CS 475: Concurrent & Distributed Systems

Prof. Sanjeev Setia
Computer Science Dept
George Mason University

About this Class

- Focus: designing and writing moderate-sized concurrent and distributed applications
  - Fundamental concepts
  - Multi-threaded and distributed programs
- See syllabus for course learning outcomes
- Prerequisites:
  - CS 367 (Computer Systems & Programming)
  - High level of competence in C/C++ and Java

What you will learn

"*I hear and I forget, I see and I remember, I do and I understand*" – Chinese proverb

- Fundamental concepts in the development of concurrent & distributed software
- Developing Concurrent Programs
  - Threads, semaphores, condition variables, monitors
- Middleware technology for distributed applications
  - Network programming using TCP/IP Sockets
  - RPC/RMI
  - Web Services

Logistics

- Grade: 65% projects, 35% exams
  - Date of midterm (late March/early April) to be announced later
- Four programming assignments
  - Can be done in groups of two
  - First two assignments require the C programming language, third assignment requires Java, fourth – your choice of programming language
  - Assignments will be graded on IT&E Linux server (zeus)
    - If you do your development elsewhere, your responsibility to make sure it runs correctly on zeus
- Occasional homework problems
  - To be done individually

Logistics cont'd

- Online Assignment submission
  - Blackboard (courses.gmu.edu)
  - Grades posted on Blackboard
- Lateness
  - 15% penalty per late day, at most two late days
- "Redo" policy for first three assignments
  - Can resubmit project for improved grade
  - Final grade is calculated by averaging two submissions
- Honor Code

Logistics cont'd

- Office Hrs
  - Tuesdays, 3-5 pm
  - Room 5305, Engineering Bldg
- Email: setia@gmu.edu
- Class Web site: http://www.cs.gmu.edu/~setia/cs475/
- Classroom Policy: Use of laptops/PDAs not permitted

Readings

- No required textbook
- Recommended books
  - Computer Systems & Programming (Bryant & O'Halloran) – used in CS 367
  - Modern Multithreading (Carver and Tai)
  - Foundations of Multithreaded, Parallel and Distributed Programming (Andrews)
  - Operating Systems Concepts (Silbershatz et al) – used in CS 471
  - Distributed Computing: Concepts & Applications (Liu)
  - Distributed Systems: Concepts & Design (Coulouris et al)
- Read class slides & notes

Programming Assignments

- Assignment 1: Shell Lab (CS 367)
  - Topic: Creating and managing concurrent processes
- Assignment 2: Proxy Lab
  - Topic: network programming, multi-threaded programming, synchronization
- Assignment 3: Calendar Lab
  - Topic: RMI, distributed application development
- Assignment 4: Web Services Lab
  - Topic: Web Services programming

Schedule (tentative)

- Concurrent Programming
- Process Synchronization
- Parallel processing on Multicores (introduction to issues)
- Distributed systems concepts
- Sockets; Application-level protocols
- RPC/RMI
- Web Services
- And if we have time.....
  - Peer-to-peer computing
  - Parallel processing on message-passing computers (introduction)

Hardware Architectures

- Uniprocessors
- Shared-memory multiprocessors
- Distributed-memory multicomputers
- Distributed systems

Concurrent Programming

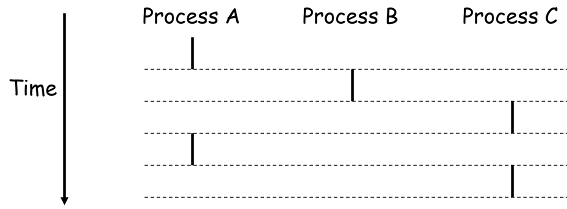
- Process = Address space + one thread of control
- Concurrent program = **multiple threads of control**
  - Multiple single-threaded processes
  - Multi-threaded process

Processes

- Def: A *process* is an instance of a running program.
  - One of the most profound ideas in computer science.
  - Not the same as "program" or "processor"
- Process provides each program with two key abstractions:
  - Logical control flow
    - Each program seems to have exclusive use of the CPU.
  - Private address space
    - Each program seems to have exclusive use of main memory.
- How are these illusions maintained?
  - Process executions interleaved (multitasking)
  - Address spaces managed by virtual memory system

