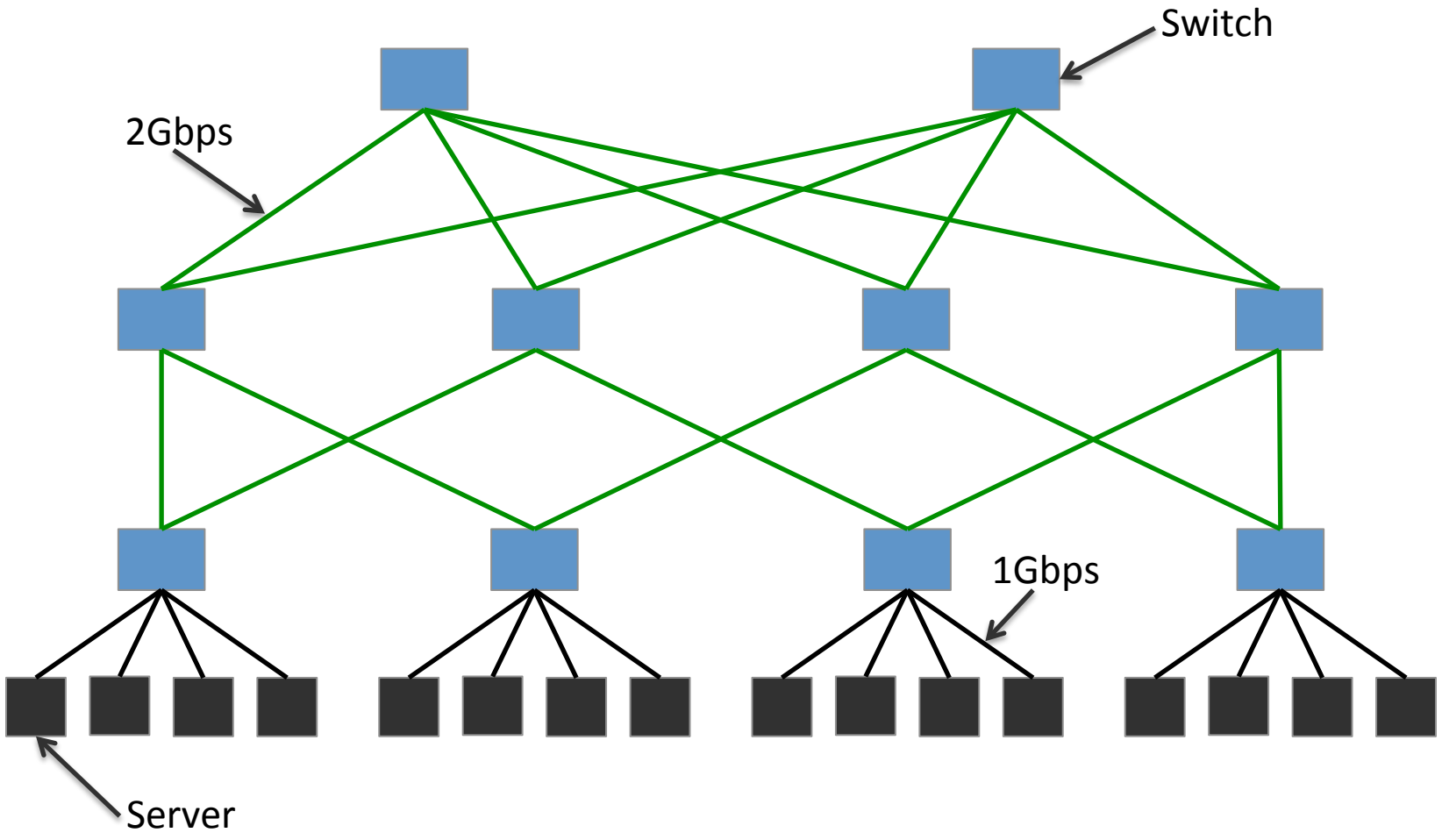
Increasing Datacenter Network Utilisation with GRIN

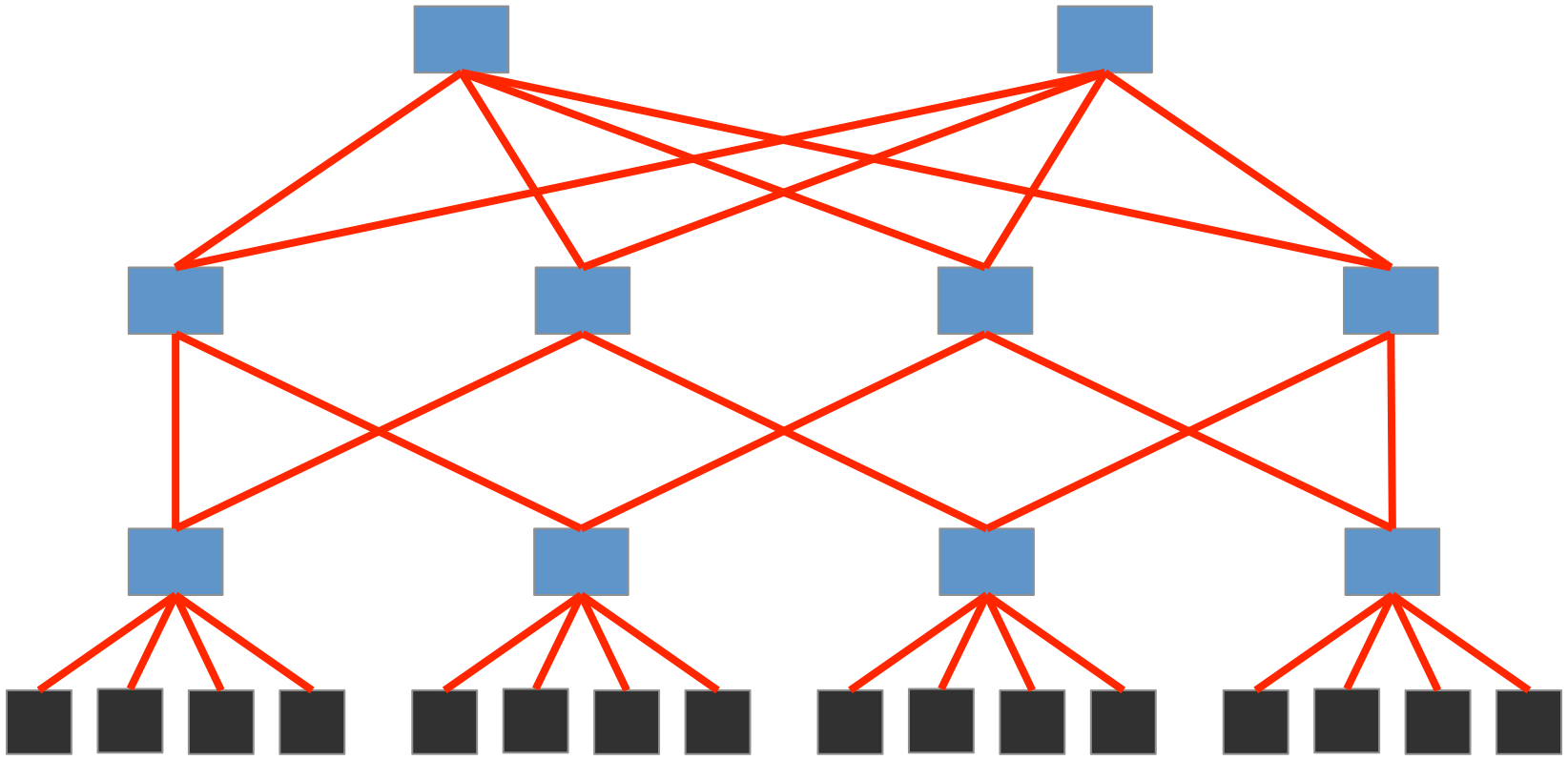
Alexandru Agache, Razvan Deaconescu, Costin Raiciu

University Politehnica of Bucharest

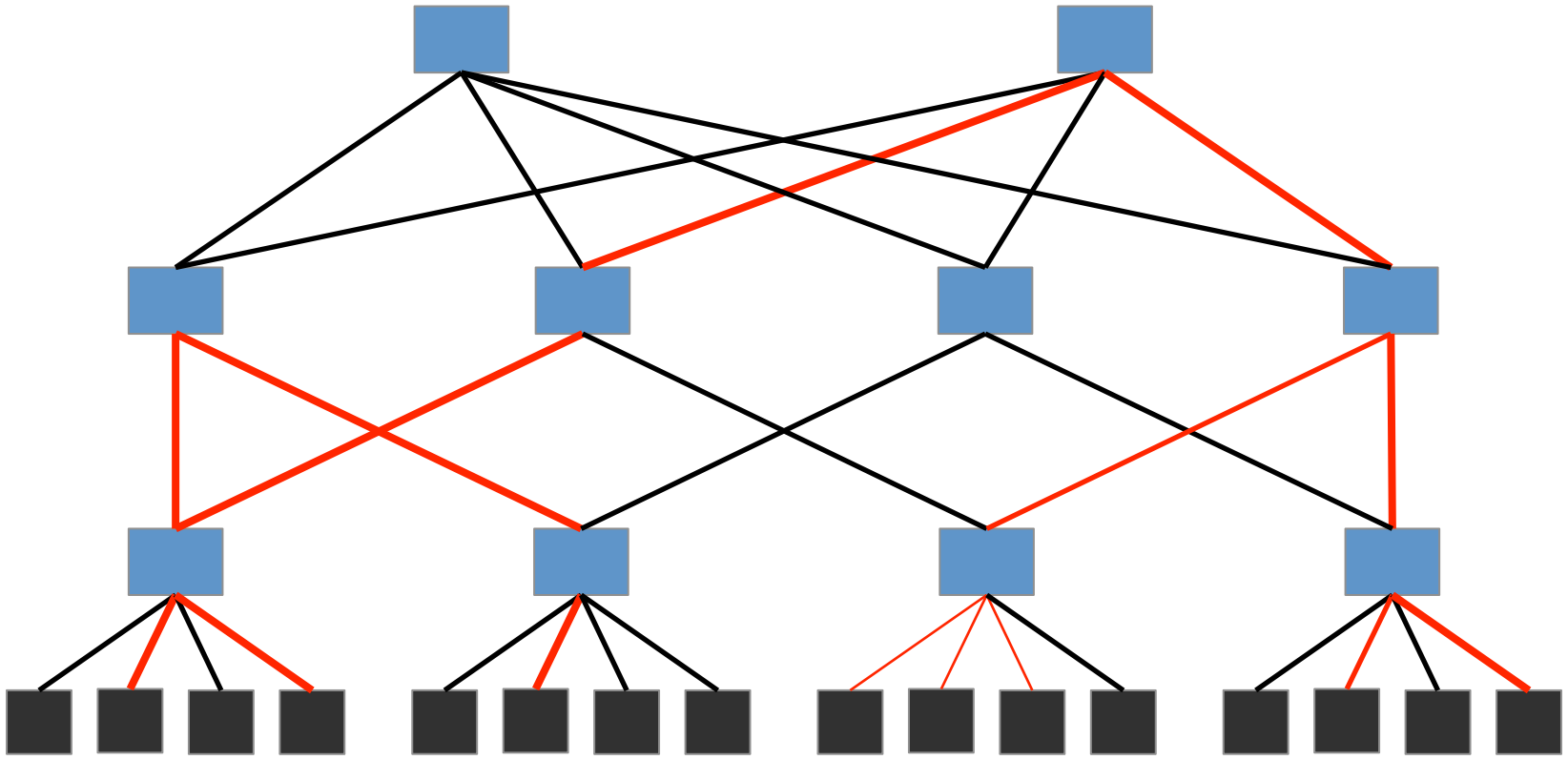
High Capacity Networks



Heavy All-to-All Traffic



Mixed Traffic



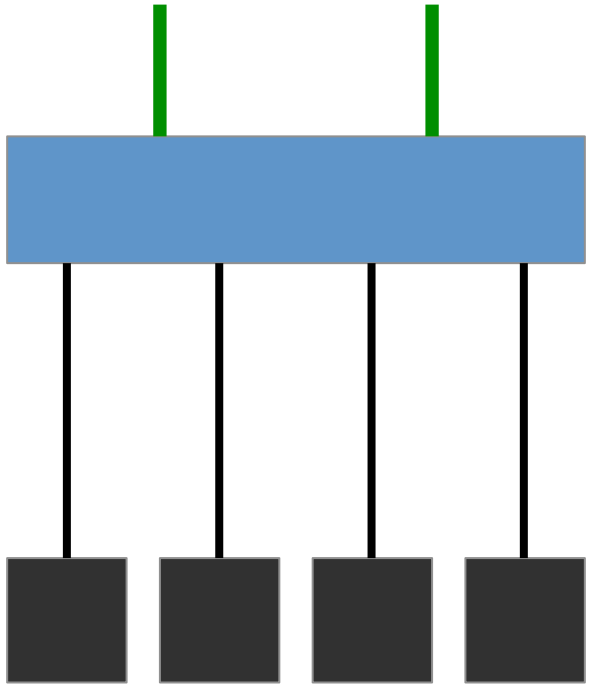
Problem

- The network is underutilised
- Servers are bottlenecked by the use of a single network interface

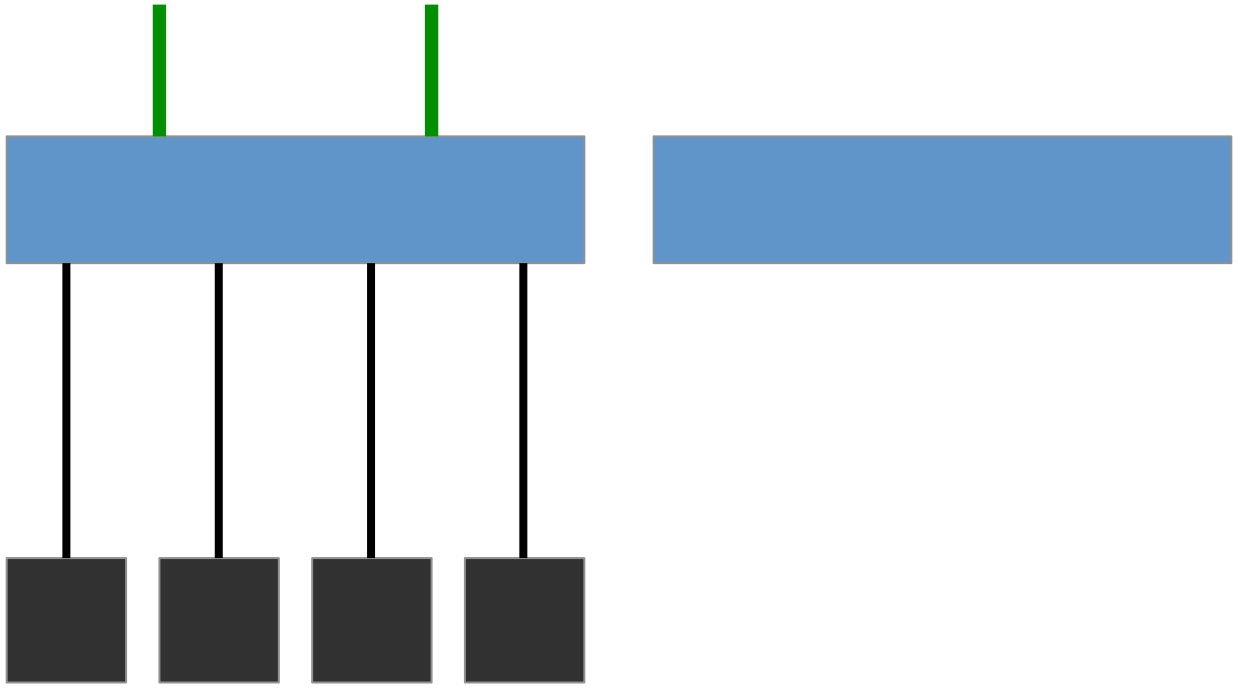
~~Problem~~ Solution

- The network is underutilised
- Servers are bottlenecked by the use of a single network interface
- Multiple NICs !

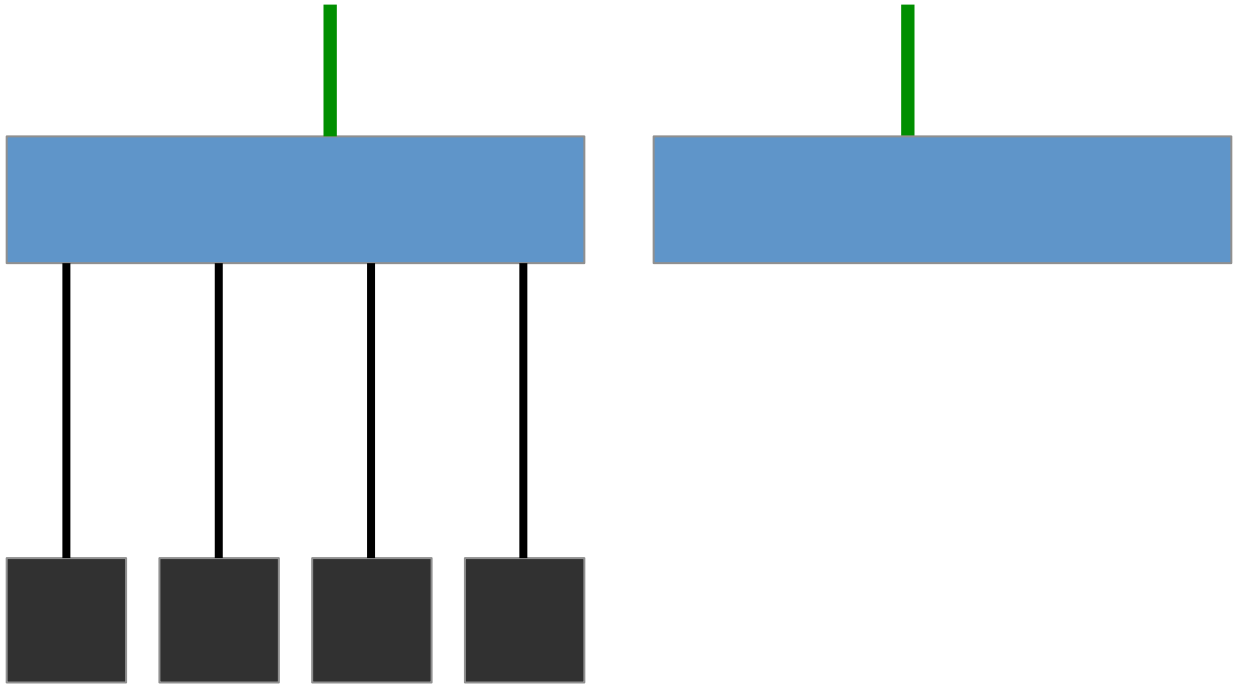
Solution



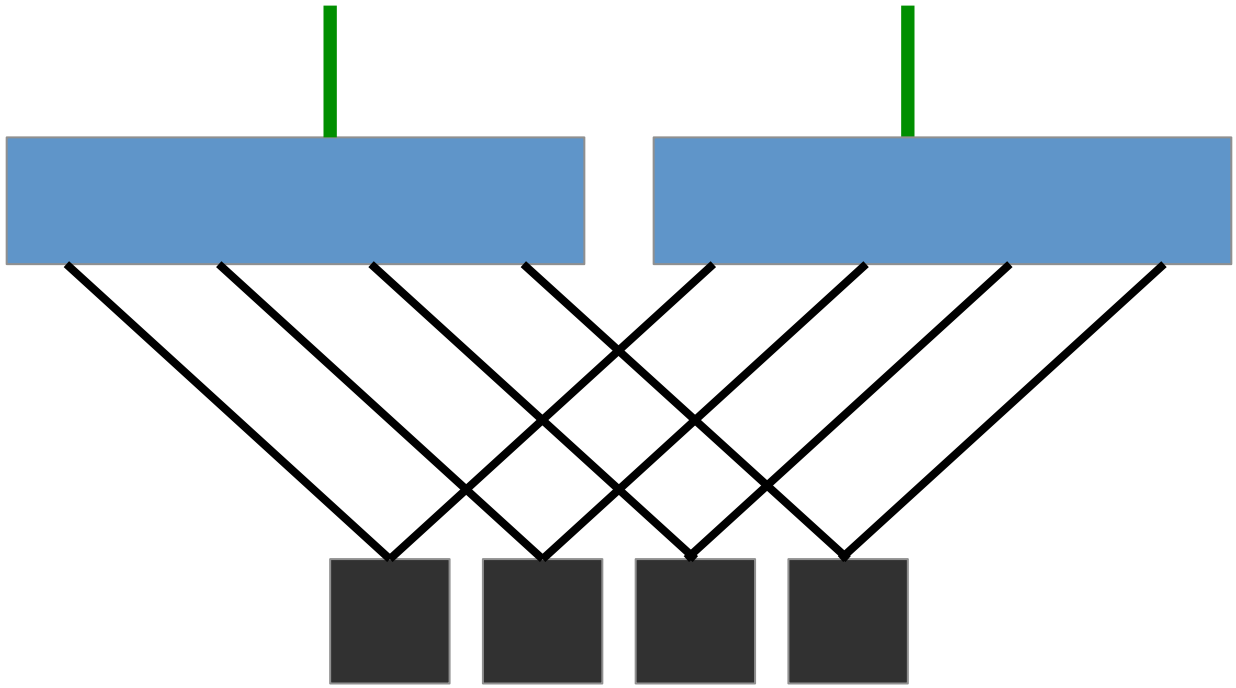
Multihoming



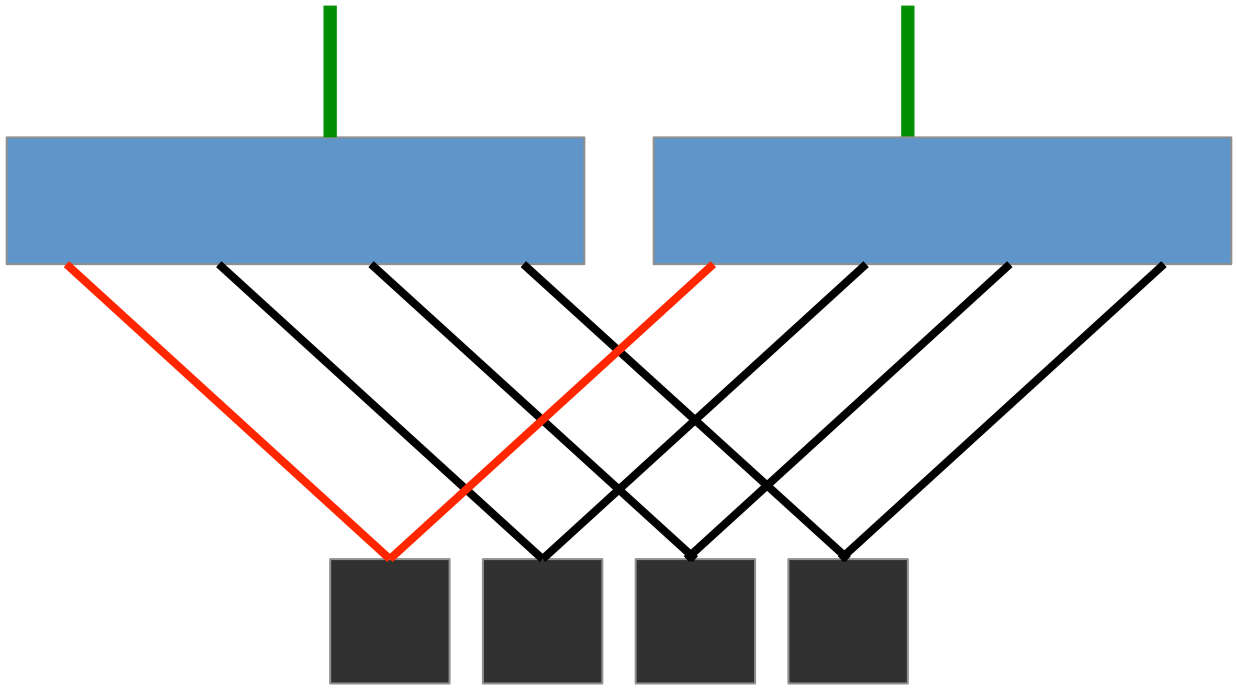
Multihoming



Multihoming



Multihoming



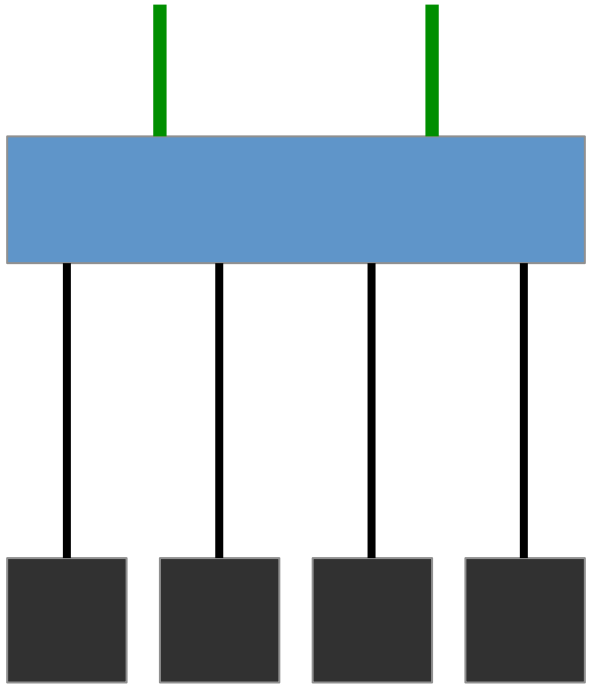
Multihoming

- Can increase performance significantly

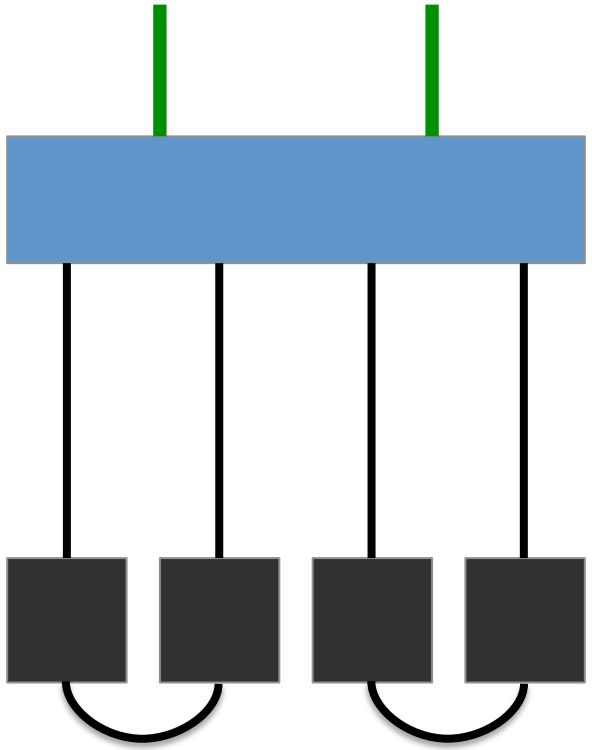
- Expensive:

- Additional switches
- Additional space
- Additional maintenance costs

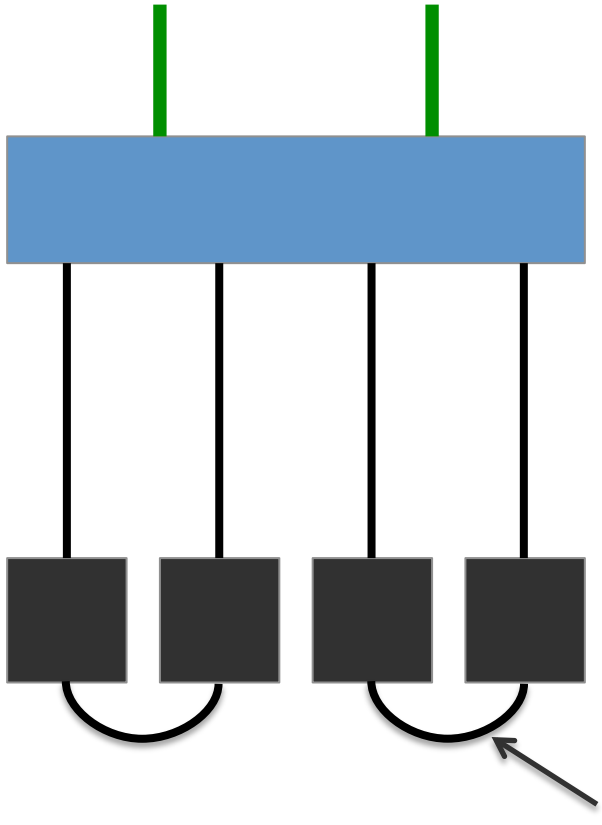
GRIN



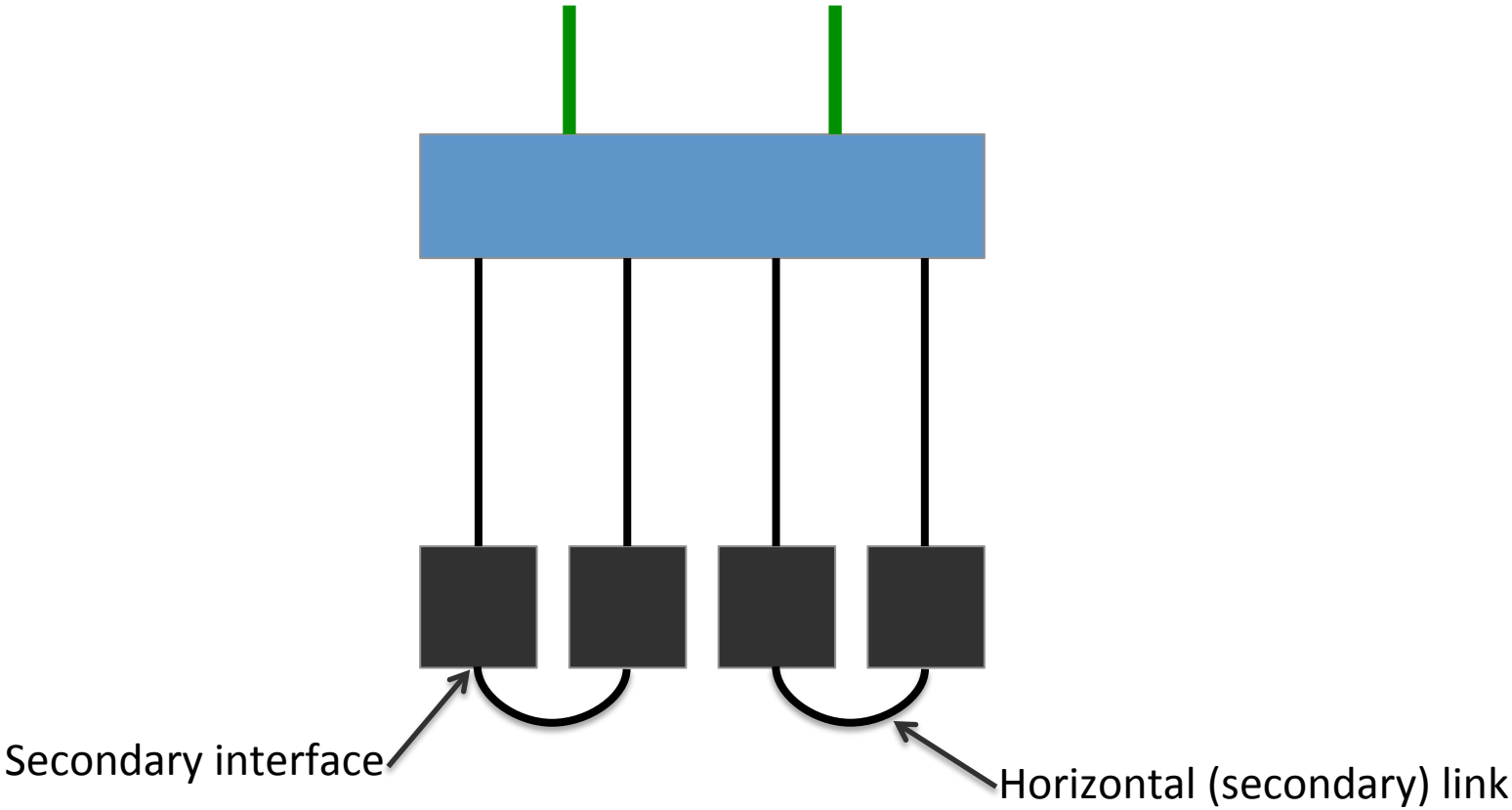
GRIN



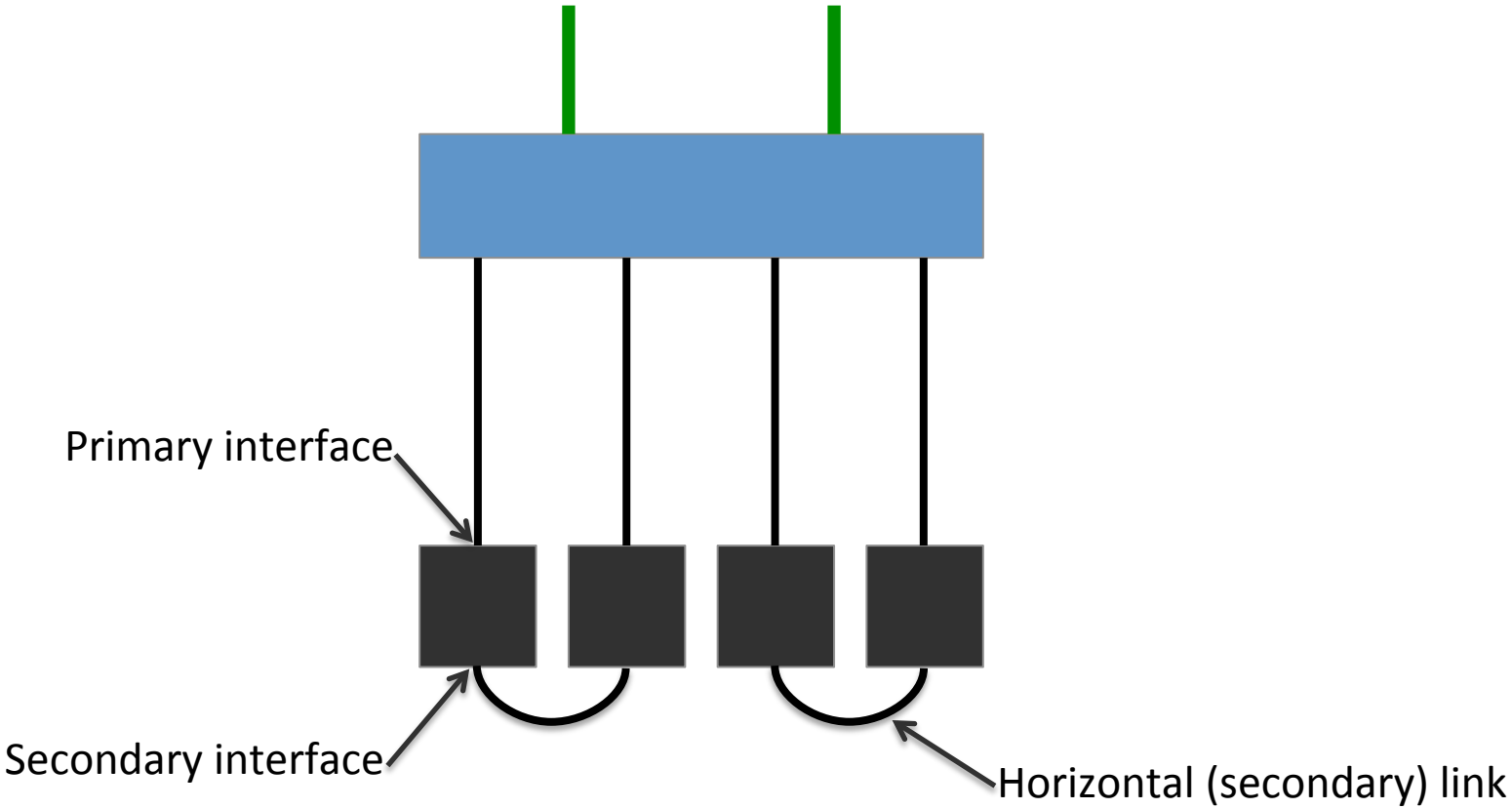
GRIN



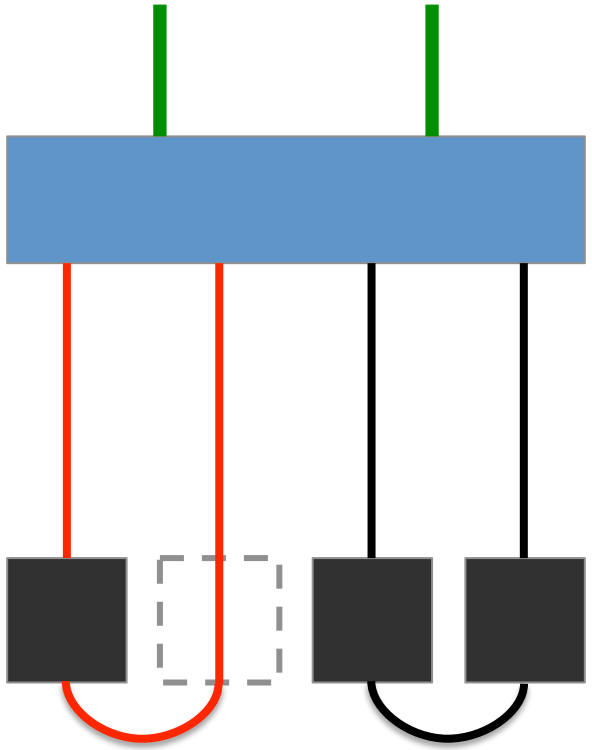
GRIN



GRIN


Primary interface
Secondary interface
Horizontal (secondary) link

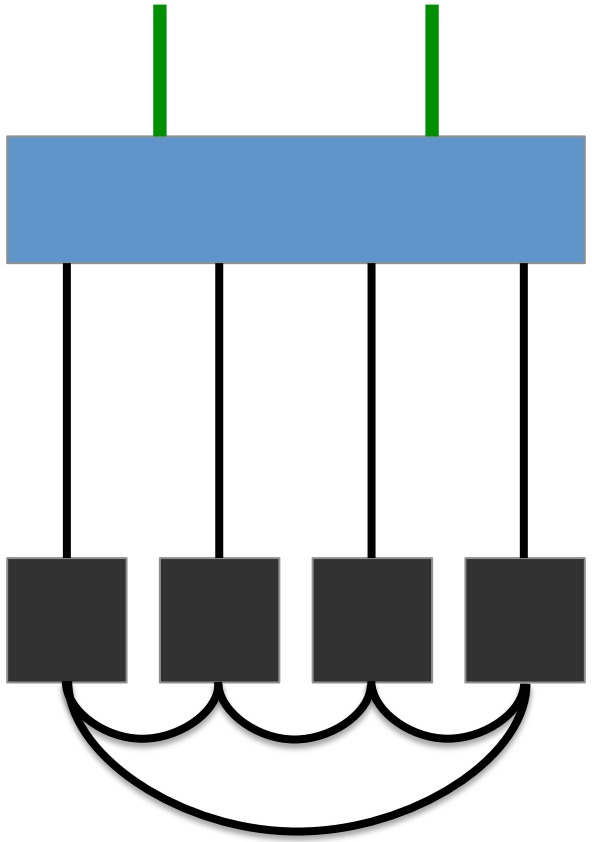
GRIN



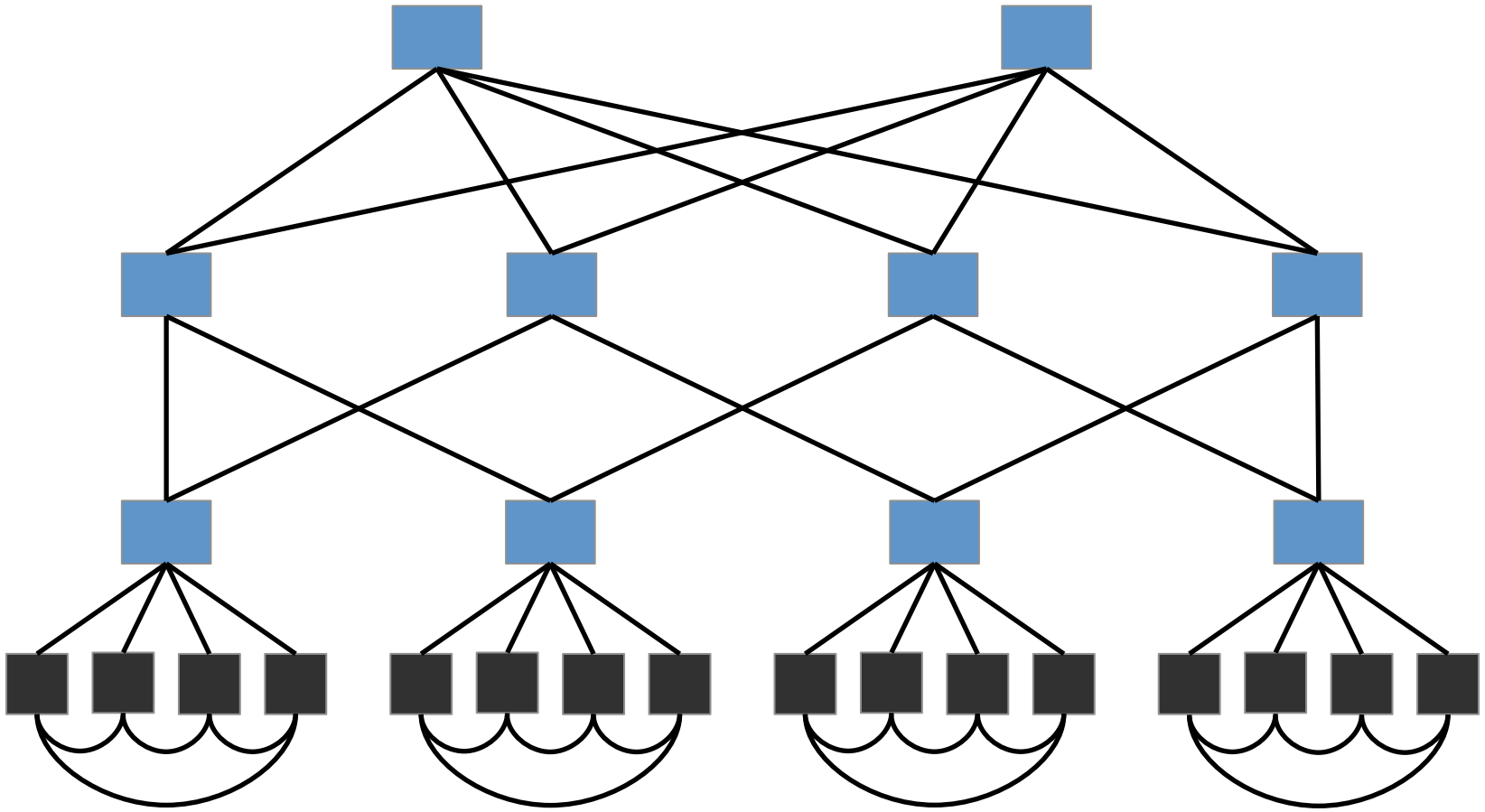
MPTCP

- One connection can be split into multiple concurrent subflows
- Allows the use multiple NICs in a transparent manner toward applications

