# WEAVER

## Efficient Coflow Scheduling in Heterogeneous Parallel Networks

Xin Sunny Huang Yiting Xia T. S. Eugene Ng

Rice University

Big Data and Optical Lightpaths Driven Lab

# This Work

**Optimizing Coflow performance** has many benefits such as avoiding application stragglers[1,2] and improving resource utilization[3,4]. Existing Coflow studies all assume a monolithic network model.

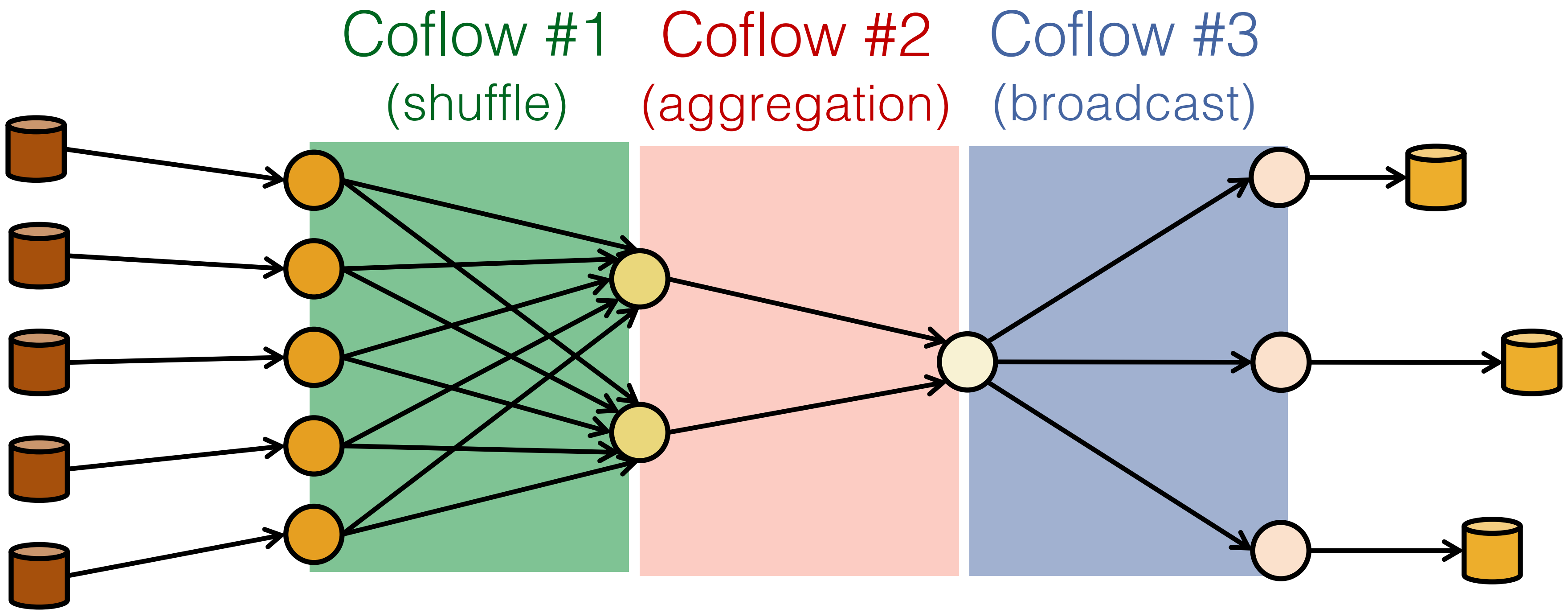
New technology trends lead to **Heterogeneous Parallel Networks** in an evolving data center.

**Weaver** is the first scheduler to address the Coflow management problem in Heterogeneous Parallel Networks.

[1] **Orchestra** (SIGCOMM '11). [2] **Varys** (SIGCOMM '14).
[3] **CARBYNE** (OSDI '16). [4] **YARN-ME** (memory elasticity, in ATC '17)

# Coflow: Traffic Abstraction for MapReduce-like Applications

- Coflow[1] : A set of related flows.
- Performance is measured by Coflow Completion Time (CCT), i.e. the last flow's completion time.
- Coflow-aware scheduling speeds up applications[2][3].

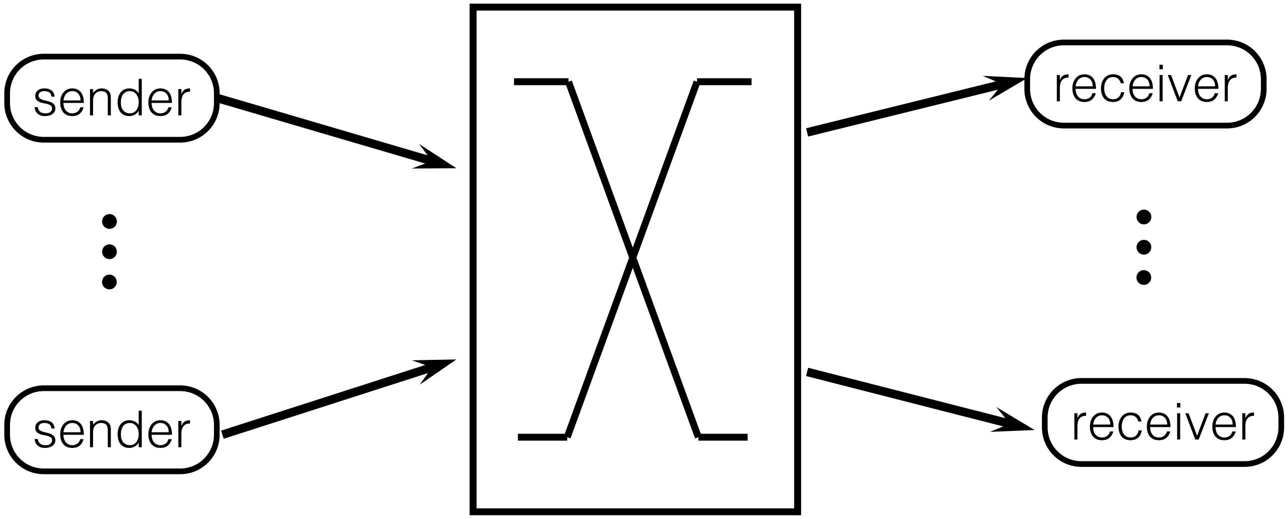


[1] Chowdhury, M. et al. Coflow: An application layer abstraction for cluster networking. (HotNets'12)
[2] Chowdhury, M. et al. Efficient coflow scheduling with Varys. (SIGCOMM'14)
[3] Chowdhury, M. et al. Efficient Coflow Scheduling Without Prior Knowledge. (SIGCOMM'15)

# Coflow Scheduling

- Prior works demonstrate benefits of Coflow scheduling.

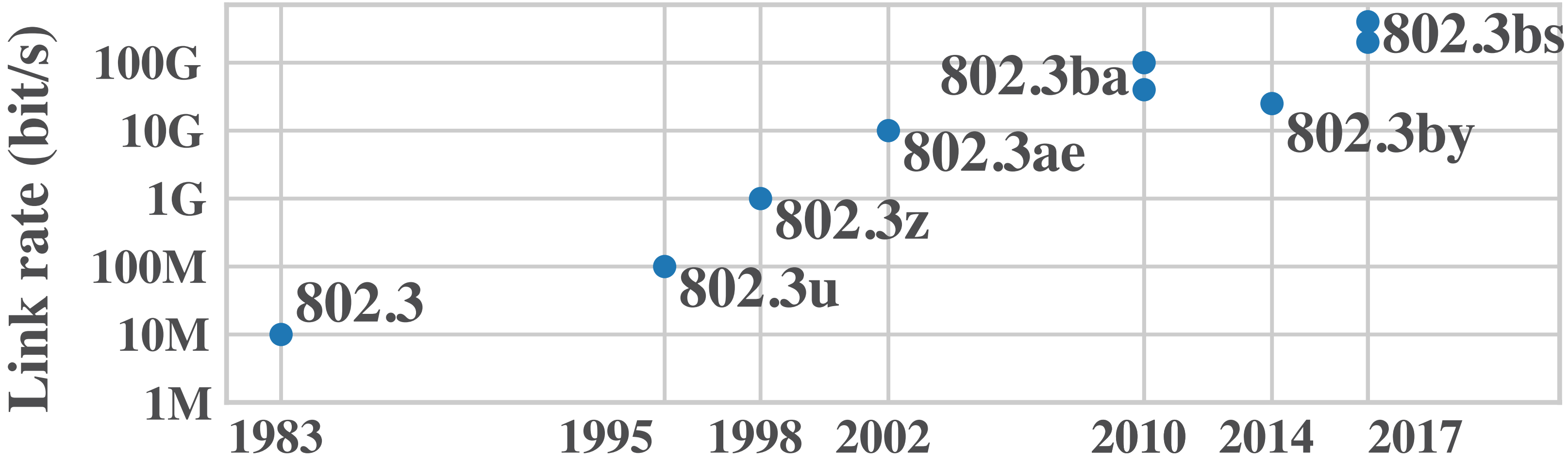
- Limitation: Assumes "big-switch" network model, which abstracts the whole network fabric as a non-blocking switch.



This network model is no longer sufficient under recent technology trends

**Varys** (SIGCOMM '14), **Aalo** (SIGCOMM '15), **CODA** (SIGCOMM '16) and **Sunflow** (CoNEXT '16), etc.

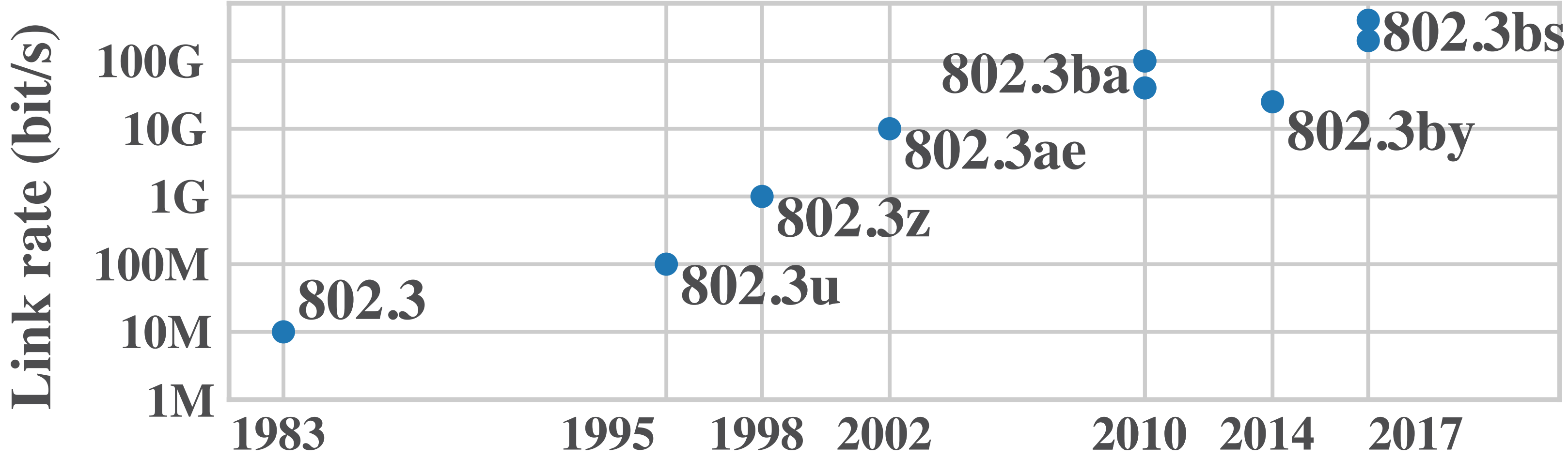
# Shrinking Generation Gap in Link Speed



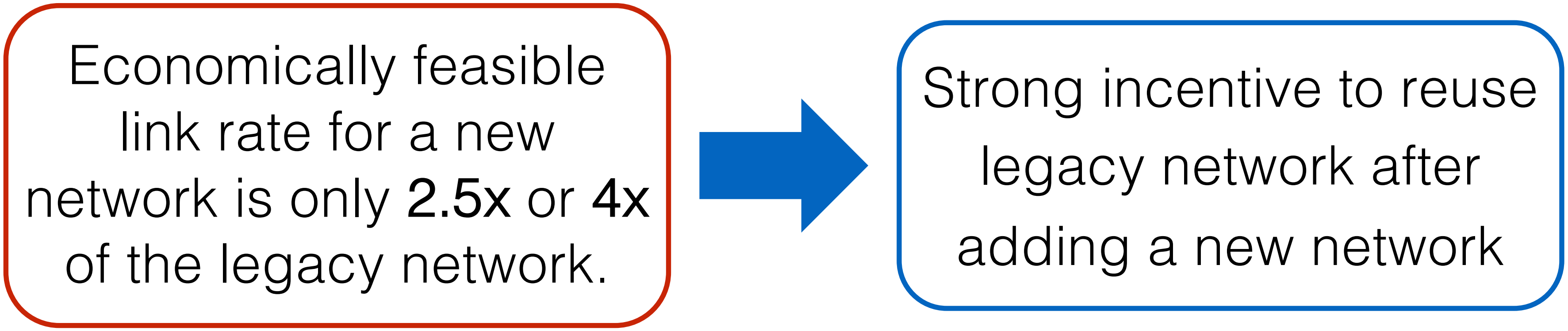
*Link rate and the year first introduced in IEEE 802.3*

Economically feasible link rate for a new network is only 2.5x or 4x of the legacy network.

# Shrinking Generation Gap in Link Speed



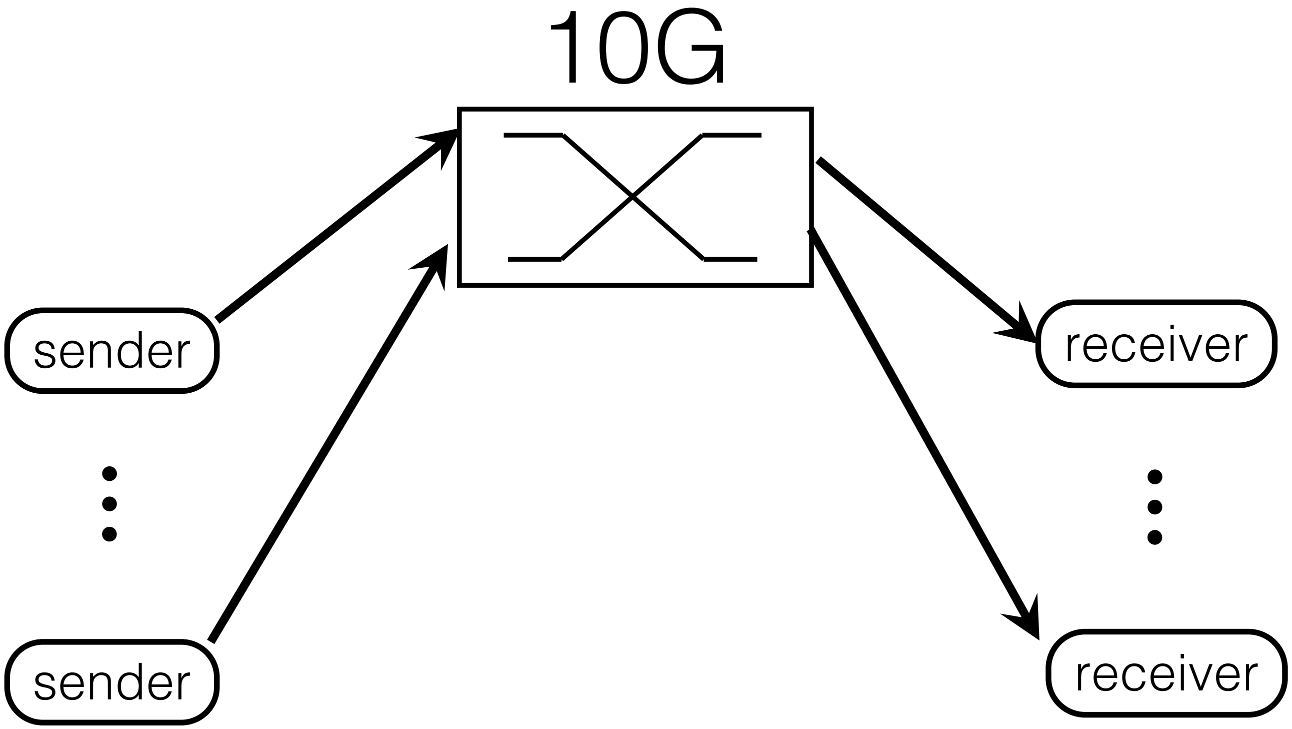
*Link rate and the year first introduced in IEEE 802.3*



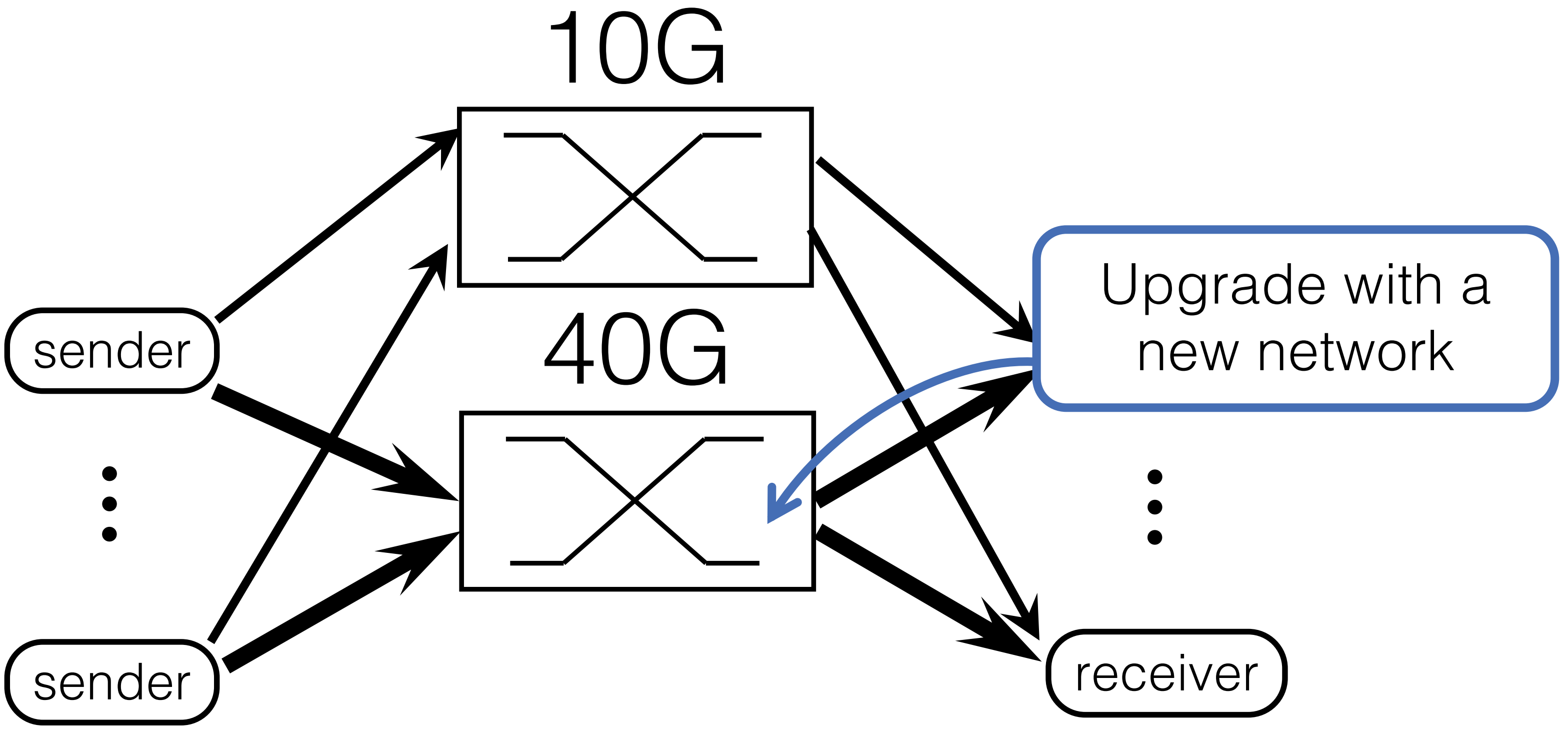
Economically feasible link rate for a new network is only **2.5x** or **4x** of the legacy network.

