CS 686: Special Topics in Big Data

Network Design

Lecture 3
8/28/17

Today's Agenda

- Upcoming assignments
- Distributed network designs
- A brief history of P2P evolution
- Designing our own storage network

Today's Agenda

- **Upcoming assignments**
- Distributed network designs
- A brief history of P2P evolution
- Designing our own storage network

Upcoming Assignments

- First paper evaluation: due Thursday (8/31)
- First paper discussion: Friday in class (9/1)
- Project 1: description posted tomorrow (8/29)
  - **Tentative** due date: 9/27
- Second paper evaluation: 9/14

Evaluation/Discussion Grading

- I'll use a simplified grading scale for these assignments:

Grade	Points
Check +	100
Check	90
Check -	80

Today's Agenda

- Upcoming assignments
- **Distributed network designs**
- A brief history of P2P evolution
- Designing our own storage network

Messaging

- Distributed systems do not have **shared memory**
- Instead, we rely on messages for exchanging state between nodes
  - **Message** – packet of information with a well-defined format
  - **State** – events occur that mutate the state space of the system
  - **Node** – one participant (machine) of the distributed system

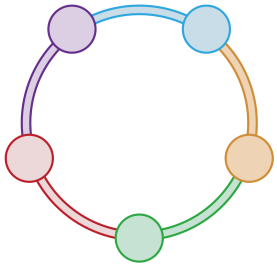
Sending a Message

1. Information to be shared is constructed in memory on **Node A**
2. The data is encapsulated and serialized for transfer
   - Well-defined **wire format**
3. The message is sent across the network
4. **Node B** receives the payload, reconstructs the message, and applies the information/event to its own state space

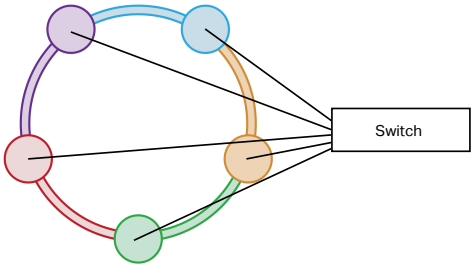
Designing the Network

- The messages exchanged between nodes are influenced by the underlying network design
  - A P2P network operates differently than set of NFS servers
  - Join notifications, file locking, periodic heartbeats
- Note that this generally doesn't mean the **physical** network
- Instead, we often design distributed systems around **overlay** networks

Overlay Network: Ring Topology



Ring Topology: Physical network



Ring Messages

- In a ring topology, we can get by with one message:
  - Send to next node
- …Or, if you want to get fancy:
  - Send to next node
  - Send to previous node
- This setup only makes sense in certain situations
  - It certainly isn't optimal for searching for a specific file/user/etc!
  - What might a good use case be?

Shared Nothing Architecture

- A popular design for distributed systems is the **shared nothing** (SN) architecture
- Each node in the system is self-sufficient
  - No specialization
  - **No centralized components**
- SN helps ensure scalability
- When all the nodes are the same, failure cases are easier to handle

Downsides of Shared Nothing

- Complicated development: assuming all nodes are the same means dealing with many corner cases
  - What if the underlying hardware is different?
  - Sometimes it's just easier to put some information in a central repository
- If excessive state information must be transferred, the system will be more susceptible to latency
  - Same goes for algorithms that require more coordination

(Semi) Centralized Networks

- An alternative is to include centralized/specialized components
  - Far simpler
  - Represents a single point of failure
- HDFS NameNode: central catalog of all files
- To help deal with failures, hierarchical designs spread out the state information
  - Example: DNS

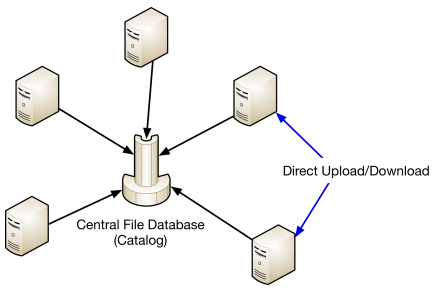
Today's Agenda

- Upcoming assignments
- Distributed network designs
- **A brief history of P2P evolution**
- Designing our own storage network

Napster

- The original Napster (not the streaming service) was one of the first popular P2P file sharing tools
  - Laid the groundwork for today's distributed systems
- On startup, the Napster client transmitted its list of files to a central database
  - Napster didn't host any files, it just maintained a database
  - The paradigm was a bit revolutionary at the time

Napster Network Layout



Napster's Database

- The central database ended up being Napster's undoing
- As the service got popular, the server was flooded with requests and couldn't keep up
- It also was an easy target for law enforcement to shut down

Gnutella

- Learning from Napster's mistakes, *Gnutella* was designed to be completely decentralized
- Instead of using a central database, queries **flooded** through the entire network
  - Unstructured network
- This poses some problems though:
  - Performance: querying gets slower with more users
  - How do you know when your query is over?

KaZaA

- Gnutella didn't catch on, but *KaZaA* did
- Struck a balance between the two approaches
- **Supernodes** functioned as local databases
- Client queries were sent to the supernodes, which would also query their peers
- In other words: a hierarchical approach

Today: BitTorrent

- Nowadays most folks download their favorite non-copyrighted songs, public domain media, and Linux distributions using BitTorrent
  - But certainly not the most recent Game of Thrones episodes!
- Puts the responsibility of indexing the files on websites/users
- Recent versions support **magnet**, backed by a distributed hash table

Today's Agenda

- Upcoming assignments
- Distributed network designs
- **Designing our own storage network**

Thought Experiment: **Storage System Design**

- Design a **backend** storage platform for Facebook
- What data types do we need to handle?
  - You can specialize for a certain subset
- What are the components in the system?
  - Client, server, … anything else?
  - Is it P2P, do we provide a lookup service?
- What messages do we need to support?
- https://classroom.github.com/g/COGDi12J