Advanced Computer Networks

CS740

Physical Connectivity at the Rack/Cluster Scale

https://pages.cs.wisc.edu/~mgliu/CS740/F25/index.html

Ming Liu
mgliu@cs.wisc.edu

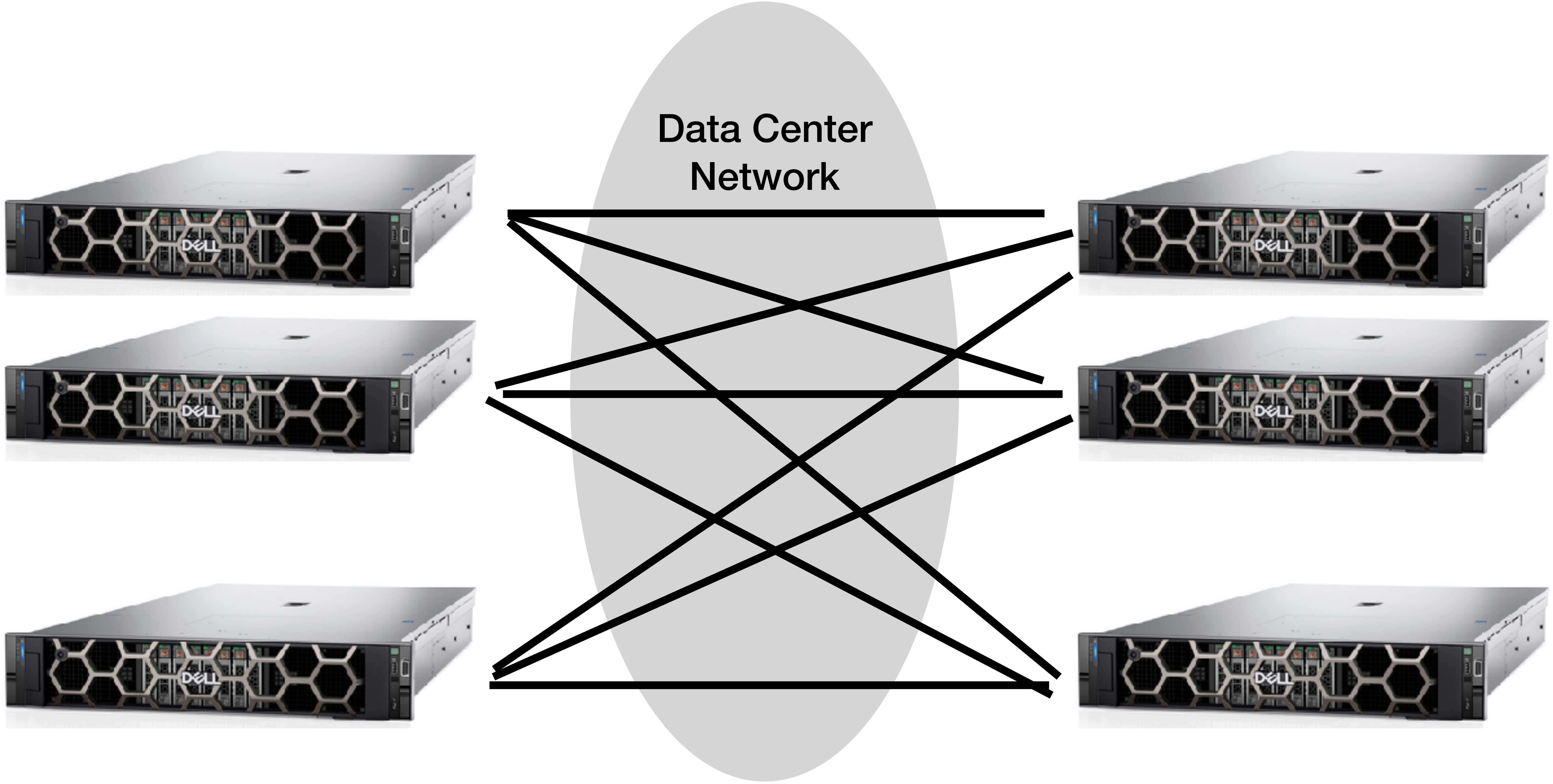
Outline

- Last lecture
  - Course logistics and schedule overview
  - Computer network design history
  - Data center network basics
  - Data center network design requirements
- Today
  - Physical connectivity at the rack/cluster scale
- Announcements
  - Lab1

What are the design requirements for data center networks?

High available server-to-server network connectivity at bandwidth Y among X NIC ports under cost efficiency

High available server-to-server network connectivity at bandwidth Y among X NIC ports under cost efficiency



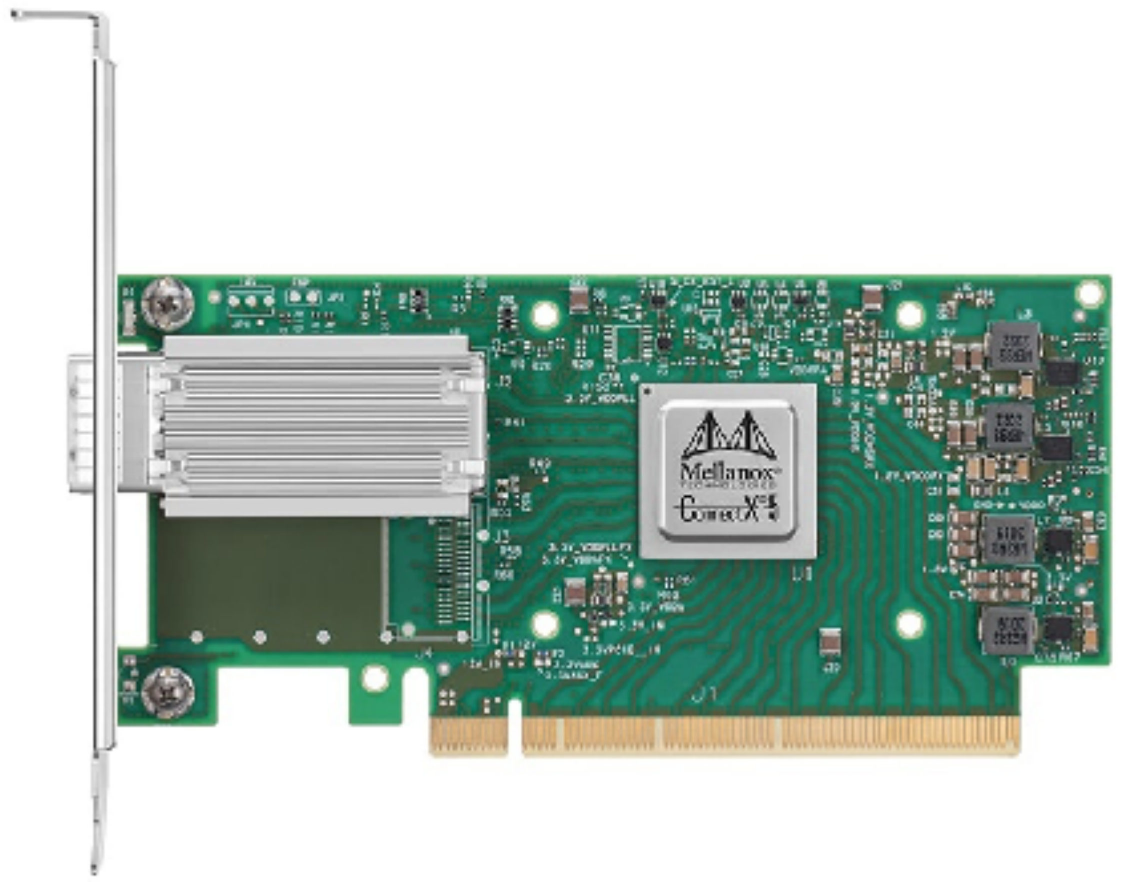
Today’s Focus

High available server-to-server network connectivity at bandwidth Y among X NIC ports under cost efficiency

How can we connect two servers?

Physical Connectivity Between Two Servers

- Network Interface Card (NIC)
  - Port bandwidth (1Gbps, 10Gbps, 25Gbps, …)
  - PCIe lane # and generation



Physical Connectivity Between Two Servers

- Networking Cable
  - Copper (Cat5e, Cat6, Cat6a, Cat7, Cat8) and Fiber (Single/Multi-Mode)
  - Transceiver: a serializer/deserializer(SerDes) converts signals at X GbE
  - Length: reliable data transfer speed, e.g., 1m, 10m, …

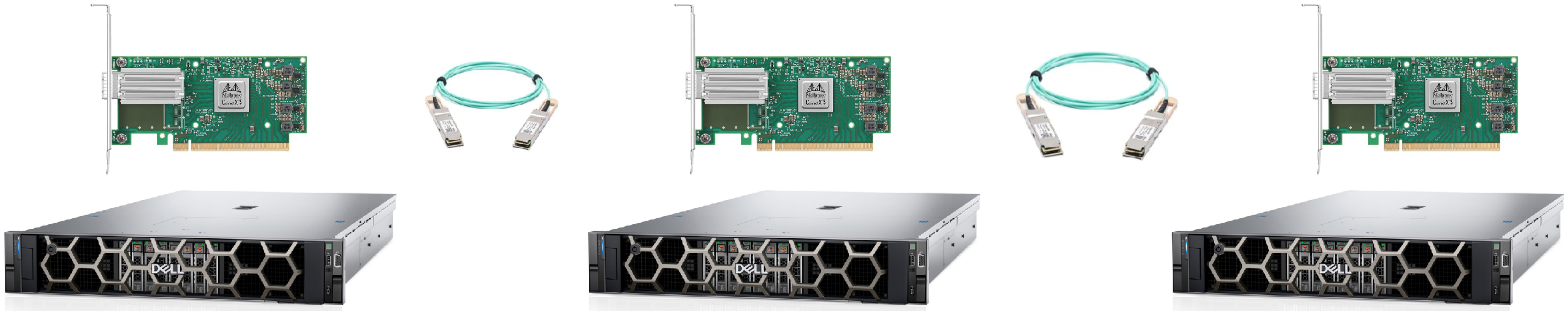
Physical Connectivity Between Two Servers

- We focus on bandwidth in this lecture
  - In practice, server physical location and cost are also important

How can we connect three servers?

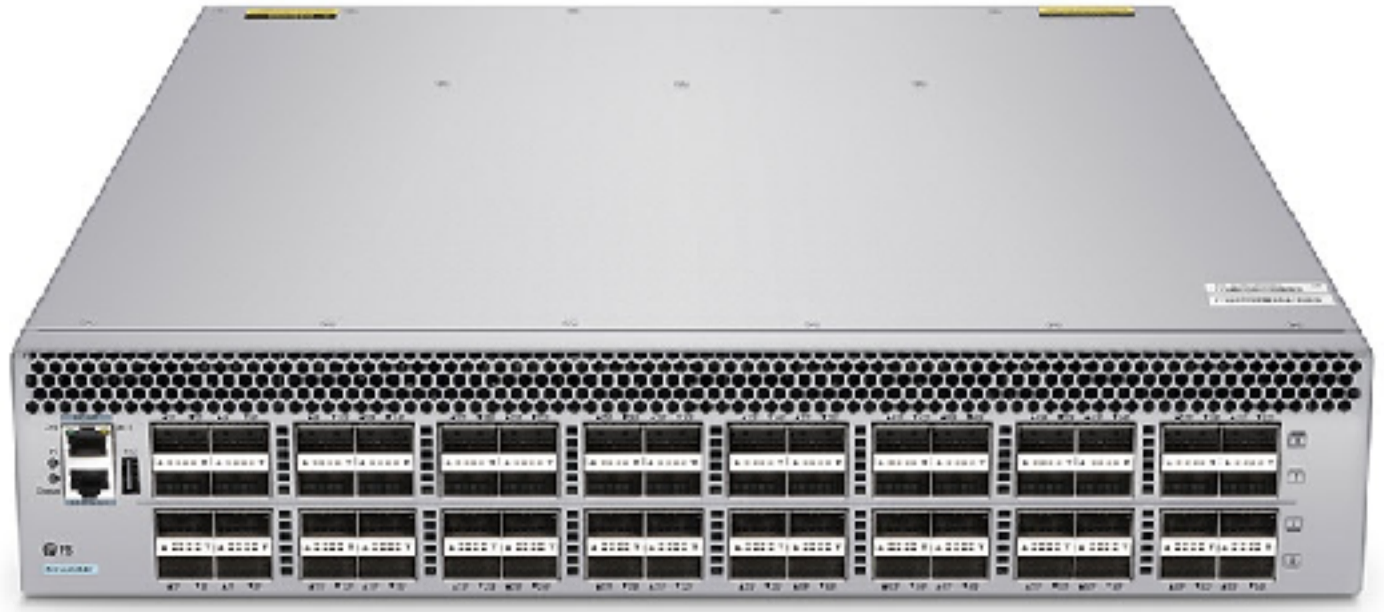


How can we connect three servers?



Physical Connectivity Among Three Servers

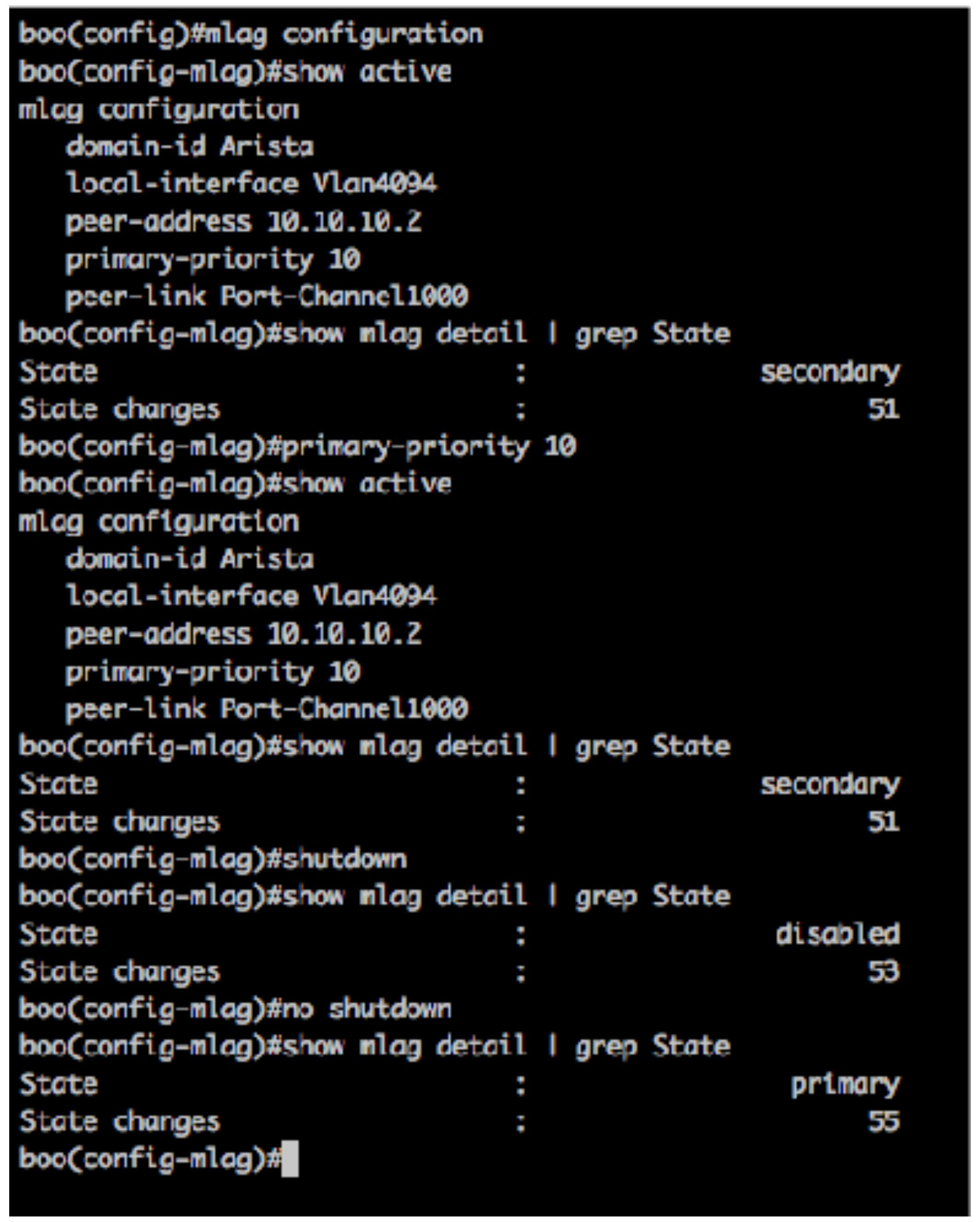
- Networking switch
  - A specialized networking gear providing fan-out connectivity
  - Vendors: Broadcom, Cisco, Dell, Arista, Nvidia, Marvell