Strong incentive to reuse legacy network after adding a new network

# Exploit the Shrinking Gap with Heterogeneous Parallel Networks

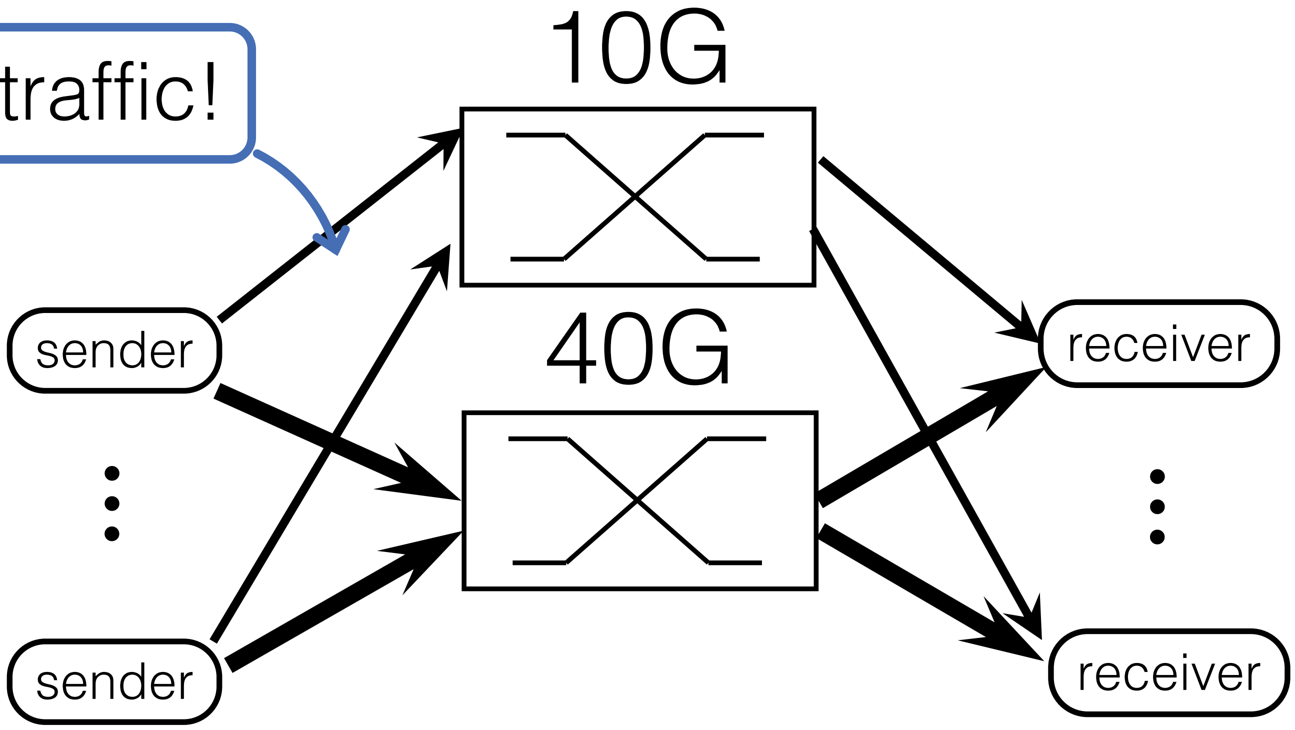


# Exploit the Shrinking Gap with Heterogeneous Parallel Networks



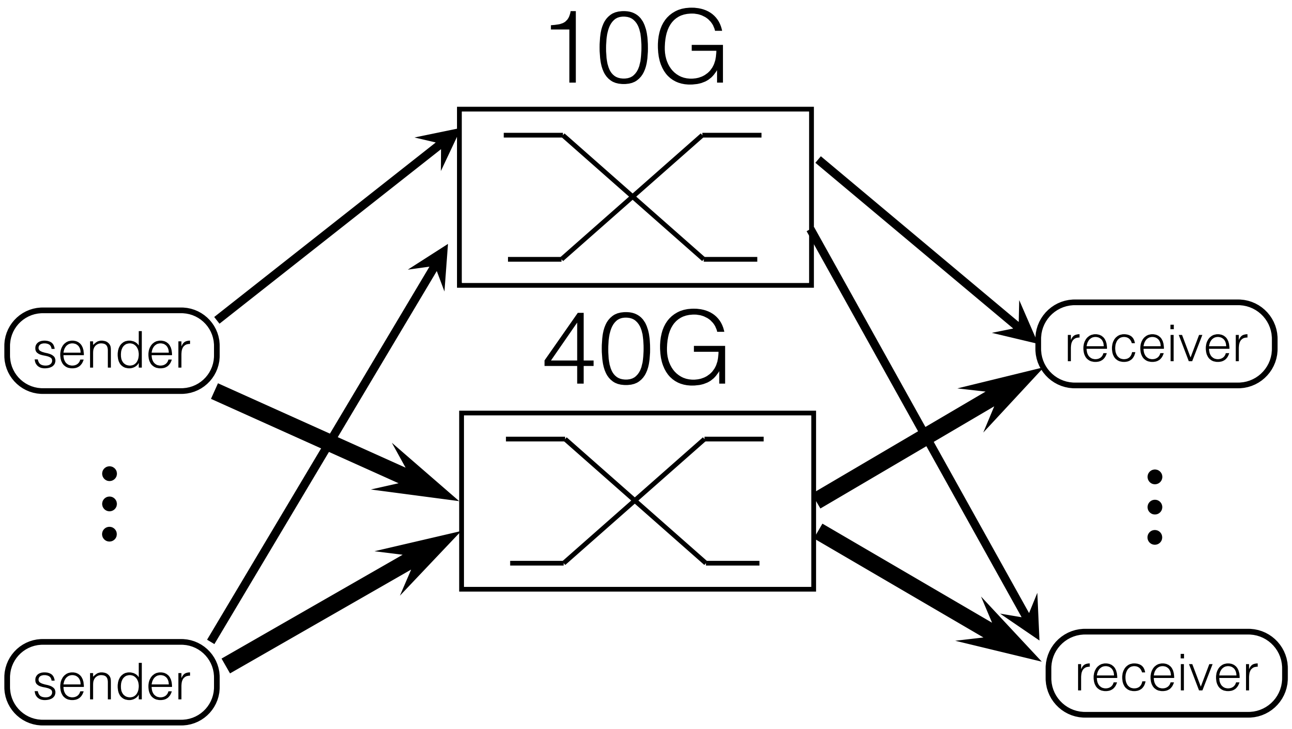
# Exploit the Shrinking Gap with Heterogeneous Parallel Networks

20% of traffic!

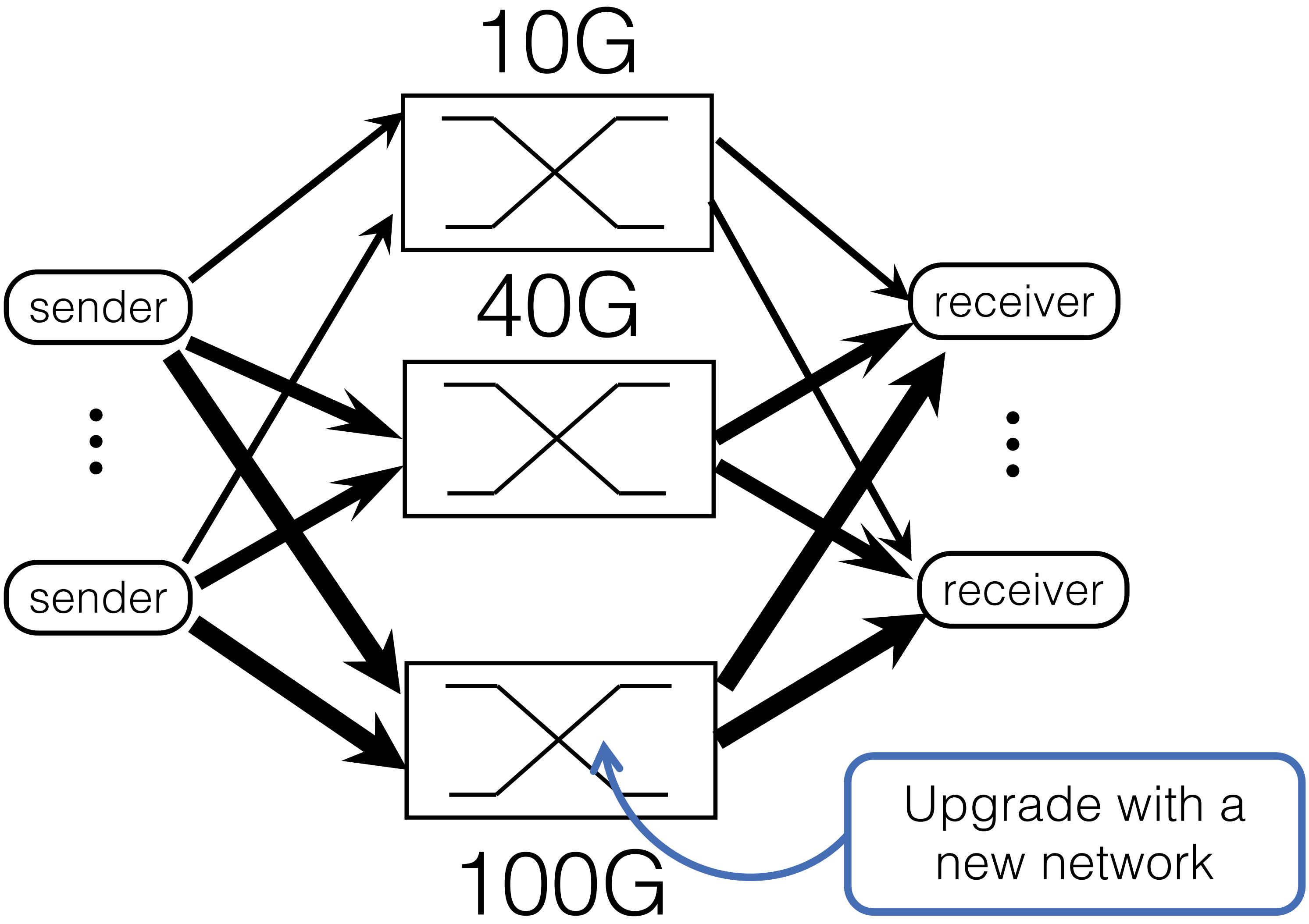


Because of the shrinking gap, a legacy network can still service a considerable amount of traffic relative to a new network.

# Exploit the Shrinking Gap with Heterogeneous Parallel Networks

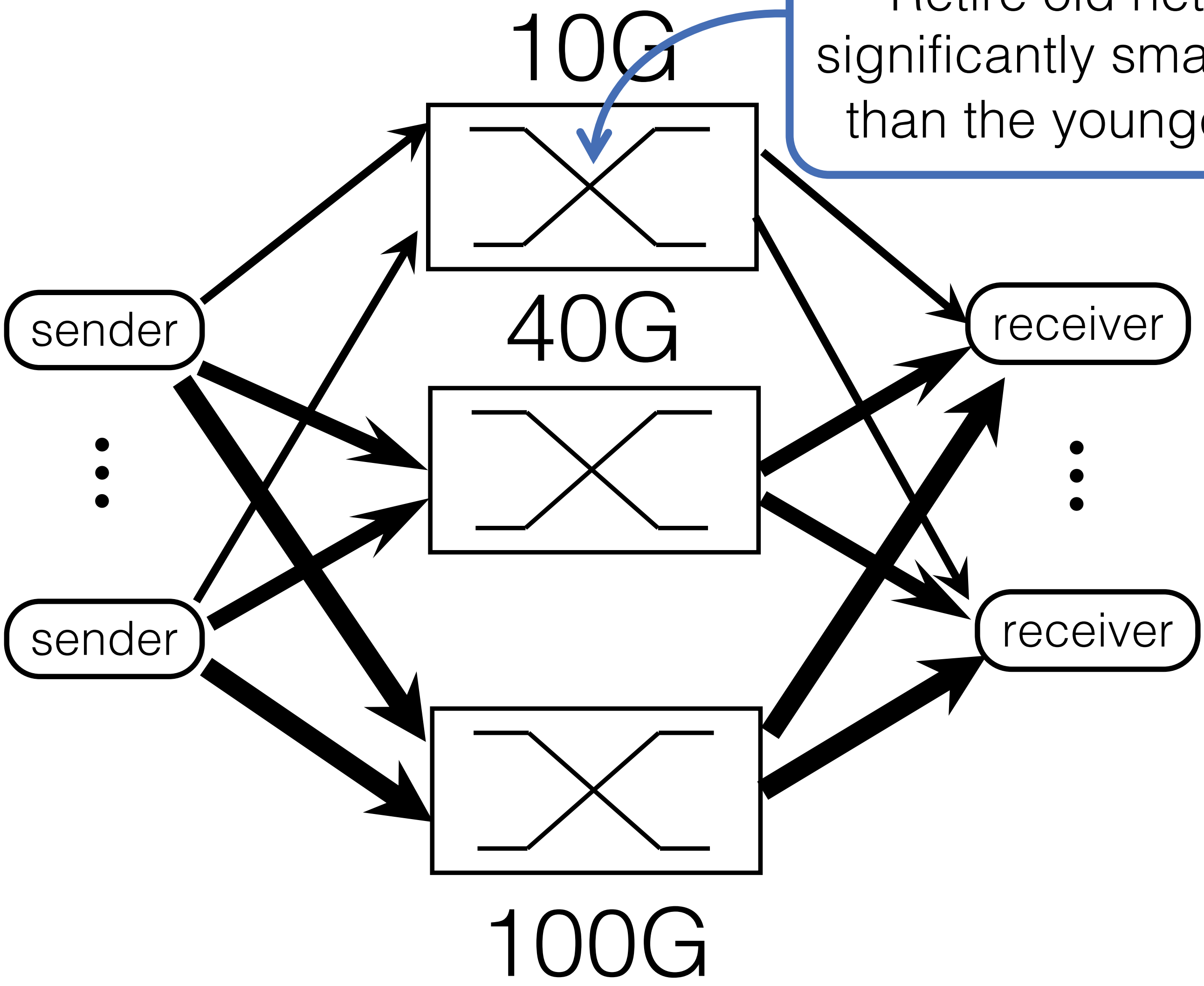


# Exploit the Shrinking Gap with Heterogeneous Parallel Networks



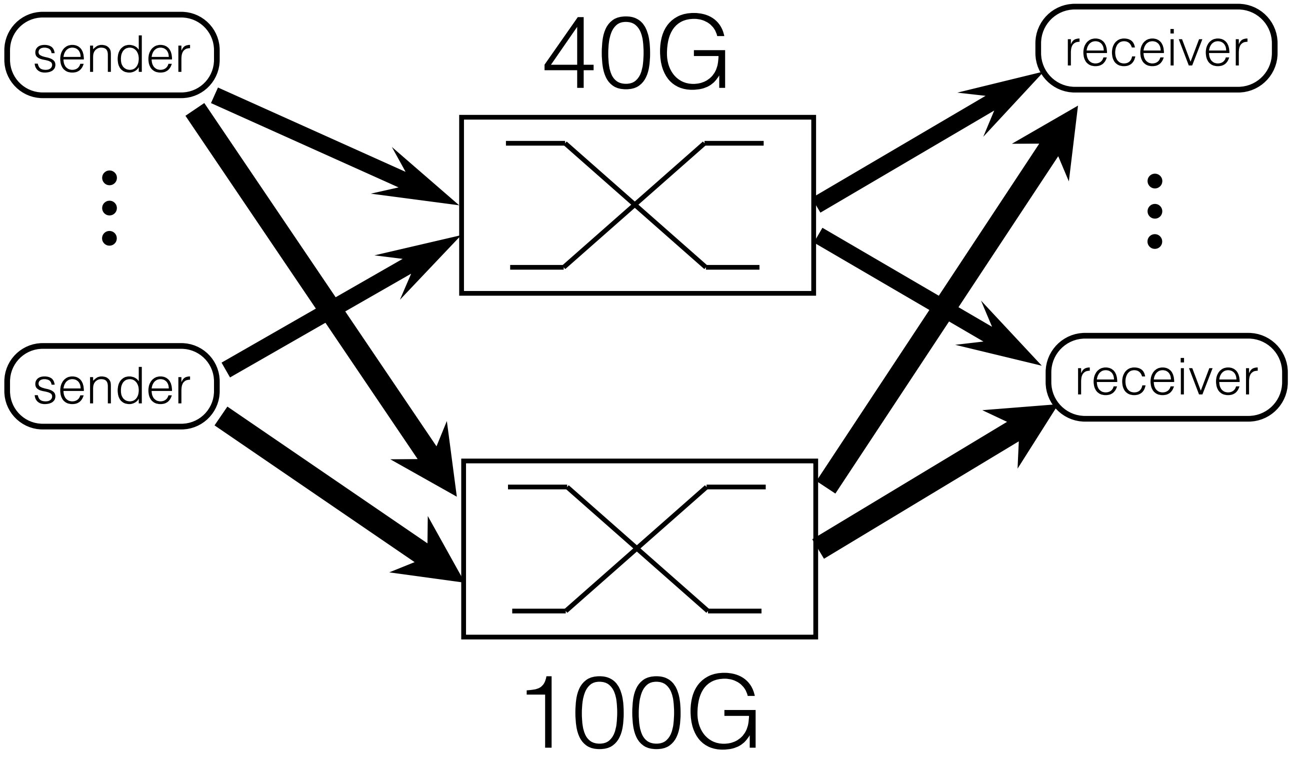
Upgrade with a new network

# Exploit the Shrinking Gap with Heterogeneous Parallel Networks



Retire old network with significantly smaller capacity than the youngest network

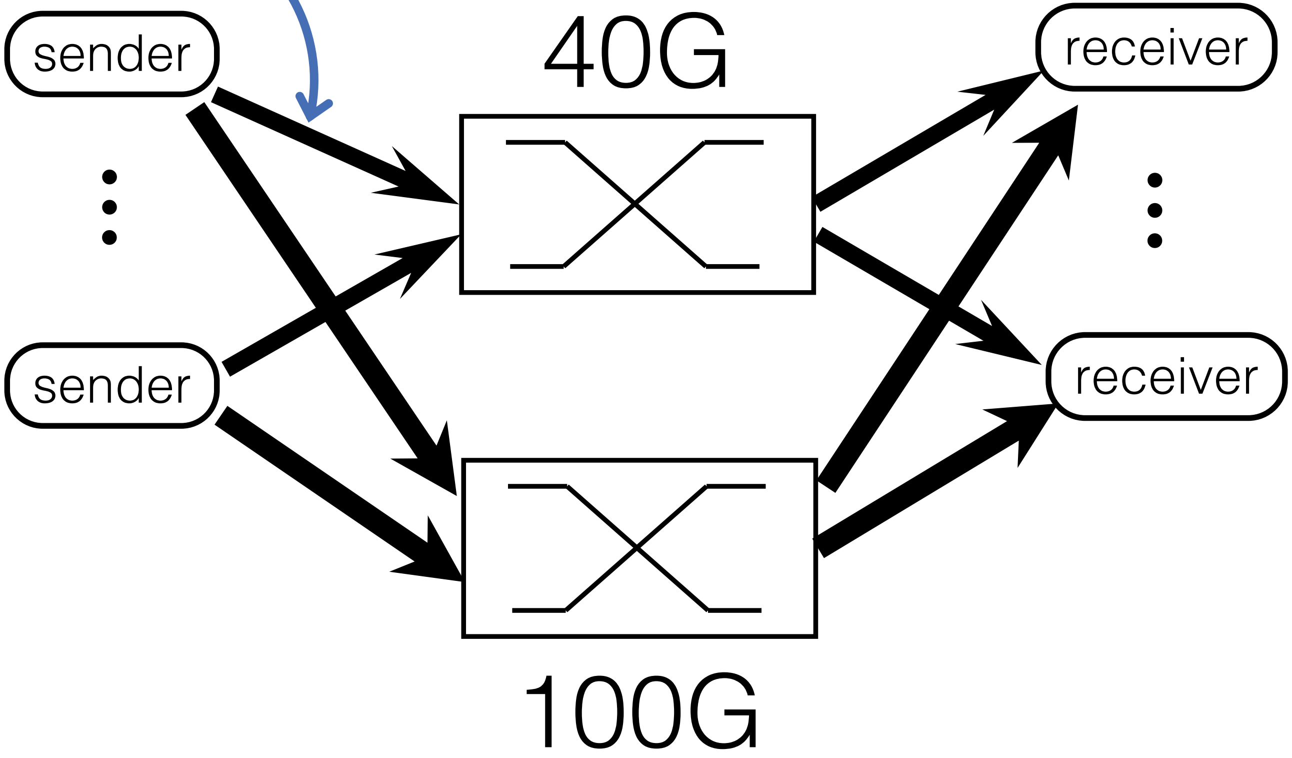
# Exploit the Shrinking Gap with Heterogeneous Parallel Networks



