Lecture 6: Software Defined Networks

CSE 222A: Computer Communication Networks

Alex C. Snoeren

UCSDCSE

Thanks: Nick McKeown, Jeremy Stribling, Kai & Lin

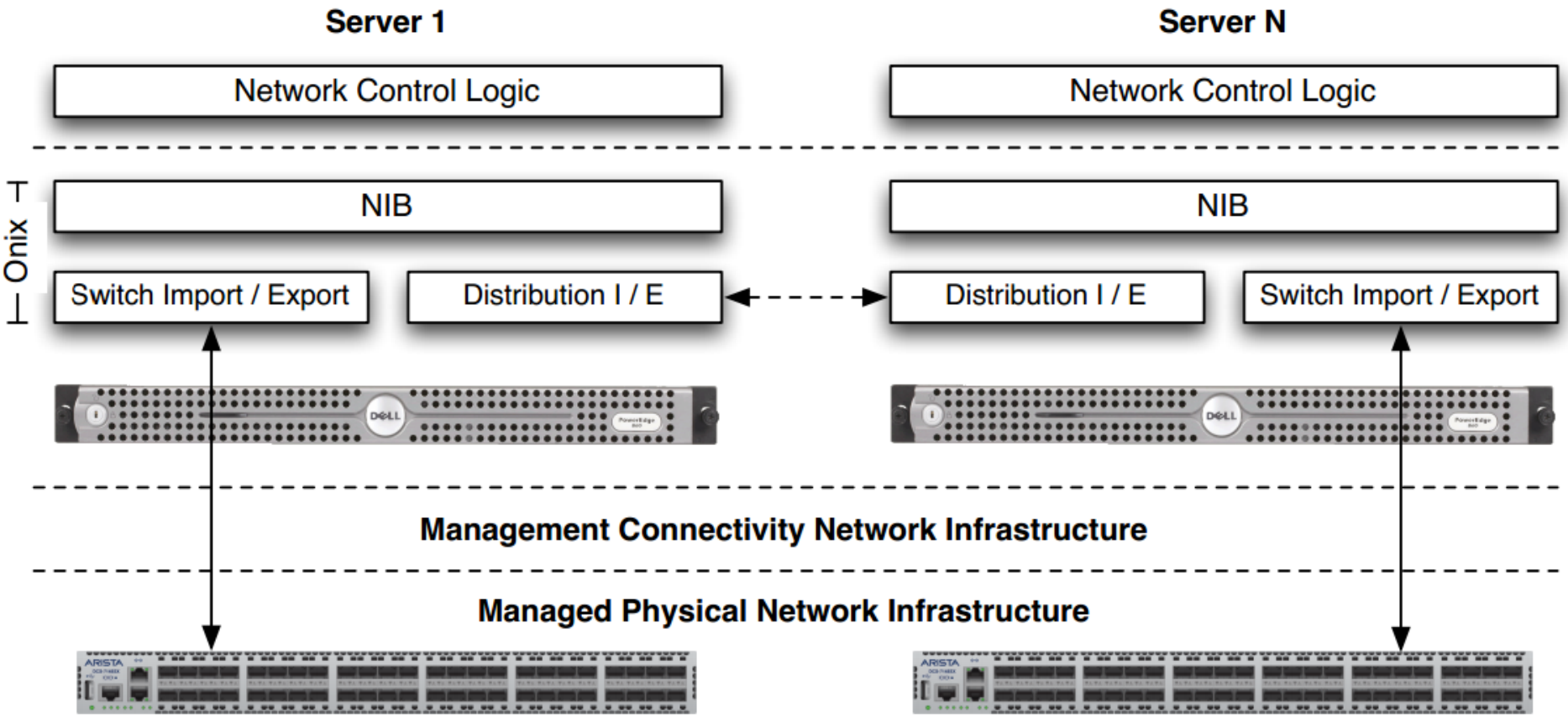
Lecture 6 Overview

- Project discussion
- Onix discussion
- Datacenter introduction

What is a Network OS?

- ONIX provides abstractions for network management
- Basic functionalities:
  - State distribution primitives between controllers and network elements.
  - Virtualized network elements
- A global view & control of the network
  - Very different from Active Networking vision (node local)

