Lecture 22: April 11

Lecturer: Prashant Shenoy *TA: Vimal Mathew & Tim Wood*

22.1 Distributed Systems

A distributed system is one composed of multiple physically separated processors that are connected by communication links. A distributed system is typically considered **loosely coupled** because each of the processing nodes have their own memory and OS, and only periodically communicate in order to maintain synchronization. In contrast, a **tightly coupled** system is one where multiple processors run under the control of a single operating system. The advantage of distributed systems is that they allow multiple systems to pool their resources. This kind of **resource sharing** can be as simple as having multiple PCs share a printer, or as complicated as a distributed set of machines all providing a shared storage system. Having the additional resources of a distributed system leads to **computation speedup**. Ideally, having n processors should give you a factor of n speedup, although in practice this is very difficult to achieve due to communication overheads and the need to cleanly decompose a problem into smaller subproblems.

Distributed systems can also provide higher **reliability** by using replication to points of failure. This allows a distributed system to keep working even if one or more nodes crash. However, the system needs to be carefully configured in order to prevent a central server crashing affecting all other nodes in the system. Finally, distributed systems form the basis for **communication** systems such as e-mail and websites. This allows software on machines all over the world to communicate and perform tasks.

While each node in a distributed system typically runs its own operating system, the distributed system as a whole also must provide services similar to an OS. For example, the entire distributed system needs a way to manage resources, provide communication, ensure security and reliability within the system, and provide high performance even as the system scales to a large number of nodes.

22.2 Networks

Networks are responsible for providing efficient, correct, and robust message passing between two separate nodes. A **Local Area Network (LAN)** is used to connect nodes within a small area (e.g. within a single building), and is designed to provide high speed communication. The nodes in a LAN may communicate wirelessly, or they may all be connected via cables such as coaxial or fiber optics. The bandwidth provided by a LAN can range from 10Mbps-100Mb/s in a typical small LAN up to 1000Mb/s or even 10Gb/s in a well provisioned network. In contrast, a **Wide Area Network (WAN)** is designed to work over longer distances, and thus typically provide slower, less reliable links. WAN connections may be run over telephone lines, microwave links, or even satellite channels. The bandwidth provided in WAN links to a home or small business is much lower than in a LAN—a T1 link for a small business provides only 1.544Mb/s transfer speeds.

22.2.1 Point-to-Point Topologies

The topology of a network is defined by how the links between nodes are created. In a **fully connected** network, all nodes are connected to all other nodes. This provides the benefit that any node can reach any other in only "one hop", and that the system is resilient to failures since one node crashing will not cause any other node to become disconnected from the rest. However, this can become expensive when there is a large number of nodes, and is not practical for WAN environments where there can be huge numbers of nodes spread across wide distances.

A **partially connected** network loosens these requirements by only using links between a subset of the nodes. As a result, a message sent between two nodes may need to be directed through several other nodes along the path, requiring some sort of routing algorithm for coordination and increasing transmission times. However, since this is significantly less expensive than a fully connected network, it is often used for WANs.

Another network topology is a **tree structure**. In this case, a root node is created, and all others connected to its children or their descendants. This provides a simple structure that often matches the logical hierarchy of an enterprise, but it can lead to slow performance since messages may have to pass up through multiple nodes to find a common ancestor for their intended recipient. Trees are also not resilient to failure, since one node failing causes all of its descendants to become disconnected from the root.

A **star** topology makes all nodes be directly connected to a single centralized node. This uses much fewer links than a fully connected network, but still allows all nodes to communicate within two hops. However, the central node is a single point of failure and can lead to the full network becoming disconnected.

22.2.2 Ring & Bus Topologies

Another class of topologies is a ring network—these topologies are very simple, but their restrictive structure can produce more predictable behavior. In a simple **one directional ring**, a node can only send messages in one direction, leading to a maximum message delay of $n - 1$ hops in a network with n nodes. Allowing messages to travel in both directions reduces the maximum delay to $n/2$ hops, but the system can still only tolerate one failure before nodes become disconnected.

