

# Decentralized Systems Engineering

CS-438 – Fall 2025

DEDIS

Pierluca Borsò-Tan and Bryan Ford



Credits: P. Tennage, C. Basescu, et al.

# Communicating with (many, unknown) peers

- Same machine  
a file in a shared directory, or Linux `wall` command
- Local networking  
shared drive, intranet website
- Global networking, centralized trust  
mailing lists, forums, Reddit, ...
- Decentralized  
??? → today's lecture

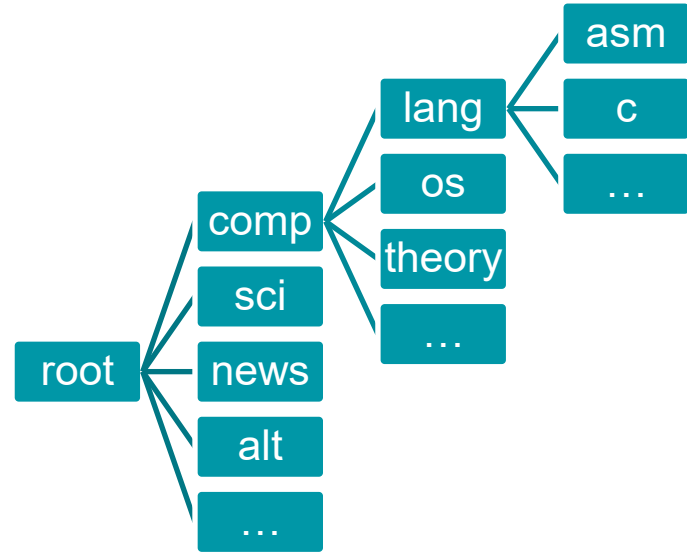
# Decentralized Communication

Usenet & Gossip

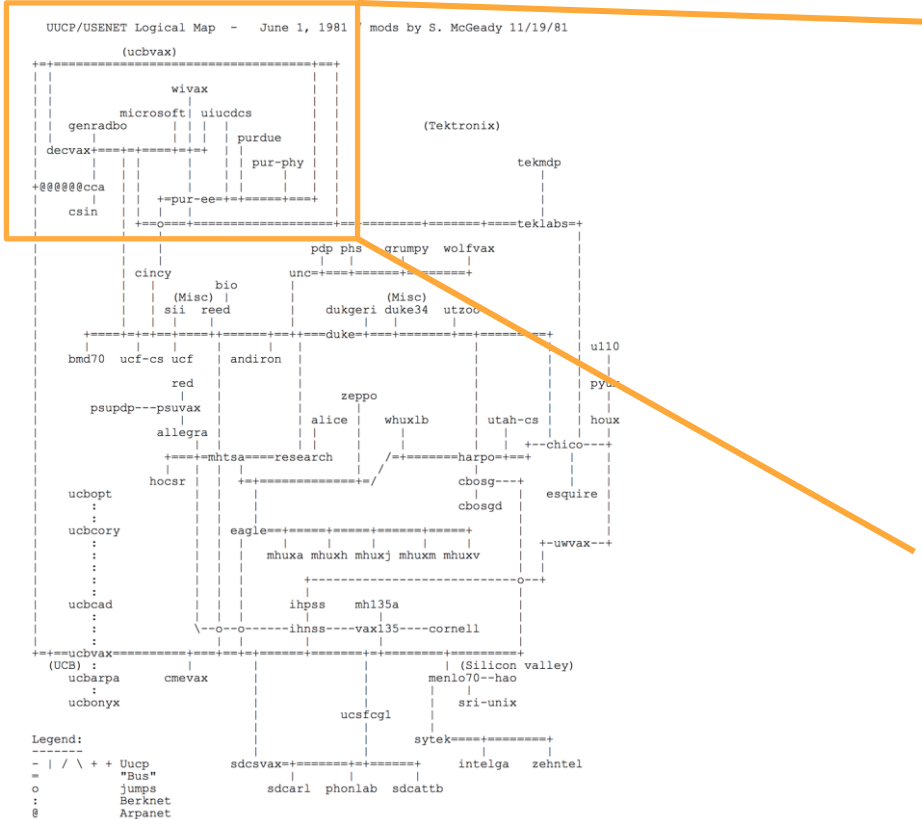
(Homework 1)

