FBR: Dynamic Memory-Aware Fast Rerouting

Nicklas S. Johansen, Lasse B. Kær, Andreas L. Madsen,
Kristian Ø. Nielsen, Stefan Schmid, Jiří Srba and Rasmus G. Tollund

Aalborg University,
Denmark

TU Berlin and University of Vienna,
Germany, Austria

IEEE Global Internet 2022

Network Failure Protection

- Increasingly strict reliability requirements for modern computer networks
- In case of network failures, instantly reroute packages around failures
- However, fast rerouting requires additional forwarding rules (memory!)

We initiate the study of **memory-aware fast rerouting** techniques. Our aim is to develop a fast rerouting algorithm that explicitly models the memory limit on the routers.

Control Plane vs. Data Plane

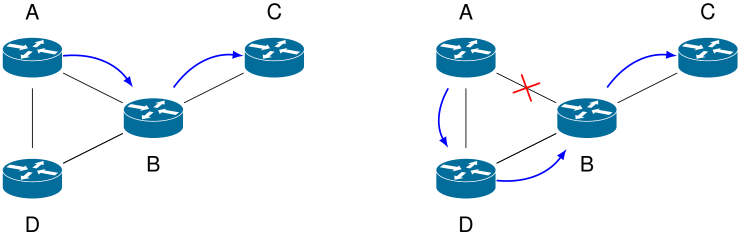
Control plane

- Determines how packets are forwarded
- Encodes the forwarding in data plane

Data plane

- Handles the packet-level forwarding using forwarding tables created by control plane logic

Motivation for Fast Rerouting



- Router and link failures are inevitable
- Convergence (in control plane) to new routing is slow
- Meanwhile, reroute based only on local failure information
  - Maintains connectivity until new routing is installed in data plane

Fast Rerouting in MPLS

- Multi-Protocol Label Switching: forwarding only by packet label
- Routers can mutate the packet label during forwarding
- Proactively compute data plane with protection paths
- Store information in label to aid routers in choosing intact protection paths

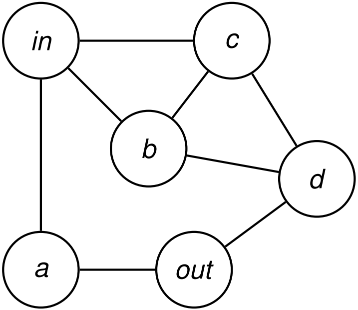
Our Contribution

We present Forward-Backward Routing (FBR) that generates highly failure resilient fast rerouting, while never exceeding the given memory limits of the routers.

Network Model

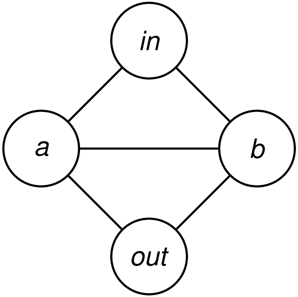
Definition

A *network topology* is an undirected graph $G = (V, E)$, where V is a set of *routers* and $E \subseteq V \times V$ is a set *links* between *routers*.



Demand: flow from *in* with label ℓ_0 to *out*

Packet Forwarding Using Label Switching



τ : forwarding table

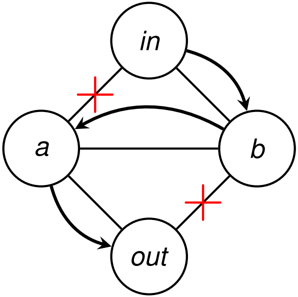
Router	Label	Priority	Router	Label
in	ℓ_1	1	*a*	ℓ_1
		2	*b*	ℓ_2
	ℓ_2	1	*b*	ℓ_2
a	ℓ_1	1	*out*	ℓ_1
		2	*b*	ℓ_1
		3	*in*	ℓ_2
b	ℓ_1	1	*out*	ℓ_1
	ℓ_2	1	*out*	ℓ_2
		2	*a*	ℓ_1

Memory usage = number of rules

$$M(\tau, in) = M(\tau, a) = M(\tau, b) = 3$$

Packet Forwarding Using Label Switching

Demand: (in, out, ℓ_1)



$F = \{(in, a), (b, out)\}$

τ_F : active forwarding table

Router	Label	Priority	Router	Label
in	ℓ_1	~~1~~	~~*a*~~	~~ℓ_1~~
		2	*b*	ℓ_2
	ℓ_2	1	*b*	ℓ_2
a	ℓ_1	1	*out*	ℓ_1
		~~2~~	~~*b*~~	~~ℓ_1~~
		~~3~~	~~*in*~~	~~ℓ_2~~
b	ℓ_1	~~1~~	~~*out*~~	~~ℓ_1~~
	ℓ_2	~~1~~	~~*out*~~	~~ℓ_2~~
		2	*a*	ℓ_1

