Scheduling Time-Constrained Communication in Linear Networks

Micah Adler*
Dept. of Computer Science,
Univ. of Toronto,
Toronto ON, M5S3G4,
Canada

Arnold L. Rosenberg†
Dept. of Computer Science,
Univ. of Massachusetts,
Amherst, MA 01003

Ramesh K. Sitaraman‡
Dept. of Computer Science,
Univ. of Massachusetts,
Amherst, MA 01003

Walter Unger§
Lehrstuhl für Informatik I,
RWTH Aachen, Ahornstr. 55,
52074 Aachen, Germany

Abstract

We study the problem of centrally scheduling multiple messages in a linear network, when each message has both a release time and a deadline. We show that the problem of transmitting optimally many messages is NP-hard, both when messages may be buffered in transit and when they may not be; for either case, we present efficient algorithms that produce approximately optimal schedules. In particular, our bufferless scheduling algorithm achieves throughput that is within a factor of two of optimal. We show that buffering can improve throughput in general by a logarithmic factor (but no more), but that in several significant special cases, such as when all messages can be released immediately, buffering can help by only a small constant factor. Finally, we show how to convert our centralized, offline bufferless schedules to equally productive fully distributed online buffered ones. Most of our results extend readily to ring-structured networks.

* Email: micah@cs.toronto.edu. Supported by an operating grant from the Natural Sciences and Engineering Research Council of Canada, and by ITRC, an Ontario Centre of Excellence. This research was conducted in part while he was at the Heinz Nixdorf Institute Graduate College, D-33095 Paderborn, Germany.

† Email: rsnbrg@cs.umass.edu. Supported in part by NSF Grant CCR-97-10367.

‡ Email: ramesh@cs.umass.edu. Supported in part by an NSF CAREER Award No. CCR-97-03017. A portion of the research of the second and third author was done while visiting the Dept. of Mathematics and Informatik, Univ. of Paderborn, Paderborn, Germany.

§ Email: quacks@il.informatik.rwth-aachen.de . The work was carried out while the author was a member of the research group of B. Monien at the University of Paderborn. This work and the visits of the second and third author was partially supported by the German Research Association (DFG) within the SFB 376 "Massive Parallelität: Algorithmen, Entwurfsmethoden, Anwendungen".

SPAA 98 Puerto Vallarta Mexico

1 The Time-Constrained Communication Problem

1.1 Introduction

Communication and interconnection networks are currently undergoing a transition from traditional best-effort data networks to networks capable of routing messages with timing constraints. In the area of communication networks, this shift is motivated by multimedia applications that use continuous media such as video and audio [27]; for instance, a real-time audio packet in a teleconferencing application must reach its destination within a specified window of time for it to have any utility. The analogous shift in the development of interconnection networks is motivated by emerging real-time applications that rely on time-constrained communication, such as industrial process control, avionics, and automated manufacturing [41].

In this paper, we consider the scheduling of the transmission of a given set of *time-constrained* messages in a multi-node network. Our goal is to deliver as many of the given messages as possible, within the following framework:

- Each message consists of a single packet. A network node may send many messages to each of the other nodes.
- Each message m has, in addition to a *source-node* s_m and a *destination-node* d_m, a *release time*, which is the earliest moment at which m can start its journey from s_m to d_m, and a *deadline*, beyond which no purpose is served by delivering m. This means that a message should be dropped as soon as it can no longer be delivered by its deadline.

Our framework allows us to model multiple classes of messages with differing timing requirements—a feature that is essential to modeling multimedia traffic. We can also model messages for which conventional best-effort transmission is sufficient, by setting these messages' associated deadlines to ∞. Our framework should be contrasted with that of more traditional routing problems, which seek to optimize global objectives such as overall completion time

or average message latency, and do not associate time constraints with individual messages.

The Network Model. We focus on routing messages in linear networks—although most of our results apply also to ring-structured networks. This focus on linear topologies is a first step towards considering more complex interconnection topologies proposed in the literature, such as higher-dimensional arrays [41]. There are often also other rationales for the focus, such as the following. (*a*) When routing messages in electro-optical interconnection networks such as hierarchical rings [22] or meshes [41], or their relatives, one might have each packet follow a path composed predominantly of long (inexpensive) bufferless hops, punctuated by a very few (costly) optical-electric conversions at certain nexus nodes. In a mesh, for instance, one might employ a dimension-order routing strategy [41] which uses our near-optimal bufferless routing along rows and along columns but that performs a single optical-electric conversion to change dimensions. (*b*) In less regularly structured communication networks, one often routes messages along subnetworks that are either linear or ring-like, because of their easy routing-path selection (coupled, in the case of rings, with a modicum of tolerance to faults).

An n-node *linear network* can be viewed as a graph whose node-set is $V_n = \{0, 1, \ldots, n-1\}$ and whose arcs are all pairs $\langle k,\ k+1\rangle$, where $0 \leq k < n-1$. We assume a *dual-ported* model where each node can pass and receive one message to/from both neighbors at each step; each arc is a full-duplex link that can accommodate one message in each direction at each step. We assume messages are routed monotonically (i.e., with no backtracking) from their sources to their destinations. Thus, our model allows us to decompose our message delivery problem into two disjoint subproblems, one for left-to-right messages, wherein $s_m < d_m$, and one for right-to-left messages, wherein $s_m > d_m$. Merely superposing optimal solutions to these subproblems yields an optimal solution to the full problem. *Henceforth, we discuss just the left-to-right subproblem.*

We study two scenarios, distinguished by buffering policies. In the first, a network's nodes are allowed to buffer messages in transit, in order to relieve contention for network links. In the second, no such buffering is allowed, so that message m must move one step closer to d_m at every moment after its departure from s_m. The first scenario, which is appropriate for purely electronic environments, has been studied extensively for decades. The second scenario, which is particularly appropriate for current and foreseeable optical technologies [22], is only beginning to receive attention in the literature [5, 37]. Significantly, we shall see that the study of the bufferless scenario provides important insights into the buffered scenario.

Definitions. An *instance* of the buffered (resp., bufferless) message-scheduling problem OPT_B (resp., OPT_{BL}) is a set $\mathcal{I}$ of messages to be routed, possibly using buffers (resp., without using buffers). The goal is to schedule a subset $OPT_B(\mathcal{I}) \subseteq \mathcal{I}$ (resp., $OPT_{BL}(\mathcal{I}) \subseteq \mathcal{I}$) of maximum cardinality. Note that with release times and deadlines, this is the natural question to study, since there is no added benefit for messages that arrive early or only marginally miss their deadline. The set of messages in $\mathcal{I}$ that are successfully delivered under a scheduling algorithm A is denoted $A(\mathcal{I})$; the *throughput* of A is $|A(\mathcal{I})|$.

Summary of Results. We present three categories of results. When messages may *not* be buffered in transit, the problem of maximizing throughput is NP-hard (Section 3.1); however, one can efficiently achieve at least one-half optimal throughput (Section 3.2). In Section 4, we ask how much message-buffering can enhance the throughput of scheduling algorithms. We show that when one or more of the three parameters that complicate message-scheduling—the release time, the source-target distance, and the allowable delay—is held constant, buffers can increase throughput by only a small constant factor (Section 4.1); however, in general, buffers can increase throughput by as much as a logarithmic factor, but no more (Section 4.2). We show that achieving optimal throughput is NP-hard also when message-buffering *is* allowed (Section 5.1); however, we provide a *distributed* (a/k/a "local control") and online algorithm that uses buffers to *exactly* mimic the performance of the (centralized and offline) bufferless approximation algorithm (Section 5.2). In this algorithm, information about a message m to be sent arrives only at the source-node s_m, and only at the time when message m is released. Each node makes all routing decisions locally, using only the information that it receives when messages are released or that it receives from other nodes during the course of routing messages. The results of Section 4 that relate buffered to bufferless routing imply that our distributed and online algorithm achieves nearly optimal throughput.

1.2 Related Work

To our knowledge, we present the first analytical results for routing time-constrained messages that have arbitrary release times and deadlines. Of course, *best-effort* routing in fixed-connection networks has a long history; see [23, 24] for a survey. Much early work routes *static* message-sets, wherein all messages are released simultaneously [45, 26, 40, 25, 31, 32]. More recent work looks at *dynamic* routing, wherein messages arrive at varying times, governed either by a random process [43, 7, 17] or an adversary [6, 2]. Several recent studies focus on *bufferless* routing algorithms, which allow simpler, faster switches. Optical networks provide strong incentives to avoid buffering, due to the cost of optical $\leftrightarrow$ electronic conversions [22]. The important class of hot-potato (a/k/a deflection) bufferless routing algorithms has been widely studied [1, 4, 10, 20, 8, 44]. More general approaches to bufferless routing can be found in [5, 37, 39, 9, 12]. These sources focus on optimizing a global parameter such as overall completion time or average message latency: individual messages do not have deadlines.

