Generalized Thinning Protocols for Routing h-Relations in Complete Networks

A. Kautonen, V. Leppänen, M. Penttonen

Report A/1999/3

ISBN 951-781-611-1
ISSN 0787-6416

UNIVERSITY OF KUOPIO

Department of Computer Science
and Applied Mathematics

P.O.Box 1627, FIN-70211 Kuopio, FINLAND

Generalized Thinning Protocols for Routing h-Relations in Complete Networks

Anssi Kautonen[*] Ville Leppänen[†] Martti Penttonen[‡]

Abstract

We present a routing algorithm called *generalized thinning* algorithm for complete networks under OCPC assumption. This algorithm generalizes earlier versions of thinning, which were proved to be competitive in comparison with other algorithms found in literature.

1 Introduction

Routing algorithms have many applications in data communication. Our work is motivated by the emulation of shared memory with distributed memory modules. If suitable techniques, such as the *slackness* principle and randomized hashing [18], are used, implementation of shared memory can be reduced to efficient routing of an h-relation in a complete network under OCPC assumption. By definition, in an *h-relation* there is a complete graph, where each node (processor) has at most h packets to send, and it is the target of at most h packets. By *OCPC* (Optical Communication Parallel Computer) or *1-collision* assumption [1], if two or more packets arrive at a node simultaneously, all fail. An implementation of an h-relation is *work-optimal* at *cost c*, if all packets arrive at their target in time ch.

The first attempt to implement an h-relation is to use *greedy* routing algorithm of Figure 1. By greedy principle, one tries to send packets as fast as one can. The fatal drawback of the greedy algorithm is the *livelock*: Some packets can cause mutual failure of sending until eternity. Consider the situation, when two processors have each one packet targeted to the same processor. Due to greediness they are forced to send — and fail for ever since then.

```
proc greedy
for all processors pardo
    while processor has packets do
        choose an unsent packet at random and try to send it
```

Figure 1: Greedy routing algorithm.

The livelock can be avoided. Such an algorithm was provided by Anderson and Miller [1], and it was improved by Valiant [17]. They realize work-optimally an h-relation for $h \in \Omega(\log p)$, where p is the number of processors. Other algorithms with even lower latency were proposed by [6, 7, 2, 3, 11]. Contrary to these theoretically strong algorithms, the algorithm of Geréb-Graus and Tsantilas [5] in Figure 1 has the advantage of being *direct*, i.e. the packets are sent to their targets directly, without intermediate nodes. For other results related with direct or indirect routing, see also [4, 16, 8, 9, 12, 15]. The algorithm of Geréb-Graus and Tsantilas routes h-relations work-optimally for $h \in \Omega(\log p \log\log p)$, where p is the number of nodes. Indeed, for $h \in \Omega(\log p \log\log p)$, GGT routes any h-relation in time

[*] University of Joensuu, Department of Computer Science, P.O.Box 111, 80101 Joensuu, Finland, email Anssi.Kautonen@cs.joensuu.fi

[†] University of Turku, Department of Computer Science, Lemminkäisenkatu 14 A, 20520 Turku, Finland, email Ville.Leppanen@cs.utu.fi

[‡] University of Kuopio, Department of Computer Science and Applied Mathematics, P.O.Box 1627, 70211 Kuopio, Finland, email Martti.Penttonen@cs.uku.fi

$O(h)$ with probability higher than $1-1/p^\alpha$ for any $\alpha \geq 1$. Due to its simplicity and directness, we choose the GGT algorithm as our reference point. Our new algorithm presented in Section 2 is inspired by [15], although the latter deals with the continuous routing problem, not the h-relation.

```
proc GGT($h$,$\epsilon$,$\alpha$)
for $i = 0$ to $\log_{1/(1-\epsilon)} h$ do
    for all processors pardo
        for $e(\epsilon h + max\{\sqrt{4\epsilon\alpha h \ln p}, 4\alpha \ln p\})/(1-\epsilon)$ times do
            choose an unsent packet $x$ at random
            attempt to send $x$ with probability # unsent packets / $h$
    $h := (1-\epsilon)h$
```

Figure 2: Gereb–Graus and Tsantilas algorithm.