Onix Architecture

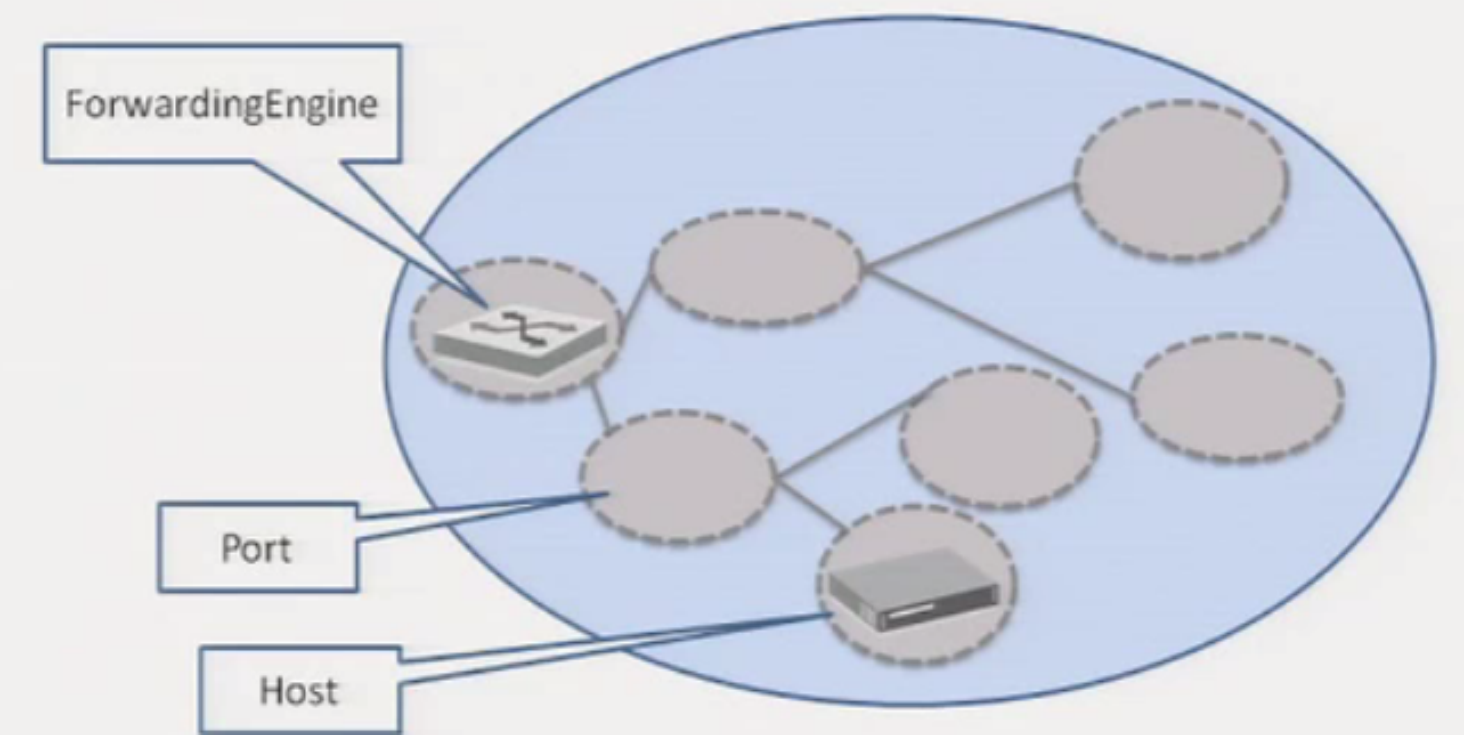


Design Requirements

- ***Generality***: Support a wide range of network management applications
- ***Scalability***: No inherent limitations due to the platform
- ***Reliability***: Graceful failure handling
- ***Performance***: Sufficient performance

