Outline

- Architectures and Algorithms
  - Matching the Architecture to the Algorithm
- Distributed Processing
  - Performance
  - Ability
  - Behavioral Principles
- Q&A
Principal Architectural Models

Client-Server

- Client Processes Interacting w/ Server Processes to Access Their Shared Resources Managed in a Centralized Manner
  - E.g., DNS Servers, Web Search Engines

Not Scalable
Principal Architectural Models (Cont’d)

Peer-to-Peer

- Processes Interacting Cooperatively to Access Their Shared Resources Collectively-Managed in a Distributed Fashion

  E.g., Napster
Flynn’s Model for Architectures

- **SISD (Single Instruction Stream, Single Data Stream)**
  - Uniprocessor

- **SIMD (SI, Multiple Data Streams)**
  - Single Instruction Memory & Control Processor w/ Memory per Processor

- **MISD (Multiple Instruction Streams, SD)**

- **MIMD**
  - Centralized Shared-Memory Architectures
  - Distributed-Memory Architectures

  - E.g., UMA, Uniform Memory Access
  - E.g., NUMA, Non-Uniform Memory Access
MIMD Architectures

- E.g., SMP (Symmetric Multiprocessing), – Problem: Poor Scalability
- E.g., MPP (Massively Parallel Processing), – Problem: Explicit Data Distribution
- E.g., SMP (Symmetric Multiprocessing), – Problem: Poor Scalability
- E.g., NUMA
Characterizing the Algorithm [Kleinrock85]

To Exploit Its Potential for Concurrency at Different Granularity Levels:
- Job
- Task
- Process
- Instruction
- Register Transfer
- Logic Device

E.g., by Using a Graph Model
Matching the Architecture to the Algorithm [Kleinrock85]

Considerations Regarding the Graph Model

- Partitioning
  - Deciding granularity levels
  - Grouping

- Scheduling
  - Assigning processors and memory modules

- Memory Access

- Interprocessor Communication

- Synchronization
  - Preserving dependency
Matching the Architecture to the Algorithm [Kleinrock85]

- Balance and Tradeoff among Communication, Processing, and Storage
  - Trading Processing for Communication
    - E.g., data compression prior to transmission
  - Trading Storage for Processing
    - E.g., storing a list of computational results
  - Trading Storage for Communication
    - E.g., storing data from a previous communication

- Failure Detection & Recovery
Parallel-Processing Performance

- **Speedup (S)**
  - Serial Computation Time $T_s$ over Parallel Execution Time $T_p$ ($S = T_s/T_p$)
    - $T_p = T_{COMP_p} + T_{COMM_p}$
    - $T_{COMP_p} = T_s (1-A)/P + T_s A$, where $A$ is the serial portion
      - E.g., If $T_s = 100$, $A=0.2$, & $P=10$, then $T_{COMP_p} = 28$
    - $T_p = T_s (1-A)/P + T_s A$, assuming that $T_{COMM_p} = 0$
    - $S = T_s/T_p = T_s / (T_{COMP_p} + T_{COMM_p}) = 1 / \{A + (1-A)/P\}$, assuming that $T_{COMM_p} = 0$
      - $1 \leq S \leq P$
Ability of Distributed Processing 
Possibly Producing the Best Result

- Greedy Solutions with Unexpected Results: E.g., Goore Game [Kleinrock85] (Repeating Actions Preferentially by Trial and Error)
  - Many Players Being Asked to Vote YES or NO in Each Interval
    - Acting w/ sufficient memory in a probabilistic fashion
    - W/ each unaware of the others
  - Referee Rewarding Each Member Independently w/ a Probability Given by $f(p)$ Unknown to It
    - $p$: fraction of the player set that votes YES

Exactly 20% of the Players Will Vote YES with Probability 1

We May Be Able to Explain How the Colony of Ants Perform Its Tasks
Ability of Distributed Processing (Cont’d)

- Possibly Producing a Globally Suboptimal Solution
  - Most Greedy Solutions: E.g., Prisoner’s Dilemma (Predicting the Behavior of the Other Players)
    - Two Men Being Held in Separate Cells for a Crime They Did
    - District Attorney w/ Hard Evidence Just for One-Year Penalty
      - Making the following offer to each prisoner:
        - Free if confessing w/ your partner remaining silent
        - In jail for 20 years in the opposite case
        - Five years in jail for each if both confess

Dilemma: Each Is Tempted to Confess, But If Both Confess, It Will Be Worse
Behavioral Principles

[Kleinrock85]

- Developing innovative architectures for parallel processing
- Providing better languages and algorithms for specification of concurrency
- More expressive models of computation
- Matching the architecture to the algorithm
- Understanding the trade-off among communication, processing, and storage
- Evaluation of the speedup factor for classes of algorithms and architectures
- Evaluation of the cost-effectiveness of distributed-processing networks
- Study of distributed algorithms in networks
- Investigation of how loosely coupled self-organizing automata can demonstrate expedient behavior
- Development of a macroscopic theory of distributed systems
- Understanding how to average over algorithms, architectures, and topologies to provide meaningful measures of system performance