CS 6323
Complex Networks and Systems:
Modeling and Inference

Lecture 1:
Overview

Prof. Gregory Provan
Department of Computer Science
University College Cork

Course Objectives

Applying Modeling Methodologies to Cloud Computing Algorithms and Architectures

- Understand notion of real-world cloud networks
  - How to define such a system?
  - What do you want to do with such a system?
- Study algorithms used for performing inference on these systems
- Study approaches used for modeling the hardware for cloud systems

Learning Outcomes

At the conclusion of this course, a participant will …

- know the *basic concepts* underlying cloud systems modeling and performance analysis
- know how to
  - Develop mathematical models for cloud-system networks
  - Design algorithms using the MapReduce framework
  - Design network topologies to achieve particular performance objectives
  - conduct and evaluate a performance analysis using Queuing models
  - conduct and evaluate a dependability analysis using Reliability Block Diagram (RBD) or Markov techniques
  - conduct and evaluate a performability analysis using various modeling techniques

Logistics

- Class time: Tues. 1-2; Thu. 12-1
- Location: WGB 1.06
- Instructor: Gregory Provan
  - Office: WGB 1-71
  - Tele: 420-5928
  - E-mail: g.provan@cs.ucc.ie
  - Office Hours:
    - Wednesday 11-1
    - By appointment

6323 Topics

- Introduction to Network Modeling
- Modeling tools (2)
- Cloud Computing (5)
  - Algorithms for Cloud Computing: MapReduce
- Performance Modeling (5)
  - Stochastic Models
  - Discrete-Event Models
- Reliability/Fault Modeling (2)
  - Model-Based Approaches
  - Reliability Models
- Integrated Modeling for Complex Systems (3)

Week	Date	Lecture	Section	Topic
1	13 January 2015	1	**Introduction**	Course Objectives
	15 January 2015	2	**Mathematical Foundations**	Topological Analysis
2	20 January 2015	3		Review: Graph Theory, Discrete Probability Theory
	22 January 2015	4	**Distributed Network Inference: Cloud Computing**	Cloud Computing
3	27 January 2015	5		MapReduce Framework
	29 January 2015	6		Distributed Graph Algorithms I
4	03 February 2015	7		Distributed Graph Algorithms II
	05 February 2015	8		Other Distributed Algorithms
5	10 February 2015	6	**Network Performance Analysis**	Probability Review: Continuous
	12 February 2015	7		Queueing Theory: M/M/1
6	17 February 2015	8		Queueing Networks
	19 February 2015	9		Priority Queues
7	24 February 2015		**Exam Week**	
	26 February 2015			
8	03 March 2015	10	**Network Performance Analysis (cont.)**	Performance Analysis Problem-Solving
	05 March 2015	11	**Network Reliability**	Reliabilty Block Diagrams
9	10 March 2015	12		Markov Models
	12 March 2015	13	**Performance and Reliability Modeling**	Integrated Modeling
10	17 March 2015	14		Performance Networks
	19 March 2015	15		
11	24 March 2015	16		Integrated Modeling Problem-Solving
	26 March 2015	17		
12	31 March 2015	18		Course Review
	02 April 2015	19		Exam Problem-Solving
	TBD		**END-OF-TERM EXAM**	

Course Evaluation

- Grading
  - Midterm Exam (3rd Nov.) 40%
  - Final exam 50%
  - Continuous Assessment 10%
- Mid-Term Exam
  - Covers the main principles introduced in class up to that point
- Final Exam
  - Covers the main principles introduced in class

Topic 1: Cloud Computing

- What is the cloud computing paradigm
- How to characterise its behaviour
- Cloud computing software
  - MapReduce

