Communication in Parallel Systems

Friedhelm Meyer auf der Heide and Christian Scheideler[⋆]

Department of Mathematics and Computer Science
and Heinz Nixdorf Institute
University of Paderborn
33095 Paderborn, Germany

Abstract. Efficient communication in networks is a prerequisite to exploit the performance of large parallel systems. For this reason much effort has been done in recent years to develop efficient communication mechanisms. In this paper we survey the foundations and recent developments in designing and analyzing efficient packet routing algorithms.

1 Introduction

Communication among the processors of a parallel computer usually requires a large portion of the runtime of a parallel algorithm. These computers are often realized as sparse networks of a large number of processors such that each processor can directly communicate with a few neighbours only. Thus, most of the communication must proceed through intermediate processors. One of the basic problems in this context is to route simultaneously many messages through the network. Telecommunicaton networks, computer networks in companies and universities, or the internet are examples for networks that have to process many communication requests in parallel (e.g., telephone calls, emails, money transfers between banks). In this paper we want to describe recent developments in the field of *universal* routing algorithms, that is, algorithms that can be used in any communication network. Universal algorithms have the advantage that, in addition to providing a unified approach to routing in standard networks, they are ideally suited to routing in irregular networks that are used in wide-area networks and that arise when standard networks develop faults. Furthermore, universal routing algorithms can be used for arbitrary patterns of communication.

In this survey paper, we focus on comparing the routing time needed by distributed routing protocols to the worst case routing time of a best offline routing protocol, the so-called *routing number*. It turns out that online protocols, even deterministic protocols, can get very close to this efficiency bound under various assumptions about the topology, bandwidth, and buffer size of the underlying network.

[⋆] email:{fmadh,chrsch}@uni-paderborn.de, fax: +49-5251-606482, http://www.uni-paderborn.de/cs/{fmadh,chrsch}.html. Supported in part by DFG-Sonderforschungsbereich 376 “Massive Parallelität: Algorithmen, Entwurfsmethoden, Anwendungen”, by DFG Leibniz Grant Me872/6-1, and by EU ESPRIT Long Term Research Project 20244 (ALCOM-IT).

Organization of the Paper

In the following chapter we introduce the basic notation about networks, messages, and protocols for routing. In Chapter 3 we introduce the routing number of a network, and relate it to the dilation and congestion of path systems. Chapter 4 contains an overview of oblivious routing protocols, and Chapter 5 describes efficient adaptive routing protocols.

2 Networks, Messages, Protocols

In this chapter we introduce the basic notions used in routing theory. In particular, we describe a typically used hardware model and message passing model, define the routing problem, and describe different classes of strategies to solve routing problems.

2.1 The Hardware Model

We model the topology of a network as an undirected graph $G = (V, E)$. V represents the computers or processors, and E represents the communication links. We assume the communication links to work bidirectional, that is, each edge represents two *links*, one in each direction. The *bandwidth* of a link is defined as the number of messages it can forward in one time step. Unless explicitly mentioned we assume that the bandwidth is 1. The *size* N of G is defined as the number of nodes G contains.

Each node in V contains an *injection buffer* and a *delivery buffer*. The task of the injection buffer of a node v is to store all messages v wants to send out, and the task of the delivery buffer is to store all messages arriving at v whose destination is v. Further we assume that each link has a *link buffer* (or *buffer* for short). The *size* of a buffer is defined as the maximal number of messages a buffer can store. In the *multi-port* model each link can forward at most one message per step, whereas in the *single-port* model each node can forward at most one message in one step. We only consider the multi-port model in the following, since it has become most common for packet routing in the last years.

Network Topologies

The most commonly used network topologies in the routing literature and in parallel computers are meshes, tori and butterflies. These classes are defined in the following. (For any $k \in \mathbb{N}$, $[k]$ denotes the set $\{0, \ldots, k-1\}$.)

Definition 1 (Butterfly). Let $d \in \mathbb{N}$. The *d-dimensional butterfly* $BF(d)$ is defined as an undirected graph with node set $V = [d+1] \times [2]^d$ and an edge set $E = E_1 \cup E_2$ with

$$E_1 = \{\{(i, \alpha), (i+1, \alpha)\} \mid i \in [d],\ \alpha \in [2]^d\}$$

and

$$E_2 = \{\{(i, \alpha), (i+1, \beta)\} \mid i \in [d],\ \alpha, \beta \in [2]^d,\ \alpha \text{ and } \beta \text{ differ only at the } i\text{th position}\}.$$

The *d-dimensional wrap-around butterfly* W-BF(d) is derived from the $BF(d)$ by identifying level d with level 0.

Definition 2 (Torus, Mesh). Let $m, d \in \mathbb{N}$. The (m,d)-mesh $M(m,d)$ is a graph with node set $V = [m]^d$ and edge set

$$E = \left\{ \{(a_{d-1} \dots a_0), (b_{d-1} \dots b_0)\} \mid a_i, b_i \in [m], \ \sum_{i=0}^{d-1} |a_i - b_i| = 1 \right\} \ .$$

The (m,d)-torus $T(m,d)$ consists of an (m,d)-mesh and additionally the edges

$$\{\{(a_{d-1} \dots a_{i+1} 0 a_{i-1} \dots a_0), (a_{d-1} \dots a_{i+1}(m-1)a_{i-1} \dots a_0)\} \mid i \in [d], \ a_j \in [m]\} \ .$$

$M(m,1)$ is also called *linear array*, $T(m,1)$ *cycle*, and $M(2,d) = T(2,d)$ *d-dimensional hypercube*.

A very general class form the node-symmetric graphs.

Definition 3 (Node-Symmetric Graph). A graph $G = (V,E)$ is called node-symmetric if for any pair of nodes $u, v \in V$ there exists an isomorphism $\varphi : V \to V$ with $\varphi(u) = v$ such that the graph $G_\varphi = (V, E_\varphi)$ with $E_\varphi = \{\{\varphi(x), \varphi(y)\} \mid \{x,y\} \in E\}$ is isomorphic to G.

Intuitively, node-symmetry means that a graph looks the same from any node. Node-symmetric graphs form a very general class and include most of the standard networks such as the d-dimensional torus, the wrap-around butterfly, the hypercube, etc. An important subclass of the node-symmetric graphs are the edge-symmetric graphs.

