CSCI 1380
Day 3 [Jan 28]

Theophilus Benson
Last Class

① Motivation for D.S. (i.e. MapReduce)
② Failures (heart beats)

This Class

③ Networking
④ Fallacies of distributed computing
⑤ Performance optimizations

First: the heartbeat dilemma

a company with 100k servers.
30k of these servers failure & restart. What sort of bad things can happen?
AWS

new cluster of machines are added
all machines try to connect to each other
a threat to manage connections
solution: more hardware

FB

\[ N \times N \]

lots of overhead
but finely

\[ O(N \times M) \]

less of overhead
but takes a little longer to get info
Protocols (TCP/UDP)

TCP

1. need to make a connection with special msg (TCP handshake)
2. after handshake: send/receive data
3. Special rate control algorithm & retransmit algorithm

Problems
1. overheads
2. lots of messages
3. no one cares about lost packets

You never worry about lost packets & TCP lets you share the network fairly.

remote function calls (RPC) You want to avoid the startup overhead of TCP

function "foo" i=1, x=5
How to go from monolithic to distributed?

1. Makes distributed calls: needs info about other components

2. Should we create each as a unique process?
   - Keep track of connections

Metadata: Where are the processes located?

IP address + port
1. Each function becomes a process
2. "App" needs information about the other process
3. "Special" to make network connections

RPC ⇒ Remote Procedure Calls

![Diagram of RPC connections with app and login functions, UDP connection, and function arguments.]
1. Heartbeats (when things go wrong)
2. N/W \Rightarrow (TCP \text{ v. UDP})
3. Socket code \mid \text{RPC}

---

Next class

RPC semantics \mid RPC structure

Performance \mid load balances