# Advanced Operating Systems Lecture notes

Clifford Neuman, Katia Obraczka
University of Southern California
Information Sciences Institute

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# CSci555: Advanced Operating Systems

## Lecture 1 - September 3, 1999

Dr. Clifford Neuman
University of Southern California
Information Sciences Institute

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Some things an operating system does

- Memory Management
- Scheduling / Resource management
- Communication
- Protection and Security
- File Management - I/O
- Naming
- Synchronization
- User Interface

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Progression of Operating Systems

Primary goal of a distributed system:

- Sharing

Progression over past years

- Dedicated machines
- Batch Processing
- Time Sharing
- Workstations and PC's
- Distributed Systems

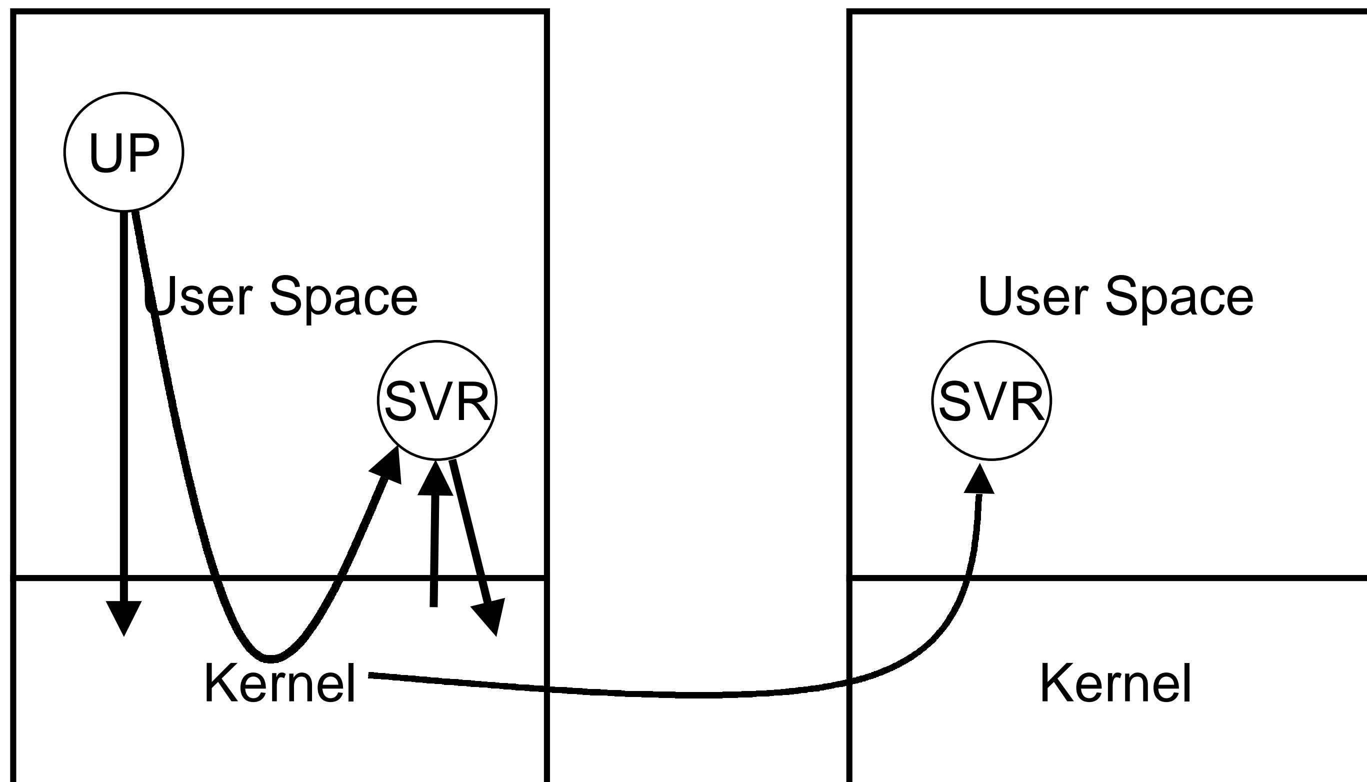
*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Structure of Distributed Systems

- Kernel
  - Basic functionality and protection
- Application Level
  - What does the real work
- Servers
  - Service and support functions needed by applications
  - Many functions that used to be in Kernel are now in servers.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Structure of Distributed Systems



# Characteristics of a Distributed System

- Basic characteristics:
  - Multiple Computers
  - Interconnections
  - Shared State

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Why Distributed Systems are Hard

- Scale:
  - Numeric
  - Geographic
  - Administrative
- Loss of control over parts of the system
- Unreliability of Messages
- Parts of the system down or inaccessible
  - Lamport: **You know you have a distributed system when the crash of a computer you have never heard of stops you from getting any work done.**

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# CSci555: Advanced Operating Systems
## Lecture 2 - September 10, 1999

Dr. Katia Obraczka
University of Southern California
Information Sciences Institute

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Administration 1

- Diagnostic exams:
  - Everyone should have gotten their grades and our recommendation by e-mail.
  - Talk to us if you haven’t heard anything.
- Academic Integrity Policy:
  - NOT A JOKE!
  - We’ll strictly enforce it.
  - Read it and talk to us if you have questions.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Administration 2

- Reading report #1 assigned.
  - Get it from the class Web page.
  - Look at guidelines for writing and submitting report.
- Office:
  - SAL 236, phone (213) 740-4777.
  - Information on Web page updated.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Last Class

- How to read a technical paper.
- What is a distributed system.
- Motivation, applications, some history.
- More: textbook chapters 1 and 2.
  - Including a chart with historic facts for more details on distributed systems history (pages 23 and 24).

# Outline

- E2E Argument [Saltzer et al.]
- Communication Models.
  - Message passing.
  - Distributed shared memory (DSM).
  - Remote procedure call (RPC) [Birrel et al.]
    - Light-weight RPC [Bershad et al.]
  - DSM case studies
    - IVY [Li et al.]
    - Linda [Carriero et al.]

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# End-to-End Argument 1

- QUESTION: Where to place distributed systems functions?
- Layered system design:
  - Different levels of abstraction for simplicity.
  - Lower layer provides service to upper layer.
  - Very well defined interfaces.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# E2E Argument 2

- E2E argument paper argues that functions should be moved closer to the application that uses them.
- Rationale:
  - Some functions can only be completely and correctly implemented with app's knowledge.
    - Example: Reliable message delivery.
  - App's that don't need certain functions should not have to pay for them.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# E2E Argument 3

- Counter argument:
  - Performance.
  - Example: File transfer
    - Reliability checks at lower layers detect problms earlier.
    - Abort transfer and re-try without having to wait till whole file is transmitted.
- Bottom line: “spread out” functionality across layers.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# E2E Argument 4

- Another perspective:
  - Why pay for something you don't need.
    - Example 1: the Internet.
    - Example 2: trend in kernel design - take away from kernel as much functionality as possible.
- We'll revisit the e2e argument throughout the course.
- Paper has more examples.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Communication Models

- Support for processes to communicate among themselves.
- Traditional (centralized) OS's:
  - Provide local (within single machine) communication support.
  - Distributed OS's: must provide support for communication across machine boundaries.
    - Over LAN or WAN.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

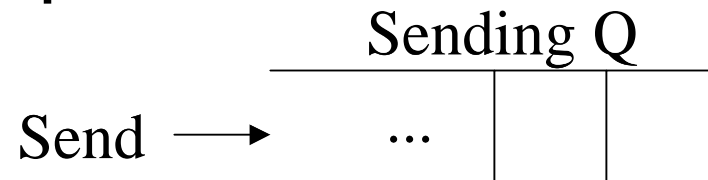
# Communication Paradigms

- 2 paradigms
  - Message Passing (MP)
  - Distributed Shared Memory (DSM)
- Message Passing
  - Processes communicate by sending messages.
- Distributed Shared Memory
  - Communication through a “virtual shared memory”.

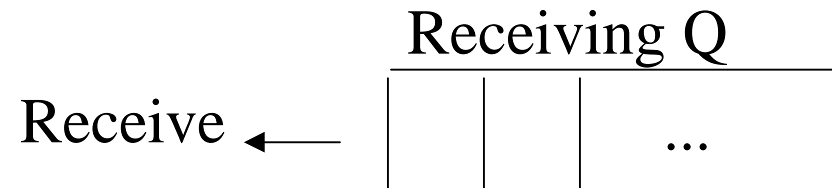
Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Message Passing

- Basic communication primitives:
  - Send message.




  - Receive message.




- Modes of communication:
  - Synchronous versus asynchronous.
- Semantics:
  - Reliable versus unreliable.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Synchronous Communication

- Blocking send: blocks until message is transmitted (out of the send queue).
- Blocking receive: blocks until message arrives in the receive queue.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Asynchronous Communication

- Non-blocking send: sending process continues as soon message is queued.
- Blocking or non-blocking receive:
  - Blocking: timeout or threads.
  - Non-blocking: proceeds while waiting for message.
    - Message is queued upon arrival.
    - Process needs to poll or be interrupted.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Reliability 1

- Unreliable communication:
  - Aka,  best effort, ie, “send and hope for the best”.
  - No ACKs or retransmissions.
  - Application programmer must design their own reliability mechanism.
  - Example: User Datagram Protocol (UDP)
    - Applications using UDP either don’t need reliability or build their own (e.g., UNIX NFS and DNS (both UDP and TCP))

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Reliability 2

- Reliable communication:
  - Different degrees of reliability.
  - Processes have some guarantee that messages will be delivered.
  - Example: Transmission Control Protocol (TCP)
  - Reliability mechanisms:
    - Positive acknowledgments (ACKs).
    - Negative Acknowledgments (NACKs).
  - Possible to build reliability atop unreliable service (E2E argument).

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Distributed Shared Memory

- Motivated by development of shared-memory multiprocessors which do share memory.
- Abstraction used for sharing data among processes running on machines that do **not** share memory.
- Processes think they read from and write to a "virtual shared memory".

# DSM 2

- Primitives: read and write.
- OS ensures that all processes (may be in different machines) sees all updates.
  - Happens transparently to processes.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# DSM and MP

- DSM is an abstraction!
  - Gives programmers the flavor of a centralized memory system, which is a well-known programming environment.
  - No need to worry about communication and synchronization.
- But, it is implemented atop MP.
  - No real shared memory.
  - OS takes care of required communication.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Caching in DSM

- For performance, DSM caches data locally.
  - More efficient access (locality).
  - But, must keep caches consistent.
  - Caching of pages in case of page-based DSM.
- Issues:
  - Page size.
  - Consistency mechanism.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Approaches to DSM 1

- Hardware-based:
  - Multi-processor architectures with processor-memory modules connected by high-speed LAN (E.g., Stanford's DASH).
  - Specialized hardware to handle reads and writes and perform required consistency mechanisms.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Approaches to DSM 2

- Paged-based:
  - Example: IVY.
  - DSM implemented as region of processor's virtual memory; occupies same address space range for every participating process.
  - OS keeps DSM data consistency as part of page fault handling.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Approaches to DSM 3

- Library-based:
  - Or language-based.
  - Example: Linda.
  - Language or language extensions.
  - Compiler inserts appropriate library calls whenever processes access DSM items.
  - Library calls access local data and communicate when necessary.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Remote Procedure Call

- Builds on MP.
- Main idea: extend traditional (local) procedure call to perform transfer of control and data across network.
- Easy to use: analogous to local calls.
- But, procedure is executed by a different process, probably on a different machine.
- Fits very well with client-server model.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# RPC Mechanism

1. Invoke RPC.
2. Calling process suspends.
3. Parameters passed across network to target machine.
4. Procedure executed remotely.
5. When done, results passed back to caller.
6. Caller resumes execution.

Is this synchronous or asynchronous?

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# RPC Advantages

- Easy to use.
- Well-known mechanism.
- Abstract data type
  - Client-server model.
  - Server as collection of exported procedures on some shared resource.
  - Example: file server.
- Reliable.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# RPC Semantics 1

- Related to delivery guarantees.
- "Maybe call":
  - Clients cannot tell for sure whether remote procedure was executed or not due to message loss, server crash, etc.
  - Usually not acceptable.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# RPC Semantics 2

- "At-least-once" call
  - Remote procedure executed at least once, but maybe more than once.
  - Retransmissions but no duplicate filtering.
  - Idempotent operations OK; e.g., reading data that is read-only.

