Computer Science 5204
Operating Systems
Fall, 2009

Dr. Dennis Kafura

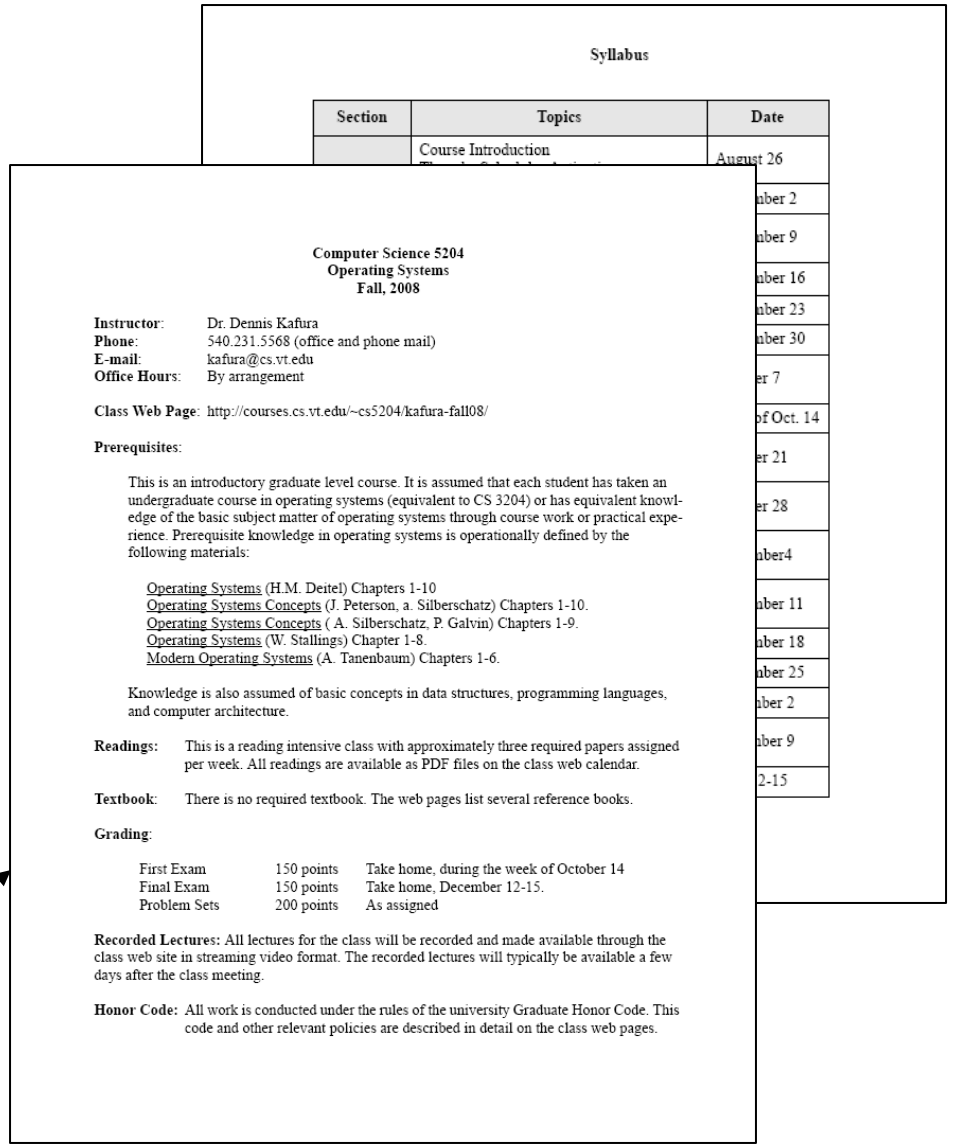
Course Overview

Virginia Tech

Organization

- Reading intensive
  - **35 +/- papers**
  - **No required text**
- Balance
  - **Theory vs. technology**
  - **Contemporary vs. classic**
  - **Survey vs. depth**
  - **Centralized vs. distributed**

Syllabus on web site



Syllabus

Section	Topics	Date
	Course Introduction	August 26
		ıber 2
		ıber 9
		ıber 16
		ıber 23
		ıber 30
		er 7
		of Oct. 14
		er 21
		er 28
		ıber4
		ıber 11
		ıber 18
		ıber 25
		ıber 2
		ıber 9
		2-15

Computer Science 5204
Operating Systems
Fall, 2008

Instructor: Dr. Dennis Kafura
Phone: 540.231.5568 (office and phone mail)
E-mail: kafura@cs.vt.edu
Office Hours: By arrangement

Class Web Page: http://courses.cs.vt.edu/~cs5204/kafura-fall08/

Prerequisites:

This is an introductory graduate level course. It is assumed that each student has taken an undergraduate course in operating systems (equivalent to CS 3204) or has equivalent knowledge of the basic subject matter of operating systems through course work or practical experience. Prerequisite knowledge in operating systems is operationally defined by the following materials:

Operating Systems (H.M. Deitel) Chapters 1-10
Operating Systems Concepts (J. Peterson, a. Silberschatz) Chapters 1-10.
Operating Systems Concepts (A. Silberschatz, P. Galvin) Chapters 1-9.
Operating Systems (W. Stallings) Chapter 1-8.
Modern Operating Systems (A. Tanenbaum) Chapters 1-6.

