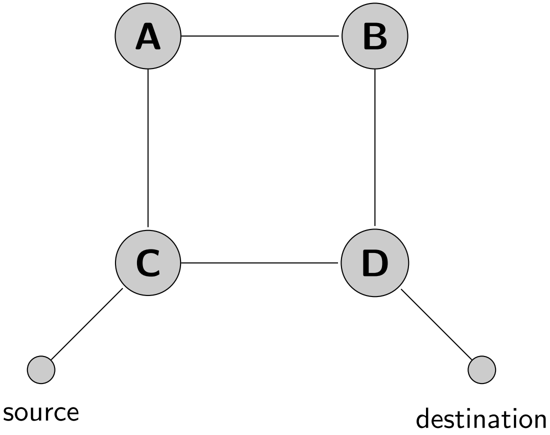
Sweet Little Lies

Fake Topologies for Flexible Routing

Nina Zinsli

Motivation

Goal: Send data packet from source to destination



Outline

Common Solutions for Network Routing

- Link-state Routing
- Software Defined Networks

Fibbing

- Using fake topologies for Network Routing
- Benefits & problems

Evaluation

Common solutions for Network Routing

Common solution

Link-state routing protocols

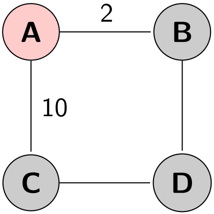
- widely used to steer network traffic

Link-state routing protocols

- every node has a map of the whole network
- compute forwarding path for every destination
  (only needs to know next hop)

Constructing network map

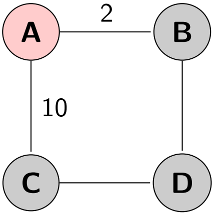
Constructing the map from router A's point of view:



1. determine neighbours and cost of connection

Constructing network map

Constructing the map from router A's point of view:



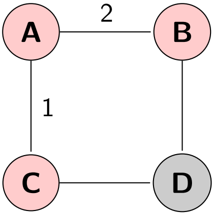
1. determine neighbours and cost of connection

link-state packet:

dist. from A	
B	2
C	1

Constructing network map

Constructing the map from router A's point of view:



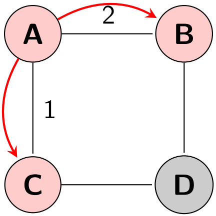
2. Flood link state packet through network

link-state packet:

dist. from A	
B	2
C	1

Constructing network map

Constructing the map from router A's point of view:



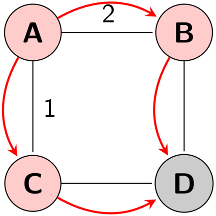
2. Flood link state packet through network

link-state packet:

dist. from A	
B	2
C	1

Constructing network map

Constructing the map from router A's point of view:



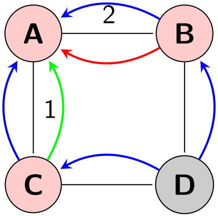
2. Flood link state packet through network

link-state packet:

dist. from A	
B	2
C	1

Constructing network map

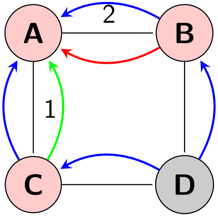
Constructing the map from router A's point of view:



3. Receive link state packet from other routers

Constructing network map

Constructing the map from router A's point of view:



3. Receive link state packet from other routers

link-state packets:

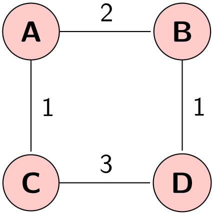
dist. from B	
A	2
D	1

dist. from C	
A	1
D	3

dist. from D	
B	1
C	3

Constructing network map

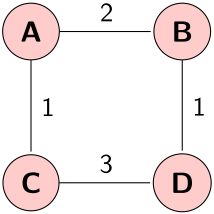
Constructing the map from router A's point of view:



4. Construct network map from link-state packets

Constructing network map

Constructing the map from router A's point of view:



