Locally Efficient On-Line Strategies for Routing Packets along Fixed Paths*

(Preliminary Version)

Petra Berenbrink and Christian Scheideler†
Department of Mathematics and Computer Science
Paderborn University, 33095 Paderborn, Germany
email: {pebe,chrsch}@uni-paderborn.de

Abstract

Most of the work done in the area of static routing concentrates on minimizing the runtime of the whole schedule rather than minimizing the runtime of individual packets, using global parameters such as the congestion and dilation of a path collection. In this paper, we study the problem of minimizing the routing time of individual packets, using local parameters. In fact, we present the first (up to a $\log\log$ factor) optimal, truly on-line routing protocols for the following problems:

- Packet switching: Assume that a fixed collection of paths is given. For every path p in this collection, let c_p denote the maximum number of paths that share an edge with p, and let d_p denote the length of p. Find a schedule (that does *not* require to know c_p and d_p) such that the routing time of a packet following a path p merely depends on c_p and d_p.
- Virtual circuit switching: Assume that a fixed set of sessions is given. For every session i, packets are injected at a rate r_i to follow a predetermined path of length d_i. If the sum of rates of the sessions using an edge is sufficiently small, find a schedule such that the routing time of a packet from session i merely depends on r_i and d_i.

We also consider dynamic versions of the two problems together with an injection model that covers all stochastic and adversarial injection models known to us.

* Authors supported in part the DFG-Sonderforschungsbereich 376 "Massive Parallelität: Algorithmen, Entwurfsmethoden, Anwendungen" and by the EU ESPRIT Research Project 20244 (ALCOM-IT).

† Part of the research was done while staying at the Weizmann Insitute, the stay was supported by a scholarship from the MINERVA foundation.

1 Introduction

For many applications such as video conferencing, scientific visualization, medical imaging, and high-speed execution of distributed algorithms across large networks (such as the Internet), it is of fundamental importance to be able to establish connections that guarantee a certain quality of service. In today's communication networks the guarantee is usually done by establishing so-called *virtual circuits*. In this communication mode, a user requests a particular share of the available bandwidth and injects a stream of packets along one particular path at the agreed-upon rate. An important consequence of the user's predictability is that the network can, in return, guarantee the user an *end-to-end delay*, *i. e.*, an upper bound on the time that any packet takes to move from its source to its destination. In order to guarantee this delay, the network must determine how to schedule the packets that contend for the same edge simultaneously. It is known that this can be done in an optimal way if we have a *static* situation, *i. e.*, no old request is cancelled and no new request arrives, and a polynomial number of time steps are allowed to compute the respective schedule [1]. However, no on-line schedule with (asymptotically) optimal end-to-end delay is known for this situation, and no schedule with (asymptotically) optimal end-to-end delay is known at all for the case that we are in a *dynamic* scenario, that is, requests can be cancelled and new requests arrive.

The reserving of bandwidth is very useful for connections of medium or long duration, since the time for establishing a virtual circuit is usually dominated by the time for sending the information across that line. However, there are communication requests (such as sending short e-mails on the Internet, or exchanging small data items between processors in parallel systems), for which it is far too expensive to establish a line (both for the bandwidth of the system and the end-to-end delay) before sending the information. Such scenarios can be modeled with the *packet switching* approach, in which individual packets follow paths that are not known to the system in advance. In almost all papers on this subject, the performance of a protocol is expressed in terms of global parameters such as the *congestion* (maximum number of paths that cross an edge) and *dilation* (the length of the longest path used by a packet). However, it would be much more desirable to be able to send packets along their paths with an end-to-end delay that only depends on the communication traffic along their own path, and not on the situation in the whole network.

Such *locally efficient* protocols could be used, for instance, to exploit topological locality in parallel systems in a sense that processors which are closely together, according to the topology of the network, can exchange data much faster than processors that are far apart. Topological locality can be due to a communication sensitive mapping of processes (*task mapping*) or global variables (*data management*) to processors. Several papers have already dealt with the question of how to exploit locality for data management in distributed systems [4, 5, 14, 15]. However, none of them considers the problem of how to actually send messages through the system. They only deal with minimizing the distance between a datum and its requesting process, or how to select paths to minimize the congestion at the links caused by the requests. In order to exploit topological locality, it is important to have a routing algorithm that is able to prefer packets accordingly.

1.1 Models and problems

In this paper, we consider arbitrary network topologies modeled as undirected graphs with n nodes. The nodes represent switches, and the edges represent bidirectional communication links with buffers for outgoing packets on either side. Every node contains an *injection buffer* and a *delivery buffer*. Initially, each packet is stored in the injection buffer of its source. Once a packet reaches its destination, it is stored in the destination's delivery buffer.

Routing is performed in a synchronous "store and forward" manner. In every step, each edge can be crossed by at most one packet in each direction. Since a packet has to store its destination, it has to consist of at least $\log n$ bits. Thus, we assume in the following that all packets are of size $O(\log n)$, *i. e.*, $O(\log n)$ bits can cross an edge in one time step. (This will be important for our algorithms, since apart from the packets we will also

send special control packets that can carry a certain amount of information.)

We present routing protocols in which the switches *locally* decide which packets they move forward in each step, *i. e.*, a decision only depends on the routing information carried by the packets and on the local history of the execution. These algorithms are called *local control* or *on-line* algorithms.

1.1.1 The injection model

In a dynamic setting, paths are continuously injected into the system. An injection model is called *bounded* if there are $T, B \in \mathbb{N}$, such that for any time interval I of length T and edge e, the number of paths generated during I that cross e is at most B (in each direction), either w.h.p.* (stochastic models), or with certainty (adversarial models). The *injection rate* λ of the system is defined as B/T. A routing protocol is called *stable* if the number of packets stored at the edges does not increase unboundedly in the course of time. Clearly, a necessary condition for a protocol to be stable in our model is that of $\lambda \leq 1$.

1.1.2 The packet switching model

In general, a packet routing scheme consists of two (not necessarily independent) parts: the first part is responsible for selecting a path for each packet, and the second part is responsible for scheduling the packets along their chosen paths. We assume that some suitable strategy for the path selection is given. Hence, in the following we only concentrate on the question of how to schedule the packets along their fixed paths. We use the following models:

Static packet switching: Assume that a fixed collection of paths is given. For every path p in this collection, let c_p denote the maximum number of paths that share any edge e of p, and let d_p denote the length of p.

Dynamic packet switching: Assume that paths are continuously injected into the system according to some arbitrary injection model. Along each path, one packet has to be send. Once a packet traversed its path, this path is removed from the system. For each path p, we define c_p as the maximum number of paths crossing any edge of p that were in the system when p was injected. d_p denotes the length of p.

Obviously, for both models a protocol is worst case optimal if it guarantees a runtime of $O(c_p + d_p)$ for every packet following a path p. (However, this is not an absolute lower bound for every individual packet since we can always give a high priority to a specific packet such that its end-to-end delay is d_p.) Often it is assumed that, in addition to local information, on-line algorithms know the size, congestion and dilation of a routing problem. We are more strict here, because we assume that nothing about the routing problem is known in advance. (We only have to know the size of the network which is independent of the routing problem.) This assumption is important, since in general it might be very difficult, if not impossible for the processors, to have sufficient knowledge about the congestion at other parts of the system. For instance, this might be very difficult if many parallel programs are executed simultaneously by the same distributed system. Also, processes and shared objects might be moved while the application is running.

1.1.3 The virtual circuit switching model

In general, a virtual circuit switching scheme consists of two parts: The first part is responsible for selecting a path and a rate for each new session request. The second part is responsible for the scheduling of the injected packets along the paths prescribed by their sessions. We assume that some suitable scheme (commonly called *admission control scheme*) is given that selects a path and a rate for each session. This scheme ensures that the

*Throughout the paper, the terms "*with high probability*" and "*w.h.p.*" mean "with probability at least $1 - n^{-\alpha}$" where $\alpha > 0$ is an arbitrary constant.

sum of the rates of the sessions using an edge is below a certain threshold. Hence, we only concentrate on the question of how to schedule the packets along their fixed paths. We use the following models:

Static virtual circuit switching: Assume that a fixed collection of sessions is given. For every session i, packets are injected at a rate r_i in order to follow a predetermined path of length d_i (*i. e.*, every $1/r_i$ steps a session i packet is injected).

Dynamic virtual circuit switching: Assume a dynamically changing set of sessions. For every session i, packets are injected at a rate r_i (*i. e.*, every $1/r_i$ steps a session i packet is injected) in order to follow a predetermined path of length d_i. After a session with rate r and path p of length d injected its last packet, its rate is reserved for $\delta(r, d, n)$ more steps at the edges of p. ($\delta(r, d, n)$ can be seen as a *deadline* given to the last packet to reach its destination.) Afterwards, its rate can be used by other, newly injected sessions.

Obviously, a protocol that guarantees an end-to-end delay of $O(1/r_i + d_i)$ for each session-i packet is worst case optimal (see also [1]). However, similar to the previous section, it is not an absolute lower bound for every individual packet.

