Thinning Protocols for Routing h-Relations in Complete Networks

Anssi Kautonen* Ville Leppänen† Martti Penttonen*

Abstract

We present two simple routing protocols, called *constant thinning* and *geometric thinning* protocol, for complete network under OCPC assumption, analyze them, and compare them with each other and some other routing protocols.

1 Introduction

Parallel programmers would welcome the "flat" shared memory, because it would make PRAM [16] style programming possible and thus make easier to utilize the rich culture of parallel algorithms written for the PRAM model. As large memories with a large number of simultaneous accesses do not seem feasible, the only possible way to build a PRAM type parallel computer appears to be to build it from processors with local memories. The fine granularity of parallelism and global memory access, what makes the PRAM model so desirable for algorithm designers, sets very high demands for data communication. Fortunately, memory accesses at the rate of the processor clock are not necessary, but due to the parallel *slackness* principle [17], latency in access does not imply inefficiency. It is enough to route an *h-relation* efficiently. 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. We assume the *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. 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 $i \in \{1, \ldots, P\}$ **pardo**
  while processor i has packets **do**
    choose an unsent packet at random and try to send it

*University of Joensuu, Department of Computer Science, P.O.Box 111, 80101 Joensuu, Finland, email {Anssi.Kautonen,Martti.Penttonen}@cs.joensuu.fi

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

Another routing algorithm was proposed by Anderson and Miller [1], and it was improved by Valiant [16]. 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 provided by [6, 7, 2, 3, 11]. Contrary to these, the h-relation algorithm of Geréb-Graus and Tsantilas [5] has the advantage of being *direct*, i.e. the packets go to their target directly, without intermediate nodes. For other results related with direct or indirect routing, see also [4, 15, 8, 9, 12, 14]. The algorithm of Geréb-Graus and Tsantilas runs work-optimally for $h \in \Omega(\log p \log\log p)$. 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 [14], although the latter deals with the continuous routing problem, not the h-relation.

proc GGT(h,ϵ,α)
for $i = 0$ **to** $\log_{1/(1-\epsilon)} h$ **do** Transmit($(1-\epsilon)^i h$,ϵ,α)

proc Transmit(h,ϵ,α)
for all processors P **pardo**
  for $\frac{e}{1-\epsilon}(\epsilon h + max\{\sqrt{4\epsilon\alpha h \ln p}, 4\alpha \ln p\})$ times **do**
    choose an unsent packet x at random
    attempt to send x with probability $\frac{\#\ unsent\ packets}{h}$

As another point of reference we take what is called Penalty algorithm in [12].

proc Penalty(f: function)
for all processors P **do**
while processor P has unsent packets **do**
  choose a packet x at random
  if $1/f$(number of failures of x) $\geq$ RandomNumber$[0 \ldots 1)$ **then**
    attempt to send x

In Penalty algorithm, each packet has an individual failure history, and failures decrease the sending probability. In literature, similar protocols are often called *backoff* protocols, and Ethernet, with $f(i) = \min\{2^i, 2^{10}\}$ is a famous representative of this class [4].

2 Thinning protocols

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

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

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 $1/1-\epsilon$, preventing the livelock. We now propose a very simple routing protocol, where thinning is more explicit. We have two flavors of the algorithm, CT for *constant thinning* [13] and GT for *geometric thinning.*

proc CT(h,h_0,t,δ)
% $t, \delta \geq 1$
for all processors **do**
  while packets remain **do**
    Transmit h packets (if so many remain) within time $[1..\lceil \delta th \rceil]$
    $h := \max\{(1-e^{-1/t})h, h_0\}$

proc GT(h,h_0,d,δ,t_{max})
% $d, \delta, t_{max} \geq 1$
for all processors **do**
  $t := 1$
  while packets remain **do**
    Transmit h packets (if so many remain) within time $[1..\lceil \delta th \rceil]$
    $h := \max\{(1-e^{-1/t})h, h_0\}; t := min(t_{max}, dt)$

In the algorithms above, transmitting (at most) h packets means that h moments of time are allocated at random to the packets from the time window $[1..\lceil \delta th \rceil]$. Thus, in the transmission phase each packet is tried only once.

A 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. But this is somewhat balanced by the success probability, which increases from $1/e$ to

$$(1-\frac{1}{th})^{h-1} \geq \frac{1}{e^{1/t}}.$$

The expected throughput with thinning is thus characterized by the function $te^{1/t}$ whose growth

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 with small values of $t > 1$, which is a small price for the robustness.
Even though a sending probability less than 1 eliminates the deadlock, it does not guarantee fast throughput. For that reason, the CT has a minimum size $h_0 \in \Omega(\log p)$ for thinning window, to prevent repeated collisions. In the GT this is not a problem, but a lower bound h_0 prevents 'starvation' of the processor. Note also that until this lower bound is achieved, the time window of one phase decreases geometrically, because $t(1-e^{-1/t}) < 1$ for all t.

