in T steps processor d uses this trace τ to deduce the pair of messages that were destined for d in every input h-relation that is in $\mathcal{F}_0$. But there are $\binom{s_T-1}{2}$ such pairs of messages, each of which is equally likely to come up in a randomly chosen member of $\mathcal{F}_1$. So the probability that A is successful given that f is in $\mathcal{F}_1$ is at most $1/\binom{s_T-1}{2}$.

Therefore the probability that A succeeds given that f_T is in case 1 is at most $(2/s_T)(1/(s_T - 1)) + (1 - 2/s_T)(1/\binom{s_T-1}{2})$ which is at most $2/\binom{s_T-1}{2}$.

CASE 2: The (T, f_T)-knowledge set of d contains a processor q which is not a member of S:

Similar arguments to those used in case 1 show that the probability that A succeeds given that f_T is in case 2 is at most $1/\binom{s_T}{2}$.

CASE 3: The (T, f_T)-knowledge set of d is the empty set:

Similar arguments to those used in case 1 show that the probability that A succeeds given that f_T is in case 3 is at most $1/\binom{s_T}{2}$.

Finally, we conclude that the probability that A successfully routes f in T steps is at most the sum of $1/n$ (an upper bound on the probability that f_T is not T-good, by Lemma 25) and $(1 - 1/n) \times 2/\binom{s_T - 1}{2}$ (an upper bound on the probability that A succeeds given that f_T is T-good). We can use Fact 7 to show that this quantity is at most 1/2.

Therefore, with probability at least 1/2, an f drawn u.a.r. from the set of relevant 2-relations will not be routed by algorithm A in T steps. □

Corollary 27. *Let A be a deterministic algorithm that routes 2-relations. Let the input to A be drawn u.a.r. from the set of relevant 2-relations. Then the expected number of communication steps used by A is at least $\sqrt{\log\log n}/4$.*

Proof: The corollary follows from the fact that $\sqrt{\log\log n}/4 \leq (1/2)(T + 1)$. *square*

Theorem 28. *Let A be a randomized algorithm that routes 2-relations. Then there is a 2-relation on which the expected number of communication steps used by A is at least $\sqrt{\log\log n}/4$.*

Proof: Using a Theorem by Yao, the expected number of communication steps used by A maximized over all possible inputs is at least the expected running time for the uniform distribution on relevant 2-relations, minimized over all deterministic algorithms, which is at least $\sqrt{\log\log n}/4$ by Corollary 27. □

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DATE
FILMED

4/7/94

END