Physical Connectivity Among Three Servers

- Networking switch
  - A specialized networking gear providing fan-out connectivity
  - Vendors: Broadcom, Cisco, Dell, Arista, Nvidia, Marvell

- Architectural internals
  - Fixed number of ports (K)
  - Switching ASIC for traffic forwarding
  - General-purpose CPU for running switch
  - L2/L3 switching



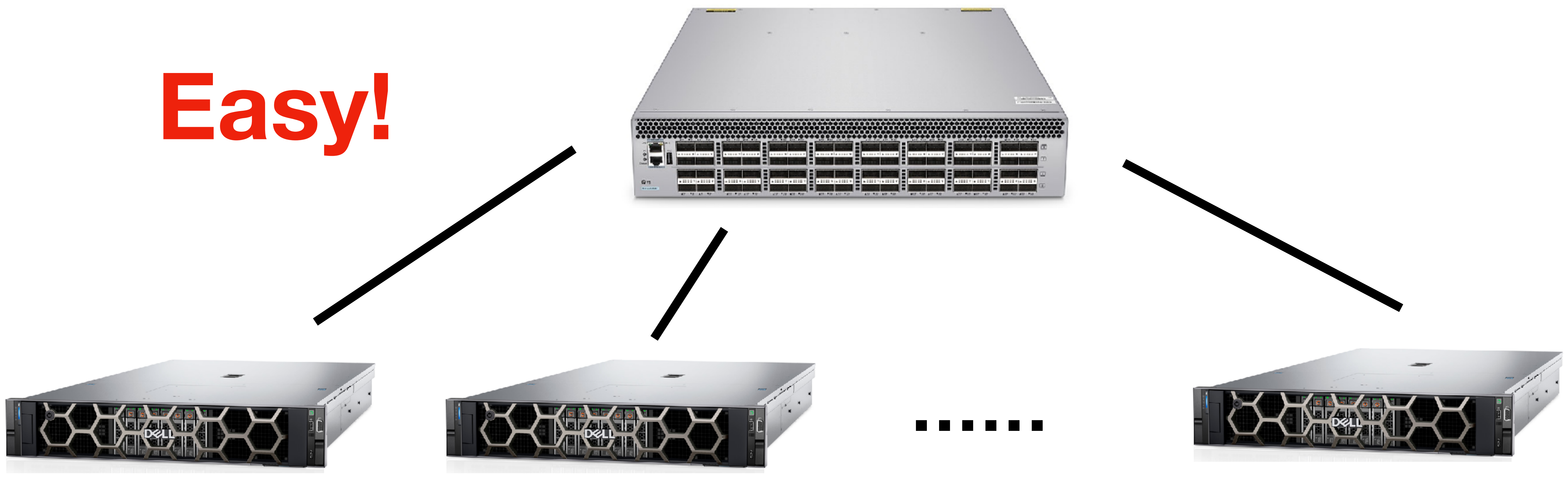
Physical Connectivity Among Three Servers

- Star topology
  - Switch port BW = NIC port BW = Cable BW = Y Gbps

Suppose a switch has K ports, how do we connect K servers?

Suppose a switch has K ports, how do we connect K servers?

Easy!



Rack-Scale Network Connectivity

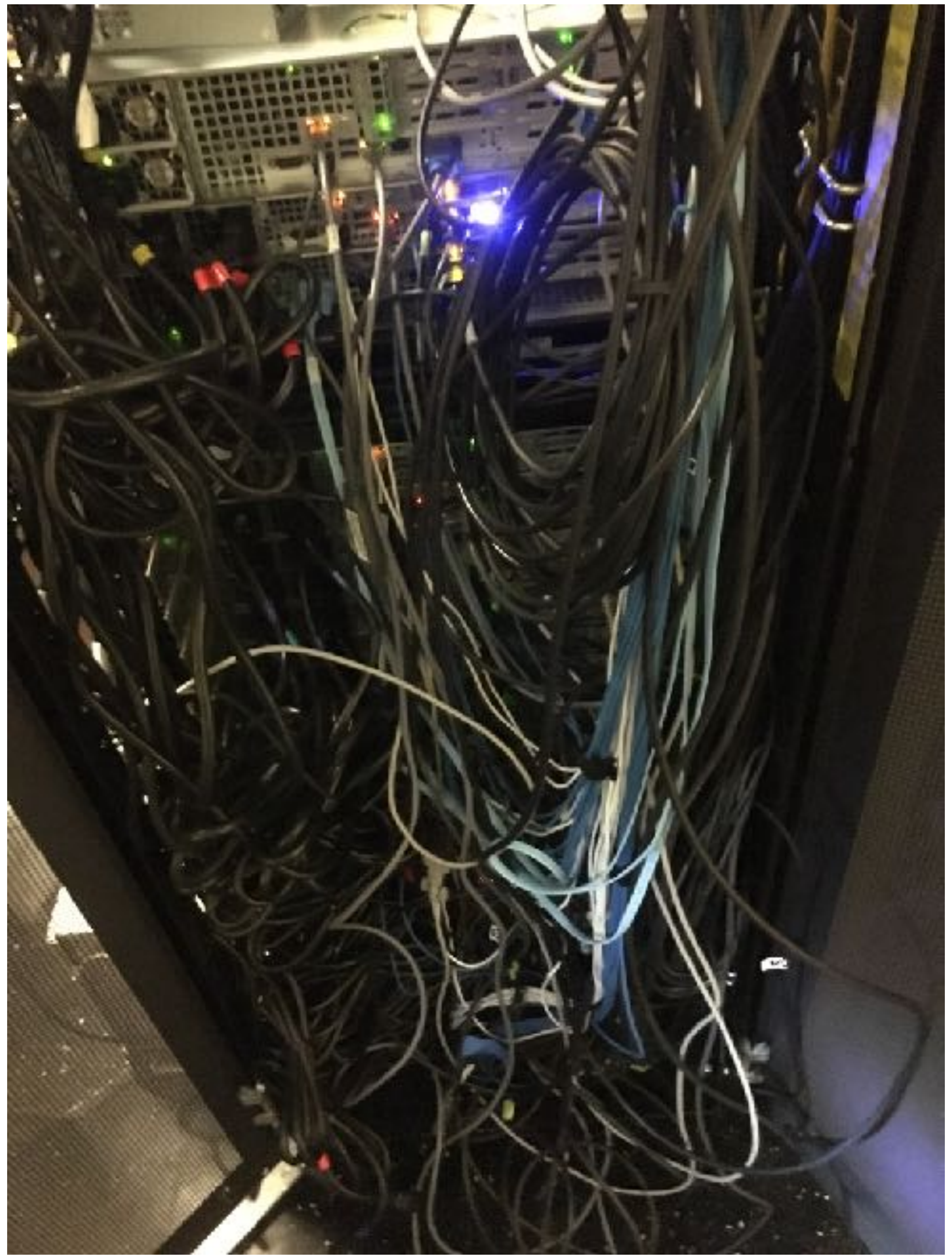
- The size depends on
  - The height of a server rack (42U)
  - The number of switching ports

- Inside a rack
  - PDU (power distribution unit)
  - Servers + Switches
  - Different cables + cable tray

Rack-Scale Network Connectivity

- Th
  - 42U)
  - ts
- In
  -)

Well-organized



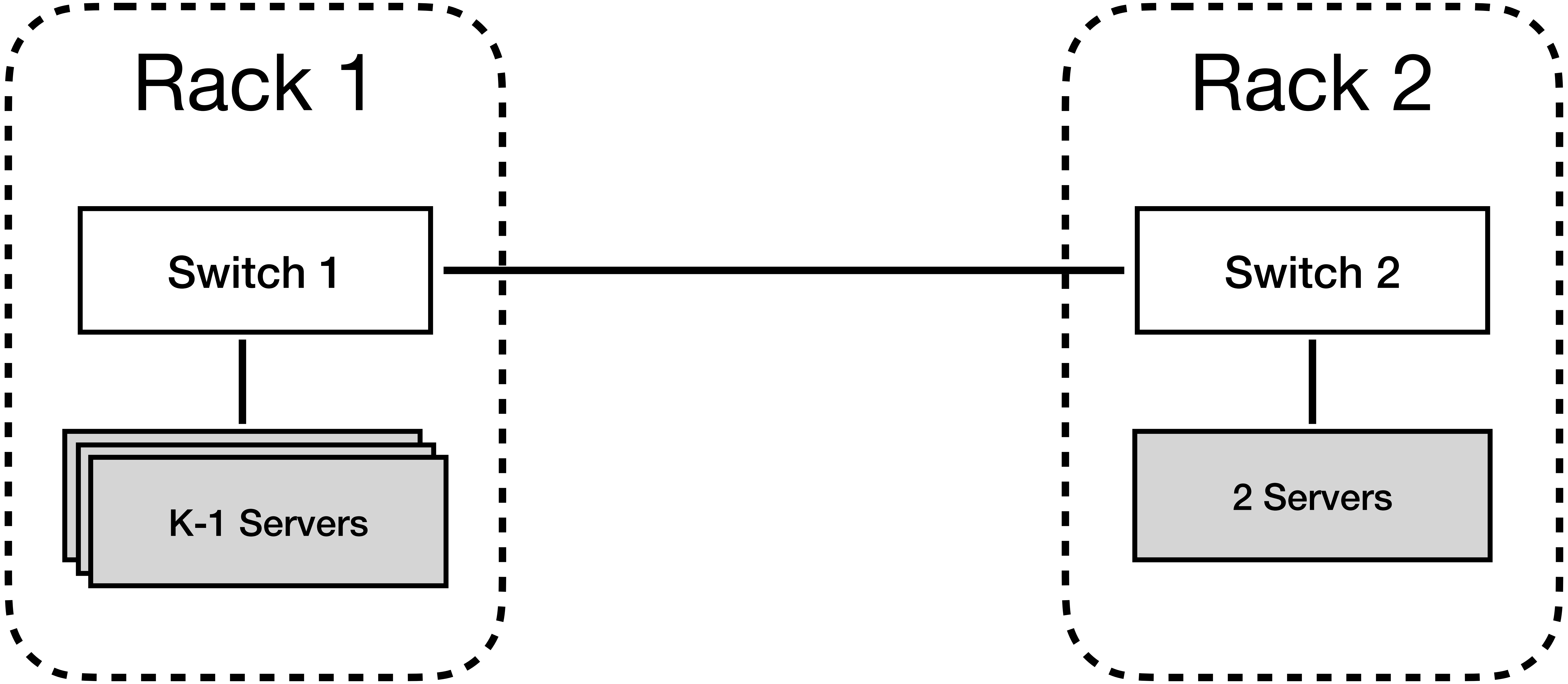
Messy (My first rack experience)

Suppose a switch has ports, how can we connect K+1 servers?

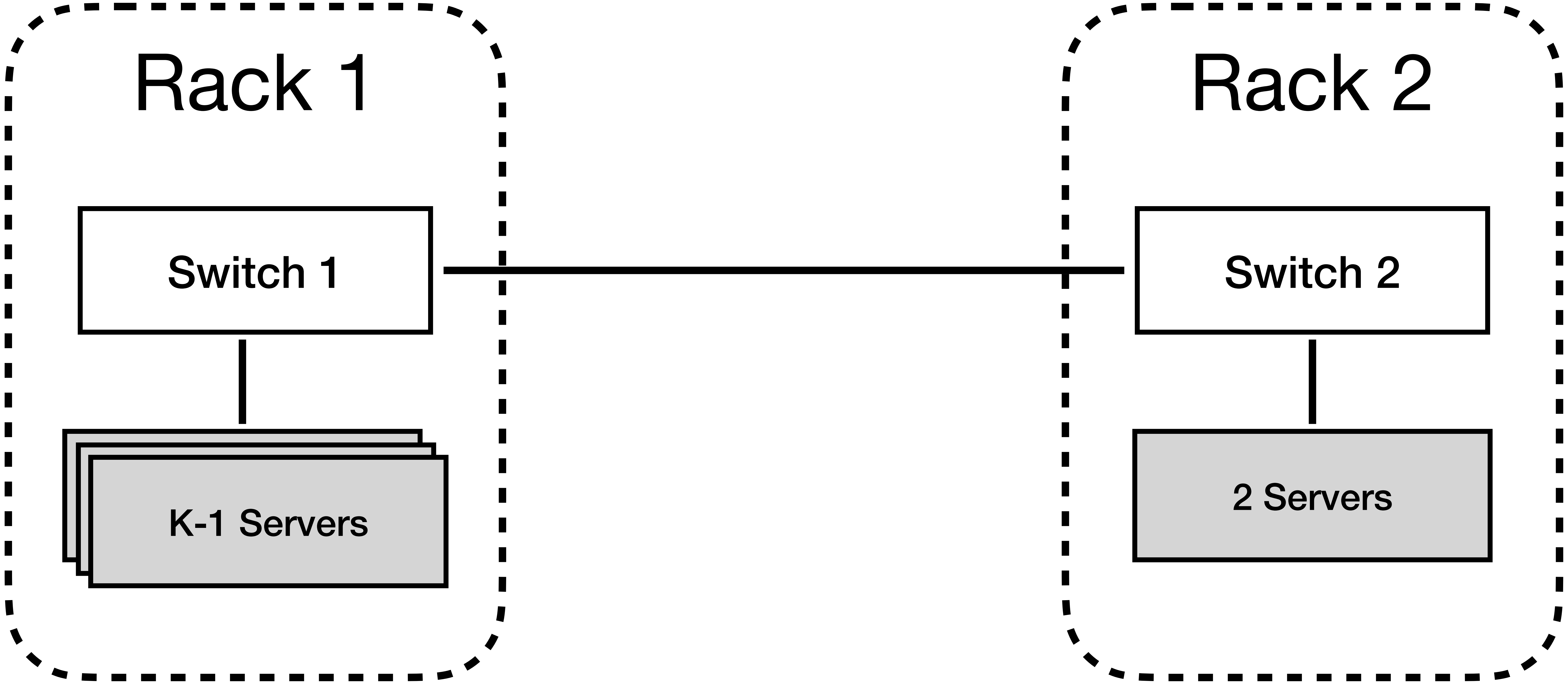
Suppose a switch has ports, how can we connect K+1 servers?

More switches!