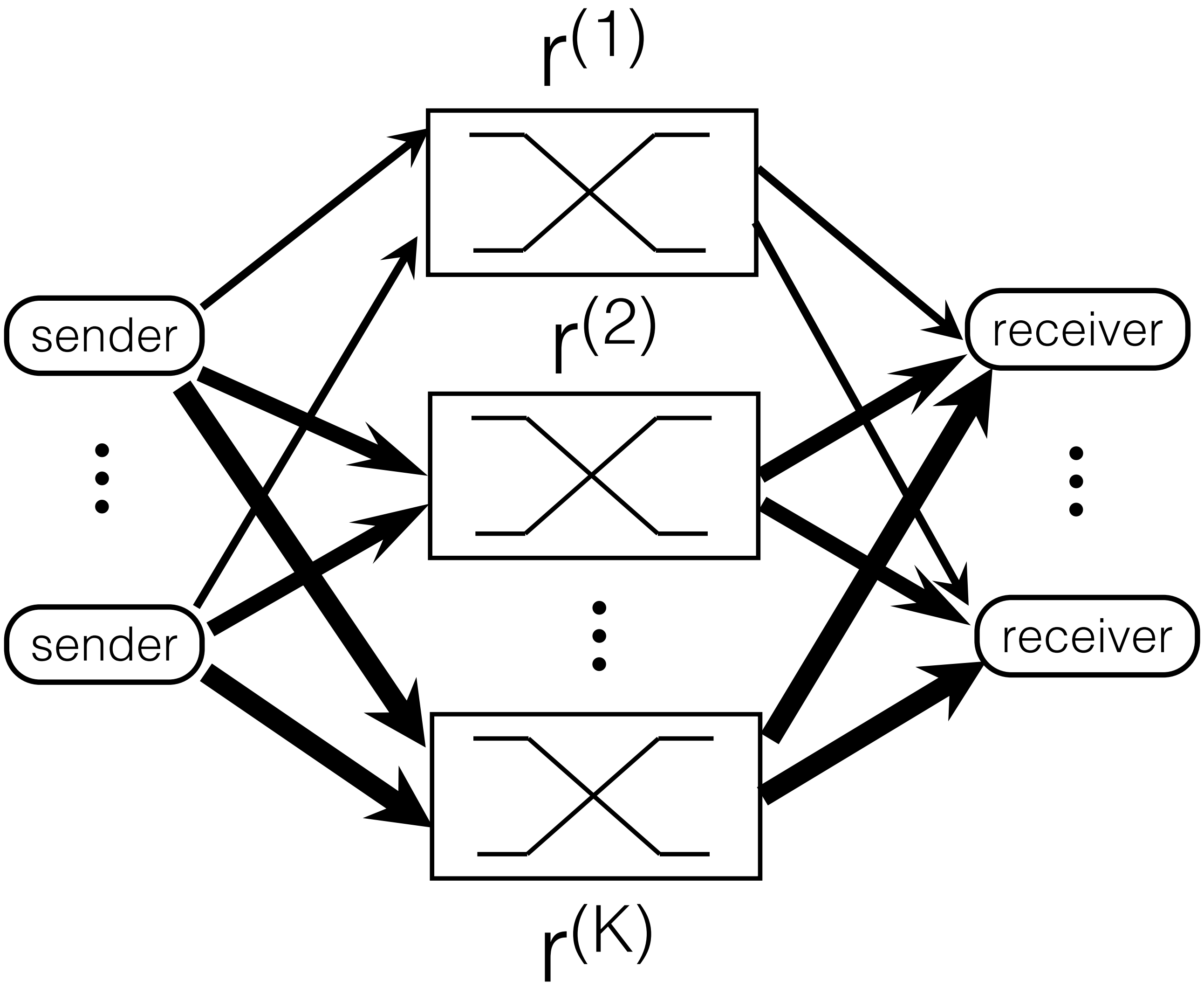
# Exploit the Shrinking Gap with Heterogeneous Parallel Networks

Because of the shrinking gap, a legacy network can still service a considerable amount of traffic relative to a new network.

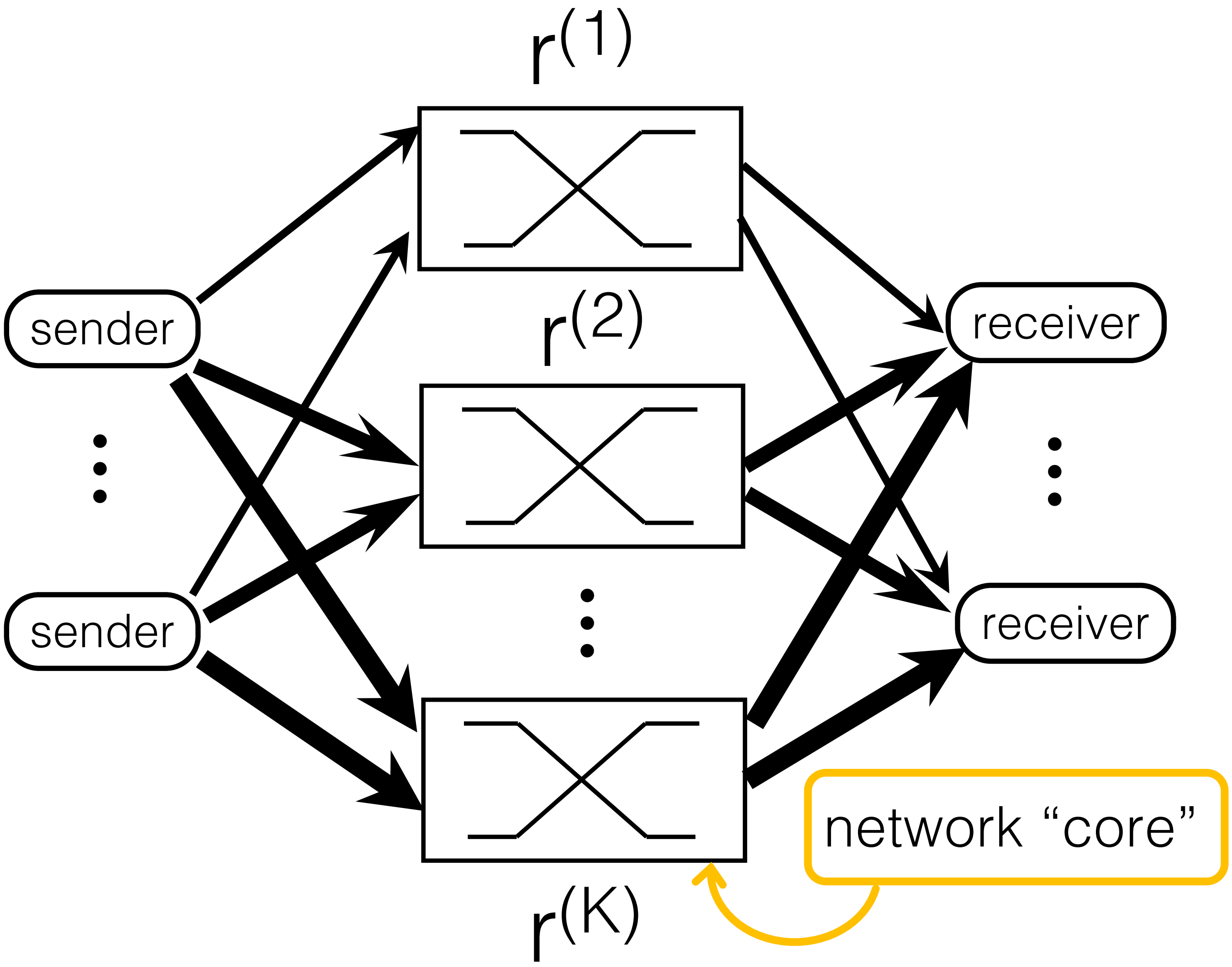
~30% of traffic!



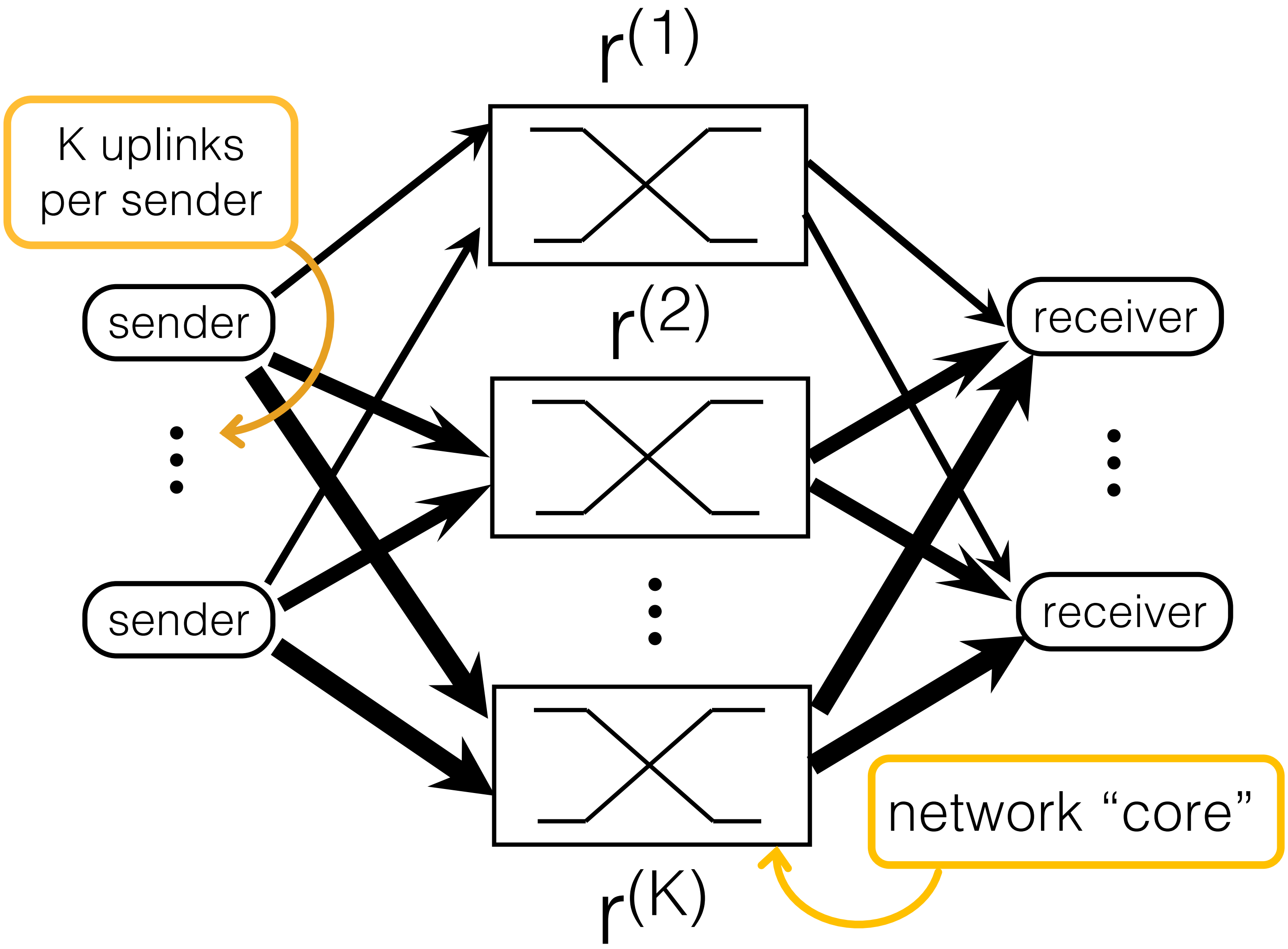
# HPNs:
# Heterogeneous Parallel Networks



# HPNs:
# Heterogeneous Parallel Networks



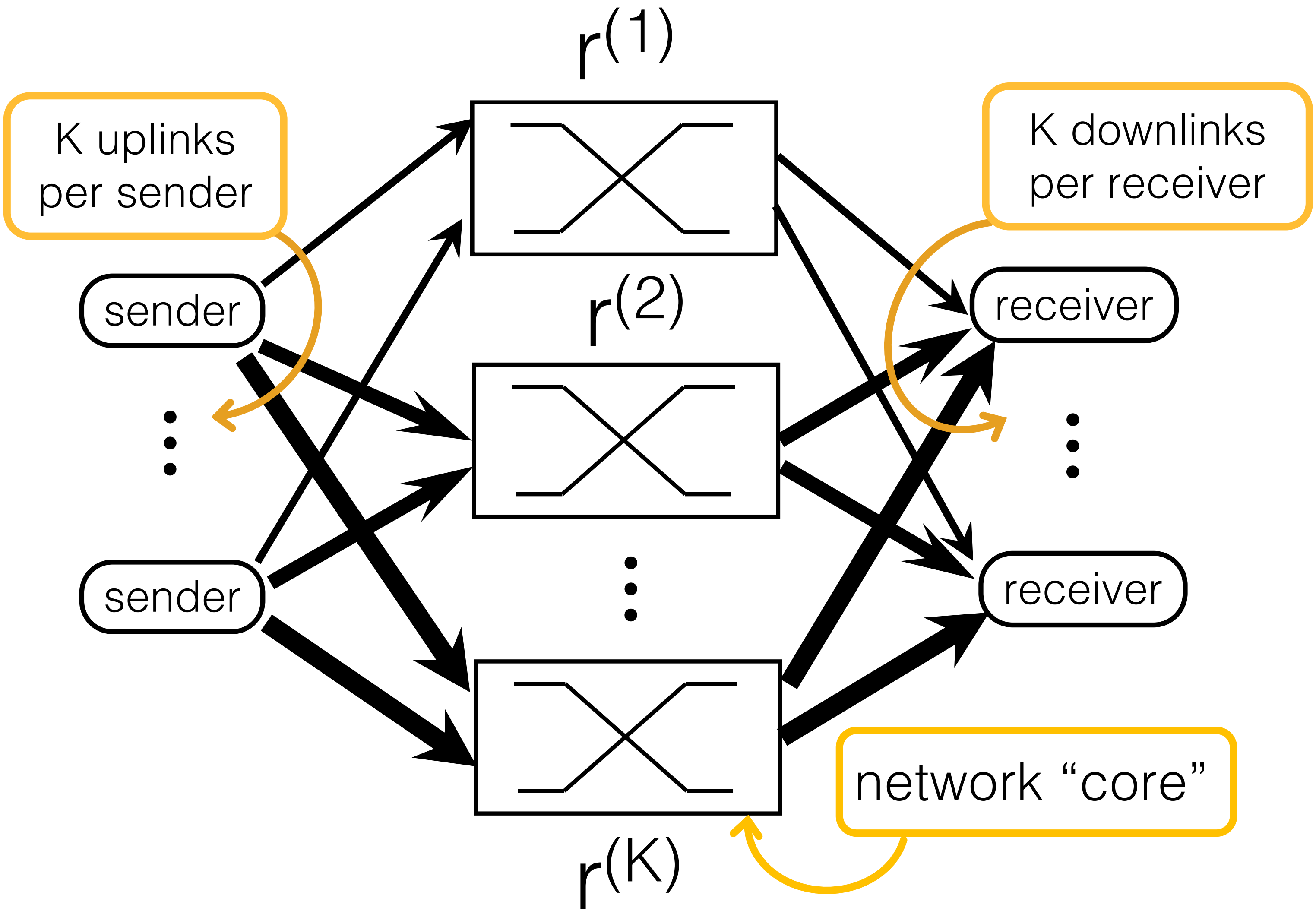
# HPNs:
# Heterogeneous Parallel Networks