# What is Usenet ?

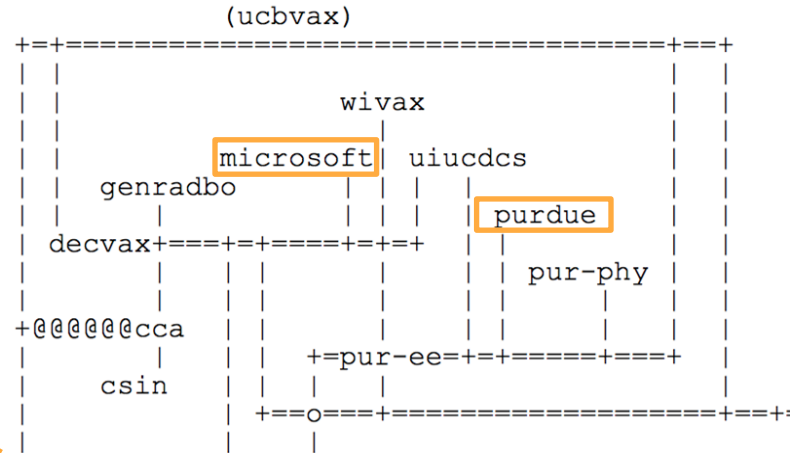
- User's Network
- Worldwide, distributed discussion system
- Hierarchical organization of topics
- Context – early 1980s:
  - Pre-Internet (1980)
  - Mainframes, then minicomputers
  - Intermittent, dial-up connections (at best 56kbit/s)
  - Resilient: censorship- and failure-resistant



# Early UUCP/Usenet Map



UUCP/USENET Logical Map - June 1, 1981 /



# Why think of Usenet in the 2020s ?

- 1980s computers are still relevant
  - ... they are just way smaller today:
    - Embedded systems, Internet of Things (IoT), Sensors, etc.
- 1980s networks are still relevant:
  - Low-Power Wide-Area Networks (e.g. LoRaNET / LoRaWAN)
    - [0.3, 50] kbit/s
    - 256 bytes / message
    - Per-message pricing (~ 2 CHF / MB)
- Power and batteries are their limit

# Building Usenet : specifications

- Worldwide, distributed discussion system
- Hierarchical organization of topics / newsgroups
- Messages are (eventually) received by all subscribers
- It is possible to respond privately (pre-email)
- Resilient: censorship- and failure-resistant
- Intermittent, dial-up connections – slow & costly !
- Limited computing resources (1980s)

# Building Usenet : network messages

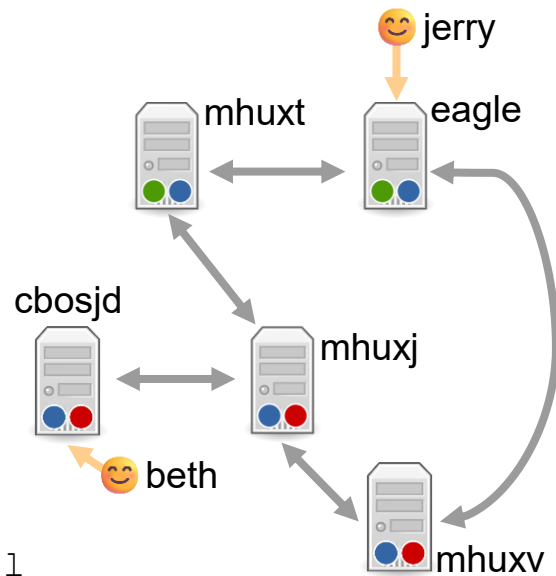
**Header**

- From: jerry@eagle.uucp (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
- Organization: AT&T Bell Laboratories, Murray Hill
- Expires: Sat, 1 Jan 83 00:00:00 -0500