Cloud Computing

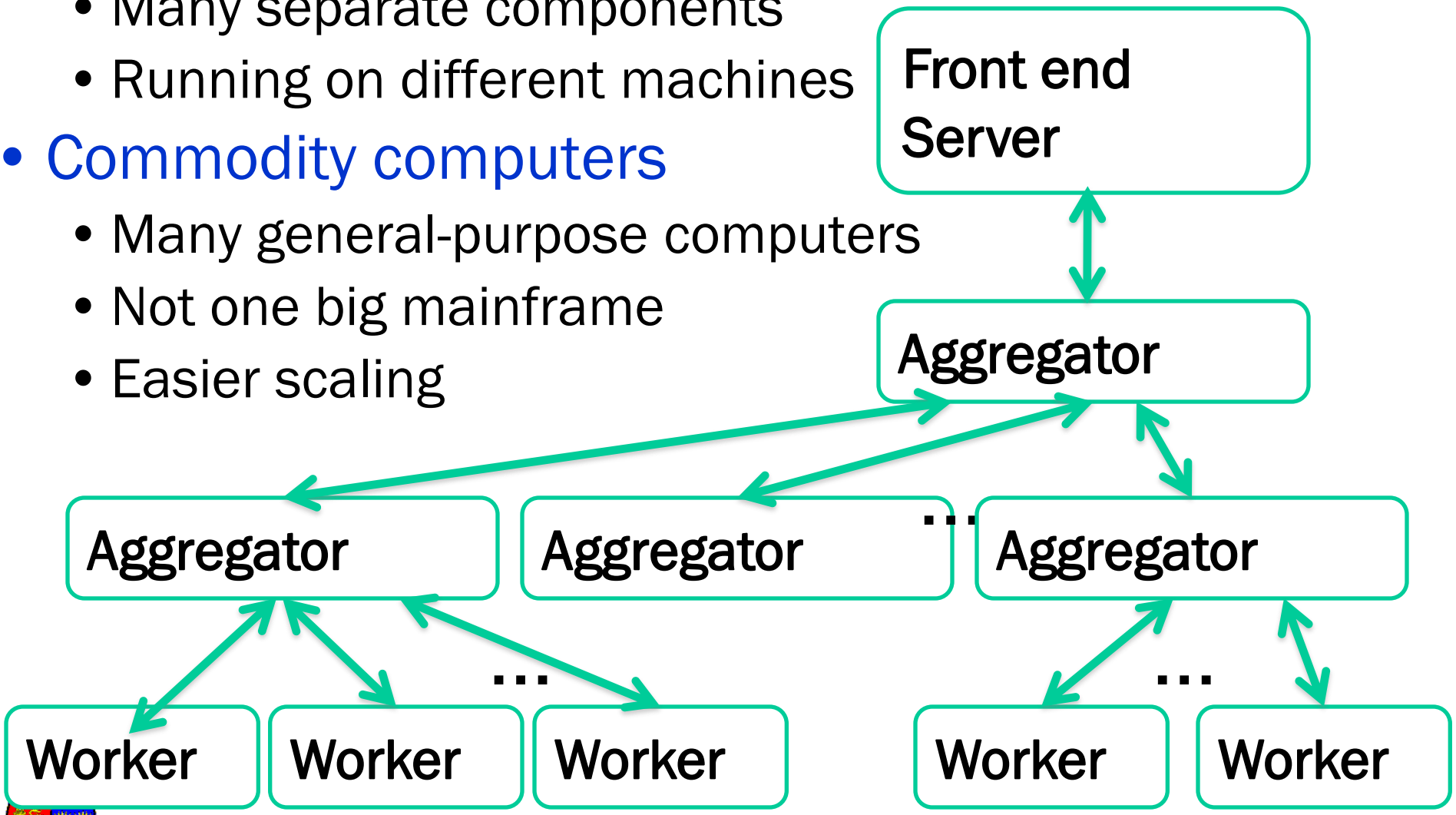
- Elastic resources
  - Expand and contract resources
  - Pay-per-use
  - Infrastructure on demand
- Multi-tenancy
  - Multiple independent users
  - Security and resource isolation
  - Amortize the cost of the (shared) infrastructure
- Flexibility service management
  - Resiliency: isolate failure of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

Cloud Service Models

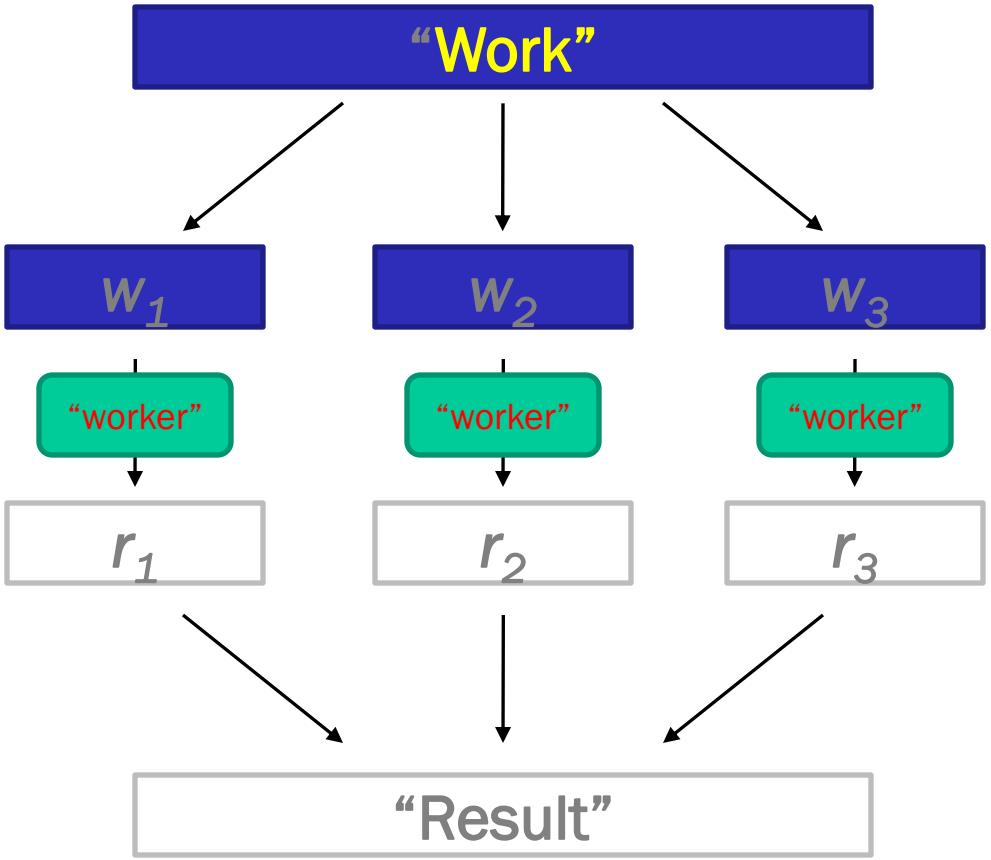
- Software as a Service
  - Provider licenses applications to users as a service
  - E.g., customer relationship management, e-mail, ..
  - Avoid costs of installation, maintenance, patches, ...
- Platform as a Service
  - Provider offers software platform for building applications
  - E.g., Google's App-Engine
  - Avoid worrying about scalability of platform
- Infrastructure as a Service
  - 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