Knowledge is also assumed of basic concepts in data structures, programming languages, and computer architecture.

Readings: This is a reading intensive class with approximately three required papers assigned per week. All readings are available as PDF files on the class web calendar.

Textbook: There is no required textbook. The web pages list several reference books.

Grading:

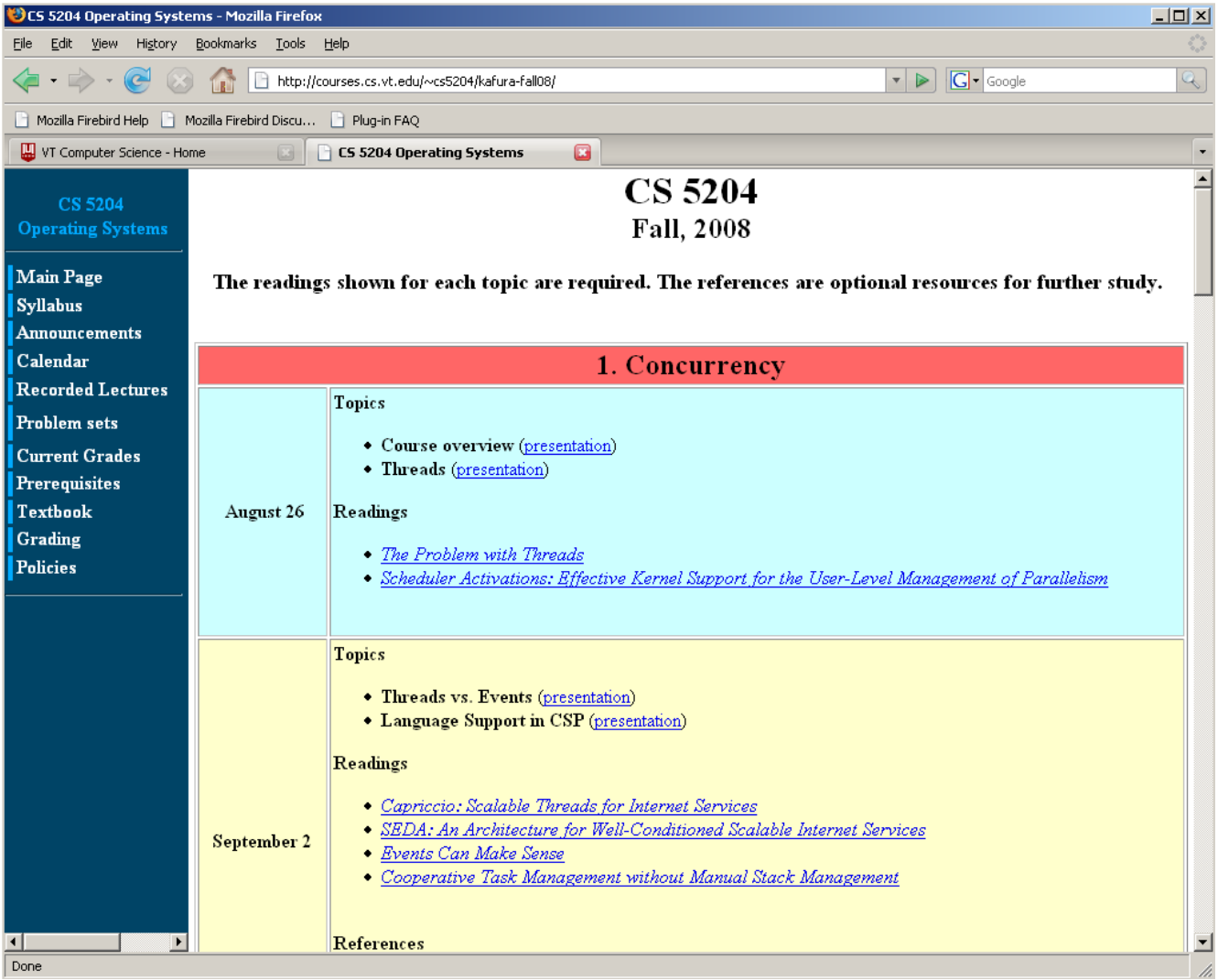
First Exam	150 points	Take home, during the week of October 14
Final Exam	150 points	Take home, December 12-15.
Problem Sets	200 points	As assigned

Recorded Lectures: All lectures for the class will be recorded and made available through the class web site in streaming video format. The recorded lectures will typically be available a few days after the class meeting.

Honor Code: All work is conducted under the rules of the university Graduate Honor Code. This code and other relevant policies are described in detail on the class web pages.

Course Web site

http://courses.cs.vt.edu/~cs5204/fall09-kafura



Major Topics

1. Concurrency
2. Security
3. Fault Tolerance
4. Virtualization & File Systems

1. Concurrency/Communication

How can concurrent processing be structured on a single processor?