Some recent papers focus on the "session model," wherein session-i packets arrive once every $1/r_i$ time steps and travel along specified length-d_i paths. A distributed routing algorithm in [35, 36] guarantees a delay of $2d_i/r_i$ for session-i packets, provided that edges are used with rate < 1; this bound is improved to $O(1/r_i + d_i)$, via a centralized algorithm, in [3]. By creating a different session for each packet, these results can be used to route each packet p_i to its destination in time $O(c_i + d_i)$, where c_i is the maximum congestion on the path of p_i. Note that while these results provide per-packet delay bounds, they do not accommodate arbitrary timing requirements for each packet, i.e., arbitrary release times and deadlines. A recent paper [30] considers the problem of routing a static set of messages with arbitrary deadlines in the linear network *without* dropping any of the messages. They show that if there exists a feasible schedule then the closest-deadline-first greedy strategy succeeds in routing all the messages.

There are several empirical, simulation-based, studies of time-constrained routing. [41] introduces a router architecture for messages with individual deadlines, using a multi-class variation of the earliest due-date algorithm [29]. [48] proposes a minimum-laxity-first protocol for transmitting messages with deadlines in a multi-access shared-bus network. Some scheduling policies—such as Virtual Clock [47], Stop-and-Go [15], Rotating Combined Queuing [21]—do not explicitly use message deadlines, but just attempt to keep the worst-case message delay small and bounded. Other relevant experimental work includes [33, 46, 28, 42, 13].

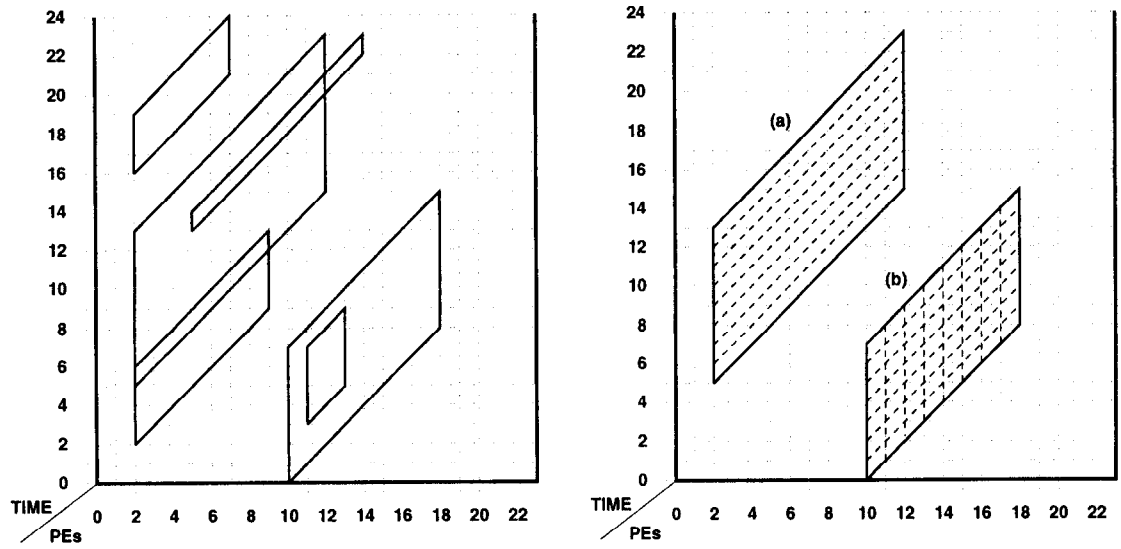

Figure 1: Left side: *Six message parallelograms on the 22-node line.* Right side: (a) *Available bufferless delivery routes for message #2 of the lefthand figure;* (b) *the mesh of available buffered delivery routes for message #5 of the lefthand figure.*

2 A Geometric View of the Problem

Fundamental to both our insights and analyses is the following geometric transformation of our message scheduling problem. Say that we are transmitting an ensemble $\mathcal{M}$ of messages in the network whose node-set is $V_n \stackrel{\text{def}}{=} \{0, 1, \ldots, n-1\}$ and whose arcs are all pairs $\langle k,\ k+1\rangle$, where $0 \le k < n-1$. Consider the "one-way infinite" subset $M_n \stackrel{\text{def}}{=} \{\langle t, v\rangle \mid t \ge 0,\ \text{and}\ 0 \le v \le n-1\}$ of the two-dimensional integer lattice.

- Each "row" $R_t \stackrel{\text{def}}{=} \{\langle t, v\rangle \mid 0 \le v \le n-1\}$ of M_n represents "time-instant" t.
- Each "column" $C_v \stackrel{\text{def}}{=} \{\langle t, v\rangle \mid t \ge 0\}$ represents node v when
  - v is the source of a message, or
  - v is the destination of a message that originates at node $v' < v$.

Then each message $m \in \mathcal{M}$ can be viewed as a *parallelogram* within M_n, whose left and right sides are vertical and whose tops and bottoms have 45-degree southwest-to-northeast (henceforth "sw-ne") slopes. Specifically, if m has source-node s_m, destination-node d_m, release-time $t_m^{(r)}$, and deadline $t_m^{(d)}$, then m's parallelogram has the following shape. The parallelogram's:

- left (vertical) side lies between rows $t_m^{(r)}$ and $t_m^{(d)} - (d_m - s_m)$ within column s_m;
- its right (vertical) side lies between rows $t_m^{(r)} + (d_m - s_m)$ and $t_m^{(d)}$ within column d_m.

$t_m^{(s)} \stackrel{\text{def}}{=} t_m^{(d)} - t_m^{(r)} + s_m - d_m$ is the *slack* of message m, and $\delta_m \stackrel{\text{def}}{=} d_m - s_m$ is its *span*.

In Fig. 1, we depict six time-constrained messages in a 22-node linear network, over a period of 25 time units. Reading from left to right in the figure, and from bottom to top in each column, the messages have the defining characteristics listed in the following table.

Message Number	Source Node	Destination Node	Release Time	Deadline
$m = 1$	$s_m = 2$	$d_m = 9$	$t_1^{(r)} = 2$	$t_1^{(d)} = 13$
$m = 2$	$s_m = 2$	$d_m = 12$	$t_2^{(r)} = 5$	$t_2^{(d)} = 23$
$m = 3$	$s_m = 2$	$d_m = 7$	$t_3^{(r)} = 16$	$t_3^{(d)} = 24$
$m = 4$	$s_m = 5$	$d_m = 14$	$t_4^{(r)} = 13$	$t_4^{(d)} = 23$
$m = 5$	$s_m = 10$	$d_m = 18$	$t_5^{(r)} = 0$	$t_5^{(d)} = 15$
$m = 6$	$s_m = 11$	$d_m = 13$	$t_6^{(r)} = 3$	$t_6^{(d)} = 9$

To understand the role of the parallelograms, focus on a message m that can be delivered. Then m must leave s_m at some time $t_m^{(r)} \le t_1 \le t_n^{(d)} - (d_m - s_m)$; it can leave no earlier, for $t_m^{(r)}$ is its release time; it can leave no later, for then it would not reach d_m before its deadline $t_m^{(d)}$. By similar reasoning, m arrives at d_m at some time $t_m^{(r)} + (d_m - s_m) \le t_2 \le t_m^{(d)}$. The trajectory of m is a path in m's parallelogram from row t_1 on the left to row t_2 on the right, having the following form.

- In the bufferless case, the trajectory is a 45-degree sw-ne line whose linearity reflects m's unimpeded progress from s_m to d_m, with no delay at an intermediate node; see Fig. 1(a).
- In the buffered case, the trajectory is a "staircase": left-to-right motion is along 45-degree sw-ne edges; "risers" (which represent m's being detained in a buffer) are upward edges; see Fig. 1(b).

A *schedule* for a set $\mathcal{M}$ of messages is a set of trajectories, at most one for each $m \in \mathcal{M}$, such that no two trajectories share a 45-degree sw-ne edge; distinct trajectories in a schedule *may*, however, share a "riser" edge or an endpoint. The *throughput* of a schedule is the number of independent trajectories that it specifies.

3 Bufferless Message-Scheduling

Our goal in this section is a centralized bufferless scheduling algorithm whose schedules are maximal in throughput. We show in

Section 3.1 that this goal is NP-hard. In Section 3.2, we devise a centralized bufferless scheduling algorithm whose schedules come within a factor of 2 of the goal.

3.1 The NP-Hardness of Optimal Bufferless Scheduling

Unfortunately, like many significant scheduling problems, the bufferless message-scheduling problem OPT_{BL} is NP-hard.

Theorem 3.1 *The bufferless message-scheduling problem OPT_{BL} is NP-hard.*

Proof. See Appendix A.

3.2 A 2-Approximation Algorithm for Bufferless Scheduling

While Theorem 3.1 suggests that we cannot efficiently achieve truly optimal throughput, we show now that we can efficiently achieve throughput that is within a factor of 2 of optimal.