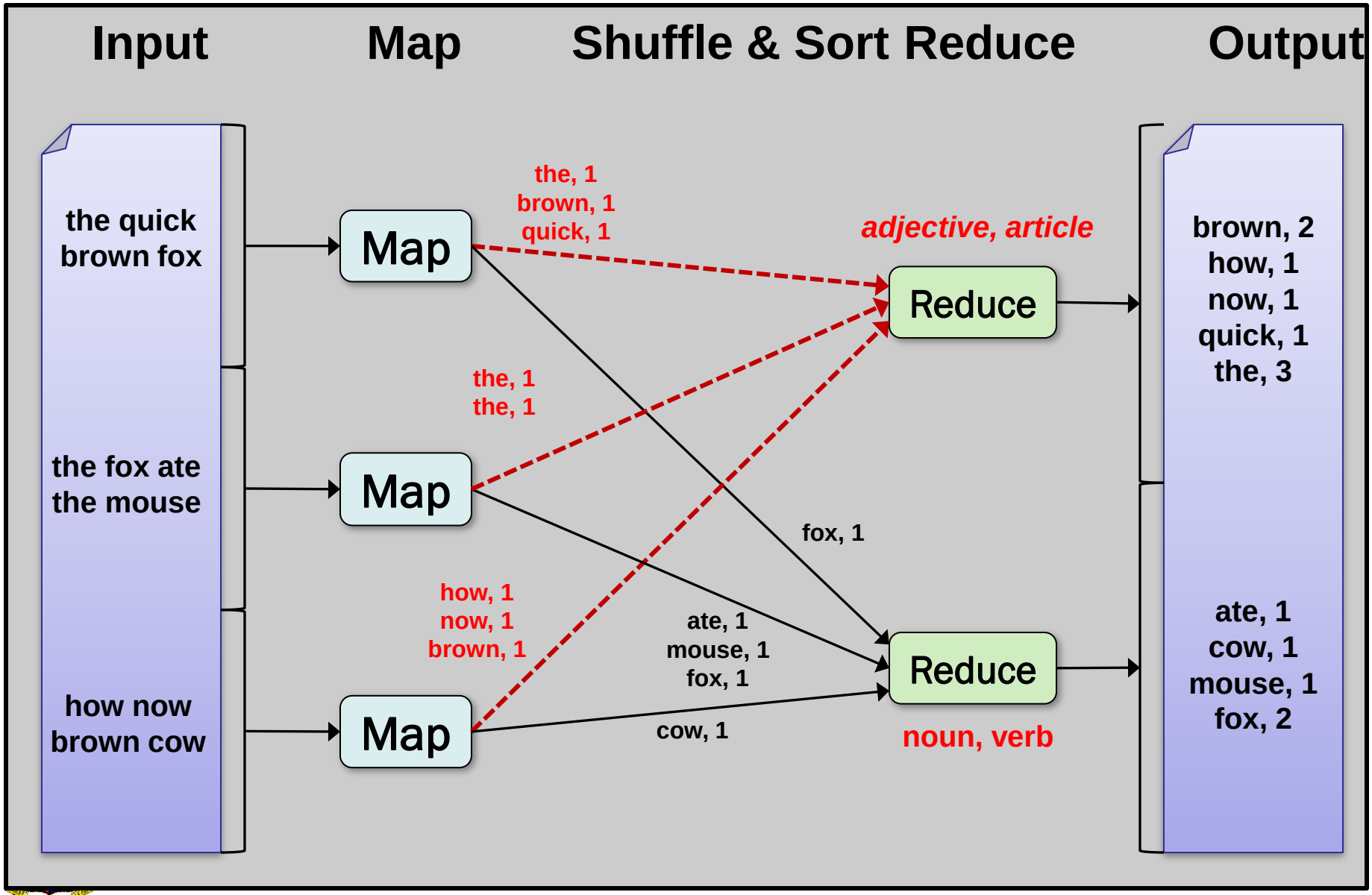
Cloud Algorithm: MapReduce



Partition

Combine

Word Count Example



Parallelization Challenges

- How do we assign work units to workers?
- What if we have more work units than workers?
- What if workers need to share partial results?
- How do we aggregate partial results?
- How do we know all the workers have finished?
- What if workers die?

What is the common theme of all of these problems?

Topic 2: Network Performance

- Goal: analyse network performance (QoS metrics)
  - Throughput
  - Response time
    - Time for an internet query to be processed due to network traffic
- Outcome
  - Can design networks to achieve target QoS
    - Cloud, Internet, mobile phone network

Realizing the 'Computer Utilities' Vision: What Consumers and Providers Want?

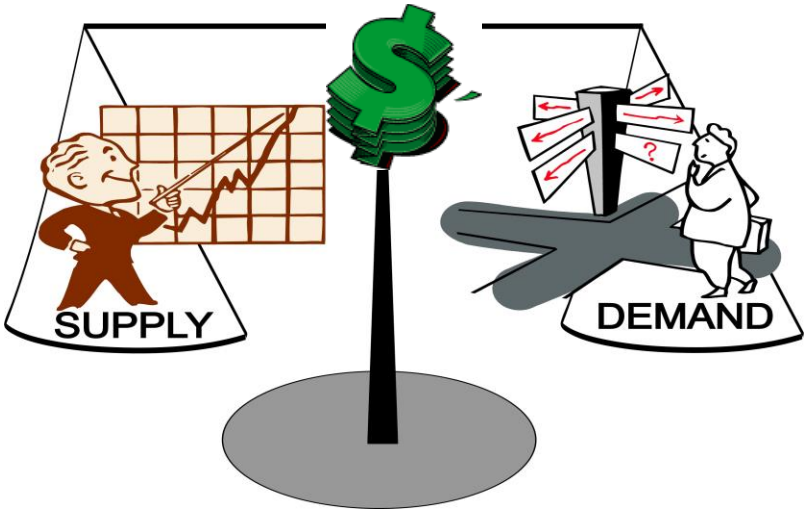
- Consumers – minimize expenses, meet QoS
  - How do I express QoS requirements to meet my goals?
  - How do I assign valuation to my applications?
  - How do I discover services and map applications to meet QoS needs?
  - How do I manage multiple providers and get my work done?
  - How do I outperform other competing consumers?
  - …
- Providers – maximise Return On Investment (ROI)
  - How do I decide service pricing models?
  - How do I specify prices?
  - How do I translate prices into resource allocations?
  - How do I assign and enforce resource allocations?
  - How do I advertise and attract consumers?
  - How do I perform accounting and handle payments?
  - …

