Resource Managers

and how do they treat INP resources

Goal

- Resource model for server and network resources
  - Capable of describing current INP solutions
  - Capable of describing any kind of network topology
  - Descriptive, yet simple
- Placement algorithm
  - It must allocate both server and network resources
  - Using the previous model as input
  - Better than a random allocation

RMs' network awareness levels

- VMs proximity-aware
  - Most of the RMs out there can spread VMs across different failure domains
  - Not worth discussing since they do not consider any other kind of network resource
  - E.g., Omega, YARN, Mesos, …
- Bandwidth-aware
  - Some RMs allow tenants to specify bandwidth demands
  - E.g., CloudMirror, Oktopus, Kraken, Proteus, …
- Network resources-aware
  - Rabbani, Md Golam, et al. "On tackling virtual data center embedding problem." *IM 2013: IFIP/IEEE International Symposium on Integrated Network Management. 2013.*

Bandwidth-aware RMs

- They use “virtual network” models
  - Virtual Custer (VC) used by Oktopus and Kraken
  - Time-Interleaved Virtual Cluster (VC with time-varying bandwidth requirements) used by Proteus
  - Virtual Oversubscribed Cluster (VOC) used by Oktopus
  - Tenant Application Graph (TAG) used by CloudMirror
- None of them handle INP resources
- Oktopus and Kraken assume that every VM can be place everywhere
  - They completely ignore server-local resource requirements
- Kraken allows tenants to *upgrade* their bandwidth requirements (replacement)

Network resource-aware RM

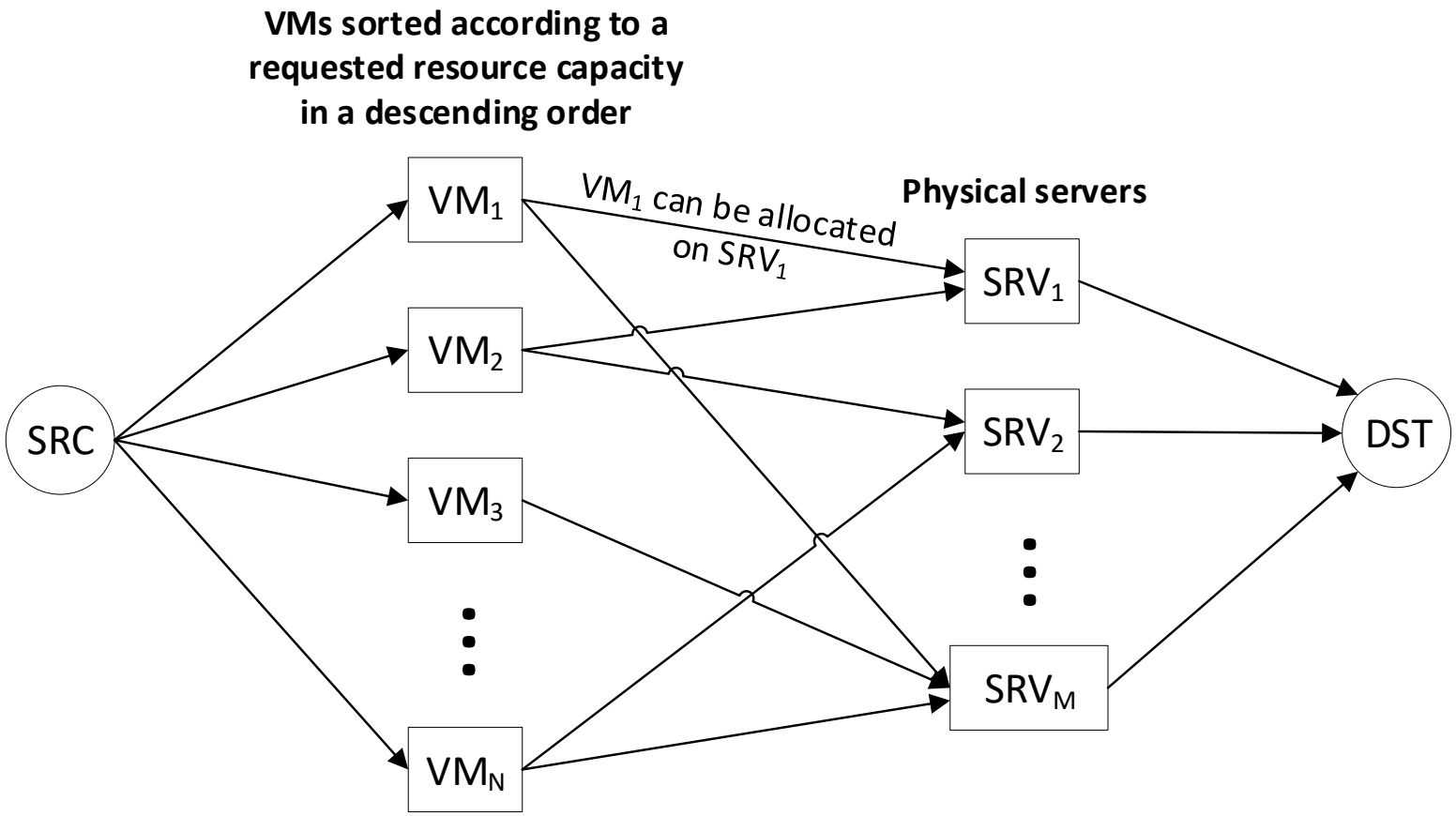
- Rabbani, Md Golam, et al. "On tackling virtual data center embedding problem." *IM 2013: IFIP/IEEE International Symposium on Integrated Network Management. 2013.*
- Embedding solution that allows tenants to explicitly specify
  - Server resources
  - **Switch** resources
  - Bandwidth demands

