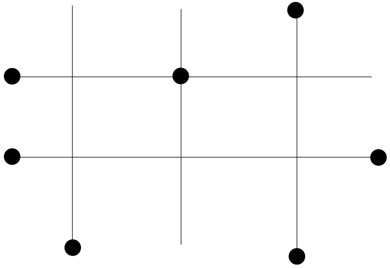
Sending Messages on Communication Networks on Time

Ronald Koch, Britta Peis, Martin Skutella, Andreas Wiese

TU Berlin

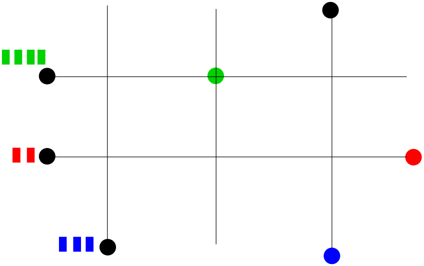
The Message Routing Problem

- In a distributed system, processes residing at different nodes of the network communicate by passing messages.



The Message Routing Problem

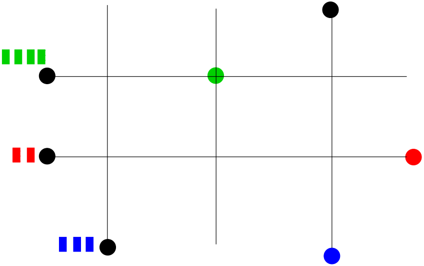
- In a distributed system, processes residing at different nodes of the network communicate by passing messages.



- It is an important question, whether a given set of messages $\{M_i\}_{i \in I}$ can be routed through the network on time.

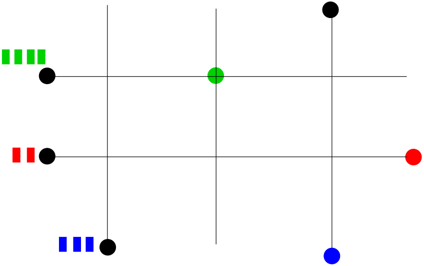
The Model

- Message $M_i = (s_i, t_i, l_i)$ consists of l_i unit-size packets that need to be send from s_i to t_i within time horizon T.



The Model

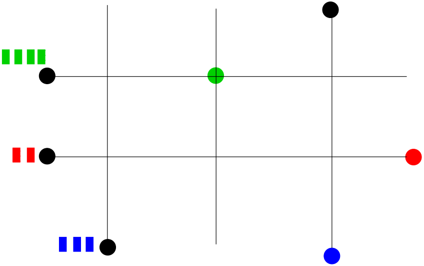
- Message $M_i = (s_i, t_i, l_i)$ consists of l_i unit-size packets that need to be send from s_i to t_i within time horizon T.



- It takes one time unit to send a packet on each link.

The Model

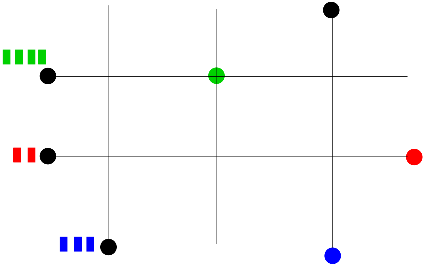
- Message $M_i = (s_i, t_i, l_i)$ consists of l_i unit-size packets that need to be send from s_i to t_i within time horizon T.



- It takes one time unit to send a packet on each link.
- At most one packet may traverse a link per time unit.

The Model

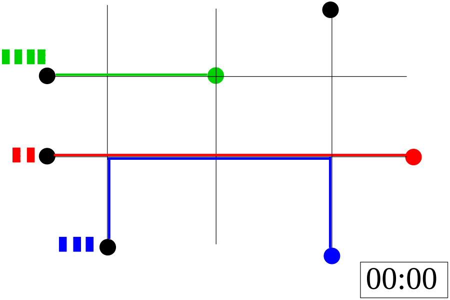
- Message $M_i = (s_i, t_i, l_i)$ consists of l_i unit-size packets that need to be send from s_i to t_i within time horizon T.



- It takes one time unit to send a packet on each link.
- At most one packet may traverse a link per time unit.
- Each message must be completely received by a node before it can be traversed to the next one.

The Model

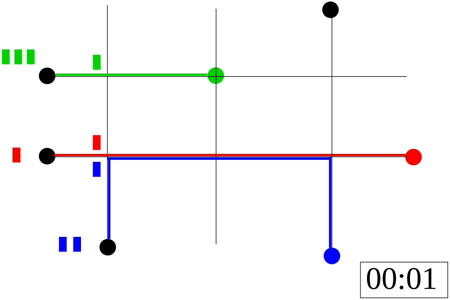
- Message $M_i = (s_i, t_i, l_i)$ consists of l_i unit-size packets that need to be send from s_i to t_i within time horizon T.



- It takes one time unit to send a packet on each link.
- At most one packet may traverse a link per time unit.
- Each message must be completely received by a node before it can be traversed to the next one.

The Model

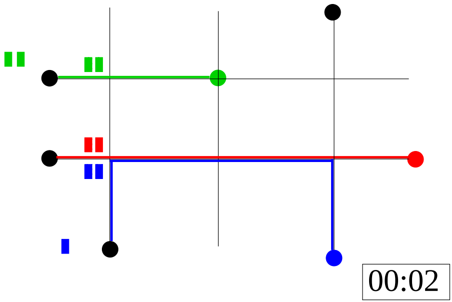
- Message $M_i = (s_i, t_i, l_i)$ consists of l_i unit-size packets that need to be send from s_i to t_i within time horizon T.