2 Thinning protocols

The throughput of the greedy routing of randomly addressed packets is characterized by

$$(1-\frac{1}{h})^{h-1} \approx \frac{1}{e}$$

where $1/h$ is the probability that one of the $h-1$ competing processors is sending to the same processor at the same time. (For all $x > 0$, $(1-1/x)^{x-1} \geq e^{-1}$.) This would be the throughput if all processors would create and send a new randomly addressed packet, which is not the case in routing an h-relation. It may happen that at the end only two processors have a packed, addressed to the same target. In this situation, under the OCPC assumption, the greedy algorithm, which always tries both packets, ends up in a livelock. The solution is to decrease the sending probability.

In the GGT algorithm, the sending probability of packets varies between 1 and $1-\epsilon$ ($0 < \epsilon < 1$). The transmission of packets is thus 'thinned' by factor 1 to $1/1-\epsilon$, which prevents the livelock.

We now propose a very simple routing protocol, where thinning is more explicit. In basic form of the algorithm, packets are routed phase by phase. In each phase, each packet (in random order) is tried once. Thinning by factor t means sending h packets in ht units of time. Thinning factor may or may not change from phase to phase. In the algorithm of Fig. 2, we have a thinning function $t(i)$ that depends on time. It is related with $\tau(i)$ that characterizes how the problem size h is guaranteed to decrease from phase to phase.

```
proc Thinning($h$,$h_0$,$t$,$\tau$)
for all processors pardo
    $i := 1$
    while $h > h_0$ do
        Transmit $h$ packets (if so many remain) within time $[1..\lceil t(i)h \rceil]$
        $h := (1-e^{-1/\tau(i)})h$; $i := i+1$
    while packets remain do
        Transmit the remaining packets in time $t(i)h_0$
```

Figure 3: General thinning algorithm.

An obvious drawback of thinning is that a processor cannot successfully send a packet at those moments, when it does not even try to send. Thinning by factor t would thus imply inefficiency by factor t. However, this is somewhat balanced by the success probability, which increases from $1/e$ to $1/e^{1/t}$. This is seen as follows.

Consider an h-relation being routed under OCPC assumption and thinning factor t. When a packet is being tried, there may be k other packets, $k < h$, that try to arrive at the same target. Thus, the success probability of our packet is

$$(1 - \frac{1}{th})^k = [(1 - \frac{1}{th})^{th}]^{k/th} \geq \frac{1}{e^{1/t}}. \tag{1}$$

Hence, the number of packets would decrease from h to $(1 - e^{1/t})h$ in time th and thus the expected throughput time would be $te^{1/t}$. It is interesting to see that the growth of this function

t	1.0	1.2	1.5	2.0	2.5	3.0	4.0
$te^{1/t}$	2.7	2.8	2.9	3.3	3.7	4.2	5.1

is very modest for small values of $t > 1$ — this is a small price for the robustness.

Even though a sending probability less than 1 eliminates the deadlock, it does not guarantee fast throughput. When h decreases, the sending becomes less and less random, and a fixed throughput cannot be guaranteed. For that reason, we set a minimum size $h_0 \in \Omega(\log p)$ for thinning window, preventing repeated collisions. Furthermore, for high probability proof, we estimate the throughput very conservatively. Therefore we compress h by factor $(1 - e^{-1/\tau(i)})$ with suitable $\tau < t$ and not by $(1 - e^{-1/t(i)})$ as can be expected, and still manage to route all packets in time $O(h)$.

3 Analysis

One can prove that

1. the number of unsent packets decreases geometrically from h to $O(\log p)$
2. the rest of the packets can be routed in time $O(\log p \log\log p)$

We will need

Lemma 3.1 $t(1 - e^{-1/t}) < \frac{1}{1+1/2t}$, *for all* $t > 0$.

Proof. By using the Taylor series for e^x, we have

$$\begin{aligned} t(1 - e^{-1/t}) &= \frac{-t + te^{1/t}}{e^{1/t}} = \frac{-t + t + t\frac{1}{t} + t\frac{1}{2!t^2} + t\frac{1}{3!t^3} + \dots}{1 + \frac{1}{t} + \frac{1}{2!t^2} + \frac{1}{3!t^3} + \dots} \\ &= \frac{1 + \frac{1}{2!t} + \frac{1}{3!t^2} + \frac{1}{4!t^3} + \dots}{1 + \frac{1}{t} + \frac{1}{2!t^2} + \frac{1}{3!t^3} + \dots} = \frac{1}{1 + \frac{1}{2t} + R} \end{aligned}$$

where $R = [\frac{1}{2!t^2}(\frac{1}{2} - \frac{1}{3}) + \frac{1}{3!t^3}(\frac{1}{2} - \frac{1}{4}) + \cdots]/[1 + \frac{1}{2!t} + \frac{1}{3!t^2} + \cdots] > 0$, for all $t > 0$. ■

