Algorithms in Nature

Distributed computing

Example Distributed systems

- Internet
- ATM (bank) machines
- Intranets/Workgroups
- Computing landscape will soon consist of ubiquitous network-connected devices
  - “The network is the computer”

Computers in a Distributed System

- Workstations: computers used by end-users to perform computing
- Server machines: computers which provide resources and services
- Mobile Devices: handheld computers connected to the system via a wireless communication link.
- ...

Goals/Benefits

- Resource sharing
- Scalability
- Fault tolerance / Robustness
- Performance / Speed
  - Parallel computing can be considered a subset of distributed computing

Challenges(Differences from Local Computing)

- Heterogeneity
- Latency
- Remote Memory vs Local Memory
- Synchronization
  - Concurrent interactions the norm
- Partial failure
  - Applications need to adapt gracefully in the face of partial failure

Challenges cont'd

- Security
  - Denial of service attacks
  - Mobile code
- Scalability
- Transparency

Scalability

- Key to scalability: decentralized algorithms and data structures
  - No machine has complete information about the state of the system
  - Machines make decisions based on locally available information
  - Failure of one machine does not ruin the algorithm
  - There is no implicit assumption that a global clock exists

Fundamental/Abstract Models

- A fundamental model captures the essential ingredients that we need to consider to understand and reason about a system's behavior
- Addresses the following questions
  - What are the main entities in the system?
  - How do they interact?
  - What are the characteristics that affect their collective and individual behavior?

Fundamental/Abstract Models

- Three issues to consider in models
  - Interaction model
    - Reflects the assumptions about the processes and the communication channels in the distributed system
  - Failure model
    - Distinguish between the types of failures of the processes and the communication channels
  - Security Model
    - Assumptions about the principals and the adversary

BASIC COMMUNICATION PRIMITIVE: MESSAGE PASSING

Paradigm:

- Send message to destination
- Receive message from origin

Nice property: can make distribution transparent, since it does not matter whether destination is at a local computer or at a remote one (except for failures).

Clean framework: "Paradigms for Process Interaction in Distributed Programs," G. R. Andrews, ACM Computing Surveys 23:1 (March 1991) pp. 49-90.

BASIC COMMUNICATION PRIMITIVE: Shared memory

Paradigm:

- All processes use the same memory space
- Need to overcome concurrent access to the same location

Both shared memory and message passing can be emulated by the other paradigm and so any algorithm that works for one would work for the other. However, it is easier to emulate shared memory using message passing than the other way around.

BLOCKING (SYNCHRONOUS) VS. NON-BLOCKING (ASYNCHRONOUS) COMMUNICATION

For sender: Should the sender wait for the receiver to receive a message or not?

For receiver: When arriving at a reception point and there is no message waiting, should the receiver wait or proceed? Blocking receive is normal (i.e., receiver waits).

Interaction Models

- Synchronous Distributed Systems: a system in which the following bounds are defined
  - The time to execute each step of a process has an upper and lower bound
  - Each message transmitted over a channel is received within a known bounded delay
  - Each process has a local clock whose drift rate from real time has a known bound
- Asynchronous distributed system
  - Each step of a process can take an arbitrary time
  - Message delivery time is arbitrary
  - Clock drift rates are arbitrary
- Some implications
  - In a synchronous system, timeouts can be used to detect failures
  - Impossible to detect failures or "reach agreement" in an asynchronous system

Interaction Models

- Synchronous Distributed Systems: a system in which the following bounds are defined
  - The time to execute each step of a process has an upper and lower bound
  - Each … a know…
  - Each … ne has a kn…
- Async…

Several computational problems can only be (provably) solved in a synchronous setting. However, asynchronous models are much more realistic.

  - Each step of a process can take an arbitrary time
  - Message delivery time is arbitrary
  - Clock drift rates are arbitrary
- Some implications
  - In a synchronous system, timeouts can be used to detect failures
  - Impossible to detect failures or “reach agreement” in an asynchronous system

Omission and arbitrary failures

Class of failure	*Affects*	*Description*
Crash	Process	Process halts and remains halted. Other processes may not be able to detect this state.
Omission	Channel	A message inserted in an outgoing message buffer never arrives at the other end's incoming message buffer.
Arbitrary (Byzantine)	Process or channel	Process/channel exhibits arbitrary behaviour: it may send/transmit arbitrary messages at arbitrary times, commit omissions; a process may stop or take an incorrect step.

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” computing represents a movement towards more “truly” distributed applications