Israel Networking Day 2014
Infocom 2014

Traffic Engineering with Equal-Cost-MultiPath: An Algorithmic Perspective

Marco Chiesa

joint work with

Guy Kindler

Michael Schapira

traffic-engineering (TE)

network operators' goal:

- provide best possible service
- minimize costs

how?

- fully exploit network resources

  → route flows of traffic along the “best” paths

traditional TE tools

ECMP (Equal-Cost-MultiPath)

- the most widely deployed TE mechanism
- load-balancing tool
- very simple mechanism

contributions

ECMP and arbitrary topologies

→ no reasonable approximation is possible ☹

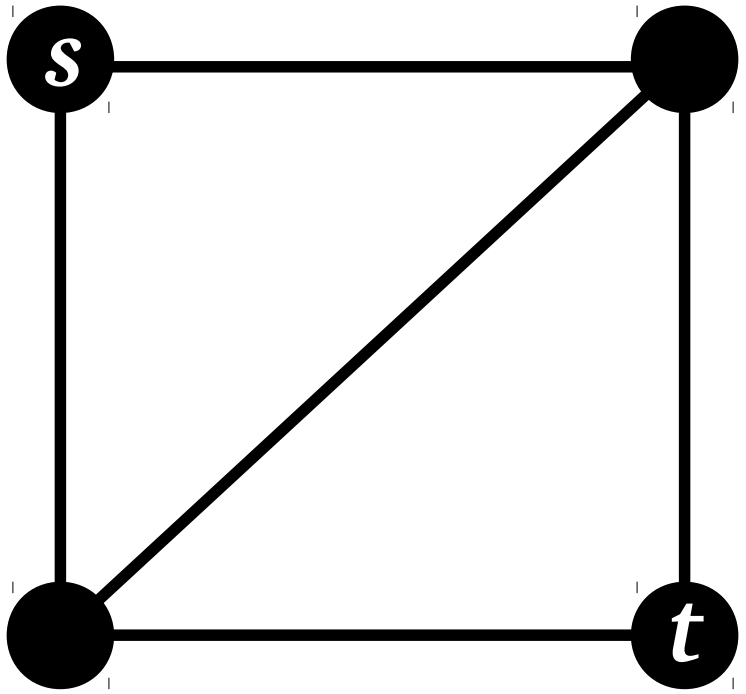
ECMP and datacenter topologies:

- hypercubes vs folded Clos networks ☺
- large flows in folded Clos networks ☺

traditional TE tools

ECMP (Equal-Cost-MultiPath)

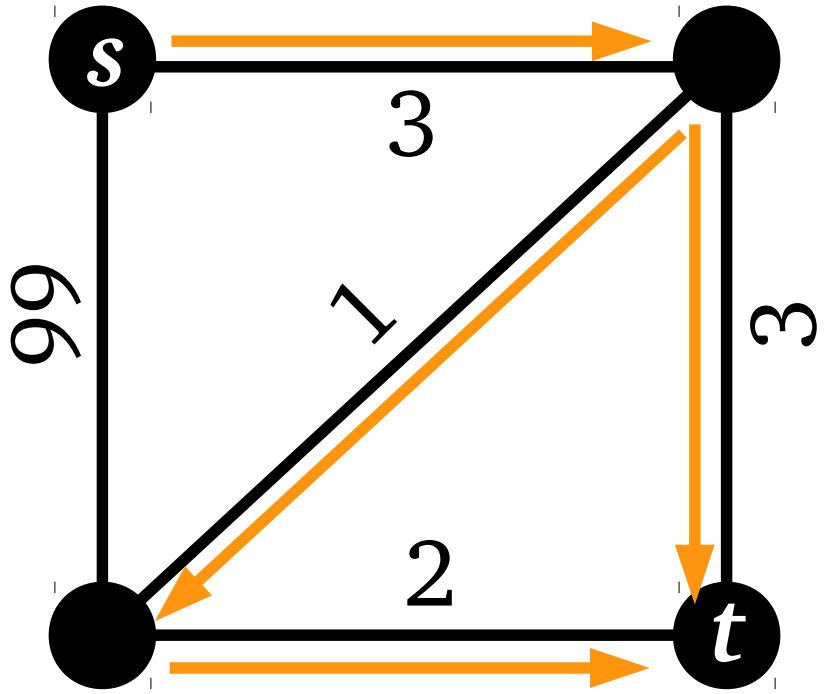
- operators set link weights
- traffic is routed along shortest-paths



traditional TE tools

ECMP (Equal-Cost-MultiPath)

- operators set link weights
- traffic is routed along shortest-paths

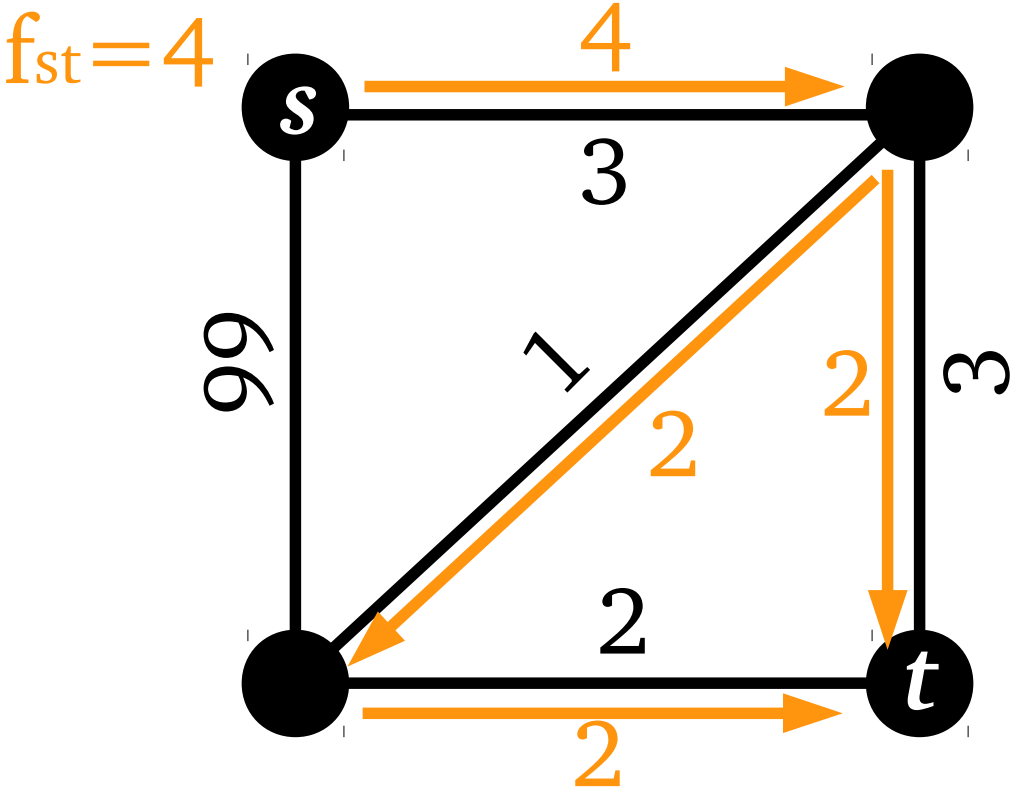


traditional TE tools

when multiple shortest paths are available:

- per-packet level equal split
- per-flow level hash-based split

  → equal split for many small flows

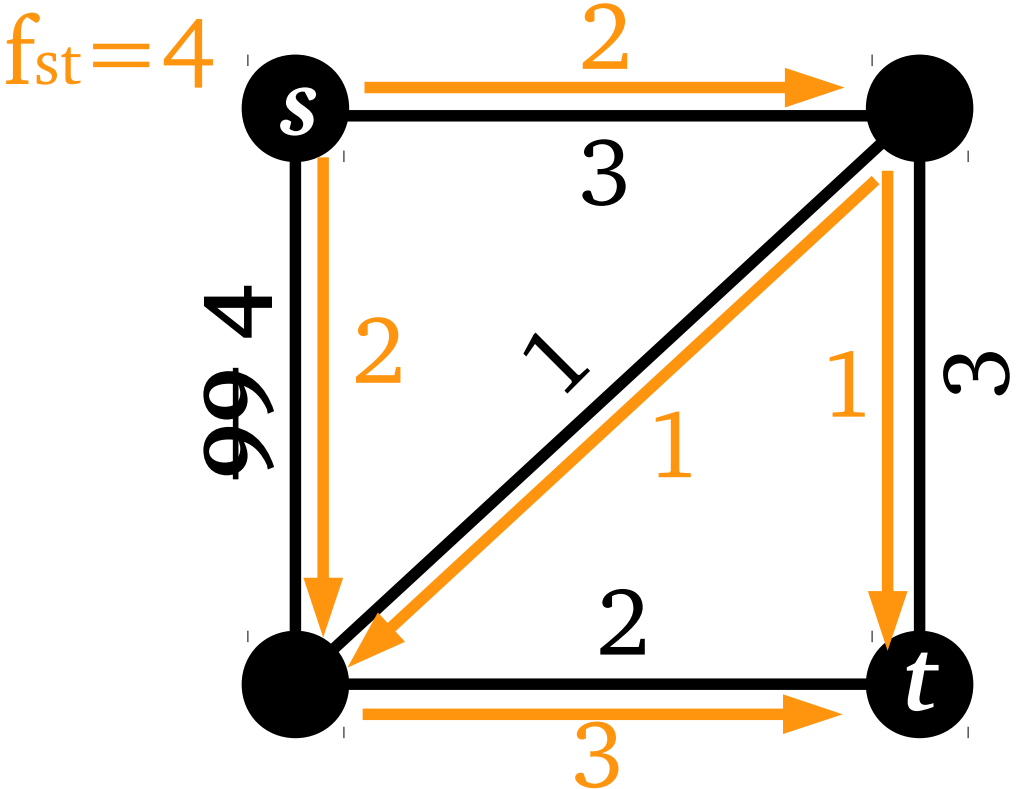


traditional TE tools

when multiple shortest paths are available:

- per-packet level equal split
- per-flow level hash-based split

  → equal split for many small flows

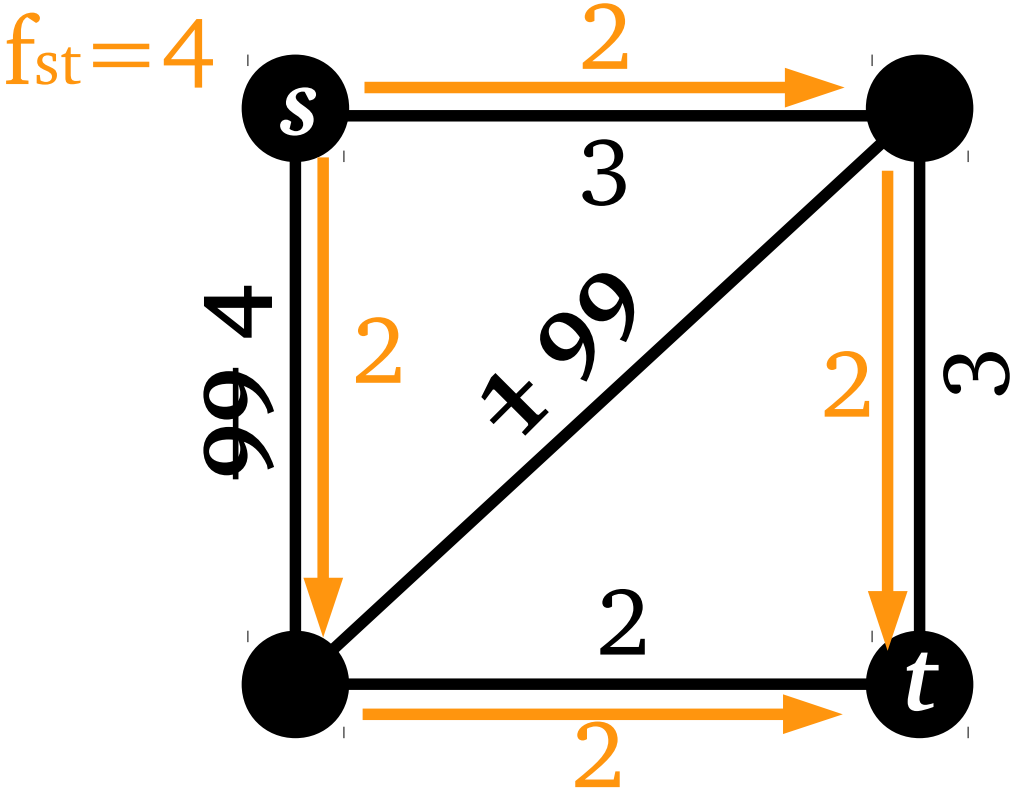


traditional TE tools

when multiple shortest paths are available:

- per-packet level equal split
- per-flow level hash-based split

  → equal split for many small flows



traditional TE tools

OSPF + ECMP (Equal-Cost-MultiPath)

- ***operators set link weights***

how? heuristic approaches:

- local search [Fortz,Thorup, 2000][Sundaresan et al, 2010]
- memetic algorithms [Buriol et al, 2002]
- genetic algorithms [Ericsson et al, 2002]
- branch-and-cut for mixed-ILP [Palmar et al, 2006]

wanted: algorithm with *provable* guarantees?