Theorem 3.2 *If* $c \leq t(1) \leq t(2) \leq \dots$ *for some constant* $c > 1$, $\tau(i) = t(i+1)/t(i) < d$ *for some constant* $1 \leq d < c$, *and* $h \in \Omega(\log p \log\log p)$, *then Thinning routes any h-relation in time $O(h)$ with high probability.*

Proof. Consider the i'th round of the while-loop, when $h = h_i$, where $h_i \geq k_0 \log p = h_0$ for a certain constant k_0. We assume that at the beginning of such a routing round the routing task is a full h_i-relation, and aim to show that after the round the routing situation has reduced to an h_{i+1}-relation with high probability. We may assume fullness, because otherwise we can add 'dummy packets' with proper destination to those processors not having initially h_i packets. The dummy packets participate routing as the other packets. In the analysis below, the dummy packets can collide with normal packets as well as with other dummy packets, but in the actual algorithm attempting to route a dummy packet corresponds to an unallocated time slot (unable to cause any collisions). Thus the number of successful normal packets is always better in the actual situation.

By (1), the expectation for successful packets is $E_i = h_i / e^{1/t(i)}$. Let N be the number of successful packets, and apply Chernoff bound [10]

$$Pr(N < (1-\epsilon)E_i) \le e^{-\frac{1}{2}\epsilon^2 E_i}$$

with $(1-\epsilon)E_i = h_i / e^{1/\tau(i)}$. Thus,

$$\epsilon = 1 - e^{\frac{1}{t(i)} - \frac{1}{\tau(i)}} \ge 1 - e^{\frac{1}{c} - \frac{1}{d}}.$$

Therefore

$$Pr(N < h_i / e^{1/\frac{1}{\tau(i)}}) \le e^{-\frac{1}{2}\epsilon^2 e^{-1/t(i)} h_i} \le e^{-\frac{\epsilon^2}{2e} h_i} \le \frac{1}{p^{2.5}}$$

for $h_i > h_0 = \frac{5e}{\epsilon^2} \log p \in O(\log p)$.

In other words, in one round and one processor, the number of outgoing packets decreases to fraction $1 - e^{-1/\tau(i)} < 1 - 1/e$ with probability higher than $1 - p^{-2.5}$. Hence, in all p processors, in $\log_{1/(1-1/e)} h/h_0 \le \log_{1/(1-1/e)} h < \sqrt{p}$ or fewer phases the number of outgoing packets decreases to h_0 with probability higher than $1 - 1/p$. Respectively, the number of incoming packets decreases to h_0 at the same rate. Removing the dummy packets from the system can only decrease the degree of the relation.

It was already stated that the level h_0 will be achieved in $O(\log(h/h_0))$ phases. We shall now show that these phases do not take more than $O(h)$ time. Indeed, consider two successive phases i and $i+1$ and the numbers S_i and S_{i+1} of their routing steps. Then

$$\frac{S_{i+1}}{S_i} = \frac{t(i+1)h_{i+1}}{t(i)h_i} = \frac{t(i+1)}{t(i)}(1 - e^{-1/\tau(i)}) = \tau(i)(1 - e^{-1/\tau(i)})$$

$$< \frac{1}{1 + 1/2\tau(i)} \le \frac{1}{1 + 1/2d} < 1$$

by Lemma 3.1. Hence, $S_1, S_2, S_3, \ldots$ form a geometric series, whose sum is $O(h)$.

Observe that by choosing a larger h_0 as in the analysis above, we can easily show the same progression in the degree of the relation with probability $1 - p^{-\alpha}$ for any positive constant α.