1.2 Previous results

1.2.1 Packet switching

In a pioneering paper, Leighton, Maggs and Rao [12] showed that there is an off-line schedule for any simple (*i. e.*, loop-free) path collection with dilation D and congestion C that sends all packets along their paths (one packet per path) in time $O(C + D)$. Since then, many on-line algorithms have been presented for various classes of path collections (see, e. g., [13, 16, 8, 21, 18]). To give two examples, in [8] Cypher *et al.* present an on-line, buffer-less algorithm that works for any simple path collection of size n. Their algorithm delivers all packets to their destinations in time $O\left(D \cdot \log\log n + C + \frac{\log n \cdot \log\log n}{\log\log(C \cdot D)}\right)$ w.h.p., provided that the communication links have a bandwidth of $\Theta\left(\log(C \cdot D)/\log\log(C \cdot D)\right)$. In [18] Ostrovsky and Rabani present an on-line protocol with runtime $O(C + D + \log^{1+\epsilon} n)$, w.h.p., for any constant $\epsilon > 0$.

If the local congestion and dilation of the paths is known in advance for every path, the protocols presented in the papers cited above can be transformed into locally efficient on-line protocols. That is, we can replace C and D by the local parameters c_p and d_p in the runtime bound for each packet P. (Simply partition the packets into sets with $\max\{c_p, d_p\} \in [2^i, 2^{i+1} - 1]$, and route these sets one after the other.) However, if the local congestion and dilation of the paths is *not* known in advance, it is not known how to transform the existing protocols into locally efficient routing protocols.

Theoretical results about dynamic routing protocols for interconnection networks are relatively new. Several papers have already dealt with dynamic routing protocols for butterflies, meshes and hypercubes (see, e. g. [17, 9, 10, 7]). The first analysis of a universal protocol can be found in [22]. Among other results, Scheideler and Vöcking present in this paper a dynamic protocol for arbitrary shortest path collections which is stable up to a constant injection rate under a stochastic injection model. Furthermore, they show that, in the steady state, their protocol delivers every packet that has to travel a distance of d in time $O(d)$, w.h.p. In [3] Broder *et al.* present conditions that are sufficient for the stability of dynamic packet routing algorithms. Their approach reduces the problem of steady state analysis to the easier question of static routing.

Whereas the results above are based on a stochastic injection model, there have also been some results based on a model called *adversarial queuing theory*. In this model, it is possible to have an adversary that can select the paths of the packets within a certain limit. This approach was introduced by Borodin *et al.* in [6]. Further results have been presented by Andrews *et al.* [2]. Among other things, they show that there are simple greedy protocols that are stable for every network, but other commonly-used protocols (such as FIFO) are not stable for every network.

1.2.2 Virtual circuit switching

The problem of virtual circuit switching is well studied. Until recently, the best known delay bound was $O(d_i/r_i)$ for packets of session i (see [19]). This was subsequently improved in [21, 18]. Ostrovsky and Rabani construct in [18] an on-line algorithm that routes every packet to its destination with probability $1-p$ in $O(1/r + D + \log^{1+\epsilon}(p^{-1}))$ steps for any constant $\epsilon > 0$, where r is the minimum injection rate and D is the length of the longest path. In [1], Andrews *et al.* show the existence of an asymptotically optimal schedule for the static virtual circuit switching model that achieves a delay bound of $O(1/r_i + d_i)$ with only constant queue size at the switches. They also present a simple on-line algorithm that delivers every session-i packet to its destination within $O(1/r_i + d_i \cdot \log(m/r_{\min}))$ steps, w.h.p., where m is the number of edges of the network and $r_{\min}$ is the minimum session rate. Furthermore, they extend their results to bursty traffic sessions.

1.3 Our results

1.3.1 Packet switching

Within the static packet switching model, we present a protocol called the *FLP (first local packet switching) protocol* that achieves the following result:

Theorem 1.1 *For any simple path collection, the FLP protocol routes every packet along its path p in expected time $O(c_p + d_p \cdot log(c_p + d_p))$. Furthermore, an additive delay of t to this runtime only occurs with a probability of at most $2^{-\Theta(\sqrt[3]{t/\log t})}$.*

The main idea behind this protocol is a strategy that allows the packets to find a suitable lower bound for their local congestion that holds with exponentially high probability. The main problem in the analysis of the protocol is to get rid of certain correlations among the packets. For this, we additionally send special *ghost packets*.

Furthermore, we present a protocol called the *SLP (static local packet switching) protocol* that achieves a faster runtime than the first one in case that c_p or d_p is larger than $\log n$.

Theorem 1.2 *For any simple path collection, the SLP protocol routes every packet along its path p in expected time $O(c_p + d_p \cdot \min\{\log(c_p + d_p), \log\log n\})$. Furthermore, an additive delay of t to this runtime only occurs with a probability of at most $2^{-\Theta(\sqrt[3]{t/\log t})}$ if $\sqrt[3]{t/\log t} \leq \log n$, and $n^{-\Theta(\log t)}$ otherwise.*

Apart from static protocols, we also present a dynamic protocol for shortcut-free paths, called *DLP (dynamic local packet switching) protocol*. A path collection is called *short-cut free* if no part of a path can be used as a short-cut of a part of another path. (This holds, for instance, if no two paths meet more than once.)

Theorem 1.3 *For any injection model restricted to short-cut free paths, the DLP protocol routes every packet along its path p in expected time $O(c_p + d_p \cdot \log\log n)$. Furthermore, an additive delay of t to this runtime only occurs with a probability of at most $2^{-\Theta(\sqrt{t/\log t})}$ if $\sqrt{t/\log t} \leq \log n$, and $n^{-\Theta(\log t)}$ otherwise. Moreover, the protocol is stable up to a constant injection rate for any bounded injection model.*

The advantage of our algorithm over previously presented algorithms is that it does not only predict results for the steady state (as usually done under stochastic injection models) or the worst case (as done under adversarial models). It can give the runtime of a packet in terms of *local* parameters at *any* time point of the dynamic routing process for *any* injection model.

1.3.2 Virtual circuit switching

Within the virtual circuit switching model we prove the following result for a protocol named *DLC (dynamic local circuit switching) protocol*.

Theorem 1.4 *For any static or dynamically changing set of sessions with a deadline of $\delta(r,d,n) = 1/r + \Theta(d \log\log\log n)$ using arbitrary simple paths, the DLC protocol routes every session-i packet in expected time at most $1/r_i + O(d_i \log\log\log n)$. Furthermore, an additive delay of t to this runtime only occurs with a probability of at most $t^{-\alpha}$ for any constant α, and an additive increase of $poly(\log n)$ only occurs with a probability of at most $n^{-\Theta(1)}$. Moreover, the protocol is stable as long as the sum of the session rates of each edge does not exceed some constant.*

This is a much better time bound for the expected end-to-end delay than has been obtained before. The protocol can be applied to bursty sessions where the packets of one session may arrive in batches and many other injection models with a constant injection rate, yielding similar performance guarantees. Furthermore, our protocol is more general than the results in [22], because they require the paths to be shortest paths and that the generation of a message and its routing path has to be independent of the generations of other messages and paths and independent of other time steps. Moreover, in contrast to [21, 18], our protocol gives flexible probabilistic guarantees and works with local parameters.

1.4 Probabilistic tools

We will frequently use Chernoff-Hoeffding bounds (or Chernoff bounds in short) in our proofs. These bounds are defined as follows.

Lemma 1.5 (Chernoff-Hoeffding) *Let $X_1,\dots,X_n$ be independent binary random variables, and let $X = \sum_{i=1}^{n} X_i$. For any $\mu \geq \mathbf{E}[X]$ and $\epsilon > 0$ it holds that*

$$\Pr[X \geq (1+\epsilon)\mathbf{E}[X]] \leq \left(\frac{\mathrm{e}^{\epsilon}}{(1+\epsilon)^{1+\epsilon}}\right)^{\mu} .$$

This can be simplified to

$$\Pr[X \geq (1+\epsilon)\mathbf{E}[X]] \leq \begin{cases} \mathrm{e}^{-\epsilon^2\mu/3} & : \quad 0 \leq \epsilon \leq 1 \\ \mathrm{e}^{-\epsilon\mu/3} & : \quad \epsilon > 1 \end{cases} .$$

Apart from independent random variables, we use the following property.

Definition 1.6 *A set of binary random variables $X_1,\dots,X_n$ is called* negatively correlated *if, for any $i \in \{1,\dots,n\}$ and subset $S \subseteq \{1,\dots,n\} \setminus \{i\}$,*

$$\Pr[X_i = 1 \mid \bigwedge_{j \in S} X_j = 1] \leq \Pr[X_i = 1] .$$

It has been shown in [20] that Lemma 1.5 also holds for negatively correlated binary random variables.

2 Strategies for the Packet Switching Model

In this section, we prove the theorems in Section 1.3.1.

2.1 Static packet switching

2.1.1 The FLP protocol

Let us consider all paths to be partitioned into sets $S_1, S_2, \ldots$, with S_i containing all paths p with $\max\{c_p/i, d_p\} \in [2^{i-1}, 2^i - 1]$. Our protocol works in *rounds*, starting with round 1. For every $r \geq 1$, the rth round has the task to route all packets with paths in S_r to their destinations. In particular, we show that

(1) round r takes $O(r \cdot 2^r)$ time steps, and

(2) any participating packet having a path in $S_{r'}$ with $r' \leq r$ successfully reaches its destination with a probability of at least $1 - 2^{-\Theta(\sqrt[3]{2^r})}$.

This means that

- the expected time to route a packet with path in S_r is bounded by

$$\begin{aligned} \sum_{i=1}^{r} O(i \cdot 2^i) + \sum_{i=r+1}^{\infty} O(i \cdot 2^i) \cdot 2^{-\Theta(\sqrt[3]{2^i})} &= O(r \cdot 2^r) = O\left(\max\left\{\frac{c_p}{r},\ d_p\right\} \cdot r\right) \\ &= O(c_p + d_p \cdot \log(c_p + d_p)) \,, \end{aligned}$$

  and

- a deviation of an additive t from this bound occurs with a probability of at most $2^{-\Theta(\sqrt[3]{t/\log t})}$.