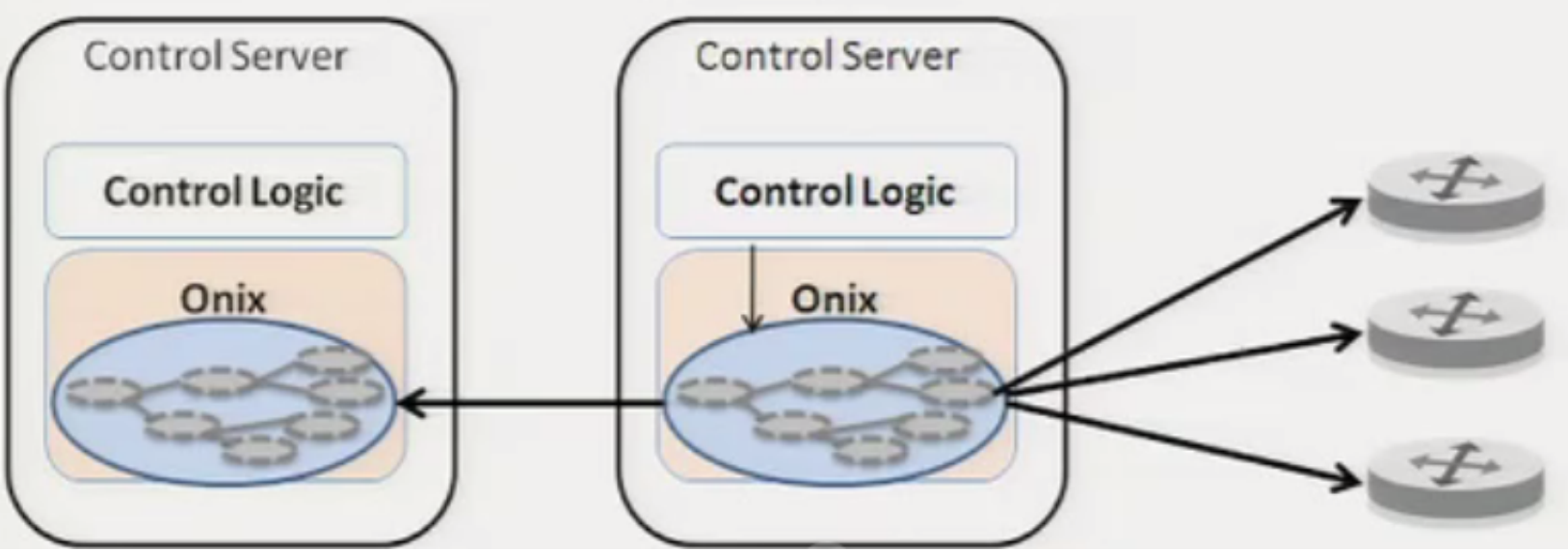
Onix API

- Developers program against a network graph
- Nodes represent physical network entities



Network Information Base

- The NIB is the focal point of the system
  - State for applications to access
  - External state changes **imported** into it
  - Local state changes **exported** from it



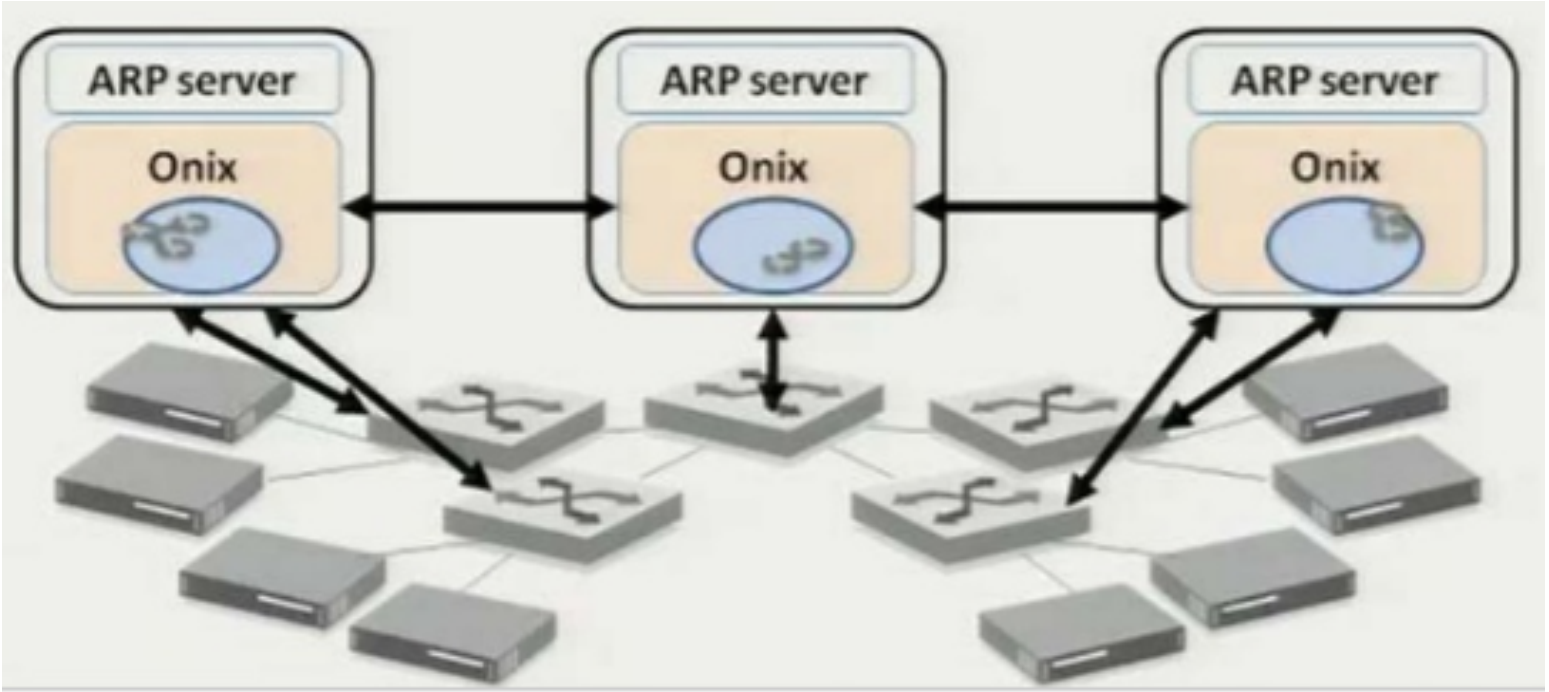
Scalability

- Physical controller becomes a bottleneck:
  - Memory: to keep NIB
  - CPU and bandwidth: to process events