Definition 4 (Edge-Symmetric Graph). A graph $G = (V,E)$ is called edge-symmetric if for any pair of edges $\{u,v\}, \{u',v'\} \in E$ there exists an isomorphism $\varphi : V \to V$ with $\{\varphi(u), \varphi(v)\} = \{u',v'\}$ such that the graph $G_\varphi = (V, E_\varphi)$ with $E_\varphi = \{\{\varphi(u), \varphi(v)\} \mid \{u,v\} \in E\}$ is isomorphic to G.

The d-dimensional torus, for instance, is edge-symmetric.

Definition 5 (Leveled Graph). A graph $G = (V,E)$ is called *leveled* with *depth D* if the nodes of G can be partitioned into D *levels* $L_0, \dots, L_D$ such that every edge in E connects nodes of consecutive levels. Nodes in level 0 are called *inputs*, and nodes in level D are called *outputs*. If, in addition, $|L_0| = |L_D|$ and L_1 is identified with L_D, then G is called a *wrapped leveled* graph with depth D.

The d-dimensional butterfly, for instance, is a leveled graph with depth d, and the d-dimensional wrap-around butterfly is a wrapped leveled graph with depth d.

2.2 The Routing Problem

Consider an arbitrary network G with node set $[N]$. Let v and w be two nodes in G. We say that a message is *routed* from v to w if it is sent along a path in G from v to w. An instance of the routing problem is defined by a multi-set of source-destination pairs

$$\mathcal{R} = \{(v_1, w_1), \dots, (v_n, w_n)\} \ .$$

Each pair $(v_i, w_i) \in [N]^2$ represents a message that has to be sent from v_i to w_i. The following routing problems are usually studied.

- *Permutation routing*: Let S_N be the set of all permutations $\pi : [N] \to [N]$. Given a permutation $\pi \in S_N$, route a message from node i to node $\pi(i)$ for all $i \in [N]$.
- *h-function routing*: Let $F_{h,N}$ be the set of all h-functions $f : [h] \times [N] \to [N]$. Given an h-function $f \in F_{h,N}$, route a message from node i to nodes $f(1,i), \ldots, f(h,i)$ for all $i \in [N]$. A 1-function is simply called function.
- *h-relation routing*: An h-relation is defined as an h-function $f \in F_{h,N}$ with $|f^{-1}(i)| = h$ for all $i \in [N]$. Given an h-relation $f \in F_{h,N}$, route a message from node i to nodes $f(1,i), \ldots, f(h,i)$ for all $i \in [N]$.
- *Broadcasting*: Given a node $i \in [N]$, route a message from i to all nodes $j \in [N]$.
- *Gossiping*: For every node $i \in [N]$, route a message from i to all nodes $j \in [N]$.

The solution to a routing problem is called *routing protocol*. It determines for each time step, which message to choose from a buffer and which edge to use. Its basic component is the *contention resolution rule* that decides which message wins if more than one message try to use the same link at the same time.

A routing protocol is called *deterministic* if all decisions during the routing are deterministic, and *randomized* otherwise.

2.3 Routing Strategies

We distinguish between two kinds of protocols: *Offline* protocols and *online* protocols. In case of offline protocols we allow the system to compute, for a given routing problem $\mathcal{R}$, a routing schedule for $\mathcal{R}$ before sending out the messages. The time an offline protocol needs is defined as the time needed to route the messages according to the schedule. This implies that the time for computing that schedule does *not* count. In case of online protocols the time for computing a schedule counts.

There are basically two classes of online protocols. In order to send a message from node v to w in G it has to traverse a contiguous sequence of links called *routing path*. Given a source-destination pair a message may either have to traverse a path specified in advance or is able to choose among several alternative paths depending on the source-destination pairs of other messages or other events. The first case is called *oblivious routing* and the second case *adaptive routing*.

Oblivious Routing

In case of oblivious routing the path a message uses only depends on its source and destination. Let $p_{v,w}$ denote the path from v to w in G that has to be taken by every message that wants to travel from v to w. Then $\mathcal{P} = \{p_{v,w} \mid v, w \in V\}$ is called *path system* of *size* $|V|$. In this case any routing problem $\mathcal{R}$ can be defined by specifying a *path collection* $\mathcal{P}_{\mathcal{R}}$ of size $|\mathcal{R}|$ that contains the path $p_{v,w}$ for every pair (v,w) in $\mathcal{R}$. A path collection is called

- *simple* if no path contains the same edge more than once,
- *shortcut-free* if no piece of a path in $\mathcal{P}$ can be shortcut by any combination of other pieces of paths in $\mathcal{P}$,
- *shortest* if all paths are shortest paths in G, and
- *leveled* if the nodes can be arranged in levels such that every edge leads from level i to level $i+1$ for some $i \geq 0$.

The following relationship holds for these types of path collections.

$$\text{leveled} \Rightarrow \text{shortest} \Rightarrow \text{shortcut-free} \Rightarrow \text{simple}$$

Important parameters for the runtime are

- the *size* n of $\mathcal{P}_{\mathcal{R}}$, that is, the number of paths $\mathcal{P}_{\mathcal{R}}$ contains,
- the *congestion* C of $\mathcal{P}_{\mathcal{R}}$, that is, the maximal number of paths in $\mathcal{P}_{\mathcal{R}}$ that contain the same edge in G, and
- the *dilation* D of $\mathcal{P}_{\mathcal{R}}$, that is, the length of a longest path in $\mathcal{P}_{\mathcal{R}}$.

In case that we want to route random functions, we often use the notion of expected congestion that is defined as follows. For an edge e and a path system $\mathcal{P}$, let the expected congestion $\bar{C}_e$ of e be defined as the expected number of messages that traverse e using paths in $\mathcal{P}$ during the routing of a randomly chosen function. The *expected congestion* of $\mathcal{P}$ is then defined as $\bar{C} = \max_{e \in E} \bar{C}_e$.

In case of bounded buffers, *deadlocks* can arise during the routing. Packets are defined to run into a deadlock if they prevent each other from moving forward because of full buffers.

Adaptive Routing

In case of adaptive routing the path a message uses is not predetermined. There are several parameters that are used to measure the performance of adaptive routing protocols: The bisection width, expansion [9], flux [13], or routing number (see Section 3) of a network.

