Lossless Migrations of Link-State IGPs

Laurent Vanbever, Student Member, IEEE, Stefano Vissicchio, Cristel Pelsser,
Pierre Francois, Member, IEEE, and Olivier Bonaventure, Member, IEEE

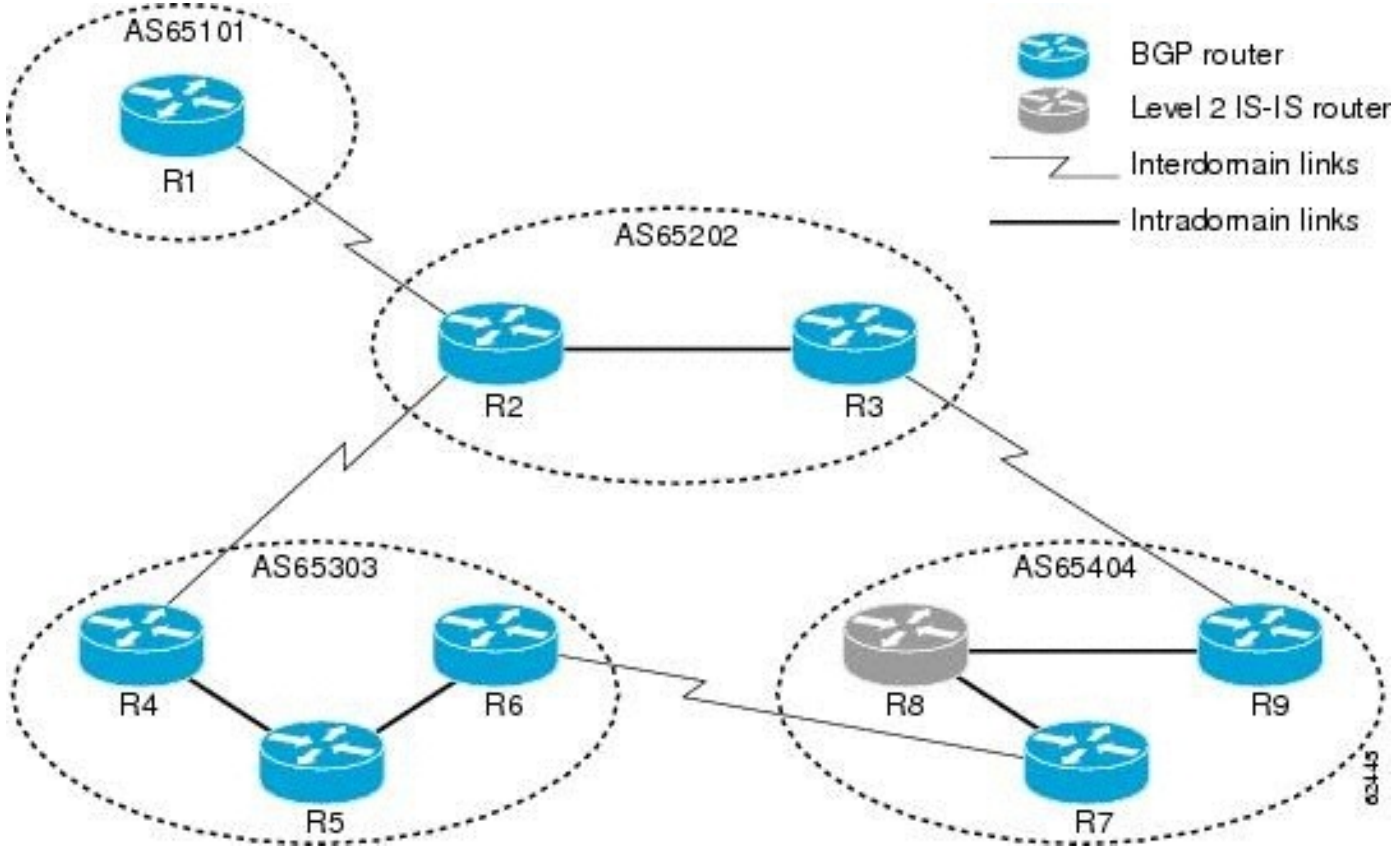
Seminar in Distributed Computing
Jochen Zehnder

Table of Contents

1. Topology of the Internet
2. Migration Problem
3. Evaluation
4. Summary
5. Q&A

Topology of the Internet

Topology of the Internet



http://www.cisco.com/c/en/us/td/docs/ios/12_2sr/12_2srb/feature/guide/tbgp_c/brbclns.html

Autonomous System (AS)

"An AS is a connected group of one or more IP prefixes run by one or more network operators which has a SINGLE and CLEARLY DEFINED routing policy." (RFC 1930)

Autonomous System (AS)

- groups of IP prefixes
  - e.g. AS559 (ETH-NET)
    - 129.132.0.0/16

Autonomous System (AS)

- groups of IP prefixes
  - e.g. AS559 (ETH-NET)
    - 129.132.0.0/16
- two types of protocols
  - Border Gateway Protocol (BGP)
  - Interior Gateway Protocol (IGP)

Interior Gateway Protocol

- flat vs. hierarchical
  - flat: forward packets along the shortest path
  - hierarchical: divided into zones

Interior Gateway Protocol

- flat vs. hierarchical
  - flat: forward packets along the shortest path
  - hierarchical: divided into zones
- route summarization
  - for hierarchical IGP
  - zone announces available prefixes and
  - length of the path