3 Analysis

The analyses for GGT, CT, and GT are very similar. One can prove that

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

Since a proof for CT was presented in [13], we present here only the proof for the GT. To show that the GT does not use too many routing steps, we need Lemma 3.1.

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

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

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

since pairwise $\frac{1}{(i+1)!t^i} < \frac{1}{i!t^i}$ for all t and natural numbers $i > 0$. ∎

Theorem 3.2 *For $h \in \Omega(\log p \log\log p)$, the GT routes any h-relation in time $O(h)$ with high probability.*

Proof. Consider the i'th round of the while-loop, when h is set to h_i and t is set to t_i, $h_i \geq k_0 \log p = h_0$ for a suitable constant k_0. For the values of h_i and t_i our algorithm sets $t_i = min(t_{max}, d^{i-1})$, $h_1 = h$ and $h_i = (1-e^{-1/t_{i-1}})h_{i-1}$, $i \geq 2$. We assume that in the beginning of each such a routing round the routing situation is an 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.

To ease the calculation of probabilities, we worsen the routing situation a little by completing each initial h_i-relation to a full h_i-relation. We do this by adding 'dummy packets' with proper destination to those processors not having initially h_i packets. We assume that 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.

We still need to define precisely the transmission phase in order to calculate the success of sending. Each packet is given a transmission moment from the time window $[1 \ldots \delta t_i h_i]$ at random. However, if several packets in the same processor get the same transmission moment, no one is transmitted at that moment. In other words, we assume the 1-collision property also inside a processor, not only between the processors.

We show first that with high probability, the number of packets decreases to $h_{i+1} = (1-e^{-1/t_i})h_i$ or below. As a packet has the probability $1/(\delta t_i h_i)$ of being sent at a given moment of time (from the interval $[1 \ldots \lceil \delta t_i h_i \rceil]$), and by the h_i-relation assumption, at most h_i packets have the same target, the probability of success for a packet at a given moment of time is at least

$$\frac{1}{\delta t_i h_i} \times (1 - \frac{1}{\delta t_i h_i})^{h_i - 1} \geq \frac{1}{e^{1/(\delta t_i)} \cdot \delta t_i h_i},$$

since

$$(1 - \frac{1}{\delta t_i h_i})^m \geq (1 - \frac{1}{\delta t_i h_i})^{h_i - 1}$$

for any $m \leq h_i - 1$ (consider m as the actual number of other packets with the same target) and

$$(1 - \frac{1}{x h_i})^{h_i - 1} = \left((1 - \frac{1}{x h_i})^{x h_i - x}\right)^{1/x} \geq \left((1 - \frac{1}{x h_i})^{x h_i - 1}\right)^{1/x} \geq e^{-1/x}$$

for $h_i > 1$ and $x \geq 1$.
Since there are $t_i h_i$ equally suitable moments of time for each packet, the probability of success for a packet during this final stage is at least

$$\frac{1}{e^{1/(\delta t_i)} \cdot \delta t_i h_i} \times \delta t_i h_i = e^{-1/(\delta t_i)}.$$

Hence, the expected number of successful packets of a processor within these $\delta t_i h_i$ units of time is $E_i = h_i / e^{1/(\delta t_i)}$. Let N be the number of successful packets, and apply Chernoff bound [10]

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

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

$$\epsilon = 1 - e^{\frac{1-\delta}{\delta t_i}} \geq 1 - e^{\frac{1-\delta}{\delta t_{max}}} \geq \epsilon_{min}$$

By suitably choosing $\delta > 1$ and $t_{max} \geq 1$, we can set $\epsilon_{min} = 0.3127$. If $\delta t_i \geq 1.6$, then $e^{1/(\delta t_i)} \leq 1.868$ and

$$Pr(N < h_i / e^{1/t_i}) \leq e^{-0.5 \times 0.3127^2 \times h_i / 1.868} = e^{-0.026 h_i} \leq \frac{1}{p^{2.5}}$$

for $h_i > h_0 = 100 \ln p$. These numerical values are just technical details and uninteresting from the practical point of view.