In adaptive protocols that do not restrict the messages to approach their destinations via shortest paths, *livelocks* can happen. Packets run into a livelock if they run infinitely often along the same cycle in the network.

2.4 The Message Passing Model

The most commonly used and well understood routing model in the literature is the *packet routing model*. In this model time is partitioned into synchronous *steps*. One step is defined as the time a message needs to be sent along an edge. (It is usually assumed that every edge needs the same amount of time to forward a packet.) A node must store an entire message before it can forward any part of the message along the next edge on its route. Hence, messages can be viewed as atomic objects called *packets*. A packet has the following format.

message	routing information	source	destination

In a network with N nodes the source and destination need $\lfloor \log N \rfloor + 1$ bits, each. Throughout this paper we restrict the routing information to be very small, namely of length at most $O(\log N)$. It is usually needed to store information about the path a packet has to follow or the priority level of a packet. We assume the messages to have uniform length.

Packet routing algorithms are used on machines such as the NCube, NASA MPP, Intel Hypercube, and Transputer-based machines. Since the packet model does not make assumptions about packet lengths, it is the easiest and therefore the most studied model in the literature.

3 The Routing Number

The aim of this paper is to survey results on relating the routing time needed by online routing protocols to the worst case routing time of a best offline routing algorithm, the so-called *routing number*. This number is defined as follows (see, e.g., [2]):

Consider an arbitrary network G with N nodes and bandwidth one. For a permutation $\pi \in S_N$, let $R(G, \pi)$ be the minimum possible number of steps required to route messages offline in G according to π using the multi-port model with unbounded buffers. Then the *routing number* $R(G)$ of G is defined by

$$R(G) = \max_{\pi \in S_N} R(G, \pi) \ .$$

In case that there is no risk of confusion about the network G we will write R instead of $R(G)$. The routing number has the following nice property.

Theorem 6. *For any network G with routing number R, the average number of steps to route a permutation in G is bounded by $\Theta(R)$.*

Proof. Let $\bar{R} = \frac{1}{|S_N|} \sum_{\pi \in S_N} R(G, \pi)$ denote the average number of steps to route a permutation in G. Consider any fixed permutation π. In order to bound the minimum number of steps to route π we will use a probabilistic argument based on Valiant's trick (see [24]) by first sending the packets to random intermediate destinations before sending them to the destinations prescribed by π. Let X be a random variable denoting the minimum number of steps necessary to route a randomly chosen permutation. According to the Markov Inequality it holds:

$$\text{Prob}(X \geq 3\bar{R}) \leq \tfrac{1}{3} \ .$$

Therefore, for a randomly chosen permutation φ for the intermediate destinations, it holds that with probability at most $\frac{1}{3} + \frac{1}{3} < 1$ the minimum number of steps to route first according to φ and then according to π exceeds $6\bar{R}$. Therefore there exists an offline protocol that routes π in at most $6\bar{R}$ steps. Thus $R \leq 6\bar{R}$, which completes the proof. ∎

Hence asymptotically the routing number is not only an upper bound, but also a lower bound for the average permutation routing time using optimal routing strategies in G.

Note that the routing number might be defined by some protocol which uses specific routing paths tailored to the permutation to be routed. This implies the following result.

Remark 7 *For any network G of size N with routing number R, there exists a collection of simple paths for any permutation routing problem $\pi \in S_N$ with congestion at most R and dilation at most R.*

The question is whether it is possible to find such a path collection in an efficient way for every permutation routing problem. This will be answered by the following two sections. The first section deals with the problem of choosing a path system with low dilation and expected congestion, and the second section shows how to choose path collections out of such a system in a distributed way such that the dilation and congestion bounds in Remark 7 can be reached up to constant factors for any permutation routing problem.

3.1 Relationship to Dilation and Expected Congestion

A necessary prerequisite for efficient oblivious routing is a path system with low dilation and low expected congestion. In this section we construct path systems that have dilation and expected congestion close to the routing number for arbitrary networks. The main result of this section is formulated in the next theorem.

Theorem 8. *For any network G of size N with routing number R there is a simple path system with dilation at most R and expected congestion at most R. Furthermore, for random functions the congestion is bounded by $R + O(\sqrt{\max\{R, \log N\} \cdot \log N})$, w.h.p.*[⋆]

Proof. Let G be a network of size N with routing number R. Then for any permutation $\pi_i : [N] \to [N]$ with $\pi_i(x) = (x + i) \bmod N$ for all $i, x \in [N]$ there is a path collection $\mathcal{P}_i$ along which packets can be routed in at most R time steps. Therefore the congestion and dilation of $\mathcal{P}_i$ is at most R. We then choose $\mathcal{P} = \bigcup_{i=0}^{N-1} \mathcal{P}_i$ to be the path system for G. Clearly, this path system has congestion at most $N \cdot R$ and dilation at most R. It remains to bound the expected congestion and the congestion that holds w.h.p. for routing a random function in G using paths in $\mathcal{P}$.

Consider any edge e in G. Let the random variable X denote the number of paths that cross e and are used by a packet . Since each node in G chooses a destination for its packet uniformly and independently at random, the probability that a path crossing e is used by a packet is $1/N$. Therefore the expected congestion is at most R.

For any node v in G, let the binary random variable X_v be one if and only if the packet with source v contains e in its routing path. Then $X = \sum_{v \in V} X_v$. Since we consider routing a random function, the probabilities of all X_v are independent. As $E(X) \leq R$, the Chernoff bounds (see [7]) therefore yield that

$$\mathrm{Prob}(X \geq (1+\epsilon)R) \leq \begin{cases} e^{-\epsilon^2 R/3} & : \text{ if } 0 \leq \epsilon \leq 1 \\ e^{-\epsilon \cdot R/2} & : \text{ if } \epsilon \geq 2 \end{cases}$$

This probability is polynomially small in N for $\epsilon = O(\max\{\sqrt{\frac{\log N}{R}}, \frac{\log N}{R}\})$ sufficiently large. Hence w.h.p. there exists no edge with a congestion greater than $R + O(\sqrt{\max\{R, \log N\} \cdot \log N})$. ∎

For node-symmetric networks we can strengthen Theorem 8 by showing that the path system may even be assumed to consist of shortest paths. This is important, because some oblivious routing protocols are especially good for shortest path systems, see Chapter 4. Since the diameter of a network is a lower bound for its routing number, we use the diameter instead of the routing number in the following theorem.