Migration Problem

next-hop function

- next-hop function $nh(u, d)$
  - u : router
  - d : destination

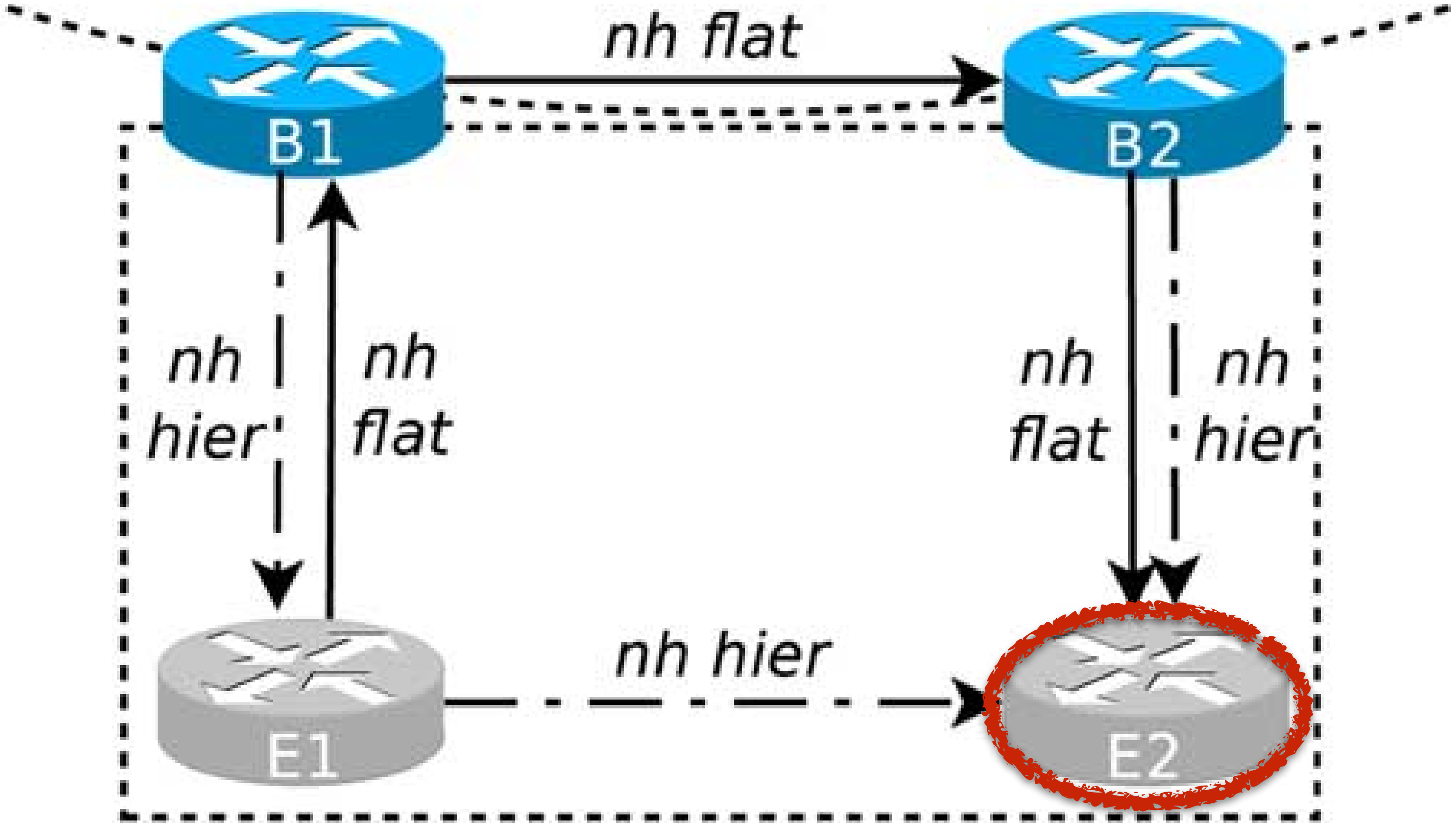
next-hop function

- next-hop function $nh(u, d)$
  - u : router
  - d : destination
- next router towards d

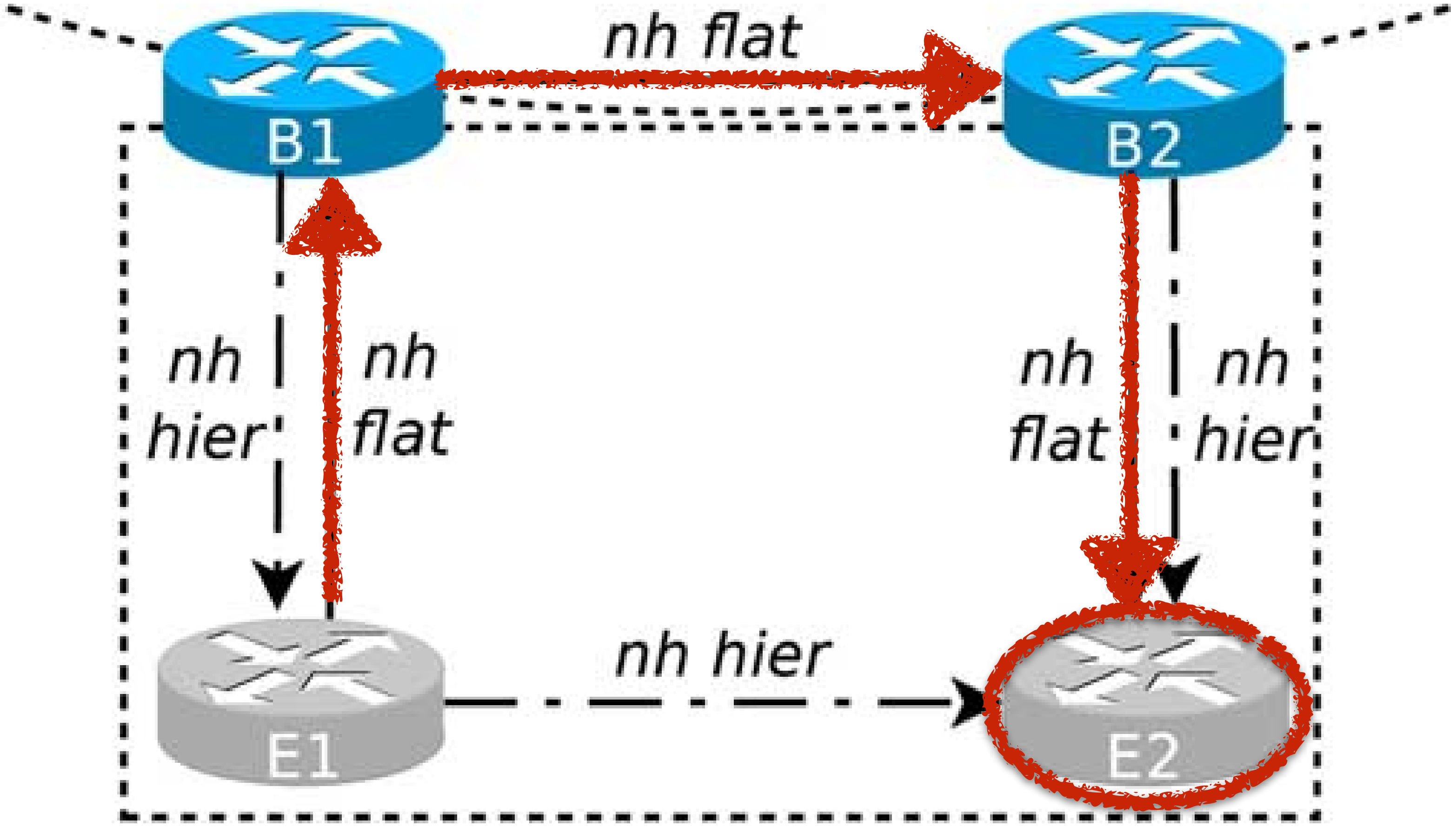
next-hop function

- next-hop function $nh(u, d)$
  - u : router
  - d : destination
- next router towards d
- $|nh(u, d)|$ does not have to be 1