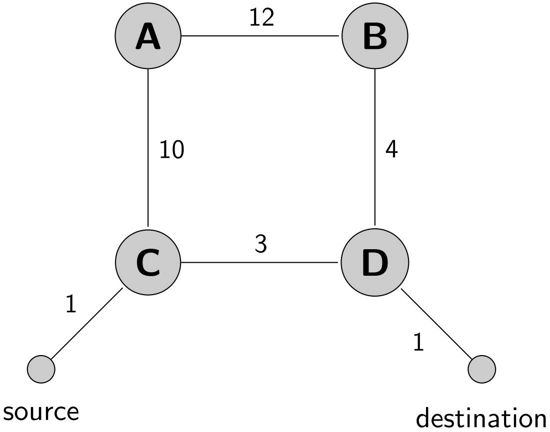
4. Construct network map from link-state packets

$\Rightarrow$ now A knows the whole topology

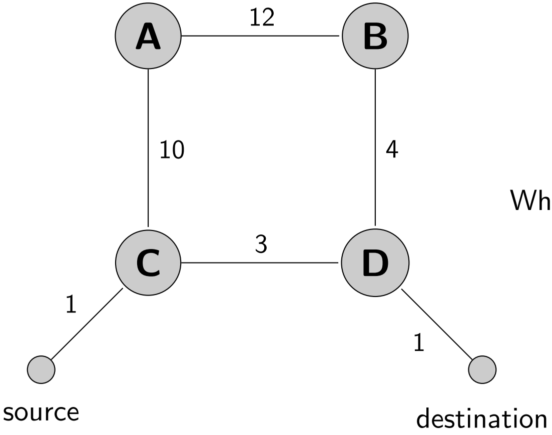
Example OSPF (Open Shortest Path First)

- Widely used link-state protocol
- Routers learn about topology like shown before
- Find shortest path

OSPF example

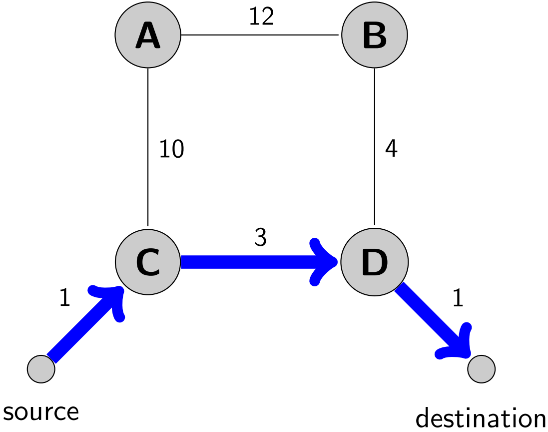


OSPF example



Which path will OSPF choose?

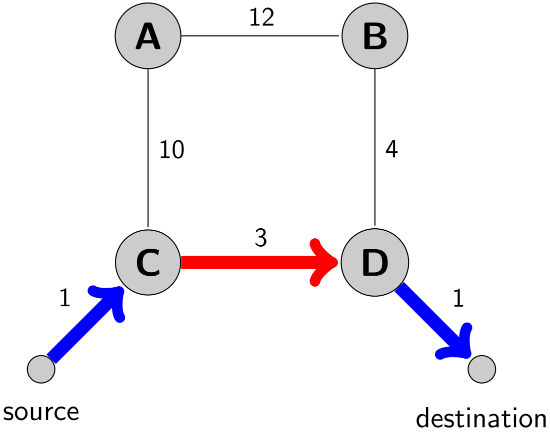
We know the solution



Advantages of this approach

- implementations are robust and widely-deployed
- deterministic algorithm
- behaviour well-understood (no surprises!)
- messages are standardized (standard protocol)

We are highly dependent on the red link



Problems with OSPF

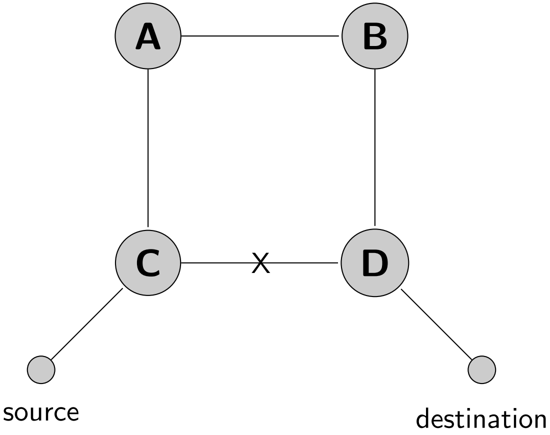
3 examples where OSPF is not ideal:

- link failure
- DDoS
- load balancing

Link Failure

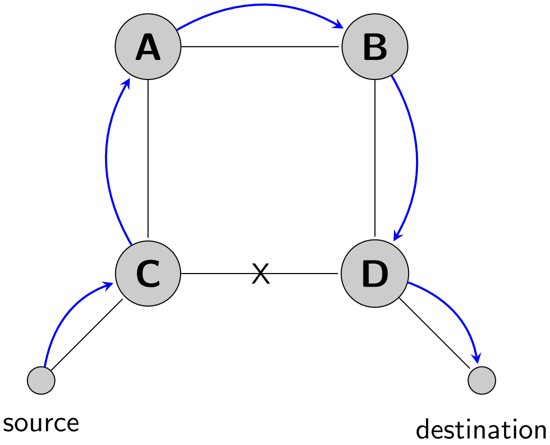
What if the link from C to D fails?

Link Failure



Link failure

We want to have a backup plan to react fast and redirect the data:

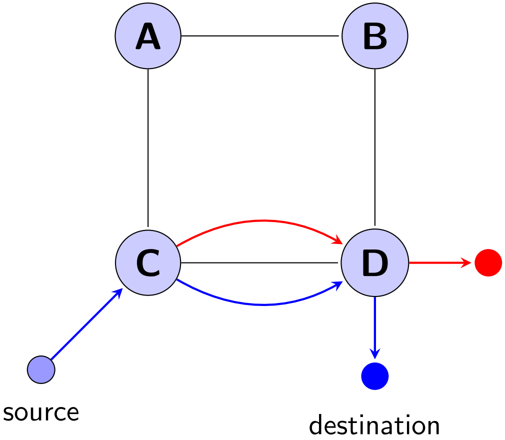


DDoS attack

Distributed Denial of Service:

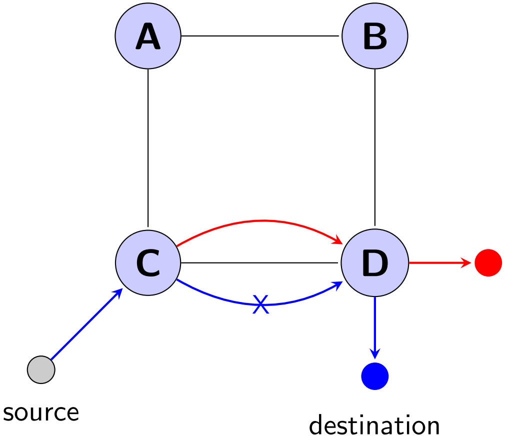
- attacker attempt to make an online service unavailable
- overwhelm it with traffic from multiple sources
- congest links

DDoS



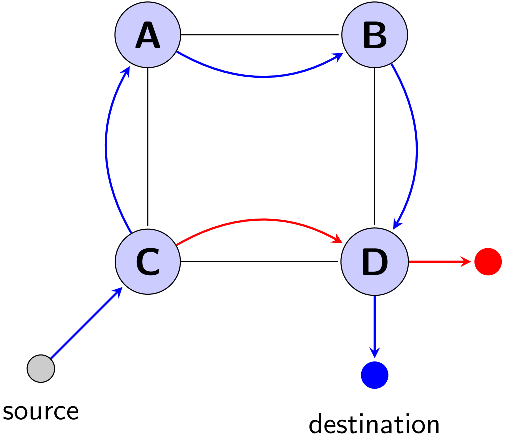
DDoS

Link between C and D congested!