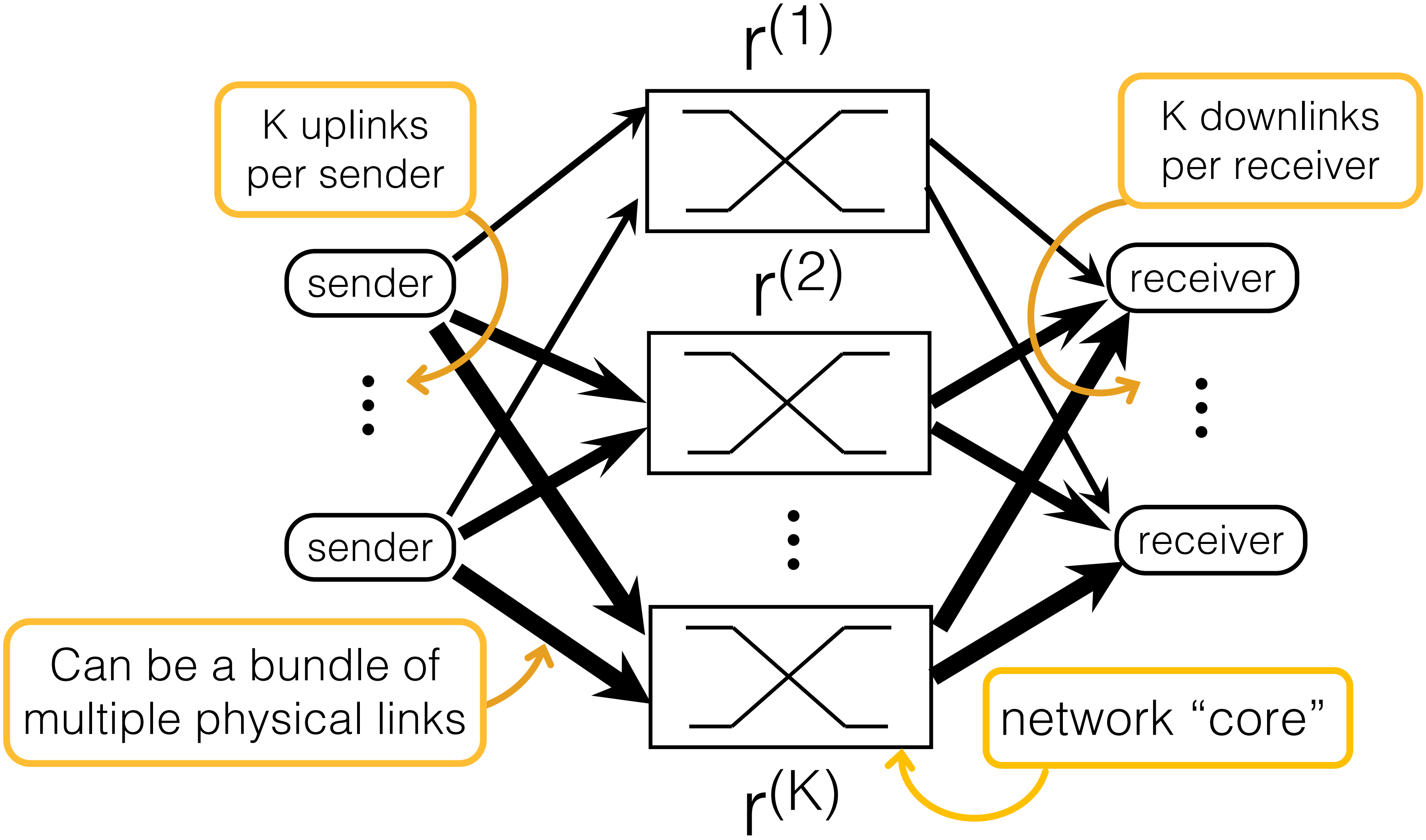
K uplinks per sender

network “core”

# HPNs: Heterogeneous Parallel Networks



# HPNs:
# Heterogeneous Parallel Networks



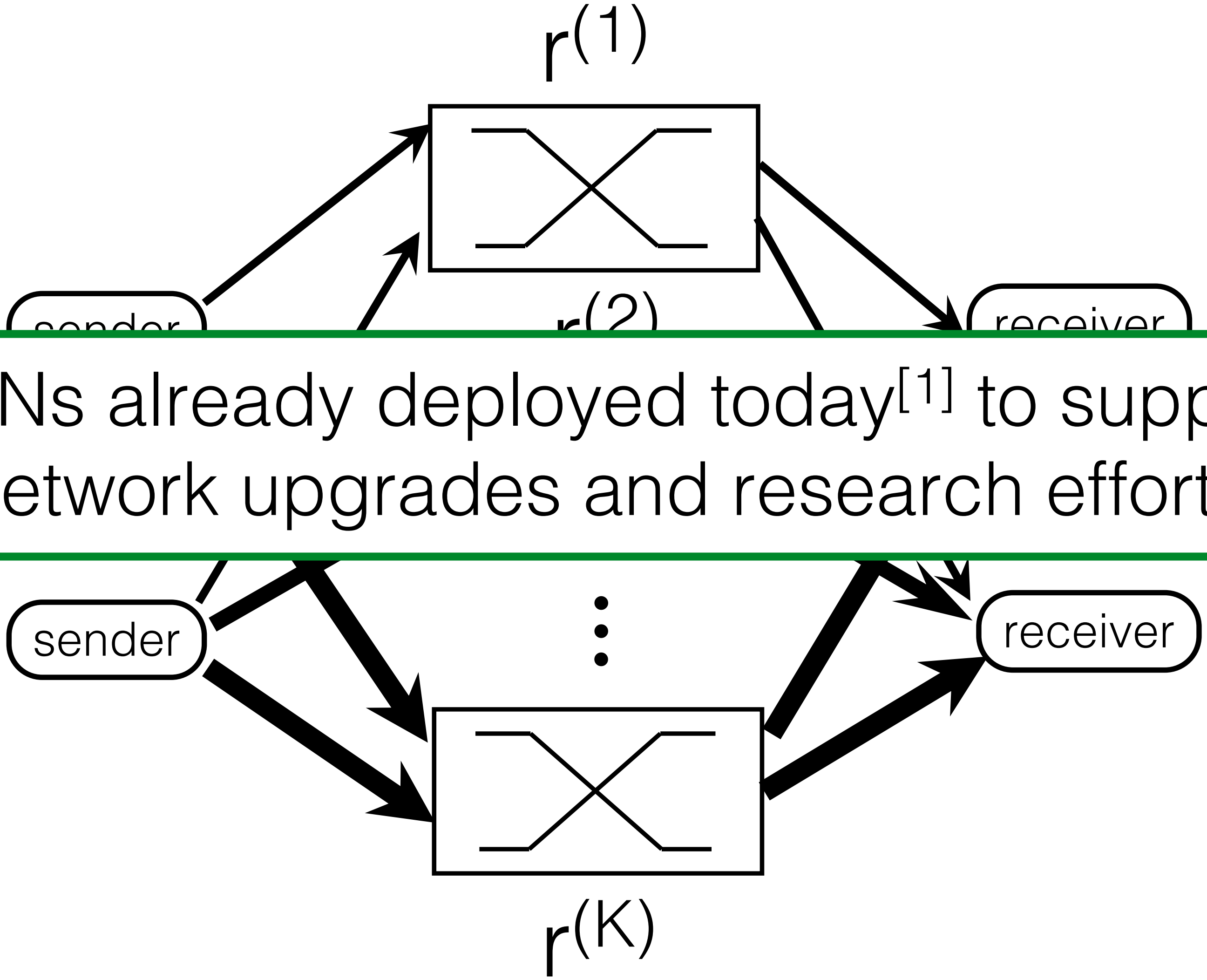
K uplinks per sender

K downlinks per receiver

Can be a bundle of multiple physical links

network “core”

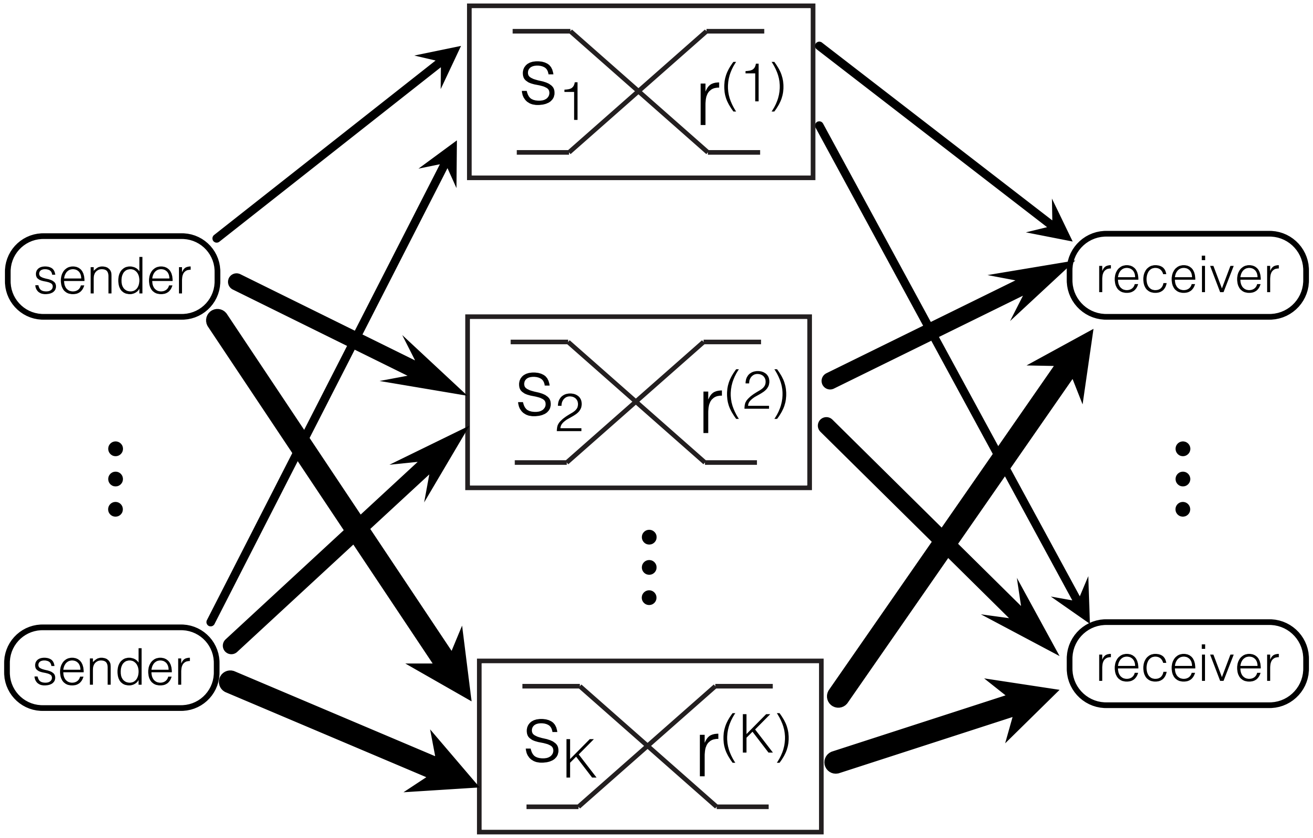
# HPNs:
# Heterogeneous Parallel Networks



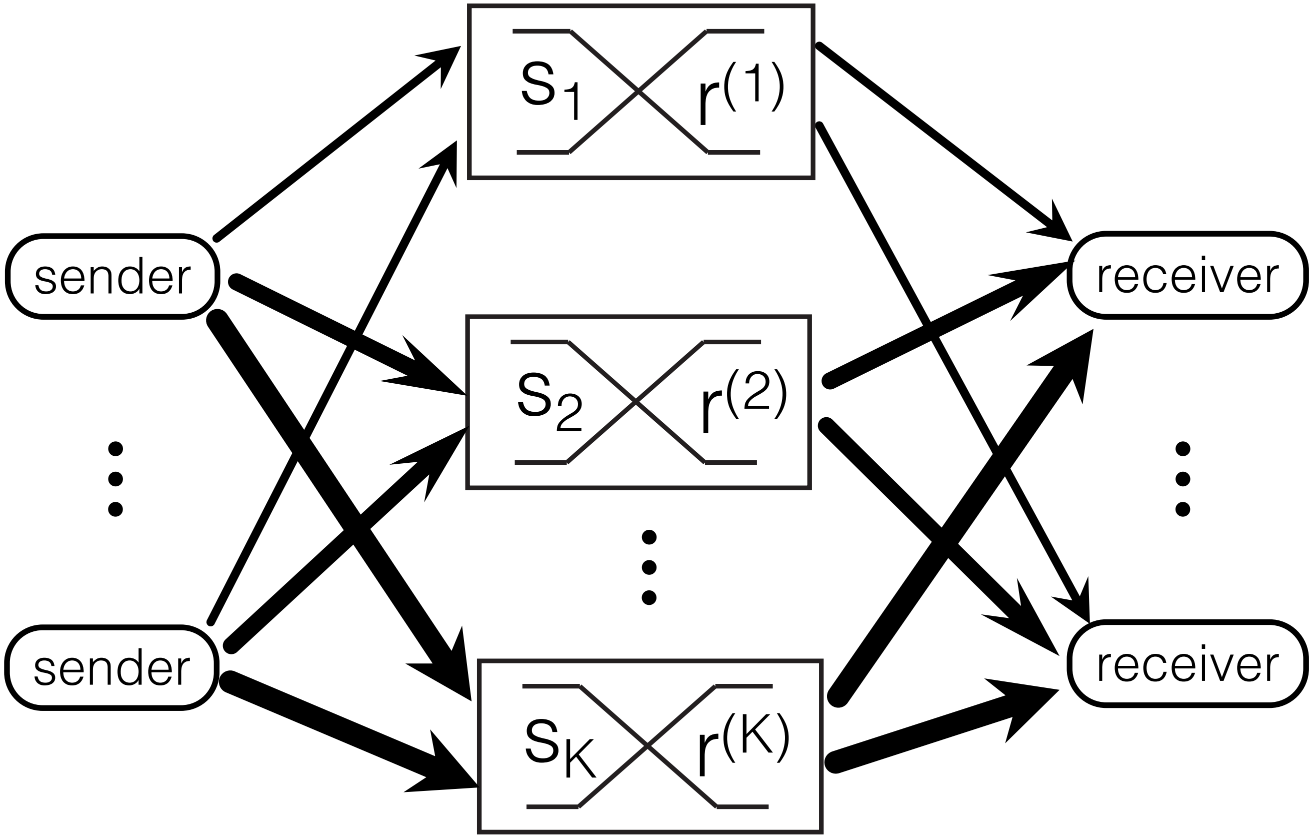
HPNs already deployed today[1] to support network upgrades and research efforts

[1] Singh, A. et al. Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network. (SIGCOMM '15)

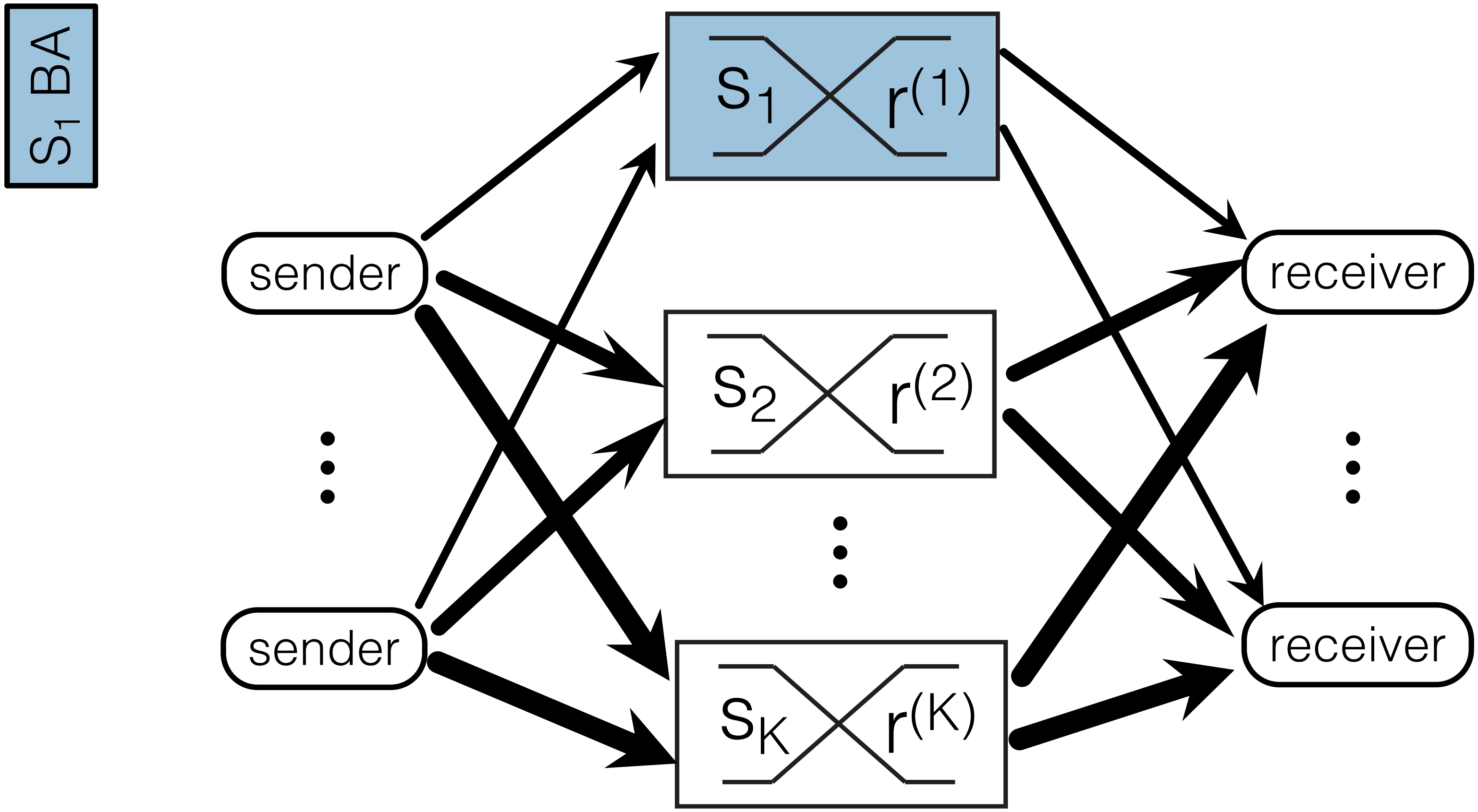
# Weaver:



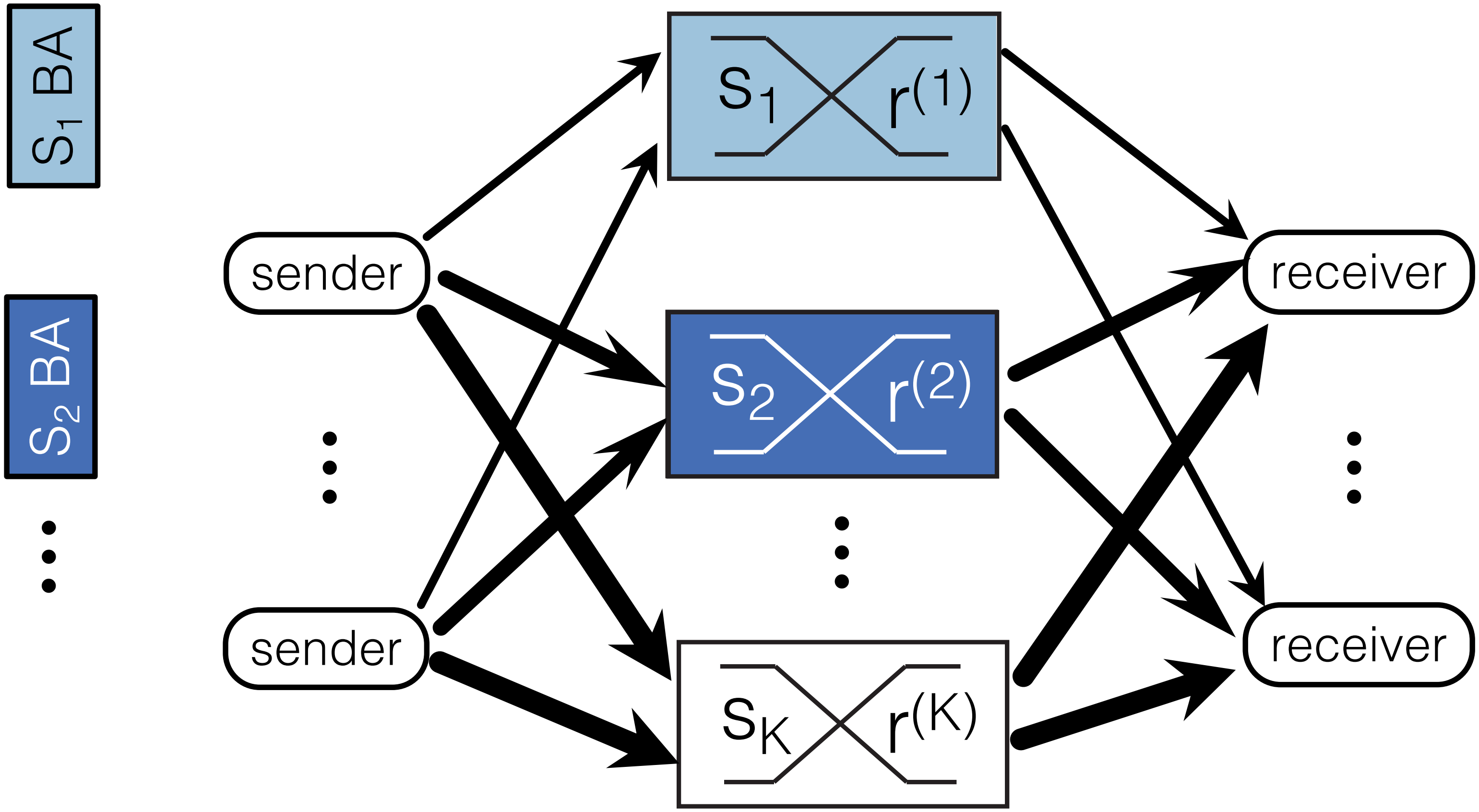
# Weaver: Bandwidth Allocation (BA)