Proposal #1



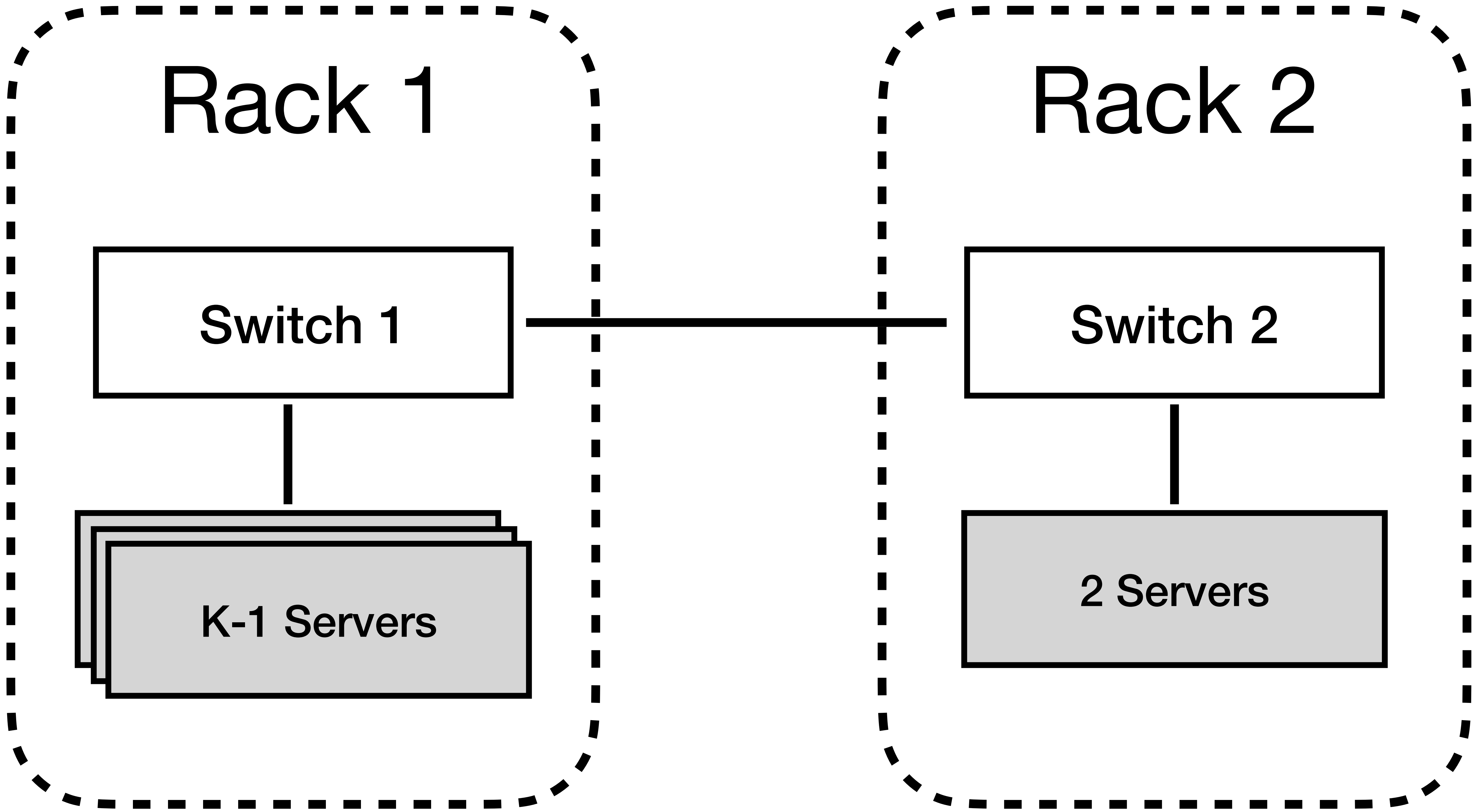
Proposal #1



Does this work?

Issues of Proposal #1

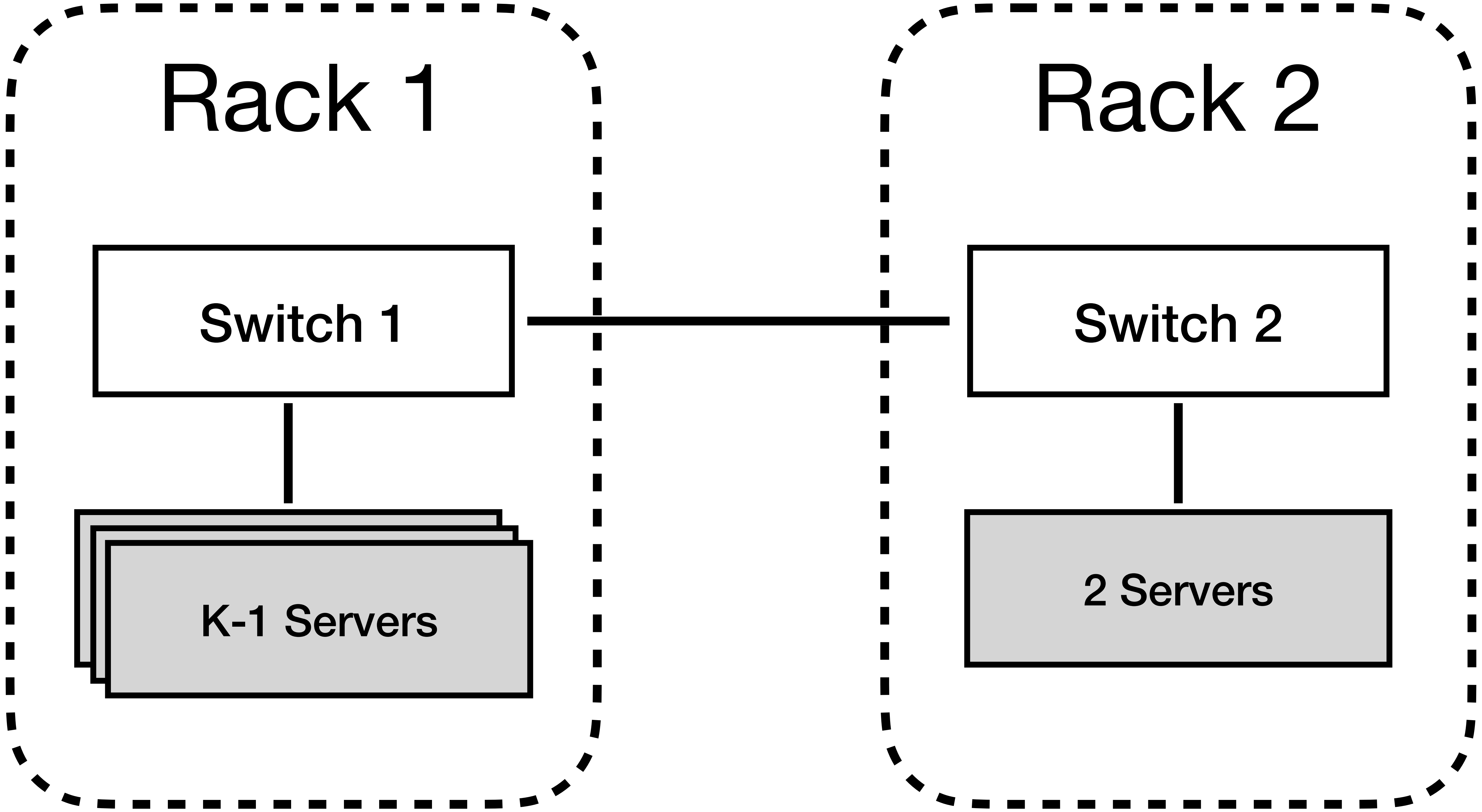
- The ingress and egress bandwidth of a switch are unmatched!
  - Egress: the aggregated bandwidth issued to the outside from a switch
  - Ingress: the aggregated bandwidth coming from the outside to the switch