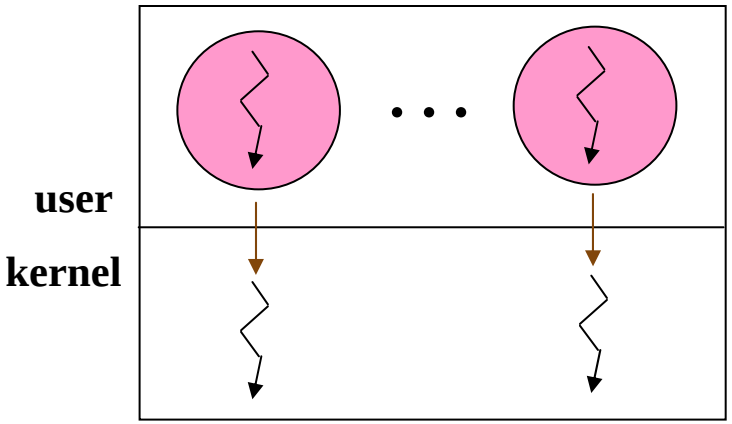
What are alternative syntaxes and semantics for interactions among concurrent entities?

How can concurrency and communication be represented formally?

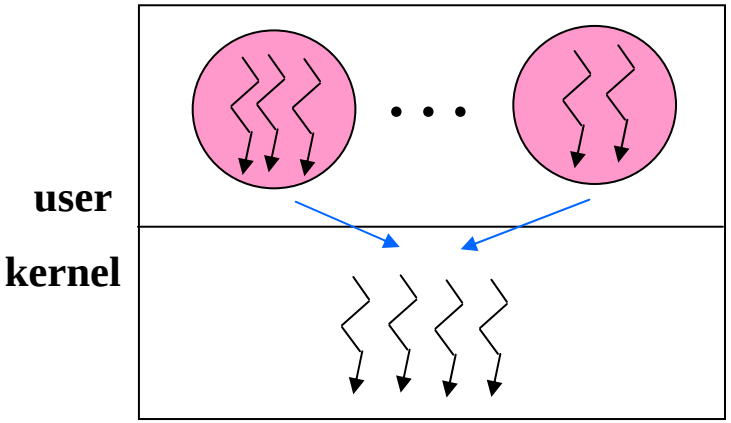
How can transaction-style semantics be supported locally in hardware or software?

How can transaction semantics be supported in a distributed system?

Structuring Activities

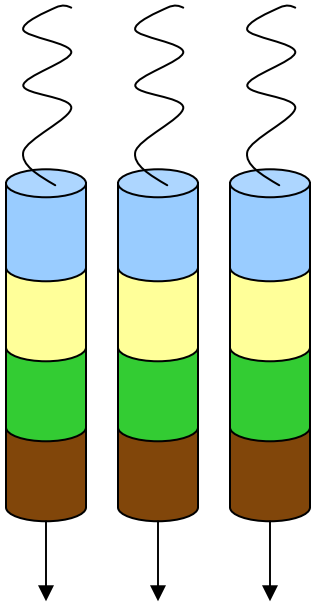


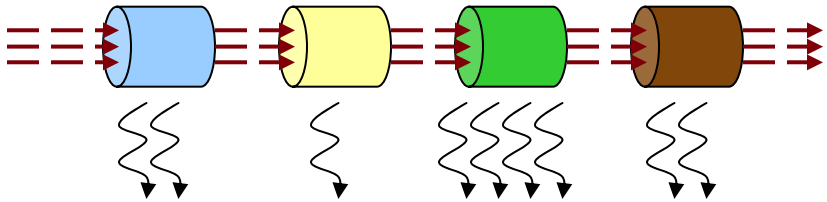
process-centered



thread-centered

Threads vs. Events

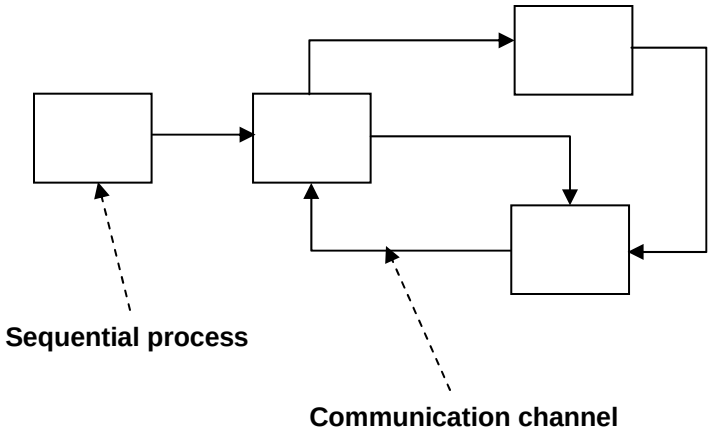




- Threads
  - Each thread executes all stages of the computation
  - Communication between stages via run-time stack
- Events
  - Each thread bound to one stage of the computation
  - Communication between stages via events