- This is a classic distributed systems problem
  - Partitioning
  - Aggregation

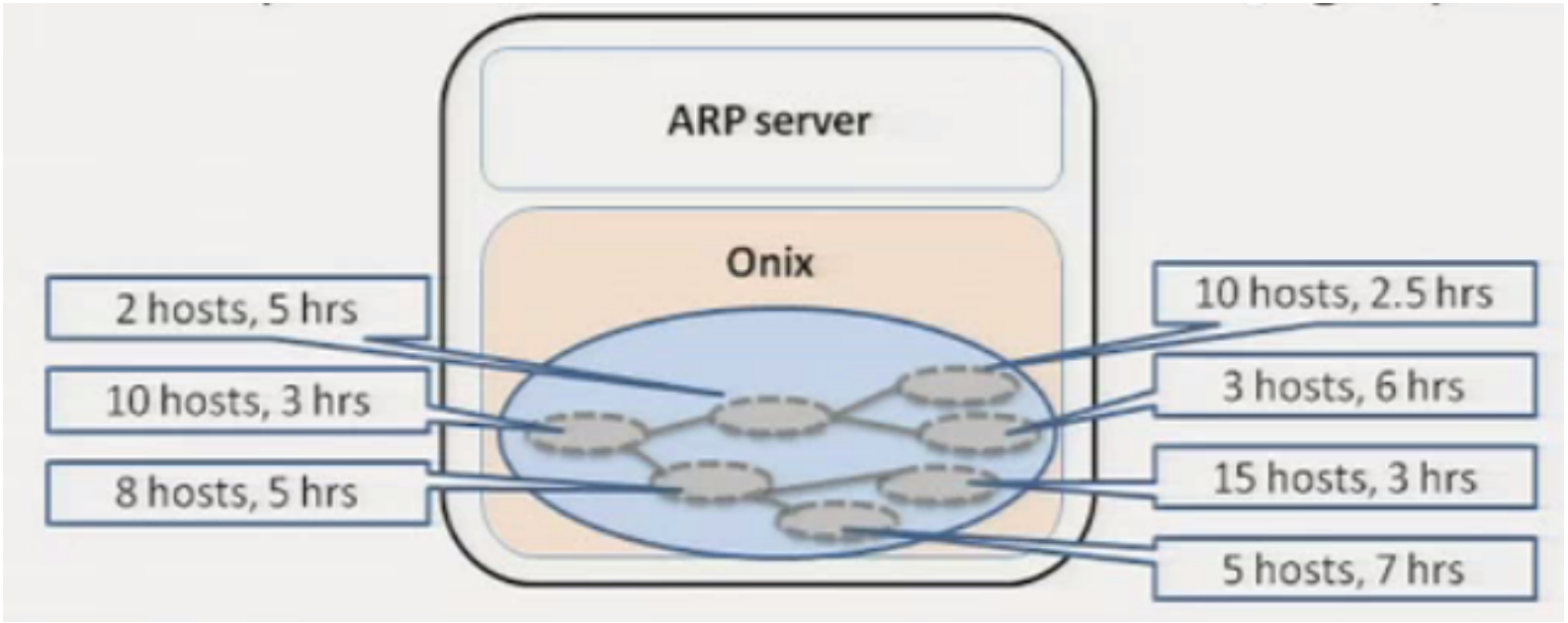
Partitioning For Scale

- Multiple dimensions available to applications:
  - Onix instances with different computations tasks
  - Onix instances have only subsets of the NIB
  - Switches connect to a subset of Onix instances