Issues of Proposal #1

- Switch 1(Rack 1—> Rack 2)
  - Ingress: (K-1) * Y Gbps
  - Egress: 1 * Y Gbps

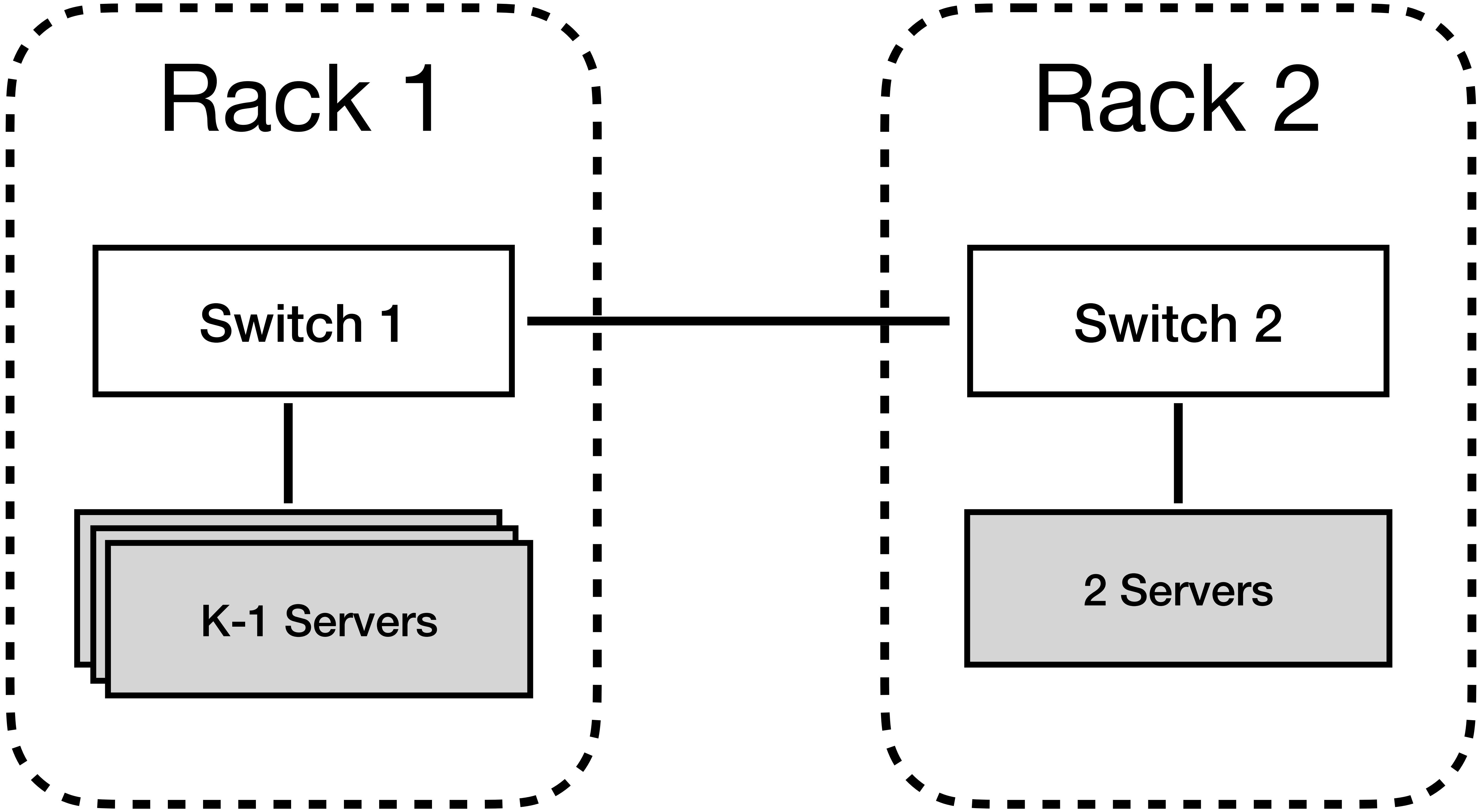
Per-server: 1/(K-1) * Y Gbps



Issues of Proposal #1

- Switch 1(Rack 2—> Rack 1)
  - Ingress: 1 * Y Gbps
  - Egress: (K-1) * Y Gbps

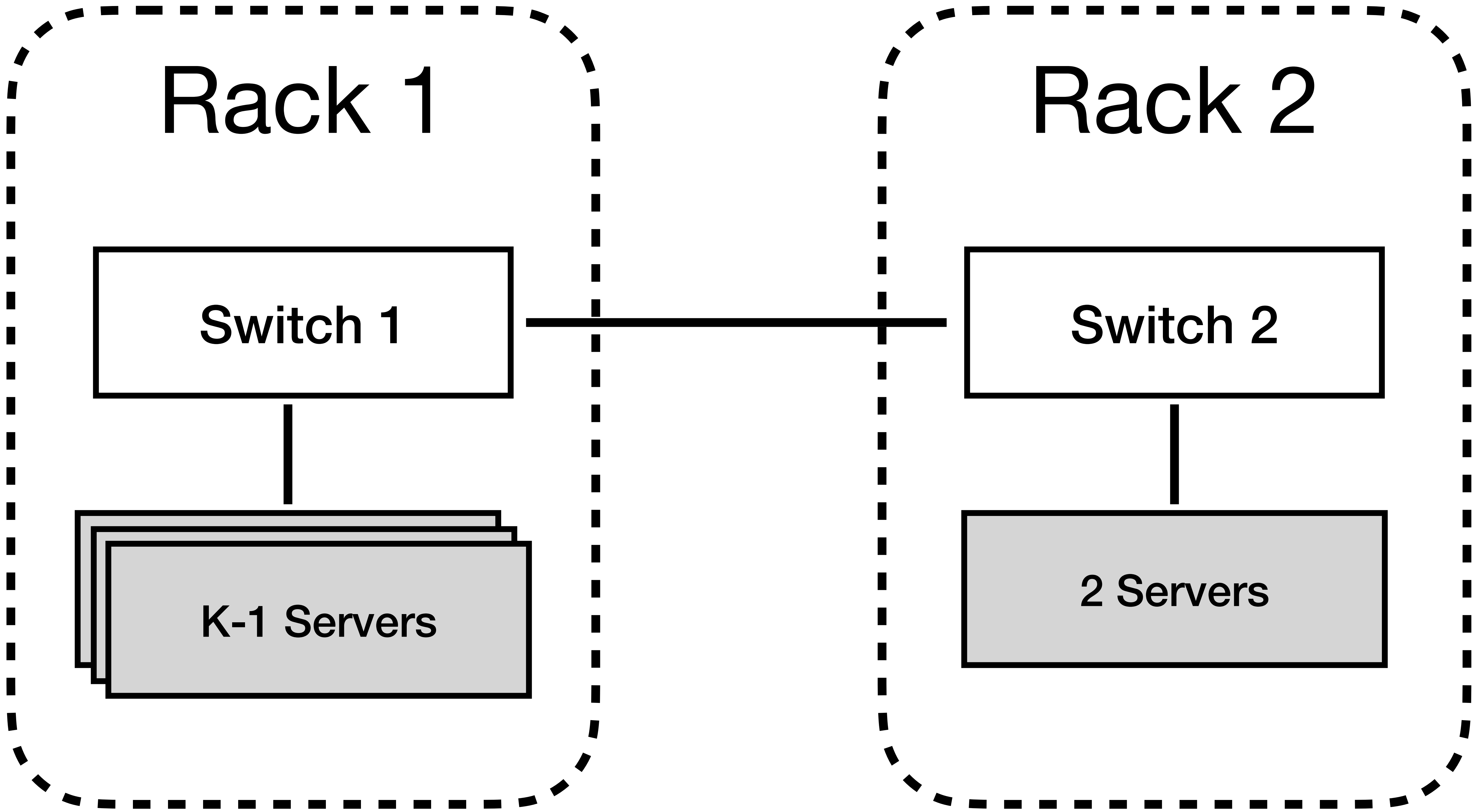
Per-server: 1/(K-1) * Y Gbps



Issues of Proposal #1

- Switch 2(Rack 1 —> Rack 2)
  - Ingress: 1 * Y Gbps
  - Egress: 2 * Y Gbps

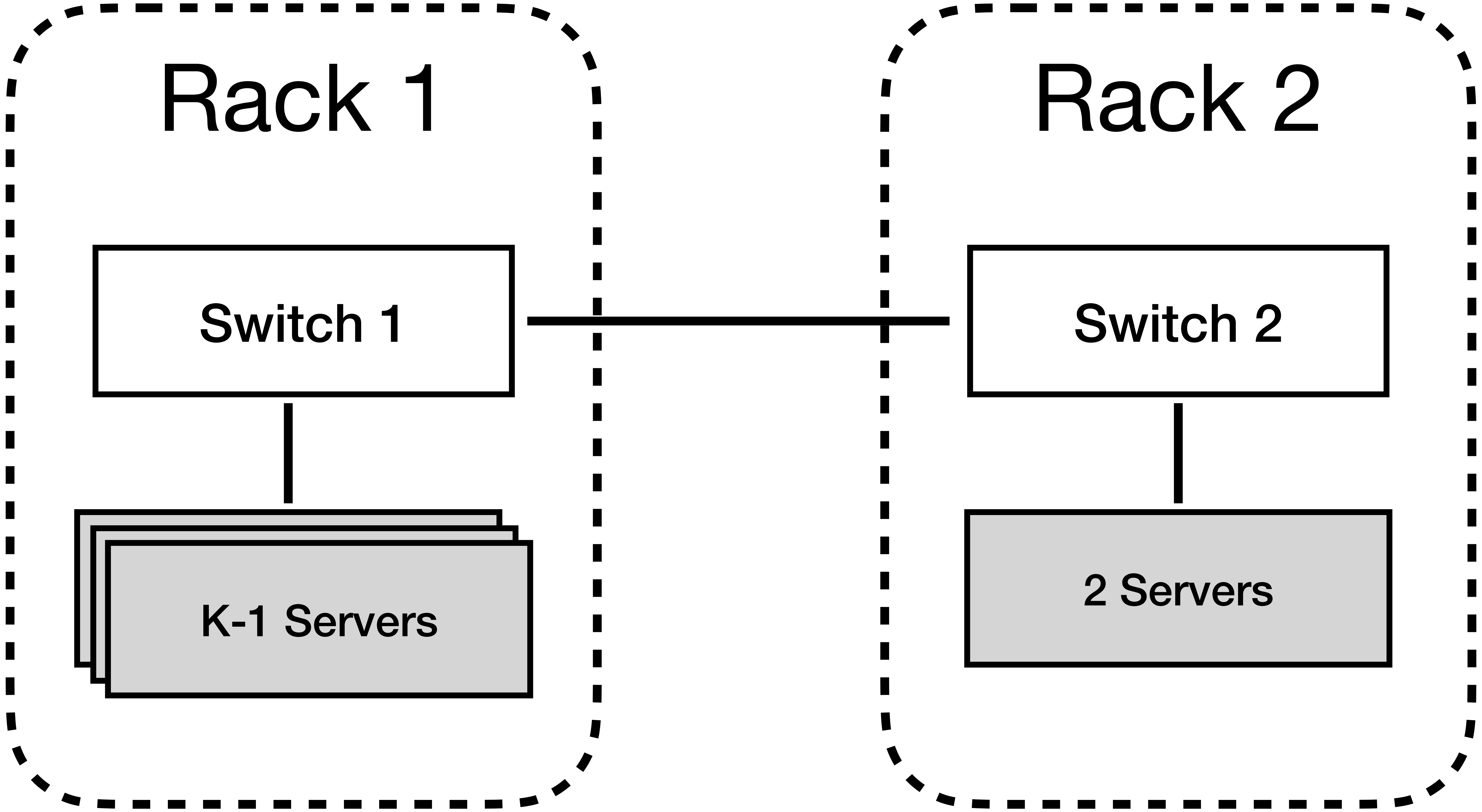
Per-server: 1/2 * Y Gbps



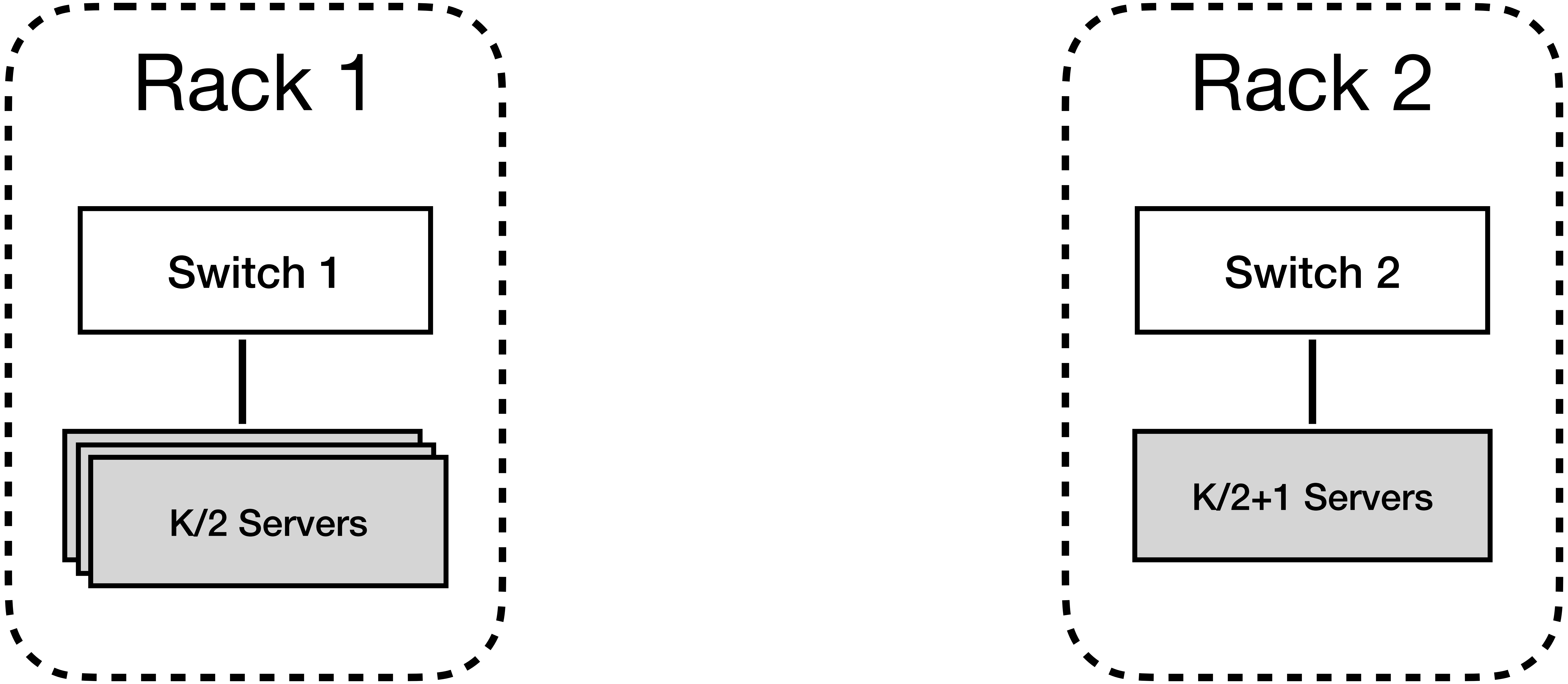
Issues of Proposal #1

- Switch 2(Rack 2 —> Rack 1)
  - Ingress: 2 * Y Gbps
  - Egress: 1 * Y Gbps

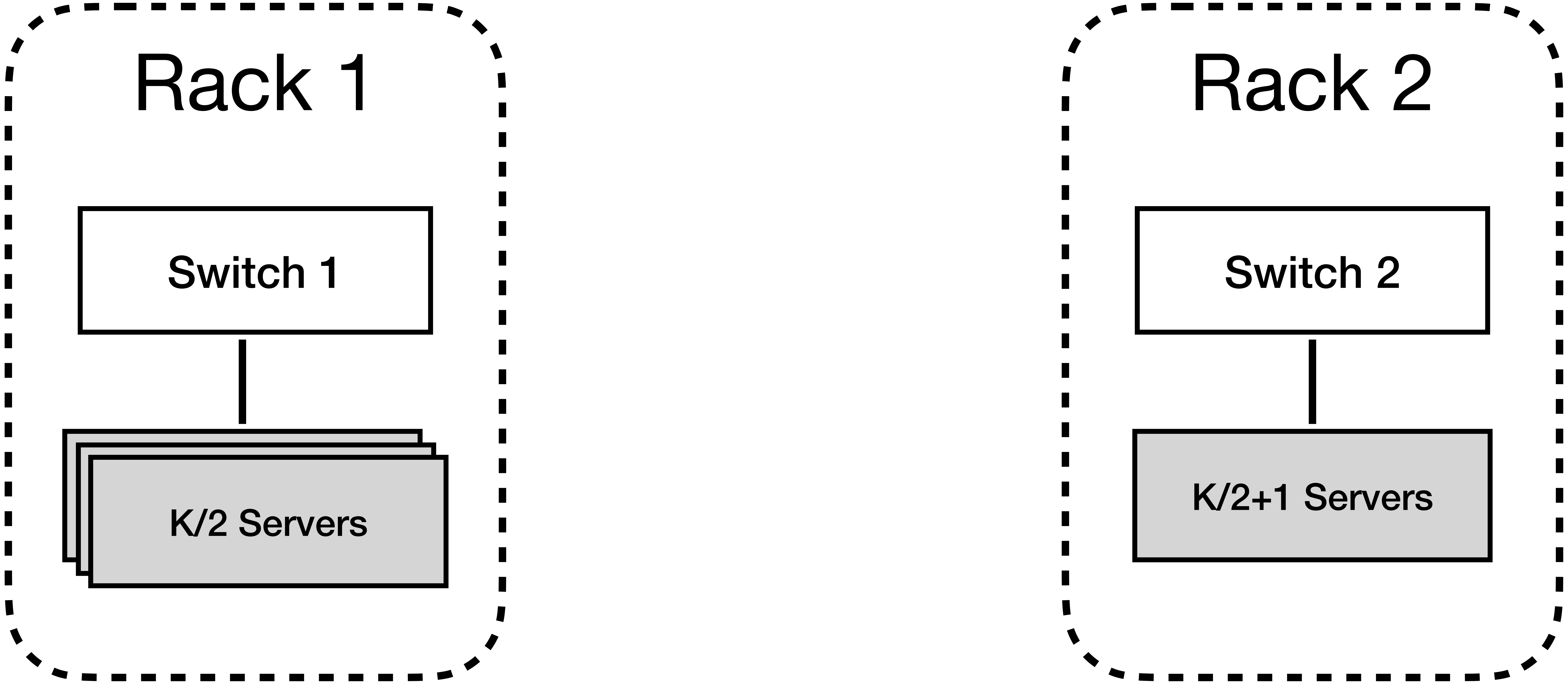
Per-server: 1/2 * Y Gbps



Proposal #2

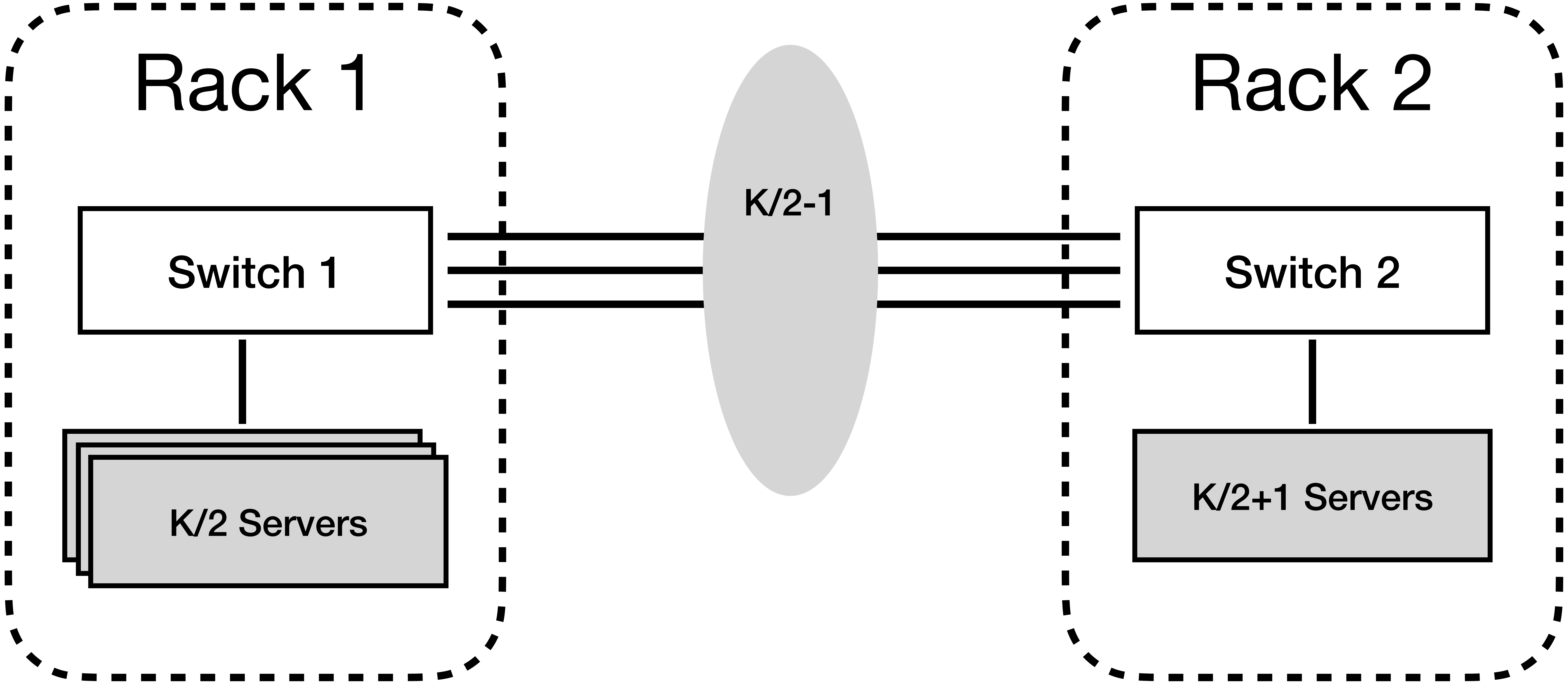


Proposal #2



How to connect switch 1 and switch 2?

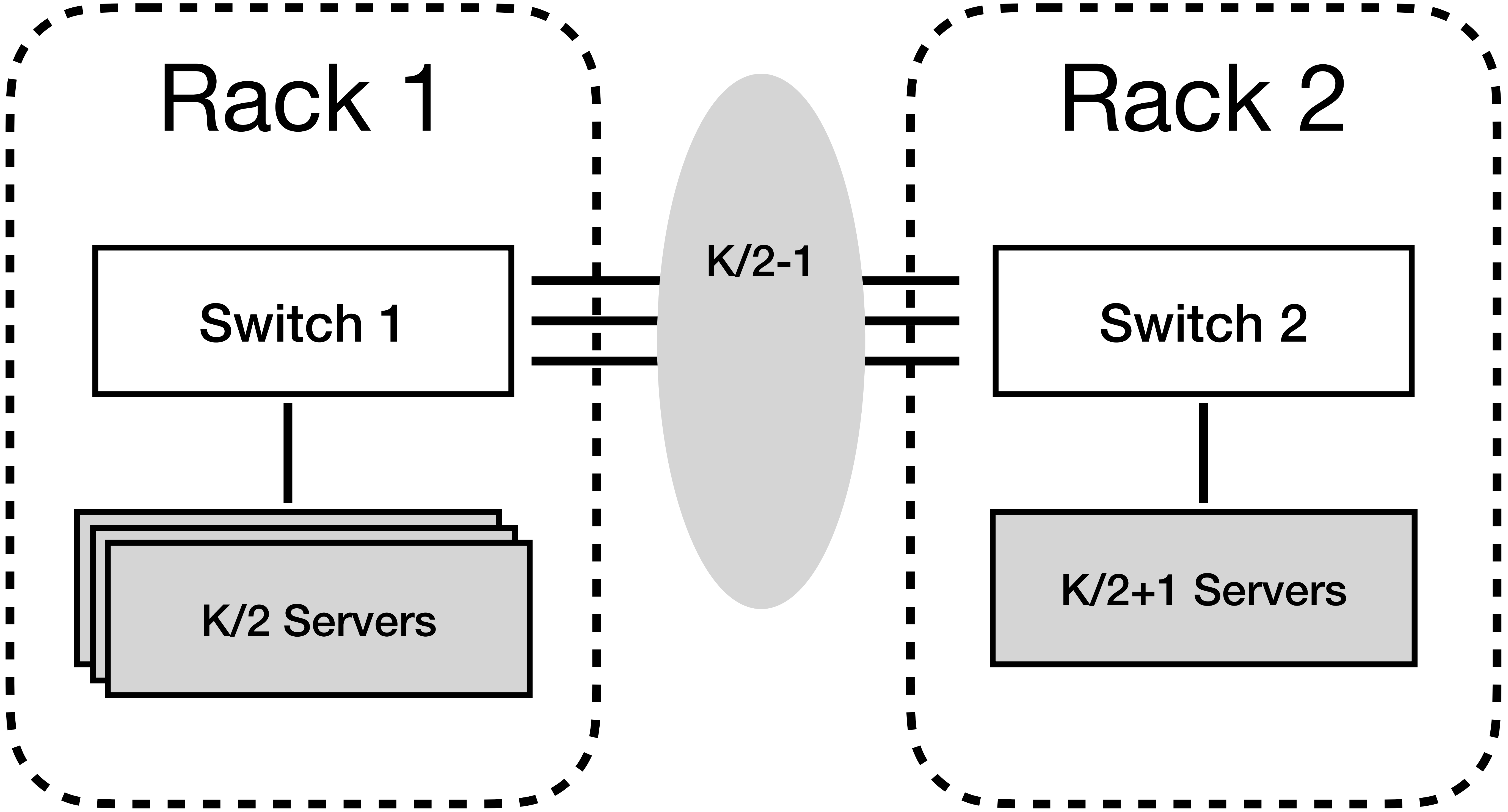
Proposal #2



Proposal #2 is better than proposal #1

- Switch 1(Rack 1—> Rack 2)
  - Ingress: K/2 * Y Gbps
  - Egress: (K/2 - 1) * Y Gbps