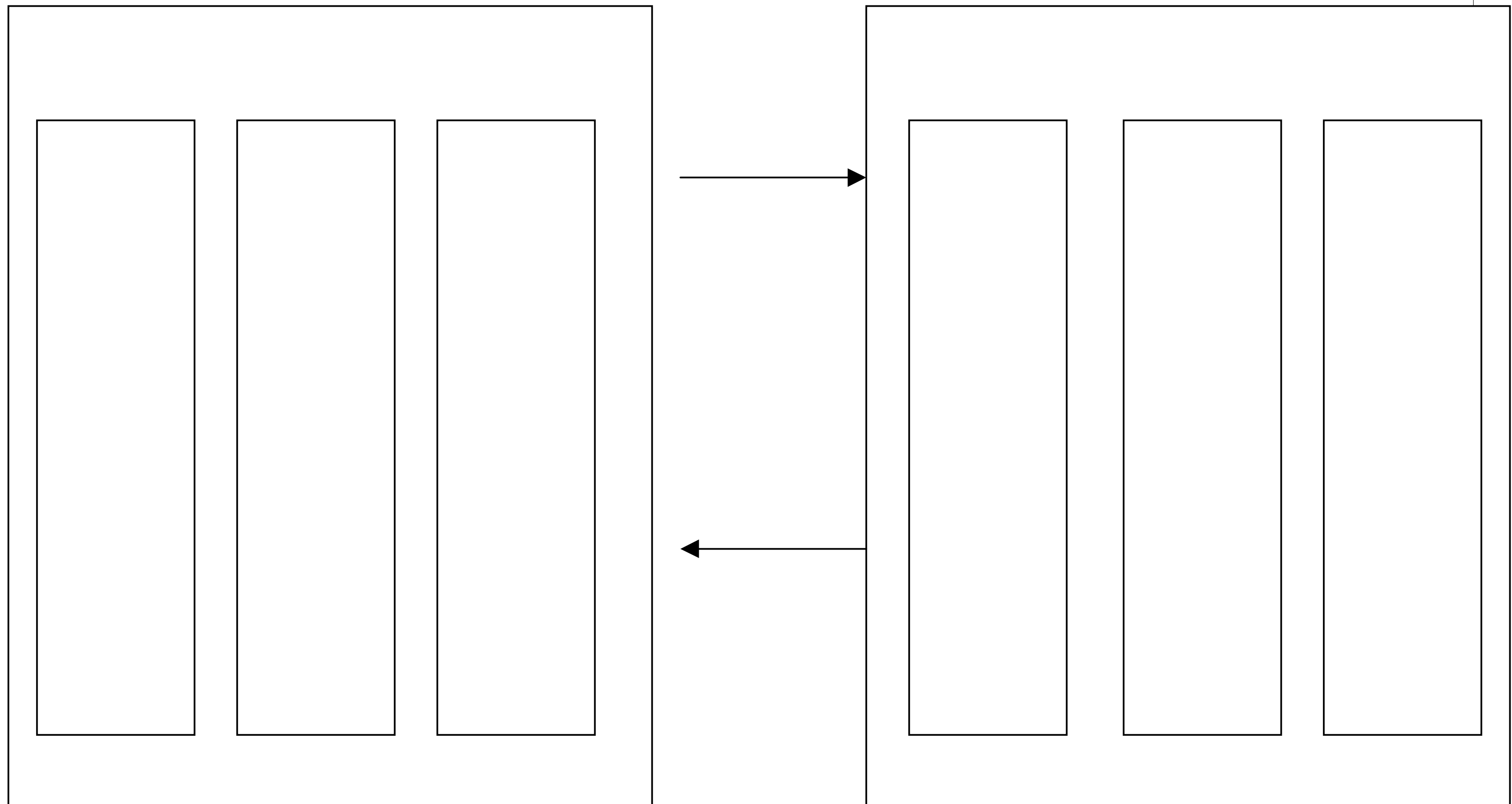
Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# RPC Semantics 3

- "At-most-once" call
  - Most appropriate for non-idempotent operations.
  - Remote procedure executed 0 or 1 time, ie, exactly once or not at all.
  - Use of retransmissions and duplicate filtering.
  - Example: Birrel et al. Implementation.
    - Use of pobes to check if server crashed.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# RPC Implementation 1



# RPC Implementation 2

- RPC runtime mechanism responsible for retransmissions, acknowledgments.
- Stubs responsible for data packaging and unpackaging; also known as marshalling and unmarshalling: putting data in form suitable for transmission. Example: Sun's XDR.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Binding

- How to determine where server is? Which procedure to call?
  - "Resource discovery" problem
    - Name service: advertises servers and services.
    - Example: Birrel et al uses Grapevine.
- Early versus late binding.
  - Early: server address and procedure name hardcoded in client.
  - Late: go to name service.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# RPC Performance

- Sources of overhead
  - data copying
  - scheduling and context switch.
- Light-Weight RPC
  - Shows that most invocations took place on a single machine.
  - LW-RPC: improve RPC performance for local case.
  - Optimizes data copying and thread scheduling for local case.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

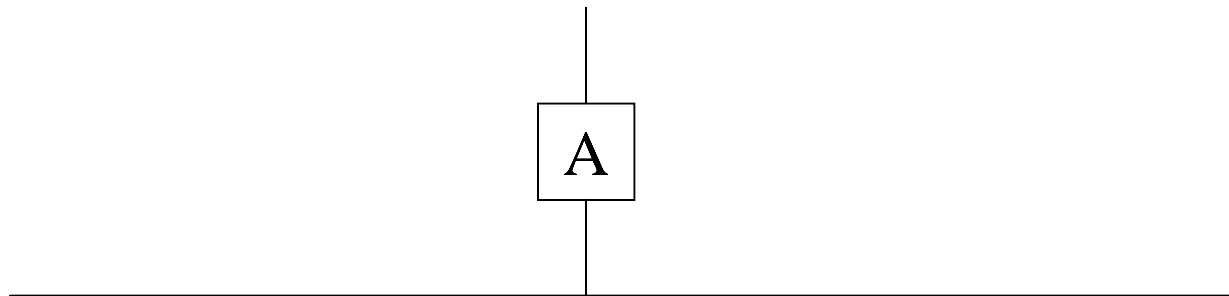
# LW-RPC 1

- Argument copying
  - RPC: 4 times (2 on call and 2 on return); copying between kernel and user space.
  - LW-RPC: common data area (A-stack) shared by client and server and used to pass parameters and results; access by client or server, one at a time.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# LW-RPC 2

- A-stack avoids copying between kernel and user spaces.
- Client and server share the same thread: less context switch (like regular calls).



# DSM Case Studies: IVY

- Environment:”loosely coupled” multiprocessor.
  - Memory is physically distributed.
  - Memory mapping managers (OS kernel):
    - Map local memories to shared virtual space.
    - Local memory as cache of shared virtual space.
    - Memory reference may cause page fault; page retrieved and consistency handled.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# IVY

- Issues:
  - Read-only versus writable data.
  - Locality of reference.
  - Granularity (1 Kbyte page size).
    - Bigger pages versus smaller pages.

# IVY

- Memory coherence strategies:
  - Page synchronization
    - Invalidation
    - Write broadcast
  - Page ownership
    - Fixed: page always owned by same processor
    - Dynamic

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# IVY Page Synchronization

- Invalidation:
  - On write fault, invalidate all copies; give faulting process write access; gets copy of page if not already there.
  - Problem: must update page on reads.
- Write broadcast:
  - On write fault, fault handler writes to all copies.
  - Expensive!

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# IVY Memory Coherence

- Paper discusses approaches to memory coherence in page-based DSM.
  - Centralized: single manager residing on a single processor managing all pages.
  - Distributed: multiple managers on multiple processors managing subset of pages.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# DSM Case Studies: Linda

- Language-based approach to DSM.
- Environment:
  - Similar to IVY, ie, loosely coupled machines connected via fast broadcast bus.
  - Instead of shared address space, processes make library calls inserted by compiler when accessing DSM.
  - Libraries access local data and communicate to maintain consistency.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Linda

- DSM: tuple space.
- Basic operations:
  - out (data): data added to tuple space.
  - in (data): removes matching data from TS; destructive.
  - read (data): same as “in”, but tuple remains in TS (non-destructive).

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Linda Primitives: Examples

- out (“P”, 5, false) : tuple (“P”, 5, false) added to TS.
  - “P” : name
  - Other components are data values.
  - Implementation reported on the paper: every node stores complete copy of TS.
  - out (data) causes data to be broadcast to every node.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Linda Primitives: Examples

- in (“P”, int I, bool b): tuple (“P”, 5, false) removed from TS.
  - If matching tuple found locally, local kernel tries to delete tuple on all nodes.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# CSci555: Lecture 3 September 17, 1999
# Concurrency, Deadlock, Transactions, Time.

Dr. Katia Obraczka
University of Southern California
Information Sciences Institute

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Administration

- Reading assignment due next class.
  - Concerns regarding grading.
- Tanenbaum book.
  - Recommended reading for background material.
  - Not required (ie, you don't have to buy it if you don't want to).

# Last Class

- E2E argument.
- Communication Models.
  - MP.
  - DSM.
  - RPC.
  - IVY.
  - ➡Linda.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Today

- Linda.
- Concurrency and Synchronization [Hauser et al.]
- Transactions [Spector et al.]
- Distributed Deadlocks [Chandy et al.]
- Time in Distributed Systems [Lamport and Jefferson]
- Replication [Birman and Gifford]

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# DSM Case Studies: Linda

- Language-based approach to DSM.
- Environment:
  - Similar to IVY, ie, loosely coupled machines connected via fast broadcast bus.
  - Instead of shared address space, processes make library calls inserted by compiler when accessing DSM.
  - Libraries access local data and communicate to maintain consistency.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Linda

- DSM: tuple space.
- Basic operations:
  - out (data): data added to tuple space.
  - in (data): removes matching data from TS; destructive.
  - read (data): same as “in”, but tuple remains in TS (non-destructive).

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Linda Primitives: Examples

- out (“P”, 5, false) : tuple (“P”, 5, false) added to TS.
  - “P” : name
  - Other components are data values.
  - Implementation reported on the paper: every node stores complete copy of TS.
  - out (data) causes data to be broadcast to every node.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Linda Primitives: Examples

- in (“P”, int I, bool b): tuple (“P”, 5, false) removed from TS.
  - If matching tuple found locally, local kernel tries to delete tuple on all nodes.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Concurrency Control and Synchronization

- How to control and synchronize possibly conflicting operations on shared data by concurrent processes?
- First, some terminology.
  - Processes.
  - Threads.
  - Tasks.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Processes

- Text book, page 165
  - Processing activity associated with an execution environment, ie, address space and resources (such as communication and synchronization resources).

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Threads

- OS abstraction of an activity/task.
  - Execution environment expensive to create and manage.
  - Multiple threads share single execution environment.
  - Single process may spawn multiple threads.
  - Maximize degree of concurrency among related activities.
  - Example: multi-threaded servers allow concurrent processing of client requests.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Other Terminology

- Process versus task.
  - Process: heavy-weight unit of execution.
  - Task/thread: light-weight unit of execution.
  - Table with other terminology page 166 of textbook.

# Threads Case Study 1

- Hauser et al.
- Examine use of user-level threads in 2 OS's:
  - Xerox Parc's Cedar (research).
  - GVX (commercial version of Cedar).
- Study dynamic thread behavior.
  - Number of threads.
  - Thread lifetime.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Thread Paradigms

- Different categories of usage:
  - Defer work: thread does work not vital to the main activity.
    - Examples: printing a document, sending mail.
  - Pumps: used in pipelining; use output of a thread as input and produce output to be consumed by another task.
  - Sleepers: tasks that repeatedly wait for an event to execute; e.g., check for network connectivity every x seconds.

# Synchronization

- So far, how one defines/activates concurrent activities.
- But how to control access to shared data and still get work done?
- Synchronization via:
  - Shared data [DSM model].
  - Communication [MP model].

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Synchronization by Shared Data

- Primitives
  - Semaphores.
  - Condition critical regions.
  - Monitors.

structure

flexibility

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Synchronization by MP.

- Explicit communication.
- Primitives: send and receive.
- Mailboxes.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Transactions 1

- Database term.
  - Execution of program that accesses a database.
- In distributed systems,
  - Concurrency control in the client/server model.
  - From client's point of view, sequence of operations executed by server in servicing client's request.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Transaction Atomicity

- "All or nothing".
- Sequence of operations to service client's request are performed in one step, ie, either all of them are executed or none are.
- Issues:
  - Multiple concurrent clients: "isolation".
  - Server failures: "failure atomicity".

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Transaction Features

- Recoverability: server should be able to "roll back" to state before transaction execution.
- Serializability: transactions have same effect whether executed concurrently or sequentially.
- Durability: effects of transactions are permanent.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Local versus Distributed Transactions

- Local transactions:
  - All transaction operations executed by single server.
- Distributed transactions:
  - Involve multiple servers.
- Both local and distributed transactions can be simple or nested.
  - Nesting: increase level of concurrency.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Concurrency Control

- Maintain transaction serializability.
- Server operations: read or write.
- 3 mechanisms:
  - Locks.
  - Optimistic concurrency control.
  - Timestamp ordering.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Locks 1

- Order transactions accessing shared data based on order of access to data.
- Lock granularity: affects level of concurrency.
  - 1 lock per shared data item.
  - Read (shared) and write locks.
  - Lock compatibility (page 381).

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Lock Implementation

- Server lock manager
  – Maintains table of locks for server data items.
  – *Lock* and *unlock* operations.
  – Clients *wait* on a lock for given data until data is released; then client is *signal*led.
  – Each client's request runs as separate server thread.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Deadlock

- Use of locks can lead to deadlock.
- Deadlock: each transaction waits for another transaction to release a lock forming a wait cycle.
- Deadlock condition: cycle in the wait-for graph.
- Deadlock prevention and detection.
- Deadlock resolution: lock timeout.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Optimistic Concurrency Control 1

- Assume that most of the time, probability of conflict is low.
- Transactions allowed to proceed in parallel until ready to commit.
- When client issues *close transaction*, check whether there was conflict; if so, some transactions aborted.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Optimistic Concurrency 2

- Read phase
  - Transactions have *tentative* version of data items it accesses.
  - *Tentative* versions allow transactions to abort without making their effect permanent.
- Validation phase
  - Executed upon *close transaction*.
  - Checks serially equivalence.
  - If validation fails, conflict resolution decides which transaction(s) to abort.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Optimistic Concurrency 3

- Write phase
  - If transaction is validated, all of its tentative versions are made permanent.
  - Read-only transactions commit immediately.
  - Write transactions commit only after their tentative versions are recorded in permanent storage.