**Blank line**

**Body**

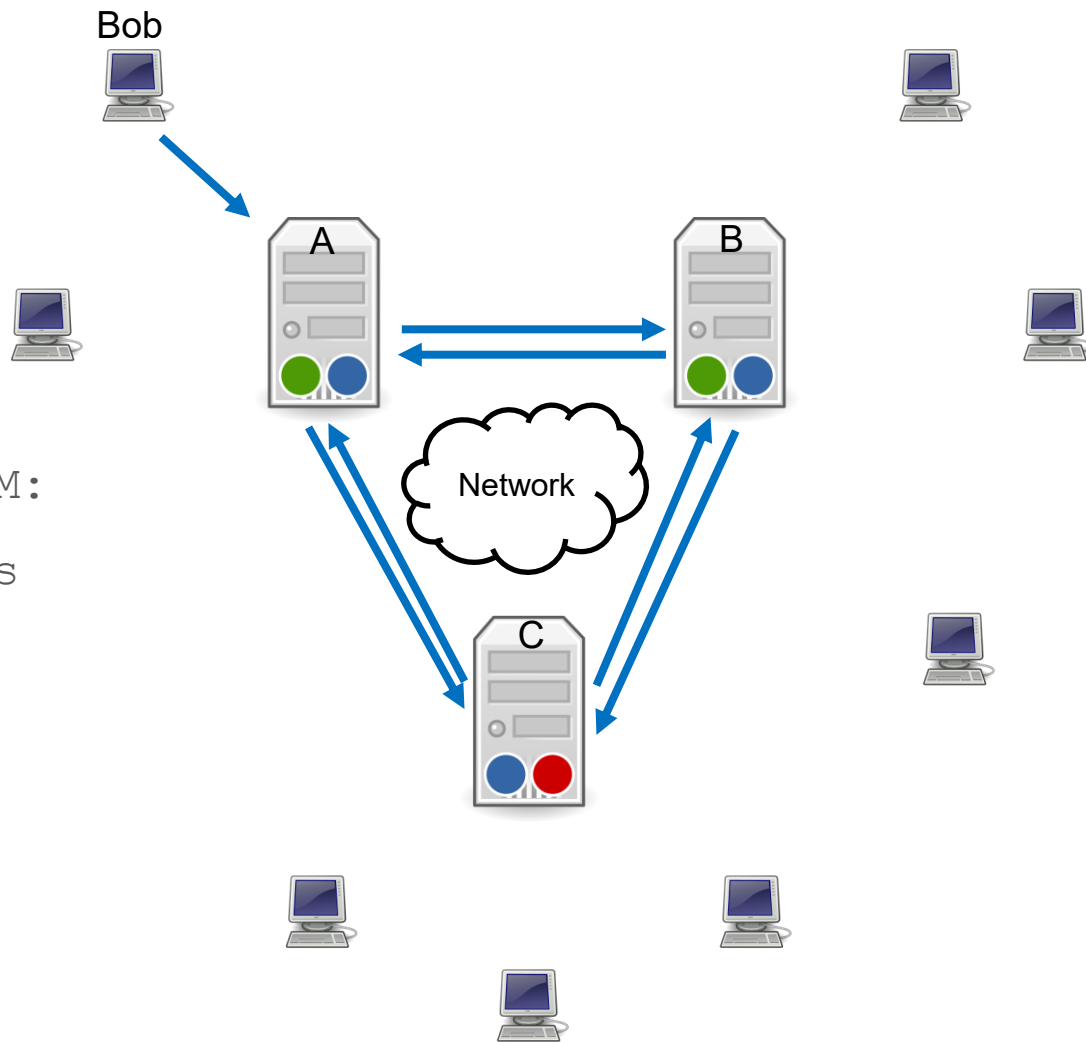
The message itself comes here, after a blank line.  
Hello, world !



