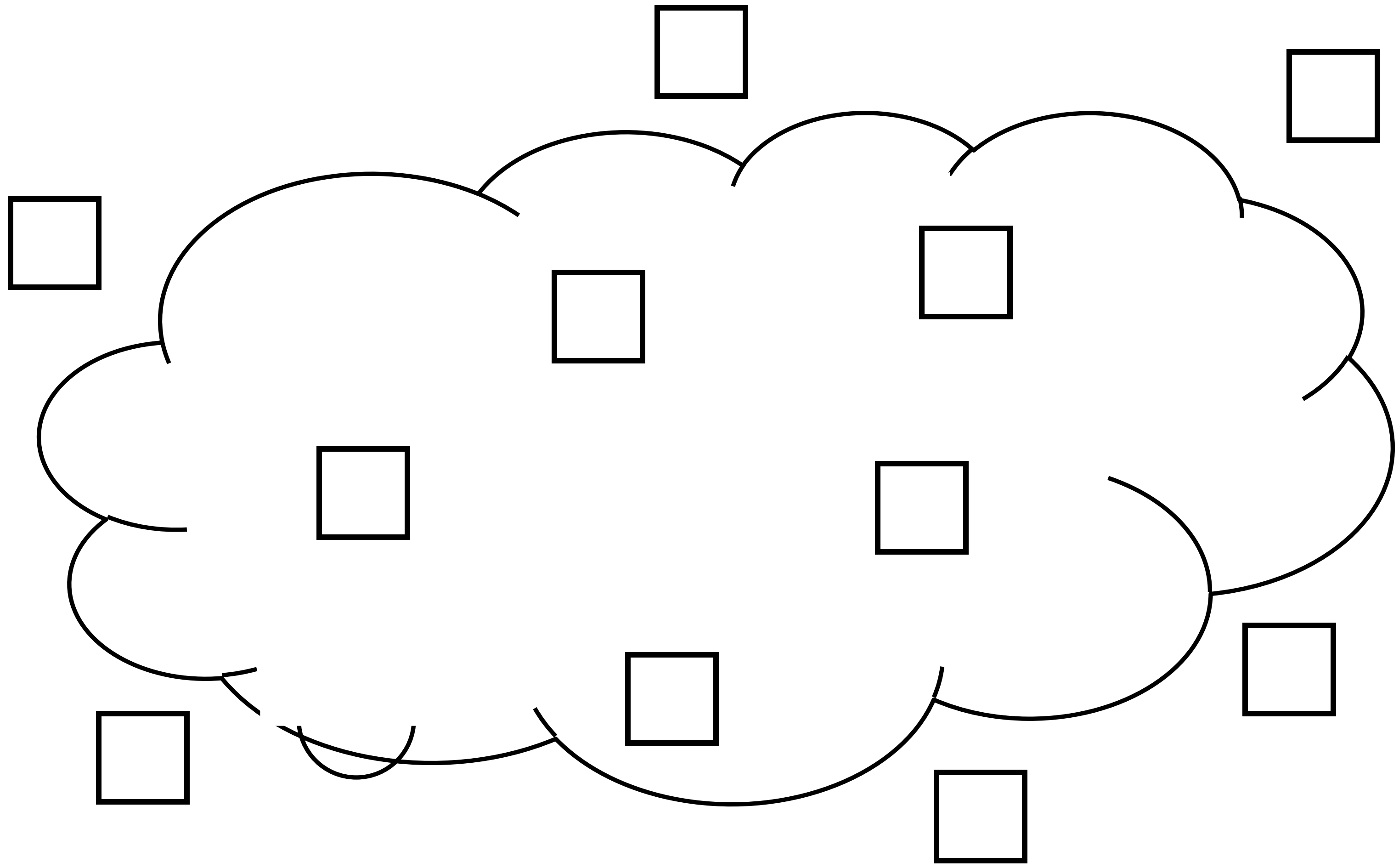
Resource Specifications

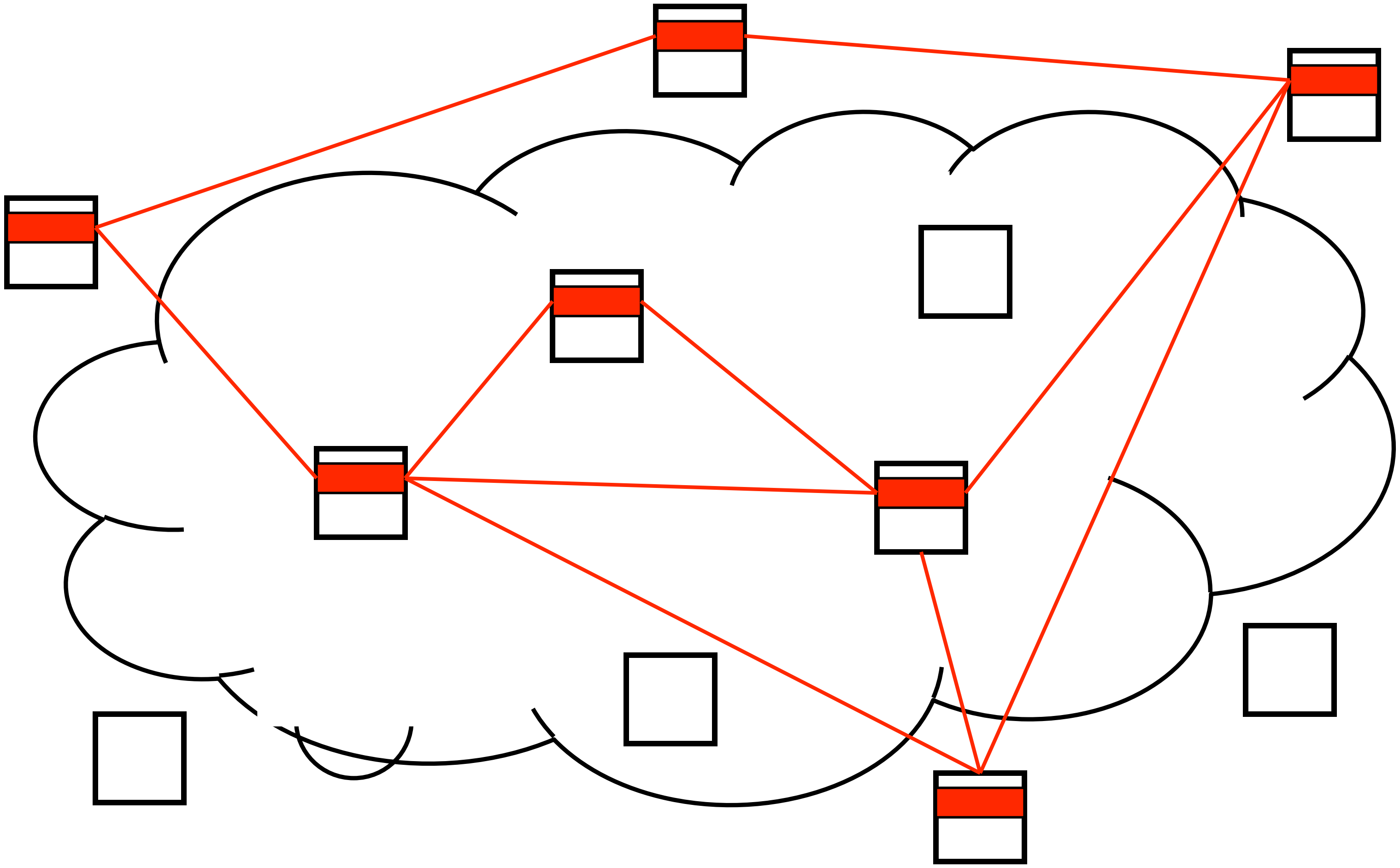
(and end-to-end slices)

Larry Peterson
Princeton University

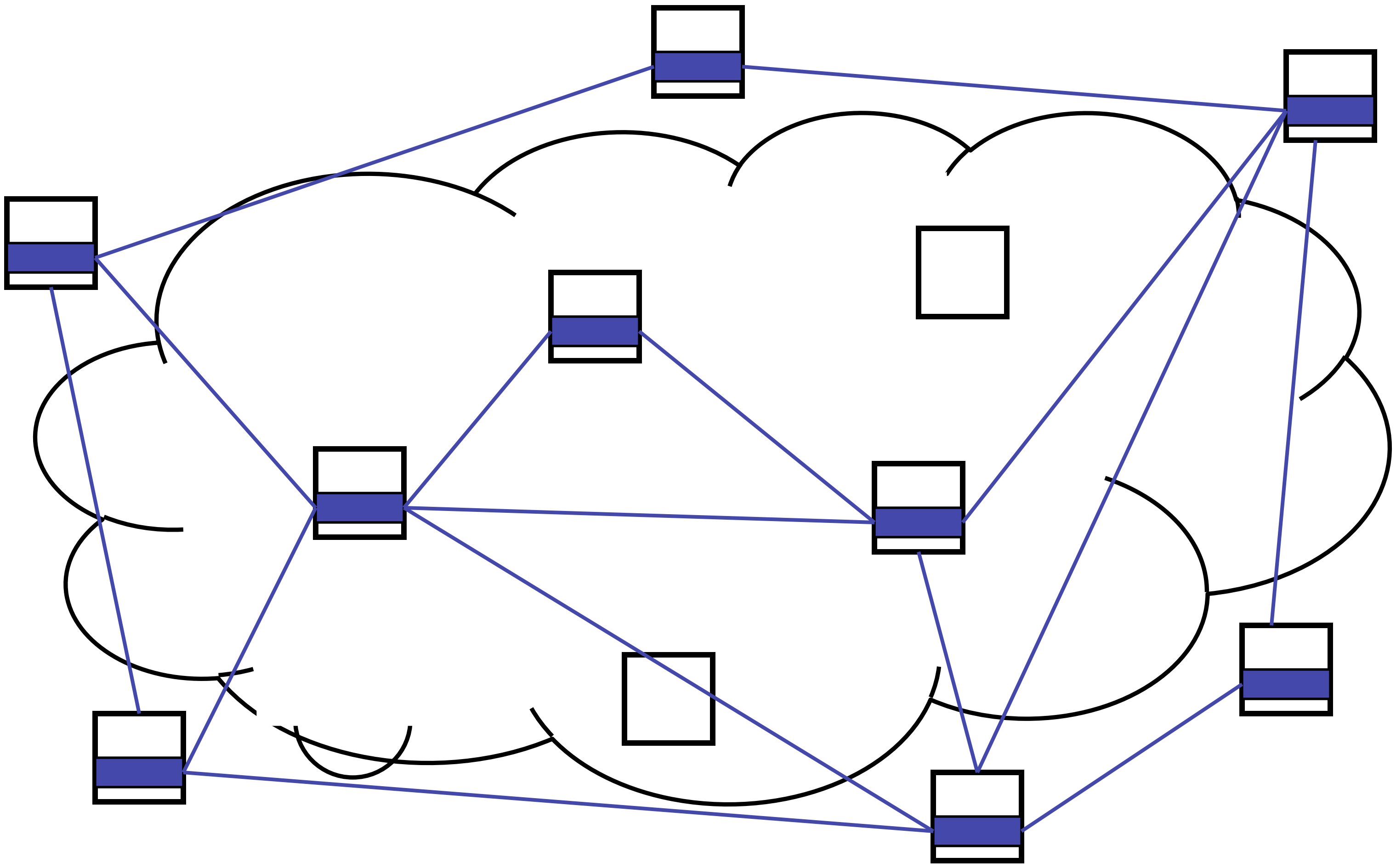
A Bunch of Nodes



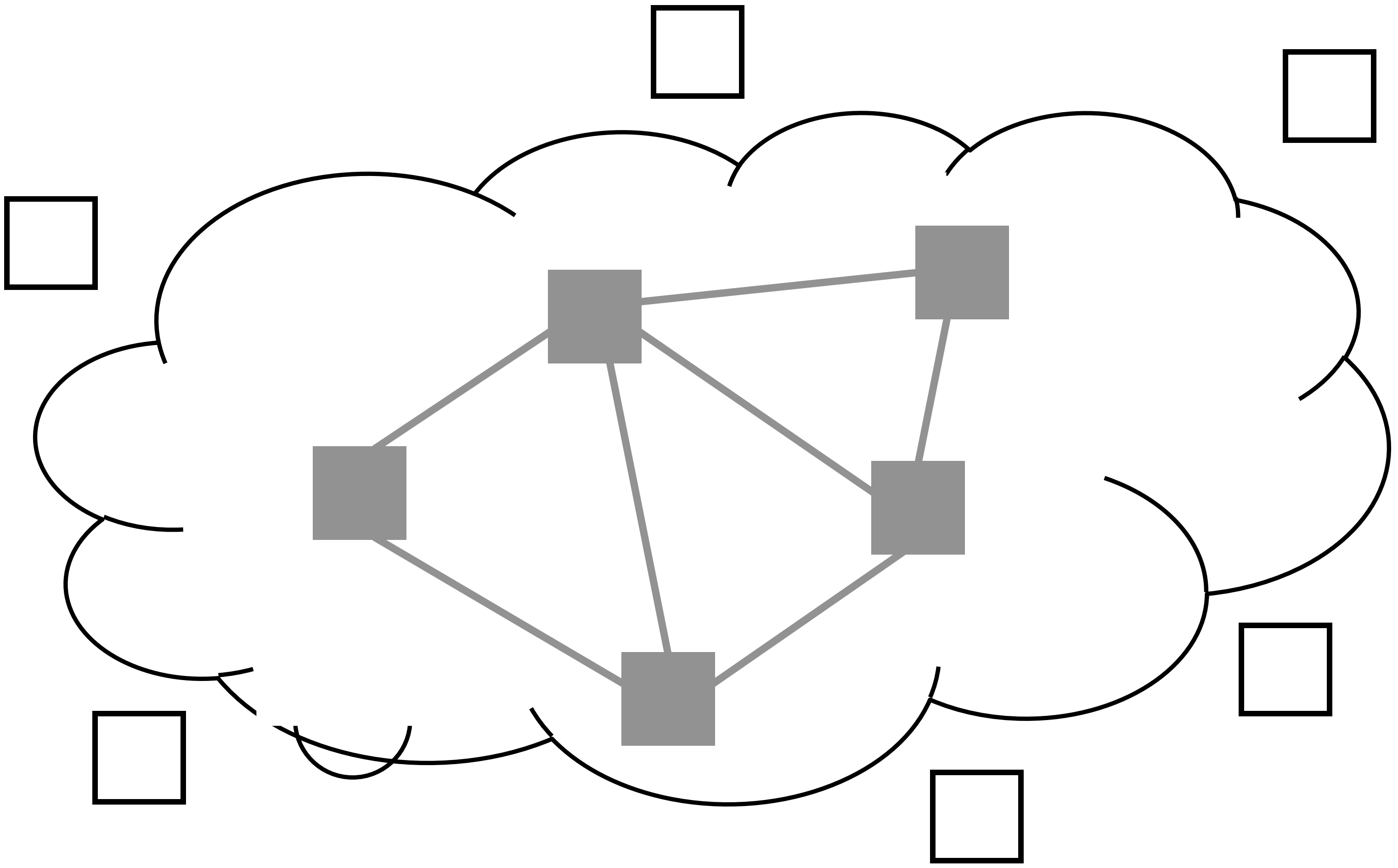
My Slice – My Topology



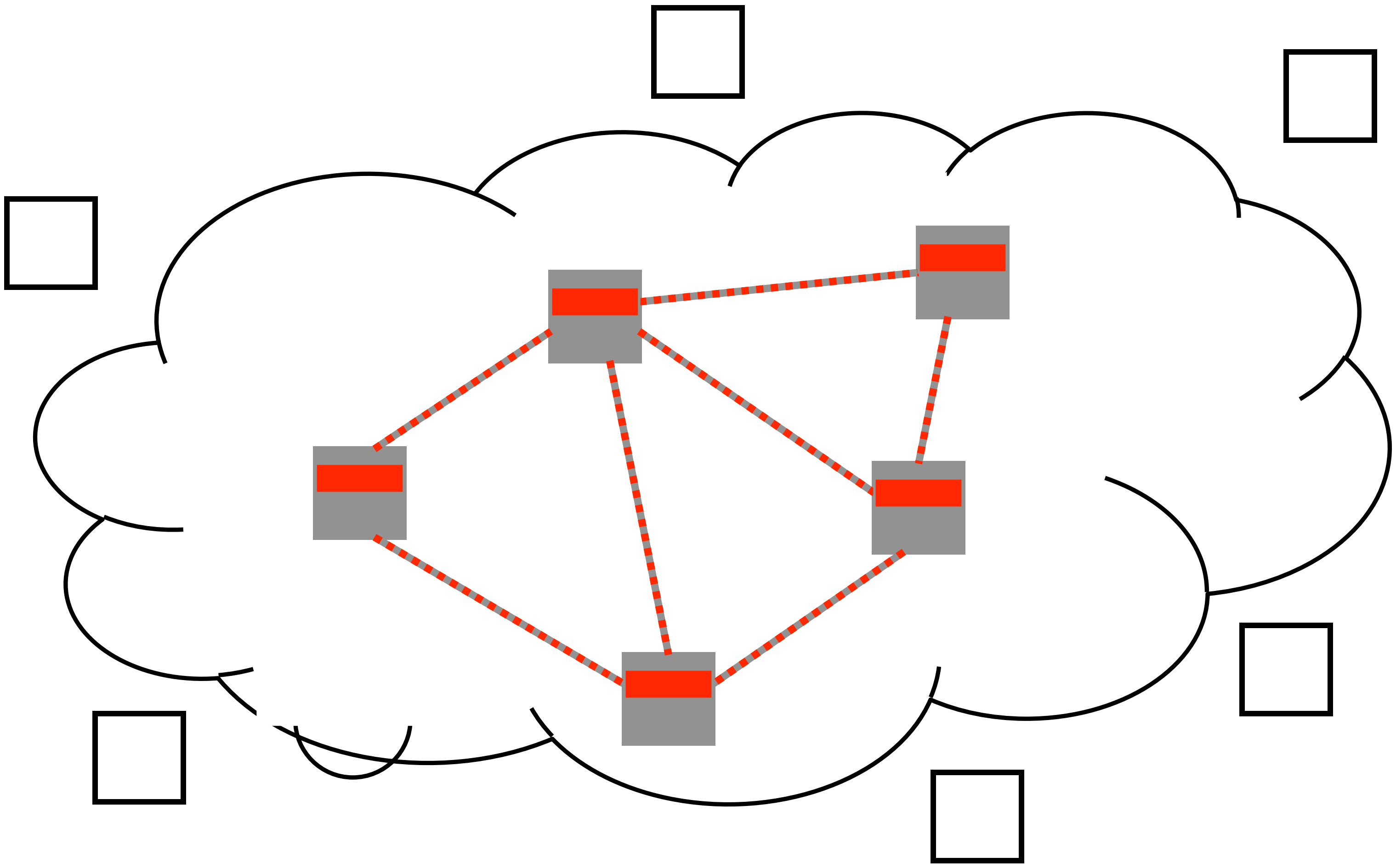
Your Slice – Your Topology



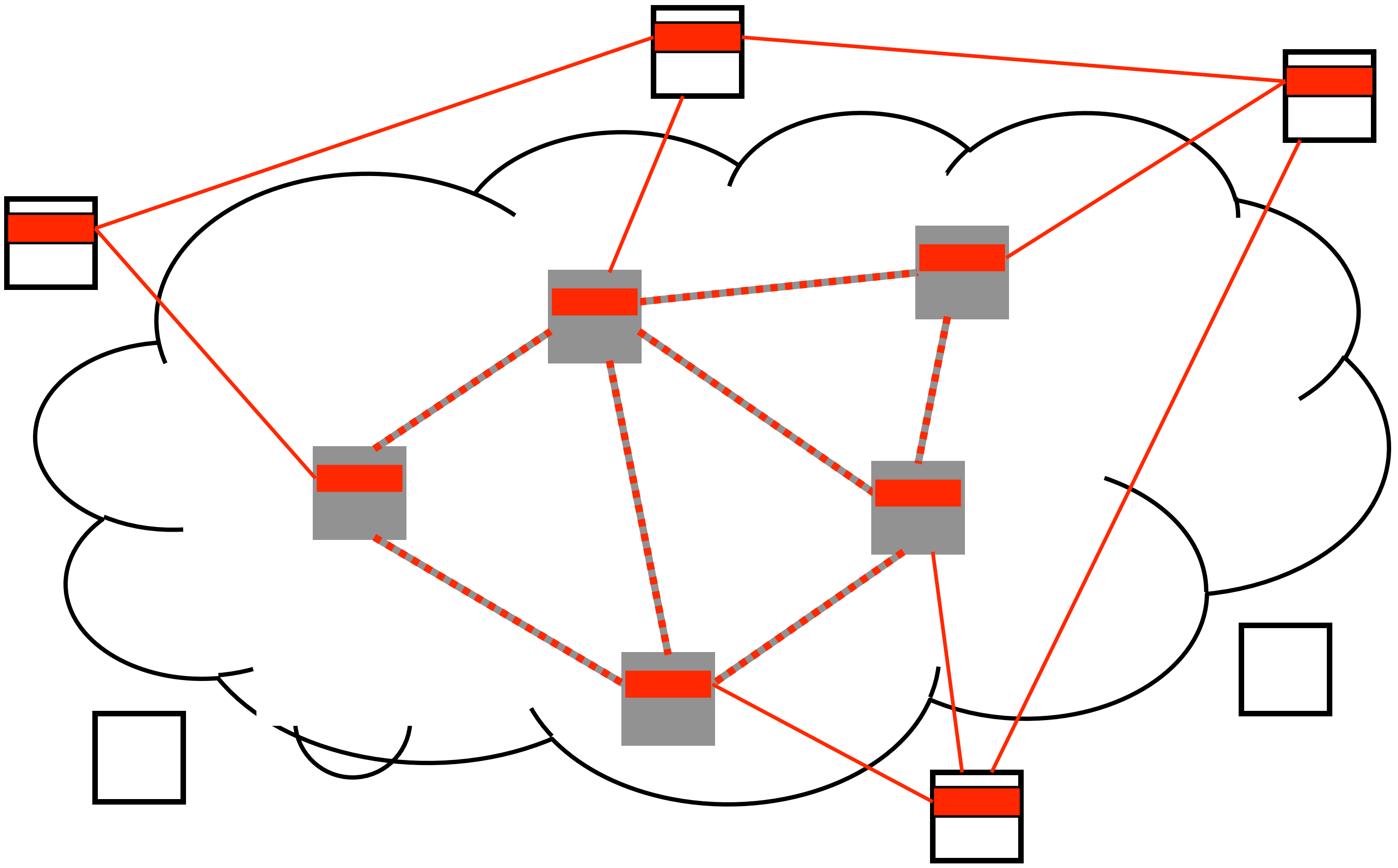
Some Nodes are Special



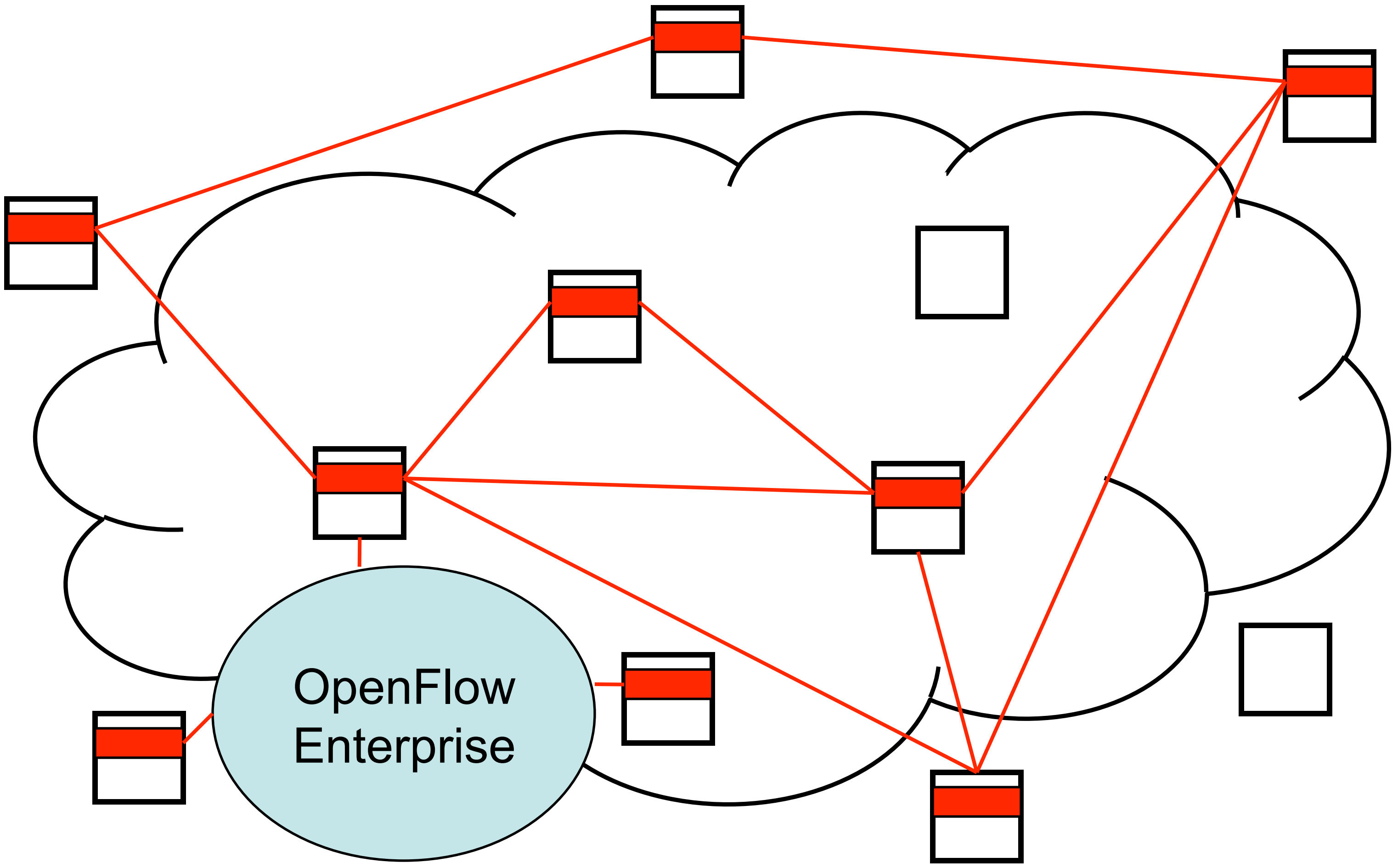