Thus it remains to prove items (1) and (2) above. Let us consider a fixed round $r \geq 1$. Let $T = \sqrt[3]{2^r}$. Every edge is assumed to have a bandwidth b of $\beta \log T$, *i. e.*, every edge can forward b packets at every time step. (Obviously, each of these time steps can be simulated in $\beta \log T$ time steps by edges with bandwidth 1.) Apart from the usual packets (called *real packets* in the following), so-called *ghost packets* are generated at each node. In fact, each node generates $T^3 \cdot \log T$ ghost packets for each of its links. Each ghost packet is assigned a delay $\delta \in [5T^3]$, chosen independently and uniformly at random (i.u.r.). Ghost packets, however, never really leave a node. As we will see later, their only purpose is to keep down certain correlations among the real packets. Furthermore, each real packet P is followed by a so-called *counting packet* P'. The packet P' counts the time steps in which at least b packets want to cross an edge of the path of P (this information has to be collected by the nodes of the network). Let us now describe how round r works. In the following, $[x]$ for some integer $x \in \mathbb{N}$ denotes the set $\{1, \ldots, x\}$, and $[x, y]$ for some integers $x < y$ denotes the set $\{x, \ldots, y\}$.

Round r of the FLP protocol:

- **Phase 1:** Every packet that has not yet reached its destination, chooses i.u.r. an initial delay $\delta \in [4T^3]$ with $T = \sqrt[3]{2^r}$. A real packet that is assigned a delay of δ waits in its current edge buffer for δ time steps and then tries to move on without waiting until it reaches its destination or traversed T^3 edges.

  If, together with the ghost packets, more than b packets want to cross an edge e at the same time, then all of the involved real packets will remain in the outgoing buffer of e until the end of the phase. Such packets are called *unsuccessful*.

- **Phase 2:** For every real packet P, we send a counting packet P' along P's path. In this phase, we do not use ghost packets any more. P' uses the same initial delay as P and then it moves on without waiting until it reaches the current edge buffer of P. Obviously, this is always possible for P', as phase 2 is a playback of phase 1.

  If a counting packet P' counts more than $(\beta T)^2$ events, where at least b packets (*not* counting P) try to cross one of the edges along the path of the corresponding real packet P at the same time, and P was

unsuccessful in the first phase, P will be *dumped*, which means that it stays in its current edge buffer for the rest of the round.

- **Phase 3:** The third phase consists of $2T$ *passes*. For every pass, every unsuccessful but not dumped real packet chooses i.u.r. an initial delay from a range of $[(\beta T)^2]$. Then it attempts to move along the next T^2 edges on its path. Once a packet reaches its destination or traversed T^3 edges in the whole round, it stops and is declared *successful*.

In order to show that the above strategy fulfills the requirements stated in (1) and (2) above, we prove several lemmas. As a worst case assumption, we assume that the local congestion of a path p is still c_p during every round, *i. e.*, every path that shares an edge with p is still participating in that round. Furthermore, we assume that T is beyond some sufficiently large constant. Since below this constant all rounds take constant time, this does not affect our runtime bounds asymptotically.

We frequently use the following notation: Given a packet P, a sub-path $q = e_1 \dots e_\ell$ of P's path from edge e_1 to e_ℓ is called its *active segment* if q represents the path which P attempts to traverse during this round. A *site* is an ordered pair (e, t) with e being an edge and t being a time step. A packet *aims for* a site (e, t) if it selects a random delay so that it intends to cross e at time t. Furthermore, a packet is said to *participate* in a site (e, t) if it indeed crosses it during the execution of the algorithm. Define a *run* of P to be the set of sites which P will aim for for some fixed delay δ, *i. e.*, a run consists of sites $(e_1, \delta), (e_2, \delta + 1), \dots, (e_\ell, \delta + \ell - 1)$. A site is called *$d$-site* (resp. *$> d$-site*, *$\geq d$-site*) if exactly d (resp. more than d or at least d) packets aim for that site.

Lemma 2.1 *Consider any fixed $r' \leq r$. A packet with a path in $S_{r'}$ is dumped with probability at most e^{-cT} for any constant c depending on b.*

Proof. First, we give a high-level description of the proof. The counting packet of a packet P counts the number of $\geq b$-sites in order to decide if P gets dumped or not. Thus, we have to bound the number of $\geq b$-sites along the edges of P's active segment q. Since the length of q is at most T^3 and the packets can aim for at most $5T^3$ different sites per edge, there are at most $5T^6$ sites that have to be considered. The proof is divided into two parts.

In the first part we show that the number of $> b$-sites along q can be upper bounded by T with a probability of at least $1 - \mathrm{e}^{-cT \cdot \log T}$. Because packets involved in a $> b$-site remain in the corresponding edge buffer, the sites can be regarded as independent of each other. Thus, the probability bound can be obtained by the use of Chernoff bounds.

In the second part we bound the number of b-sites along q. This is much more difficult because now there are dependencies among sites at different edges of q. For example, the event that a packet traverses a b-site at time t may significantly influence the event whether there is a b-site at time $t + 1$ at its next edge. In the case that the paths are not short-cut free, there might be additional dependencies among the sites. In order to cope with the dependencies, we need the ghost packets. These packets allow us to show that the probability of a packet to be involved in more than $\beta^{3/2}T$ different b-sites is at most $\mathrm{e}^{-c'T}$. Therefore, we can bound the number of b-sites along q by $\beta^2 T \cdot (T - 1)$ with a probability of at least $1 - \mathrm{e}^{-c''T}$. Thus, the number of $\geq b$-sites along the edges of q can be upper bounded by $(\beta T)^2$ with a probability of at least $1 - \mathrm{e}^{-cT}$. We now give a detailed proof.

Let P be a packet with active segment q in $S_{r'}$, $r' \leq r$. First, we upper bound the number of $> b$-sites on the active segment of P. Consider marking any m sites along the edges of q. There are at most T^3 edges in q and for each of these edges there are $4T^3 + T^3 = 5T^3$ possible time steps at which real packets might arrive. Hence, there are at most $\binom{5T^6}{m}$ possibilities of marking m sites.

Suppose the m marked sites to be numbered from 1 to m. Let the random variable X_i denote the number of packets *participating* in site i (*i. e.*, that reach this site in Phase 1), and let $X = \sum_{i=1}^{m} X_i$. Since $r' \leq r$, there are at most $r \cdot 2^r = 3T^3 \log T$ real packets that cross an edge of q. Together with the $T^3 \log T$ ghost packets we

therefore have a total of at most $4T^3 \log T$ packets that cross an edge of q. Since each of these packets chooses a random delay from a range of size at least $4T^3$, the expected number of packets which aim for any one site is at most $\log T$. Therefore, $\mathbf{E}[X] \leq m \log T$. Since a packet can only participate in one $>b$-site, we have: If all m marked sites are $>b$-sites, then $X > b \cdot m$ and the random variables X_i can be viewed as sums of negatively correlated, binary random variables. Thus, in order to bound the number of $>b$-sites, we can apply the Chernoff bounds to obtain

$$\begin{aligned} \Pr[\geq m \text{ many } >b\text{-sites}] &\leq \Pr[X \geq \beta m \log T \mid X_1, \ldots, X_m \text{ negatively correlated}] \\ &\leq \left(\frac{\mathrm{e}^{\beta-1}}{\beta^\beta}\right)^{m \log T} . \end{aligned}$$

Let $m = T$. Then we have, for $\beta = \max[5 + c, \mathrm{e}^2]$,

$$\begin{aligned} \binom{5T^6}{m} \left(\frac{\mathrm{e}^{\beta-1}}{\beta^\beta}\right)^{m \log T} &\leq \left(\frac{5\mathrm{e}T^6}{m}\right)^m \cdot \mathrm{e}^{-\beta m \log T} \leq \mathrm{e}^{m(5 \ln T + \ln(5\mathrm{e})) - \beta m \log T} \\ &\leq \mathrm{e}^{(5-\beta) m \log T} = \mathrm{e}^{-cT \log T} \end{aligned}$$

if T is sufficiently large. Thus, the number of $>b$-sites along q can be upper bounded by T with a probability of at least $1 - \mathrm{e}^{-cT \log T}$. In order to prove Lemma 2.1 it remains to bound the number of b-sites along q (note that a packet counts the number of $\geq b$-sites in order to decide if it gets dumped or not!). This is more difficult than bounding the number of $>b$-sites, because now there may be dependencies causing positive correlations among the sites of q. In order to cope with these dependencies, we need the ghost packets. These packets allow us to show the following proposition.

Proposition 2.2 *Consider any fixed $r' \leq r$. For any constant $\gamma > 0$ there is a constant $\beta > 0$ for b so that a packet with path in $S_{r'}$ participates in at most $\beta^{3/2} T$ b-sites with probability at most $\mathrm{e}^{-\gamma T}$.*

Proof. Consider a fixed packet P with active segment $q = e_1 \ldots e_\ell$ in $S_{r'}$, $r' \leq r$. In the following, let the random variable X_i denote the number of packets participating in the site of P's run at edge e_i *including* P, $i \in [\ell]$, and let $X = \sum_{i=1}^{\ell} X_i$. In order to bound the number of b-sites in which P participates, we prove the following claim.