# Weaver: Bandwidth Allocation (BA)



# Weaver: Bandwidth Allocation (BA)



$s_1$ BA

$s_2$ BA

sender

sender

$s_1$ $r^{(1)}$

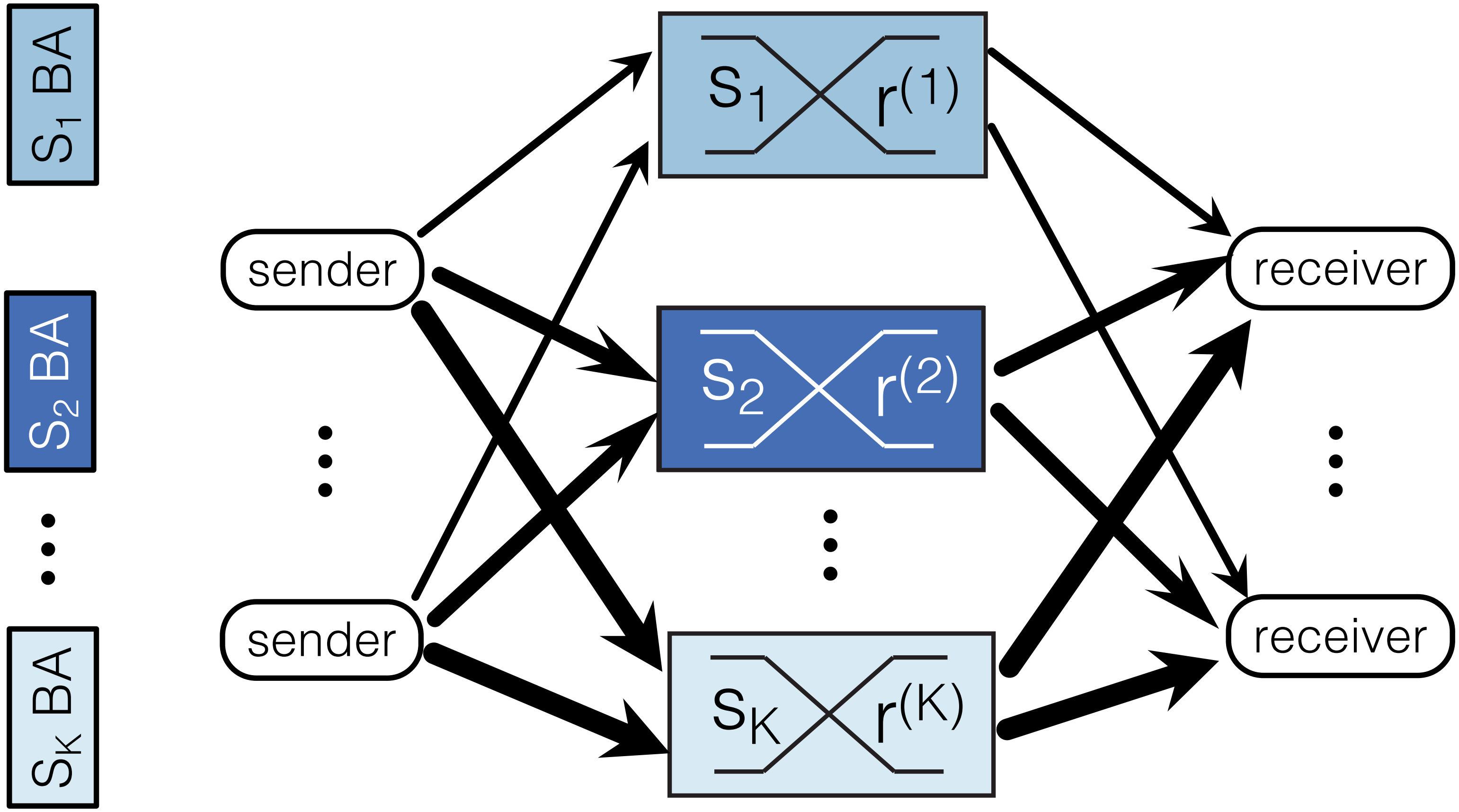
$s_2$ $r^{(2)}$

$s_K$ $r^{(K)}$

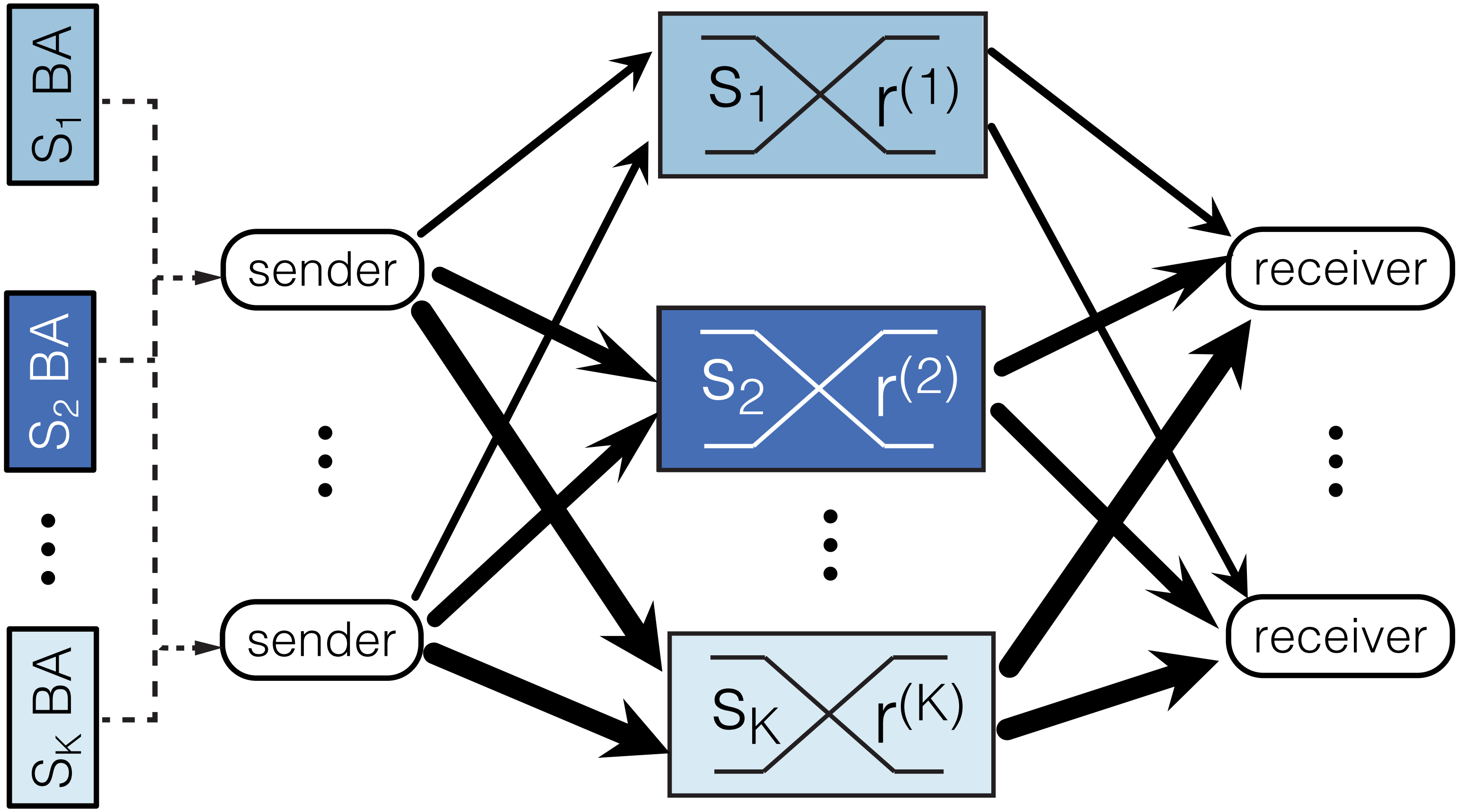
receiver

receiver

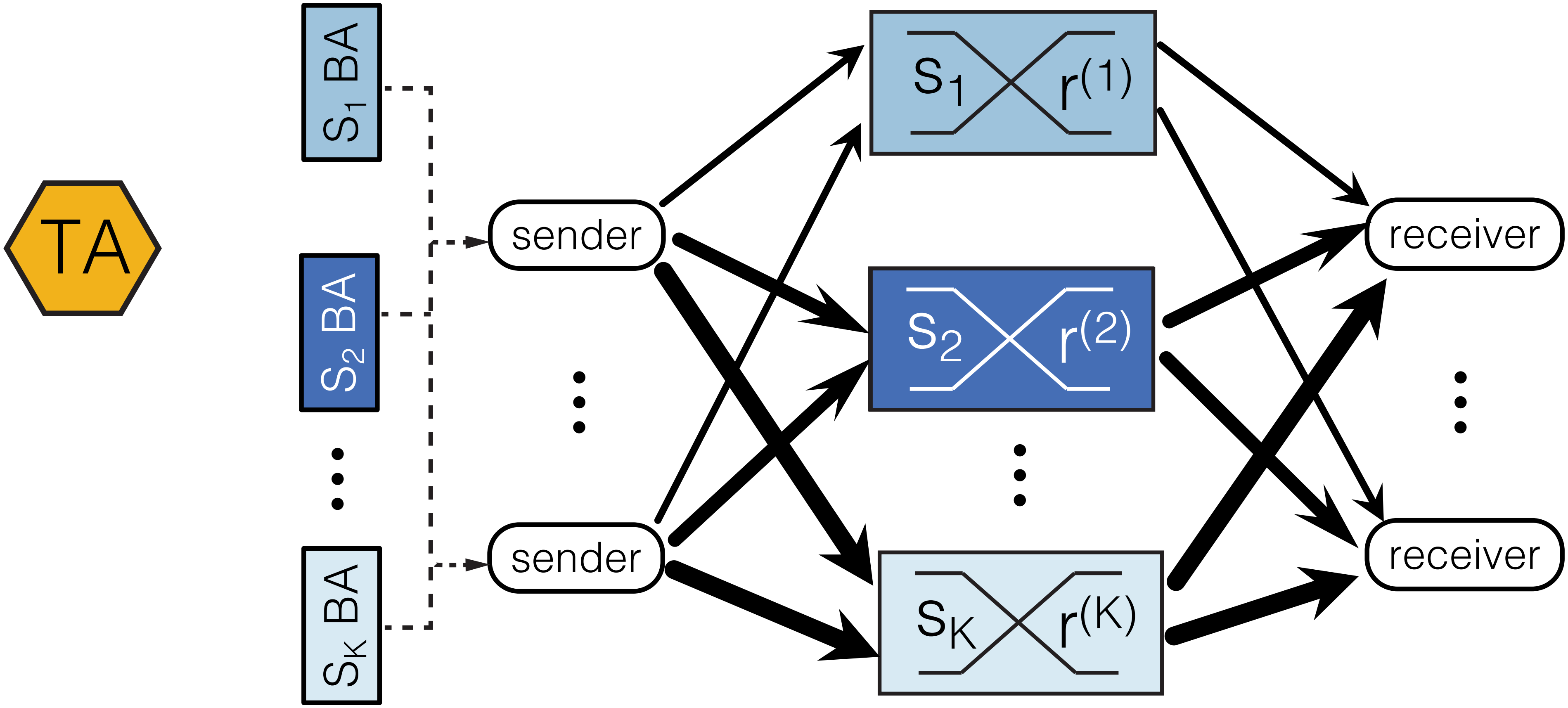
# Weaver: Bandwidth Allocation (BA)



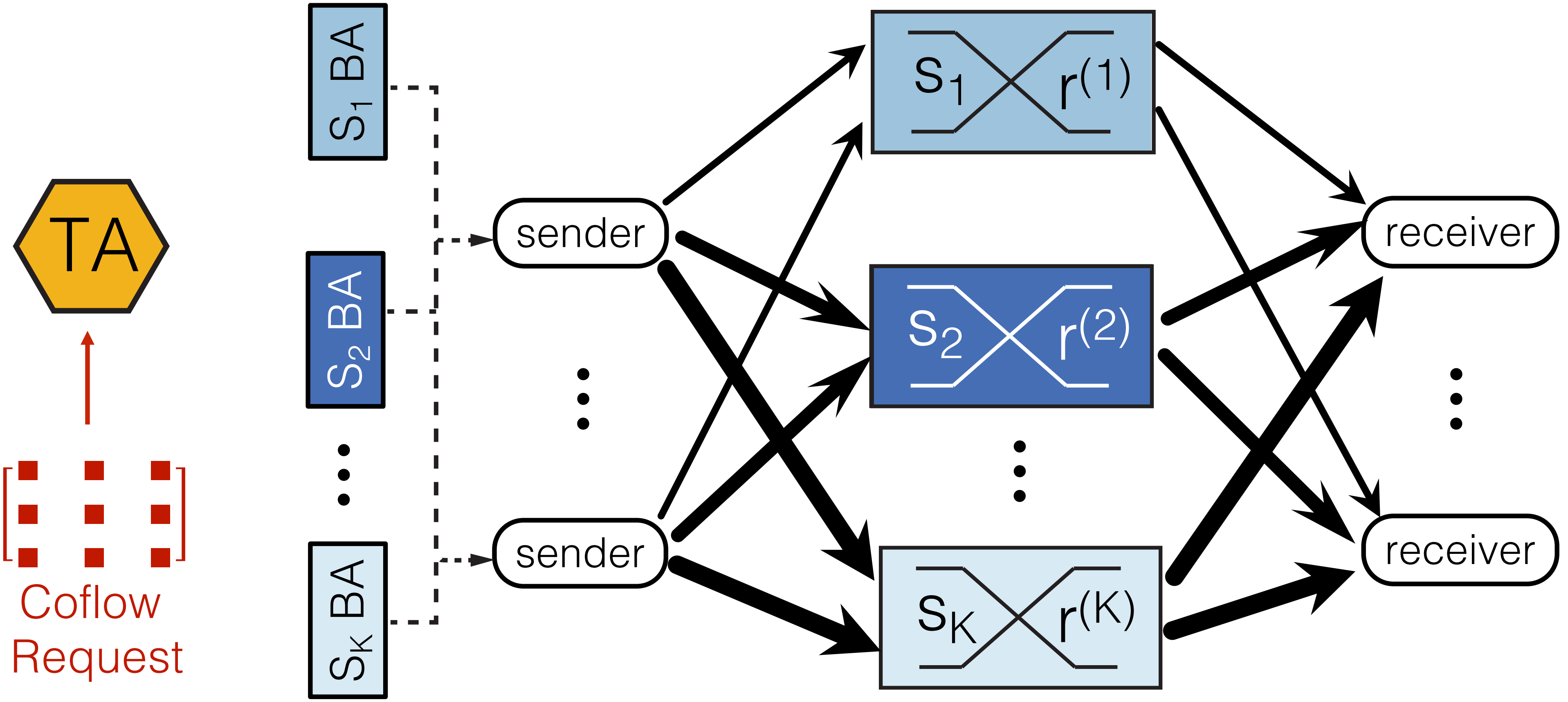
# Weaver: Bandwidth Allocation (BA)



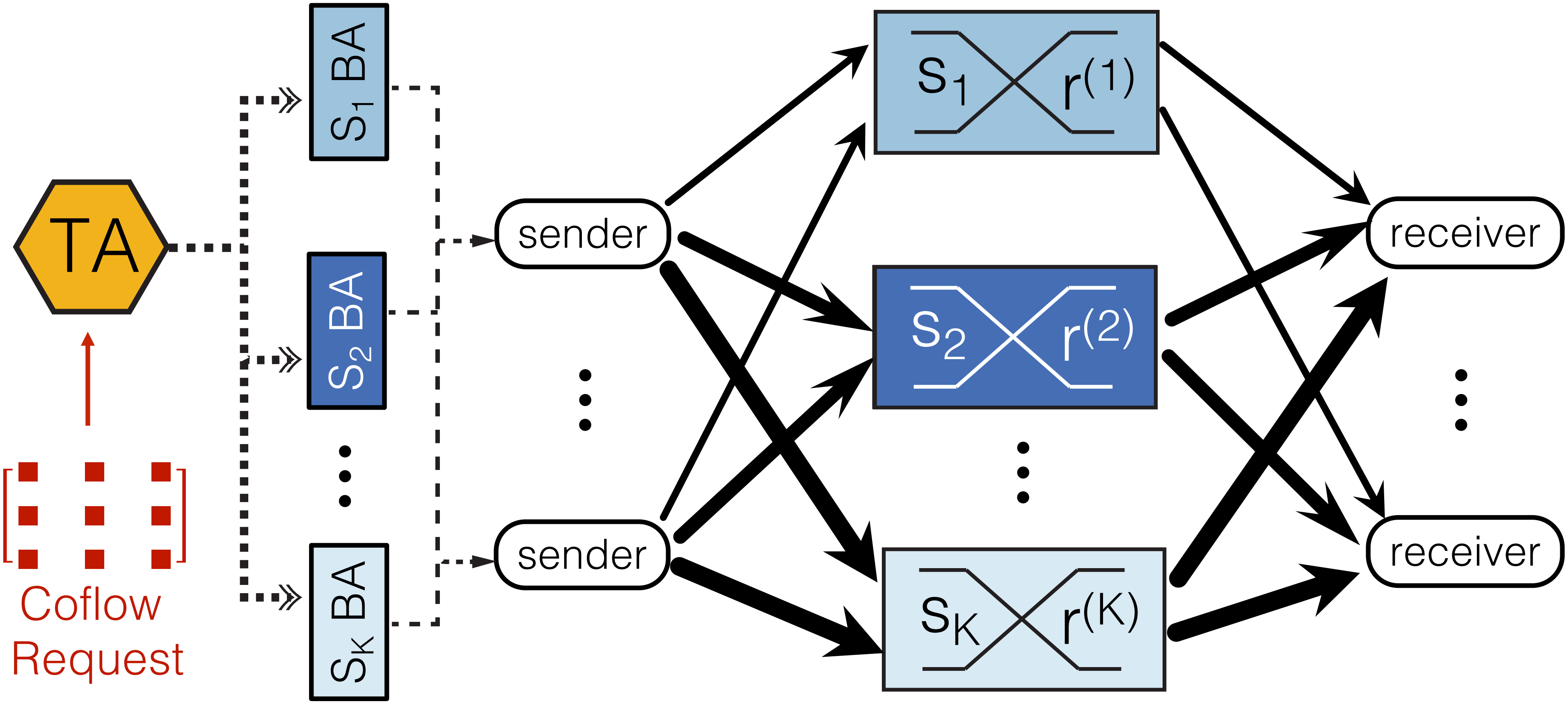
# Weaver: Bandwidth Allocation (BA) and Traffic Assignment (TA)



# Weaver: Bandwidth Allocation (BA) and Traffic Assignment (TA)

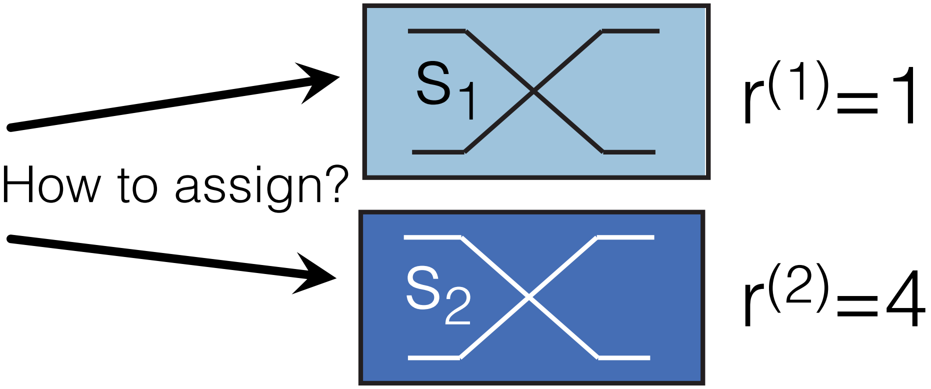


# Weaver: Bandwidth Allocation (BA) and Traffic Assignment (TA)



# Weaver’s TA Algorithm

|  | out.4 | 5 | 6 |
|---|---|---|---|
| in.1 | 90 | 10 | 10 |
| 2 | 90 |  | 5 |
| 3 | 90 |  |  |

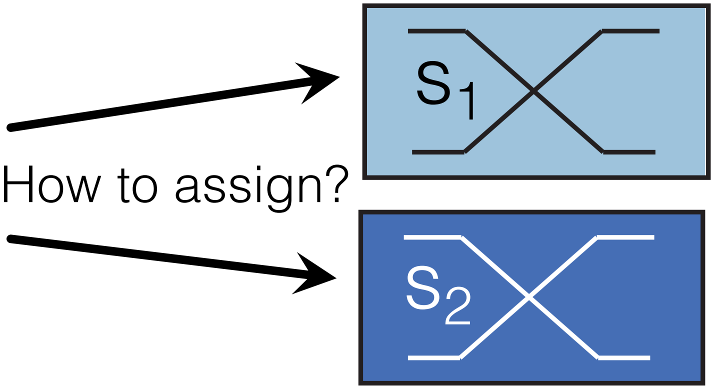


How to assign?