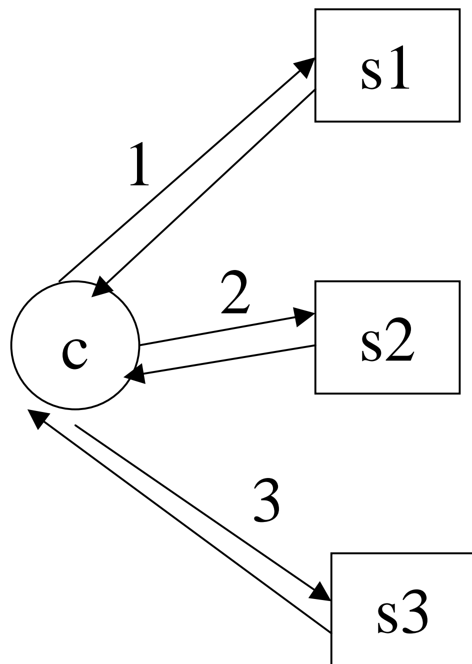
Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Timestamp Ordering

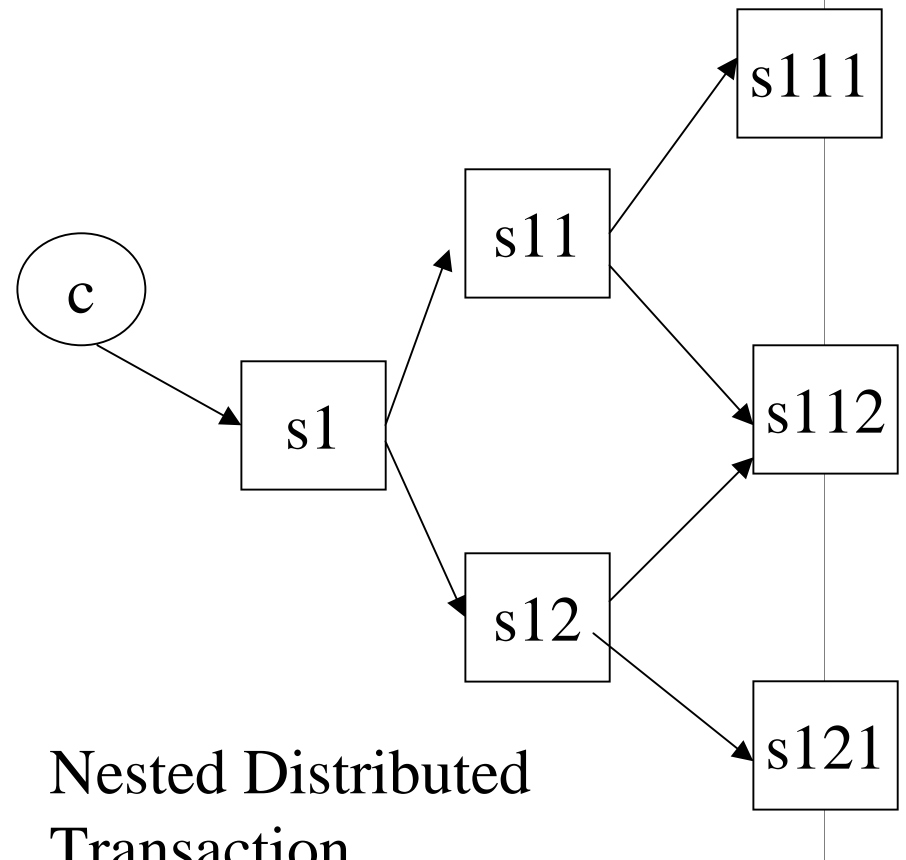
- Uses timestamps to order transactions accessing same data items according to their starting times.
- Assigning timestamps:
  - Clock based.
  - Monotonically increasing counter.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Distributed Transactions 1




Simple Distributed Transaction




Nested Distributed Transaction

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Distributed Transactions 2

- Transaction coordinator
  - First server contacted by client.
  - Responsible for aborting/commiting.
  - Adding *workers*.
- Workers
  - Other servers involved.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Atomicity in Distributed Transactions

- Harder: several servers involved.
- Atomic commit protocols
  - 1-phase commit
    - Example: coordinator sends "commit" or "abort" to workers; keeps re-broadcasting until it gets ACK from all of them that request was performed.
    - Inefficient.
    - Client makes decision to commit which may not be feasible.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

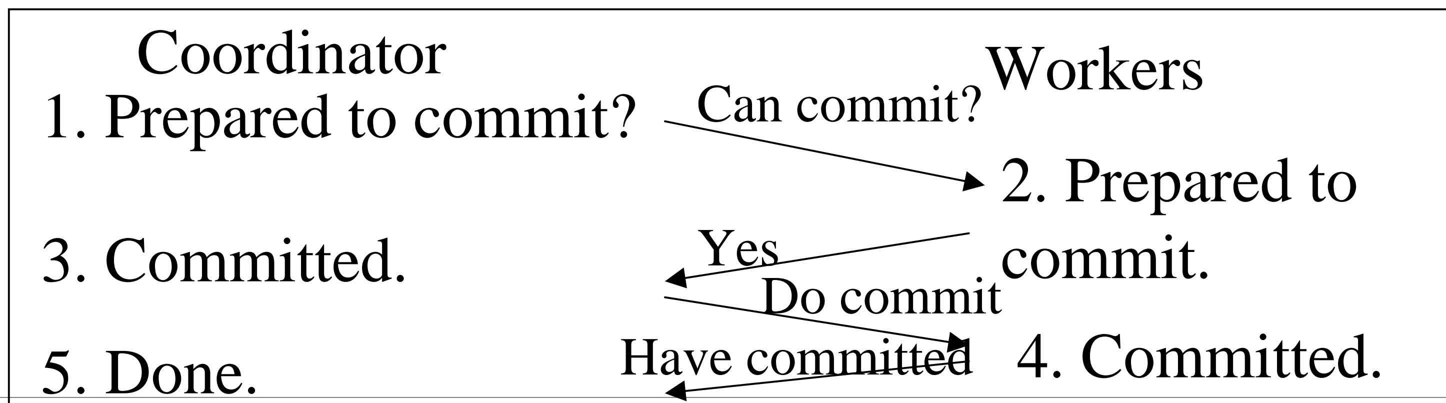
# 2-Phase Commit 1

- First phase: voting
  - Each server votes to commit or abort transaction.
- Second phase: carrying out joint decision.
  - If any server votes to abort, joint decision is to abort.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# 2-Phase Commit 2

- Issues:
  - Ensure all servers vote and reach same decision despite of failures.
  - When vote to commit, ensure server will commit even in the event of failure.




Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Concurrency Control in Distributed Transactions 1

- Locks
  - Each server manages locks for its own data.
  - Decides whether to grant lock or make transaction wait.
  - Locks cannot be released until transaction committed or aborted on all servers involved.
  - Distributed deadlocks!

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Concurrency Control in Distributed Transactions 2

- Timestamp Ordering
  - Globally unique timestamps.
    - First server issues globally unique TS and passes it around.
    - TS: <server id, local TS>
  - Servers must ensure same order of operations on all servers.
  - Clock synchronization issues

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

## Concurrency Control in Distributed Transactions 3

- Optimistic concurrency control
  - Makes even more sense?
  - Validation phase occurs during 1st. phase of 2-phase commit.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Camelot [Spector et al.]

- Supports execution of distributed transactions.
- Specialized functions:
  - Disk management
    - Allocation of large contiguous chunks.
  - Recovery management
    - Transaction abort and failure recovery.
  - Transaction management
    - Abort, commit, and nest transactions.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Distributed Deadlock 1

- When locks are used, deadlock can occur.
- Circular wait in wait-for graph means deadlock.
- Centralized deadlock detection, prevention, and resolutions schemes.
  - Examples:
    - Detection of cycle in wait-for graph.
    - Lock timeouts: hard to set TO value, aborting unnecessarily.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Distributed Deadlock 2

- Much harder to detect, prevent, and resolve. Why?
  - No global view.
  - No central agent.
  - Communication-related problems
    - Unreliability.
    - Delay.
    - Cost.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Distributed Deadlock Detection

- Cycle in the **global** wait-for graph.
- Global graph can be constructed from local graphs: hard!
  - Servers need to communicate to find cycles.
- Example from book (page 425).

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Distributed Deadlock Detection Algorithms 1

- [Chandy et al.]
- Message sequencing is preserved.
- Resource versus communication models.
  - Resource model
    - Processes, resources, and controllers.
    - Process requests resource from controller.
  - Communication model
    - Processes communicate directly via messages (request, grant, etc) requesting resources.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Resource versus Communication Models

- In resource model, controllers are deadlock detection agents; in communication model, processes.
- In resource model, process cannot continue until all requested resources granted; in communication model, process cannot proceed until it can communicate with at least one process it's waiting for.
- Different models, different detection alg's.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Distributed Deadlock Detection Schemes

- Graph-theory based.
- Resource model: deadlock when cycle among dependent processes.
- Communication model: deadlock when knot (all vertices that can be reached from *i* can also reach *i*) of waiting processes.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Deadlock Detection in Resource Model

- Use probe messages to follow edges of wait-for graph (aka edge chasing).
- Probe carries transaction wait-for relations representing path in global wait-for graph.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Deadlock Detection Example

1. Server 1 detects transaction T is waiting for U, which is waiting for data from server 2.
2. Server 1 sends probe T->U to server 2.
3. Server 2 gets probe and checks if U is also waiting; if so (say for V), it adds V to probe T->U->V. If V is waiting for data from server 3, server 2 forwards probe.
4. Paths are built one edge at a time.

Before forwarding probe, server checks for cycle (e.g., T->U->V->T).

5. If cycle detected, a transaction is aborted.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Replication 1

- Keep more than one copy of data item.
- Technique for improving performance in distributed systems.
- In the context of concurrent access to data, replicate data for increase availability.
  - Improved response time.
  - Improved availability.
  - Improved fault tolerance.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Replication 2

- But nothing comes for free.
- What's the tradeoff?
  - Consistency maintenance.
- Consistency maintenance approaches:
  - Lazy consistency (gossip approach).
  - Quorum consensus.
  - Process group.

Stronger consistency

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Quorum Consensus

- Goal: prevent partitions from from producing inconsistent results.
- Quorum: subgroup of replicas whose size gives it the right to carry out operations.
- Quorum consensus replication:
  - Update will propagate successfully to a subgroup of replicas.
  - Other replicas will have outdated copies but will be updated off-line.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Weighted Voting [Gifford] 1

- Every copy assigned a number of votes (weight assigned to a particular replica).
- Read: Must obtain *R* votes to read from any up-to-date copy.
- Write: Must obtain write quorum of *W* before performing update.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Weighted Voting 2

- W > 1/2 total votes, R+W > total votes.
- Ensures non-null intersection between every read quorum and write quorum.
- Read quorum guaranteed to have current copy.
- Freshness is determined by version numbers.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Weighted Voting 3

- On read:
  - Try to find enough copies, ie, total votes no less than R. Not all copies need to be current.
  - Since it overlaps with write quorum, at least one copy is current.
- On write:
  - Try to find set of up-to-date replicas whose votes no less than W.
  - If no sufficient quorum, current copies replace old ones, then update.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# ISIS 1

- Goal: provide programming environment for development of distributed systems.
- Assumptions:
  - DS as a set of processes with disjoint address spaces, communicating over LAN via MP.
  - Processes and nodes can crash.
  - Partitions may occur.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# ISIS 2

- Distinguishing feature: group communication mechanisms
  - Process group: processes cooperating in implementing task.
  - Process can belong to multiple groups.
  - Dynamic group membership.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Virtual Synchrony

- Real synchronous systems
  - Events (e.g., message delivery) occur in the same order everywhere.
  - Expensive and not very efficient.
- Virtual synchronous systems
  - Illusion of synchrony.
  - Weaker ordering guarantees when applications allow it.

# ISIS Features

- Atomic multicast
  - One or none.
  - As long as 1 member is up, message will be delivered to remaining members.
  - FBCAST: unordered multicast.
  - CBCAST: causally ordered multicast.
    - “Happen before” order.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Time in Distributed Systems

- Notion of time is critical.
- “Happen before” notion.
  - Example: concurrency control using TS’s.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Atomic Multicast 1

- One or none.
- As long as 1 member is up, message will be delivered to remaining members.
- FBCAST: unordered multicast.
- CBCAST: causally ordered multicast.
  - "Happened before" order.
  - Messages from given process in order.
  - UDP+ACKs+reXmissions.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Atomic Multicast 2

- ABCAST: totally ordered multicast.
  - Ordered delivery of messages across processes.
  - Uses token site to serialize messages.

# Other Features

- Process groups
  - Group membership management.
- Broadcast and group RPC
  - RPC-like interface to CBCAST, ABCAST, and FBCAST protocols.
  - Delivery guarantees
    - Caller indicates how many responses required.
      - No responses: asynchronous.
      - 1 or more: synchronous.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Implementation

- Set of library calls on top of UNIX.
- Commercially available.
- In the paper, example of distributed DB implementation using ISIS.
- HORUS: extension to WANs.

# Time in Distributed Systems

- Notion of time is critical.
- "Happened before" notion.
  - Example: concurrency control using TS's.
  - "Happened before" notion is not straightforward in distributed systems.
    - No guarantees of synchronized clocks.
    - Communication latency.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Event Ordering

- Lamport defines partial ordering:
  1. If 2 events occurred in the same process, then they occurred in the order observed.
  2. Whenever message sent between 2 processes, event sending message occurred before event receiving message.
  3. If X->Y and Y->Z, then X->Z.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Causal Ordering

- “Happened before” also called causal ordering.
- Example: text book page 298.
- In summary, possible to draw happened-before relationship between events if they happen in same process or there’s chain of messages between them.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Logical Clocks

- Monotonically increasing counter.
- No relation with real clock.
- Each process keeps its own logical clock Cp used to timestamp events.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Causal Ordering and Logical Clocks

1. $Cp$ incremented before each event.

   $Cp=Cp+1$.