Theorem 9. *Let G be a node-symmetric network with N nodes and diameter D. Then there exists a shortest path system that has an expected congestion of $D + O(\sqrt{\frac{D \log N}{N}})$.*

Proof. Consider the problem of gossiping in G. Perform the random experiment of choosing at random shortest path for all source-destionation pairs $(u, w) \in V^2$ independently from all other source-destionation pairs. For every node v in G, let the binary random variable $X_{u,w}^v$ be one if and only if the path chosen from u to w traverses v and $p_{u,w}^v = \mathrm{Prob}(X_{u,w}^v = 1)$. Further let the random variable C_v denote the number of paths traversing v. Then it holds

$$C_v = \sum_{(u,w) \in V^2} X_{u,w}^v \quad .$$

[⋆] By "with high probability" (or w.h.p. for short) we mean a probability of at least $1 - 1/N^k$ for any constant $k > 0$.

Since G is node-symmetric there exists for any fixed node v' in G an automorphism φ that maps v to v'. Thus it holds

$$\begin{aligned} E(C_v) &= \sum_{(u,w)\in V^2} p^v_{u,w} = \sum_{(u,w)\in V^2} p^{\varphi(v)}_{\varphi(u),\varphi(w)} \\ &= \sum_{(u,w)\in V^2} p^{v'}_{u,w} = E(C_{v'}) \end{aligned}$$

Hence, $E(C_v)$ is the same for every node in G, namely at most $N \cdot D$. Since the paths are chosen independently at random, applying Chernoff bounds yields that $C_v = N \cdot D + O(\sqrt{D \cdot N \log N})$, w.h.p. Thus there exists a path system in G with such a congestion. Since, for a randomly chosen function, each of the paths has a probability of $\frac{1}{N}$ to be chosen, Theorem 9 follows. ∎

Theorem 9 implies, for instance, that there exists a shortest path system in a butterfly of size N with expected congestion at most $\log N + 1$. Similarly, the following result holds for edge-symmetric networks.

Theorem 10. *Let G be a edge-symmetric network with N nodes, degree d, and diameter D. Then there exists a shortest path system that has expected congestion $\frac{D}{d} + O(\frac{1}{N}\sqrt{\max\{\frac{N\cdot D}{d}, \log N\} \cdot \log N})$.*

3.2 Valiant's Trick

In Section 3.1 we have seen that for any network with routing number R there is a fixed path system that yields asymptotically optimal parameters C and D for almost all functions. However, Borodin and Hopcroft [4] could prove a very high lower bound for the maximal congestion that can be reached by a permutation routing problem in a network using a fixed path system. Their result has been improved by Kaklamanis *et al.* [8]. Together with an extension by Parberry [18] we obtain the following theorem.

Theorem 11. *Let G be an arbitrary network of size N with degree d, and let $\mathcal{P}$ be an arbitrary path system in G. Let n nodes in G be determined as sources and destinations. Then there is a permutation $\pi \in S_n$ that has a congestion C_π of $\Omega(\frac{n}{d\sqrt{N}})$.*

Hence, if G has constant degree and all nodes in G are source and destination, that is $n = N$, then there exists a permutation with congestion $\Omega(\sqrt{N})$. Therefore the congestion might be much higher than the dilation since networks of bounded degree can have a diameter of $O(\log N)$.

For the d-dimensional hypercube $M(2,d)$ Theorem 11 implies a worst case congestion of $C = \Omega(\sqrt{2^d}/d)$. Kaklamanis *et al.* present a path system in [8] that reaches this bound.

In case that messages have to be sent from the top level to the top level of a d-dimensional wrap-around butterfly according to some arbitrary permutation, Bock [3] presents a path system that reaches the lower bound $C = \Omega(\sqrt{2^d}/d)$.

According to Theorem 11, oblivious routing might perform very poorly for some functions. Thus, in order to get close to the routing number, we have to turn to non-oblivious strategies. A beautiful, simple idea was presented in [24].

Valiant's trick:
Consider routing an arbitrary h-relation. Route the packets first to intermediate destinations according to a randomly chosen h-function, before routing them to their true destinations.

Applying this trick to a fixed path system $\mathcal{P}$ yields the following strategy:

Each node first chooses random intermediate destinations for its packets. Then each packet is first sent to this intermediate destination, and from there to its final destination, both times using the path prescribed in $\mathcal{P}$.

With this strategy, the congestion of *every* permutation routing problem can be brought close to the expected congestion of $\mathcal{P}$, w.h.p. In particular, the following theorem holds.

Theorem 12. *Let $\mathcal{P}$ be an arbitrary path system of size n with dilation D and expected congestion $\bar{C}$. Then every h-relation can be routed along paths in $\mathcal{P}$ using Valiant's trick with dilation at most $2D$ and congestion at most $2h \cdot \bar{C} + O(\sqrt{\max\{h \cdot \bar{C}, \log n\} \cdot \log n})$, w.h.p.*

Proof. Let V be the set of sources in $\mathcal{P}$. Consider an arbitrary edge e in $\mathcal{P}$. For every $v \in V$ and $i \in \{1, \ldots, h\}$, let the binary random variable $X_{v,i}$ be one if and only if the ith packet in v traverses e. Further let the random variable X denote the number of packets that traverse e. Then $X = \sum_{v \in V} \sum_{i=1}^{h} X_{v,i}$. According to the definition of the expected congestion, $E(X) \leq h \cdot \bar{C}$ holds. If we choose a random h-function, then the probablities for all $X_{v,i}$ are independent. Hence the Chernoff bounds yield that

$$\text{Prob}(X \geq (1+\epsilon)h\bar{C}) \leq \begin{cases} e^{-\epsilon^2 h\bar{C}/3} & : \text{ if } 0 \leq \epsilon \leq 1 \\ e^{-\epsilon \cdot h\bar{C}/2} & : \text{ if } \epsilon \geq 2 \end{cases}$$

which is polynomially small in n for sufficiently large $\epsilon = O(\max\{\sqrt{\frac{\log n}{h \cdot \bar{C}}}, \frac{\log n}{h \cdot \bar{C}}\})$. Therefore the collection of paths in $\mathcal{P}$ used for routing packets from their sources to intermediate destinations and from the intermediate destinations to their destinations has a congestion of at most $2h \cdot \bar{C} + O(\sqrt{\max\{h \cdot \bar{C}, \log n\} \cdot \log n})$, w.h.p., if the intermediate destinations are chosen according to a random h-function. Further, the dilation of the resulting paths is at most $2D$. ∎

Theorems 8 and 12 yield the following result.