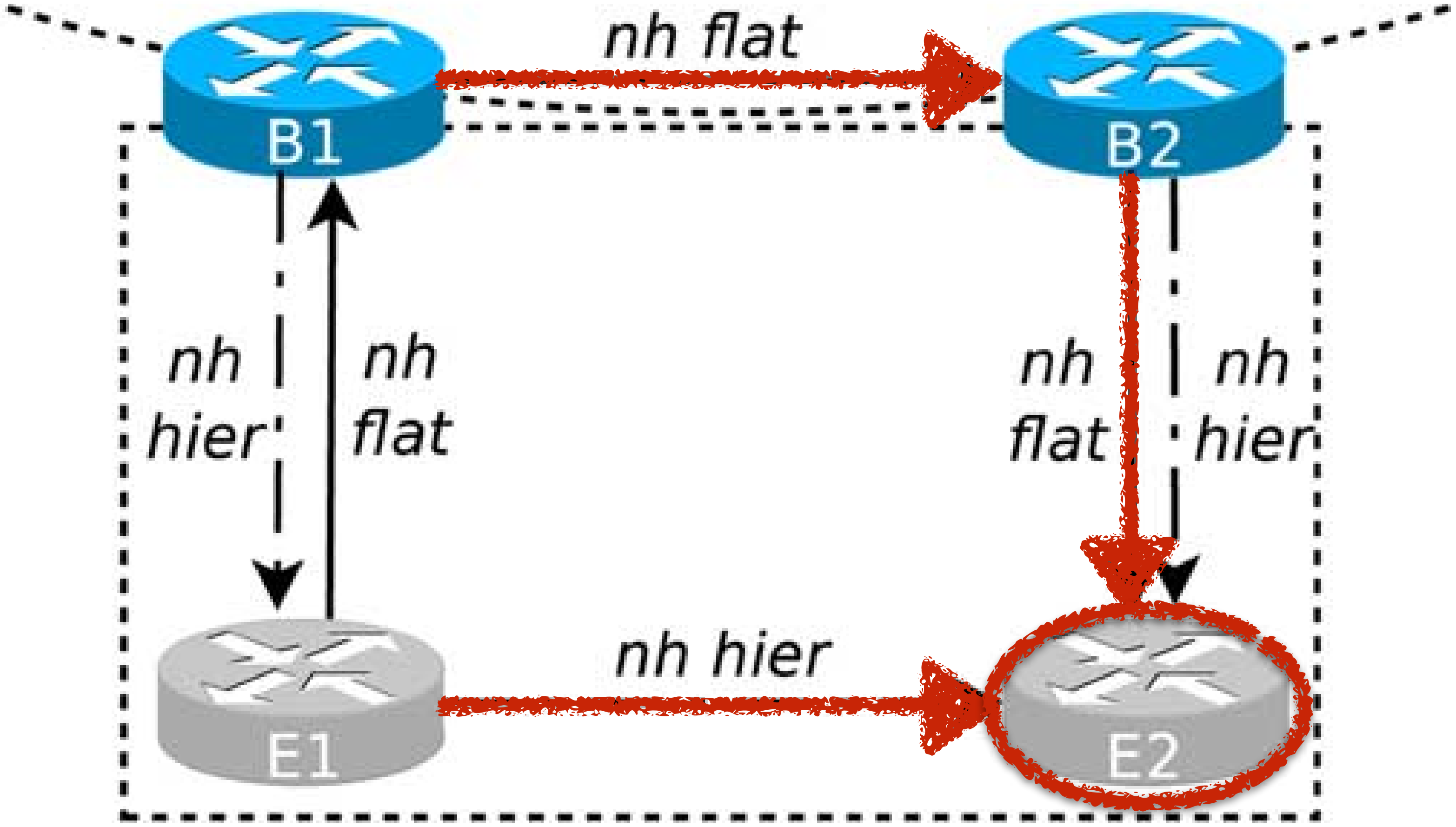
Migration Loop



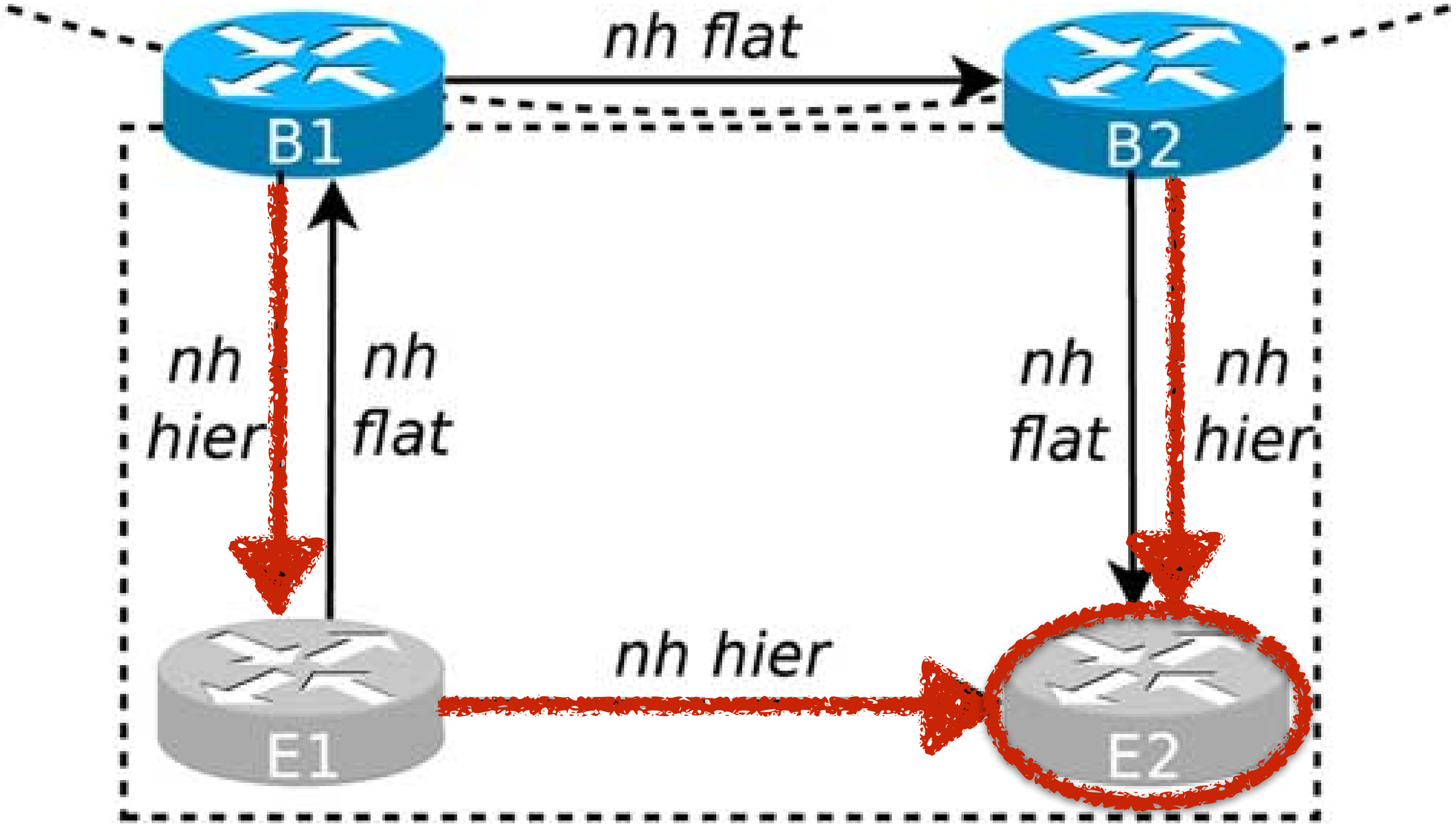
Migration Loop



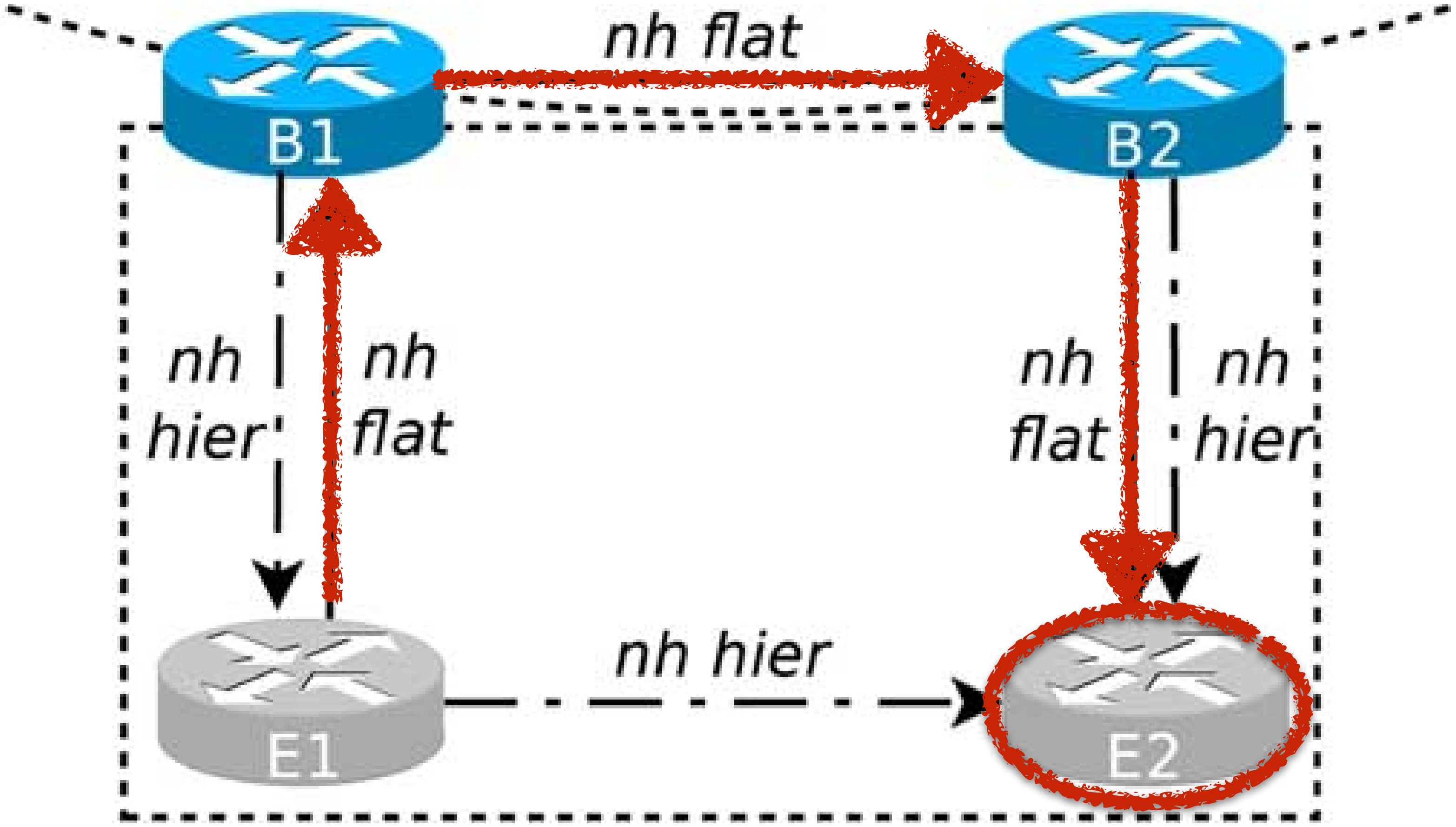
Migration Loop



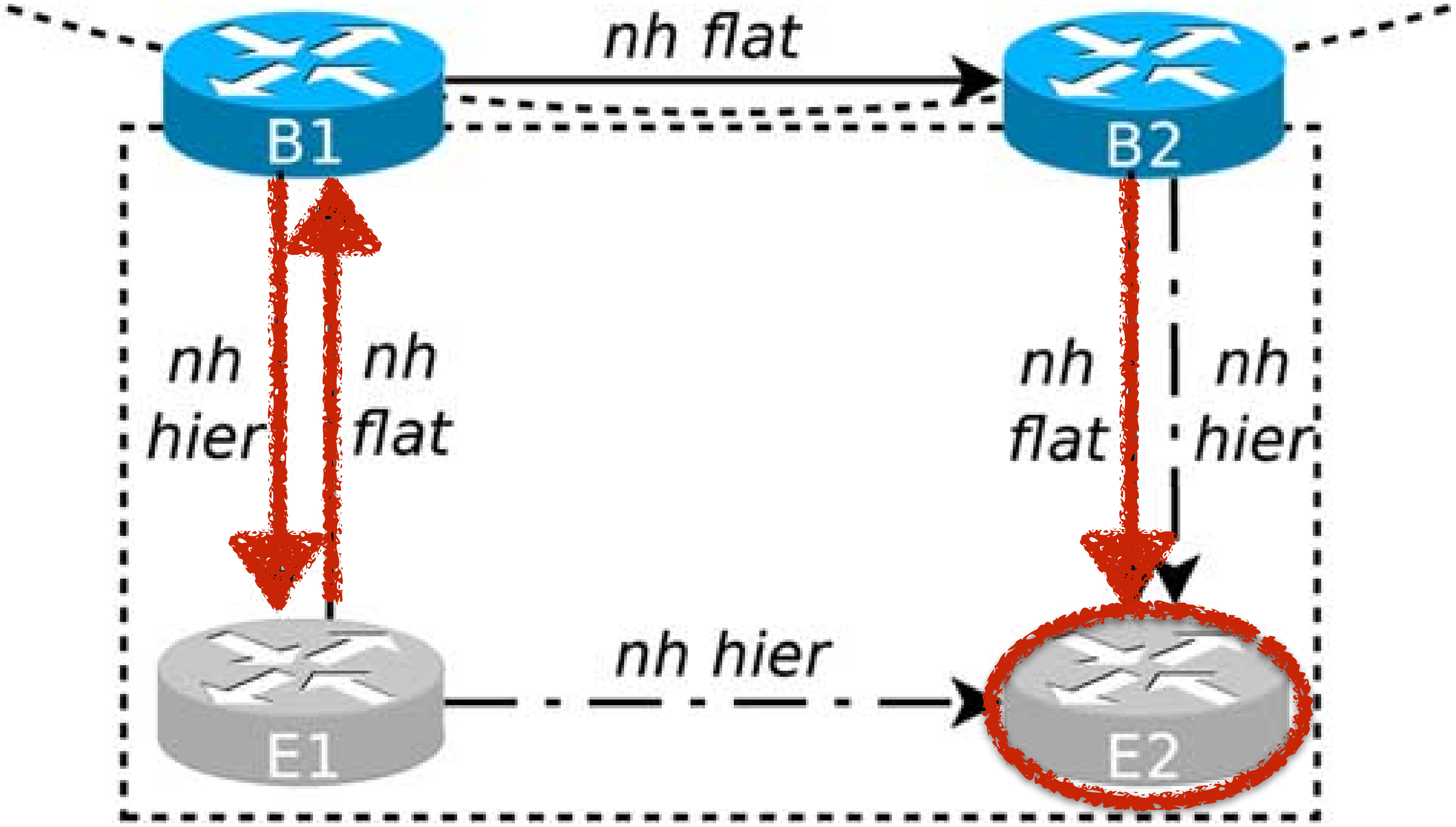
Migration Loop



Migration Loop



Migration Loop



Migration Problem

- given unicast IP network

Migration Problem

- given unicast IP network
- replace IGP configuration
    - from nh_{init} to nh_{final}

Migration Problem

- given unicast IP network
- replace IGP configuration
    - from nh_{init} to nh_{final}
- minimal configuration changes

Migration Problem

- given unicast IP network
- replace IGP configuration
  - from nh_{init} to nh_{final}
- minimal configuration changes
- no migration loops

Migration Scenarios

scenario	**IGP configuration changes**
protocol	protocol replacement
flat2hier	zone introduction
hier2flat	zone removal
hier2hier	zone reshaping
summarization	summarization introduction/removal

ships-in-the-night

- run separate routing protocols on one router

ships-in-the-night

- run separate routing protocols on one router
- share hardware and software resources

ships-in-the-night

- run separate routing protocols on one router
- share hardware and software resources
- but do not interact on a protocol level

Proposed methodology

- Seamless IGP Migration Methodology

Proposed methodology

- Seamless IGP Migration Methodology

    1. Compute a lossless router migration order

Proposed methodology

- Seamless IGP Migration Methodology

    1. Compute a lossless router migration order

    2. Introduce the final IGP configuration

Proposed methodology

- Seamless IGP Migration Methodology