Claim 2.3 *For any $i \in [\ell]$ and set $J \subseteq [\ell] \setminus \{i\}$ and any $c_j \in \mathbb{N}_0$ with $j \in J$ it holds*

$$\Pr[X_i > b \mid (\textstyle\bigwedge_{j \in J} X_j = c_j) \ \wedge \ X_i \geq b] \geq \frac{1}{6\beta} \cdot \Pr[X_i = b \mid (\textstyle\bigwedge_{j \in J} X_j = c_j) \ \wedge \ X_i \geq b] .$$

Proof. W.l.o.g., we assume that $J = \{1, \ldots, k\}$ and $i = k+1$ for some k. Let M be the set of all possibilities to select packets such that $X_j = c_j$ for all $j \in [k]$. Each of these possibilities may be represented by a *configuration* $C = (C_1, \ldots, C_k)$, where C_j is the set containing those c_j packets that participate in site j. Let E_C be the event that configuration C is true. Then

$$\Pr[\textstyle\bigwedge_{j \in [k]} X_j = c_j] = \Pr[\bigcup_{C \in M} E_C] .$$

Now, let the random variable Z_r denote the number of real packets that participate site $k+1$, and let Z_g denote the number of ghost packets that aim for (resp. participate in) site $k+1$. Given a fixed Z_r, the probability that exactly $b - Z_r$ ghost packets aim for site $k+1$ is either 0 if $Z_r > b$ or

$$\binom{T^3 \log T}{b - Z_r} \left(\frac{1}{5T^3}\right)^{b - Z_r} \left(1 - \frac{1}{5T^3}\right)^{T^3 \log T - (b - Z_r)} .$$

Thus, for any fixed configuration C,

$$\Pr[X_{k+1} > b \mid E_C \ \wedge \ Z_r > b \ \wedge \ X_{k+1} \geq b] \geq \Pr[X_{k+1} = b \mid E_C \ \wedge \ Z_r > b \ \wedge \ X_{k+1} \geq b] = 0$$

and otherwise

$$\begin{aligned}
&\frac{\Pr[X_{k+1} > b \mid E_C \ \wedge \ Z_r \leq b \ \wedge \ X_{k+1} \geq b]}{\Pr[X_{k+1} = b \mid E_C \ \wedge \ Z_r \leq b \ \wedge \ X_{k+1} \geq b]} \\
&= \frac{\Pr[X_{k+1} > b \mid E_C \ \wedge \ Z_r \leq b]}{\Pr[X_{k+1} = b \mid E_C \ \wedge \ Z_r \leq b]} \\
&\geq \frac{\Pr[Z_g = b+1-Z_r \mid E_C \ \wedge \ Z_r \leq b]}{\Pr[Z_g = b-Z_r \mid E_C \ \wedge \ Z_r \leq b]} \\
&= \frac{\sum_{i=0}^{b} \Pr[Z_r = i \mid E_C] \cdot \Pr[Z_g = b+1-Z_r \mid E_C \ \wedge \ Z_r = i]}{\sum_{i=0}^{b} \Pr[Z_r = i \mid E_C] \cdot \Pr[Z_g = b-Z_r \mid E_C \ \wedge \ Z_r = i]} \\
&= \frac{\sum_{i=0}^{b} \Pr[Z_r = i \mid E_C] \cdot \binom{T^3 \log T}{b+1-i} \left(\frac{1}{5T^3}\right)^{b+1-i} \left(1-\frac{1}{5T^3}\right)^{T^3 \log T-(b+1-i)}}{\sum_{i=0}^{b} \Pr[Z_r = i \mid E_C] \cdot \binom{T^3 \log T}{b-i} \left(\frac{1}{5T^3}\right)^{b-i} \left(1-\frac{1}{5T^3}\right)^{T^3 \log T-(b-i)}} \\
&\geq \frac{T^3 \log T - b}{b+1} \cdot \frac{1/(5T^3)}{1-1/(5T^3)} \geq \frac{\log T}{6b} = \frac{1}{6\beta}
\end{aligned}$$

if T is sufficiently large. Therefore,

$$\begin{aligned}
&\frac{\Pr[X_{k+1} > b \mid (\bigwedge_{j\in[k]} X_j = c_j) \ \wedge \ X_{k+1} \geq b]}{\Pr[X_{k+1} = b \mid (\bigwedge_{j\in[k]} X_j = c_j) \ \wedge \ X_{k+1} \geq b]} \\
&= \frac{\sum_{C\in M} \Pr[X_{k+1} > b \mid E_C \ \wedge \ X_{k+1} \geq b]}{\sum_{C\in M} \Pr[X_{k+1} = b \mid E_C \ \wedge \ X_{k+1} \geq b]} \\
&\geq \frac{\frac{1}{6\beta} \sum_{C\in M} \Pr[E_C] \cdot \Pr[X_{k+1} = b \mid E_C]}{\sum_{C\in M} \Pr[E_C] \cdot \Pr[X_{k+1} = b \mid E_C]} \\
&\geq \frac{1}{6\beta} \, .
\end{aligned}$$

□

The above claim implies that for each additional $\geq b$-site in which P is involved, the probability that this is a $>b$-site (and therefore P stops) is at least a constant times the probability that this is a b-site (and therefore P can continue). Let A_b (resp. $A_{\geq b}$, $A_{>b}$) be the event that some given site i is a b-site (resp. $\geq b$-site, $>b$-site), assuming any event $\bigwedge_{j\in J} X_j = c_j$ with $J \subseteq [\ell] \setminus \{i\}$. Then

$$\begin{aligned}
\Pr[A_b \mid A_{\geq b}] &= \frac{\Pr[A_b]}{\Pr[A_{\geq b}]} = \left(\frac{\Pr[A_b]}{\Pr[A_b] + \Pr[A_{>b}]}\right) \\
&\leq \left(\frac{\Pr[A_b]}{\Pr[A_b] + \frac{1}{6\beta}\Pr[A_b]}\right) = \left(\frac{1}{1+\frac{1}{6\beta}}\right) = \left(1 - \frac{1}{6\beta+1}\right) \leq \mathrm{e}^{-\frac{1}{6\beta+1}} \, .
\end{aligned}$$

Set $p = \mathrm{e}^{-\frac{1}{6\beta+1}}$. Now we are ready to bound the number of b-sites along the active segment q which P will see on its run. Note that the number of b-sites in which P participates is at least the number of $\geq b$-sites minus one, because P stops as soon as it participates in a $>b$-site. Let $D = \mathcal{P}([\ell])$ represent the set of all possible distributions of $<b$ and $\geq b$-sites along P's run in the following way: For any $U \in D$, let the event E_U denote

the set of all outcomes such that U contains exactly those sites that are $\geq b$-sites. Let the binary random variable Y_i be one if and only if site i is a b-site and there is no site $j < i$ that is a $> b$-site. Furthermore, let $Y = \sum_{i=1}^{\ell} Y_i$. Then it holds:

$$\begin{aligned}
\Pr[Y \geq k] &\leq \sum_{U \in D} \Pr[E_U] \cdot \Pr[\bigwedge_{\text{first } k \text{ numbers } i \in U} Y_i = 1 \mid E_U] \\
&= \sum_{U \in D} \Pr[E_U] \cdot \prod_{\text{first } k \text{ numbers } i \in U} \Pr[Y_i = 1 \mid E_U \wedge (\bigwedge_{j \in U:\, j < i} Y_j = 1)] \\
&\leq \sum_{U \in D} \Pr[E_U] \cdot p^k = p^k
\end{aligned}$$

Hence, for $\beta^{3/2}/(6\beta + 1) \geq \gamma$ it holds

$$\Pr[Y \geq \beta^{3/2} T] \leq p^{\beta^{3/2} T} = e^{-\frac{\beta^{3/2}}{6\beta+1} T} \leq \mathrm{e}^{-\gamma T} \ .$$

This completes the proof of Proposition 2.2. □

We are now ready to prove Lemma 2.1. First, we define a set of binary random variables $X_1, \ldots, X_m$. Each X_i corresponds to one of the at most $5T^6$ possible sites along the edges of q (*i. e.*, $m \leq 5T^6$). X_i is equal to 1 if and only if at least $b/2$ packets *aim* for site i. Clearly,

$$\Pr[X_i = 1] \leq \binom{4T^3 \log T}{b/2} \left(\frac{1}{4T^3}\right)^{b/2} \leq \left(\frac{2\mathrm{e}}{\beta}\right)^{b/2} \ .$$

Let $X = \sum_{i=1}^{m} X_i$. Then $\mathbf{E}[X] \leq m \cdot (2\mathrm{e}/\beta)^{b/2} \leq 1$ if $\beta \geq 4\mathrm{e}$ and T is sufficiently large. Let the random variable X' denote the maximum number of sites along q for each of which a set of at least $b/2$ packets aiming for it can be marked that is disjoint to all other sets of marked packets. Clearly, $X' \leq X$. On the other hand, if there are at least k many b-sites and no packet participates in more than d of them, then there are at least k/d of these sites for each of which a set of $b/2$ packets can be marked that is disjoint to all other sets of marked packets. Assume that this is not the case, and let $k' < k/d$ be the maximum number of b-sites with the property above. The corresponding $k' \cdot b/2$ marked packets can be used to construct a participation by at least $b/2$ packets for at most $k' \cdot b/2 \cdot d/(b/2) = k' \cdot d < k$ sites. Thus, there must be at least one additional b-site that has at least $b/2$ packets that have not been marked by the k' selected sites, which contradicts the assumption.