- Mechanisms, tools, and technologies
  - value expression, translation, and enforcement

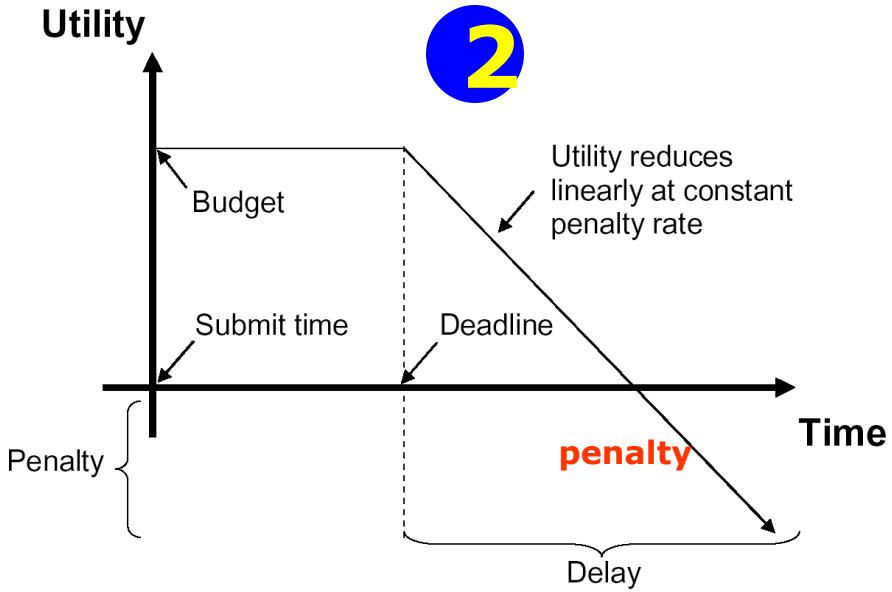
Market-based Systems = Self-managed and self-regulated systems.