  1. Compute a lossless router migration order
  2. Introduce the final IGP configuration
  3. Monitor the final IGP status

Proposed methodology

- Seamless IGP Migration Methodology

  1. Compute a lossless router migration order
  2. Introduce the final IGP configuration
  3. Monitor the final IGP status
  4. Progressively migrate routers

Proposed methodology

- Seamless IGP Migration Methodology

  1. Compute a lossless router migration order

  2. Introduce the final IGP configuration

  3. Monitor the final IGP status

  4. Progressively migrate routers

  5. Remove initial IGP configuration

Router migration ordering

- Given
  - initial and final next-hop functions
  - logical graph *G*
  - set of destinations *D*

Router migration ordering

- Given
  - initial and final next-hop functions
  - logical graph *G*
  - set of destinations *D*
- Compute router migration
  - no forwarding loops in *G* for $d \in D$

Router migration ordering

- Router Migration Ordering Problem is NP-complete

Router migration ordering

- Router Migration Ordering Problem is NP-complete
  - is in NP

Router migration ordering

- Router Migration Ordering Problem is NP-complete
  - is in NP
  - Reduction from 3-SAT

Router migration ordering

- Router Migration Ordering Problem is NP-complete
    - is in NP
    - Reduction from 3-SAT
        - e.g. $F = (x_1 \lor \bar{x}_2 \lor x_3) \land (\bar{x}_1 \lor \bar{x}_2 \lor x_3)$

Router migration ordering

- Router Migration Ordering Problem is NP-complete
  - is in NP
  - Reduction from 3-SAT
    - e.g. $F = (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3)$
    - is F satisfiable?