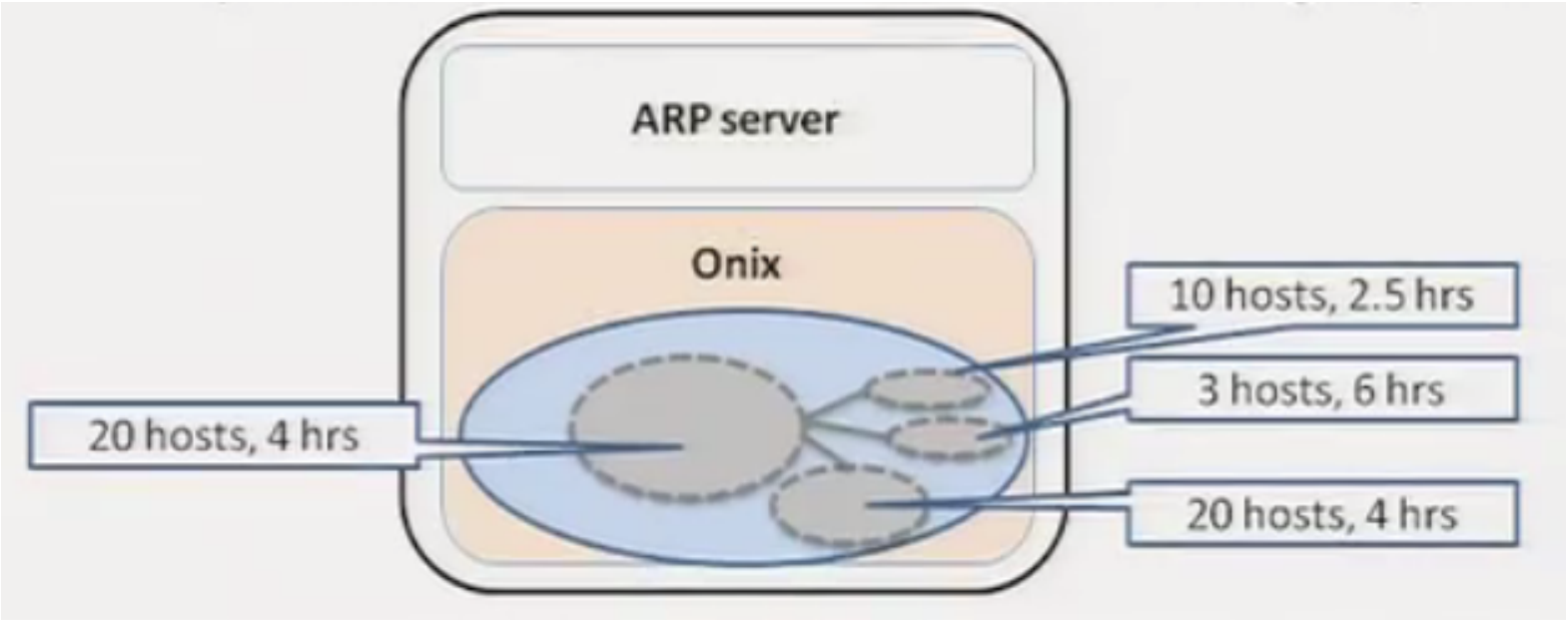
Aggregating for Scale

- Reduce fidelity of information before disseminating within the cluster



Aggregating for Scale

- Reduce fidelity of information before disseminating within the cluster



Consistency

- Onix provides two storage options
  - Replicated transactions (SQL) storage
  - One-hop memory-based DHT
- Do we need strong consistency for forwarding state between the controller and routers?
- Do we need strong consistency for NIB stored in controllers?
- Is it reasonable to ask applications for consistency preference and resolving conflicts?

Reliability

- ***Network Element & Link Failures***: Applications' responsibility
- ***Connectivity Infrastructure Failures***: Assumed reliable
- ***Onix Failures***: Onix provides distributed coordination facilities provided for app failover

Discussion

- What are the security implications of Onix?
- Is Onix flexible and general enough for networking management?
- Are there applications you can run on a centralized network but not on a distributed one?