DDoS

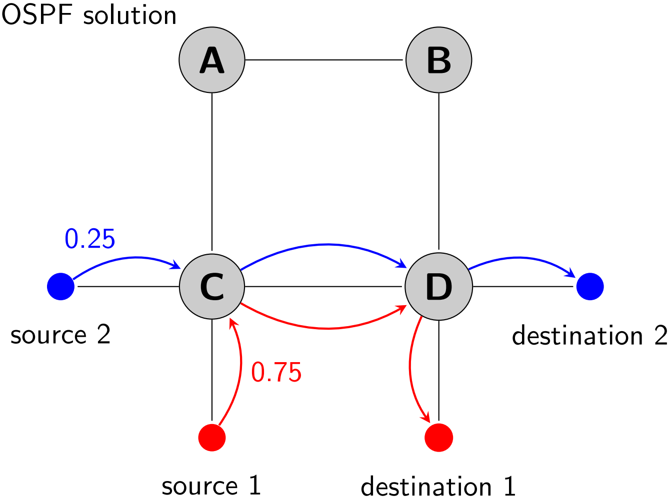
What we want:



Load balancing

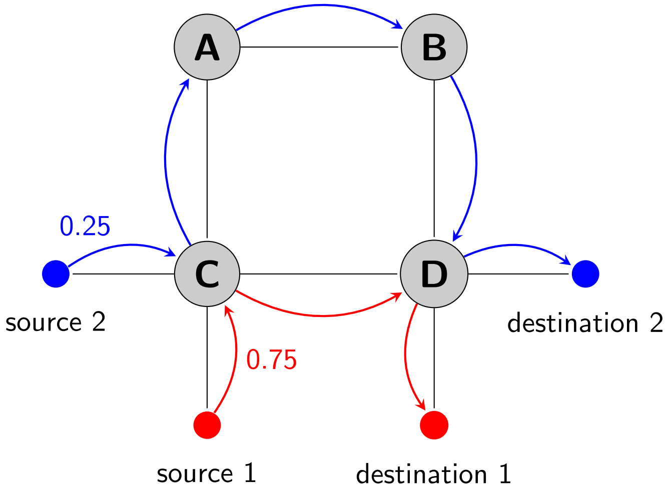
Huge amount of traffic from two sources
$\Rightarrow$ we want to split it on two different paths

Load balancing



Load balancing

What we want:



Better solution, maybe?

SDN (Software Defined Networks)

- can also be used to steer network traffic
- central controller chooses path for all traffic
- used by Google, Microsoft, ...

Better solution, maybe?

SDN (Software Defined Networks)

- can also be used to steer network traffic
- central controller chooses path for all traffic
- used by Google, Microsoft, ...
- does not scale to big networks
- cannot be used with most current routers (e.g. Cisco)

Better solution?

We want a solution which combines the benefits of both OSPF and SDN!

Better solution?

What we want:

- scales to big networks
- no central controller
- routers calculate the paths
- more flexible than OSPF
- works on existing routers (no large deviations from OSPF)

Fibbing

Fibbing

to fib: to lie about something minor or unimportant

Solution: Fibbing

New way to make network routing more flexible.

Solution: Fibbing

New way to make network routing more flexible.

$\Rightarrow$ Shortest-Path-Violations

Solution: Fibbing

Idea: Show the routers a **fake** topology.

Solution: Fibbing

Idea: Show the routers a **fake** topology.

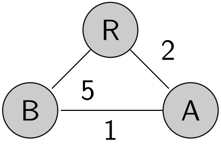
- add fake nodes to real topology (not physically)
  $\Rightarrow$ Router sees a different topology

Solution: Fibbing

Idea: Show the routers a **fake** topology.

- add fake nodes to real topology (not physically)
  $\Rightarrow$ Router sees a different topology

How the network looks like:

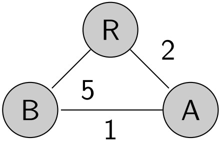


Solution: Fibbing

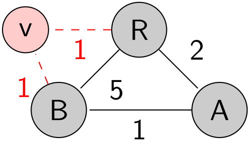
Idea: Show the routers a **fake** topology.

