Real-Time Message Routing and Scheduling[⋆]

Ronald Koch, Britta Peis, Martin Skutella, and Andreas Wiese

TU Berlin, Institut für Mathematik, Straße des 17. Juni 136, 10623 Berlin, Germany
`www.math.tu-berlin.de/coga/`

Abstract. Exchanging messages between nodes of a network (e.g., embedded computers) is a fundamental issue in real-time systems involving critical routing and scheduling decisions. In order for messages to arrive on time, one has to determine a suitable (short) origin-destination path for each message and resolve conflicts between messages whose paths share a communication link of the network. We provide efficient routing strategies yielding origin-destination paths of bounded dilation and congestion. In particular, we can give good a priori guarantees on the time required to send a given set of messages which, under certain reasonable conditions, implies that all messages can be scheduled to reach their destination on time. Our algorithm uses a path-based LP-relaxation and iterative rounding. Finally, for message routing along a directed path (which is already $\mathcal{NP}$-hard), we identify a natural class of instances for which a simple scheduling heuristic yields provably optimal solutions.

1 Introduction

In a distributed real-time system, processes residing at different nodes of the network communicate by passing messages. One of the most challenging and important tasks for the design of a distributed system is the problem of sending a given set of messages through the network from the respective origin- to the destination nodes on time.

The message routing problem. To model the problem we represent the communication network by a (directed or undirected) graph $G = (V, E)$, whose edges correspond to the communication links of the network. In the *message routing problem*, each message $M_i = (s_i, t_i, d_i)$ of a given set of messages $\{M_i\}_{i \in I}$ consists of d_i packets of unit size that have to be sent from the origin node $s_i \in V$ to the destination node $t_i \in V$ within a certain time horizon $T > 0$. Usual constraints are (see e.g., [1,2], or [3, Chapter 37]):

(i) it takes one time unit to send a packet on any edge $e \in E$,
(ii) at most one packet can traverse an edge per time unit,
(iii) a message has to be completely received by a node before the node can start to transmit it to any other node.

⋆ This work was partially supported by Berlin Mathematical School, by DFG research center MATHEON and by the DFG Focus Program 1307 within the project "Algorithm Engineering for Real-time Scheduling and Routing".

The last constraint is due to integrity checks performed by each node and implies that each message M_i has to be sent along a unique path P_i from its origin to its destination node.

Example 1. Consider the problem illustrated in Figure 1 where three messages need to be routed through a grid graph within a time horizon of twelve time units. Suppose we decide to send each message along the (unique) shortest path. Then, after three time steps there is a conflict between the second packet of message 1 and the first packet of message 2 that both want to traverse edge e in time step four. No matter which message is assigned a higher priority, we need at least 13 time steps to send all message from their sources to their destinations. On the other hand, if we choose the longer path $\{a, b, c, d, f, g\}$ for message 1, all messages can be sent within twelve time units since all paths are edge-disjoint.

Store-and-forward packet routing. In the special case where each message consists of only one packet, message routing reduces to *store-and-forward packet routing*, a fundamental routing problem in interconnection networks (see, e.g., Leighton's survey [4]). Store-and-forward packet routing can be formulated as an *integral dynamic multicommodity flow problem* with unit capacities and unit transit times on the edges. While this problem is known to be $\mathcal{NP}$-hard [5], store-and-forward packet routing can be solved efficiently by calculating a maximum flow over time in case all packets share the same origin and destination. In contrast, the message routing problem turns out to be $\mathcal{NP}$-hard even in the special case where all messages have the same origin and destination [1]. Thus, message routing is considerably harder than packet routing.

We would like to mention that the possibility of storing packets is crucial in the message routing model we consider, since packets need to wait at intermediate nodes for the entire message. Therefore, our problem considerably differs from the well-studied *direct routing problem* in which the packets are not allowed to be stored at intermediate nodes on the way to their destination.

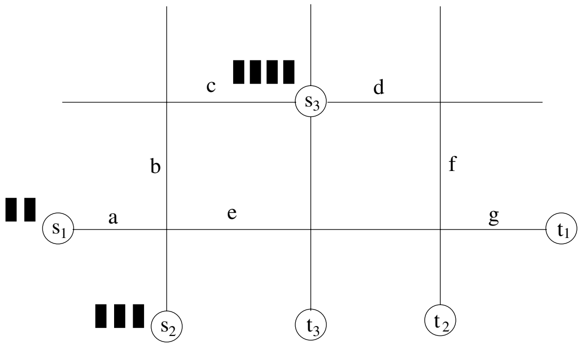

Fig. 1. Message routing problem with three messages and time horizon twelve. The messages consist of two, three, and four packets, respectively.