Network resource-aware RM

- The used resource model is a graph containing:
  - A set of VM resources $(k_i = v_i, \dots)$
  - A set of virtual switches resources $(k_i = v_i, \dots)$
  - A set of virtual links connecting the above entities (bandwidth)
- Placement algorithm divided in three steps:
  - VMs placement
  - Virtual switches placement
  - Virtual links placement

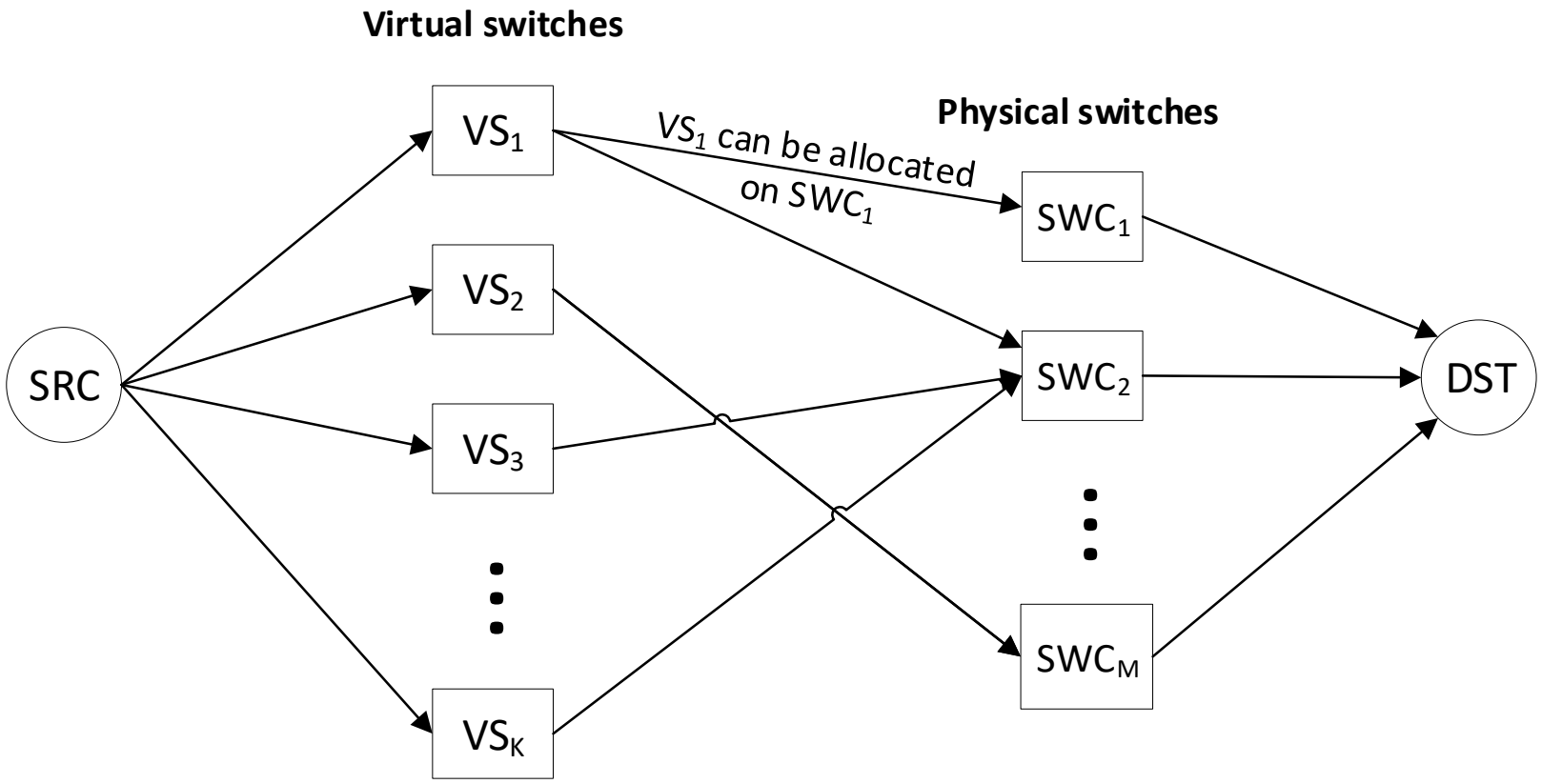