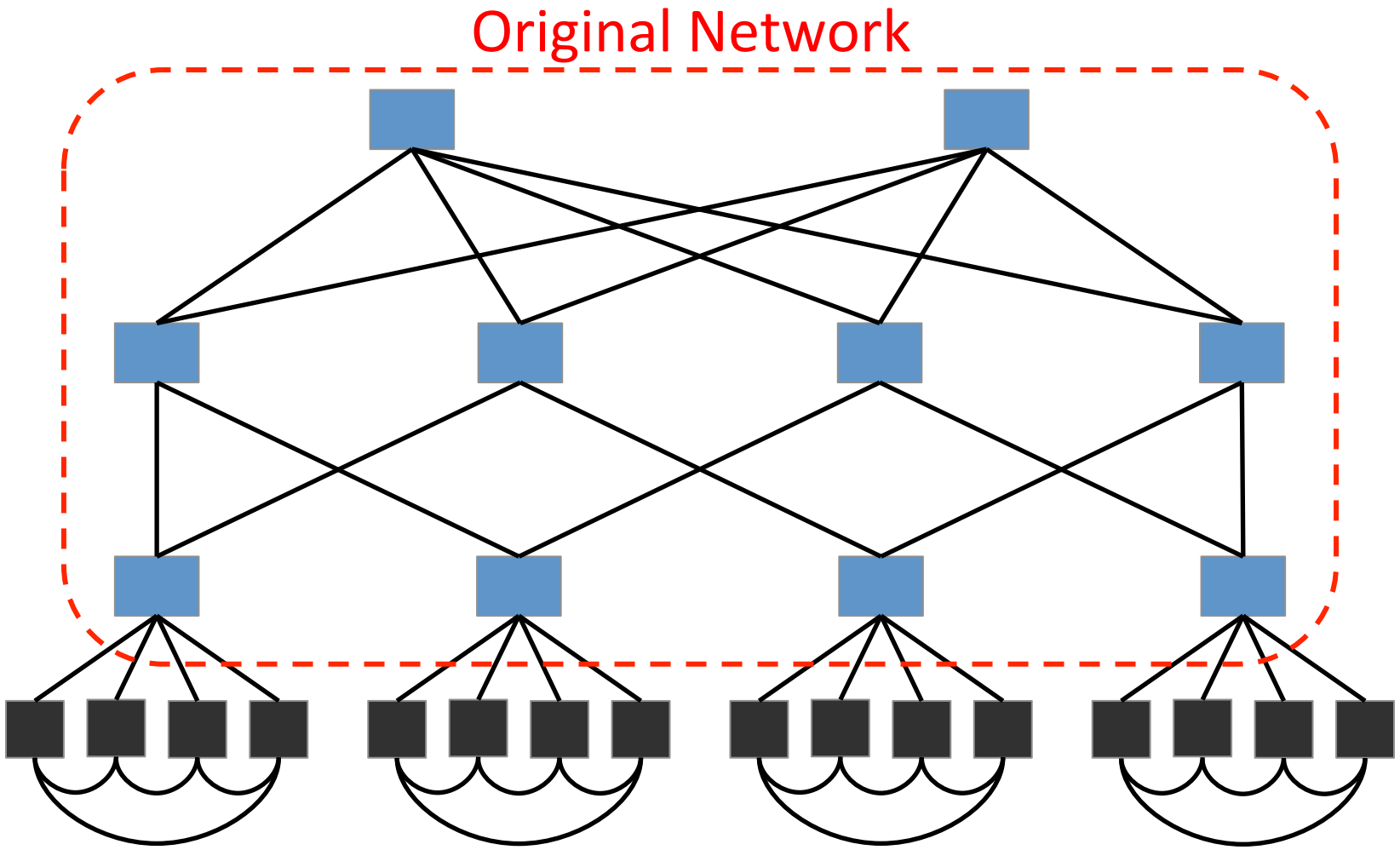
Using Two Additional Ports



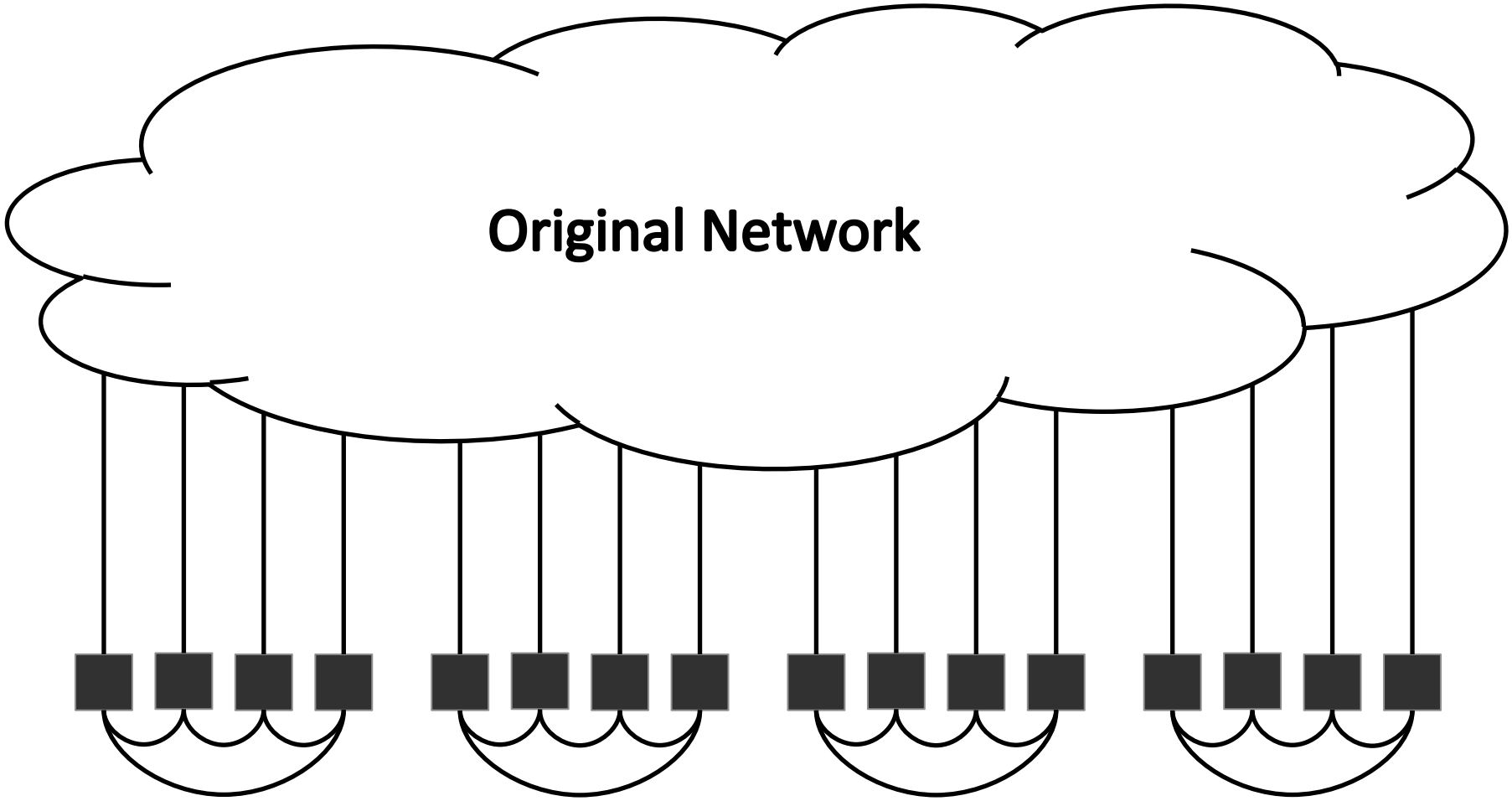
GRIN2 Topology



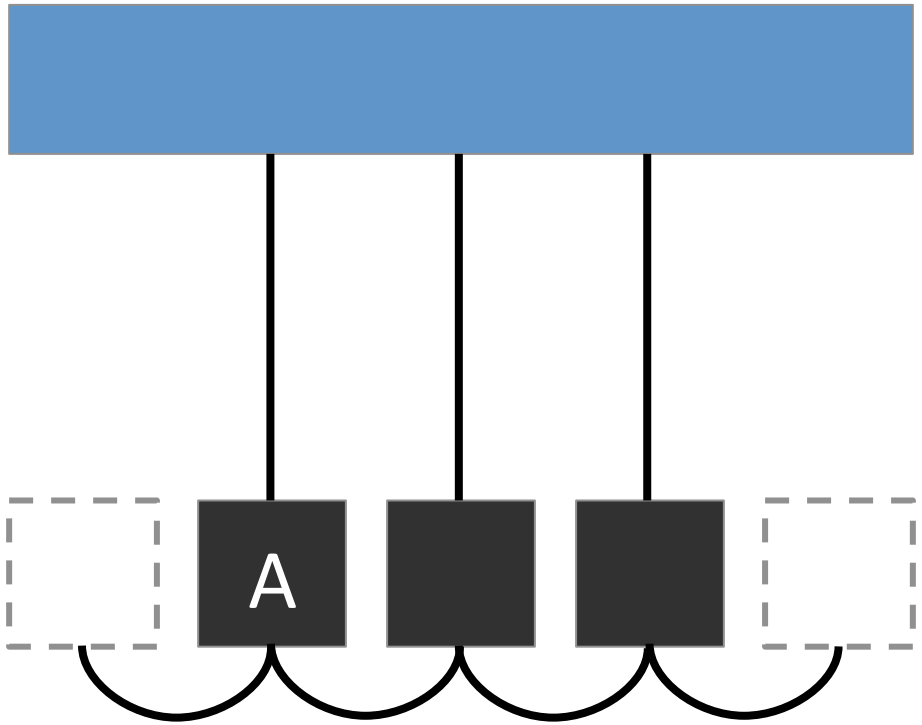
GRIN2 Topology



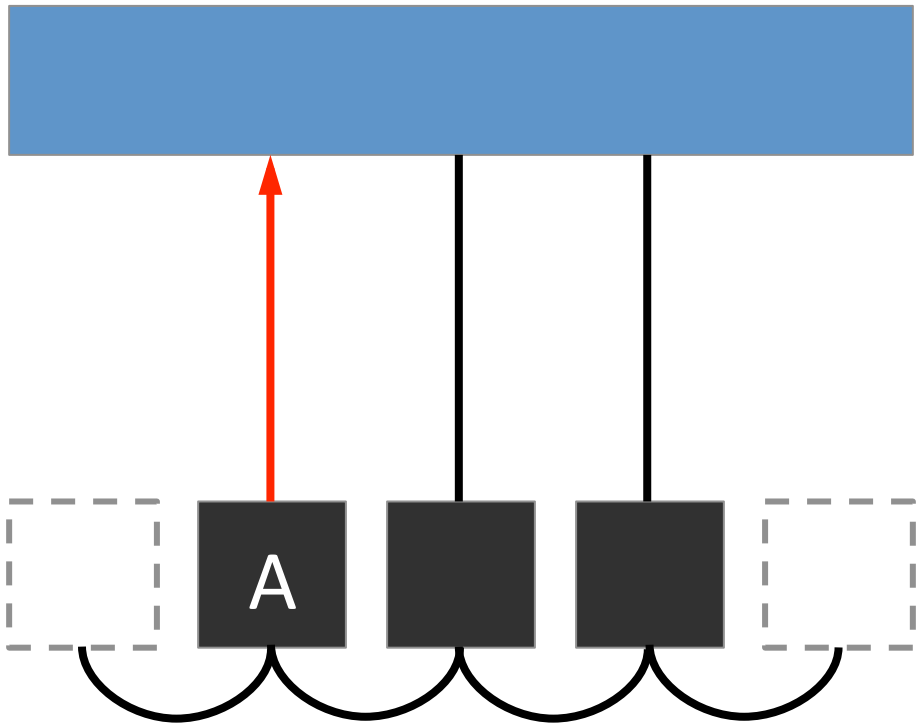
GRIN2 Topology



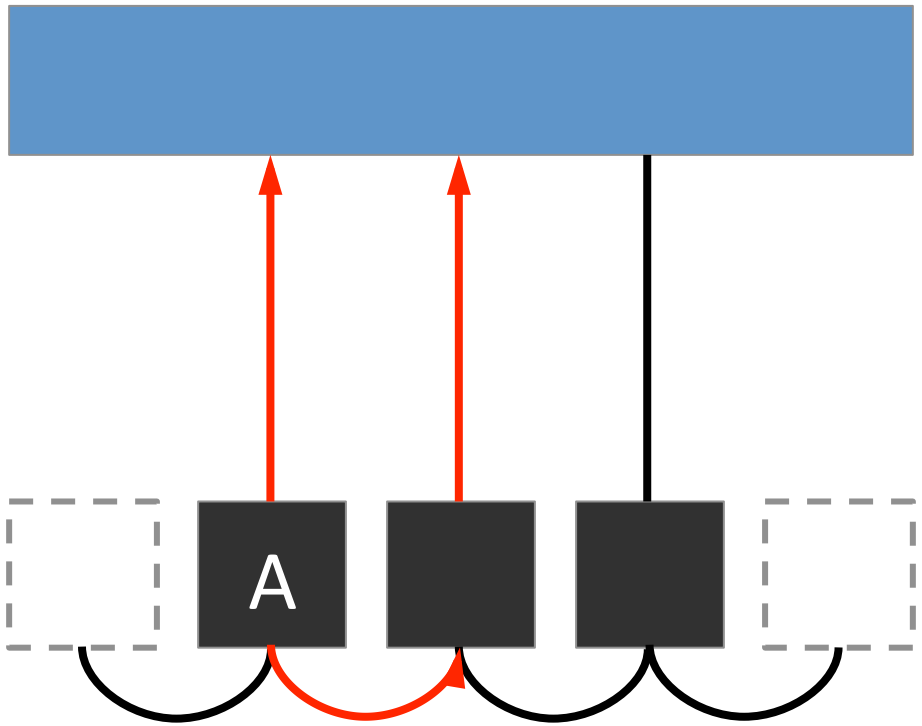
Using GRIN - Sender



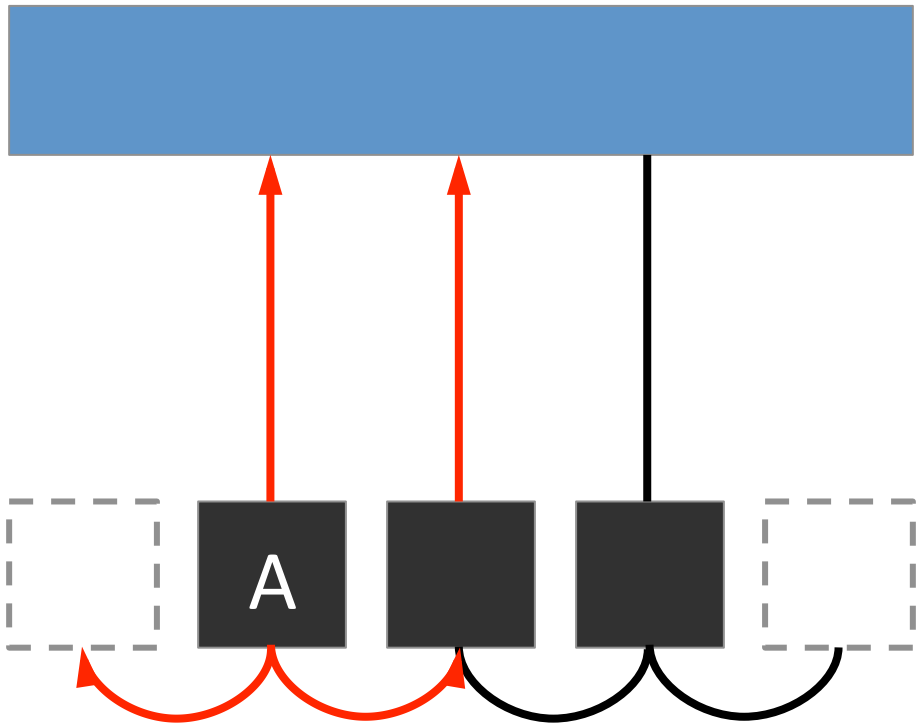
Using GRIN - Sender



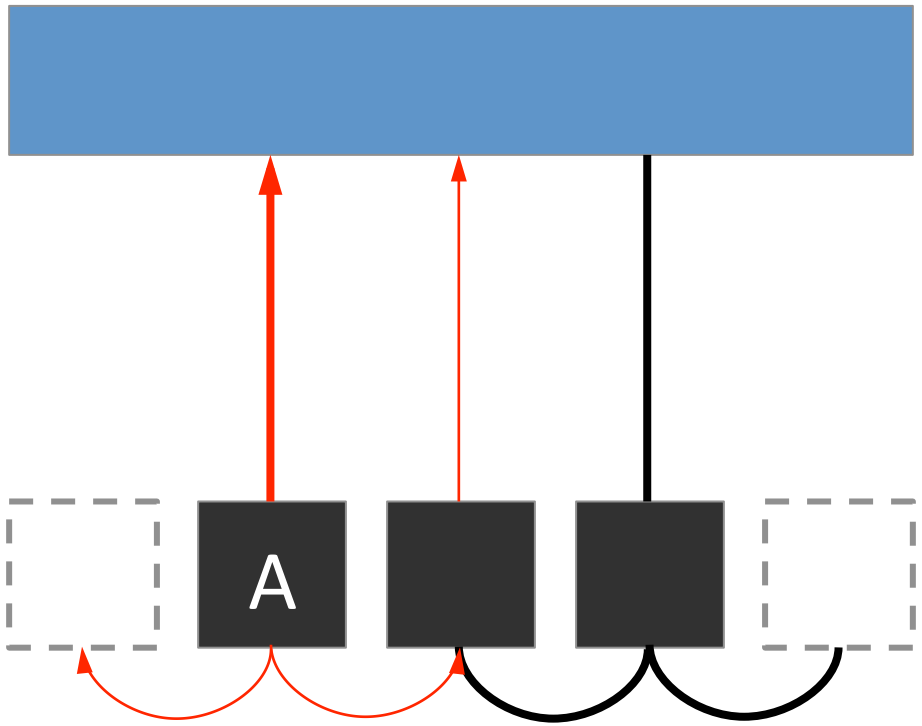
Using GRIN - Sender



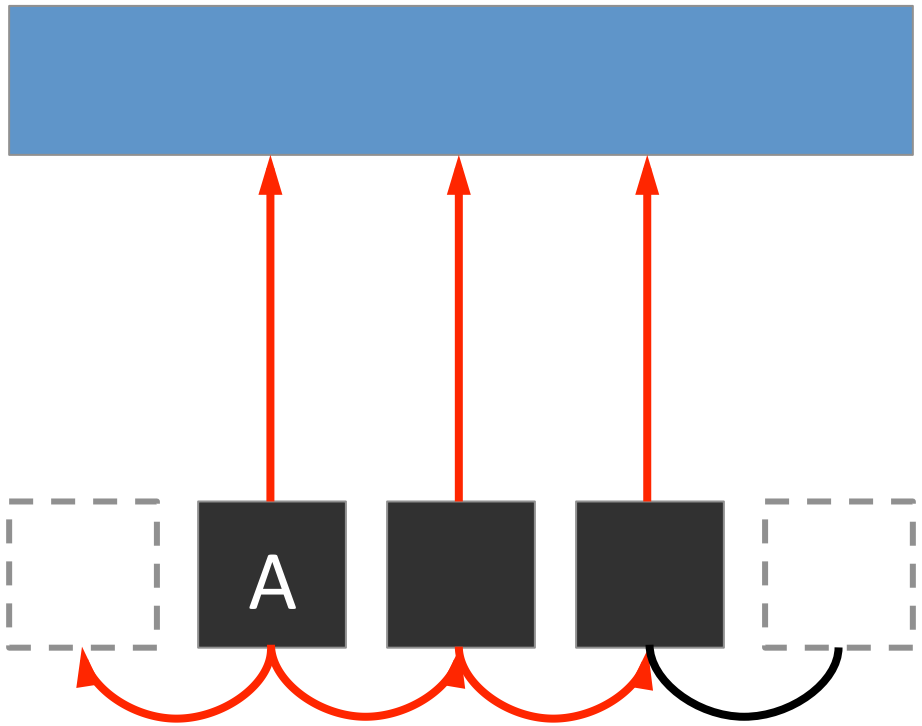
Using GRIN - Sender



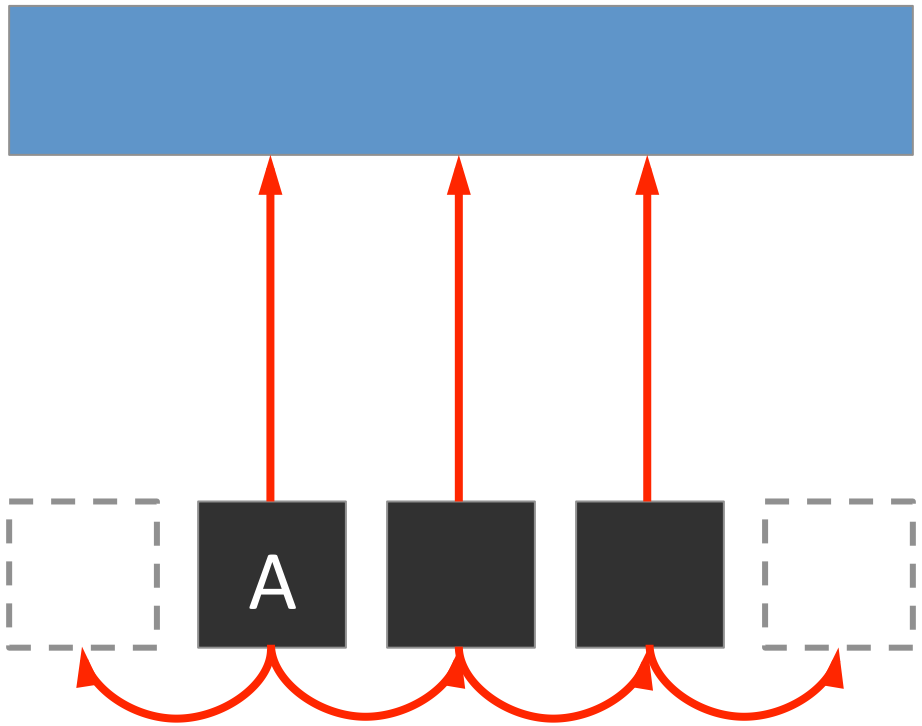
Using GRIN - Sender



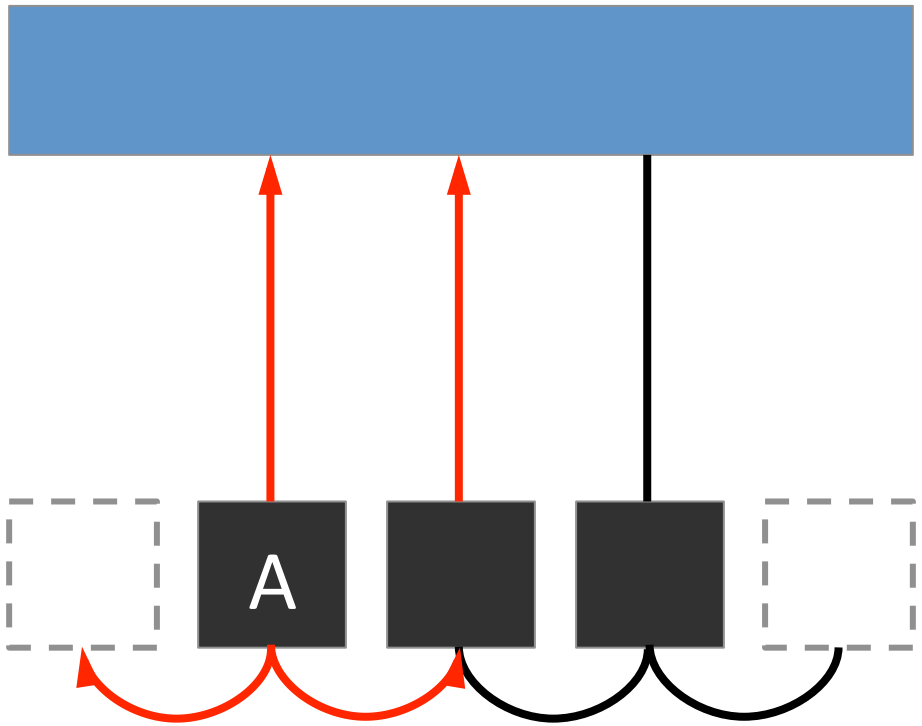
Using GRIN - Sender



H_n Routing



H_1 Routing



How Far Do We Go ?

- Compromise ?

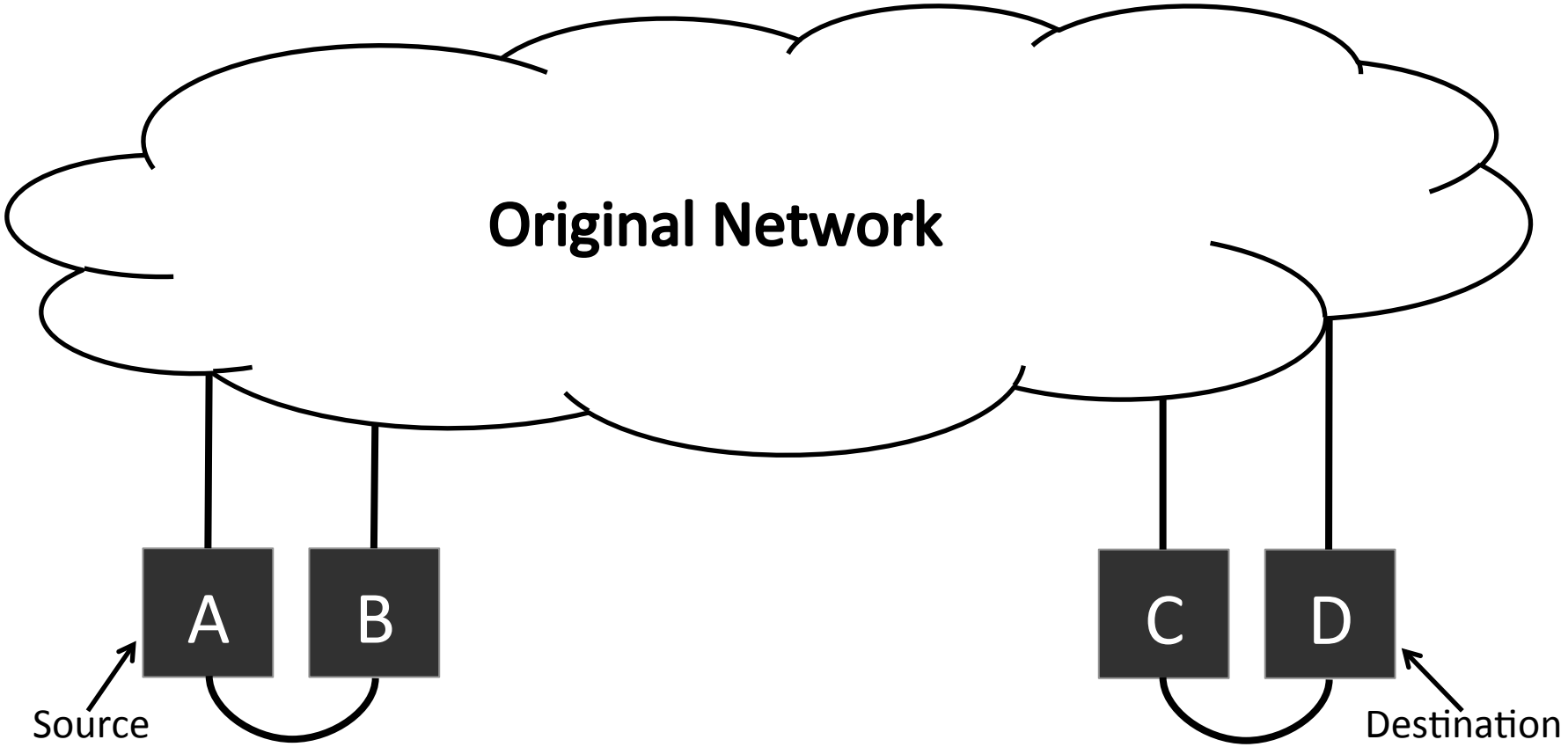
How Far Do We Go ?

- ~~Compromise ?~~
- Using multiple horizontal hops is not actually better than h_1 routing at discovering capacity

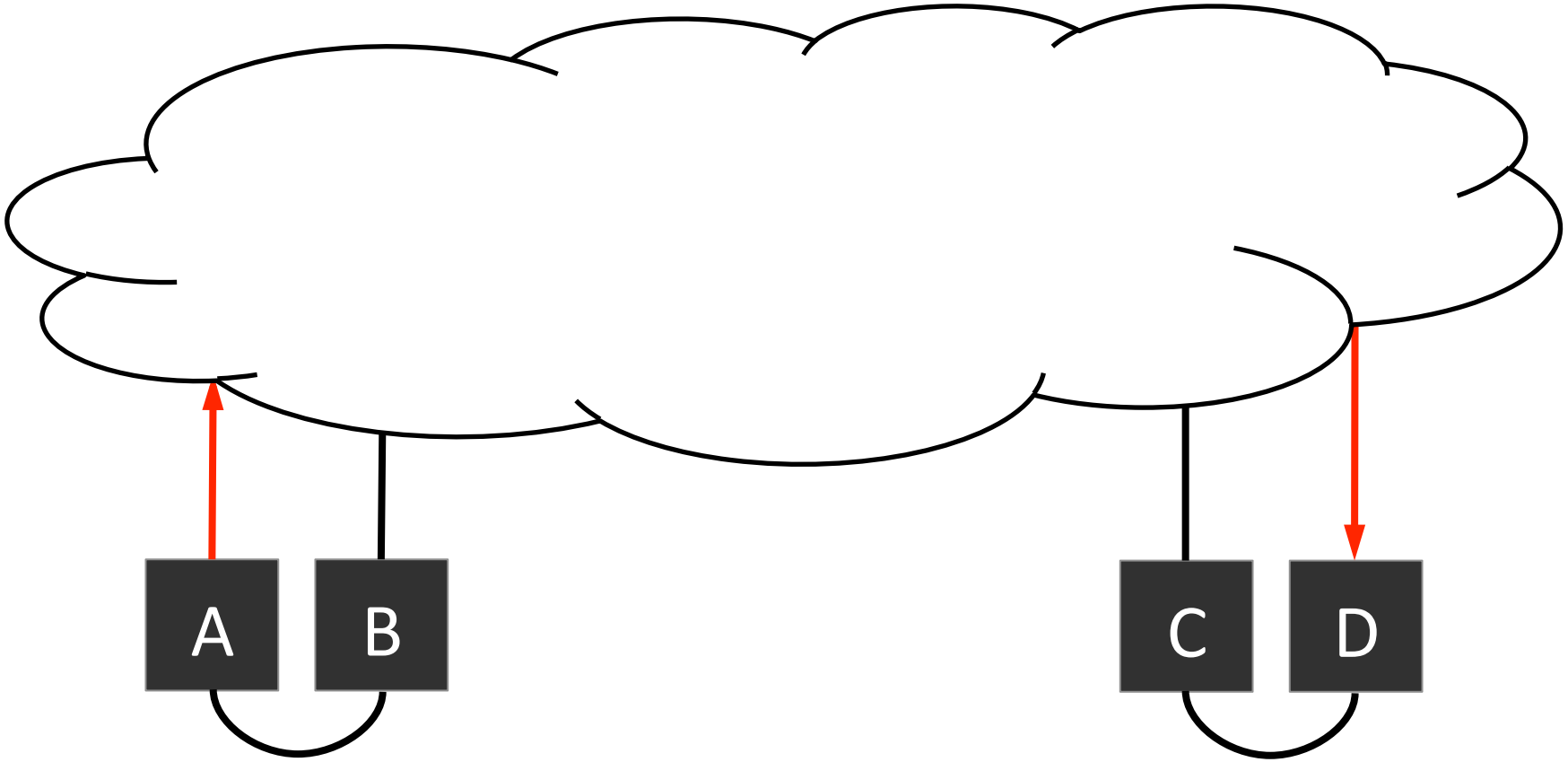
How Far Do We Go ?

- ~~Compromise ?~~
- Using multiple horizontal hops is not actually better than h_1 routing at discovering capacity
- Thus, GRIN only forwards traffic through direct neighbours