# Naive Broadcast

On receiving message M:  
Send M to all peers  
(except sender)

What's the problem here ?



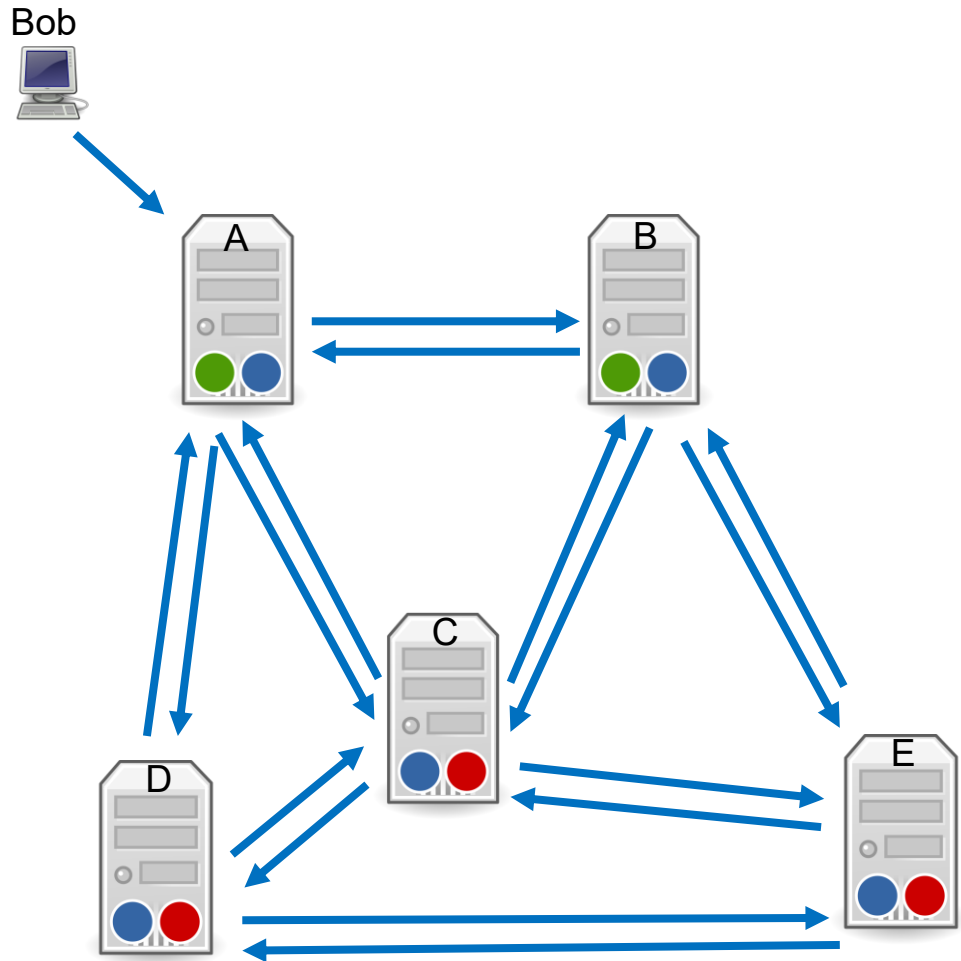
# Naive Broadcast

Network meltdown !

- Exponential # of messages
- Too costly to operate

How do we fix this ?