Routing and scheduling. A natural approach for solving the message routing problem is the following two-stage strategy. In the first stage (the *routing stage*), determine the set of paths $\{P_i\}_{i \in I}$. Then, in the second stage (the *scheduling stage*), resolve conflicts between messages sharing an edge. Of course, in order to determine good solutions, the paths chosen in the routing stage must feature certain desirable properties that guarantee the existence of good solutions to the second stage scheduling problem.

Congestion and dilation. If the paths $\{P_i\}_{i \in I}$ are given, we immediately obtain two trivial lower bounds on the minimum amount of time needed to send all messages, which we call the *makespan* of the problem. The first one is the *congestion*

$$C = \max_{e \in E} \sum_{i \in I : e \in P_i} d_i,$$

i.e., the maximum number of packets that have to traverse a single edge. The second one is the *dilation*

$$D = \max_{i \in I}(d_i |P_i|),$$

i.e., the maximum time necessary to send a message without any delays from its origin to its destination. As usual, $|P_i|$ denotes the number of edges in path P_i. As we will see in the following, C and D not only provide lower bounds on the makespan but also good upper bounds in terms of C and D can be determined.

A related job shop scheduling problem. Given paths $\{P_i\}_{i \in I}$, it remains to declare priorities on the messages whenever two packets of different messages meet at an intermediate node and want to use the same outgoing edge. However, this is exactly an instance of the well-studied *acyclic preemptive job shop scheduling problem.* Every edge corresponds to a machine and a message is a job that has to be consecutively processed on the machines corresponding to the edges on its path. In shop scheduling, the processing requirement of a job is usually machine-dependent. In our case, however, we have the special property that the processing requirement of a job/message is identical (namely equal to the size of the message) on each machine/edge on its path.

It is well-known that even this special case of acyclic preemptive job shop scheduling is $\mathcal{NP}$-hard and even $\mathcal{NP}$-hard to approximate[1] with performance guarantee $\frac{5}{4} - \epsilon$ for any $\epsilon > 0$ [6]. On the positive side, Feige and Scheideler [7] prove the existence of a schedule with makespan $\mathcal{O}(C + D \log\log d_{\max})$ for the preemptive job shop scheduling problem in general by using the non-constructive General Lovász Local Lemma (LLL). (Here, $d_{\max}$ denotes the maximum operation-length, resp. message-size.) An algorithmic version of the General LLL can be found in [8]. In the special case where all operation lengths

[1] An *α-approximation algorithm* for an optimization problem is a polynomial-time algorithm which computes a solution whose value is at most a factor α away from the optimum. The number α is called the performance guarantee of the algorithm.

are identical, Leighton et al. [9,10] even establish an efficient randomized algorithm which computes a schedule with makespan $\mathcal{O}(C + D)$. Busch, Magdon-Ismail, and Mavronicolas [11] prove that intermediate storage of packets can be avoided at the cost of an additional poly-logarithmic factor in the makespan.

Desirable properties of paths. We now return to our discussion of the two-stage approach to message routing discussed above. As a consequence of the scheduling results mentioned in the previous paragraph, a promising approach is to determine a set of paths in the routing stage such that $C + D$ is relatively small. The first constant-factor approximation for the special case of store-and-forward packet routing, established by Srinivasan and Teo [12], is also based on this idea. Basically, Srinivasan and Teo establish a constant-factor approximation for the problem to find paths minimizing $C + D$. Combining this result with the $\mathcal{O}(C + D)$-schedule for acyclic job shop scheduling with constant operation lengths (proved in [9,10]), they obtain a constant-factor approximation for packet routing. A similar idea has been used by Fleischer and Skutella [13] in the general context of dynamic network flow problems.

Our contributions. In Section 2 we describe an algorithm that, given a set of messages $\{M_i\}_{i\in I}$ on a communication network, and a desired dilation $\varDelta$, finds a set of paths of dilation at most $\varDelta$ and congestion smaller than $C^*(\varDelta)+\varDelta$, where $C^*(\varDelta)$ denotes the congestion of an optimal fractional solution with dilation at most $\varDelta$. The dilation $\varDelta$ that is given to the algorithm as an input can be chosen arbitrarily (e.g., $\varDelta = T/2$). Of course, the smaller the dilation $\varDelta$ is, the larger is the optimal congestion $C^*(\varDelta)$. In practice it is thus reasonable to try several values of $\varDelta \leq T$ in order to find a good tradeoff between dilation and congestion. In theory, one can, for example, use binary search in order to determine $\varDelta$ such that $\varDelta + C^*(\varDelta)$ or $\varDelta + (C^*(\varDelta) + \varDelta)$ (or some other function of $\varDelta$ and $C^*(\varDelta)$) is minimal.

