Scalable Routing for Tag-Based Information-Centric Networking

ICN 2014

Michele Papalini (University of Lugano)
Antonio Carzaniga (University of Lugano)
Koorosh Khazaei (University of Lugano)
Alexander L. Wolf (Imperial College London)

Our Vision

Our Vision

Addressing Information

Our Vision

Addressing Information

descriptors:

```
{ICN14, conference, Paris}
```

Our Vision

Addressing Information

descriptors:

`{ICN14, conference, Paris}`

1. expressive ($>$ names)

Our Vision

Addressing Information

descriptors:

`{ICN14, conference, Paris}`

1. expressive ($>$ names)
2. user-defined

Our Vision

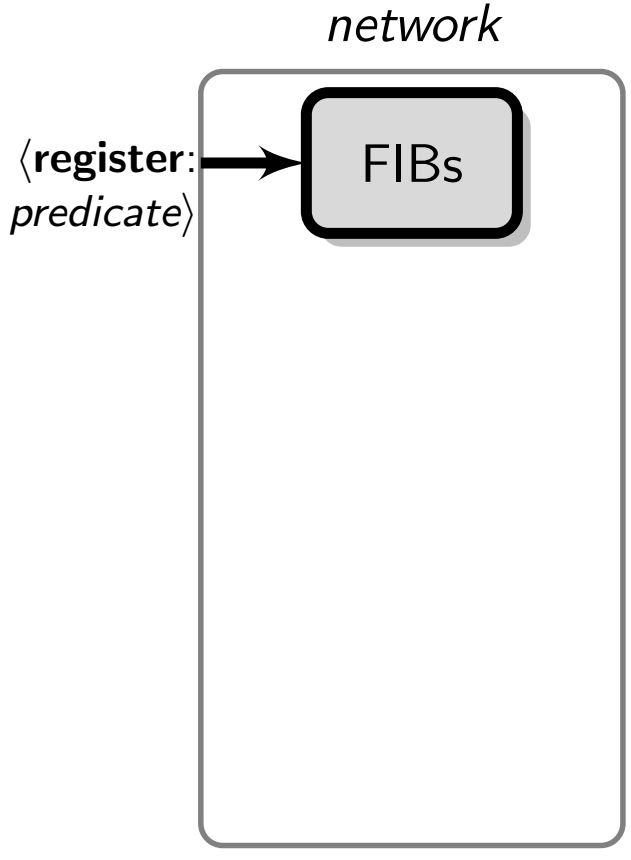
Addressing Information

descriptors:

`{ICN14, conference, Paris}`

1. expressive ($>$ names)
2. user-defined

Rich Service Model



Our Vision

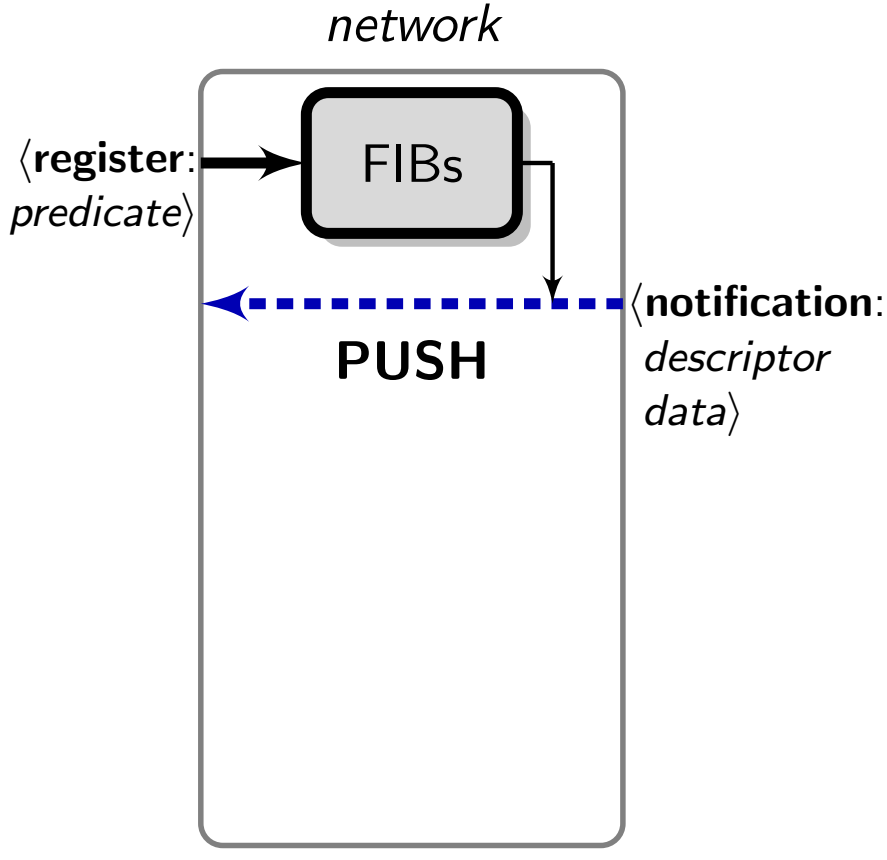
Addressing Information

descriptors:

`{ICN14, conference, Paris}`

1. expressive ($>$ names)
2. user-defined

Rich Service Model



Our Vision

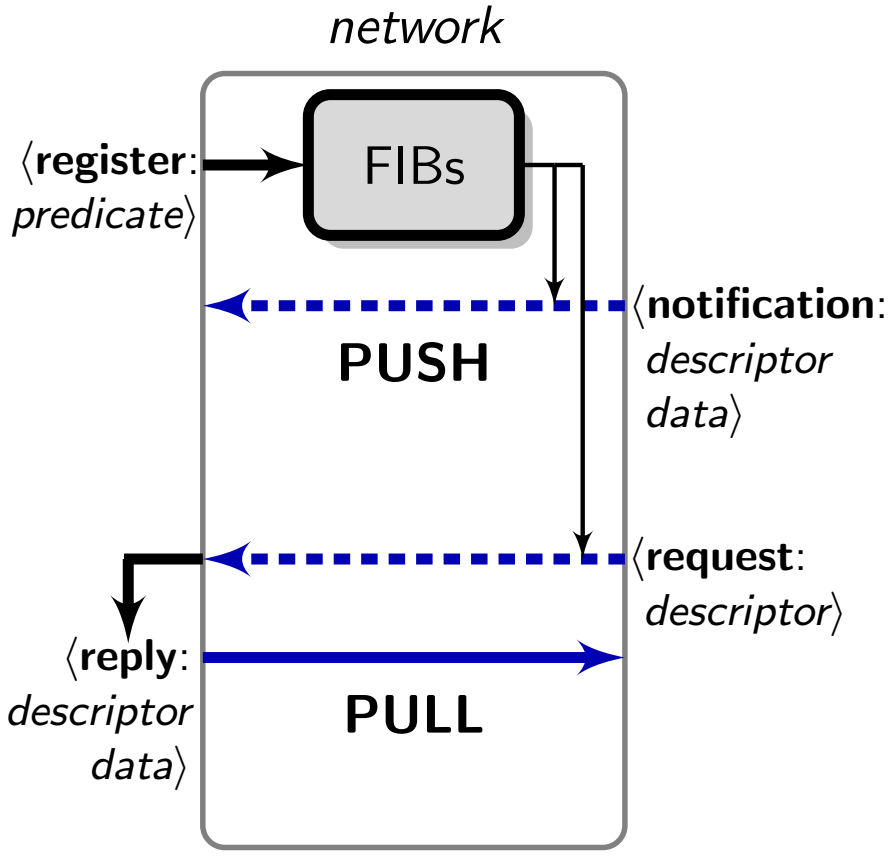
Addressing Information

descriptors:

`{ICN14, conference, Paris}`

1. expressive ($>$ names)
2. user-defined

Rich Service Model



Architectural Principles: Addressing Scheme

Architectural Principles: Addressing Scheme

Content Descriptor

set of **tags**

D0: `{ICN14, conference}`

Architectural Principles: Addressing Scheme

Content Descriptor

set of **tags**

D0: `{ICN14, conference}`

matching: **subset** relation ($\subseteq$)

Architectural Principles: Addressing Scheme

Content Descriptor

set of **tags**

D0: {ICN14, conference}

matching: **subset** relation ($\subseteq$)

D1: {ICN14, conference, banquet}

D2: {ICN14, paper, routing}

Architectural Principles: Addressing Scheme

Content Descriptor

set of **tags**

D0: {ICN14, conference}

matching: **subset** relation ($\subseteq$)

D1: {ICN14, conference, banquet} ✓

D2: {ICN14, paper, routing}

Architectural Principles: Addressing Scheme

Content Descriptor

set of **tags**

D0: {ICN14, conference}

matching: **subset** relation ($\subseteq$)

D1: {ICN14, conference, banquet} ✓

D2: {ICN14, paper, routing} ✗

Architectural Principles: Addressing Scheme

Architectural Principles: Addressing Scheme

Content Descriptor

Function: Describe content, meta-data, user interests, ...

Architectural Principles: Addressing Scheme

Content Descriptor

Function: Describe content, meta-data, user interests, ...

Characteristics: True content-based
User-defined
Location independent

Architectural Principles: Addressing Scheme

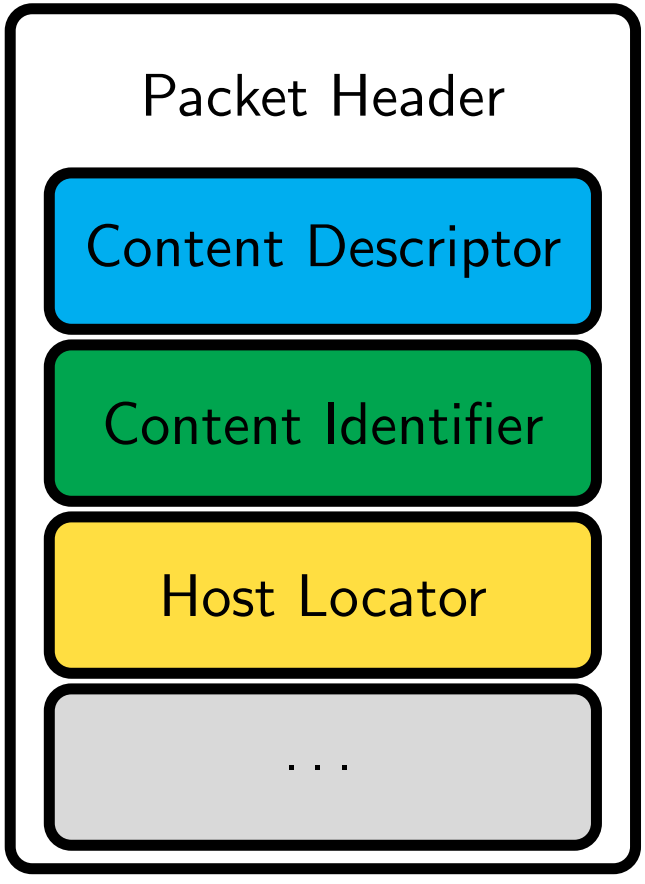
Content Descriptor

Function: Describe content, meta-data, user interests, ...

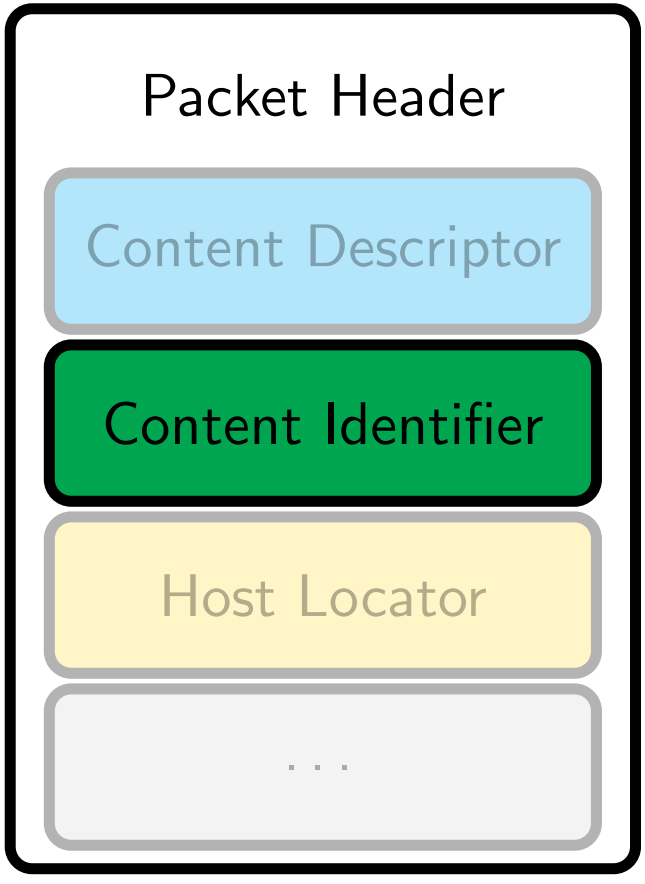
Characteristics: True content-based
User-defined
Location independent

Issues: Not a unique identifier
Addressing objects/blocks?
High-throughput forwarding?

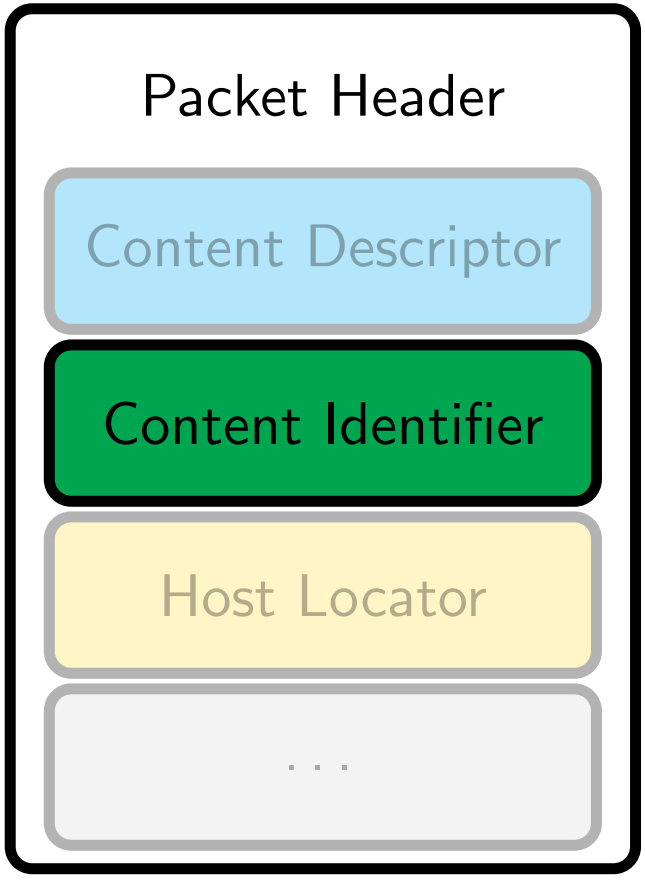
Architectural Principles: Addressing Scheme



Architectural Principles: Addressing Scheme

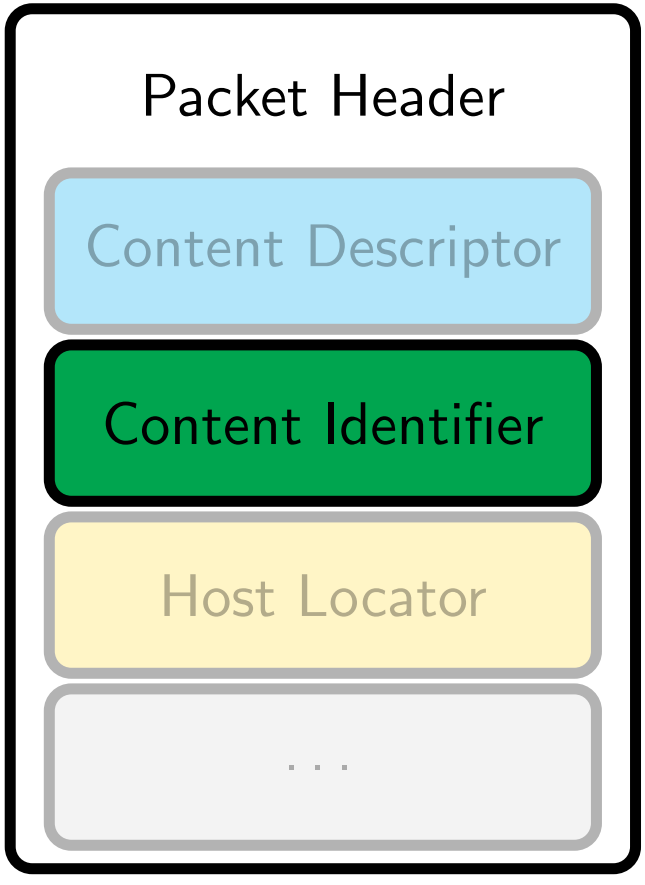


Architectural Principles: Addressing Scheme



Function: Identify a specific object/block

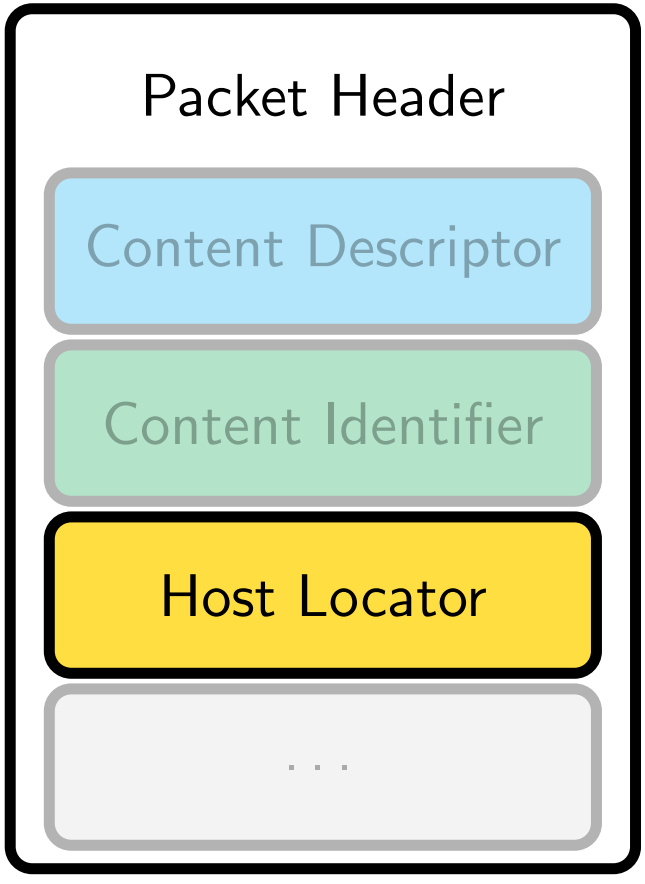
Architectural Principles: Addressing Scheme



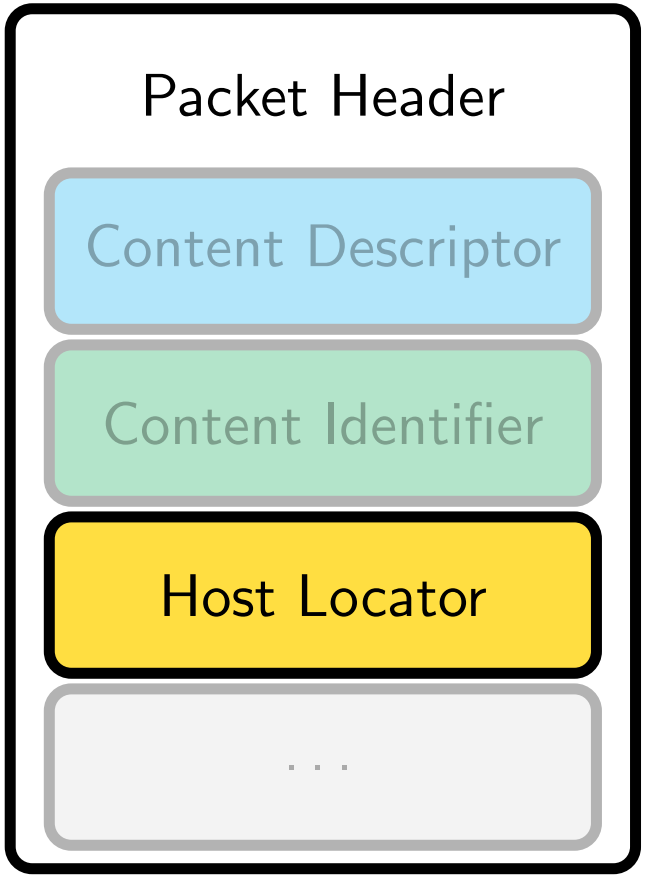
Function: Identify a specific object/block

Characteristics: Location independent
Globally unique

Architectural Principles: Addressing Scheme

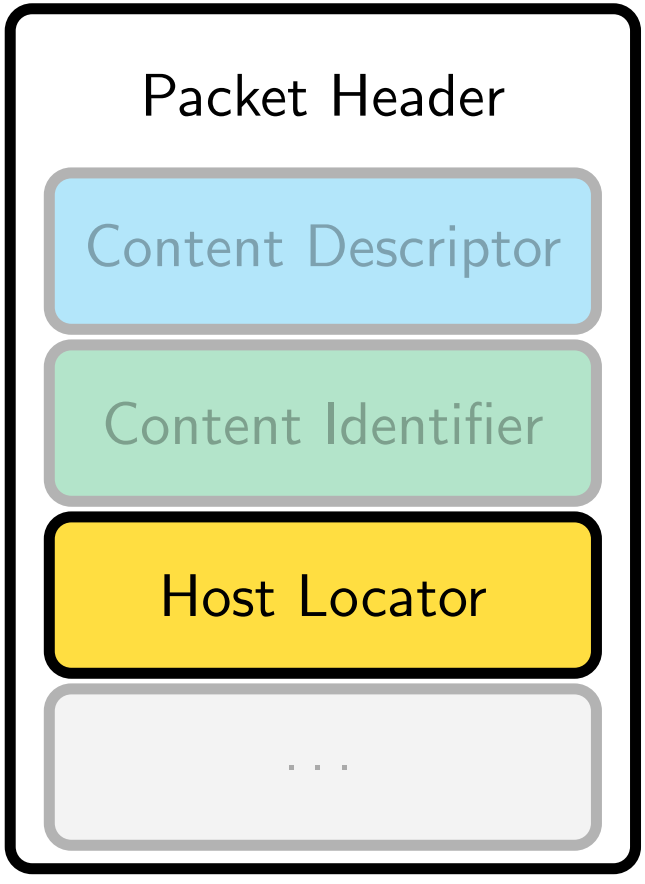


Architectural Principles: Addressing Scheme



Function: Locate and route to a specific host

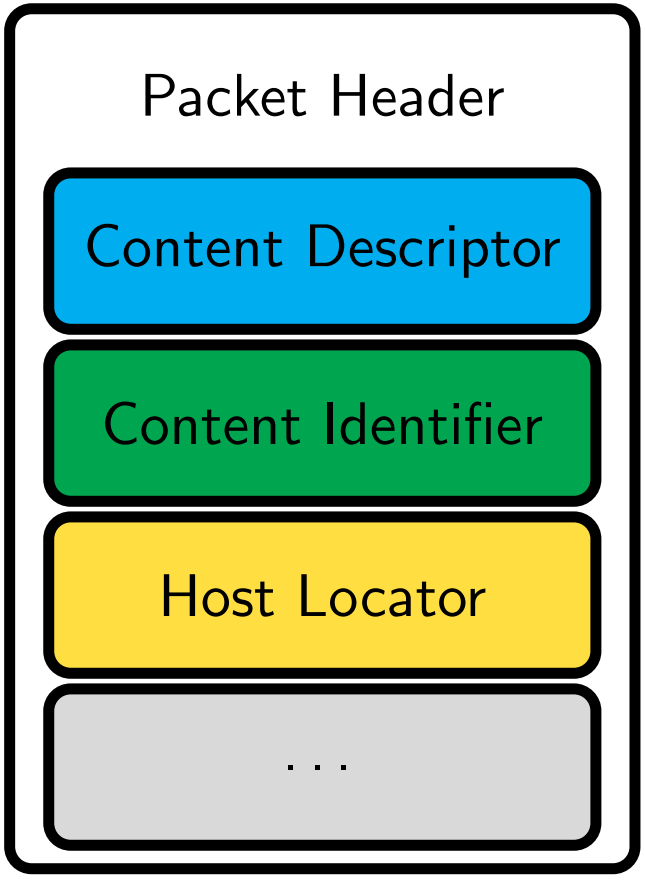
Architectural Principles: Addressing Scheme



Function: Locate and route to a specific host

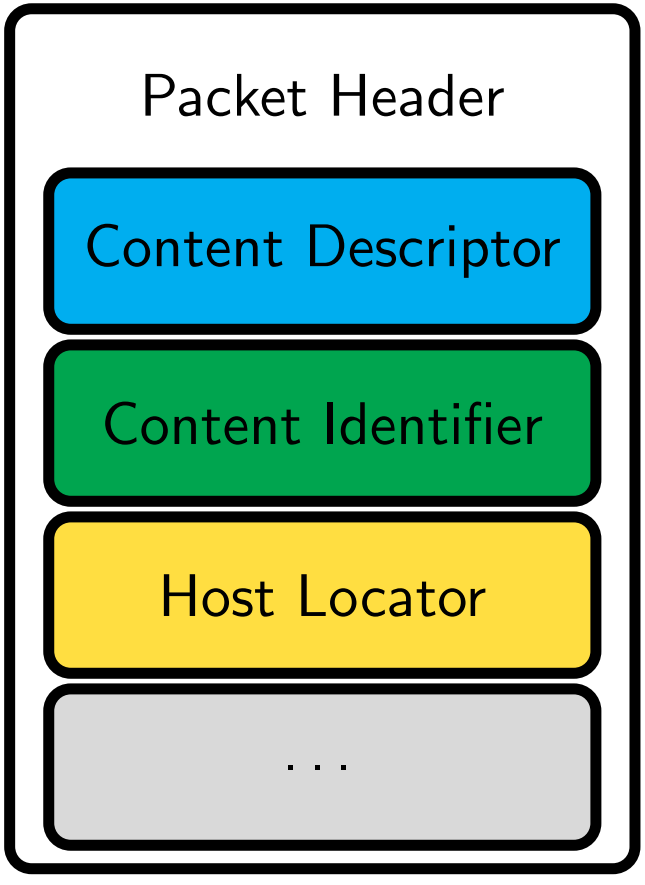
Characteristics: Network-defined

Architectural Principles: Addressing Scheme



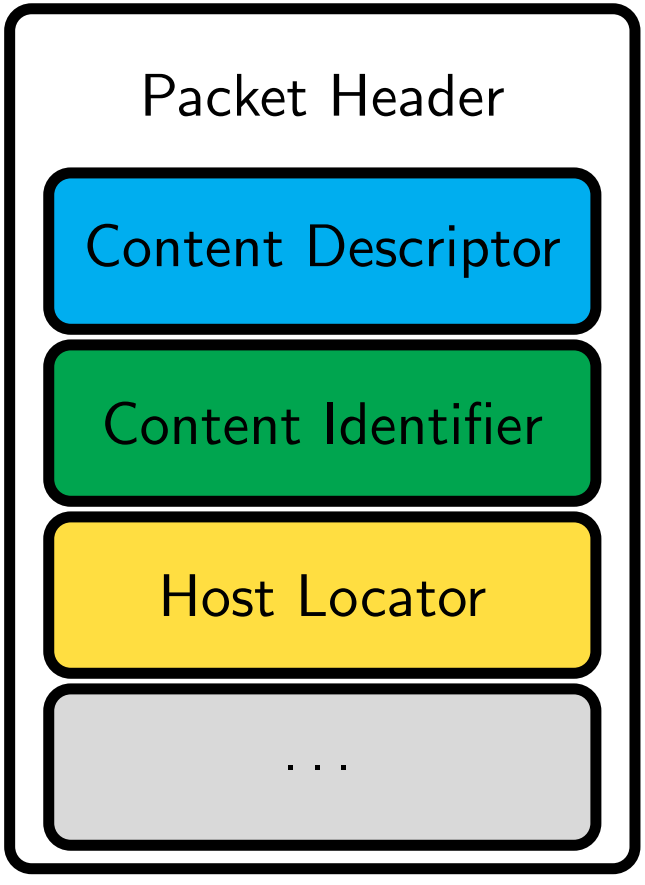
Network **Transport** **Application**

Architectural Principles: Addressing Scheme



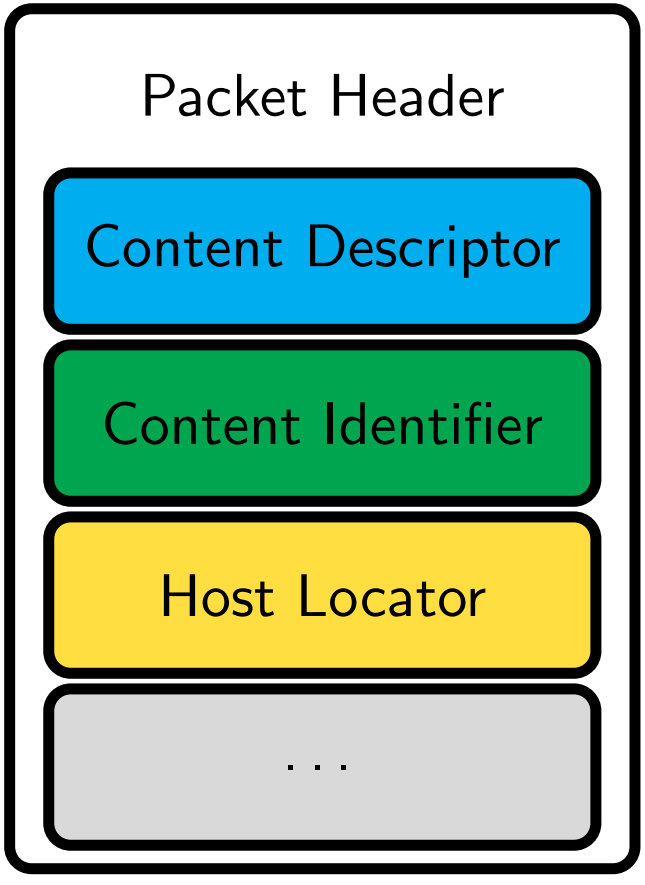
Network	Transport	Application
Content-based forwarding	—	Describe content

Architectural Principles: Addressing Scheme



Network	Transport	Application
Content-based forwarding	—	Describe content
Caching	Stream ID, object ID, blocks	—

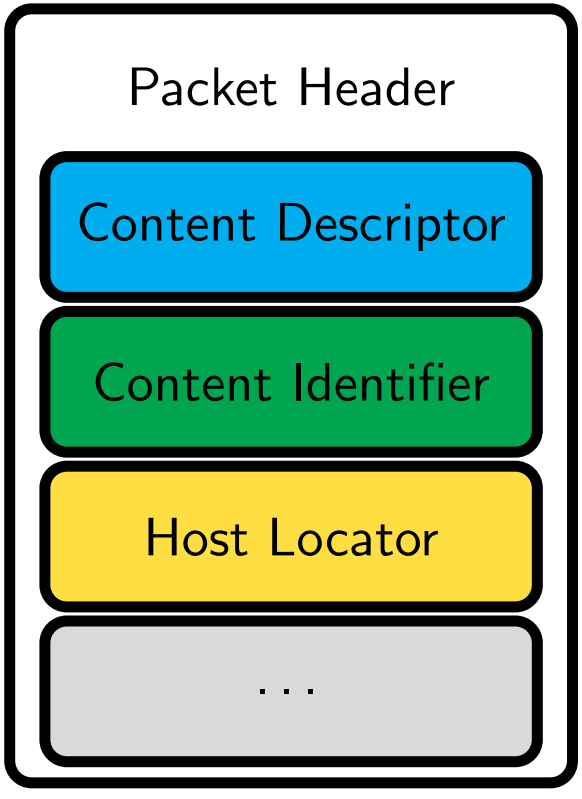
Architectural Principles: Addressing Scheme



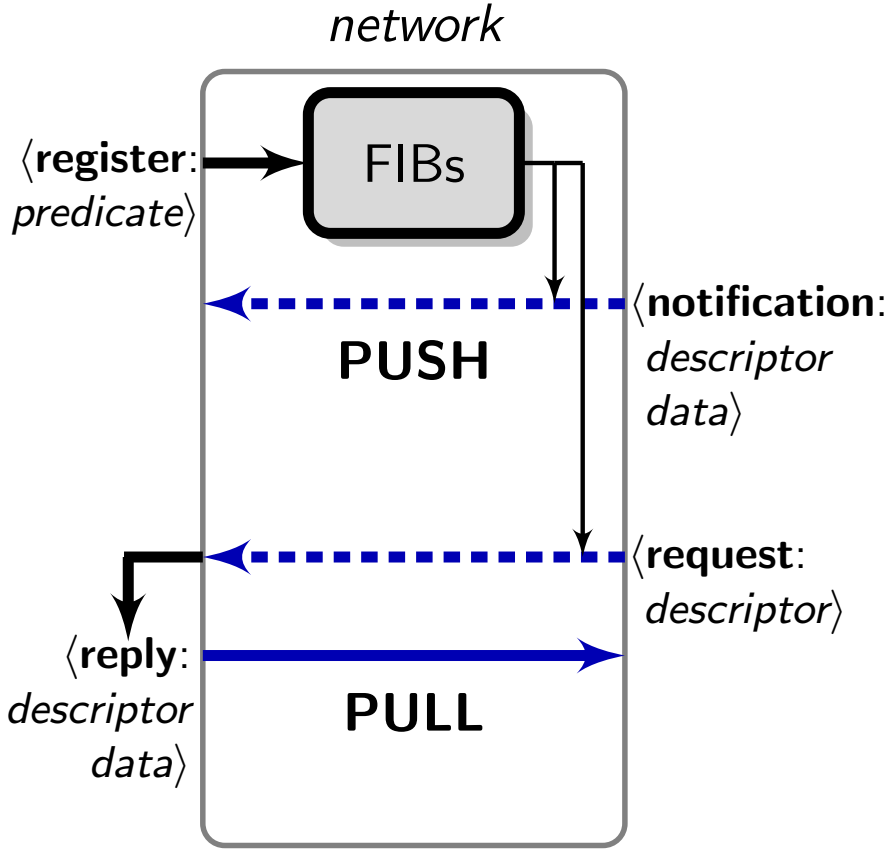
Network	Transport	Application
Content-based forwarding	—	Describe content
Caching	Stream ID, object ID, blocks	—
Locator-based forwarding	End-point address	—

Our Vision

Address Information



Rich Service Model



From Architecture to Engineering

- Routing protocol based on trees
  - Supports rich "push/pull" primitives
  - Supports both descriptors and very efficient locators

From Architecture to Engineering

- Routing protocol based on trees
  - Supports rich "push/pull" primitives
  - Supports both descriptors and very efficient locators
  - Heuristics to build trees

From Architecture to Engineering

- Routing protocol based on trees
  - Supports rich "push/pull" primitives
  - Supports both descriptors and very efficient locators
  - Heuristics to build trees
  - Hierarchical (inter and intra AS) routing

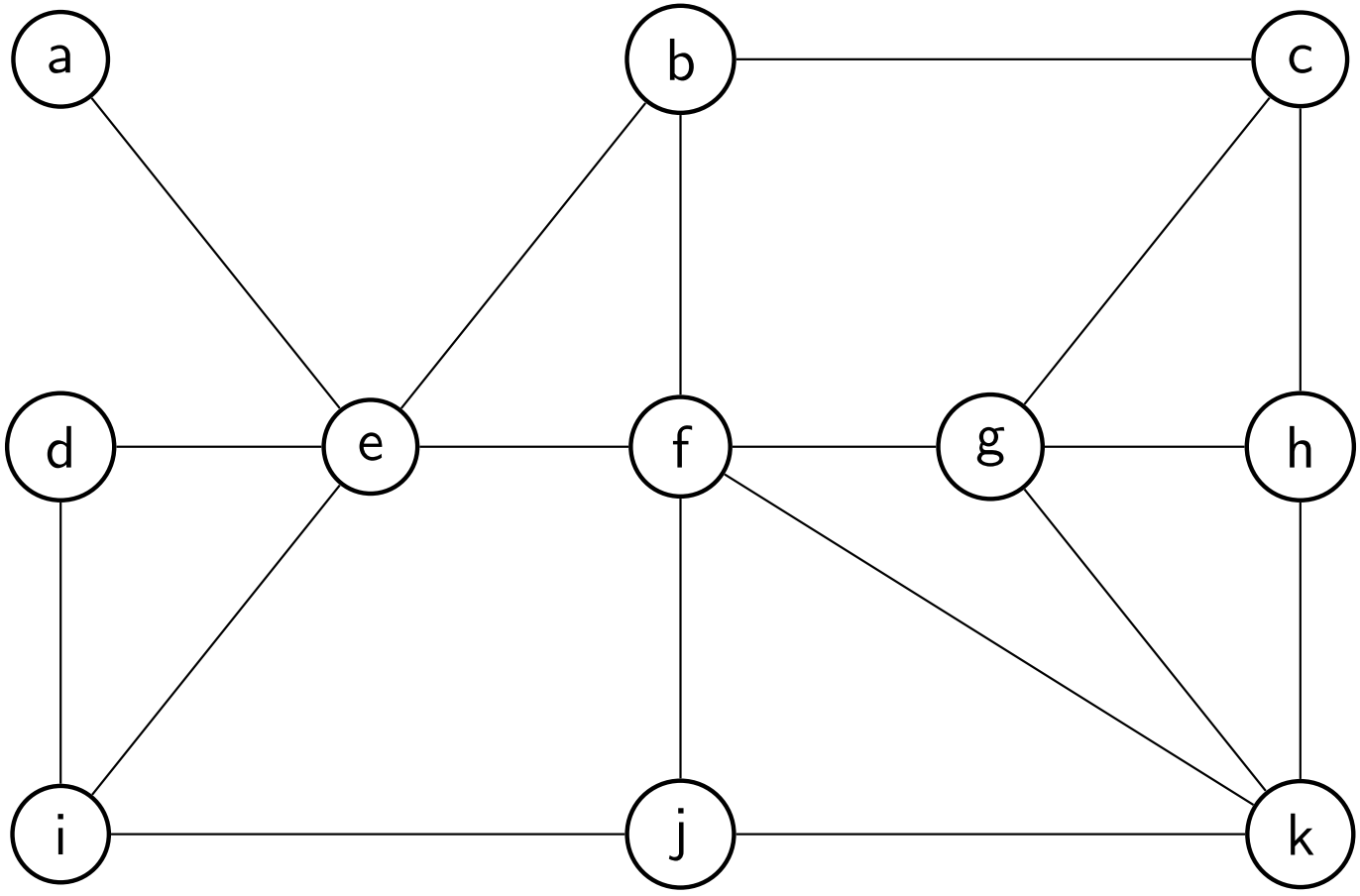
From Architecture to Engineering

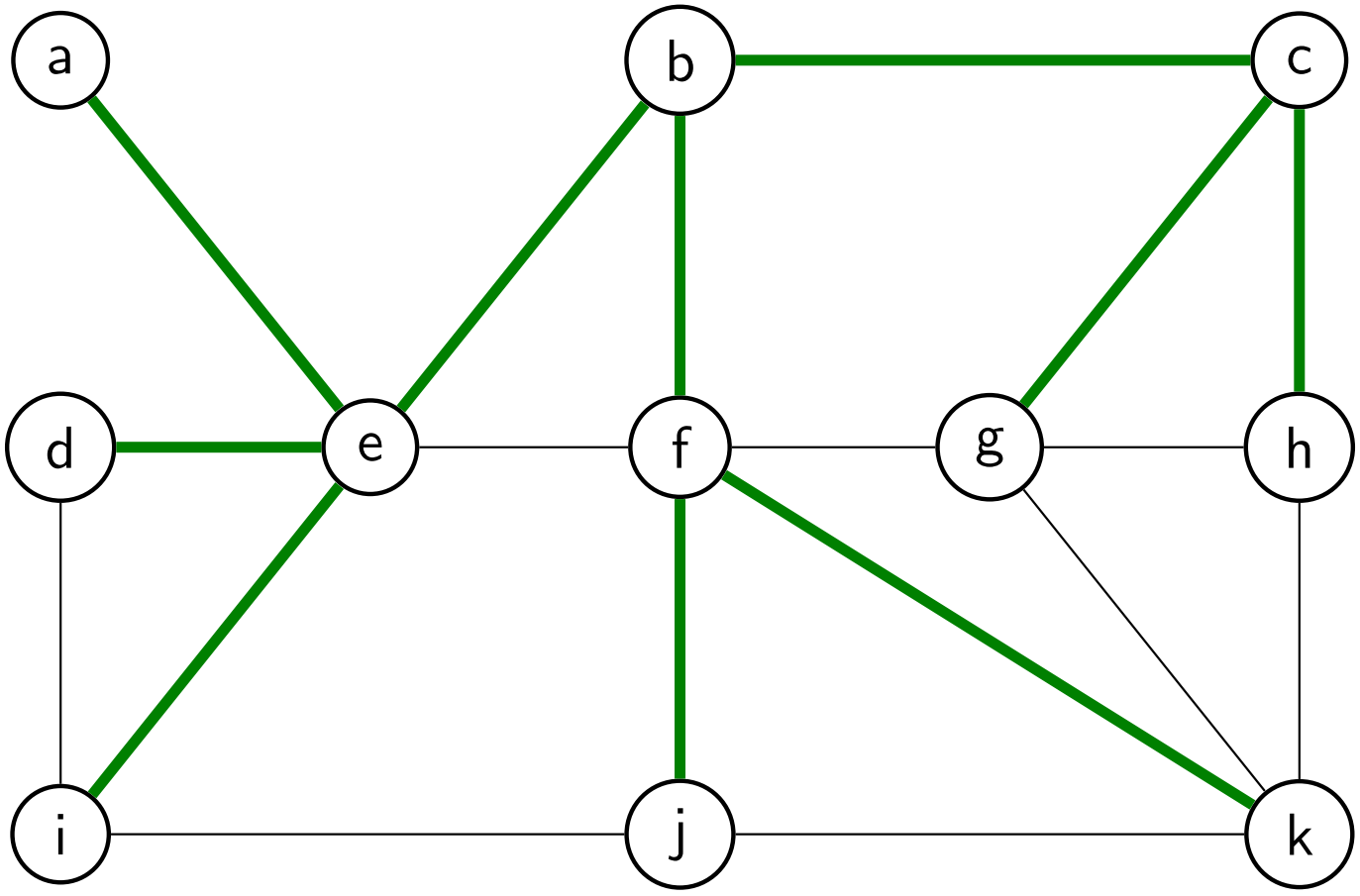
- Routing protocol based on trees
  - Supports rich "push/pull" primitives
  - Supports both descriptors and very efficient locators
  - Heuristics to build trees
  - Hierarchical (inter and intra AS) routing
- Data structure for RIBs
  - Efficient compression
  - Maintenance algorithm