Hence, in all p processors, in all $\log_{1/(1-e)} h < \sqrt{p}$ or fewer rounds (when the current degree of h_i-relation satisfies $h_0 \leq h_i \leq h$), the number of outgoing packets decreases by compression factor $c_i = 1 - 1/e^{1/t_i}$ with probability $1 - 1/p$. Guaranteeing that the number of incoming packets decreases to at most $c_i h_i$ for each processor is analyzed respectively. Clearly, the full h_i-relation (completed with dummy packets) decreases to $c_i h_i = (1 - 1/e^{1/t_i}) h_i$-relation. Removing the dummy packets from the system can only decrease the degree of the relation.

Since the compression factor at round i is $c_i \leq 1 - e^{-1}$ the level h_0 will be achieved. We show that – excluding the first two rounds – the number of routing steps $S(i) = \delta t_i h_i$ used at round i satisfies $S(i) \geq S(i+1) \times min(1 - e^{-1}, 1/d), i \geq 3$. I.e., if $d > 1$, the numbers $S(3), S(4), \ldots$ form a geometrically converging series, and thus achieving level h_0 requires $O(\delta h)$ routing steps.

We need to study the situation before and after t_i reaches level t_{max}. After level t_{max}, the ratio $S(i)/S(i+1)$ clearly is $c_i = 1 - e^{-1/t_i} \leq 1 - e^{-1}$. Before level t_{max}, the ratio $S(i)/S(i+1)$ is

$$\frac{t_i}{t_{i+1}(1 - e^{-1/t_i})} = \frac{1}{d(1 - e^{-1/d^{i-1}})} \leq \frac{1}{d^{i-2}} \leq \frac{1}{d}$$

by Lemma 3.1 and since $i \geq 3$.

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 $h' \leq h_0$, the success probability of one such packet is

$$h' \times \frac{1}{\delta t_i h_0} \times (1 - \frac{1}{\delta t_i h_0})^{h'-1} \geq \frac{h'}{\delta t_i h_0} \times (1 - \frac{1}{\delta t_i h_0})^{h_0 - 1}$$

$$\geq \frac{h'}{h_0} \frac{1}{\delta t_i e^{1/(\delta t_i)}} \geq \frac{h'}{h_0} \frac{1}{\delta t_{max} e^{1/(\delta t_{max})}},$$

since $xe^{1/x} < (x + \gamma)e^{1/(x+\gamma)}$ for any $\gamma > 0$ and $x \geq 1$. Thus the expectation of sending times for one such packet is at most $\delta t_{max} e^{1/(\delta t_{max})} h_0 / h'$. The sum of all these expectations, until all such packets have been sent, is

$$\delta t_{max} e^{1/(\delta t_{max})} h_0 (\frac{1}{h'} + \frac{1}{h' - 1} + \ldots + \frac{1}{2} + 1)$$

$$\leq \delta t_{max} e^{1/(\delta t_{max})} h_0 (\frac{1}{h_0} + \frac{1}{h_0 - 1} + \ldots + \frac{1}{2} + 1) = E \in O(h_0 \log h_0).$$

By another form of Chernoff bound [10]

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

we see that

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

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. ∎

4 Experiments

We ran some experiments to get practical experience of the new algorithms, see Tables 1 and 2. In the experiments of Table 1, the number of processors was $p = 1024$, and each result is the average of 250 experiments. In all cases, slackness factors $h = 16, 32, 64, 128, 256$ were tried. It was assumed that the acknowledgement of packets does not need extra time. Furthermore, it was assumed that collisions within processors are avoided by allocating a unique sending

h	Penalty	GGT	CT1.1	CT2.0	GT1.1	GT1.5
16	5.8	5.8	5.7	7.7	5.3	5.6
32	4.8	5.8	4.5	5.5	4.4	4.8
64	4.2	5.7	3.9	4.7	3.9	3.9
128	3.9	5.6	3.6	4.2	3.5	3.6
256	3.9	5.6	3.4	3.9	3.2	3.4

Table 1: Routing cost of the Penalty, GGT, CT, and GT algorithms. In the Penalty protocol, penalty function $f(i) = 1 + i$ was used. In the GGT, $\epsilon = 0.5$, $\alpha = 0.01$ were safe and fast. In the CT1.1 $h_0 = 10, \delta = 1.1, t = 1.1$. CT2.0 is similar otherwise but $t = 2.0$. In the GT1.1 $h_0 = 5, \delta = 1.1, d = 1.1$, and $t_{max} = 2.0$, while in CT2.0 $d = 2.0$. The standard deviation was about 0.5 on top lines and about 0.1 on bottom lines.

moment of time for each packet from the time window $[1 \ldots \delta th]$.