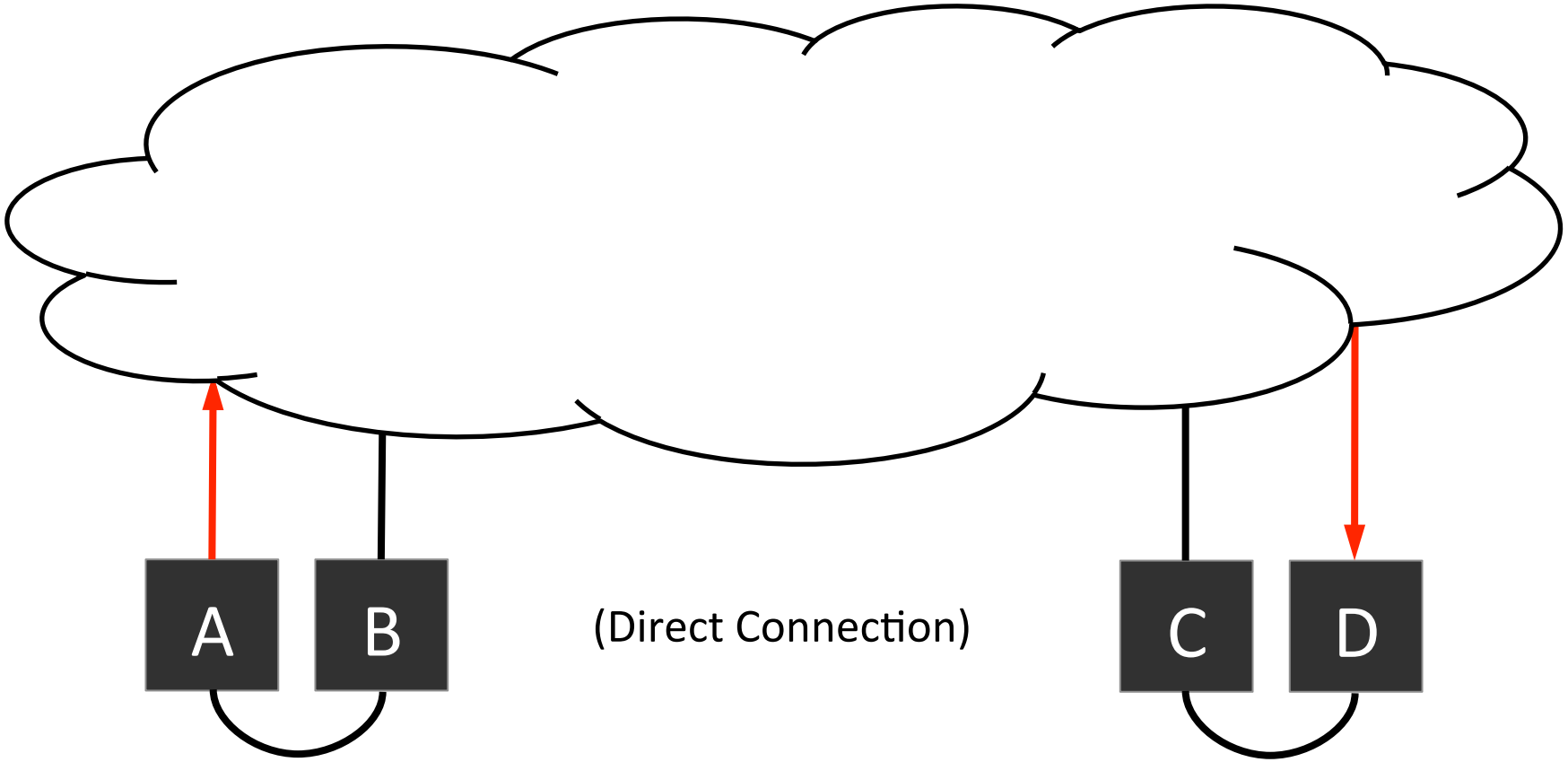
Routing In a GRIN Topology



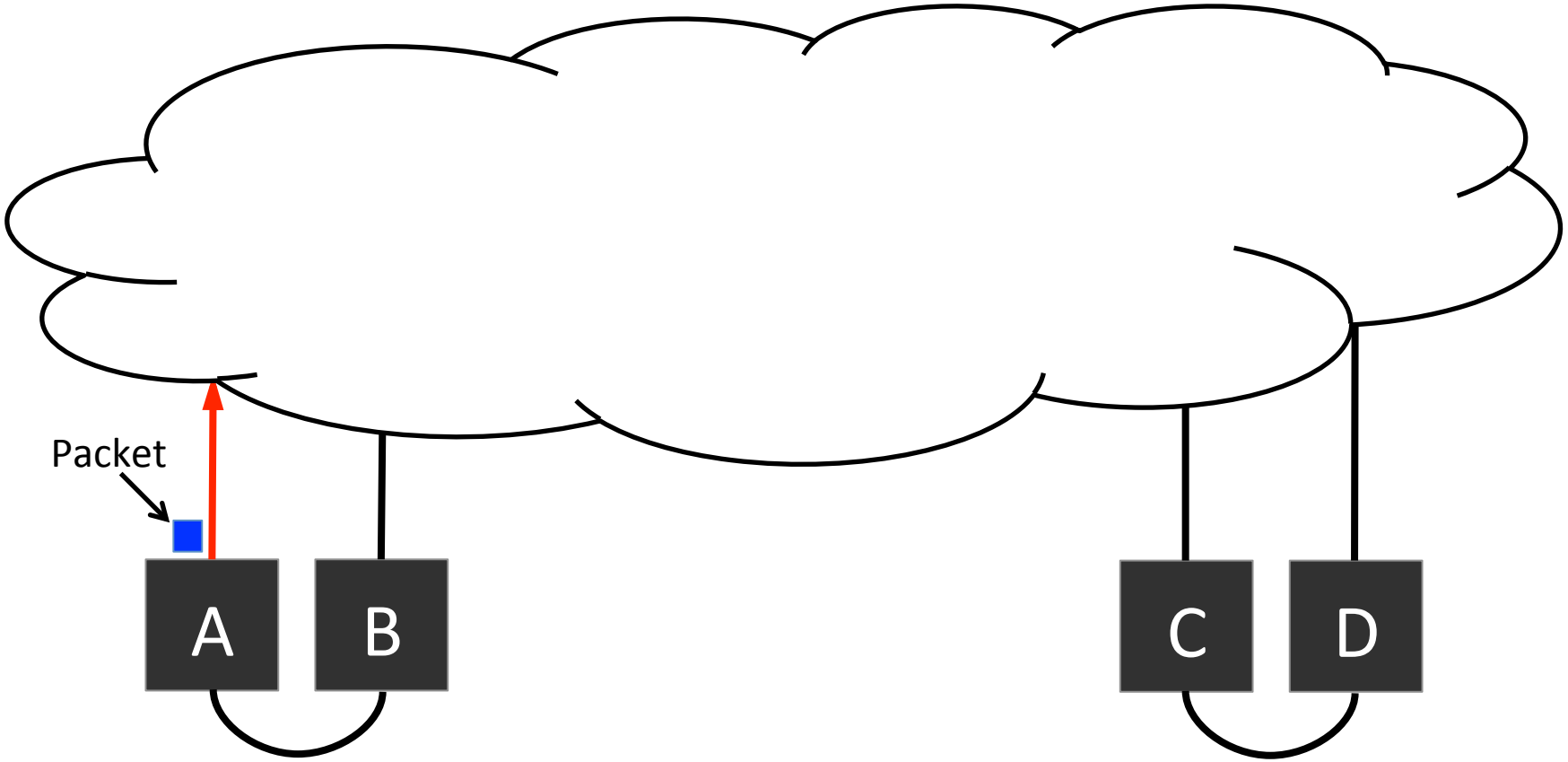
Routing In a GRIN Topology



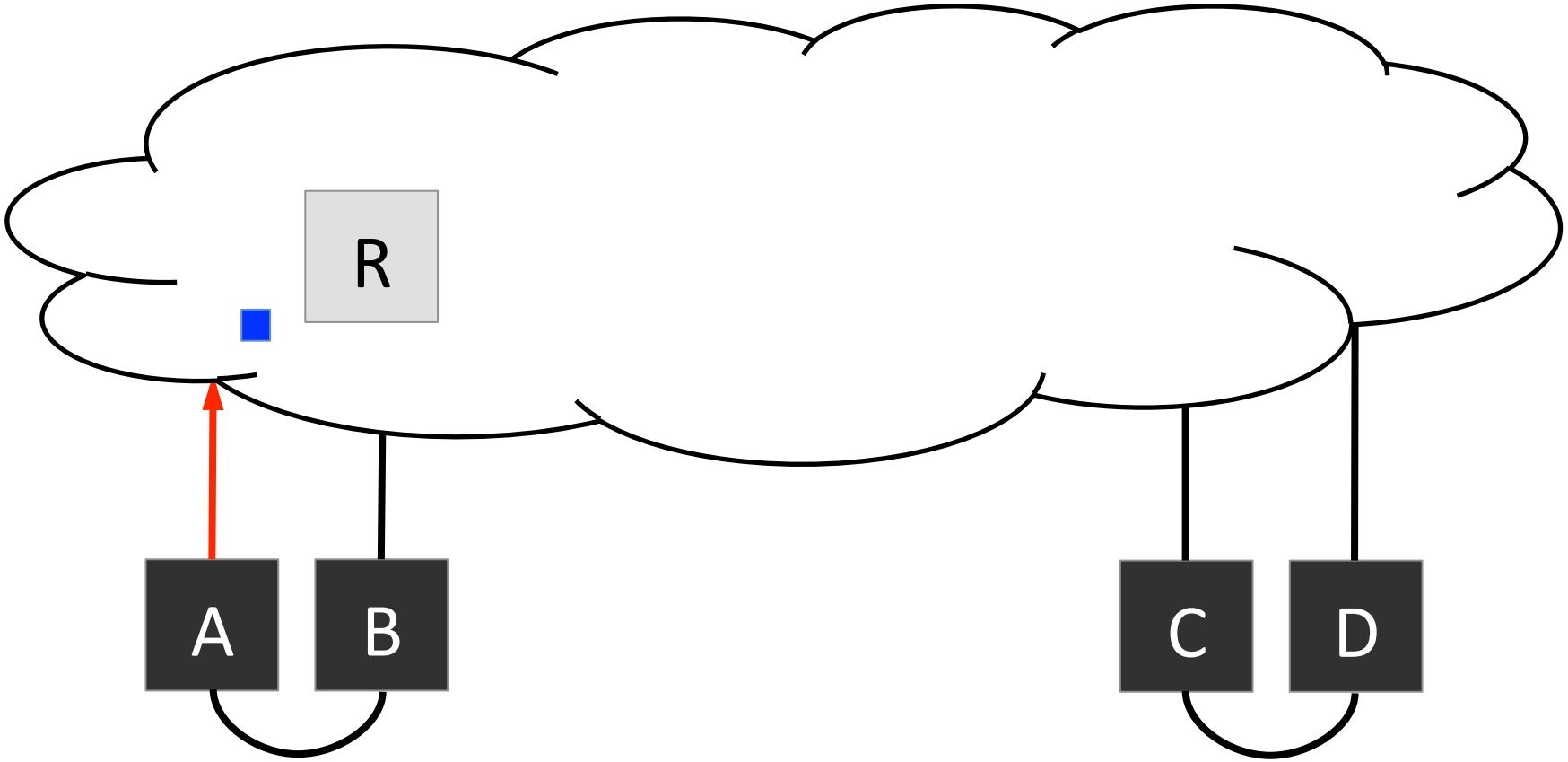
Routing In a GRIN Topology



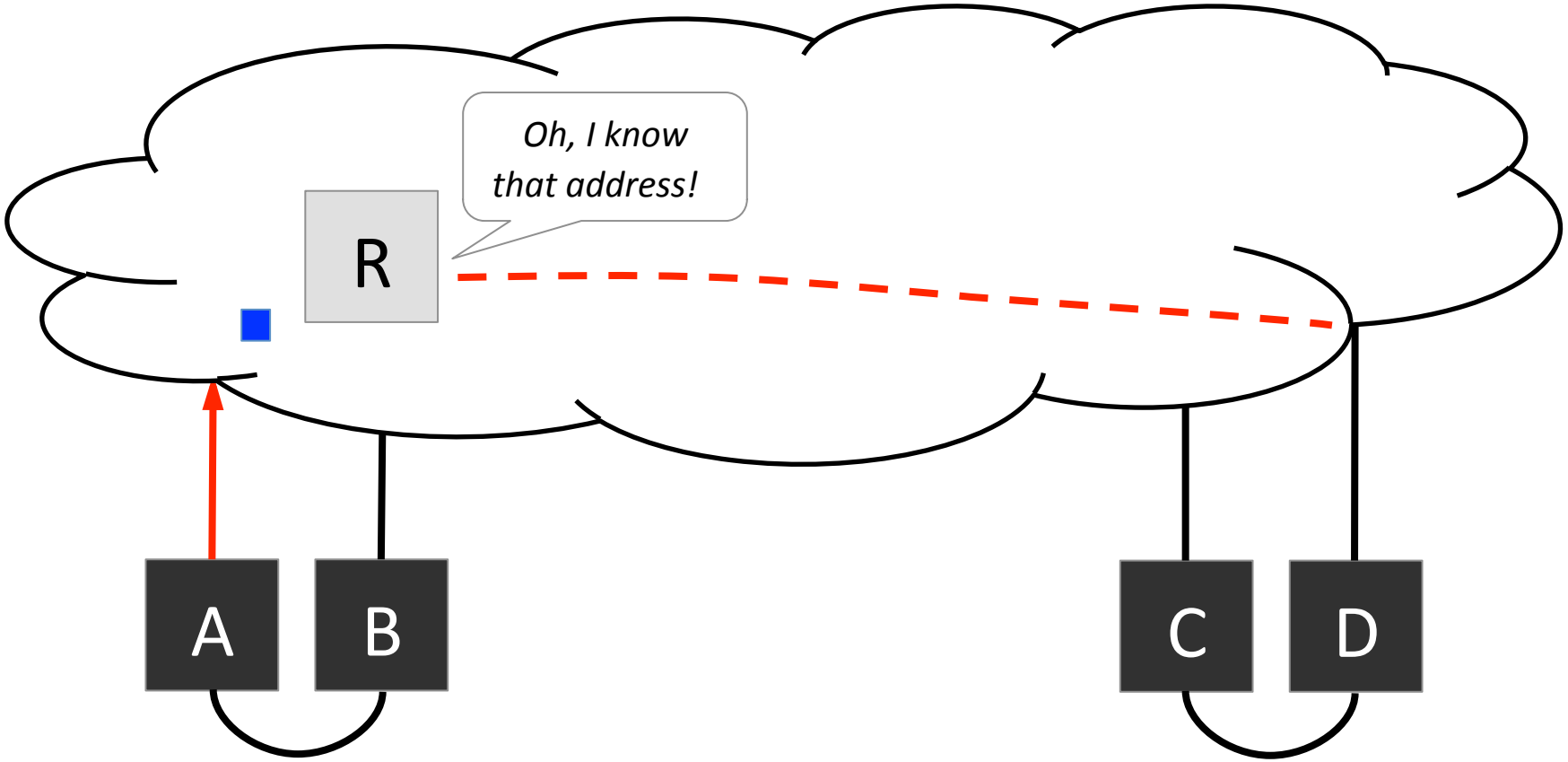
Routing In a GRIN Topology



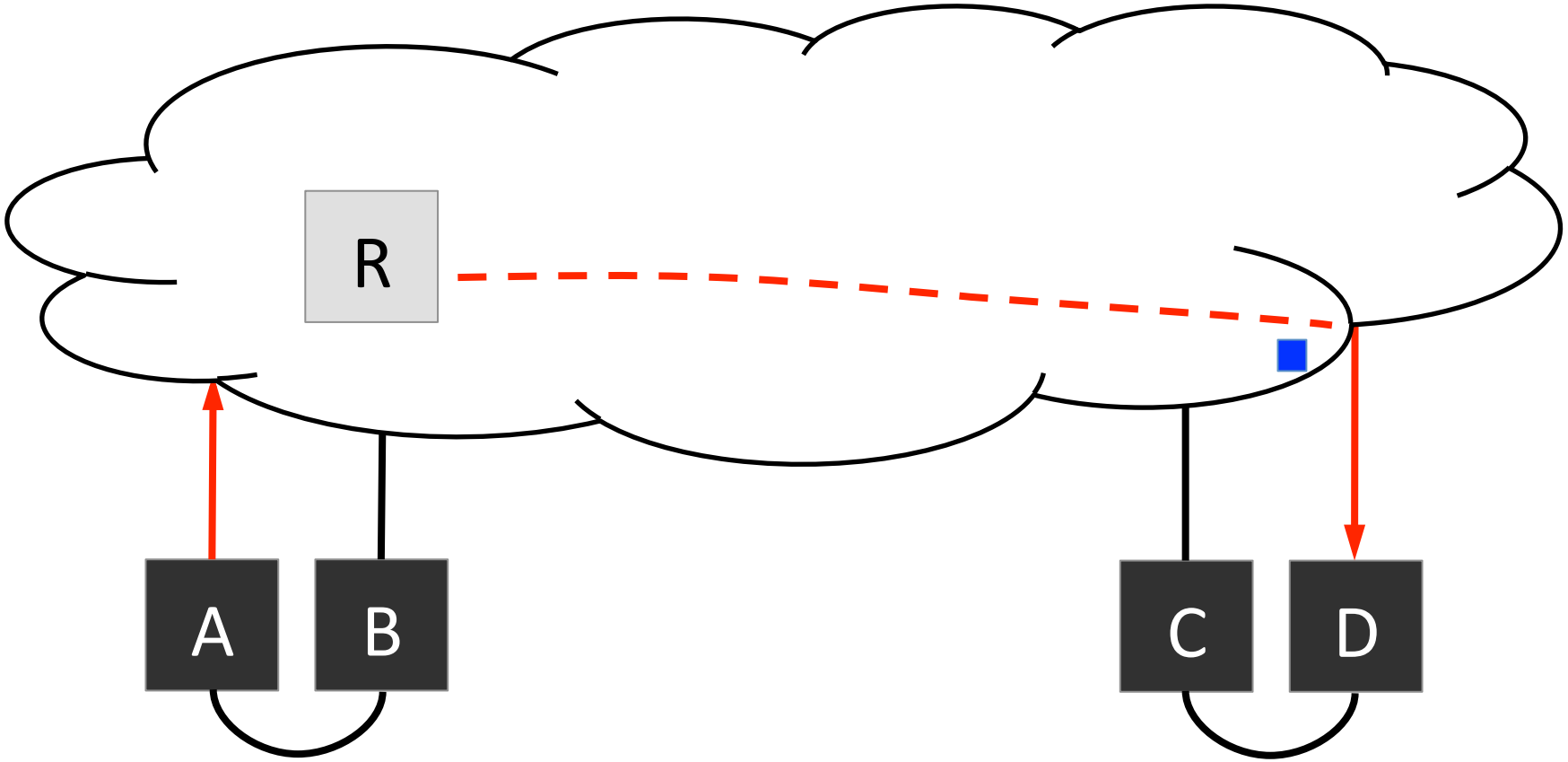
Routing In a GRIN Topology



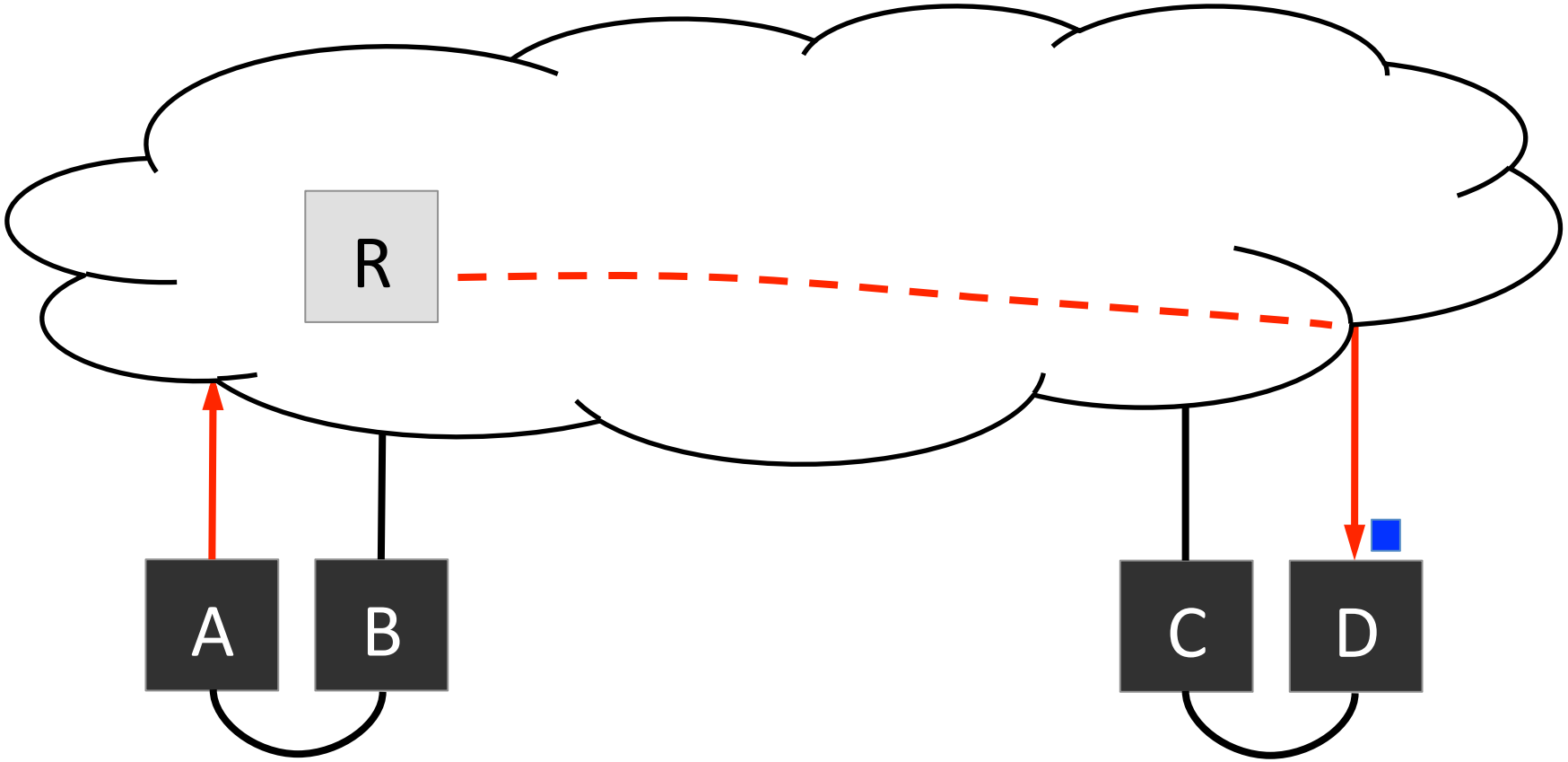
Routing In a GRIN Topology



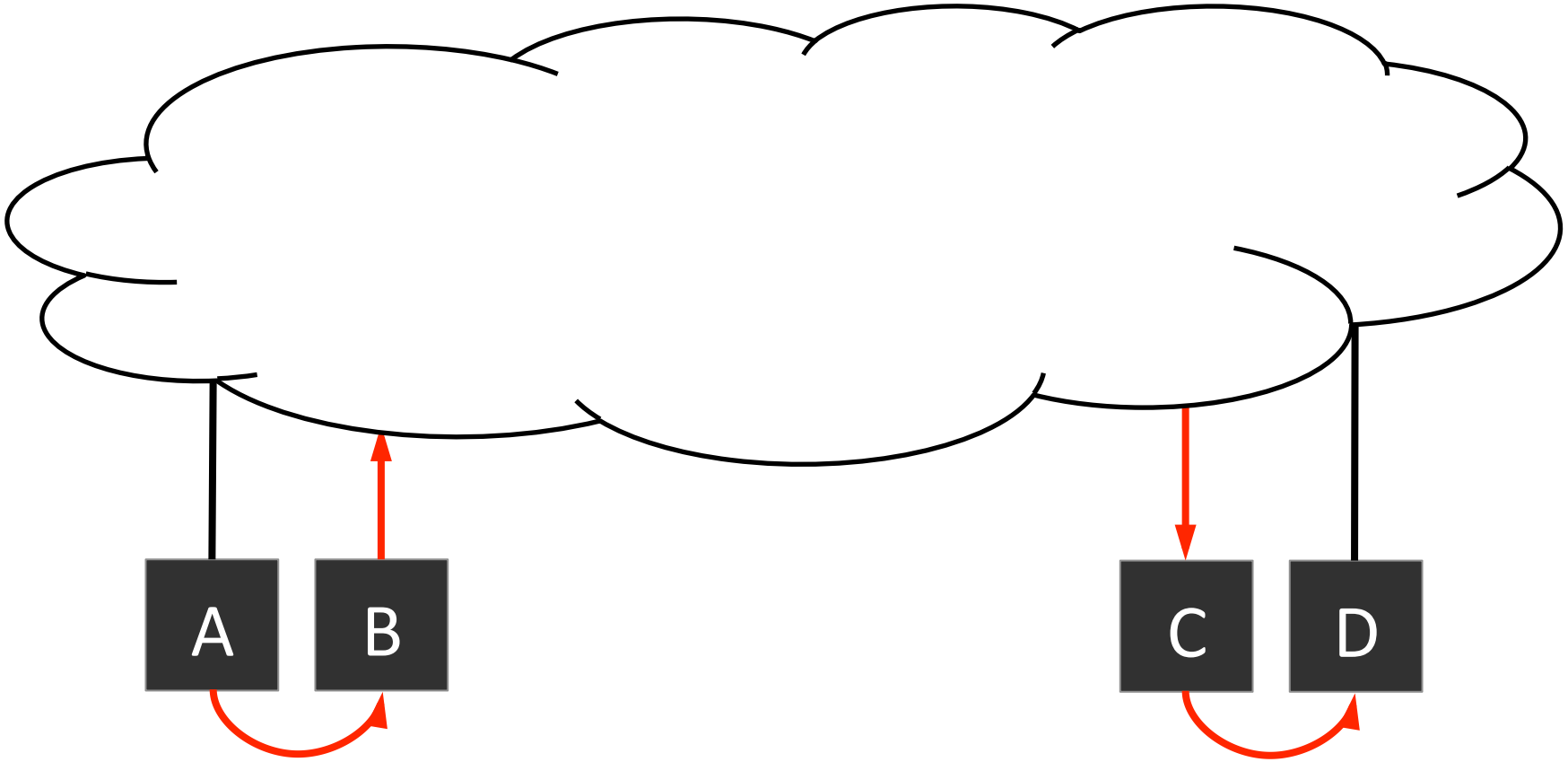
Routing In a GRIN Topology



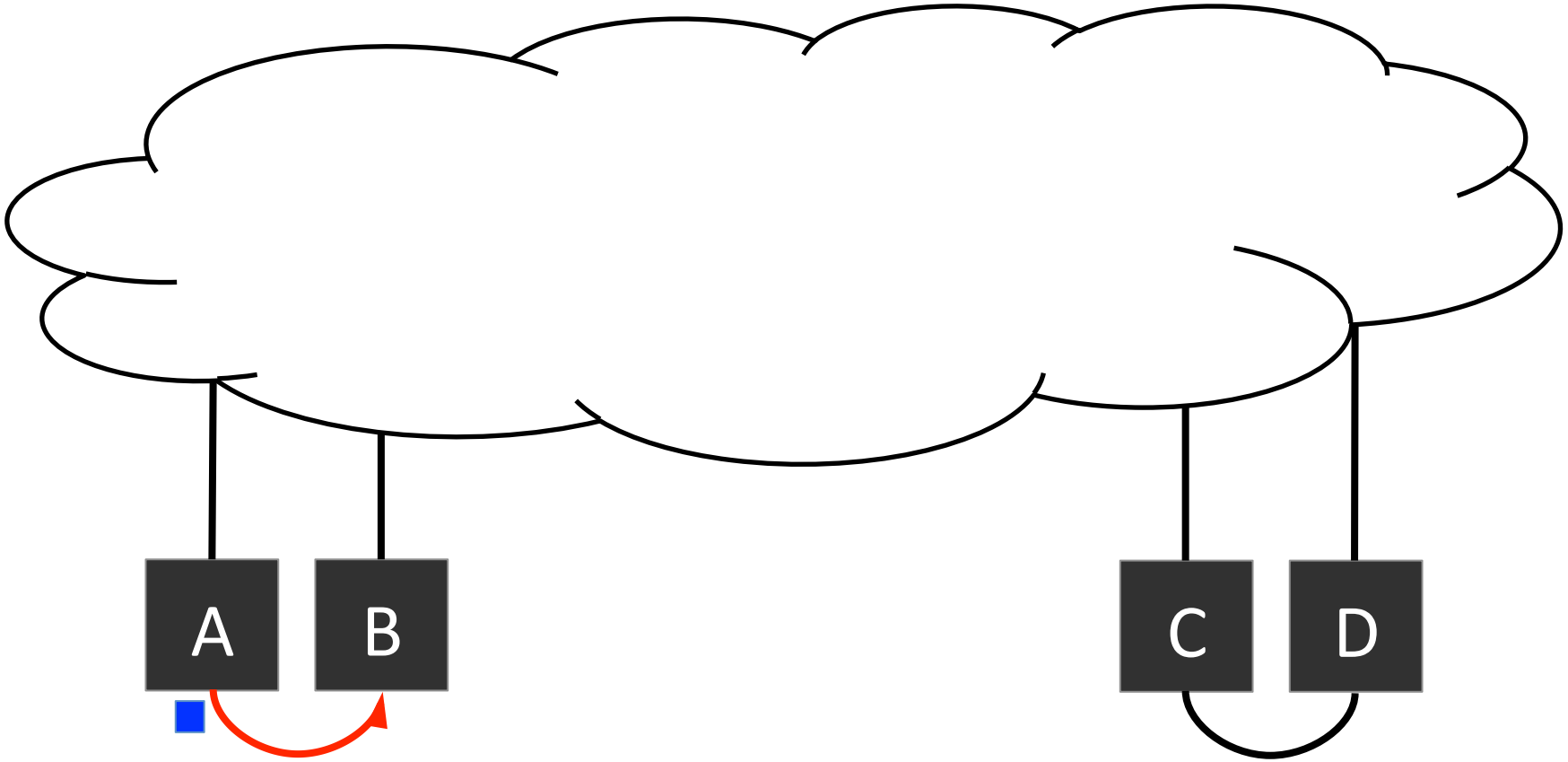
Routing In a GRIN Topology



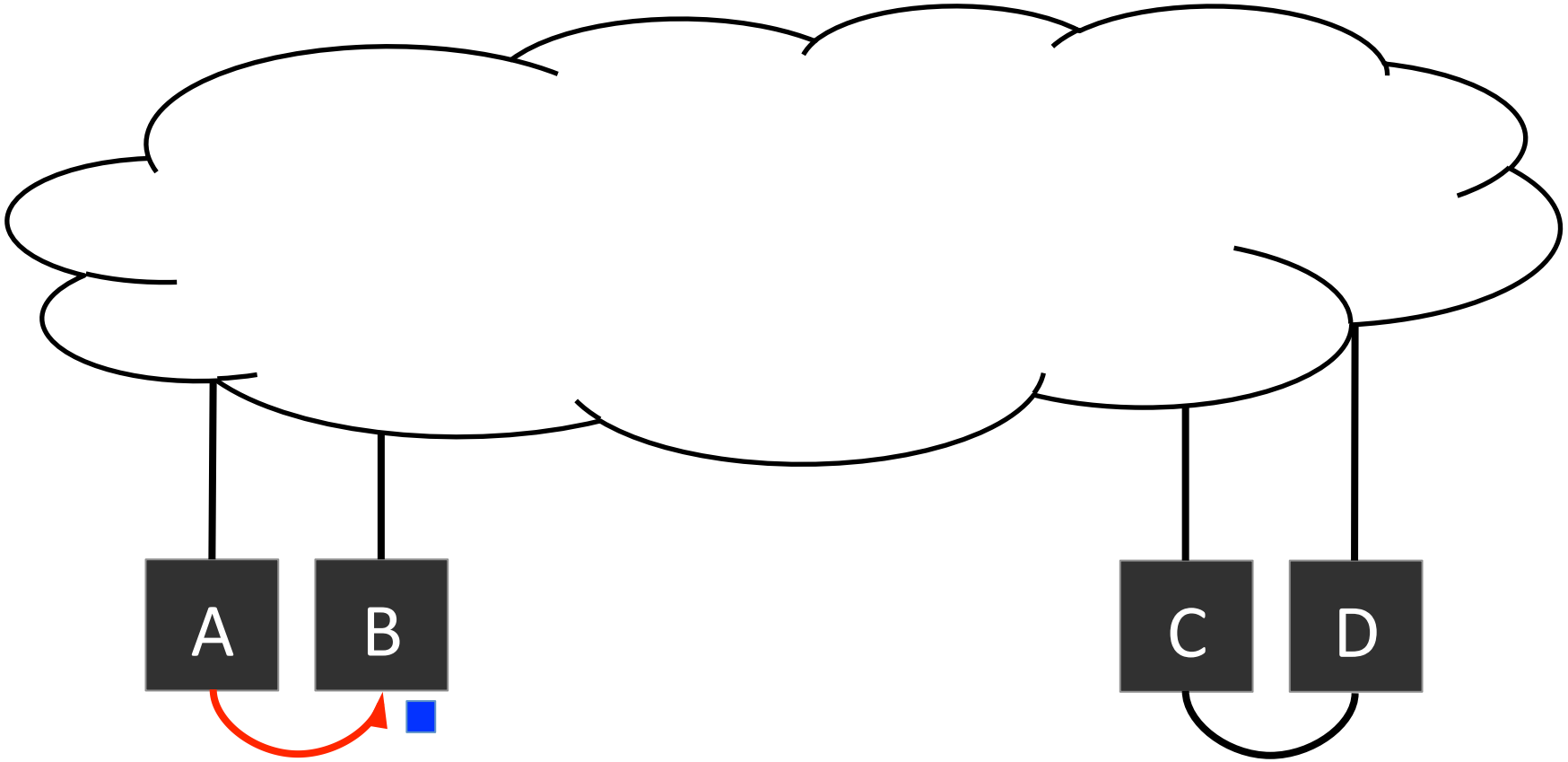
Routing In a GRIN Topology



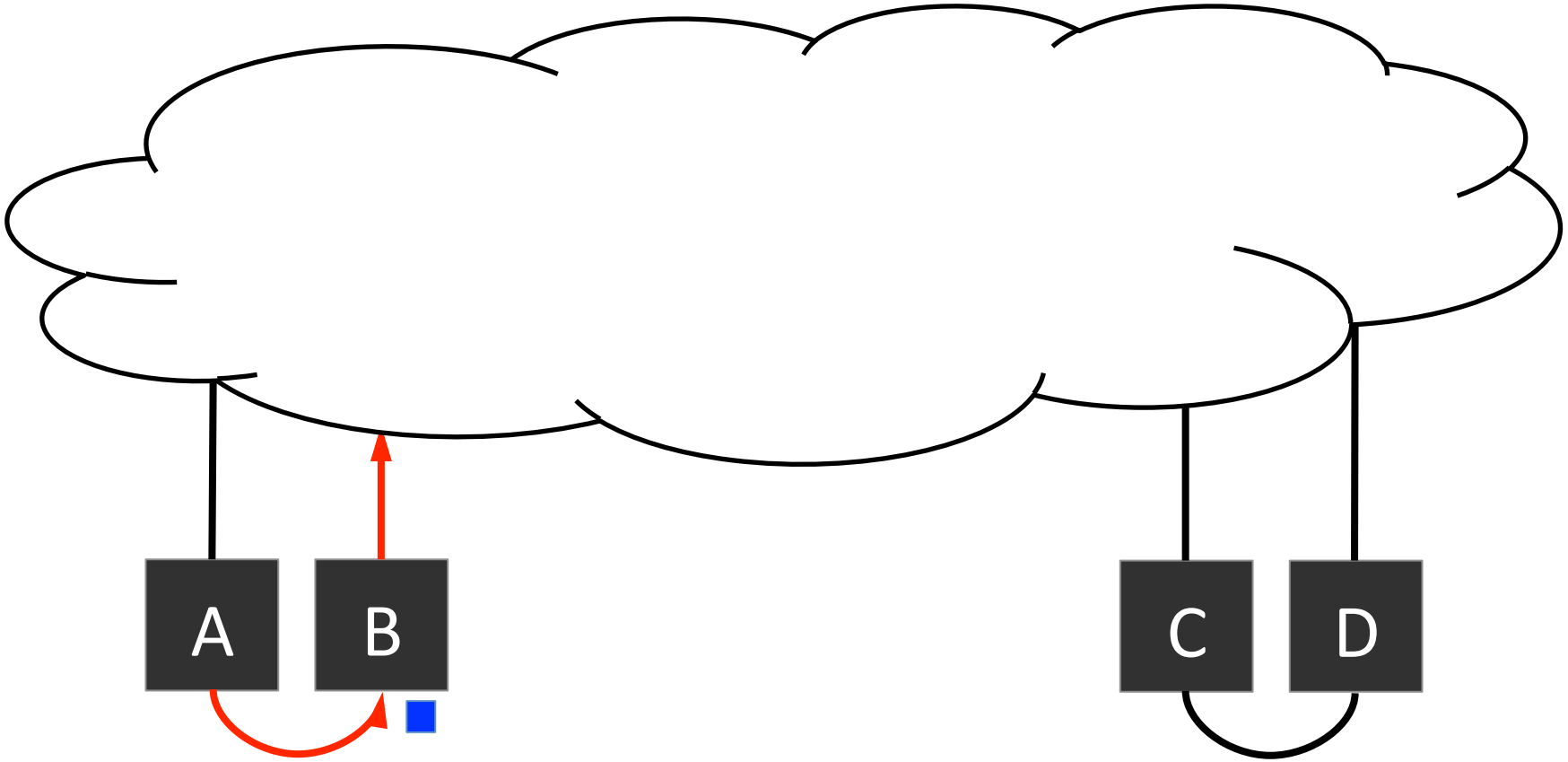