Router migration ordering

- given formula F

Router migration ordering

- given formula F
- transform into migration ordering instance

Router migration ordering

- given formula F
- transform into migration ordering instance
- show that

Router migration ordering

- given formula F
- transform into migration ordering instance
- show that
    - F satisfiable ⇒ loop free migration ordering

Router migration ordering

- given formula F
- transform into migration ordering instance
- show that
    - F satisfiable ⇒ loop free migration ordering
    - F not satisfiable ⇒ no loop free migration ordering

Router migration ordering

- central vertex P

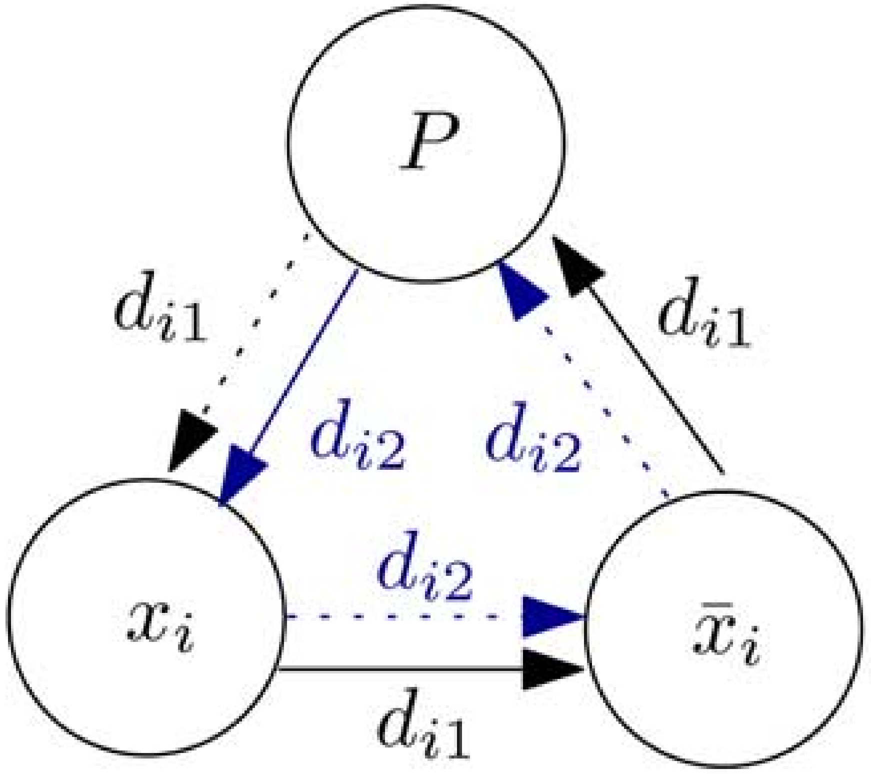
Router migration ordering

- central vertex P
- true: x_i migrated before P

Router migration ordering

- central vertex P
- true: x_i migrated before P
- false: x_i migrated after P

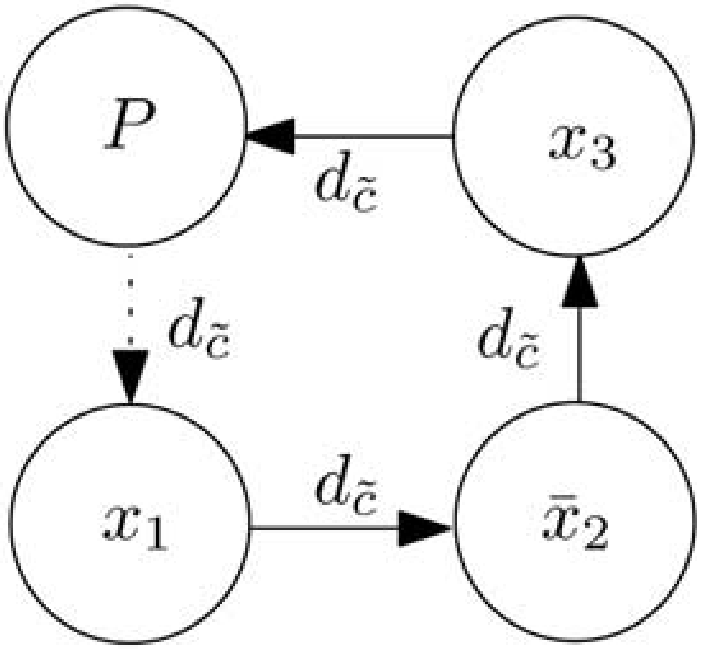
Router migration ordering



(a)