Step	Router	Label	Forwarding rules
1	*in*	ℓ_1	$\cancel{(1, a, \ell_1)}, (2, b, \ell_2)$
2	*b*	ℓ_2	$\cancel{(1, out, \ell_2)}, (2, a, \ell_1)$
3	*a*	ℓ_1	$(1, out, \ell_1)$
4	*out*	ℓ_1	

Objective for Memory-Aware Fast Rerouting

Goal: High resilience to failures without exceeding any router's memory

Measure for failure resilience:

$$connectedness(\tau, D) = \sum_{F \subseteq E} p_F \cdot connectivity(\tau_F, D),$$

where

$connectivity(\tau_F, D) = \textit{ratio of successful deliveries to achievable deliveries}$

$connectivity, connectedness \in [0, 1]$

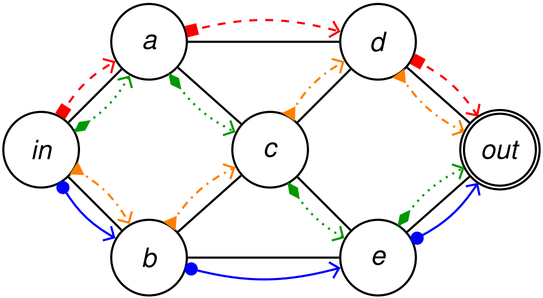
Forward-Backward Routing (FBR)

Intuition

- Find a list of alternative paths from ingress to egress
- In case of failure: route along the first path
- If that path fails, backtrack towards ingress
- When the next path is reached, switch to that path
- Repeat

FBR Example

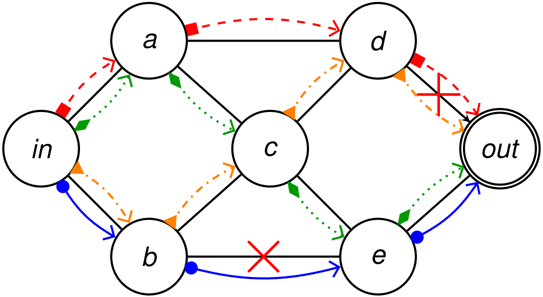
Demand: *in* to *out* with initial label ●



Path order: ● ■ ⧫ ▲

FBR Example

Demand: *in* to *out* with initial label ●



Path order: ● ■ ⧫ ▲

$$F = \{(b, e), (d, out)\}$$

Trace: *in* – *b* – *b* – *in* – *a* – *d* – *d* – *a* – *c* – *e* – *out*

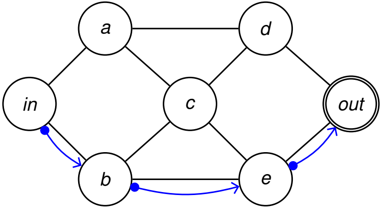
Encoding the Paths

1 unique label for every path encoded

For each path:

- Skip encoding if it exceeds memory!
- Encode forwarding with a unique label ℓ_i
- (If not last path) For each router in the path:
  - Add a local lookup to upgrade to the next label ℓ_{i+1}
  - Add backtracking rules with label ℓ_{i+1}

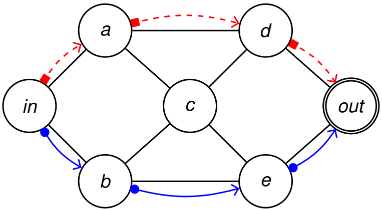
FBR Example: Encoding



τ : forwarding table

Router	Label	Priority	Router	Label
in	ℓ_1	1	*b*	ℓ_1
a				
b	ℓ_1	1	*e*	ℓ_1
c				
d				
e	ℓ_1	1	*out*	ℓ_1

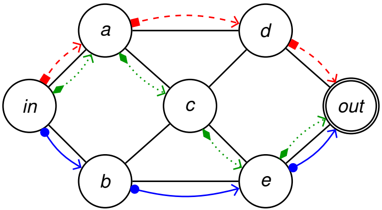
FBR Example: Encoding



τ : forwarding table

Router	Label	Priority	Router	Label
in	ℓ_1	1	*b*	ℓ_1
		2	*in*	ℓ_2
	ℓ_2	1	*a*	ℓ_2
a	ℓ_2	1	*d*	ℓ_2
b	ℓ_1	1	*e*	ℓ_1
		2	*b*	ℓ_2
	ℓ_2	1	*in*	ℓ_2
c				
d	ℓ_2	1	*out*	ℓ_2
e	ℓ_1	1	*out*	ℓ_1
		2	*e*	ℓ_2
	ℓ_2	1	*b*	ℓ_2

FBR Example: Encoding



If $m(v) = 5$ for *in* then no more paths can be added.