Hence, to summarize our observations, if there are at least k many b-sites along q and none of the participating packets participates in more than d of them, then $X' \geq k/d$. Furthermore, the corresponding X_i's are negatively correlated, as the sets of marked packets are disjoint. Since $\mathbf{E}[X'] \leq 1$ if β and T are sufficiently large, it follows from the Chernoff bounds that

$$\Pr[X' \geq k/d] \leq \left(\frac{\mathrm{e}}{k/d}\right)^{k/d} \ .$$

If we set $k = \beta^2 T(T-1)$ and $d = \beta^{3/2} T$, we obtain

$$\Pr[X' \geq \beta^{1/2}(T-1)] \leq \left(\frac{\mathrm{e}}{T}\right)^{\beta^{1/2}(T-1)} \leq \mathrm{e}^{-\gamma T}$$

if T and β are sufficiently large.

According to Proposition 2.2, the probability that some fixed packet participates in more than $\beta^{3/2} T$ many b-sites is at most $\mathrm{e}^{-\gamma' T}$ for any constant γ' depending on β. Thus, the probability that there are at least $\beta^{3/2} T \cdot$

$\beta^{1/2}(T-1) = \beta^2 T(T-1)$ many b-sites along the edges of q can be made as small as $\mathrm{e}^{-\gamma'' T}$ for any constant $\gamma'' > 0$.

Set $c = \gamma'' - 1$ and assume T to be sufficiently large. Then, together with the bound on the number of $> b$-sites, we obtain a probability of at most e^{-cT} for any constant c (depending on β) that packet P has more than $\beta^2 T(T-1) + T \leq (\beta T)^2$ many $\geq b$-sites along its active segment. □

Lemma 2.4 *Consider any active segment q in $S_{r'}$ with $r' \leq r$. It holds for any constant c depending on b that, with probability at least $1 - \mathrm{e}^{-cT^2 \cdot \log T}$, every edge of q is contained in at most $(\beta T)^2 \cdot \log T$ active segments of packets that were unsuccessful in Phase 1 but have not been dumped.*

Proof. Consider some fixed edge e of q. First, we bound the expected number of active segments of unsuccessful, non-dumped packets containing e. For this, consider some fixed packet P. Let A_P be the event that it fails without being dumped, that is, from the start of its active segment to the place where it fails the other packets form at most $(\beta T)^2$ many $\geq b$-sites. Let D denote the set of all possible distributions of the packets among the sites of P, and let E_d be the event that a given distribution $d \in D$ is true. Then

$$\Pr[A_P] = \sum_{d \in D} \Pr[E_d] \cdot \Pr[A_P \mid E_d] \leq \sum_{d \in D} \Pr[E_d] \cdot \frac{\beta^2}{4T} = \frac{\beta^2}{4T} \,,$$

since the probability that P aims for any of the $(\beta T)^2$ first $\geq b$-sites along the path of P given by d is at most $(\beta T)^2/(4T^3)$. As the edge e chosen above belongs to a path in $S_{r'}$ for some $r' \leq r$, e is contained in at most $T^3 \log T$ active segments. Hence, the expected number of unsuccessful, non-dumped packets containing e in their active segment is at most $\frac{1}{4}(\beta T)^2 \log T$.

Consider now any set of packets $P_1, \ldots, P_k$ crossing e. Set $p = \beta^2/(4T)$. Let D denote the set of all possible distributions of the packets, not containing any of the packets $P_1, \ldots, P_k$, among all the sites in the active segments of $P_1, \ldots, P_k$, and let E_d be the event that a given distribution $d \in D$ is true. Furthermore, let D_i denote the set of all possible distributions of all the packets, excluding P_i, among the sites along the active segment of P_i. Since the packets choose their delays independently at random,

$$\begin{aligned}
&\Pr[A_{P_1} \cap \ldots \cap A_{P_k}] \\
&= \sum_{d \in D} \Pr[E_d] \cdot \Pr[A_{P_1} \cap \ldots \cap A_{P_k} \mid E_d] \\
&= \sum_{d \in D} \Pr[E_d] \prod_{i=1}^{k} \Pr[A_{P_i} \mid A_{P_1} \cap \ldots \cap A_{P_{i-1}} \cap E_d] \\
&= \sum_{d \in D} \Pr[E_d] \prod_{i=1}^{k} \left(\sum_{d' \in D_i} \Pr[E_{d'} \mid A_{P_1} \cap \ldots \cap A_{P_{i-1}} \cap E_d] \cdot \Pr[A_{P_i} \mid E_{d'}] \right) \\
&\leq \sum_{d \in D} \Pr[E_d] \prod_{i=1}^{k} \left(\sum_{d' \in D_i} \Pr[E_{d'} \mid A_{P_1} \cap \ldots \cap A_{P_{i-1}} \cap E_d] \cdot p \right) = p^k \,.
\end{aligned}$$

Hence, for an upper bound on the number of unsuccessful, non-dumped packets the events A_{P_i} can be treated as if independent with a probability of p to be true. Thus, using the Chernoff bounds, the probability that segment q has an edge that is contained in at most $(\beta T)^2 \log T$ active segments of unsuccessful, non-dumped packets, where $\beta^2/3 \geq c+1$, is at most

$$T^3 \cdot \mathrm{e}^{-((\beta T)^2 \log T)/3} \leq \mathrm{e}^{3 \ln T - (c+1) T^2 \log T} \leq \mathrm{e}^{-cT^2 \log T}$$

if T is sufficiently large. □

Lemma 2.5 *Consider any $r' \leq r$ and packet P with a path in $S_{r'}$, that was unsuccessful in Phase 1 but was not dumped. If the congestion bound in Lemma 2.4 is true for P's path then it holds: The probability that P is unsuccessful in Phase 3 is at most $\mathrm{e}^{-cT\cdot\log T}$ for any constant $c > 0$ depending on b.*

Proof. Let β be defined as in the proof of Lemma 2.1. Since the packets choose a random delay out of a range of $[(\beta T)^2]$, and at most $(\beta T)^2 \log T$ active segments cross any edge, the expected number of packets that aim for one site is at most $\log T$. Hence, together with the Chernoff bounds, the probability that P is not successful in one sub-pass is at most

$$T^2 \cdot \left(\frac{\mathrm{e}^{\beta-1}}{\beta^\beta}\right)^{\log T} \leq T^2 \cdot \mathrm{e}^{-\beta \log T} \leq \mathrm{e}^{-(\beta-2)\log T}$$

for $\beta \geq \mathrm{e}^2$. Thus, for $\beta \geq \max\{c+4, \mathrm{e}^2\}$ the probability that P fails in at least T of $2T$ sub-passes is at most

$$\binom{2T}{T} \cdot \mathrm{e}^{-(\beta-2)T\log T} \leq \mathrm{e}^{2T-(\beta-2)T\log T} \leq \mathrm{e}^{-cT\log T} .$$

□

Hence, the probability that some fixed packet with path in $S_{r'}$ ($r' \leq r$) fails in round r is at most

$$\underbrace{\mathrm{e}^{-cT}}_{\text{Lemma 2.1}} + \underbrace{\mathrm{e}^{-cT^2\cdot\log T}}_{\text{Lemma 2.4}} + \underbrace{\mathrm{e}^{-cT\cdot\log T}}_{\text{Lemma 2.5}} \leq 2^{-cT} ,$$

which completes the proof of item (2) above. Since β is a constant, Phase 1, 2 and 3 each take $O(T^3 \log T) = O(2^r \cdot r)$ time steps. From this item (1) follows, which completes the analysis of the protocol.

2.1.2 The SLP protocol

In this section we transform the FLP protocol into a protocol called SLP with a packet delay of $O(c_p + d_p \cdot \min\{\log(c_p + d_p), \log\log n\})$ steps. As before, we consider all paths to be partitioned into sets $S_1, S_2, \ldots$. For $i \leq 3\log\log n$, S_i contains all paths p with $\max\{c_p/i, d_p\} \in [2^{i-1}, 2^i - 1]$. For $i > 3 \cdot \log\log n$, S_i contains all paths p with $\max\{\frac{c_p}{3\log\log n}, d_p\} \in [2^{i-1}, 2^i - 1]$. Again, our protocol works in *rounds*. For every $r \geq 1$, round r is responsible for routing all packets with paths in S_r to their destinations.

Rounds 1 to $r' = 3 \cdot \log\log n$ work as in the FLP protocol. For round $r > r'$, the paths of all participating packets are divided into *path segments* of length $\log^3 n$. Round r consists of $3 \cdot 2^{r-r'}$ sub-rounds that work like round r' in the FLP protocol. Each of these sub-rounds is responsible for the routing of the packets along one of their path segments. Initially, every packet chooses a *global delay* Δ i.u.r. from the range $[2 \cdot 2^{r-r'}]$. A packet with a global delay of Δ waits for Δ sub-rounds and afterwards participates in $2^{r-r'}$ consecutive sub-rounds. If, for some $i \geq 1$, it fails to cross its ith path segment in sub-round $\Delta + i$, it gets dumped, *i. e.*, it remains at its current edge buffer until the end of round r.

It is easy to see that w.h.p. the congestion within every sub-round is at most $\log^3 n$ for every participating packet as the packets choose their initial delays i.u.r. Under this assumption, it follows from the previous section that w.h.p. a packet fails in none of its sub-rounds. Since the runtime of round r is at most

$$O(2^{r-r'} \cdot (r' \cdot 2^{r'})) = O(r' \cdot 2^r) = O(2^r \cdot \log\log n) = O(c_p + d_p \cdot \log\log n))$$

for $r > r'$, Theorem 1.2 follows.