Refer to our paper for a detailed description on this example and the Weaver’s TA algorithm.

# Weaver’s TA Algorithm

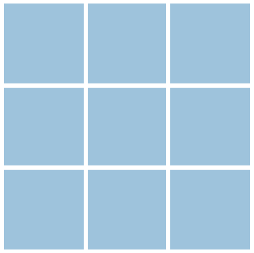
|  | out.4 | 5 | 6 |
|---|---|---|---|
| in.1 | 90 | 10 | 10 |
| 2 | 90 |  | 5 |
| 3 | 90 |  |  |



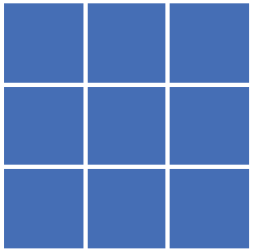
$r^{(1)}=1$

$r^{(2)}=4$

$r^{(1)}=1$



$r^{(2)}=4$

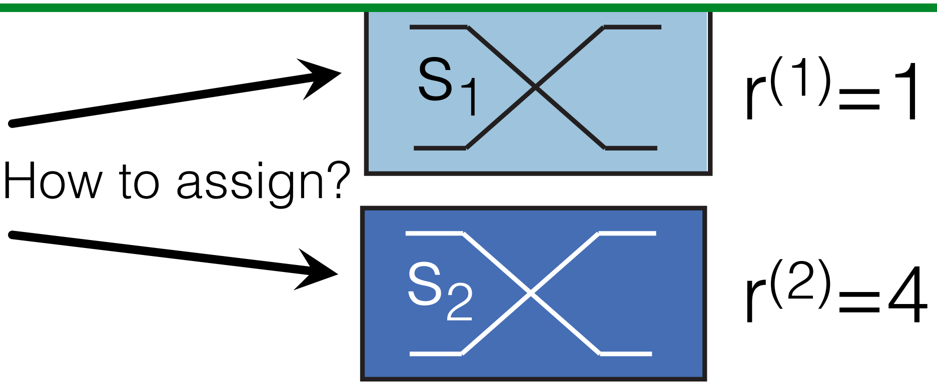


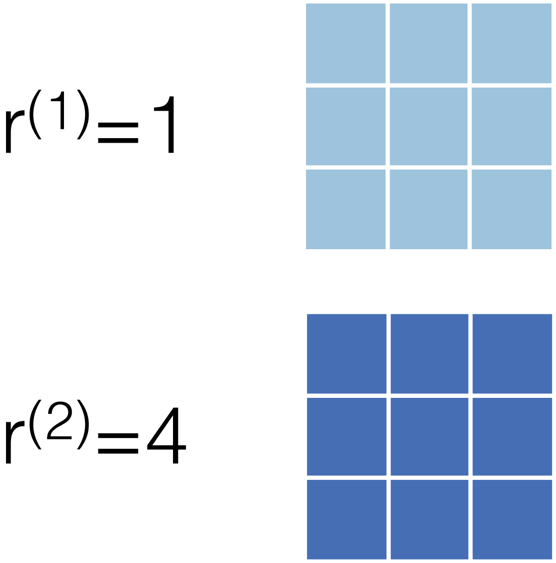
Refer to our paper for a detailed description on this example and the Weaver’s TA algorithm.

# Weaver's TA Algorithm

Start assignment from larger flows that are more likely to finish later and determine CCT

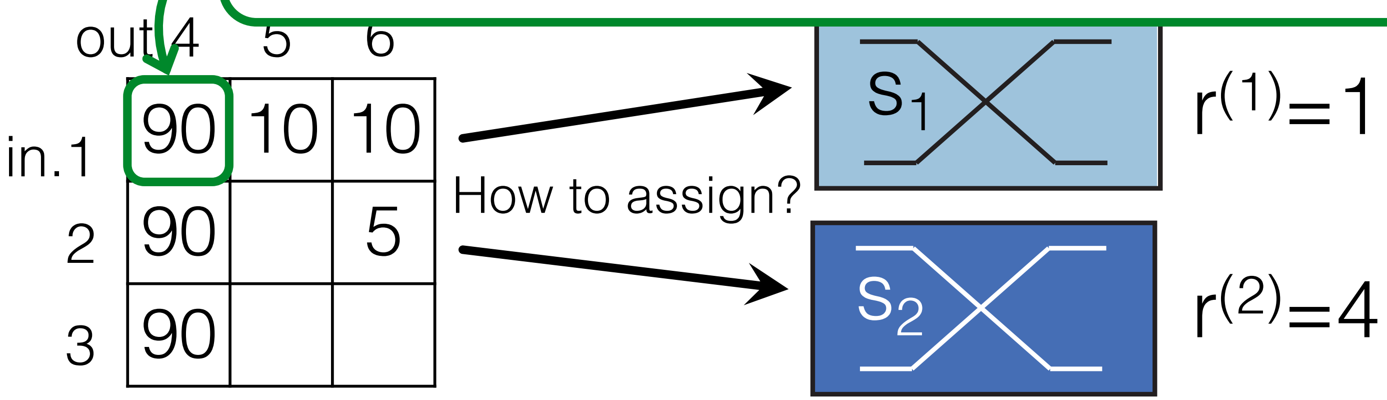
| | out 4 | 5 | 6 |
|---|---|---|---|
| in.1 | 90 | 10 | 10 |
| 2 | 90 | | 5 |
| 3 | 90 | | |





Refer to our paper for a detailed description on this example and the Weaver's TA algorithm.

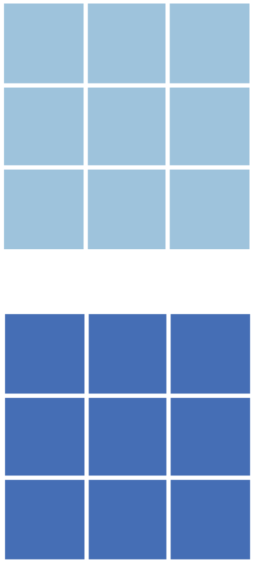
Start assignment from larger flows that are more likely to finish later and determine CCT



| out | 4 | 5 | 6 |
|---|---|---|---|
| in. 1 | 90 | 10 | 10 |
| 2 | 90 | | 5 |
| 3 | 90 | | |

$r^{(1)}=1$

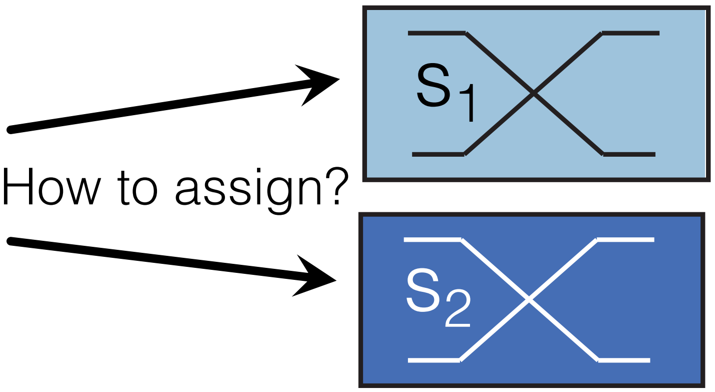
$r^{(2)}=4$



Refer to our paper for a detailed description on this example and the Weaver's TA algorithm.

# Weaver’s TA Algorithm

| | out.4 | 5 | 6 |
|---|---|---|---|
| in.1 | 90 | 10 | 10 |
| 2 | 90 | | 5 |
| 3 | 90 | | |

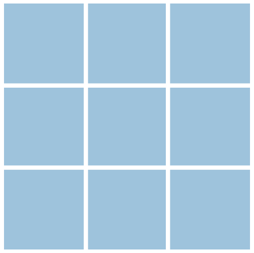


How to assign?

$r^{(1)}=1$

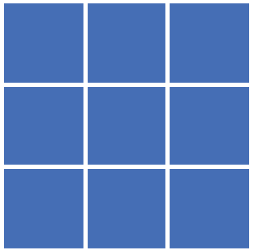
$r^{(2)}=4$

$r^{(1)}=1$



CCT = 90/1 = 90

$r^{(2)}=4$



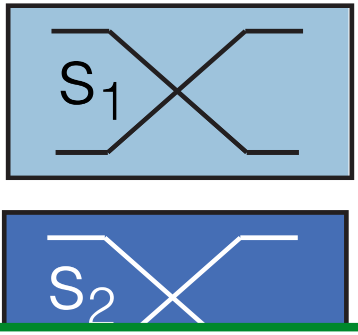
CCT = 90/4 = 22.5

Refer to our paper for a detailed description on this example and the Weaver’s TA algorithm.

# Weaver’s TA Algorithm

|  | out.4 | 5 | 6 |
|---|---|---|---|
| in.1 | 90 | 10 | 10 |
| 2 | 90 |  | 5 |
| 3 | 90 |  |  |

How to assign?

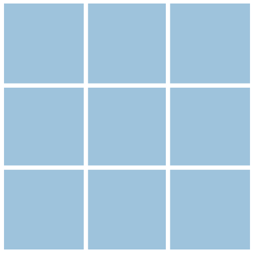


$r^{(1)}=1$

$r^{(2)}=4$

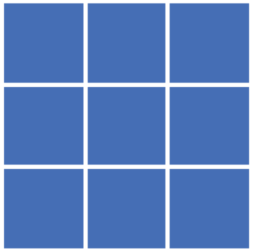
**Critical flow**: increases CCT on any network core after adding

$r^{(1)}=1$



CCT = 90/1 = 90

$r^{(2)}=4$



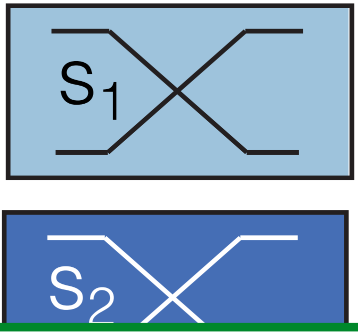
CCT = 90/4 = 22.5

Refer to our paper for a detailed description on this example and the Weaver’s TA algorithm.

# Weaver's TA Algorithm

|  | out.4 | 5 | 6 |
|---|---|---|---|
| in.1 | 90 | 10 | 10 |
| 2 | 90 |  | 5 |
| 3 | 90 |  |  |

How to assign?

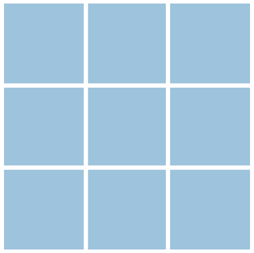


$r^{(1)}=1$

$r^{(2)}=4$

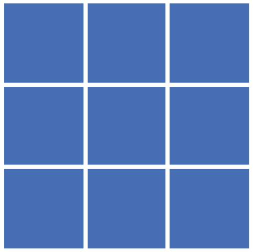
**Critical flow**: increases CCT on any network core after adding

$r^{(1)}=1$



CCT = 90/1 = 90

$r^{(2)}=4$



CCT = 90/4 = 22.5

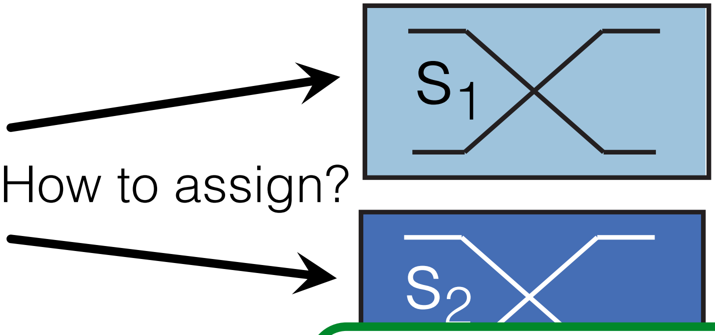
Assign critical flow to min CCT to obtain **optimality guarantee**

Refer to our paper for a detailed description on this example and the Weaver's TA algorithm.

# Weaver's TA Algorithm

| out. | 4 | 5 | 6 |
|---|---|---|---|
| in.1 | 90 | 10 | 10 |
| 2 | 90 | | 5 |
| 3 | 90 | | |

How to assign?

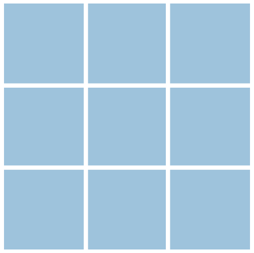


$r^{(1)}=1$

$r^{(2)}=4$

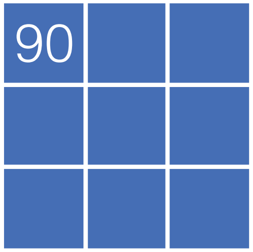
**Critical flow**: increases CCT on any network core after adding

$r^{(1)}=1$



CCT = 90/1 = 90

$r^{(2)}=4$



CCT = 90/4 = 22.5

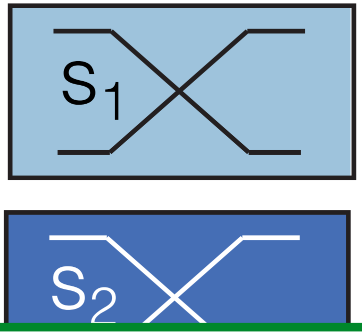
Assign critical flow to min CCT to obtain **optimality guarantee**

Refer to our paper for a detailed description on this example and the Weaver's TA algorithm.

# Weaver’s TA Algorithm

|  | out.4 | 5 | 6 |
|---|---|---|---|
| in.1 | 90 | 10 | 10 |
| 2 | 90 |  | 5 |
| 3 | 90 |  |  |

How to assign?

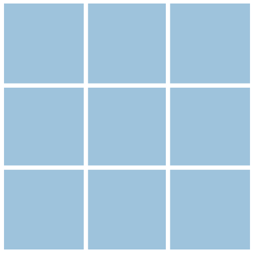


$r^{(1)}=1$

$r^{(2)}=4$

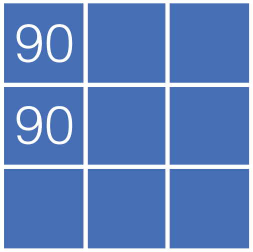
**Critical flow**: increases CCT on any network core after adding

$r^{(1)}=1$



CCT = 90/1 = 90

$r^{(2)}=4$



CCT = 180/4 = 45

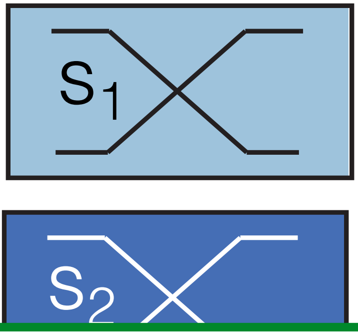
Assign critical flow to min CCT to obtain **optimality guarantee**

Refer to our paper for a detailed description on this example and the Weaver’s TA algorithm.

# Weaver’s TA Algorithm

|  | out.4 | 5 | 6 |
|---|---|---|---|
| in.1 | 90 | 10 | 10 |
| 2 | 90 |  | 5 |
| 3 | 90 |  |  |

How to assign?

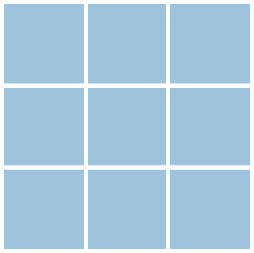


$r^{(1)}$=1

$r^{(2)}$=4

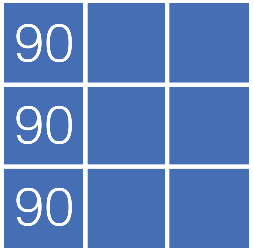
**Critical flow**: increases CCT on any network core after adding

$r^{(1)}$=1



CCT = 90/1 = 90

$r^{(2)}$=4

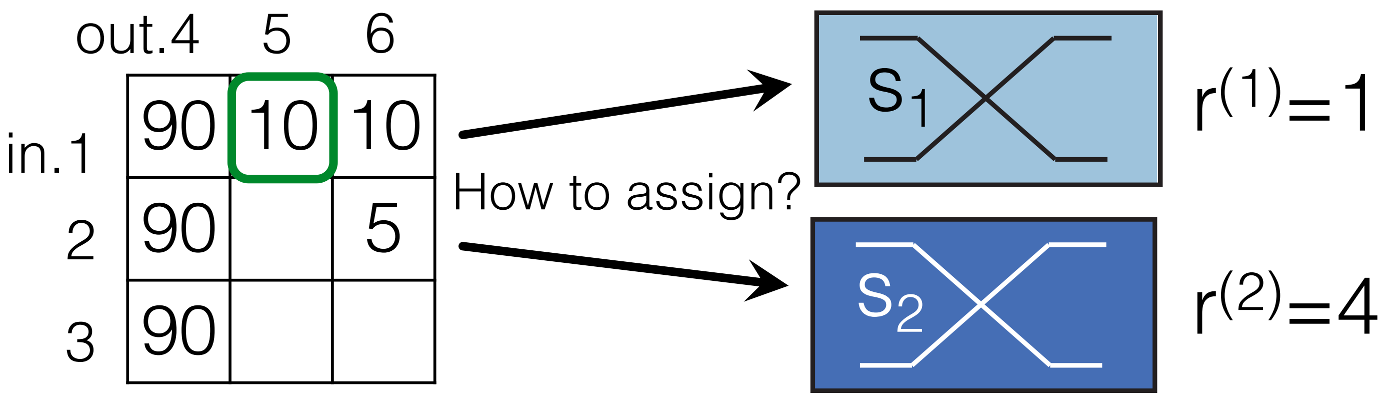


CCT = 270/4 = 67.5

Assign critical flow to min CCT to obtain **optimality guarantee**

Refer to our paper for a detailed description on this example and the Weaver’s TA algorithm.