A ring network can be created with additional links in order to gradually increase the level of connectivity. A **double connected ring** is one where each node is connected to its immediate neighbors, as well as the nodes two hops away. This allows messages to hop through the network more quickly, giving a maximum delay of $n/4$ hops and increasing the resilience to failure. Further increasing the number of links provides better performance and reliability at greater expense. Note that a ring network with links to all nodes is simply a fully connected point-to-point network.

A **bus network** relies on special links that can be shared by multiple nodes. For example, multiple components within a single computer communicating over the shared system bus can be considered a form of distributed system. A **linear bus** simply has multiple nodes connected to one bus that supports multiple access. This is inexpensive, but nodes may need to coordinate to ensure that only a single node uses the bus at any one time. Ethernet LAN links use this structure.

22.2.3 Communication Protocols

A communication protocol is a set of rules that are agreed upon by both nodes that want to send messages to each other. These rules define things such as the order that nodes should talk, the format that data should be sent, etc. The rules are typically implemented as part of a **protocol stack**—a set of software layers that

convert application level requests into the format required by the network, transmit the data between nodes, and finally converts it back into messages that the destination application can understand.

The International Standards Organization Open Systems Interconnect **(ISO/OSI)** model defines the standard network protocol stack used in modern operating systems. It uses a layered architecture with a set of layers that communicate only with those immediately above or below them. The layers, from highest to lowest are:

- Application Layer: contains the user processes which desire to send messages over the network, e.g. web browsers or IM clients.
- Presentation Layer: performs data conversion (e.g. big/little endian format) since the destination and source nodes may not be running the same architecture.
- Session Layer: includes libraries that implement the communication strategy such as Remote Procedure Calls (RPC) or simple message passing.
- Transport Layer: ensures reliable, end-to-end communication between two nodes. This layer is provided within the operating system, and the OS may support multiple different policies (e.g. TCP and UDP).
- Network Layer: performs routing and congestion control to ensure that messages can reach their destination and that resources are used efficiently within the network. Usually implemented within the OS.
- Data Link Control Layer: ensures reliable point-to-point communication of packets even over unreliable network paths. Can be implemented either within the network card hardware or in software.
- Physical Layer: controls the electrical or optical signaling across a "wire", e.g. the actual transmission of bits between two nodes. Implemented in the network card hardware.

22.2.4 TCP/IP Protocol Stack

The Transmission Control Protocol/Internet Protocol (TCP/IP) is the most common protocol stack used in network communication. In practice, many systems implement only the TCP/IP stack which contains four layers, rather than the full seven layer model proposed by ISO/OSI.

TCP/IP defines a suite of protocols to be used for the Transport Layer, that specifies how messages should be sent between hosts over the full end-to-end path. **TCP** is the primary transport protocol used on the Internet today because it provides a **reliable** transfer mechanism. TCP enforces the rules that any packet sent will eventually be received by the destination (assuming it does not become disconnected) and that packets will arrive in order. A second protocol, **UDP** can be used instead of TCP, but it is an **unreliable** protocol—no guarantee is given that packets sent will arrive or arrive in any particular order. While it may seem like TCP would always be preferable to UDP, TCP must give up some performance in order to make its guarantees about reliability. As a result, UDP is still often used for applications which can deal with some amount of packet loss but require low latency, e.g. audio streaming or video games. The TCP/IP stack also defines **IP**, a protocol which is used as a lower layer for transporting packets underneath of either TCP or UDP.

A **packet** is the minimal unit of transfer in a TCP or UDP network flow. When an application desires to send a message, the TCP/IP stack will split it up into multiple fixed size packets. Each packet acts as a self contained message that knows its source, destination, and ordering. This means that while a destination host may receive a set of packets out of order, it is able to rearrange them into the proper order. When packets are received at the destination, they are combined to rebuild the original message.