Concurrent Processes

- Two processes *run concurrently* (*are concurrent)* if their flows overlap in time.
- Otherwise, they are *sequential.*
- Examples:
  - Concurrent: A & B, A & C
  - Sequential: B & C

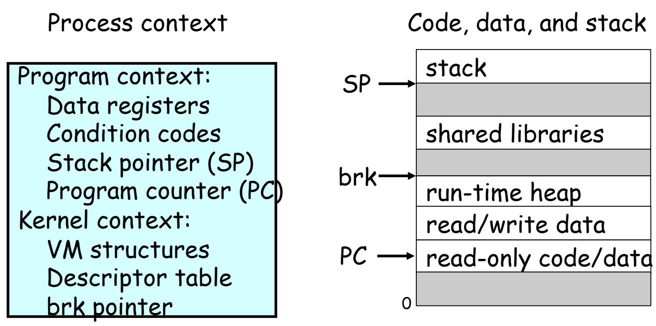


Cooperating Concurrent Processes

- Concurrent processes part of the same application
- Processes “cooperate” on task
- Motivation
  - Support inherent concurrency in application
    - Window systems, web servers
  - Improved performance - can make use of multiple processors

Traditional View of a Process

- Process = process context + code, data, and stack



CS 475 15

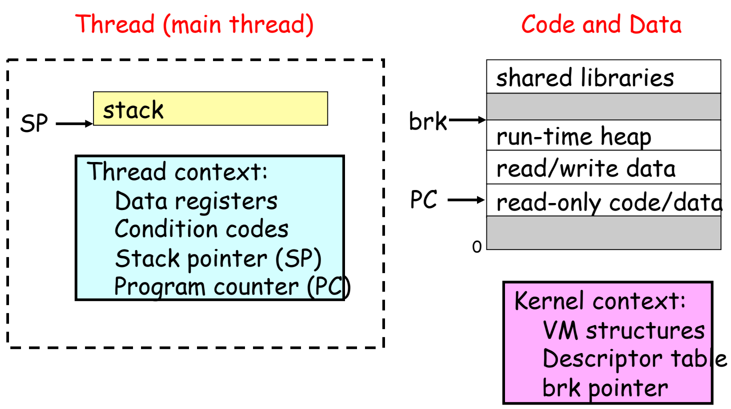
Threads: Motivation

- Traditional processes created and managed by the OS kernel
- Process creation expensive - fork system call in UNIX
- Context switching expensive
- Cooperating processes - no need for memory protection (separate address spaces)

CS 475 16

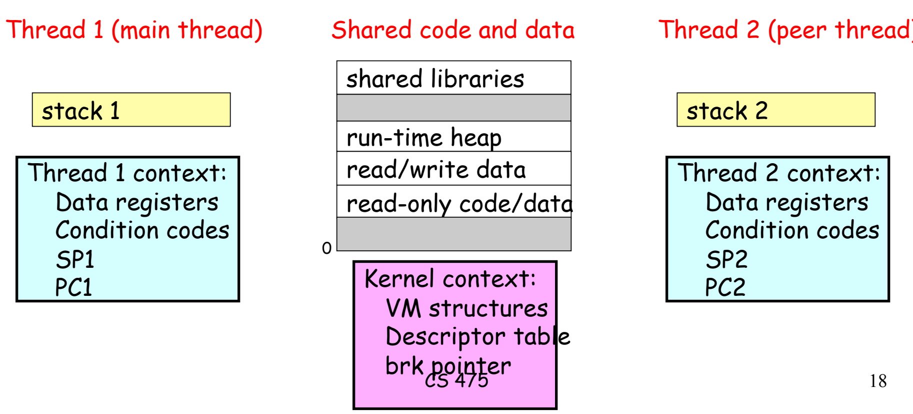
Alternate View of a Process

- Process = thread + code, data, and kernel context



A Process With Multiple Threads

- Multiple threads can be associated with a process
  - Each thread has its own logical control flow (sequence of PC values)
  - Each thread shares the same code, data, and kernel context
  - Each thread has its own thread id (TID)



Threads

- Execute in same address space
  - separate execution stack, share access to code and (global) data
- Smaller creation and context-switch time
- Can exploit fine-grain concurrency

Challenges in multi-threaded/ concurrent programming

- Synchronizing multiple processes/threads
  - Locks
  - Semaphores
  - Monitors
  - Deadlocks
  - Livelocks
- Testing/debugging concurrent applications is a lot harder!

