On Rerouting Connection Requests in Networks with Shared Bandwidth

David Coudert, Dorian Mazauric, Nicolas Nisse

MASCOTTE, INRIA, I3S, CNRS, UNS, Sophia Antipolis, France

DIMAP workshop on Algorithmic Graph Theory
Warwick, March 24th 2009
Routing in WDM Networks

Physical Network, Links provide several wavelengths

**multi-graph** \( G = (V, E) \)

an edge \((u, v) \iff \) one wavelength on the link \((u, v)\)

Routing of a set of requests/connections

set of requests \( \mathcal{R} \subseteq 2^{V \times V} \)

routing: for each request \((u, v)\),

a path from \(u\) to \(v\) and 1 wavelength.

A wavelength on a link used by **AT MOST 1** request

Problem: due to dynamicity of traffic, failures

how to maintain an efficient routing?
Network = Path with two wavelengths per link (2 parallel edges).
Basic Example

Network = Path with two wavelengths per link (2 parallel edges).

request for a A-F connection
Basic Example

Network = Path with two wavelengths per link (2 parallel edges).

A → B → C → D → E → F

request for a A-C connection
Network = Path with two wavelengths per link (2 parallel edges).

request for a E-F connection
Network = Path with two wavelengths per link (2 parallel edges).

end of the A-F connection
Network = Path with two wavelengths per link (2 parallel edges).

request for a D-F connection
Network = Path with two wavelengths per link (2 parallel edges).

request for a A-B connection
Basic Example

Network = Path with two wavelengths per link (2 parallel edges).

What if there is a request for a B-E connection? (using ONE wavelength) Impossible with the current routing...
Network = Path with two wavelengths per link (2 parallel edges).

... While it is possible !!
What can we do?

- Reject the new request → blocking probabilities
- Stop all requests and restart with new “optimal” routing
- Sequence of switching to converge to new routing
- Find the most suitable route for incoming request with eventual rerouting of pre-established connections

Our problem:

Inputs: Set of connection requests + current and new routing
Output: Scheduling for switching connection requests from current to new routes
Constraint: A connection is switched only once
What can we do?

- Reject the new request → *blocking probabilities*
- Stop all requests and restart with new “optimal” routing
- Sequence of switching to converge to new routing
- Find the most suitable route for incoming request with eventual rerouting of pre-established connections

**Our problem:**

**Inputs:** Set of connection requests
+ current *and* new routing

**Output:** Scheduling for switching connection requests from current to new routes

**Constraint:** A connection is switched only once
Tool: the Dependency Digraph (Jose & Somani, DRCN’03)

**Dependancy Digraph**
- one vertex per connection with different routes in $\mathcal{I}$ and $\mathcal{F}$
- arc from $u$ to $v$ if resources needed by $u$ in $\mathcal{F}$ are used by $v$ in $\mathcal{I}$

If cycles exist $\Rightarrow$ cyclic dependencies $\Rightarrow$
some requests must be interrupted
Tool: the Dependency Digraph (Jose & Somani, DRCN’03)

**Initial routing** $\mathcal{I}$ + request BE

**Final routing** $\mathcal{F}$ (pre-computed)

**Dependency Digraph**
- One vertex per connection with different routes in $I$ and $F$
- Arc from $u$ to $v$ if resources needed by $u$ in $F$ are used by $v$ in $I$

If cycles exist $\Rightarrow$ cyclic dependencies $\Rightarrow$
some requests must be interrupted

interrupt DF-connection (Break-before-Make)
initial routing $\mathcal{I} +$ request BE

\[
\begin{array}{cccccc}
A & B & C & D & E & F \\
\end{array}
\]

final routing $\mathcal{F}$ (pre-computed)

\[
\begin{array}{cccccc}
A & B & C & D & E & F \\
\end{array}
\]

**Dependancy Digraph**
- one vertex per connection with different routes in $\mathcal{I}$ and $\mathcal{F}$
- arc from $u$ to $v$ if resources needed by $u$ in $\mathcal{F}$ are used by $v$ in $\mathcal{I}$

If cycles exist $\Rightarrow$ cyclic dependencies $\Rightarrow$

some requests must be interrupted

reroute EF-connection

(Make-before-Break)
initial routing $\mathcal{I} + \text{request BE}$

final routing $\mathcal{F}$ (pre-computed)