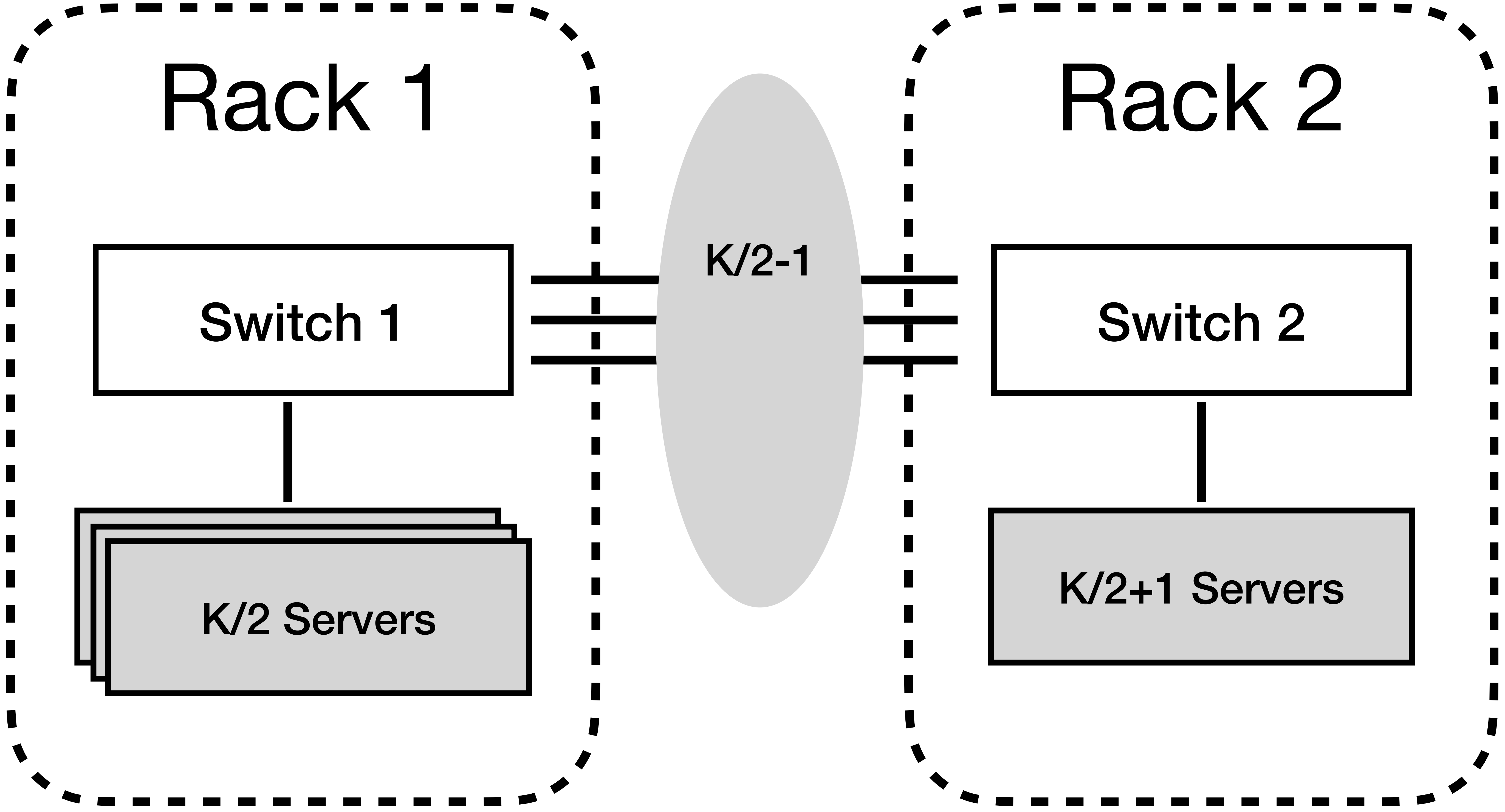
Per-server: (K-2)/K * Y Gbps



Proposal #2 is better than proposal #1

- Switch 1(Rack 2—> Rack 1)
  - Ingress: (K/2-1) * Y Gbps
  - Egress: K/2 * Y Gbps

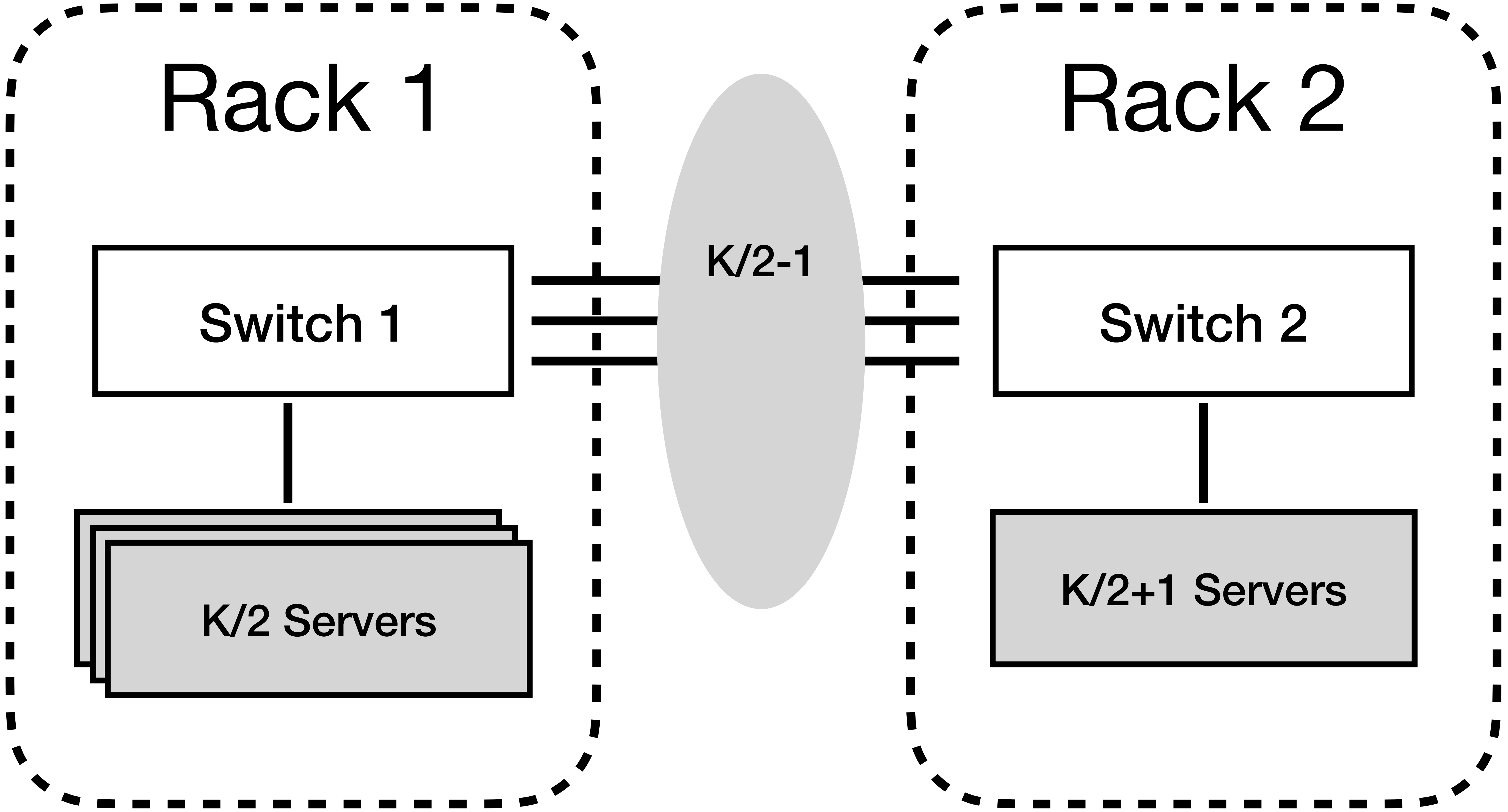
Per-server: (K-2)/K * Y Gbps



Proposal #2 is better than proposal #1

- Switch 2(Rack 1—> Rack 2)
  - Ingress: (K/2 - 1) * Y Gbps
  - Egress: (K/2 + 1) * Y Gbps

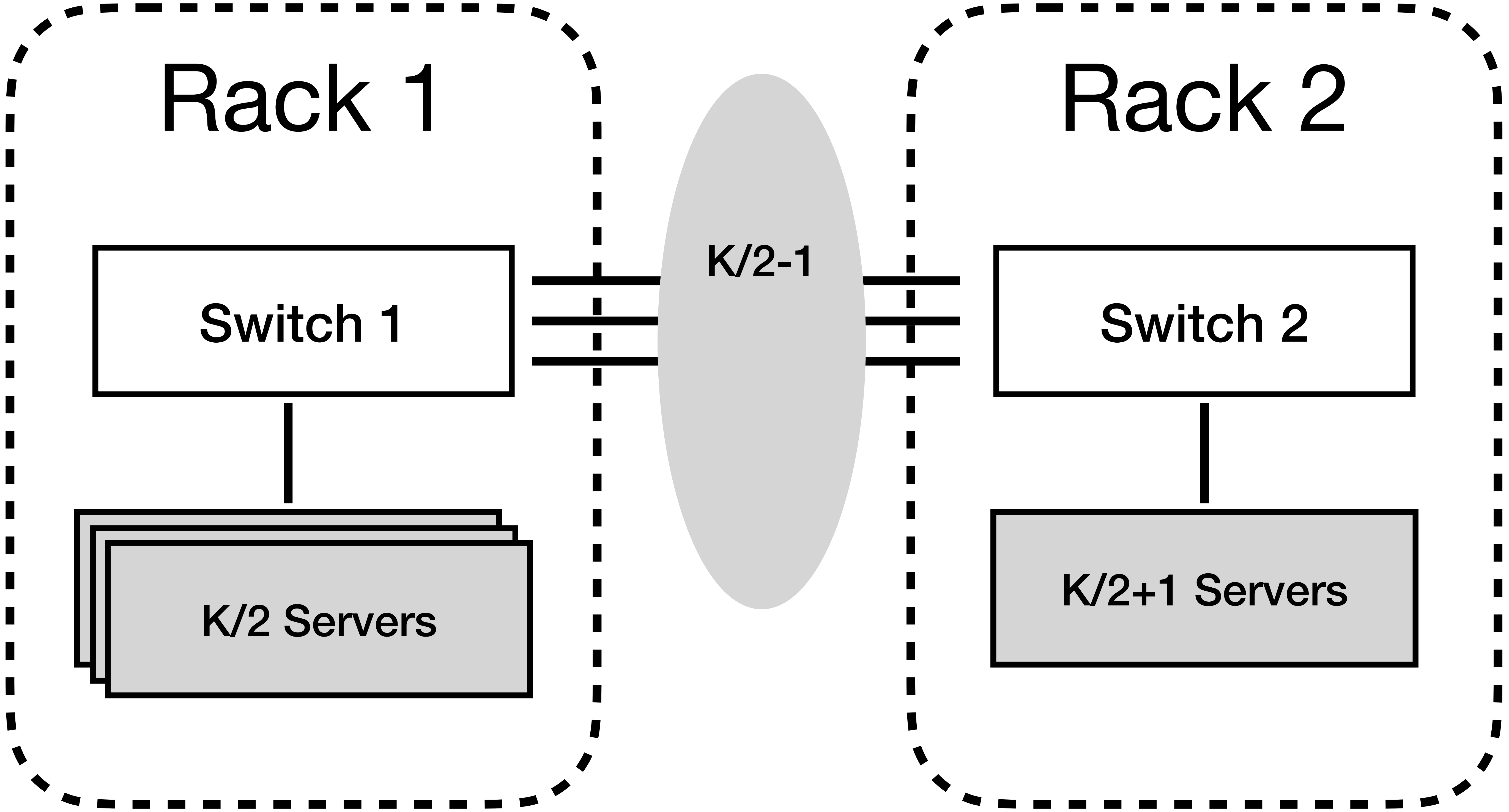
Per-server: (K-2)/(K+2) * Y Gbps



Proposal #2 is better than proposal #1

- Switch 2(Rack 2—> Rack 1)
  - Ingress: (K/2 + 1) * Y Gbps
  - Egress: (K/2 - 1) * Y Gbps

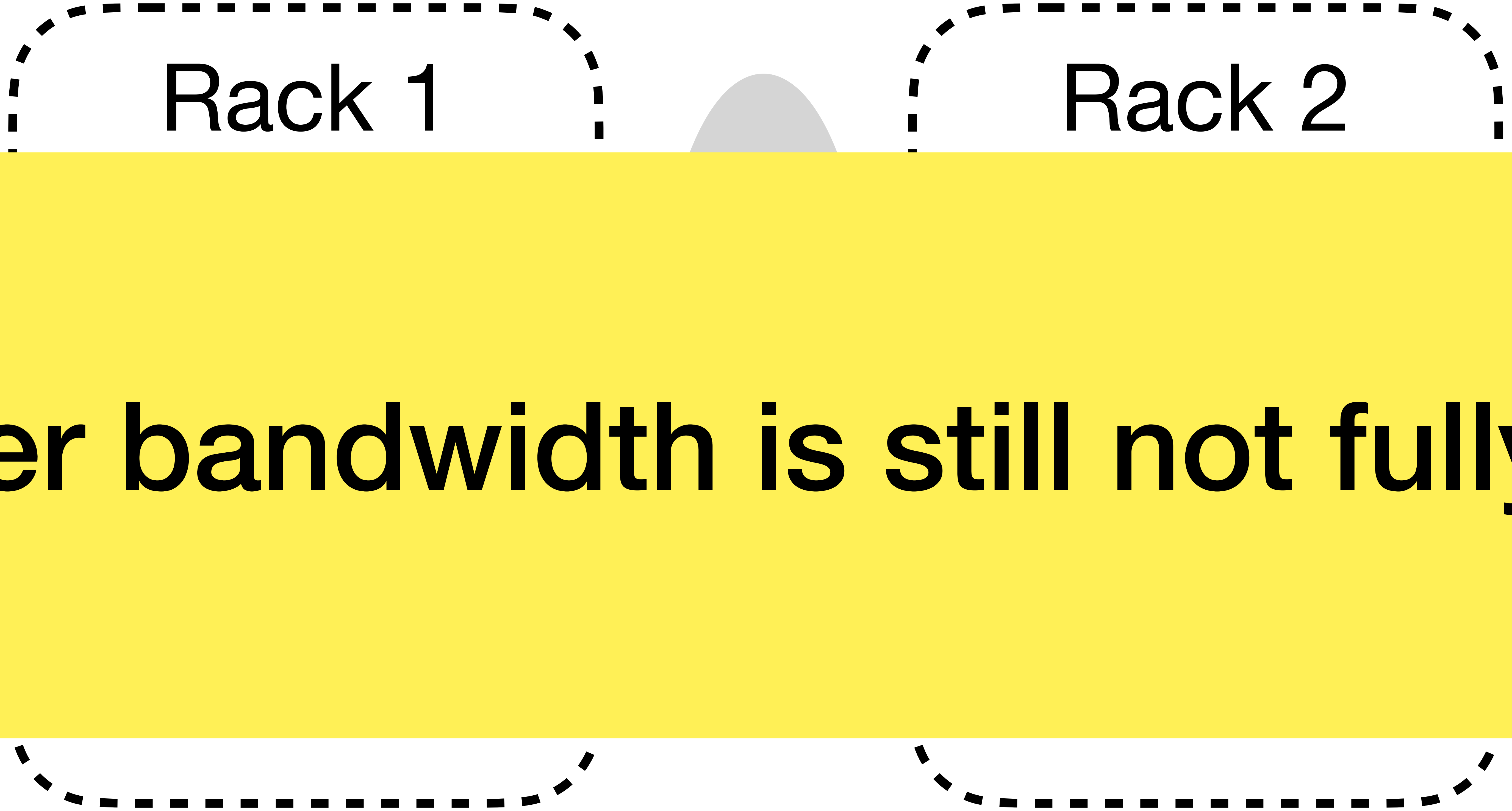
Per-server: (K-2)/(K+2) * Y Gbps



Proposal #2 is better than proposal #1

- Switch 2(Rack 2—> Rack 1)
  - Ingress: (K/2 + 1) * Y Gbps
  - Egress: (K/2 - 1) * Y Gbps