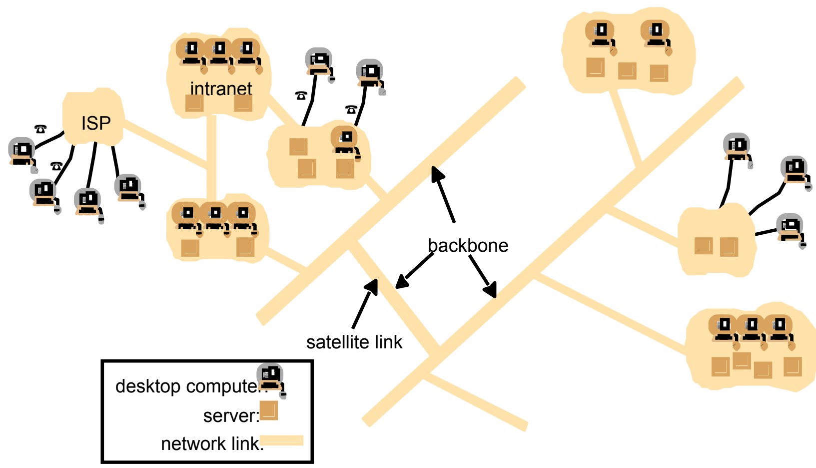
Application classes

- Multi-threaded Programs
  - Processes/Threads on same computer
  - Window systems, Operating systems
- Distributed computing
  - Processes/Threads on separate computers
  - File servers, Web servers
- Parallel computing
  - On same (multiprocessor) or different computers
  - Goal: solve a problem faster or solve a bigger problem in the same time

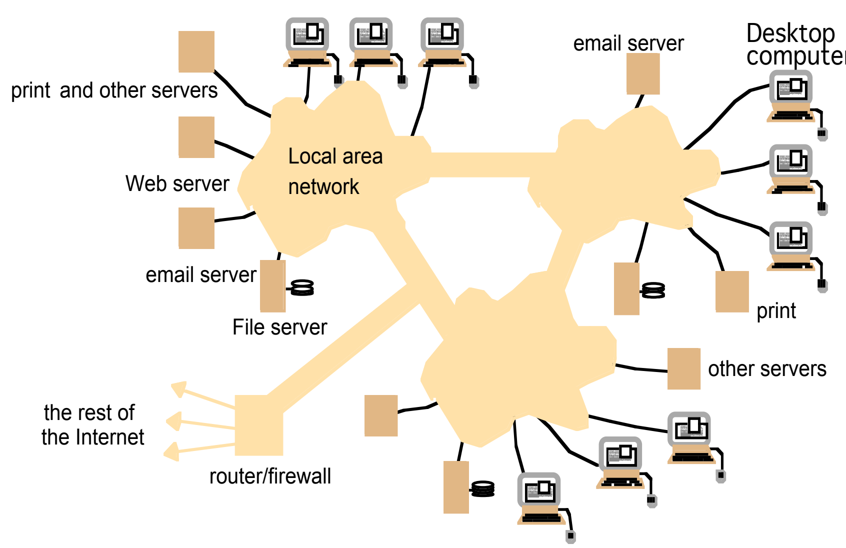
Distributed systems

- "Workgroups"
- ATM (bank) machines
- WWW
- Multimedia conferencing
- Ubiquitous network-connected devices
  - Cell phones, PDAs, sensors
  - "The network is the computer"