Routing In a GRIN Topology



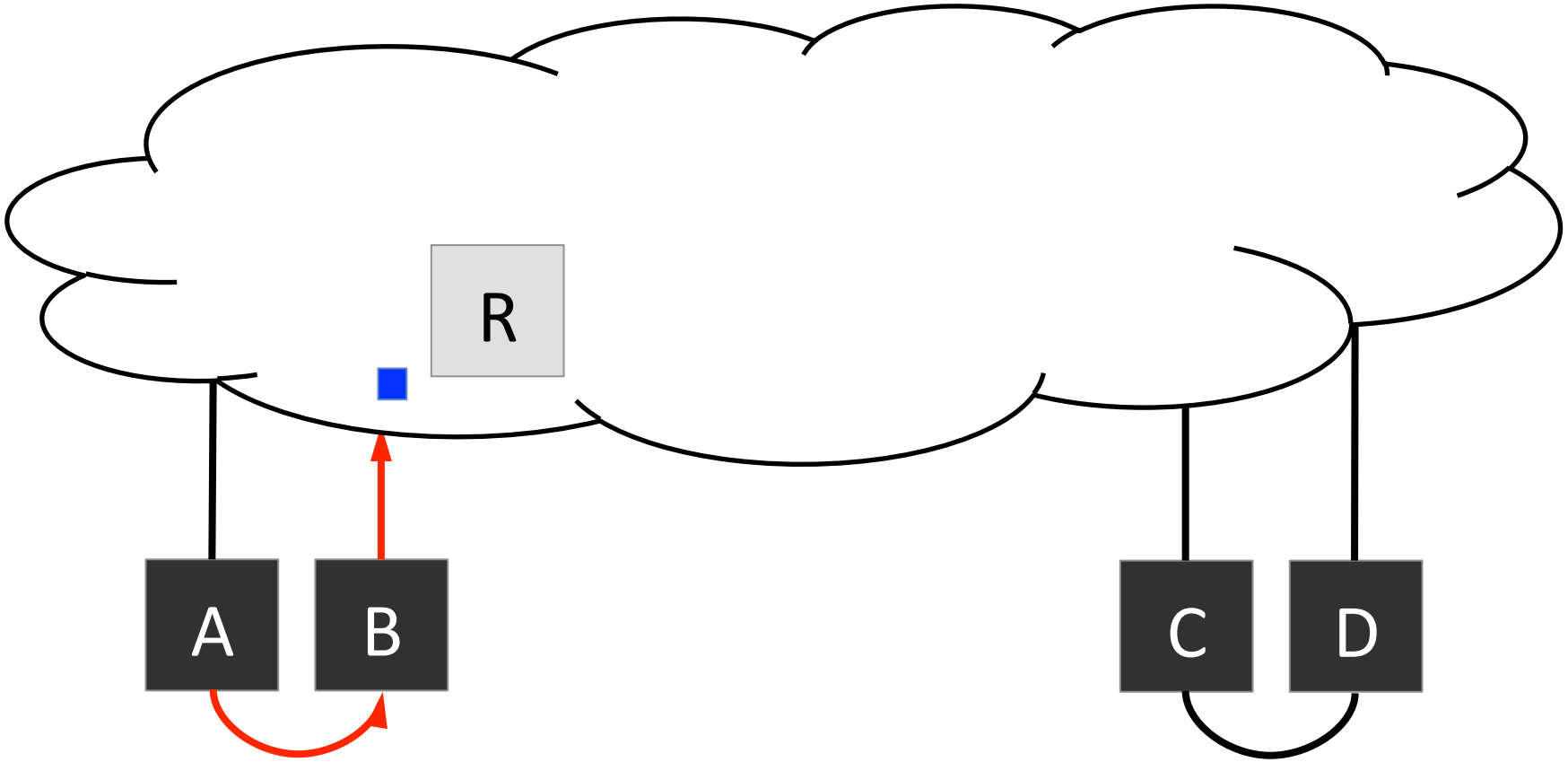
Routing In a GRIN Topology



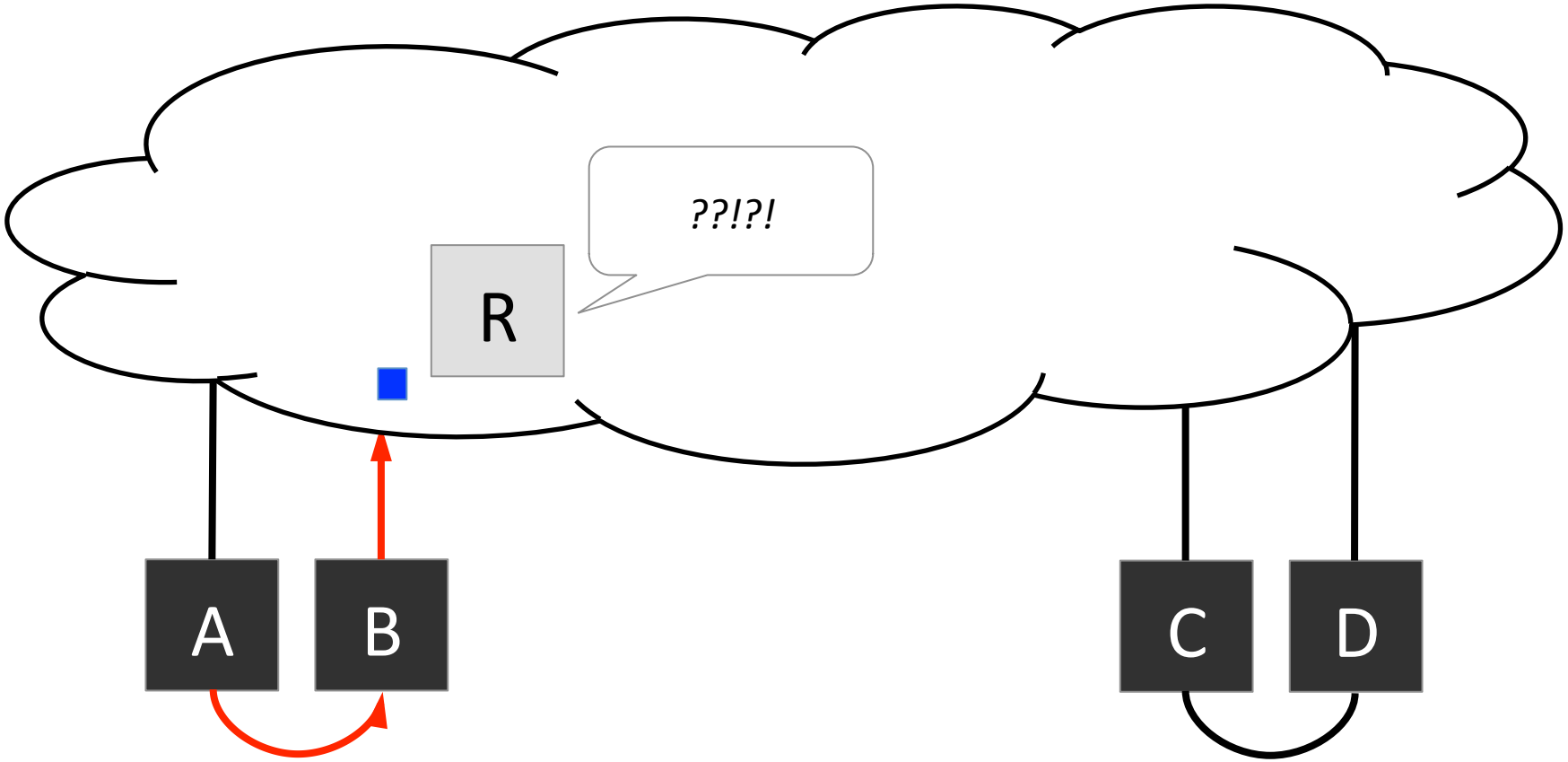
Routing In a GRIN Topology



Routing In a GRIN Topology



Routing In a GRIN Topology



The GRIN Routing Problem

- Secondary interfaces should be reachable
- We want to rely as much as possible on the routing scheme of the original network

The GRIN Routing Problem

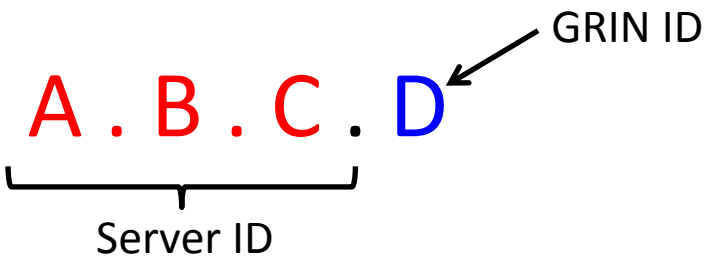
- Secondary interfaces should be reachable
- We want to rely as much as possible on the routing scheme of the original network
- Unsatisfactory solutions:

- source routing
- running the routing protocol (OSPF, IS-IS) down to the servers