Although our algorithm can be applied for arbitrary message lengths, it even improves upon the performance guarantee of [12] for the special case of store-and-forward packet routing by a multiplicative factor of two. The main difference between our approach and the approach in [12] is our use of a path-based linear programming formulation which turns out to be efficiently solvable as the corresponding separation problem is a special case of the *length-bounded shortest path problem.* (The latter can be solved with a modification of Dijkstra's algorithm). Given an optimal solution to the linear program, we apply iterative rounding to turn the fractional solution into an integral one, and guarantee that the congestion is not increased by more than $\varDelta$.

Our path-finding algorithm works for arbitrary directed or undirected graphs. Combined with either approximation algorithms for the acyclic job shop scheduling problem, or with suitable priority heuristics, it therefore returns solutions for the message routing problem in general. In many situations in practice, however, the communication graphs are very simple. It therefore makes sense to consider the problem on special graph classes. In Section 3 we consider the message routing problem on directed paths (which is already $\mathcal{NP}$-hard [1]), and show that the *Farthest-Destination-First Algorithm* works optimally on a directed path P in case the messages are not nested, i.e., in case

$$s_i <_P s_j \quad \Longrightarrow \quad t_i \leq_P t_j \quad \forall i, j \in I.$$

2 Routing with Small Congestion and Dilation

Note that any set of edge-disjoint paths $\{P_i\}_{i \in I}$, where the length of each path P_i is bounded by $\frac{T}{d_i}$, forms a solution to the message routing problem: all messages can be sent directly without any delay from their origin to their destination nodes where they arrive before time T. Of course, such length-bounded edge-disjoint paths do not necessarily exist (it is $\mathcal{NP}$-hard to decide whether they do exist or not [14]). However, some delays are allowed if the path-lengths do not meet the upper bounds $(\frac{T}{d_i})_{i \in I}$. Thus, we restrict to shorter paths on which we minimize the congestion.

Given a suitable value $\Delta \leq T$ (which can, for example, be determined by binary search), we define for each $i \in I$ the set of paths

$$\mathcal{P}_i := \left\{ s_i, t_i\text{-paths in } G \text{ of length at most } \frac{\Delta}{d_i} \right\}$$

and $\mathcal{P} := \bigcup_{i \in I} \mathcal{P}_i$. Among $\mathcal{P}$, we are looking for a set of representatives $\{P_i\}_{i \in I}$ with minimal congestion. That is, we are interested in an optimal integral solution to the following linear program

$$\begin{aligned}
\min\ & C \\
\text{s.t.}\ & \sum_{P \in \mathcal{P}_i} x_P \geq 1 && \forall i \in I, \\
& \sum_{i \in I} \sum_{P \in \mathcal{P}_i : e \in P} d_i x_P \leq C && \forall e \in E, \\
& x_P \geq 0 && \forall P \in \mathcal{P}.
\end{aligned}$$

Note that the paths in the support of any feasible integral solution $\hat{x} \in \{0,1\}^{|\mathcal{P}|}$ of the linear program above with objective value $\hat{C}$ yield a set of representatives $\{P_i\}_{i \in I}$ with dilation at most Δ and congestion $\hat{C}$: the first set of constraints ensures that at least one path is found for each message, while the second set of constraints guarantees that the total number of packets traversing a single edge does not exceed $\hat{C}$.

2.1 Optimal Fractional Solutions

To find a good integral solution to the linear program above, we first determine an optimal fractional solution x^* with objective value C^*, and then, in a second step, round x^* to an integral solution $\hat{x} \in \{0,1\}^{|\mathcal{P}|}$ whose congestion is at most $C^* + \Delta$. At first sight, it seems to be impossible to find an optimal fractional solution in polynomial time, since the number of variables is in general exponential in the size of the underlying network G. However, if we consider the dual linear program, we get

$$\begin{aligned}
\max\ & \sum_{i \in I} z_i \\
\text{s.t.}\ & \sum_{e \in E} y_e \leq 1 \\
& \sum_{e \in P} y_e \geq \frac{z_i}{d_i} && \forall P \in \mathcal{P}_i, i \in I \\
& y_e, z_i \geq 0 && \forall e \in E, i \in I.
\end{aligned}$$

The corresponding separation problem can be formulated as a *length-bounded shortest path problem*: find a shortest s_i, t_i-path with respect to the edge costs y_e among those paths containing at most $\frac{\Delta}{d_i}$ edges. In contrast to the general length-bounded shortest path problem with arbitrary edge lengths (which is known to be $\mathcal{NP}$-hard [14]), this problem can be solved efficiently with a modification of Dijkstra's algorithm (*sketch:* in each iteration of Dijkstra's algorithm determine a shortest path among those with at most $1, 2, \ldots$ edges). Thus, by the equivalence of optimization and separation [15], an optimal fractional solution to the dual and thus also to the primal linear program can be found in polynomial time. (I.e., we do not need to consider all path-variables in the LP. Instead, we iteratively solve the LP for small subsets of variables, where in each step a variable corresponding to the shortest length-bounded path is added to the LP in case the reduced costs are negative.) In practice, column generation seems to be the most suitable technique to actually solve the primal linear programming problem.