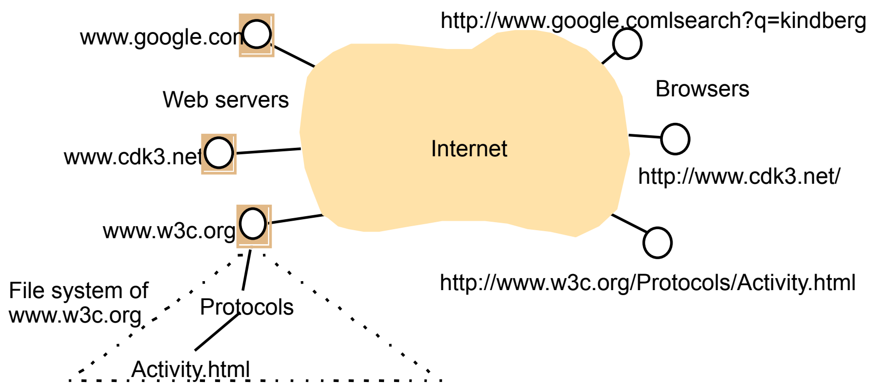
A typical portion of the Internet



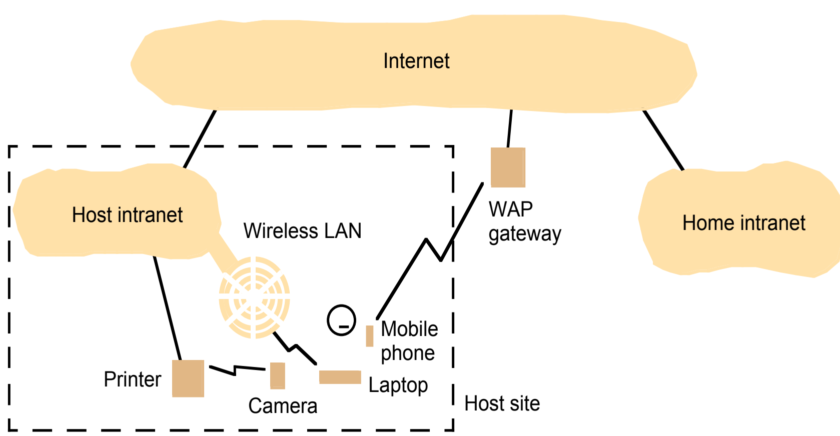
A typical intranet



Web servers and web browsers



Portable and handheld devices in a distributed system



Distributed applications

- Applications that consist of a set of processes that are distributed across a network of machines and work together as an ensemble to solve a common problem
- In the past, mostly "client-server"
  - Resource management centralized at the server
- Peer-to-peer applications represent "truly" distributed applications

CS 475 27

Goals/Benefits

- Resource sharing
- Scalability
- Fault tolerance and availability
- Performance
  - Parallel computing can be considered a subset of distributed computing

CS 475 28

Challenges (Differences from Local Computing)

- Heterogeneity
- Latency
  - Interactions between distributed processes have a higher latency
- Memory Access
  - Remote memory access is not the same as local memory access
    - Local pointers are meaningless outside address space of process

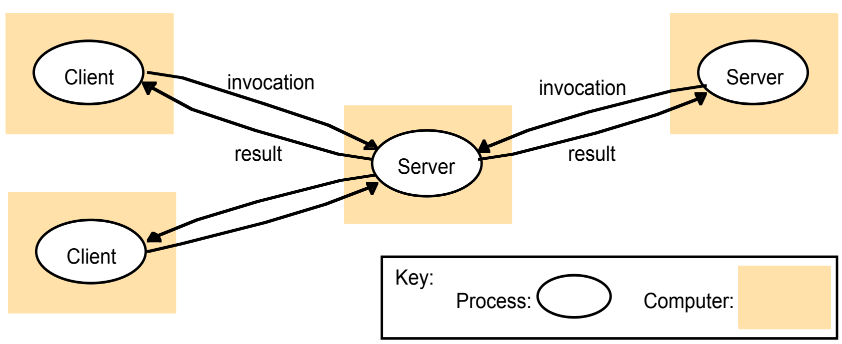
Challenges cont'd

- Synchronization
  - Concurrent interactions the norm
- Partial failure
  - Applications need to adapt gracefully in the face of partial failure
  - Leslie Lamport (a famous computer scientist) once defined a distributed system as "One on which I cannot get any work done because some machine I have never heard of has crashed"