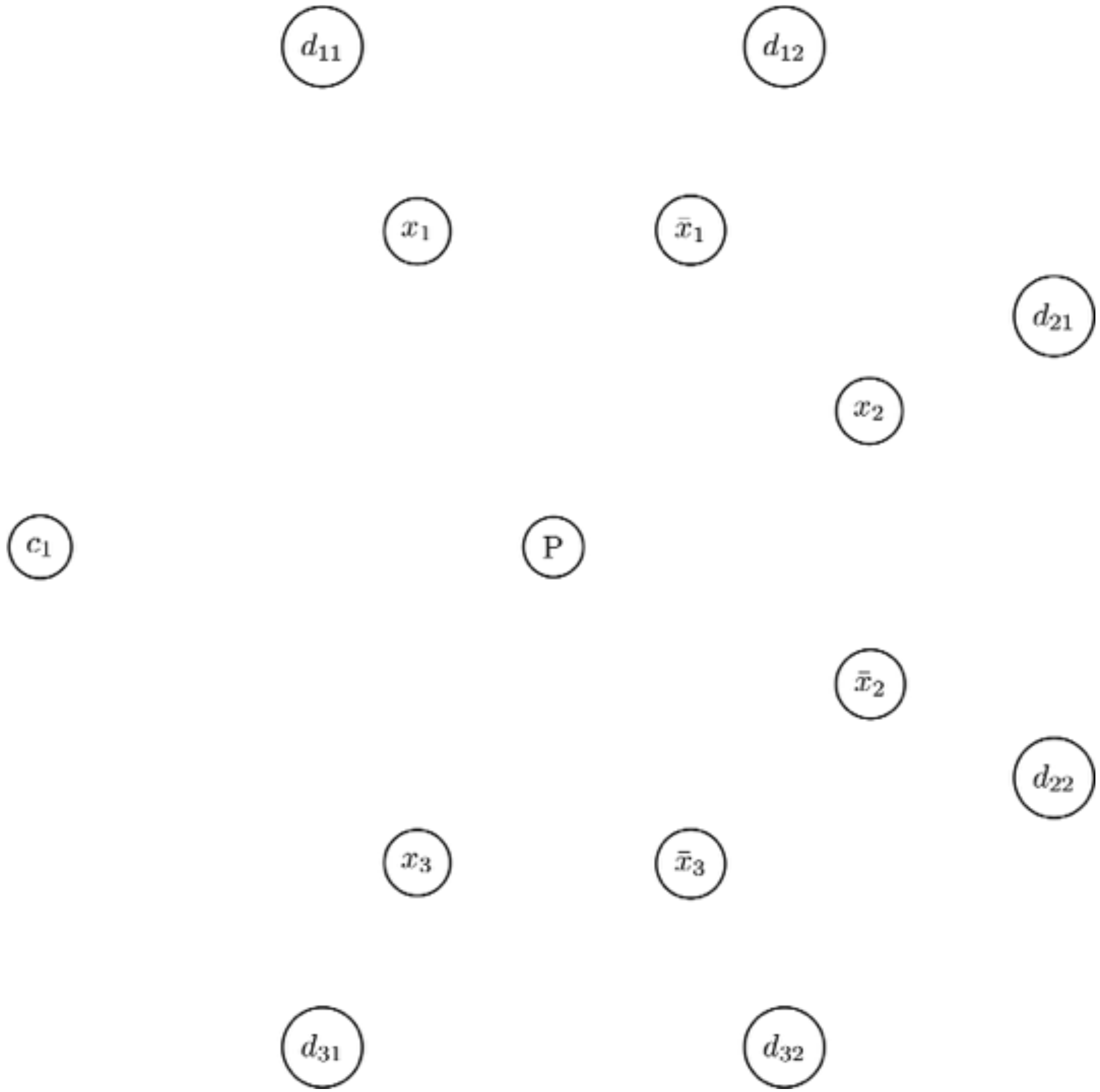
Router migration ordering

$$\tilde{C} = (X_1 \vee \bar{X}_2 \vee X_3)$$

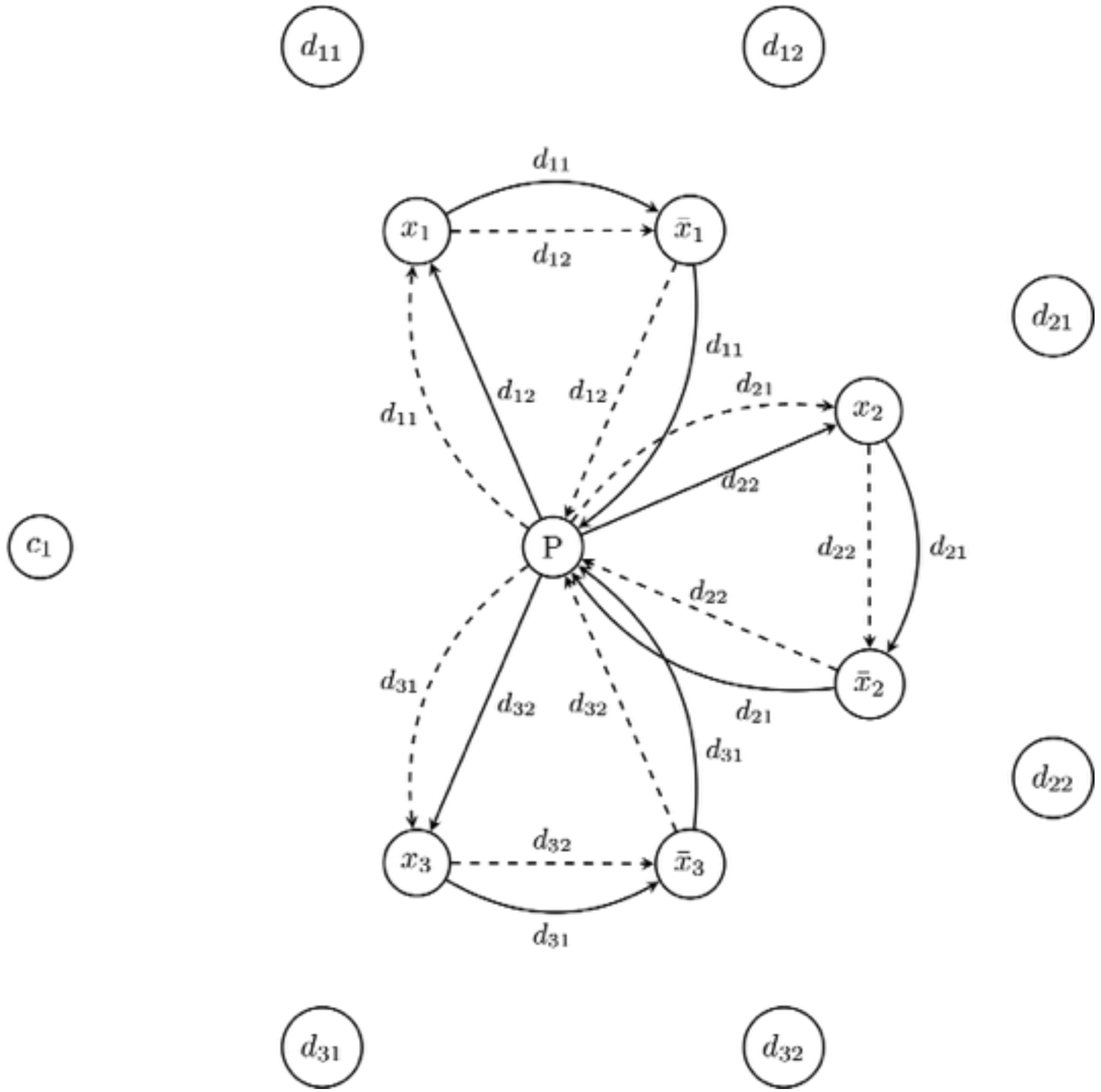


(b)