2. When $p$ sends message $m$, it piggybacks $t=Cp$.

3. When q receives $(m, t)$, it computes:

   $Cq=max(Cq, t)$ before timestamping message receipt event.

Example: text book page 299.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Total Ordering

- Extending partial to total order.
- Global timestamps:
  - $(T_a, p_a)$, where $T_a$ is local TS and $p_a$ is the process id.
  - $(T_a, p_a) < (T_b, p_b)$ iff $T_a < T_b$ or $T_a = T_b$ and $p_a < p_b$
  - Total order consistent with partial order.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Virtual Time [Jefferson] 1

- Time warp mechanism.
- May or may not have connection with real time.
- Uses optimistic approach, ie, events and messages are processed in the order received: “look-ahead”.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Virtual Time 2

- Process virtual clock (local virtual clock) set to TS of next message in input queue.
- If next message's TS is in the past, rollback!
  - Can happen due to different computation rates and communication latency.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Rolling Back

- Process goes back to TS(latest message).
- Cancels all intermediate effects.
- Then, executes forward.
- Rolling back is expensive!
  - Messages may have been sent to other processes causing them to send messages, etc.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Anti-Messages 1

- For every message, there is an anti-message with same content but different *sign*.
- When sending message, message goes to receiver input queue and a copy with "-" sign is enqueued in the sender's output queue.
- Message is retained for use in case of roll back.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Anti-Message 2

- Message + its antimessage = 0 when in the same queue.
- Processes must keep log to "undo" operations.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Implementation

- Local control.
- Global control
  - How to make sure system as a whole progresses.
  - Errors and I/O when roll back occurs.
  - Avoid running out of memory.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Global Virtual Clock

- Snapshot of system at given real time.
- Minimum of all local virtual times.
- Lower bound on how far processes roll back.
- Purge state before GVT.
- GVT computed concurrently with rest of time warp mechanism.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# CSci555: Advanced Operating Systems
## Lecture 4 - September 24, 1999

Dr. Clifford Neuman
University of Southern California
Information Sciences Institute

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Naming Concepts

- Name
  - What you call something
- Address
  - Where it is located
- Route
  - How one gets to it

  What is http://www.isi.edu/people/bcn ?

- But it is not that clear anymore, it depends on perspective. A name from one perspective may be an address from another.
  - Perspective means layer of abstraction

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# What are the things we name

- Users
  - To direct, and to identify
- Hosts (computers)
  - High level and low level
- Services
  - Service and instance
- Files and other “objects”
  - Content and repository
- Groups
  - Of any of the above

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# How we name things

- Host-Based Naming
  - Host-name is required part of object name
- Global Naming
  - Must look-up name in global database to find address
  - Name transparency
- User/Object Centered Naming
  - Namespace is centered around user or object
- Attribute-Based Naming
  - Object identified by unique characteristics
  - Related to resource discovery / search / indexes

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Namespace

- A name space maps:

$$\Sigma^* \rightarrow X \, \varepsilon \, O$$

At a particular point in time.

- The rest of the definition, and even some of the above, is open to discussion/debate.
- What is a “flat namespace”
  – Implementation issue

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Case Studies

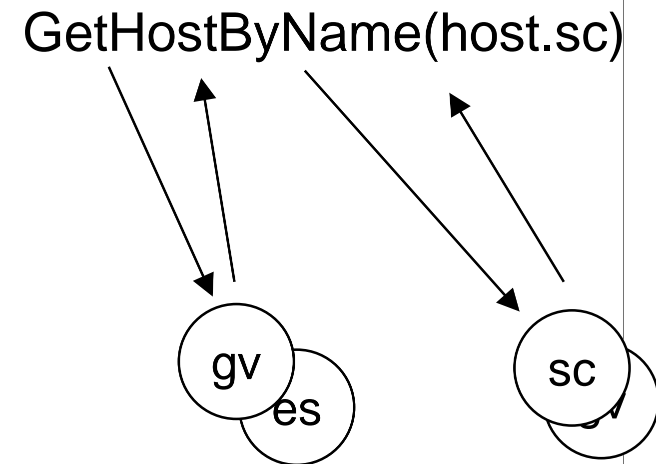
- Host Table
  - Flat namespace (?)
  - Global namespace (?)
- Grapevine
  - Two-level, iterative lookup
  - Clearinghouse 3 level
- Domain name system
  - Arbitrary depth
  - Iterative or recursive(chained) lookup
  - Multi-level caching

```
GetHostByName(usc.arpa){
    scan(host file);
    return(matching entry);
    }
```

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

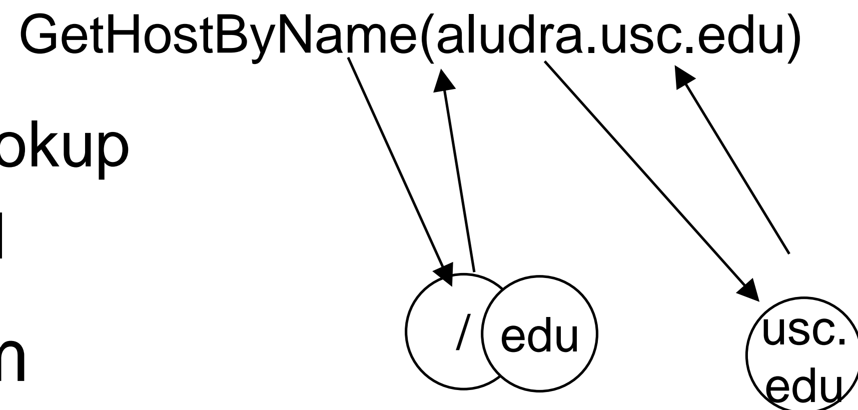
# Case Studies

- Host Table
  - Flat namespace (?)
  - Global namespace (?)
- Grapevine
  - Two-level, iterative lookup
  - Clearinghouse 3 level
- Domain name system
  - Arbitrary depth
  - Iterative or recursive(chained) lookup
  - Multi-level caching



# Case Studies

- Host Table
  - Flat namespace (?)
  - Global namespace (?)
- Grapevine
  - Two-level, iterative lookup
  - Clearinghouse 3 level
- Domain name system
  - Arbitrary depth
  - Iterative or recursive(chained) lookup




Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Scalability of naming

- Scalability
  - Ability to continue to operate efficiently as a system grows large, either numerically, geographically, or administratively.
- Affected by
  - Frequency of update
  - Granularity
  - Evolution/reconfiguration
- DNS characteristics
  - Multi-level implementation
  - Replication of root and other servers
  - Multi-level caching

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Other implementations of naming

- Broadcast
  - Limited scalability, but faster local response
- Prefix tables
  - Essentially a form of caching
- Capabilities
  - Combines security and naming
  - Traditional name service built over capability based addresses

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Closure

- Closure binds an object to the namespace within which names embedded in the object are to be resolved.
  - Namespace is dynamic
  - “Object” may as small as the name itself

# Advanced Name Systems

- DEC's Global Naming
  - Support for reorganization the key idea
  - Little coordination needed in advance
- Prospero Directory Service
  - Multiple namespace centered around a "root" node that is specific to each namespace.
  - Closure binds objects to this "root" node.
  - Used today as an embedded directory service.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Non Hierarchical Naming

- Examples
  - Prospero
  - The Web
- User centered naming
  - Prospero
  - Tilde
  - Quicksilver

# Resource Discovery

- Similar to naming
  - Browsing related to directory services
  - Indexing and search similar to attribute based naming
- Attribute based naming
  - Profile
  - Multi-structured naming
- Search engines

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Resource Discovery

- Similar to naming
  - Browsing related to directory services
  - Indexing and search similar to attribute based naming
- Object handles
  - Uniform Resource Locators (URL's)
    - Is it a name or an address?
  - Uniform Resource Names (URN's)
    - Is a directory service required

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# CSci555: Advanced Operating Systems
## Lecture 5 - October 1, 1999

Dr. Clifford Neuman
University of Southern California
Information Sciences Institute

# The Web

- Object handles
  - Uniform Resource Locators (URL's)
    - Is it a name or an address?
  - Uniform Resource Names (URN's)
    - Is a directory service required
  - How URL's are misused
- XML
  - Definitions provide a form of closure
    - Conceptual level rather than the "namespace" level.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Security Goals

- Protection
  - Enforced by hardware
  - Depends on trusted kernel
- Authentication
  - Determining identity of principal
- Integrity
  - Authenticity of document
  - That it hasn't changes
- Privacy
  - That inappropriate information is not disclosed

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

## Security Policy

- Access Matrix

| Subject | OBJ1 | OBJ2 |
| --- | --- | --- |
| bcn | RW | R |
| gost-group | RW | - |
| obraczka | R | RW |
| tyao | R | R |
| Csci555 | R | - |

- implemented as:
  - Capabilities or
  - Access Control list

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Access Control Lists

- Advantages
  - Easy to see who has access
  - Easy to change/revoke access
- Disadvantages
  - Time consuming to check access
- Extensions to ease management
  - Groups
  - EACLs

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Extended Access Control Lists

- Conditional authorization
  - Implemented as restrictions on ACL entries and embedded as restrictions in authentication and authorization credentials

| Principal | Rights | Conditions |
|---|---|---|
| bcn | RW | HW-Authentication<br>Retain Old Items |
| gost-group | RW | TIME: 9AM-5PM |
| authorization server | R | Delegated-Access |
| * | R | Load Limit 8<br>Use: Non-Commercial |
| * | R | Payment: $Price |

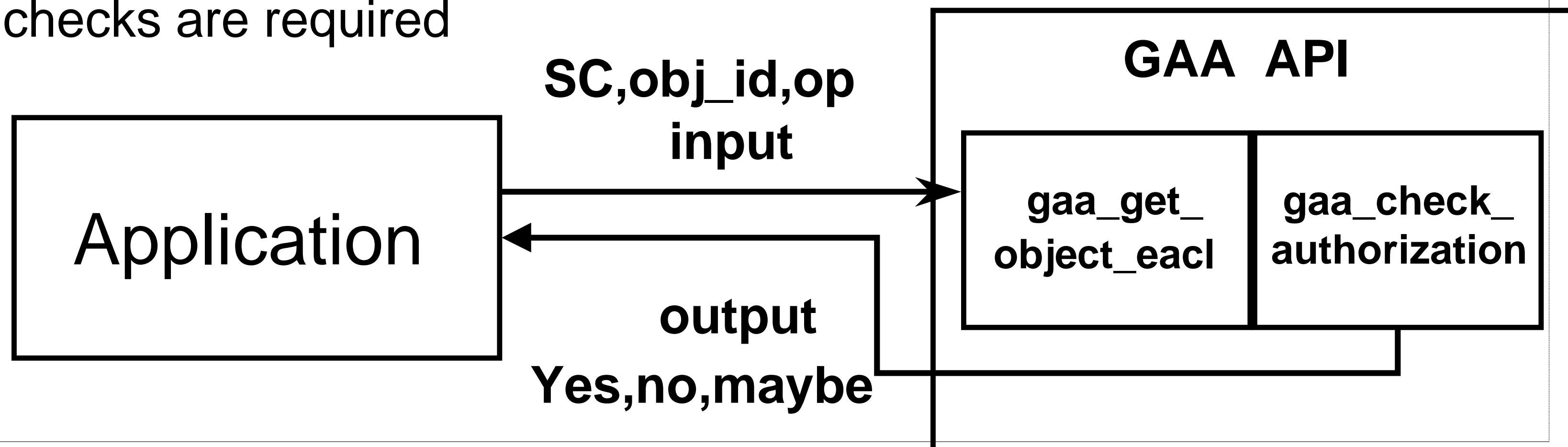
Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Generic Authorization and Access-control API

Allows applications to use the security infrastructure to implement security policies.

**gaa_get_object_eacl** function called before other GAA API routines, each of which requires a handle to object EACL to identify EACLs on which to operate.