Corollary 13. *For any network G of size N with routing number R there exists a simple path system $\mathcal{P}$, such that routing any h-relation along paths in $\mathcal{P}$ using Valiant's trick has dilation at most $2R$ and congestion $2h \cdot R + O(\sqrt{\max\{h \cdot R, \log n\} \cdot \log n})$, w.h.p.*

This result implies that if there is an oblivious protocol that can route packets along an arbitrary simple path collection with dilation D and congestion C in time $O(D + C)$ then a combination of the path selection strategy in Corollary 13 and this protocol would yield a routing protocol that reaches the optimal routing performance of permutation routing in arbitrary networks. Unfortunately, no oblivious protocol is known so far that reaches this bound for arbitrary simple path collections. However, the following result shown in [11, 12] lets us hope that such a protocol may exist.

Theorem 14. *For any simple path collection $\mathcal{P}$ with congestion C and dilation D there exists an offline protocol that finishes routing in $\mathcal{P}$ in $O(C + D)$ steps, using only constant size buffers.*

In the next chapter we summarize what kind of online oblivious protocols have already been found.

4 Oblivious Routing Protocols

In the previous chapter we have seen that oblivious routing has the potential to achieve an efficiency that is close to the routing number, because the dilation and expected congestion are essentially of the same order of magnitude as the routing number. Hence what we need are oblivious routing protocols with runtime close to O(dilation+congestion). In this chapter we present some universal oblivious routing protocols that (nearly) reach this time bound for different classes of path collections.

4.1 The Random Rank Protocol

The *random rank protocol* has its origin in a paper by Aleliunas [1] and Upfal [23], and has been extended to constant size buffers in [19] (see also [9] and [10]). It routes packets along an arbitrary leveled path collection of size n with congestion C and depth D in $O(C + D + \log n)$ steps, w.h.p., using edge buffers of size at least one. In the following we only describe the protocol for unbounded buffers. (See [17] for an easy proof of the runtime bound.)

At the beginning, every packet p gets a random rank denoted by rank(p) that is stored in its routing information. We require rank(p) to be chosen uniformly and independent from the choices of the other packets from some fixed range $[K]$ ($K = O(C + D + \log n)$ is chosen sufficiently large). Additionally, each packet stores an identification number id(p) $\in [n]$ in its routing information that is different from all identification numbers of the other packets. The random rank protocol uses the following contention resolution rule.

> **Priority rule:**
> It two or more packets contend to use the same link at the same time then the one with minimal rank is chosen.

If two packets have the same rank then, in order to break ties, the one with the lowest id wins. The protocol then works as follows in each time step

> For each link with nonempty buffer, select a packet according to the priority rule and send it along that link.

Consider an arbitrary leveled network with N input and N output nodes. Let the routing number be defined as the worst case time needed by the best offline protocol to send packets from the inputs to the outputs according to an arbitrary permutation. Then the random rank protocol and Corollary 13 together yield the following result.

Theorem 15. *Any wrapped leveled network with N inputs, N outputs, and routing number R can route any h-relation from the inputs to the outputs in time $O(h \cdot R + \log N)$, w.h.p., using constant size buffers.*

4.2 The Growing Rank Protocol

Now we present a protocol that routes packets along an arbitrary shortcut-free path collection of size n with congestion C and dilation D in $O(C+D+\log n)$ steps, w.h.p., using buffers of size C. It is called *growing-rank protocol* [15] and works as follows.

Initially, each packet is assigned an integer rank chosen randomly, independently, and uniformly from $[K]$ ($K = O(C + D + \log n)$ is chosen sufficiently large). For each step, the protocol works as follows.

For each link with nonempty buffer,

- choose a packet p according to the priority rule,
- increase the rank of p by K/D, and
- move p forward along the link.

This protocol together with Valiant's trick yields the following result.

Theorem 16. *For any shortcut-free path system of size n with dilation D and expected congestion $\bar{C}$, any permutation can be routed using the growing rank protocol in time $O(\bar{C} + D + \log n)$, w.h.p.*

This theorem applied to node-symmetric networks yields the following result together with Theorem 9.

Corollary 17. *Let G be a node-symmetric network of size N with diameter D. Then the growing rank protocol routes packets according to an arbitrary h-relation in time $O(h \cdot D + \log N)$, w.h.p.*

This result is optimal for permutation routing in arbitrary node-symmetric networks with diameter $D = \Omega(\log N)$, since Theorem 9 together with Theorem 12 and Theorem 14 implies that the routing number of any node-symmetric network with diameter $D = \Omega(\log N)$ is bounded by $\Theta(D)$.

Unfortunately, it has been shown in [16] that the growing rank protocol can not be efficiently applied to arbitrary simple path collections. Furthermore, it is not clear yet whether efficient shurtcut-free path systems exist for any network. In case of shortest path systems, however, networks exist such that any shortest path system has a much higher expected congestion then the best simple path system.

4.3 The Trial-and-Failure Protocol

In the following we present a protocol that routes packets along an arbitrary simple path collection of size n with link bandwidth $B \leq \log n$, congestion C, and dilation D in time

$$O\left(\frac{C \cdot D^{1/B} + D \log n}{B}\right)$$

w.h.p., without buffering. This protocol is called *trial-and-failure protocol* and has been presented in [6]. The idea of the trial-and-failure protocol is that once a packet leaves its source it has to move along the edges of its path without waiting until it reaches its destination. If too many packets want to use the same link at the same time then some are discarded (and therefore have to be rerouted).

Initially, each packet p_i chooses uniformly and independently from the other packets a random rank $r_i \in [K]$ ($K = O(C \cdot D^{-1+1/B} + \log n)$ is chosen sufficiently large) and a random delay $d_i \in [D]$. Additionally, p_i stores an identification number $\mathrm{id}(p_i) \in [n]$ in its routing information that is different from all identification numbers of the other packets. Let us define the following contention resolution rule.