Cloud Computing

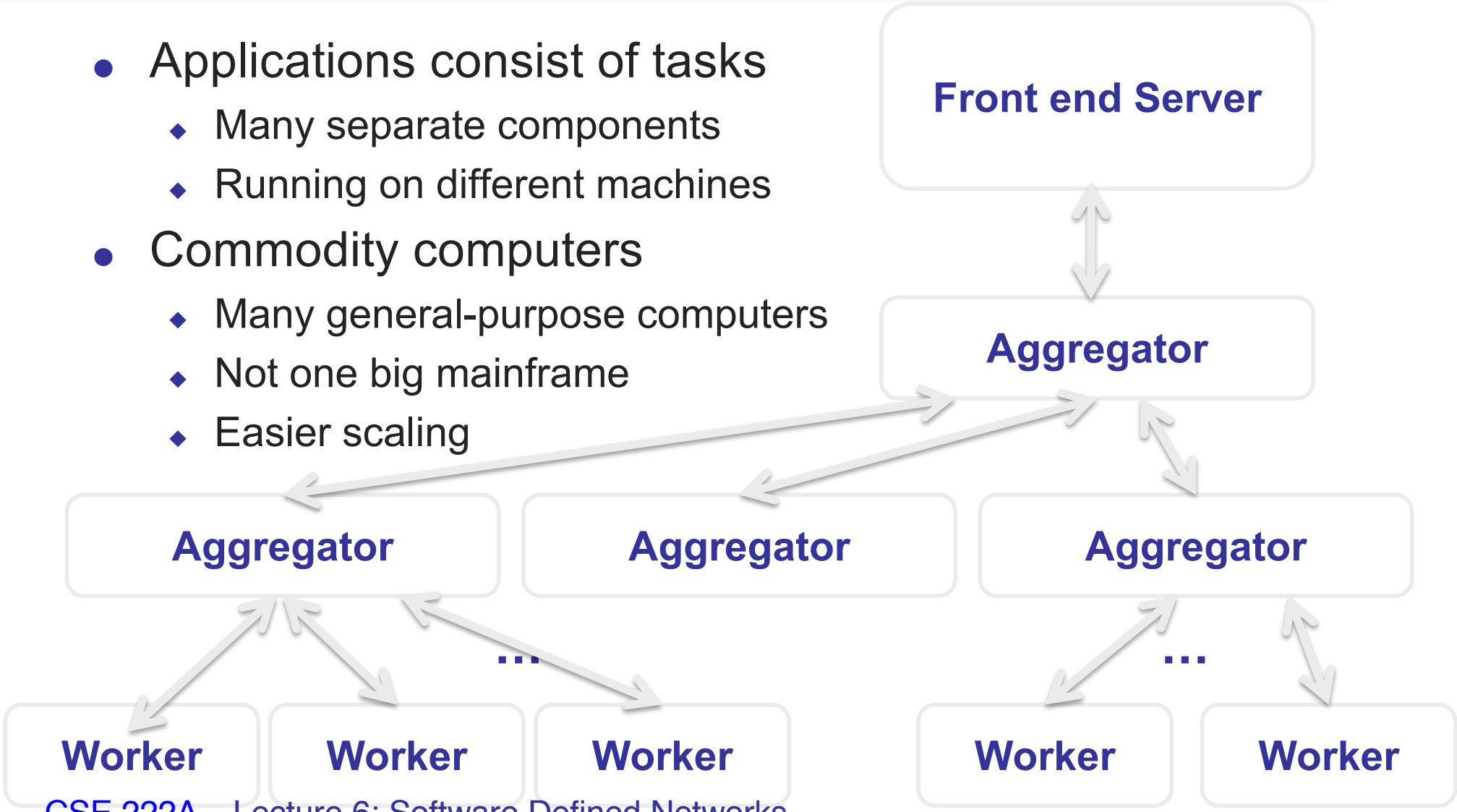
- Elastic resources
  - Expand and contract resources
  - Pay-per-use
  - Infrastructure on demand
- Multi-tenancy
  - Multiple independent users
  - Security and resource isolation
  - Amortize the cost of the (shared) infrastructure
- Flexible service management
  - Resiliency: isolate failure of servers and storage
  - Workload movement: move work to other locations

Cloud Service Models

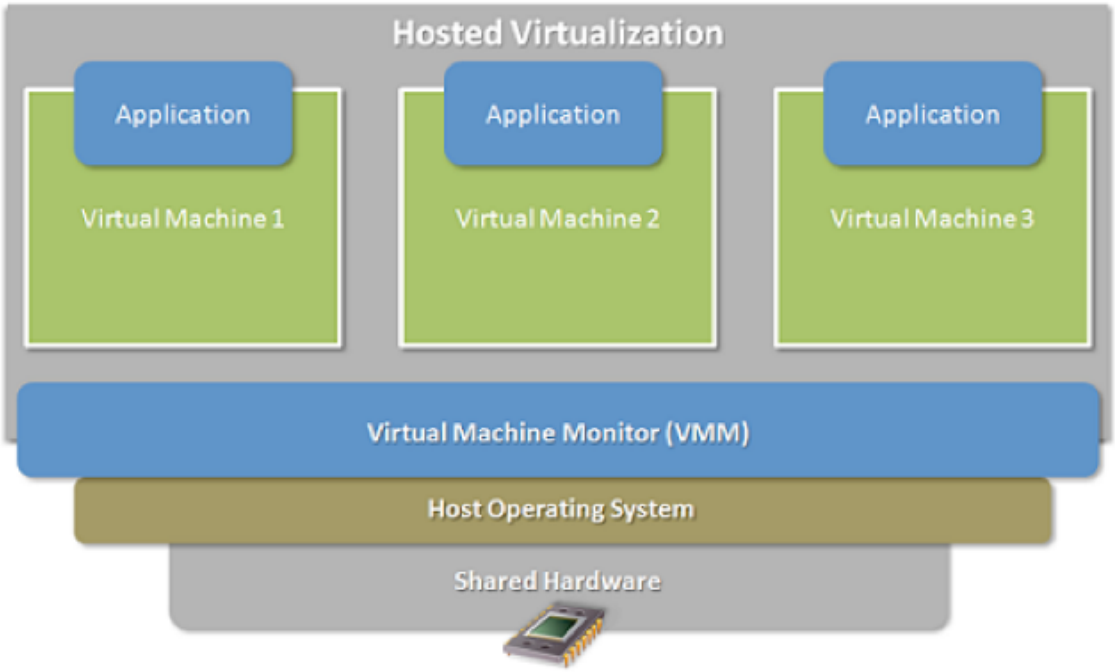
- Software as a Service (Saas)
  - Provider licenses applications to users as a service
  - e.g., customer relationship management, email, …
  - Avoid costs of installation, maintenance, patches, …
- Platform as a Service (Paas)
  - Provider offers software platform for building applications
  - e.g., Google's App-Engine
  - Avoid worrying about scalability of platform
- Infrastructure as a Service (Iaas)
  - Provider offers raw computing, storage, and network
  - e.g., Amazon's Elastic Computing Cloud (EC2)
  - Avoid buying servers and estimating resource needs