For the rest of the algorithm, the while-loop with h-relation level at most h_0, consider sets of packets with the same target. When the size of such a set is $h' \le h_0$, the success probability of one such packet is

$$h' \times \frac{1}{h_0} \times (1 - \frac{1}{h_0})^{h'-1} \ge \frac{h'}{h_0}(1 - \frac{1}{h_0})^{h_0 - 1} \ge \frac{h'}{h_0}\frac{1}{e}$$

Thus the expectation of sending times for one such packet is at most eh_0/h'. The sum of all these expectations, until all such packets have been sent, is

$$eh_0(\frac{1}{h'} + \frac{1}{h'-1} + \ldots + \frac{1}{2} + 1) = E \in O(h_0 \log h_0).$$

By another form of Chernoff bound [10]

$$Pr(T > r) \le \frac{1}{2^r} \quad \text{for } r > 6E$$

we see that

$$Pr(T > k_1 h_0 \log h_0) \le \frac{1}{p^{2.5}}$$

for some k_1 and therefore it is possible to transmit all such packets to their target in time $O(\log p \log\log p)$ with high probability. Finally, observe that there are at most p such groups of packets.

By combining the two phases we see that all packets can be routed in time $O(h + \log p \log\log p)$ with high probability. By choosing $h \in \Omega(\log p \log\log p)$ we complete the proof. ■

The purpose of Theorem 3.2 is mainly to show the existence of such $t(i)$- and $\tau(i)$-sequences that the thinning algorithm works with high probability work-optimally for $h \in \Omega(\log p \log\log p)$. Especially, Theorem 3.2 does not characterize all such sequences. The limiting condition $1 \leq d < c \leq t(1)$ of Theorem 3.2 is due to lower bounding ϵ by a constant, when $\tau(i)$ is bound to $t(i+1)/t(i)$. The selection of $\tau(i)$ is due to forcing window sizes to form a simple geometric series. In general, window sizes, S_i's, do not need to form a geometric series – the only requirement is that $\sum S_i = O(h)$. Thus, $t(i)$ does not need to be a monotonically increasing series. Necessarily, $\tau(i) < t(i)$ and the difference (or ratio) of $t(i)$ and $\tau(i)$ is used to gain high probability in decreasing the problem size by $1 - e^{-1/\tau(i)}$. Decrease by factor of $1 - e^{-1/\tau(i)}$ could also be achieved by repeating the routing attempt by thinning factor $t(i)$ by some constant number of times. Observe that the selection of $\tau(i)$ in the theorem does not allow $t(1) = 1$ — but it is easy to see that having $t(1) = 1$ gives the best throughput in the first round. Finally, notice that Theorem 3.2 provides running time $O(h)$ with high probability — giving up of high probability would allow us to decrease h_0 and narrow the gap between $t(i)$ and $\tau(i)$.

The condition for thinning factor t is quite general. Consider three cases

1. $1 < t(1) = t(2) = t(3) = \ldots$. This is the case of *constant* thinning (CT) [13]
2. $t(i) = 1 + i \times t$ for some $t > 0$. This is called *linear* thinning (LT) in [13]
3. $t(i) = t^i$ for some $t > 1$. This case is called *geometric* thinning (GT) in [13, 14]
4. $t(i) = t_1 + (t_2 - t_1)(\tanh(i-k) - \tanh(-k))/(1 - \tanh(-k))$ for some $1 < t_1 < t_2$ and $k \geq 1$. We call this function *sigmoid* thinning (ST).

From Theorem 3.2 we get

Corollary 3.3 *If $h \in \Omega(\log p \log\log p)$, then constant, linear, exponential, and sigmoid thinning route any h-relation in time $O(h)$ with high probability.*

4 Experiments

We ran some preliminary experiments to get practical experience of the new algorithm, see Tables 1 and 2. In the experiments of Table 1, the number of processors was $p = 1024$, and slackness factors $h = 16, 32, 64, 128, 256$ were tried. The results for GGT are taken from [14] and the results for CT, GT and ST are averages of 100 experiments. In CT and GT experiments, $h_0 = 8$ was used. However, in ST experiments $h_0 = 1$ was used, somewhat surprisingly. It proved out that if t grows high enough soon enough, h can decrease towards 0 continuously. The acknowledgement of packets does not counted. Collisions within processors were avoided by allocating a unique sending moment of time for each packet from the time window $[1 \ldots th]$.

h	GGT	CT1.1	LT1.1	GT1.1	ST4.0
16	5.8	5.2	5.7	5.9	5.2
32	5.8	4.4	4.6	4.7	4.6
64	5.7	3.8	3.9	4.0	3.9
128	5.6	3.4	3.5	3.6	3.5
256	5.6	3.2	3.4	3.4	3.2