> ***B*-priority rule:**
> If more than B packets attempt to use the same link during the same time step, then those B with lowest rank win.

If two or more packets have the same rank then, in order to break ties, the ones with the lowest id's win. Then the protocol works as follows.

```
repeat
  • forward pass: Each active packet $p_i$ waits for $d_i$ steps. Then it is routed
    along its path, obeying the $B$-priority rule.
  • backward pass: For each packet that reached its destination during
    the forward pass, an acknowledgment is sent back to the source. Upon
    receipt of the acknowledgment, the source declares the packet inactive.
until no packet is active
```

Clearly, the forward pass needs $2D$ steps to be sure that every packet that has not been discarded during the routing reaches its destination. In the backward pass, the forward pass is run in reverse order. Therefore, no collisions can occur in the backward pass, and $2D$ steps suffice to send all acknowledgments back.

The trial-and-failure protocol has several applications. If we simulate link bandwidth by buffers we arrive at the following result.

Corollary 18. *Given any simple path collection with buffer size B, congestion C, and dilation D, the trial-and-failure protocol requires $O(C \cdot D^{1/B} + D \log n)$ time to route all packets, w.h.p.*

This result is optimal if $C \geq D \log n$ and $B \geq \log D$. Together with Valiant's trick it yields the following result.

Corollary 19. *Let G be an arbitrary network of size N with buffer size B and routing number R. Then the trial-and-failure protocol routes packets according to an arbitrary h-relation in time $O((h \cdot R^{1/B} + \log n)R)$, w.h.p.*

In the case of shortest path collections, the trial-and-failure protocol can be combined with the growing rank protocol to yield the following tradeoff between routing time and buffer size for oblivious routing.

Theorem 20. *Given any shortest path collection of size n with buffer size B, congestion C, and dilation D there exists an on-line routing protocol that requires*

$$O\left(\frac{C \cdot D^{1/B} + \log n}{B} \cdot (C + D + \log n)\right)$$

time to route all packets, w.h.p.

Together with Theorem 9 and Valiant's trick we get the following result.

Corollary 21. *Let G be an arbitrary node-symmetric network of size N with buffer size B and diameter D. Then the trial-and-failure protocol routes packets according to an arbitrary h-relation in time*

$$O\left(\frac{h \cdot D^{1+1/B} + \log n}{B} \cdot (h \cdot D + \log N)\right) ,$$

w.h.p.

4.4 The Duplication Protocol

In [6] a protocol is presented that routes packets along an arbitrary simple path collection of size n with congestion C, dilation D, and bandwidth $\Theta(\log(C \cdot D)/\log\log(C \cdot D))$ in

$$O\left(D \log\log n + C + \frac{\log n \cdot \log\log n}{\log\log(C \cdot D)}\right)$$

time steps, w.h.p., without buffering. It is called *duplication protocol*. The structure of the duplication protocol is similar to that of the trial-and-failure protocol with the difference that each new trial the number of copies sent out for a packet is duplicated. This significantly increases the chance that finally one of the copies will be able to reach the destination. The protocol uses the following rule in case of collisions between packets:

> ***B*-collision rule:**
> If more than B packets attempt to use the same link during the same time step, then *all* of them are discarded.

If we simulate bandwidth by buffer size, we arrive at the following result.

Corollary 22. *Given any simple path collection $\mathcal{P}$ of size n with congestion C and dilation D, there exists a routing protocol that requires*

$$O\left(\left(D \log\log n + C + \frac{\log n \cdot \log\log n}{\log\log(C \cdot D)}\right) \cdot \frac{\log(C \cdot D)}{\log\log(C \cdot D)}\right)$$

time, w.h.p., and uses buffers of size $O(\log(C \cdot D)/\log\log(C \cdot D))$.

Together with Corollary 13 this protocol yields the following result.

Corollary 23. *For any network G of size N with routing number $R = \Omega(\log N)$ and buffer size $O(\log R/\log\log R)$, the duplication protocol can be used to route any permutation in time*

$$O\left(\frac{\log R \cdot \log\log N}{\log\log R} \cdot R\right) ,$$

w.h.p.

4.5 The Protocol by Rabani and Tardos

In [20], Rabani and Tardos present a protocol that routes packets along an arbitrary simple path system with congestion C and dilation D in

$$O(C) + (\log^* n)^{O(\log^* n)} D + poly(\log n)$$

time, w.h.p., using buffers of size C. Together with Corollary 13 their protocol yields the following result.

Corollary 24. *For any network G of size N with routing number $R \geq \log^k N$, $k \geq 0$ sufficiently large, there exists an oblivious routing protocol for routing any h-relation in time $(O(h) + (\log^* N)^{O(\log^* N)})R$, w.h.p.*

4.6 Open Problems

In the following we state some important open questions. Is there an online protocol that can route packets along an arbitrary simple (not necessarily shortcut-free) path system of size N with congestion C and dilation D in time $O(C + D + \log N)$? If this is true then, in case of unbounded buffers, randomized oblivious routing can asymptotically reach the performance of offline protocols. How do bounded buffers influence the runtime of oblivious routing strategies ? Are ghost packets (see [10]) necessary for efficient routing with bounded buffers in leveled networks ?

5 Adaptive Routing Protocols

Adaptive protocols are very appealing compared to oblivious protocols, since they allow a parallel system to react more flexibly in case of faulty or overloaded communication links or processors. Another motivation for adaptive routing is that bounded buffers are difficult to handle for oblivious routing strategies. Usually, the only way to avoid deadlocks using oblivious routing strategies is simply to delete the packets in case of full buffers and restart them again from the source (see,i.e., the trial-and-failure protocol). However, in case of unbounded buffers we know from Corollary 13 and Theorem 14 that randomized oblivious routing strategies have at least a chance to be as efficient as randomized adaptive routing strategies. This, of course, crucially depends on whether it is possible to design an efficient protocol for routing packets along an arbitrary simple path collection.

In the deterministic case, however, adaptive routing protocols are usually far superior against oblivious routing protocols, since the worst case congestion for routing an arbitrary permutation using a fixed path system can be fairly large (see Theorem 11). But even if the congestion is small, deterministic oblivious routing strategies might still perform very poorly as will be shown in the following theorem.