Per-server: (K-2)/(K+2) * Y Gbps



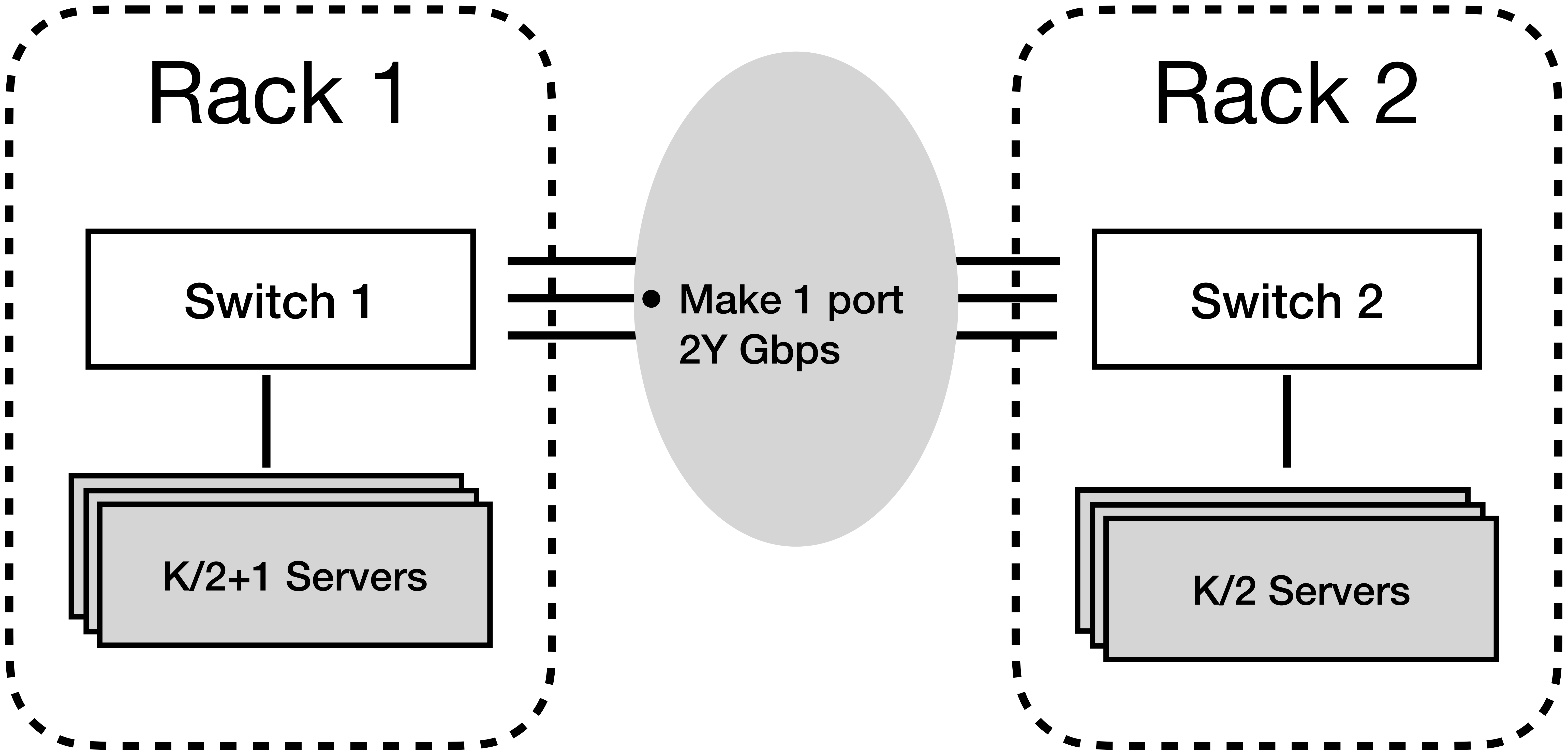
But server bandwidth is still not fully used!

Key: Match ingress and egress bandwidth at each switching point!

What we can do?

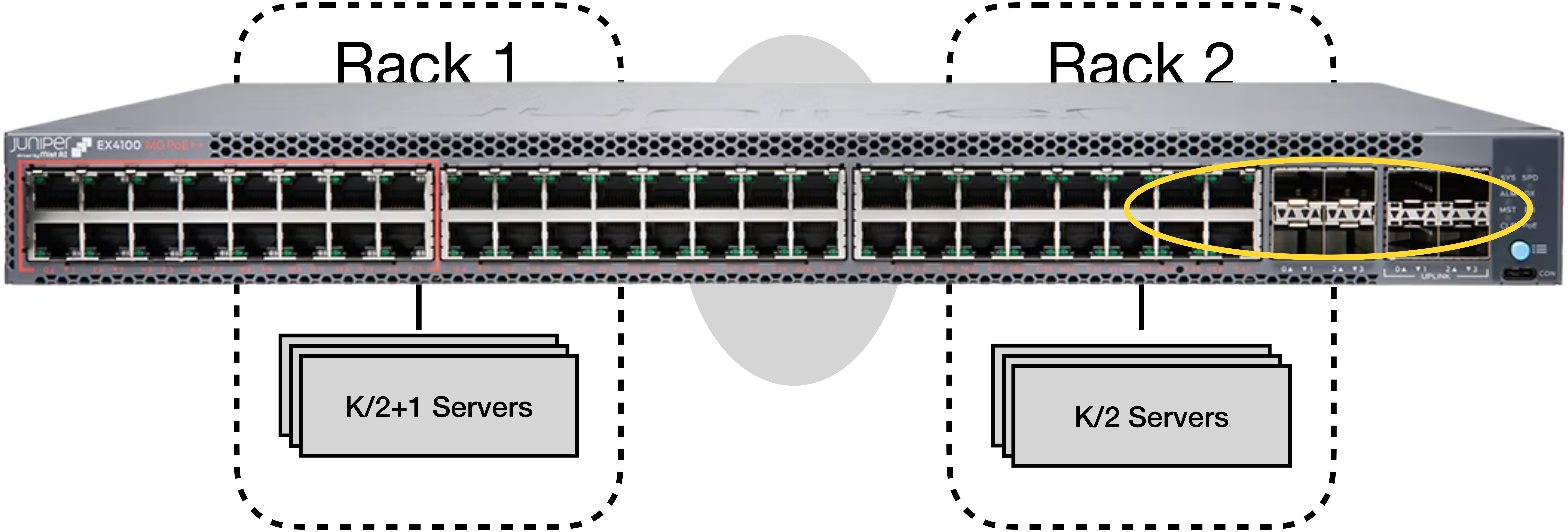
What we can do?

- #1: scale-out strategy
  - Enhance the switch
  - Slim (slow) port + Fat (fast) port



What we can do?

- #1: scale-out strategy
  - Enhance the switch
  - Slim (slow) port + Fat (fast) port



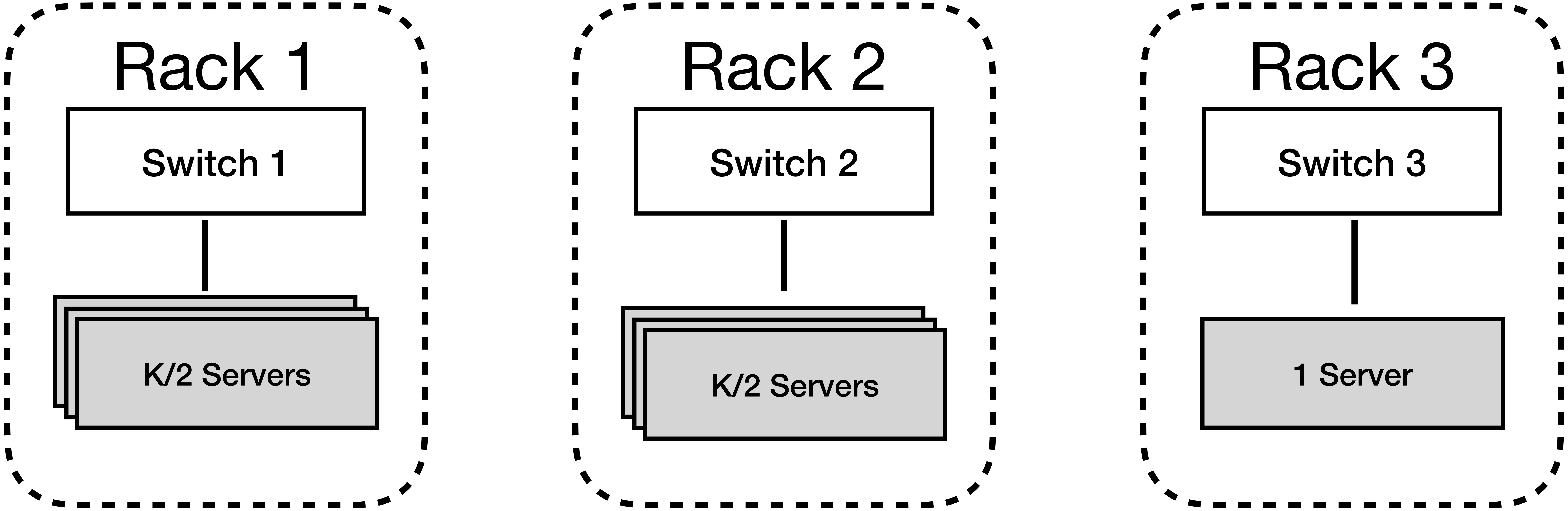
What we can do?

- #1: scale-out strategy
  - Enhance the switch
  - Slim (slow) port + Fat (fast) port

- A heavy solution, requiring all networking gear support
- Hardware-dependent, not generally applicable

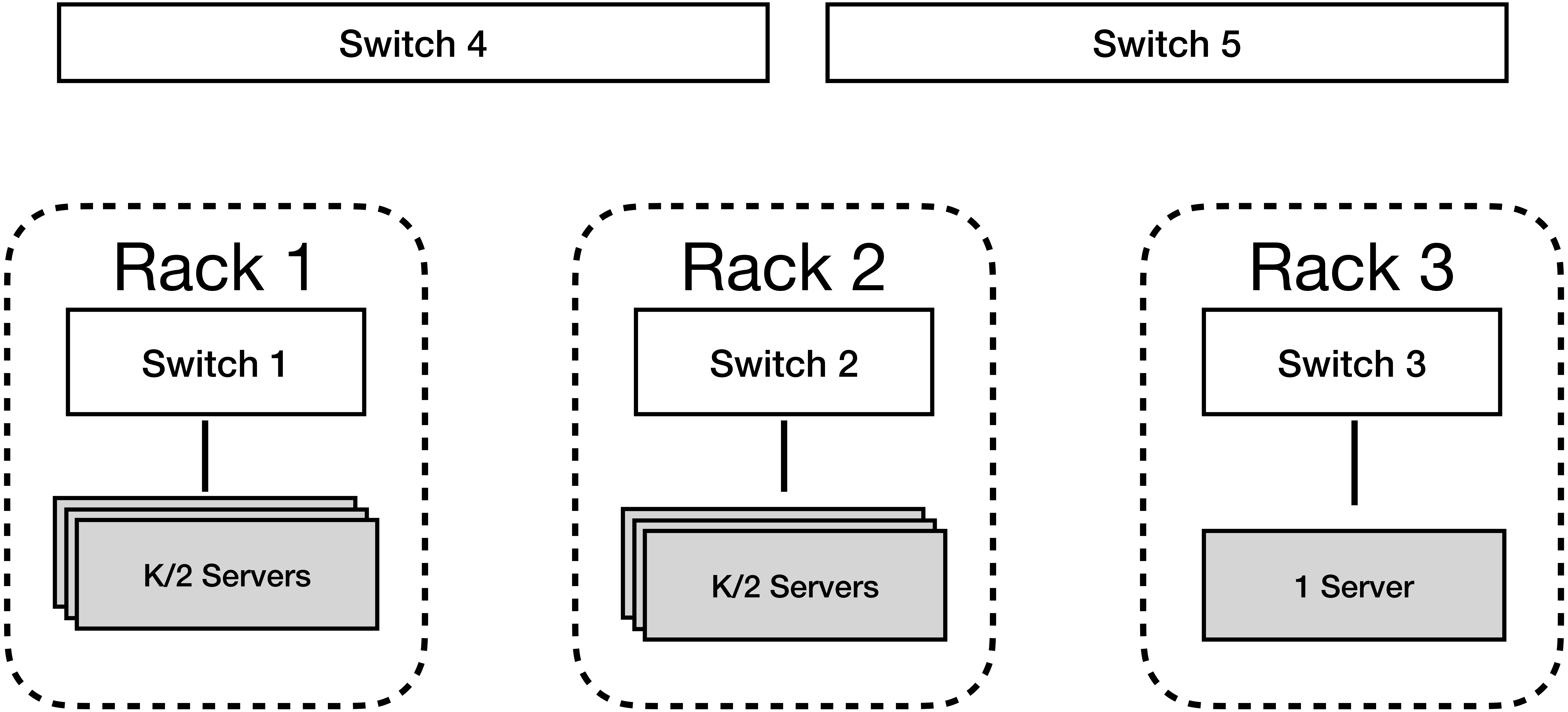
What we can do?