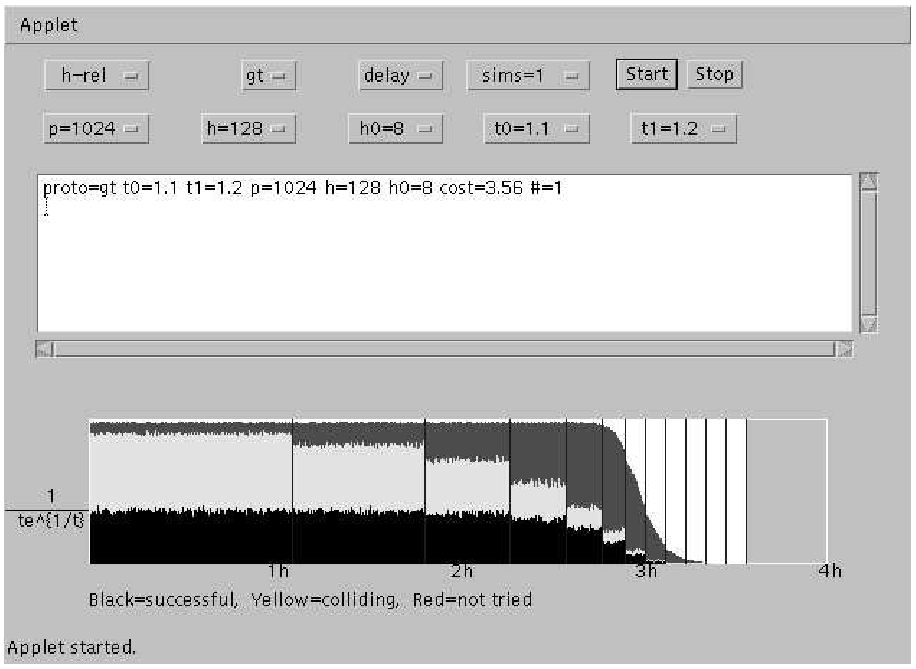
Table 1: Routing cost of the GGT, CT, LT, GT and ST algorithms. In GGT, $\epsilon = 0.5$, $\alpha = 0.01$ were safe and fast. In CT1.1, $t(0) = t(1) = \cdots = 1.1$. In LT1.1, $t(0) = 1.1, t(1) = 1.2$, etc, $t(i)$ grows linearly. In GT1.1, $t(0) = 1.1, t(1) = 1.2$ etc, $t(i)$ grows geometrically. In ST4.0, t grows from 1.1 to 4.0 guided by a "sigmoid" function $t(i) = t_1 + (t_2 - t_1)(\tanh(i-k) - \tanh(-k))/(1 - \tanh(-k))$. The point of maximal growth k was between 6 and 10. In CT, LT, GT, and SG experiments, $\tau(i) = t(i+1)/t(i)$ as in Theorem 3.2.

h_0	CT1.1	GT1.1
4	3.5	3.6
6	3.7	3.7
8	3.7	3.5
12	3.9	3.5
16	3.8	3.6
24	4.2	3.8
32	3.8	4.0

Table 2: The effect of h_0 on the cost, when $p = 1024$ and $h = 128$. In CT1.1 $t(0) = t(1) = \cdots = 1.1$. In GT1.1 $t(0) = 1.1, t(1) = 1.2$ and t grows geometrically.

The Table 2 demonstrates the effect of h_0. Note that the high reliability requires a lower bound for h_0 (except in case ST, where $h_0 = 1$ was used), but obviously too high value of h_0 implies inefficiency. By the results in Table 1 and Table 2, the constant thinning algorithm CT and the geometric thinning algorithm GT appear to work better than the GGT algorithm.

In addition to mere numbers, our routing simulator shows the progress of routing graphically, see Figure 4.



h-rel gt delay sims=1

p=1024 h=128 h0=8 t0=1.1 t1=1.2

```
proto=gt t0=1.1 t1=1.2 p=1024 h=128 h0=8 cost=3.56 #=1
```

Black=successful, Yellow=colliding, Red=not tried

Applet started.

Figure 4: Routing simulator. The bottom band of the graphical representation shows the proportion of successful processors, the middle band failed processors, and the top band passive processors. The vertical lines at intervals of $t(i)h$, $i = 0, 1, \ldots$, until $t(i) < h_0$, and after that at intervals of $t(i)h_0$, separate the phases of the algorithm.