# Weaver’s TA Algorithm

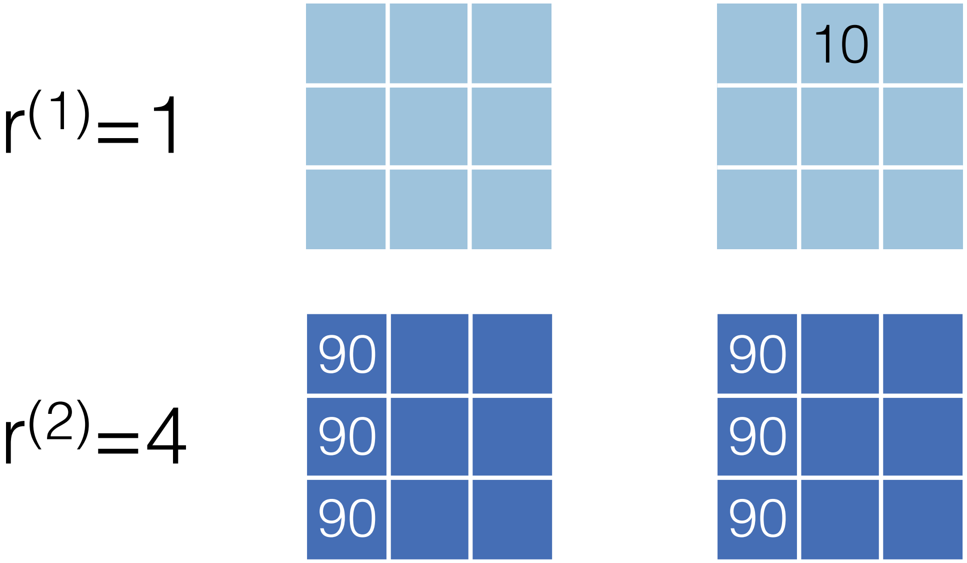


|  | out.4 | 5 | 6 |
|---|---|---|---|
| in.1 | 90 | 10 | 10 |
| 2 | 90 |  | 5 |
| 3 | 90 |  |  |

How to assign?

$r^{(1)}=1$

$r^{(2)}=4$



$r^{(1)}=1$

|  |  |  |
|---|---|---|
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
|---|---|---|
|  | 10 |  |
|  |  |  |
|  |  |  |

$r^{(2)}=4$

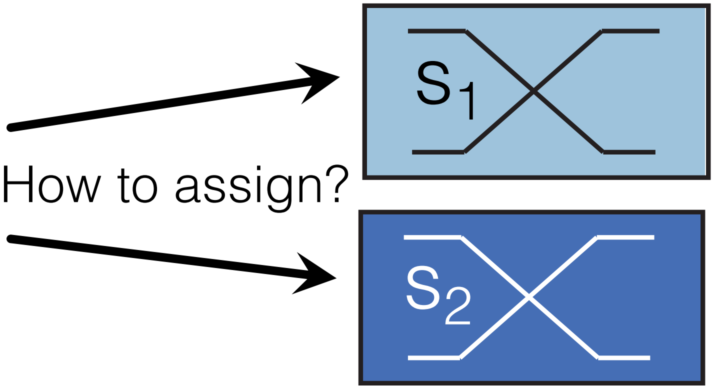
|  |  |  |
|---|---|---|
| 90 |  |  |
| 90 |  |  |
| 90 |  |  |

|  |  |  |
|---|---|---|
| 90 |  |  |
| 90 |  |  |
| 90 |  |  |

Refer to our paper for a detailed description on this example and the Weaver’s TA algorithm.

# Weaver's TA Algorithm

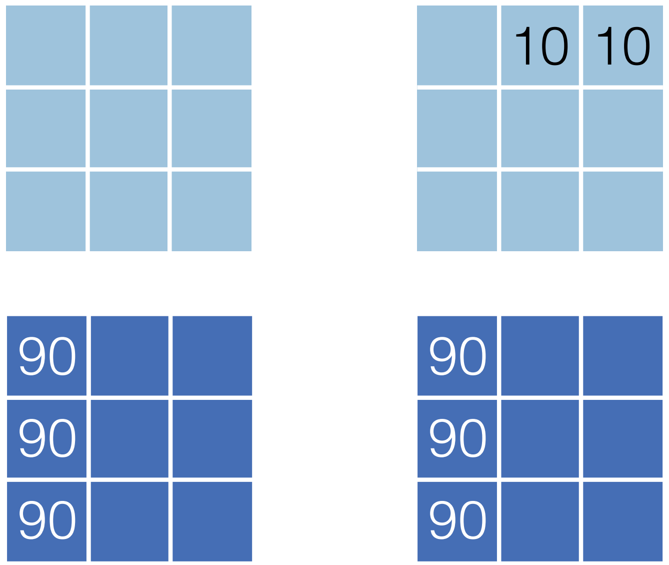
|  | out.4 | 5 | 6 |
|---|---|---|---|
| in.1 | 90 | 10 | 10 |
| 2 | 90 |  | 5 |
| 3 | 90 |  |  |



How to assign?

$r^{(1)}=1$

$r^{(2)}=4$



$r^{(1)}=1$

|  |  |  |
|---|---|---|
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
|---|---|---|
|  | 10 | 10 |
|  |  |  |
|  |  |  |

$r^{(2)}=4$

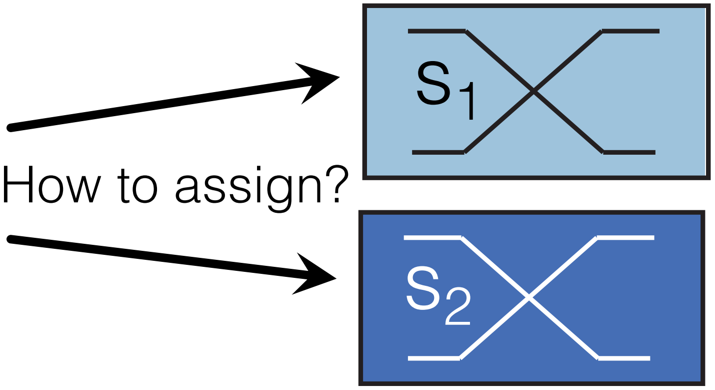
|  |  |  |
|---|---|---|
| 90 |  |  |
| 90 |  |  |
| 90 |  |  |

|  |  |  |
|---|---|---|
| 90 |  |  |
| 90 |  |  |
| 90 |  |  |

Refer to our paper for a detailed description on this example and the Weaver's TA algorithm.

# Weaver's TA Algorithm

|  | out.4 | 5 | 6 |
|---|---|---|---|
| in.1 | 90 | 10 | 10 |
| 2 | 90 |  | 5 |
| 3 | 90 |  |  |



How to assign?

$r^{(1)}=1$

$r^{(2)}=4$

$r^{(1)}=1$

|  |  |  |
|---|---|---|
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
|---|---|---|
|  | 10 | 10 |
|  |  |  |
|  |  |  |

$r^{(2)}=4$

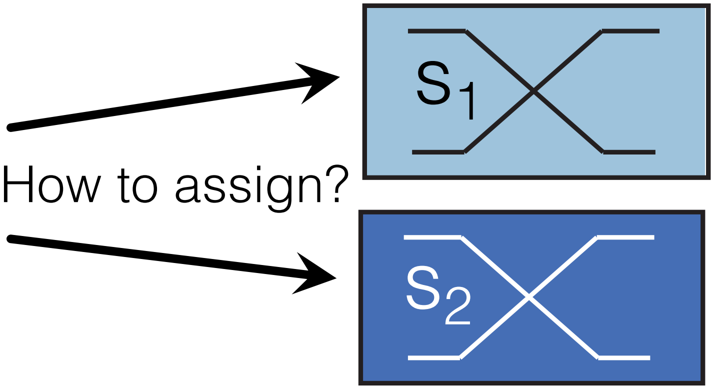
|  |  |  |
|---|---|---|
| 90 |  |  |
| 90 |  |  |
| 90 |  |  |

|  |  |  |
|---|---|---|
| 90 |  |  |
| 90 |  |  |
| 90 |  |  |

Refer to our paper for a detailed description on this example and the Weaver's TA algorithm.

# Weaver’s TA Algorithm

|  | out.4 | 5 | 6 |
|---|---|---|---|
| in.1 | 90 | 10 | 10 |
| 2 | 90 |  | 5 |
| 3 | 90 |  |  |



How to assign?

$r^{(1)}=1$

$r^{(2)}=4$

$r^{(1)}=1$

|  |  |  |
|---|---|---|
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
|---|---|---|
|  | 10 | 10 |
|  |  |  |
|  |  |  |

CCT determined by in.1

$r^{(2)}=4$

|  |  |  |
|---|---|---|
| 90 |  |  |
| 90 |  |  |
| 90 |  |  |

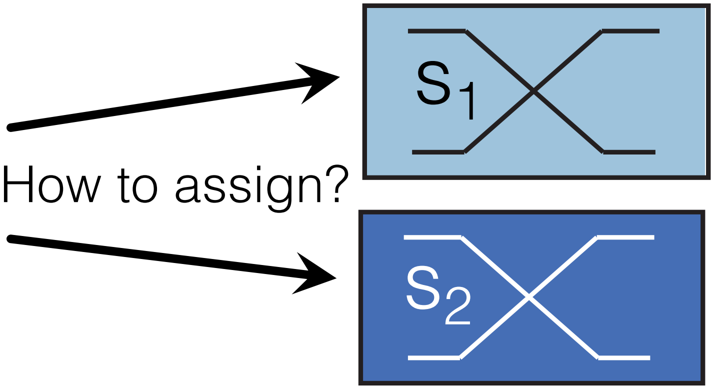
|  |  |  |
|---|---|---|
| 90 |  |  |
| 90 |  |  |
| 90 |  |  |

CCT determined by out.4

Refer to our paper for a detailed description on this example and the Weaver’s TA algorithm.

# Weaver’s TA Algorithm

|  | out.4 | 5 | 6 |
|---|---|---|---|
| in.1 | 90 | 10 | 10 |
| 2 | 90 |  | 5 |
| 3 | 90 |  |  |

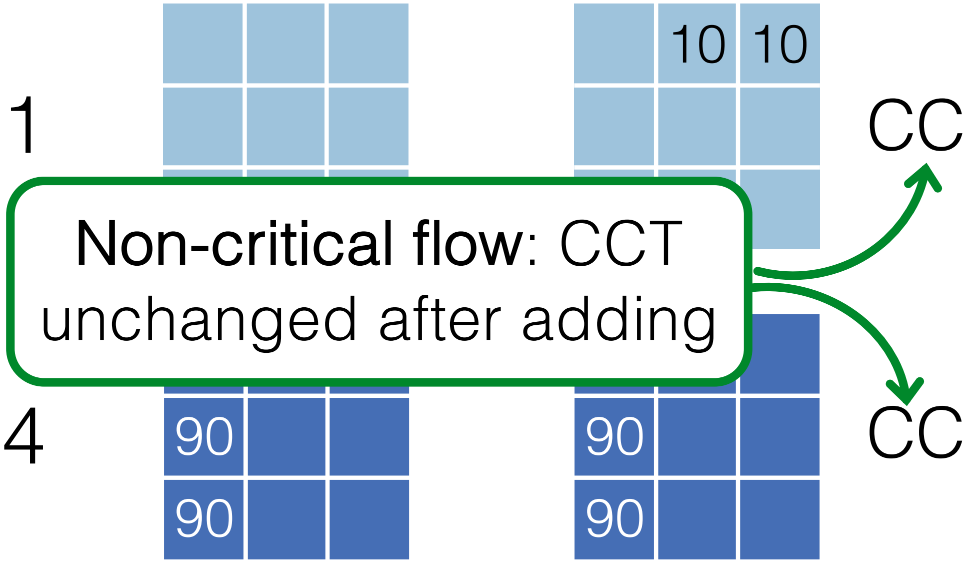

How to assign?

$r^{(1)}=1$

$r^{(2)}=4$

$r^{(1)}=1$

$r^{(2)}=4$



**Non-critical flow**: CCT unchanged after adding

CCT determined by in.1

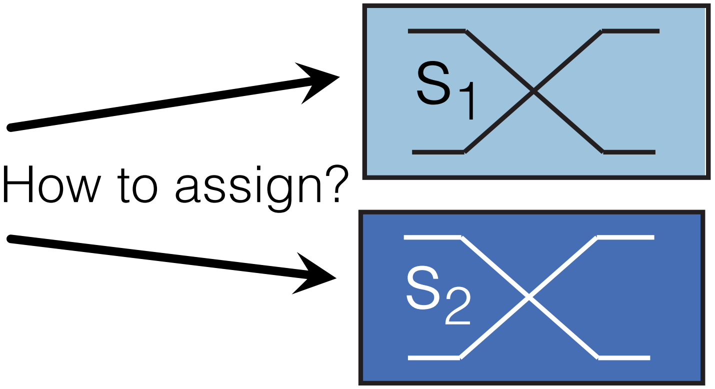
CCT determined by out.4

Refer to our paper for a detailed description on this example and the Weaver’s TA algorithm.

# Weaver’s TA Algorithm

|  | out.4 | 5 | 6 |
|---|---|---|---|
| in.1 | 90 | 10 | 10 |
| 2 | 90 |  | 5 |
| 3 | 90 |  |  |

How to assign?



$r^{(1)}=1$

$r^{(2)}=4$

$r^{(1)}=1$

|  |  |  |
|---|---|---|
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
|---|---|---|
|  | 10 | 10 |
|  |  |  |
|  |  |  |

FCT = (10 + 5)/1 =15

$r^{(2)}=4$

|  |  |  |
|---|---|---|
| 90 |  |  |
| 90 |  |  |
| 90 |  |  |

|  |  |  |
|---|---|---|
| 90 |  |  |
| 90 |  |  |
| 90 |  |  |

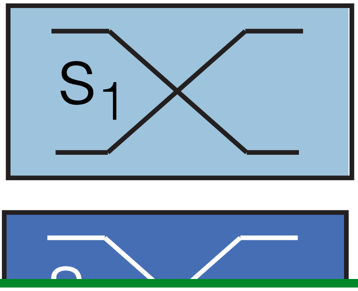
FCT = (90 + 5)/4 = 23.75

Refer to our paper for a detailed description on this example and the Weaver’s TA algorithm.

# Weaver’s TA Algorithm

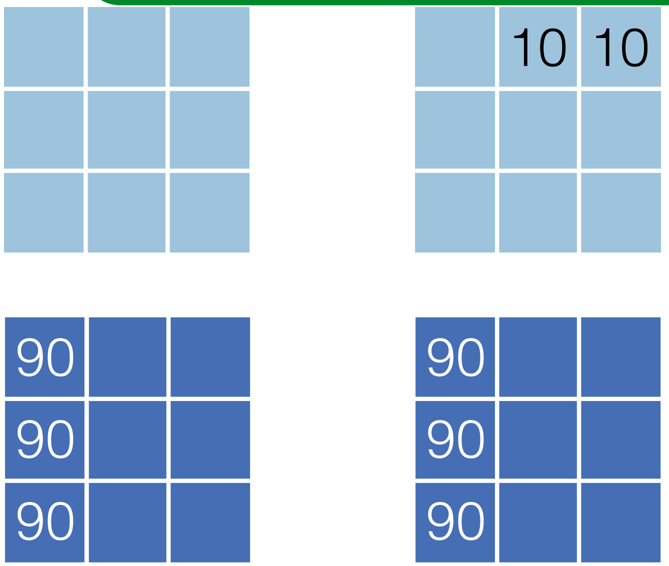
|  | out.4 | 5 | 6 |
|---|---|---|---|
| in.1 | 90 | 10 | 10 |
| 2 | 90 |  | 5 |
| 3 | 90 |  |  |

How to assign?



$r^{(1)}$=1

assign non-critical flow to less busy core on flow path to balance load



$r^{(1)}$=1

FCT = (10 + 5)/1 =15

$r^{(2)}$=4

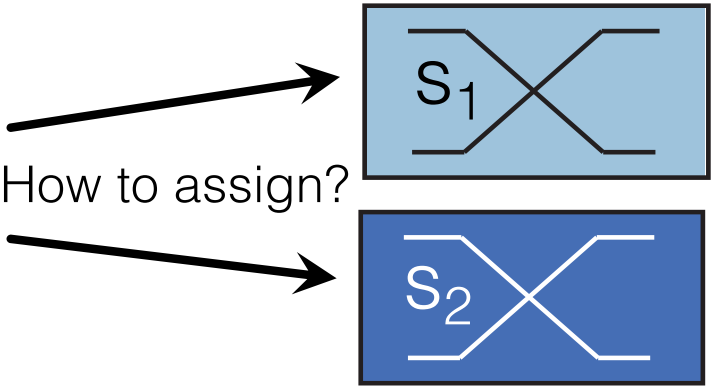
FCT = (90 + 5)/4 = 23.75

Refer to our paper for a detailed description on this example and the Weaver’s TA algorithm.

# Weaver's TA Algorithm

|  | out.4 | 5 | 6 |
|---|---|---|---|
| in.1 | 90 | 10 | 10 |
| 2 | 90 |  | 5 |
| 3 | 90 |  |  |

How to assign?



$r^{(1)}=1$

$r^{(2)}=4$

$r^{(1)}=1$

|  |  |  |
|---|---|---|
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
|---|---|---|
|  | 10 | 10 |
|  |  |  |
|  |  |  |

|  |  |  |
|---|---|---|
|  | 10 | 10 |
|  |  | 5 |
|  |  |  |

$r^{(2)}=4$

|  |  |  |
|---|---|---|
| 90 |  |  |
| 90 |  |  |
| 90 |  |  |

|  |  |  |
|---|---|---|
| 90 |  |  |
| 90 |  |  |
| 90 |  |  |

|  |  |  |
|---|---|---|
| 90 |  |  |
| 90 |  |  |
| 90 |  |  |

Refer to our paper for a detailed description on this example and the Weaver's TA algorithm.