- Manage
  - Complexity
  - Supply and Demand
- Enhance Utility

1

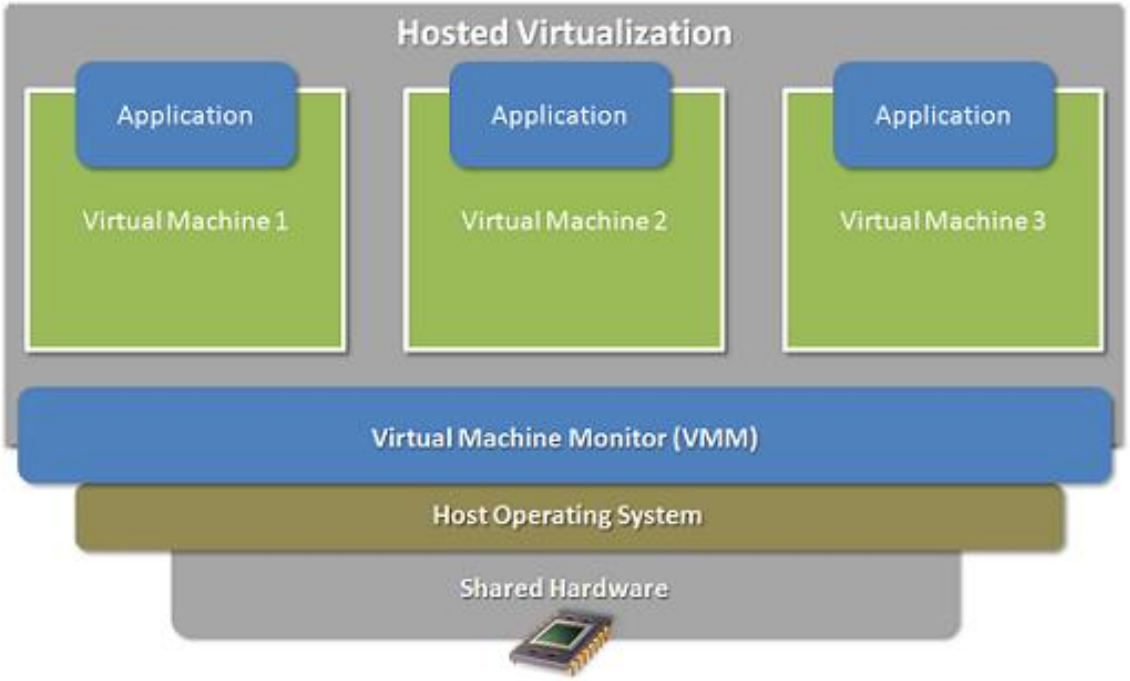


2



3

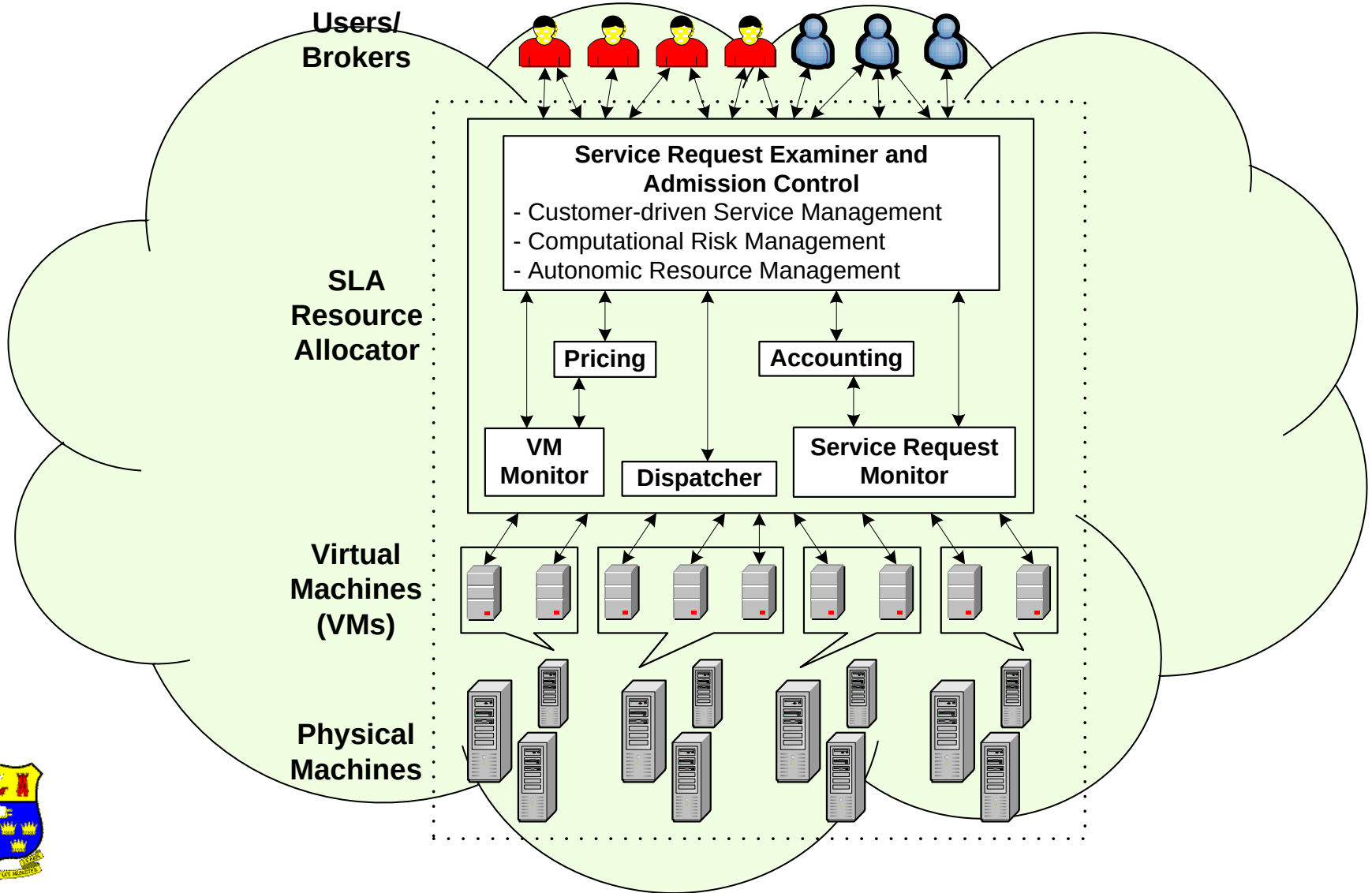
Enabling Technology: Virtualization



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

Market-oriented Cloud Architecture:

QoS negotiation and SLA-based Resource Allocation



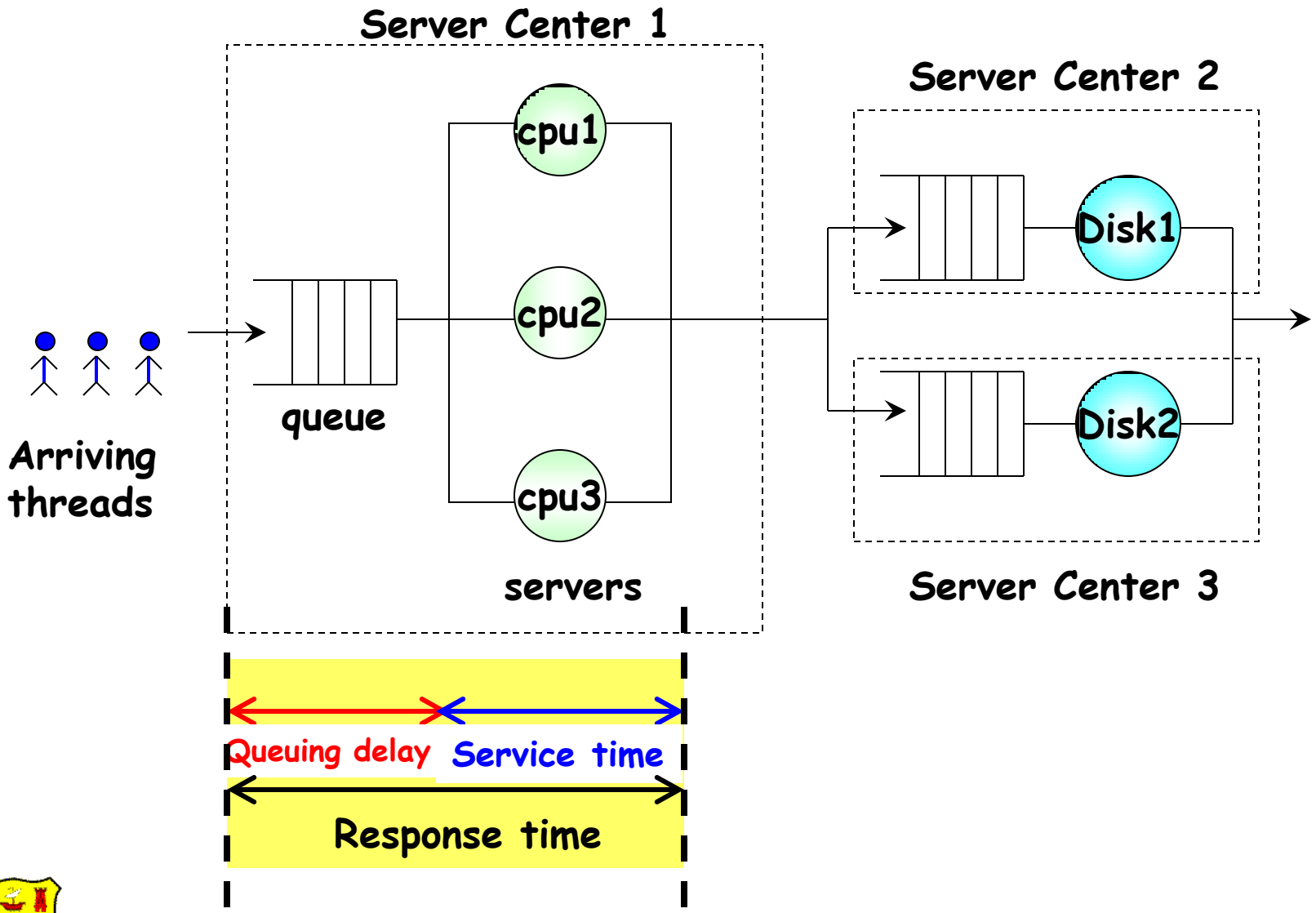
How to Characterise Networks?