Dependancy Digraph
- one vertex per connection with different routes in $I$ and $F$
- arc from $u$ to $v$ if resources needed by $u$ in $\mathcal{F}$ are used by $v$ in $\mathcal{I}$

If cycles exist $\Rightarrow$ cyclic dependencies $\Rightarrow$
some requests must be interrupted

reroute DF-connection
Tool: the Dependency Digraph (Jose & Somani, DRCN’03)

initial routing $I +$ request BE

final routing $F$ (pre-computed)

Dependancy Digraph
- one vertex per connection with different routes in $I$ and $F$
- arc from $u$ to $v$ if resources needed by $u$ in $F$ are used by $v$ in $I$

If cycles exist $\implies$ cyclic dependencies $\implies$ some requests must be interrupted

route BE-connection
Possible objectives

Minimize overall number of interrupted requests

Minimum Feedback Vertex Set (MFVS), here $N/4$
Possible objectives

**Minimize overall number of interrupted requests**

**Minimum Feedback Vertex Set (MFVS), here \( N/4 \)**
Possible objectives

Minimize overall number of interrupted requests

Minimum Feedback Vertex Set (MFVS), here $N/4$

Minimize number of simultaneous interrupted requests

**Process Number**, $pn = \text{smallest number of requests that have to be simultaneously interrupted.}$

Here, $pn = 1 \Rightarrow \text{Gap with MFVS up to } N/2$
Possible objectives

Minimize overall number of interrupted requests
Minimum Feedback Vertex Set (MFVS), here $N/4$

Minimize number of simultaneous interrupted requests
Process Number, $pn =$ smallest number of requests that have to be simultaneously interrupted.
Here, $pn = 1$ $\Rightarrow$ Gap with MFVS up to $N/2$
Possible objectives

Minimize overall number of interrupted requests
Minimum Feedback Vertex Set (MFVS), here $N/4$

Minimize number of \textit{simultaneous} interrupted requests

\textbf{Process Number}, $pn =$ smallest number of requests that have to be \textit{simultaneously} interrupted.
Here, $pn = 1 \Rightarrow$ Gap with MFVS up to $N/2$
Possible objectives

Minimize overall number of interrupted requests

Minimum Feedback Vertex Set (MFVS), here $N/4$

Minimize number of simultaneous interrupted requests

Process Number, $pn =$ smallest number of requests that have to be simultaneously interrupted.
Here, $pn = 1 \Rightarrow$ Gap with MFVS up to $N/2$
Possible objectives

Minimize overall number of interrupted requests

Minimum Feedback Vertex Set (MFVS), here $N/4$

Minimize number of **simultaneous** interrupted requests

**Process Number**, $pn =$ smallest number of requests that have to be **simultaneously** interrupted.

Here, $pn = 1 \Rightarrow$ Gap with MFVS up to $N/2$
Reconfiguration and Process number (Coudert, Sereni)

Game with Agents on the Dependency digraph $D$

Sequence of three basic operations, ...

1. Place a agent at a node = interrupt the request;
2. Process a node if all its out-neighbors are either processed or occupied by an agent = (Re)route a connection when final resources are available;
   A processed node is removed from the dependency digraph.
3. Remove an agent from a node, after having processed it.

...that must result in processing all nodes

Process number $pn(D) = \min p \mid D$ can be processed with $p$ agents

Remark: In undirected graphs or symmetric digraphs:

Graph Searching game when a fugitive is captured when surrounded
Example: DAG

Only one operation is used

1. Place an agent at a node = interrupt the request;
2. Process a node if all its out-neighbors are either processed or occupied by an agent = (Re)route a connection when final resources are available;
3. Remove an agent from a node, after having processed it.

DAG

Theorem

\( pn(D) = 0 \) iff \( D \) is a DAG
Digraphs with process number 1

One agent is used

1. Place a agent at a node = **interrupt the request**;
2. Process a node if all its out-neighbors are either processed or occupied by an agent = **(Re)route a connection when final resources are available**;
3. Remove an agent from a node, after having processed it.