- add fake nodes to real topology (not physically)
  ⇒ Router sees a different topology

How the network looks like:



How R thinks the network looks like:

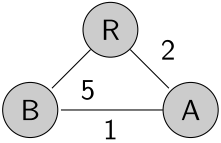


Solution: Fibbing

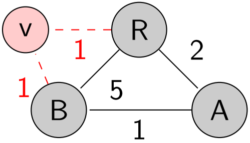
Idea: Show the routers a **fake** topology.

- add fake nodes to real topology (not physically)
  ⇒ Router sees a different topology

How the network looks like:



How R thinks the network looks like:



Router R computes shortest path on the second network

Fibbing

This allows us to make Router R choose a path which is not the shortest.
(if a path with a fake node is shorter)

Fibbing

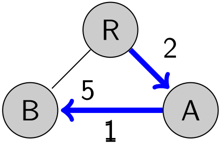
This allows us to make Router R choose a path which is not the shortest.
(if a path with a fake node is shorter)

But a data packet cannot be sent over a fake node ☹

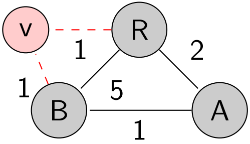
Fibbing

Assume R wants so send a packet to B:

Shortest path in real network:



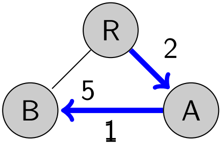
Fake network:



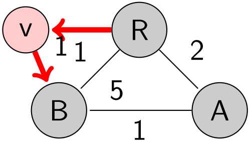
Fibbing

Assume R wants so send a packet to B:

Shortest path in real network:



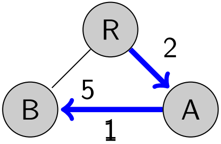
Shortest path in fake network:



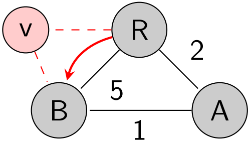
Fibbing

Fibbing redirects data over existing link:

Shortest path in real network:



Shortest path in fake network:



The Fibbing controller

- announces fake node to routers
  - local (seen by single router)
  - global (seen by all routers)
- chooses them such that routers send traffic over desired path

The Fibbing controller

- just used to insert fake nodes!
  does not compute paths
- mostly only a few Shortest-Path-Violations
- multiple controllers can be used for different subnets

The Fibbing controller

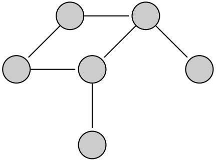
- just used to insert fake nodes!
  does not compute paths
- mostly only a few Shortest-Path-Violations
- multiple controllers can be used for different subnets

$\Rightarrow$ Fibbing controller can be used in big networks

Input

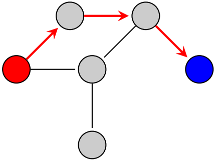
Input

Physical topology



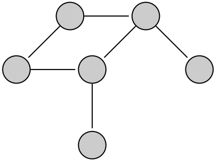
$+$

desired path



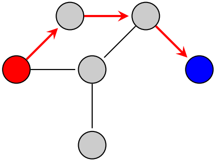
Output

Physical topology



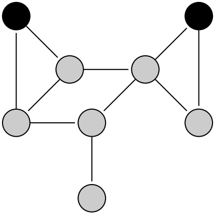
$+$

desired path



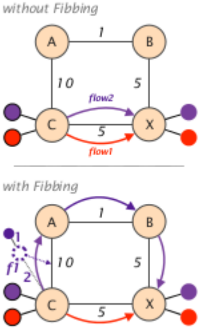
$\Rightarrow$

Topology with fake nodes

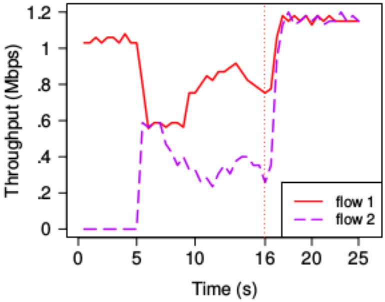


How fibbing solves all three problems

Load Balancing Example



(a) Topology



(b) Throughput evolution

Fibbing is expressive

Good news:

Theorem Any set of desired paths can be enforced by Fibbing.