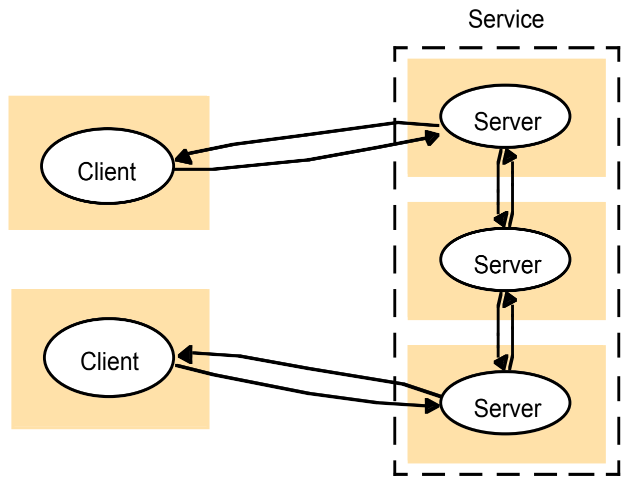
Communication Patterns

- Client-server
- Group-oriented
  - Applications that require reliability
- Function-shipping
  - Java applets

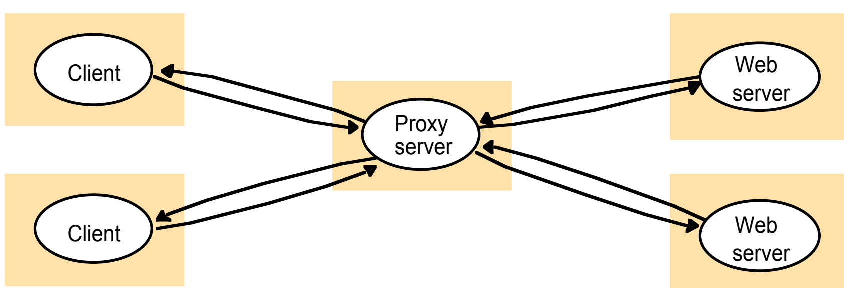
Clients invoke individual servers



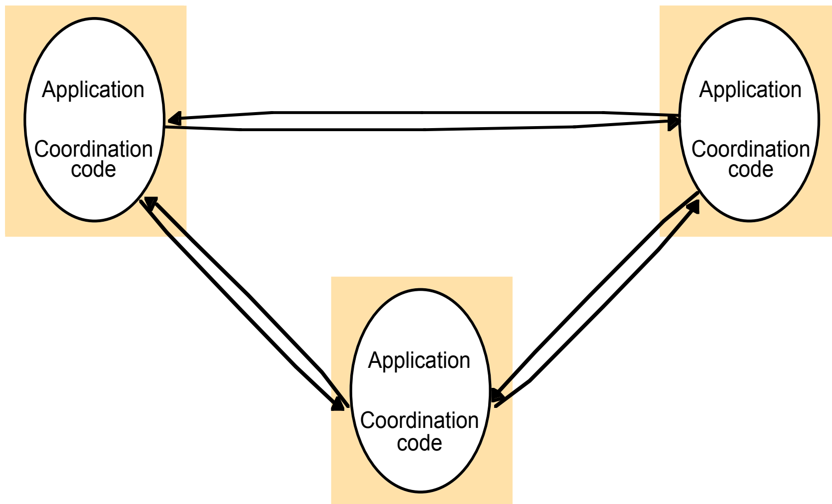
A service provided by multiple servers



Web proxy server

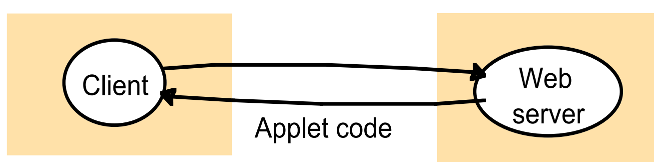


A distributed application based on peer processes



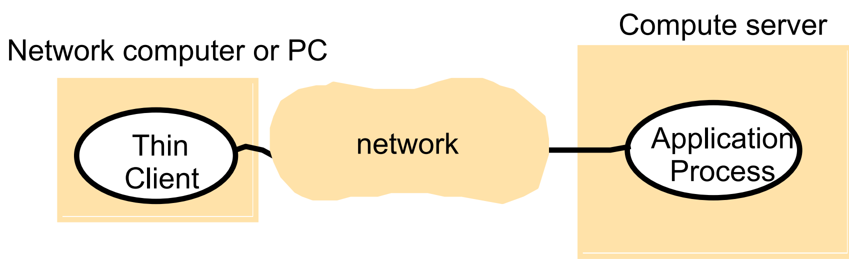
Web applets

a) client request results in the downloading of applet code



b) client interacts with the applet

Thin clients and compute servers



Cloud Computing latest industry buzzword

Software and hardware service layers in distributed systems