Theorem 3.2 *There is a polynomial-time algorithm that produces bufferless schedules whose throughput is within a factor of 2 of optimal.*

Proof. We represent each message in the set $\mathcal{I}$ to be scheduled via its parallelogram. At each step of our algorithm, we maintain the set $\mathcal{U} \subseteq \mathcal{I}$ of as-yet unscheduled messages and the set $\mathcal{S}$ of assigned message-trajectories; initially, $\mathcal{U} = \mathcal{I}$ and $\mathcal{S} = \emptyset$.

A *scan line* is any sw-ne line segment in M_n that originates either on the X-axis or in Column C_0 and terminates either on row $\max_{m \in \mathcal{I}}\{t_m^{(d)}\}$ or in Column C_{n-1}. Each scan line is a segment of a level line of the function $x - y$, hence is uniquely specified by its *ao-parameter*, namely, the (abscissa $-$ ordinate) difference of any point on the line. This specification affords us a natural "left-to-right" ordering of scan lines, in increasing order of ao-parameters.

Algorithm BFL: preliminaries. A scan line is *active* if no scan line to its "left" has already been scanned; initially all scan lines are active. A scan line is *relevant* to a message m if it intersects m's parallelogram. We maintain a priority queue Q of scan lines, ordered by ao-parameter, from which we can extract the "rightmost" active scan line that is relevant to some $m \in \mathcal{U}$. We denote by $BFL(\mathcal{I})$ the subset of $\mathcal{I}$ actually scheduled by Algorithm BFL.

Algorithm BFL

1. Extract a scan line ℓ from the priority queue Q.
2. Determine the sequence of segments of ℓ determined by all $m \in \mathcal{U}$ which ℓ is relevant to.
3. Use a se-nw scan of ℓ to find a maximal set $\mathcal{S}(\ell)$ of segments that are independent in the sense of not intersecting, except perhaps at their endpoints. In more detail:
   a. Start scanning ℓ at the "lowest" left endpoint of a parallelogram that ℓ intersects.
   b. Find the "lowest" intersection of ℓ with the right endpoint of a parallelogram p. Use the segment of ℓ that intersects p to schedule the message associated with p. For the remainder of this step, ignore all parallelograms whose left endpoints along ℓ lie "below" the right endpoint of p.
   c. Continue scanning ℓ at the "lowest" left endpoint of a parallelogram that either coincides with, or lies "above" the right endpoint of p along ℓ. If no such left endpoint exists, then this step is complete; else repeat sub-step (b).

   Note that we never schedule any parallelogram whose intersection with ℓ properly contains some other parallelogram's intersection with ℓ.
4. Add the segments in $\mathcal{S}(\ell)$ to set $\mathcal{S}$; remove the associated messages from $\mathcal{U}$.
5. If Q is not empty, go to step 1; else return the schedule $\mathcal{S}$.

Claim. $|BFL(\mathcal{I})| \geq \frac{1}{2}|OPT_{BL}(\mathcal{I})|$.
Verification. Consider an arbitrary m that is assigned a scan line segment by OPT_{BL} but not by BFL. Let the *right endpoint* of a scan line segment be the last edge of the segment. It must be that the scan line segment $\ell(m)$ assigned to m by OPT_{BL} contains the right endpoint of at least one scan line segment in $\mathcal{S}$; associate $\ell(m)$ with the "leftmost" of these. Since OPT_{BL} produces a valid schedule, at most one such $\ell(m)$ can be assigned to each scan line segment in BFL. We thus have a one-to-one mapping from $OPT_{BL}(\mathcal{I}) - \mathcal{S}$ into $\mathcal{S}$, whence the desired inequality. ■

The implementational details and a tighter time analysis of Algorithm BFL appear in the full paper. Here, we prove the following claim.
Claim. Algorithm BFL can be implemented in time polynomial in $n + |\mathcal{I}|$, independent of the message slacks.
Verification. The slack of a message m can be set to $\min\{|\mathcal{I}| - 1, t_m^{(s)}\}$, without altering the throughput. Thus, the number of scan lines considered in step 1 is polynomial in $|\mathcal{I}|$. For a given scan line ℓ with ao-parameter α, a message m is relevant to ℓ if and only if ℓ intersects the message-parallelogram of m, i.e., $d_m - t_m^{(d)} \leq \alpha \leq s_m - t_m^{(r)}$. The left (or lower) endpoint of the segment of ℓ contained in the parallelogram of message m is $\langle s_m, s_m - \alpha \rangle$, while the right (or upper) endpoint is $\langle d_m, d_m - \alpha \rangle$. Thus, computing the set of segments of ℓ that correspond to relevant messages in $\mathcal{U}$ in step 2 takes time $O(|\mathcal{U}|)$, which is $O(|\mathcal{I}|)$. Choosing a subset $\mathcal{S}(\ell)$ of these segments takes time polynomial in $n + |\mathcal{U}|$. Therefore, steps 3 and 4 take time polynomial in $n + |\mathcal{I}|$. ■

4 Comparing Buffered and Bufferless Message Scheduling

How much can the ability to buffer messages enhance the throughput of time-constrained communication? We answer this question with bounds on $OPT_B(\mathcal{I})$ in terms of $OPT_{BL}(\mathcal{I})$. We show that, when all messages have the same slack, or the same span, or the same release time, buffering can enhance throughput by at most a small constant factor (Section 4.1), while in general, it can improve throughput by a logarithmic factor, but no more (Section 4.2).
Notation. For each message m in $BFL(\mathcal{I})$ (resp., $OPT_B(\mathcal{I})$) (resp., $OPT_{BL}(\mathcal{I})$), we denote by $\pi(m)$ (resp., $\pi_B(m)$) (resp., $\pi_{BL}(m)$) the trajectory assigned to m by algorithm BFL (resp., an optimal buffered schedule) (resp., an optimal bufferless schedule).

4.1 Important Special Cases

We focus on three natural restrictions of the routing problem, assuming in turn that all messages have the same slack, or the same span, or the same release time.

4.1.1 The Power of Buffers when Message-Slacks are Uniform

Theorem 4.1 *If all messages in problem instance $\mathcal{I}$ have the same slack S, then $OPT_B(\mathcal{I}) \leq 3 \cdot OPT_{BL}(\mathcal{I})$.*

Proof. We compare $|OPT_B(\mathcal{I})|$ with $|BFL(\mathcal{I})|$, using a scheme in which messages in $OPT_B(\mathcal{I}) - BFL(\mathcal{I})$ donate "credits" to messages in $BFL(\mathcal{I})$. A message $m \in OPT_B(\mathcal{I}) - BFL(\mathcal{I})$ is not included in $BFL(\mathcal{I})$ because each of the $S+1$ potential bufferless trajectories specified by its parallelogram contains the right endpoint of at least one trajectory $\pi(m')$ of a message $m' \in BFL(\mathcal{I})$. We collect a set D_m of some $S+1$ messages from $BFL(\mathcal{I})$ that collectively block all of m's potential bufferless trajectories, and we have m "donate" $1/(S+1)$ units of credit to each $m' \in D_m$. (Note that some $m' \in BFL(\mathcal{I})$ may receive credits from more than one $m \in OPT_B(\mathcal{I}) - BFL(\mathcal{I})$.)

Clearly, this scheme allocates $|OPT_B(I) - BFL(I)|$ units of credit in all. To bound the total credits a message $m' \in BFL(\mathcal{I})$ can receive, let $R_{m'} \subseteq OPT_B(\mathcal{I}) - BFL(\mathcal{I})$ be the set of messages that donated credit to m'. The parallelogram of each $m \in R_{m'}$ must contain the right endpoint, $\langle v, t\rangle \in M_n$, of $\pi(m')$. Hence, m's optimal buffered trajectory $\pi_B(m)$ must "reach" node v, say at time τ_m. Since at most one message arrives at v in a single timestep, τ_m is unique to message m. Further, since all messages have slack S, each $m \in R_{m'}$ has $|\tau_m - t| \leq S$, whence $|R_{m'}| \leq 2S+1$. Since each $m \in R_{m'}$ contributes exactly $1/(S+1)$ units of credit to m', the total credit received by m' is

$$\sum_{m \in R_{m'}} \frac{1}{S+1} \leq \frac{2S+1}{S+1} \leq 2. \qquad (1)$$

It follows that the aggregate credit received by messages in $BFL(\mathcal{I})$ does not exceed $2 \cdot |BFL(\mathcal{I})|$. The desired bound now follows from the fact that the total credits donated by messages in $OPT_B(\mathcal{I}) - BFL(\mathcal{I})$ equals the total credits received by messages in $BFL(\mathcal{I})$, combined with the definition of optimality:

$$|OPT_B(\mathcal{I})| \leq 3 \cdot |BFL(\mathcal{I})| \leq 3 \cdot |OPT_{BL}(\mathcal{I})|.$$

■

4.1.2 The Power of Buffers when Message-Spans are Uniform

Theorem 4.2 *If all messages in problem instance $\mathcal{I}$ have the same span δ, then $OPT_B(\mathcal{I}) \leq 2 \cdot OPT_{BL}(\mathcal{I})$.*

Proof. We show how to route at least half the messages in $OPT_B(\mathcal{I})$ without buffers. Partition $OPT_B(\mathcal{I})$ into sets S_0 and S_1, by placing each $m \in OPT_B(\mathcal{I})$ into S_j iff m's parallelogram intersects a column $C_{i(\delta+1)}$, where $0 \leq i \leq \lfloor (n-1)/(\delta+1) \rfloor$ and $i \bmod 2 = j$: each $m \in OPT_B(I)$ intersects *exactly* one such column. At least one $|S_j| \geq |OPT_B(\mathcal{I})|/2$; assume without loss of generality that set is S_0. We construct a bufferless trajectory $\widetilde{\pi}(m)$ for each $m \in S_0$ as follows. Let $\pi_B(m)$ "reach" a column $C_{2i(\delta+1)}$ at time τ_m. (By construction, such an i exists for each $m \in S_0$.) The bufferless trajectory $\widetilde{\pi}(m)$ is the unique sw-ne segment that passes through point $\langle 2i(\delta+1), \tau_m\rangle$.