Multi-Tier Applications

- Applications consist of tasks
  - Many separate components
  - Running on different machines
- Commodity computers
  - Many general-purpose computers
  - Not one big mainframe
  - Easier scaling

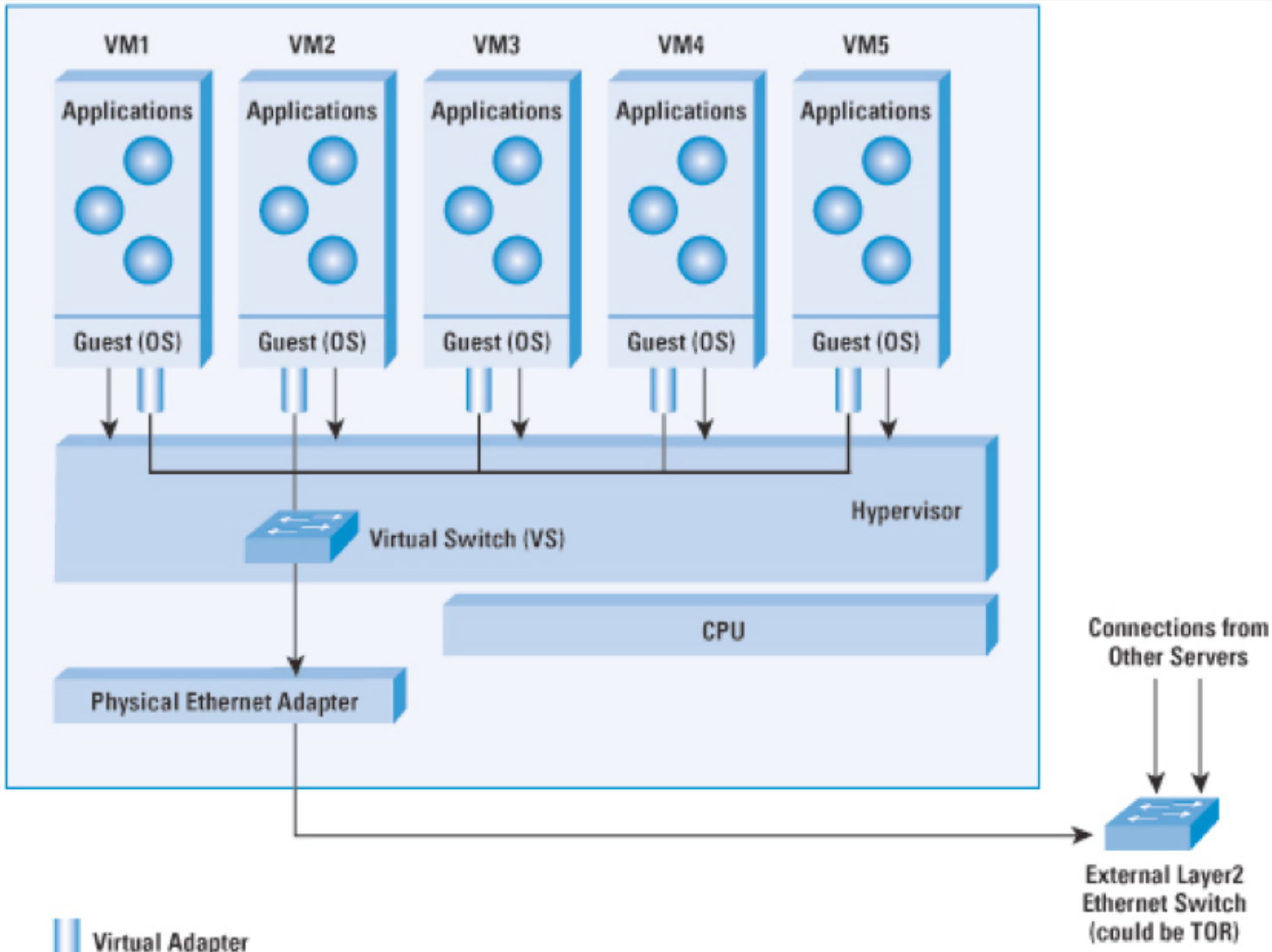


Host Virtualization



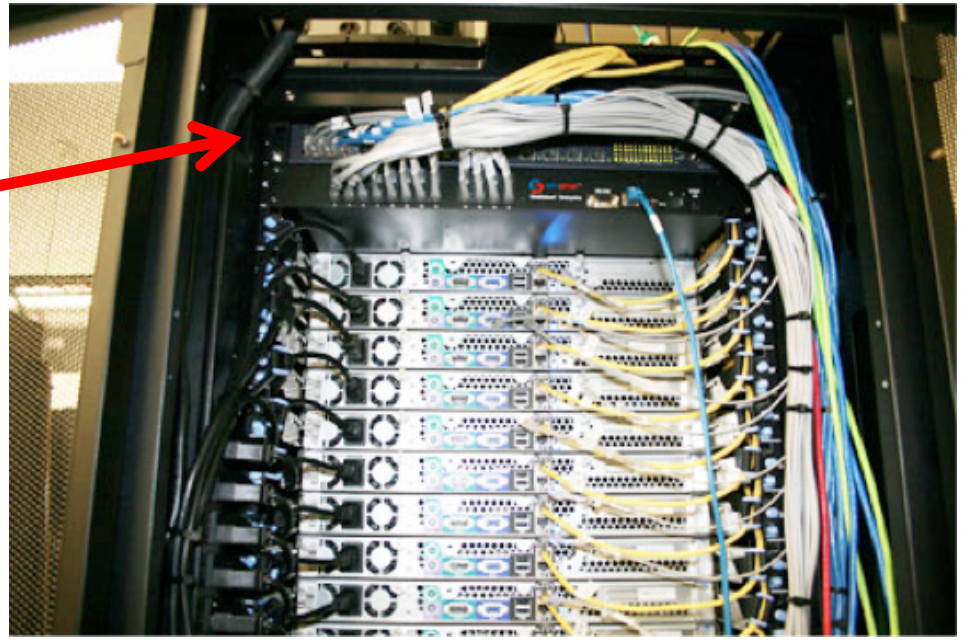
- Multiple virtual machines on one physical machine
- Applications run unmodified as on real machine
- VM can migrate from one computer to another

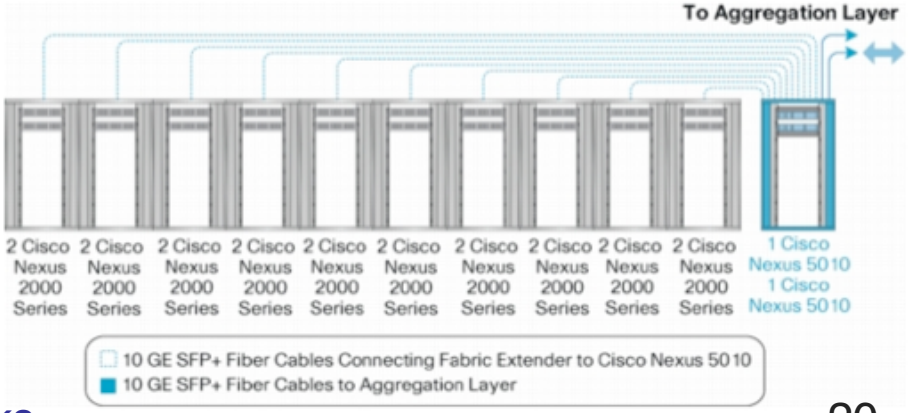
VMM Virtual Switches



Top-of-Rack Architecture

- Rack of servers
  - Commodity servers
  - And top-of-rack switch
- Modular design
  - Preconfigured racks
  - Power, network, and storage cabling
- Aggregate to the next level





Extreme Modularity

- Containers

- Many containers

Data Center Challenges

- Traffic load balance
- Support for VM migration
- Achieving bisection bandwidth
- Power savings / Cooling
- Network management (provisioning)
- Security (dealing with multiple tenants)

Data Center Costs

- James Hamilton published basic 2008 breakdown

- Servers: 45%
  - CPU, memory, disk
- Infrastructure: 25%
  - UPS, cooling, power distribution
- Power draw: 15%
  - Electrical utility costs
- Network: 15%
  - Switches, links, transit

For Next Class…

- Read and review Vahdat's Fat Tree paper
- Submit project groups by tonight
  - Email me list of members & initial idea(s)
  - Make sure to CC all project members