**gaa_check_authorization** function tells application whether requested operation is authorized, or if additional application specific checks are required




Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Capabilities

- Advantages
  - Easy and efficient to check access
  - Easily propagated
- Disadvantages
  - Hard to protect capabilities
  - Easily propagated
  - Hard to revoke
- Hybrid approach
  - EACL's/proxies

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Protecting capabilities

- Stored in TCB
  - Only protected calls manipulate
- Limitations ?
  - Works in centralized systems
- Distributed Systems
  - Tokens with random or special coding
  - Possibly protect through encryption
  - How does Amoeba do it? (claimed)

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Basic Security Services

- Authentication
- Authorization
- Accounting
- Audit
- Assurance
- Payment
- Protection
- Policy
- Privacy
- Confidentiality

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Network Threats

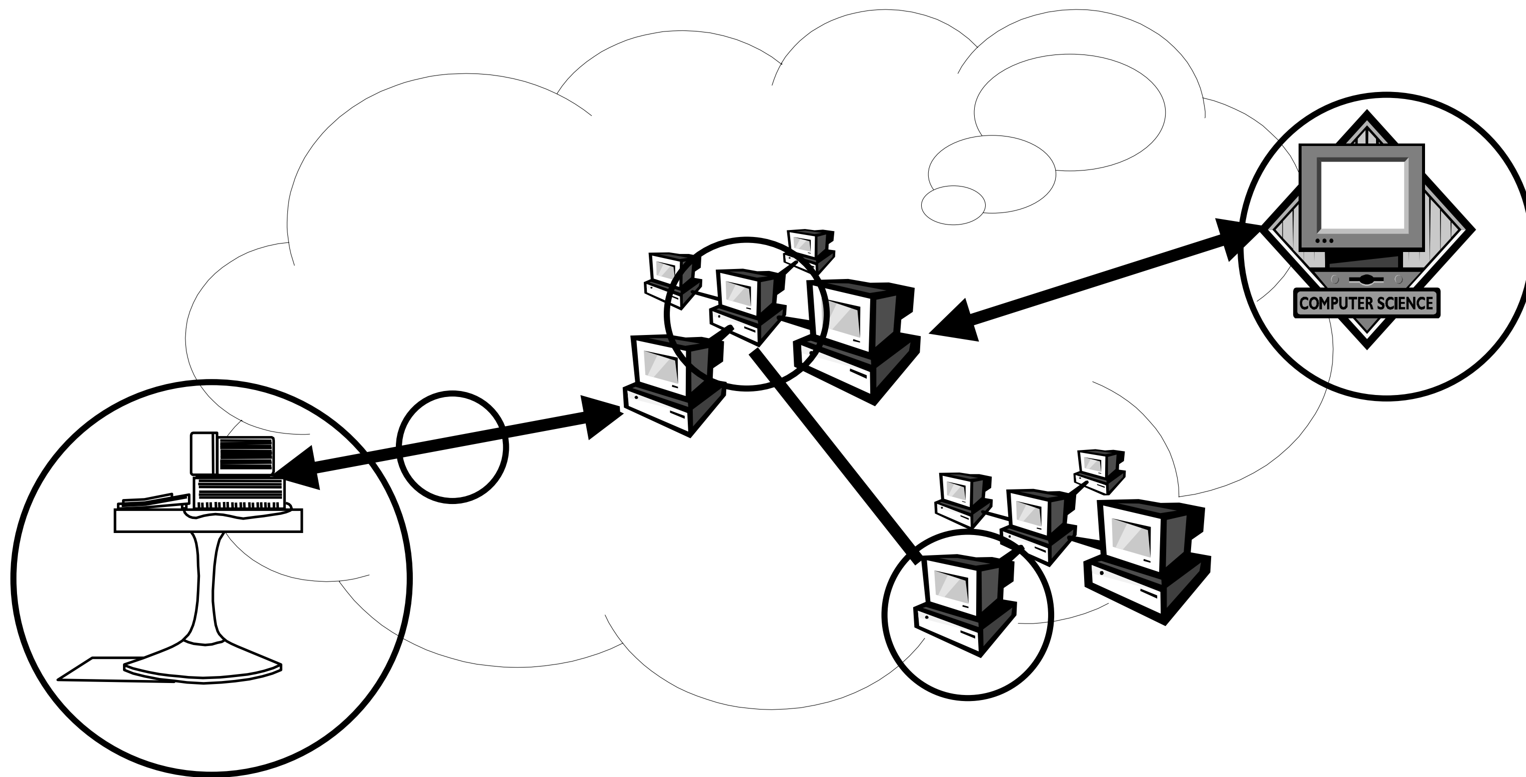
- Unauthorized release of data
- Unauthorized modification of data
- Spurious association initiation
  - Impersonation
- Denial of use
- Traffic analysis

❖ Attacks may be

- Active or passive

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Likely points of attack (location)



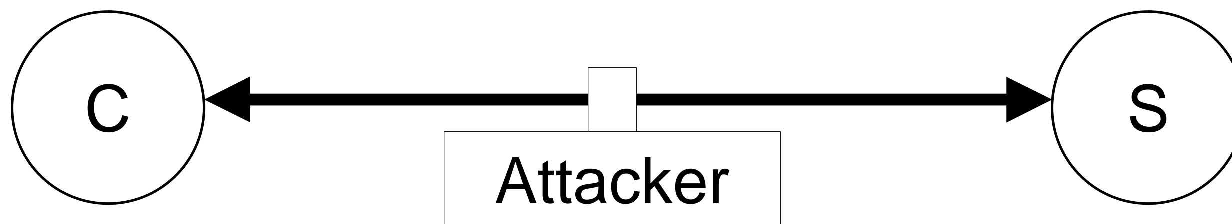

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Likely points of attack (module)

- Against the protocols
  - Sniffing for passwords and credit card numbers
  - Interception of data returned to user
  - Hijacking of connections
- Against the server
  - The commerce protocol is not the only way in
  - Once an attacker is in, all bets are off
- Against the client's system
  - You have little control over the client's system

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

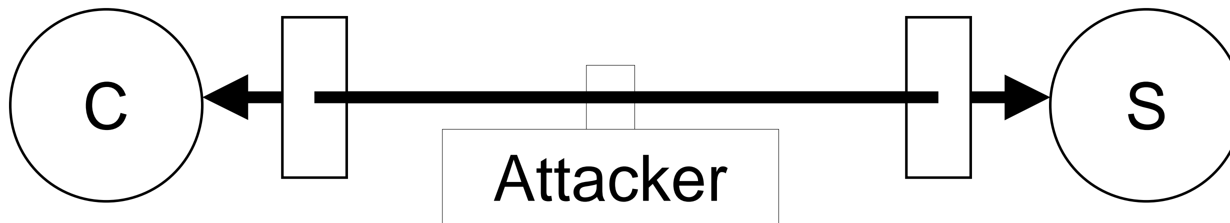
# Network Attacks




- Eavesdropping
  - Listening for passwords or credit card numbers
- Message stream modification
  - Changing links and data returned by server
- Hijacking
  - Killing client and taking over connection

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Network Attack Countermeasures

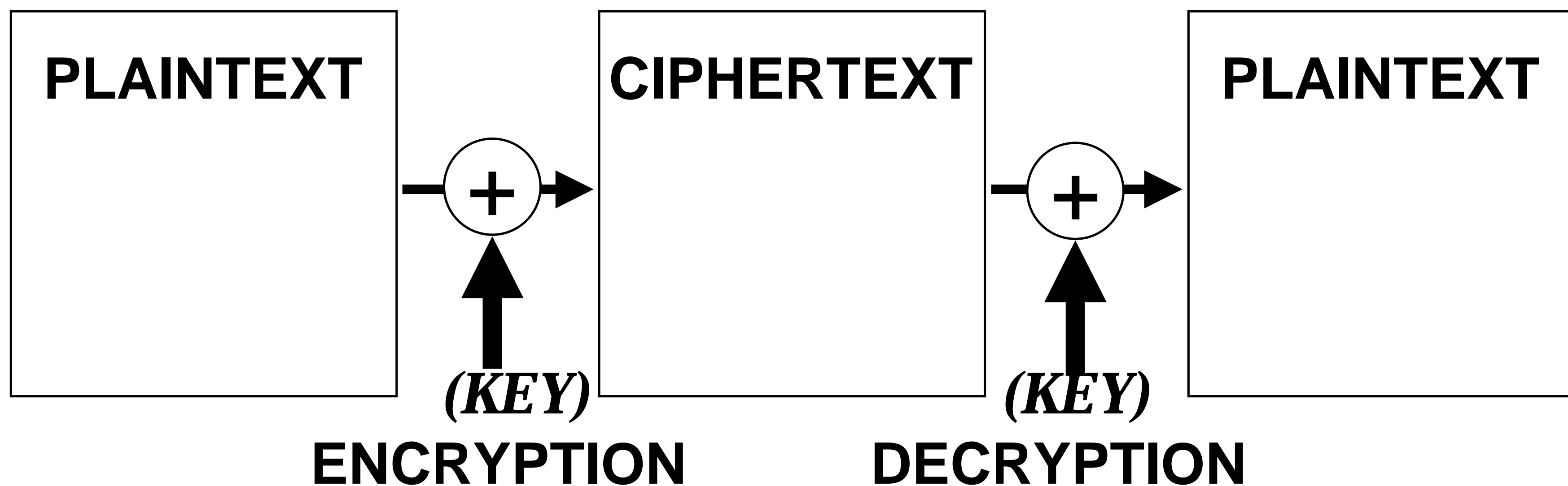



- Don't send anything important
  - Not everything needs to be protected
- Encryption
  - For everything else
  - Mechanism limited by client side software

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Encryption for confidentiality and integrity

- Encrypton used to scramble data



# Authentication

- Proving knowledge of encryption key
  - Nonce = Non repeating value

{Nonce or timestamp}$K_c$

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# CSci555: Advanced Operating Systems
## Lecture 6 - October 8, 1999

Dr. Clifford Neuman
University of Southern California
Information Sciences Institute

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

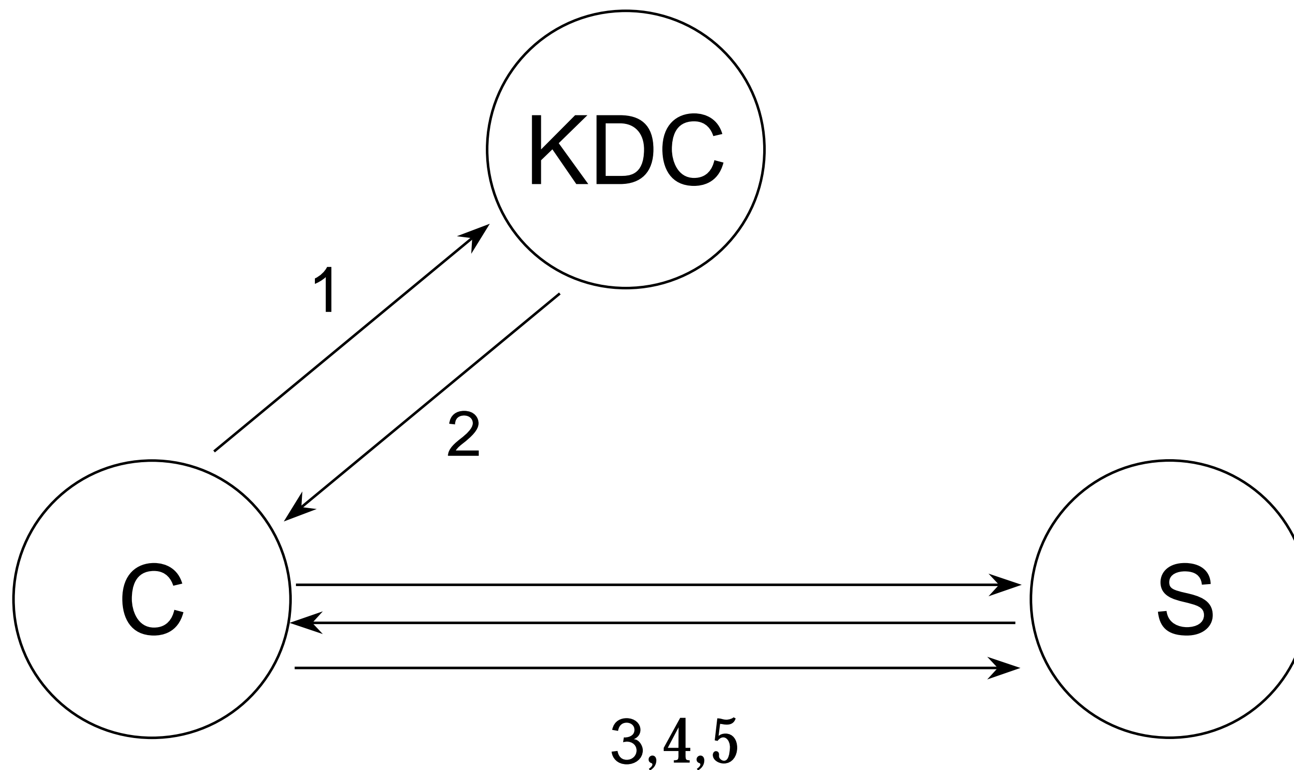
# Key distribution

- Conventional cryptography
  - Single key shared by both parties
- Public Key cryptography
  - Public key published to world
  - Private key known only by owner
- Third party certifies or distributes keys
  - Certification infrastructure
  - Authentication