2.2 Dynamic packet switching

In this section we prove Theorem 1.3. For this, we first show how to modify the SLP protocol to obtain an $O(c_p + d_p \cdot \log\log n)$-protocol for the dynamic packet switching model, called *FDLP protocol*. In the following, we call the time step at which a packet is injected its *birth date*. We define a path p to belong to set S_i if, at the time of its injection, $\max\{\frac{c_p}{2\log\log n}, d_p\} \in [2^{i-1}, 2^i - 1]$. (Recall that c_p is the congestion caused by all packets that are at least as old as the packet following p.) One of the differences to the static situation is that we now assume that all edges have a bandwidth of $3\beta \log\log n$. The available bandwidth is divided into three parts. Each part has a bandwidth of $b = \beta \log\log n$. The ith part is reserved for packets that are in phase i of their current round of the FDLP protocol (see below). As noted in Section 2.1.1, each step using edges of this bandwidth can be simulated by $3\beta \log\log n$ steps using edges of bandwidth 1. We assume that all paths used by the packets are short-cut free. Due to this assumption, the FDLP protocol does not require ghost packets any more. It works as follows:

Every newly injected packet starts *immediately* with round 1. As soon as it finishes round r, it starts with round $r+1$, and so on, until it reaches its destination. For $r \leq 2\log\log n$, round r works as follows:

Round r of the FDLP protocol for packet P:

- **Phase 1:** This phase takes $3T^2$ time steps. If P has not yet reached its destination, it chooses i.u.r. an initial delay $\delta \in [2T^2]$ with $T = \sqrt{2^r}$. If P chooses a delay of δ, it waits in its current edge buffer for δ time steps and then tries to move on without waiting until it reaches its destination or traversed T^2 edges.

  In the case that P wants to cross an edge e at the same time as at least b other packets that are at least as old as P, then P remains in the outgoing buffer of e until the end of the phase. Otherwise, P tries to reserve bandwidth for its counting packet for exactly $3T^2$ time steps later. If this is not possible, because at that time step the bandwidth of e has already been completely reserved by b non-younger packets, then P also remains in the outgoing buffer of e until the end of the phase. In both cases, P is called *unsuccessful*.

- **Phase 2:** This phase also takes $3T^2$ time steps. For packet P, a counting packet P' is sent along P's path, using the same initial delay as P. Since P successfully reserved bandwidth for P' till its current position, P' will manage to reach P in phase 2. The task of P' is to count the number of possible delays for P for which either P or P' would have been unsuccessful (*i. e.*, P' has to count the number of delays for P for which either the run of P or the run of P' contains a site with at least b non-younger packets, not counting P and P'). If P' counts more than $\min\{\beta T/\log T, \beta\log n/\log\log n\}$ of these delays, P is dumped, which means that it stays in its current edge buffer for the rest of the round. Obviously, P' needs only $O(\log n)$ bits to store the necessary information.

- **Phase 3:** The third phase consists of $2T$ *passes*. If P was unsuccessful but not dumped, than P chooses for every pass i.u.r. an initial delay from a range of $[\gamma T]$, where γ is a sufficiently large constant depending on β. Then it attempts to move along the next T edges of its path. Once a P reaches its destination or traversed altogether T^2 edges in the current round, it stops and is declared *successful*.

For $r > 2\log\log n$, round r works as round r of the SLP protocol with the difference that $r' = 2\log\log n$ and a sub-round works as round r' above.

Since we restrict ourselves to short-cut free paths, the probabilities that different runs have a $\geq b$-site are negatively correlated and therefore can be regarded as independent of each other for an upper bound on these runs. (Thus, we do not need ghost packets any more to destroy dependencies.) As we will see, this can be used to prove that, for any round r of a packet whose path belongs to $S_{r'}$ with $r' \leq r$, the probability that this packet is not successful is at most $\mathrm{e}^{-c\cdot\min\{T,\log n\}}$ for any constant $c > 0$. Since round r now takes only $O(T^2)$ steps (instead of $O(T^3)$ steps for FLP and SLP), the probability for an additive delay of at least t can be improved to

at most $2^{-\Theta(\sqrt{t/\log t})}$. We will prove this by three lemmas. In the following, let the function ϕ be defined as $\phi(x) = \min\{x, \log n\}$.

Lemma 2.6 *If the path of a packet P belongs to $S_{r'}$ for some $r' \leq r$, then the probability that it is dumped in round r is at most $\mathrm{e}^{-c\cdot\phi(T)}$ for any constant c depending on β.*

Proof. Suppose that P is in round $r \geq r'$. If $r \leq 2\log\log n$, then P has to follow a path of length $T^2 = 2^r$ and the congestion along this path caused by packets that are at least of the same age as P is at most $r \cdot 2^r = 2T^2 \log T$. Clearly, each of these packets has to be in a round of number at least r. Thus, the range from which these packets have chosen their most recent delay is at least $2T^2$. Since the delays are chosen independently at random, the probability that $b = \beta \log\log n$ of these packets (or their counting packets) aim for some fixed site in the run of P or P' is at most

$$2\binom{2T^2\log T}{b}\left(\frac{1}{2T^2}\right)^b \leq 2\left(\frac{\mathrm{e}}{\beta}\right)^{\beta\log T} .$$

Let $X_1, \ldots, X_{4T^2}$ denote a collection of binary random variables with the property that X_i is 1 if and only if the ith run of P or the ith run of P' has at least one $\geq b$-site. Furthermore, let $X = \sum_{i=1}^{2T^2} X_i$. For each X_i it holds that

$$\Pr[X_i = 1] \leq T^2 \cdot 2\left(\frac{\mathrm{e}}{\beta}\right)^{\beta\log T} .$$

If $\beta \geq 2\mathrm{e}$ and T is large enough, then

$$\mathbf{E}[X] \leq \sum_{i=1}^{2T^2} 2T^2 \cdot \left(\frac{\mathrm{e}}{\beta}\right)^{\beta\log T} \leq 4T^4 \cdot T^{-\beta} \leq 1 .$$

Since we only allow short-cut free paths, the X_i are negatively correlated. Hence, we can use the Chernoff bounds to bound the probability that at least $\beta T/\log T$ of P's delays are bad by

$$\Pr[X \geq \beta T/\log T] \leq \left(\frac{\mathrm{e}}{\beta T/\log T}\right)^{\beta T/\log T} \leq \mathrm{e}^{-\beta T/2}$$

if T is sufficiently large and $\beta \geq \mathrm{e}$. Thus, the probability that P is dumped is at most e^{-cT} with $c = \beta/2$.

If $r > 2\log\log n$, then it can be shown in a similar way as above that the probability that P is dumped is at most $\mathrm{e}^{-c\log n}$. □

Lemma 2.7 *Consider any packet P with path in $S_{r'}$ that is in Phase 3 of round $r \geq r'$. The probability that there is an edge along P's path with a congestion of at least γT is at most $\mathrm{e}^{-c\cdot\phi(T)}$ for any constant $c > 0$ depending on β.*

Proof. Recall that Phase 3 has its extra part of the bandwidth. Hence, the congestion we have to bound is solely due to packets that are also in Phase 3 of some round.

Consider some fixed edge e along the path of P. First, we bound the expected number of active path segments of packets participating in Phase 3 containing e. Suppose that $r \leq 2\log\log n$. We know that there are at most $r \cdot 2^r = 2T^2 \log T$ packets in the system that intend to cross e and that are at least as old as P. Since P is already in Phase 3 of its round, this means that all the other packets must be in Phase 3 of some round at least r in order to contribute to the congestion. Let the random variable X denote the congestion caused by these pacekts. Similar to Lemma 2.4, it can be shown that the probability of a packet to be unsuccessful and not dumped is at most $(\beta T/\log T)/(2T^2)$. Thus,

$$\mathbf{E}[X] \leq 2T^2 \log T \cdot \frac{\beta T/\log T}{2T^2} = \beta T .$$

Since, according to the proof of Lemma 2.4, the probability of a packet to be unsuccessful and not dumped can be viewed as being independent of the other packets for an upper bound on the congestion, we can use the Chernoff bounds to obtain

$$\Pr[X \geq 2\beta T] \leq \mathrm{e}^{-\beta T/3} .$$

Thus, the probability that there is an edge along P's path with congestion at least $2\beta T$ is at most

$$T^2 \cdot \mathrm{e}^{-\beta T/3} \leq \mathrm{e}^{-cT}$$

if $\beta \geq 3c + 1$ and T is sufficiently large.

For $r > 2 \log \log n$, a similar proof shows that the probability that P has a congestion of at least $2\beta \log n$ along its path it at most $\mathrm{e}^{-c \log n}$. □

Lemma 2.8 *Consider any packet P with a path in $S_{r'}$ that was unsuccessful in Phase 1 of round $r \geq r'$ but has not been dumped. If the congestion bound in Lemma 2.7 is true for P's path, then it holds: The probability that P is unsuccessful in Phase 3 is at most $\mathrm{e}^{-c \cdot \phi(T)}$ for any constant $c > 0$ depending on β.*

Proof. The proof of this lemma is similar to the proof of Lemma 2.5. □

The three lemmas above imply that the probability of a packet with path in $S_{r'}$ to fail a round $r \geq r'$ is at most $2^{-c \cdot \phi(T)}$ for any constant $c > 0$, as desired. Next we show how to transform the FDLP protocol into a dynamic protocol, called DLP, that is stable.