The Table 2 demonstrates the cost that is achieved when an h-relation with $h = \log_2^2 p$ is routed. Remember, again, that if random packets are created continuously, each packet requires expected time e for succeeding. Therefore, the cost e is the best one can hope.

p	CT1.1	CT2.0	GT1.1	GT1.5
128	3.7	4.4	3.7	3.7
256	3.7	4.4	3.7	3.7
512	3.7	4.4	3.6	3.7
1024	3.7	4.3	3.6	3.7
2048	3.7	4.3	3.6	3.7
4096	3.6	4.3	3.6	3.7

Table 2: The cost of routing an h-relation, where $h = \log_2^2 p$. In CT1.1 and CT2.0 $h_0 = \log_2 p$, while in GT1.1 and GT1.5 $h_0 = 0.5 \log_2 p$. In all cases $\delta = 1.1$. In CT1.1 $t = 1.1$ and in CT2.0 $t = 2.0$. In GT1.1 $d = 1.1$, in GT1.5 $d = 1.5$, and in both cases $t_{max} = 2.0$.

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 Penalty algorithm and the GGT algorithm.

In addition to mere numbers, our routing simulator shows the progress of routing graphically, see Figure 1. In CT and GT the throughput is constant, unlike the througput of GGT, which follows a saw blade pattern. The parameters δ, t, and h_0 have an important effect on the shape and the length of the "tail" in Figure 1, i.e. on the routing of the last packets. By some experimentation, or better theoretical reasoning, it may be possible to decrease the cost a little more.

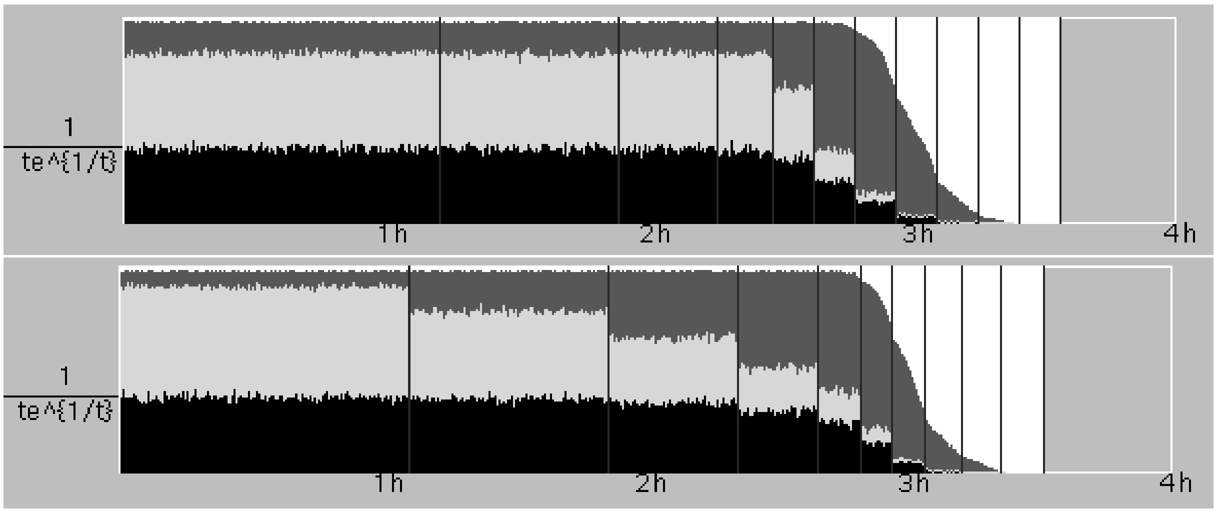

Figure 1: Graphical output of a CT simulation (upper graph) and a GT simulation (lower graph). In both simulations $p = 1024$ and $h = 128$. In CT simulation $h_0 = 16$, $\delta = 1, t = 1.2$. In GT simulation $h_0 = 8, \delta = d = 1.1$. The CT graph is divided in three horizontal bands by ragged borderlines at $1/t = 83\%$ and at $1/te^{1/t} = 36\%$ of all processors. The bottom band represents successful processors, the middle band failed processors, and the top band passive processors. The vertical lines at intervals of δch, $\delta c^2 th$, ..., until $\delta c^i th < h_0 = 16$ ($c = 1 - 1/e^{1/t}$) separate the phases of the algorithm. The total time in picture is $456 = 3.56h$. In GT simulation, the number of passive processors increases due to geometrically increasing thinning. The total time is $450 = 3.52h$.

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

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

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

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