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

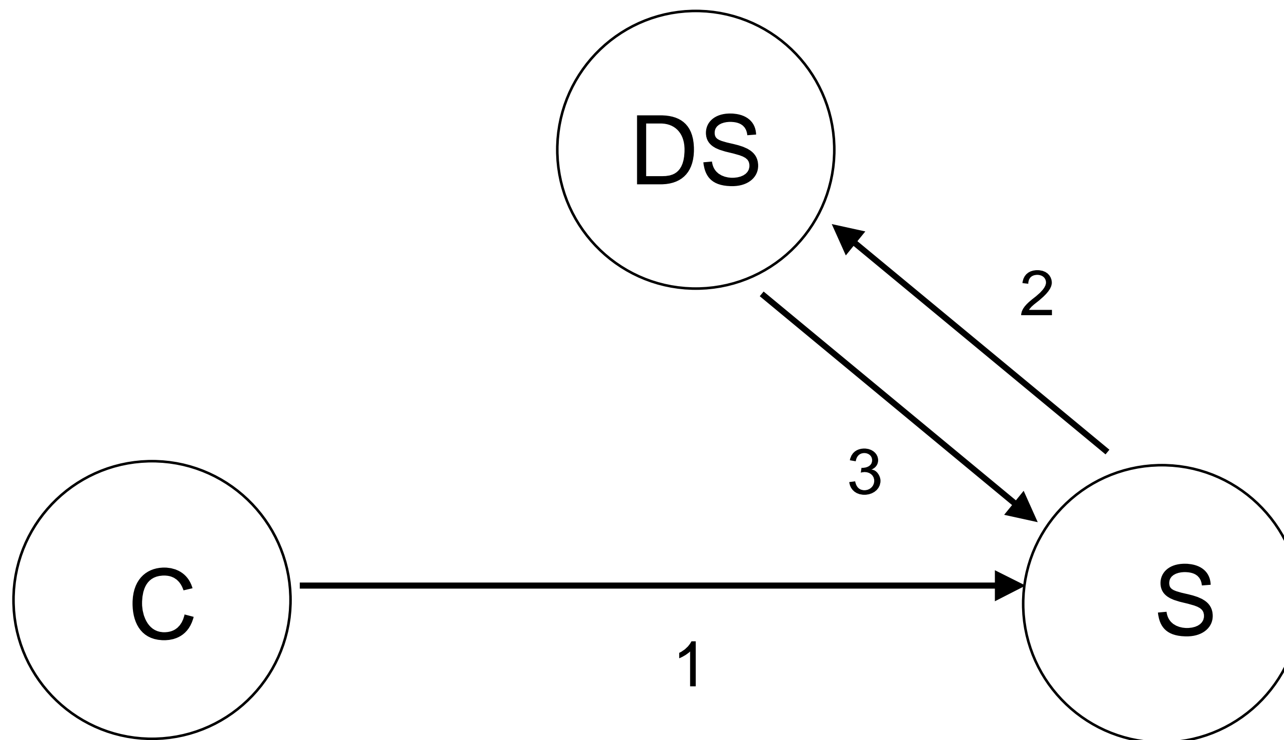
# Authentication w/ Conventional Crypto

- Kerberos or Needham Schroeder



# Authentication w/ PK Crypto

- Based on public key certificates

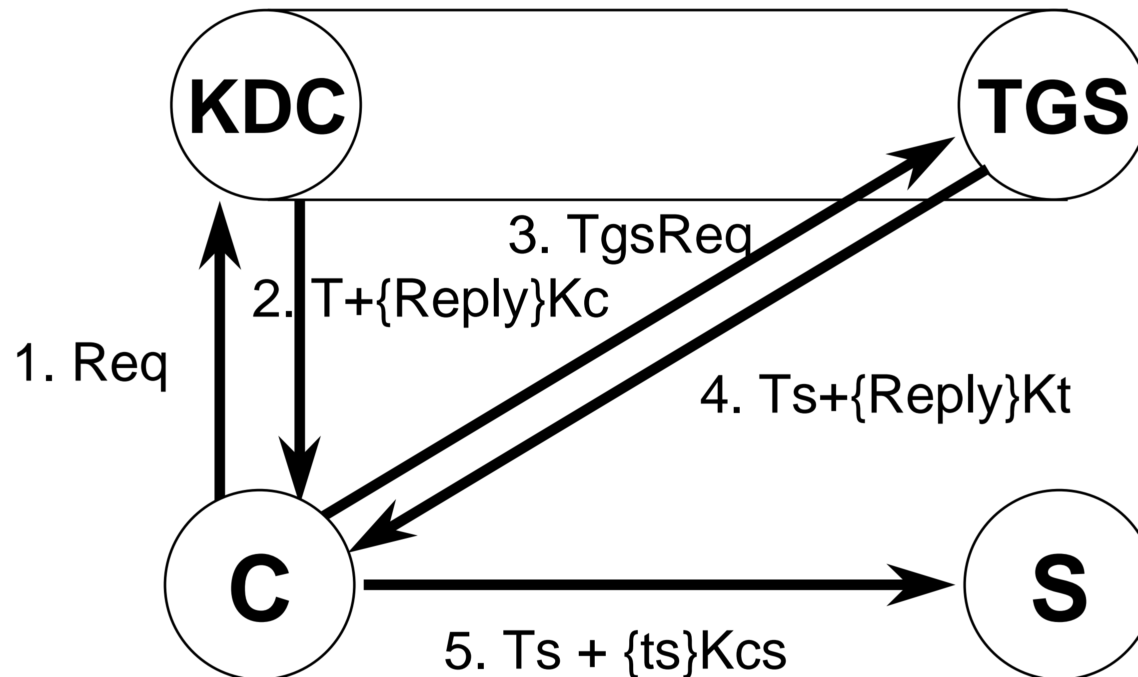



Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Kerberos

Third-party authentication service

- Distributes session keys for authentication, confidentiality, and integrity




*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Public Key Cryptography (revisited)

- Key Distribution
  - Confidentiality not needed for public key
  - Solves $n^2$ problem
- Performance
  - Slower than conventional cryptography
  - Implementations use for key distribution, then use conventional crypto for data encryption
- Trusted third party still needed
  - To certify public key
  - To manage revocation
  - In some cases, third party may be off-line

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Certificate-Based Authentication

## Certification authorities issue signed certificates

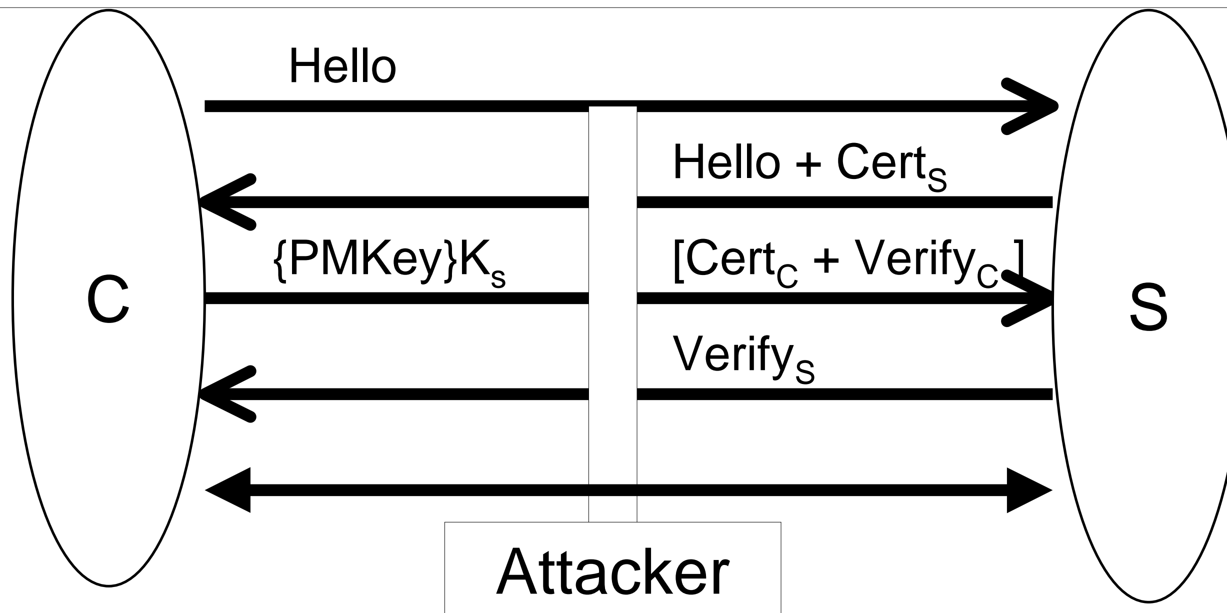
- Banks, companies, & organizations like Verisign act as CA's
- Certificates bind a public key to the name of a user
- Public key of CA certified by higher-level CA's
- Root CA public keys configured in browsers & other software
- Certificates provide key distribution

## Authentication steps

- Verifier provides nonce, or a timestamp is used instead.
- Principal selects session key and sends it to verifier with nonce, encrypted with principal's private key and verifier's public key, and possibly with principal's certificate
- Verifier checks signature on nonce, and validates certificate.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Secure Sockets Layer (and TLS)




**Encryption support provided between**

Browser and web server - below HTTP layer

**Client checks server certificate**

Works as long as client starts with the correct URL

**Key distribution supported through cert steps**

**Authentication provided by verify steps**

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Trust models for certification

- X.509 Hierarchical
  - Single root (original plan)
  - Multi-root (better accepted)
  - SET has banks as CA's and common SET root
- PGP Model
  - "Friends and Family approach" - S. Kent
- Other representations for certifications
- No certificates at all
  - Out of band key distribution
  - SSH

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Global Authentication Service

- Pair-wise trust in hierarchy
  - Name is derived from path followed
  - Shortcuts allowed, but changes name
  - **Exposure of path is important for security**
- Compared to Kerberos
  - Transited field in Kerberos - doesn't change name
- Compared with X.509
  - X.509 has single path from root
  - X.509 is for public key systems
- Compared with PGP
  - PGP evaluates path at end, but may have name conflicts

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

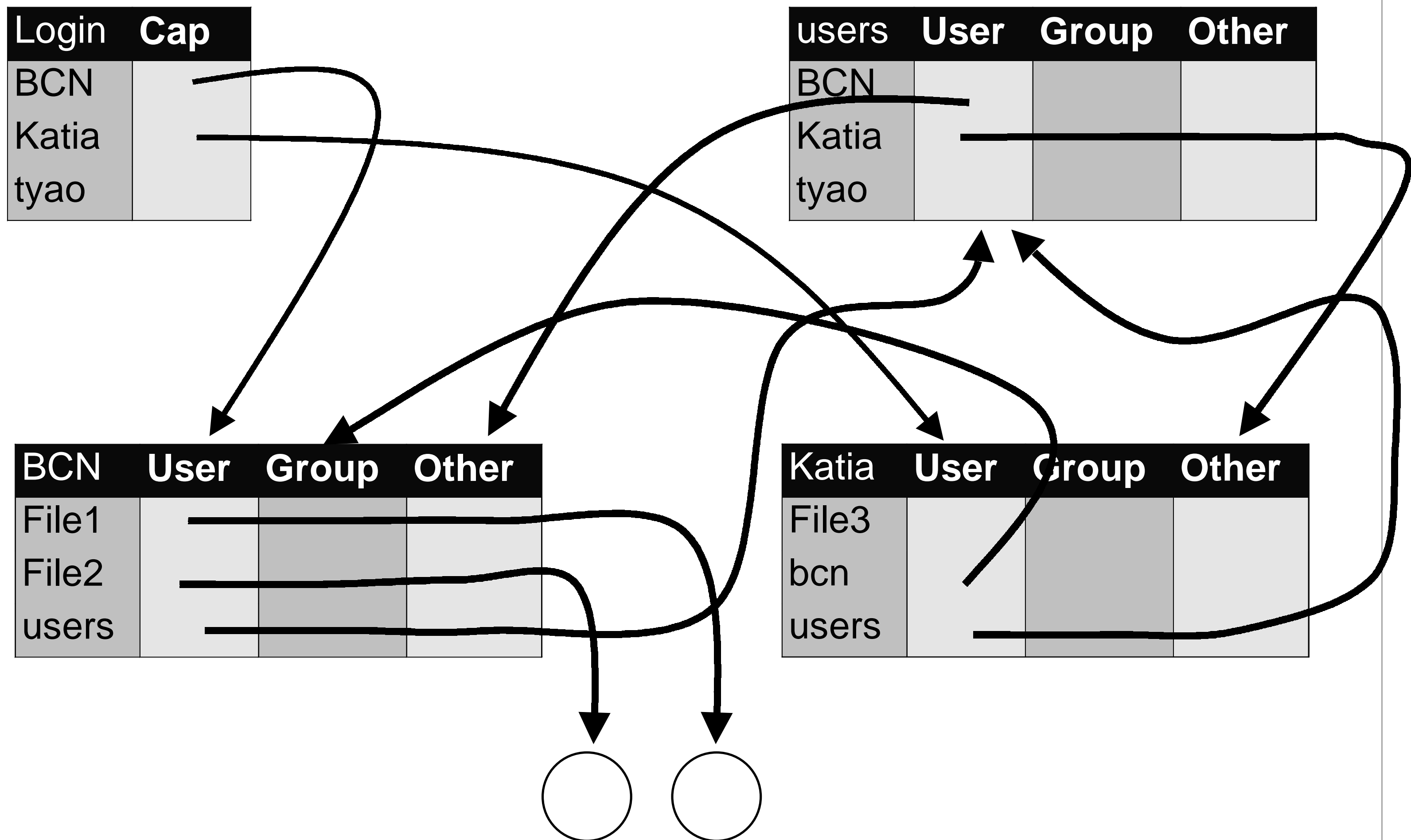
# Capability Based Systems - Amoeba

*"Authentication not an end in itself"*

- Theft of capabilities an issue
  - Claims about no direct access to network
  - Replay an issue
- Modification of capabilities a problem
  - One way functions provide a good solution
- Where to store capabilities for convenience
  - In the user-level naming system/directory
  - 3 columns
- Where is authentication in Amoeba
  - To obtain initial capability

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Capability Directories in Amoeba



# Security Architectures

- DSSA
  - Delegation is the important issue
    - Workstation can act as user
    - Software can act as workstation - if given key
    - Software can act as developer - if checksum validated
  - Complete chain needed to assume authority
  - Roles provide limits on authority - new sub-principal
- Proxies - Also based on delegation
  - Limits on authority explicitly embedded in proxy
  - Works well with access control lists

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Distributed Authorization

- It must be possible to maintain authorization information separate from the end servers
  - Less duplication of authorization database
  - Less need for specific prior arrangement
  - Simplified management
- Based on restricted proxies which support
  - Authorization servers
  - Group Servers
  - Capabilities
  - Delegation

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Proxies

- A proxy allows a second principal to operate with the rights and privileges of the principal that issued the proxy
  - Existing authentication credentials
  - Too much privilege and too easily propagated
- Restricted Proxies
  - By placing conditions on the use of proxies, they form the basis of a flexible authorization mechanism

