CSCI 251: Concepts of Parallel and Distributed Systems

Alvin Lin

December 4th, 2017

Networking Architectures

Topics:

1. Principles of network applications
2. Web and HTTP
3. Email: SMTP, POP3, IMAP
4. DNS
5. P2P applications
6. Video streaming and CDNs
7. Socket programming with UDP and TCP

Networked Applications

Creating a networked app requires writing a program that runs on different end systems and communicate over a network. Network core devices do not need this software since they do not run the end user applications.

Client-server Architecture

A client server architecture involves an always-on host with a permanent IP address and uses data centers for scaling. Clients communicate with the server, may be intermittently connected, may have dynamic IP addresses, and do not communicate directly with each other.

P2P architecture

Peer to peer architecture does not have an always on server. An arbitrary number of end streams directly communicate with each other. Peers request and provide service to other peers. This architecture has the property of self-scalability, new peers bring new service capacity and meet the new service demands. The peers may be intermittently connected and change IP addresses.

Processes

Processes are programs running within a host. Within the same host, two processes communicate using inter-process communication. Processes in different hosts communicate by exchanging messages. Generally, the client process is the one that initiates communication while the server process waits to be contacted. Applications with P2P architectures have client and server processes.

Sockets

A process sends and receives messages to and from its socket. A socket is analogous to a door: the sending process shoves a message out the door and relies on the underlying transport infrastructure on the other side of the door to deliver the message.

Addressing Processes

To receive messages, a process must have an identifier. A host device has a unique 32-bit IP address. Many processes can be running on the same host, so the identifier includes both the IP address and port numbers associated with the process on the host.

Messages

The application layer protocol defines the types of messages exchanged, the message syntax, message semantics, and rules for when and how processes send and respond to messages. Open protocols like those defined in the IETF's RFCs, HTTP, and SMTP allow for application interoperability.

Transport Services

- Data integrity: some applications (file transfer, web transactions) require 100% reliable data transfer. Audio streaming applications, however, can tolerate some loss.
- Timing: some applications (Internet telephony, multiplayer games) require low delays in order to be effective.
- Throughput: some apps (multimedia) require a higher minimum throughput in order to be effective. Other apps may make use of whatever throughput they get.
- Security: some applications require encryption and data integrity.

TCP

- reliable transport between sending and receiving process
- flow control: sending won't overwhelm receiver
- congestion control: throttle sender when the network is overloaded
- does not provide timing, minimum throughput guarantee, and security
- connection-oriented: setup required between client and server processes
- bottlenecked by acknowledgement

For security, applications use SSL libraries that talk to the TCP layer. This provides an encrypted TCP connection, data integrity, and end-point authentication.

UDP

- unreliable data transfer between sending and receiving process
- does not provide reliability, flow control, congestion control, timing, throughput guarantee, security, or connection setup

HTTP/Web

Refer to the slides on MyCourses.

Reminders

Work on Project 2.
Professor Mohan Kumar:
mjkvcs@rit.edu
https://cs.rit.edu/~mjk

Rahul Dashora (TA):
rd5476@mail.rit.edu

You can find all my notes at http://omgimanerd.tech/notes. If you have any questions, comments, or concerns, please contact me at alvin@omgimanerd.tech