- #2: scale-up strategy
  - Adding more intermediate stages



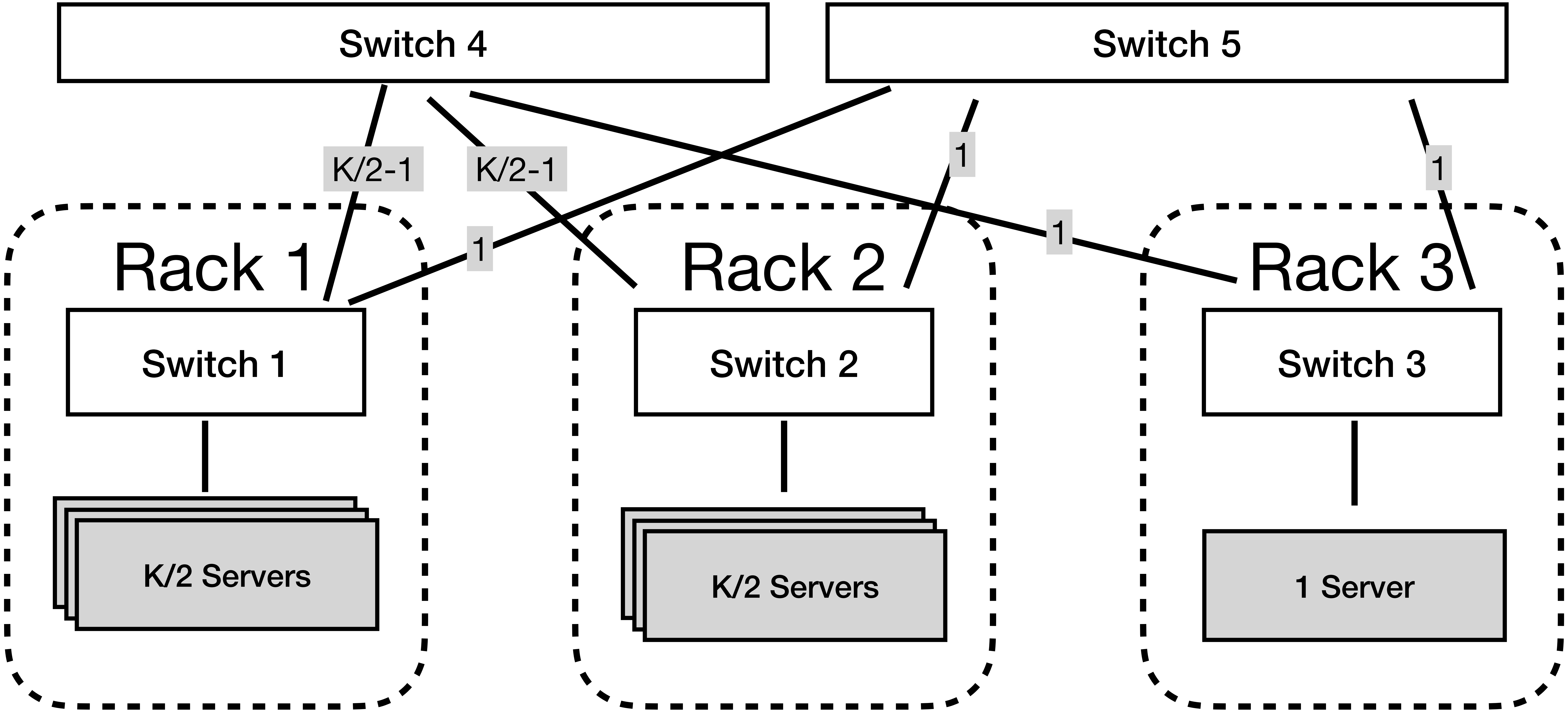
What we can do?

- #2: scale-up strategy
  - Adding more intermediate stages



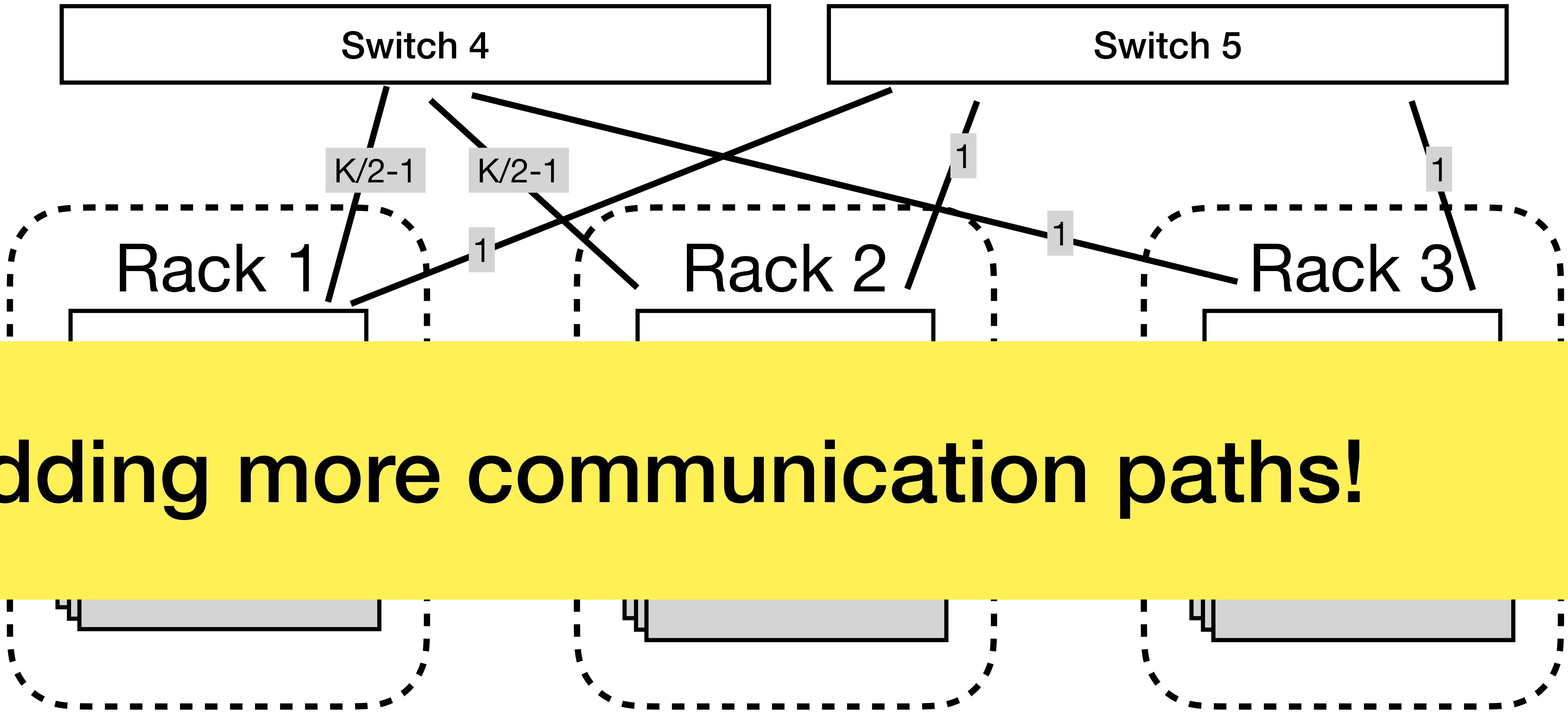
What we can do?

- #2: scale-up strategy
  - Adding more intermediate stages



What we can do?

- #2: scale-up strategy
  - Adding more intermediate stages



Adding more communication paths!

How can we connect X servers?

A Multistage Switching Network

- Clos networks, originally proposed in the telecommunications
  - Invented by Edson Erwin in 1938 and formalized by Charles Clos in 1952
- Fat-Tree topology
  - First proposed for parallel supercomputers

Fat-Trees: Universal Networks for Hardware-Efficient Supercomputing

CHARLES E. LEISERSON, MEMBER, IEEE

Abstract — This paper presents a new class of universal routing networks called *fat-trees*, which might be used to interconnect the processors of a general-purpose parallel supercomputer. A fat-tree routing network is parameterized not only in the number of processors, but also in the amount of simultaneous communication it can support. Since communication can be scaled independently from number of processors, substantial hardware can be saved over, for example, hypercube-based networks, for such parallel processing applications as finite-element analysis, but without resorting to a special-purpose architecture.

Of greater interest from a theoretical standpoint, however, is a proof that a fat-tree of a given size is nearly the best routing network of that size. This *universality theorem* is proved using a three-dimensional VLSI model that incorporates wiring as a direct cost. In this model, hardware size is measured as physical volume. We prove that for any given amount of communications hardware, a fat-tree built from that amount of hardware can simulate every other network built from the same amount of hardware, using only slightly more time (a polylogarithmic factor greater). The basic assumption we make of competing networks is the following. In unit time, at most $O(a)$ bits can enter or leave a closed three-dimensional region with surface area a. (This paper proves the universality result for *off-line* simulations only.)

Index Terms — Fat-trees, interconnection networks, parallel supercomputing, routing networks, universality, VLSI theory.

I. Introduction

MOST routing networks for parallel processing supercomputers have been analyzed in terms of performance and cost. Performance is typically measured by how long it takes to route permutations, and cost is measured by the number of switching components and wires. This paper presents a new routing network called fat-trees, but analyzes it in a somewhat different model. Specifically, we use a three-dimensional VLSI model in which *pin boundedness* has a direct analog as the bandwidth limitation imposed by the surface of a closed three-dimensional region. Performance is measured by how long it takes to route an arbitrary set of messages, and cost is measured as the volume of a physical implementation of the network. We prove a *universality* theorem which shows that for a given volume of hardware, no network is much better.

Unlike a computer scientist's traditional notion of a tree, fat-trees are more like real trees in that they get thicker further from the leaves. In physical structure, a fat-tree resembles, and is based on, the *tree of meshes* graph due to Leighton [12], [14]. The processors of a fat-tree are located at the leaves of a complete binary tree, and the internal nodes are switches. Going up the fat-tree, the number of wires connecting a node with its father increases, and hence the communication bandwidth increases. The rate of growth influences the size and cost of the hardware as well.

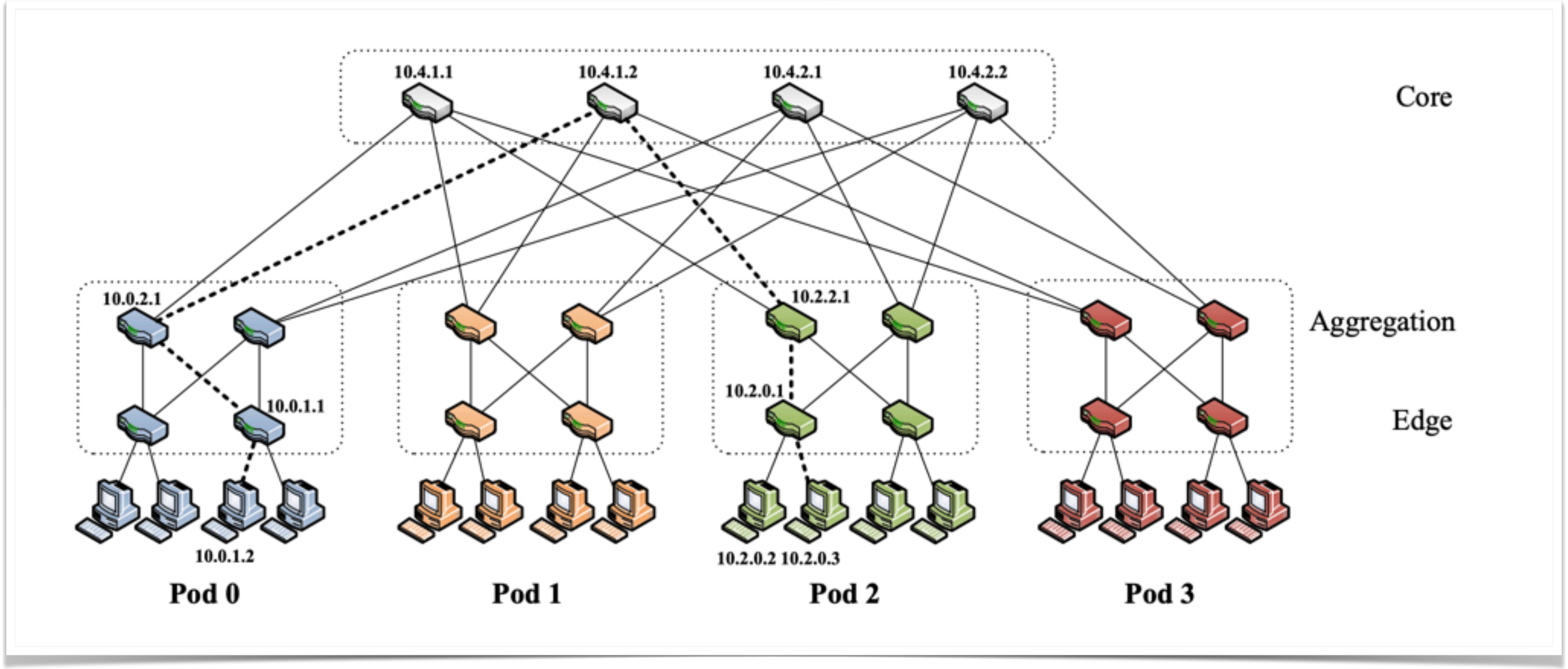
Most networks that have been proposed for parallel processing are based on the Boolean hypercube, but these networks suffer from wirability and packaging problems and require nearly order $n^{3/2}$ physical volume to interconnect n processors. In his influential paper on "ultracomputers" [27], Schwartz demonstrates that many problems can be solved efficiently on a supercomputer based on a shuffle network [28]. But afterwards, Schwartz comments, "The most problematic aspect of the ultracomputer architecture suggested in the preceding section would appear to be the very large number of intercabinet wires which it implies." Schwartz then goes on to consider a "layered" architecture, which seems easier to build, but which may not have all the nice properties of the original architecture.

On the other hand, there are many applications that do not require the full communication potential of a hypercube-based network. For example, many finite-element problems are planar, and planar graphs have a bisection width of size $O(\sqrt{n})$, as was shown by Lipton and Tarjan [19]. Moreover, any planar interconnection strategy requires only $O(n)$ volume. Thus, a natural implementation of a parallel finite-element algorithm would waste much of the communication bandwidth provided by a hypercube-based routing network.

Fat-trees are a family of general-purpose interconnection strategies which effectively utilize any given amount of hardware resource devoted to communication. This paper proves that for a given physical volume of hardware, no network is much better than a fat-tree. Section II introduces fat-tree architectures and gives the logical structure of one feasible implementation. Section III shows how communication on a fat-tree can be scheduled off-line in a near-optimal fashion. Section IV defines the class of *universal* fat-trees and investigates their hardware cost in a three-dimensional VLSI model. Section V contains several combinatorial theorems concerning the recursive decomposition of an arbitrary routing network, and Section VI uses these results to demonstrate that fat-trees are indeed a class of hardware-efficient universal routing networks. Finally, Section VII offers some remarks about the practicality of fat-trees.

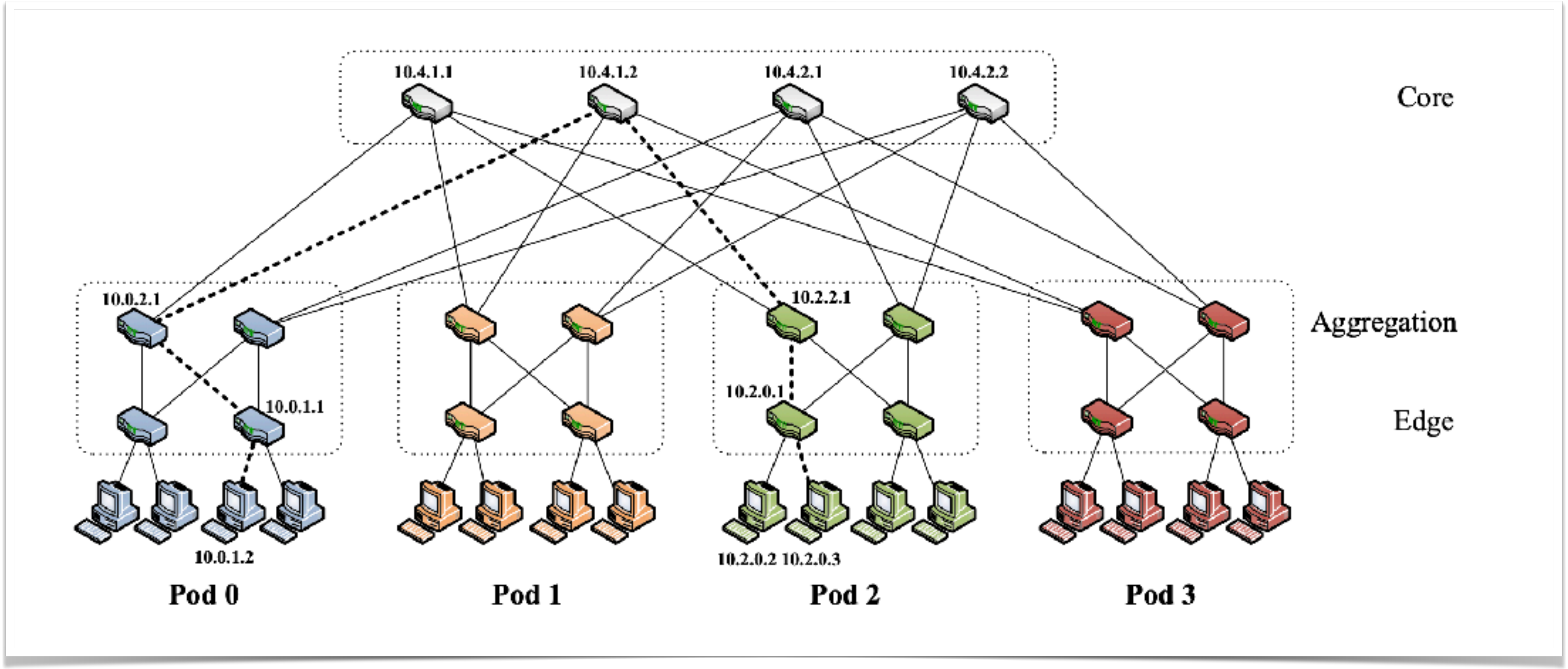
Manuscript received February 1, 1985; revised May 30, 1985. This work was supported in part by the Defense Advanced Research Projects Agency under Contract N00014-80C-0622. A preliminary version of this paper was presented at the IEEE 1985 International Conference on Parallel Processing, St. Charles, IL, Aug. 1985.