Theorem

\[ pn(D) = 1 \iff \forall SCC, \text{MFVS}(SCC) = 1 \]

\[ O(N + M) \]

Coudert, Mazauric, Nisse  Routing Reconfiguration
Digraphs with process number 1

One agent is used

1. Place a agent at a node = interrupt the request;
2. Process a node if all its out-neighbors are either processed or occupied by an agent = (Re)route a connection when final resources are available;
3. Remove an agent from a node, after having processed it.

Theorem

\[ pn(D) = 1 \iff \forall SCC, \text{MFVS}(SCC) = 1 \]

\[ O(N + M) \]
Digraphs with process number 1

One agent is used

1. Place a agent at a node = interrupt the request;
2. Process a node if all its out-neighbors are either processed or occupied by an agent = (Re)route a connection when final resources are available;
3. Remove an agent from a node, after having processed it.

Theorem

\[ pn(D) = 1 \iff \forall SCC, \text{MFVS}(SCC) = 1 \]

\[ O(N + M) \]
Process number versus Other Parameters

- A parameter of directed (and undirected) graphs vs, vertex separation

in undirected graph or symmetric digraph: $\text{vs} = \text{pathwidth}$

$\text{vs}(G) = \text{pw}(G)$

Kinnersley [IPL 92]

Theorem (Coudert & Sereni, 2007)

$\text{vs}(D) \leq \text{pn}(D) \leq \text{vs}(D) + 1$

Complexity: NP-Complete, Not APX

- Characterization of digraphs with process number 0, 1, 2

(Coudert & Sereni, 2007)
State of the Art

- distributed $O(n \log n)$-time exact algorithm in trees
  (Coudert, Huc, Mazauric [DISC 08])

- generalized Model handling **priority connections**
  connections that cannot be interrupted
  heuristic
  (Coudert, Huc, Mazauric, Nisse, Sereni [ONDM 09])
Now: 1 wavelength on a link can be shared by several requests. More freedom, but reconfiguration becomes more difficult.

Example: Symmetric grid, one wavelength per link can be shared by 2 requests.

Routing 1, \( r \) and \( s \) cannot be accepted.

Routing 2

Theorem

NP-complete to decide whether the reconfiguration can be done without interruptions. This is true even if capacities of wavelength are at most 3.

Recall that if capacities equal 1, this problem is equivalent to recognize a DAG.
Ideas of Proof: generalized Dependency Digraph

Routing 1, r and s cannot be accepted

Routing 2

Dependancy Digraph (a color ⇔ a network's link)

generalized Dependancy Digraph: multi digraph with labeled edges

- one vertex per request with different routes in \( I \) and \( F + \)
- \( \forall \) network's link \( e \), 1 virtual vertex per free unit of capacity (in Routing 1) on \( e \)
- for any network's link \( e \), arc \( (u, v) \) labeled with \( e \)
  - from request \( u \) to vertex \( v \) if \( e \) is used by \( u \) in \( F \) and
  - either \( v \) is a request using \( e \) in \( I \), or \( v \) is a virtual vertex corresponding to \( e \)
Ideas of Proof: generalized Dependency Digraph

Routing 1, r and s cannot be accepted

Routing 2

Dependency Digraph (a color ⇔ a network's link)

Possible reconfiguration

Remarks

1 color of the dependency digraph ⇒ directed complete bipartite graph

possible reconfiguration ⇒
1: maximum matching for any color ⇒ “classical” Dependency Digraph
2: compute the process number of the obtain Dependency Digraph
Ideas of Proof: NP-Hardness

**Problem:** Is there a possible reconfiguration without interrupting requests?

From previous remarks:

⇔ Find a set of maximal matchings (1 per color) s.t. the obtained digraph is a DAG

**Reduction of 3-SAT**

Reduction of Formula \((a \lor b \lor \neg c) \land (\neg b \lor d \lor \neg e)\)

∃ set of matchings inducing DAG ⇔ Formula satisfiable
Further work

Lot of questions remain:

- Heuristics
- Distributed algorithms
- Realistic scenarios
- Complexity when ≤ 2 requests can share a link?
- Other objectives, Tradeoffs: simultaneous interruptions / interruption time / overall time of reconfiguration
- ...

Coudert, Mazaric, Nisse
Routing Reconfiguration