My Slice is Special



Or Includes a Special Subset



Another Special Subset



RSpec – Two Problems

- Interface Negotiation – Introspection
  - Learn the set of resources an aggregate supports
  - Program-heavy (return WSDL)
    - ➤ SetMemory(value)
    - ➤ SetCPU(value)
    - ➤ SetLink(value)
    - ➤ …
  - Data-heavy (return XSD)
    - ➤ SetResources(type=value)
- Resource Negotiation
  - Learn the “amount” of resource an aggregate will grant you

Resource Negotiation

- Today

```
RSpec = GetResources( )
SetResources(RSpec)
```

- Generalize

```
until successful {
   result = SetResources(Request)
   …modify Request…
 }
```

- How do we ensure progress (and termination)?

Resource Negotiation

- Aggregate returns…
  - *Capacity* – what it will say yes to (XSD)
  - *Policy* – how to interpret this capacity (XSLT)

    P(Request, Capacity) = True => request will be honored

    P(Request, Capacity) = False => request will be honored
- Examples
  - P(R, C) → Yes if R and C are the same graph
    - ➤ VINI today
  - P(R, C) → Yes if R is a subset C
    - ➤ VINI tomorrow
  - P(R, C) → Yes if R is subset of C and site sliver cnt ok
    - ➤ PlanetLab today

Resource Negotiation

- Best Part…
  - Policies can be composed (multi-aggregate slice mgrs)
  - Peering policies can be expressed and verified
  - Maintaining polices simplified (defined in single place)
  - Greater degree of automation (load-dependent)