2.2.1 Stability under any bounded injection model

In order to complete the proof of Theorem 1.3 it remains to show how to ensure stability. Consider an arbitrary path selection strategy based on an arbitrary bounded injection model, and applied to an arbitrary network. Let T be the minimum number for which there is a λ such that, for any fixed time interval of length T and edge e, at most λT paths are generated (w.h.p.), that contain e. Furthermore, set $D = d_{\max} \cdot \log \log n$ with $d_{\max}$ being the length of a longest possible path injected into the system.

We extend our dynamic protocol presented in the previous section to a protocol called DLP in the following way: Give every edge an additional buffer called *overflow stack*. Packets that are longer in the system than some threshold t are moved on top of the overflow stack. In order to re-inject the packets into the system, we divide the time into frames of length m/λ_r for some $\lambda_r < 1$, where m is the number of edges in the network. During each frame, each edge is allowed to re-inject the first packet (if it is not empty) of its overflow stack. In fact, the ith edge is allowed to re-inject a packet in the i/λ_rth time step of each frame. These re-injected packets, of course, increase the injection rate. For some fixed time interval I of length T and edge e, up to $(\lambda + \lambda_r) \cdot T$ paths are injected into the system (w.h.p.) during I that contain e. Now set $\lambda_{\text{eff}} = \lambda + \lambda_r$.

We already know that the time needed by the FDLP protocol to send a packet along a path p is at most $\alpha \cdot (c_p + d_p \cdot \log \log n) + \beta \log^2 n$, w.h.p., where α and β are constants. Let t be chosen as $\max[2(\alpha D + \beta \log^2 n), T]$. Since the congestion of a newly generated path p is at most $\lambda_{\text{eff}} \cdot t$ (w.h.p.), the time a packet needs to traverse p is at most

$$\begin{aligned} \alpha \cdot (\lambda_{\text{eff}} \cdot t + D) + \beta \log^2 n &\leq \alpha \cdot \left(\lambda_{\text{eff}} t + \frac{t}{2\alpha} - \frac{\beta \log^2 n}{\alpha} \right) + \beta \log^2 n \\ &= \alpha \cdot (\lambda_{\text{eff}} + 1/(2\alpha)) \cdot t \end{aligned}$$

w.h.p. This is at most t if $\lambda_{\text{eff}} \leq \frac{1}{2\alpha}$. Therefore, if we choose λ_r as $\frac{1}{4\alpha}$, the DLP protocol will be stable for any $\lambda \leq \frac{1}{4\alpha}$.

Using methods in [22] it can be shown that, if the probability of a packet to get moved to an overflow stack is sufficiently small (a probability of $n^{-\Theta(1)}$ suffices), then the number of packets stored in overflow stacks does not grow unboundedly in the course of time. Hence, the DLP protocol is stable up to some constant λ. Furthermore, the expected routing time of a packet can be shown to be similar to the expected routing time given by the SLP protocol.

3 Strategies for the Virtual Circuit Switching Model

In this section, we present a protocol called DLC that can be applied either to the static or to the dynamic virtual circuit switching model with $\delta(r, d, n) = 1/r + \Theta(d \log\log\log n)$. Furthermore, we show that it achieves the performance stated in Theorem 1.4. The DLC protocol uses a recursive refinement strategy over two levels, and applies techniques presented in the previous section and by Rabani and Tardos in [21].

In order to simplify our analysis, we will transform the possibly deterministic or bursty injection process into a simple stochastic injection process: All packets generated for session i are first moved to their injection buffer. From there, we take a packet every $1/r_i$ steps. For this packet, a delay is chosen i.u.r. from a range of $[1/r_i]$. After this delay, the packet finally starts to participate in DLC. Since in our virtual circuit switching models it is guaranteed that, for every edge e, the sum of the rates of the sessions using it is bounded by some parameter $\lambda < 1$, and in the dynamic model overlaps of old and new sessions will be prevented by $\delta(r, d, n)$, we will be able to design a routing protocol with the help of our stochastic injection process that ensures delay bounds for the packets that depend on local parameters.

3.1 Description of the DLC protocol

To simplify the description, let us assume that all edges have a bandwidth of $b = \Theta(\log\log\log n)$, *i. e.*, every edge can forward up to b packets in one time step. Furthermore, we assume that the injection rate λ is bounded by $\epsilon \log\log\log n$ for some (sufficiently small) constant $\epsilon > 0$. It is easy to see that this model can be transferred to a model in which every edge has a bandwidth of 1 and $\lambda' \leq \lambda/b$ such that a packet that has an end-to-end delay of t in the former model has an end-to-end delay of at most $t \cdot b$ in the latter model.

We partition the time into disjoint intervals of length $O((\log n \cdot \log\log n)^3)$, called *global phases*. The starting points of the global phases are synchronized among all processors. Each global phase has two tracks: a *normal track* and a *catch-up track*. The bandwidth of the edges is divided evenly among the tracks, *i. e.*, every track can forward up to $b/2$ packets over an edge in each time step. Let the global phases be numbered consecutively, starting with 1. A packet participates alternately in a normal track and a catch-up track. In particular, a packet generated in a global phase with even (resp. odd) number only participates (up to a special warm-up phase, which will be explained later) in normal tracks of global phases with even (resp. odd) number and catch-up tracks of global phases with odd (resp. even) number.

Let us call a packet that is already in the system for at least 3 global phases an *old* packet. Otherwise it is called a *young* packet. In the following two subsections we explain how to route these packets.

3.1.1 Routing the old packets

Every old packet tries to cross exactly $(\log n \cdot \log\log n)^3$ edges in each of its normal tracks. If it fails in its normal track, it uses the subsequent catch-up track to cross the remaining edges. If it also fails in the catch-up track it is moved to a special buffer called *overflow stack*. From there it is injected back into the system as described in Section 2.2.1. In the following we describe how the normal tracks and the catch-up tracks work.

Structure of a normal track

A normal track consists of $\log^2 n \cdot (\log\log n)^3$ time intervals of equal length called *normal passes*. Each normal pass is responsible for sending the packets along $\log n$ further edges of their paths. It consists of $(\alpha+1)\frac{\log n}{(\log\log n)^3}$ time intervals of equal length, called *local phases*, where $\alpha > 0$ is some constant. A local phase works as round r of Static Protocol I with $r = 3\log\log\log n$. The task of a local phase is to send the packets along $(\log\log n)^3$ further edges of their paths. (We assume the bandwidth b to be chosen such that the packets can traverse their paths in phase 1 of the local phase without waiting.) If a packet fails to cross this number of edges in a local phase, it stops for the rest of the normal track (and therefore fails it).

At the beginning of a normal pass, every old packet chooses i.u.r. an initial delay $\delta \in [\alpha\frac{\log n}{(\log\log n)^3}]$. A packet that is assigned a delay of δ waits in its current edge buffer for δ local phases. Afterwards it participates in $\frac{\log n}{(\log\log n)^3}$ consecutive local phases.

From Section 2 we know that a local phase takes $O((\log\log n)^3)$ time steps (recall that the edges have a bandwidth of $b = \Theta(r)$). Hence a normal pass takes $O(\log n)$ time steps, and therefore a normal track takes $O((\log n \cdot \log\log n)^3)$ time steps, as desired.

Structure of a catch-up track

A catch-up track consists of $2\log n$ time intervals of equal length called *catch-up passes*. Each catch-up pass is responsible for sending the packets along $\log^2 n \cdot (\log\log n)^3$ further edges of their paths. Once a packet traversed $(\log n \cdot \log\log n)^3$ edges together with the previous normal track, it stops for the rest of the catch-up track and is declared *successful*. A catch-up pass consists of $(\alpha+1)\log^2 n$ local phases for some constant $\alpha > 0$. A local phase in a catch-up track works as a local phase in a normal track.

At the beginning of each catch-up pass, every still unsuccessful packet chooses i.u.r. an initial delay $\delta \in [\alpha \log^2 n]$. A packet that is assigned a delay of δ waits in its current edge buffer for δ local phases. Afterwards it participates in $\log^2 n$ consecutive local phases.

Since a local phase takes $O((\log\log n)^3)$ time steps, each catch-up track requires $O((\log n \cdot \log\log n)^3)$ time steps, as desired.

3.1.2 Routing the young packets

We distinguish between young and old packets to allow packets to choose no delay or only very small delays right after they are injected into the system. This has the advantage that packets that only have to travel a very short path, *e. g.*, of length $\log\log n$, do not have to choose initial delays usually required at the beginning of each pass and local phase. Otherwise, such delays would make it impossible for us to guarantee an expected runtime of $O(1/r_i + d_i \log\log\log n)$ for every d_i.

We now describe how to route the young packets. For every packet, we declare the global phase in which it is generated and the following global phase as its *warm-up* normal track. The two phases after its warm-up normal track are called its *warm-up* catch-up track. Let the normal pass in which it is generated together with the following normal pass be called its *warm-up* normal pass. Furthermore, let the first two local phases in which it participates be called its *warm-up* local phase.

A newly injected packet immediately participates in all remaining passes of its warm-up normal track. If it fails in one of the local phases of this track, it participates in its warm-up catch-up track to ensure that it traverses all edges that it was supposed to traverse in its warm-up normal track.

In its warm-up normal pass, the packet immediately (that is, without an initial delay) participates in all remaining local phases of the first pass of its warm-up pass and the first $\frac{\log n}{(\log\log n)^3}$ local phases of the second pass of its warm-up pass. All other passes work as usual.