1. Recognize messages
2. Restrict the graph to a tree  
e.g., Ethernet (R)STP

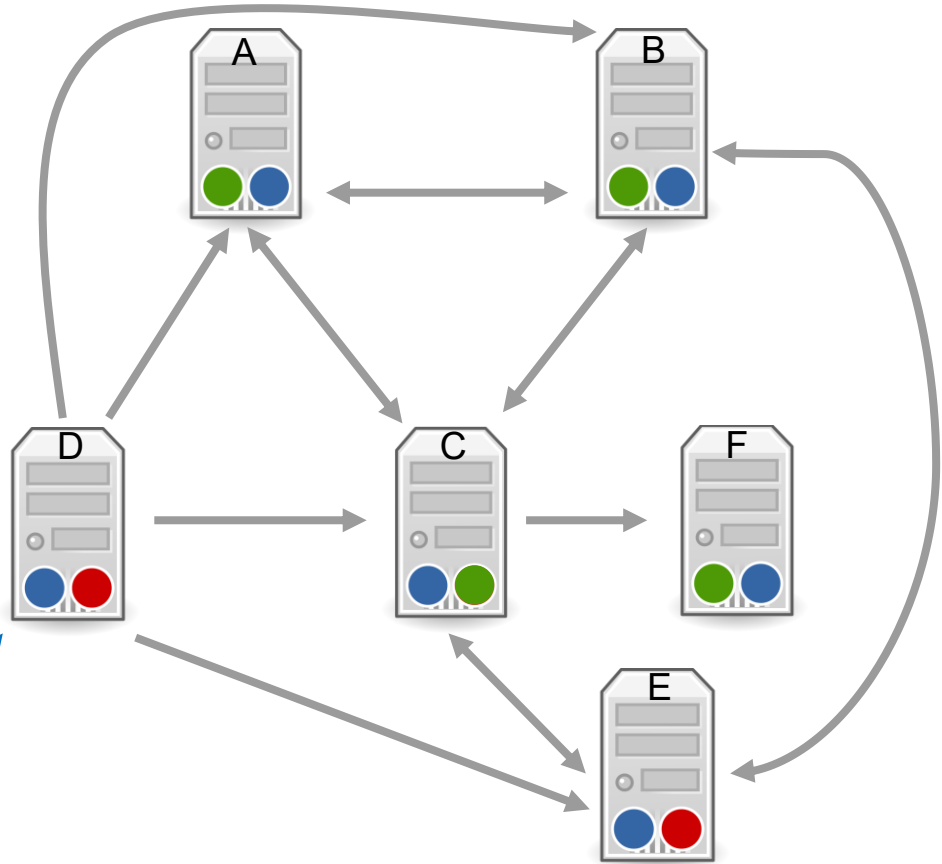


# Recognizing Messages with IDs

- How do you generate message IDs ?
  - Big random number (e.g., 256 bits)
  - Hash
  - Usenet: <sequence number>@<node>
- How do you detect / trace node misbehavior ?
  - Usenet: use message propagation path

# Naive Broadcast (Fixed!)

```
On receiving message M:  
  if M is known:  
    ignore  
  else:  
    Send M to all peers  
    (except sender)
```



Which issues do you foresee ?

# Broadcast : Limitations

- Well-connected nodes often receive the same message many times
- What happens if some nodes failed ?
- Do we need to send the whole message every time ?
  - Early Usenet (UUCP)

“better than accepting the delay of round-trips”
  - Late Usenet (NNTP) - binaries, high traffic volume, etc.

“we can't afford to”

# Broadcast efficiency

How can we minimize duplication and reduce traffic ?

By comparing sets of messages !