TE model: multi commidity flow

input:

- capacitated graph $G=(V,E)$
- demand matrix $D=\{d_{ij}\}$

constraints:

- flows cannot exceed links capacities
- flows are ***equally*** split among all shortest-paths

optimization functions:

- maximize total throughput (flow)
- minimize congestion
- minimize sum of link-costs

known results: inapproximability for max-flow

Theorem [FortzThorup,2000]. Given an instance $I=(G,D)$, it is NP-hard to distinguish whether:

$$OPT(I) = 1 \text{ or } OPT(I) = \frac{2}{3}$$

→ **no efficient algorithm can provably route at least a fraction $\frac{2}{3}+e$ of *OPT(I)***

our first results: inapproximability for max-flow

Theorem ~~[FortzThorup,2000]~~. Given an instance $I=(G,D)$, with a single entry in D, it is NP-hard to distinguish whether:

$$OPT(I) = 1 \text{ or } OPT(I) = \frac{2}{3}$$

our first results: no inapproximability for max-flow

Theorem ~~[FortzThorup,2000]~~. Given an instance $I=(G,D)$, with a single entry in D, it is NP-hard to distinguish whether:

$$OPT(I) = 1 \text{ or } OPT(I) = \not{\frac{2}{3}}\ q,$$

for any $q>0$

our first results: any constant inapproximability for max-flow

Theorem ~~[FortzThorup,2000]~~. Given an instance $I=(G,D)$, with a single entry in D, it is NP-hard to distinguish whether:

$$OPT(I) = 1 \text{ or } OPT(I) = \not{\frac{2}{3}}\ q,$$

for any $q>0$

→ no efficient algorithm can provably route at least a fraction q of $OPT(I)$

key tool:
amplification operator X

operator X: instance $I \rightarrow$ instance I_{new}

such that

$$OPT(I_{new}) = OPT(I)^{2}$$

amplifying the gap

$OPT(I) = 1$ *or* $OPT(I) = \frac{2}{3}$

it is NP-hard to distinguish between ***1*** *and* ***~0.6***

amplifying the gap

$OPT(I) = 1$ *or* $OPT(I) = \frac{2}{3}$

it is NP-hard to distinguish between ***1*** *and* ***~0.6***

$OPT(X(I)) = 1$ *or* $OPT(X(I)) = \frac{4}{9}$

it is NP-hard to distinguish between ***1*** *and* ***~0.4***

amplifying the gap

$OPT(I) = 1$ or $OPT(I) = \frac{2}{3}$

it is NP-hard to distinguish between ***1*** *and* ***~0.6***

$OPT(X(I)) = 1$ or $OPT(X(I)) = \frac{4}{9}$

it is NP-hard to distinguish between ***1*** *and* ***~0.4***

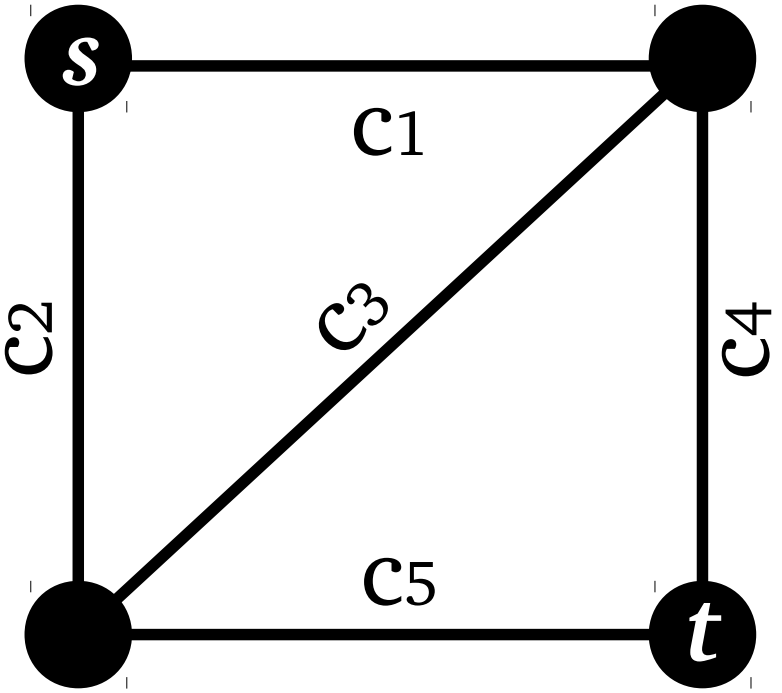
$OPT(X^2(I)) = 1$ or $OPT(X^2(I)) = \frac{16}{81}$