How Fibbing works

2 Algorithms:

1. Simple
2. Merge

Simple

Simple

- is used if we want to react fast
- local fake node for every shortest-path violation

Simple

- is used if we want to react fast
- local fake node for every shortest-path violation
- might introduce a lot of new fake nodes!

Merge

- is used to reduce the number of fake nodes
- can be used to compute backup plans
- can be used after Simple to clean up

Merge

Goal: Merge local fake nodes to global fake nodes whenever possible to reduce number of fake nodes

Merge

Goal: Merge local fake nodes to global fake nodes whenever possible to reduce number of fake nodes

- for every local fake node, safe the minimum and maximum weight
- take two nodes together if possible

Problems with implementation

Problems with implementation

- with current routers not possible to lie about direct neighbour
- if desired path differs from shortest path in the first hop, we can not achieve it

Problems with implementation

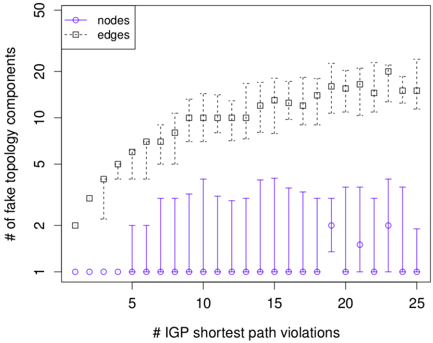
- with current routers not possible to lie about direct neighbour
- if desired path differs from shortest path in the first hop, we can not achieve it
- with small changes in routers it should be possible

Evaluation

Evaluation

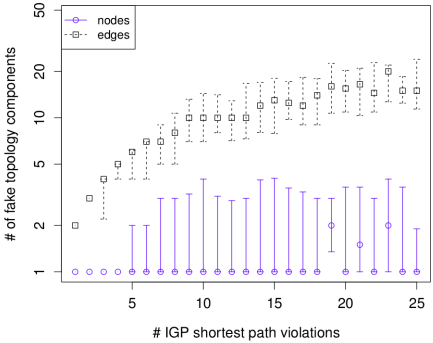
- test how number of desired Shortest-Path-Violations affects number of fake nodes

Evaluation



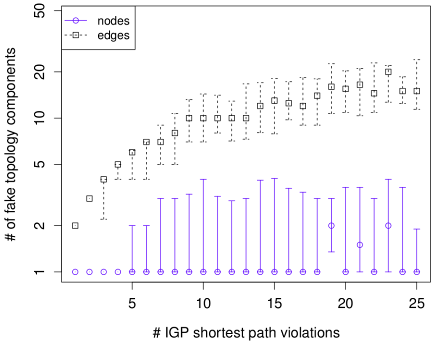
- ○: Median # of nodes
- □: Median # of edges
- solid bar: 95th percentile
- dashed bar: 5th percentile

Evaluation



- ○: Median # of nodes
- □: Median # of edges
- solid bar: 95th percentile
- dashed bar: 5th percentile
- real network (AS 6461, 141 nodes, 748 edges)
- random desired paths

Evaluation



- not many fake components needed, max # nodes: 5, edges: 26
- not strictly increasing

Memory and Time

# fake nodes	RIB memory (MB)	OSPF memory (MB)
1,000	0.09	0.56
5,000	1.58	5.19
10,000	3.56	10.96
50,000	19.67	56.37
100,000	39.78	113.17

small memory and CPU overhead

# fake nodes	installation time (s)	avg time/entry (μs)
1,000	0.89	886.00
5,000	4.46	891.40
10,000	8.96	894.50
50,000	44.74	894.78
100,000	89.50	894.98

Conclusion

- Fibbing achieves what we want:
  more flexible routing with few overhead

Conclusion

- Fibbing achieves what we want:
  more flexible routing with few overhead
- tests on small networks seem to work

Problems

- Fibbing controller takes desired path as an input, does not find an alternative path itself
- We know that Fibbing always works, but there are no guarantees for speed and number of fake nodes

Questions

?