CPSC 426/526
UseNet and Gossip Protocol

Ennan Zhai
Computer Science Department
Yale University
Recall: Lec-1

- Understanding:
  - Distributed systems vs. decentralized systems
  - Why we need both?
Lecture Roadmap

- Network Basics
- UseNet
- Gossip
- Location and Identity
Network Basics

- We connect computers via point-to-point links:
  - Local area network, DNS and ISP routers
  - Communications are unreliable
  - No global control of the network
We connect computers via point-to-point links:
- Local area network, DNS and ISP routers
- Communications are unreliable
- No global control of the network
Example: HTTP Layer Encapsulation

User A:
- Application
- App-to-app channels
- Host-to-host connectivity
- Link hardware

User B:
- Get index.html
- Connection ID
- Source/Destination
- Link Address
End Hosts vs. Routers

HTTP message

TCP segment

http Hosts vs. Routers
End Hosts vs. Routers

System Developer

Network Developer

host

HTTP

TCP

IP

Ethernet interface

router

http

TCP

IP

Ethernet interface

SONET interface

host

HTTP

TCP

IP

Ethernet interface

SONET interface

Ethernet interface
Finding Nodes

Hey
What's your address? 7:05 PM

173.168.15.10 7:05 PM

No man. Your local address. 7:05 PM

127.0.0.1 7:06 PM

Oh you geeky nerd!!!
I mean your physical address. 11:46 PM

Network Basics

- Each interface on a host has a unique MAC address:
  - My machine 48-bit ethernet address = 32:00:19:ac:b1:40
Network Basics

- Each interface on a host has a unique MAC address:
  - My machine 48-bit ethernet address = 32:00:19:ac:b1:40

Why we need a physical address?
Network Basics

• Each interface on a host has a unique MAC address:
  - My machine 48-bit ethernet address = 32:00:19:ac:b1:40

Why we need a physical address?

Which layer in OSI model it belongs to?
Network Basics

• Each interface on a host has a unique MAC address:
  - My machine 48-bit ethernet address = 32:00:19:ac:b1:40

• This is *not* too interesting to us as programmers
  - We usually do not communicate at the data link layer
Network Basics

- Addressing applications:
  - IP address (32-bit for IPv4) and port number (16-bit)
  - Well-known port numbers (0-1023), e.g., ftp, ssh and http
Network Basics

• Addressing applications:
  - IP address (32-bit for IPv4) and port number (16-bit)
  - Well-known port numbers (0-1023), e.g., ftp, ssh and http

• We have two transport-layer protocols
  - TCP (SSH and FTP) and UDP (Streaming and local broadcast)
  - What is the difference?
End Hosts vs. Routers

System Developer

Network Developer

host

HTTP

TCP

IP

Ethernet interface

router

IP packet

IP

Ethernet interface

SONET interface

IP

SONET interface

Ethernet interface

TCP

HTTP

host

IP

Ethernet interface

SONET interface

Ethernet interface

IP packet

IP packet

IP packet
End Hosts vs. Routers

HTTP message

TCP segment

Ethernet interface

SONET interface

Ethernet interface

Ethernet interface

Ethernet interface
Today's Cluster

PC
Today’s Cluster

PC

Server
Today's Cluster

PC
Server
Cluster
Today’s Cluster
Rack

Today’s Cluster
Rack

Today's Cluster

Network switches (connects nodes with each other and with other racks)
Today's Cluster

Rack

Network switches (connects nodes with each other and with other racks)

Many nodes/blades (often identical)
Today’s Cluster

Rack

Network switches (connects nodes with each other and with other racks)

Many nodes/ blades (often identical)

Storage device(s)
Today's Cluster

- What if cluster is too big to fit into machine room?

PC  Server  Cluster
• What if cluster is too big to fit into machine room?
  - Build a separate building for the cluster
  - Building can have lots of cooling and power
• What if cluster is too big to fit into machine room?
  - Build a separate building for the cluster
  - Building can have lots of cooling and power
  - Result: Data center
Google Datacenter in Oregon
Google Datacenter in Oregon

Data centers (size of a football field)
A warehouse-sized computer

- A single data center can easily contain 10,000 racks with 100 cores in each rack (1,000,000 cores total)
Google Datacenters in the US
Google Datacenters in this World
End Hosts vs. Routers

HTTP message

TCP segment

Ethernet interface
IP packet
IP
TCP
host
HTTP
router
IP packet
IP
TCP
host
HTTP
Network APIs

- Programmers need to access the network
- A network application programming interface (API)
  - Socket programming
  - Remote procedure calls
Socket (TCP)

Client

Create a socket
Name the socket (assign local address, port)
Connect to the other side
read / write byte streams
close the socket

Server

Create a socket
Name the socket (assign local address, port)
Set the socket for listening
Wait for and accept a connection; get a socket for the connection
read / write byte streams
close the socket

socket
bind
listen
accept
read/write
close

Copyright © 2014-2016 Paul Krzyzanowski

September 15, 2016
import socket

Server

Create a socket

Name the socket (assign local address, port)

Set the socket for listening

Wait for and accept a connection; get a socket for the connection

read / write byte streams

Client

socket

bind

listen

accept

read/write

close the socket

close the listening socket
import socket

s = socket.socket(AF_INET, SOCK_STREAM)

Create a socket
Name the socket
(assign local address, port)
Set the socket for listening
Wait for and accept a connection; get a socket for the connection
read / write byte streams
close the socket
close the listening socket

Client

Server

socket
bind
listen
accept
read/write
close
close
import socket

s = socket.socket(AF_INET,\
                  SOCK_STREAM)
s.bind((host, port))

Create a socket
Name the socket
(set local address, port)
Set the socket for listening
Wait for and accept a
connection; get a socket for
the connection
read / write byte streams
close the socket
close the listening socket

socket
bind
listen
accept
read/write
close
close
import socket

s = socket.socket(AF_INET, \ 
                 SOCK_STREAM)

s.bind((host, port))

s.listen(5)

while 1:
    conn, addr = s.accept()
    msg = conn.recv()
    conn.close()

s.close
import socket

s = socket.socket(AF_INET, SOCK_STREAM)
s.bind((host, port))
s.listen(5)

while 1:
    conn, addr = s.accept()
    msg = conn.recv()
    conn.close
import socket

s = socket.socket(AF_INET,\  
    SOCK_STREAM)

s.bind(host, port)

s.listen(5)

while 1:
    conn, addr = s.accept()
    msg = conn.recv()
    conn.close

s.close
Socket (TCP)

Client

- Create a socket
- Name the socket (assign local address, port)
- Connect to the other side
- read / write byte streams
- close the socket

Server

- Create a socket
- Name the socket (assign local address, port)
- Set the socket for listening
- Wait for and accept a connection; get a socket for the connection
- close the listening socket
- read / write byte streams
- close

Python code:

```python
import socket

s = socket.socket(AF_INET,\    SOCK_STREAM)

a = socket.gethostbyname(host)

s.connect(a, port)

s.sendall(msg)
```
Socket (TCP)

Client

Create a socket
Name the socket (assign local address, port)
Connect to the other side
read / write byte streams
close the socket

Server

Create a socket
Name the socket (assign local address, port)
Set the socket for listening
Wait for and accept a connection; get a socket for the connection
read / write byte streams
close the socket
close the listening socket
Connectionless (UDP) socket operations

- Create a socket
- Name the socket (assign local address, port)
- Send a message
- Receive a message
- Close the socket

Client

- `socket`
- `bind`
- `sendto`
- `recvfrom`
- `close`

Server

- `socket`
- `bind`
- `recvfrom`
- `sendto`
- `close`
Lecture Roadmap

- Network Basics
- UseNet
- Gossip
- Location and Identity
UseNet

- UseNet is the first successful decentralized messaging system

Brief history:
- 1979: Created by Truscott and Ellis (based on gossip scheme)
- 1981: An early UseNet map in hand-drawn ASCII
- 1986: Shifting to NNTP to propagate news over TCP
- 1990s: Public interest in UseNet peaks, then declines quickly
UseNet

• Several important questions:
  - Why we need to understand UseNet?
  - What are assumptions of UseNet?
  - Why I should care about a simple messaging system?
UseNet

- Several important questions:
  - Why we need to understand UseNet?
  - What are assumptions of UseNet?
  - Why I should care about a simple messaging system?

Let’s build a simple decentralized messaging system now.
UseNet

• How does it work?
  - Example UseNet message from RFC 1036

```plaintext
From: jerry@eagle.ATT.COM (Jerry Schwarz)
Path: cbosgd!mhuxj!mhuxt!eagle!jerry
Newsgroups: news.announce
Subject: Usenet Etiquette -- Please Read
Message-ID: <642@eagle.ATT.COM>
Date: Fri, 19 Nov 82 16:14:55 GMT
Followup-To: news.misc
Expires: Sat, 1 Jan 83 00:00:00 -0500
Organization: AT&T Bell Laboratories, Murray Hill

The body of the message comes here, after a blank line.
```
• How does it work?
  - Example UseNet message from RFC 1036

```
From: jerry@eagle.ATT.COM (Jerry Schwarz)
Path: cbosgd!mhuxj!mhuxt!eagle!jerry
Newsgroups: news.announce
Subject: Usenet Etiquette -- Please Read
Message-ID: <642@eagle.ATT.COM>
Date: Fri, 19 Nov 82 16:14:55 GMT
Followup-To: news.misc
Expires: Sat, 1 Jan 83 00:00:00 -0500
Organization: AT&T Bell Laboratories, Murray Hill

The body of the message comes here, after a blank line.
```

- From: E-mail address of sender
UseNet

- How does it work?
  - Example UseNet message from RFC 1036

```
From: jerry@eagle.ATT.COM (Jerry Schwarz)
Path: cbosgd!mhuxj!mhuxt!eagle!jerry
Newsgroups: news.announce
Subject: Usenet Etiquette -- Please Read
Message-ID: <642@eagle.ATT.COM>
Date: Fri, 19 Nov 82 16:14:55 GMT
Followup-To: news.misc
Expires: Sat, 1 Jan 83 00:00:00 -0500
Organization: AT&T Bell Laboratories, Murray Hill

The body of the message comes here, after a blank line.
```

- From: E-mail address of sender
- Path: log of message’s propagation through network
UseNet

- How does it work?
  - Example UseNet message from RFC 1036

```
From: jerry@eagle.ATT.COM (Jerry Schwarz)
Path: cbosgd!mhuxj!mhuxt!eagle!jerry
Newsgroups: news.announce
Subject: Usenet Etiquette -- Please Read
Message-ID: <642@eagle.ATT.COM>
Date: Fri, 19 Nov 82 16:14:55 GMT
Followup-To: news.misc
Expires: Sat, 1 Jan 83 00:00:00 -0500
Organization: AT&T Bell Laboratories, Murray Hill

The body of the message comes here, after a blank line.
```

- From: E-mail address of sender
- Path: log of message’s propagation through network
- Subject: message subject, just like in Email
How does it work?

- Example UseNet message from RFC 1036

```
From: jerry@eagle.ATT.COM (Jerry Schwarz)
Path: cbosgd!mhuxj!mhuxt!eagle!jerry
Newsgroups: news.announce
Subject: Usenet Etiquette -- Please Read
Message-ID: <642@eagle.ATT.COM>
Date: Fri, 19 Nov 82 16:14:55 GMT
Followup-To: news.misc
Expires: Sat, 1 Jan 83 00:00:00 -0500
Organization: AT&T Bell Laboratories, Murray Hill

The body of the message comes here, after a blank line.
```

- From: E-mail address of sender
- Path: log of message’s propagation through network
- Subject: message subject, just like in Email
- Message-ID: Unique ID of message (duplicate detection)
Why was UseNet so popular?
- It worked, even over slow connections
- It was decentralized
- It was algorithmically simple -- can be easily understood
- Anyone can post, create a group, etc.
UseNet

• Why was UseNet so popular?
  - It worked, even over slow connections
  - It was decentralized
  - It was algorithmically simple -- can be easily understood
  - Anyone can post, create a group, etc.

• Why did it collapse?
  - Spam!!!
  - No privacy
Lecture Roadmap

- Network Basics
- UseNet
- Gossip
- Location and Identity
Strawman Solution

- UseNet distributes its messages in a gossip way
Strawman Solution

• UseNet distributes its messages in a gossip way
• Strawman solution:
  - On receiving a message, rebroadcast it to all neighbors
  - Issue: graph cycles result in forwarding loops
Strawman Solution

- UseNet distributes its messages in a gossip way
- Strawman solution:
  - On receiving a message, rebroadcast it to all neighbors
  - Issue: graph cycles result in forwarding loops

- Another strawman solution:
  - Assign each message a unique ID
  - On receiving a message, first check ID. If already received, ignore it
  - Only if message is new, re-broadcast it as above
Gossip Protocol

- Randomized technique: rumor-mongering
  - Keep DB of message IDs received

Advantages:
- Fast: when the rumor is new, it propagates quickly
- Efficient: when rumor is old, most nodes have it, so it stops quickly

Disadvantage:
- Rumor may not spread to all the nodes before it becomes "cold"
Gossip Protocol

• Randomized technique: rumor-mongering
  - Keep DB of message IDs received
  - On receiving a new message, save its ID in DB, then loop:
Gossip Protocol

• Randomized technique: rumor-mongering
  - Keep DB of message IDs received
  - On receiving a new message, save its ID in DB, then loop:
    1. Pick a random neighbor (other than source of message)

• Advantages:
  - Fast: when the rumor is new, it propagates quickly
  - Efficient: when rumor is old, most nodes have it, so it stops quickly

• Disadvantage:
  - Rumor may not spread to all the nodes before it becomes "cold"
Gossip Protocol

- Randomized technique: rumor-mongering
  - Keep DB of message IDs received
  - On receiving a new message, save its ID in DB, then loop:
    1. Pick a random neighbor (other than source of message)
    2. Rebroadcast the message to the chosen neighbor

- Advantages:
  - Fast: when the rumor is new, it propagates quickly
  - Efficient: when rumor is old, most nodes have it, so it stops quickly

- Disadvantage:
  - Rumor may not spread to all the nodes before it becomes "cold"
Gossip Protocol

- Randomized technique: rumor-mongering
  - Keep DB of message IDs received
  - On receiving a new message, save its ID in DB, then loop:
    1. Pick a random neighbor (other than source of message)
    2. Rebroadcast the message to the chosen neighbor
    3. Neighbor replies indicating whether it had seen the msg

- Advantages:
  - Fast: when the rumor is new, it propagates quickly
  - Efficient: when rumor is old, most nodes have it, so it stops quickly

- Disadvantage:
  - Rumor may not spread to all the nodes before it becomes "cold"
Gossip Protocol

• Randomized technique: rumor-mongering
  - Keep DB of message IDs received
  - On receiving a new message, save its ID in DB, then loop:
    1. Pick a random neighbor (other than source of message)
    2. Rebroadcast the message to the chosen neighbor
    3. Neighbor replies indicating whether it had seen the msg
    4. If msg was new to neighbor, repeat this loop

• Advantages:
  - Fast: when the rumor is new, it propagates quickly
  - Efficient: when rumor is old, most nodes have it, so it stops quickly

• Disadvantage:
  - Rumor may not spread to all the nodes before it becomes “cold”
Gossip Protocol

• Randomized technique: rumor-mongering
  - Keep DB of message IDs received
  - On receiving a new message, save its ID in DB, then loop:
    1. Pick a random neighbor (other than source of message)
    2. Rebroadcast the message to the chosen neighbor
    3. Neighbor replies indicating whether it had seen the msg
    4. If msg was new to neighbor, repeat this loop
    5. If neighbor has msg, then repeat this loop with \( p \); otherwise stop the loop

Advantages:
- Fast: when the rumor is new, it propagates quickly
- Efficient: when rumor is old, most nodes have it, so it stops quickly

Disadvantage:
- Rumor may not spread to all the nodes before it becomes “cold”
Gossip Protocol

• Randomized technique: rumor-mongering
  - Keep DB of message IDs received
  - On receiving a new message, save its ID in DB, then loop:
    1. Pick a random neighbor (other than source of message)
    2. Rebroadcast the message to the chosen neighbor
    3. Neighbor replies indicating whether it had seen the msg
    4. If msg was new to neighbor, repeat this loop
    5. If neighbor has msg, then repeat this loop with $p$; otherwise stop the loop

• Advantages:
  - Fast: when the rumor is new, it propagates quickly
  - Efficient: when rumor is old, most nodes have it, so it stops quickly
Gossip Protocol

- Randomized technique: rumor-mongering
  - Keep DB of message IDs received
  - On receiving a new message, save its ID in DB, then loop:
    1. Pick a random neighbor (other than source of message)
    2. Rebroadcast the message to the chosen neighbor
    3. Neighbor replies indicating whether it had seen the msg
    4. If msg was new to neighbor, repeat this loop
    5. If neighbor has msg, then repeat this loop with $p$; otherwise stop the loop

- Advantages:
  - Fast: when the rumor is new, it propagates quickly
  - Efficient: when rumor is old, most nodes have it, so it stops quickly

- Disadvantage:
  - Rumor may not spread to all the nodes before it becomes “cold”
Gossip Protocol

- Another randomized technique: anti-entropy

Another randomized technique: anti-entropy
Gossip Protocol

- Another randomized technique: anti-entropy
  - Independently of receiving msg, each node occasionally does:
    1. Pick a random neighbor

  Advantages:
  - Guarantee each msg will eventually propagate to all nodes

Disadvantage:
- Substantial bandwidth to compare databases
- New messages spread slowly
Gossip Protocol

- Another randomized technique: anti-entropy
  - Independently of receiving msg, each node occasionally does:
    1. Pick a random neighbor
    2. Compare list of message IDs received so far

- Advantages:
  - Guarantee each msg will eventually propagate to all nodes

- Disadvantages:
  - Substantial bandwidth to compare databases
  - New messages spread slowly
Gossip Protocol

• Another randomized technique: anti-entropy
  - Independently of receiving msg, each node occasionally does:
    1. Pick a random neighbor
    2. Compare list of message IDs received so far
    3. For all msg IDs one node has received and the other hasn’t, transfer the corresponding msg

Advantages:
- Guarantee each msg will eventually propagate to all nodes

Disadvantage:
- Substantial bandwidth to compare databases
- New messages spread slowly
Gossip Protocol

• Another randomized technique: anti-entropy
  - Independently of receiving msg, each node occasionally does:
    1. Pick a random neighbor
    2. Compare list of message IDs received so far
    3. For all msg IDs one node has received and the other hasn’t, transfer the corresponding msg
    4. Result: received message ID database should be identical
Gossip Protocol

• Another randomized technique: anti-entropy
  - Independently of receiving msg, each node occasionally does:
    1. Pick a random neighbor
    2. Compare list of message IDs received so far
    3. For all msg IDs one node has received and the other hasn’t, transfer the corresponding msg
    4. Result: received message ID database should be identical

• Advantages:
  - Guarantee each msg will eventually propagate to all nodes
Gossip Protocol

• Another randomized technique: anti-entropy
  - Independently of receiving msg, each node occasionally does:
    1. Pick a random neighbor
    2. Compare list of message IDs received so far
    3. For all msg IDs one node has received and the other hasn’t, transfer the corresponding msg
    4. Result: received message ID database should be identical

• Advantages:
  - Guarantee each msg will eventually propagate to all nodes

• Disadvantage:
  - Substantial bandwidth to compare databases
  - New messages spread slowly
Gossip Protocol

- Combining rumor-mongering with anti-entropy:
  - Rumor-mongering spreads new messages quickly to most nodes
  - Anti-entropy ensures “cold” nodes to get all the messages
Gossip Protocol

• Combining rumor-mongering with anti-entropy:
  - Rumor-mongering spreads new messages quickly to most nodes
  - Anti-entropy ensures “cold” nodes to get all the messages

• Result: A fast, efficient and reliable gossip scheme:
  - All nodes guaranteed to receive each message eventually
  - Messages propagate quickly with high probability
  - Not too much bandwidth is wasted
Lecture Roadmap

• Network Basics
• UseNet
• Gossip
• Location and Identity
Location and Identity

- Basic questions in any decentralized system:
  - How do you identify someone you want to communicate with?
  - How do you locate them in order to get a message to them?
Location and Identity

- Basic questions in any decentralized system:
  - How do you identify someone you want to communicate with?
  - How do you locate them in order to get a message to them?

- In distributed system field:
  - Identity: supposed to stay the same wherever you move
  - Location: contains structure useful for finding you
Location and Identity in UseNet

- Identity and location representations in UseNet:
  - Identity is a short, administratively unique string, say, decvax
  - Location represented by a bang path

```
From: jerry@eagle.ATT.COM (Jerry Schwarz)
Path: cbosgd!mhuxj!mhuxt!eagle!jerry
Newsgroups: news.announce
Subject: Usenet Etiquette -- Please Read
Message-ID: <642@eagle.ATT.COM>
Date: Fri, 19 Nov 82 16:14:55 GMT
Followup-To: news.misc
Expires: Sat, 1 Jan 83 00:00:00 -0500
Organization: AT&T Bell Laboratories, Murray Hill

The body of the message comes here, after a blank line.
```
Location and Identity in the Internet

- Identity and location representations in today’s Internet:
  - You are identified by a DNS name, located by an IP address
  - Develop a modern UseNet (in Lab1)
Peerster Demo for Lab1
Next Lecture

• In the lec-3, I will cover:
  - Peer-to-Peer content systems’ history
  - How to lookup object in a huge P2P network (say $2^{160}$ nodes)
  - Comparing different types of P2P systems