Language models and syntax

Communicating Sequential Processes

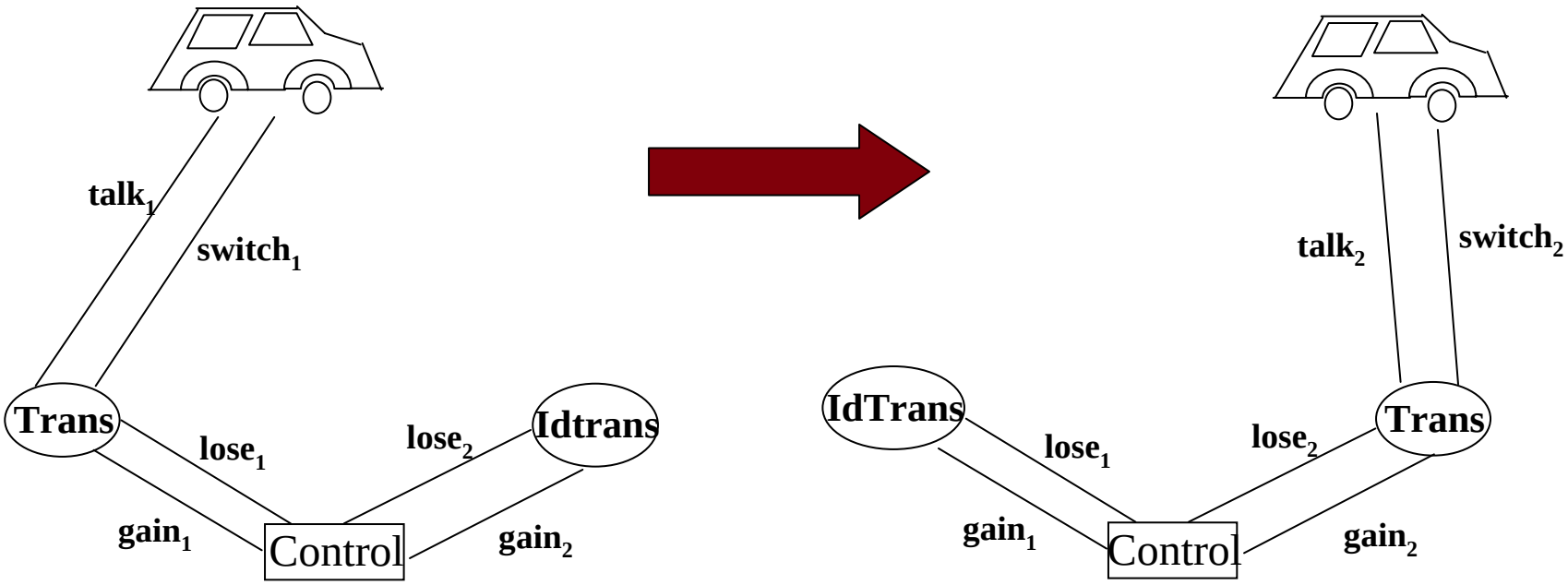


Chords in C#

```
public class Buffer {
                    public string Get() & public async Put (string s) { return s; }
                }
```

π-Calculus

An algebra that captures the notions of communication, interaction, and synchronization among concurrently executing entities.