it is NP-hard to distinguish between ***1*** *and* ***~0.2***

…

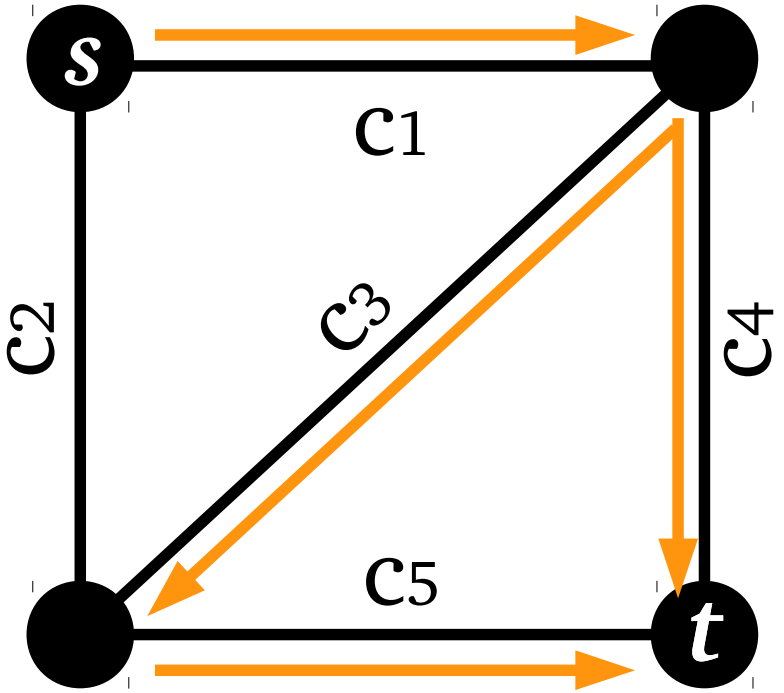
amplification gap technique:
graph G

- source `s`
- target `t`
- capacitated edges



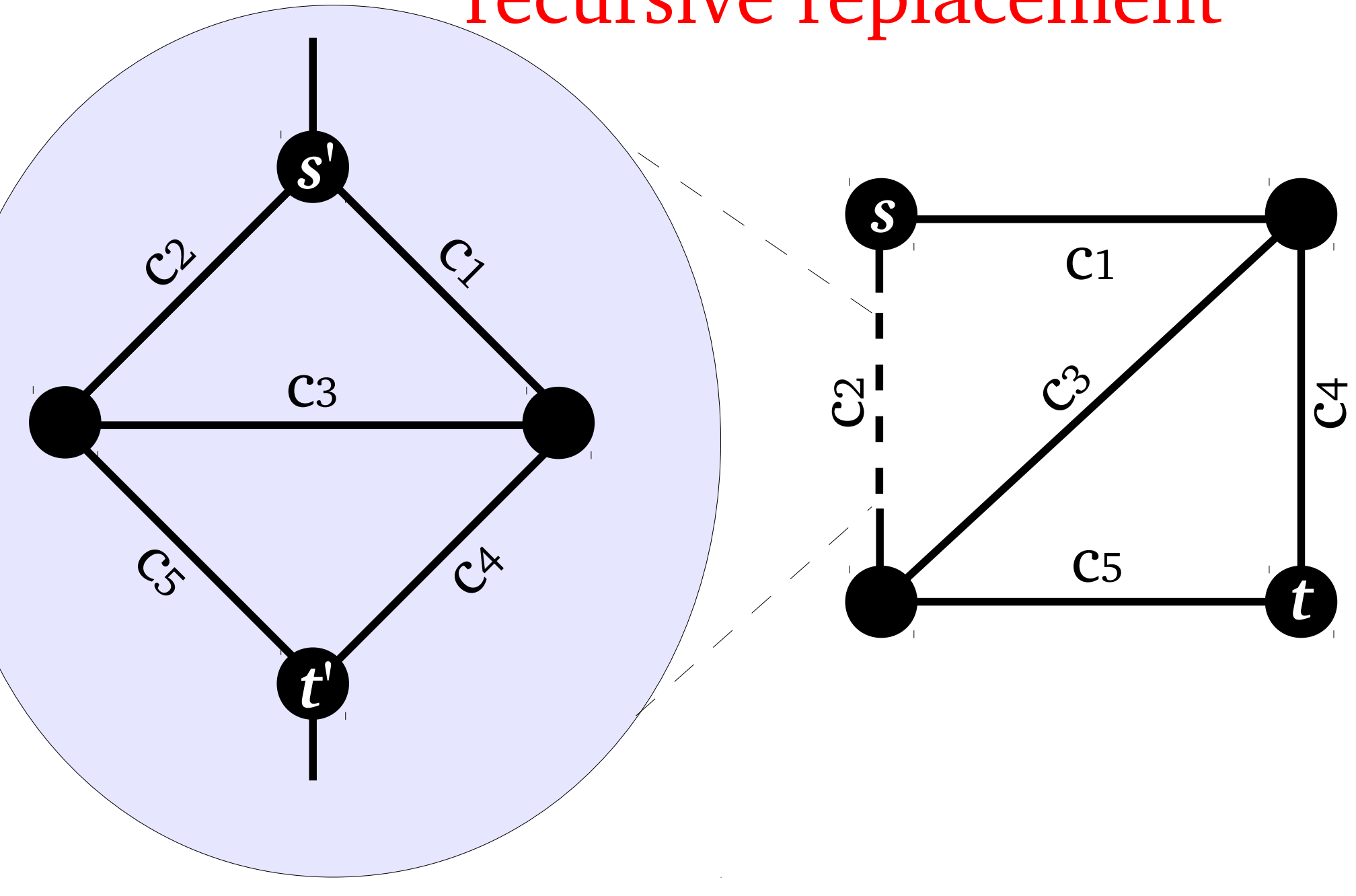
amplification gap technique:
graph G

- source `s`
- target `t`
- capacitated edges

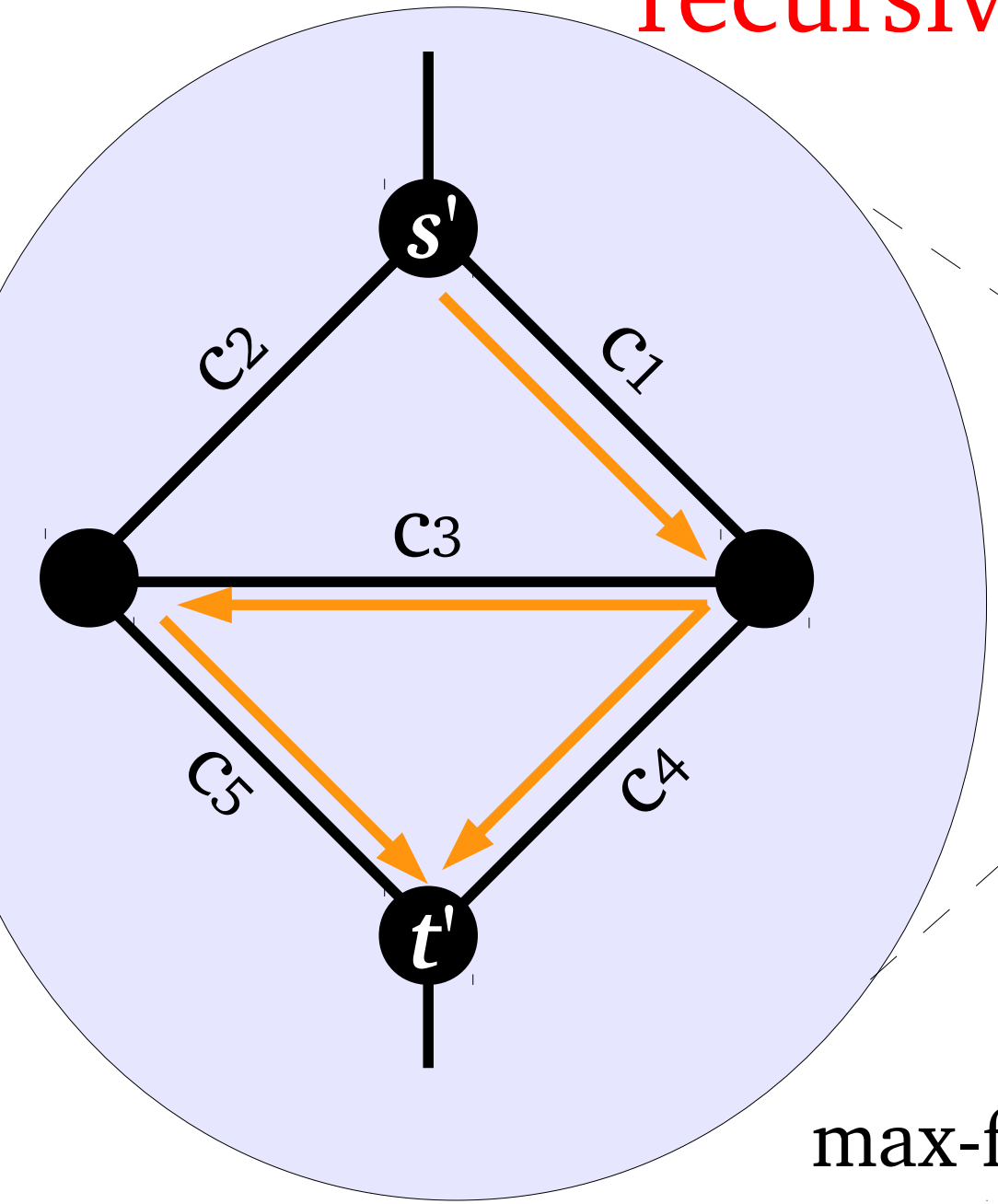


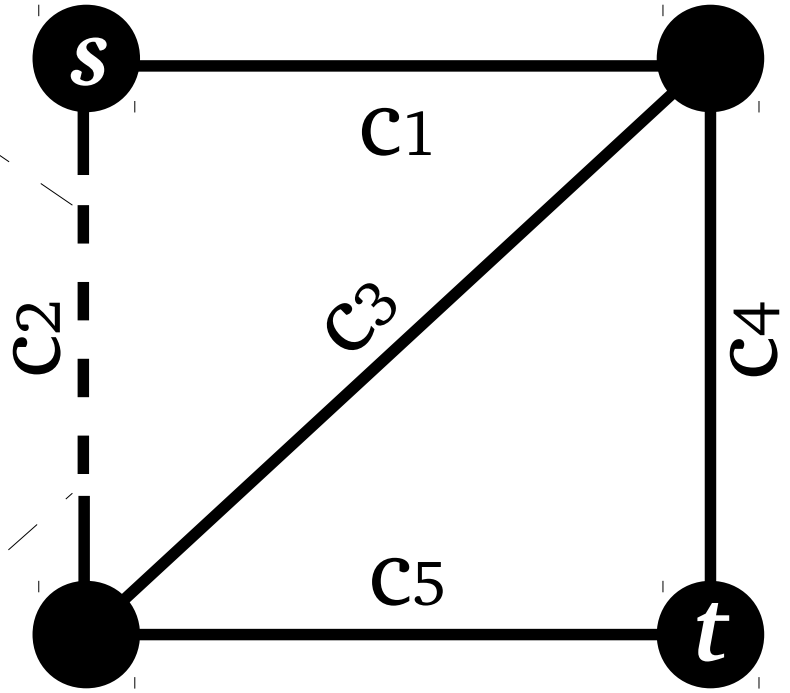
→ max-flow (s → t) = ***OPT***

amplification gap technique:
recursive replacement



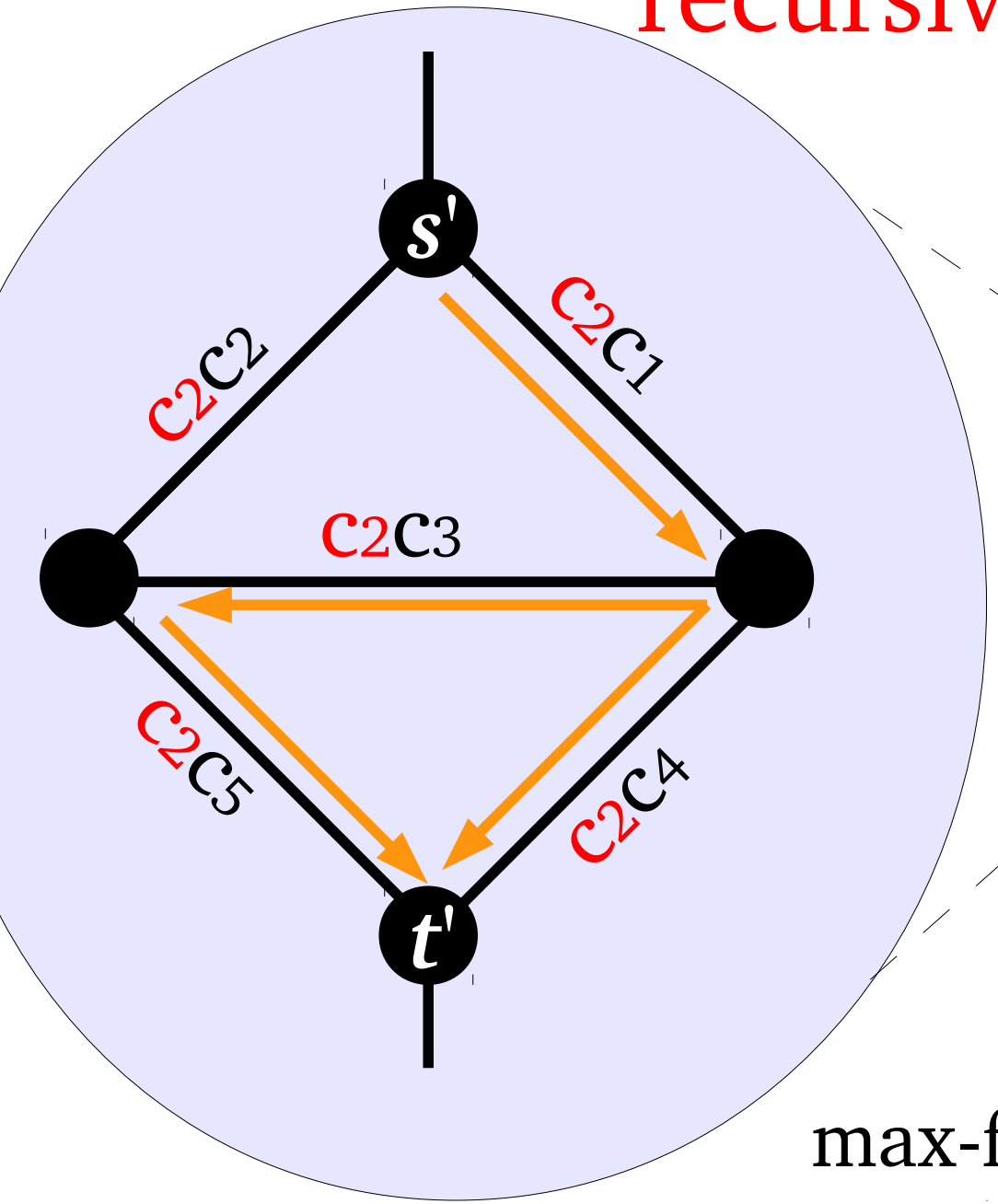
amplification gap technique:
recursive replacement

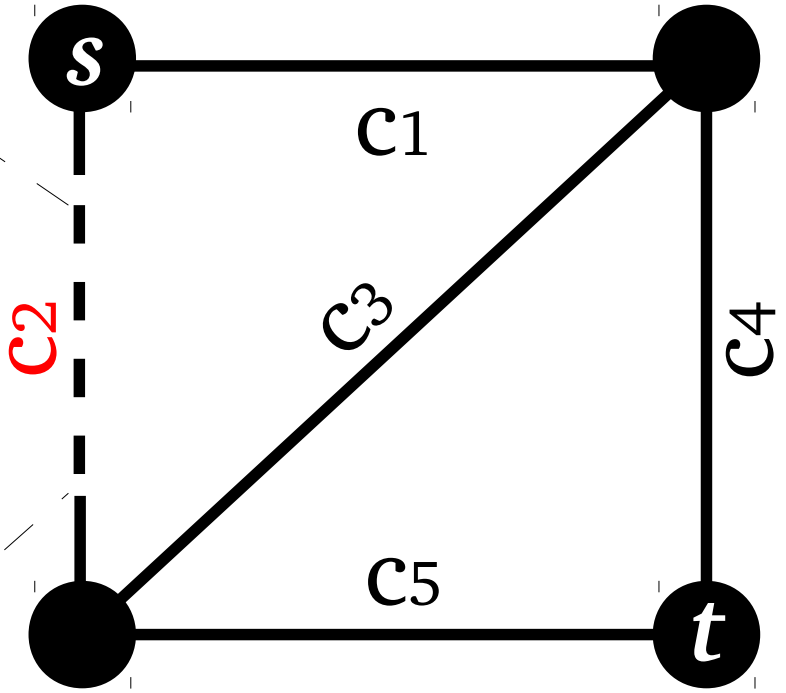




max-flow($s' \rightarrow t'$) = ***OPT***

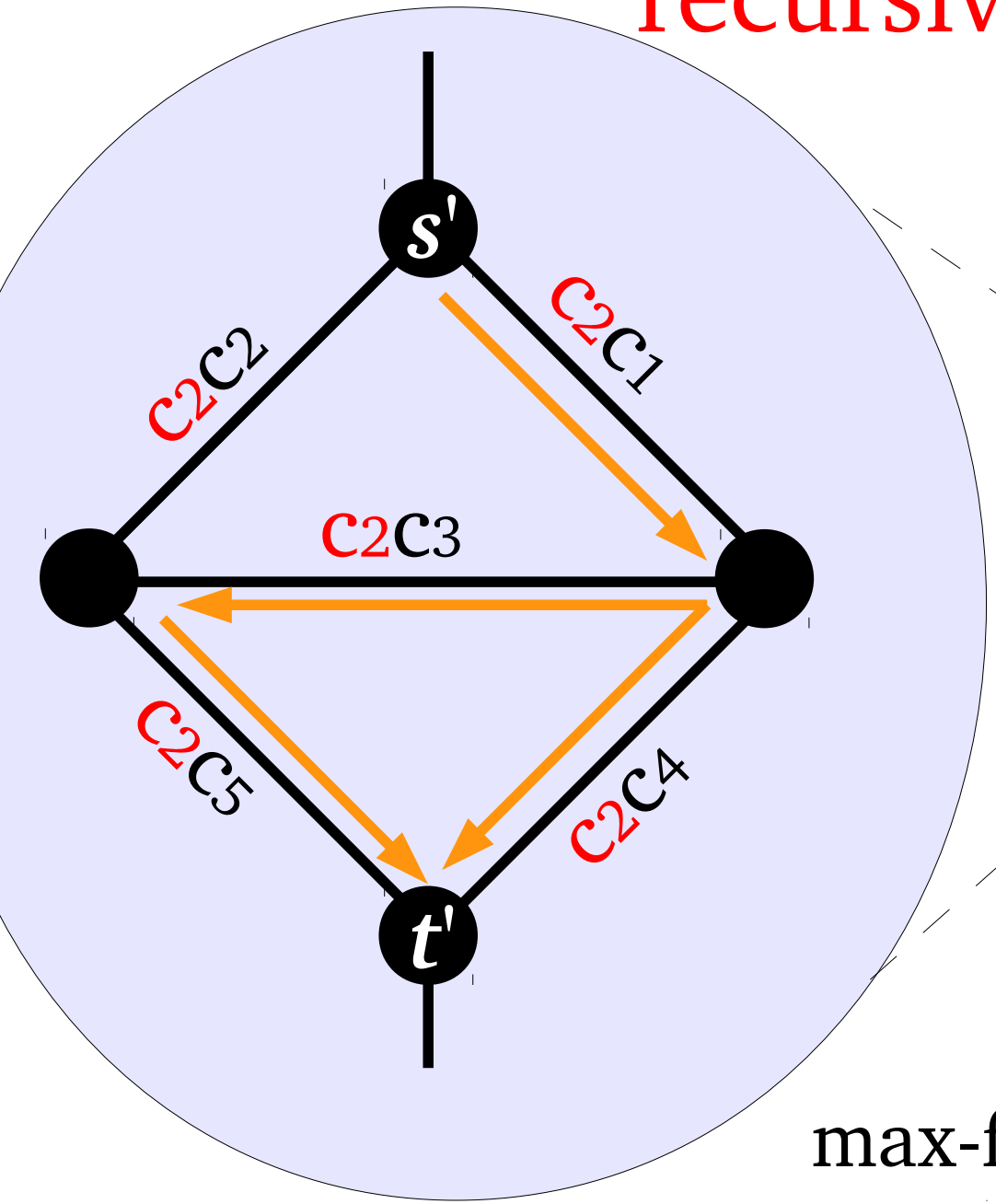
amplification gap technique:
recursive replacement

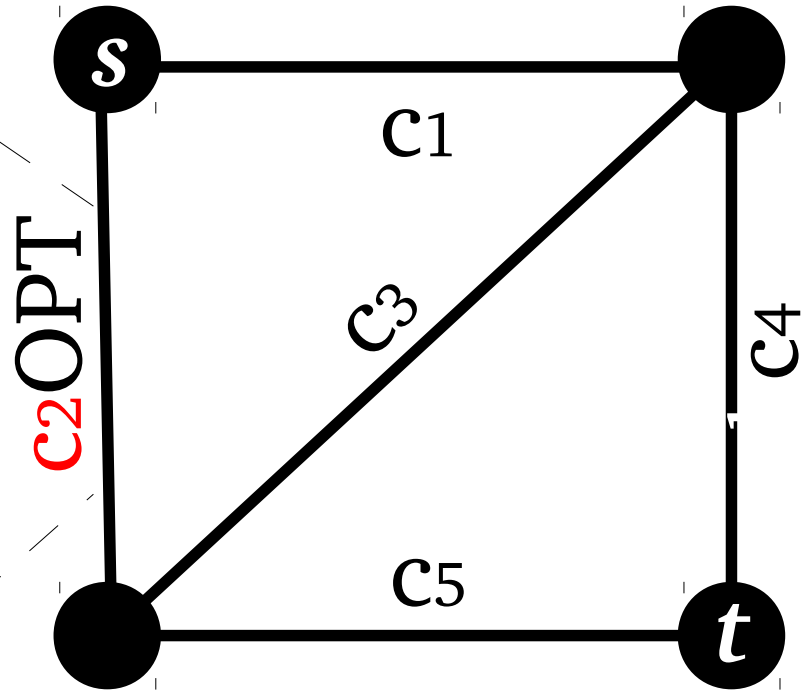




max-flow($s' \rightarrow t'$) = $c_2 OPT$

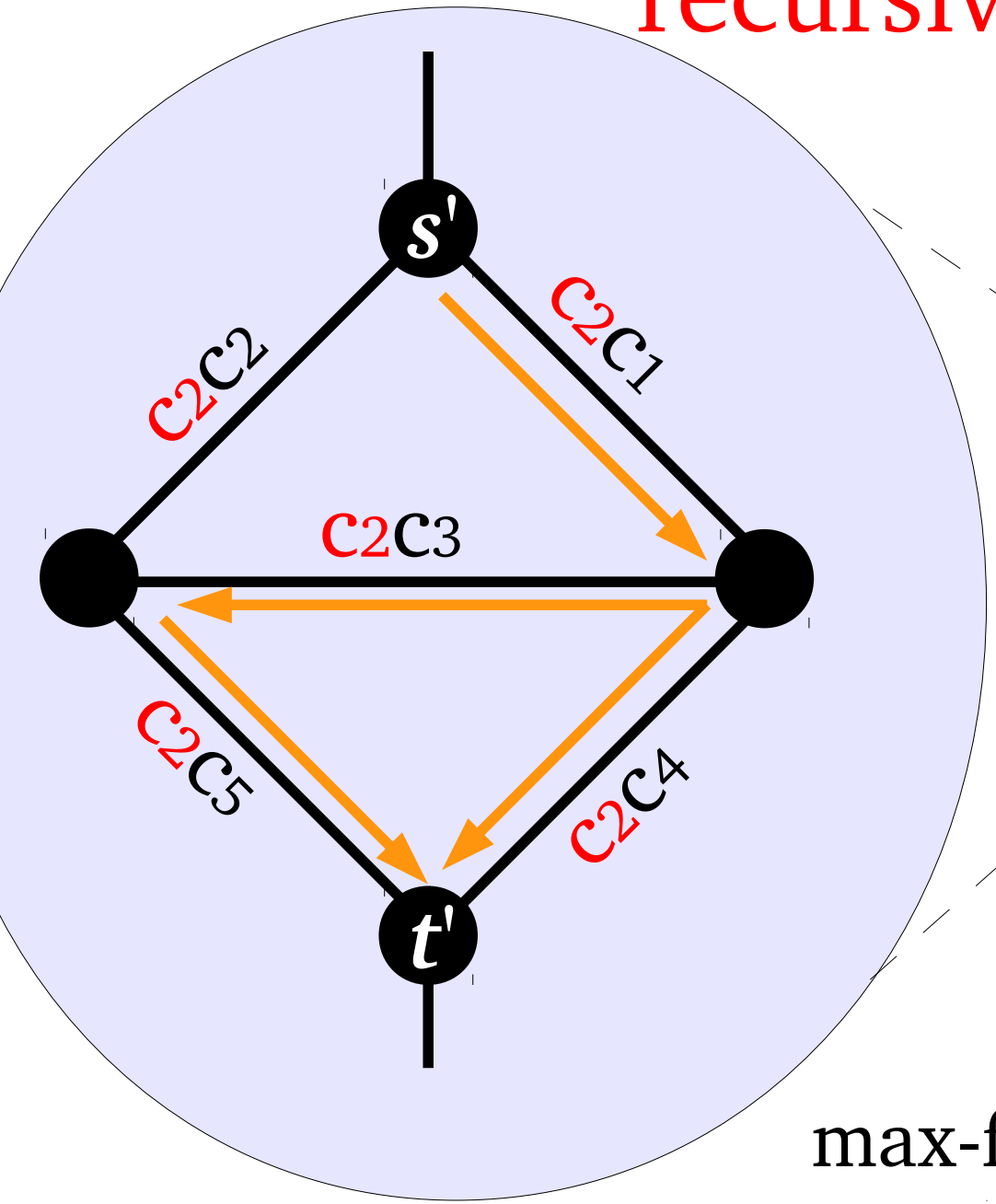
amplification gap technique:
recursive replacement