We investigate the behavior of non-predictive routing protocols in which all scheduling decisions have to be independent from the future routing paths of the packets. Note that the growing rank protocol is not deterministic and hence not nonpredictive. However, for any fixed setting of the initial ranks it is nonpredictive. The same holds for the random rank protocol and its extensions [21, 9, 10]. The following example shows that all these protocols perform poorly in a deterministic setting even on leveled networks. The proof can be found in [16].

Theorem 25. *Suppose we are given any deterministic non-predictive routing protocol $\mathcal{Q}$ for routing on the D-dimensional butterfly. Then, for any C, there is a routing problem with congestion C for which $\mathcal{Q}$ takes time $\Omega(C \cdot D)$.*

Therefore in case of deterministic routing, efficient adaptive routing protocols are highly needed. In the following section we present a network for which an efficient deterministic adaptive routing protocol is already known. This result will be used in Section 5.2 to develop, for any network, a deterministic adaptive routing protocol that has a performance close to the routing number of that network.

5.1 Deterministic Routing in the Multibutterfly

In this section describe a network in which deterministic routing can be done efficiently. It is called the (elementary) *s-ary multibutterfly.*

The basic building block of the s-ary multibutterfly is an *s-ary m-splitter* (see [5]).

The s-ary m-splitter (or (s, m)-splitter) is a bipartite graph with m input nodes and m output nodes. In this graph the output nodes are partitioned into $\sqrt{s}$ output sets, each with $m/\sqrt{s}$ nodes. Every input node has $\frac{\sqrt{s}}{2}$ edges to each of the $\sqrt{s}$ output sets. The edges connecting the input set to each of the output sets define an expander graph with properties described in [5].

The *s-ary d-dimensional multibutterfly* (s, d)-MBF has d levels. The vertices at level $0 \leq i \leq d-1$ are partitioned into $\sqrt{s}^{i}$ sets of $m_i = \sqrt{s}^{d-i}$ consecutive nodes. Each of these sets in level i is an input set of an s-ary m_i-splitter. The output sets of that splitter are $\sqrt{s}$ sets of size m_{i+1} in level $i+1$. Thus each node in the (s, d)-MBF is the endpoint of at most $2 \cdot \sqrt{s}(\sqrt{s}/2) = s$ edges.

For this network the following result can be shown. It's proof can be found in [5].

Theorem 26. *For sufficiently large s, the s-ary multibutterfly of size N can route any permutation from the top level to the bottom level in time $O(\log_s N)$.*

This result has recently been improved in [14] for an extended version of the (s, d)-MBF that has a degree of $O(s)$.

Theorem 27. *Given an extended s-ary multibutterfly of size N with $s \geq 2$, $s \cdot N$ packets, s per node, can be routed deterministically according to some arbitrary s-relation in time $O(\log_s N)$.*

5.2 Routing via Simulation

In this section we present an efficient deterministic adaptive routing protocol for arbitrary networks, as it is described in [14]. Let H be an arbitrary network of size N with routing number R. The idea is that, in order to route packets in H deterministically according to an arbitrary permutation, a suitably chosen multibutterfly is embedded in H, and H simulates the routing steps of the multibutterfly.

Consider first the more general problem of simulating one routing step of an arbitrary network G by a network H. We start with describing how to embed G in H. In order to simplify the construction, let H be a network of size N, and G be a network of size M with at most $N/2$ edges. Let $d_1, \ldots, d_M$ be the degree sequence of G, i.e., d_i is the degree of node i in G. Then $\sum_{i=1}^{M} d_i \leq N$. Our strategy is to partition the nodes of H into clusters $C_1, \ldots, C_M$ such that for all $i \in \{1, \ldots, M\}$ cluster C_i consists of d_i nodes simulating the d_i endpoints of node i in G. For this we choose an arbitrary spanning tree T in H. Let r be an arbitrary node in T. We mark the nodes in T with numbers in $\{1, \ldots, M\}$ starting with r by calling Mark$(1, true, r)$:

```
**Algorithm Mark($i, m, v$):**

$i$: number of nodes already marked
$m$: boolean variable indicating whether father has been marked
$v$: actual node to be considered

**if** $m = false$ **then**
    mark $v$ with the number $\ell$ obeying $\sum_{j=1}^{\ell-1} d_j < i \leq \sum_{j=1}^{\ell} d_j$
    i=i+1
    for every son $w$ of $v$: call Mark$(i, true, w)$
**else**
    for every son $w$ of $v$: call Mark$(i, false, w)$
    mark $v$ with the number $\ell$ obeying $\sum_{j=1}^{\ell-1} d_j < i \leq \sum_{j=1}^{\ell} d_j$
    i=i+1
**return** the value of $i$
```

Basically, the algorithm ensures that on a pass downwards through the tree only every second node is marked such that afterwards on a pass upwards the other half of the nodes can be marked. Hence it is easy to see that the following two results hold.

(a) cluster i has diameter at most $3d_i$ for all $i \in \{1, \ldots, M\}$, and
(b) the maximal number of clusters that share the same link is constant.

Let the nodes of each cluster be connected by an Euler tour along edges in T. (An Euler tour in a tree is defined as a directed cycle that uses any edge in T in any direction at most once.) Because of (a) this tour can have a length of at most $6d_i$.

Further we want to simulate every edge in G by a path in H that connects the nodes simulating its endpoints. Let R be the routing number of H. Since our clustering allows the endpoints of edges in G to be distributed in H such that every node in H has to simulate at most one endpoint, there is a path collection in H for simulating the edges in G with congestion at most R and dilation at most R.

Consider now the problem of simulating an arbitrary routing step in G. Clearly, any routing step can be extended to the situation that along every edge in G a packet has to be sent. This event can be simulated in the following way by H.

- Moving the packets to the nodes simulating the endpoints of the edges they want to use in G: This can be done by sending the packets along an Euler tour connecting the nodes of the respective cluster in T. Because of (b) this can be coordinated among the clusters deterministically in time $O(\max_i d_i)$ using only constant size buffers.
- Moving the packets along an edge in G: This can be done by sending the packets along the paths simulating edges in G. Since these paths have congestion at most R and dilation at most R, this can be done deterministically in time $O(R)$ using only constant size buffers (see Theorem 14).