2.2 Iterative Rounding

Given the upper bound Δ on the dilation of paths and an optimal fractional solution x^* with objective value C^* to the corresponding linear program, we now describe how to round the fractional solution to an integral one while increasing congestion at most by Δ.

In the rounding algorithm described below, we iteratively solve a linear programming relaxation and fix a path P_i for message i as soon as the corresponding variable x_{P_i} attains value 1. In the following, F is the set of those messages i for which a path P_i has already been fixed. Initially, F is empty. The messages in F are removed from I such that I only contains the messages for which a path remains to be fixed. In each step of the algorithm, we thus solve the following linear program (LP):

$$
\begin{aligned}
&\min\ C && && \\
&\sum_{P\in\mathcal{P}_i} x_P \geq 1 && \forall i\in I && (1)\\
&\sum_{i\in I}\ \sum_{P\in\mathcal{P}_i: e\in P} d_i x_P \leq C - \sum_{i\in F: e\in P_i} d_i && \forall e\in E && (2)\\
&x_P \geq 0 && \forall P\in\mathcal{P}. &&
\end{aligned}
$$

The basic idea of the algorithm is as follows: in each iteration, we fix the integral variables and drop at least one of the constraints, before we solve the (LP) again. That is, in each iteration, whenever there is an index i with $x^*_P = 1$ for some $P \in \mathcal{P}_i$, we move index i from I to F. Moreover, we remove all paths not in the support of x^* from $\mathcal{P}$. After fixing the integral variables, we can easily find a constraint of type (2) which can be dropped from the updated (LP): the reason is that even if all remaining variables are rounded up to 1, the right-hand side of the inequality is not violated by more than Δ (see Theorem 1).

Algorithm 1 (Iterative Rounding Algorithm)

```
1. Initialize: $F \leftarrow \emptyset$;
2. Compute a basic optimum solution $x^*$ to $(LP)$;
3. For $i \in I$, let $\mathcal{P}_i \leftarrow \{P \in \mathcal{P}_i \mid x^*_P > 0\}$;
4. WHILE $\exists i \in I$ and $P_i \in \mathcal{P}_i$ with $x^*(P_i) = 1$ DO
     – Set $I \leftarrow I \setminus \{i\}$;
     – Set $F \leftarrow F \cup \{i\}$;
5. Set $\mathcal{P} \leftarrow \bigcup_{i\in I} \mathcal{P}_i$;
6. WHILE $\mathcal{P} \neq \emptyset$ DO
     – Drop a constraint of type (2) with
         $\sum_{i\in I} \sum_{P\in\mathcal{P}_i: e\in P} d_i < C^* - \sum_{i\in F: e\in P_i} d_i + \Delta$;
     – GoTo step 2;
```

Note that in a single iteration of our algorithm, we do not round fractional variables explicitly, but simply fix the integral variables. The "rounding" is thus done by solving in each iteration the modified linear program corresponding to the remaining fractional variables. It remains to show that the algorithm is well-defined, i.e., we need to show the following: in case the set $\mathcal{P}$ of non-integral components is non-empty, we can find an edge $e \in E$ such that the congestion cannot be violated by more than Δ, even if all non-integral components are rounded up to one.

Theorem 1. *If x^* is a basic optimum solution to (LP) with $0 < x^*_P < 1$ for all $P \in \mathcal{P}$, then there exists a constraint of type (2) such that for the corresponding edge $e \in E$ holds*

$$\sum_{i\in I}\ \sum_{P\in\mathcal{P}_i: e\in P} d_i < C^* - \sum_{i\in F: e\in P_i} d_i + \Delta.$$

The theorem can be derived from a more general result shown in [16], stating that any fractional solution x^* of a linear equality system $Ax = b$ can be rounded to an integral vector $\hat{x}$ satisfying $A\hat{x} < b + \Delta$, whenever the sum of positive entries in each column of matrix A is bounded from above by Δ, and the sum of negative entries in each column is bounded from below by $-\Delta$. However, the proof turns out to be much simpler for our special inequality system:

Proof. Let $n = |\mathcal{P}|$. Since x^* is a basic feasible solution, there exist linearly independent tight constraints $\mathcal{T}_1$ and $\mathcal{T}_2$ of type (1) and (2), respectively, such that

$$n = |\mathcal{T}_1| + |\mathcal{T}_2|.$$

Observe that for each constraint $j \in \mathcal{T}_1$ we have

$$\Delta \sum_{P \in \mathcal{P}_j} x^*_P = \Delta. \tag{3}$$

Suppose by contradiction that for each e corresponding to a constraint in $\mathcal{T}_2$, we have

$$\sum_{i \in I} \sum_{P \in \mathcal{P}_i : e \in P} d_i \geq C^* - \sum_{i \in F : e \in P_i} d_i + \Delta. \tag{4}$$

Since

$$\sum_{i \in I} \sum_{P \in \mathcal{P}_i : e \in P} d_i x^*_P = C^* - \sum_{i \in F : e \in P_i} d_i$$

holds by the tightness of the constraint, equation (4) turns out to be equivalent to

$$\sum_{i \in I} \sum_{P \in \mathcal{P}_i : e \in P} d_i (1 - x^*_P) \geq \Delta. \tag{5}$$

Summing up the inequalities of type (3) and (5) for all constraints in $\mathcal{T}_1$ and $\mathcal{T}_2$, we get

$$\begin{aligned} n\Delta &\leq \sum_{j \in \mathcal{T}_1} \Delta \sum_{P \in \mathcal{P}_j} x^*_P + \sum_{e \in \mathcal{T}_2} \sum_{i \in I} \sum_{P \in \mathcal{P}_i : e \in P} d_i (1 - x^*_P) \\ &= \sum_{i \in I} \sum_{P \in \mathcal{P}_i} \Big(\chi_i^{\mathcal{T}_1} \Delta x^*_P + \sum_{e \in \mathcal{T}_2 \cap P} d_i (1 - x^*_P) \Big) \\ &\leq \sum_{i \in I} \sum_{P \in \mathcal{P}_i} \big(\Delta x^*_P + \Delta (1 - x^*_P) \big) = n\Delta, \end{aligned}$$

where $\chi_i^{\mathcal{T}_1} \in \{0, 1\}$ is an indicator variable with $\chi_i^{\mathcal{T}_1} = 1$ iff $i \in \mathcal{T}_1$. Since $0 < x^*_P < 1$ for all $P \in \mathcal{P}$, the second inequality in the derivation above is an equality only if for all $i \in I$ and all paths $P \in \mathcal{P}_i$ the following two conditions are satisfied.

1. $\chi_i^{\mathcal{T}_1} = 1$, and
2. $\sum_{e \in \mathcal{T}_2 \cap P} d_i = \Delta$.

If we now consider each column of (LP) separately, add the column's entries corresponding to constraints of type (2) and subtract the column's entries corresponding to constraints of type (1), we achieve a result of 0 in each column. This demonstrates that $\mathcal{T}_1$ and $\mathcal{T}_2$ must be linearly dependent constraints. A contradiction! $\square$

Thus, after at most $|E|$ iterations, the algorithm terminates with an integral vector $\hat{x} \in \{0,1\}^{|\mathcal{P}|}$, whose support contains a path P_i for each message $i \in I$. It is guaranteed that each path P_i does not contain more than $\frac{\Delta}{d_i}$ edges, and that the congestion of the paths violates the congestion of the optimal fractional solution by at most Δ.

Corollary 1. *Given Δ, the rounding algorithm determines a set of paths $\{P_i\}_{i\in I}$ with dilation $\leq \Delta$ and congestion $\leq C^* + \Delta$, where C^* is the minimum possible congestion of fractional paths with dilation Δ.*

2.3 Individual Deadlines

In a more general model of the message routing problem, each message M_i is additionally equipped with a certain deadline $D_i > 0$, denoting the latest point in time when the message must be received by the destination node t_i. We want to emphasize that our algorithm might as well be applied in this more general setting: we simply restrict the path lengths with respect to the deadlines. That is, instead of choosing a value Δ which is not greater than the overall time horizon T, we choose a factor $q \in (0,1]$ and consider for each message M_i the collection of paths

$$\mathcal{P}_i := \left\{ s_i, t_i\text{-paths in } G \text{ of length at most } q\frac{D_i}{d_i} \right\}.$$

This guarantees a dilation of at most

$$\Delta = \max_{i\in I} qD_i$$

and a congestion of at most $C^* + \Delta$.

2.4 Arbitrary Travel Times

The algorithm above can also be applied in a further extension of the message routing problem, where travel times $\tau(e) \in \mathbb{N}_{>0}$ are associated with all edges $e \in E$. Here $\tau(e)$ denotes the time it takes for one packet to traverse e. Thus, a message of size d_i completely traverses edge e in $\tau(e)+d_i-1$ time units. Further, if message $i \in I$ is to be sent along path P_i, it takes at least

$$\tau^i(P_i) := \sum_{e\in P_i} (d_i + \tau(e) - 1)$$

time steps before the message is completely received by its destination node t_i. These observations show that the dilation for a given set of paths $\{P_i\}_{i\in I}$ in this more general model becomes

$$D := \max_{i\in I} \tau^i(P_i),$$

while the congestion $C = \max_{e\in E} \sum_{i\in I: e\in P_i} d_i$ remains unchanged. Note that we can adopt our algorithm to handle travel times by defining for a given value $\Delta \leq T$ the collections of paths

$$\mathcal{P}_i := \{s_i, t_i\text{-paths with } \tau^i(P) \leq \Delta\} \quad \forall i \in I.$$

However, with arbitrary travel times, the corresponding separation problem to our linear relaxation (LP) is the *general* length-bounded shortest path problem. While this problem is $\mathcal{NP}$-hard, it can be solved approximately in the following sense: for any $\epsilon > 0$, one can find in time polynomial in the size of the network G and $\frac{1}{\epsilon}$ an s_i, t_i-path P with $\tau^i(P) \leq (1+\epsilon)\Delta$ whose cost is bounded from above by the cost of a shortest path in $\mathcal{P}^i$ [17,18,19]. As before, the fractional solution (which is now a $(1+\epsilon)$-approximation to the optimal one) can be turned into an integral solution with the rounding algorithm described above, since the inequality $\sum_{e\in P} d_i \leq \Delta$ still holds for each path $P \in \mathcal{P}_i$ and $i \in I$. Thus, we achieve the following result.

Corollary 2. *Even if each edge $e \in E$ is equipped with a travel time $\tau(e) \in \mathbb{N}_{>0}$, a slight modification of the algorithm above returns a set of paths whose dilation is bounded by $(1+\epsilon)\Delta$ and whose congestion differs from the optimal congestion by an additive factor of at most $(1+\epsilon)\Delta$. Here, $\epsilon > 0$ can be chosen arbitrarily small.*

Given the set of paths $\{P_i\}_{i\in I}$ with congestion C and dilation D, the remaining problem of determining priority rules in order to minimize the makespan, can again be formulated as an acyclic job shop scheduling problem: to incorporate the travel times, we simply define for each message $i \in I$ and each edge $e \in E$ with $e \in P_i$ an additional machine e^i. After job i has been executed on machine e for d_i time steps, it needs to be processed on machine e^i for $\tau(e)-1$ time steps, before it can proceed to the next machine corresponding to the successive edge of e in P_i.

Note that processing times in the resulting acyclic job shop scheduling problem depend on both, the job and the machine. However, as already mentioned in the introduction, schedules of length $\mathcal{O}(C + D \log\log \ell_{\max})$ can be found for this more general problem. (In our model, $\ell_{\max}$ denotes the maximum of all travel times and message sizes).

3 Message Routing on Paths

In this section we consider instances of the message routing problem where the underlying network is a directed path. Since the path taken by any message is unique on such instances, no routing decisions but only scheduling decisions have to be taken. That is, an algorithm for the message routing problem must only resolve conflicts if two messages want to traverse the same edge at the same point in time. This can be done by assigning priorities to the messages such that a message with higher priority is sent first. More precisely, even if a message is currently being sent while a message with a strictly higher priority arrives, the latter message is sent instantaneously. Thus, an interruption of the message of lower priority occurs.

The following example illustrates that a wrong choice of a priority rule can lead to arbitrarily bad schedules.

Example 2. Suppose n messages $\{M_i\}_{i=0}^{n-1}$ start at the same origin node and need to be sent along a directed path. Each message M_i consists of $d_i = 2^i$ packets and needs to traverse 2^{n-i} edges before it reaches its destination. First we consider a schedule, where messages with farther destination get a higher priority. In order to send message i we wait at the origin until the first $i-1$ messages are sent and then traverse the path without any additional delay. Thus message i arrives at its destination at time $\sum_{k=0}^{i-1} 2^k + 2^i \cdot 2^{n-i} \leq 2^{n+1}$. Therefore the optimal makespan is at most 2^{n+1}.

In contrast, we next consider a schedule where messages with farther destination are assigned lower priorities. Then the makespan is determined by the completion time of the smallest message 0. Furthermore, any message i is sent without additional delay on its last $2^{n-i} - 2^{n-i-1} + 1$ edges and each message smaller than i is sent immediately after i on these edges. Thus each messages i adds at least $(2^{n-i} - 2^{n-i-1})2^i = 2^n - 2^{n-1}$ time units to the completion time of message 0. Thus the makespan of this schedule is at least $n(2^n - 2^{n-1}) = \frac{n}{4}2^{n+1}$. This shows that the gap to the optimal makespan can grow linearly in the number of messages.