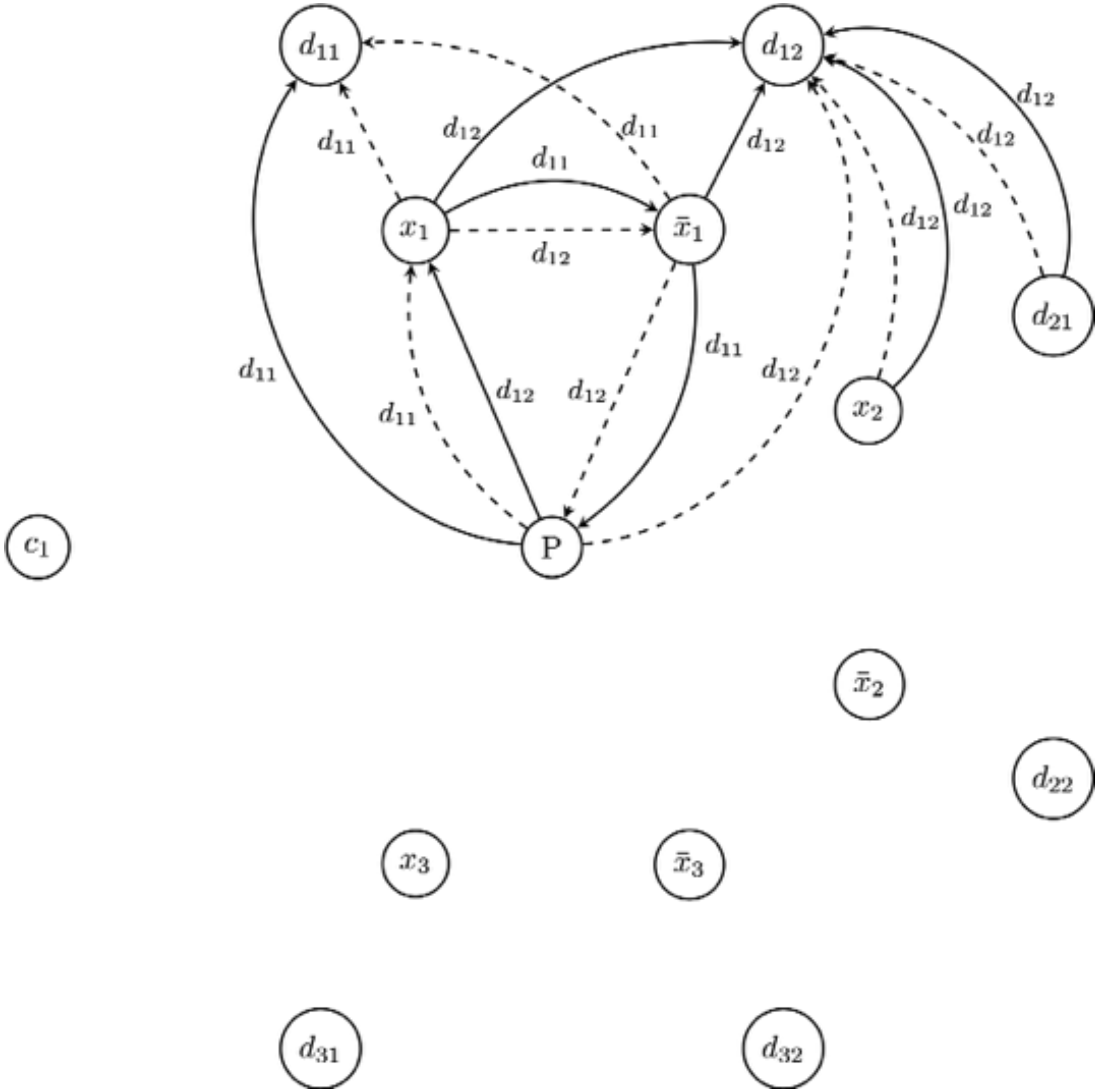
$$F = (x_1 \vee x_2 \vee x_3)$$



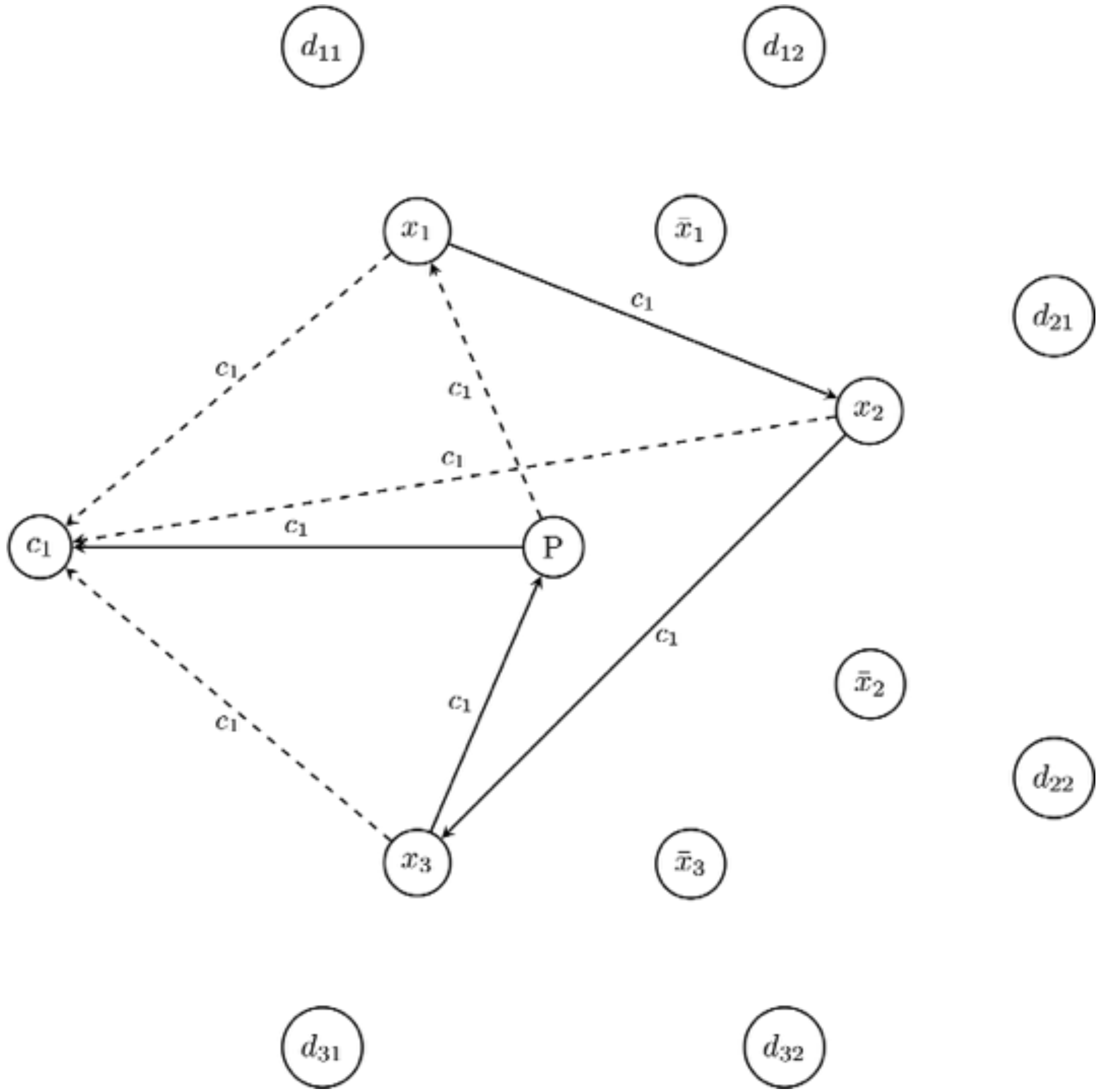
$$F = (x_1 \vee x_2 \vee x_3)$$



$$F = (x_1 \vee x_2 \vee x_3)$$



$$F = (x_1 \lor x_2 \lor x_3)$$



Loop enumeration Algorithm

Loop enumeration Algorithm

- inefficient algorithm

Loop enumeration Algorithm

- inefficient algorithm
- requires exponential time
  - cycles can be exponential in the number of nodes

Loop enumeration Algorithm

1. for each destination d

Loop enumeration Algorithm

1. for each destination d
    1. Build graph G_d

Loop enumeration Algorithm

1. for each destination d
    1. Build graph G_d
        - initial and final next hops for destination *d*

Loop enumeration Algorithm

1. for each destination d
    1. Build graph G_d
        - initial and final next hops for destination *d*
    2. for each loop in G_d create ordering constraint

Loop enumeration Algorithm

1. for each destination d
    1. Build graph G_d
        - initial and final next hops for destination *d*
    2. for each loop in G_d create ordering constraint
2. solve the problem using a Linear Programm

Routing Tree Heuristic

Routing Tree Heuristic

- computes constraints for each destination
    - migrate next-hop changing routers after
    - forwarding path is established

Routing Tree Heuristic

- computes constraints for each destination
    - migrate next-hop changing routers after
    - forwarding path is established
- polynomial respect to input size

Routing Tree Heuristic

- computes constraints for each destination
  - migrate next-hop changing routers after
  - forwarding path is established
- polynomial respect to input size
- not guaranteed to find a solution
  - rare in carefully designed networks

Routing Tree Heuristic

1. for each destination d

Routing Tree Heuristic

1. for each destination d
    1. greedy run to generate set S_d

Routing Tree Heuristic

1. for each destination d
    1. greedy run to generate set S_d
    2. generate set V_d

Routing Tree Heuristic

1. for each destination d
    1. greedy run to generate set S_d
    2. generate set V_d
    3. build graph G_d

Routing Tree Heuristic

1. for each destination d
    1. greedy run to generate set S_d
    2. generate set V_d
    3. build graph G_d
    4. constraints: migrate router after all its successors

Routing Tree Heuristic

1. for each destination d
    1. greedy run to generate set S_d
    2. generate set V_d
    3. build graph G_d
    4. constraints: migrate router after all its successors
2. topological sort of the final graph

Per-Destination Ordering

- per-router ordering does not exist or

Per-Destination Ordering

- per-router ordering does not exist or
- Routing Tree Heuristic does not find a solution

Per-Destination Ordering

- per-router ordering does not exist or
- Routing Tree Heuristic does not find a solution
- applied to problematic destinations only

Per-Destination Ordering

- per-router ordering does not exist or
- Routing Tree Heuristic does not find a solution
- applied to problematic destinations only
- use constraints generated in Routing Tree Heuristic

Method discussion

- always migrate one router after the other
  - why not migrate a subset of routers?
  - why not migrate only part of the routers?

Method discussion

- always migrate one router after the other
  - why not migrate a subset of routers?
  - why not migrate only part of the routers?
- other approach introducing version numbers
  - packets need to be adapted
  - when can I delete old configurations?

Evaluation

Dataset

- Rocketfuel project

Dataset

- Rocketfuel project
  - AS of different sizes

Dataset

- Rocketfuel project
  - AS of different sizes
  - smallest: 79 nodes and 294 edges

Dataset

- Rocketfuel project
  - AS of different sizes
  - smallest: 79 nodes and 294 edges
  - biggest: 315 nodes and 1944 edges

Dataset

- Rocketfuel project
  - AS of different sizes
  - smallest: 79 nodes and 294 edges
  - biggest: 315 nodes and 1944 edges
- pan-European research network (Geant)