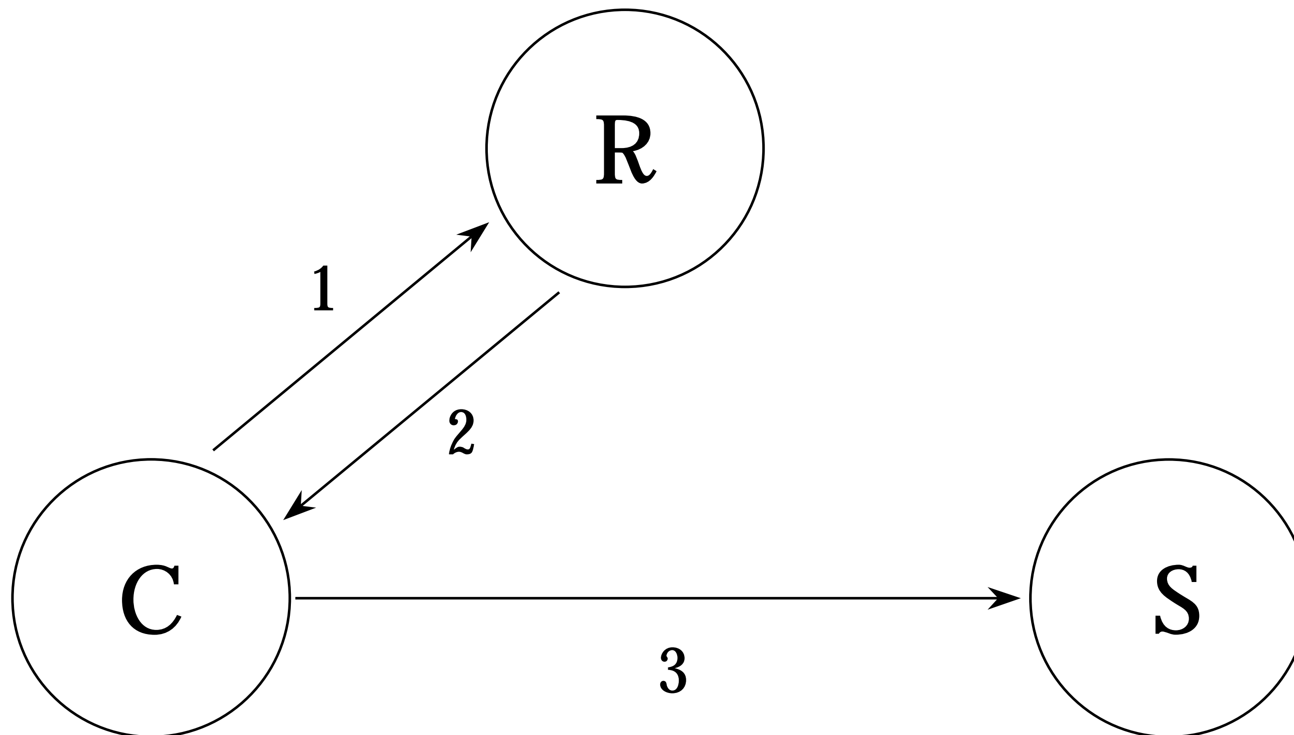
*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Restricted Proxies

- Two Kinds of proxies
  - Proxy key needed to exercise bearer proxy
  - Restrictions limit use of a delegate proxy
- Restrictions limit authorized operations
  - Individual objects
  - Additional conditions

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Authorization and Group Services




1. Authenticated authorization request (operation X)

2. [operation X only]R, {Kproxy} Ksession

3. [operation X only]R, authentication using Kproxy

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Central Authorization

- Authorization server uses extended ACLs
  - Conditions are not evaluated, but instead attached to credentials
- Groups implemented by auth server
  - Server grants right to assert group membership
- Application servers configured to use authorization server
  - Minimal local ACL
  - Can use multiple Authorization servers

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Applied Security

- Electronic commerce
  - SSL Applies authentication and encryption
  - NetCheque applies proxies
  - SET applies certification
  - End system security a major issue
- What we have today
  - Firewalls
  - Web passwords, encryption, certificates
  - Windows 2000 uses Kerberos

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# CSci555: Advanced Operating Systems
## Lecture 7 - October 15, 1999
## File Systems

Dr. Katia Obraczka
University of Southern California
Information Sciences Institute

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Outline

- Time in distributed systems.
- File systems.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# File Systems

- Provide set of primitives that abstract users from details of storage access and management.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Distributed File Systems

- Promote sharing across machine boundaries.
- Transparent access to files.
- Make diskless machines viable.
- Increase disk space availability by avoiding duplication.
- Balance load among multiple servers.

# Sun Network File System 1

- De facto standard:
  - Mid 80's.
  - Widely adopted in academia and industry.
- Provides transparent access to remote files.
- Uses Sun RPC and XDR.
  - NFS protocol defined as set of procedures and corresponding arguments.
  - Synchronous RPC:
    - Client blocks until it gets results from server.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

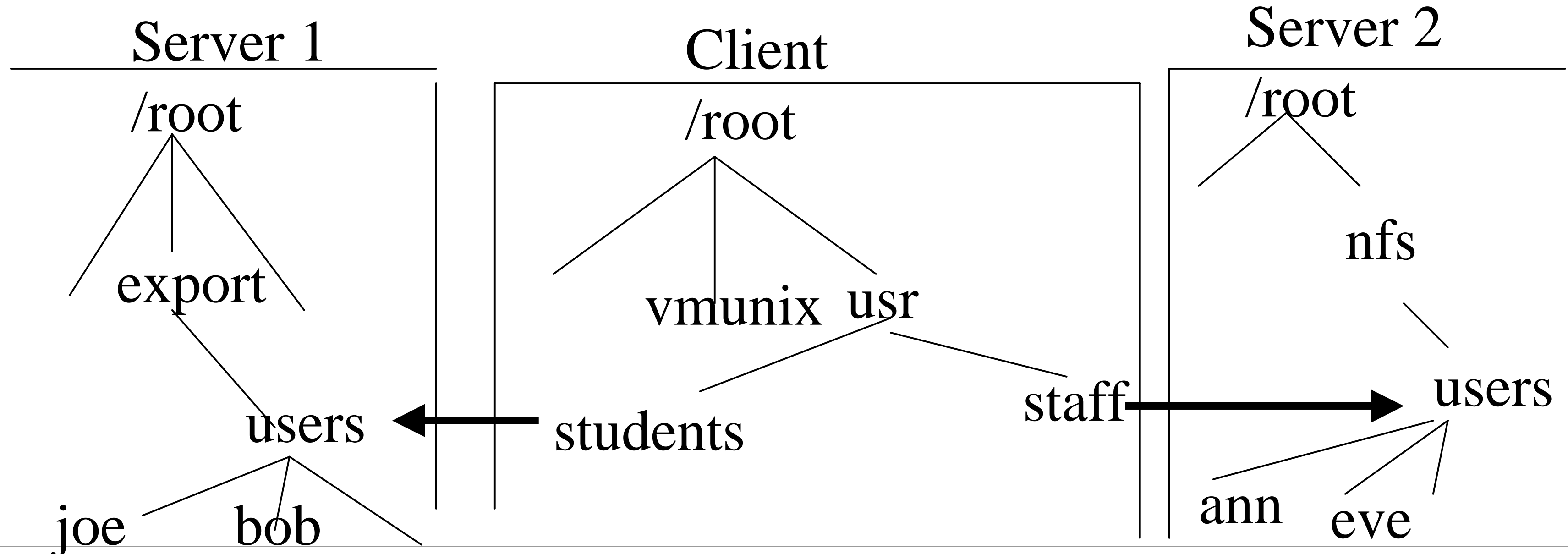
# Sun NFS 2

- Stateless server:
  - Remote procedure calls are self-contained.
  - Servers don't need to keep state about previous requests.
    - Flush all modified data to disk before returning from RPC call.
  - Robustness.
    - No state to recover.
    - Clients retry.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Location Transparency

- Client's file name space includes remote files.
  - Shared remote files are *exported* by server.
  - They need to be *remote-mounted* by client.




Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Achieving Transparency 1

- Mount service.
  - Mount remote file systems in the client's local file name space.
  - Mount service process runs on each node to provide RPC interface for mounting and unmounting file systems at client.
  - Runs at system boot time or user login time.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Achieving Transparency 2

- Automounter.
  - Dynamically mounts file systems.
  - Runs as user-level process on clients (daemon).
  - Resolves references to unmounted pathnames within files managed by the automounter by mounting them on demand.
  - Maintains a table of mount points and the corresponding server(s); sends probes to server(s).
  - Primitive form of replication.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Transparency?

- Early binding.
  - Mount system call attaches remote file system to local mount point.
  - Client deals with host name once.
  - But, mount needs to happen before remote files become accessible.

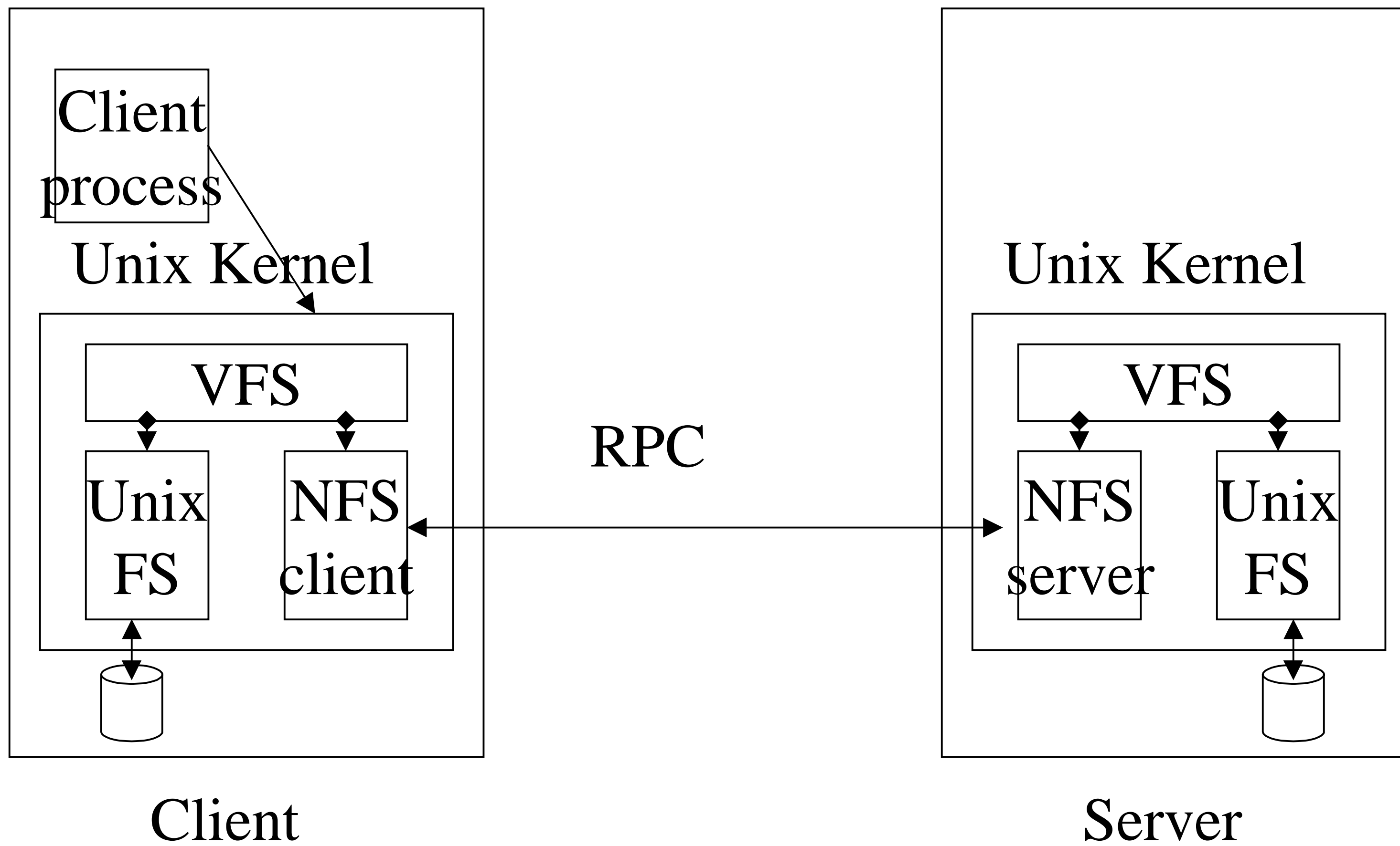
*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Other Functions

- NFS file and directory operations:
  - read, write, create, delete, getattr, etc.
- Access control:
  - File and directory access permissions.
- Path name translation:
  - Lookup for each path component.
  - Caching.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Implementation




*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Virtual File System 1

- VFS added to UNIX kernel.
  - Location-transparent file access.
  - Distinguishes between local and remote access.
- @ client:
  - Processes file system system calls to determine whether access is local (passes it to UNIX FS) or remote (passes it to NFS client).
- @ server:
  - NFS server receives request and passes it to local FS through VFS.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

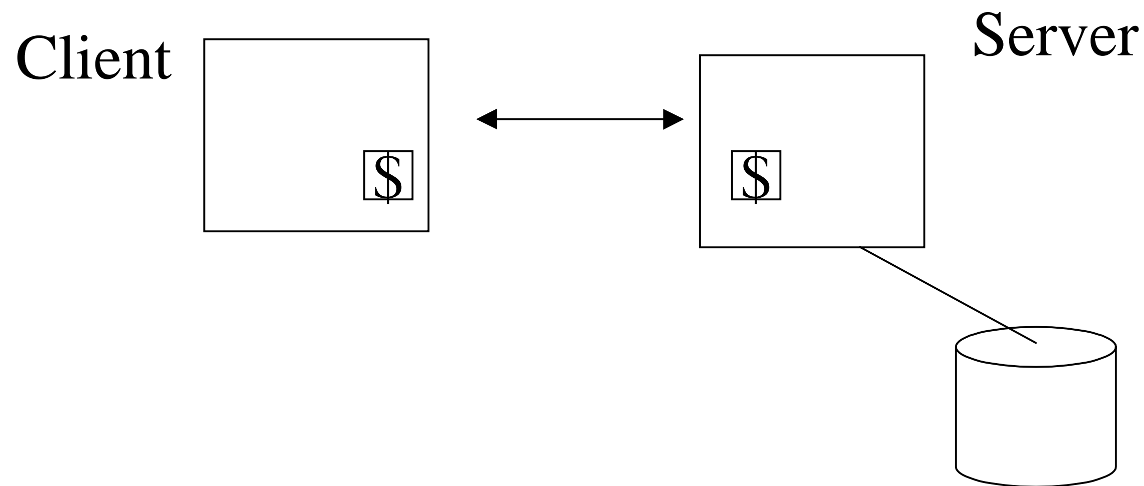
# VFS 2

- If local, translates file handle to internal file id's (in UNIX i-nodes).
- V-node:
  - If file local, reference to file's i-node.
  - If file remote, reference to file handle.
- File handle: uniquely distinguishes file.

| File system id | I-node # | I-node generation # |
| --- | --- | --- |

# NFS Caching

- File contents and attributes.
- Client versus server caching.