Finally, we describe how a packet behaves in its warm-up local phase. Let t_l be the number of time steps covered by a local phase. A packet injected into the system t time steps after the beginning of its warm-up local phase immediately tries to traverse $\frac{2t_l-t}{t_l}(\log\log n)^3$ edges. If it fails to traverse $\frac{t_l-t}{t_l}(\log\log n)^3$ edges, it tries to traverse them in the second phase of its warm-up local phase. The remaining of the $\frac{2t_l-t}{t_l}(\log\log n)^3$ edges are traversed in the local phase after the warm-up local phase.

Starting with the local phase after its warm-up local phase, the movement of a young packet is synchronized with all other young packets that have been generated at least two local phases before, in a sense that the number of edges traversed so far divided by the age is the same for all of these packets at the beginning of each normal pass. However, young packets are only synchronized with the old packets after their warm-up tracks. Nevertheless we can route the young and old packets together in the same tracks, since our routing strategy ensures that the expected number of packets traversing any edge at any time step is $O(\log\log\log n)$ for both groups of packets.

3.2 Analysis of the DLC protocol

In this section we sketch the proof of Theorem 1.4. We will only concentrate on bounding the success probability of old packets, simply called packets in the following. The analysis for the young packets is similar. Given a packet P and a track (resp. pass or local phase) π in which P participates, let the *π-segment* of P denote the part of its path it intends to traverse in π. Let us consider in the following some fixed normal track Π. The injection strategy together with our strategy to synchronize the movements of the packets yields the following two lemmata. (The proof of the first lemma is easy and therefore omitted.)

Lemma 3.1 *For any $c > 0$ there is a constant ϵ, so that the probability that the congestion at any edge in any pass in Π exceeds $\log n \cdot \log\log\log n$ is at most n^{-c}.*

Lemma 3.2 *Under the assumption that Lemma 3.1 holds, the probability that the congestion at some fixed edge in some fixed local phase in Π is at least $s = (\log\log n)^3 \log\log\log n$ is at most $(\mathrm{e}/\alpha)^s$.*

Proof. As the packets independently choose random initial delays from a range of $[\alpha\frac{\log n}{(\log\log n)^3}]$ at the beginning of each pass, the bound above can easily be shown with the help of Chernoff bounds. □

Let Π' be the catch-up track following Π. The above two lemmata allow us to prove the following lemma.

Lemma 3.3 *For any $c > 0$ there is a $\gamma > 0$ such that the probability that the congestion at any edge in Π' exceeds $\gamma \log^2 n \cdot (\log\log n)^3 \cdot \log\log\log n$ is at most n^{-c}.*

Proof. Fix some edge e. Set $s = (\log\log n)^3 \cdot \log\log\log n$. Assume that at least $\Gamma = \gamma \log^2 n \cdot (\log\log n)^3 \cdot \log\log\log n$ packets fail in Π whose Π-segments contain e. Then

(1) there must be either $\Gamma/2$ packets that failed in a local phase π of Π in which the congestion at some edge of their π-segments was larger than s, or

(2) there must be $\Gamma/2$ packets that failed in a local phase π of Π in which the congestion at every edge of their π-segments was at most s.

We first consider case 1. Set $g = \Gamma/2$. Since the failures of packets in different passes are independent, we only have to bound the dependencies among failures within one normal pass. Assume that g' of the failures fall into one pass. It is not difficult to show that, by selecting a suitable set of s packets to witness a too high congestion for each of these g' packets, each of these s packets can occur in at most $2s \cdot \log n$ other packet sets. This allows us to conclude that, if altogether g packets fail then there must be $g/(2s^2 \cdot \log n)$ sets of s packets that are either

independent, because they represent congestion events in different normal passes or because the packet sets are disjoint. Thus the probability that there are g packets that fail in a local phase π with a too high congestion is at most

$$\binom{(\log n \cdot \log\log n)^6 \log\log\log n}{g/(2s^2 \cdot \log n)} \left(\frac{e}{\alpha}\right)^{\frac{s \cdot g}{2s^2 \cdot \log n}} \leq n^{-c}$$

if α and γ are sufficiently large.

Next we consider case 2. It is not difficult to see that here the failure of a packet P in some local phase π only depends (in our analysis) on a failure of a packet P' in some local phase π' if $\pi = \pi'$ and there is a packet that intends to traverse an edge of the π-segments of P and P' in π. This implies a $poly(\log\log n)$ dependency among the failure events. Together with the probability bound for a packet to fail in a local phase (see Section 2), we therefore get a similar probability bound as above, which concludes the proof. □

With this lemma and the proof of Lemma 2.5 it is easy to show that the probability that a packet fails Π' is polynomially small in n. This completes the proof of Theorem 1.4.

4 Conclusion

In this paper we presented locally efficient on-line routing protocols for static and dynamic packet switching and virtual circuit switching models. Many open problems remain. For instance, is it possible in the packet switching model to route a packet on-line along a path p in expected time $O(c_p + d_p)$? Can our result for the dynamic packet switching model be generalized to arbitrary simple paths? We assume that, when using the techniques in [18] together with our ideas, it is possible for the virtual circuit switching model to have a protocol that guarantees an expected delay of $O(d_i + 1/r_i)$ for a session-i packet. However, is it possible to have one protocol that guarantees a deviation of more than t from this delay bound with a probability that is exponentially small in t for every t?

References

[1] M. Andrews, A. Fernández, M. Harcol-Balter, T. Leighton, L. Zhang. *General dynamic routing with per-packet delay guarantees of O(distance + 1/session rate)*. In Proc. of the 38th Ann. IEEE Symp. on Foundations of Computer Science, pp. 294-302, 1997.

[2] M. Andrews, B. Awerbuch, A. Fernández, J. Kleinberg, T. Leighton, Z. Liu. Universal stability results for greedy contention-resolution protocols. In *Proc. of the 37th IEEE Symp. on Foundations of Computer Science*, pp. 380-389, 1996.

[3] Andrei Broder, Alan Frieze, Eli Upfal. A general approach to dynamic packet routing with bounded buffers. In *Proc. of the 37th IEEE Symp. on Foundations of Computer Science*, pp. 390-399, 1997.

[4] B. Awerbuch, Y. Bartal, A. Fiat. Competitive distributed file allocation. In *Proc. of the 25th ACM Symp. on Theory of Computing*, pp. 164-173, 1993.

[5] B. Awerbuch, Y. Bartal, A. Fiat. Distributed paging for general networks. in *Proc. of the 7th ACM Symp. on Discrete Algorithms*, pp. 574-583, 1996.

[6] Allan Borodin, Jon Kleinberg, Prabhakar Raghavan, Madhu Sudan, David Williamson. Adversarial queuing theory. In *Proc. of the 28th ACM Symposium on Theory of Computing*, pp 376-385, 1996.

[7] A. Broder, E. Upfal. Dynamic deflection routing on arrays. In *Proc. of the 28th ACM Symposium on Theory of Computing*, pp. 348-355, 1996.

[8] R. Cypher, F. Meyer auf der Heide, C. Scheideler, B. Vöcking. Universal algorithms for store-and-forward and wormhole routing. In *Proc. of the 28th ACM Symp. on Theory of Computing*, pp. 356-365, 1996.

[9] M. Harchol-Balter, D. Wolfe. Bounding delays in packet-routing networks. In *Proc. of the 27th ACM Symp. on Theory of Computing*, pp. 248-257, 1995.

[10] N. Kahale, T. Leighton. Greedy dynamic routing on arrays. In *Proc. of the 6th ACM-SIAM Symp. on Discrete Algorithms*, 1995.

[11] F.T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays · Trees · $Hypercubes$*. Morgan Kaufmann Publishers (San Mateo, CA, 1992)

[12] F.T. Leighton, B.M. Maggs, S.B. Rao. Universal packet routing algorithms. In *Proc. of the 29th IEEE Symp. on Foundations of Computer Science*, pp. 256-271, 1988.

[13] 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.

[14] C. Lund, R. Reingold, J. Westbrook, D. Yan. On-line distributed data management. In *Proc. of the 2nd European Symposium on Algorithms*, 1996.

[15] B. Maggs, F. Meyer auf der Heide, B. Vöcking, M. Westermann. Exploiting locality for data management in systems of limited bandwidth. In *Proc. of the 38th IEEE Symp. on Foundations of Computer Science*, pp. 284-293, 1997.

[16] F. Meyer auf der Heide and B. Vöcking. A packet routing protocol for arbitrary networks. In *Proc. of the 12th Symp. on Theoretical Aspects of Computer Science*, pp. 291-302, 1995.

[17] M. Mitzenmacher. Bounds on the greedy routing algorithm for array networks. In *Proc. of the 6th ACM Symp. on Parallel Algorithms and Architectures*, pp. 346-353, 1994.

[18] R. Ostrovsky, Y. Rabani. Universal O(congestion + dilation + $\log^{1+\epsilon} N$) Local Control Packet Switching Algorithms. In *Proc. of the 29th ACM Symp. on Theory of Computing*, pp. 644-653, 1997.

[19] A.K. Parekh, R.G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The multi-node case. *IEEE/ACM Trans. on Networking* **2**(2), pp. 137-150, 1994.

[20] A. Panconesi, A. Srinivasan. Fast randomized algorithms for distributed edge coloring. In *Proc. of the 11th ACM Symp. on Principles of Distributed Computing*, pp. 251-262, 1992.

[21] Y. Rabani, E. Tardos. Distributed packet switching in arbitrary networks. In *Proc. of the 28th ACM Symp. on Theory of Computing*, pp. 366-375, 1996.

[22] C. Scheideler, B. Vöcking. Universal continuous routing strategies. In *Proc. of the 8th ACM Symp. on Parallel Algorithms and Architectures*, pp. 142-151, 1996.