τ : forwarding table

Router	Label	Priority	Router	Label
in	ℓ_1	1	*b*	ℓ_1
		2	*in*	ℓ_2
	ℓ_2	1	*a*	ℓ_2
		2	*in*	ℓ_3
	ℓ_3	1	*a*	ℓ_3
a	ℓ_2	1	*d*	ℓ_2
		2	*a*	ℓ_3
	ℓ_3	1	*c*	ℓ_3
b	ℓ_1	1	*e*	ℓ_1
		2	*b*	ℓ_2
	ℓ_2	1	*in*	ℓ_2
c	ℓ_3	1	*e*	ℓ_3
d	ℓ_2	1	*out*	ℓ_2
		2	*d*	ℓ_3
	ℓ_3	1	*a*	ℓ_3
e	ℓ_1	1	*out*	ℓ_1
		2	*e*	ℓ_2
	ℓ_2	1	*b*	ℓ_2
	ℓ_3	1	*out*	ℓ_3

Theoretical Properties

Theorem

Forward-Backward Routing is loop-free.

Proof sketch: Paths are loop-free and are only used once.

Theorem

For a k-connected network topology, FBR can achieve $(k-1)$-resilience using $3k-2$ rules per demand per router.

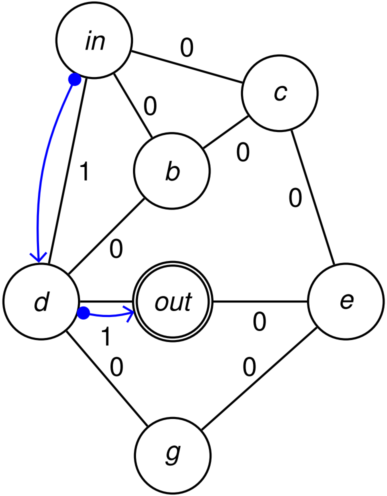
Proof sketch: In a k-connected graph, at least k edges must be removed to disconnected the graph. FBR creates k edge-disjoint paths to attempt.

Simple Path Generation Approach

Idea: Increase weight of used edges to increase disjointedness of paths.

For each iteration do:

- Pick next demand d
- Find shortest path p in G
- Update weights $W(d, e) := W(d, e) \cdot 2 + 1$ for each $e \in p$

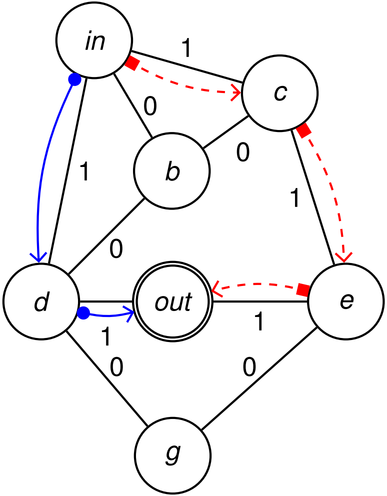


Simple Path Generation Approach

Idea: Increase weight of used edges to increase disjointedness of paths.

For each iteration do:

- Pick next demand d
- Find shortest path p in G
- Update weights $W(d, e) := W(d, e) \cdot 2 + 1$ for each $e \in p$

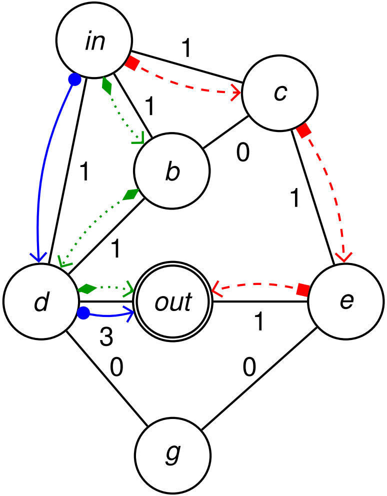


Simple Path Generation Approach

Idea: Increase weight of used edges to increase disjointedness of paths.

For each iteration do:

- Pick next demand d
- Find shortest path p in G
- Update weights $W(d,e) := W(d,e) \cdot 2 + 1$ for each $e \in p$

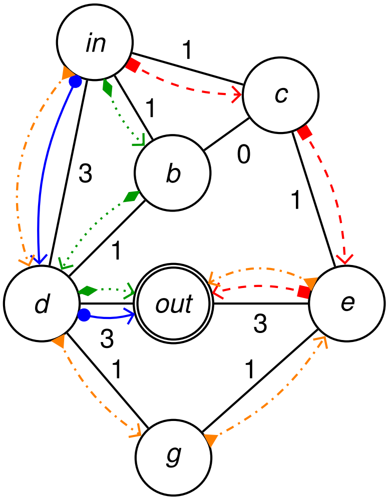


Simple Path Generation Approach

Idea: Increase weight of used edges to increase disjointedness of paths.