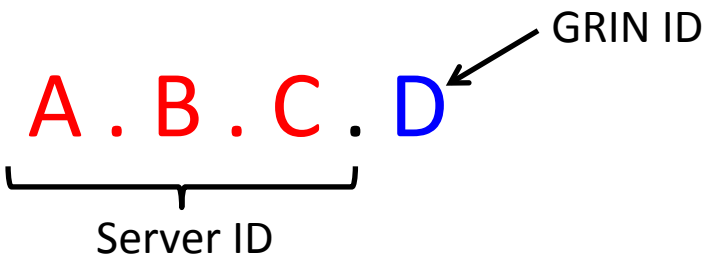
GRIN Addressing Scheme

- We use a custom addressing scheme that also conveys routing information

GRIN Addressing Scheme

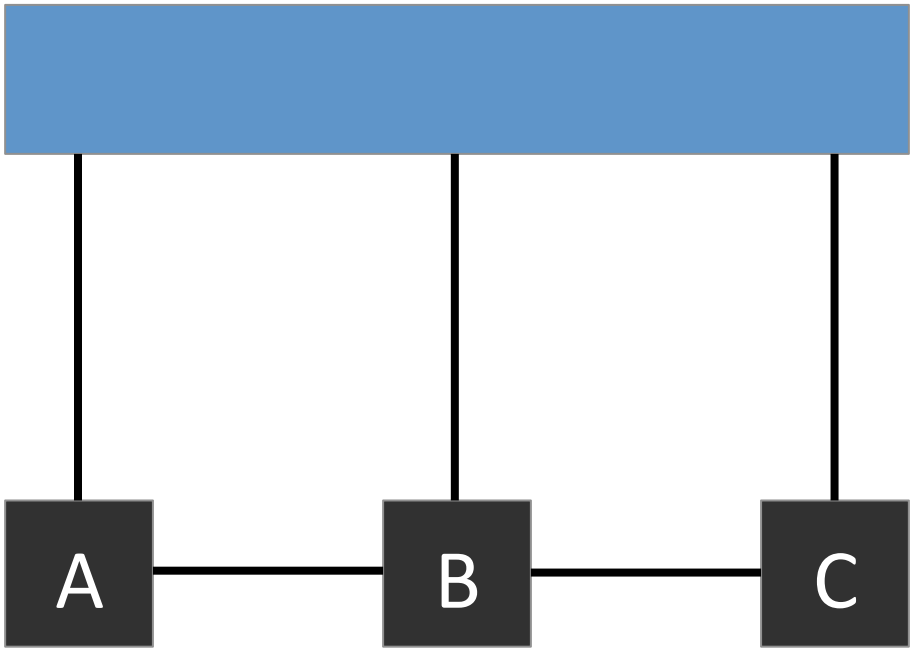


GRIN Addressing Scheme

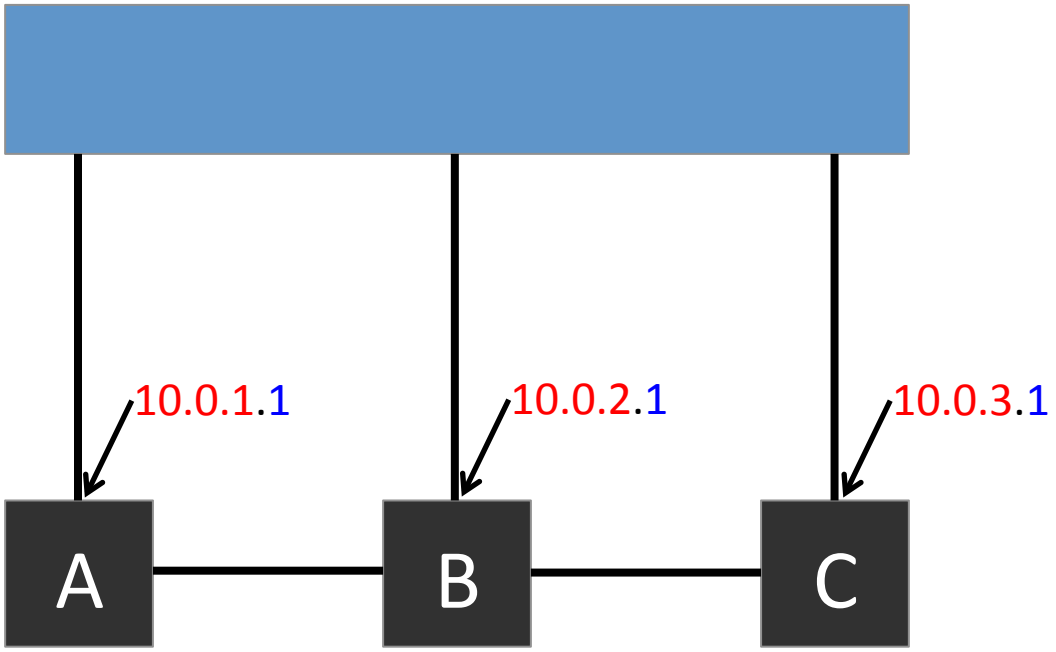


Example:
Server ID(*127.0.0.1*) = 127.0.0
GRIN ID(*127.0.0.1*) = 1

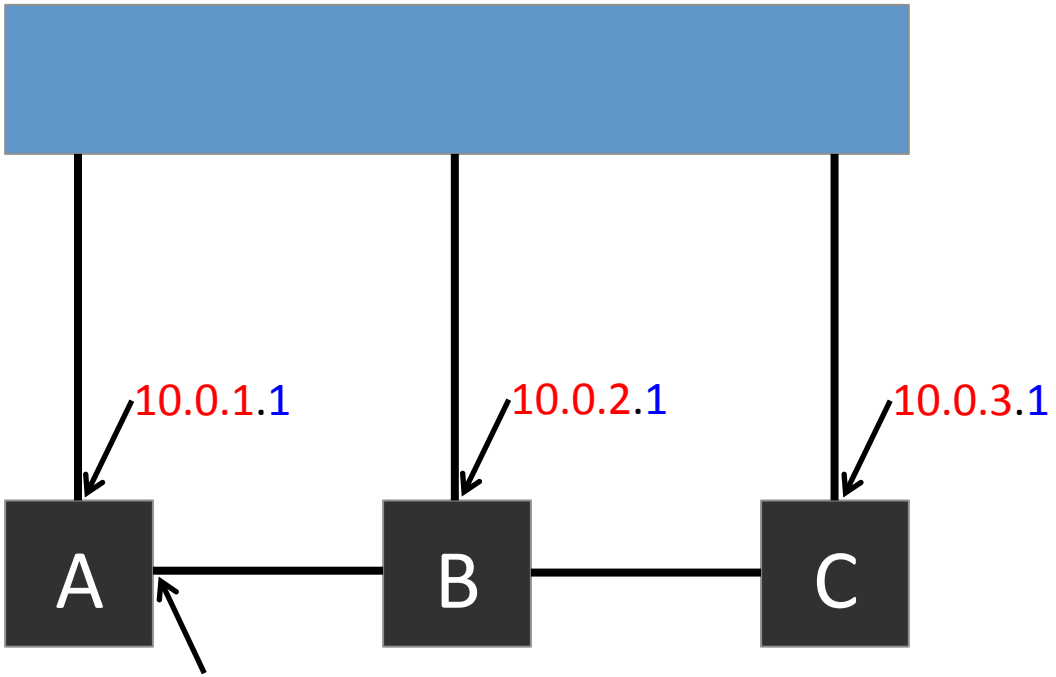
Address Assignment



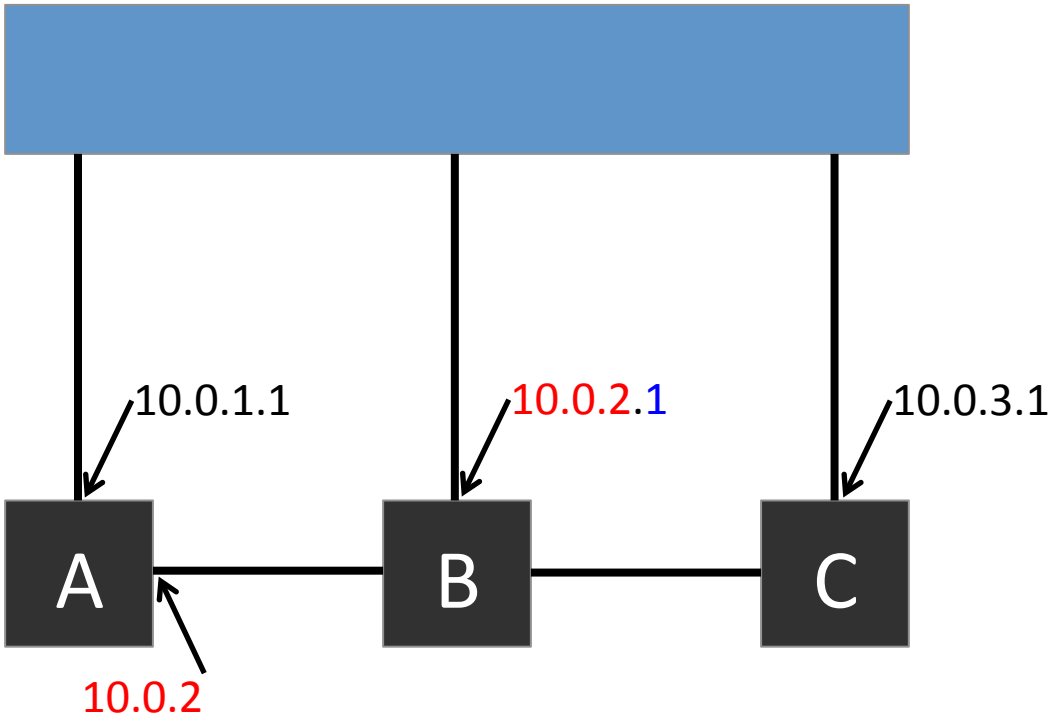
Primary Addresses



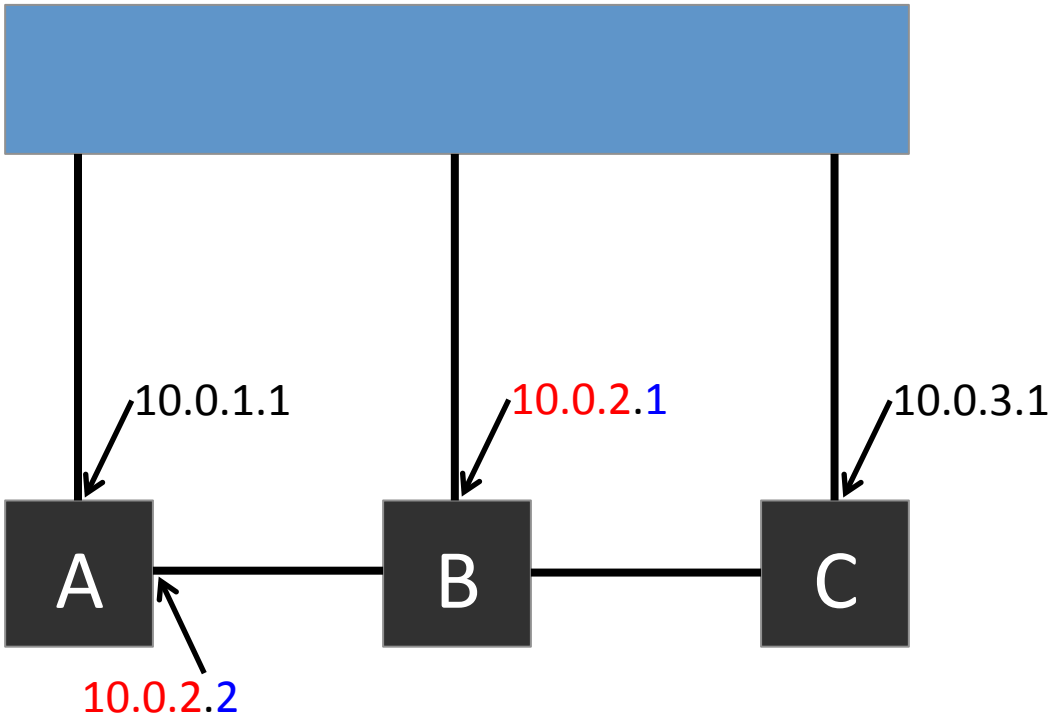
Secondary Addresses



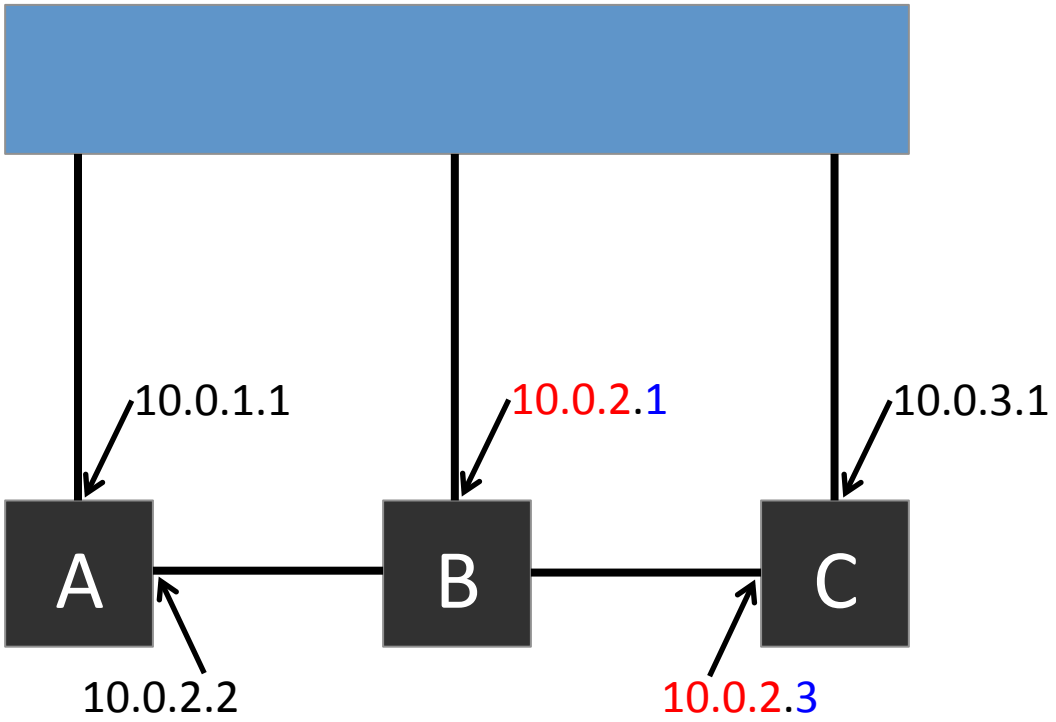
Secondary Addresses



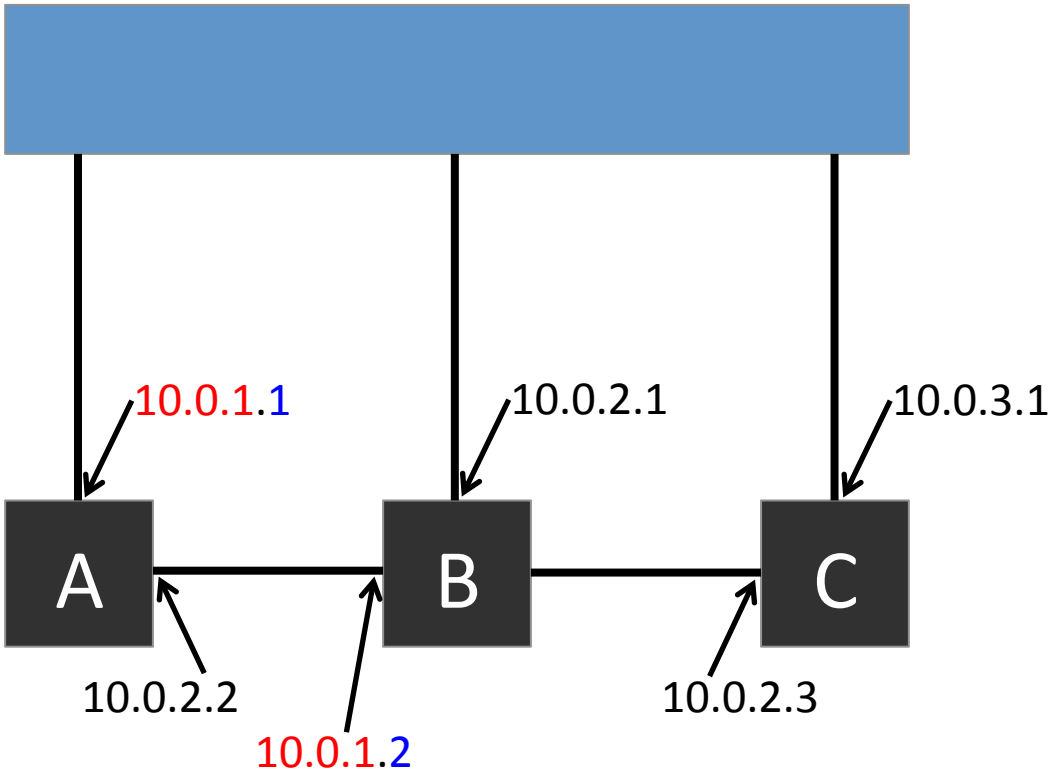
Secondary Addresses



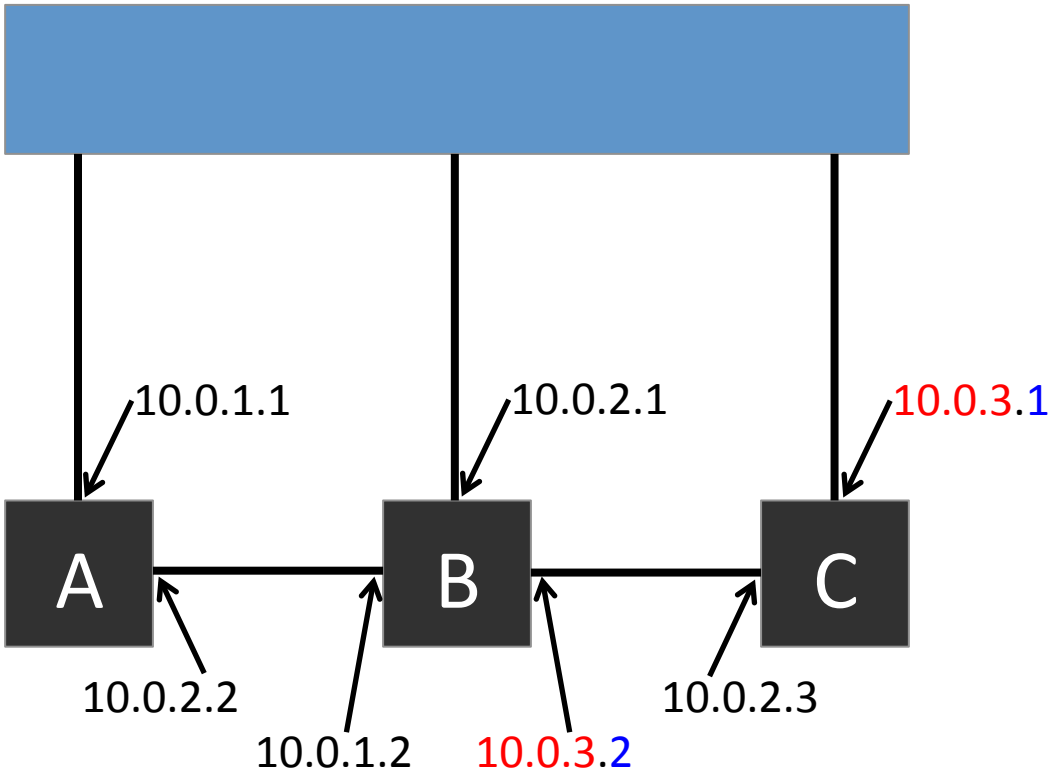
Secondary Addresses



Secondary Addresses



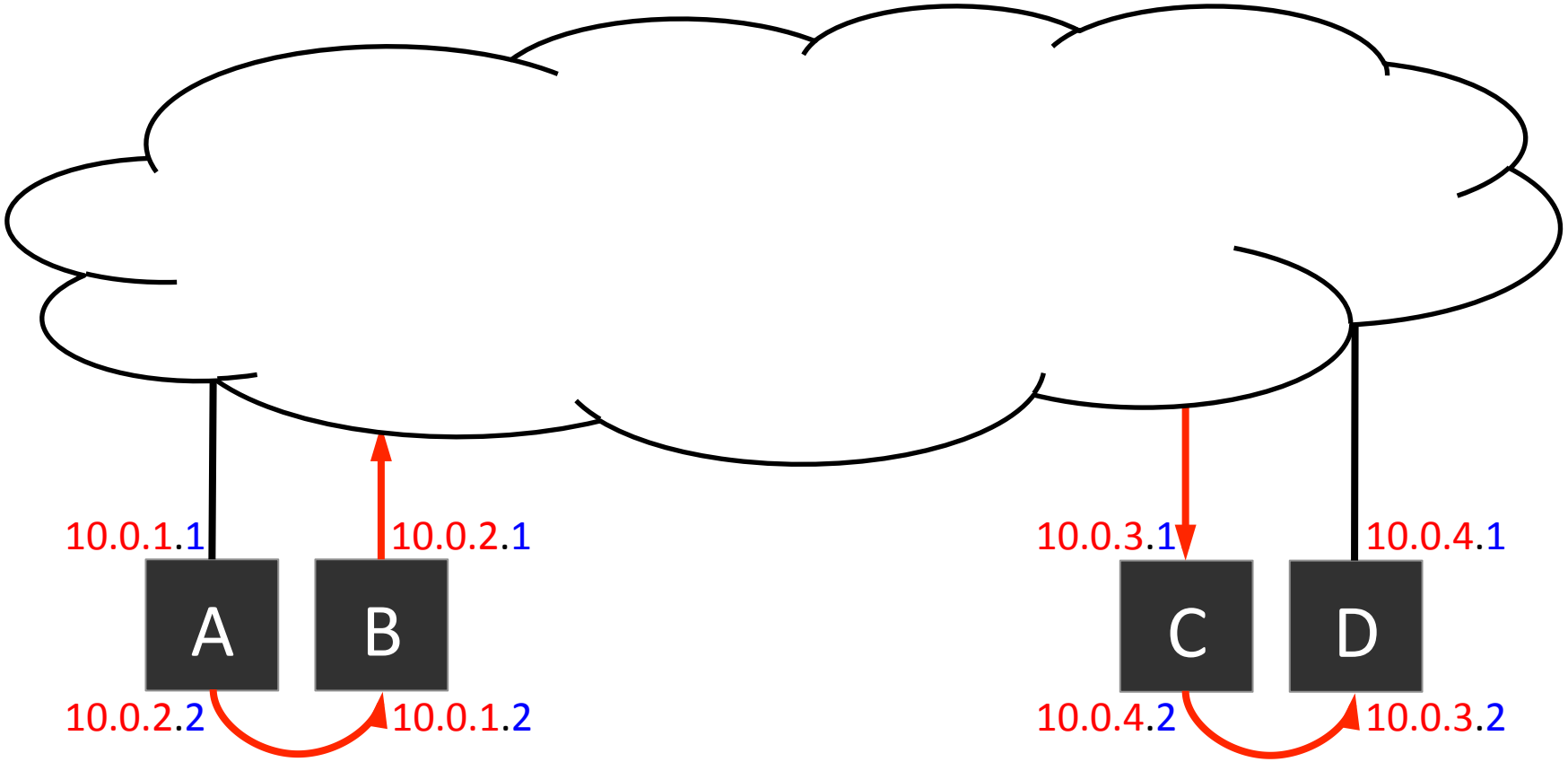
Secondary Addresses



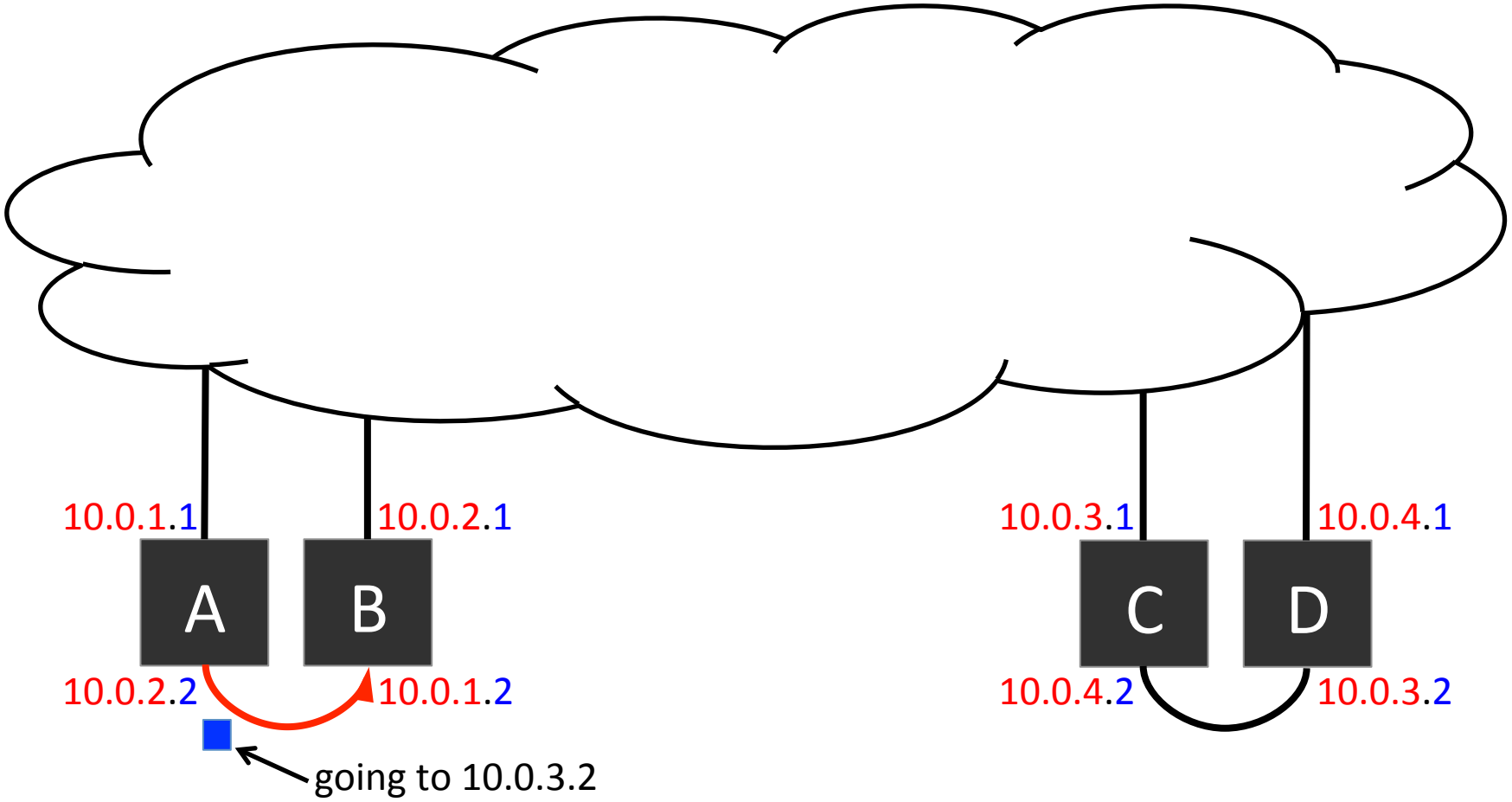
GRIN Addressing Scheme

- Easy to distinguish primary from secondary addresses
- Easy to determine the primary address of the neighbour associated with a secondary link