From Architecture to Engineering

- Routing protocol based on trees
  - Supports rich "push/pull" primitives
  - Supports both descriptors and very efficient locators
  - Heuristics to build trees
  - Hierarchical (inter and intra AS) routing
- Data structure for RIBs
  - Efficient compression
  - Maintenance algorithm
- Scalability analysis

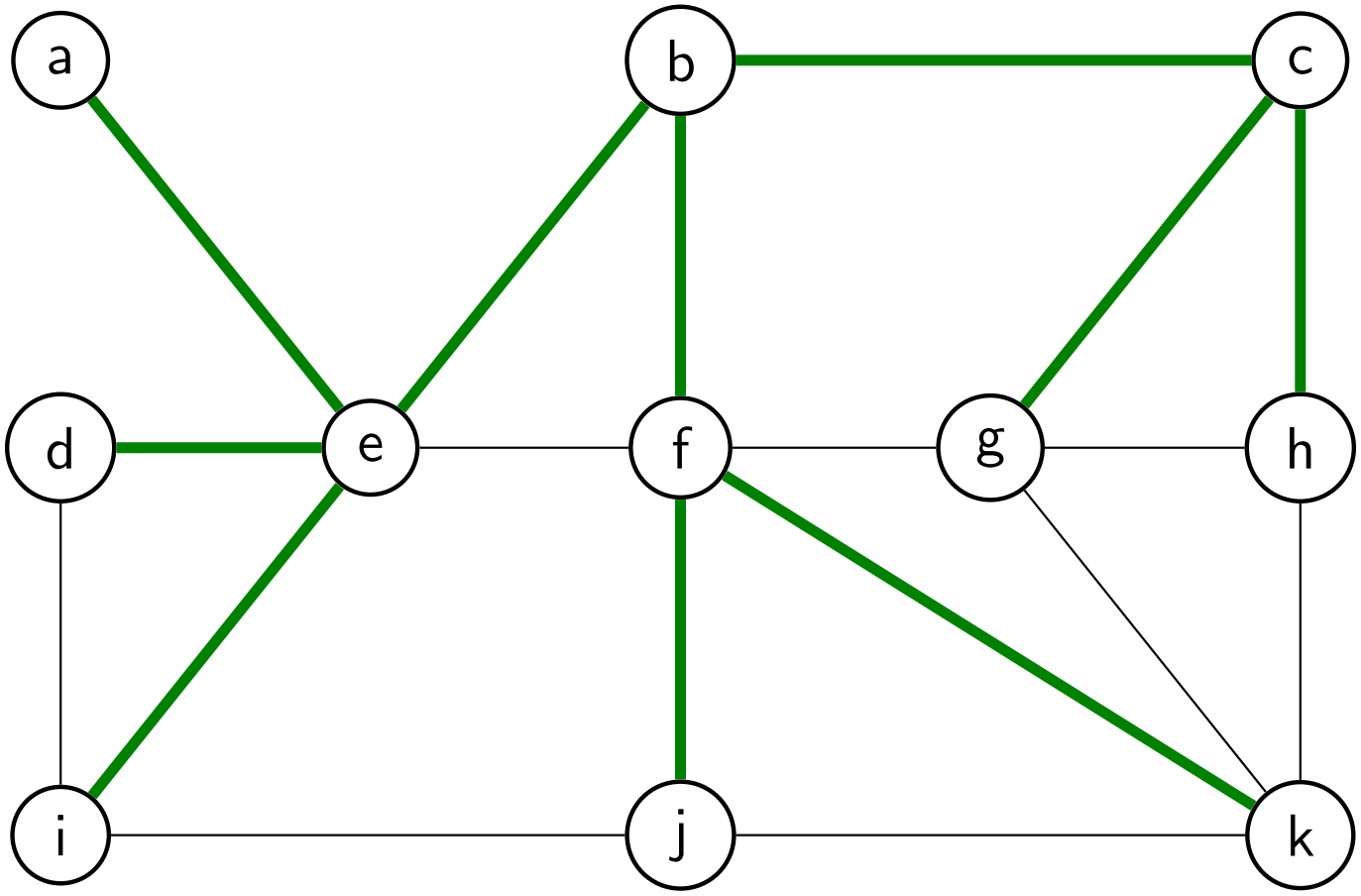
Routing on One Tree


a
b
c
d
e
f
g
h
i
j
k


a
b
c
d
e
f
g
h
i
j
k

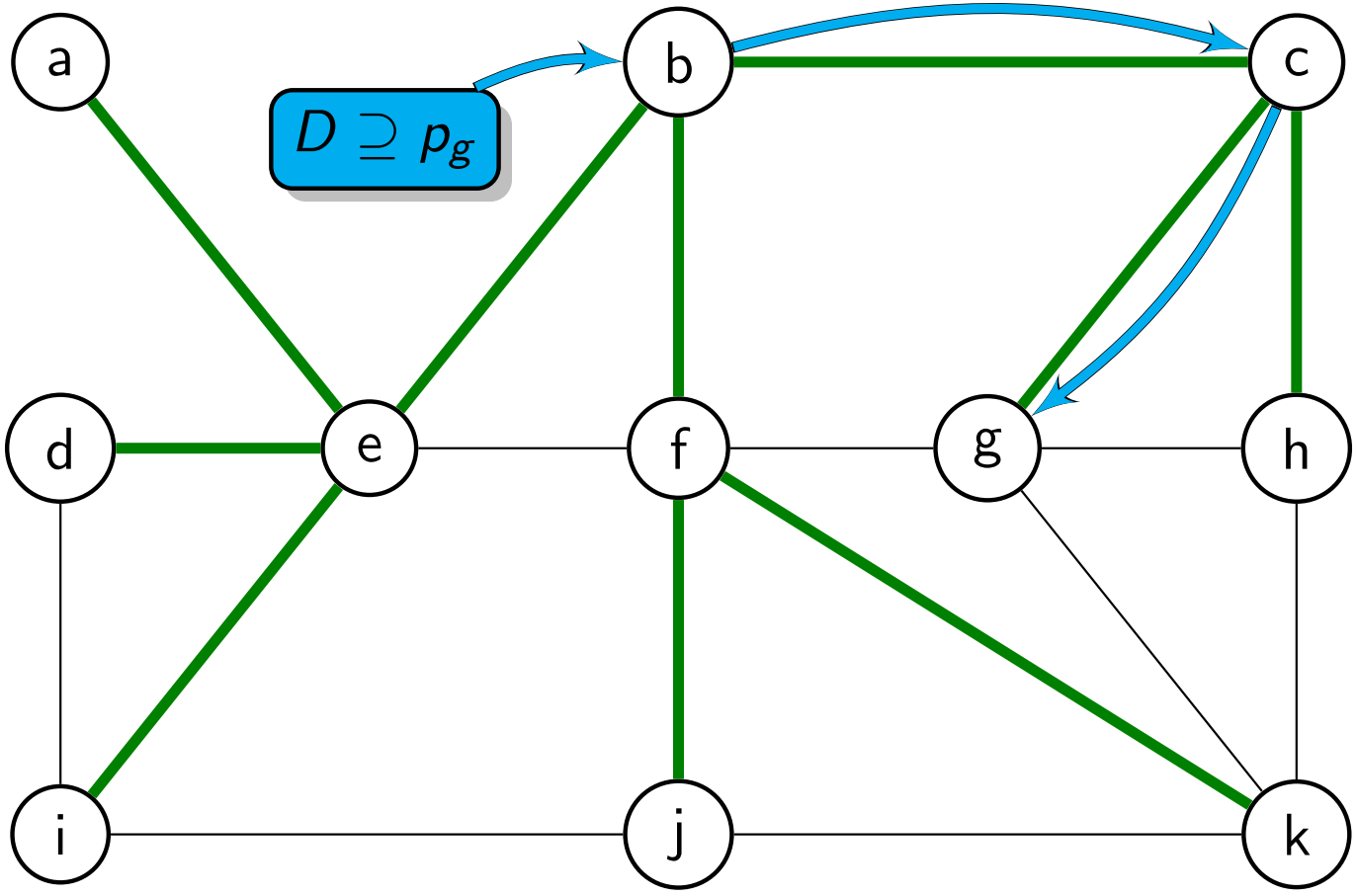
Descriptor-Based Forwarding

FIB router b	
tree T, next-hop w	predicate $P_{T,w}$
T_{green}, c	$p_c \vee p_g \vee p_h$
T_{green}, f	$p_f \vee p_j \vee p_k$
T_{green}, e	$p_e \vee p_a \vee p_d \vee p_i$

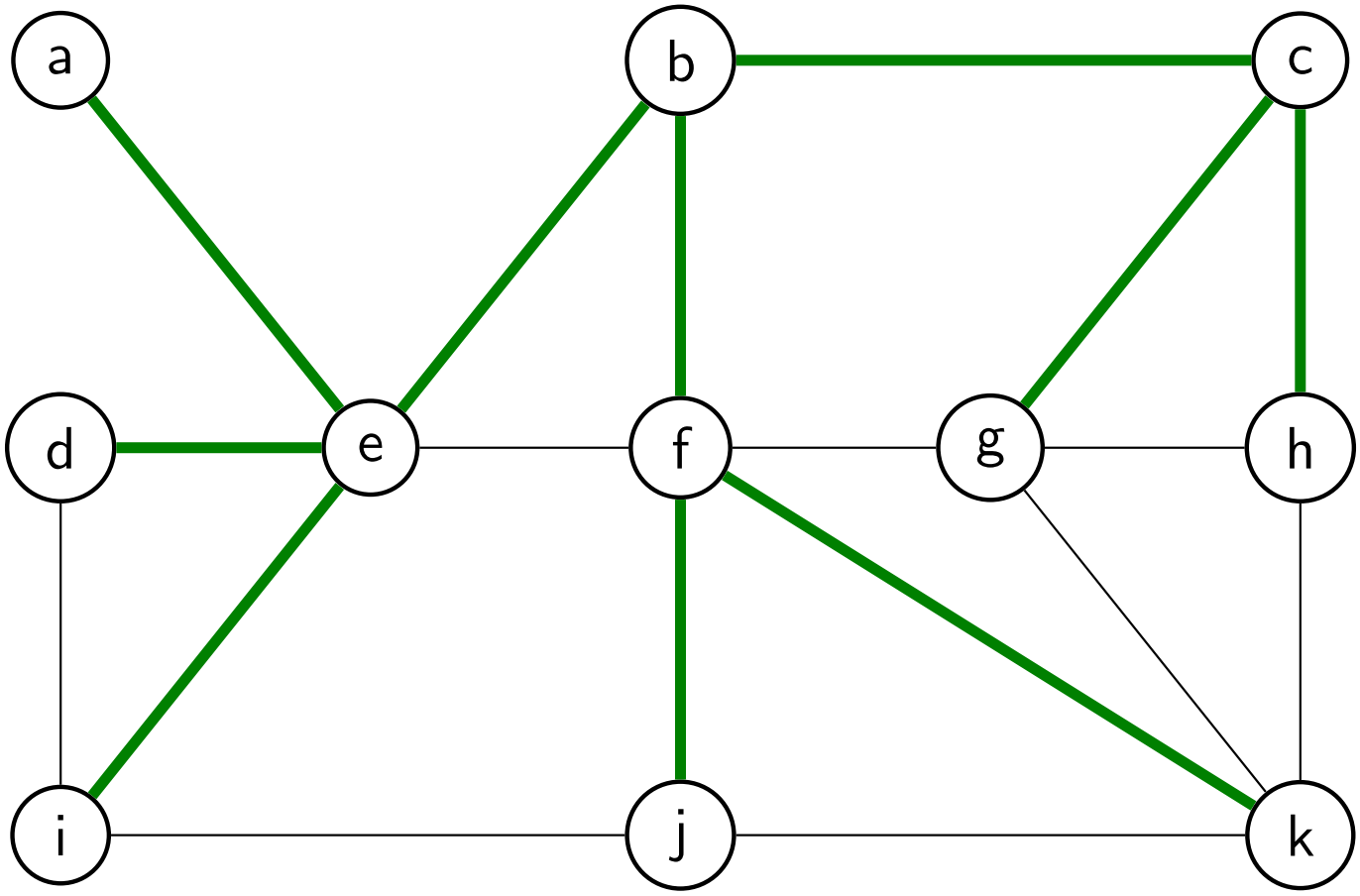


Descriptor-Based Forwarding

FIB router b	
tree T, next-hop w	**predicate** $P_{T,w}$
T_{green}, c	$p_c \vee p_g \vee p_h$
T_{green}, f	$p_f \vee p_j \vee p_k$
T_{green}, e	$p_e \vee p_a \vee p_d \vee p_i$



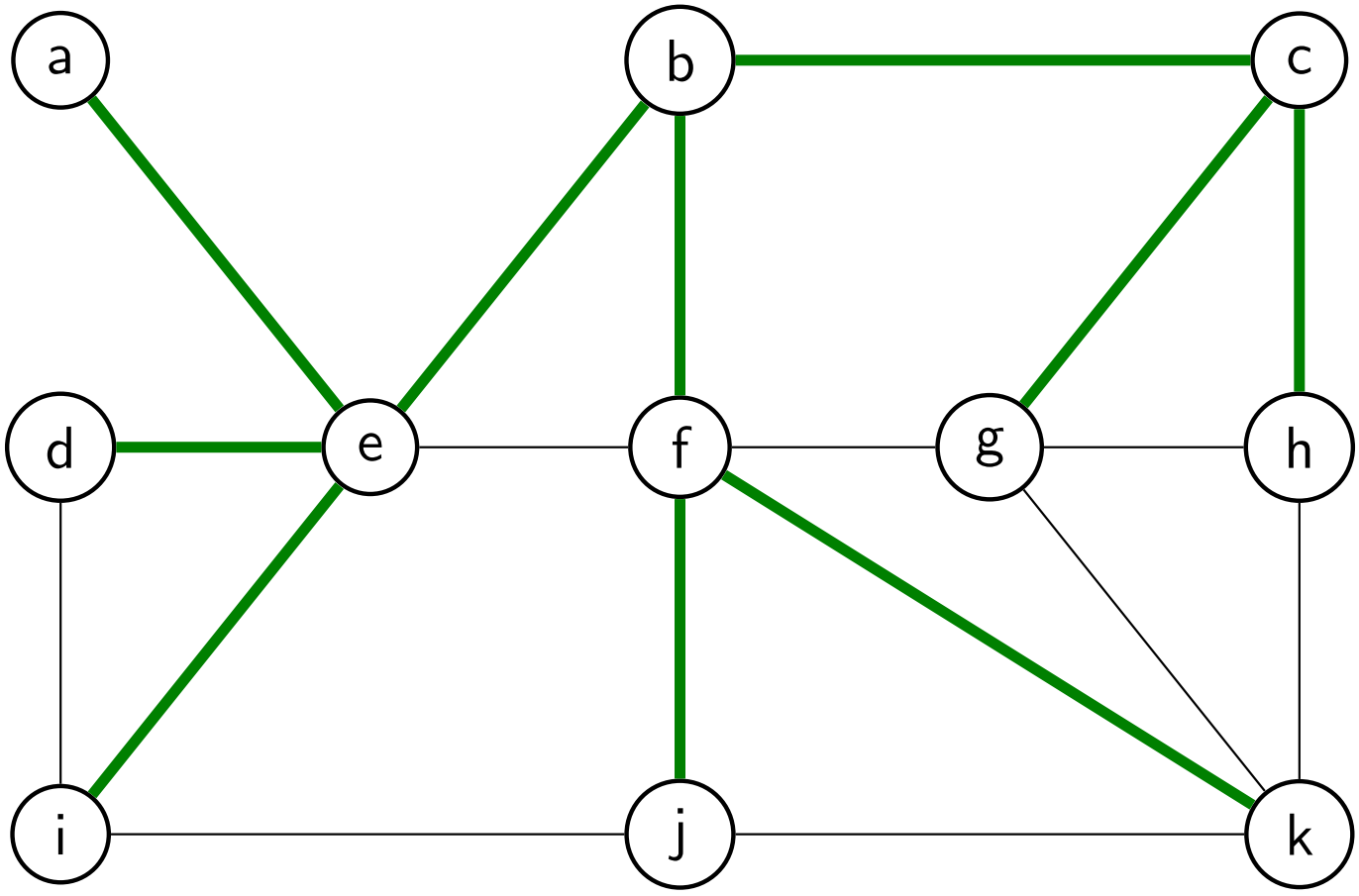
Locator-Based Forwarding



Locator-Based Forwarding

TZ-labels

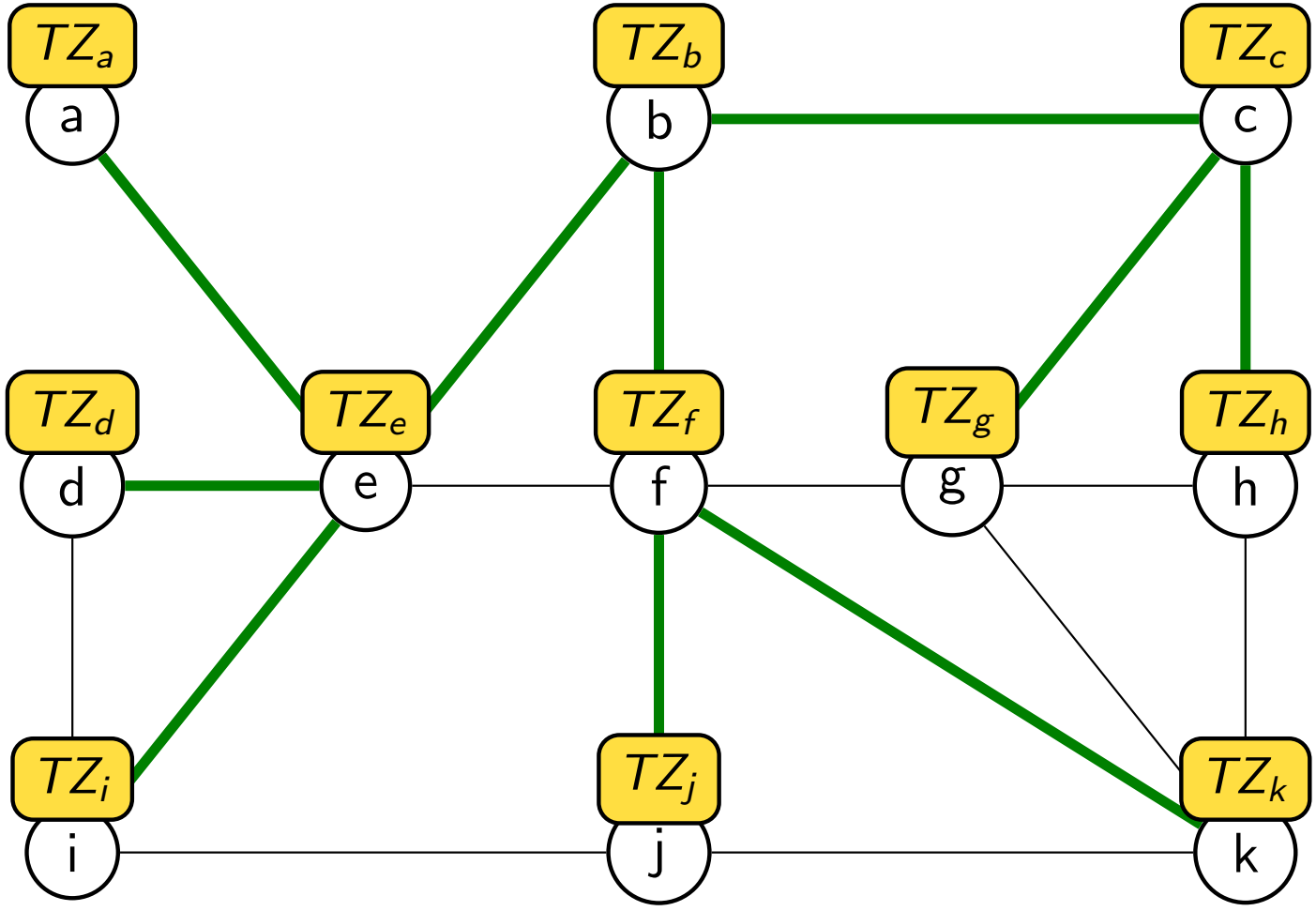
[Thorup M. and Zwick U., *Compact Routing Schemes*, SPAA 2001]



Locator-Based Forwarding

TZ-labels

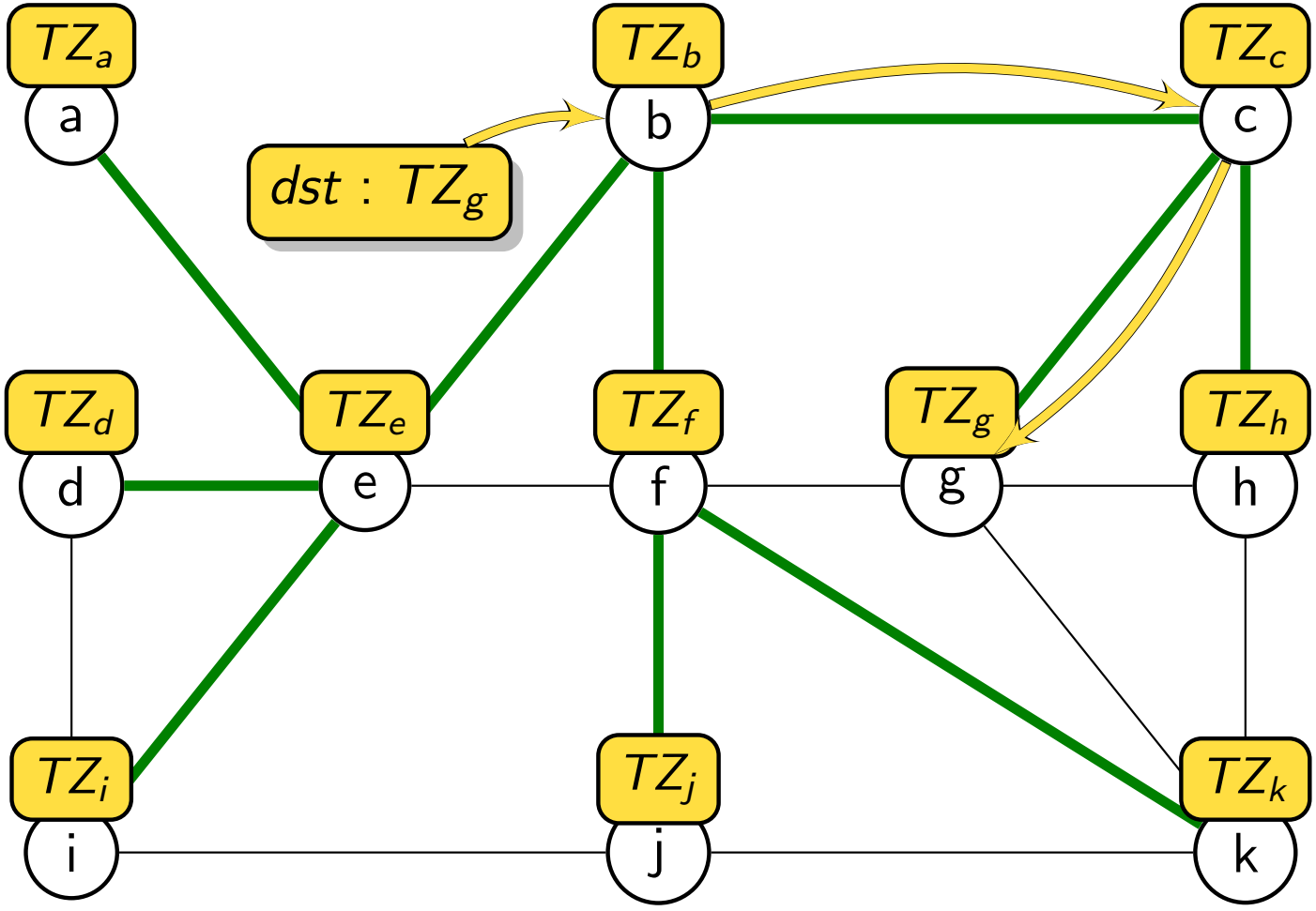
[Thorup M. and Zwick U., *Compact Routing Schemes*, SPAA 2001]



Locator-Based Forwarding

TZ-labels

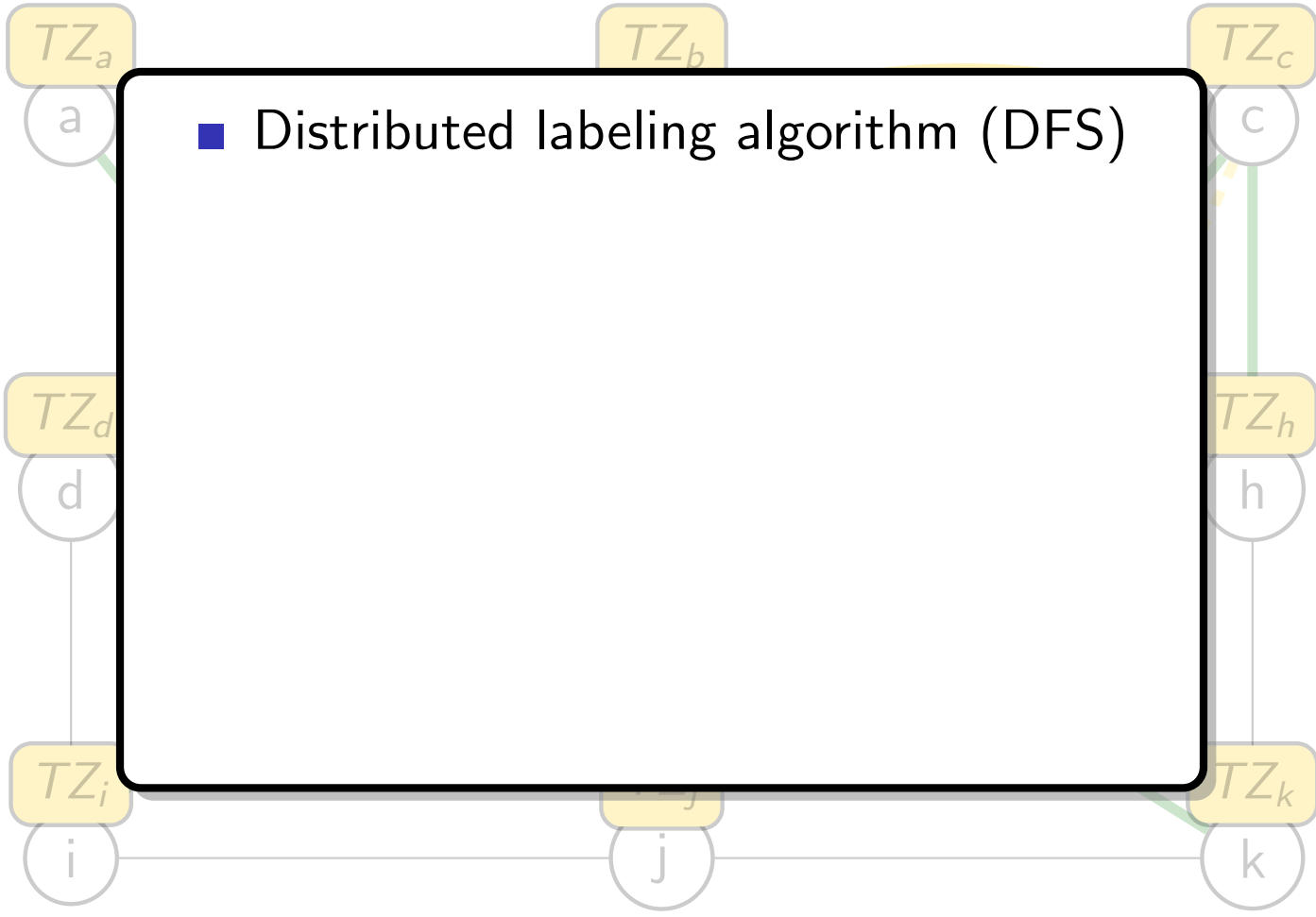
[Thorup M. and Zwick U., *Compact Routing Schemes*, SPAA 2001]



Locator-Based Forwarding

TZ-labels

[Thorup M. and Zwick U., *Compact Routing Schemes*, SPAA 2001]

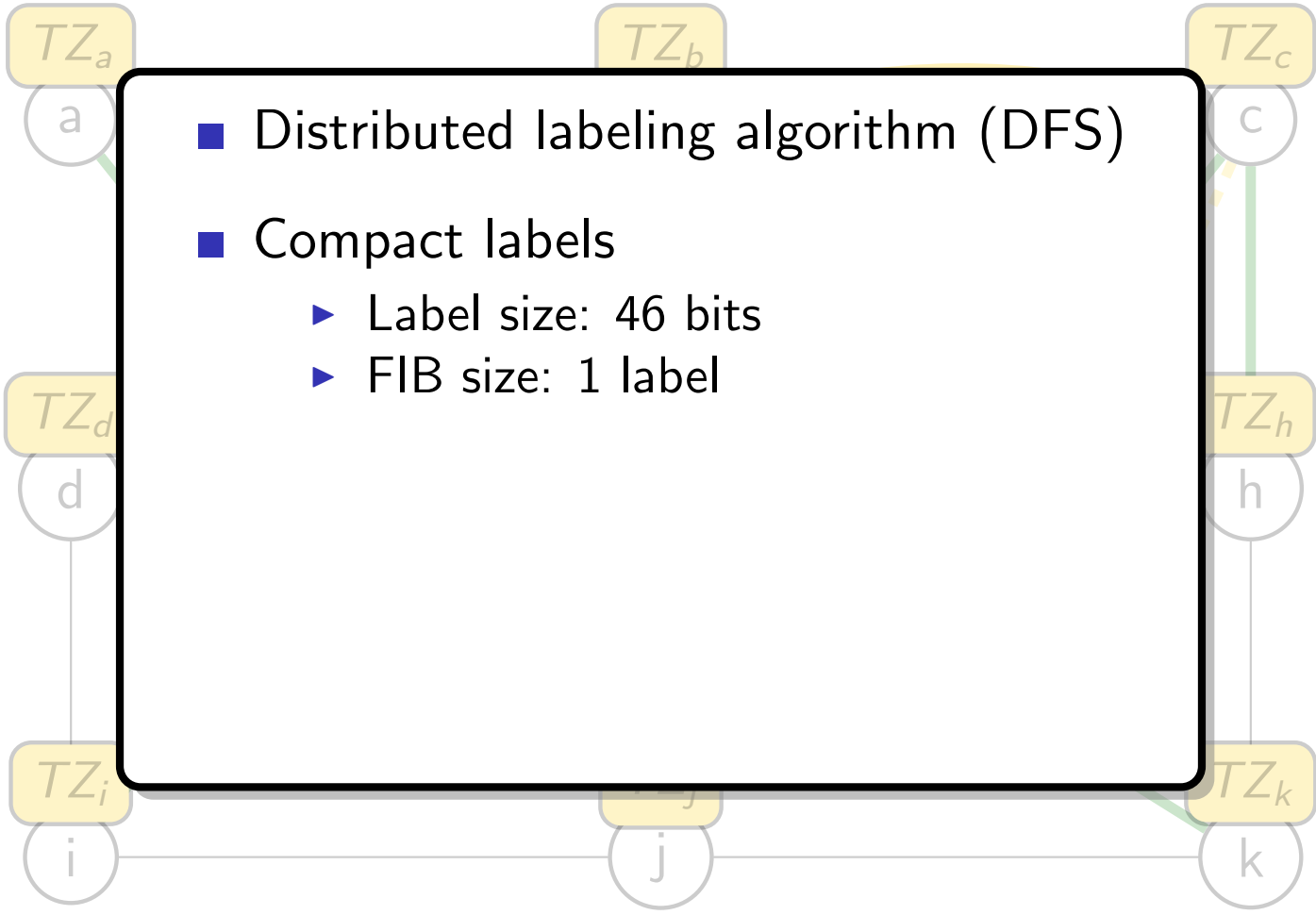


- Distributed labeling algorithm (DFS)

Locator-Based Forwarding

TZ-labels

[Thorup M. and Zwick U., *Compact Routing Schemes*, SPAA 2001]

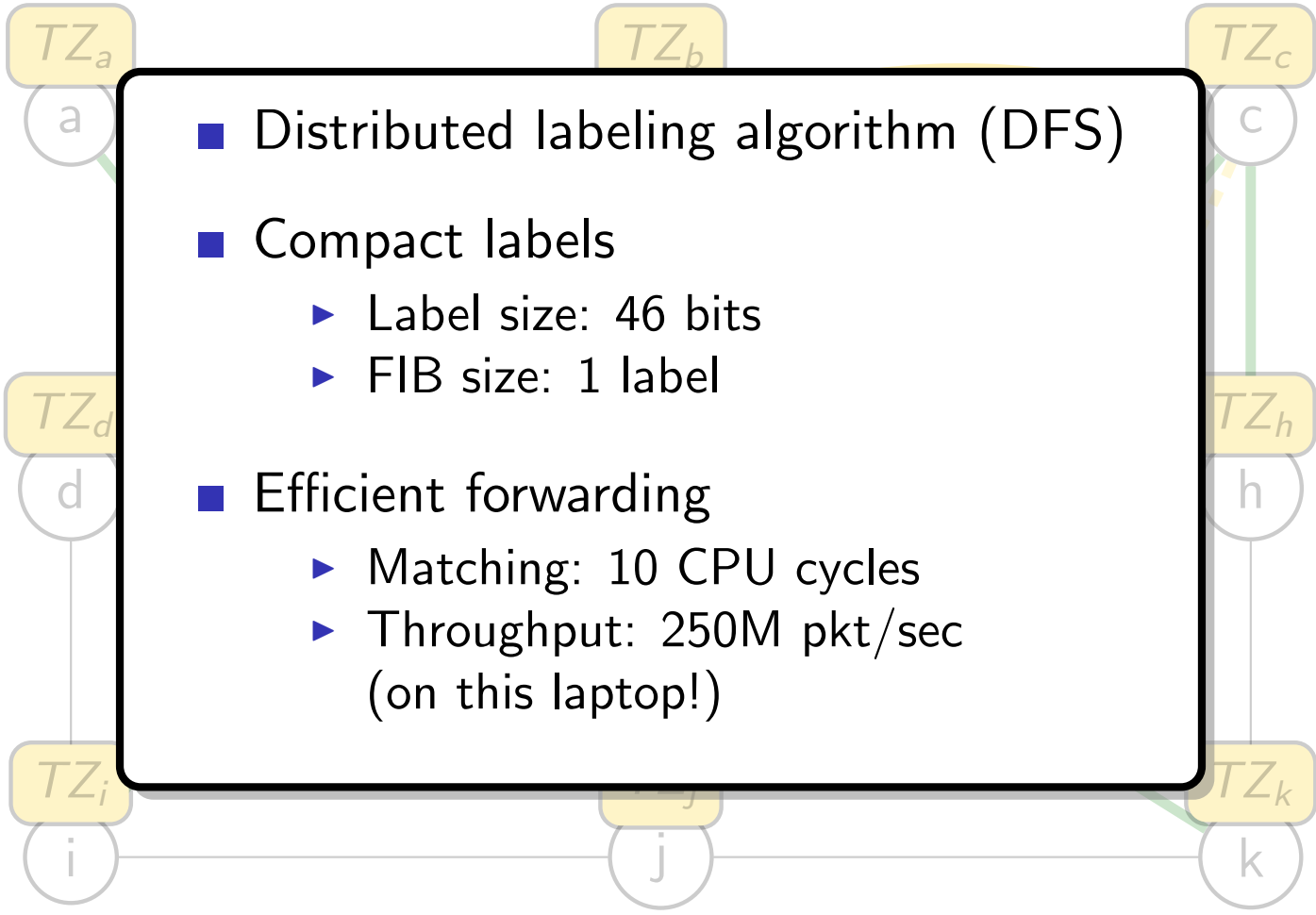


- Distributed labeling algorithm (DFS)
- Compact labels
  - Label size: 46 bits
  - FIB size: 1 label