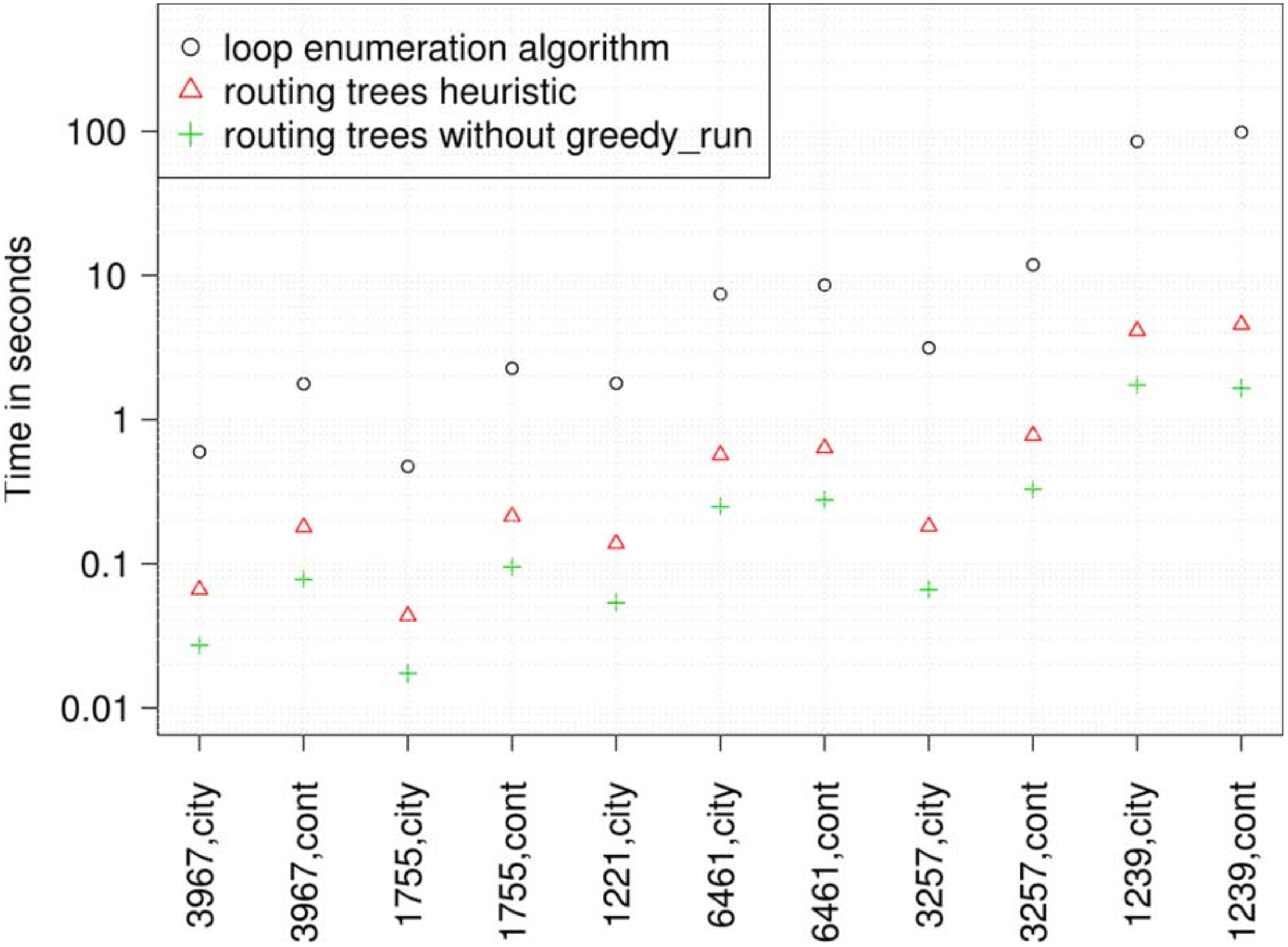
Dataset

- Rocketfuel project
  - AS of different sizes
  - smallest: 79 nodes and 294 edges
  - biggest: 315 nodes and 1944 edges
- pan-European research network (Geant)
  - 36 routers and 53 links

Dataset

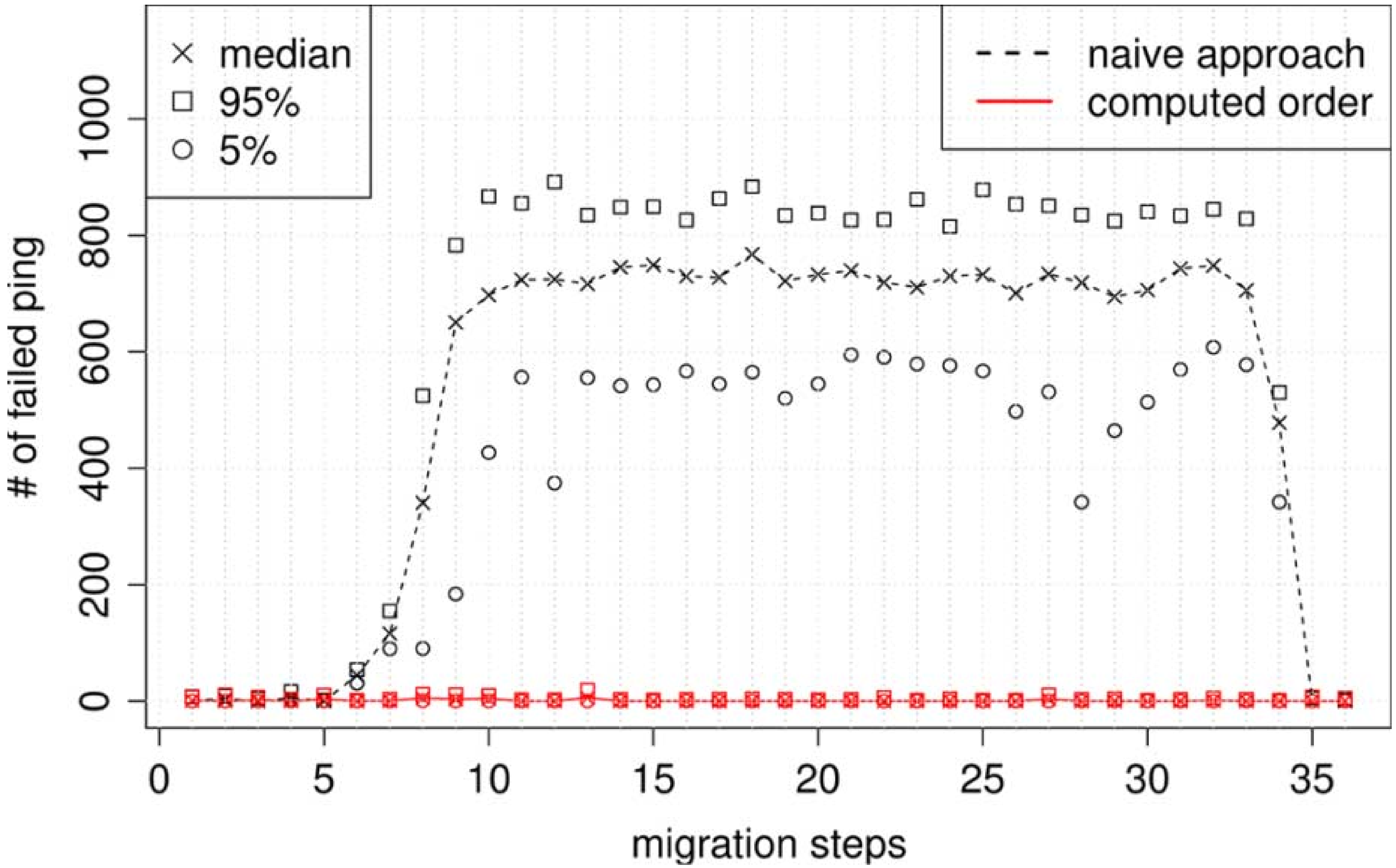
- Rocketfuel project
  - AS of different sizes
  - smallest: 79 nodes and 294 edges
  - biggest: 315 nodes and 1944 edges
- pan-European research network (Geant)
  - 36 routers and 53 links
  - emulation

Algorithms compared



Time taken to compute an ordering in flat2hier (Rocketfuel topologie)

Packet loss



Packet loss during flat2hier migration (Geant topology)

Summary

Summary

- changing IGP configurations can lead to loops

Summary

- changing IGP configurations can lead to loops
- proposed a methodology
    - router migration ordering

Summary

- changing IGP configurations can lead to loops
- proposed a methodology
  - router migration ordering
- finding loop free migration ordering is NP-complete

Summary

- changing IGP configurations can lead to loops
- proposed a methodology
    - router migration ordering
- finding loop free migration ordering is NP-complete
- loop enumeration algorithm

Summary

- changing IGP configurations can lead to loops
- proposed a methodology
    - router migration ordering
- finding loop free migration ordering is NP-complete
- loop enumeration algorithm
- routing tree heuristic

Q&A