We claim that the bufferless schedule just constructed is valid, in that distinct messages are assigned disjoint trajectories. To wit, say that columns $C_{2i_1(\delta+1)}$ and $C_{2i_2(\delta+1)}$ intersect $\widetilde{\pi}(m_1)$ and $\widetilde{\pi}(m_2)$, respectively. First, if $i_1 \neq i_2$, then these columns are at least $2(\delta+1)$ apart, whence $\widetilde{\pi}(m_1)$ and $\widetilde{\pi}(m_2)$, both having span δ, cannot intersect. Say next that $i_1 = i_2 \stackrel{\text{def}}{=} i$. Since the buffered trajectories $\pi_B(m_1)$ and $\pi_B(m_2)$ reach column $C_{2i(\delta+1)}$ at τ_{m_1} and τ_{m_2}, respectively, we have $\tau_{m_1} \neq \tau_{m_2}$; hence, $\widetilde{\pi}(m_1)$ does not intersect $\widetilde{\pi}(m_2)$. Our bufferless schedule for S_0 is thus valid, whence $|OPT_{BL}(\mathcal{I})| \geq |S_0| \geq \frac{1}{2}|OPT_B(\mathcal{I})|$. ■

4.1.3 The Power of Buffers when Release Times are Uniform

Theorem 4.3 *If all messages in problem instance $\mathcal{I}$ have release time zero, then $OPT_B(\mathcal{I}) \leq 2 \cdot OPT_{BL}(\mathcal{I})$.*

Proof. Let Σ be any buffered schedule that specifies trajectories for a set of messages $\Sigma(\mathcal{I}) \subseteq \mathcal{I}$. We say that $m' \in \Sigma(\mathcal{I})$ *conflicts* with $m \in \Sigma(\mathcal{I})$ if (*a*) m' reaches its destination on the same scan line as m, and (*b*) $s_{m'} < d_m < d_{m'}$. Σ is a *single-conflict* schedule if for each $m \in \Sigma(\mathcal{I})$, there is at most one other $m' \in \Sigma(\mathcal{I})$ that conflicts with m.

Claim 1. *At least half the messages in any single-conflict buffered schedule Σ can be routed without using buffers.*

Verification. We filter the messages in $\Sigma(\mathcal{I})$, dropping some and routing others without buffers along the scan lines on which they reach their destinations under Σ. We select the messages to route greedily, by performing a sw-ne traversal of each scan line and scheduling a message iff it does not conflict with any previously scheduled message. By definition of single-conflict, each message that we schedule can block at most one other message, whence the Claim.

Claim 2. *Any* static *message-set can be optimally scheduled via a buffered single-conflict schedule Σ.*

Verification. We start with an optimal buffered schedule Σ^* for the messages in $OPT_B(\mathcal{I})$ and convert it in stages to a single-conflict schedule Σ. We process scan lines from left to right, rerouting some messages if necessary, to ensure the single-conflict property. When we reschedule a message m, we use only scan lines to the right of the current one and only messages whose destinations are to the right of d_m; therefore, a single left-to-right pass over the scan lines converts Σ^* to a single-conflict schedule Σ.

We describe a single iteration of the rerouting procedure. We convert a schedule Σ' under which message m has (potentially) multiple conflicts along scan line ℓ—call them $m_1, \ldots, m_k$, where $d_{m_1} < d_{m_2} < \cdots < d_{m_k}$—to a schedule Σ''' under which m has at most one conflict along ℓ. Say that $k \geq 2$ (or else no rerouting is needed). We transform Σ' to Σ''' in two steps.

Step 1. We reschedule m_k, routing it as before until it reaches d_m, and then routing it to d_{m_k} along ℓ.

Step 2. We use the space freed by delaying m_k to "push" all other messages on ℓ between d_m and d_{m_k}, including $m_1, m_2, \ldots, m_{k-1}$, to a scan line to the right of ℓ.

This procedure ensures that m conflicts along ℓ only with m_k.

We accomplish Step 2 as follows. Let q be the distance traveled by m_k along ℓ under Σ'. Denote by Σ'' the "schedule" obtained by performing Step 1 on Σ'. Now, Σ'' may not be a valid schedule: there may be a $v \in \{d_m, \ldots, d_{m_k} - q - 1\}$ such that more than one message moves along ℓ from v to $v+1$ at the same time. We transform Σ'' into a valid schedule Σ''' by removing all other messages on ℓ between d_m and $d_{m_k} - q$. We produce, in stages, a sequence of schedules, $\Sigma_{d_m}, \Sigma_{d_m+1}, \ldots, \Sigma_{d_{m_k}-q}$, such that

- $\Sigma_{d_m} = \Sigma''$, and $\Sigma_{d_{m_k}-q} = \Sigma'''$.
- Each Σ_v is valid along scan line ℓ up to node v, for $d_m \leq v \leq d_{m_k} - q$.

We exploit the static nature of $\mathcal{I}$ to transform each Σ_v to Σ_{v+1}. Say that $m' \neq m_k$ moves from v to $v+1$ along ℓ. (If no m' exists, then $\Sigma_{v+1} = \Sigma_v$.) We create Σ_{v+1} by altering Σ_v so that m' moves from v to $v+1$ on a scan line to the right of ℓ. Let ℓ_v be the scan line along which m_k moves from v to $v+1$ under Σ'. Note that under Σ_v, no message moves from node v to $v+1$ along ℓ_v.

Now, if $v = s_{m'}$, then since $t_{m'}^{(r)} = 0$, we can simply reschedule m' to travel from v to $v+1$ along ℓ_v, and we are done. If $v > s_{m'}$, then we must be more careful. Let ℓ' be the scan line along which m' moves from $v-1$ to v under Σ_v. If ℓ' is *not* to the left of ℓ_v, then we are done, since again we can simply reschedule m' from node v to $v+1$ along ℓ_v; however, we cannot do this if ℓ' lies to the left of ℓ_v. In this case, we reschedule m' from v to $v+1$ along ℓ'. If no message was previously routed from v to $v+1$ along ℓ', then again we are done. Otherwise, since Σ_v is valid up to v, and since m' is routed from $v-1$ to v along ℓ', some other message m'' must encounter ℓ' at v. We repeat the same process with m'', and we keep repeating until we reach either ℓ_v or any other scan line that does not forward a message from v to $v+1$ under Σ_v. With each new message, we arrive at least one scan line closer to ℓ_v; hence we reach either ℓ_v or some other empty scan line eventually. Claim 2 follows.

If we use Claim 2 to construct a single-conflict schedule for $OPT_B(\mathcal{I})$, then use Claim 1 to route at least half the messages in $OPT_B(\mathcal{I})$ without buffers, we achieve the theorem. ■

4.2 The General Comparison

We now derive a tight (to within constant factors) relationship between $|OPT_B(\mathcal{I})|$ and $|OPT_{BL}(\mathcal{I})|$ for arbitrary problem instances $\mathcal{I}$. We express our bounds in terms of the parameter $\Lambda(\mathcal{I}) \stackrel{\text{def}}{=} \min\{\sigma(\mathcal{I}),\ \delta(\mathcal{I}),\ |\mathcal{I}|\}$, where $\sigma(\mathcal{I}) \stackrel{\text{def}}{=} \max_{m\in\mathcal{I}}\{t_m^{(s)}\}$ is the *maximum slack* in $\mathcal{I}$, and $\delta(\mathcal{I}) \stackrel{\text{def}}{=} \max_{m\in\mathcal{I}}\{\delta_m\}$ is the *maximum span*.

Theorem 4.4 *For any problem instance $\mathcal{I}$,*

$$|OPT_B(\mathcal{I})| \geq \frac{1}{2}\log\Lambda(\mathcal{I}) \cdot |OPT_{BL}(\mathcal{I})|,$$

and

$$|OPT_B(\mathcal{I})| \leq 4(\log\Lambda(\mathcal{I}) + 1) \cdot |OPT_{BL}(\mathcal{I})|.$$

The upper bound in the theorem is a direct consequence of the following three lemmas. We prove the lower bound in Theorem 4.5.