If we restrict the maximum degree in G to be $O(R)$, we get the following result.

Theorem 28. *Any network H of size N with routing number R can simulate any routing step in a network G with degree $O(R)$ and $O(N)$ edges in $O(R)$ steps using only constant size buffers.*

Theorem 28 and Theorem 27 together yield the following result.

Theorem 29. *Let H be an arbitrary network of size N with routing number R. Then there is a deterministic online protocol that routes any permutation in time $O(\log_R N \cdot R)$, using constant size buffers.*

Theorem 29 has the following implications.

Corollary 30. *For any network H of size N with routing number $R = \Omega(N^\epsilon)$ for some constant $\epsilon > 0$ there exists a deterministic routing strategy that routes any permutation in H in $O(R)$ steps.*

$R = \Omega(N^\epsilon)$ holds, for instance, for all networks with diameter $\Omega(N^\epsilon)$ or bisection width $O(N^{1-\epsilon})$. According to [22], every N-node graph of genus g and maximal degree d has bisection width $O(\sqrt{gdN})$. Thus the following result is true.

Corollary 31. *For any network H of size N with genus g and degree d such that $g \cdot d = O(N^{1-\epsilon})$, $\epsilon > 0$ constant, there exists a deterministic routing strategy that routes any permutation in H in $O(R)$ steps.*

This result implies that for any planar network with degree $O(N^{1-\epsilon})$ there is an asymptotically optimal deterministic permutation routing strategy.

5.3 Open Problems

The results above only yield asymptotically optimal routing strategies if the routing number is large enough. How efficient is deterministic online routing for networks with small routing number ? No non-trivial lower bounds are known so far.

References

1. R. Aleliunas. Randomized Parallel Communication. In *Proc. of the ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing*, pp. 60-72, 1982.
2. N. Alon, F.R.K. Chung, R.L. Graham. Routing Permutations on Graphs via Matchings. *SIAM J. Discrete Math.* 7(3), pp. 513-530, 1994.
3. S. Bock. Optimales Wormhole Routing im hochdimensionalen Torus. Diploma thesis, Paderborn University, March 1996.
4. A. Borodin, J.E. Hopcroft. Routing, merging, and sorting on parallel models of computation. *Journal of Computer and System Sciences* 30, pp. 130-145, 1985.
5. A. Borodin, P. Raghavan, B. Schieber, E. Upfal. How much can hardware help routing? In *Proc. of the 25th Ann. ACM Symposium on Theory of Computing*, pp. 573-582, 1993.
6. R. Cypher, F. Meyer auf der Heide, C. Scheideler, B. Vöcking. Universal Algorithms for Store-and-Forward and Wormhole Routing. In *28th Ann. ACM Symp. on Theory of Computing*, pp. 356-365, 1996.

7. T. Hagerup, C. Rüb. A Guided Tour of Chernoff Bounds. *Information Processing Letters* 33, pp. 305-308, 1989/90.
8. C. Kaklamanis, D. Krizanc, T. Tsantilas. Tigth Bounds for Oblivious Routing in the Hypercube. *Mathematical Systems Theory* 24, pp. 223-232, 1991.
9. F.T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes.* Morgan Kaufmann, San Mateo, CA, 1992.
10. F.T. Leighton, B.M. Maggs, A.G. Ranade, S.B. Rao. Randomized Routing and Sorting on Fixed-Connection Networks. *Journal of Algorithms* 17, pp. 157-205, 1994.
11. F.T. Leighton, B.M. Maggs, S.B. Rao. Universal Packet Routing Algorithms. In *Proc. of the 29th Ann. Symp. on Foudations of Computer Science*, pp. 256-271, 1988.
12. F.T. Leighton, B.M. Maggs, S.B. Rao. Packet Routing and Job-Shop Scheduling in O(Congestion + Dilation) Steps. *Combinatorica* 14, pp. 167-186, 1994.
13. T. Leighton, S. Rao. An Approximate Max-Flow Min-Cut Theorem for Uniform Multicommodity Flow Problems with Applications to Approximation Algorithms. In *Proc. of the 29th Ann. IEEE Symp. on Foundations of Computer Science*, pp. 422-431, 1988.
14. F. Meyer auf der Heide, C. Scheideler. Deterministic Routing with Bounded Buffers: Turning Offline into Online Protocols. To appear at *Proc. of the 37th Ann. IEEE Symp. on Foundations of Computer Science*, 1996.
15. F. Meyer auf der Heide, B. Vöcking. A Packet Routing Protocol for Arbitrary Networks. In *12th Symp. on Theoretical Aspects of Computer Science* (STACS 95), pp. 291-302, 1995.
16. F. Meyer auf der Heide, B. Vöcking. Universal Store-and-Forward Routing. Technical Report, Paderborn University, 1996.
17. F. Meyer auf der Heide , R. Wanka. Kommunikation in parallelen Rechnernetzen (in German). In *Highlights aus der Informatik*, I. Wegener (editor), Springer Verlag, pp. 177-198, 1996.
18. I. Parberry. An Optimal Time Bound for Oblivious Routing. *Algorithmica* 5, pp. 243-250, 1990.
19. N. Pippenger. Parallel Communication with Limited Buffers. In *Proc. of the 25th IEEE Symp. on Foundations of Computer Science*, pp. 127-136, 1984.
20. Y. Rabani, É. Tardos. Distributed Packet Switching in Arbitrary Networks. In *28th Ann. ACM Symp. on Theory of Computing*, pp. 366-375, 1996.
21. A.G. Ranade. How to Emulate Shared Memory. *Journal of Computer and System Sciences* 42, pp. 307-326, 1991.
22. O. Sýkora, I. Vrťo. Edge Seperators for Graphs of Bounded Genus with Applications. *Theoretical Computer Science* 112, pp. 419-429, 1993.
23. E. Upfal. Efficient Schemes for Parallel Communication. In *Proc. of the ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing*, pp. 241-250, 1982.
24. L.G. Valiant. A Scheme for Fast Parallel Communication. *SIAM Journal of Computing* 11(2), pp. 350-361, 1982.

This article was processed using the LaTeX macro package with LLNCS style