A->B    ihave <id\_1>, <id\_2>, <id\_3>, ...    <B\_address>

A<-B    sendme <id\_3>, <id\_5>    <A\_address>

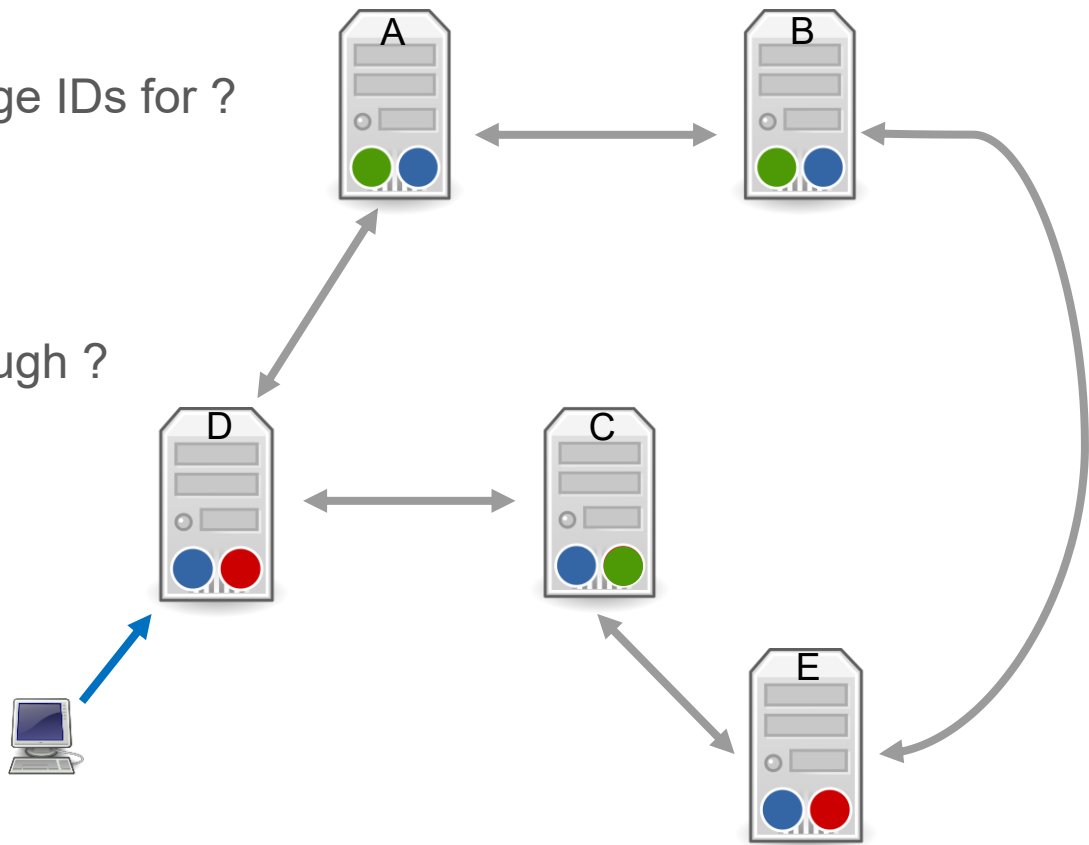
What implementation issues can you foresee ?

# Message IDs : Trade-offs

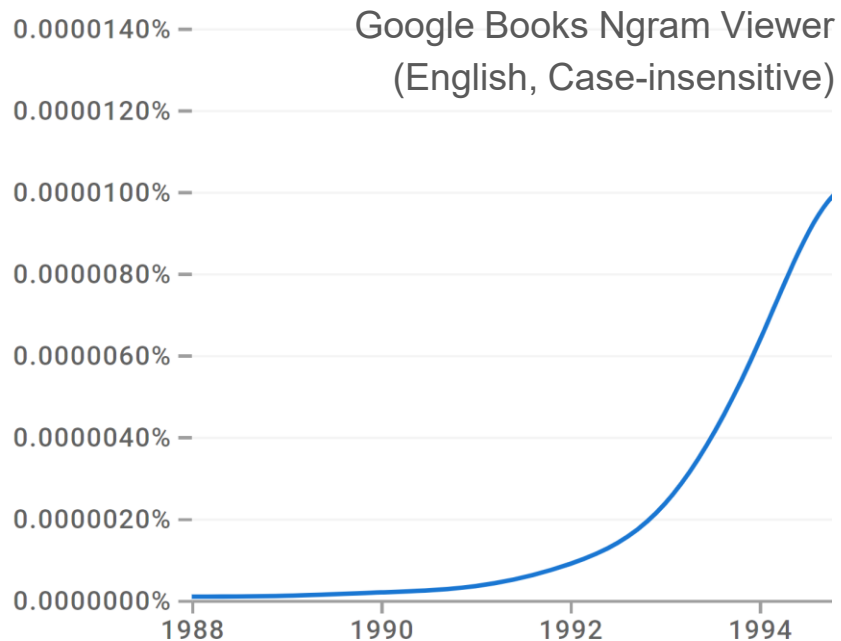
- How long do you store message IDs for ?

