Intro to Distributed Systems

Hank Levy

Distributed Systems

- Nearly all systems today are distributed in some way, e.g.:
  - they use email
  - they access files over a network
  - they access printers over a network
  - they are backed up over a network
  - they share other physical or logical resources
  - soon: they receive video, audio, etc.

Why use distributed systems?

- Distributed systems are now a requirement:
  - economics dictate that we buy small computers
  - everyone needs to communicate
  - we need to share physical devices (printers) as well as information (files, etc.)
  - many applications are by their nature distributed (bank teller machines, airline reservations, ticket purchasing)
  - in the future, to solve the largest problems, we will need to get large collections of small machines to cooperate together (parallel programming)

Loosely-Coupled Systems

- Most distributed systems are "loosely-coupled."
- Each CPU runs an independent autonomous OS.
- Hosts communicate through message passing.
- Computer don’t really trust each other.
- Some resources are shared, but most are not.
- The system may look differently from different hosts.
- Typically, communication times are long.

What is a distributed system?

- There are several levels of distribution.
- Earliest systems used simple explicit network programs:
  - FTP: file transfer program
  - Telnet (rlogin): remote login program
  - mail
  - remote job entry (rsh): run jobs remotely
- Each system was a completely autonomous independent system, connected to others on the network

Closely-Coupled Systems

- A distributed system becomes more "closely coupled" as it:
  - appears more uniform in nature
  - runs a "single" operating system
  - has a single security domain
  - shares all logical resources (e.g., files)
  - shares all physical resources (CPUs, memory, disks, printers, etc.)
- In the limit, a distributed system looks to the user as if it were a centralized timesharing system, except that it’s constructed out of a distributed collection of hardware and software components.
Tightly-Coupled Systems

- A “tightly-coupled” system usually refers to a multiprocessor.
  - Runs a single copy of the OS with a single job queue
  - has a single address space
  - usually has a single bus or backplane to which all processors and memories are connected
  - has very low communication latency
  - processors communicate through shared memory

Some Issues in Distributed Systems

- Transparency (how visible is the distribution)
- Security
- Reliability
- Performance
- Scalability
- Programming models
- Communications models

Transparency

- In a true distributed system with transparency:
  - It would appear as a single system
  - different modes would be invisible
  - jobs would migrate automatically from node to node
  - a job on one node would be able to use memory on another

Distribution and the OS

- There are various issues that the OS must deal with:
  - how to provide efficient network communication
  - what protocols to use
  - what is the application interface to remote apps (although this might be a language issue)
  - protection of distributed resources

The Network

- There are various network technologies that can be used to interconnect nodes.
- In general, Local Area Networks (LANs) are used to connect hosts within a building. Wide Area Networks (WANs) are used across the country or planet.
- We are at an interesting point, as network technology is about to see an order-of-magnitude performance increase. This will have a huge impact on the kinds of systems we can build.

Issues in Networking

- Routing
- Bandwidth and contention
- Latency
- Reliability
- Efficiency
- Cost
- Scalability
Network Topologies

Point to Point

Ring

Broadcast

Star

Tree

Switch

Two ways to handle networking

- **Circuit Switching**
  - what you get when you make a phone call
  - good when you require constant bit rate
  - good for reserving bandwidth (refuse connection if bandwidth not available)

- **Packet Switching**
  - what you get when you send a bunch of letters
  - network bandwidth consumed only when sending
  - packets are routed independently
  - packetizing may reduce delays (using parallelism)

Packet switching is preferable for data communications

- From the perspective of the network
  - but may not be preferable for some application

- Applications are bursty
  - variable amounts of info at irregular intervals
  - a diskless workstation: needs all bandwidth to transfer a page, so can’t reserve it
  - circuit switching may have high cost to set up connection
  - maintaining the connection may waste bandwidth if connection is used infrequently

New Applications

- Video and Voice may be different (more like phone system)
  - but with data compression, makes circuit switching less attractive:
    - compressed video generates a variable bit rate signal
    - signal needs to be transported within a certain max. delay, but bandwidth needed is variable

- New applications will be very bursty and will require guarantees about latency.

Messages

- At a low level, network communication is via messages.
- A message is simply a typed byte string passed between two levels of the system (e.g., OS to OS, app to app).
- A message usually contains a header, indicating what kind of information it contains, and some data.
- What the message “means,” i.e., how to interpret the bytes in the message, is an agreement between the two communicating parties (the protocol).

The anatomy of a message

- The msg data may contain a header and some data for another level of communication, and so on.
The OSI Model

- The Open Systems Interconnect model is a standard way of understanding the conceptual layers of network communication.
- This is a model, nobody builds systems like this.
- Each level provides certain functions and guarantees, and communicates with the same level on remote notes.
- A message is generated at the highest level, and is passed down the levels, encapsulated by lower levels, until it is sent over the wire.
- On the destination, it makes its way up the layers, until the high-level msg reaches its high-level destination.

OSI Levels

- Physical Layer: electrical details of bits on the wire
- Data Link: sending "frames" of bits and error detection
- Network Layer: routing packets to the destination
- Transport Layer: reliable transmission of messages, disassembly/assembly, ordering, retransmission of lost packets
- Session Layer: really part of transport, typ. Not impl.
- Presentation Layer: data representation in the message
- Application: high-level protocols (mail, ftp, etc.)

Addressing and Packet Format

- Every network card has a unique address in HARDWARE.
- The "Data" segment contains higher level protocol information.
  - Which protocol is this packet destined for?
  - Which process is the packet destined for?
  - Which packet is this in a sequence of packets?
  - What kind of packet is this?
- This is the stuff of the OSI reference model.

Ethernet packet dispatching

- An incoming packet comes into the ethernet controller.
- The ethernet controller reads it off the network into a buffer.
- It interrupts the CPU.
- A network interrupt handler reads the packet out of the controller into memory.
- A dispatch routine looks at the Data part and hands it to a higher level protocol
- The higher level protocol copies it out into user space.
- A program manipulates the data.
- The output path is similar.
- Consider what happens when you send mail.

Example: Mail

- Mail Composition And Display
- Mail Transport Layer
- Network Transport Layer
- Link Layer
- Network
- Physical

- Start (7 bytes)
- Destination (6)
- Source (6)
- Length (2)
- Mag Data (1500)
- Checksum (4)
Routing

- Moving packets from one network to another.
- Routers run their own address distribution protocol to ensure connectivity
  - decisions based on a distance metric

Finally

- TCP/IP (Transmission Control Protocol/Internet Protocol) provides reliable, ordered byte streams between pairs of processes
- UDP/IP (User Datagram Protocol) provides unreliable, unordered messages between pairs of processes
- A network interface delivers packets to the operating system.
- The operating system delivers messages to an application according to the destination specified in the packet
- The rest is all about distributed programming!