The author is with the Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139.



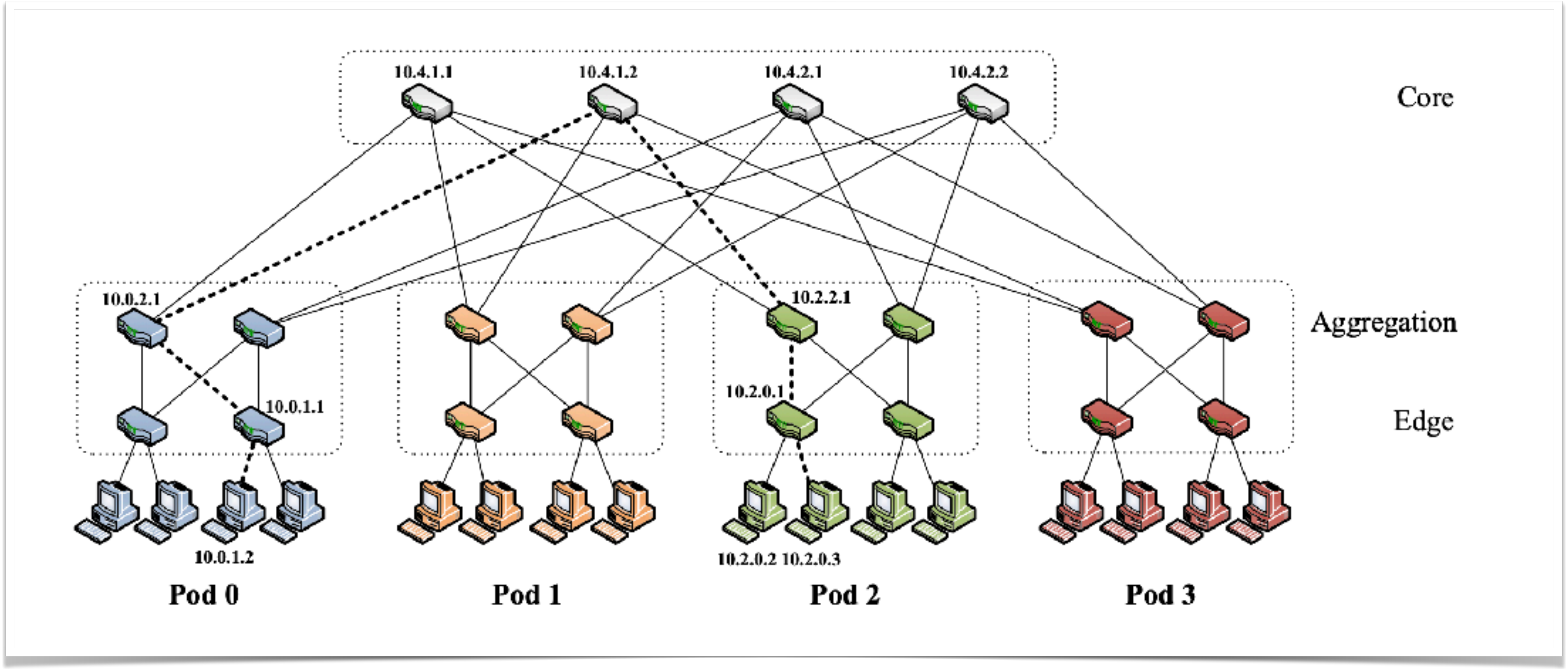
Zoom in a Fat-Tree Example

- K-ary fat tree: three-layer topology (edge, aggregation, and core)



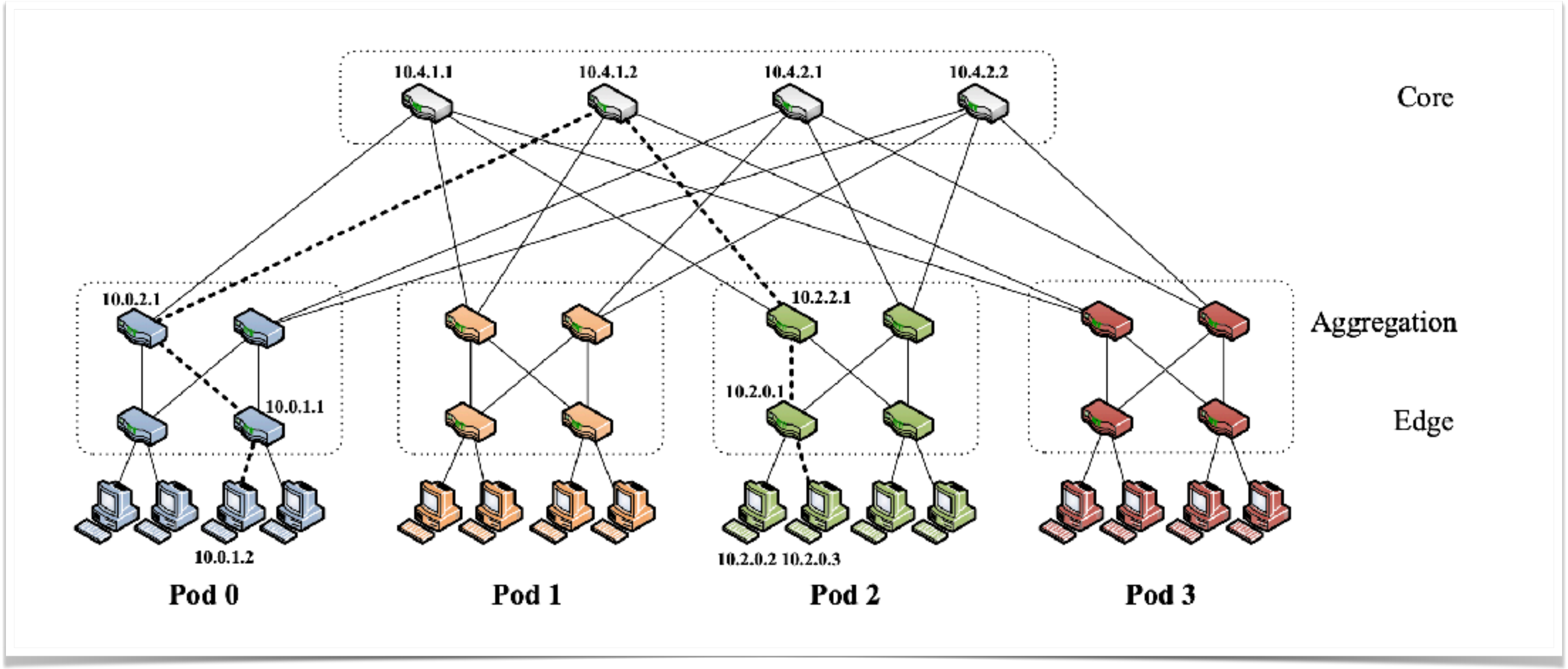
Zoom in a Fat-Tree Example

- K-ary fat tree: three-layer topology (edge, aggregation, and core)
  - Each edge switch connects to K/2 servers and K/2 aggregation switches



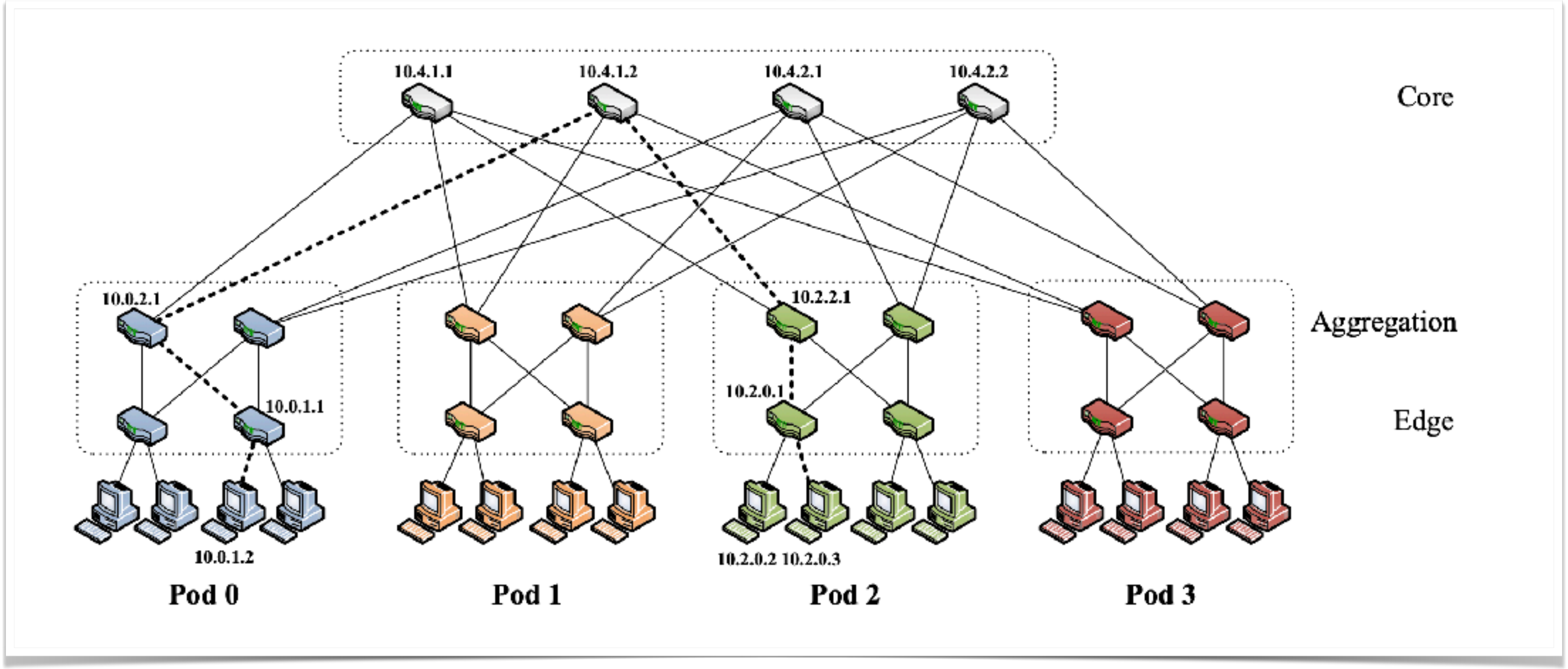
Zoom in a Fat-Tree Example

- K-ary fat tree: three-layer topology (edge, aggregation, and core)
  - Each edge switch connects to K/2 servers and K/2 aggregation switches
  - Each aggregation switch connects to K/2 edge and K/2 core switches



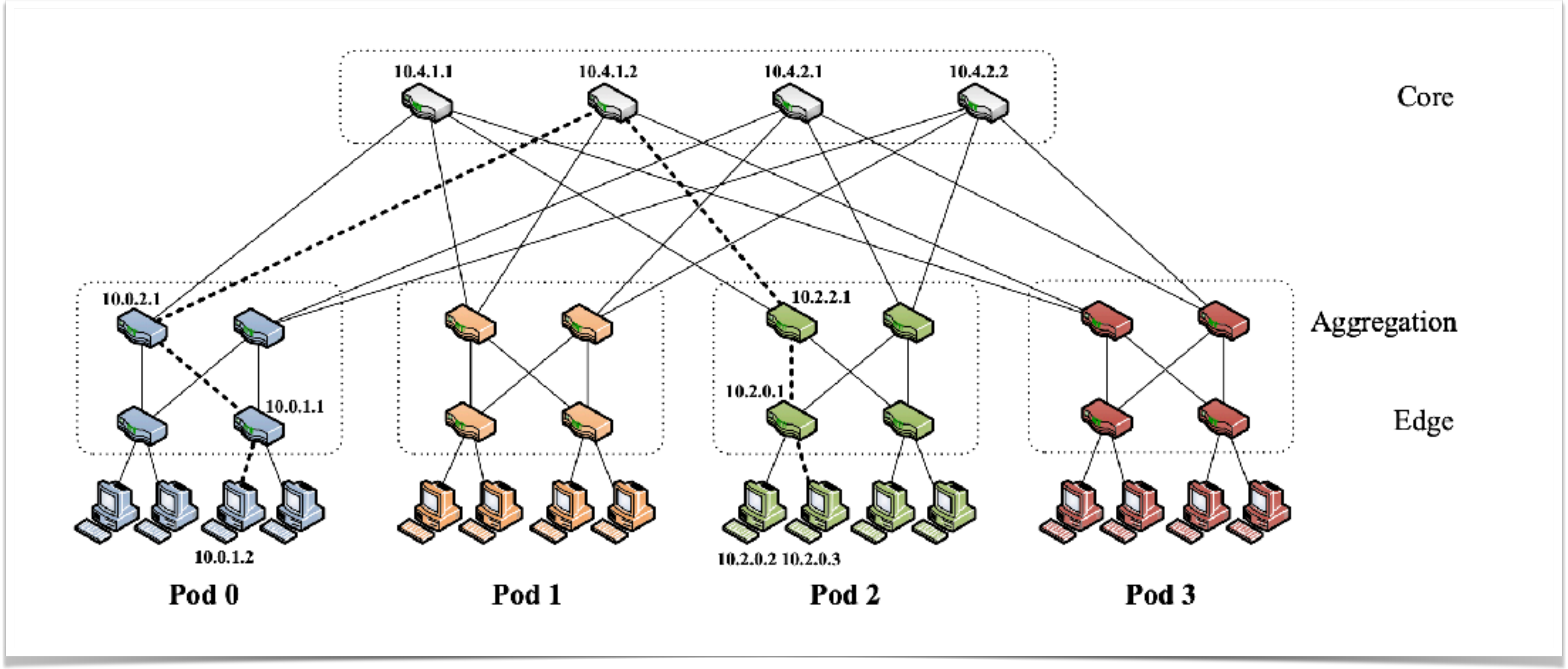
Zoom in a Fat-Tree Example

- K-ary fat tree: three-layer topology (edge, aggregation, and core)
  - Each edge switch connects to K/2 servers and K/2 aggregation switches
  - Each aggregation switch connects to K/2 edge and K/2 core switches
  - (K/2)^2 cores switches



Zoom in a Fat-Tree Example

- K-ary fat tree: three-layer topology (edge, aggregation, and core)
  - Each edge switch connects to K/2 servers and K/2 aggregation switches
  - Each aggregation switch connects to K/2 edge and K/2 core switches
  - (K/2)^2 cores switches
  - Support K^3/4 servers



A Generic Workflow to Construct Fat-Tree Networks

- Step 1: Determine the networking configuration
  - E.g., bandwidth of server NIC and switch port, switching port #
- Step 2: Add intermediate switching stages to match the BW
  - Ingress BW == Egress BW at any switching point (Bandwidth rule)
- Step 3: Apply the scale-out strategy to merge connections
  - Use the Slim and Fat port ratio to decide (Scale-out rule)
- Step 4: Apply the scale-up strategy to add communication paths
  - Added path # relates to switching hops # in the next stage (Scale-up rule)

Summary

- Today
  - Physical connectivity at the rack/cluster scale

- Next lecture
  - Physical connectivity for inter-data centers