Lemma 4.1 *For any problem instance $\mathcal{I}$,*

$$|OPT_B(\mathcal{I})| \leq 2(\ln(\sigma(\mathcal{I}) + 1) + 1) \cdot |OPT_{BL}(\mathcal{I})|.$$

Proof. We use the credit-distribution scheme of Theorem 4.1 to compare $|OPT_B(\mathcal{I})|$ with $|BFL(\mathcal{I})|$. This scheme has each $m \in OPT_B(\mathcal{I}) - BFL(\mathcal{I})$ donate $1/(t_m^{(s)}+1)$ units of credit to each of $t_m^{(s)}+1$ messages in $BFL(\mathcal{I})$ that collectively block all of m's potential bufferless trajectories. Clearly, a grand total of $|OPT_B(\mathcal{I}) - BFL(\mathcal{I})|$ credits are donated. Reversing our focus, we use the reasoning (and notation) of Theorem 4.1, tempered by the fact that slacks are not uniform here, to generate the following analogue of the upper bound (1) on the total credit received by any $m' \in BFL(\mathcal{I})$.

$$\sum_{m\in R_{m'}} \frac{1}{t_m^{(s)}+1} \leq \sum_{m\in R_{m'}} \frac{1}{|\tau_m - t| + 1}$$
$$\leq 1 + 2\sum_{i=2}^{\sigma(\mathcal{I})+1} \frac{1}{i} \leq 2\ln(\sigma(\mathcal{I}) + 1) + 1. \qquad (2)$$

(Recall that $R_{m'}$ comprises those messages that donate credit to m'.) Hence, the total number of credits received by messages in $BFL(\mathcal{I})$ is $\leq (2\ln(\sigma(\mathcal{I})+1)+1) \cdot |BFL(\mathcal{I})|$, which yields the lemma, because $|BFL(\mathcal{I})| \leq |OPT_{BL}(\mathcal{I})|$. ■

Lemma 4.2 *For any problem instance $\mathcal{I}$,*

$$|OPT_B(\mathcal{I})| \leq 2(\ln(\frac{1}{2}|\mathcal{I}|) + 1) \cdot |OPT_{BL}(\mathcal{I})|.$$

Proof. The proof is identical to that of Lemma 4.1, except that we bound the sum (2) differently. Using the facts that (*a*) $|R_{m'}| \leq |\mathcal{I}| - 1$ (since $m' \notin R_{m'}$), and (*b*) no two messages reach any node simultaneously, we find (using the notation of Theorem 4.1) that

$$\sum_{m\in R_{m'}} \frac{1}{|\tau_m - t| + 1} \leq 1 + 2\sum_{1<i\leq|\mathcal{I}|/2} \frac{1}{i} \leq 2\ln\left(\frac{1}{2}|\mathcal{I}|\right) + 1.$$

The lemma follows. ■

Lemma 4.3 *For any problem instance $\mathcal{I}$,*

$$|OPT_B(\mathcal{I})| \leq 4(\lfloor\log\delta(\mathcal{I})\rfloor + 1) \cdot |OPT_{BL}(\mathcal{I})|.$$

Proof. We partition $OPT_B(\mathcal{I})$ into sets R_i, $0 \leq i \leq \lfloor\log\delta(\mathcal{I})\rfloor$, by placing each m into $R_{\lfloor\log\delta_m\rfloor}$. Let R be an R_i of largest cardinality, so that

$$|R| \geq \frac{|OPT_B(\mathcal{I})|}{\lfloor\log\delta(\mathcal{I})\rfloor + 1}. \qquad (3)$$

Much as in Theorem 4.2, we show that at least $1/4$ of the messages in R can be routed without buffers. To this end, let δ be an integer such that for all $m \in R$, $\delta \leq \delta_m < 2\delta$. Divide R into four (possibly intersecting) sets S_j, $0 \leq j \leq 3$ by placing each m into S_j iff m's parallelogram intersects a column $C_{i(\delta+1)}$ of M_n, where $0 \leq i \leq \lfloor(n-1)/(\delta+1)\rfloor$ and $i \bmod 4 = j$. The S_j collectively cover R because each $m \in R$ intersects at least one $C_{i(\delta+1)}$. Let S denote the largest S_j, say S_0. We note that $|S| \geq \frac{1}{4}|R|$.

We conclude the proof by showing that all $m \in S$ can be routed without buffers. Say that m's buffered trajectory, $\pi_B(m)$, reaches a column $C_{4k(\delta+1)}$ at time τ_m. (By construction, such a column exists for each $m \in S$.) We assign m the bufferless trajectory $\widetilde{\pi}(m)$ that is the unique sw-ne segment that passes through point $\langle 4k(\delta+1), \tau_m\rangle$. We claim that the bufferless schedule $\{\widetilde{\pi}(m) \mid m \in S\}$ is valid. To this end, let k_1 and k_2 be integers such that columns $C_{4k_1(\delta+1)}$ and $C_{4k_2(\delta+1)}$ intersect $\widetilde{\pi}(m_1)$ and $\widetilde{\pi}(m_2)$, respectively. First, if $k_1 \neq k_2$, then columns $C_{4k_1(\delta+1)}$ and $C_{4k_2(\delta+1)}$ are at least $4(\delta+1)$ apart. Since $\widetilde{\pi}(m_1)$ and $\widetilde{\pi}(m_2)$ each have span $\leq 2\delta - 1$ each, the trajectories cannot intersect. Alternatively, if $k_1 = k_2 \stackrel{\text{def}}{=} k$, then the buffered trajectories $\pi_B(m_1)$ and $\pi_B(m_2)$ reach column $C_{4k(\delta+1)}$ at times τ_{m_1} and $\tau_{m_2} \neq \tau_{m_1}$, respectively so that $\widetilde{\pi}(m_1)$ and $\widetilde{\pi}(m_2)$ do not intersect. ■

Theorem 4.5 *There exists a problem instance $\mathcal{I}$ such that*

$$|OPT_B(\mathcal{I})| \geq \frac{1}{2}\log\Lambda(\mathcal{I}) \cdot |OPT_{BL}(\mathcal{I})|.$$

Proof. We construct the "bad" problem instance $\mathcal{I}$ recursively. The base instance $\mathcal{I}_0$ contains a single message m such that $s_m = 0$, $d_m = 1$, $t_m^{(r)} = 0$, and $t_m^{(d)} = 1$. Inductively, we construct instance $\mathcal{I}_k$ from $\mathcal{I}_{k-1}$, by setting $\mathcal{I}_k = S_k \cup \mathcal{I}_{k-1}^{(1)} \cup \mathcal{I}_{k-1}^{(2)}$, where: (*a*) S_k consists of 2^{k-1} identical messages, each having $s_m = 0$, $d_m = 2^k$, $t_m^{(r)} = 0$, and $t_m^{(d)} = 2^{k+1} - 1$; (*b*, *c*) $\mathcal{I}_{k-1}^{(1)}$ and $\mathcal{I}_{k-1}^{(2)}$ are obtained by translating two copies of $\mathcal{I}_{k-1}$, positioning the first with its "origin" at $\langle 0, 2^{k-1}\rangle \in M_n$ and the second with its "origin" at $\langle 2^{k-1}, 2^{k-1}\rangle \in M_n$. See Fig. 2.

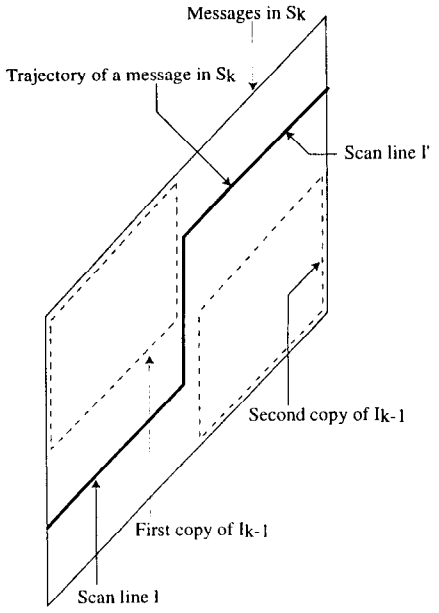

Figure 2: Problem instance $\mathcal{I}_k$.