Step 1: VMs placement

- The problem is reduced to a min-cost flow problem



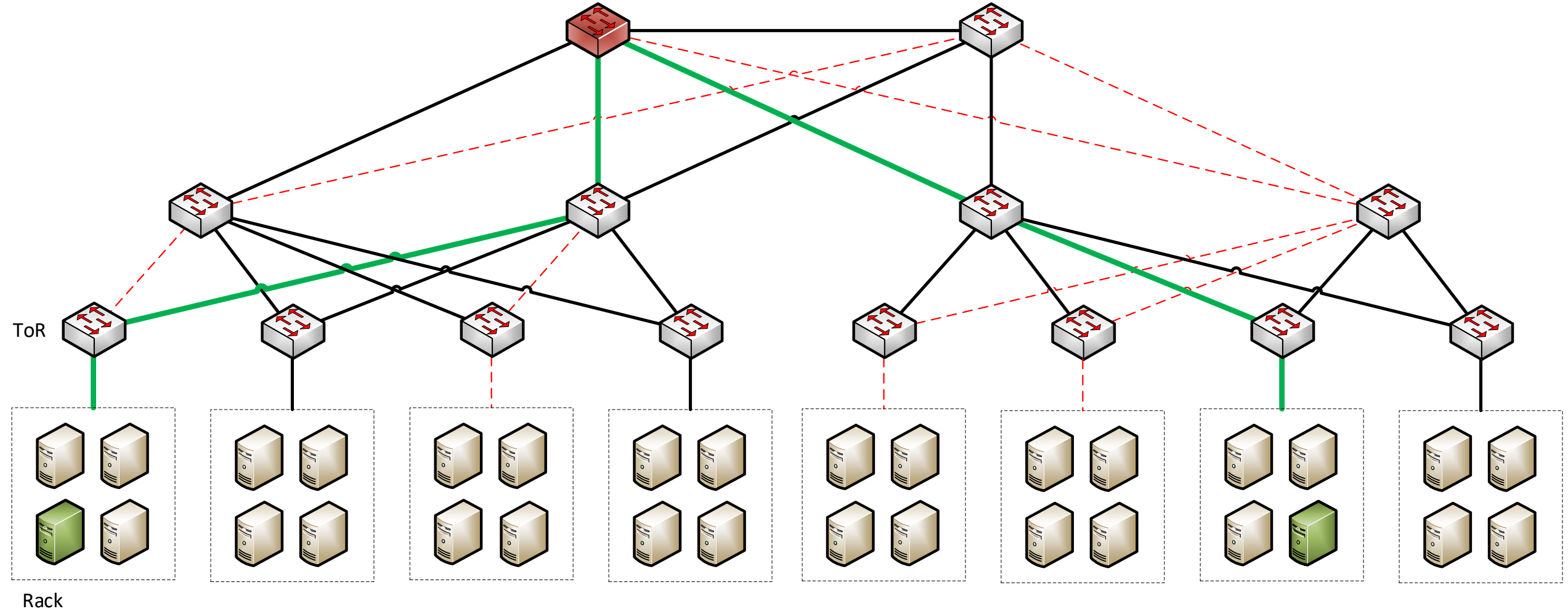
Step 2: virtual switches placement

- Same min-cost flow problem as before

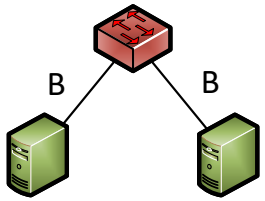


Step 3: virtual links placement

- Given two already-placed entities (e.g., a VM and a virtual link), a virtual link is mapped to the physical shortest path



Allocation request:



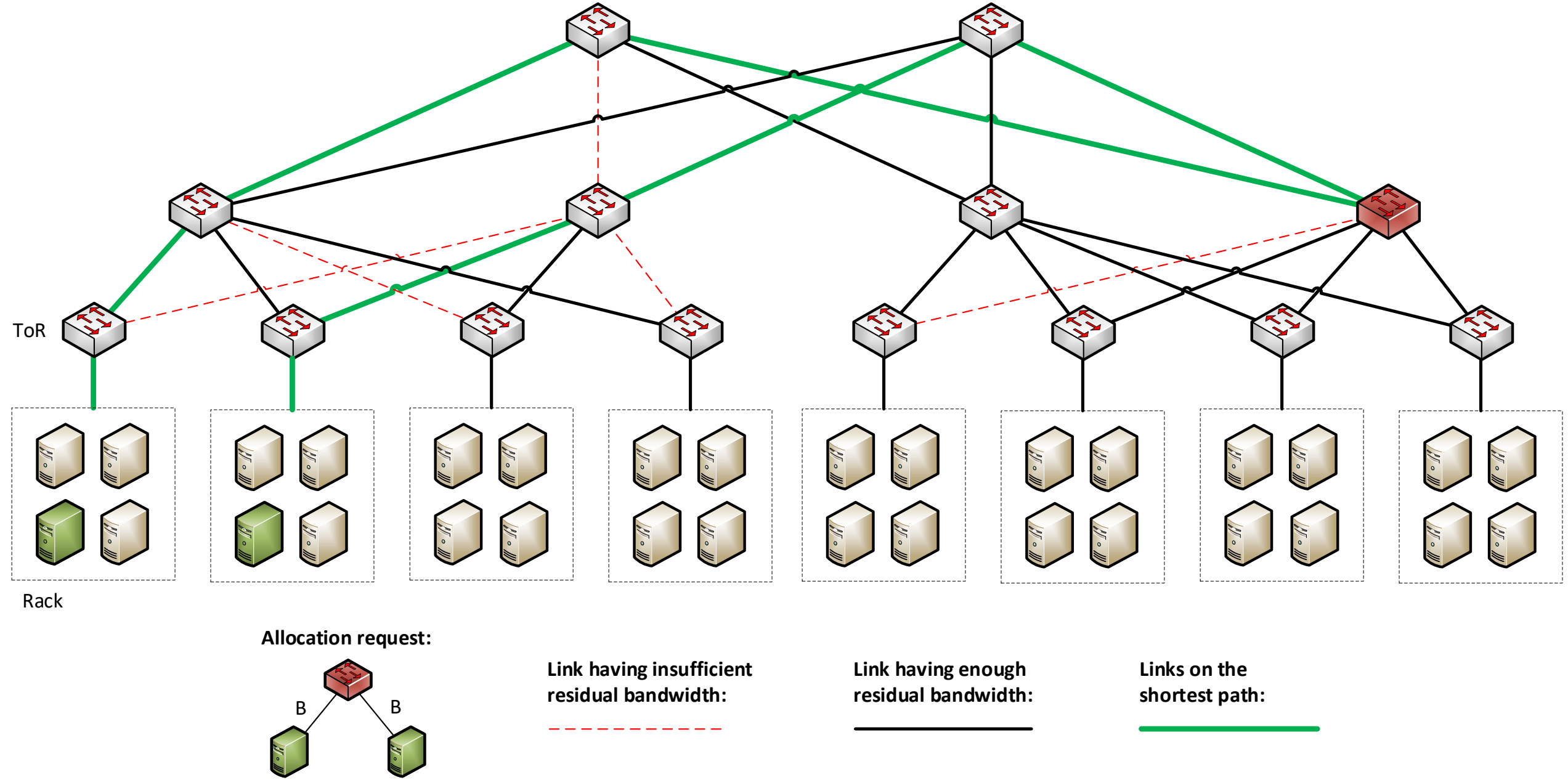
Link having insufficient residual bandwidth:

Link having enough residual bandwidth:

Links on the shortest path:

Network resource-aware RM: 1st problem

- Inefficient virtual switches placement
  - Virtual switches are mapped to physical ones without considering where VMs have been previously mapped in the physical topology
- Bad switch mapping example in the next slide


ToR
Rack
Allocation request:
B
B
Link having insufficient residual bandwidth:
Link having enough residual bandwidth:
Links on the shortest path:

Network resource-aware RM: other problems

- One resource dimension considered
  - The model supports multiple resource dimensions (CPU cores, memory, etc.)
  - The algorithm and the example consider just one dimension
    - Crucial when ordering VMs during the VM mapping phase
    - VMs are ordered according to just one "*requested resource capacity*" so that a request can be instantly rejected in case there is at least one VM requesting for too many resources
- The placement algorithm tries to map as most VMs to the same physical server in order to minimize server resource fragmentation
  - It is not aware of failure domains
  - Not fault tolerant