- Issues
  - *Stochastic behaviour:* randomness in traffic, faults, etc.
  - *Complexity*: measure of network "density"
  - *Performance*: what is target QoS?
  - *Dependability*: how reliable is the network functionality
- Issues are in constant tension
- Examine how to define good tradeoffs

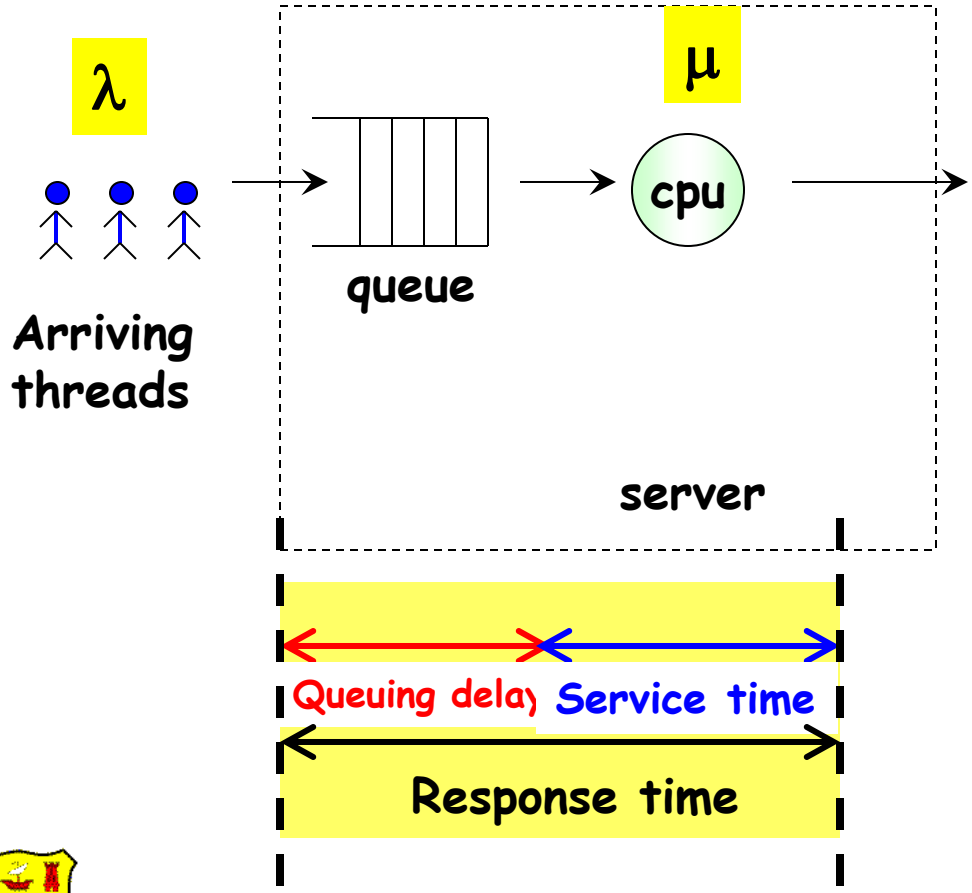
Integrating "Performance" in Analysis of Complex Networks

- Two aspects to performance modeling
  - Optimal performance
  - Performance under degraded conditions
- Study issues separately and integrate them
  - Performance
  - System failure