Claim. *All messages in $\mathcal{I}_k$ can be routed using a buffered schedule, so that $|OPT_B(\mathcal{I}_k)| \geq k2^{k-1}$.*
Verification. We proceed by induction on k, the case $k = 0$ being obvious. Assuming the claim for $\mathcal{I}_{k-1}$, we route the messages in $\mathcal{I}_k$ as follows. We adapt the schedule for $\mathcal{I}_{k-1}$ to its non-overlapping copies, $\mathcal{I}_{k-1}^{(1)}$ and $\mathcal{I}_{k-1}^{(2)}$. We then route each $m \in S_k$ in turn, so that its trajectory does not conflict with any previously routed message: m starts at some time t, $0 \leq t < 2^{k-1}$, along a scan line ℓ that is not used by any other $m'' \in S_k$, until it reaches node 2^k; it then waits in node 2^k's buffer for 2^{k-1} steps, to reach a scan line ℓ' that is not used by any other $m'' \in S_k$; it finally travels along ℓ' to d_m (cf. Fig. 2). We thus have $|OPT_B(\mathcal{I}_k)| = |\mathcal{I}_k|$, whence, by construction,

$$|\mathcal{I}_k| = 2 \cdot |\mathcal{I}_{k-1}| + 2^{k-1}$$

for $k \geq 1$, with initial condition $|\mathcal{I}_0| = 1$. The Claim follows from the above recurrence relation.

We now bound $|OPT_{BL}(\mathcal{I}_k)|$. For each $0 \leq j \leq 2^k - 1$, let ℓ_j be the scan line that passes through point $\langle 0, j \rangle$. In our construction of $\mathcal{I}_k$, any bufferless message that uses scan line ℓ_j must be sent from point $\langle 2^k - j - 1, 2^k - 1 \rangle$ to point $\langle 2^k - j, 2^k \rangle$. Thus, no bufferless schedule for $\mathcal{I}_k$ can route more than one message along any ℓ_j. This means, however, that $|OPT_{BL}(\mathcal{I}_k)| \leq 2^k$. We now note that $\sigma(\mathcal{I}_k) = 2^k - 1$, $\delta(\mathcal{I}_k) = 2^k$, and $|\mathcal{I}_k| \geq k2^{k-1}$, so that $\Lambda(\mathcal{I}_k) = 2^k - 1$. Combining this fact with our bounds on $|OPT_B(\mathcal{I}_k)|$ and $|OPT_{BL}(\mathcal{I}_k)|$, we have the claimed inequality. ■

5 Message Scheduling with Buffers

In this section, we seek buffered scheduling algorithms that are maximal in throughput, making no attempt to limit the number of buffers. We show that maximizing throughput in this scenario is NP-hard (Section 5.1); however, we show that, surprisingly, a distributed and online buffered algorithm (which makes all routing decisions *locally*) can mimic the performance of the centralized and offline bufferless algorithm BFL—in both action and throughput (Section 5.2). Using our results in Section 4, we infer that our distributed and online algorithm achieves throughput to within a constant factor of optimal when slacks or spans or release times are uniform, and to within a logarithmic factor in general.

5.1 The NP-Hardness of Optimal Buffered Scheduling

Theorem 5.1 *The bufferless message-scheduling problem OPT_B is NP-hard.*

Proof. See Appendix A.

5.2 A Distributed and Online Approximation for Buffered Scheduling

A scheduling algorithm is *distributed* if only the source-node of a message initially receives information about each message to be sent. Information can be transmitted to other nodes, for example, when a message is forwarded to another node, but this information travels no faster than messages, i.e., across one link per time step. All decisions made by a node must be derived using only local information. In the algorithms discussed here, we transmit a small amount of additional information along with each packet. However, this additional information can be encoded in $\log n$ bits—which is exactly the number of bits required to describe message destinations. This does not increase the required packet size for messages by more than a factor of 2, and when the size of the message is much larger than the size of the destination description, this has no significant effect on the packet size. The scheduling algorithms we consider here are also *online*: no node has any information about a message m prior to $t_m^{(r)}$, the release time of m.

In our distributed and online algorithm, $D\text{-}BFL$, each node v maintains a list of messages that are available to be forwarded to node $v + 1$. Let P_{iv}^D be the set of messages that are available for transmission along scan line ℓ_i by node v. At time t, node v selects a message $m_{iv} \in P_{iv}^D$, for $i = v - t$, and forwards it along scan line ℓ_i. We now show how to make the choice of m_{iv} using only local information. For any algorithm A, let $L_{iv}(A)$ denote the *most recent endpoint* on ℓ_i in column C_v, i.e., the largest $v' \leq v$ such that some m with $d_m = v'$ reaches d_m along ℓ_i. In $D\text{-}BFL$, we ensure that each v always knows the value of $L_{iv}(D\text{-}BFL)$ at time step t, by always forwarding the current value along each scan line. This is the *only* additional information that $D\text{-}BFL$ forwards with messages.

Algorithm $D\text{-}BFL$

Each node v performs the following steps at time $t = v - i$.

1. Node v forwards $L_{iv}(D\text{-}BFL)$ to node $v + 1$ along ℓ_i.
2. Let S be the set of all messages m' such that $m' \in P_{*v}^D$ and $s_{m'} \geq L_{iv}(D\text{-}BFL)$. Node v selects a message m_{iv} such that m_{iv} has the nearest destination of any message in S (breaking ties as in BFL). The message m_{iv} is forwarded to node $v + 1$ along ℓ_i. Note that no message is forwarded if $S = \emptyset$.

Theorem 5.2 *For any problem instance $\mathcal{I}$, $D\text{-}BFL(\mathcal{I}) = BFL(\mathcal{I})$.*

Proof. Let $\ell_{DB}[m]$ (resp., $\ell_{BFL}[m]$) be the scan line along which $m \in D\text{-}BFL(\mathcal{I})$ (resp., $m \in BFL(\mathcal{I})$) reaches d_m. We claim that $\ell_{BFL}[m] \equiv \ell_{DB}[m]$ for all m.

Say that m_{iv} is *premature* if $d_{m_{iv}} = v + 1$ and $\ell_{DB}[m_{iv}] > \ell_{BFL}[m_{iv}]$, i.e., if, under $D\text{-}BFL$, m_{iv} is the final "hop" for a

message, and m_{iv} reaches $d_{m_{iv}}$ earlier under $D\text{-}BFL$ than under BFL. We say that m_{iv} is *preemptive* if there is an $m' \in P_{iv}^D$ such that $\ell_{BFL}[m'] = i$, but $m_{iv} \neq m'$. Note that if $\ell_{BFL}[m] \neq \ell_{DB}[m]$ for some message m, then some m_{iv} is either premature or preemptive. We wish to show that this cannot occur.

Let the pair (i, v) specify the segment within scan line ℓ_i, from column C_v to column C_{v+1}. For each pair (i, v), consider the pairs (k, w) defined by the three relations: (1) $i \leq k \leq i + v - 1$; (2) $0 \leq w \leq v$; (3) $(i, v) \neq (k, w)$. These pairs consist of all the segments in a region of M_n. When $i \geq 0$, this region is a right triangle, minus the segment that originates where $i = k$ and $v = w$. The legs of this triangle lie along the X-axis and column C_v; its hypotenuse is the portion of ℓ_i between the legs. When $i < 0$, the region has the same shape, except the portion of the triangle where $w < 0$ is not present. In either case, we call this the *triangle of influence of* (i, v).

Lemma 5.1 *If there is no pair (k, w) in the triangle of influence for (i, v) such that m_{kw} is either preemptive or premature, then m_{iv} is neither preemptive nor premature.*

Proof Let P_{iv}^N be the BFL-analogue of P_{iv}^D, comprising those messages that have not been scheduled by BFL prior to scan line ℓ_i, but that could be sent from v to $v + 1$ along ℓ_i (because their parallelograms contain the appropriate segment of ℓ_i). We prove the lemma via three claims, the first of which is immediate from the specification of BFL.

Claim 1. *For any $m \in P_{iv}^N$, if $\ell_{BFL}[m] \neq \ell_i$, then ℓ_i contains the right endpoint of some other message m' between nodes s_m and d_m.*

Claim 2. *If no m_{kw}, for a (k, w) in the triangle of influence for (i, v), is preemptive, then $P_{iv}^D \subseteq P_{iv}^N$.*

Verification. Any $m \in P_{iv}^D - P_{iv}^N$ must be sent by BFL along a scan line ℓ_k such that $i < k \leq i + v - 1$. Since m is sent on ℓ_k by BFL, and thus it must have a release time making it available at least as early as scan line ℓ_k. However, since $m \in P_{iv}^D$, message m must "pass through" scan line ℓ_k, and thus there must be some w, $s_m \leq w \leq v$ such that $m \in P_{kw}^D$, but m was not sent from node w on scan line k. However, this means that m_{kw} is preemptive. By hypothesis, no $m \in P_{iv}^D - P_{iv}^N$ can exist. ■

Claim 3. *If no m_{kw}, for a (k, w) in the triangle of influence for (i, v), is premature or preemptive, then $L_{iv}(BFL) = L_{iv}(D\text{-}BFL)$.*

Verification. Let m be the message with $d_m = L_{iv}(BFL)$ that reaches its destination on ℓ_i. Since no m_{kw} in the triangle of influence is premature or preemptive, m must reach d_m along ℓ_i under $D\text{-}BFL$ also. Thus, we need only show that no m' with $d_m < d_{m'} \leq v$ has $\ell_{DB}[m'] = i \neq \ell_{BFL}[m']$. Since no edge in the triangle of influence is preemptive, if $\ell_{DB}[m'] = i$, then we cannot have $\ell_{BFL}[m'] > i$. Also, since $m_{i(d_{m'}-1)}$ is not premature, we cannot have $\ell_{BFL}[m'] < i$. ■

To prove the lemma, we first show that m_{iv} is not preemptive. To this end, let m_{iv}^N be the message in $BFL(\mathcal{I})$ that was forwarded along ℓ_i from v to $v + 1$; m_{iv}^N is NULL if no message is sent. We need only show that, if m_{iv}^N is in P_{iv}^D and is not NULL, then $m_{iv}^N = m_{iv}$. In BFL, m_{iv}^N is the message in P_{iv}^N with the leftmost destination, that has a source after $L_{iv}(BFL)$. Furthermore, by Claim 3, under $D\text{-}BFL$ all messages in P_{iv}^D with a source preceding $L_{iv}(BFL)$ are removed from consideration. If $m_{iv}^N \in P_{iv}^D$, then m_{iv}^N is not removed. By Claim 2, if $m_{iv}^N \in P_{iv}^D$, then it must be one of the remaining messages with the leftmost destination. Since we break ties in the same manner as is in BFL, if $m_{iv}^N \in P_{iv}^D$, then $m_{iv} = m_{iv}^N$.

Lastly, we assume, for contradiction, that m_{iv} is premature. If so, it cannot be NULL. By Claim 2, then, $m_{iv} \in P_{iv}^N$. Moreover, by Claim 1, we must have $L_{iv}(BFL) > s_{m_{iv}}$. However, by Claim 3, $L_{iv}(D\text{-}BFL) > s_{m_{iv}}$ as well. This implies that m_{iv} is not delivered under $D\text{-}BFL$, which is a contradiction. ■

We show how this lemma implies that no m_{iv} is preemptive or premature, which in turn implies the theorem. We proceed inductively along rows of M_n (= time steps). For row $t = 0$: the triangle of influence for any pair (i, v) such that $t_{m_{iv}}^{(r)} = 0$ is empty, so the hypothesis of Lemma 5.1 is satisfied trivially; hence, no such m_{iv} is preemptive or premature. Now, assume inductively that for some time $t > 0$, every message forwarded at time $t' < t$ is neither preemptive nor premature. This implies that for any i and $v = i + t$, no m_{kw} in the triangle of influence for (i, v) is preemptive or premature. Then, by Lemma 5.1, no message forwarded at time t is preemptive or premature, which extends the induction. ■

References

[1] A. Acampora and S. Shah (1992): Multihop lightwave networks: a comparison of store-and-forward and hot-potato routing. *IEEE Trans. Commun. 40*, 1082-1090.

[2] M. Andrews, B. Awerbuch, A. Fernandez, J. Kleinberg, F.T. Leighton, Z. Liu (1996): Universal stability results for greedy contention-resolution protocols. *37th IEEE Symp. on Foundations of Computer Science*.

[3] M. Andrews, A. Fernandez, M. Harchol-Balter, F.T. Leighton, L. Zhang (1997): General dynamic routing with per-packet delay guarantees of O(distance + 1/session rate). *38th IEEE Symp. on Foundations of Computer Science*.

[4] I. Ben-Aroya, T. Eilam, A. Schuster (1995): Greedy hot-potato routing on the two-dimensional mesh. *Distr. Computing 9*, 3-19.

[5] S.N. Bhatt, G. Bilardi, G. Pucci, A.G. Ranade, A.L. Rosenberg, E.J. Schwabe (1996): On bufferless routing of variable-length messages in leveled networks. *IEEE Trans. Comp. 45*, 714-729.

[6] A. Borodin, J. Kleinberg, P. Raghavan, M. Sudan, D.P. Williamson (1996): Adversarial queuing theory. *28th ACM Symp. on Theory of Computing*.

[7] A.Z. Broder, A.M. Frieze, E. Upfal (1996): A general approach to dynamic packet routing with bounded buffers. *37th IEEE Symp. on Foundations of Computer Science*, 390-399.

[8] A.Z. Broder and E. Upfal (1996): Dynamic deflection routing on networks. *28th ACM Symp. on Theory of Computing*, 348-355.

[9] R. Cypher, F. Meyer auf der Heide, C. Scheideler, B. Vocking (1996): Universal algorithms for store-and-forward and wormhole routing. *28th ACM Symp. on Theory of Computing*.

[10] U. Feige and P. Raghavan (1992): Exact analysis of hot-potato routing. *33rd IEEE Symp. on Foundations of Computer Science*, 553-562.

[11] P. Fizzano, D. Karger, C. Stein, J. Wein (1994): Job scheduling in rings. *6th ACM Symp. on Parallel Algorithms and Architectures*, 210-219.

[12] M. Flammini and C. Scheideler (1997): Simple, efficient routing schemes for all-optical networks. *9th ACM Symp. on Parallel Algorithms and Architectures*, 170-179.

[13] R. Games, A. Kevsky, P. Krupp, L. Monk (1995): Real-time communications scheduling for massively parallel processors. *Real-Time Technology and Applications Symp.*, 76-85.

[14] M.R. Garey and D.S. Johnson (1979): *Computers and Intractability.* W.H. Freeman and Co., San Francisco.

[15] S.J. Golestani (1991): Congestion-free communication in high-speed packet networks. *IEEE Trans. Communications 39*, 1802-1812.

[16] M.C. Golumbic (1980): *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York.

[17] M. Harchol-Balter and D. Wolfe (1995): Bounding delays in packet-routing networks. *27th ACM Symp. on Theory of Computing*, 248-257.

[18] S. Hinrichs, C. Kosak, D.R. O'Hallaron, T.M. Stricker, R. Take (1994): An architecture for optimal all-to-all personalized communication. *6th ACM Symp. on Parallel Algorithms and Architectures*, 310-319.

[19] S.L. Johnsson and C.-T. Ho (1989): Optimum broadcasting and personalized communication in hypercubes. *IEEE Trans. Comp. 38*, 1249-1268.

[20] C. Kaklamanis, D. Krizanc, S.B. Rao (1993): Hot-potato routing on processor arrays. *5th ACM Symp. on Parallel Algorithms and Architectures*, 273- 282.

[21] J.H. Kim and A.A. Chien (1996): Rotating Combined Queuing (RCQ): Bandwidth and latency guarantees in low-cost, high-performance networks. *23rd Intl. Symp. on Computer Architecture*, 226-236.

[22] C. Lam, H. Jiang, V.C. Hamacher (1995): Design and analysis of hierarchical ring networks for shared-memory multiprocessors. *Intl. Conf. on Parallel Processing*, I:46-50.

[23] F.T. Leighton (1992): Methods for message routing in parallel machines (invited survey). *24th ACM Symp. on Theory of Computing*.

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

[25] F.T. Leighton, B.M. Maggs, A.G. Ranade, S.B. Rao (1994): Randomized routing and sorting on fixed-connection networks. *J. Algorithms 17*, 157-205.

[26] F.T. Leighton, F. Makedon, I. Tollis (1989): A $2N - 2$ step algorithm for routing in an $N \times N$ mesh. *1st ACM Symp. on Parallel Algorithms and Architectures*, 328-335.

[27] J. Liebeherr (1995): Multimedia networks: issues and challenges. *IEEE Computer 28* (4) 68-69.

[28] J.-P. Li and M.W. Mutka (1994): Priority based real-time communication for large scale wormhole networks. *Intl. Parallel Proc. Symp.*, 433-438.

[29] C.L. Liu and J.W. Layland (1973): Scheduling algorithms for multiprogramming in a hard real-time environment. *J. ACM 20*, 46-61.

[30] K.-S. Lui and S. Zaks (1998): Scheduling in synchronous networks and the greedy algorithm. *Theoretical Comp. Sci.*

[31] B.M. Maggs and R.K. Sitaraman (1992): Simple algorithms for routing on butterfly networks with bounded queues. *24th ACM Symp. on Theory of Computing*, 150-161.

[32] F. Meyer auf der Heide and B. Vocking (1995): A packet routing protocol for arbitrary networks. *Symp. on Theoretical Aspects of Computer Sci.*

[33] M.W. Mutka (1994): Using rate monotonic scheduling technology for real-time communications in a wormhole network. *Wkshp. on Parallel and Distr. Real-Time Computing Systs. and Applications.*

[34] R. Ostrovsky and Y. Rabani (1997): Universal (congestion + dilation $+ \log^{1+\epsilon} N$) local control packet switching algorithms. *29th ACM Symp. on Theory of Computing.*

[35] A.K. Parekh and R.G. Gallager (1993): A generalized processor sharing approach to flow control in integrated services networks: the single-node case. *IEEE/ACM Trans. Networking 1*, 344-357.

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

[37] A. Pietracaprina and F.P. Preparata (1995): Bufferless packet routing in high-speed networks. Typescript, Brown Univ.