What happens if ...

- you store them forever ?
- you don't store them long enough ?



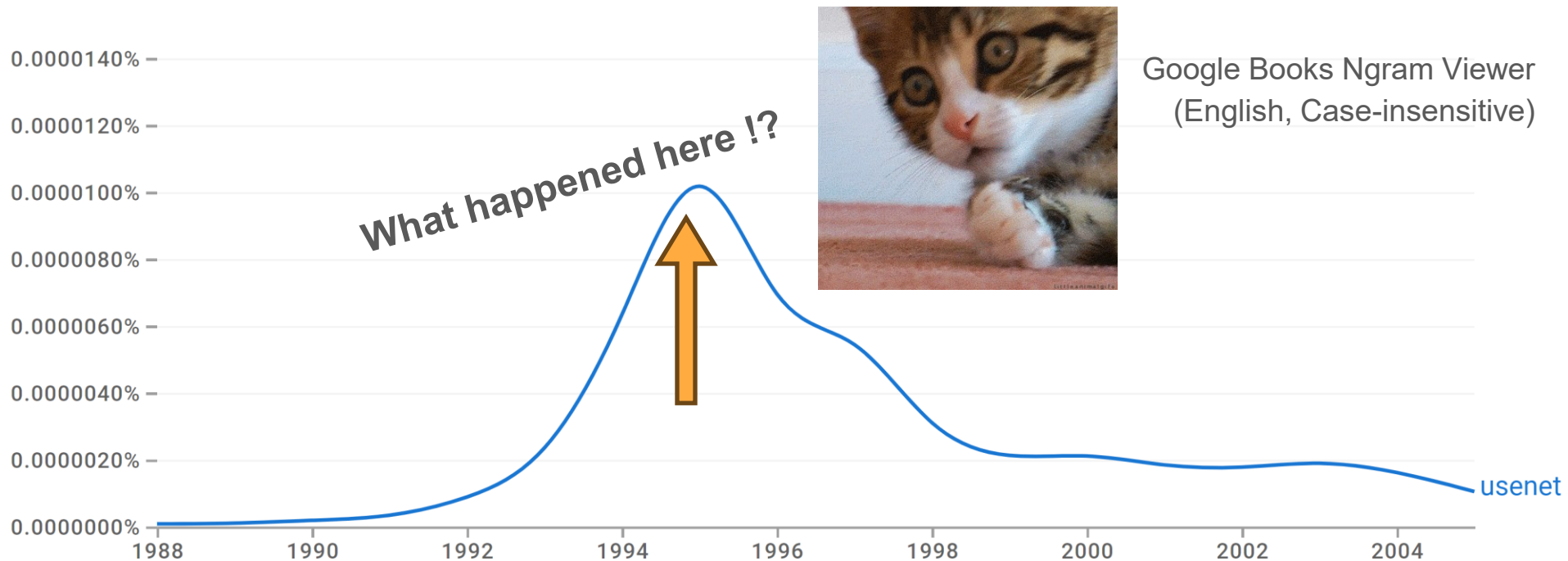
# The Life of Usenet



What made it so successful ?

- It worked !
- Engineering simplicity
- Decentralization
- “Democratizing”

# The Life and Death of Usenet



# What killed Usenet ?

Cause #1: Spam !

- Jan 1994 : Global Alert for All : Jesus is Coming Soon
- Apr 1994 : Green Card Lottery – Final One?

Other causes:

- Better alternatives
- Slow evolution

# Beyond Usenet: Gossip Efficiency

What if we wanted to further minimize traffic ?

A closer look at `ihave/sendme` :

- **V** message content is only sent once per node
- **X** still requires P2P interaction, sending IDs redundantly

Can we minimize bandwidth usage without interaction ?

- Naive broadcast  $O(n \cdot d)$  ,  $n = \#$  nodes,  $d = \text{max. degree of any node}$
- Can we reduce to  $O(n)$  ?

# Improved Gossiping

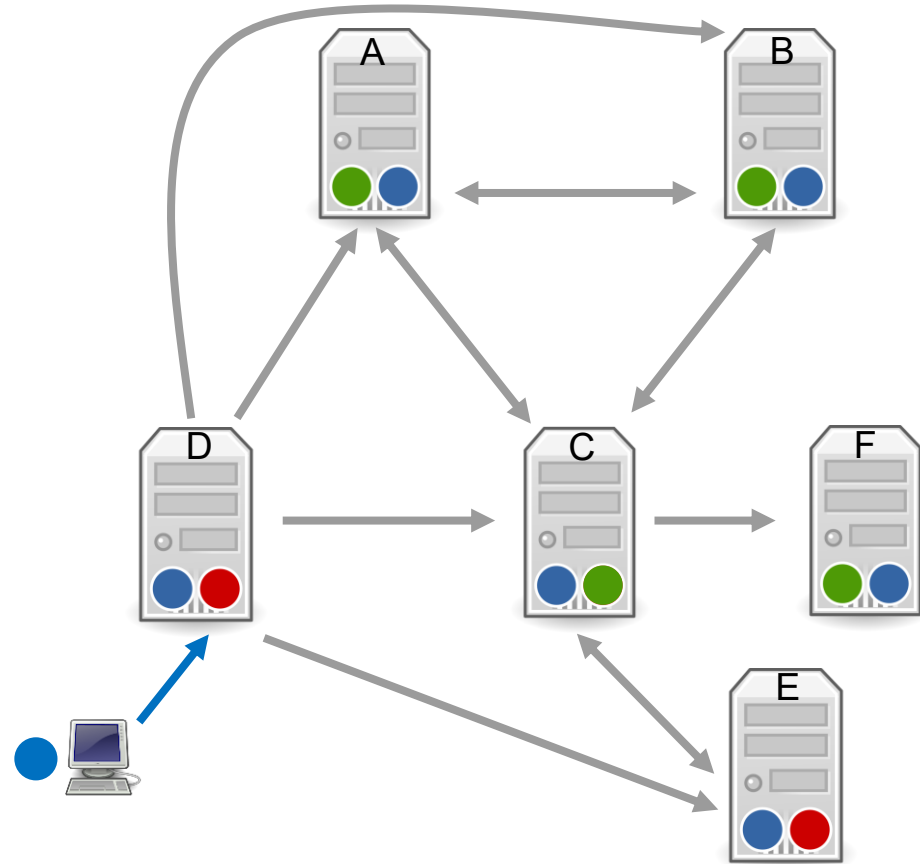
What can we learn from people ?

## Rumor mongering

On receiving message M:

```
pick random neighbor, send M  
neighbor replies: new rumor?
```

```
if new:    repeat  
else:     flip_coin()  
          if head: repeat  
          else:    stop
```



What's good about this ? Which issues do you foresee ?

# Improved Gossiping (cont'd)

How can make sure messages reach *every* node?

## **Anti-entropy**

Periodically (when timer fires):

```
pick random neighbor
send "anything new?"
reduce entropy (ihave/sendme)
```

What's good about this ? What's limiting ?

This (slowly) ensures complete dissemination, at  $O(n)$  per period