- It takes one time unit to send a packet on each link.
- At most one packet may traverse a link per time unit.
- Each message must be completely received by a node before it can be traversed to the next one.

The Model

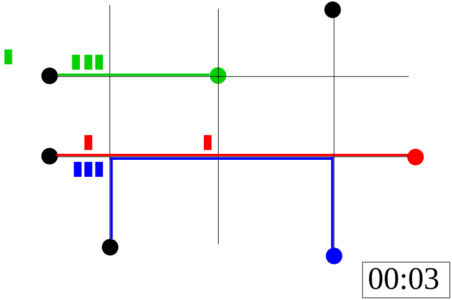
- Message $M_i = (s_i, t_i, l_i)$ consists of l_i unit-size packets that need to be send from s_i to t_i within time horizon T.



- It takes one time unit to send a packet on each link.
- At most one packet may traverse a link per time unit.
- Each message must be completely received by a node before it can be traversed to the next one.

The Model

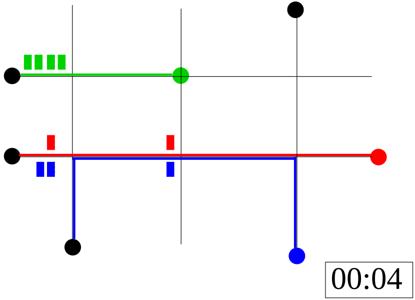
- Message $M_i = (s_i, t_i, l_i)$ consists of l_i unit-size packets that need to be send from s_i to t_i within time horizon T.



- It takes one time unit to send a packet on each link.
- At most one packet may traverse a link per time unit.
- Each message must be completely received by a node before it can be traversed to the next one.

The Model

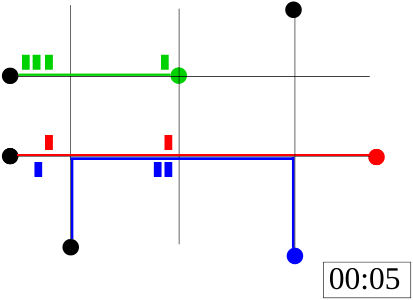
- Message $M_i = (s_i, t_i, l_i)$ consists of l_i unit-size packets that need to be send from s_i to t_i within time horizon T.



- It takes one time unit to send a packet on each link.
- At most one packet may traverse a link per time unit.
- Each message must be completely received by a node before it can be traversed to the next one.

The Model

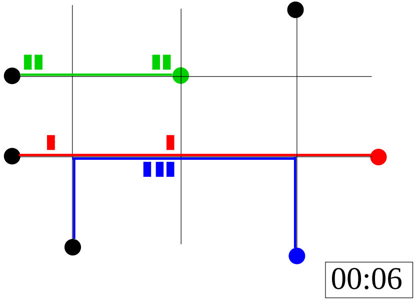
- Message $M_i = (s_i, t_i, l_i)$ consists of l_i unit-size packets that need to be send from s_i to t_i within time horizon T.



- It takes one time unit to send a packet on each link.
- At most one packet may traverse a link per time unit.
- Each message must be completely received by a node before it can be traversed to the next one.

The Model

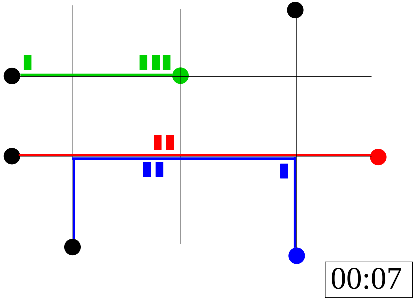
- Message $M_i = (s_i, t_i, l_i)$ consists of l_i unit-size packets that need to be send from s_i to t_i within time horizon T.



- It takes one time unit to send a packet on each link.
- At most one packet may traverse a link per time unit.
- Each message must be completely received by a node before it can be traversed to the next one.

The Model

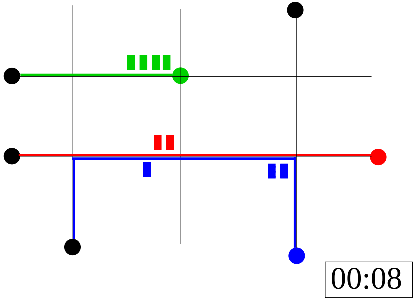
- Message $M_i = (s_i, t_i, l_i)$ consists of l_i unit-size packets that need to be send from s_i to t_i within time horizon T.



- It takes one time unit to send a packet on each link.
- At most one packet may traverse a link per time unit.
- Each message must be completely received by a node before it can be traversed to the next one.

The Model

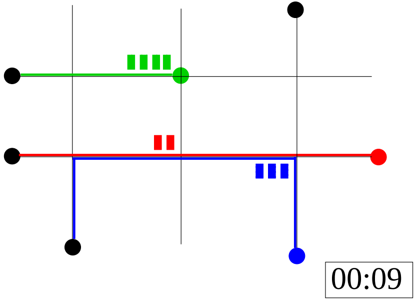
- Message $M_i = (s_i, t_i, l_i)$ consists of l_i unit-size packets that need to be send from s_i to t_i within time horizon T.



- It takes one time unit to send a packet on each link.
- At most one packet may traverse a link per time unit.
- Each message must be completely received by a node before it can be traversed to the next one.

The Model

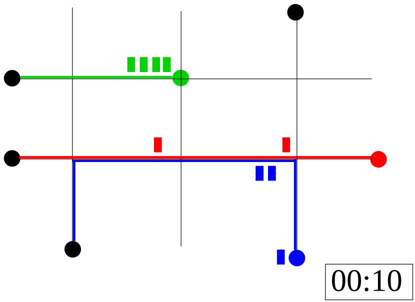
- Message $M_i = (s_i, t_i, l_i)$ consists of l_i unit-size packets that need to be send from s_i to t_i within time horizon T.



- It takes one time unit to send a packet on each link.
- At most one packet may traverse a link per time unit.
- Each message must be completely received by a node before it can be traversed to the next one.

The Model

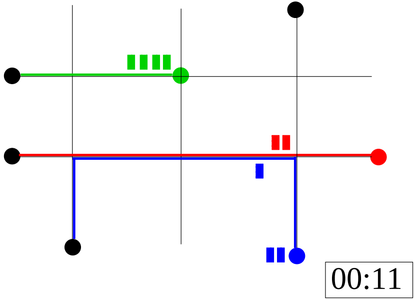
- Message $M_i = (s_i, t_i, l_i)$ consists of l_i unit-size packets that need to be send from s_i to t_i within time horizon T.



- It takes one time unit to send a packet on each link.
- At most one packet may traverse a link per time unit.
- Each message must be completely received by a node before it can be traversed to the next one.

The Model

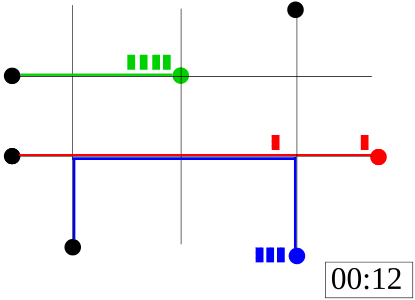
- Message $M_i = (s_i, t_i, l_i)$ consists of l_i unit-size packets that need to be send from s_i to t_i within time horizon T.



- It takes one time unit to send a packet on each link.
- At most one packet may traverse a link per time unit.
- Each message must be completely received by a node before it can be traversed to the next one.

The Model

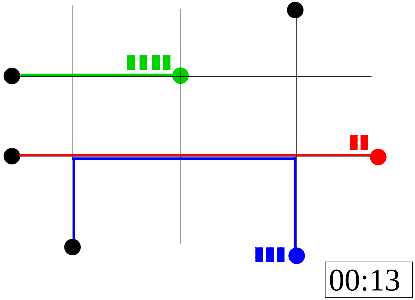
- Message $M_i = (s_i, t_i, l_i)$ consists of l_i unit-size packets that need to be send from s_i to t_i within time horizon T.



- It takes one time unit to send a packet on each link.
- At most one packet may traverse a link per time unit.
- Each message must be completely received by a node before it can be traversed to the next one.

The Model

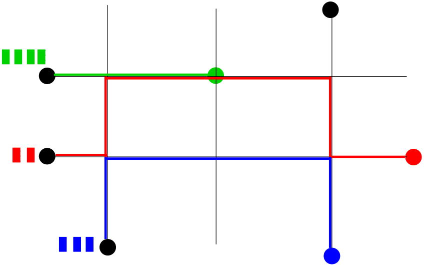
- Message $M_i = (s_i, t_i, l_i)$ consists of l_i unit-size packets that need to be send from s_i to t_i within time horizon T.



- It takes one time unit to send a packet on each link.
- At most one packet may traverse a link per time unit.
- Each message must be completely received by a node before it can be traversed to the next one.

The Model

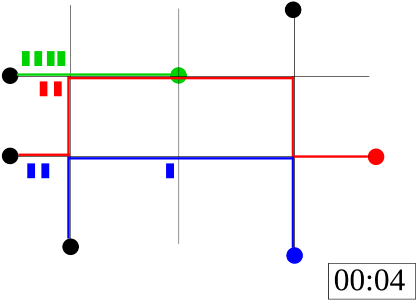
- Message $M_i = (s_i, t_i, l_i)$ consists of l_i unit-size packets that need to be send from s_i to t_i within time horizon T.



- It takes one time unit to send a packet on each link.
- At most one packet may traverse a link per time unit.
- Each message must be completely received by a node before it can be traversed to the next one.

The Model

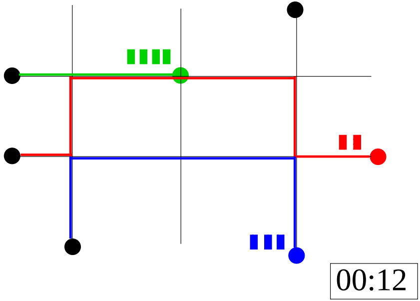
- Message $M_i = (s_i, t_i, l_i)$ consists of l_i unit-size packets that need to be send from s_i to t_i within time horizon T.



- It takes one time unit to send a packet on each link.
- At most one packet may traverse a link per time unit.
- Each message must be completely received by a node before it can be traversed to the next one.

The Model

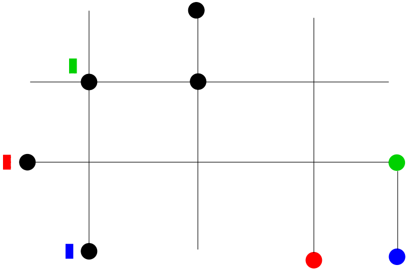
- Message $M_i = (s_i, t_i, l_i)$ consists of l_i unit-size packets that need to be send from s_i to t_i within time horizon T.



- It takes one time unit to send a packet on each link.
- At most one packet may traverse a link per time unit.
- Each message must be completely received by a node before it can be traversed to the next one.

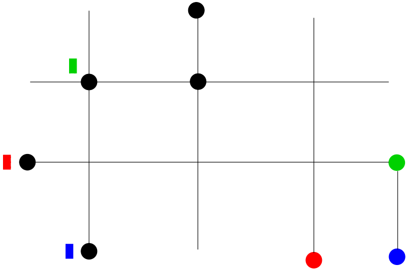
Special Case: Packet Routing

- In the **packet routing problem**, each message consists of only one packet.



Special Case: Packet Routing

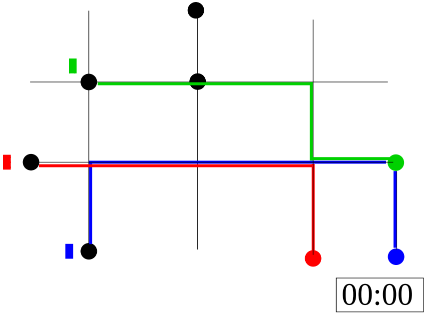
- In the **packet routing problem**, each message consists of only one packet.



- **Integral multicommodity flow problem over time** with unit travel times and capacities.

Special Case: Packet Routing

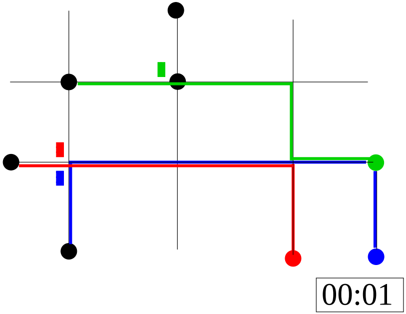
- In the **packet routing problem**, each message consists of only one packet.



- **Integral multicommodity flow problem over time** with unit travel times and capacities.

Special Case: Packet Routing

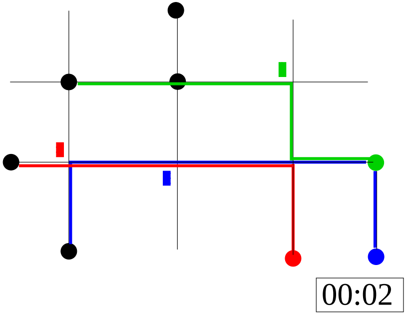
- In the **packet routing problem**, each message consists of only one packet.



- **Integral multicommodity flow problem over time** with unit travel times and capacities.

Special Case: Packet Routing

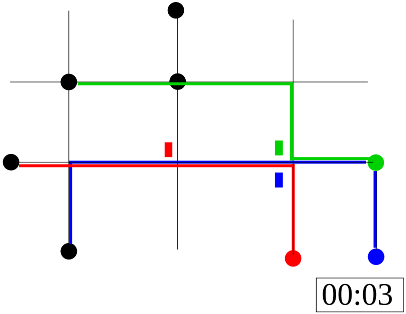
- In the **packet routing problem**, each message consists of only one packet.



- **Integral multicommodity flow problem over time** with unit travel times and capacities.

Special Case: Packet Routing

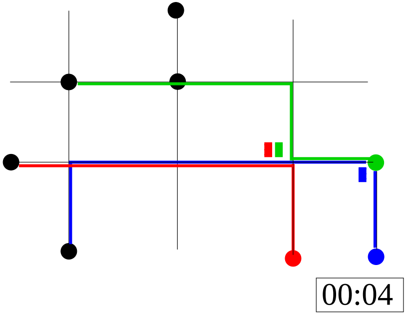
- In the **packet routing problem**, each message consists of only one packet.



- **Integral multicommodity flow problem over time** with unit travel times and capacities.

Special Case: Packet Routing

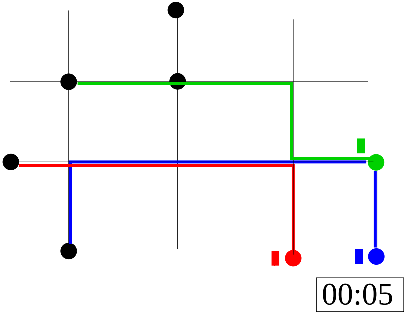
- In the **packet routing problem**, each message consists of only one packet.



- **Integral multicommodity flow problem over time** with unit travel times and capacities.

Special Case: Packet Routing

- In the **packet routing problem**, each message consists of only one packet.



- **Integral multicommodity flow problem over time** with unit travel times and capacities.

Single-Sink-Single-Source Packet and Message Routing

Observation

1-sink-1-source packet routing *can be solved efficiently.*

Proof.

Calculate a maximum s-t-flow over time with time horizon T. $\square$

Observation

1-sink-1-source packet routing *can be solved efficiently.*

Proof.

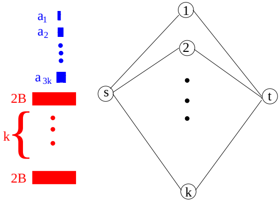
Calculate a maximum s-t-flow over time with time horizon T. □

Observation

1-sink-1-source message routing *is NP-complete.*

Proof.

Reduction from 3-PARTITION.

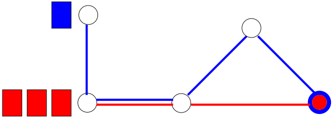


Contents

Contents

Congestion and Dilation

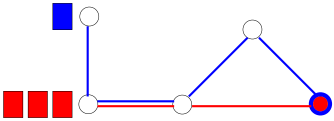
- In case the paths $\{P_i\}_{i \in I}$ are known in advance,



we have to trivial lower bounds on T:

Congestion and Dilation

- In case the paths $\{P_i\}_{i \in I}$ are known in advance,



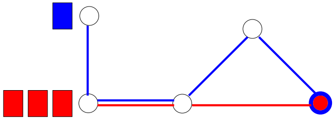
we have to trivial lower bounds on T:

- the **congestion**

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

Congestion and Dilation

- In case the paths $\{P_i\}_{i \in I}$ are known in advance,



we have to trivial lower bounds on T:

- the **congestion**

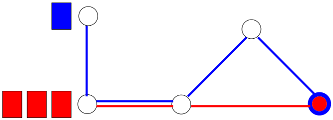
$$C := \max_{e \in E} \sum_{i: e \in P_i} l_i,$$

- and the **dilation**

$$D := \max_{i \in I} |P_i| l_i.$$

Congestion and Dilation

- In case the paths $\{P_i\}_{i \in I}$ are known in advance,



we have to trivial lower bounds on T:

- the **congestion**

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

- and the **dilation**

$$D := \max_{i \in I} |P_i| l_i.$$

- Assigning priorities $\longrightarrow$ acyclic job shop scheduling problem!

Job Shop Scheduling

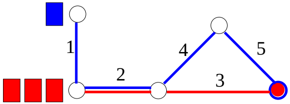
- Jobs $J_1, \ldots, J_n$, machines $M_1, \ldots, M_m$, each job consists of a sequence of operations $J_i = ((M_{i_1}, p_{i_1}), ..., (M_{i_k}, p_{i_k}))$ to be performed in order.

  Goal: Find feasible schedule with minimal makespan.

Job Shop Scheduling

- Jobs $J_1, \dots, J_n$, machines $M_1, \dots, M_m$, each job consists of a sequence of operations $J_i = ((M_{i_1}, p_{i_1}), ..., (M_{i_k}, p_{i_k}))$ to be performed in order.
  Goal: Find feasible schedule with minimal makespan.
- **Example**:

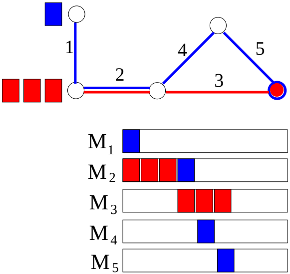
$$
\begin{aligned}
J_R &= ((M_2, 3), (M_3, 3)) \\
J_B &= ((M_1, 1), (M_2, 1), (M_4, 1), (M_5, 1)).
\end{aligned}
$$



Job Shop Scheduling

- Jobs $J_1, \ldots, J_n$, machines $M_1, \ldots, M_m$, each job consists of a sequence of operations $J_i = ((M_{i_1}, p_{i_1}), ..., (M_{i_k}, p_{i_k}))$ to be performed in order.
  Goal: Find feasible schedule with minimal makespan.
- **Example**:

$$\begin{aligned} J_R &= ((M_2, 3), (M_3, 3)) \\ J_B &= ((M_1, 1), (M_2, 1), (M_4, 1), (M_5, 1)). \end{aligned}$$



Results on Job Shop Scheduling

- AJSS is NP-hard to approximate within a factor of $\frac{5}{4}$ [*Sevast'janov et al. 93*]

Results on Job Shop Scheduling

- AJSS is NP-hard to approximate within a factor of $\frac{5}{4}$ [*Sevast'janov et al. 93*]
- $\mathcal{O}(C + D \log\log l_{\max})$-schedule exists [*Feige, Scheideler 02*]

Results on Job Shop Scheduling

- AJSS is NP-hard to approximate within a factor of $\frac{5}{4}$ [*Sevast'janov et al. 93*]
- $\mathcal{O}(C + D \log\log l_{\max})$-schedule exists [*Feige, Scheideler 02*]
- $\mathcal{O}(C + D)$-schedule if all operation lengths are one [*Leighton, Maggs, Richa 94*]

Results on Job Shop Scheduling

- AJSS is NP-hard to approximate within a factor of $\frac{5}{4}$ [*Sevast'janov et al. 93*]
- $\mathcal{O}(C + D \log\log l_{\max})$-schedule exists [*Feige, Scheideler 02*]
- $\mathcal{O}(C + D)$-schedule if all operation lengths are one [*Leighton, Maggs, Richa 94*]
- First constant factor approximation for packet routing [*Srinivasan, Teo 01*]

Results on Job Shop Scheduling

- AJSS is NP-hard to approximate within a factor of $\frac{5}{4}$ [*Sevast'janov et al. 93*]
- $\mathcal{O}(C + D \log\log l_{\max})$-schedule exists [*Feige, Scheideler 02*]
- $\mathcal{O}(C + D)$-schedule if all operation lengths are one [*Leighton, Maggs, Richa 94*]
- First constant factor approximation for packet routing [*Srinivasan, Teo 01*]
- Our algorithm finds paths for the message routing problem with C and D small.

Results on Job Shop Scheduling

- AJSS is NP-hard to approximate within a factor of $\frac{5}{4}$ [*Sevast'janov et al. 93*]
- $\mathcal{O}(C + D \log\log l_{\max})$-schedule exists [*Feige, Scheideler 02*]
- $\mathcal{O}(C + D)$-schedule if all operation lengths are one [*Leighton, Maggs, Richa 94*]
- First constant factor approximation for packet routing [*Srinivasan, Teo 01*]
- Our algorithm finds paths for the message routing problem with C and D small.
- It improves the result of Srinivasan and Teo for packet routing by a factor of 2.

Contents

Paths with C and D small

- For some fixed $D \leq T$ define

$$\mathcal{P}_i := \{s_i, t_i\text{-paths of length } \leq \frac{D}{l_i}\} \quad \forall i \in I.$$

Paths with C and D small

- For some fixed $D \leq T$ define

$$\mathcal{P}_i := \{s_i, t_i\text{-paths of length } \leq \frac{D}{l_i}\} \quad \forall i \in I.$$

- We are interested in an optimal $\{0,1\}$-solution of

$$\begin{array}{ll} \min_{x \geq 0} & C \\ \forall i \in I: & \sum_{P \in \mathcal{P}_i} x_P \geq 1 \\ \forall e \in E: & \sum_{i \in I} \sum_{P \in \mathcal{P}_i : e \in P} l_i x_P \leq C \end{array}$$

Paths with C and D small

- For some fixed $D \leq T$ define

$$\mathcal{P}_i := \{s_i, t_i\text{-paths of length } \leq \frac{D}{l_i}\} \quad \forall i \in I.$$

- We are interested in an optimal $\{0,1\}$-solution of

$$\begin{array}{ll} \min_{x \geq 0} & C \\ \forall i \in I: & \sum_{P \in \mathcal{P}_i} x_P \geq 1 \\ \forall e \in E: & \sum_{i \in I} \sum_{P \in \mathcal{P}_i : e \in P} l_i x_P \leq C \end{array}$$

Theorem

Our algorithm finds a $\{0,1\}$-solution $\hat{x}$ such that

$$\hat{C} < C^* + D,$$

where C^* is the congestion of an optimal fractional solution.

An optimal fractional solution

Observation

An **optimal fractional solution** x^* *of*

$$\begin{array}{ll} \min_{x \geq 0} & C \\ \forall i \in I: & \sum_{P \in \mathcal{P}_i} x_P \geq 1 \\ \forall e \in E: & \sum_{i \in I} \sum_{P \in \mathcal{P}_i : e \in P} l_i x_P \leq C \end{array}$$

can be found efficiently.

An optimal fractional solution

Observation

An **optimal fractional solution** x^* *of*

$$\begin{array}{ll} \min_{x \geq 0} & C \\ \forall i \in I: & \sum_{P \in \mathcal{P}_i} x_P \geq 1 \\ \forall e \in E: & \sum_{i \in I} \sum_{P \in \mathcal{P}_i : e \in P} l_i x_P \leq C \end{array}$$

can be found efficiently.

Proof.

The pricing problem is the **constant-length-bounded shortest path** problem ($\rightarrow$ modified Dijkstra). □

Rounding algorithm

Algorithm

$F \leftarrow$ messages with fixed paths (initially empty).

Iteratively:

1. *Compute a basic optimal solution x^* of*

$$\begin{array}{ll} \min_{x \geq 0} & C \\ \forall i \in I: & \sum_{P \in \mathcal{P}_i} x_P \geq 1 \\ \forall e \in E: & \sum_{i \in I} \sum_{P \in \mathcal{P}_i : e \in P} l_i x_P \leq C - \sum_{i \in F : e \in P_i} l_i \end{array}$$

2. *Fix variables with $x_P^* = 1$; (move corresponding i from I to F;)*
3. *Remove variables with $x_P^* = 0$;*
4. *Remove constraint e with*

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

The algorithm is well-defined

Theorem

If $0 < x_P^* < 1$ *for all paths* P*, there exists a constraint* e *with*

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

Proof.

Since x^* is b.f.s., there exist linearly independent tight constraints $\mathcal{T}_1$ and $\mathcal{T}_2$ of type (1) and (2) with $n = |\mathrm{supp}(x^*)| = |\mathcal{T}_1| + |\mathcal{T}_2|$. If

$$\forall e \in \mathcal{T}_2 : \quad \sum_{i \in I} \sum_{P \in \mathcal{P}_i : e \in P} l_i(1 - x_P^*) \geq D, \quad \text{then}$$

$$\begin{aligned} nD &\leq \textstyle\sum_{i \in \mathcal{T}_1} D \sum_{P \in \mathcal{P}_i} x_P^* + \sum_{e \in \mathcal{T}_2} \sum_{i \in I} \sum_{P \in \mathcal{P}_i : e \in P} l_i(1 - x_P^*) \\ &\leq \textstyle\sum_{i \in I} \sum_{P \in \mathcal{P}_i} (D x_P^* + \sum_{e \in \mathcal{T}_2 : e \in P} l_i(1 - x_P^*)) \\ &\leq \textstyle\sum_{i \in I} \sum_{P \in \mathcal{P}_i} (D x_P^* + D - D x_P^*) = Dn. \end{aligned}$$

Contradiction to linear independency! $\square$

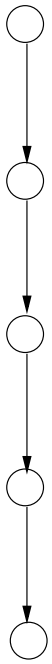
Corollary

Corollary

The algorithm determines paths with small congestion and dilation.

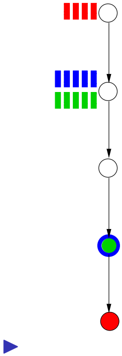
Contents

Message Routing on a Directed Path

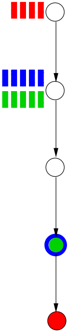


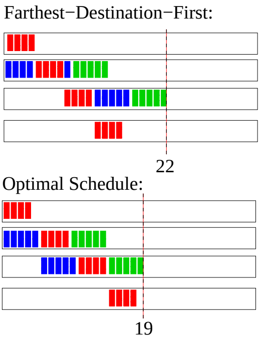
▶

Message Routing on a Directed Path

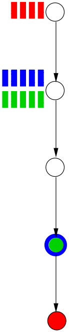


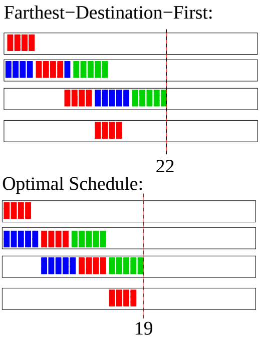
Message Routing on a Directed Path





Message Routing on a Directed Path

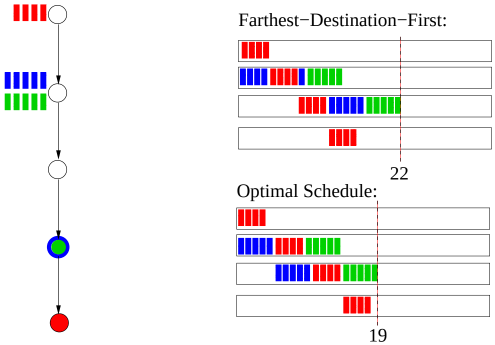




- Theorem (Leung, Tam, Wong, Young 96)

  Message routing on a directed path is NP-complete (reduction from 3-PARTITION).

Message Routing on a Directed Path



- **Theorem (Leung, Tam, Wong, Young 96)**
  Message routing on a directed path is NP-complete (reduction from 3-PARTITION).

- **Theorem**
  Farthest-Destination-First is optimal if $s_i <_P s_j \implies t_i \leq_P t_j$.

Further results on special graph classes

Message routing problem:

- NP-hard to approximate with a factor $< \frac{7}{6}$ on a **tree** even if message lengths of 1 and 2, only.
- NP-hard on a **grid** even in the single-sink, single-source case.

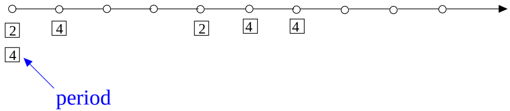
Packet routing problem:

- FDF optimal on directed **paths, in-trees** and **out-trees**.
- FDF arbitrarily bad on **trees**.
- 2-approximation on **trees**.
- $C + D - 1$-approximation on **directed trees**.
- NP-hard to approximate with a factor $< \frac{10}{9}$ on **trees**.
- NP-hard to approximate with a factor $< \frac{7}{6}$ on **planar graphs**.
- NP-hard to approximate with a factor $< \frac{6}{5}$ on **general graphs**.
- 2-approximation on a **grid** with pairwise different origins and destinations.

Contents

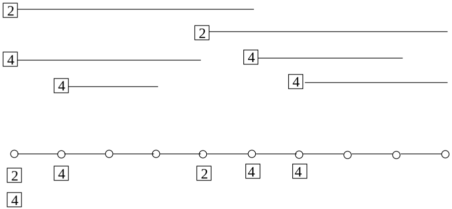
Periodic message routing

- Each message (here: packet) is released periodically.



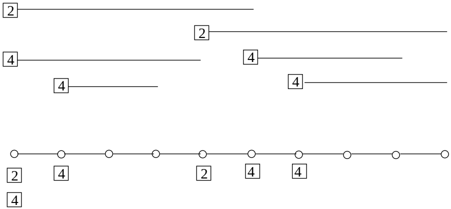
Periodic message routing

- Each message (here: packet) is released periodically.



Periodic message routing

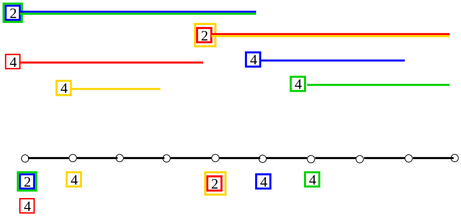
- Each message (here: packet) is released periodically.



- Feasible schedule: each packet is sent before the next one is released.

Periodic message routing

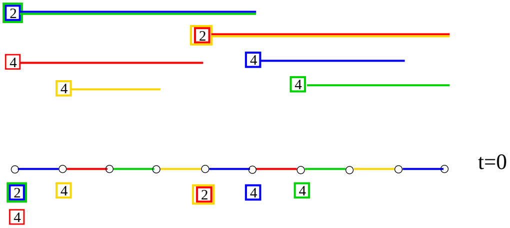
- Each message (here: packet) is released periodically.



- Feasible schedule: each packet is sent before the next one is released.

Periodic message routing

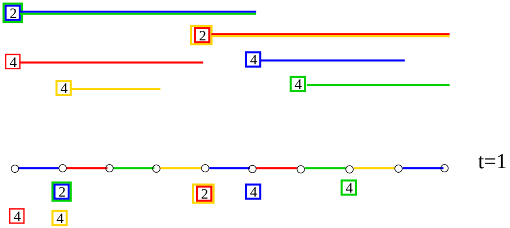
- Each message (here: packet) is released periodically.



- Feasible schedule: each packet is sent before the next one is released.

Periodic message routing

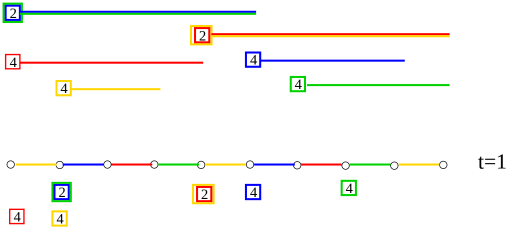
- Each message (here: packet) is released periodically.



- Feasible schedule: each packet is sent before the next one is released.

Periodic message routing

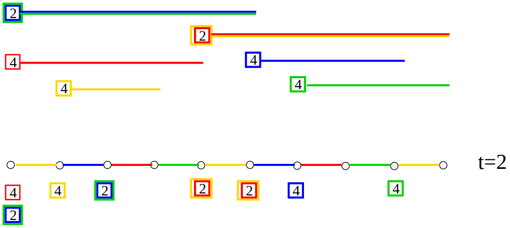
- Each message (here: packet) is released periodically.



- Feasible schedule: each packet is sent before the next one is released.

Periodic message routing

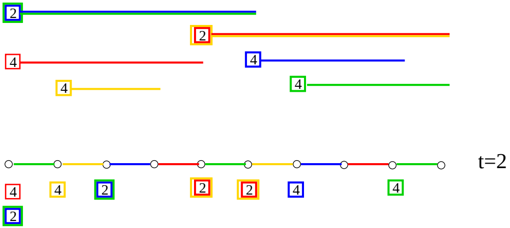
- Each message (here: packet) is released periodically.



- Feasible schedule: each packet is sent before the next one is released.

Periodic message routing

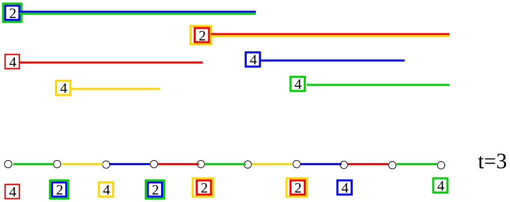
- Each message (here: packet) is released periodically.



- Feasible schedule: each packet is sent before the next one is released.

Periodic message routing

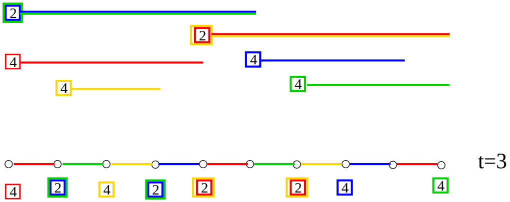
- Each message (here: packet) is released periodically.



- Feasible schedule: each packet is sent before the next one is released.

Periodic message routing

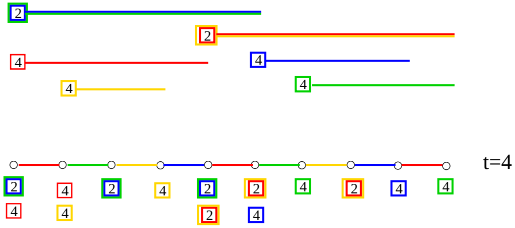
- Each message (here: packet) is released periodically.



- Feasible schedule: each packet is sent before the next one is released.

Periodic message routing

- Each message (here: packet) is released periodically.



- Feasible schedule: each packet is sent before the next one is released.

Color algorithm

- Send packets according to the **color algorithm:**
  1. Order the paths by increasing periods;
  2. Color a path P with period p with a color class
  $$[c_P] \in \{[0 \mod p], [1 \mod p], \ldots, [(p-1) \mod p]\}$$
  such that $[c_P] \cap [c_{P'}] = \emptyset$ if P and P' overlap.
  3. Color the edges e_k of $\{e_0, e_1, \ldots, e_n\}$ with time-dependent color
  $$(k+t) \mod p_{max} \quad \text{for } t = 0, \ldots;$$
  4. Send packet P along e_k at time step t if $e_k + t \in [c_P]$;

Color algorithm

- Send packets according to the **color algorithm:**
  1. Order the paths by increasing periods;
  2. Color a path P with period p with a color class

     $$[c_P] \in \{[0 \mod p], [1 \mod p], \ldots, [(p-1) \mod p]\}$$

     such that $[c_P] \cap [c_{P'}] = \emptyset$ if P and P' overlap.
  3. Color the edges e_k of $\{e_0, e_1, \ldots, e_n\}$ with time-dependent color

     $$(k+t) \mod p_{max} \quad \text{for } t = 0, \ldots;$$

  4. Send packet P along e_k at time step t if $e_k + t \in [c_P]$;
- **Theorem:** Feasible schedule if all periods are powers of 2.

Color algorithm

- Send packets according to the **color algorithm:**
  1. Order the paths by increasing periods;
  2. Color a path P with period p with a color class

     $$[c_P] \in \{[0 \mod p], [1 \mod p], \ldots, [(p-1) \mod p]\}$$

     such that $[c_P] \cap [c_{P'}] = \emptyset$ if P and P' overlap.
  3. Color the edges e_k of $\{e_0, e_1, \ldots, e_n\}$ with time-dependent color

     $$(k+t) \mod p_{max} \quad \text{for } t = 0, \ldots;$$

  4. Send packet P along e_k at time step t if $e_k + t \in [c_P]$;
- **Theorem:** Feasible schedule if all periods are powers of 2.
- Thus, for general periods feasible if utilities

$$u(e) := \sum_{i:e \in P_i} \frac{1}{p_i} \leq \frac{1}{2} \quad \forall e \in E.$$

Further Research

Solve all the open questions!