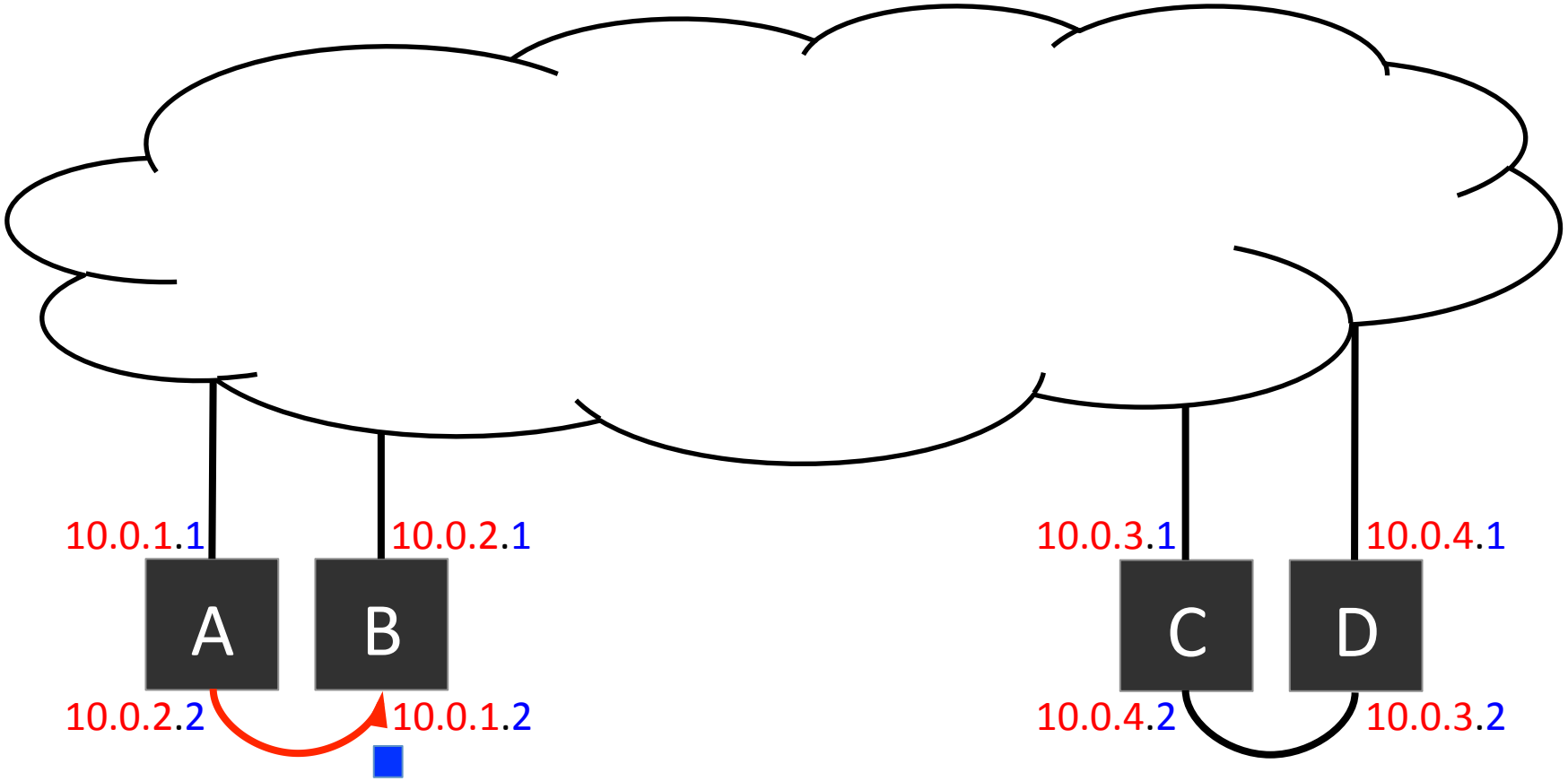
Routing In a GRIN Topology



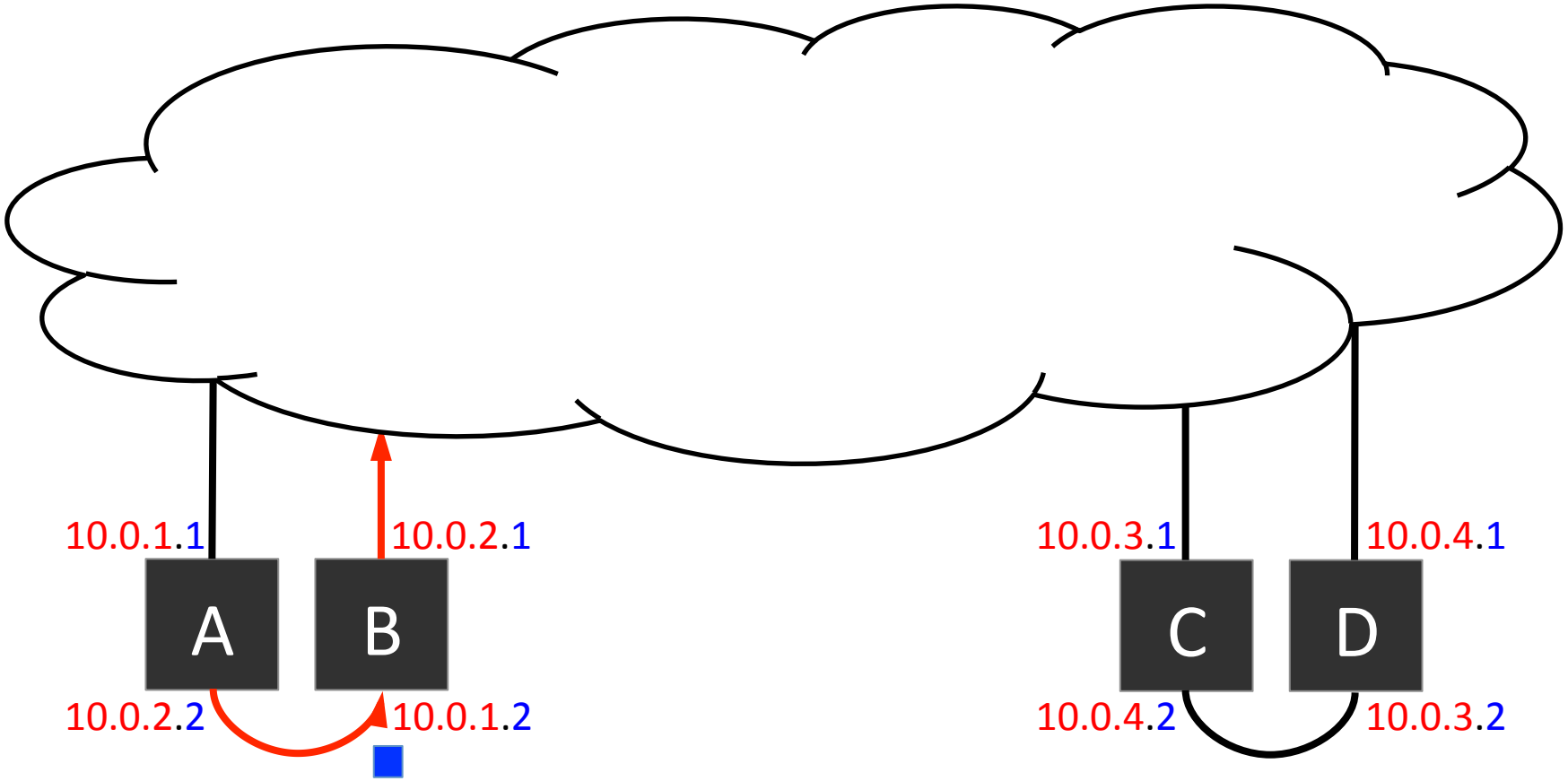
Routing In a GRIN Topology



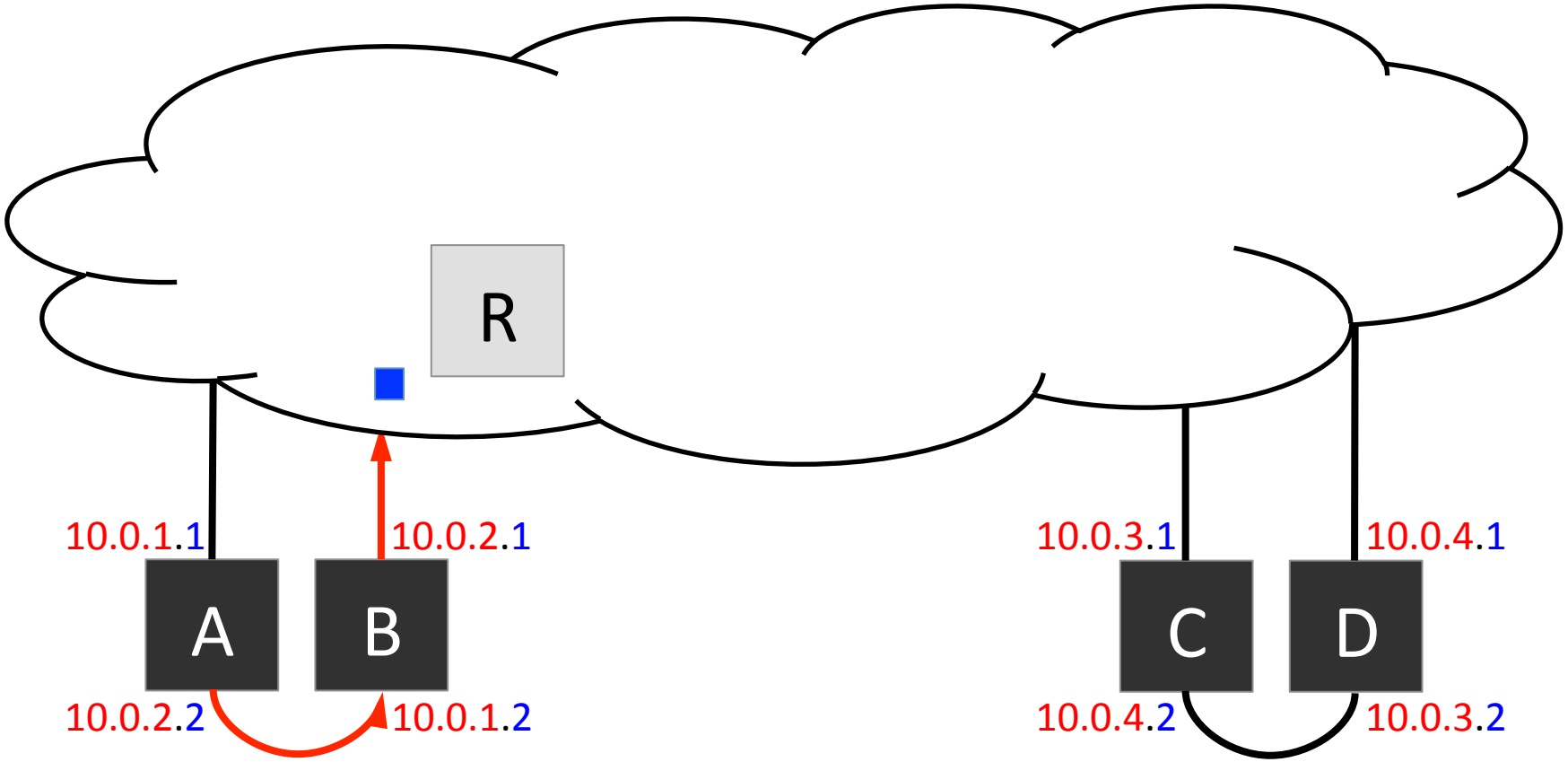
Routing In a GRIN Topology



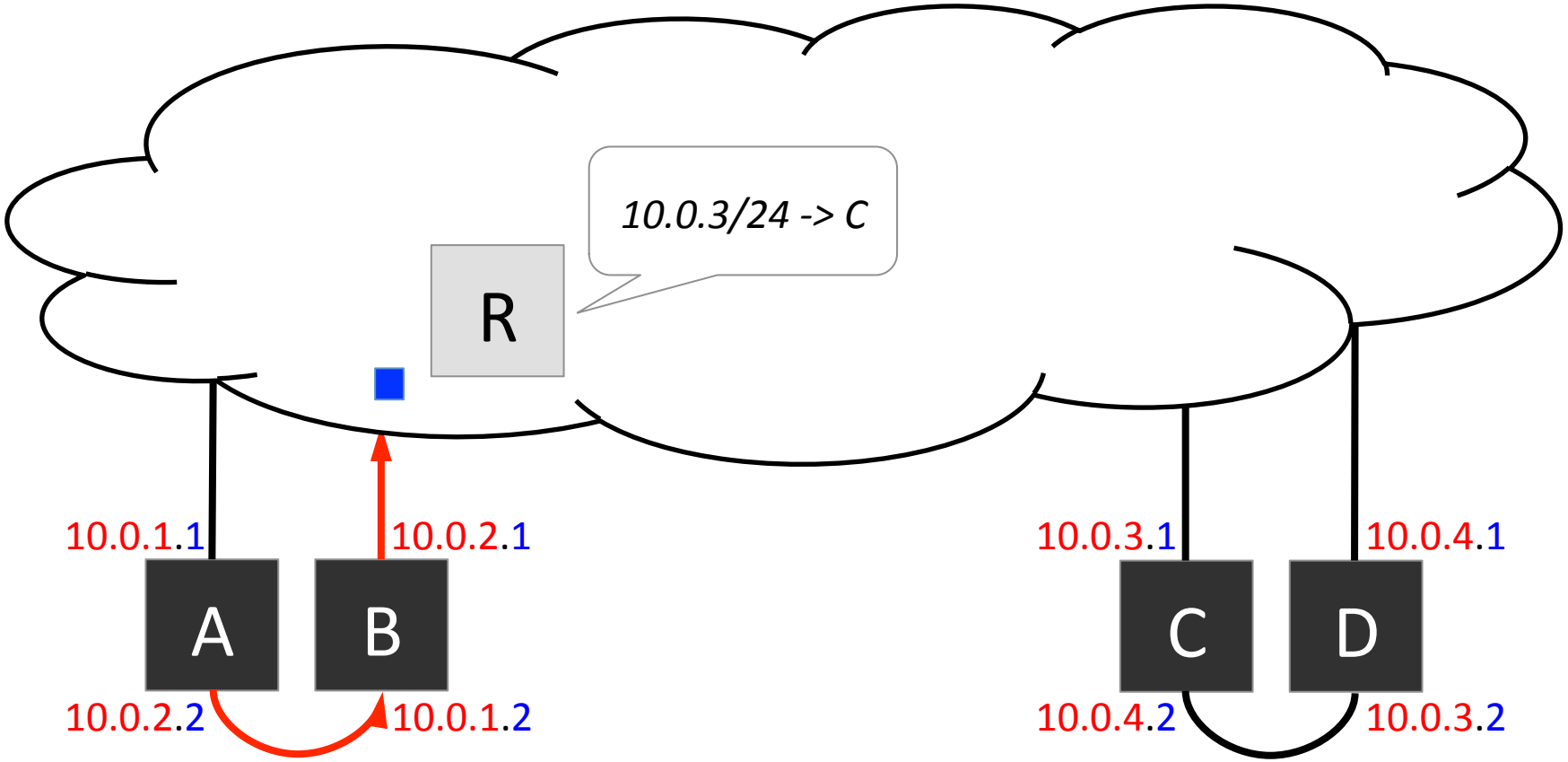
Routing In a GRIN Topology



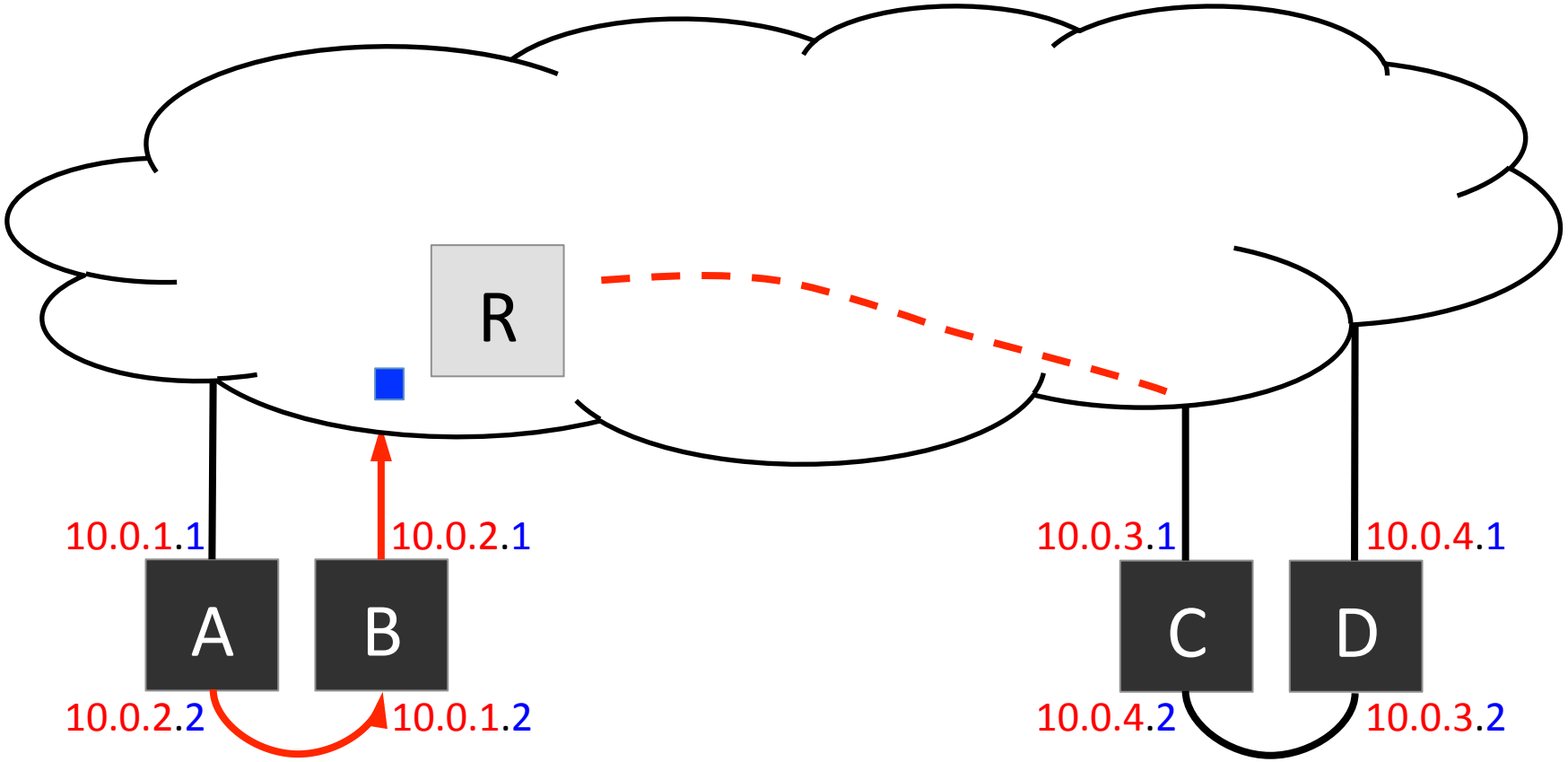
Routing In a GRIN Topology



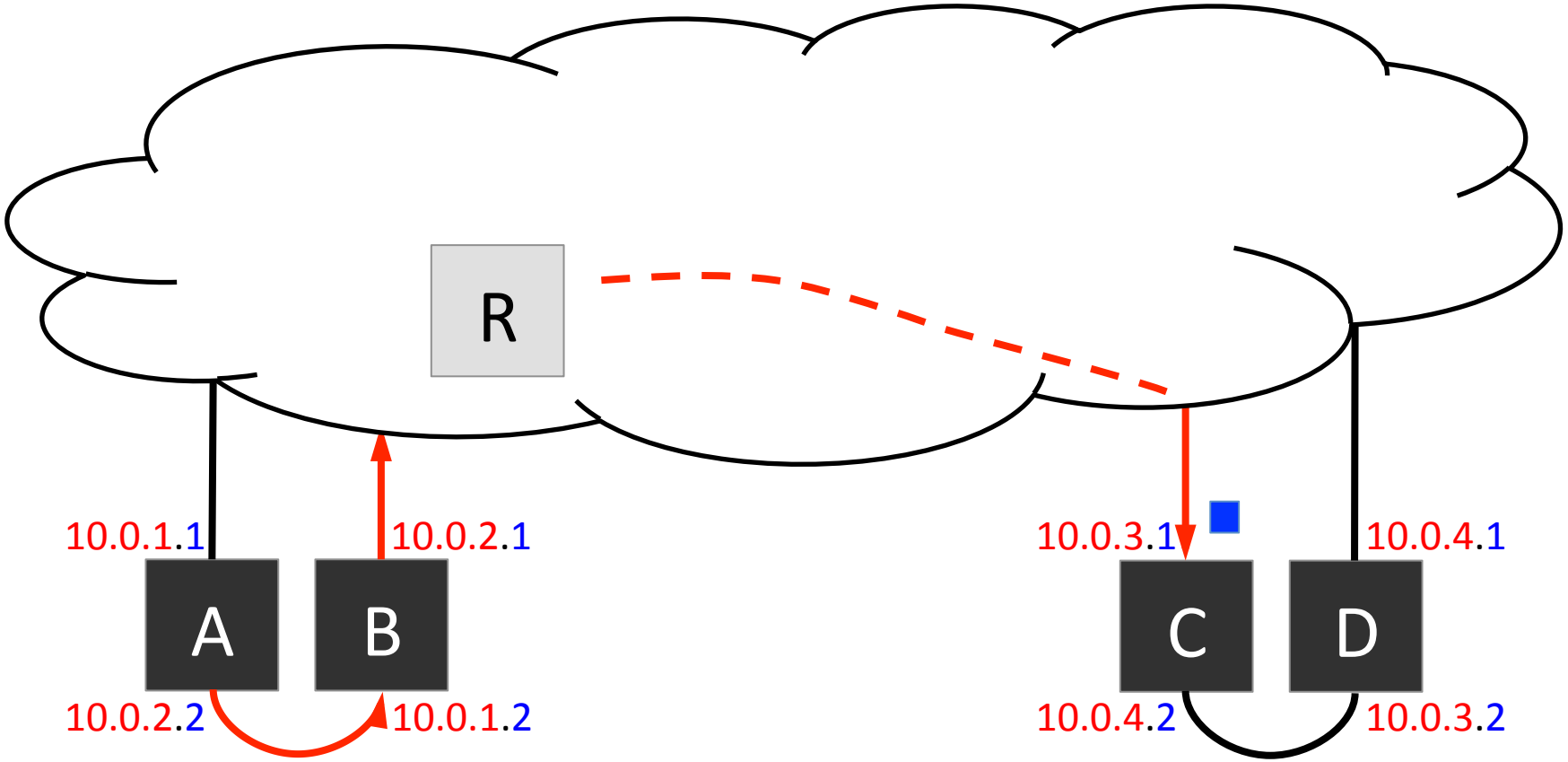
Routing In a GRIN Topology



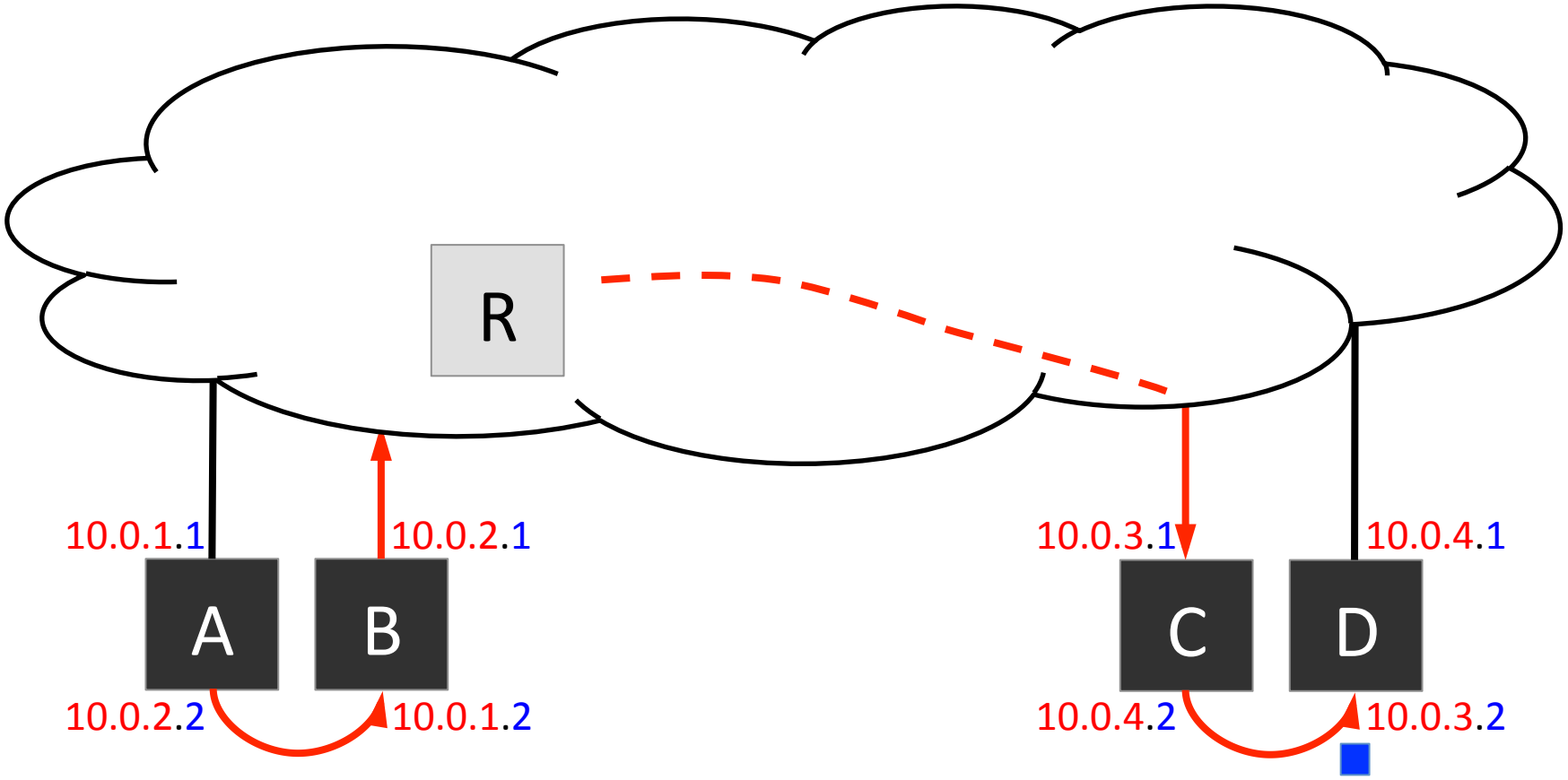
Routing In a GRIN Topology



Routing In a GRIN Topology



Routing In a GRIN Topology



Using GRIN

Opportunistic Usage

- Don’t change anything about applications and the way we use them

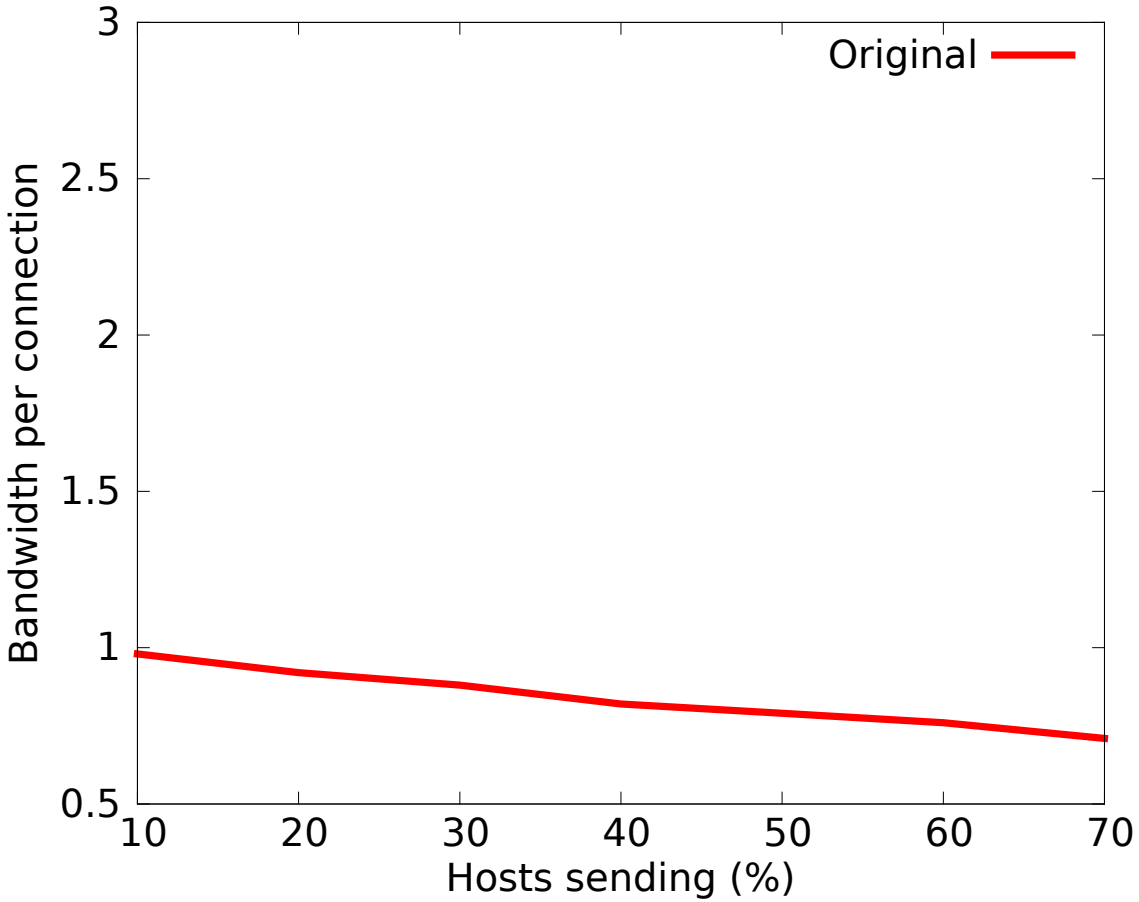
Evaluation

- Simulated (120 server topology)
- Micro cluster (10 servers)
  - 1Gbps
  - 10Gbps
- EC2 (1000 c3.large instances)
  - interfaces limited to 100Mbps

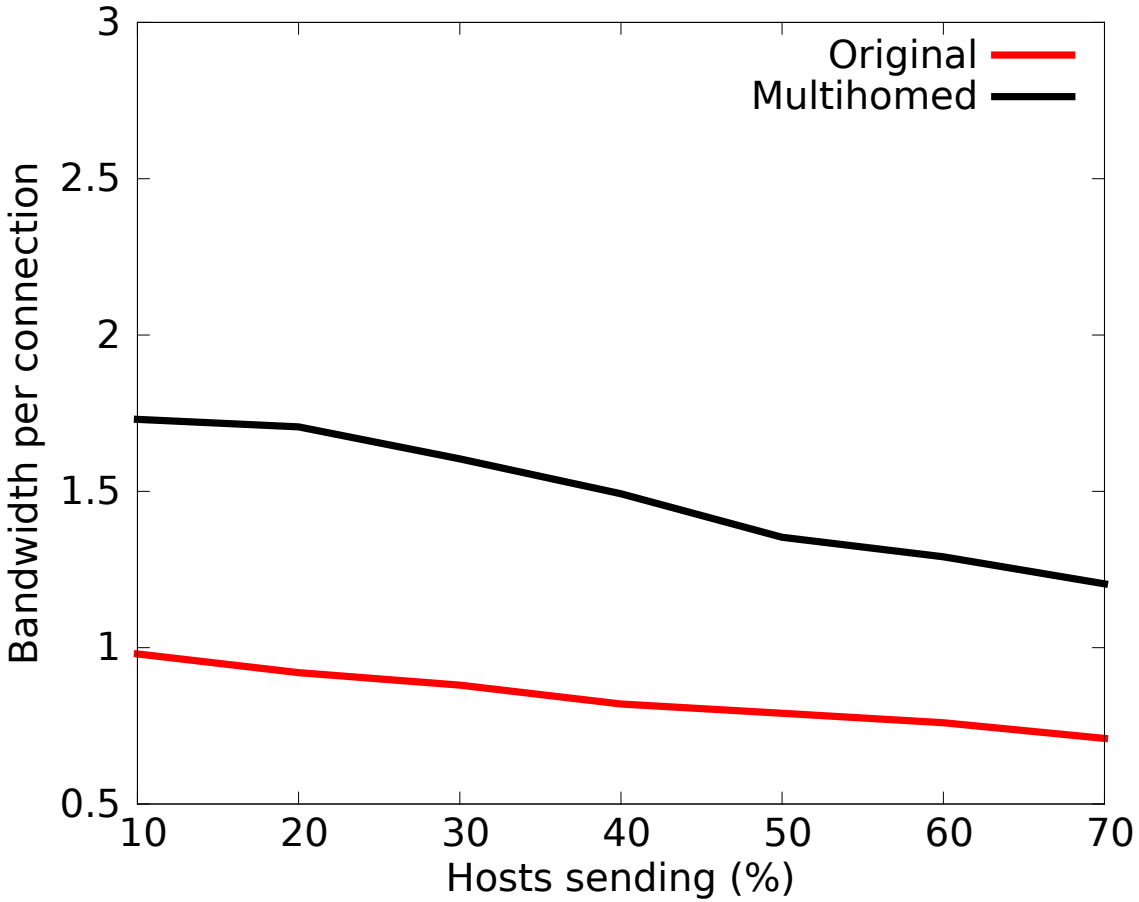
Random Traffic

- A certain number of servers are active
- Each active server send data flat-out to a destination chosen at random