[38] Y. Rabani and E. Tardos (1996): Distributed packet switching in arbitrary networks. *28th ACM Symp. on Theory of Computing.*

[39] P. Raghavan and E. Upfal (1994): Efficient routing in all-optical networks. *26th ACM Symp. on Theory of Computing.*

[40] A.G. Ranade (1991): How to emulate shared memory. *J. Comp. Syst. Scis. 42*, 307-326.

[41] J. Rexford, J. Hall, K.G. Shin. (1996): A router architecture for real-time point-to-point networks. *23rd Intl. Symp. Computer Architecture.*

[42] A. Saha (1995): Simulator for real-time parallel processing architectures. *IEEE Ann. Simulation Symp.*, 74-83.

[43] G.D. Stamoulis and J.N. Tsitsiklis (1991): The efficiency of greedy routing in hypercubes and butterflies. *3rd ACM Symp. on Parallel Algorithms and Architectures*, 248-259.

[44] T.H. Szymanski (1990): An analysis of hot-potato routing in a fiber-optic packet-switched hypercube. *IEEE INFOCOM* 918-926.

[45] L.G. Valiant (1982): A scheme for fast parallel communication. *SIAM J. Comput. 11*, 350-361.

[46] L.R. Welch and K. Toda (1994): Architectural support for real-time systems: issues and trade-offs. *Intl. Wkshp. on Real-Time Computing Systs. and Applications.*

[47] L. Zhang (1990): Virtual clock: A new traffic control algorithm for packet switching networks. *ACM SIGCOMM*, 19-29.

[48] W. Zhao, J.A. Stankovic, K. Ramamritham (1990): A window protocol for transmission of time-constrained messages. *IEEE Trans. Computers 39*, 1186-1203.

A The NP-Completeness Proofs

We reduce 3-SAT [14] to the problem of determining a maximum cardinality subset of messages that can be routed with or without buffers. Given any 3-SAT formula Φ, we show how to construct an equivalent time-constrained message routing problem $\mathcal{I}(\Phi)$ such that $OPT_B(\mathcal{I}(\Phi)) = OPT_{BL}(\mathcal{I}(\Phi))$. For this, we use three types of structures: a structure to represent variables of Φ, a structure to represent clauses of Φ, and a structure called a *chain* that is used as an interface between the two.

The structure for a variable x consists of two messages m_x and $m_{\bar{x}}$: one for each literal of x. These messages have span 2 and slack 0. Messages m_x and $m_{\bar{x}}$ must be sent on the same scan line, and overlap for one unit of the distance traveled. Thus, at most one of m_x and $m_{\bar{x}}$ can ever be routed successfully. In our construction, the message corresponding to the literal that is true is the message that is dropped. For all variables x, m_x and $m_{\bar{x}}$ are placed on the smallest numbered (leftmost) scan line in any arrangement where only the messages corresponding to the same variable overlap. Call the one unit of distance to be traveled by m_x and $m_{\bar{x}}$ where they do not overlap with the opposite literal the *critical time slot* of that message.

The critical time slot is used to interface with a chain. Each chain is associated with a single literal x (or $\bar{x}$). Each message in the chain has to travel the same 1 unit of distance as the critical time slot for x. In the simplest chains, there are k messages, each with slack k. The deadline for all of these messages is the same, and coincides with the critical time slot for x. This means that if the literal that the chain is associated with is false (i.e., message m_x has NOT been dropped) one message of the chain has to be routed as soon as the messages in the chain are released. We call this time slot the *bottom* of the chain. When the literal is true (i.e., message m_x is dropped), on the other hand, the bottom of the chain does not need to be occupied.

The bottom of each chain is connected to a clause structure. Each clause structure uses 6 consecutive scan lines $\ell_1 \ldots \ell_6$, and these scan lines are allocated only to a single clause structure. The structure is depicted in Fig. 3. This structure represents a clause composed of the literals A, B, and C, where A is the earliest of

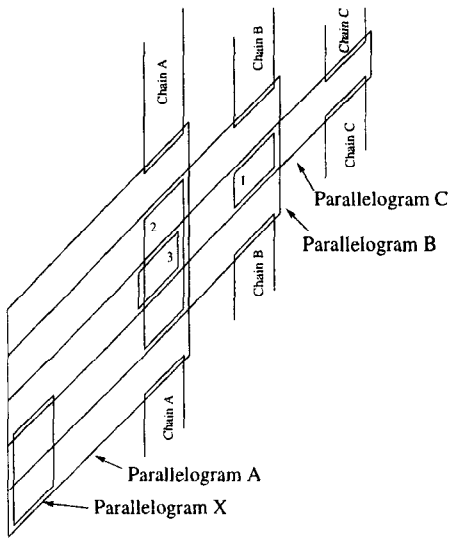

Figure 3: *The clause structure.*

the three literal in the linear order imposed by the ordering of the variables in the leftmost scan line, and C is the latest. In this structure, parallelogram p_A (resp. p_B and p_C) is lined up so that the upper right corner of its parallelogram exactly coincides with the bottom of the chain originating from message m_A (resp. messages m_B and m_C). The release times of these parallelograms, as well as all of parallelogram p_X, are all at a node to the left of all m_x. Parallelogram p_X is available on ℓ_1 and has slack 2. Parallelogram p_A is also available on ℓ_1, and has slack 5. Parallelogram p_B has slack 3 and is available on ℓ_2. Parallelogram p_C has slack 1, and is available on ℓ_3. Parallelogram p_1 is available on ℓ_3, has slack 1, and its source and destination are the same as the messages in the chain for B. Parallelograms p_2 and p_3 have the same source and destination as the messages in the chain for A. Parallelograms p_2 is available on ℓ_2 with slack 3, and p_3 is available on ℓ_3 with slack 1.

For each clause of Φ, there is one such structure in $\mathcal{I}(\Phi)$. When a literal Y appears in more than one clause, the chain is extended, starting at the lower right corner of the parallelogram Y. This extension to the chain interfaces with the next clause containing Y. The behavior of the chain extension mimics that of the original chain: if the bottom of the original chain is occupied, and all messages in the clause structure are successfully routed, then the bottom of the chain extension is also occupied. However, if the bottom of the original chain need not be occupied, then the bottom of the chain extension need not be occupied either.

It is also the case that some chains need to cross over a clause structure CS that does not contain the literal the chain represents. However, in all such cases, we know exactly how many messages for CS pass through the chain. To construct a chain of height k with j messages passing through the chain, we simply use $k - j$ messages with slack k. The resulting chain has the same properties as a chain without messages passing through it.

Let n be the total number of messages in $\mathcal{I}(\Phi)$, and let v be the number of variables in Φ. We first show that when $OPT_B(\mathcal{I}(\Phi)) = n - v$ or $OPT_{BL}(\mathcal{I}(\Phi)) = n - v$, then there must be a satisfying truth assignment for Φ.

Claim A. *If the messages in p_A, p_B, p_C, p_X, p_1, p_2 and p_3 can all be successfully routed (with or without buffers), then the bottom of at least one of the chains for A, B and C is not occupied.*

Verification. Note what happens when the bottom of chain C is occupied: the message for p_C must be routed by sending it entirely along the first scan line of its parallelogram, regardless of whether buffers are being used or not. Similarly, if the messages for p_C and p_1 are both successfully routed and the bottom of chain B is occupied, then the message for p_B must be routed along the earliest scan line of its parallelogram. Likewise, when the messages for p_B, p_C, p_2 and p_3 are all successfully routed and the bottom of chain A is occupied, then the message for p_A must be routed along the earliest scan line of its parallelogram. However, if the messages for p_A, p_B and p_C are all routed along the bottoms of their respective parallelograms, there is no room for the message in p_X to be routed. ■

For each variable x, at least one of m_x and $m_{\bar{x}}$ must be dropped, and thus if $OPT_B(\mathcal{I}(\Phi)) = n - v$, then for each variable x, exactly one of m_x and $m_{\bar{x}}$ is dropped, and no other messages are dropped. This and Claim A implies that every clause is connected to at least one chain D such that the bottom of D is not occupied. Thus, every clause is associated with at least one message m_D that has been dropped. By setting every literal D corresponding to a message m_D that has been dropped to true, we produce a satisfying assignment for Φ. The same holds if $OPT_{BL}(\mathcal{I}(\Phi)) = n - v$.

We complete the proof by showing that if there exists a satisfying truth assignment for Φ, then $OPT_B(\mathcal{I}(\Phi)) = OPT_{BL}(\mathcal{I}(\Phi)) = n - v$. Given a satisfying assignment for Φ, we route all but v of the messages by dropping only the messages corresponding to a true literal. We route all the messages in each chain at as late a time as possible. Since each clause has a true literal, this means that at least one of the chain bottoms in each clause structure is unoccupied. The message in the corresponding parallelogram can be routed along the last scan line in its parallelogram, and since at least one message is so routed in each clause structure, all messages in every clause structure can be successfully routed. ■