5 Conclusions

We have presented a simple direct routing protocol for routing h-relations work-optimally in complete networks. Studying the h-relation as opposed to a continuous routing setting is motived by implementation of shared memory abstraction and the BSP model [18]. On the other hand, in contemporary data communication situations optical and wireless networks can be seen as complete networks.

Our thinning protocol is very simple and since our preliminary results indicate the routing cost to be rather close to the optimum, we believe our thinning protocol to be practical. Theorem 3.2 shows the existence of proper $t(i)$- and $\tau(i)$-sequences, but does not attempt to fully characterize all such sequences that guarantee the algorithm to solve the h-relation routing problem work-optimally. Quite obviously, there exists many such sequences not characterized by Theorem 3.2.

In the future, we aim to study $t(i)$- and and $\tau(i)$-sequences more extensively to achieve smaller routing cost. On the other hand, we would like to apply the ideas behind the thinning algorithm to continuous routing settings.

References

[1] R.J. Anderson and G.L. Miller. Optical communication for pointer based algorithms. Technical Report CRI-88-14, Computer Science Department, University of Southern California, LA, 1988.

[2] A. Czumaj and F. Meyer auf der Heide, and V. Stemann. Shared memory simulations with triple-logarithmic delay. *Proc. ESA95*, 46–59, 1995.

[3] M. Dietzfelbinger and and F. Meyer auf der Heide. Simple, efficient shared memory simulations. *Proc. SPAA93*, 110–119.

[4] J. Håstad, T. Leighton, and B. Rogoff. Analysis of backoff protocols for multiple access channels. *SIAM J. Comput.* 25:740–774, 1996

[5] M. Geréb-Graus and T. Tsantilas. Efficient optical communication in parallel computers. In *SPAA'92, 4th Annual Symposium on Parallel Algorithms and Architectures, San Diego, California*, pages 41 – 48, June 1992.

[6] L.A. Goldberg, M. Jerrum, T. Leighton, and S. Rao. A doubly logarithmic communication algorithm for the completely connected optical communication parallel computer. In *SPAA'93, 5th Annual Symposium on Parallel Algorithms and Architectures, Velen, Germany*, pages 300 – 309, June 1993.

[7] L.A. Goldberg, Y. Matias, and S. Rao. An optical simulation of shared memory. In *SPAA'94, 6th Annual Symposium on Parallel Algorithms and Architectures, Cape May, New Jersey*, pages 257 – 267, June 1994.

[8] L.A. Goldberg and P.D. MacKenzie, Analysis of Practical Backoff Protocols for Contention Resolution with Multiple Servers. Proceedings of SODA 7 (1996) 554-563.

[9] L.A. Goldberg and P.D. MacKenzie, Contention Resolution with Guaranteed Constant Expected Delay, Proceedings of the Symposium on Foundations of Computer Science 38 (1997) 213-222.

[10] T. Hagerup, C. Rüb. A guided tour of Chernoff bounds. *Information Processing Letters* 33:305–308, 1989.

[11] R.M. Karp, M. Luby, and F. Meyer auf der Heide. Efficient PRAM simulation on a distributed memory machine. *Proc. 24th Annual ACM Symposium on Theory of Computing*, 318–326, 1992.

[12] A. Kautonen and V. Leppänen and M. Penttonen. Simulations of PRAM on Complete Optical Networks. In *Proc. EuroPar'96*, LNCS 1124:307–310.

[13] A. Kautonen and V. Leppänen and M. Penttonen. Constant thinning protocol for routing h-relations in complete networks. To appear in *Proc. EuroPar'98*, LNCS.

[14] A. Kautonen, V. Leppänen and M. Penttonen). Thinning protocols for routing h-relations in complete networks. In: R. Freivalds (ed.). *Proceedings of International Workshop on Randomized Algorithms*, Brno, Czech Republic, August 27-28, 1998, p. 61-69. 1998.

[15] M. Paterson and A. Srinivasan. Contention resolution with bounded delay. In *Proc. FOCS'95*, IEEE Computer Society Press, 104–113.

[16] P. Raghavan, and E. Upfal. Stochastic Contention Resolution With Short Delays. In *Proc. STOC'95*, 229–237.

[17] L.G. Valiant. General purpose parallel architectures. In *Handbook of Theoretical Computer Science, Vol. A*, 943–971, 1990.

[18] L.G. Valiant. A bridging model for parallel computation. *Communications of the ACM*, 33:103-111, 1990.