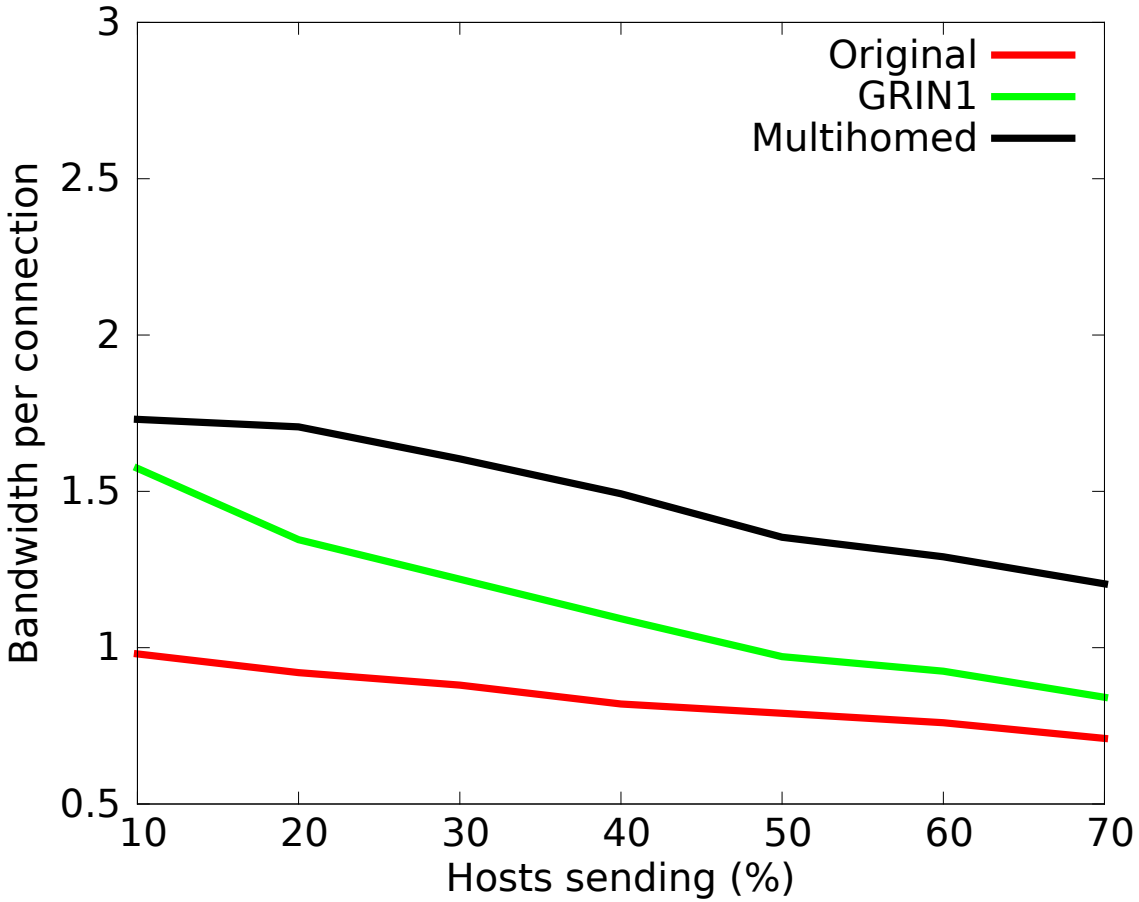
Random Traffic



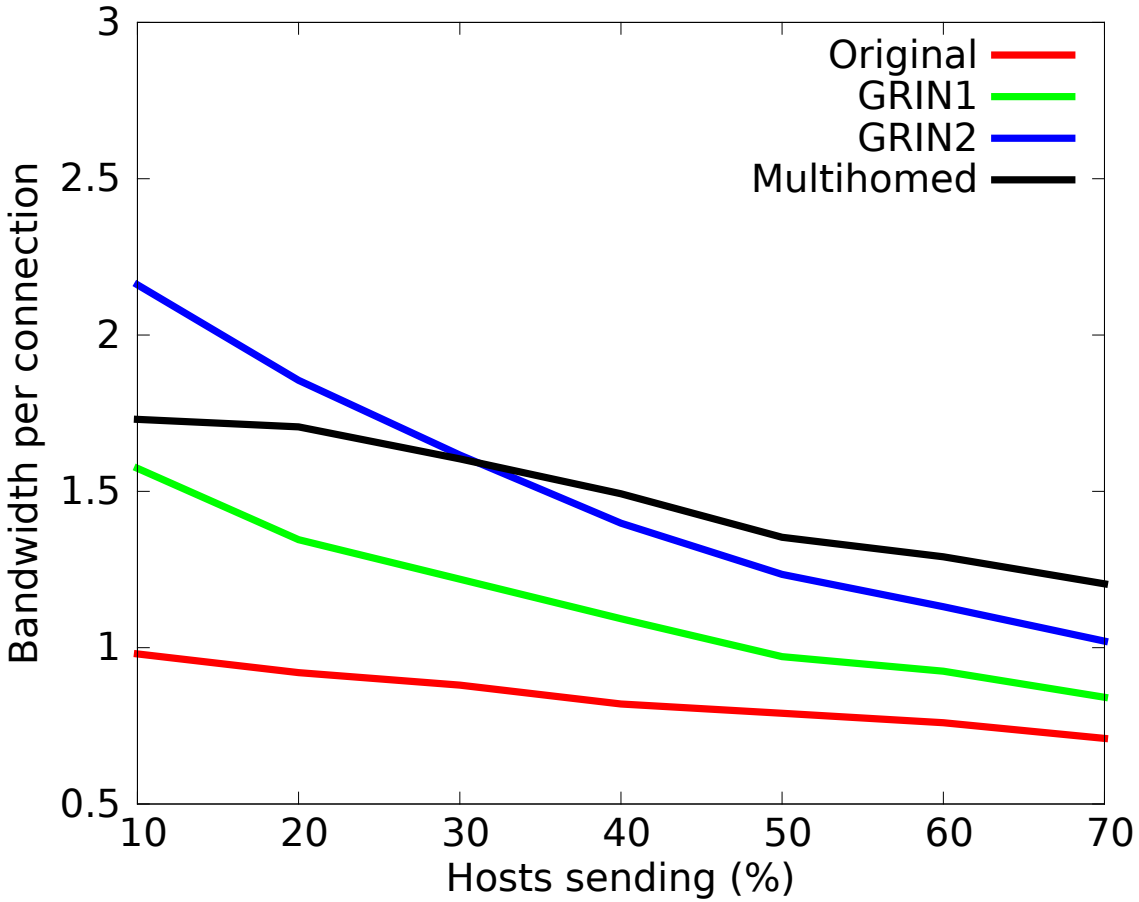
Random Traffic



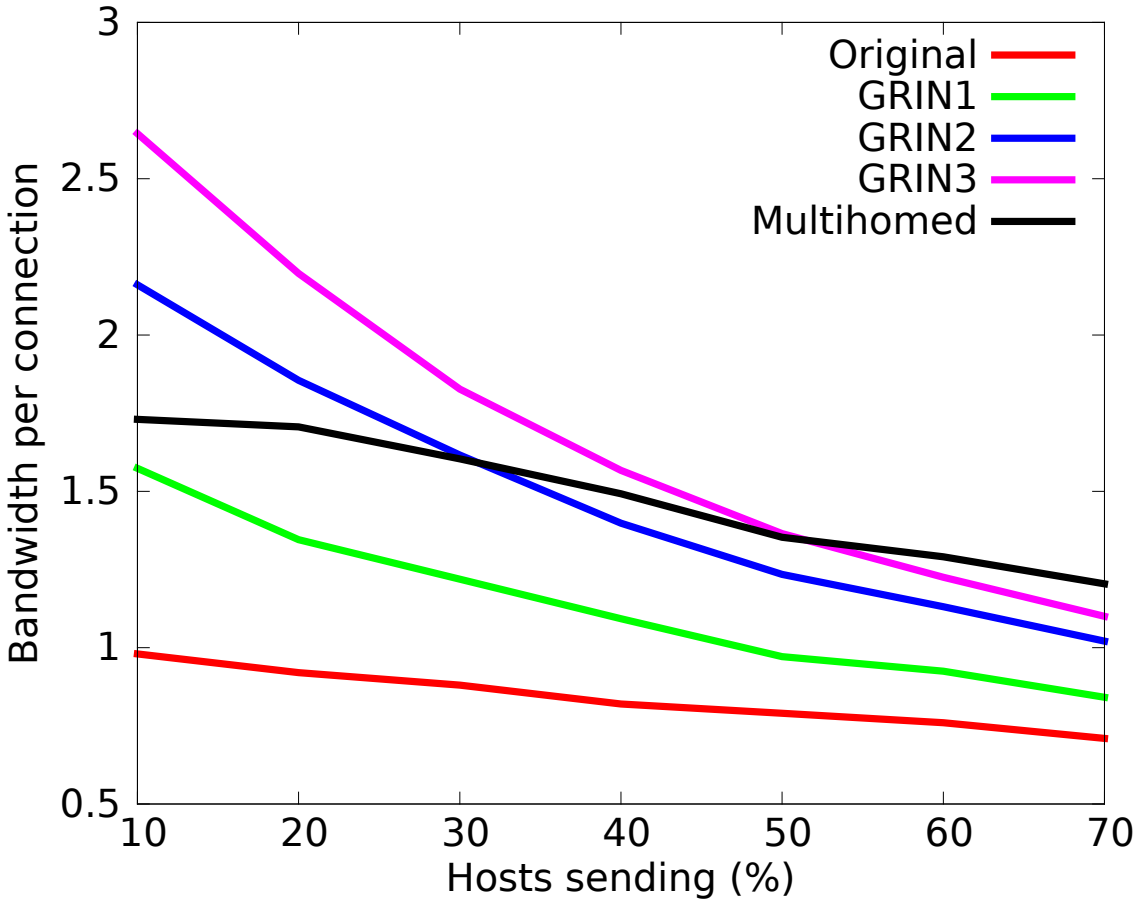
Random Traffic



Random Traffic



Random Traffic



Application Performance

- With 1Gbps links

Application	GRIN1	GRIN2
NFS (files > 1MB)	1.9x	2.5x
HDFS	2x	2.6x (CPU bound) 3x (for 2 clients)
Xen Virtual Machine Migration	1.6x	2x

Application Performance

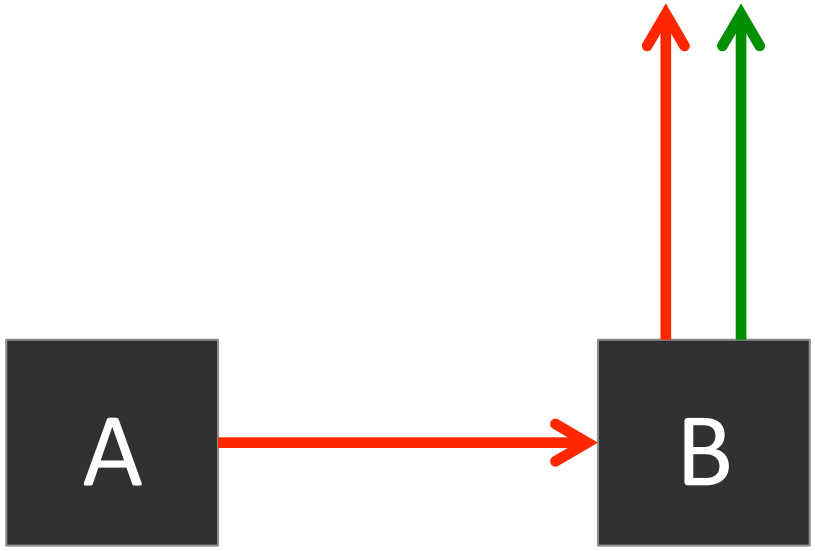
- At 10Gbps, apps easily become CPU bound (on our machines)

- We needed multiple concurrent clients to increase bandwidth utilisation at the server

- Two NFS clients fully utilised both links available to one server
- Three Spark workers needed to fully utilise one DFS node for word count

Negative Impact of Opportunistic Usage

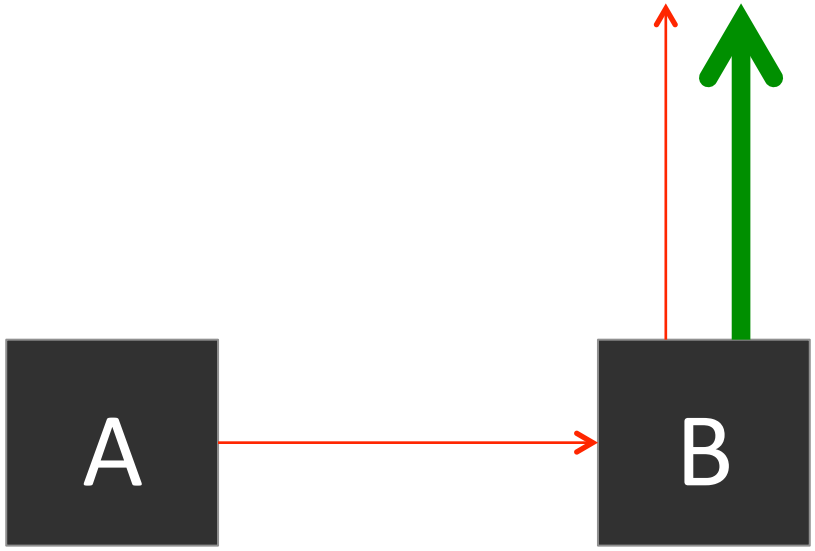
Fair Use of Uplinks



Fair Use of Uplinks

- DSCP-based prioritization scheme
- Direct flows receive high priority
- Secondary flows receive low priority

Fair Use of Uplinks



Latency

- Increased buffer pressure
- One latency-sensitive direct flow vs. multiple secondary flows

Latency

- Outgoing on a server uplink

- At an egress port of a switch

Latency

- Outgoing on a server uplink
  - 1.7 ms increase without priority
  - 240 us increase with priority (can be improved)

- At an egress port of a switch
  - 2.3 ms increase w/o priority
  - 10 us increase w priority

Forwarding Overhead

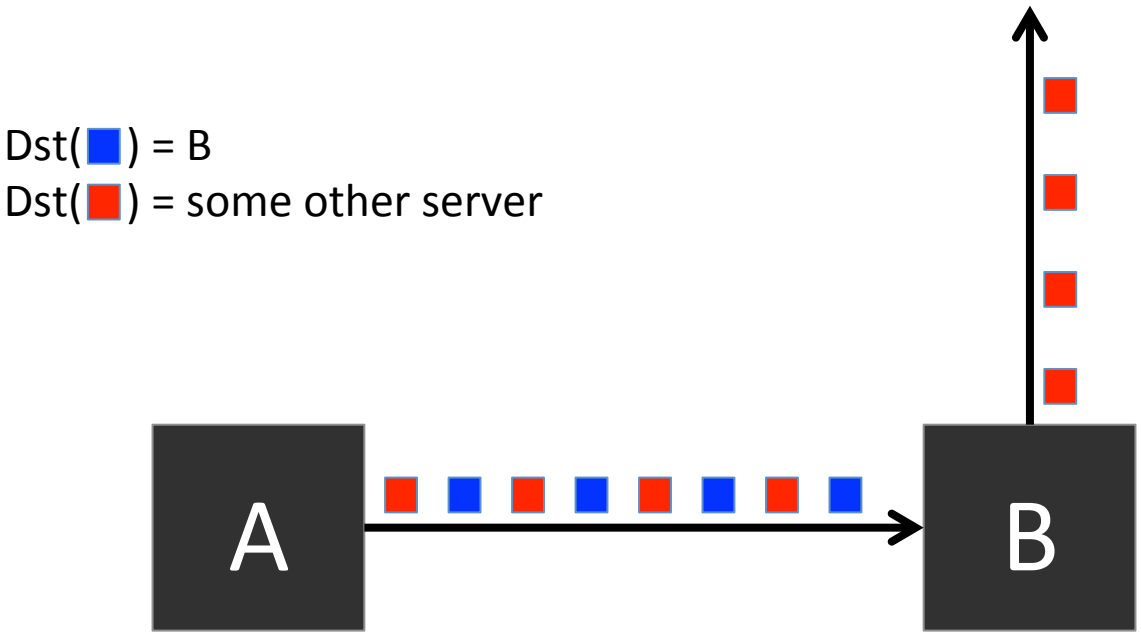
- Compute-intensive applications may suffer
- Three cases:
- Kernel compilation in a ramdisk
- Video transcoding
- A memcached server

Forwarding Overhead

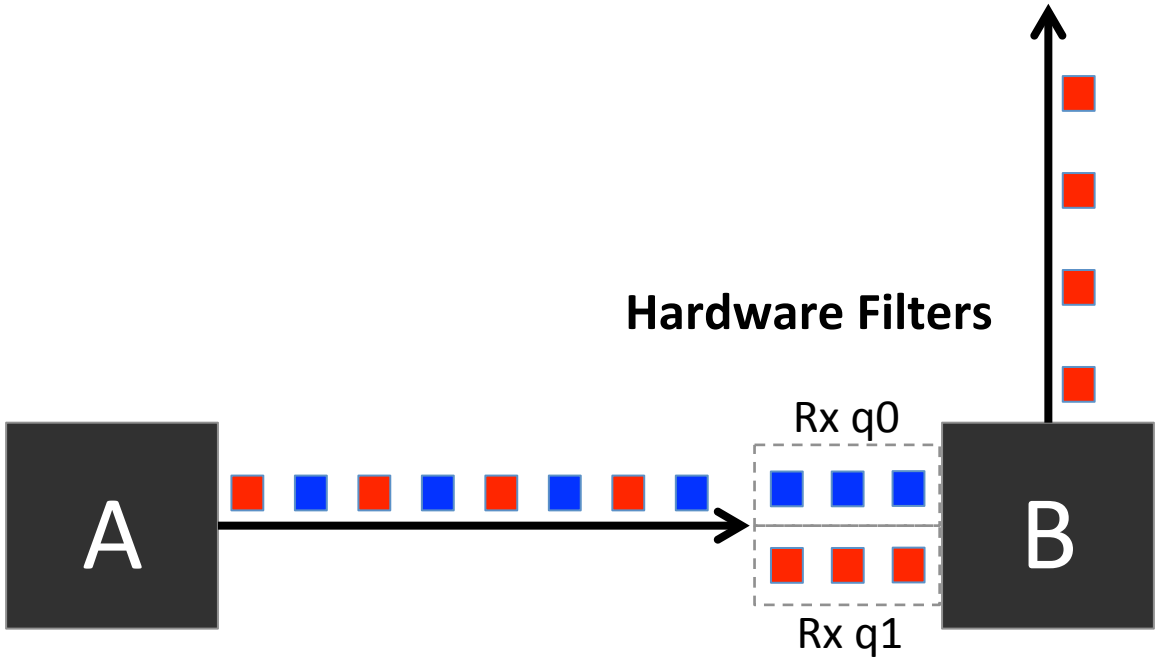
- Bidirectional 10Gbps traffic
- If one core is reserved for traffic forwarding, the impact is minimal < 1%
- Otherwise, apps may suffer a 8-13% performance hit

Zero-Overhead Forwarding

Packet Differentiation



Packet Differentiation

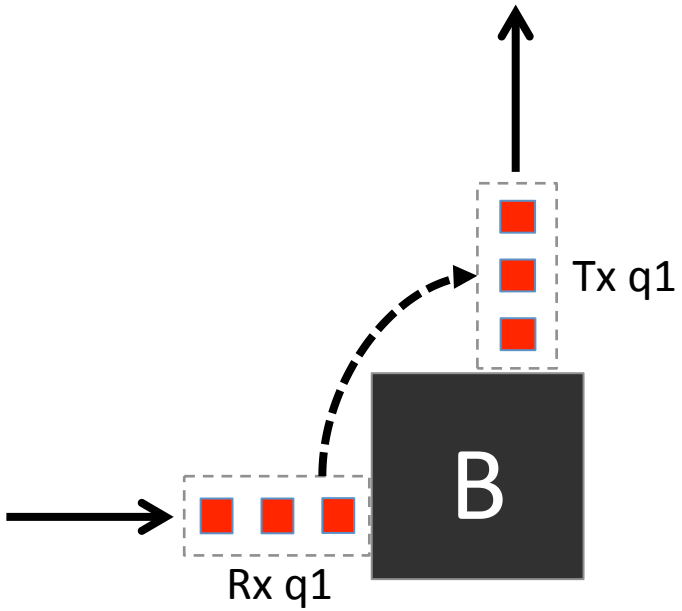


Header Field Rewriting

- Most important: update the destination MAC address

- We argue no such thing is required at forwarding servers in a GRIN topology

Queue Bridging



Queue Bridging

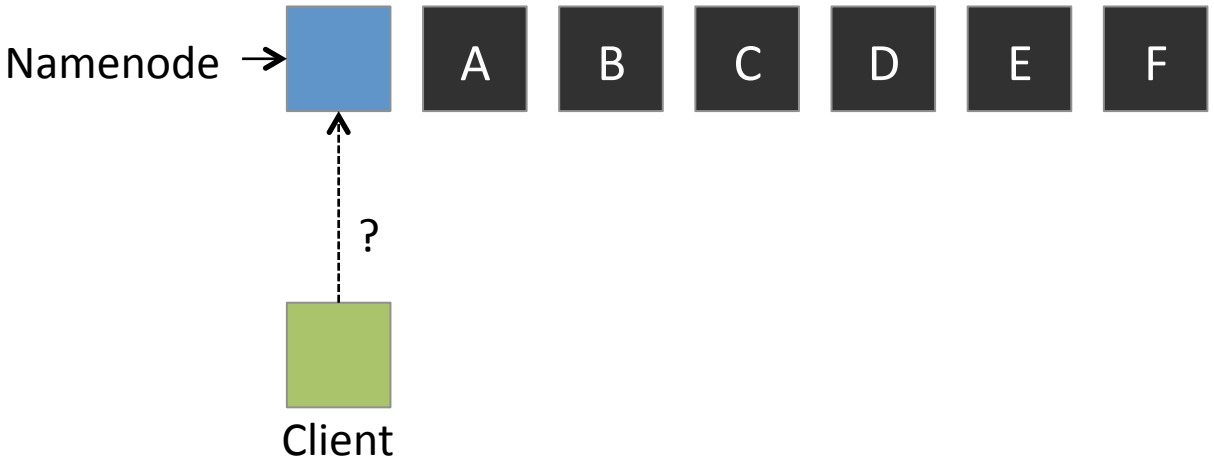
- Missing hardware functionality
- So far only possible from Tx to Rx and for queues of the same physical port

GRIN Aware Applications

GRIN Aware Applications

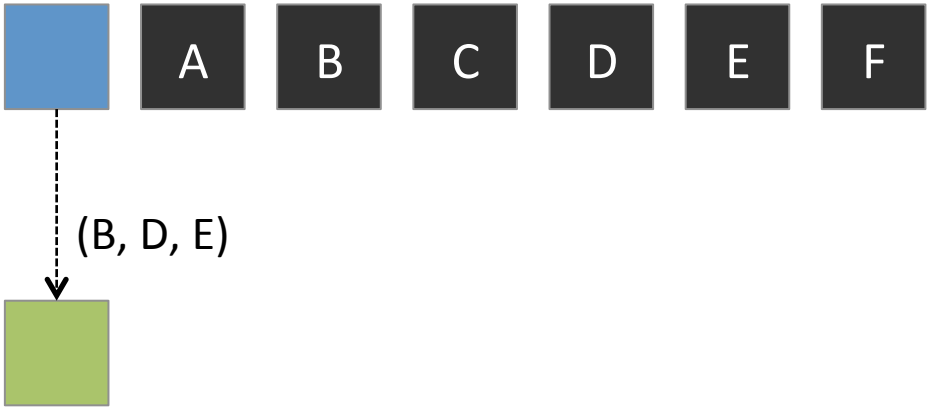
- Distributed applications (ex. Hadoop)
- Attempt to schedule tasks such that negative interactions are kept to a minimum

HDFS Namenode



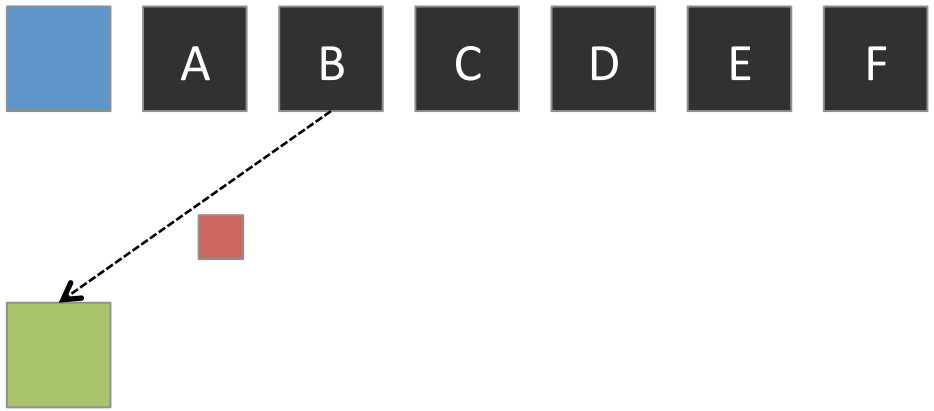
Client requests information about a block

HDFS Namenode



Namenode responds with a list of replicas where that node is available

HDFS Namenode



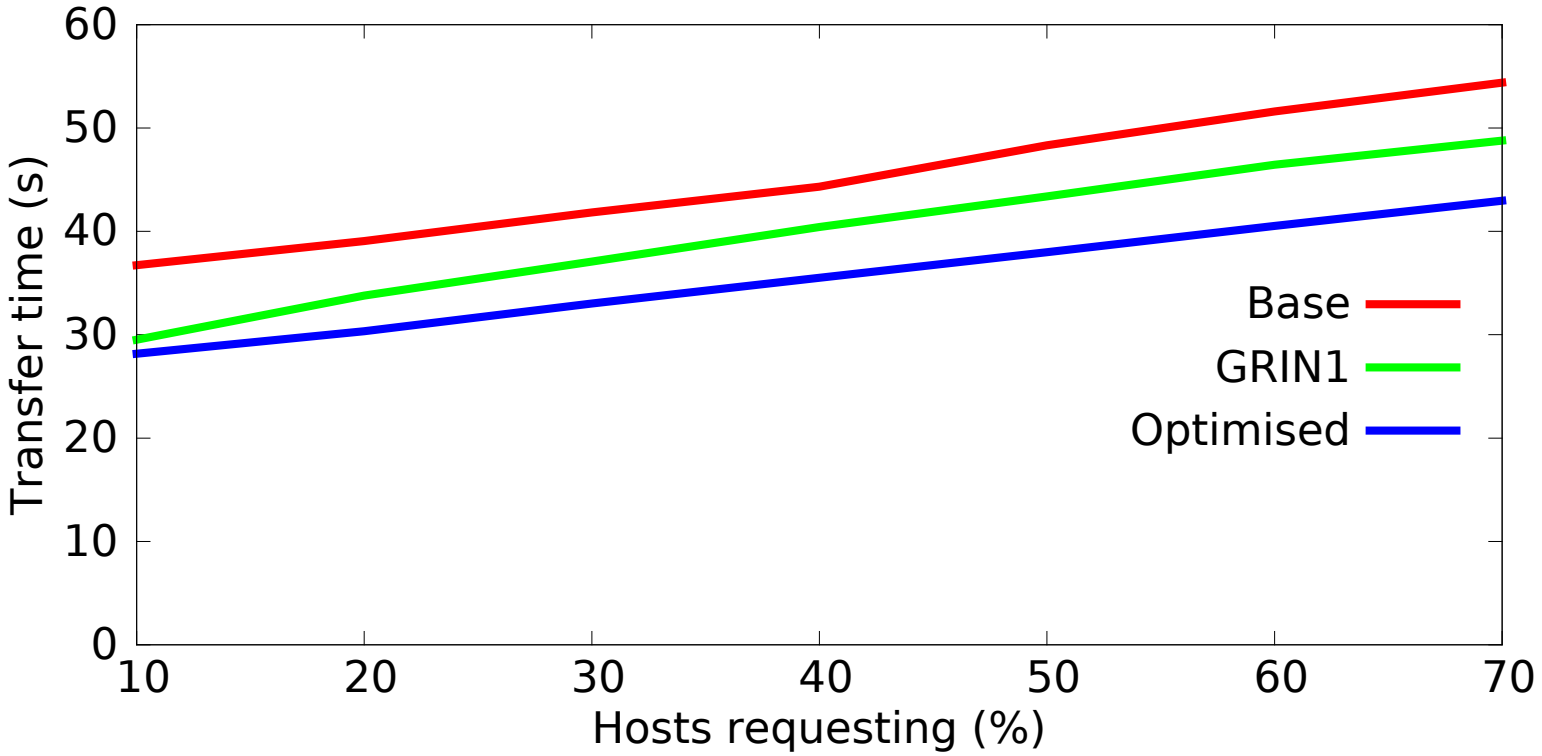
Client attempts to contact a replica, in the order provided by the NameNode

HDFS Namenode Optimisation

- Simple change: before returning a list of replicas, sort it based on expected available uplink capacity

- Heuristic based on last recommendation time

HDFS Namenode Optimisation



Conclusions

- GRIN can bring significant performance benefits for underutilised networks
- Works with almost any existing topology
- Cheap and incrementally deployable
- GRIN aware applications may lessen the negative impact of opportunistic usage (while we hope for better hardware to become available)