Computer-System Example



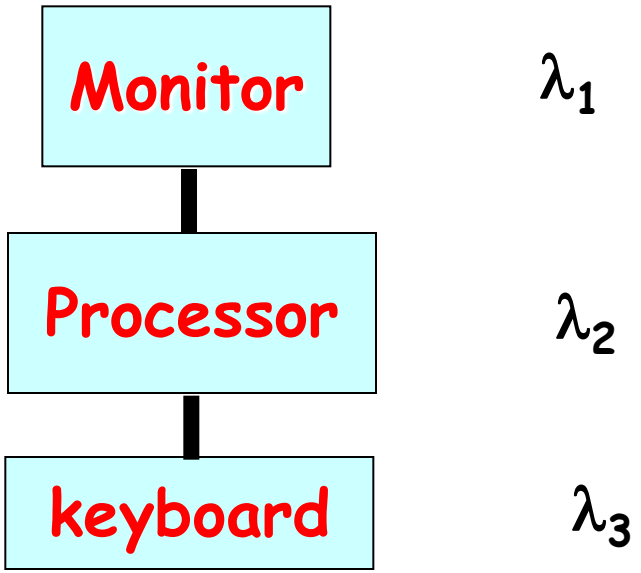
Network Performance Analysis



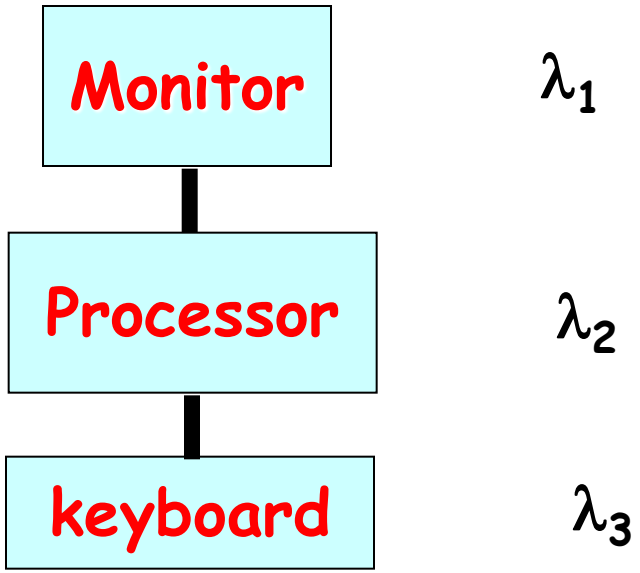
- Mathematical tool: queueing theory
  - Probabilistic analysis
- Arrival rate: λ
- Service rate: μ
- Response time: $\frac{1}{(\mu - \lambda)}$

Topic 3: Network Failure Analysis

- What is the probability that this system functions normally?
  - Define failure rates for individual components



Composite System Reliability



$$R_{system} = R_{monitor} * R_{processor} * R_{keyboard}$$

$$= e^{-(\lambda 1+\lambda 2+\lambda 3)t}$$

Topic 4: Integrated Performance/Failure Analysis

- Compute network performance in the presence of faults
- Integrate 2 models
  - Performance model
  - Failure-rate model

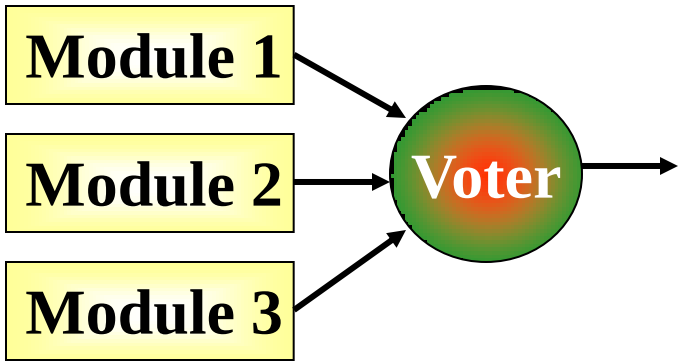
Performability =

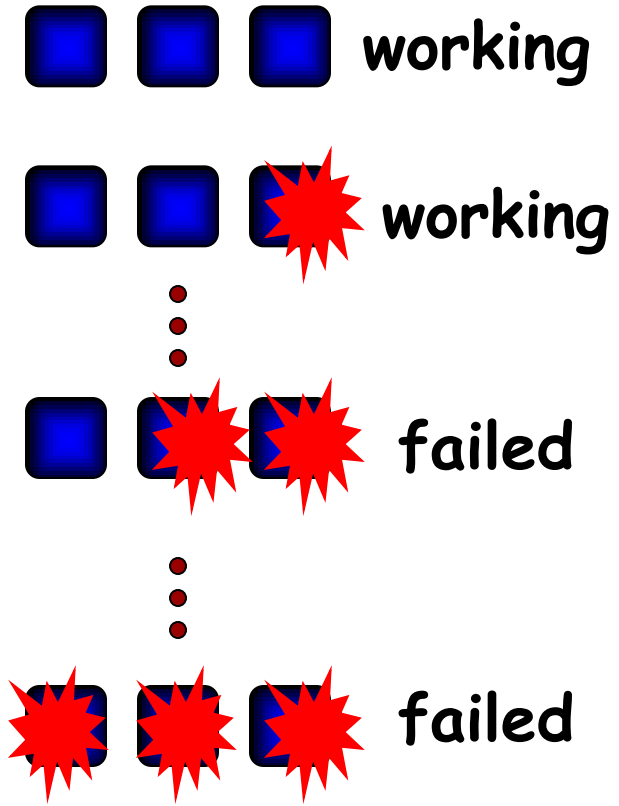
Performance + Dependability

The needs of High Performance, Fault Tolerant Computing

Integrated Model: Example

- Derive the Performability as a function of time
- The performance metric is considered only “working” or “failed”. (Reliability)