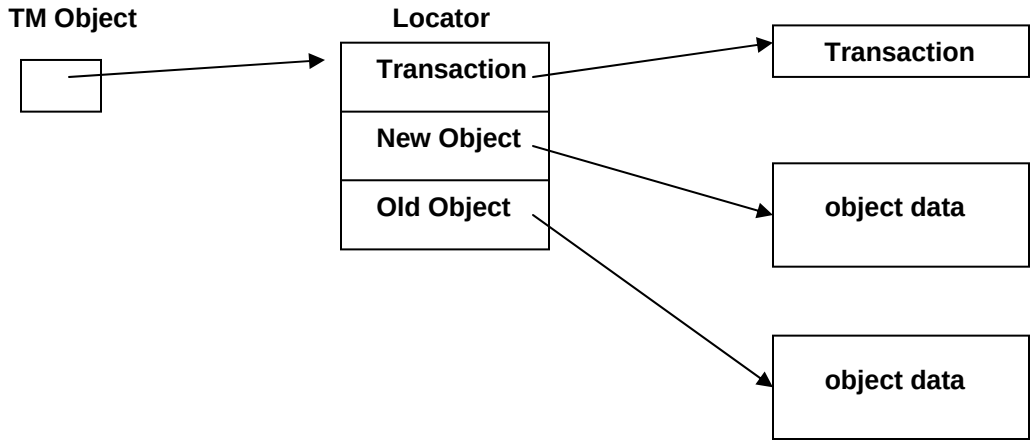


Supporting Transaction Semantics

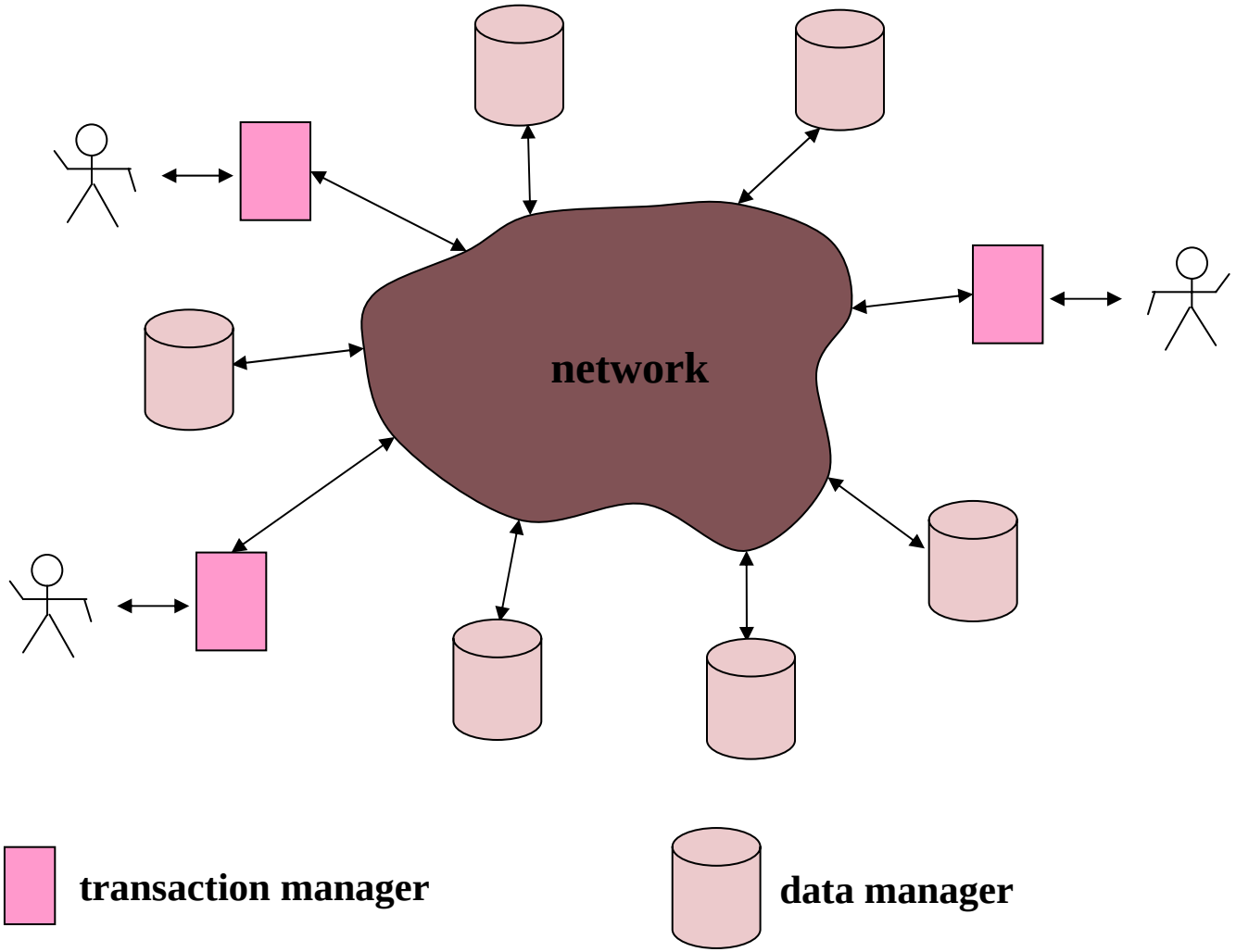
```
repeat {

   BeginTransaction();     /* initialize transaction */
   <read input values>
   success = Validate();   /* test if inputs consistent */
   if (success) {
      <generate updates>
      success = Commit(); /* attempt permanent update */
      if (!success)
         Abort();/* terminate if unable to commit */
   }
   EndTransaction();       /* close transaction */

} until (success);
```



Transaction Model



2. Security

How can rights for access control be structured for effective use and management?