For each iteration do:

- Pick next demand d
- Find shortest path p in G
- Update weights $W(d,e) := W(d,e) \cdot 2 + 1$ for each $e \in p$



Methodology

Evaluation specifications

- 259 real-world networks from Topology Zoo
- Failure scenarios of size 0 to 4
- 1000 failure scenario limit for each size of F
- We assume link failures are independent, with probability 0.001 for a single link failure
- All routers are assumed to have the same memory limit

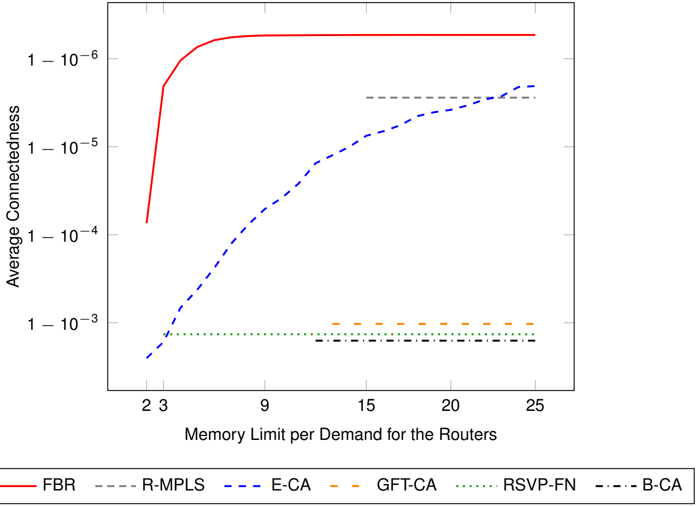
All experiments were conducted using the data plane simulation toolkit MPLS-Kit [2022] on a compute-cluster.

Related Work

Fast rerouting algorithms

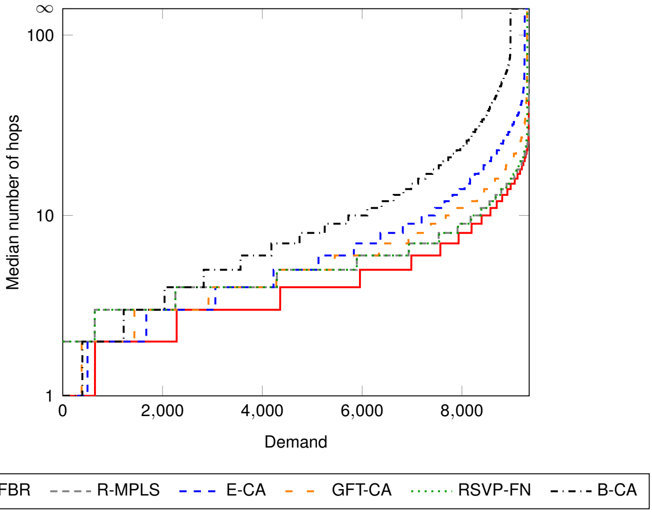
- RSVP-FN: industry standard [RFC 4090, 2005]
- R-MPLS: recent work that augments RSVP-FN [CoNEXT, 2022]
- B-CA: static arborescence approach [Trans. on Netw., 2017]
- GFT-CA: static DAG approach [INFOCOM, 2021]
- E-CA: dynamic memory-aware arborescence approach

Results: Connectedness and Memory Usage



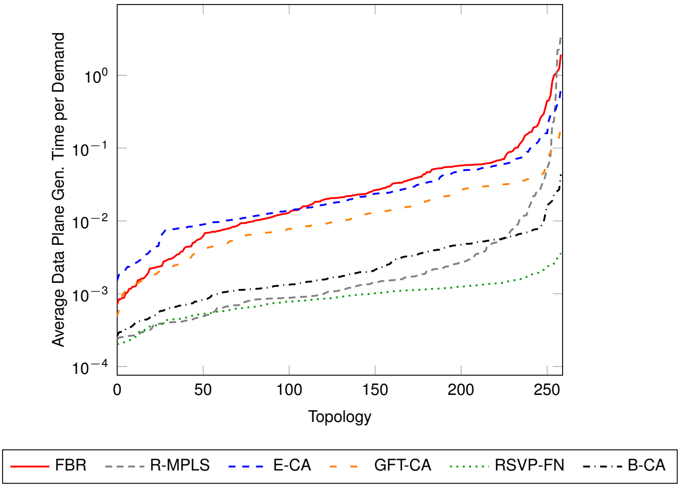
FBR achieves highest connectedness at any memory limitation!

Results: Number of Hops



FBR uses the fewest number of hops.

Results: Data Plane Generation Time



FBR is generally slower than the others, but still always under 2 seconds.

Conclusion

Forward-Backward Routing

- Memory-aware fast rerouting
- Outperforms previous work w.r.t. failure resilience and memory usage
- Uses the fewest number of hops
- Maximum generation time per demand is 2 seconds

Future Work

Improved path generation algorithm.