

# Challenges of Distributed Systems

CS 451 Lecture 2, 3, 4



Prof. R.K. Joshi  
Dept. of CSE  
IIT Bombay

# Communication

- ◆ Paradigms for interaction  
Message passing vs. shared memory

- ◆ Primary Communication:

Bytes - unix sockets

Typed data PVM, CSP

Function invocations - RPC

Member function invocation RMI, Distributed Processes

Web services - HTTP/SOAP/XML based services

- ◆ Events and Synchronizers:

Signals thread notifications

Monitors - Distributed Processes of Hansen

Mutexes/locks - distributed semaphores, centralized locking

Exceptions - CORBA

# Communication

## ◆ Shared memories

Sharing each others resources (networked RAM)

Sharing Bytes (DSM)

Sharing typed data and computations (Linda)

Sharing objects and services



# Communication

## ◆ Modes of interaction

Synchronous

call/reply

connect/wait for connect

send/receive

Asynchronous

call/callback

send/pickup



# Basic Coordination Techniques

## ◆ Ordering of events

In absence of a common physical clock  
- e.g. Leslie Lamport's Logical Clocks

## ◆ Distributed/Decentralized Decision making based on local view

Each process decides locally

Local decision is consistent with global decision

- All distributed algorithms

◆ e.g. distributed mutual exclusion, leader election, consensus

# Consistency

- ◆ Consistency in presence of concurrent computations
- ◆ Consistency in presence of replication or multiple copies
- ◆ Availability vs. consistency

# Failures

- ◆ Different kinds of failures
- ◆ Decisions in presence of failures
- ◆ Fault tolerant Algorithms/Protocols  
E.g. 3 phase commit protocol,  
Fault tolerant routing, broadcast protocols

# Byzantine Failures

- ◆ Node does not stop working, but it sends a wrong value

- ◆ Can good nodes tolerate failed nodes?

i.e. identify them and isolate them



# Interoperability

- ◆ Cross platform communication

- ◆ Across

  - Type systems

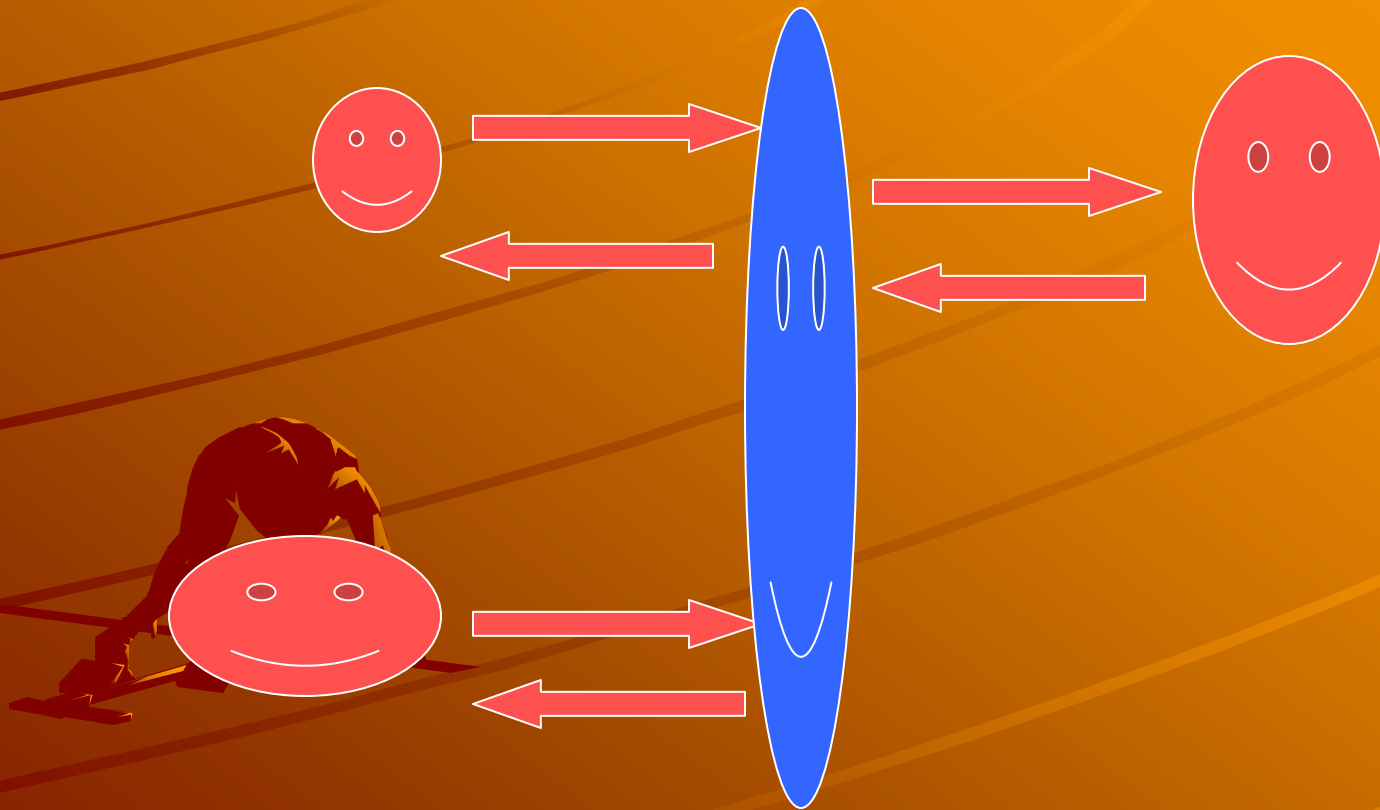
  - Architectures

  - Languages, Execution environments

  - Operating systems

- ◆ Standardization

# A Technique for Interoperability



# Transparency

- ◆ Access Transparency  
hides the difference in data representation and invocation mechanism enables heterogeneous systems to communicate
- ◆ Failure Transparency  
Hides failures and possible recoveries of objects for fault tolerance
- ◆ Location Transparency  
Hides the location information while finding and bind to an object
- ◆ Relocation Transparency  
Masks the changes in the location of an object from its clients

# Transparency

- ◆ Migration Transparency  
Masks the awareness of changes in location of the object from itself and from others
- ◆ Replication Transparency  
Masks the existence of replicated objects
- ◆ Persistence Transparency  
Masks activation and deactivation of objects
- ◆ Transaction Transparency  
Masks coordination of activities to achieve consistency

# Service Orientation

- ◆ Services and contracts
- ◆ clients and servers
- ◆ Stateless vs. statefull
- ◆ Ability to announce services
- ◆ Ability to locate services
- ◆ Services over Web



# Self Stabilization

- ◆ Self correcting systems

- ◆ Local nodes may corrupt their local states

- ◆ System heals itself/stabilizes after a bounded number of iterations/steps

# And ..

- ◆ Synchrony vs. asynchrony
- ◆ Caching and Locality
- ◆ Interconnection architecture  
Broadcast network, point to point, meshes, grids, graphs
- ◆ Failure recovery
- ◆ Migration
- ◆ Handling problems of deadlocks, busy wait etc.
- ◆ Thread safe computing

# Distributed Systems Applications

- ◆ Distributed File systems
- ◆ Networked file systems
- ◆ Distributed Shared memory
- ◆ Distributed Operating systems
- ◆ Distributed Databases
- ◆ Parallel and distributed computing environments
- ◆ Components and distributed services
- ◆ Distributed Web services -- technology yet to mature

# Reference Reading Material for the course

- ◆ Papers from journals, conferences
- ◆ Selected articles from books on distributed systems

◆ Your TA: Sattu