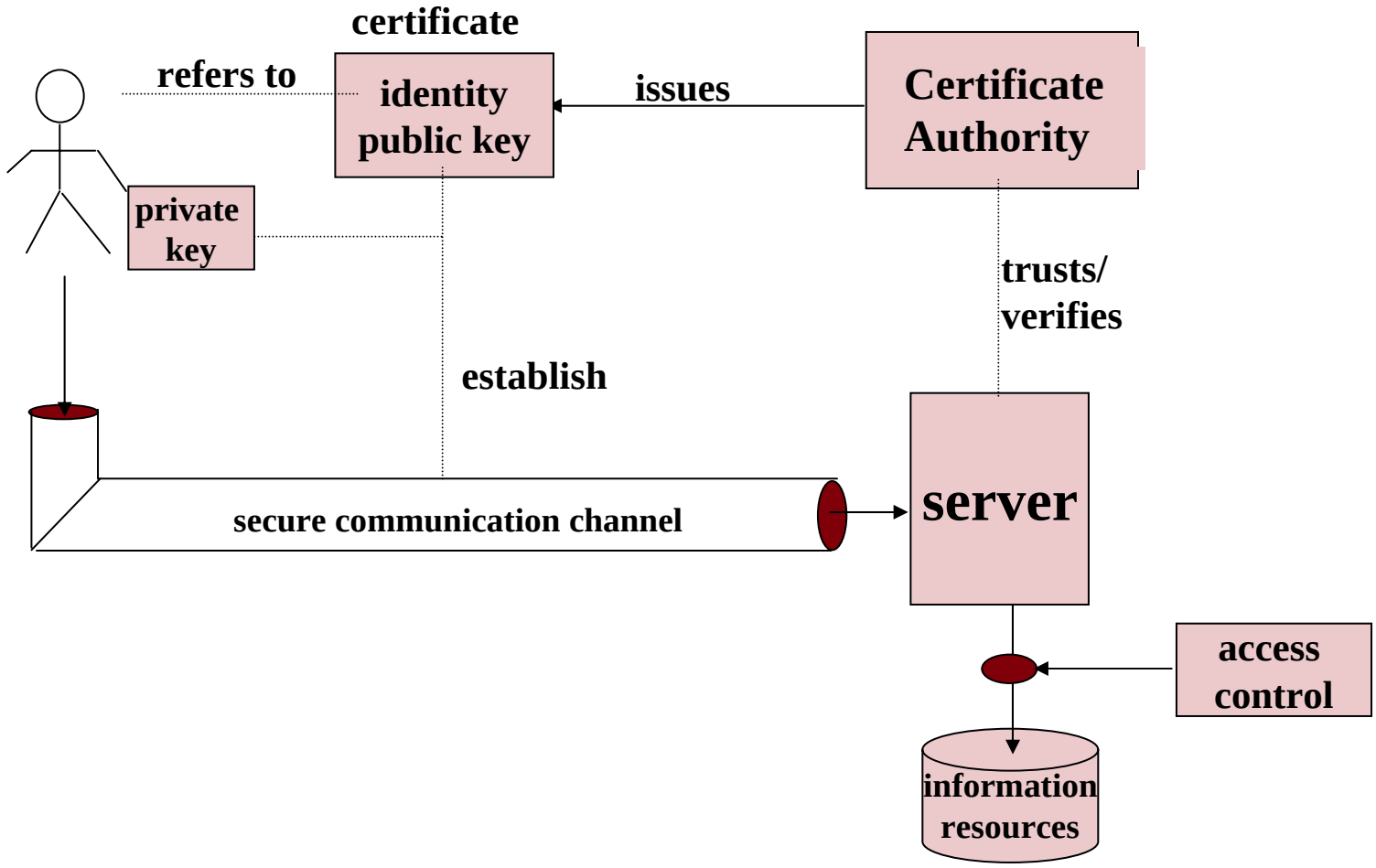
How can a digital document be "signed" so as to identify authorship?

How can communicating parties be confident of each other's identities?

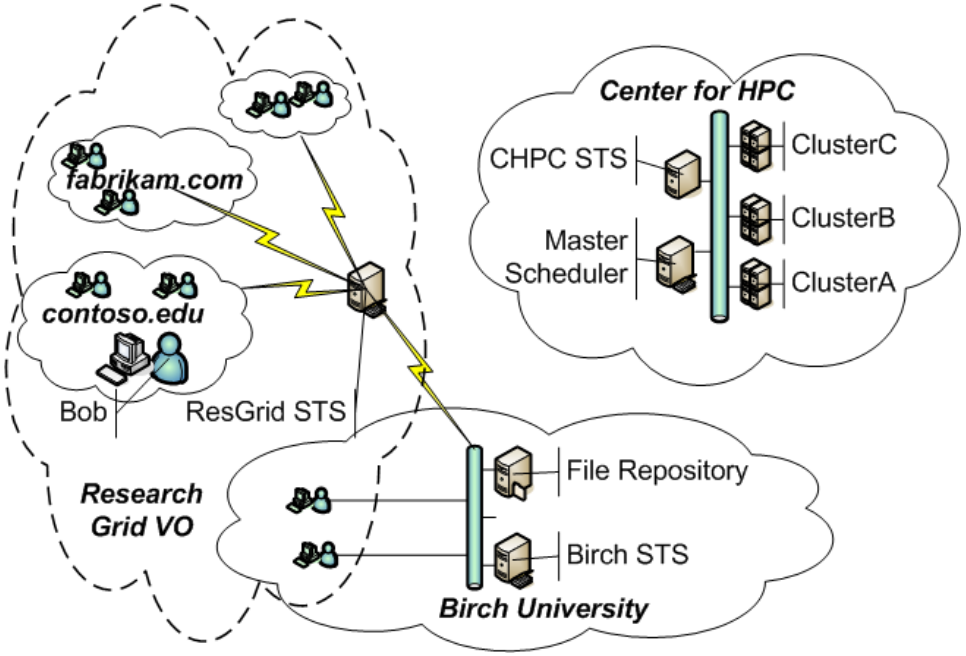
How can distributed systems authenticate clients and servers to each other?

How can access policies be expressed and enforced?

Security Overview



Security in distributed systems



- Describe explicit trust relationships
- Express security token issuance policies
- Provide security tokens that contain identities, capabilities, and/or delegation policies
- Express resource authorization and delegation policies

3. Fault Tolerance

How can events be ordered in a distributed system lacking a shared clock?

How can this ordering give rise to a form of virtual time?