# Weaver to manage Coflows in HPNs

| | |
|---|---|
| TA | • Optimality guarantee: within a constant factor of the optimal<br>  • By assigning critical flows to minimize CCT<br>• Further optimize assignment by …<br>  • Starting from larger flows<br>  • Assigning non-critical flows to balance load |
| BA | |

[1] Chowdhury, M. et al. Efficient coflow scheduling with Varys. (SIGCOMM'14)
[2] Chowdhury, M. et al. Efficient Coflow Scheduling Without Prior Knowledge. (SIGCOMM'15)

# Weaver to manage Coflows in HPNs

TA

- Optimality guarantee: within a constant factor of the optimal
  - By assigning critical flows to minimize CCT
- Further optimize assignment by …
  - Starting from larger flows
  - Assigning non-critical flows to balance load

BA

- Flexible framework to accommodate state-of-the-art Coflow scheduling policies to achieve the desired scheduling goal
  - Reuse state-of-the-art inter-Coflow schedulers for BAs
  - E.g. Varys[1] and Aalo[2], both designed to min avg CCT

[1] Chowdhury, M. et al. Efficient coflow scheduling with Varys. (SIGCOMM'14)

[2] Chowdhury, M. et al. Efficient Coflow Scheduling Without Prior Knowledge. (SIGCOMM'15)

# Evaluations

- [Simulations] Intra-Coflow TA efficiency
  - Weaver's TA has the best performance guarantee among competitive algorithms
- [Simulations] Inter-Coflow Scheduling (TA+BA)
  - Weaver achieves Coflow performance close to the ideal monolithic network.
  - Weaver improves TA by better assignment ordering
  - Weaver improves TA by load balancing non-critical flows
  - Weaver remains robust under different BA policies
- [Testbed] Inter-Coflow Scheduling (TA+BA)
  - Weaver achieves Coflow performance close to the ideal monolithic network

# Simulation setup

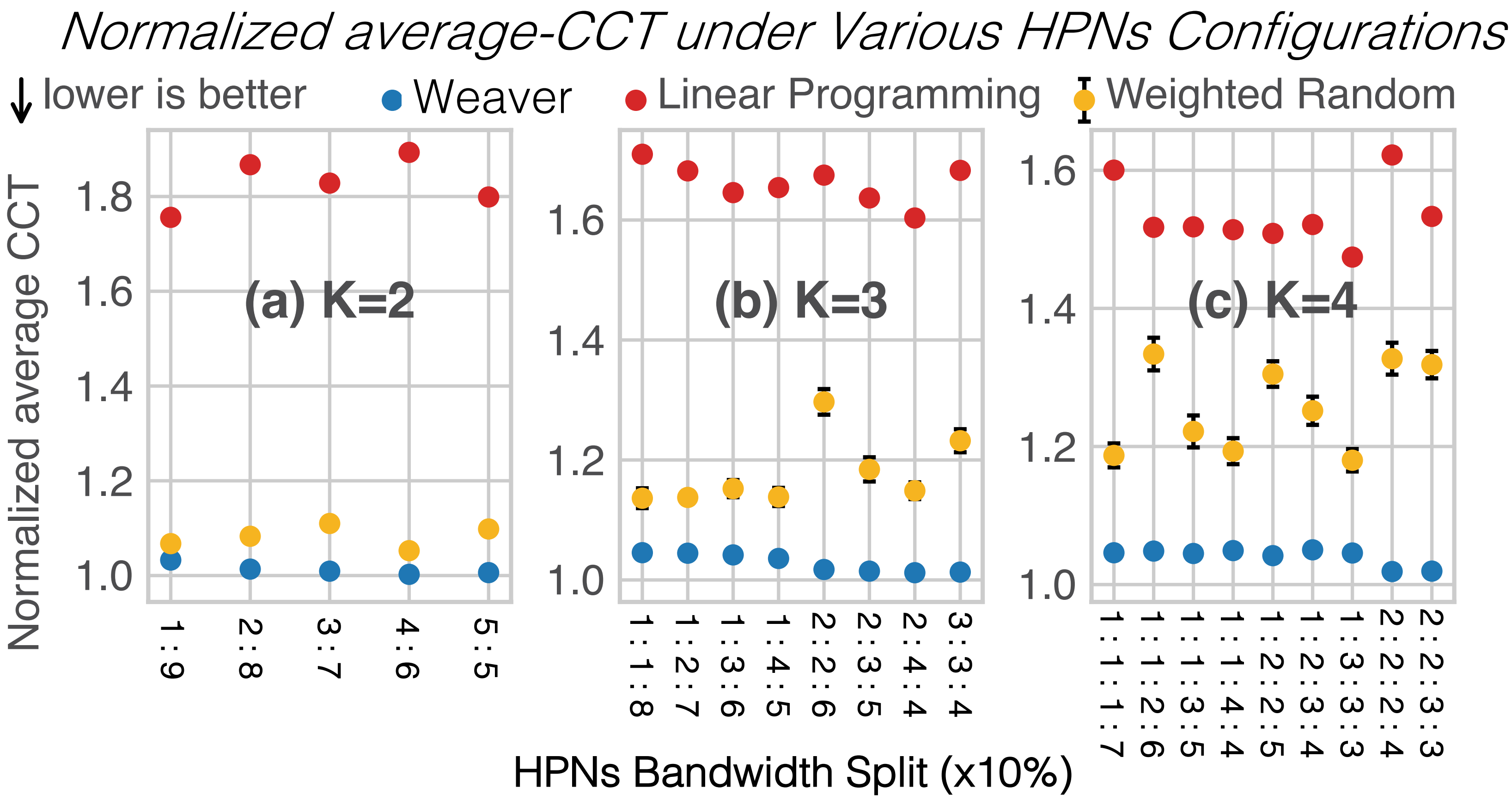
- Flow-level simulator and realistic Coflow trace
- Various HPNs configurations under K=2, 3, 4
  - Various bandwidth splits under each K
  - E.g. a 20%:80% split (K=2) is relevant for the 10G/40G HPNs
- Baseline: ideal monolithic network providing 100% bandwidth
- Scheduling schemes compared

| | TA | BA |
|---|---|---|
| Weaver | Weaver TA | Varys[1] |
| Weighted Random | Naïve Weighted  Random TA | Varys[1] |
| Rapier[2] | Linear Programming based Coflow scheduling in generic topology (Control both TA and BA) | |

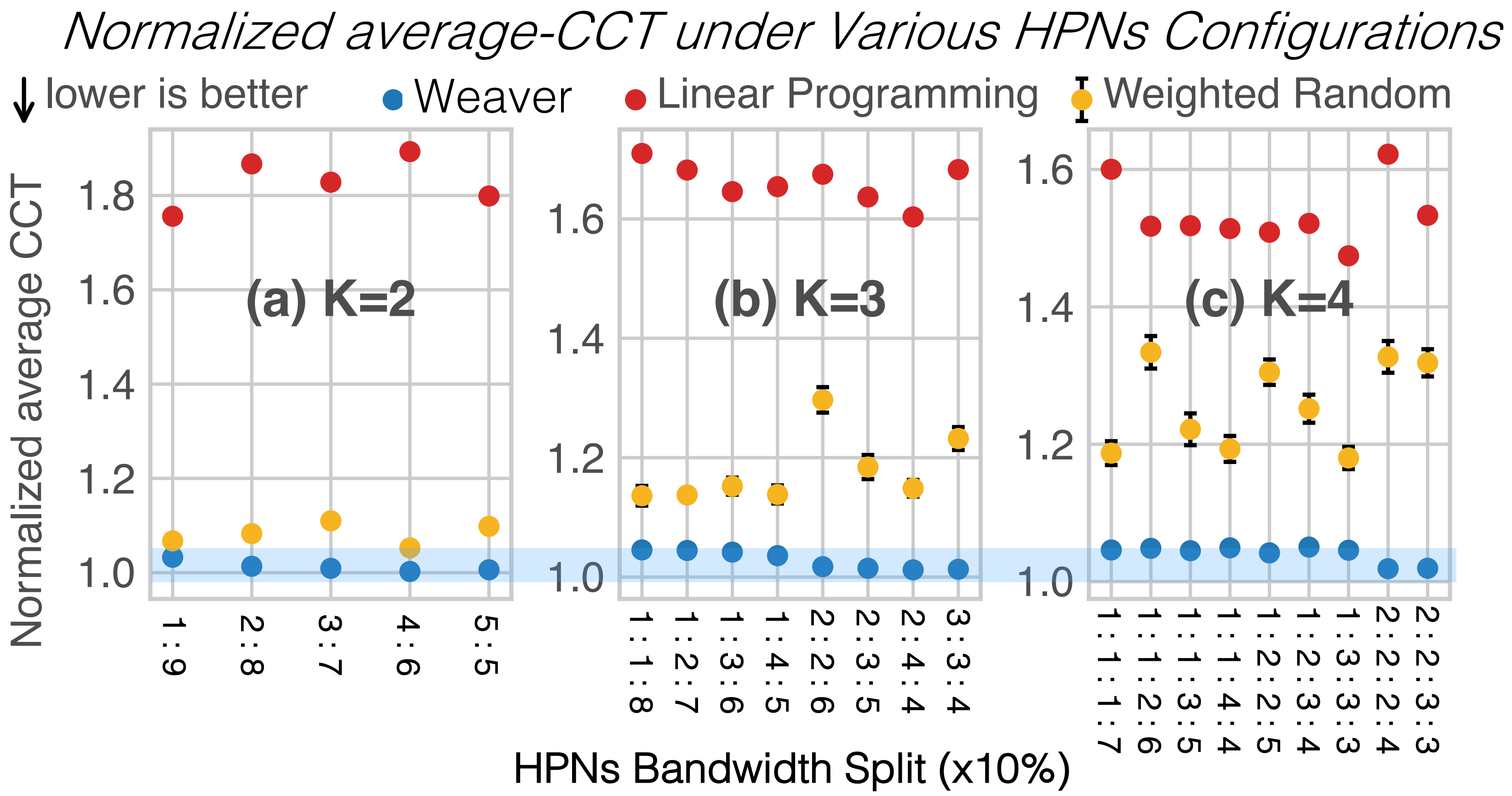
[1] Chowdhury, M. et al. Efficient coflow scheduling with Varys. (SIGCOMM'14)

[2] Zhao, Y. et al. Rapier: Integrating Routing and Scheduling for Coflow-Aware Data Center Networks (INFOCOM'15)

# Improvement in Average CCT



# Improvement in Average CCT



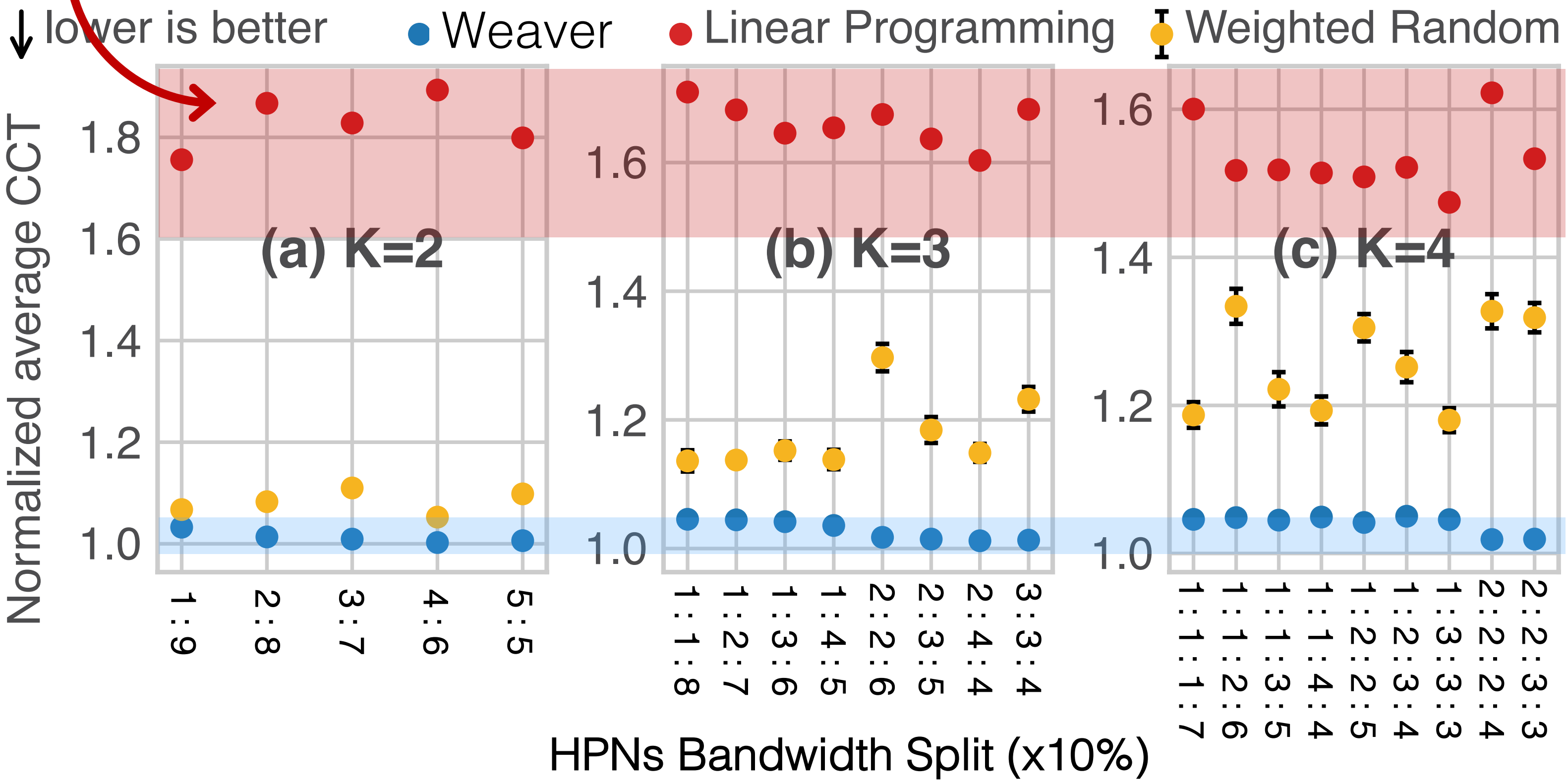
The Weaver-orchestrated HPNs achieve Coflow performance comparable to the monolithic network.

We have also validated the inter-Coflow scheduling efficiency with testbed experiments.
Our testbed results generally resemble those of simulations. See paper for details.

# ge CCT

LP-based Rapier: Less efficient TA algorithm and less efficient inter-Coflow scheduling in HPNs

*onfigurations*



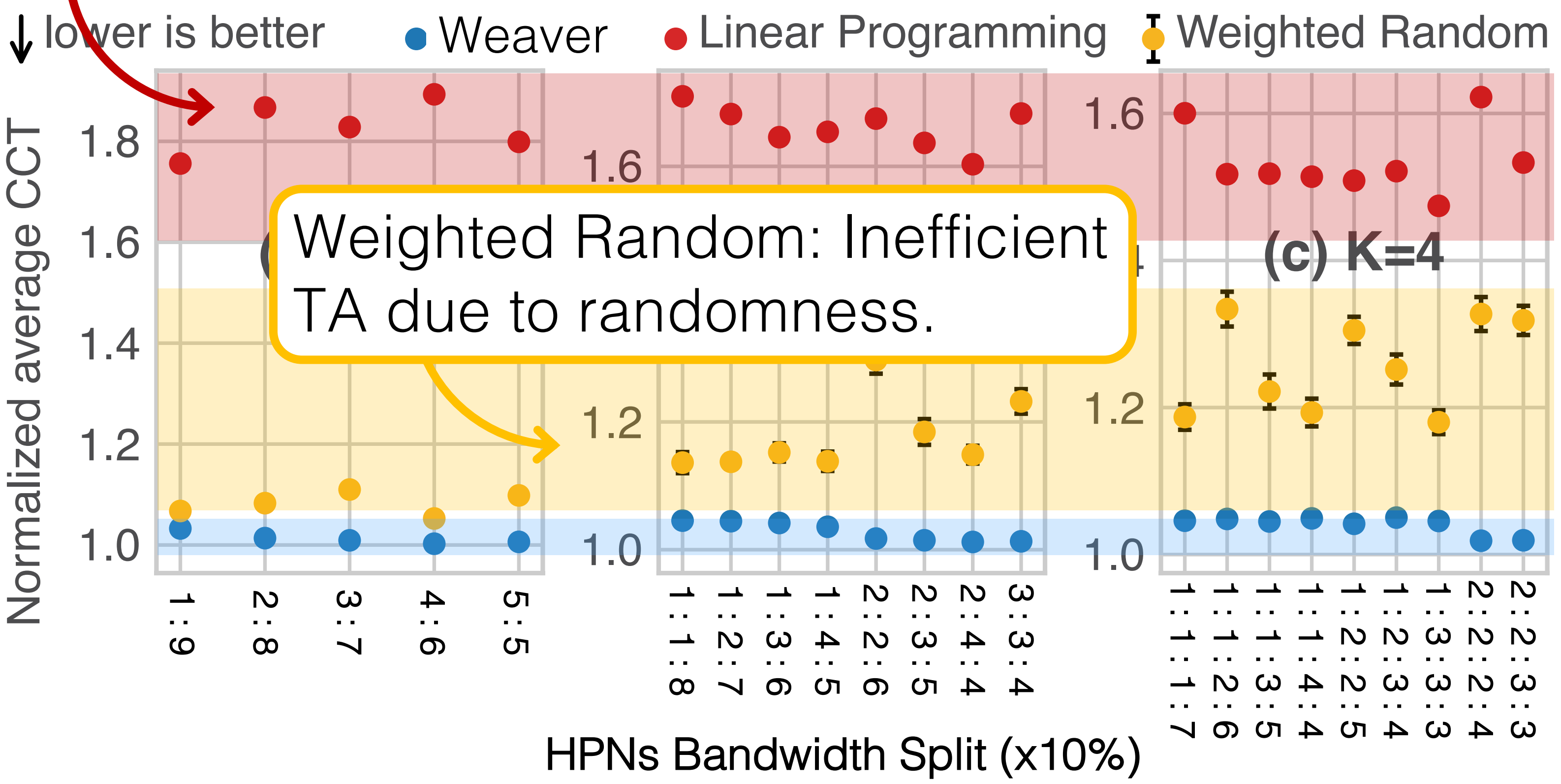
The Weaver-orchestrated HPNs achieve Coflow performance comparable to the monolithic network.

We have also validated the inter-Coflow scheduling efficiency with testbed experiments.
Our testbed results generally resemble those of simulations. See paper for details.

# ge CCT

LP-based Rapier: Less efficient TA algorithm and less efficient inter-Coflow scheduling in HPNs

*Normalized average CCT under various HPNs Configurations*



Weighted Random: Inefficient TA due to randomness.

The Weaver-orchestrated HPNs achieve Coflow performance comparable to the monolithic network.

We have also validated the inter-Coflow scheduling efficiency with testbed experiments.
Our testbed results generally resemble those of simulations. See paper for details.

# Refer to our paper for more results

- [Simulations] Intra-Coflow TA efficiency
    - Weaver's TA has better performance guarantee
- [Simulations] Inter-Coflow Scheduling (TA+BA)
    - Weaver achieves Coflow performance close to the ideal monolithic network.
    - Weaver improves TA by better assignment ordering
    - Weaver improves TA by load balancing non-critical flows
    - Weaver remains robust under different BA policies
- [Testbed] Inter-Coflow Scheduling (TA+BA)
    - Weaver achieves Coflow performance close to the ideal monolithic network

Open Source Code & Benchmark

**https://github.com/sunnyxhuang/weaver**

# Conclusions

- The Weaver-orchestrated HPNs achieve Coflow performance comparable to the ideal monolithic network.
- Weaver exploits HPNs at two levels: efficient traffic assignment for each Coflow and coordinated bandwidth allocation among multiple Coflows.
- Weaver inspires how an evolving data center can make the most out of its multiple generations of network fabrics.

Open Source Code & Benchmark

**https://github.com/sunnyxhuang/weaver**

**Thank You!**