In this example the *Farthest-Destination-First Algorithm* (FDFA for short) leads to an optimal schedule. FDFA assigns a higher priority to messages which have a farther destination according to the order of the underlying path. In case of ties, messages with a later origin node get higher priority. If both origin and destination of two messages coincide, ties are broken arbitrarily.

FDFA seems to be a good choice for the message routing problem on directed paths in general. But, since the problem is known to be $\mathcal{NP}$-hard [1], there surely exist examples where FDFA is not optimal:

Example 3. Consider a directed path consisting of four edges and three messages 1, 2, and 3. Message 1 must be sent from the first to the last edge and has size 1, whereas messages 2 and 3 must be sent from the second to the third edge and have both size $1 + \epsilon$ for small enough $\epsilon > 0$ (see Figure 2).

Then the optimum solution has a makespan of $4+2\epsilon$ and the solution of FDFA has a makespan of $5 + 2\epsilon$. Thus the performance guarantee of FDFA cannot be better than $\frac{5}{4}$.

In this section, we identify a large class of problems where FDFA is guaranteed to be optimal. But before, let us introduce some notation. For a message routing

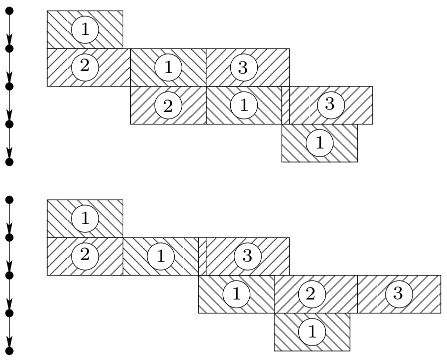

Fig. 2. Schedules of Example 3 showing that the approximation ratio of FDFA is not less than $\frac{5}{4}$. The optimum schedule is illustrated above the FDFA-schedule.

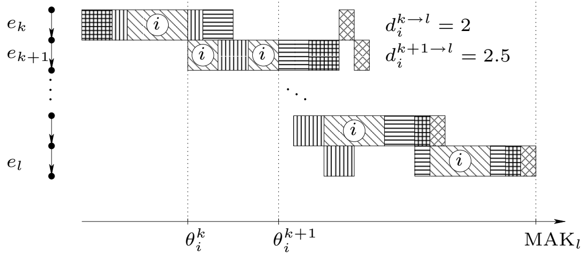

Fig. 3. Setting of Lemma 1

instance the underlying directed path P is given by node set $V(P) := \{v_1, \ldots, v_n\}$ and edge set $E(P) := \{e_k := (v_k, v_{k+1}) \mid k = 1, \ldots, n-1\}$. We say that a message *experiences additional delay* or *is additionally delayed* on edge e in a given schedule, if the starting time of i on e is strictly greater than the end time of i on the predecessor edge. The *makespan* on an edge e is the earliest point in time when all its messages have been sent through e. A time interval where no message traverses a particular edge is called *idle time*. (The infinitely long time interval after the makespan of an edge is not called idle time).

We show that the Farthest-Destination-First algorithm computes an optimum solution on non-nested instances. For this we need improved bounds on the minimum makespan combining dilation and congestion.

Lemma 1. *Consider an arbitrary feasible schedule. Let $e_k, e_l \in E(P)$ with $k \leq l$ be two edges of P and $i \in I$ be a message which must pass these edges. Let θ_i^k be the time when i has completely traversed e_k and let $d_i^{k\to l}$ be the total amount of messages passing e_k and e_l and traversing e_k after time θ_i^k (see Figure 3). Then a lower bound on the makespan occurring on e_l is $\theta_i^k + d_i^{k\to l} + d_i(l-k)$.*

Proof. The proof is illustrated in Figure 3. We prove this by induction over $l-k$. If $l-k=0$ then the statement is of course true. Let MAK_l be the makespan of edge l. By the induction hypothesis we know for given k and l with $l - k \geq 1$ that

$$\mathrm{MAK}_l \geq \theta_i^{k+1} + d_i^{k+1 \to l} + d_i(l-k-1). \tag{6}$$

Further let Δ be the total size of messages passing e_k and e_l, traversing e_k after time θ_i^k and e_{k+1} before time θ_i^{k+1}. Then we get:

$$\theta_i^{k+1} \geq \theta_i^k + d_i + \Delta \tag{7}$$

$$\Delta \geq d_i^{k \to l} - d_i^{k+1 \to l} \tag{8}$$

Combining these inequalities leads to

$$\mathrm{MAK}_l \geq \theta_i^k + d_i^{k \to l} + d_i(l-k). \tag{9}$$

This completes the proof. □

Note that the bounds in the previous lemma depend on the considered schedule. The following corollary states a lower bounds on the minimum makespan on a particular edge over all feasible schedules.

Corollary 3. *Let $e_k, e_l \in E(P)$ with $k \leq l$ be two edges of P and $i \in I$ be a message which must pass both of these edges. Let $d^{k \to l}$ be the total size of messages passing e_k and e_l. Then a lower bound on the minimum makespan on edge e_l is $d^{k \to l} + d_i(l-k)$.*

Proof. Given an arbitrary schedule and a message i passing e_k and e_l we know $d^{k \to l} \leq \theta_i^k + d_i^{k \to l}$. Since $d^{k \to l}$ and d_i are independent of the considered schedule, the corollary follows directly from Lemma 1. □

Next we show that FDFA computes a schedule minimizing the makespan if the underlying instance does not contain nested messages. Let $<_P$ be the topological order of P. Recall that two messages $i_1, i_2 \in I$ are nested if one is strictly contained in the other one, i.e., if $s_{i_1} <_P s_{i_2} \leq_P t_{i_2} <_P t_{i_1}$ or vice versa.

Theorem 2. *Consider an instance of the message routing problem where no two messages are nested. Then FDFA computes a schedule which minimizes the makespan on each edge simultaneously.*

Proof. Due to the lack of space, we refer for the proof to the full version of the paper. □

Acknowledgements. The authors are much indebted to Fritz Eisenbrand for helpful discussions on the topic of this paper.

References

1. Leung, J.Y.T., Tam, T.W., Wong, C.S., Young, G.H.: Routing messages with release time and deadline constraints. Journal of Parallel and Distributed Computing 31, 65–76 (1995)
2. Leung, J.Y.T., Tam, T.W., Wong, C.S., Young, G.H.: Online routing of real-time messages. Journal of Parallel and Distributed Computing 34, 211–217 (1996)

3. Leung, J.Y.T.: Handbook of Scheduling: Algorithms, Models and Performance Analysis. Chapman and Hall/CRC, Boca Raton (2004)
4. Leighton, F.: Methods for message routing in parallel machines. In: Proceedings of the ACM Symposium on the Theory of Computing, pp. 77–96 (1992)
5. Hall, A., Hippler, S., Skutella, M.: Multicommodity flows over time: Efficient algorithms and complexity. Theoretical Computer Science 379, 387–404 (2007)
6. Williamson, D.P., Hall, L.A., Hoogeveen, J.A., Hurkens, C.A.J., Lenstra, J.K., Sevast'janov, A.V., Shmoys, D.B.: Short shop schedules. Operations Research 45, 288–294 (1997)
7. Feige, U., Scheideler, C.: Improved bounds for acyclic job shop scheduling. Combinatorica 3(22), 361–399 (2002)
8. Czumaj, A., Scheideler, C.: A new algorithmic approach to the general Lovász Local Lemma with applications to scheduling and satisfiability problems. In: Proc. of the thirty-second annual ACM symposium on Theory of computing, Portland, USA, pp. 38–47 (2000)
9. Leigthon, F.T., Maggs, B.M., Rao, S.B.: Packet routing and job shop scheduling in O(congestion+dilation) steps. Combinatorica 14, 167–186 (1994)
10. Leigthon, F.T., Maggs, B.M., Richa, A.: Fast algorithms for finding O(congestion+ dilation) packet routing schedules. Combinatorica 19, 375–401 (1999)
11. Busch, C., Magdon-Ismail, M., Mavronicolas, M.: Universal bufferless packet switching. SIAM Journal on Computing 37, 1139–1162 (2007)
12. Srinivasan, A., Teo, C.-P.: A constant-factor approximation algorithm for packet routing and balancing local vs. global criteria. SIAM Journal on Computing 30(6), 2051–2068 (2001)
13. Fleischer, L., Skutella, M.: Quickest flows over time. SIAM Journal on Computing 36, 1600–1630 (2007)
14. Garey, M.R., Johnson, D.S.: Computers and Intractability. W. H. Freeman and Co., New York (1979)
15. Grtschel, M., Lovsz, L., Schrijver, A.: The ellipsoid method and its consequences in combinatorial optimization. Combinatorica 1(2), 169–197 (1981)
16. Karp, R.M., Leigthon, F.T., Rivest, R.L., Thompson, C.D., Vazirani, U.V., Vazirani, V.V.: Global wire routing in two-dimensional arrays. Algorithmica 2, 113–129 (1987)
17. Hassin, R.: Approximation schemes for the restricted shortest path problem. Math. Oper. Research 17, 36–42 (1992)
18. Lorenz, D.H., Raz, D.: A simple efficient approximation scheme for the restricted shortest path problem. Oper. Res. Letters 28, 213–219 (2001)
19. Phillips, C.A.: The network inhibition problem. In: Proc. of the 25th Annual ACM Symposium on the Theory of Computing, San Diego, CA, pp. 776–785 (1993)