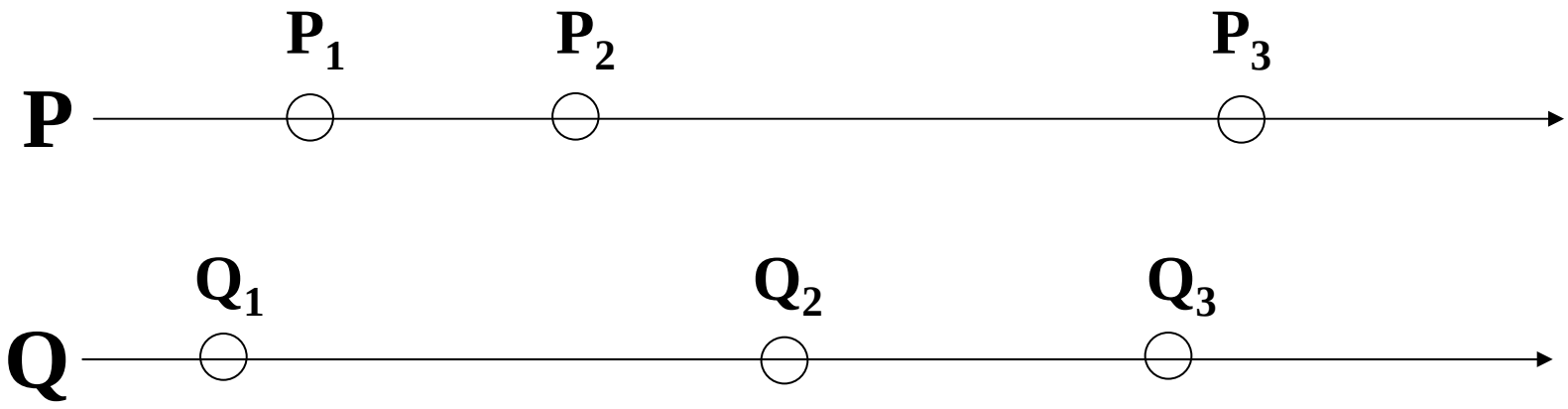
What are basic approaches to recovery from failure?

What is the taxonomy of strategies of "backward" recovery?

How can the state of system be captured so that it can be recovered in the event of failure?

How can distributed elements agree on commit to accepting a change in the system state?

Event Ordering

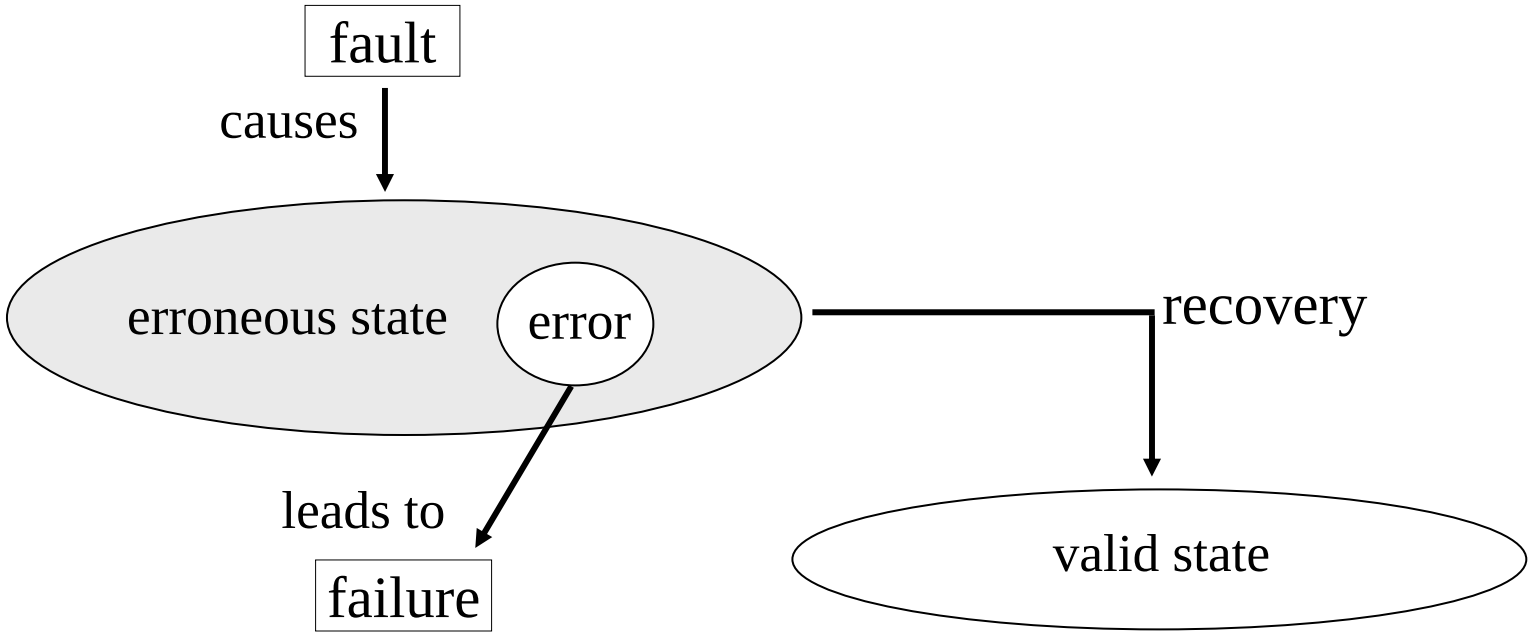


How can the events on P be related to the events on Q?