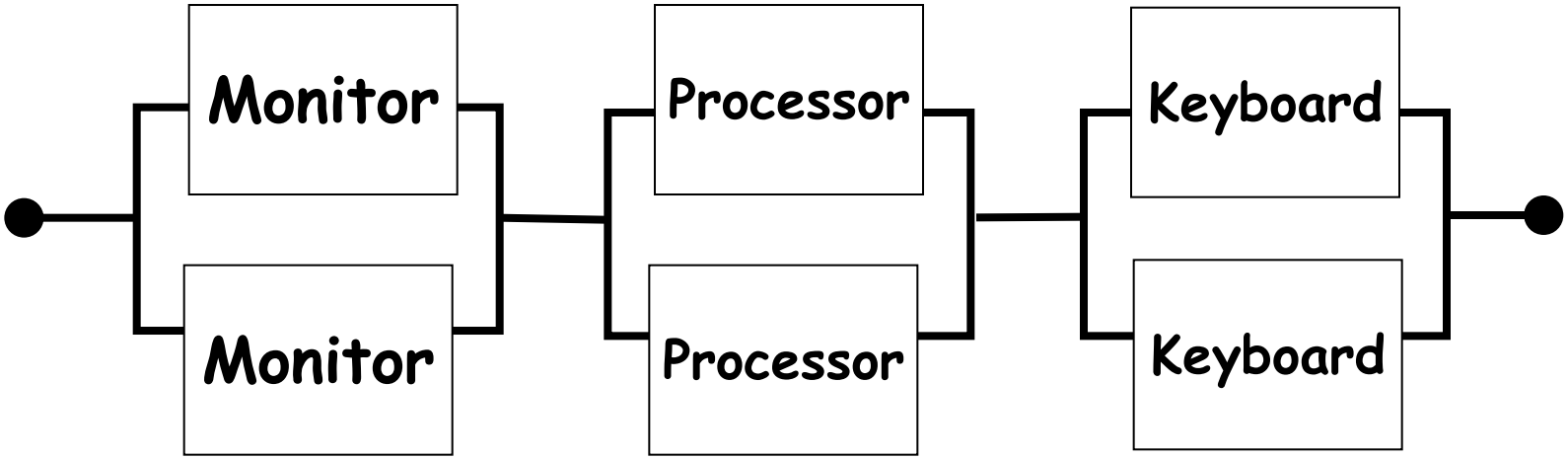


Topic 5: Designs to Tolerate Faults—Fault-Tolerant Computing

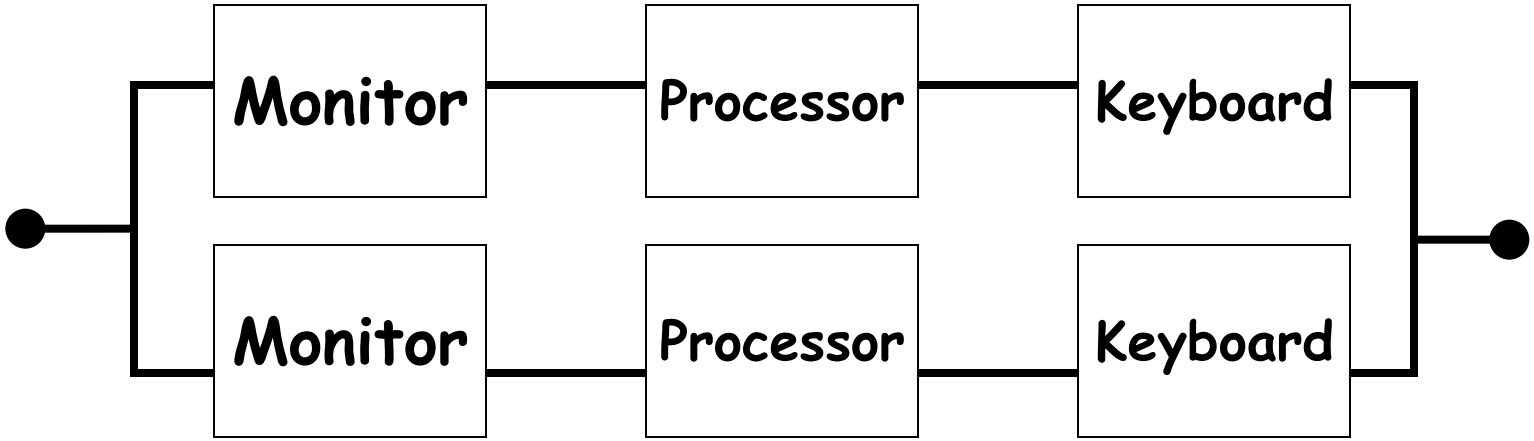
- Fault-tolerant computing is a generic term describing redundant design techniques with duplicate components or repeated computations enabling uninterrupted (tolerant) operation in response to component failure (faults).

Comparing System Topologies

Assuming Buses are perfect



Which topology has better reliability?



Course Pre-Requisites

- Pre-requisites
  - Basic mathematics background necessary
    - Discrete mathematics
  - Use of simple simulation tools
- Mathematical tools used
  - Graph theory
  - Probability theory
- We will review these tools

Text & References

Kishor S. Trivedi, *Probability and Statistics with Reliability, Queuing and Computer Science Applications*, second edition, John Wiley & Sons, Inc. 2002, ISBN 0-471-33341-7.

Introduction to Wireless and Mobile Systems, by Dharma Prakash Agrawal and Qing-An Zeng, ISBN No. 0534-40851-6.

References:

[1] Robin A. Sahner, Kishor S. Trivedi, Antonio Puliafito,
Performance and Reliability Analysis of Computer Systems –
An Example-Based Approach Using the SHARPE Software Package,
Kluwer Academic Publishers, 1996. ISBN 0-7923-9650-2.

[2]Martin L. Shooman, *Reliability of Computer Systems and Networks,Fault Tolerance, Analysis, and Design,* John Wiley & Sons, Inc., 2002. ISBN 0-471-29342-3.

[3] http://www.crhc.uiuc.edu/PERFORM/home.html

[4] http://www.eecs.umich.edu/~jfm/

[5] http://www.ee.duke.edu/~kst/

Links

http://www.crhc.uiuc.edu/PERFORM/home.html
http://www.eecs.umich.edu/~jfm/

http://www.dsn.org
http://www.rams.org