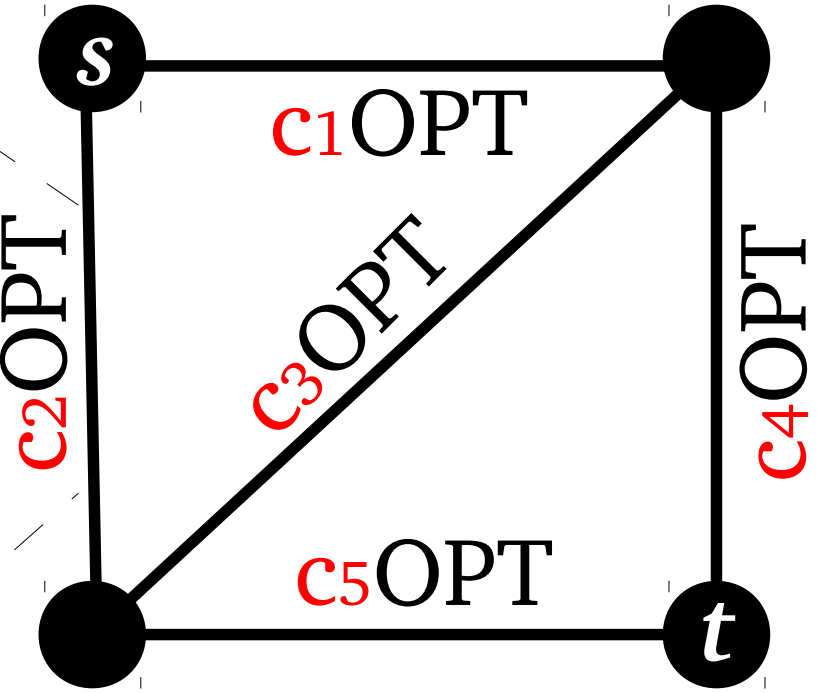




max-flow($s' \rightarrow t'$) = $c_2 OPT$

amplification gap technique:
recursive replacement

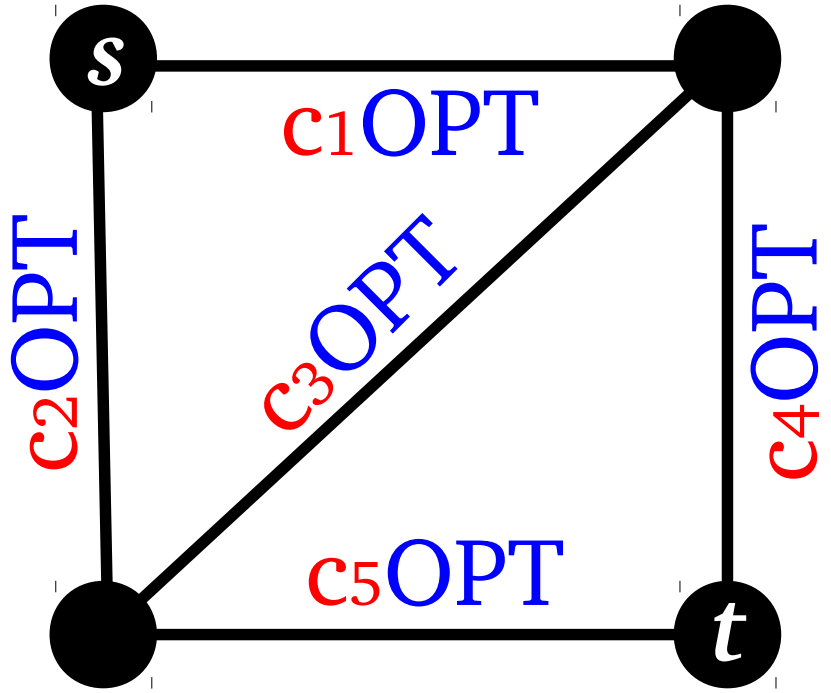




max-flow($s' \rightarrow t'$) = $c_2 OPT$

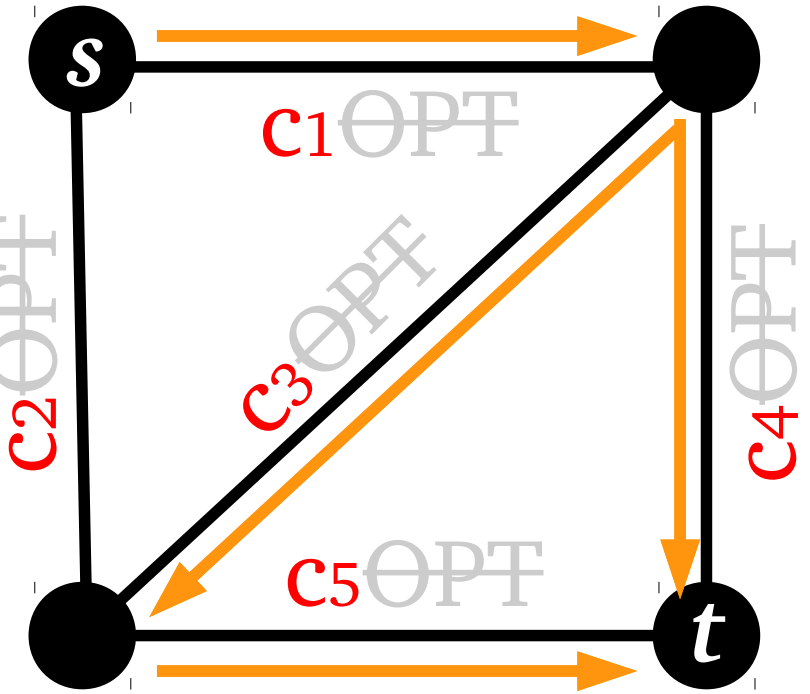
amplification gap technique: graph *X(G)*

max-flow(s → t,G') = OPT · ...



amplification gap technique: graph *X(G)*

max-flow(s → t,G') =
OPT · OPT = OPT ^{2}



data-center topologies and ECMP

topology constraints:

- d-dimensional hypercubes (e.g., bCube-like)

  → NP-hard
- l-layers folded Clos networks (e.g., VL2-like)

  → easy
- random regular graphs (e.g., Jellyfish)

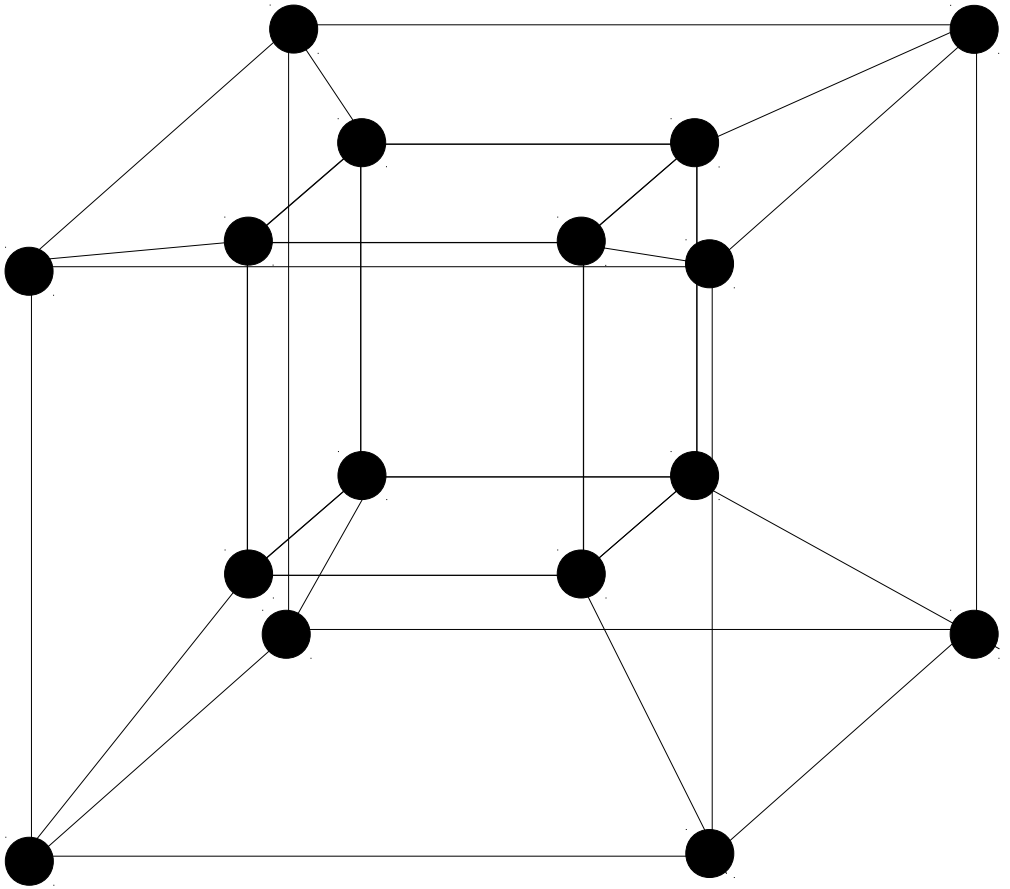
  → future work

data-center topologies and ECMP

topology constraints:

- *d*-dimensional hypercubes (e.g., bCube)

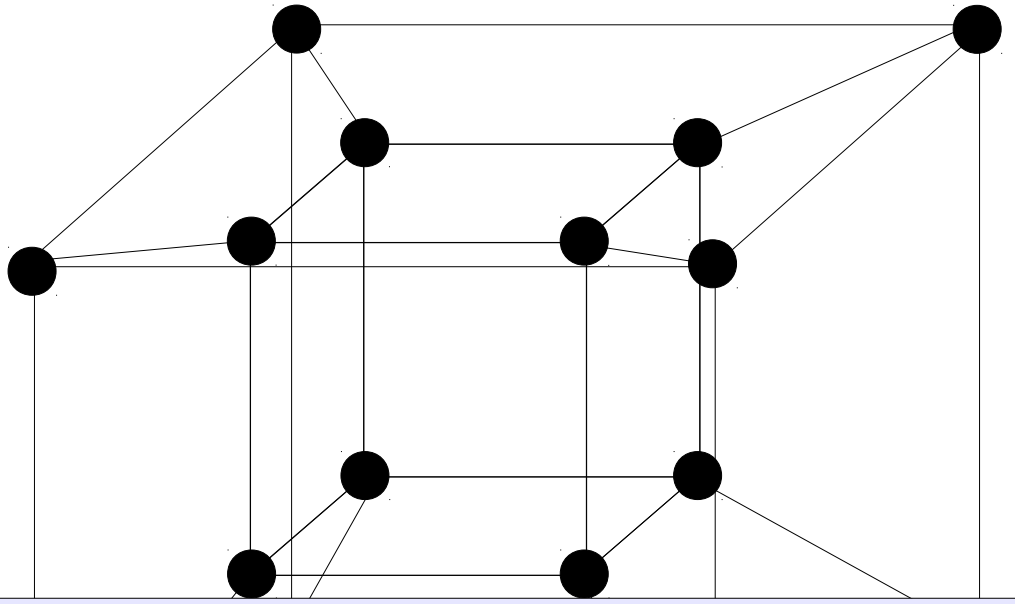
d=4



data-center topologies and ECMP

topology constraints:

- *d*-dimensional hypercubes (e.g., bCube)



d=4

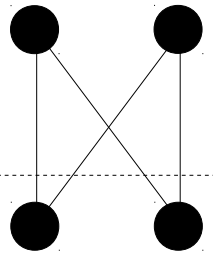
computing the best weight assignment is computationally intractable

data-center topologies and ECMP

topology constraints:

- *l*-layers folded Clos networks (e.g.,VL2)

l=2



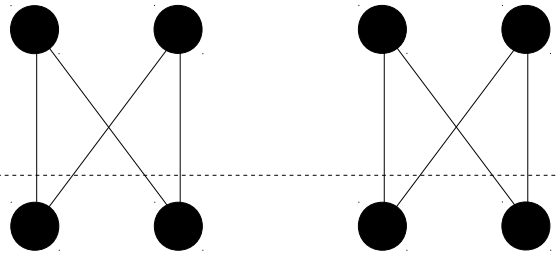
servers

data-center topologies and ECMP

topology constraints:

- *l*-layers folded Clos networks (e.g.,VL2)

l=3

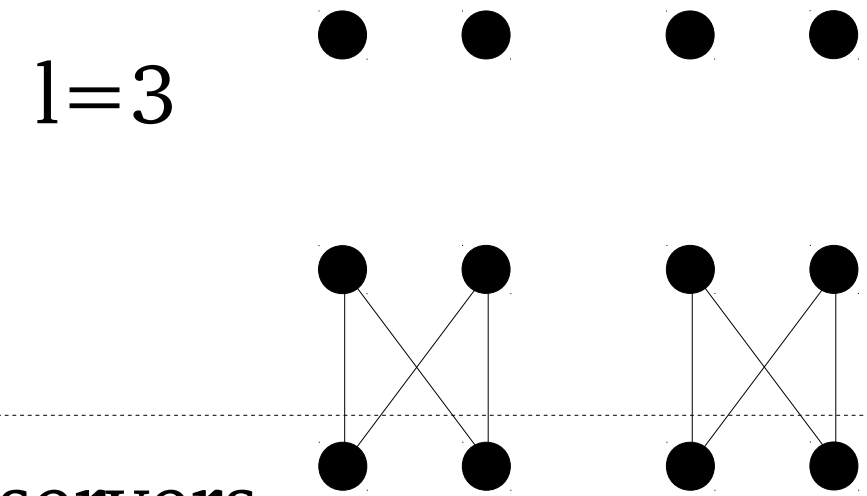


servers

data-center topologies and ECMP

topology constraints:

- *l*-layers folded Clos networks (e.g.,VL2)

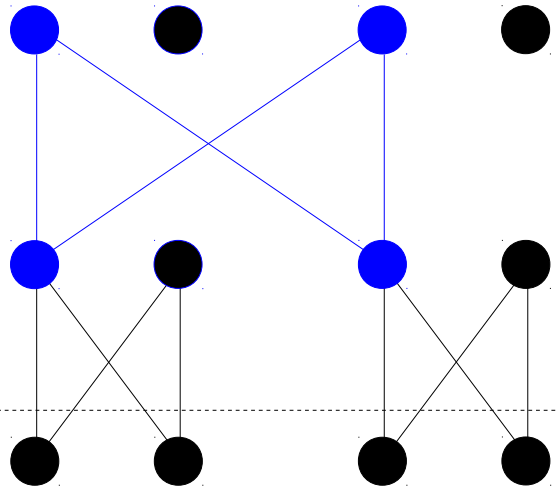


data-center topologies and ECMP

topology constraints:

- *l*-layers folded Clos networks (e.g.,VL2)

l=3



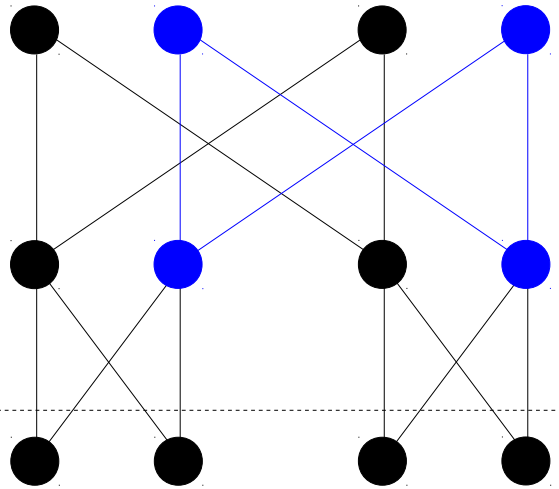
servers

data-center topologies and ECMP

topology constraints:

- *l*-layers folded Clos networks (e.g.,VL2)

l=3

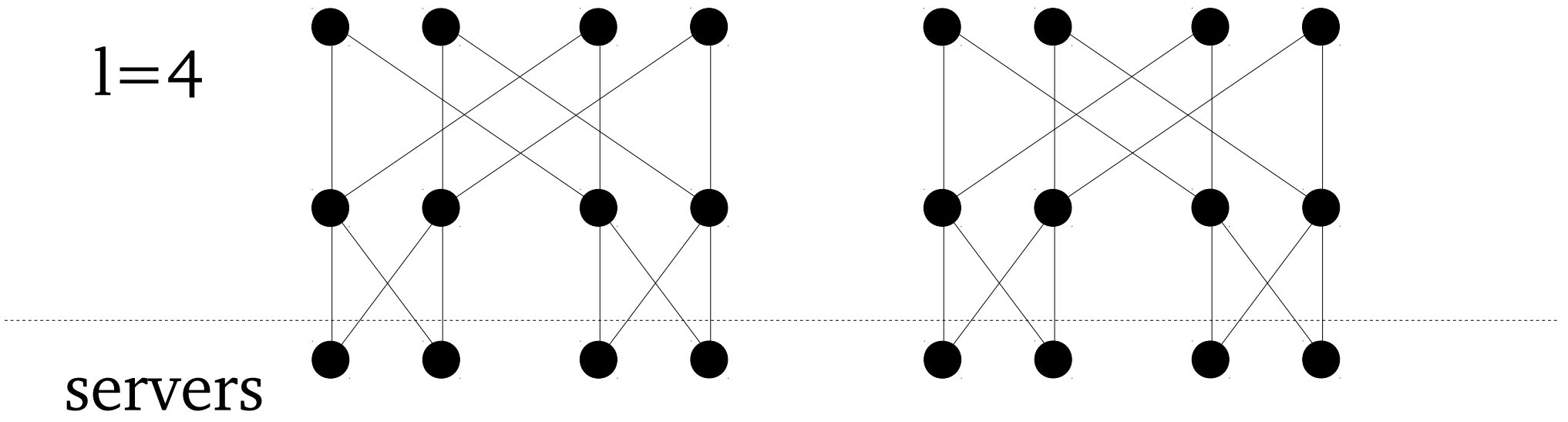


servers

data-center topologies and ECMP

topology constraints:

- *l*-layers folded Clos networks (e.g.,VL2)

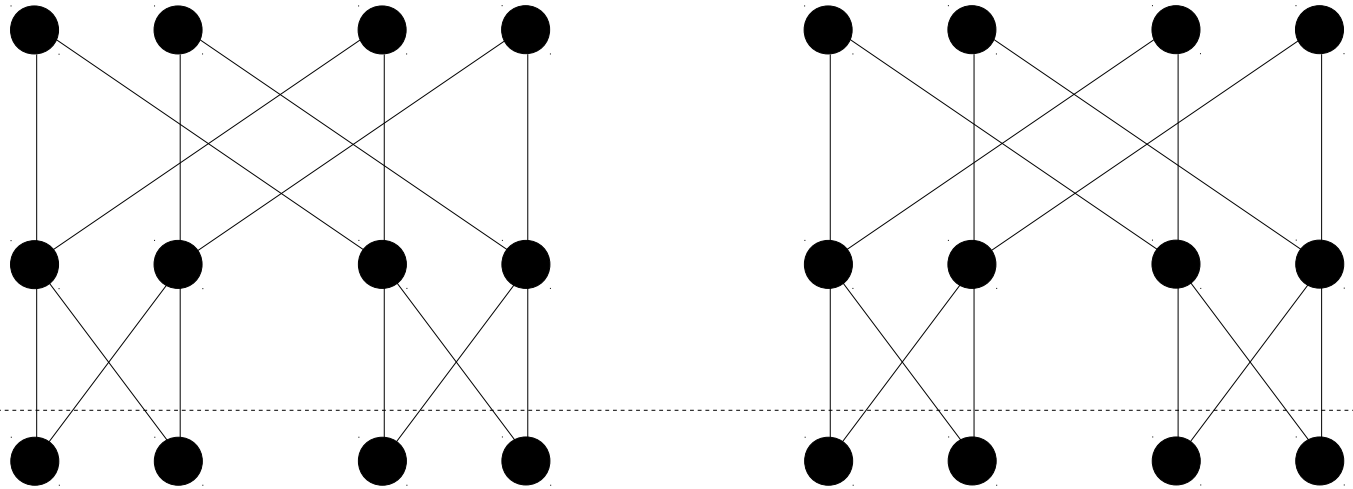


data-center topologies and ECMP

topology constraints:

- *l*-layers folded Clos networks (e.g.,VL2)

l=4

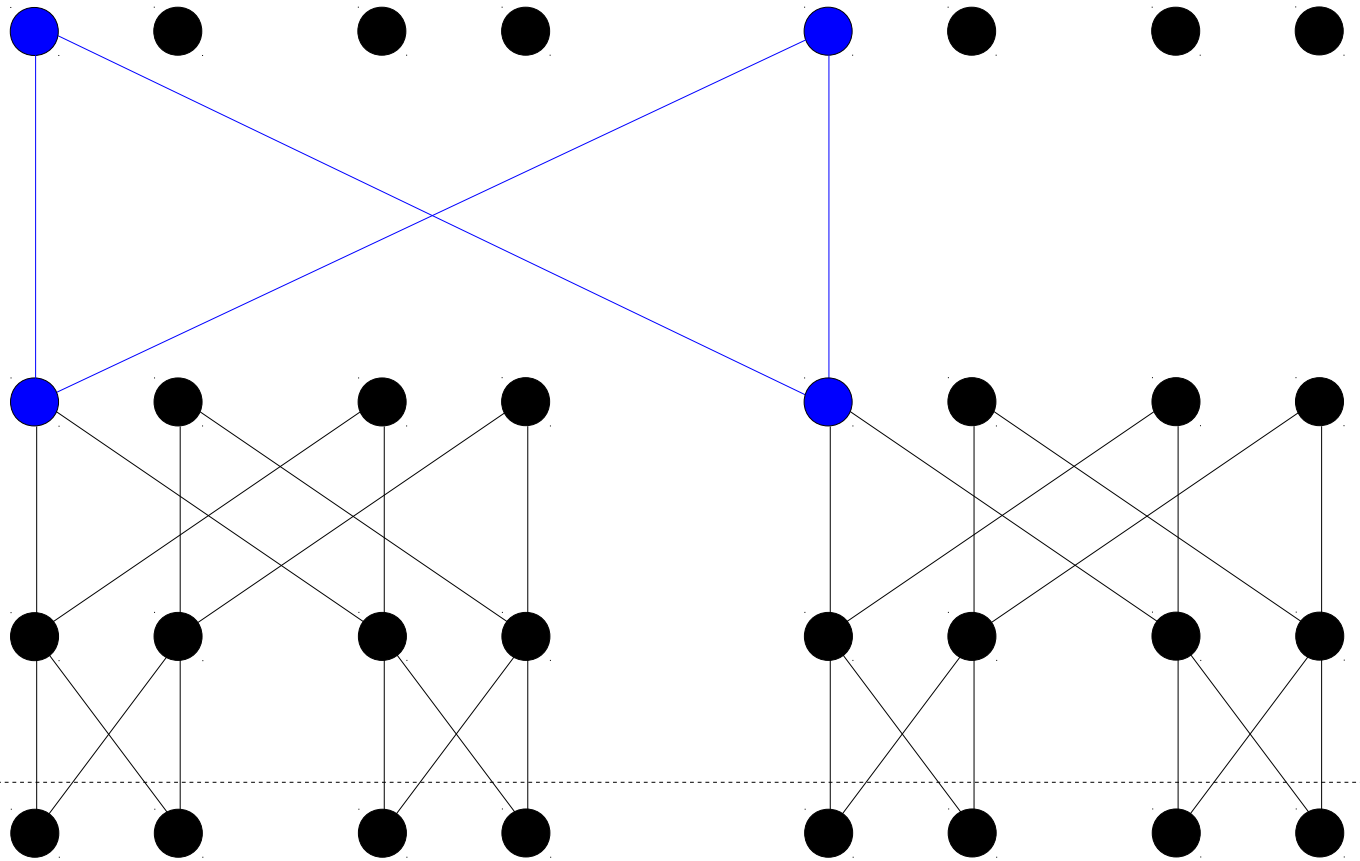


servers

data-center topologies and ECMP

topology constraints:

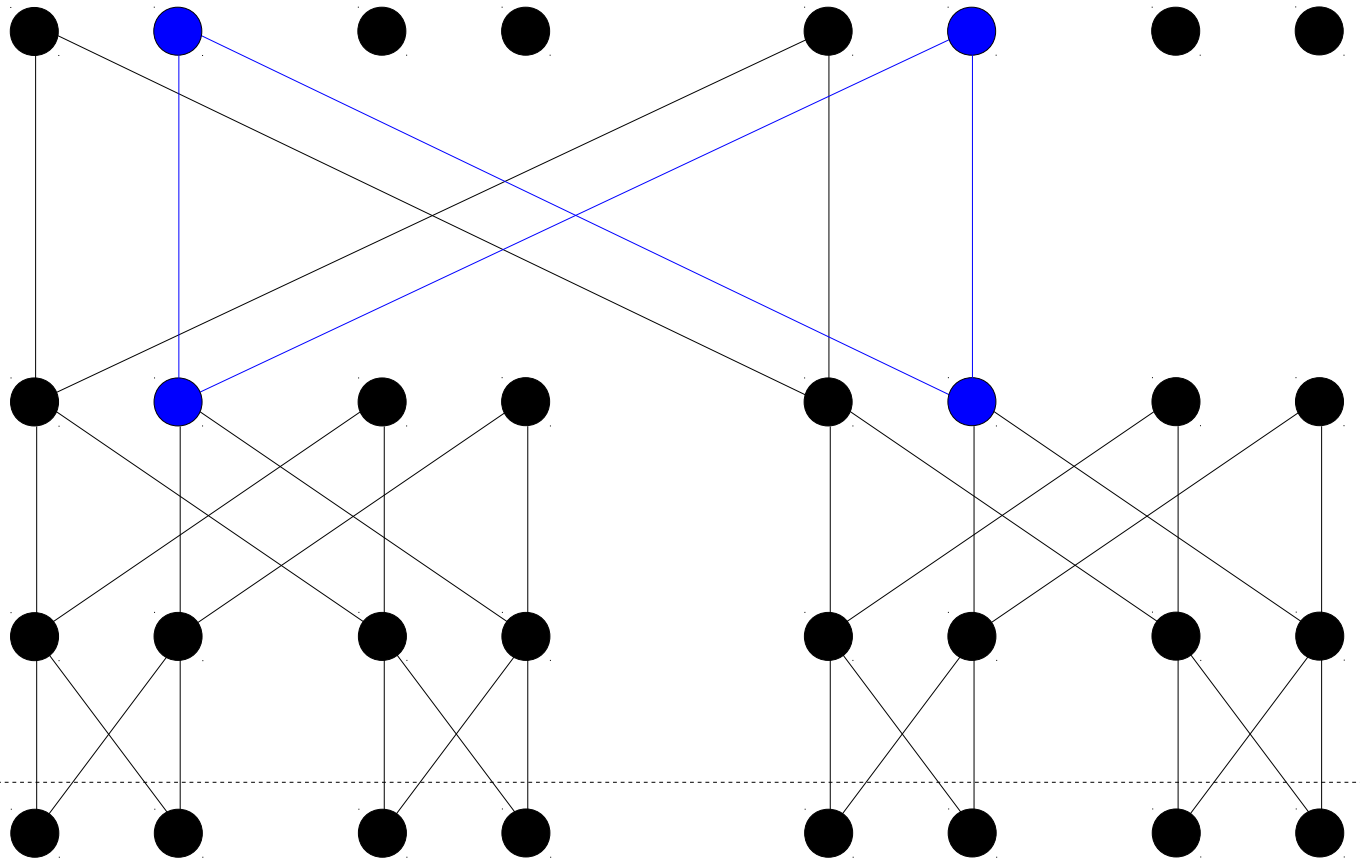
- *l*-layers folded Clos networks (e.g.,VL2)



data-center topologies and ECMP

topology constraints:

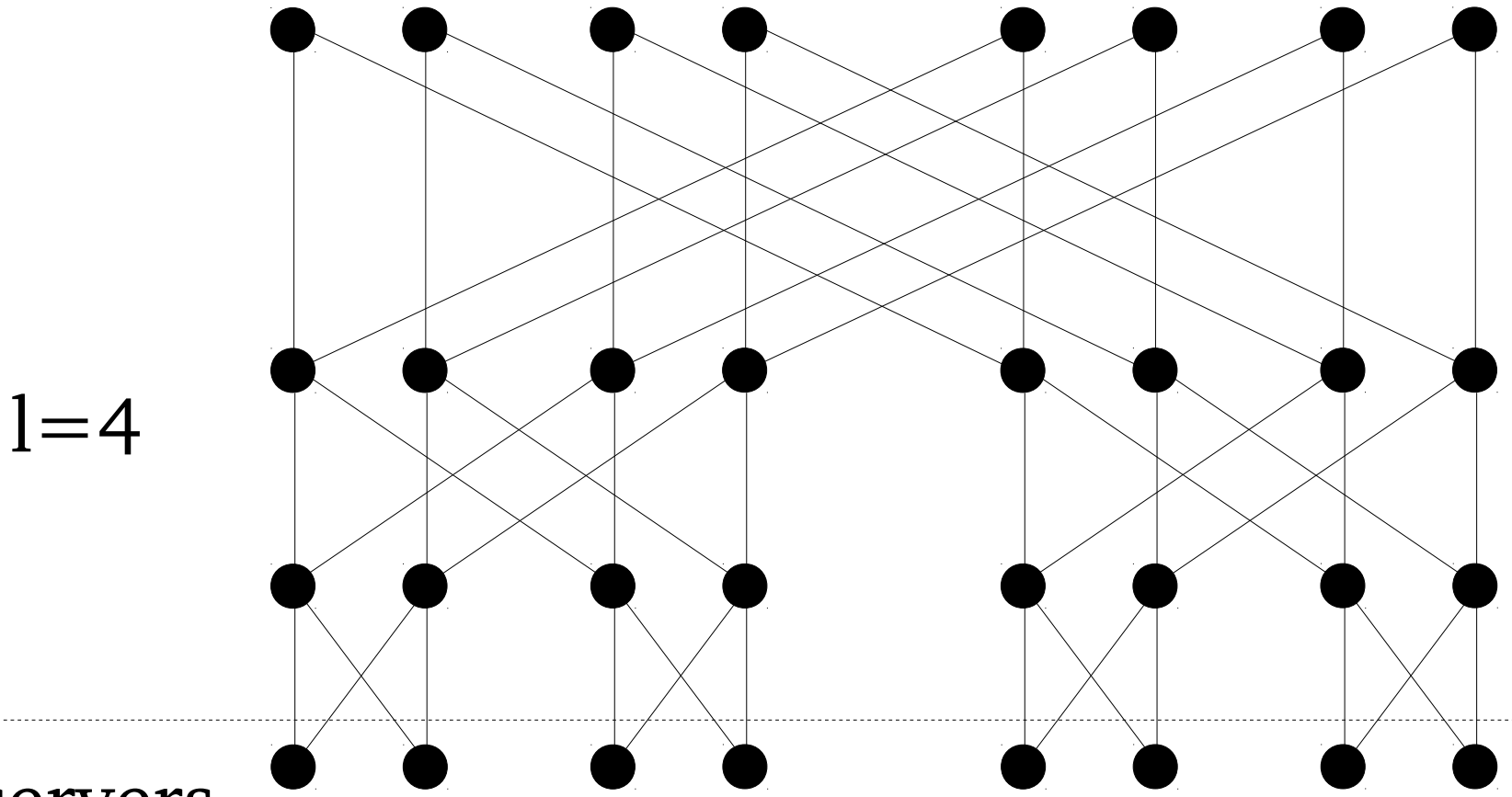
- *l*-layers folded Clos networks (e.g.,VL2)



data-center topologies and ECMP

topology constraints:

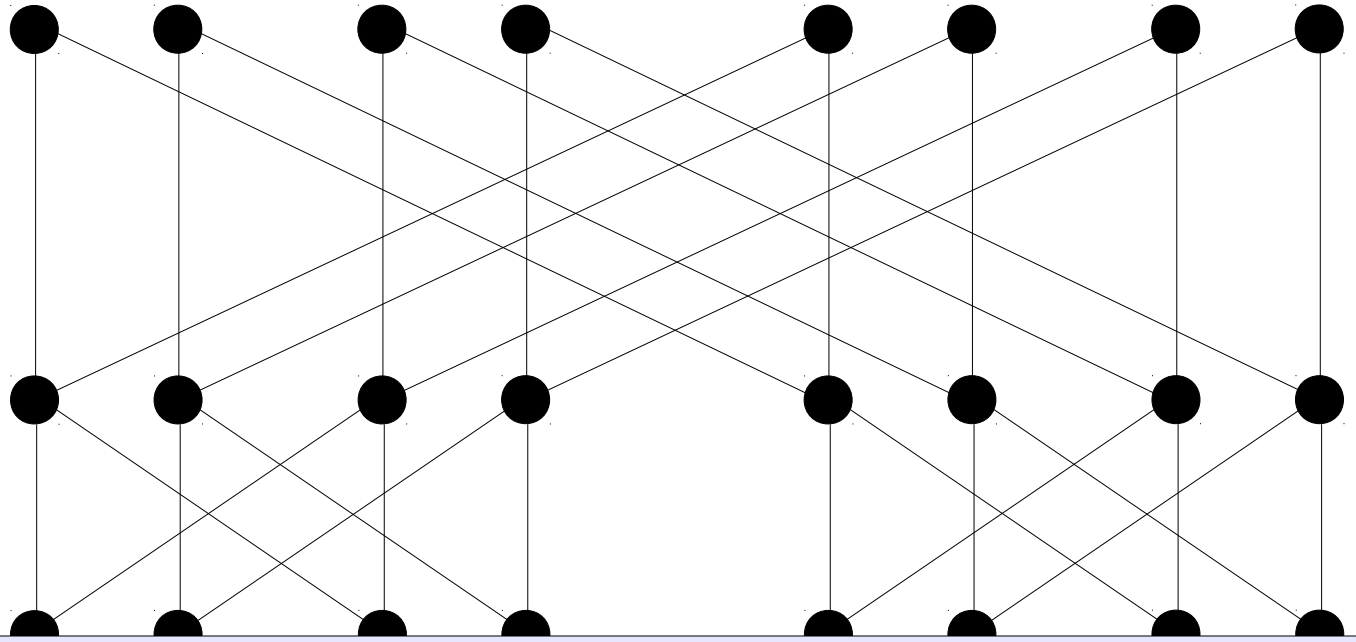
- *l*-layers folded Clos networks (e.g.,VL2)



data-center topologies and ECMP

topology constraints:

- *l*-layers folded Clos networks (e.g.,VL2)

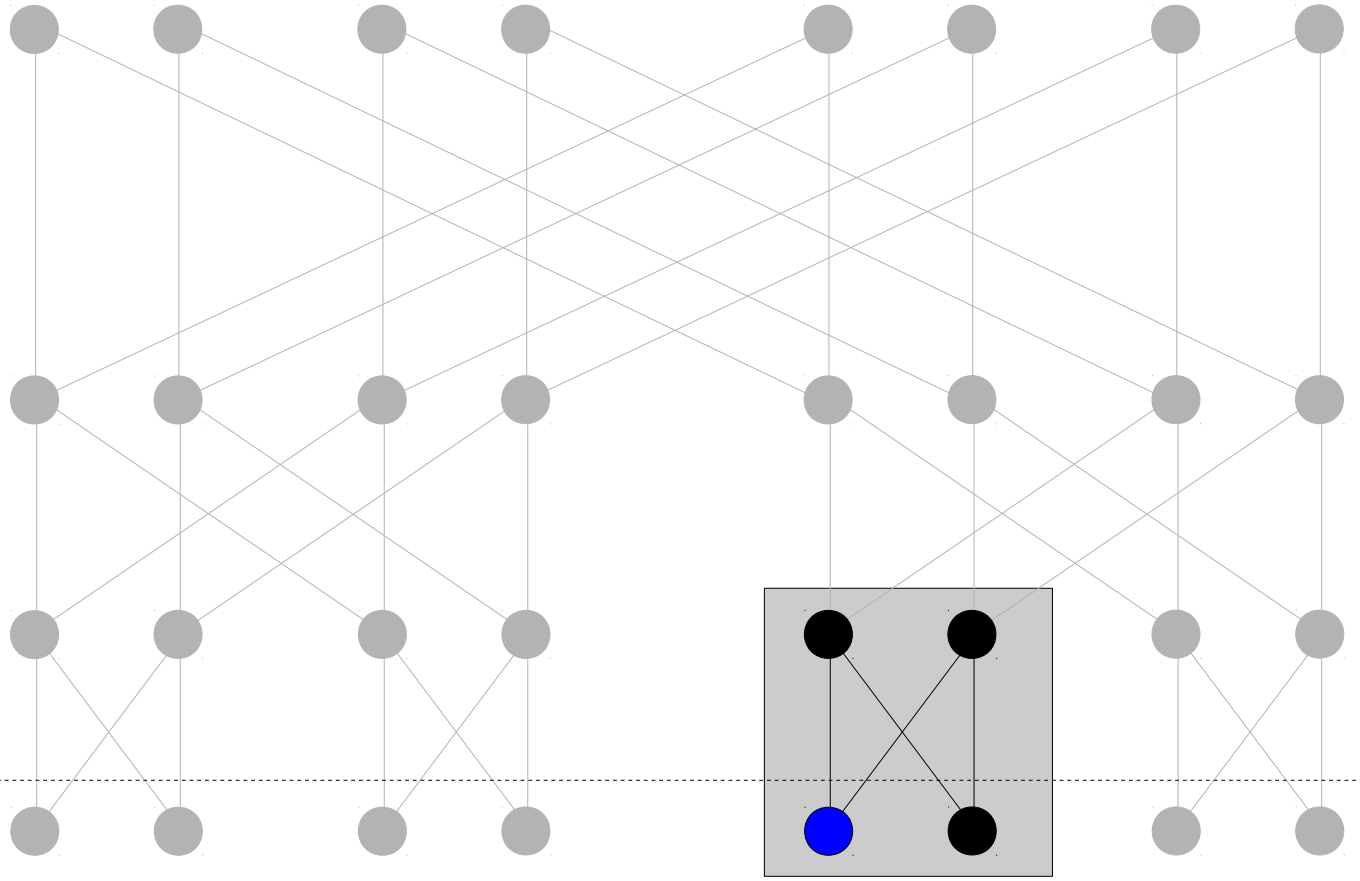


setting all link weights to 1 is optimal

optimality proof sketch

topology constraints:

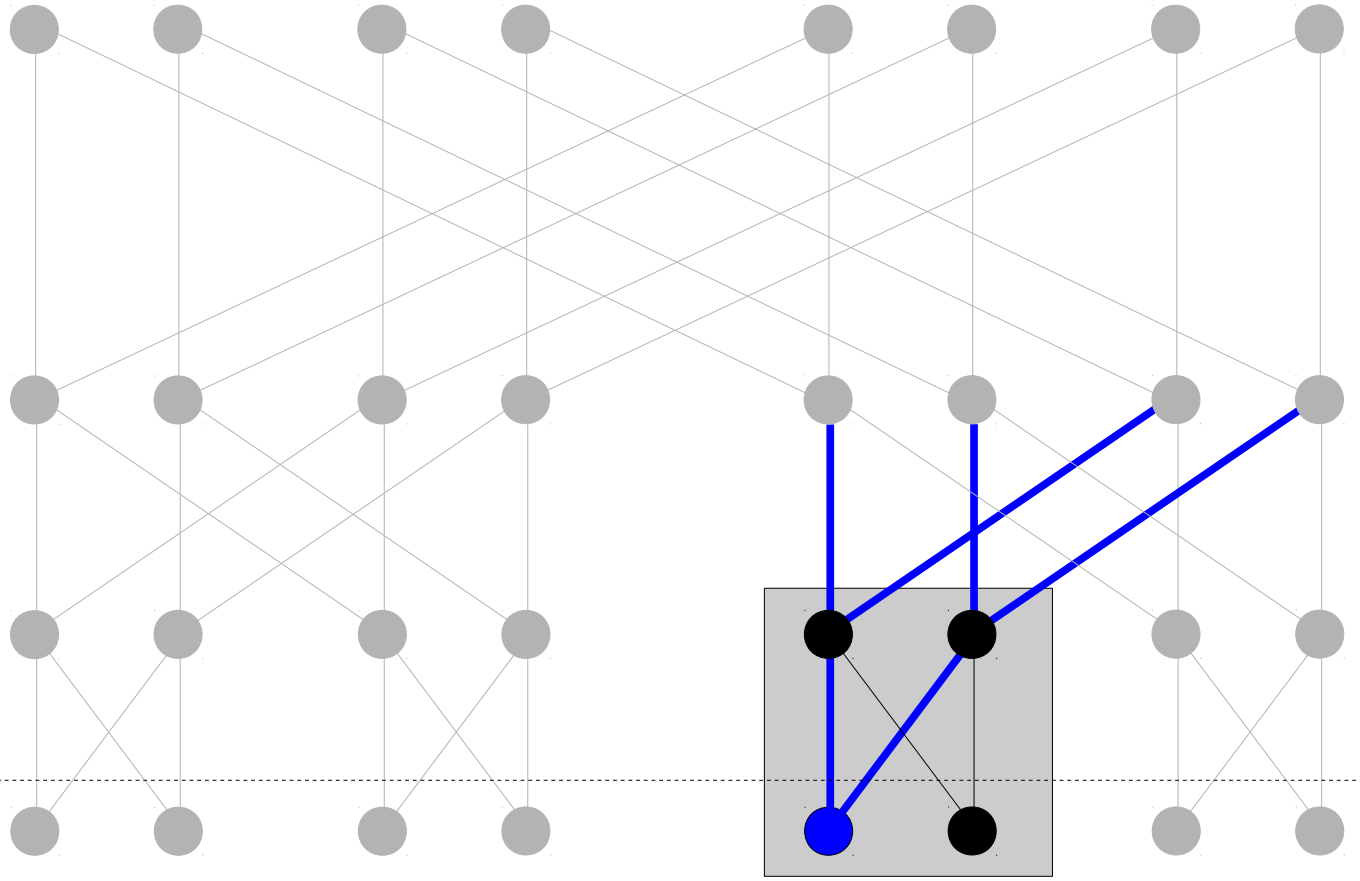
- *l*-layers folded Clos networks (e.g.,VL2)



optimality proof sketch

topology constraints:

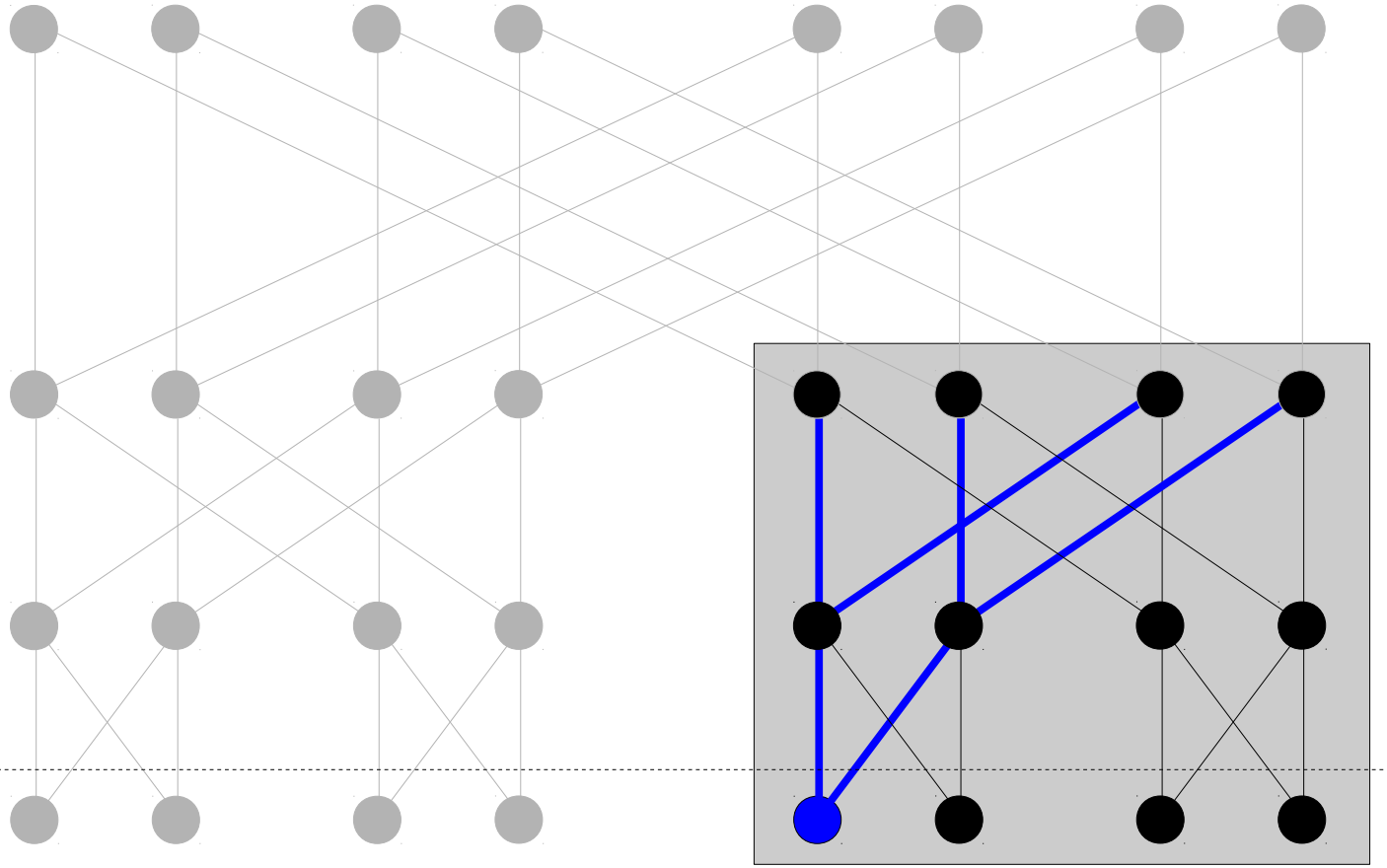
- *l*-layers folded Clos networks (e.g.,VL2)



optimality proof sketch

topology constraints:

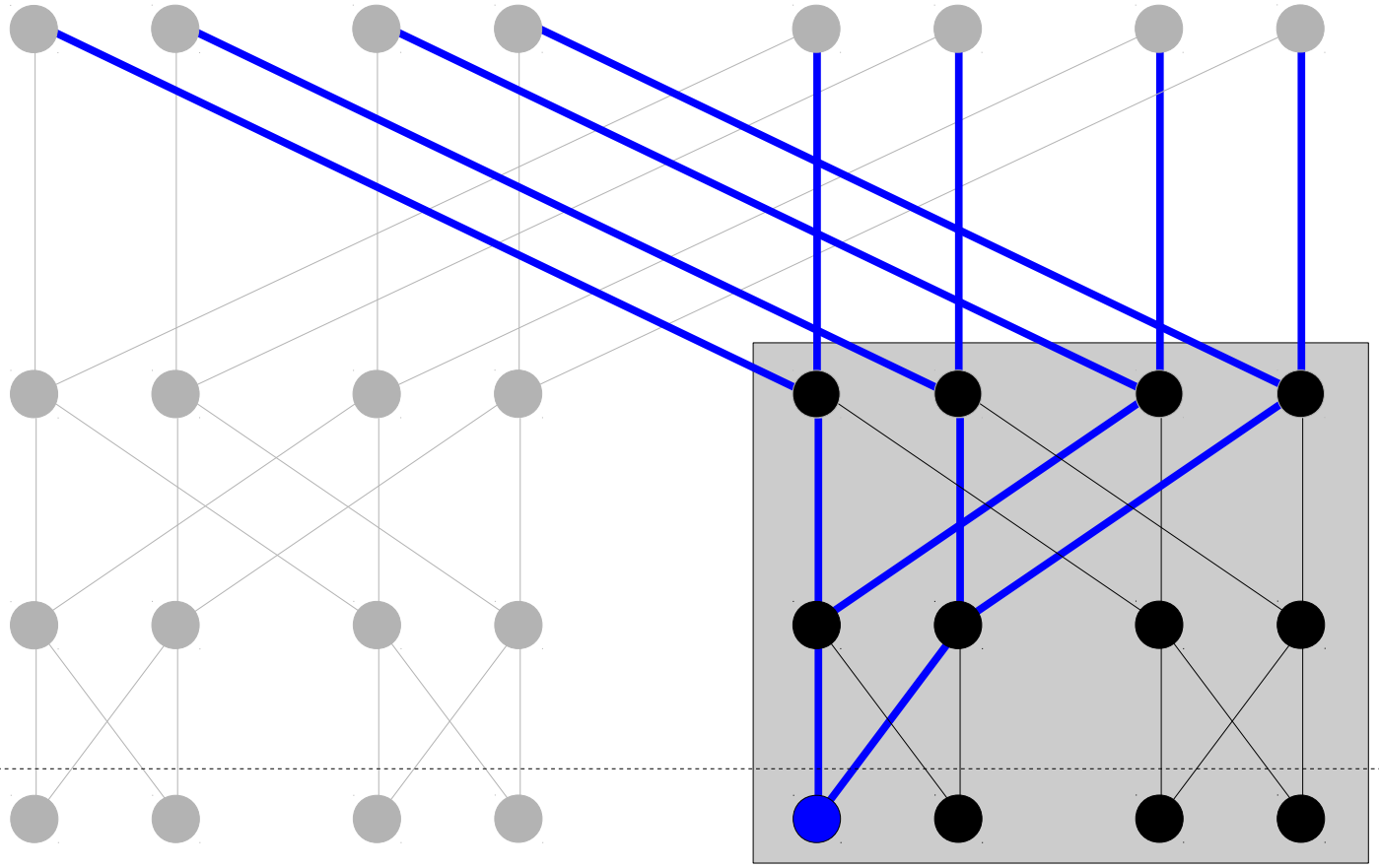
- *l*-layers folded Clos networks (e.g.,VL2)



optimality proof sketch

topology constraints:

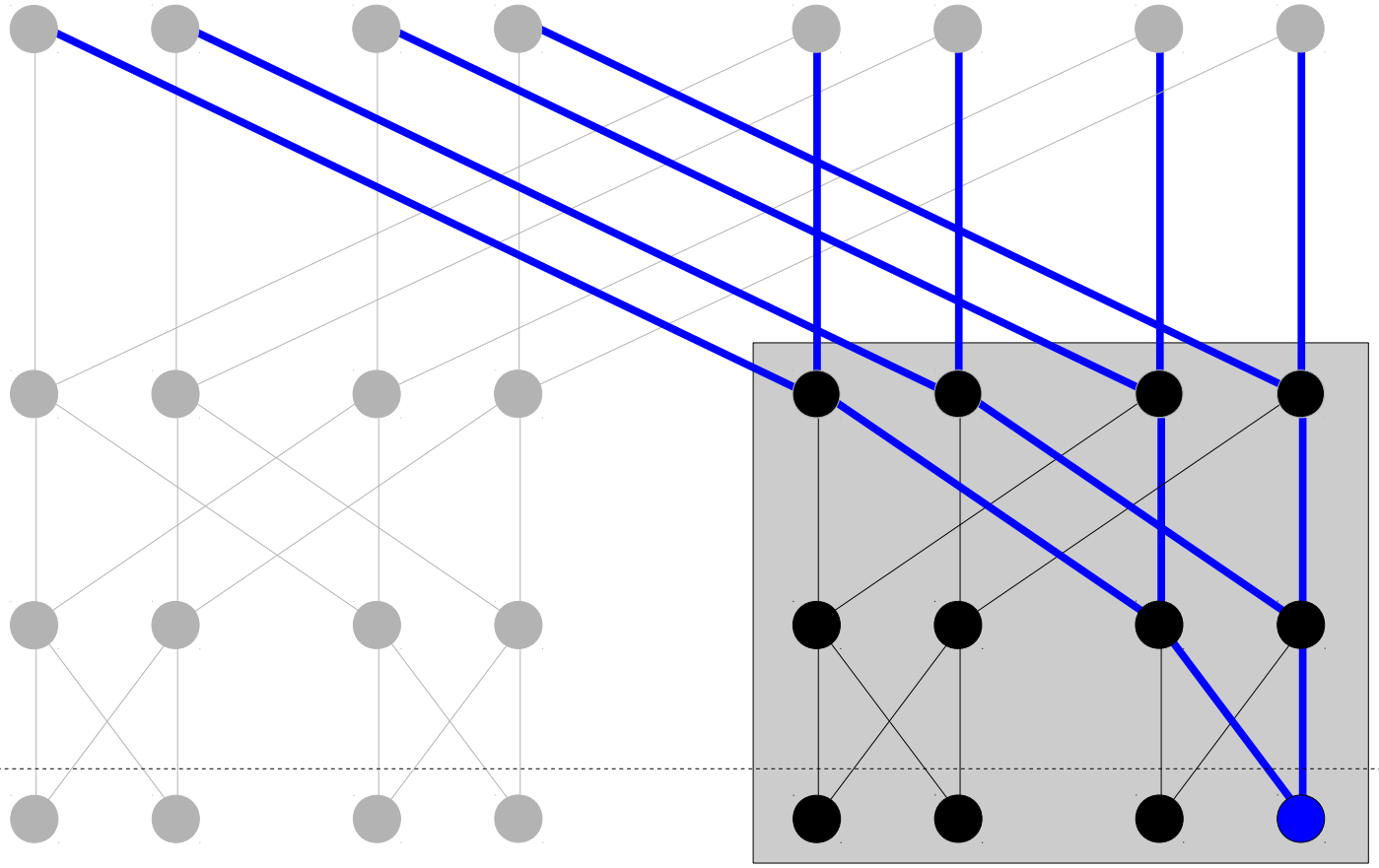
- *l*-layers folded Clos networks (e.g.,VL2)



optimality proof sketch

topology constraints:

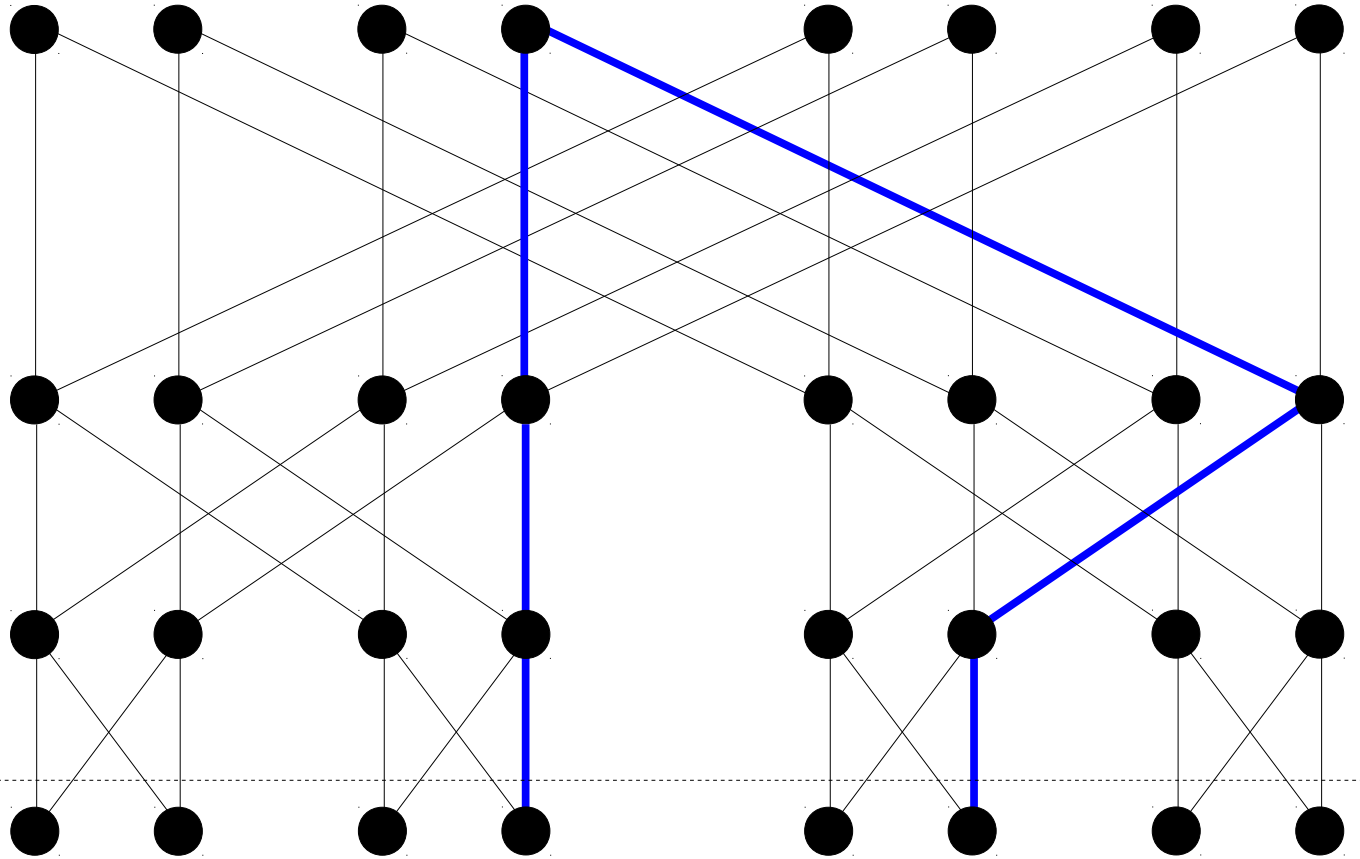
- *l*-layers folded Clos networks (e.g.,VL2)



ECMP and large flows

per-flow level hash-based split

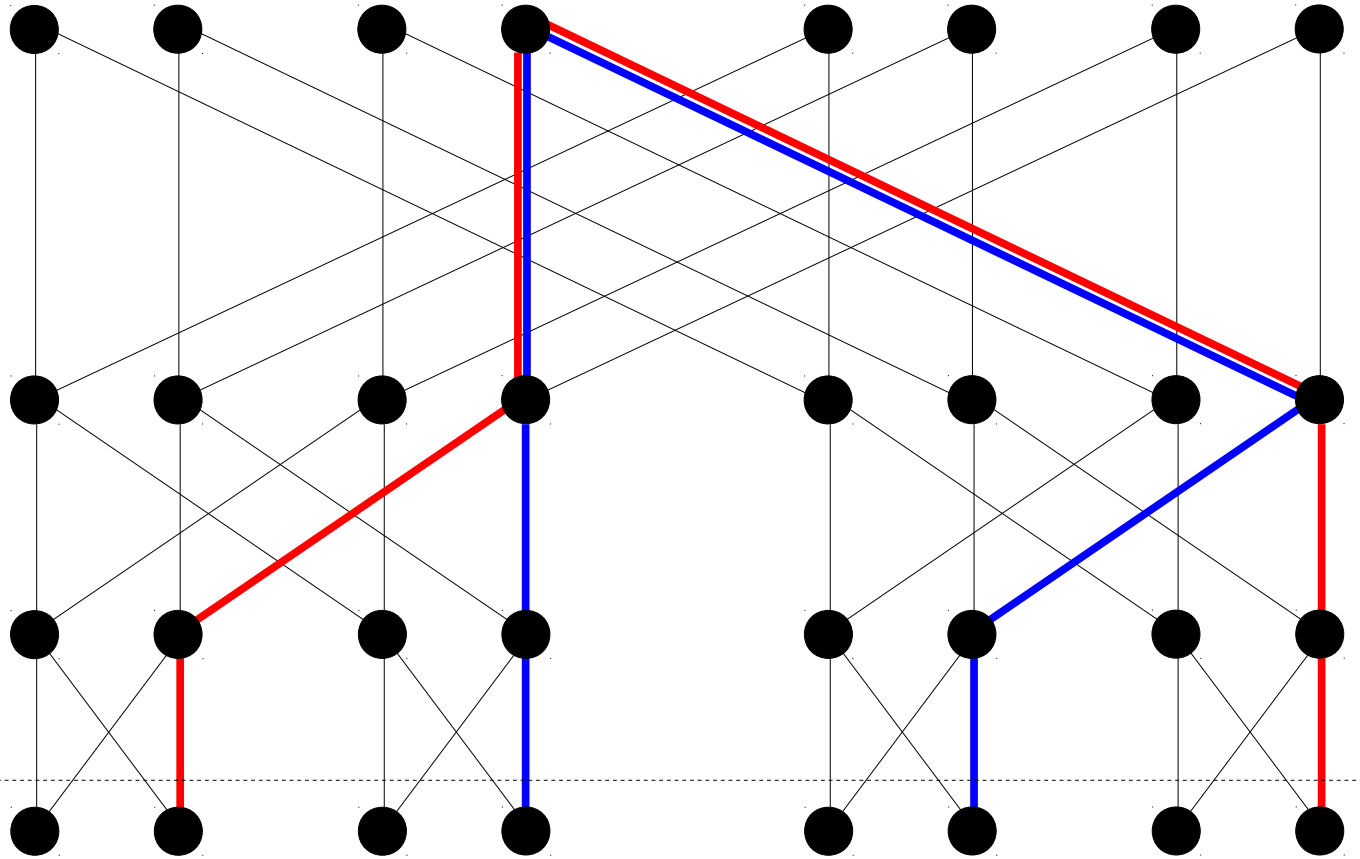
→ when there are a few large flows, traffic may not be properly load-balanced



ECMP and large flows

per-flow level hash-based split

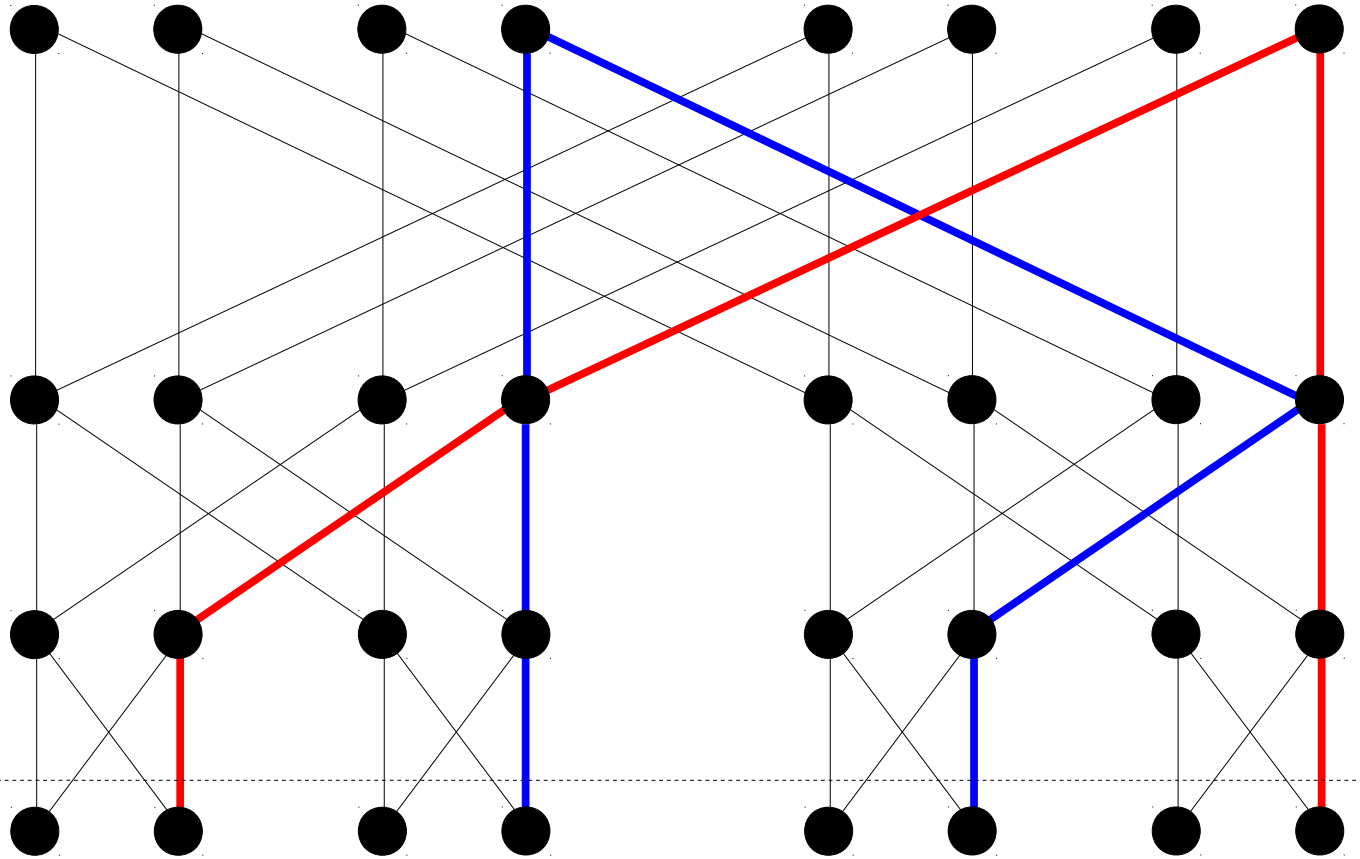
→ when there are a few large flows, traffic may not be properly load-balanced



ECMP and large flows

per-flow level hash-based split

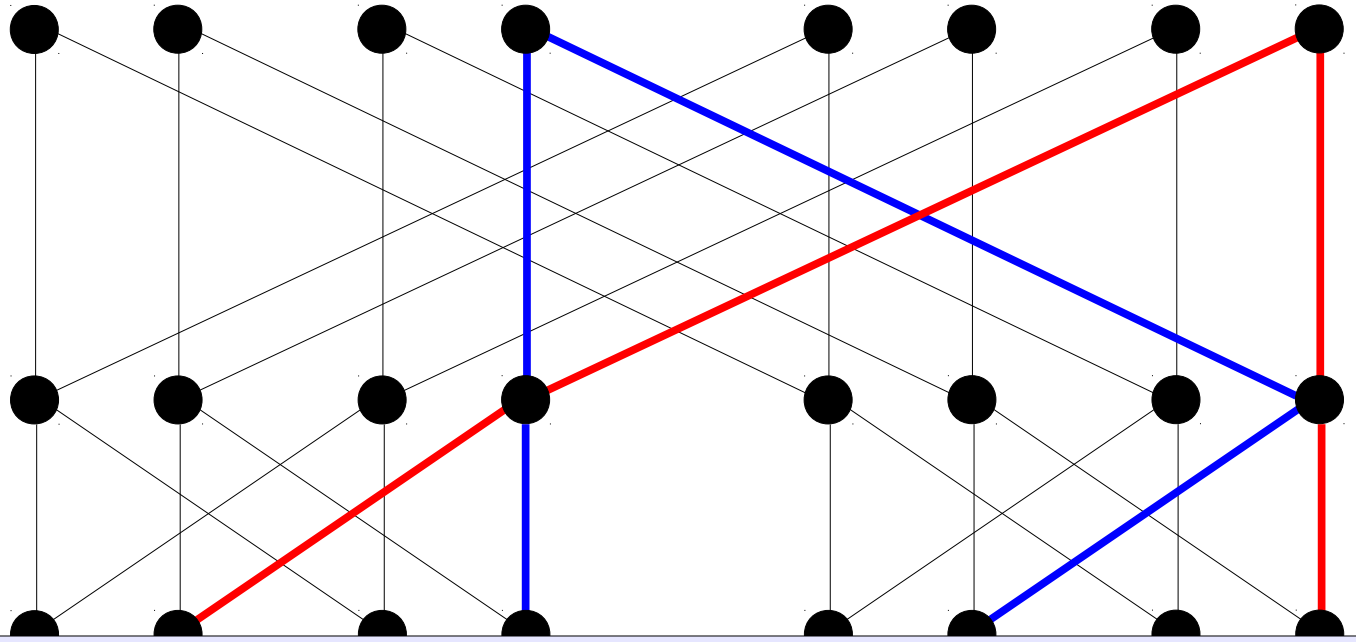
→ when there are a few large flows, traffic may not be properly load-balanced



ECMP and large flows

per-flow level hash-based split

→ when there are a few large flows, traffic may not be properly load-balanced



severe performance degradation [Al Fares et al, 2010]**:**
30% of the bandwidth in a datacenter is wasted

ECMP, Clos networks, and large flows

proposed solution [Al-Fares et al, 2010]:

- route small flows (mice) using ECMP
- route large flows (elephant) using a `greedy` algorithm

our results:

- $\frac{1}{2}$-inapproximability
- `greedy` is a $\frac{1}{5}$-approximation algorithm
  - $\frac{1}{4}$-approximation if all flows have equal size

conclusions

- in general, no efficient algorithm exists to assign the best link weights

conclusions

- in general, no efficient algorithm exists to assign ~~the best~~ reasonable link weights

conclusions

- in general, no efficient algorithm exists to assign ~~the best~~ reasonable link weights
- datacenter topologies:
  - hypercubes → hard to find the best weights
  - folded Clos networks → set all weights to 1
    - `greedy` algorithm for routing large flows is a $\frac{1}{5}$-approximation in a 3-layers Folded Clos network (VL2-like)

thank you