Which events of P "happened before" which events of Q?

When does it matter how we answer these questions?

Recovery



An error is a manifestation of a fault that can lead to a failure.

Failure Recovery:

- backward recovery
  - operation-based (do-undo-redo logs)
  - state-based (checkpoints)
- forward recovery

4. Virtualization and File Systems

What are the principles of virtualization?

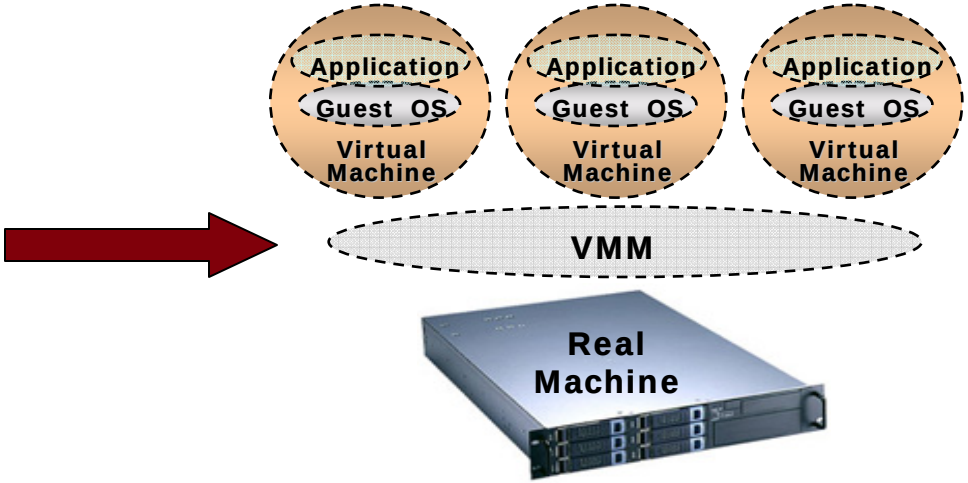
Why is virtualization difficult to achieve on modern architectures?

What strategies are there for building virtual machine monitors?

How can file systems be structured to handle tera-bytes of information?

Virtualization

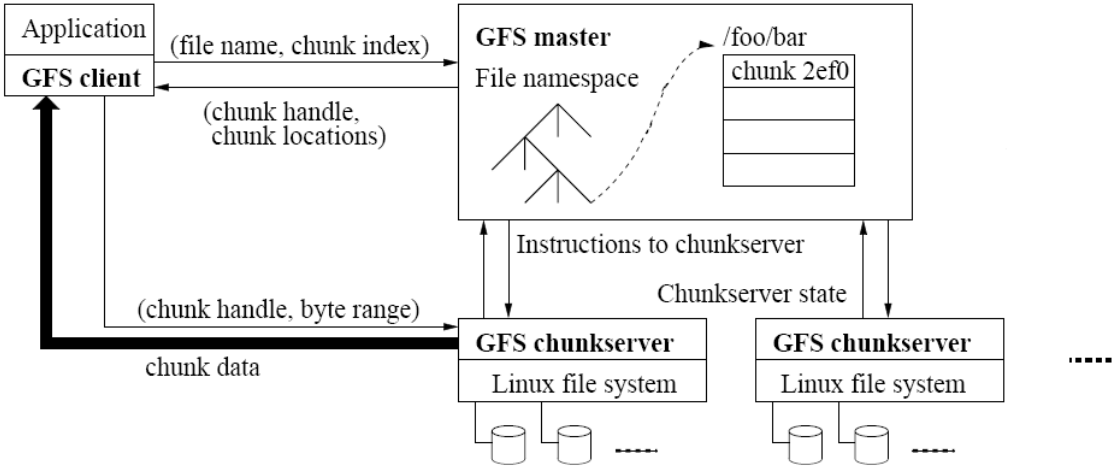
Circa 1970s

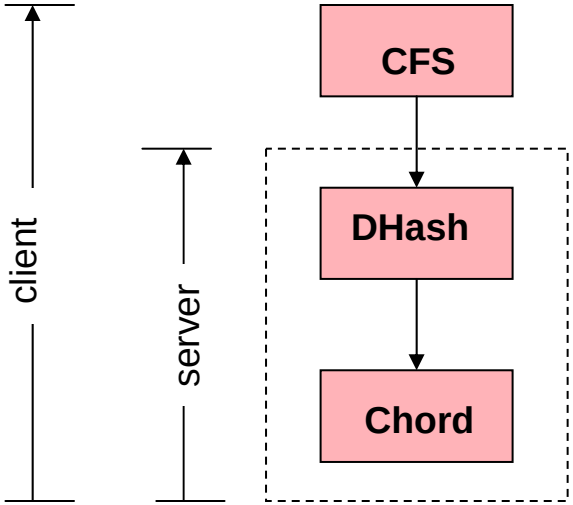


today

File Systems

Google File System





A peer-to-peer file system structure