Locator-Based Forwarding

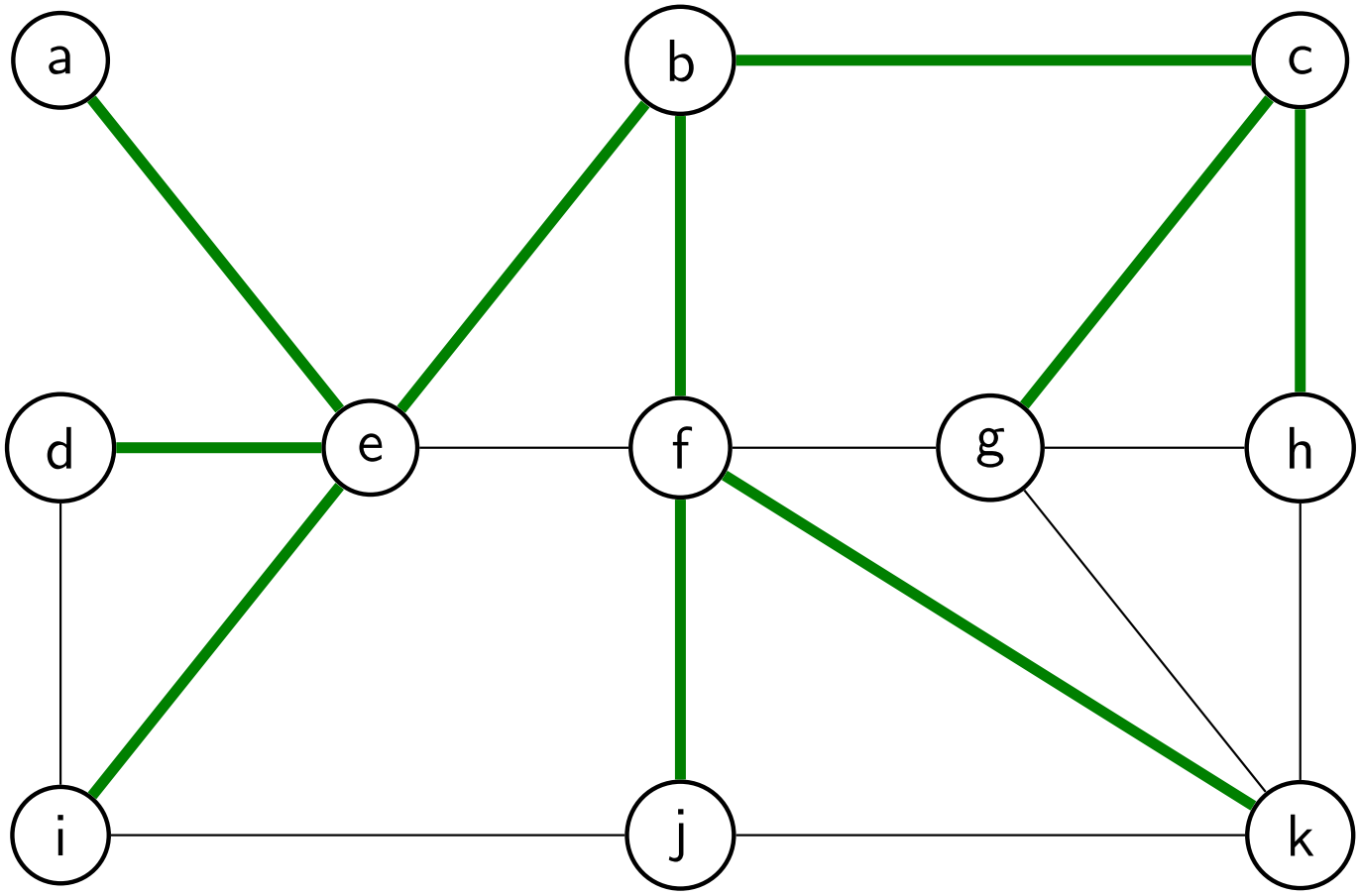
TZ-labels

[Thorup M. and Zwick U., *Compact Routing Schemes*, SPAA 2001]

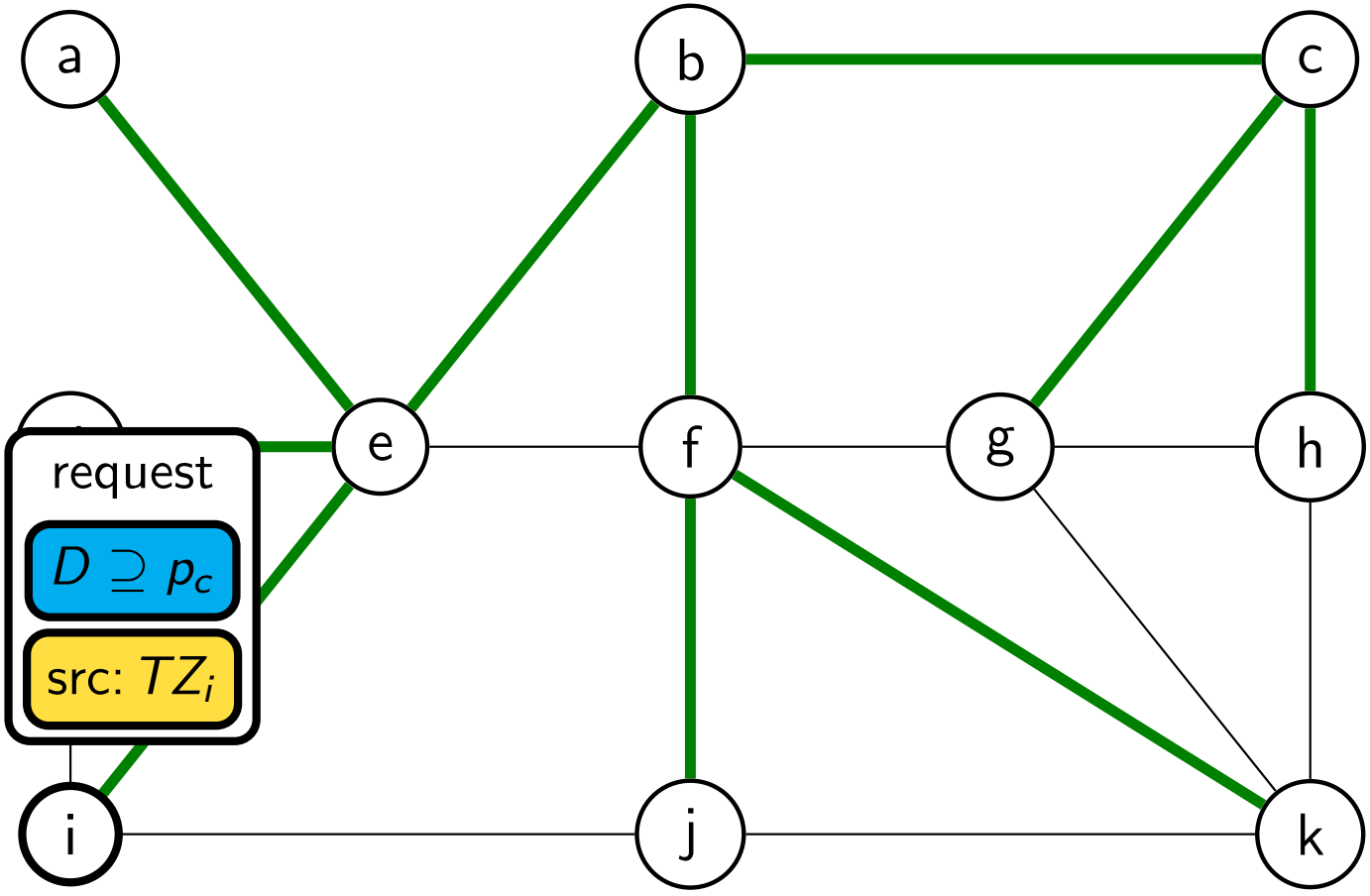


- Distributed labeling algorithm (DFS)
- Compact labels
  - Label size: 46 bits
  - FIB size: 1 label
- Efficient forwarding
  - Matching: 10 CPU cycles
  - Throughput: 250M pkt/sec (on this laptop!)

Communication Flow

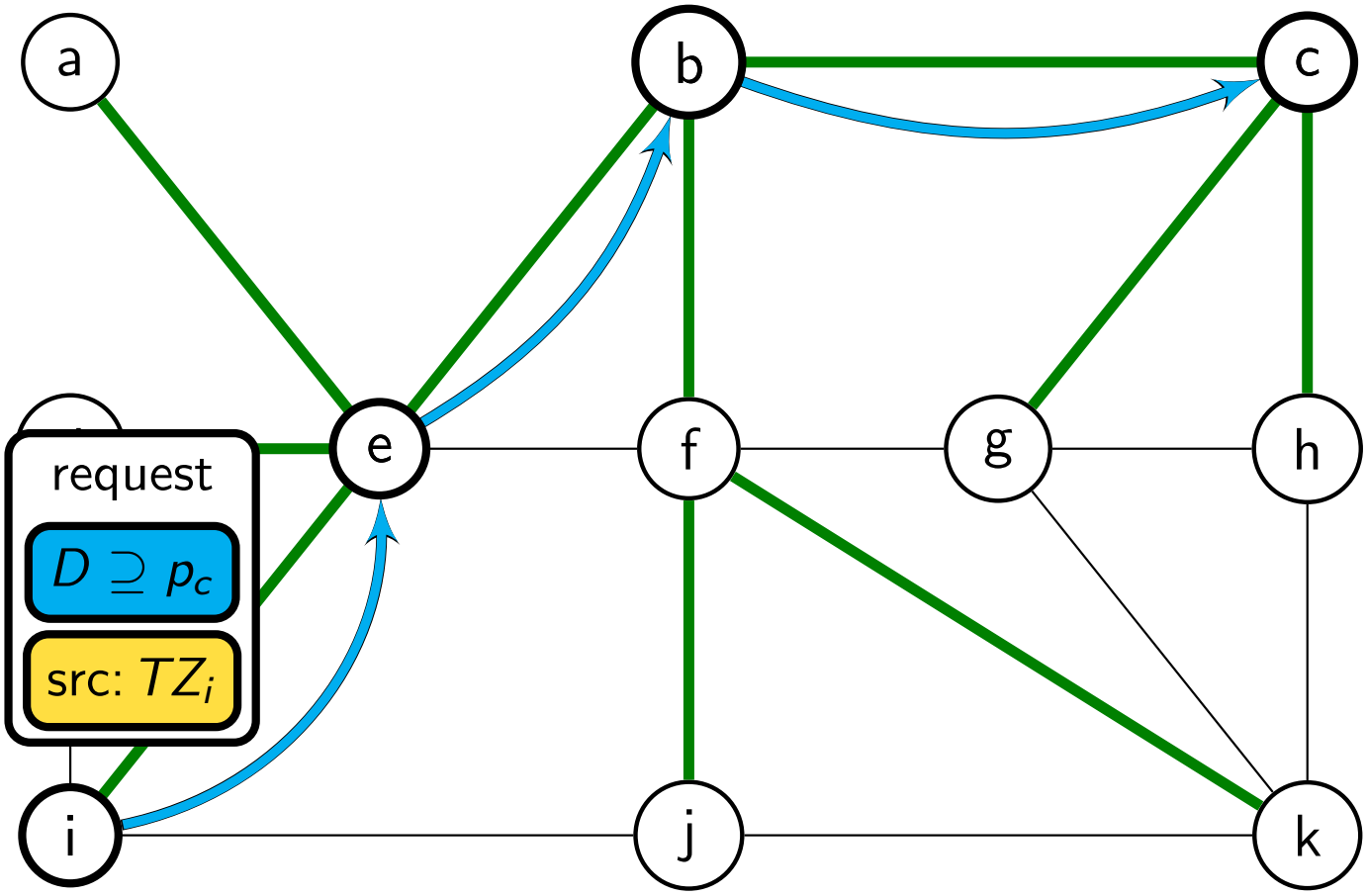


Communication Flow

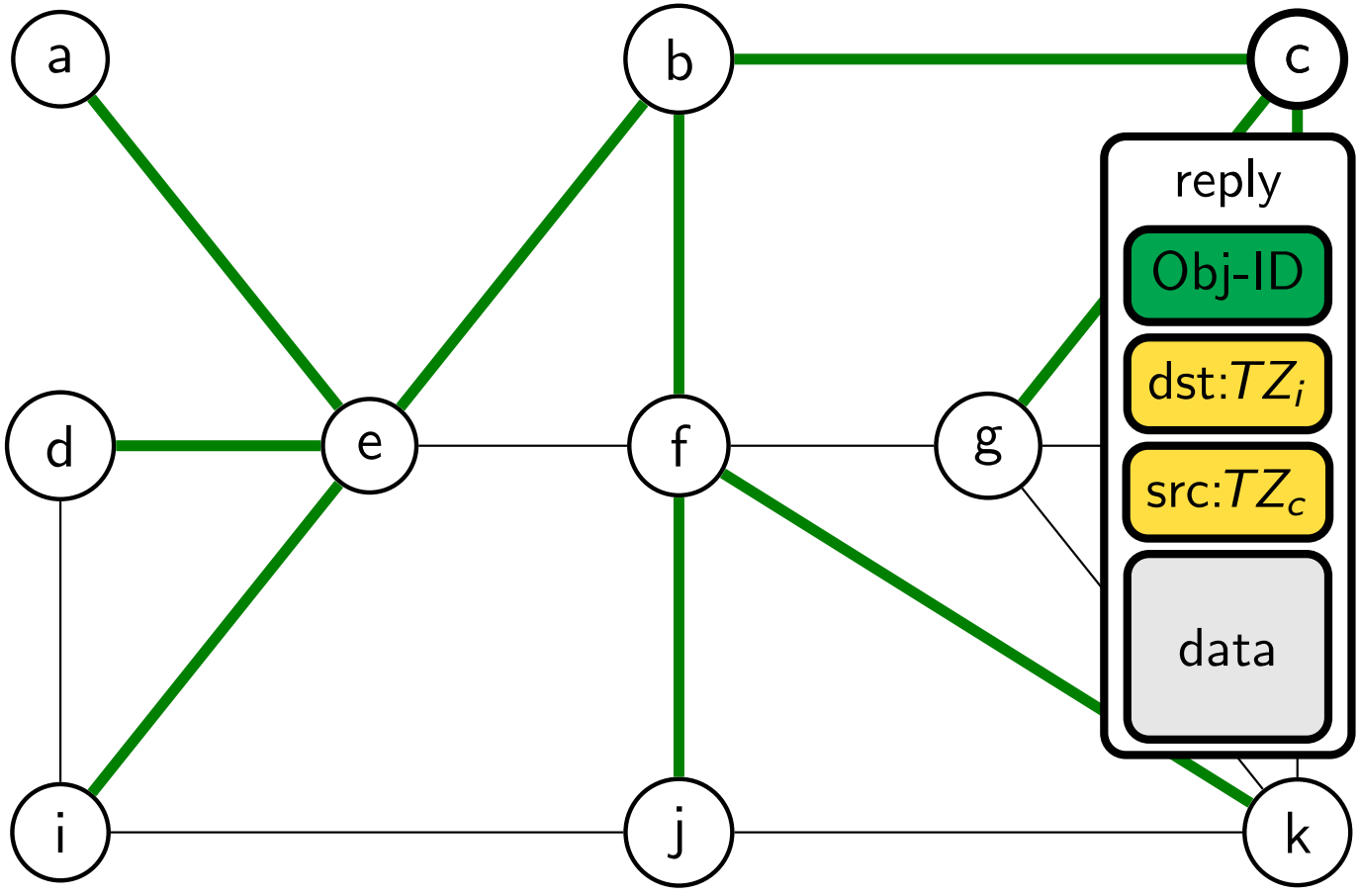


Communication Flow

Content-based matching, using descriptor D

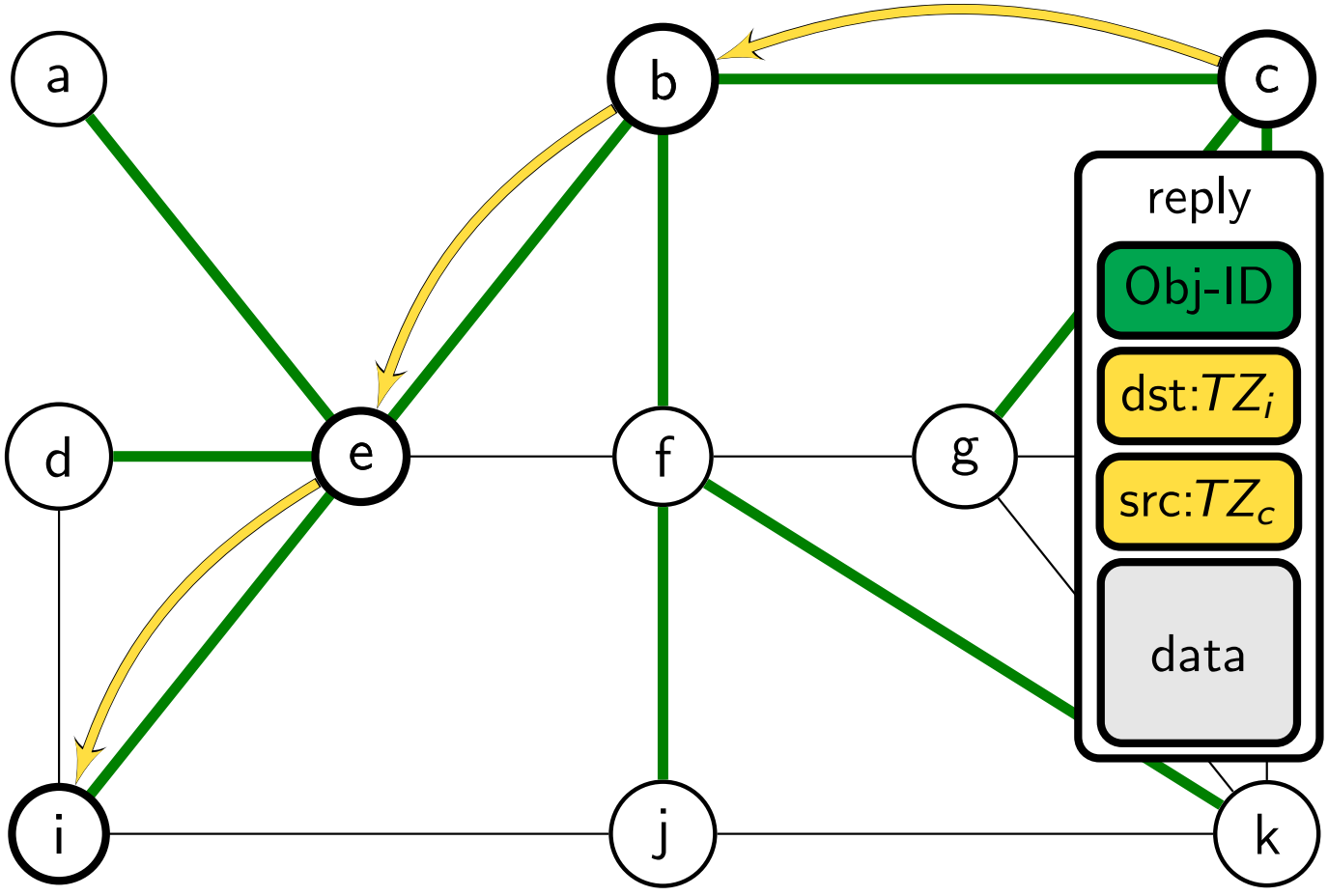


Communication Flow



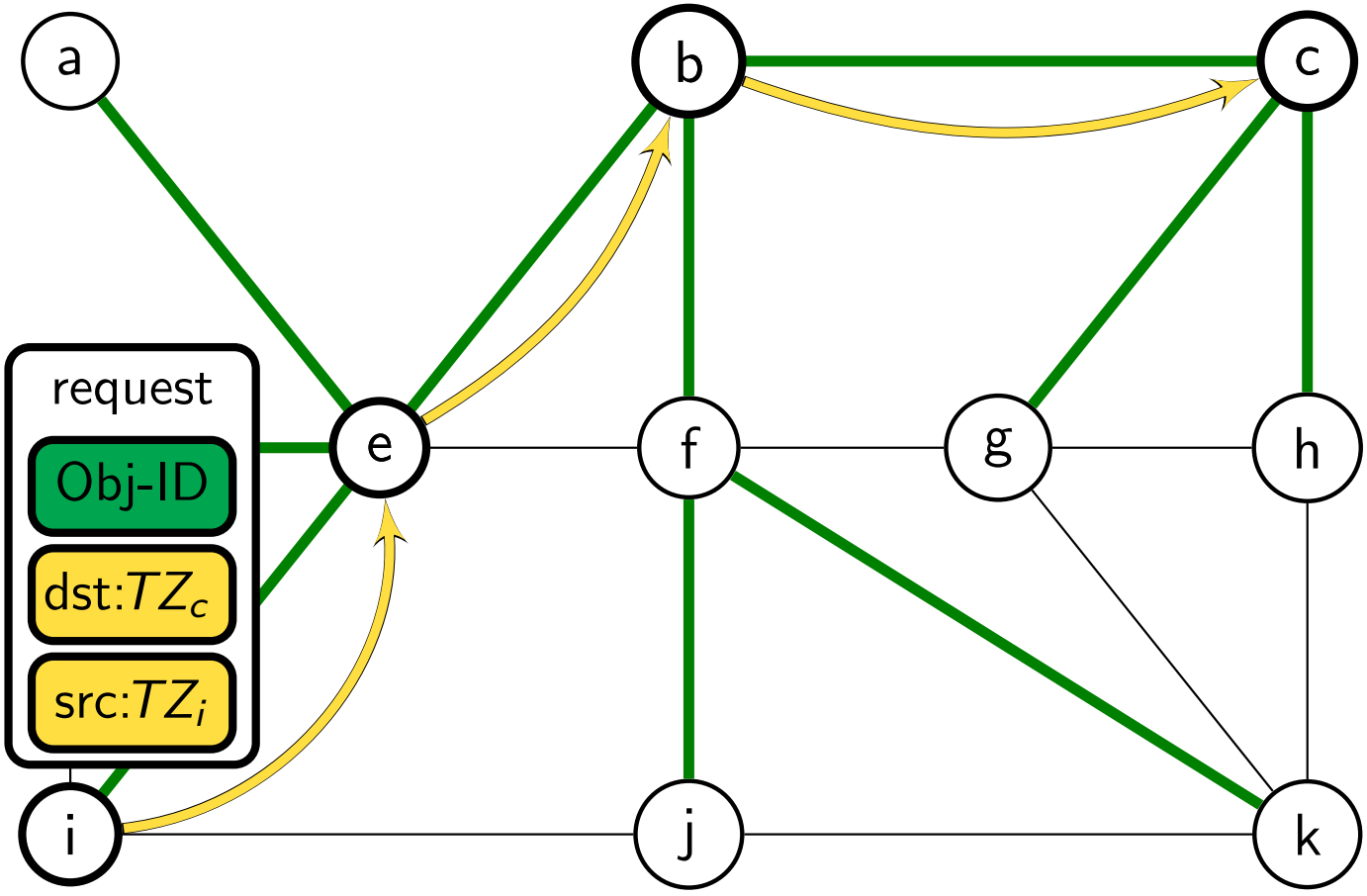
Communication Flow

Locator-based forwarding, using TZ-label TZ_i



Communication Flow

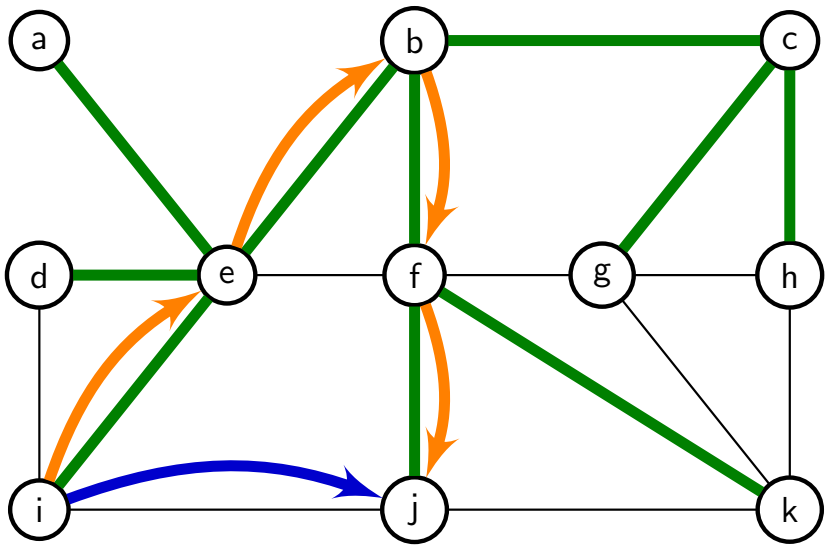
Locator-based forwarding, using TZ-label TZ_c
Caching, using Obj-ID



Issues with a Single Tree

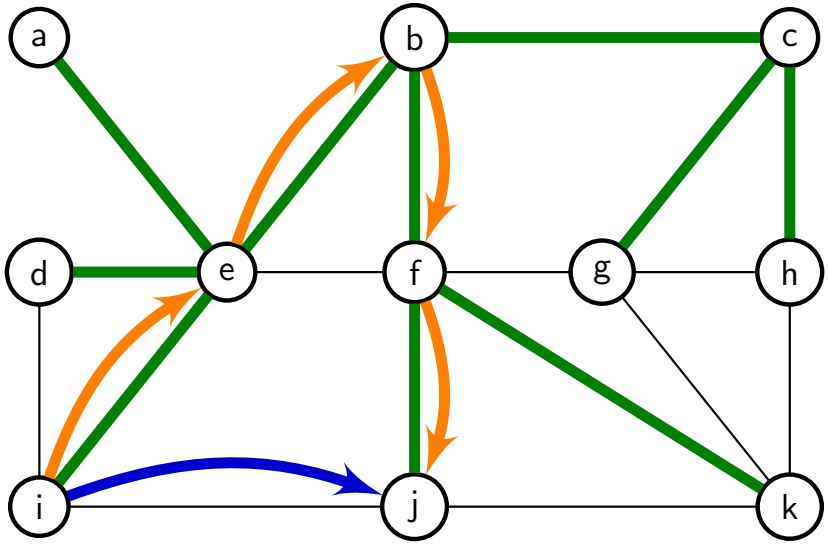
Issues with a Single Tree

Higher latency

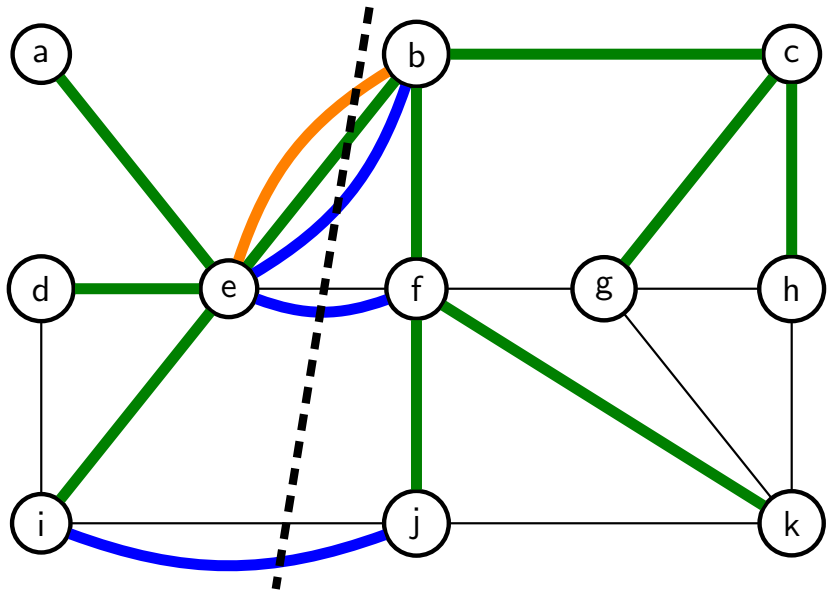


Issues with a Single Tree

Higher latency

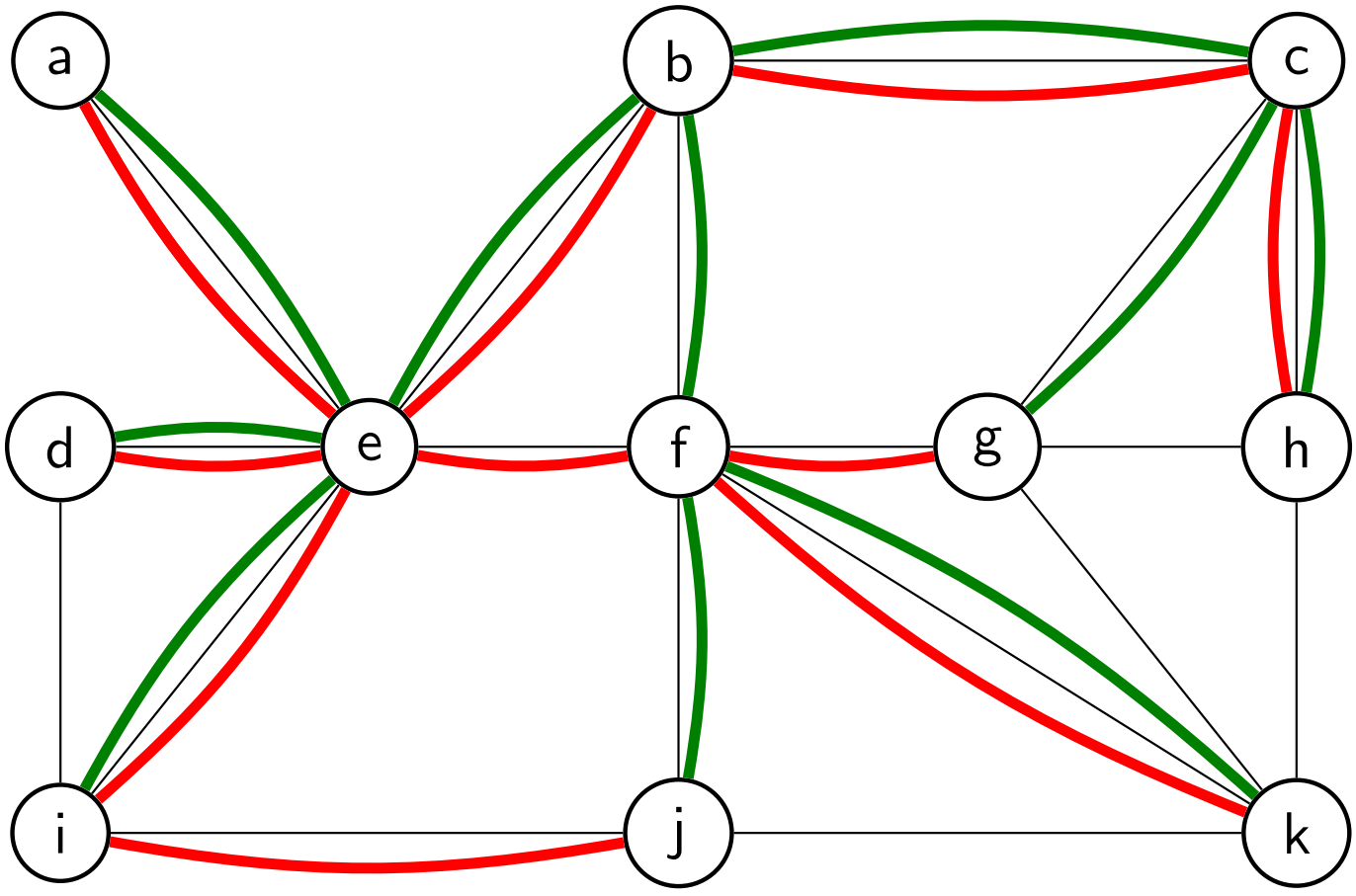


Lower throughput



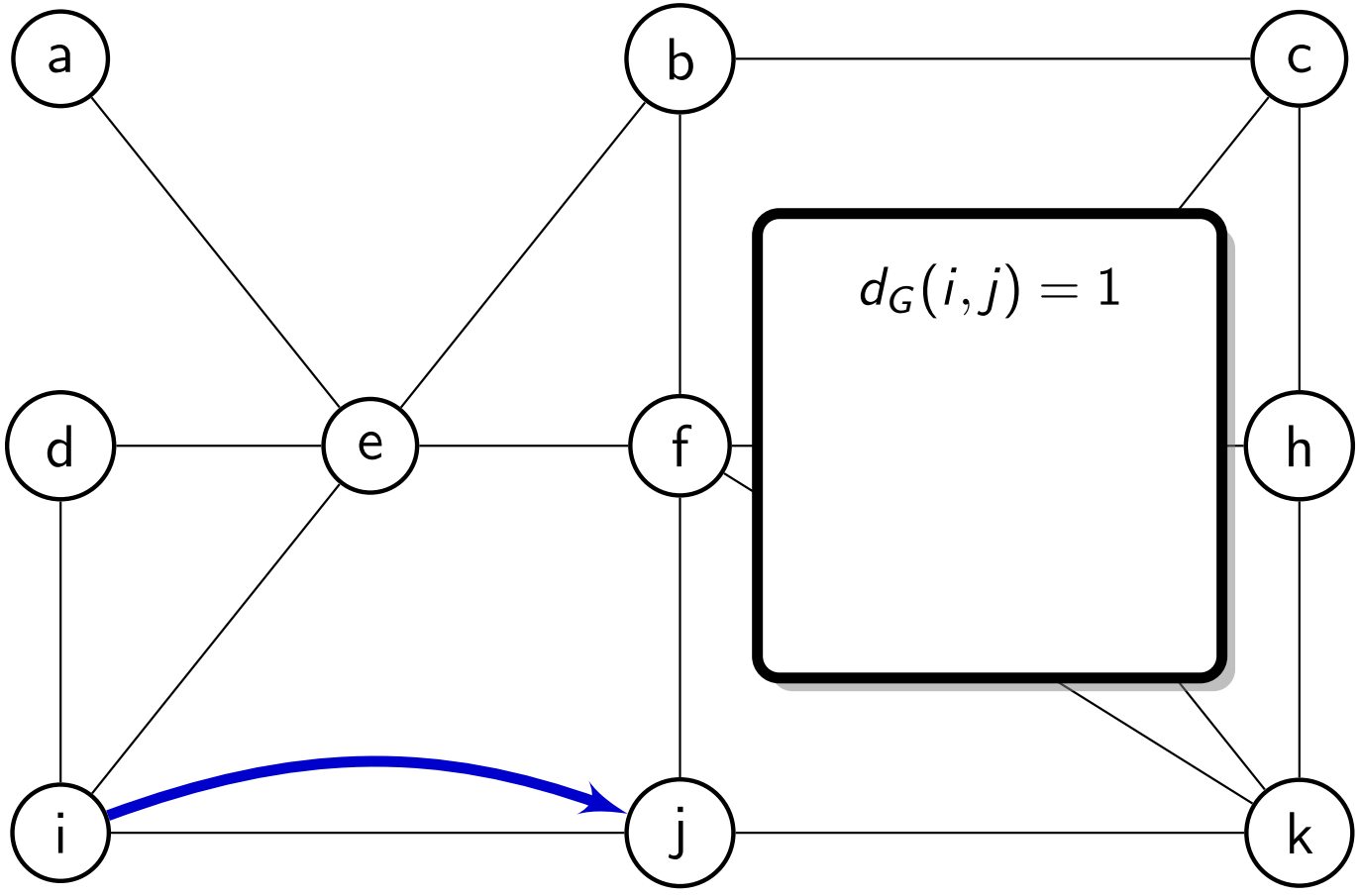
Multiple Trees

Low latency and **high throughput** (in expectation)



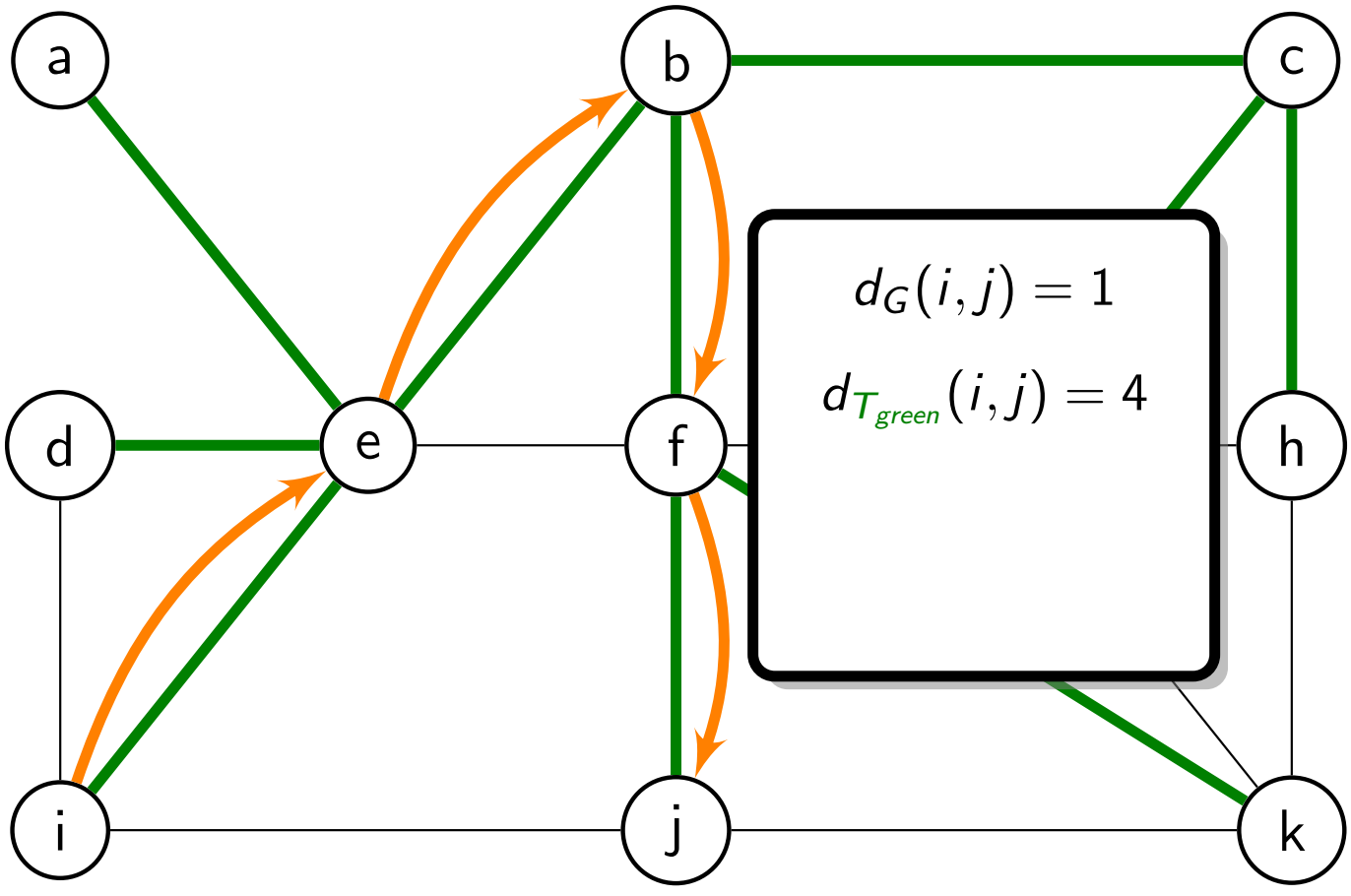
Multiple Trees

Low latency and **high throughput** (in expectation)



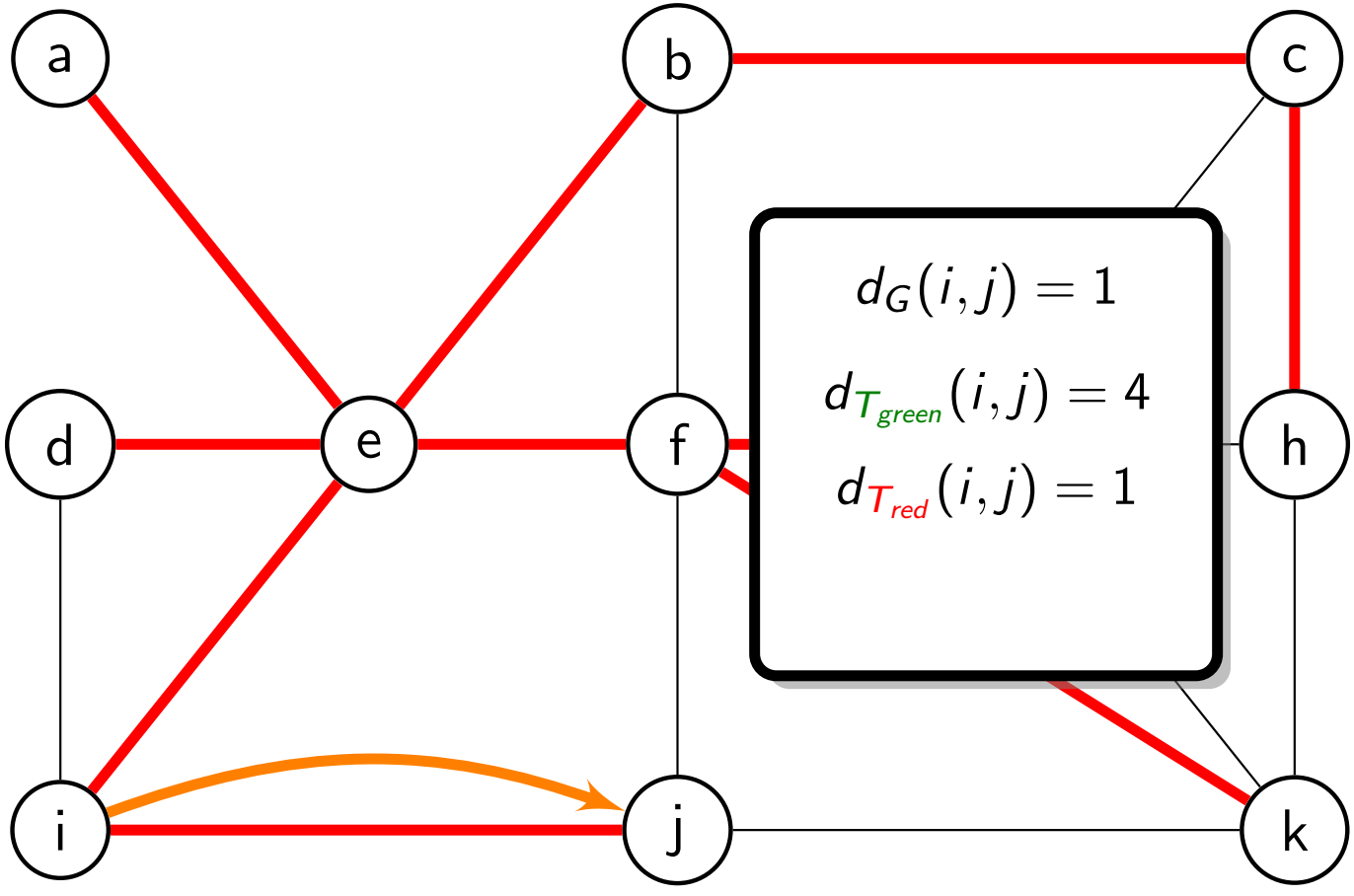
Multiple Trees

Low latency and **high throughput** (in expectation)



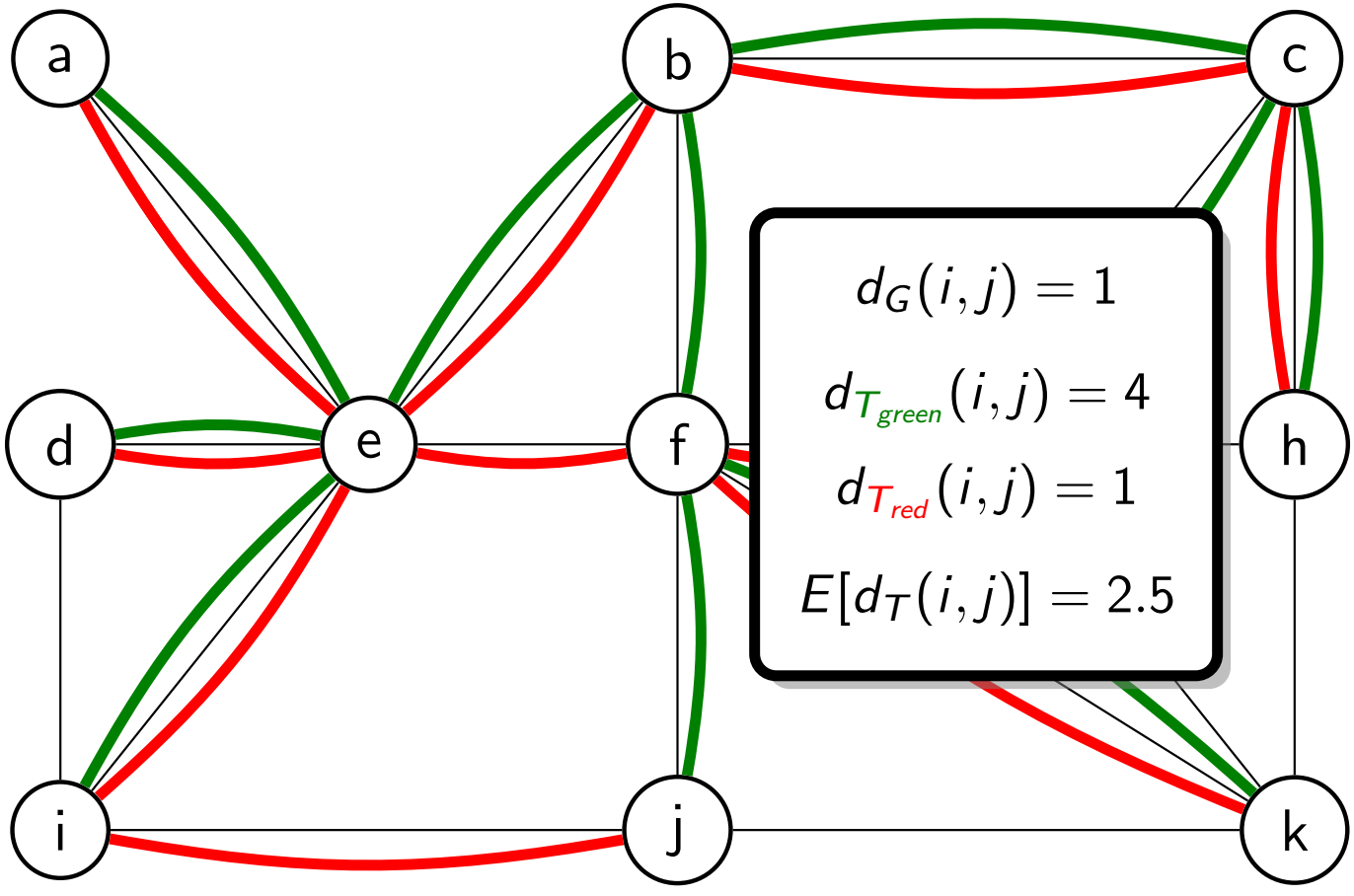
Multiple Trees

Low latency and **high throughput** (in expectation)



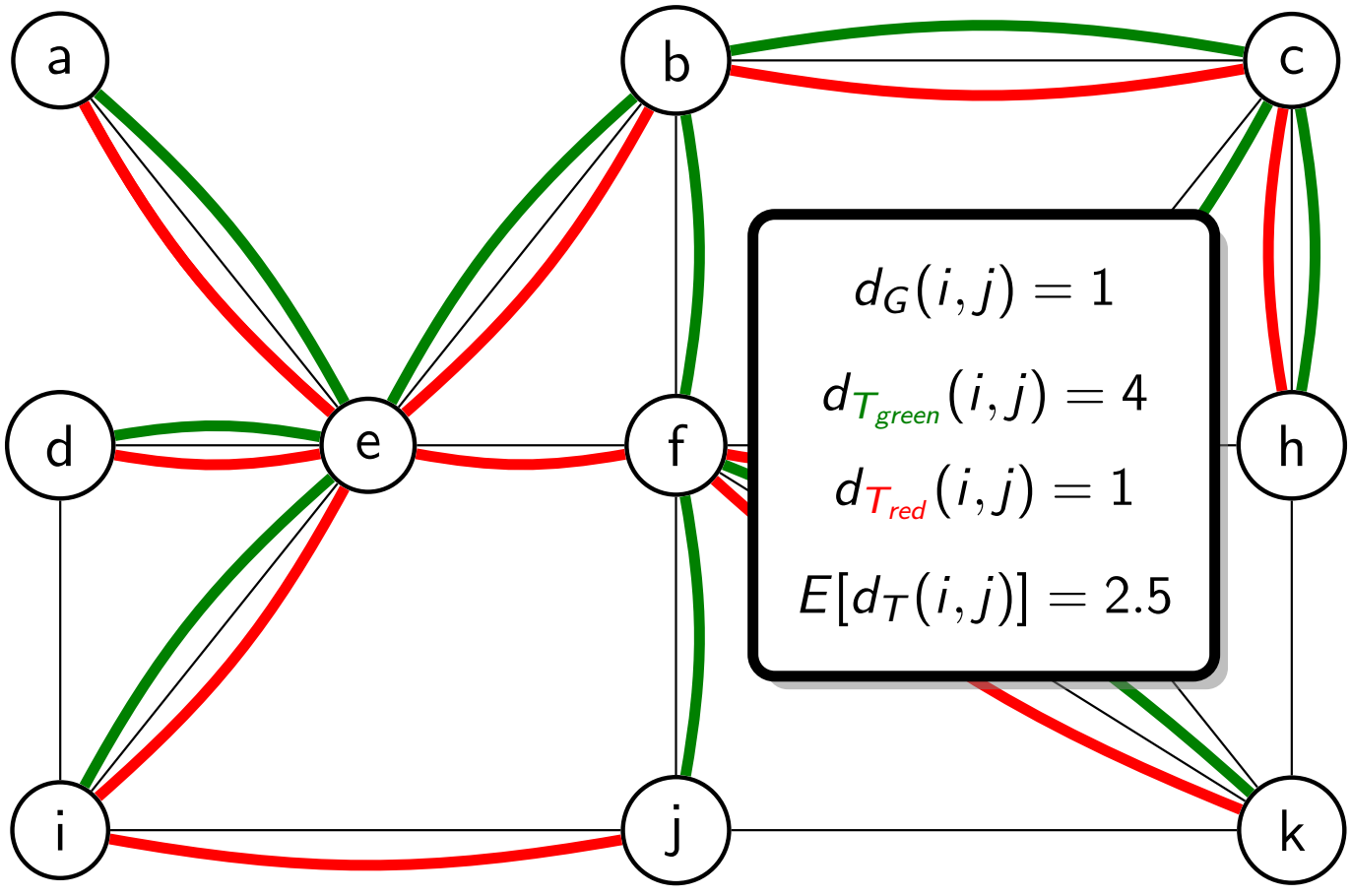
Multiple Trees

Low latency and **high throughput** (in expectation)



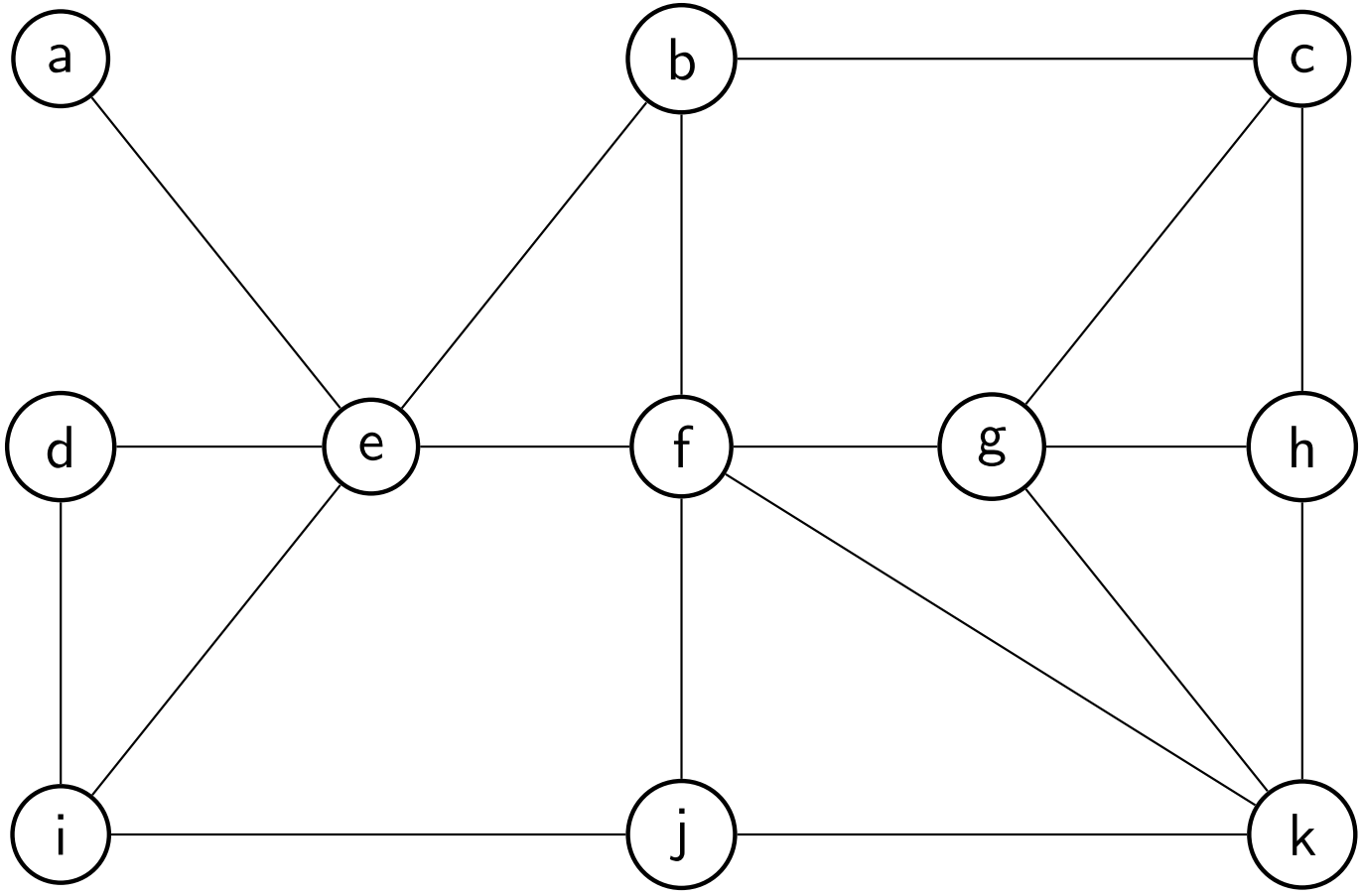
Multiple Trees

Low latency and **high throughput** (in expectation)

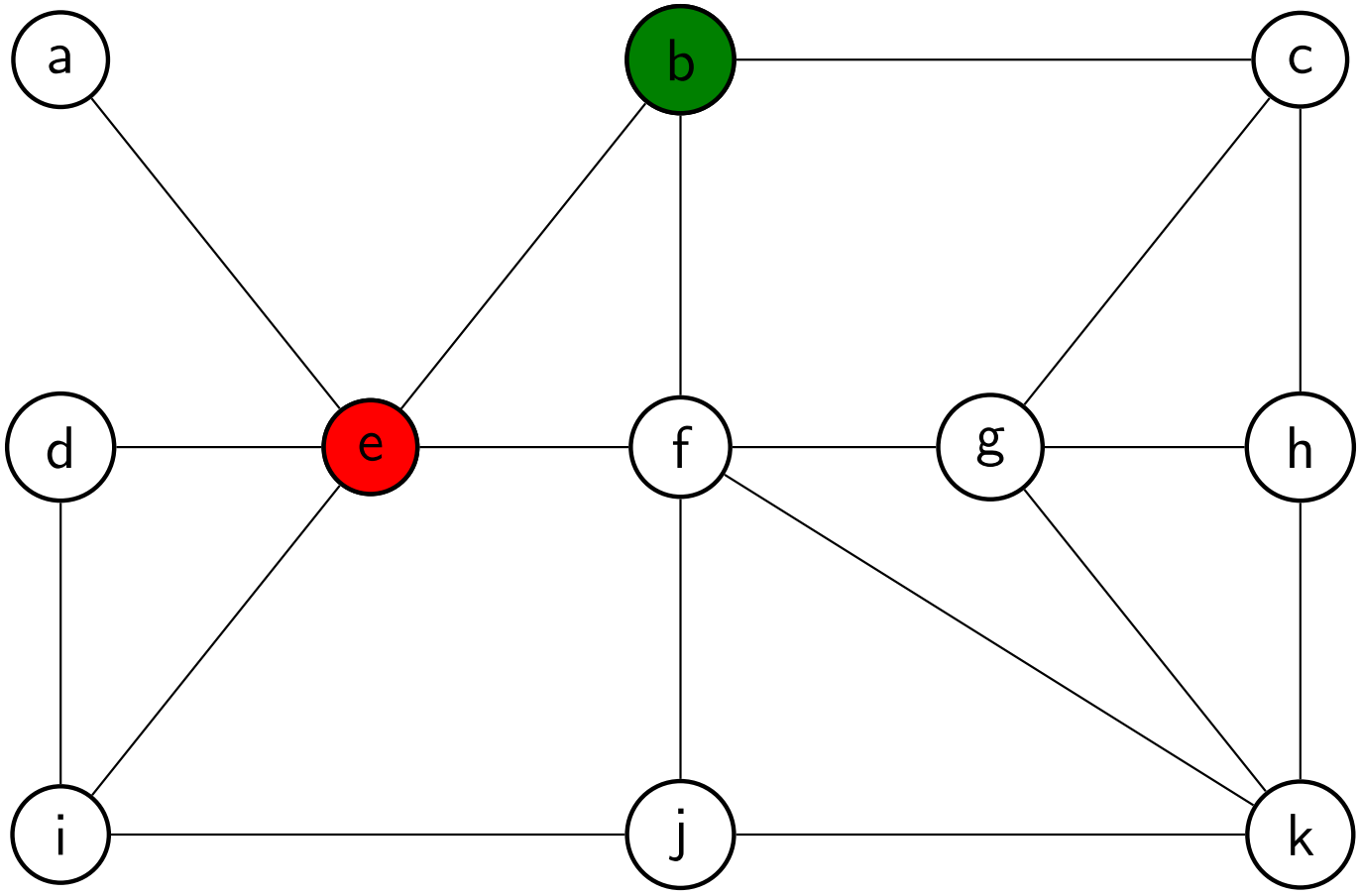


How do we build the trees?

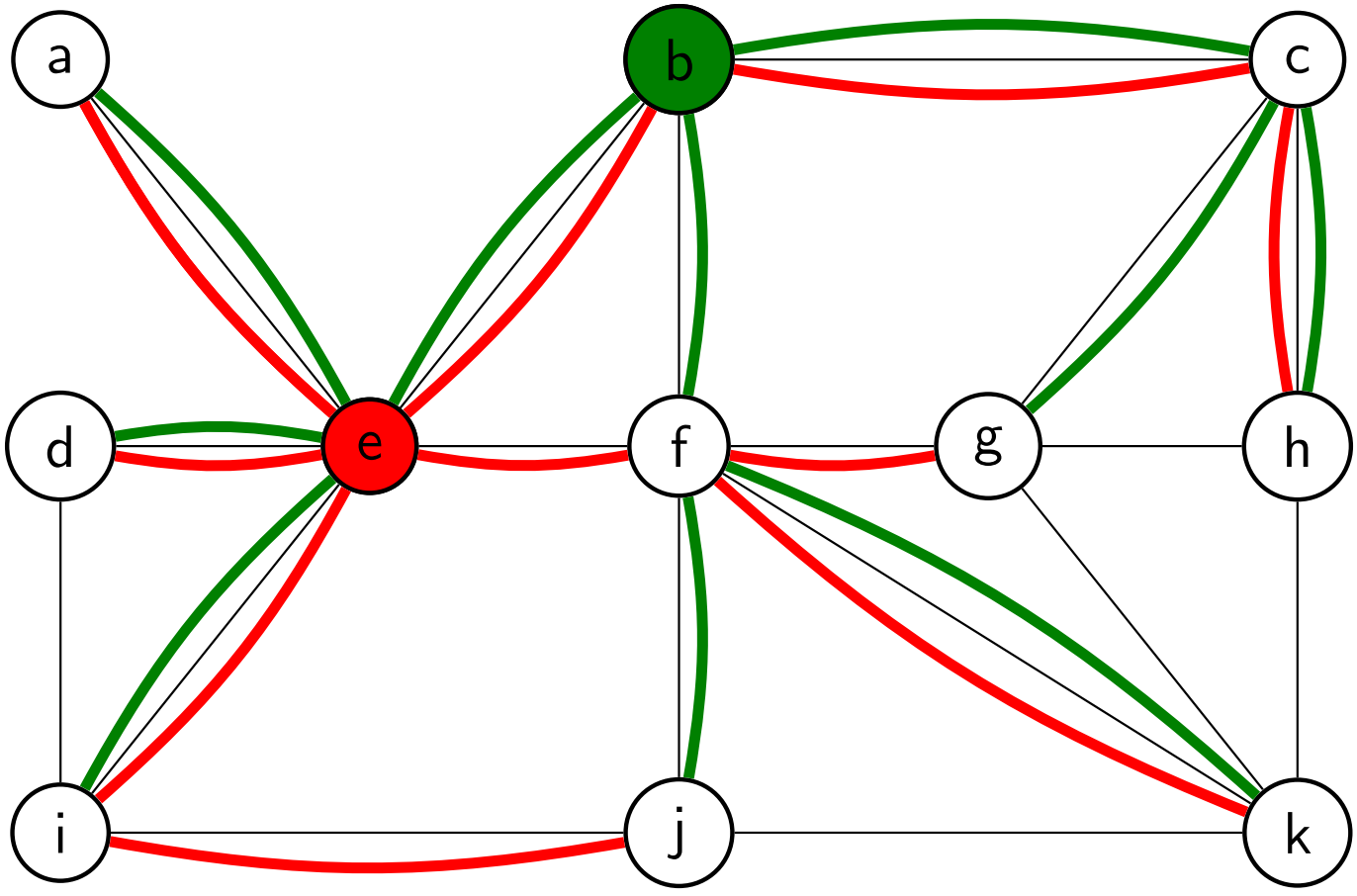
Building Trees: Latency Only



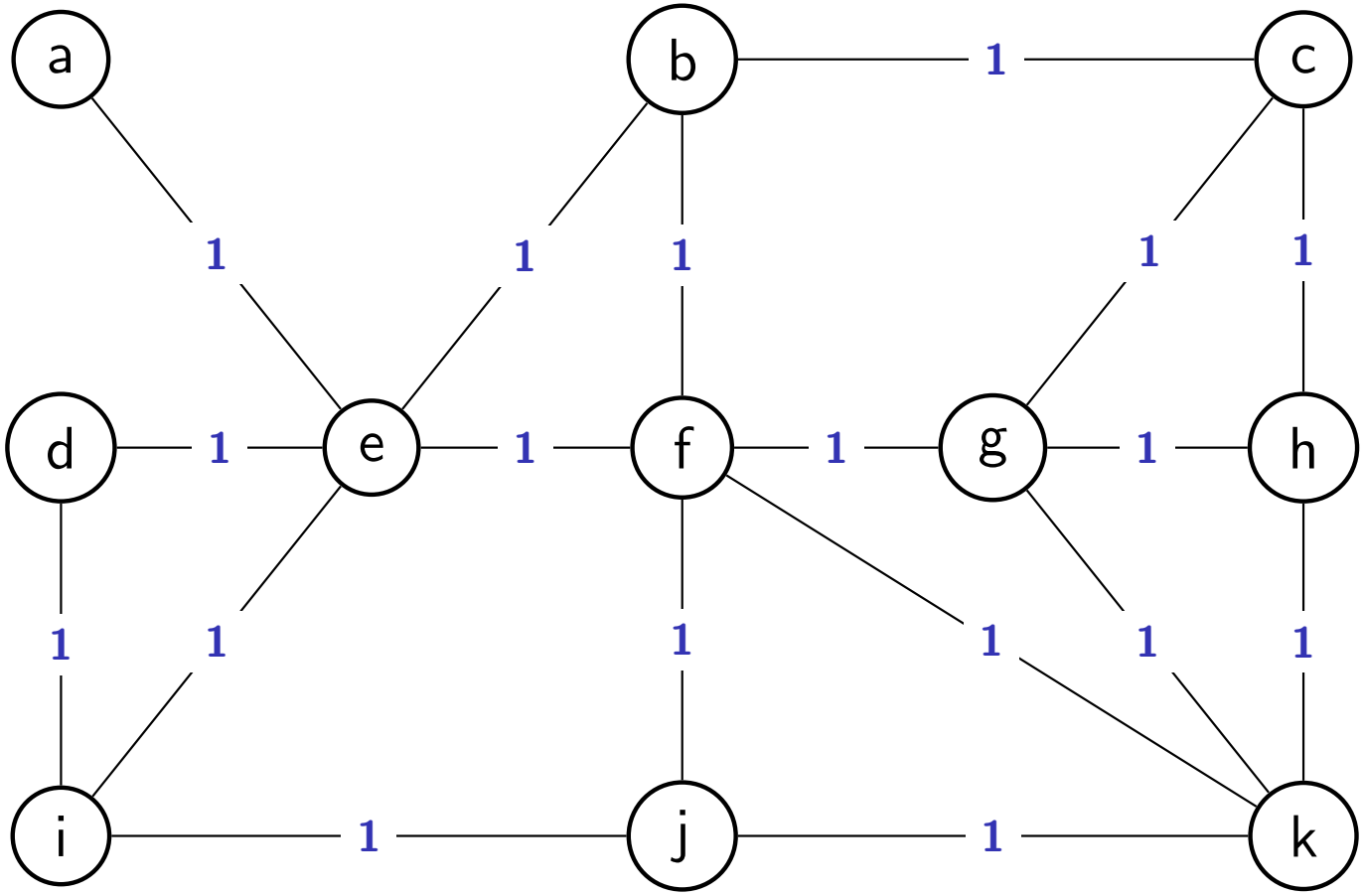
Building Trees: Latency Only



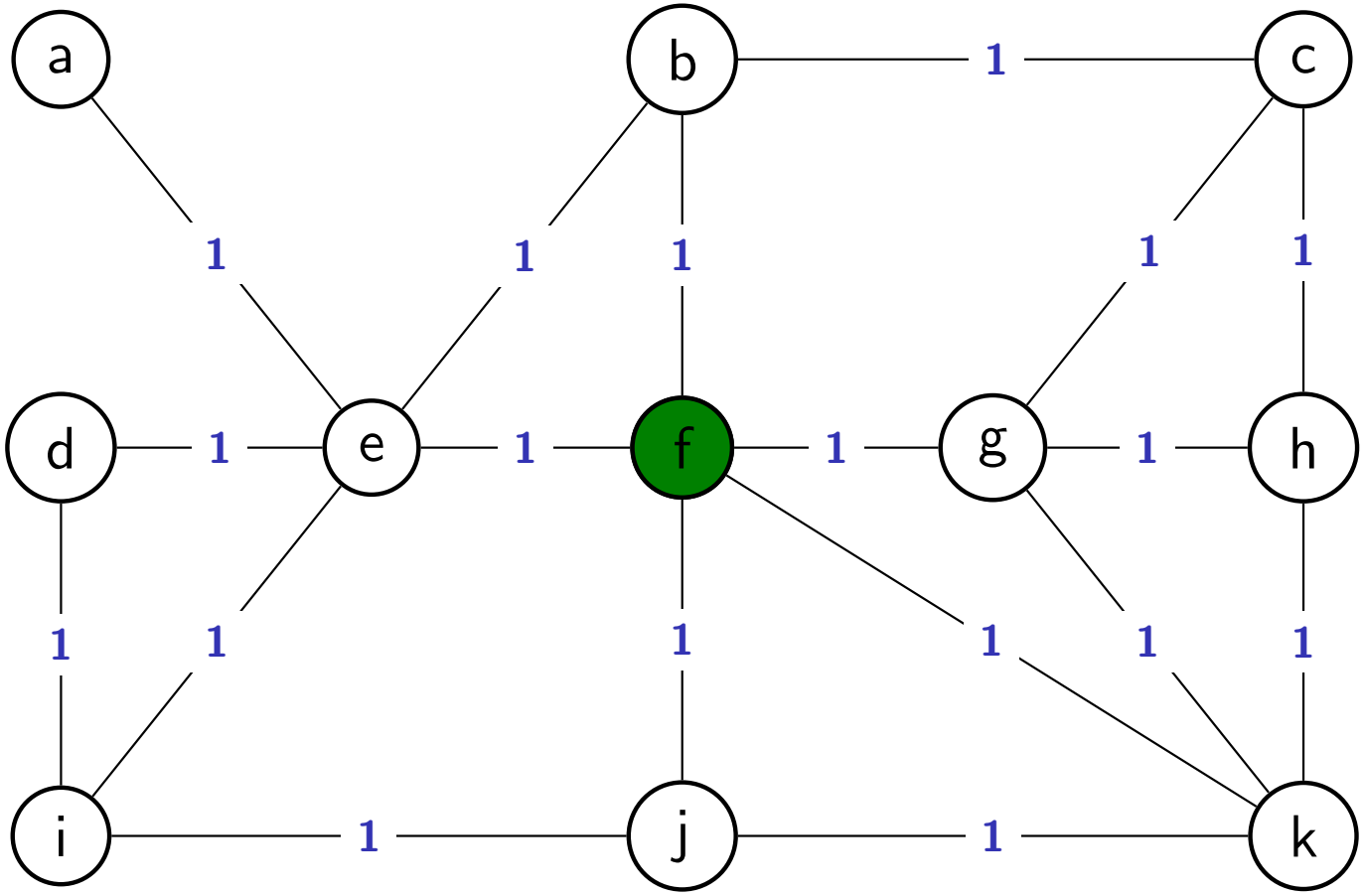
Building Trees: Latency Only



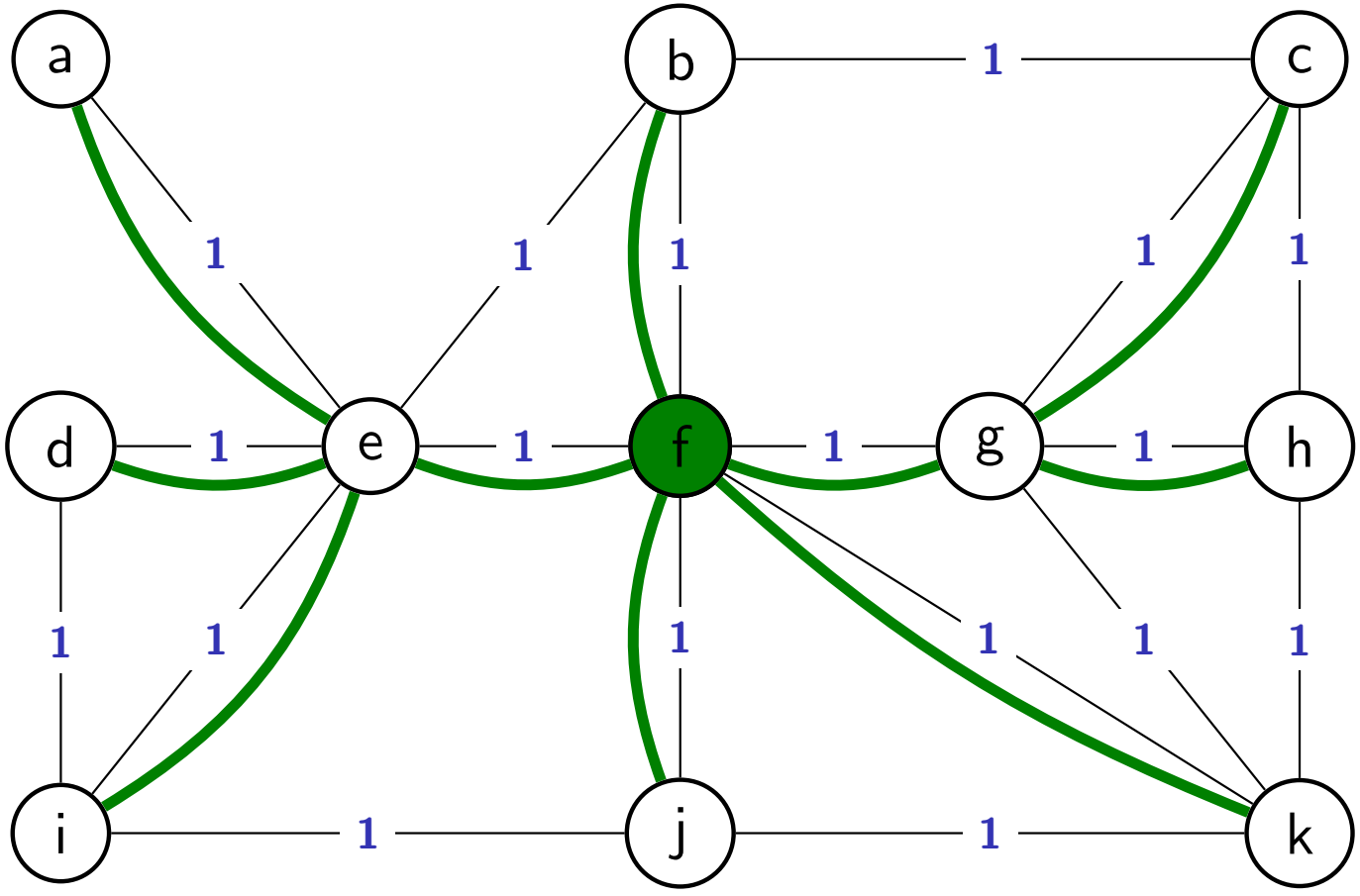
Building Trees: Latency and Congestion



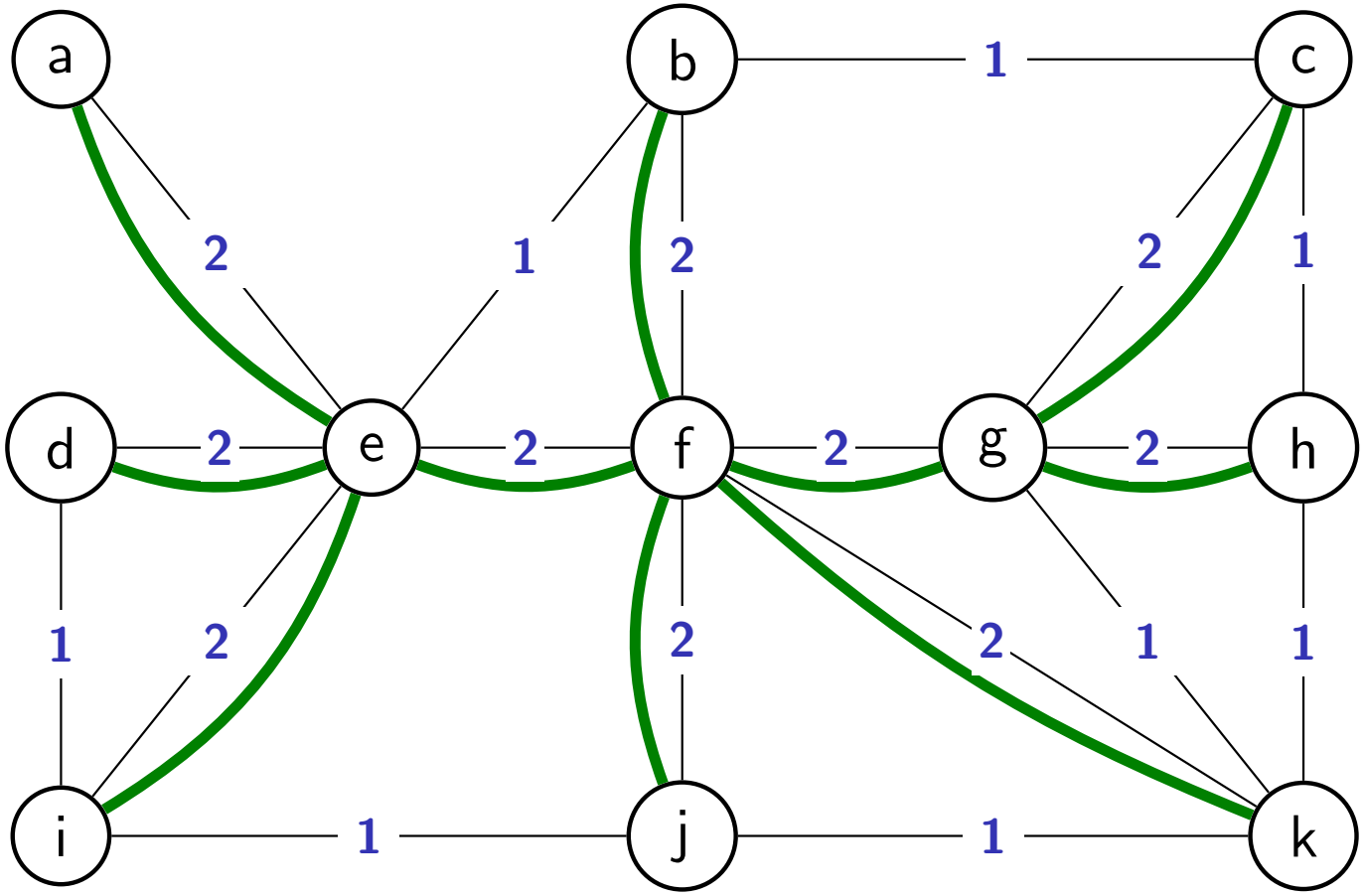
Building Trees: Latency and Congestion



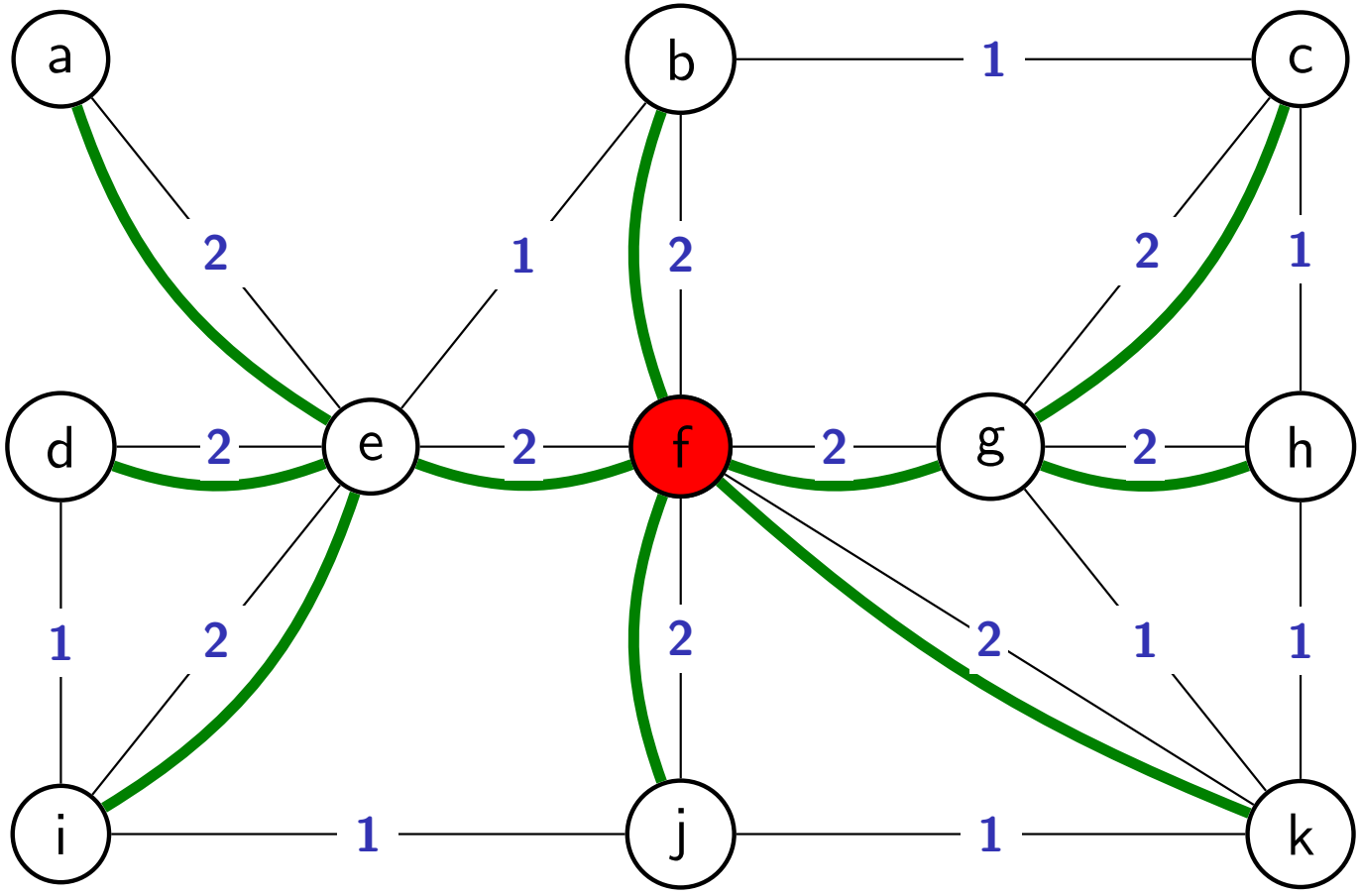
Building Trees: Latency and Congestion



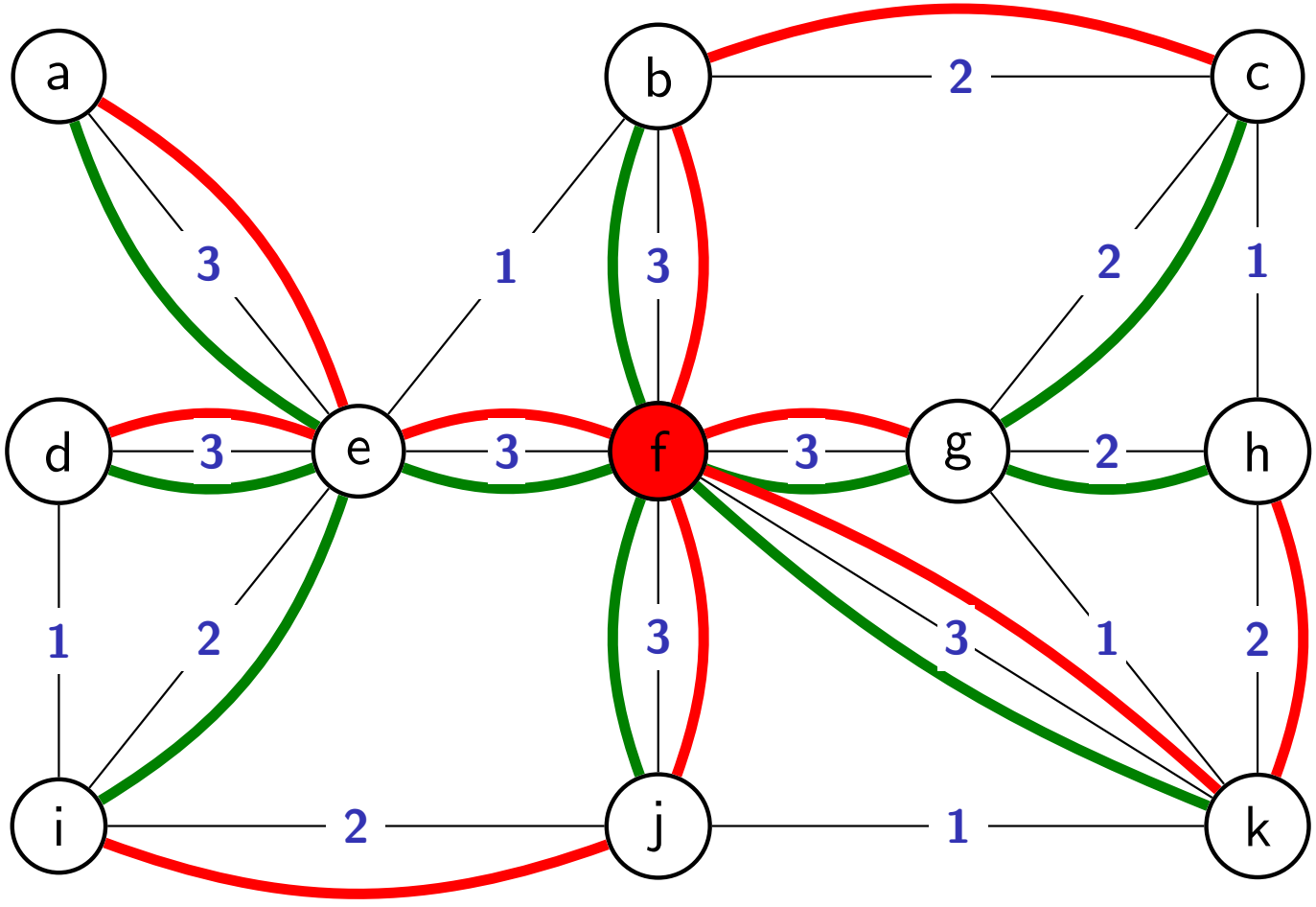
Building Trees: Latency and Congestion



Building Trees: Latency and Congestion

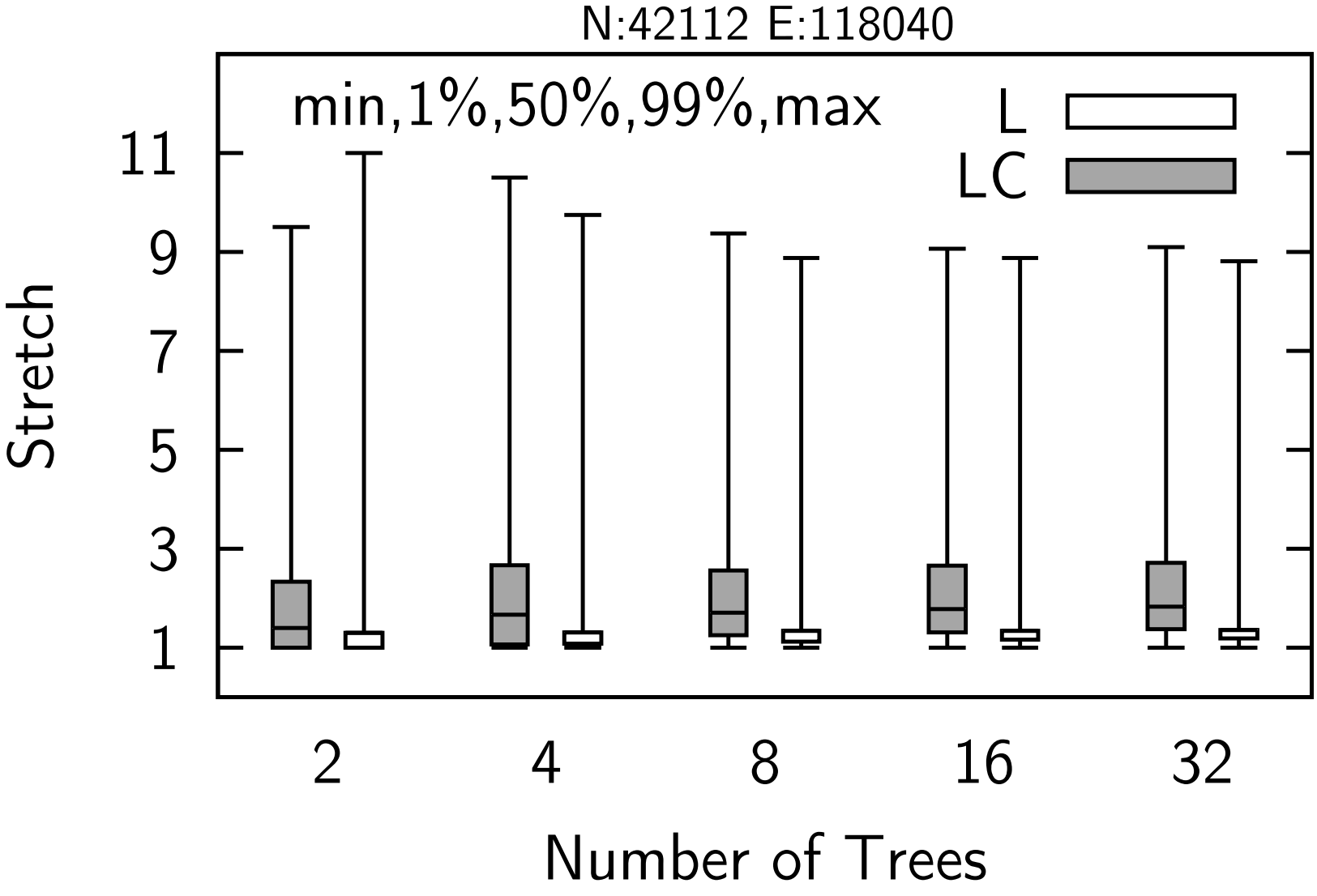


Building Trees: Latency and Congestion

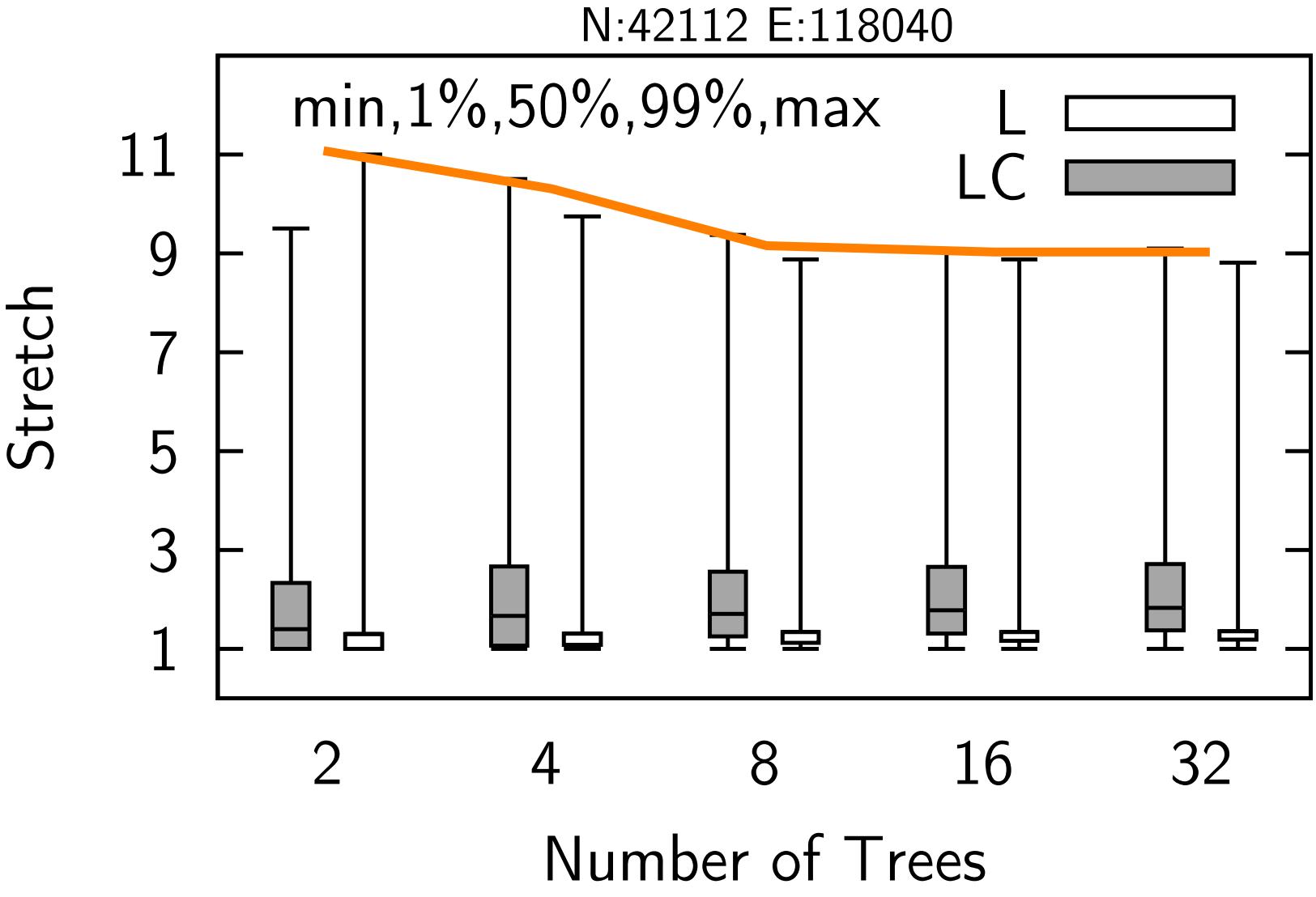


Routing Over Multiple Trees: Evaluation

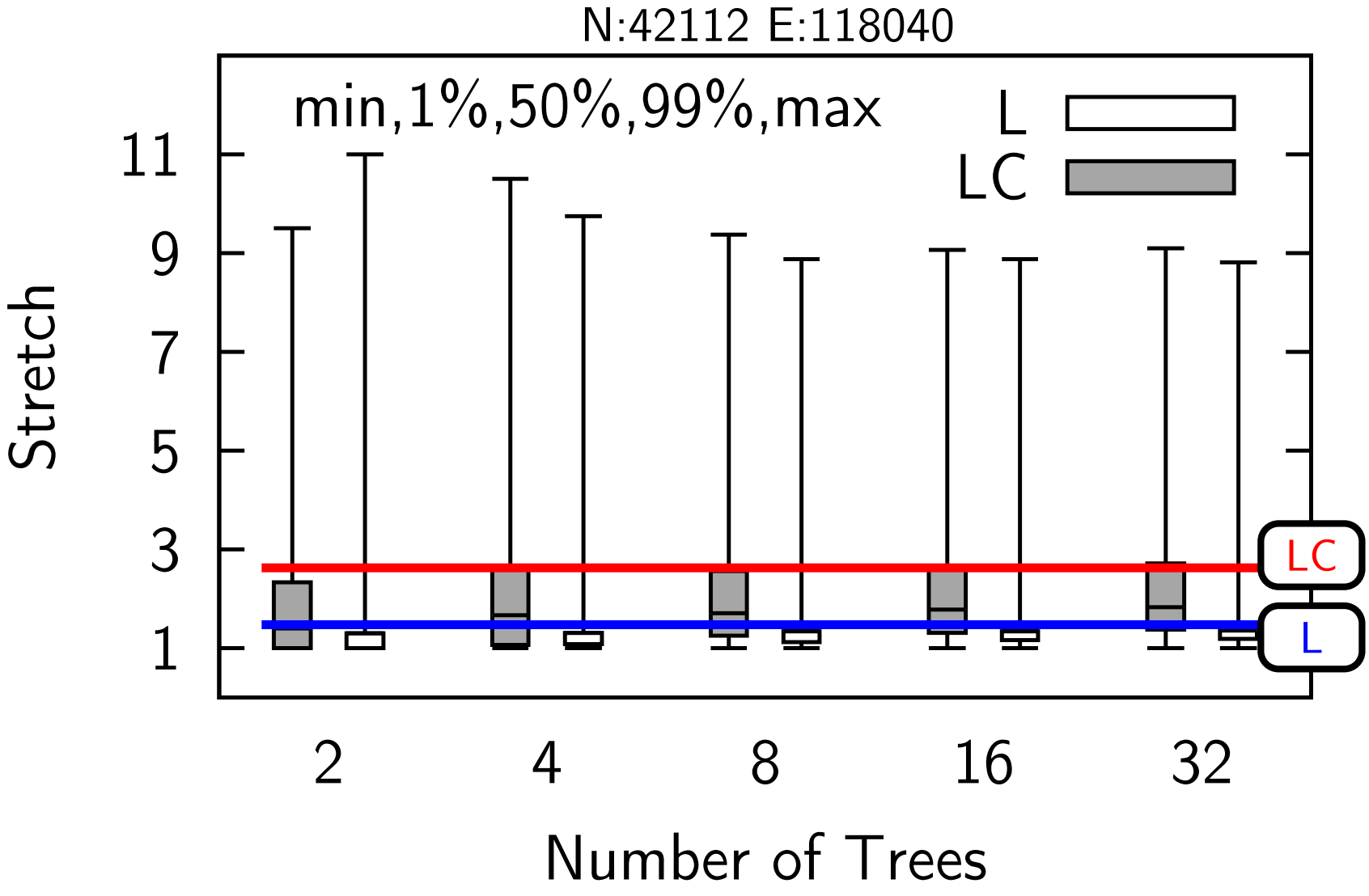
Expected Stretch, AS-level Topology



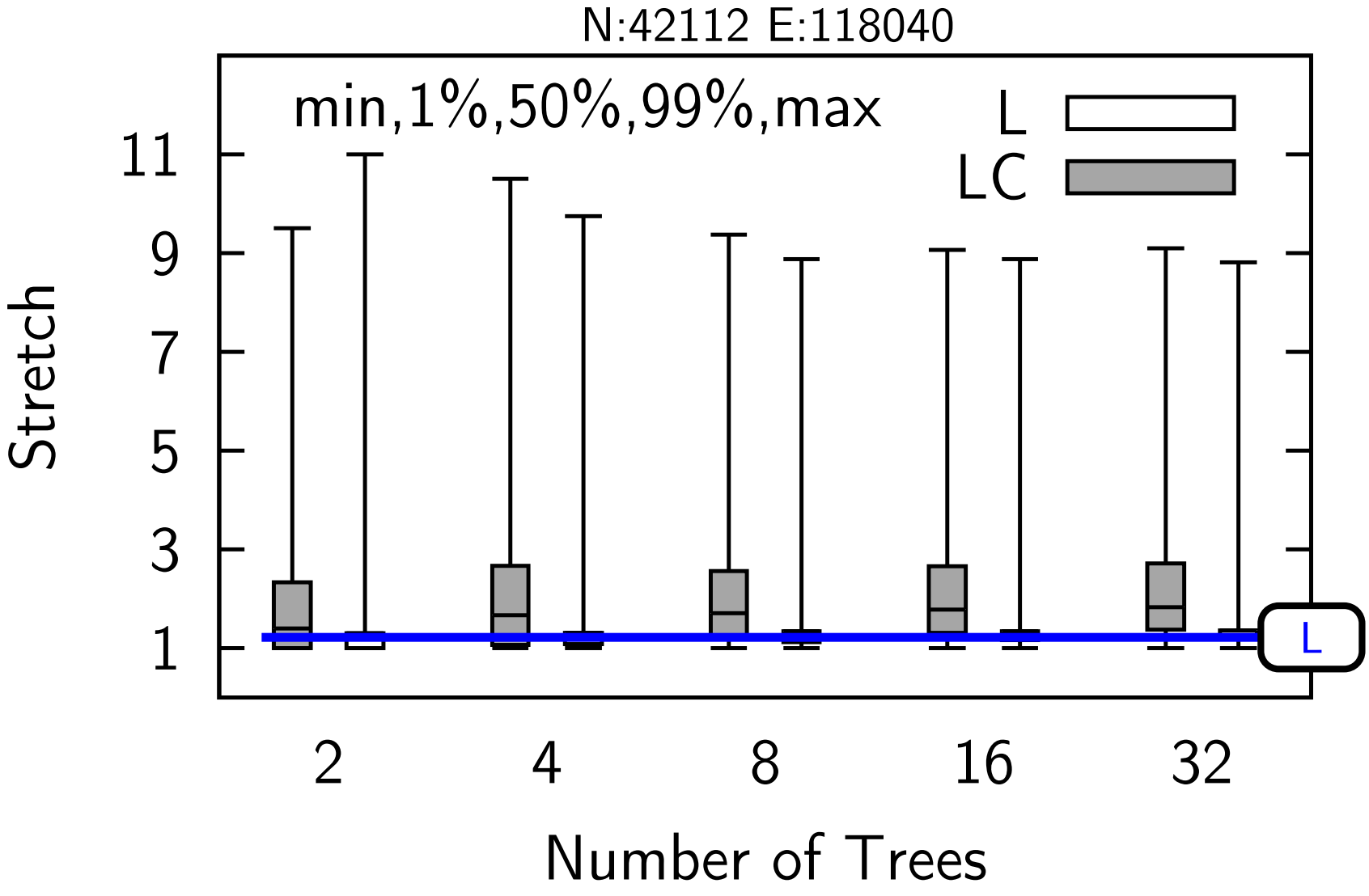
Expected Stretch, AS-level Topology



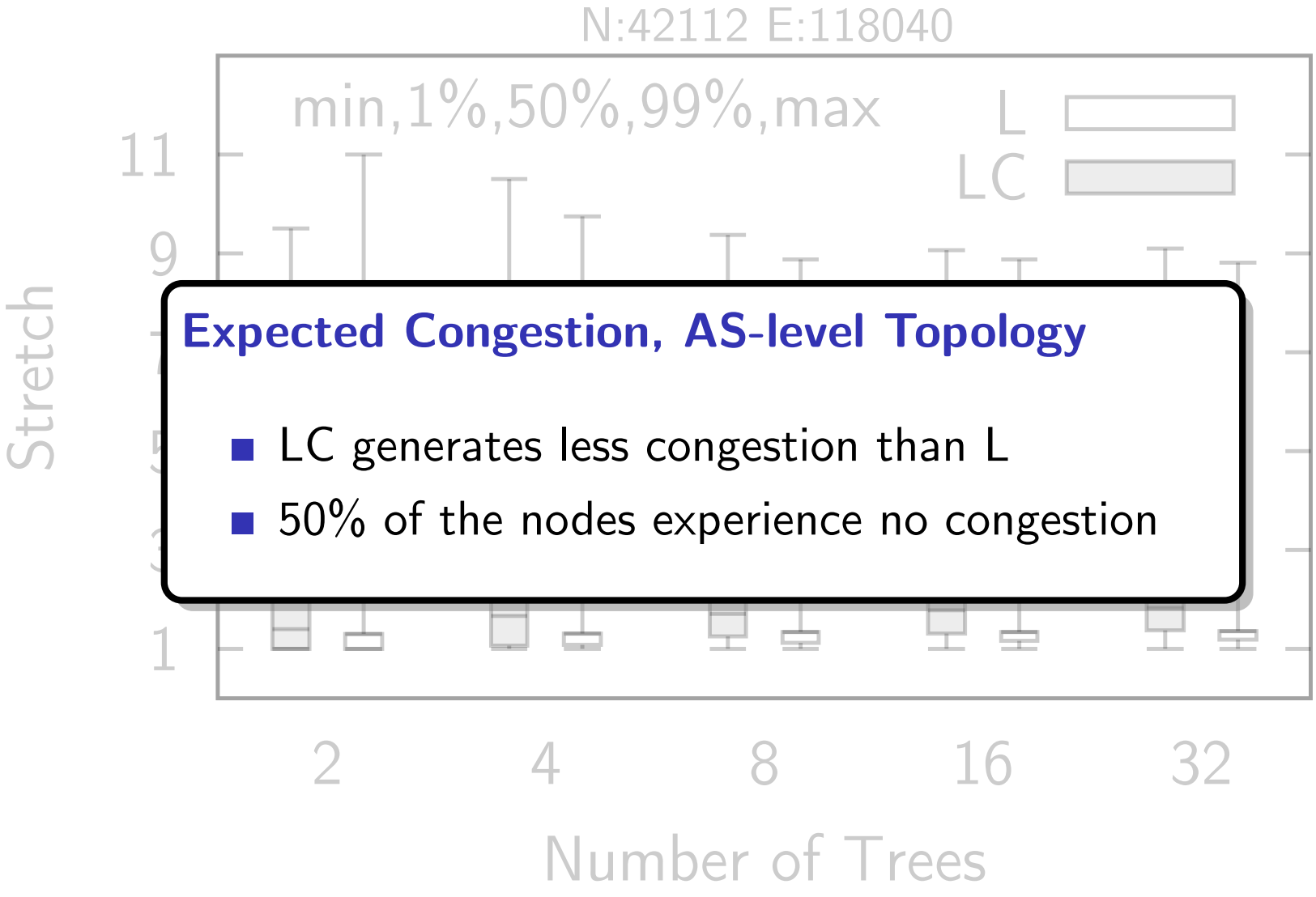
Expected Stretch, AS-level Topology



Expected Stretch, AS-level Topology



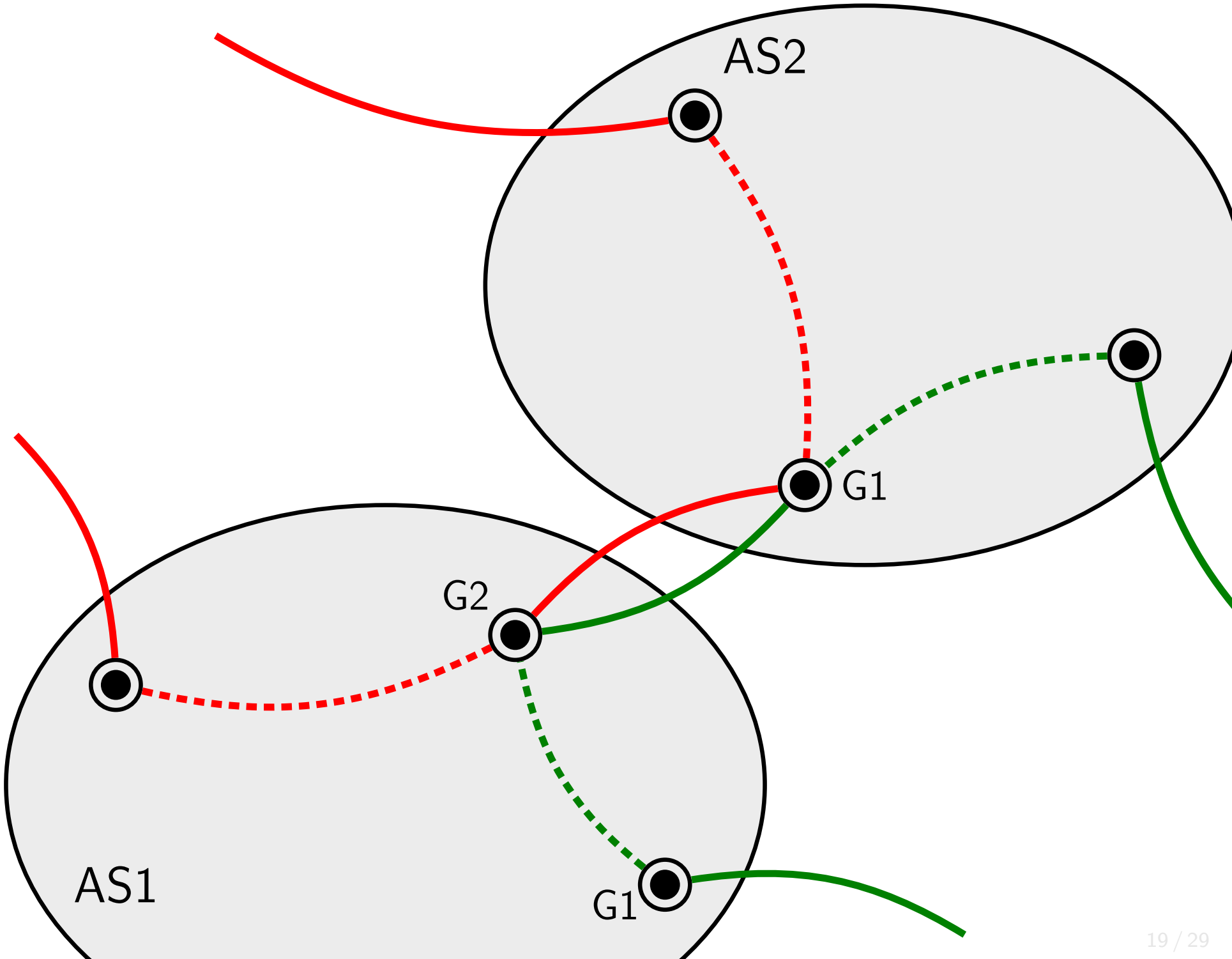
Expected Stretch, AS-level Topology

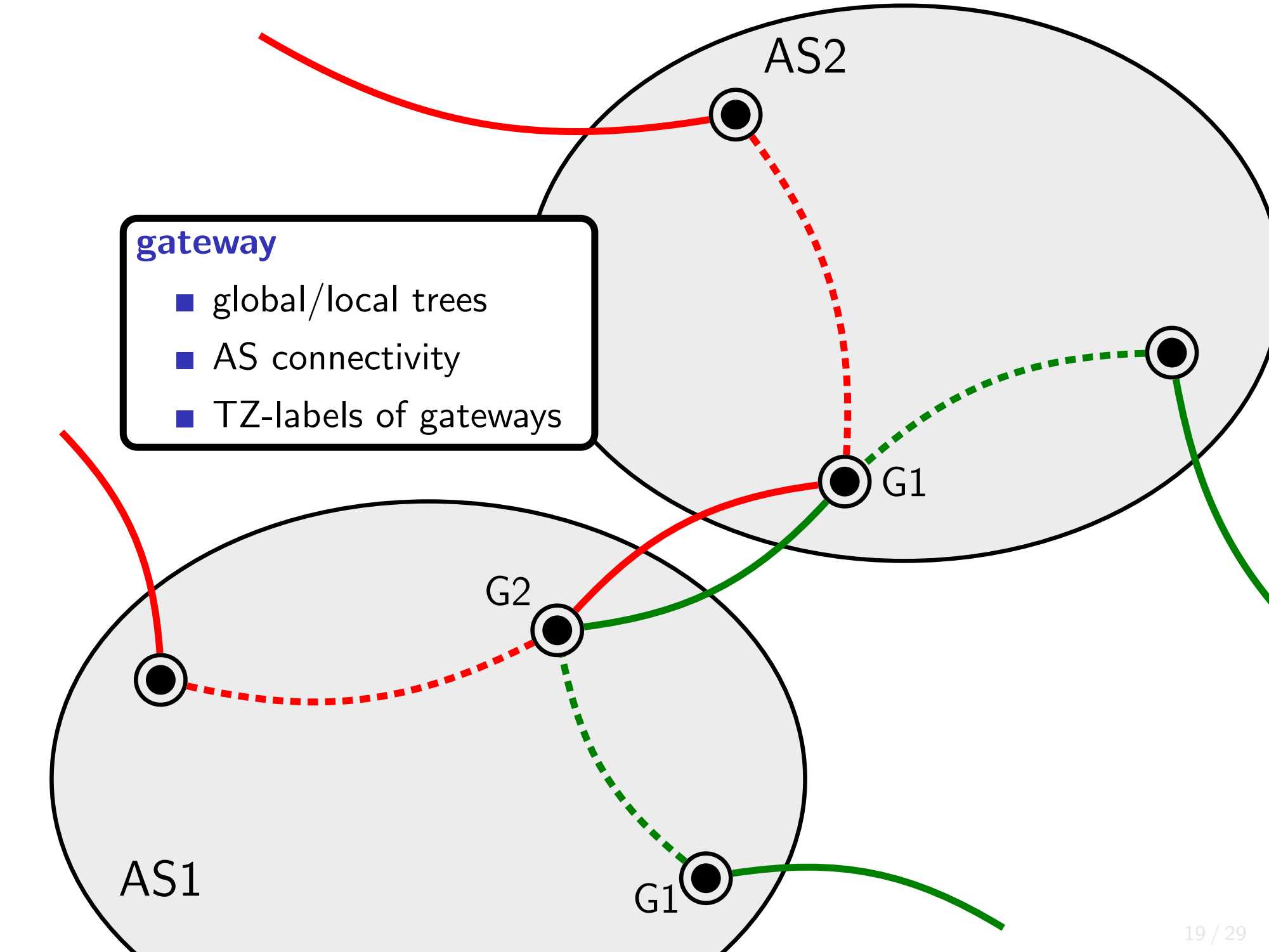


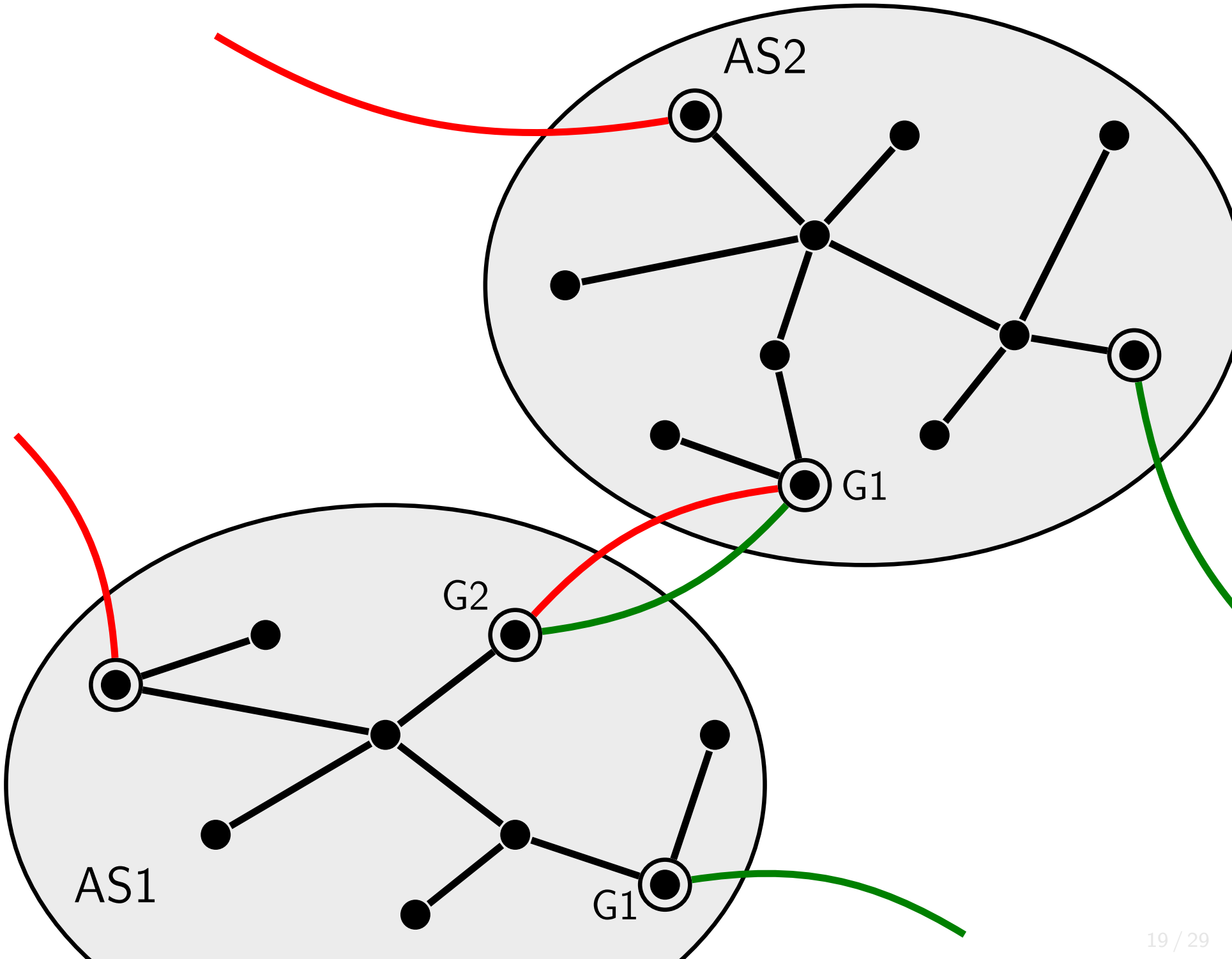
Expected Congestion, AS-level Topology

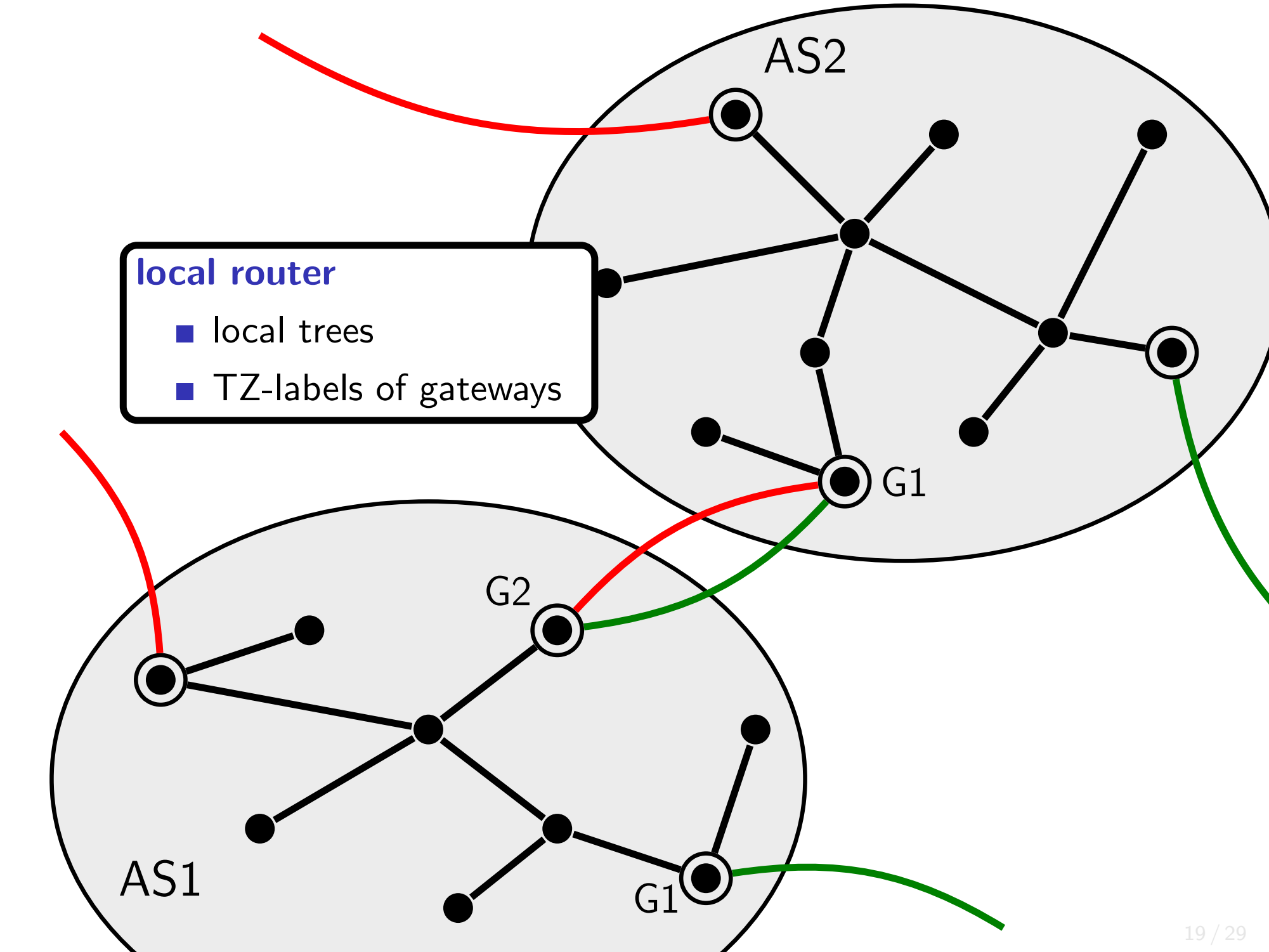
- LC generates less congestion than L
- 50% of the nodes experience no congestion

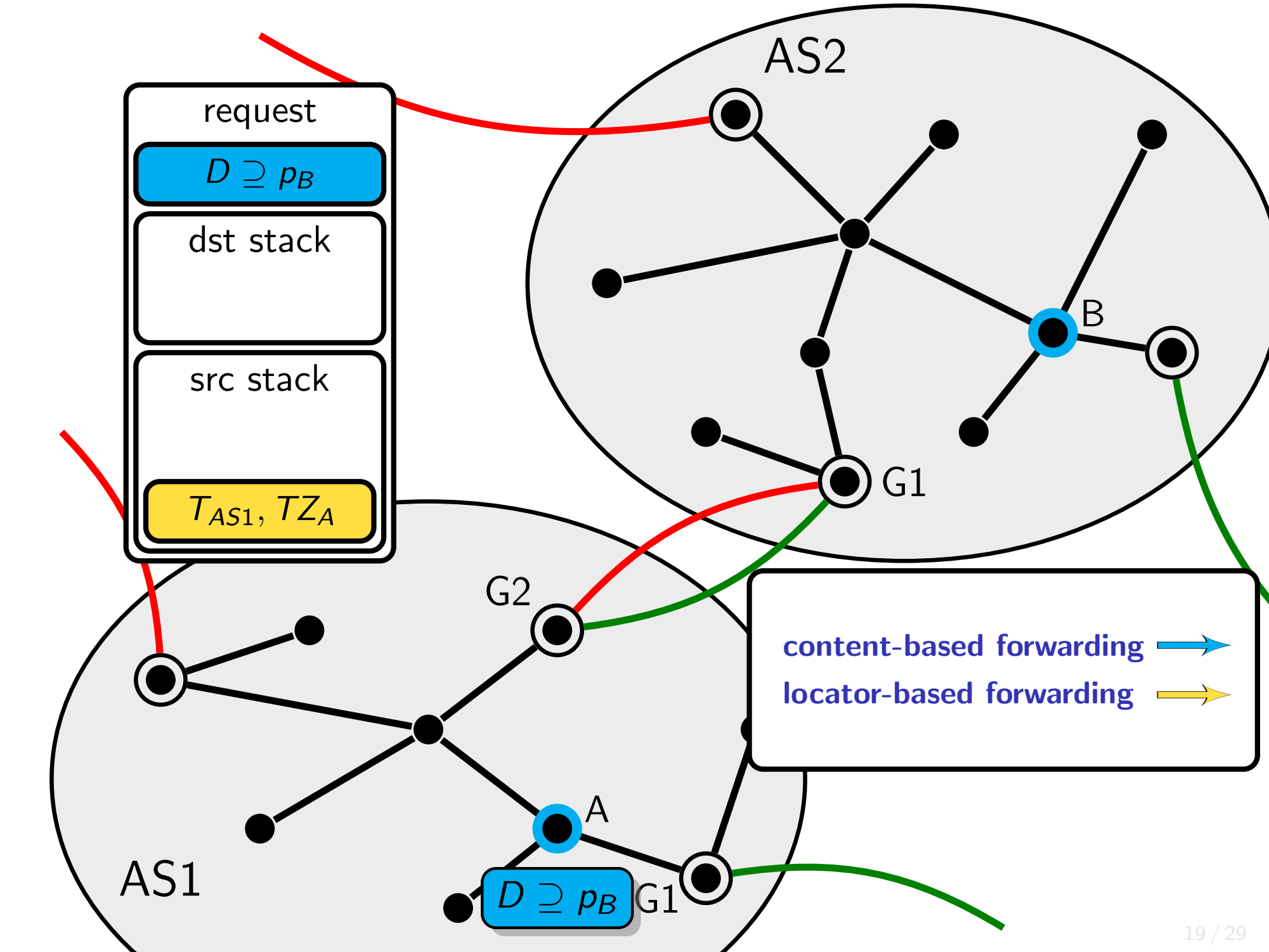
Hierarchical Multi-Tree Routing

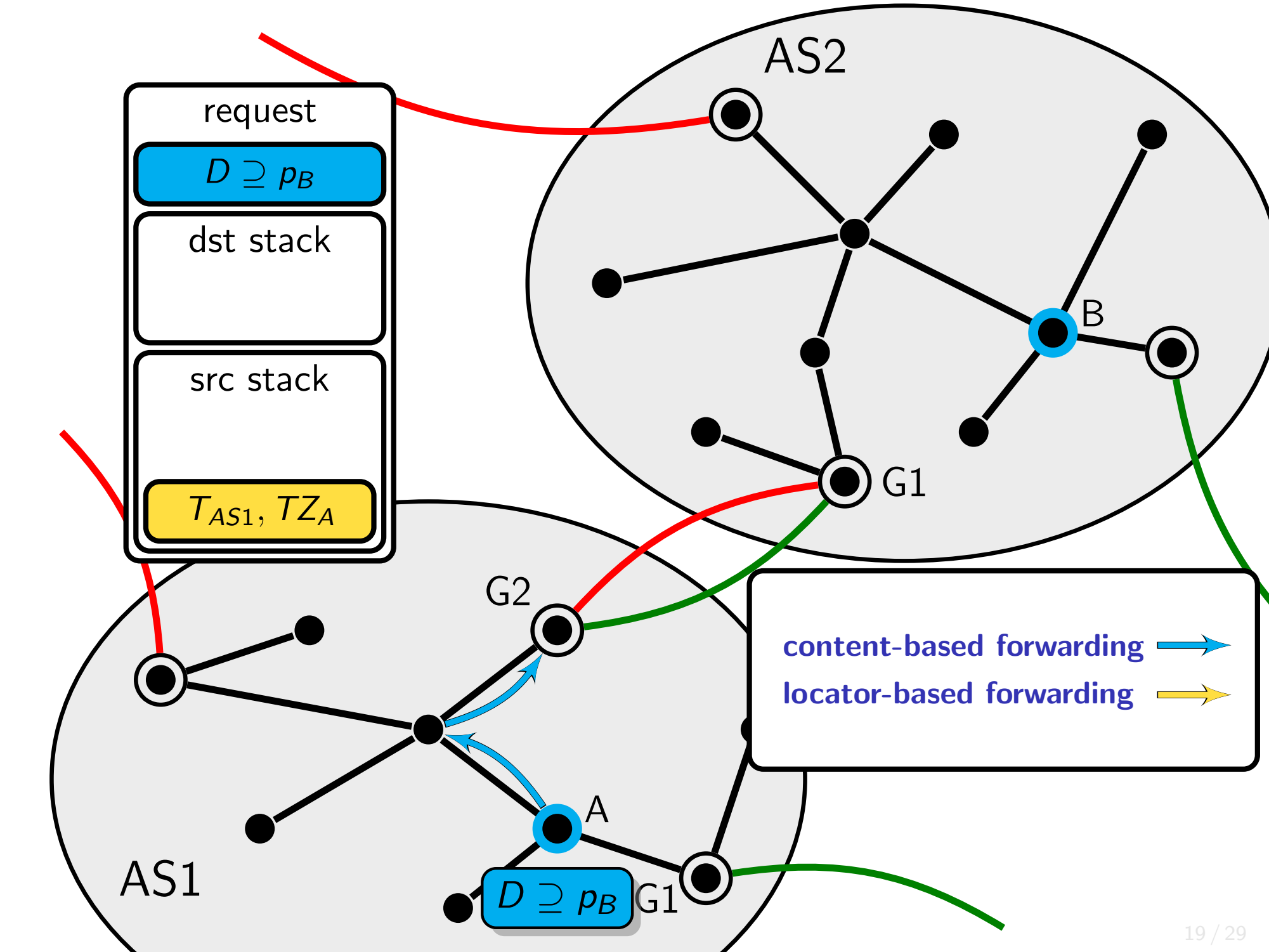

AS2
G1
G2
AS1
G1

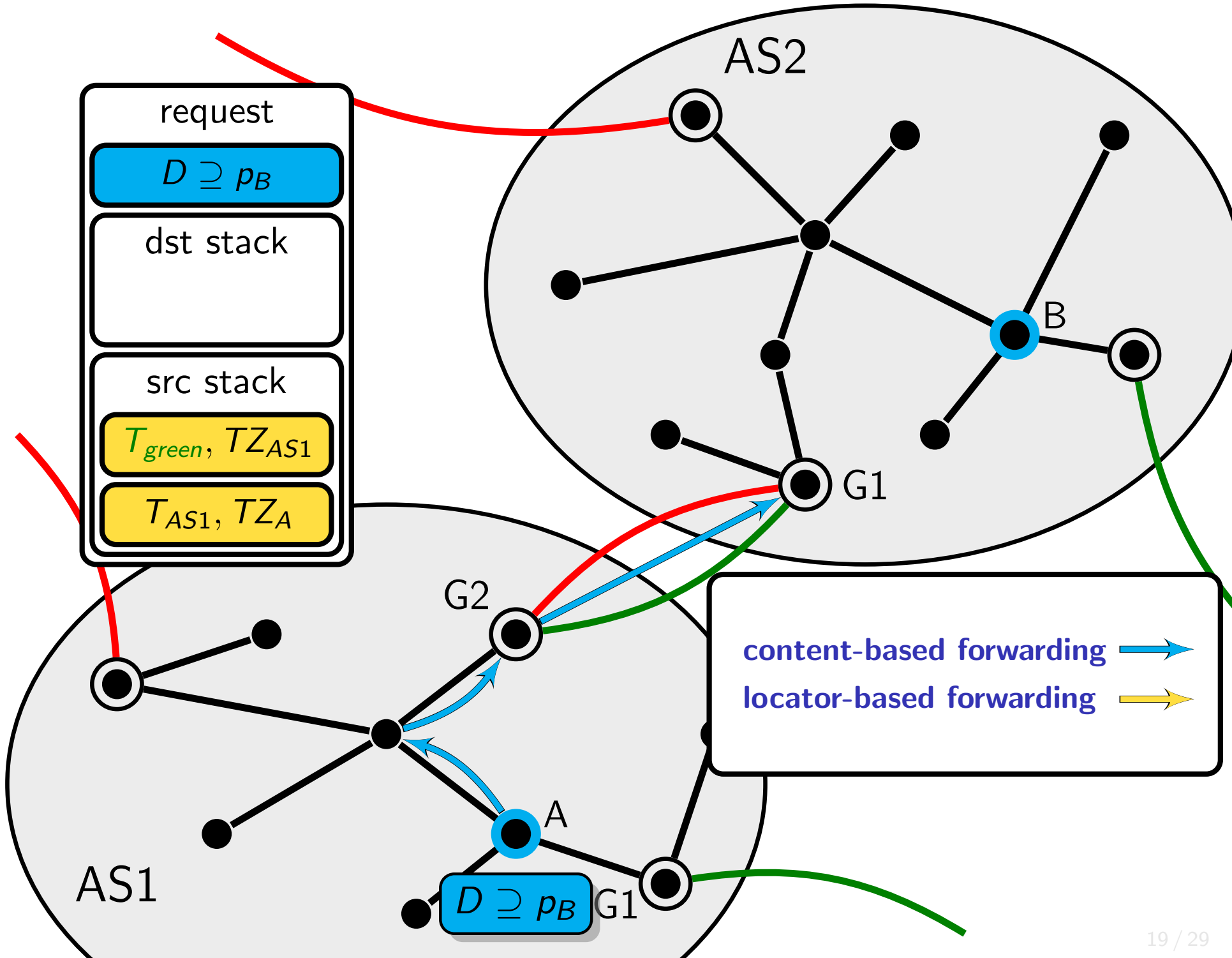

AS2
gateway
global/local trees
AS connectivity
TZ-labels of gateways
G1
G2
AS1
G1

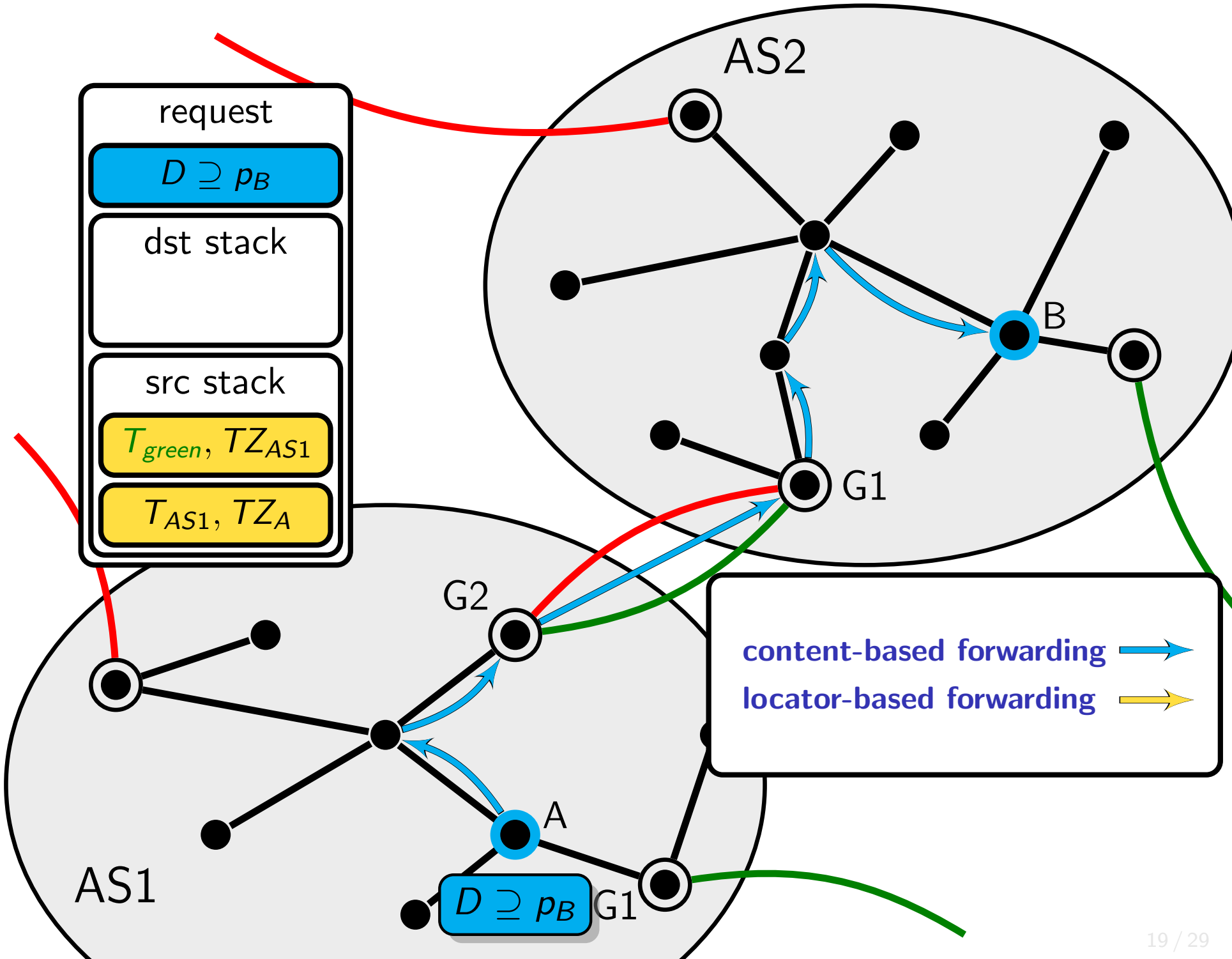

AS2
G1
G2
AS1
G1

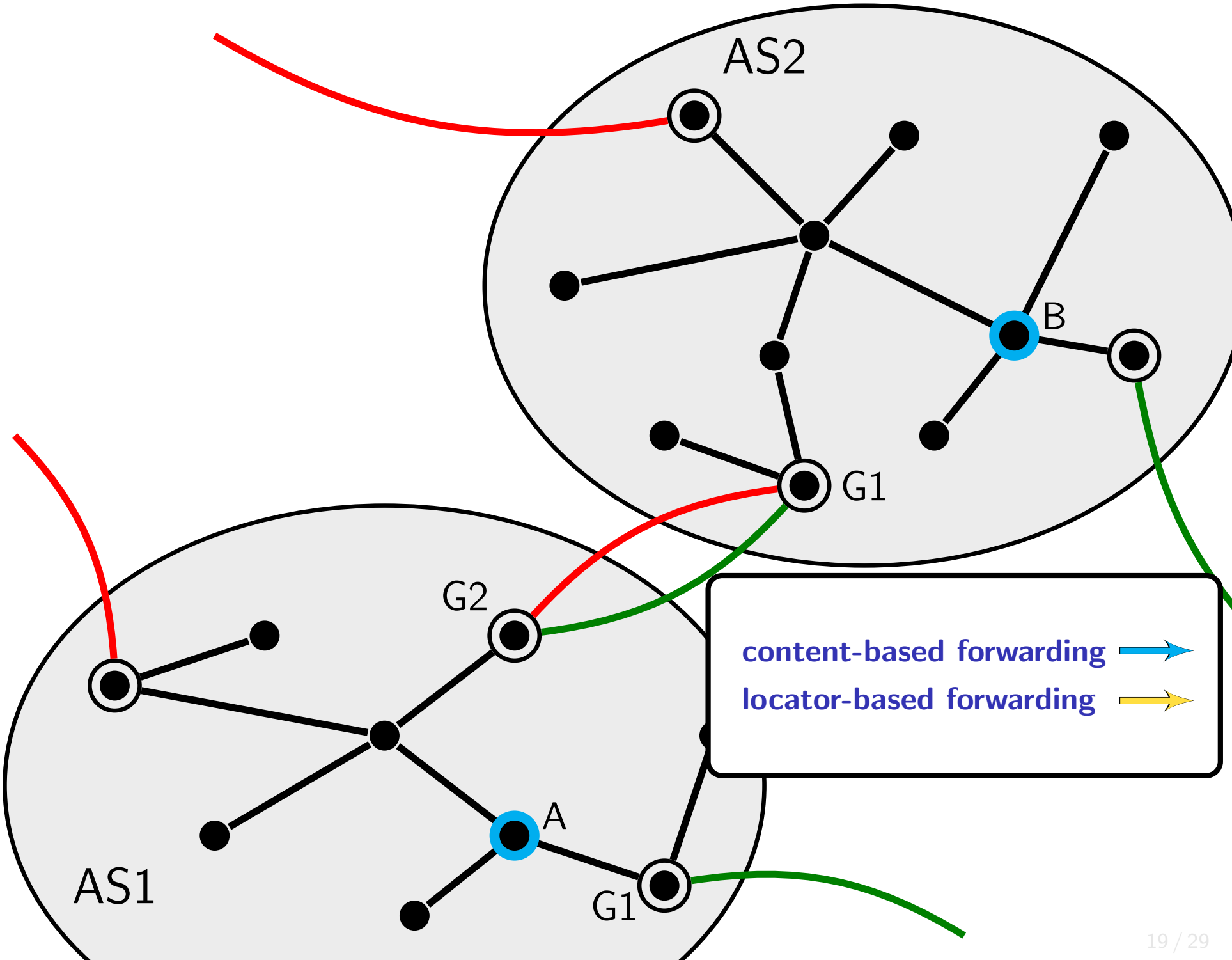

AS2
local router
local trees
TZ-labels of gateways
G1
G2
AS1
G1

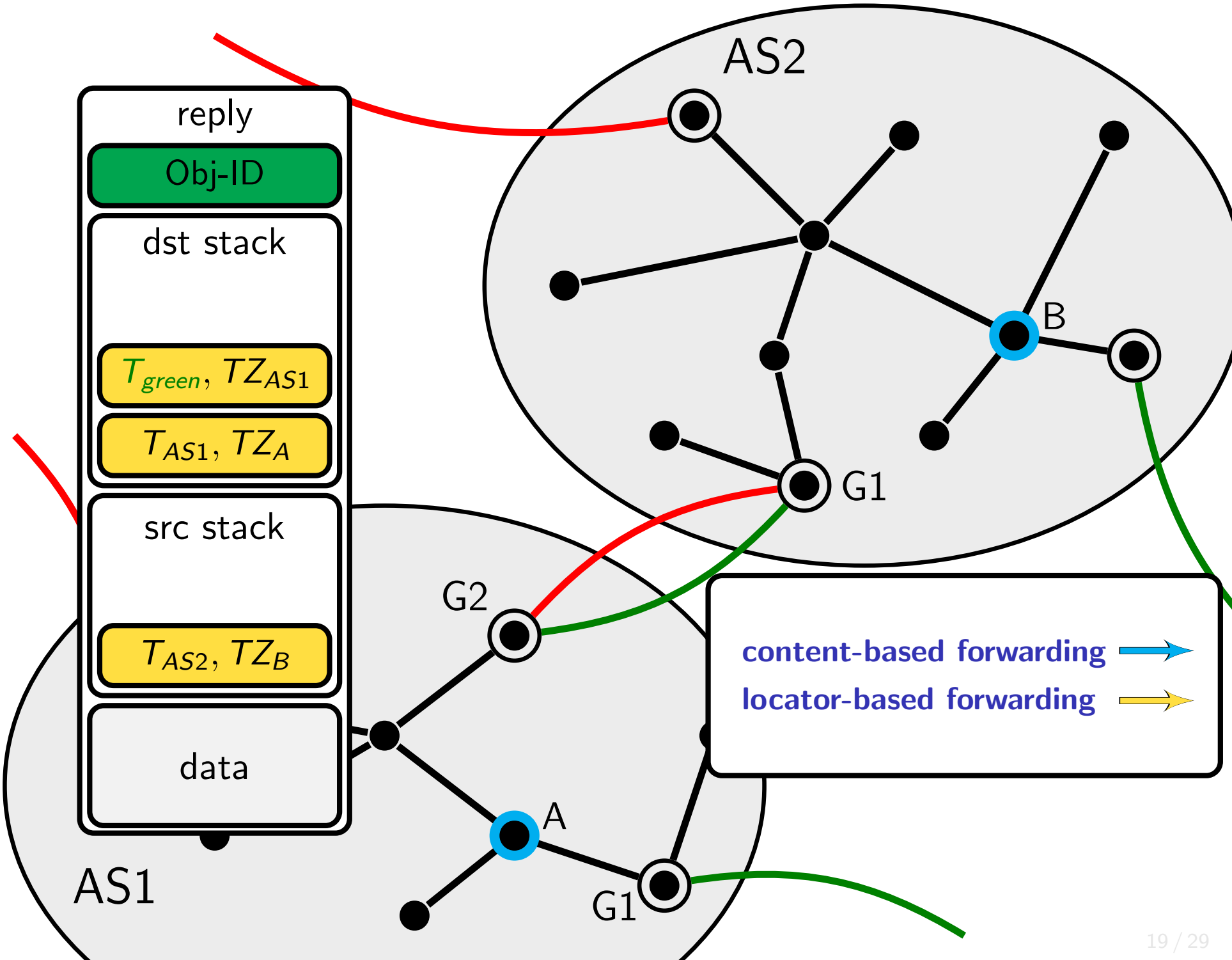

request
$D \supseteq p_B$
dst stack
src stack
T_{AS1}, TZ_A
AS2
B
G1
G2
A
AS1
$D \supseteq p_B$
G1
content-based forwarding
locator-based forwarding

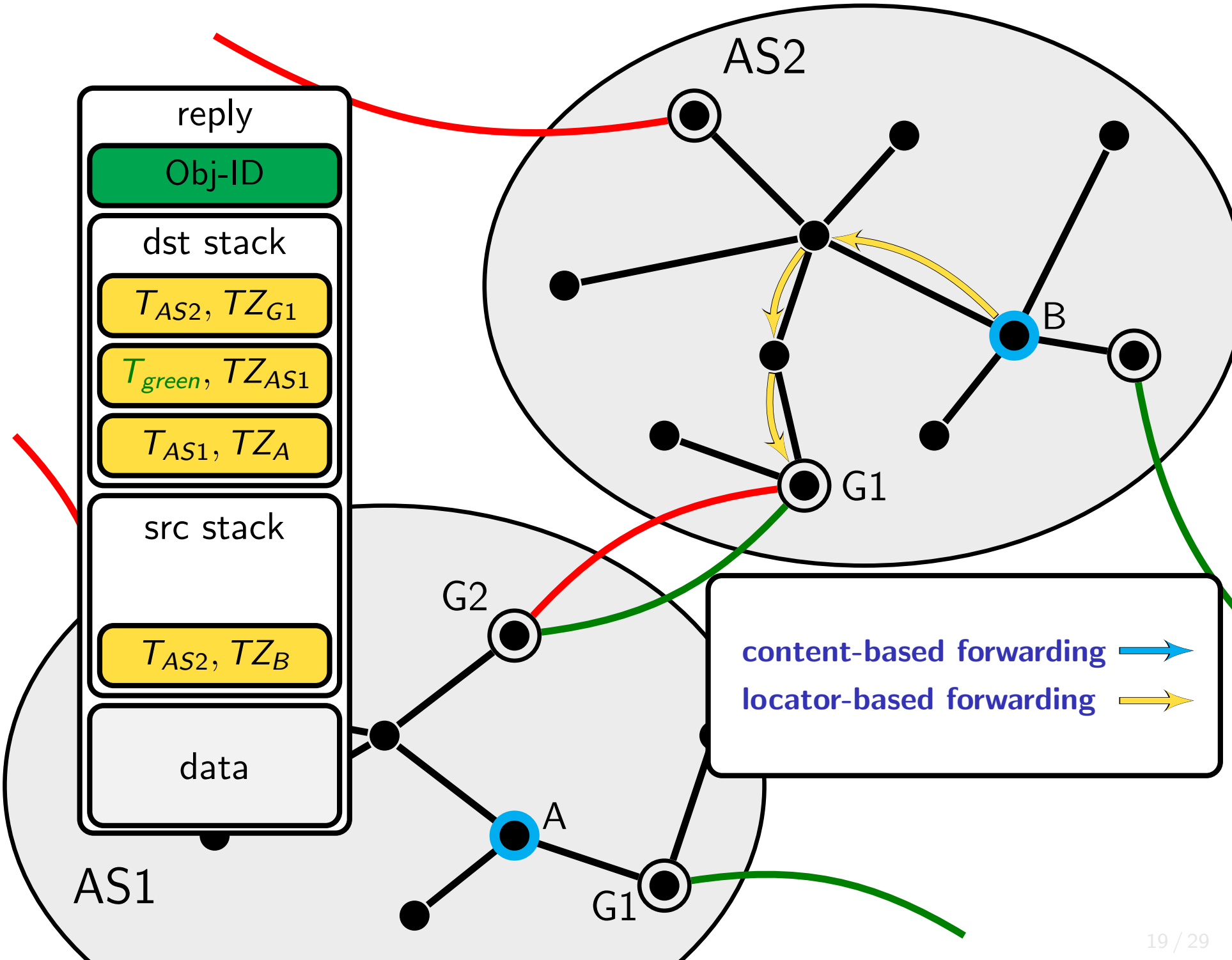

request
$D \supseteq p_B$
dst stack
src stack
T_{AS1}, TZ_A
AS2
B
G1
G2
A
AS1
$D \supseteq p_B$
G1
content-based forwarding
locator-based forwarding

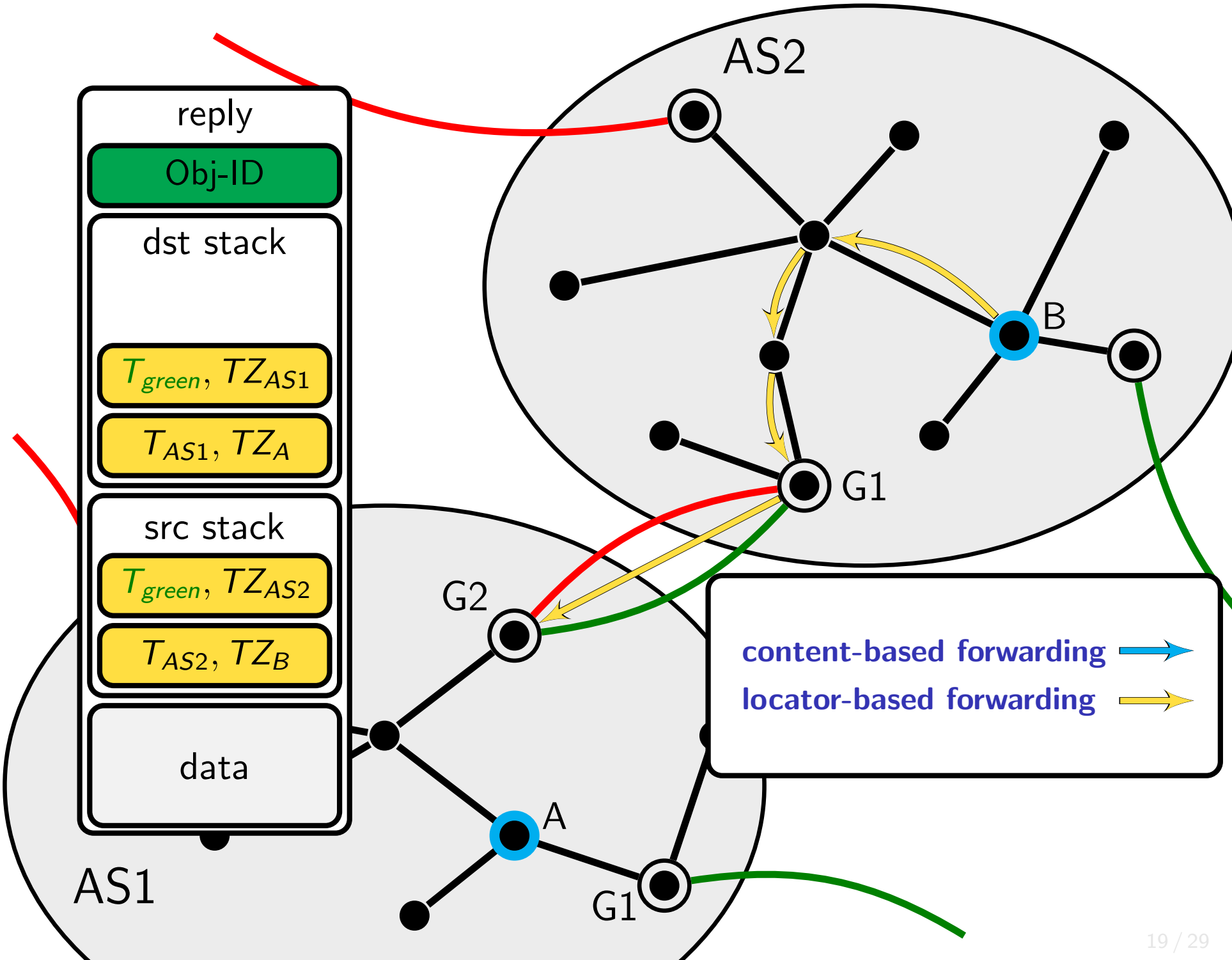

request
$D \supseteq p_B$
dst stack
src stack
T_{green}, TZ_{AS1}
T_{AS1}, TZ_A
AS2
B
G1
G2
A
AS1
$D \supseteq p_B$
G1
content-based forwarding
locator-based forwarding

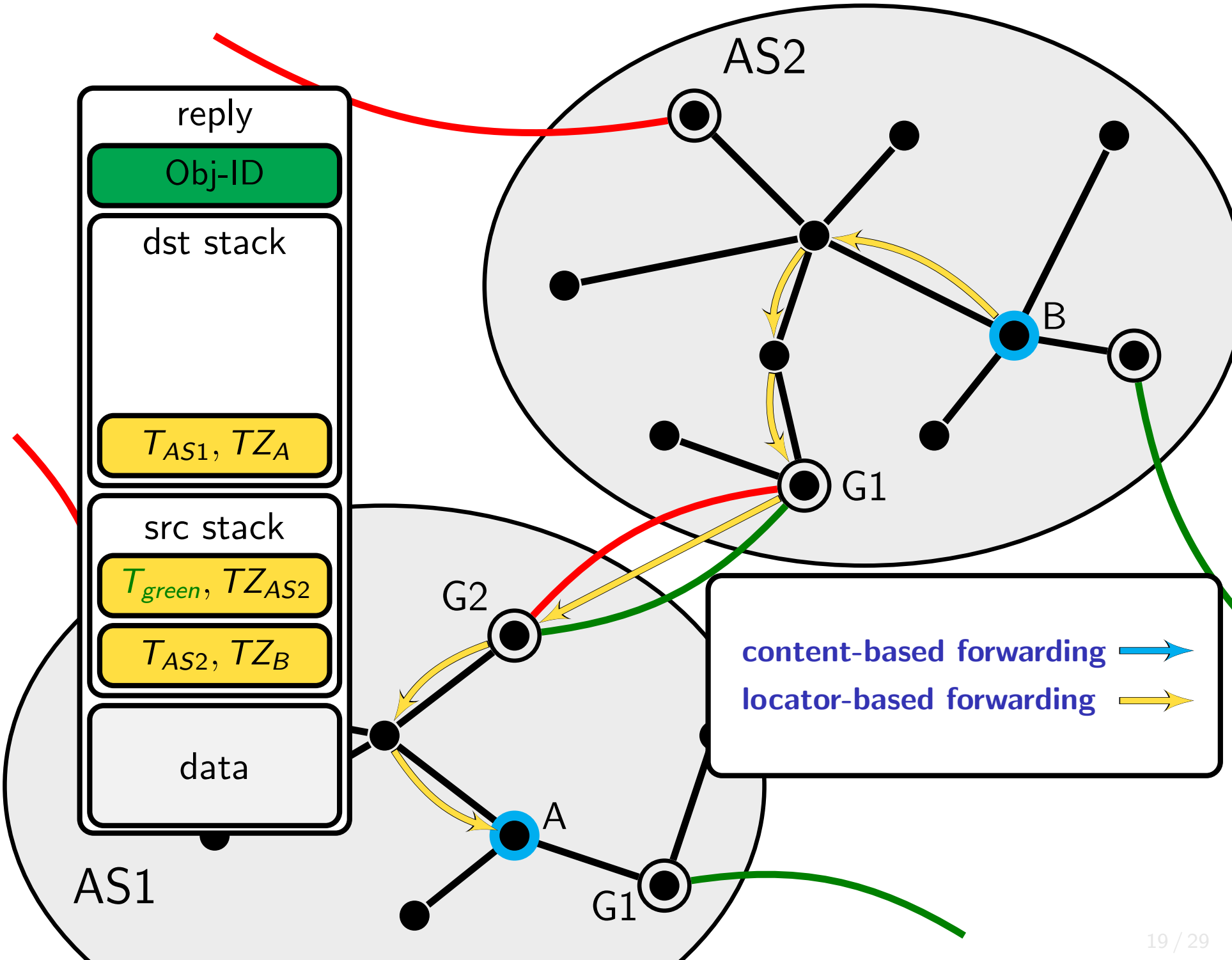

request
$D \supseteq p_B$
dst stack
src stack
T_{green}, TZ_{AS1}
T_{AS1}, TZ_A
AS2
B
G1
G2
A
AS1
$D \supseteq p_B$
G1
content-based forwarding
locator-based forwarding


AS2
B
G1
G2
content-based forwarding
locator-based forwarding
A
AS1
G1

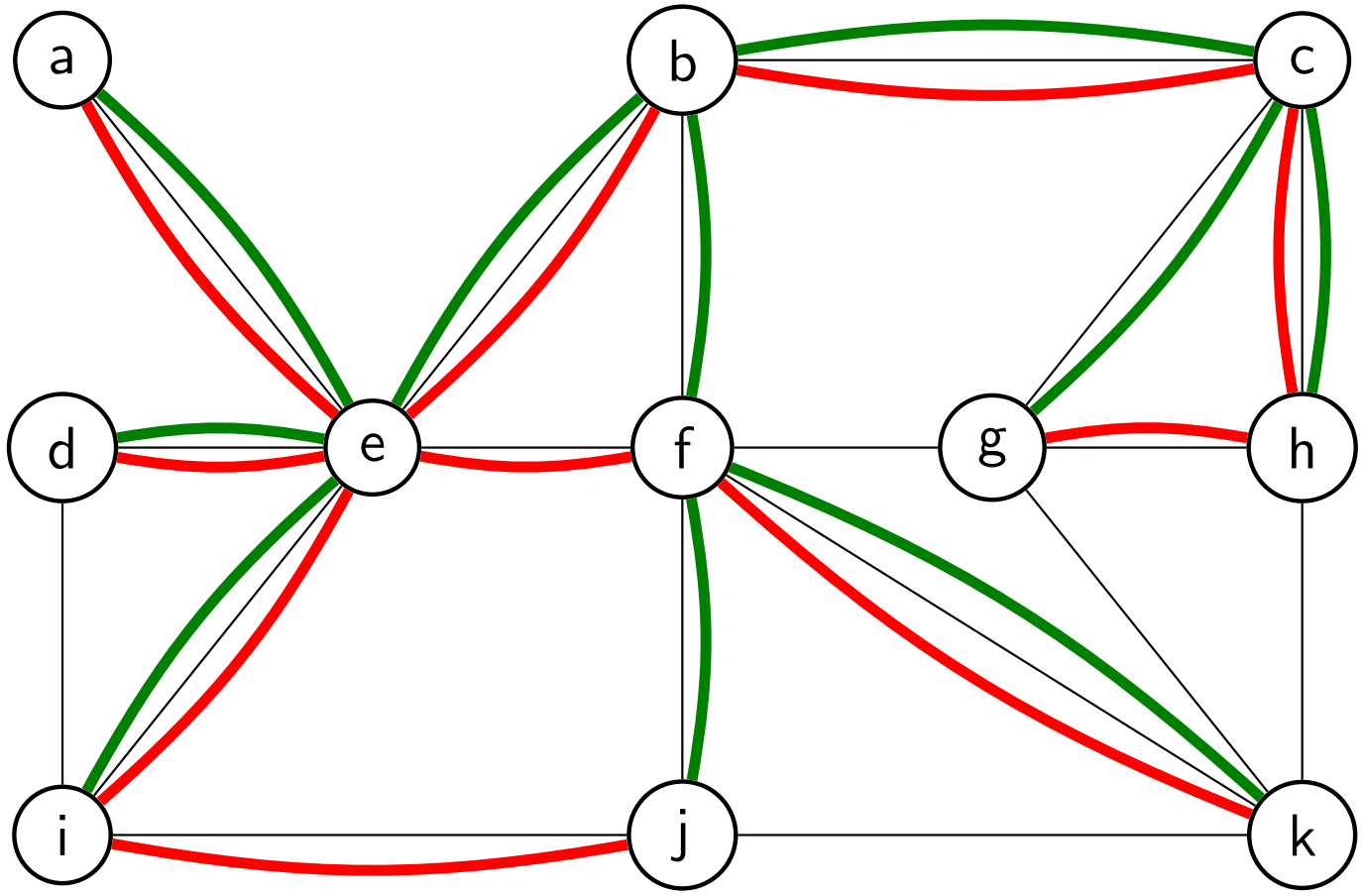

reply
Obj-ID
dst stack
T_{green}, TZ_{AS1}
T_{AS1}, TZ_A
src stack
T_{AS2}, TZ_B
data
AS2
B
G1
G2
A
G1
AS1
content-based forwarding
locator-based forwarding

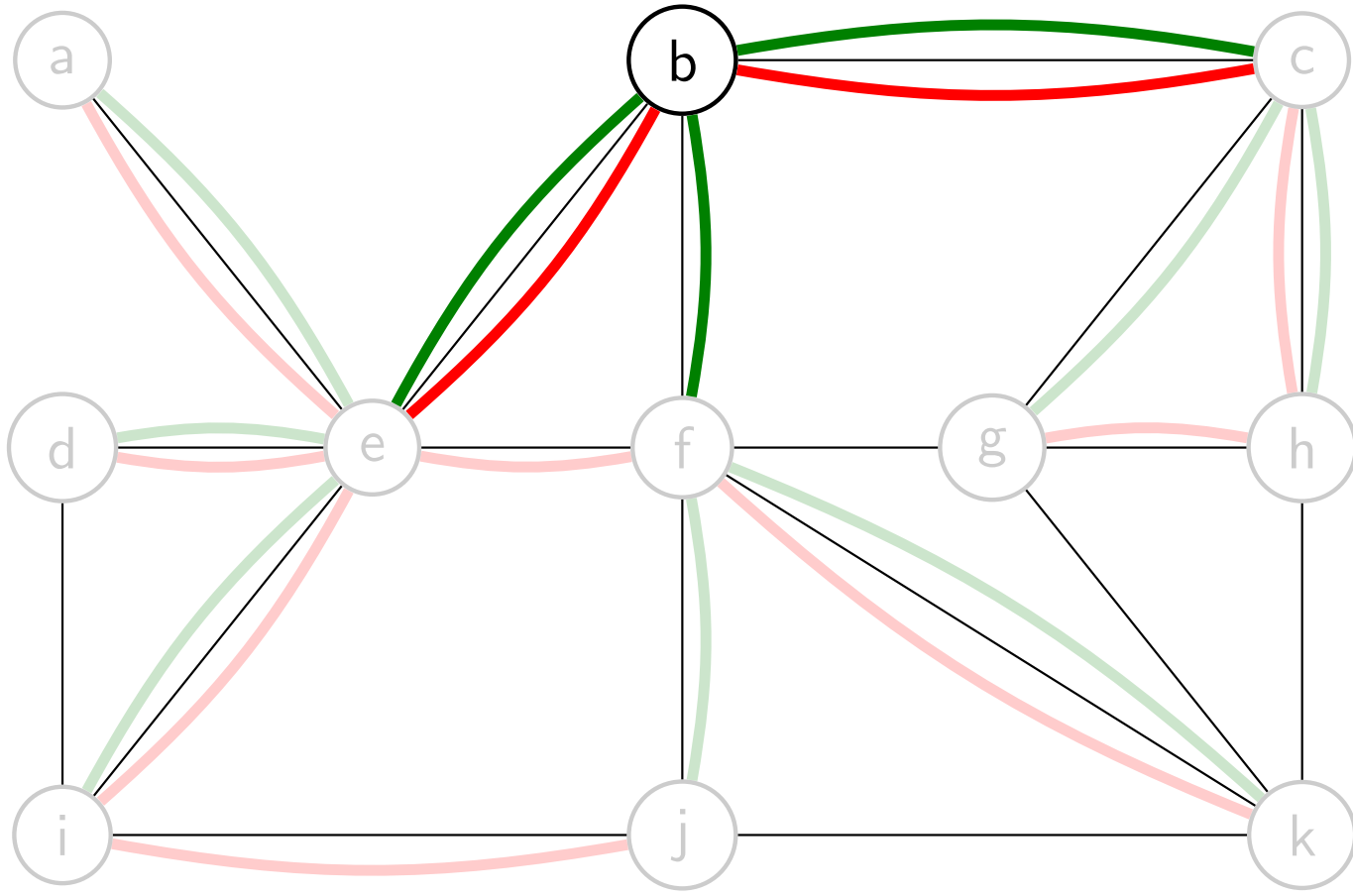

reply
Obj-ID
dst stack
T_{AS2}, TZ_{G1}
T_{green}, TZ_{AS1}
T_{AS1}, TZ_A
src stack
T_{AS2}, TZ_B
data
AS2
B
G1
G2
A
G1
AS1
content-based forwarding
locator-based forwarding

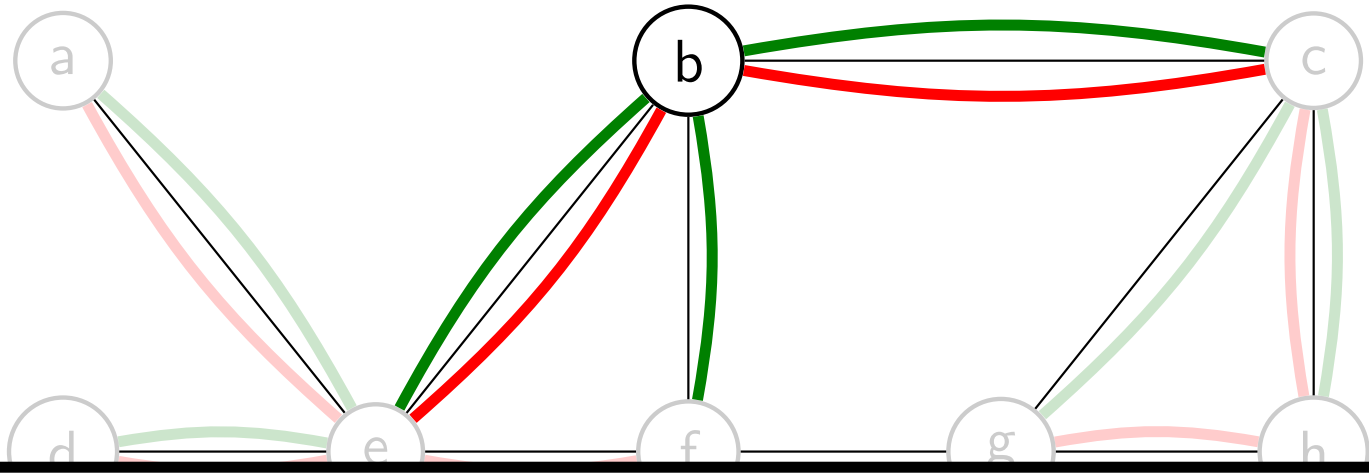

reply
Obj-ID
dst stack
T_{green}, TZ_{AS1}
T_{AS1}, TZ_A
src stack
T_{green}, TZ_{AS2}
T_{AS2}, TZ_B
data
AS2
B
G1
G2
A
G1
AS1
content-based forwarding
locator-based forwarding


reply
Obj-ID
dst stack
T_{AS1}, TZ_A
src stack
T_{green}, TZ_{AS2}
T_{AS2}, TZ_B
data
AS2
B
G1
G2
A
G1
AS1
content-based forwarding
locator-based forwarding

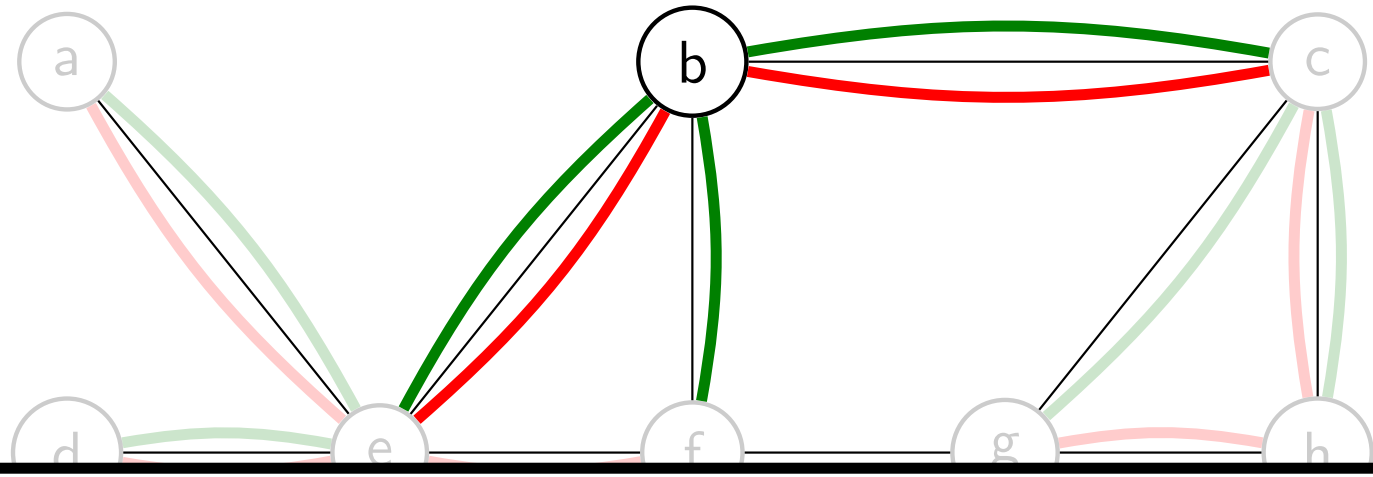
RIB Representation and Maintenance


a
b
c
d
e
f
g
h
i
j
k


a
b
c
d
e
f
g
h
i
j
k

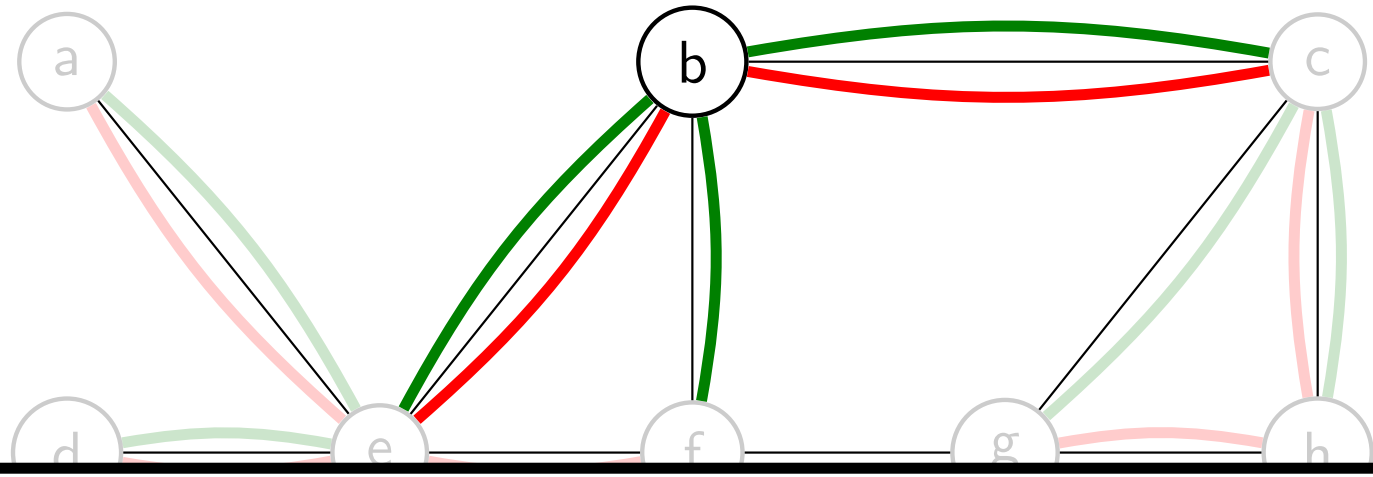


Neighbors	
tree T	neighbors
T_{green}	e, f, c
T_{red}	e, c



Neighbors	
tree T	neighbors
T_{green}	e, f, c
T_{red}	e, c

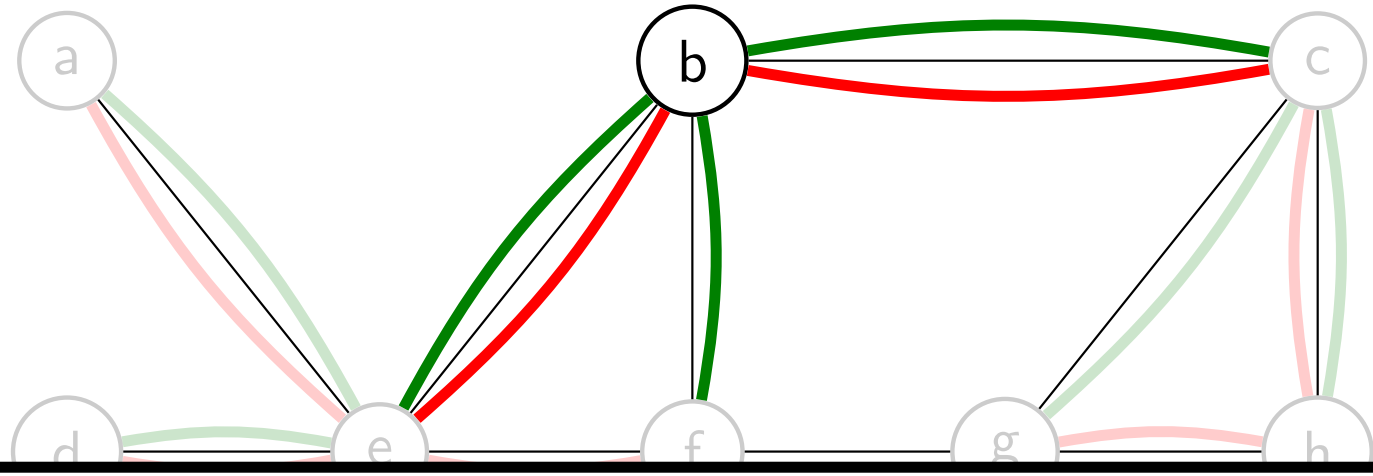
TZ-labels
TZ_b^{green}
TZ_b^{red}



Neighbors	
tree T	neighbors
T_{green}	e, f, c
T_{red}	e, c

TZ-labels
TZ_b^{green}
TZ_b^{red}

RIB	
tree T, next-hop w	predicate $P_{T,w}$
T_{green}, c	$p_c \vee p_g \vee p_h$
T_{green}, f	$p_f \vee p_j \vee p_k$
T_{green}, e	$p_e \vee p_a \vee p_d \vee p_i$
T_{red}, c	$p_c \vee p_g \vee p_h$
T_{red}, e	$p_a \vee p_d \vee p_e \vee p_f \vee p_i \vee p_j \vee p_k$



Neighbors	
tree T	neighbors
T_{green}	e, f, c
T_{red}	e, c

TZ-labels
TZ_b^{green}
TZ_b^{red}

RIB	
tree T, next-hop w	predicate $P_{T,w}$
T_{green}, c	$p_c \vee p_g \vee p_h$
T_{green}, f	$p_f \vee p_j \vee p_k$
T_{green}, e	$p_e \vee p_a \vee p_d \vee p_i$
T_{red}, c	$p_c \vee p_g \vee p_h$
T_{red}, e	$p_a \vee p_d \vee p_e \vee p_f$ $\vee p_i \vee p_j \vee p_k$

Compress and **update** the RIB

Compression

Compression

(1) Subset Aggregation

Compression

(1) Subset Aggregation

{ICN14, Paris, routing, paper}

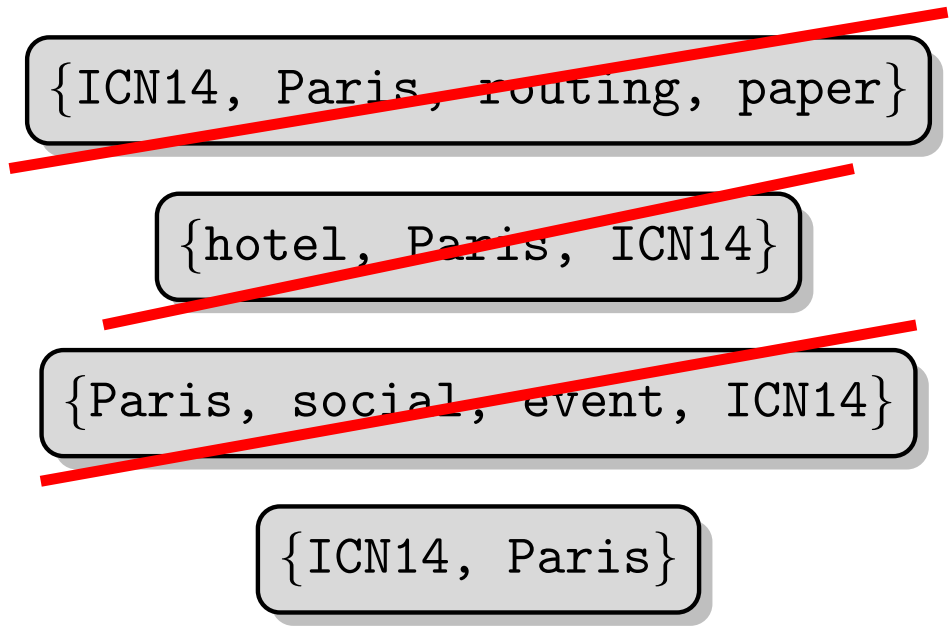
{hotel, Paris, ICN14}

{Paris, social, event, ICN14}

{ICN14, Paris}

Compression

(1) Subset Aggregation



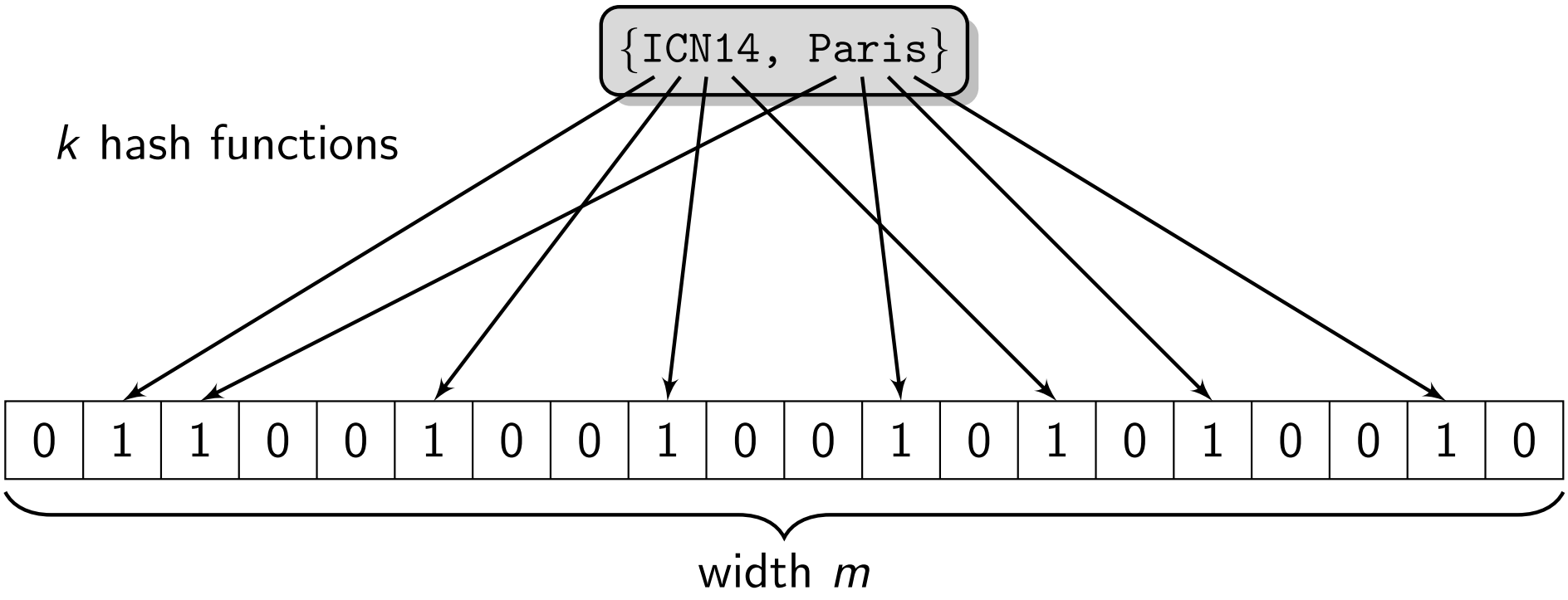
(2) Bloom Filters

(2) Bloom Filters

{ICN14, Paris}

Compression

(2) Bloom Filters



our setting: k=7 and m=192 (24B)

(3) Index by Descriptors

Compression

(3) Index by Descriptors

RIB	
tree T, next-hop w	predicate $P_{T,w}$
T_{green}, c	00100101 01010000 01000001
T_{green}, f	00100100 01010000 01100100
T_{green}, e	00010000 10000101
T_{red}, c	00100101 01000001
T_{red}, e	00010000 10000101 00100100

Compression

(3) Index by Descriptors

RIB	
tree T, next-hop w	predicate $P_{T,w}$
T_{green}, c	00100101 ● 01010000 ● 01000001 ●
T_{green}, f	00100100 ● 01010000 ● 01100100 ●
T_{green}, e	00010000 ● 10000101 ●
T_{red}, c	00100101 ● 01000001 ●
T_{red}, e	00010000 ● 10000101 ● 00100100 ●

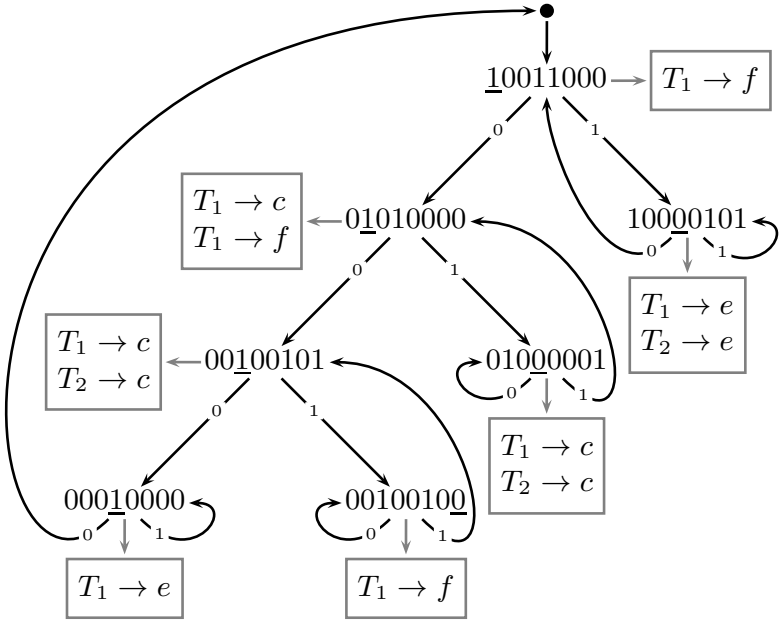
Compression

(3) Index by Descriptors

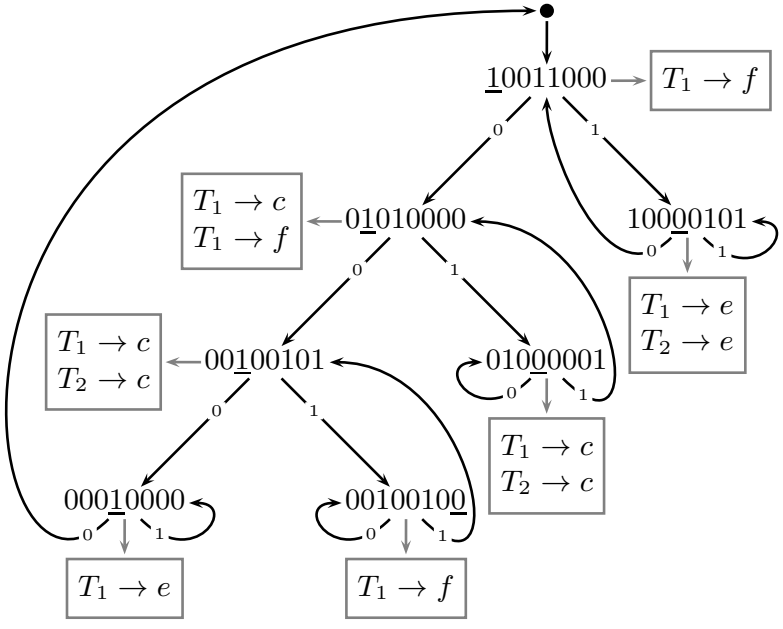
RIB	
tree T, next-hop w	predicate $P_{T,w}$
T_{green}, c	00100101 🔴 01010000 🔵 01000001 🟢
T_{green}, f	00100100 🟠 01010000 🔵 01100100 ⚫
T_{green}, e	00010000 🟡 10000101 🟣
T_{red}, c	00100101 🔴 01000001 🟢
T_{red}, e	00010000 🟡 10000101 🟣 00100100 🟠

Compressed RIB	
descriptor D	tree T, next-hop w
🔴 00100101	(T_{green}, c), (T_{red}, c)
🔵 01010000	(T_{green}, c), (T_{green}, f)
🟢 01000001	(T_{green}, c), (T_{red}, c)
🟠 00100100	(T_{green}, f), (T_{red}, e)
⚫ 01100100	(T_{green}, f)
🟡 00010000	(T_{green}, e), (T_{red}, e)
🟣 10000101	(T_{green}, e), (T_{red}, e)

RIB as PATRICIA Trie and Updates

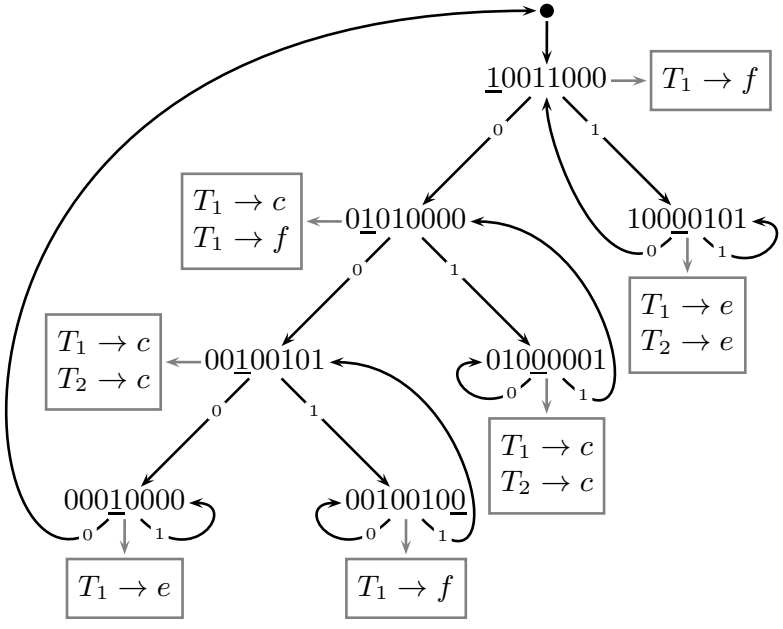


RIB as PATRICIA Trie and Updates



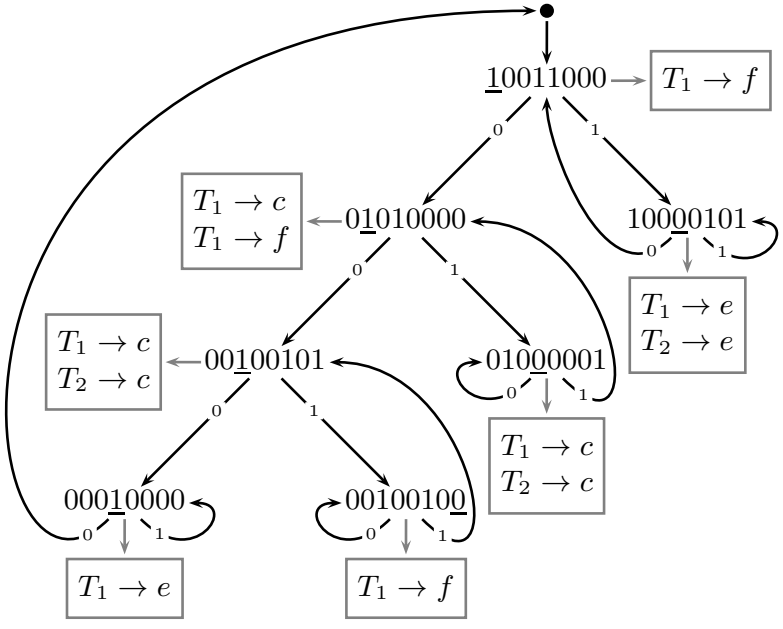
- Minimal size

RIB as PATRICIA Trie and Updates



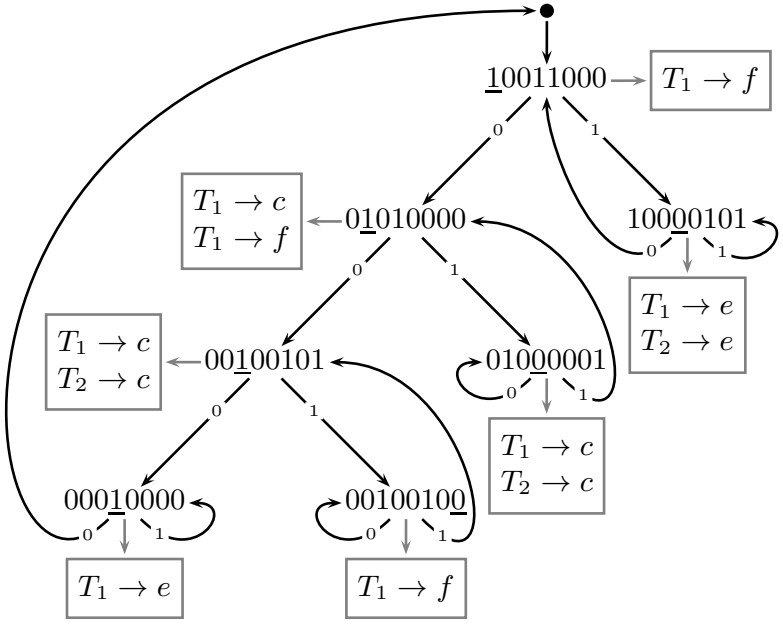
- Minimal size
- Subset/superset check = walk

RIB as PATRICIA Trie and Updates



- Minimal size
- Subset/superset check = walk
- Shortcut by prefix

RIB as PATRICIA Trie and Updates



- Minimal size
- Subset/superset check = walk
- Shortcut by prefix

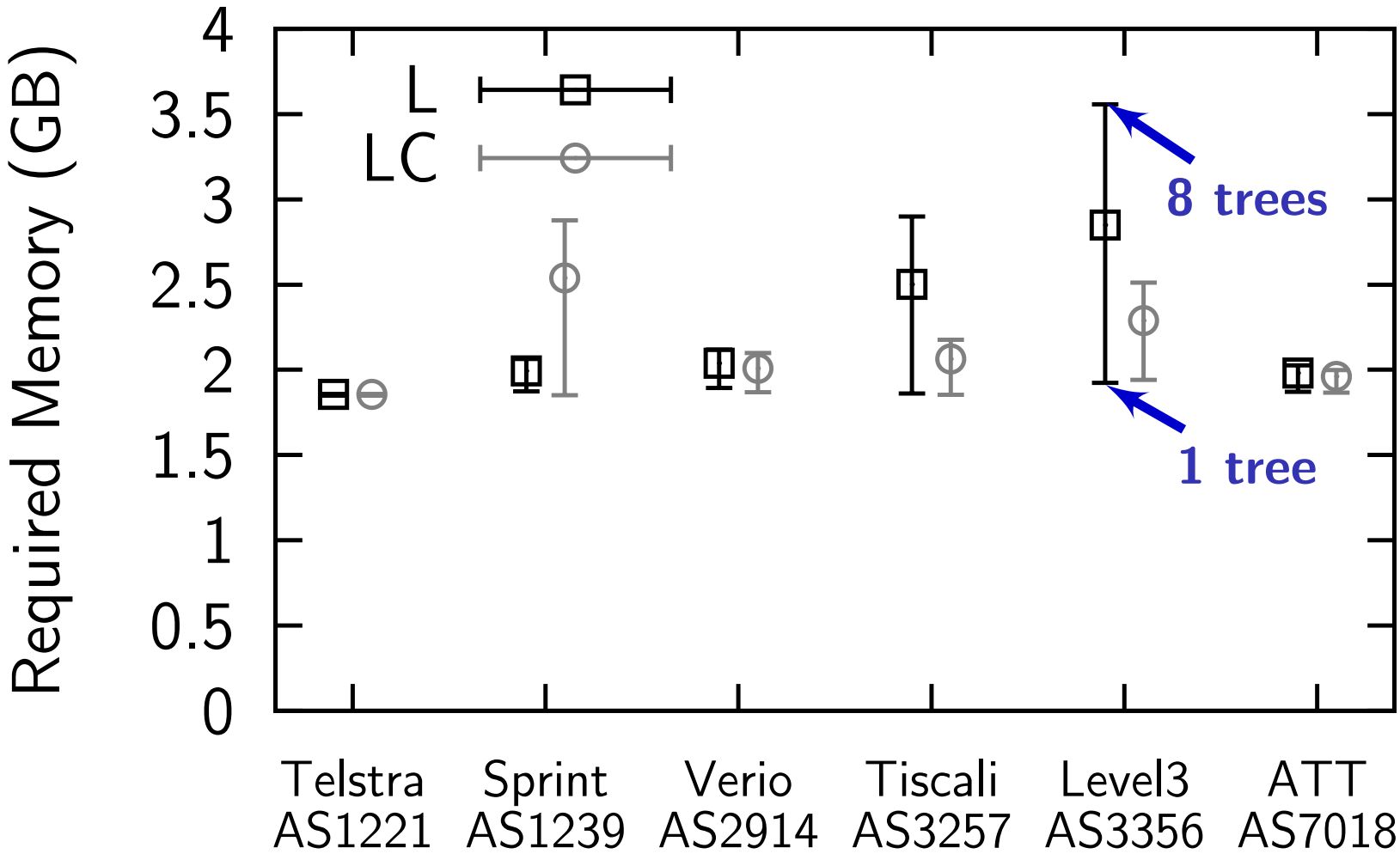
```
void apply_delta (map<int,delta> & result,
        delta update, int ifx, int tree) {
  for (filter f : update.removals)
    remove_filter(result, f, ifx, tree);
  for (filter f : update.additions)
    add_filter(result, f, ifx, tree); }

void add_filter (map<int,delta> & result,
                 filter f, int ifx, int tree) {
  if (!exists_subset_of(f, ifx, tree)) {
    add(f, ifx, tree);
    remove_supersets_of(f, ifx, tree);
    for (int i : interfaces[tree])
      if (i != ifx && no_subsets_on_other_ifx(f, i, tree))
        result[i].additions.add(f); } }

void remove_filter (map<int,delta> & result,
                    filter f, int ifx, int tree) {
  if (exists_filter(f, ifx, tree)) {
    remove(f, ifx, tree);
    for (i : interfaces[tree]) {
      if (i != ifx && no_subsets_on_other_ifx(f, i, tree)) {
        result[i].removals.add(f);
        result[i].additions.add(supersets_of(f, tree)); } } } }
```

Evaluation: 500 descriptors/sec

RIB Compression (Aggregation): Evaluation

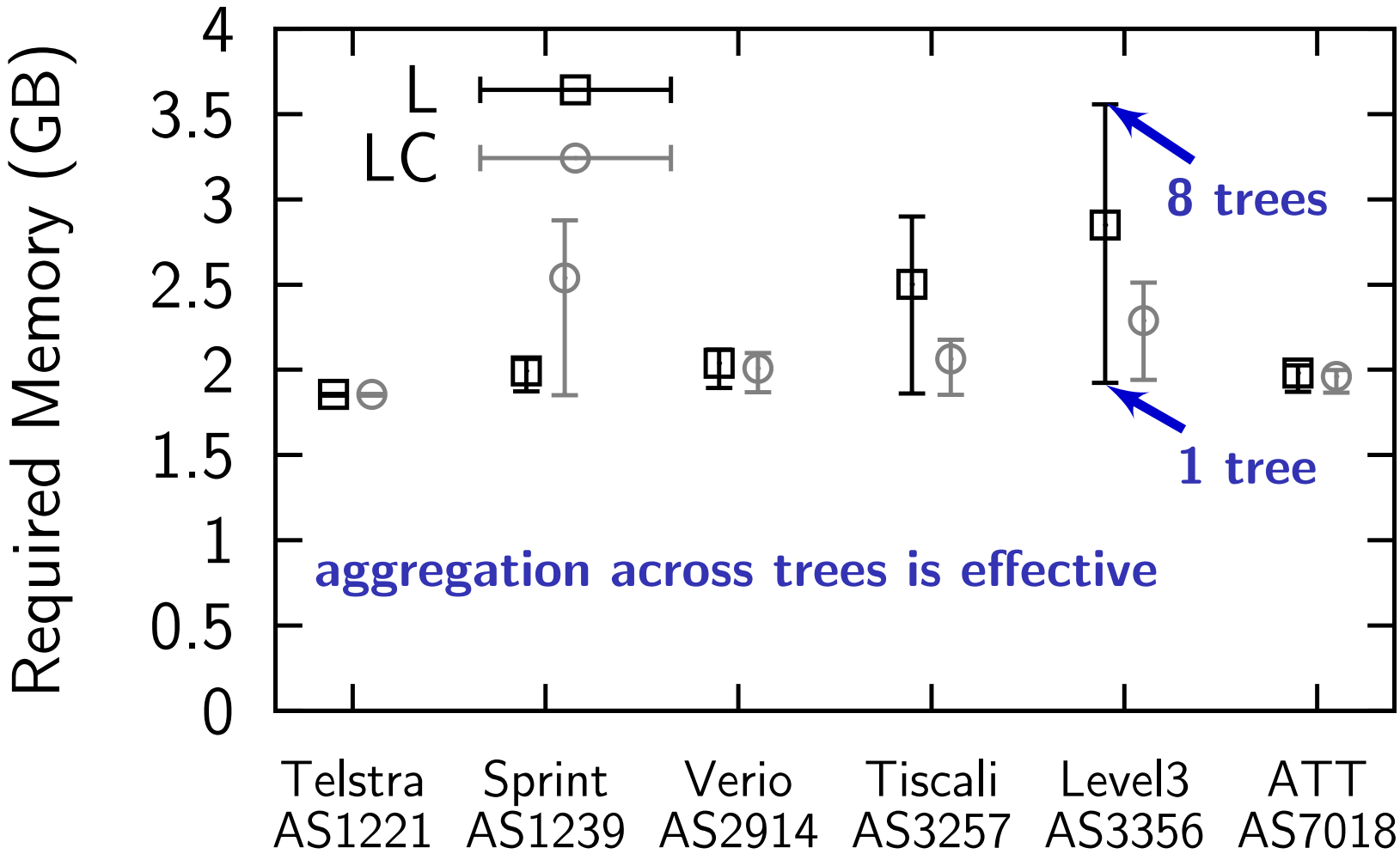
Gateway RIB Sizes

50M users, 8 trees



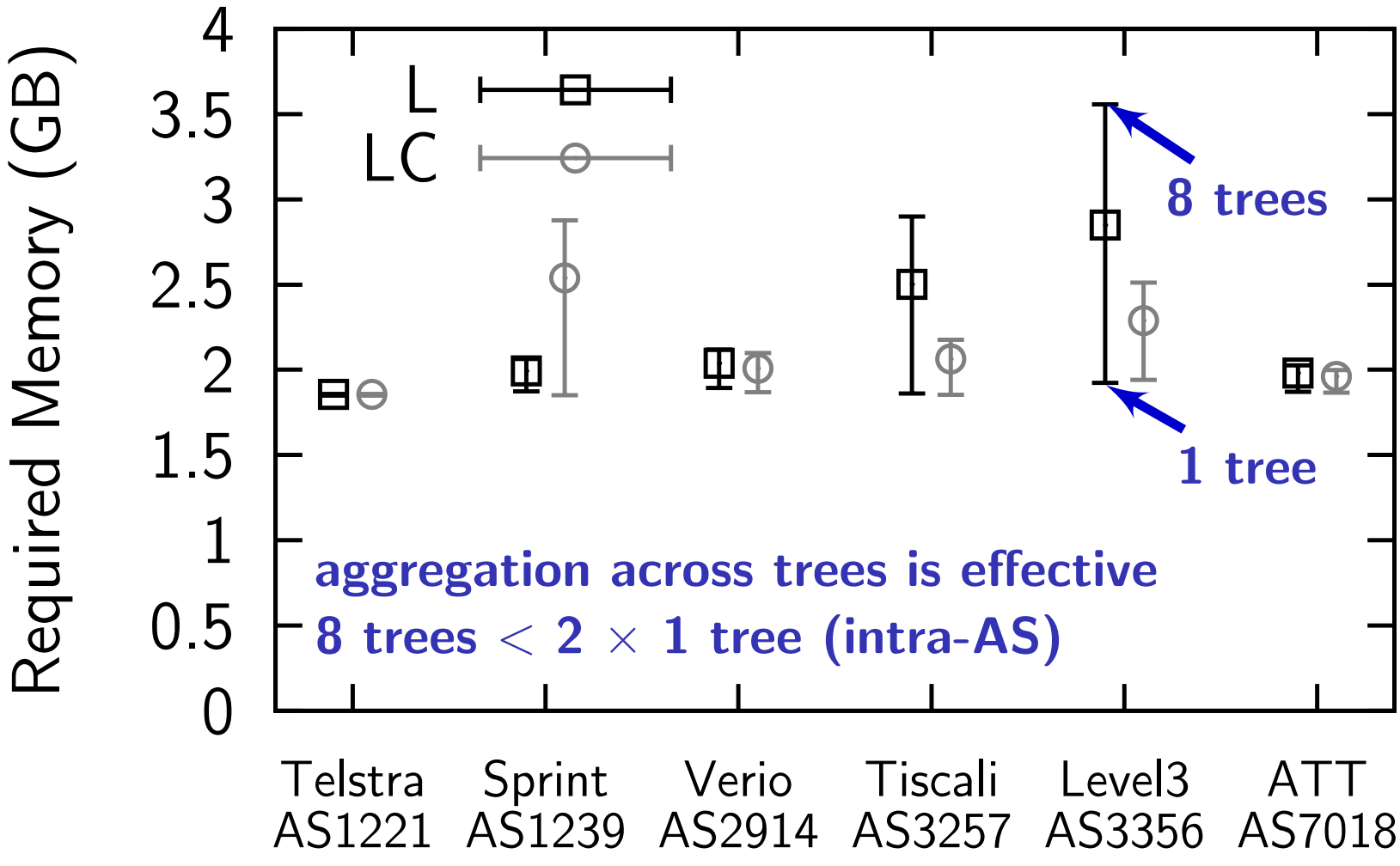
Gateway RIB Sizes

50M users, 8 trees



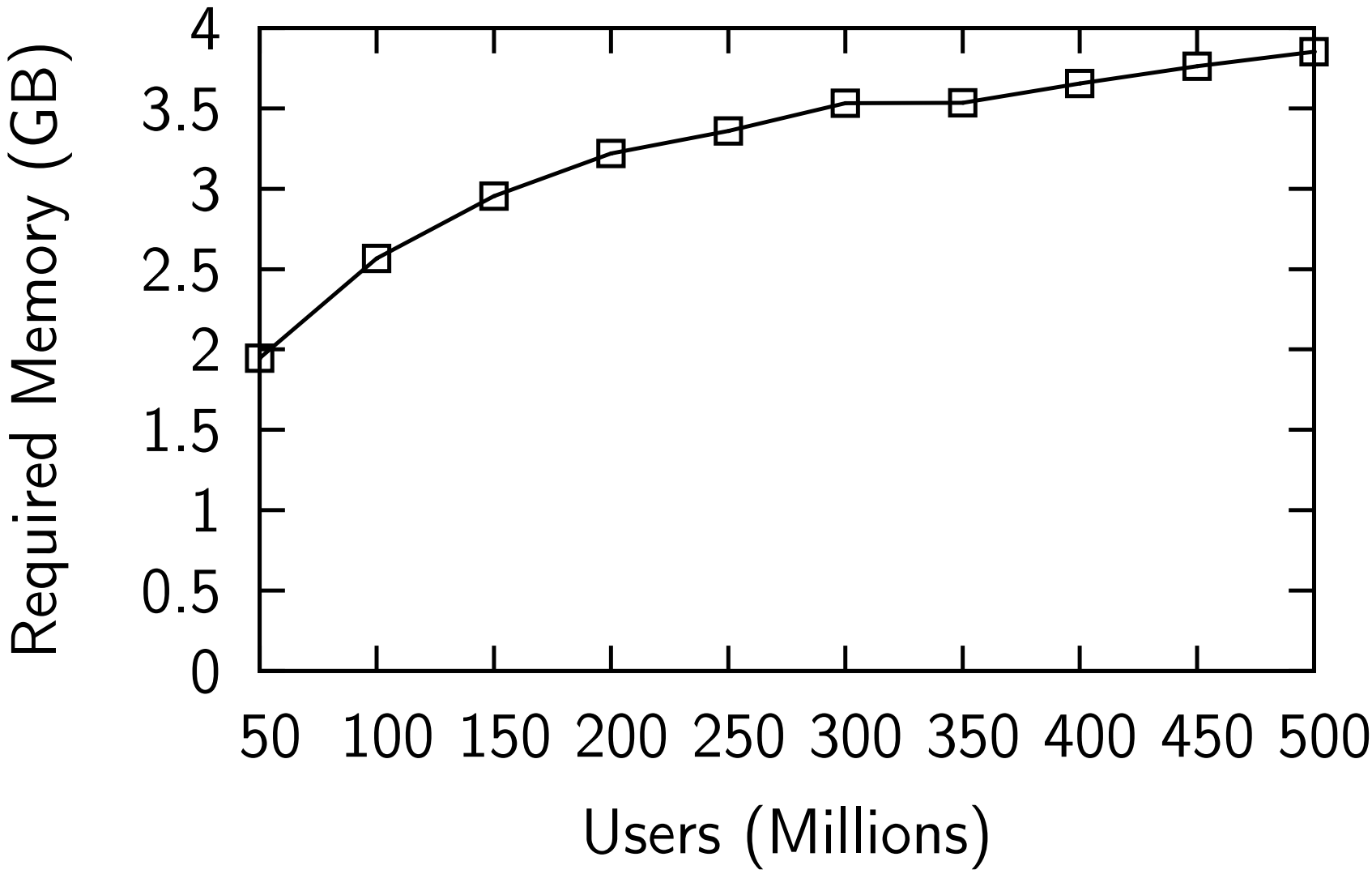
Gateway RIB Sizes

50M users, 8 trees



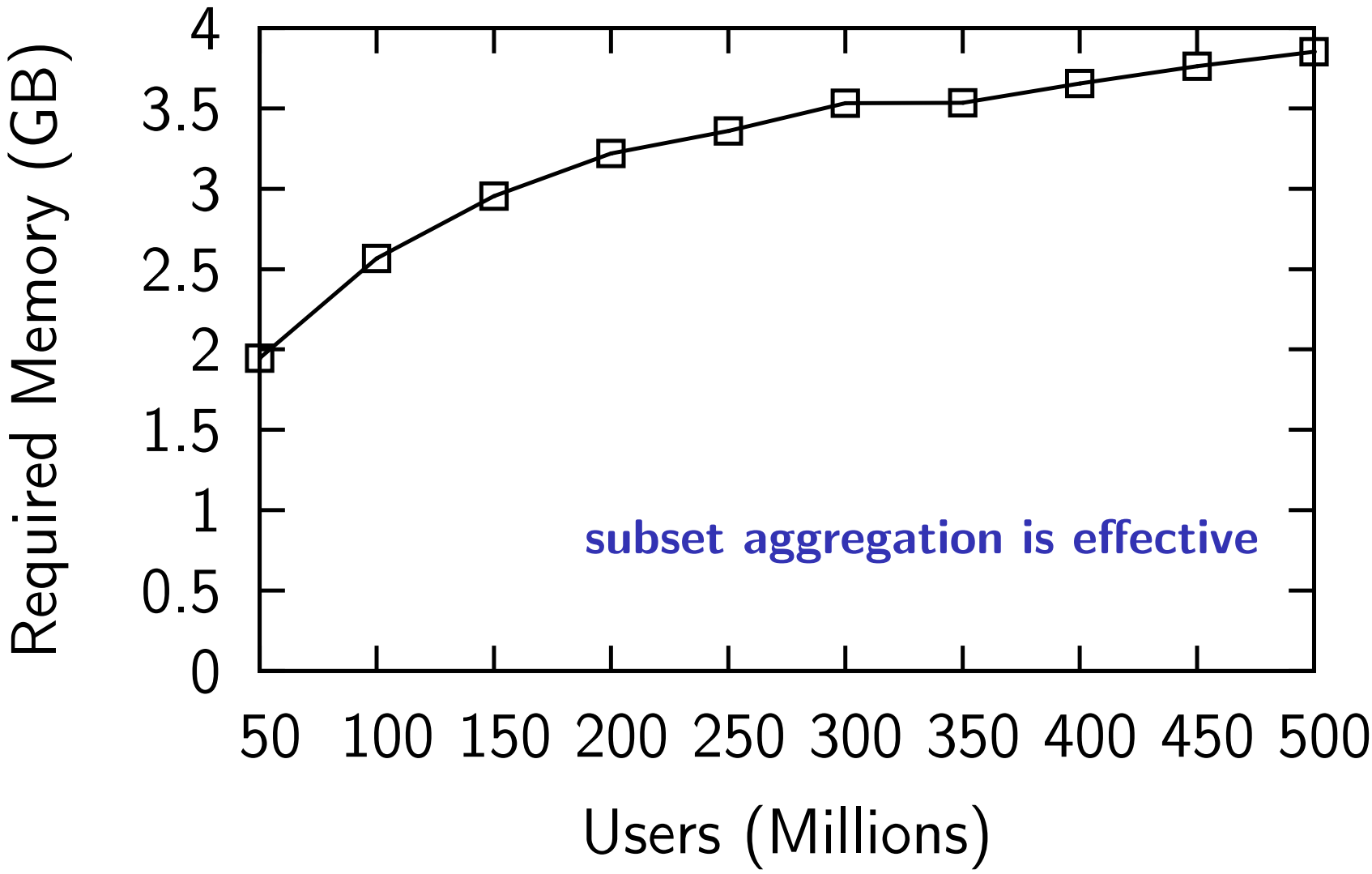
Single AS

50–500M users, 1 tree



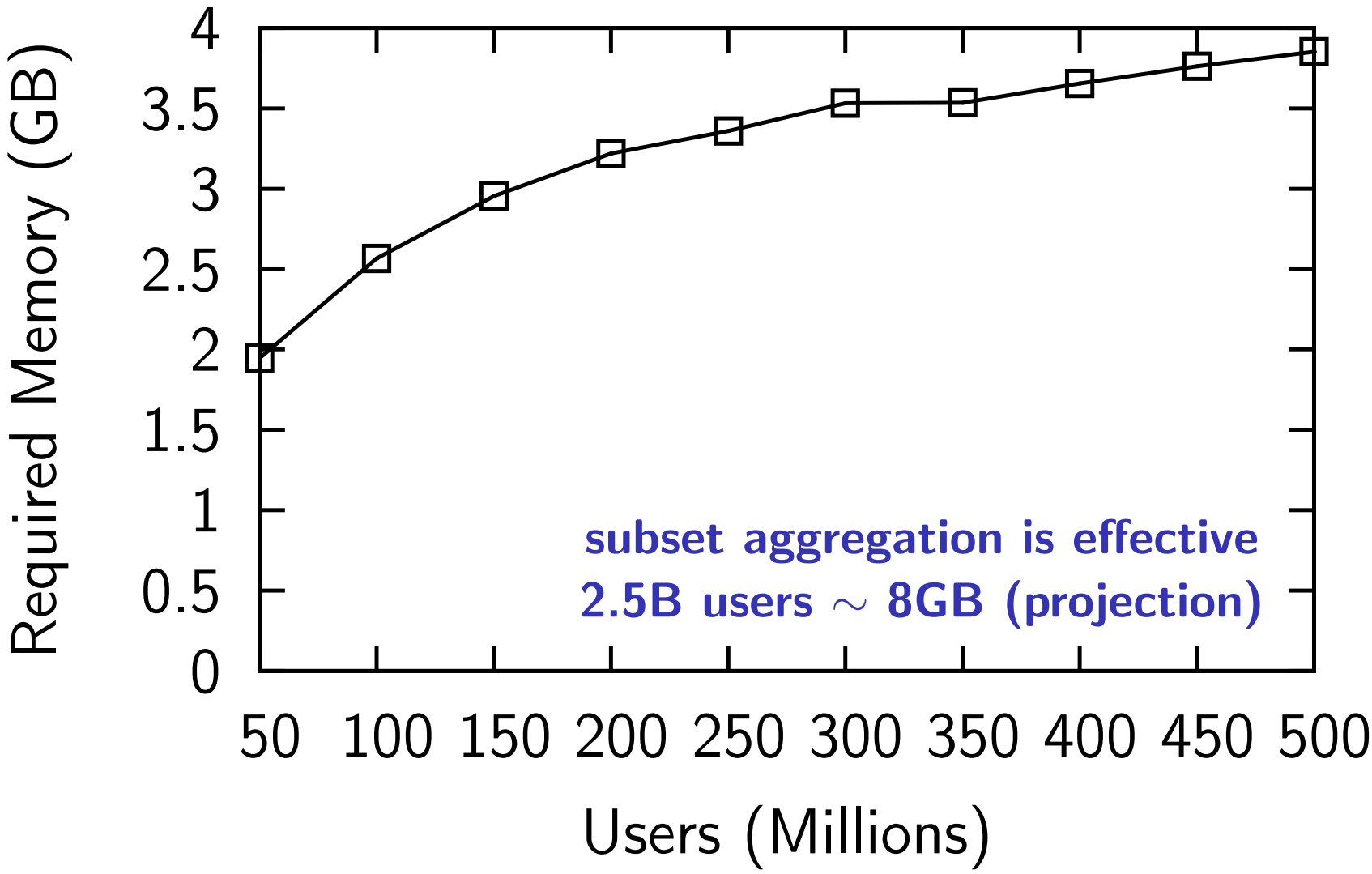
Single AS

50–500M users, 1 tree



Single AS

50–500M users, 1 tree



Conclusion and Future Work

Conclusion and Future Work

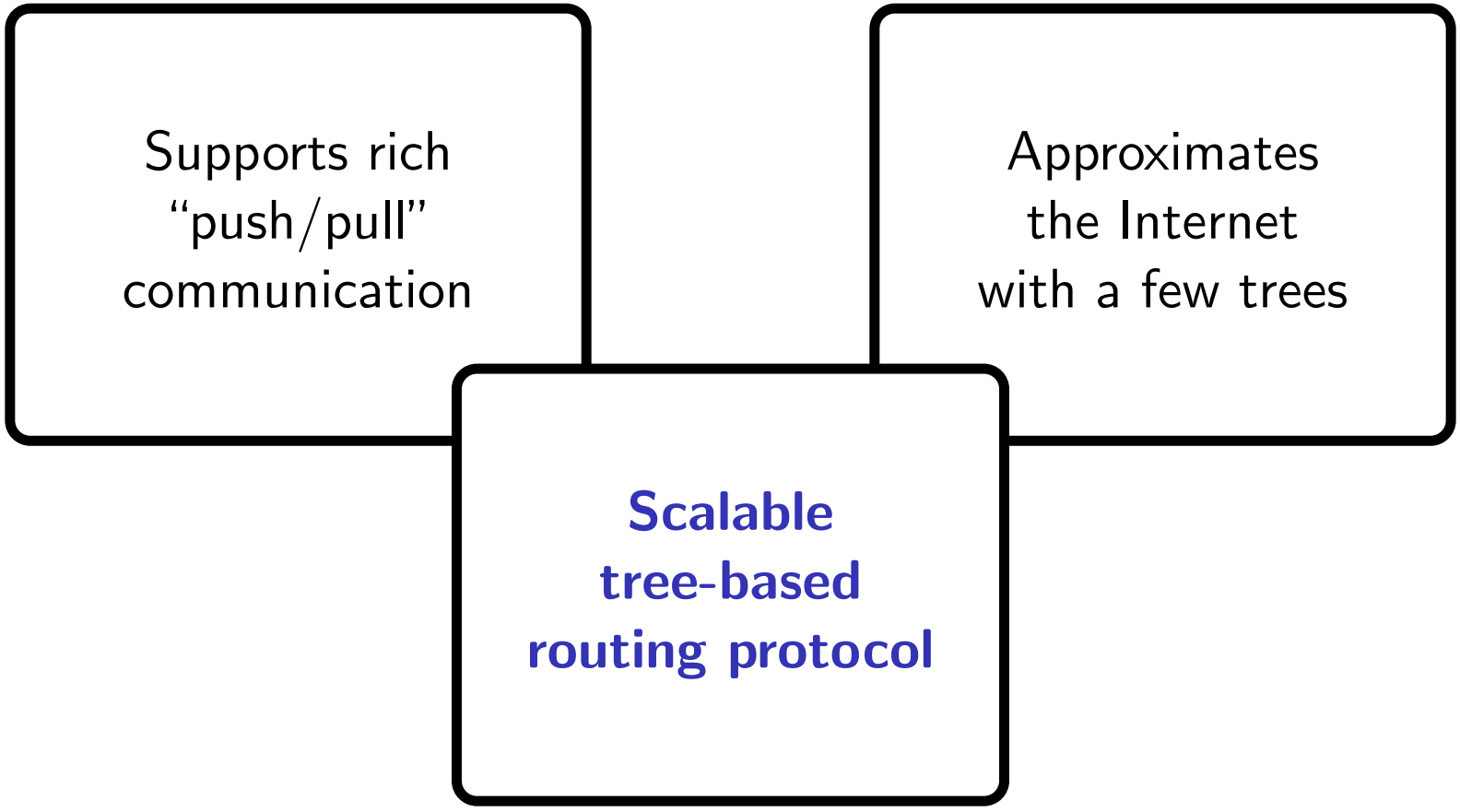
Scalable
tree-based
routing protocol

Conclusion and Future Work

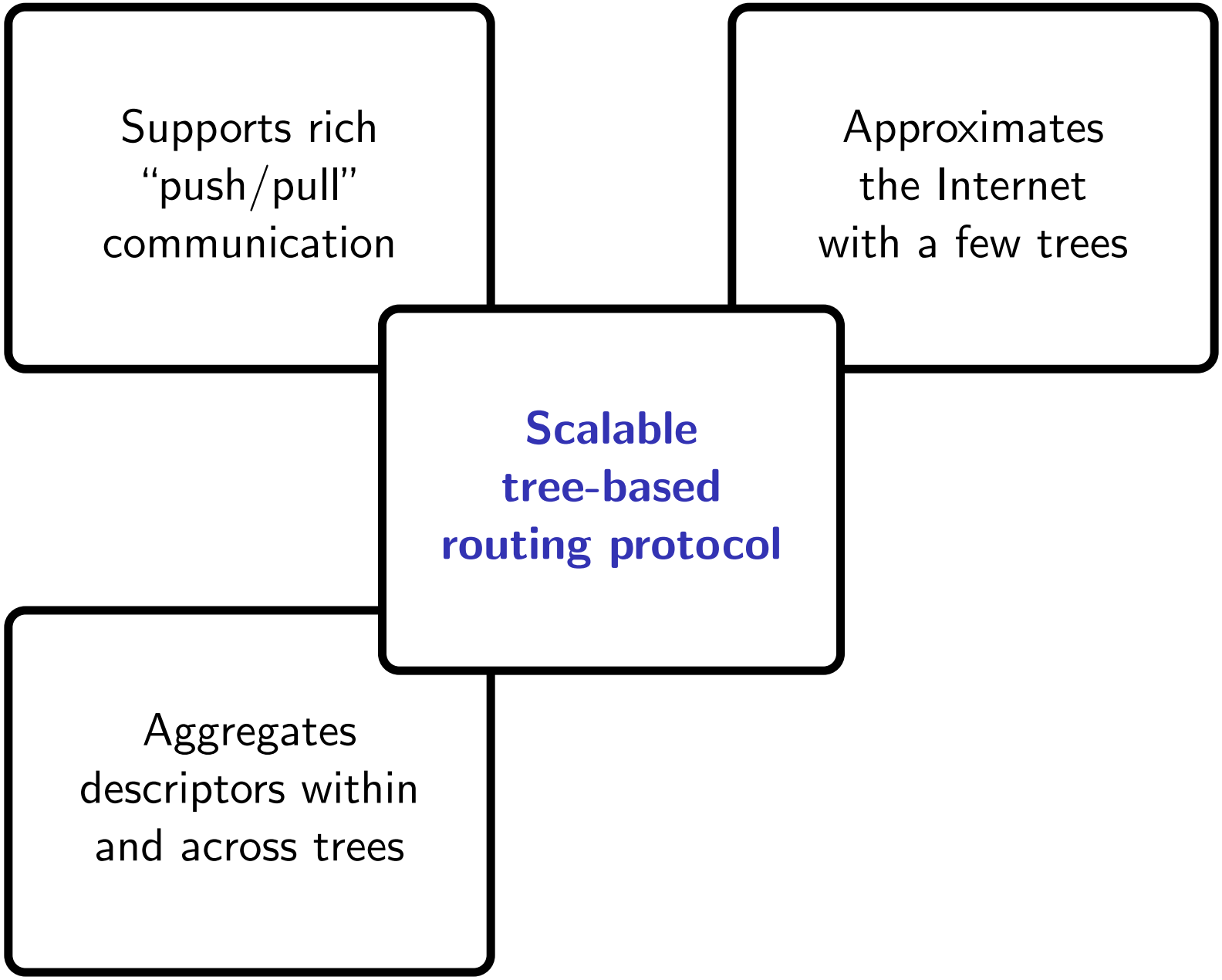
Supports rich "push/pull" communication

Scalable tree-based routing protocol

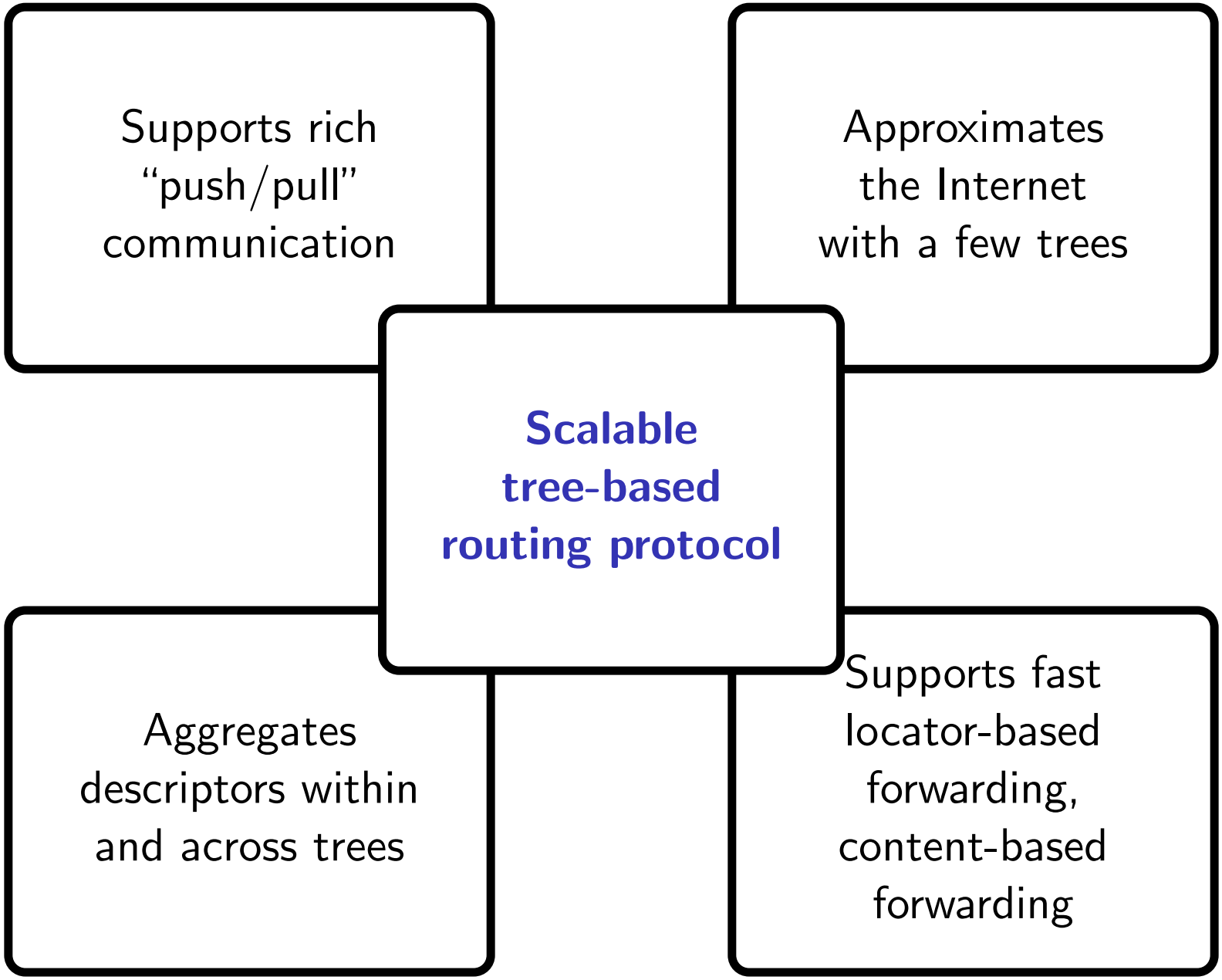
Conclusion and Future Work



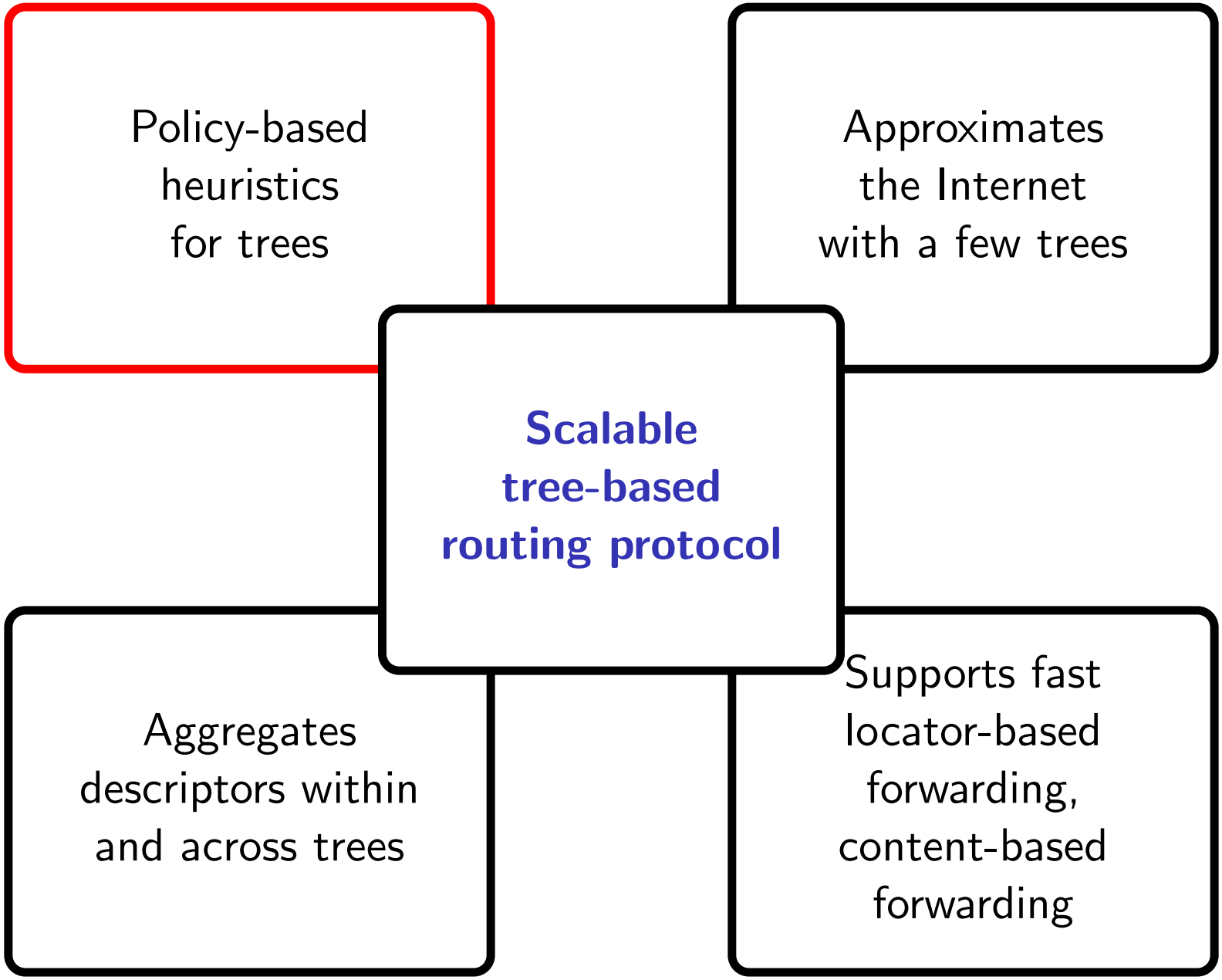
Conclusion and Future Work



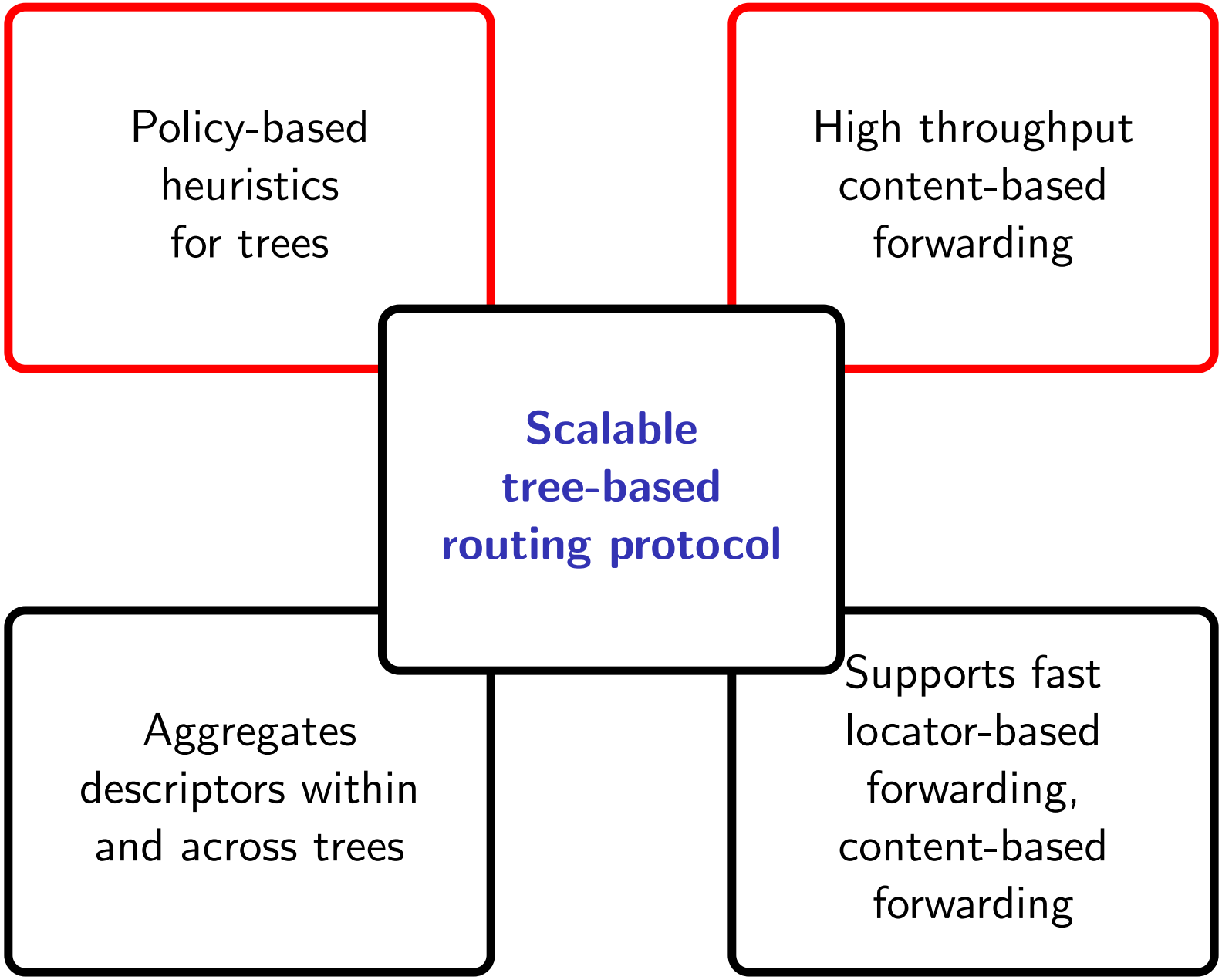
Conclusion and Future Work



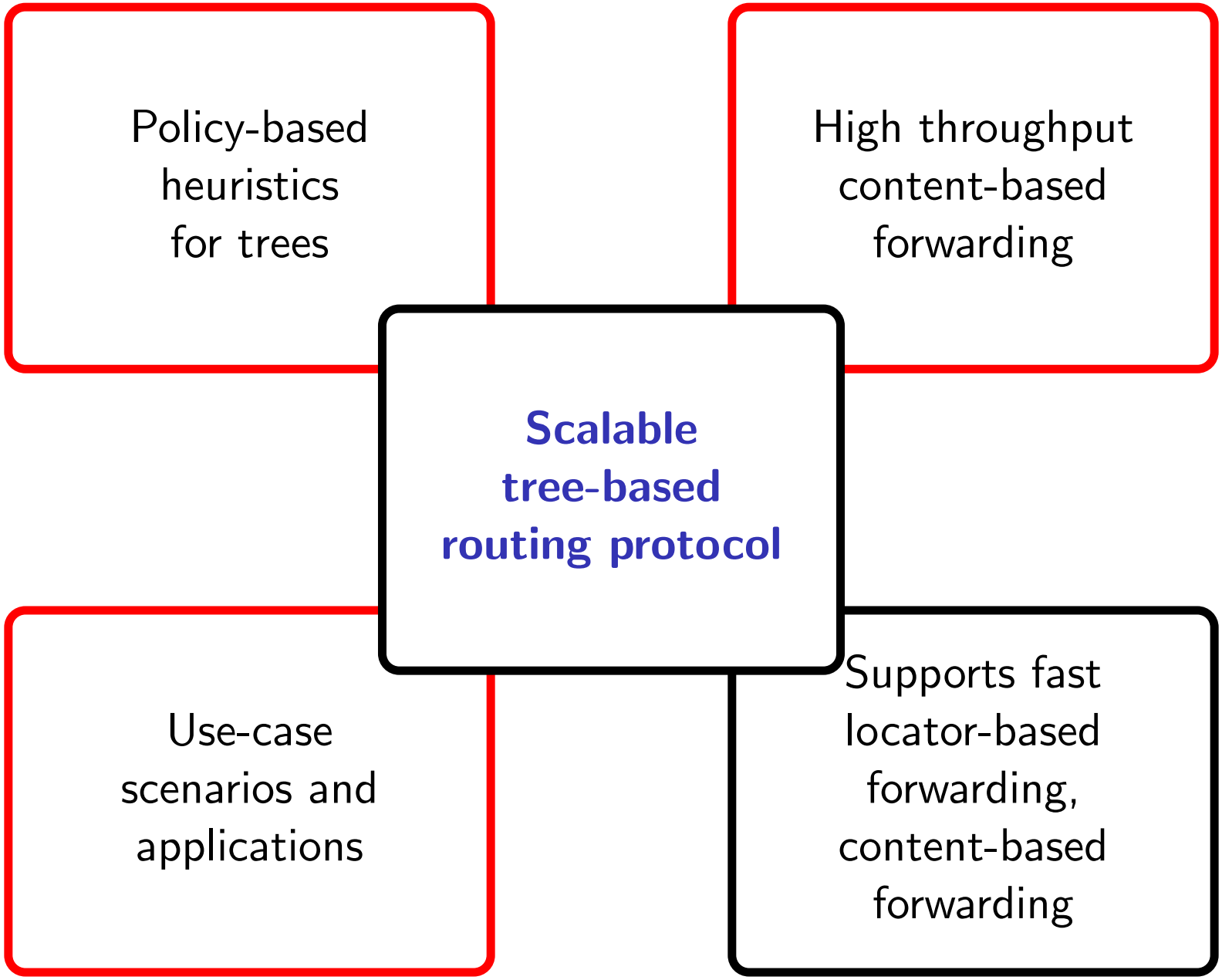
Conclusion and Future Work



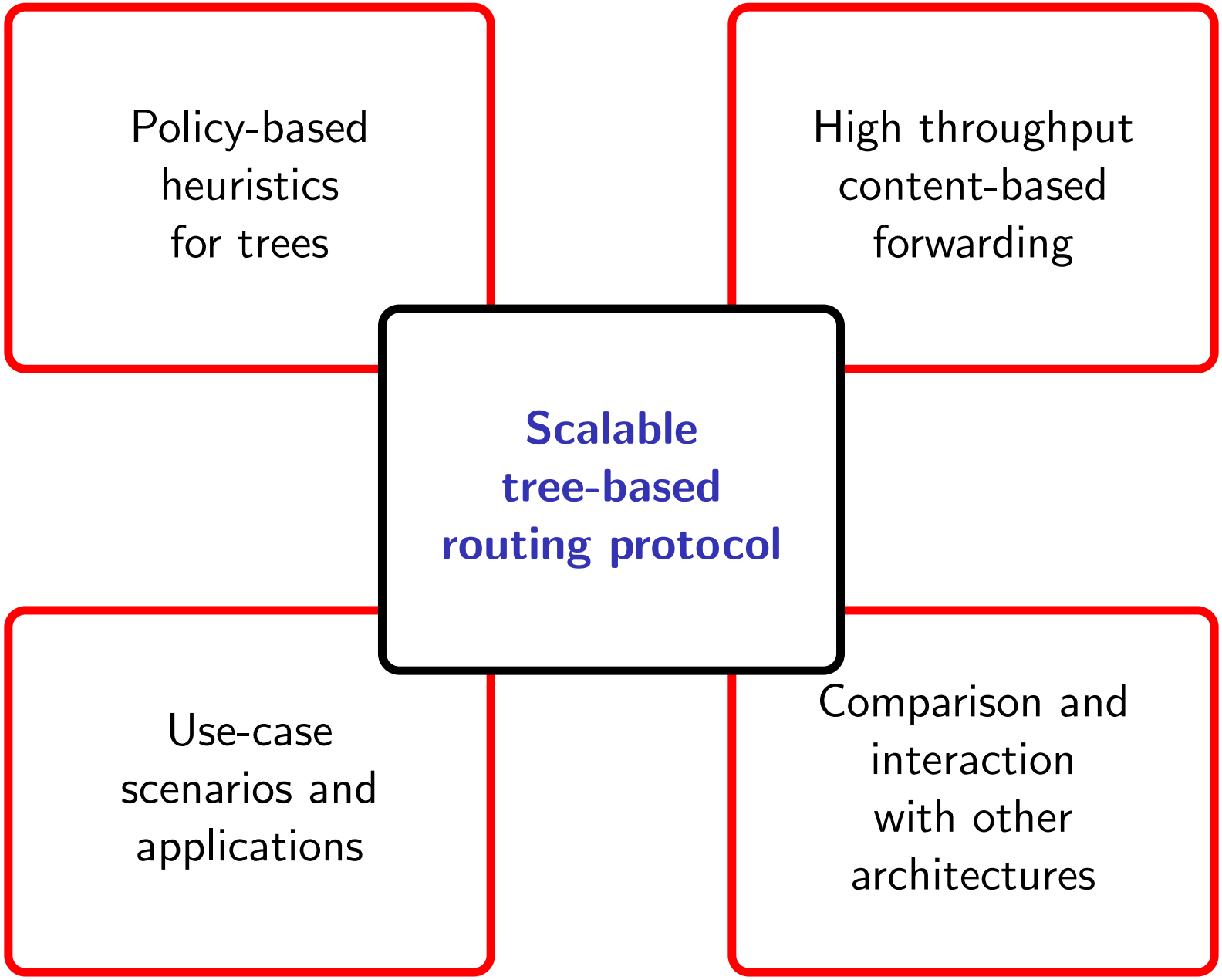
Conclusion and Future Work



Conclusion and Future Work



Conclusion and Future Work



Scalable Routing for Tag-Based Information-Centric Networking

ICN 2014

code available at:
http://www.inf.usi.ch/carzaniga/maketrees.git