# Server Caching

- Read:
  - Same as UNIX FS.
  - Caching of file pages and attributes.
  - Cache replacement uses LRU.
- Write:
  - Write through (as opposed to delayed writes of conventional UNIX FS). Why?
  - [Delayed writes: modified written to disk when buffer space needed, sync operation (every 30 sec), file close].

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Client Caching 1

- Timestamp-based cache invalidation.
- Read:
  - Cached entries have TS with last-modified time.
  - Blocks assumed to be valid for TTL.
    - TTL specified at mount time.
    - Typically 3 sec for files.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Client Caching 2

- Write:
  - Modified pages marked and flushed to server at file close or sync (every 30 sec).
- Consistency?
  - Not always guaranteed!
  - E.g., client modifies file; delay for modification to reach servers + 3-sec window for cache validation from clients sharing file.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Cache Validation

- Validation check performed when:
  - First reference to file after TTL expires.
  - File open or new block fetched from server.
- Done for all files (even if not being shared).
- Expensive!
  - Potentially, every 3 sec get file attributes.
  - If needed invalidate all blocks.
  - Fetch fresh copy when file is next accessed.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# The Sprite File System 1

- Main memory caching on both client and server.
- Write-sharing consistency guarantees.
- Variable size caches.
  - VM and FS negotiate amount of memory needed.
  - According to caching needs, cache size changes.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Sprite 2

- Sprite supports concurrent writes by disabling caching of write-shared files.
  - If file shared, server notifies client that has file open for writing to write modified blocks back to server.
  - Server notifies all client that have file open for read that file is no longer cacheable; clients discard all cached blocks, so access goes throu server.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Sprite 3

- Sprite servers are stateful.
  - Need to keep state about current accesses.
  - Centralized points for cache consistency.
    - Bottleneck?
    - Single point of failure?
- Tradeoff: consistency versus performance.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Andrew

- Distributed computing environment developed at CMU.
- Campus wide computing system.
  - Between 5 and 10K workstations.
  - 1991: |~ 800 ws's, 40 servers.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

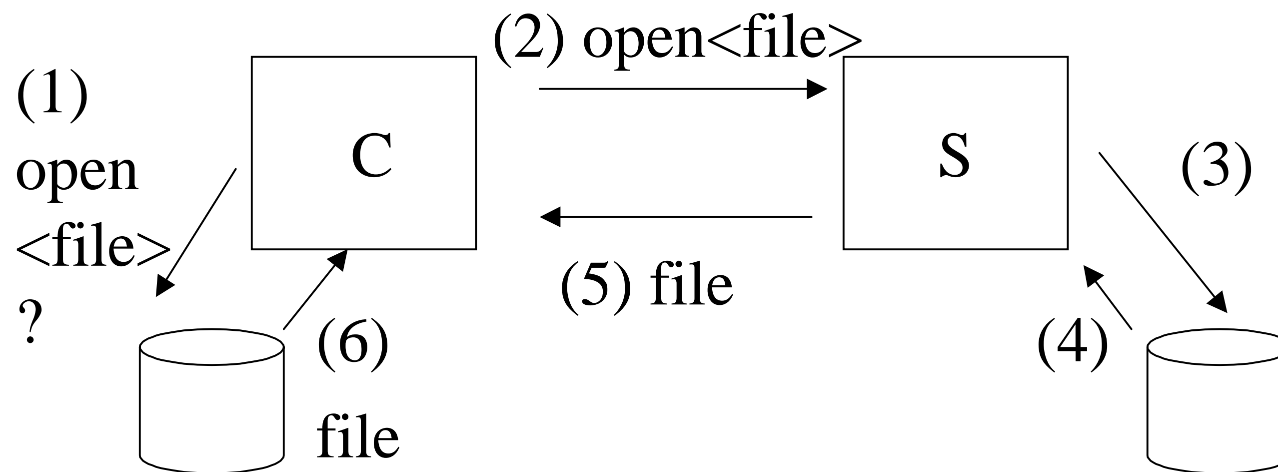
# Andrew FS

- Goals:
  - Information sharing.
  - Scalability.
    - Key strategy: caching of **whole** files at client.
    - Whole file serving
      - Entire file transferred to client.
    - Whole file caching
      - Local copy of file cached on client's local disk.
      - Survive client's reboots and server unavailability.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Whole File Caching

- Local cache contains several most recently used files.




- Subsequent operations on file applied to local copy.
- On close, if file modified, sent back to server.

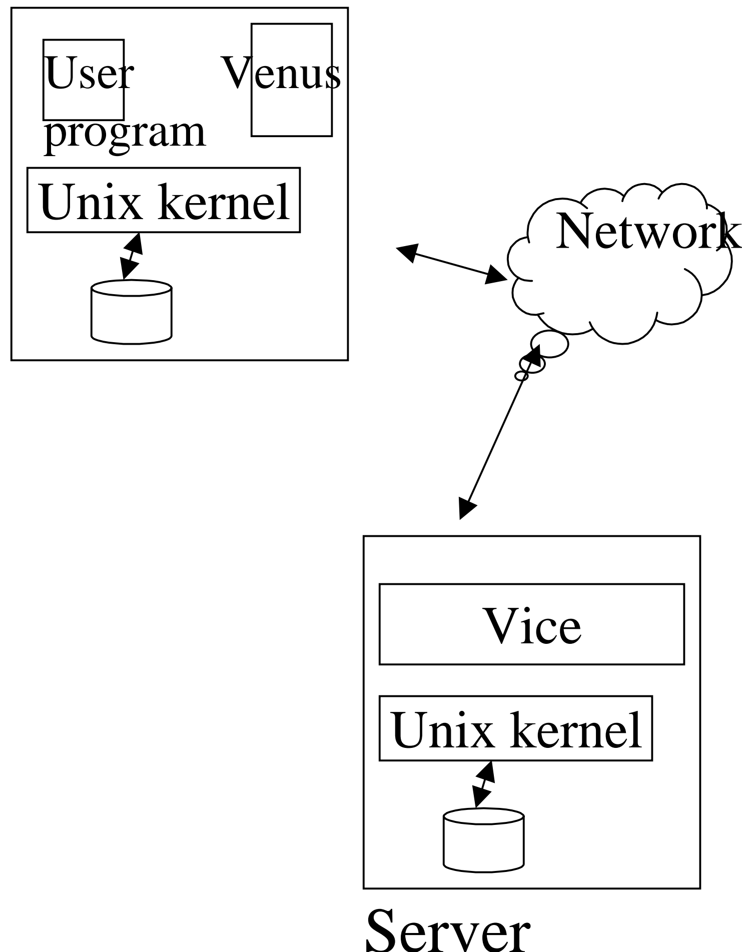
Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Implementation 1

- Network of ws's running Unix BSD 4.3 and Mach.
- Implemented as 2 user-level processes:
  - Vice: runs at each Andrew server.
  - Venus: runs at each Andrew client.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Implementation 2




- Modified BSD 4.3 Unix kernel.
  - At client, intercept file system calls (open, close, etc.) and pass them to Venus when referring to shared files.
- File partition on local disk used as cache.
- Venus manages cache.
  - LRU replacement policy.
  - Cache large enough to hold 100's of average-sized files.

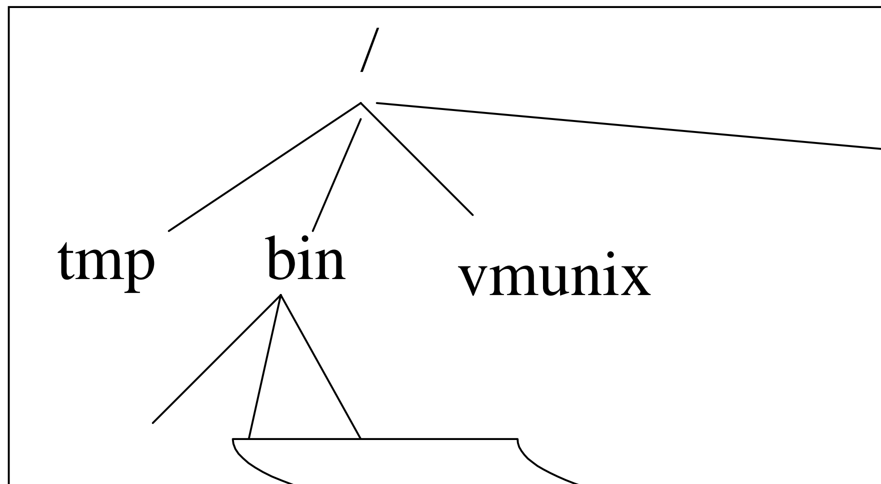
*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

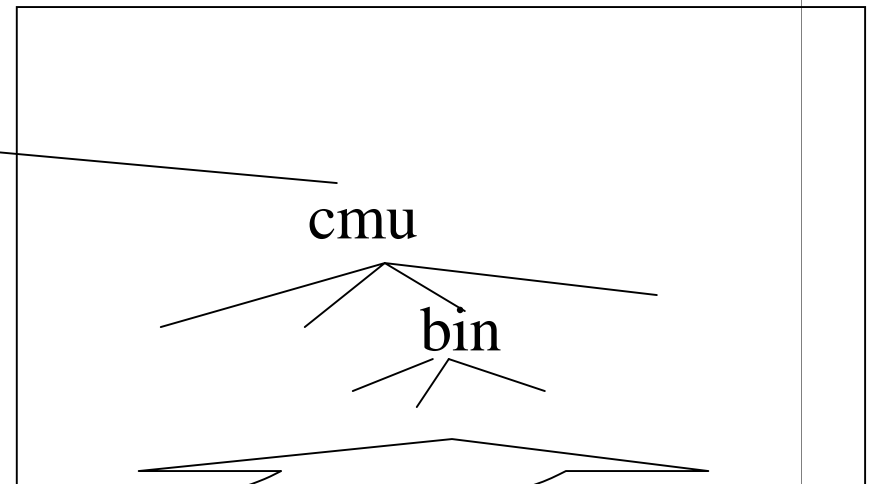
# File Sharing

- Files are *shared* or *local*.
  - Shared files
    - Utilities (/bin, /lib): infrequently updated or files accessed by single user (user's home directory).
    - Stored on servers and cached on clients.
    - Local copies remain valid for long time.
  - Local files
    - Temporary files (/tmp) and files used for start-up.
    - Stored on local machine's disk.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# File Name Space




- Regular UNIX directory hierarchy.
- "cmu" subtree contains shared files.
- Local files stored on local machine.
- Shared files: symbolic links to shared files.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# AFS Caching 1

- AFS-1 uses timestamp-based cache invalidation.
- AFS-2 and + use *callbacks*.
  - When serving file, Vice server promises to notify Venus client when file is modified.
  - Stateless servers?
  - Callback stored with cached file.
    - Valid.
    - Canceled: when client is notified by server that file has been modified.

# AFS Caching 2

- Calbacks implemented using RPC.
- When accessing file, Venus checks if file exists and if callback valid; if canceled, fetches fresh copy from server.
- Failure recovery:
  - When restarting after failure, Venus checks each cached file by sending validation request to server.
  - Also periodic checks in case of communication failures.

# AFS Caching 3

- @ file close time, Venus on client modifying file sends update to Vice server.
- Server updates its own copy and sends callback cancellation to all clients caching file.
- Consistency?
- Concurrent updates?

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# AFS Replication

- Read-only replication.
  - Only read-only files allowed to be replicated at several servers.

# Coda

- Evolved from AFS.
- Goal: constant data availability.
  - Improved replication.
    - Replication of read-write volumes.
  - Disconnected operation: mobility.
    - Extension of AFS's whole file caching mechanism.
- Access to shared file repository (servers) versus relying on local resources when server not available.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Replication in Coda

- Replication unit: file volume (set of files).
- |Set of replicas of file volume: volume storage group (VSG).
- Subset of replicas available to client: AVSG.
  - Different clients, different AVSGs.
  - AVSG membership changes as server availability changes.
  - On write: when file is closed, copies of modified file broadcast to AVSG.

# Optimistic Replication

- Goal is availability!
- Replicated files are allowed to be modified even in the presence of partitions or during disconnected operation.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*

# Disconnected Operation

- AVSG = { }.
- Network/server failures or host on the move.
- Rely on local cache to serve all needed files.
- Loading the cache:
  - User intervention: list of files to be cached.
  - Learning usage patterns over time.
- Upon reconnection, cached copies validated against server's files.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Normal and Disconnected Operation

- During normal operation:
  - Coda behaves like AFS.
  - Cache miss transparent to user; only performance penalty.
  - Load balancing across replicas.
  - Cost: replica consistency + cache consistency.
- Disconnected operation:
  - No replicas are accessible; cache miss prevents further progress; need to load cache before disconnection.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

# Replication and Caching

- Coda integrates server replication and client caching.
  - On cache hit and valid data: Venus does not need to contact server.
  - On cache miss: Venus gets data from an AVSG server, i.e., the preferred server (PS).
    - PS chosen at random or based on proximity, load.
  - Venus also contacts other AVSG servers and collect their versions; if conflict, abort operation; if replicas stale, update them off-line.

*Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE*