



Reliable Distributed Systems

Peer to Peer

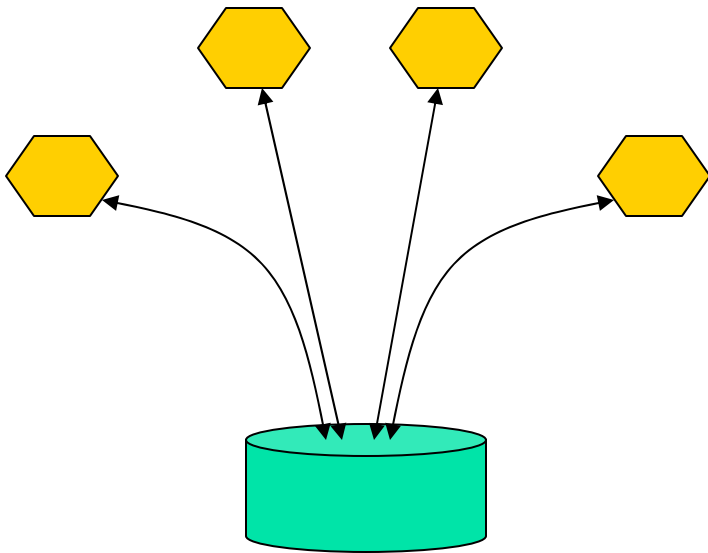


Peer-to-Peer (p2p) Systems

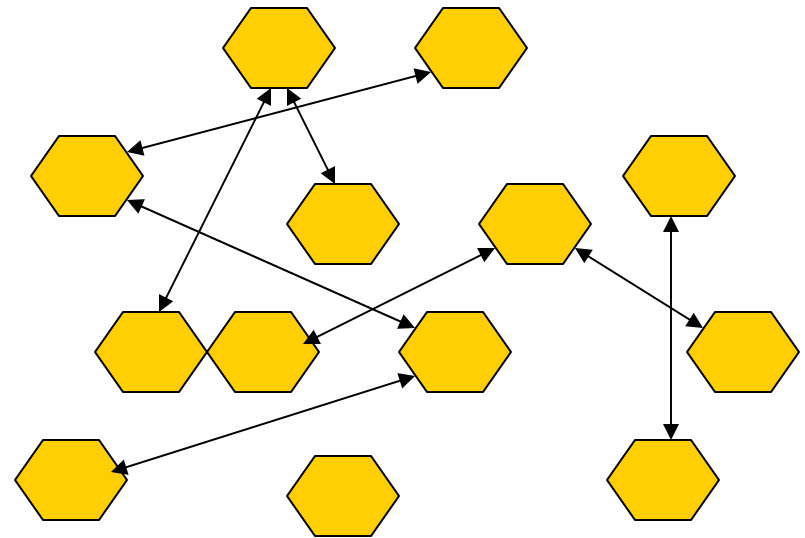
- The term refers to a kind of distributed computing system in which the “main” service is provided by having the client systems talk directly to one-another
- In contrast, traditional systems are structured with servers at the core and clients around the edges

p2p systems

Standard systems:
Client/Server structured



P2P systems: Clients help
one-another out

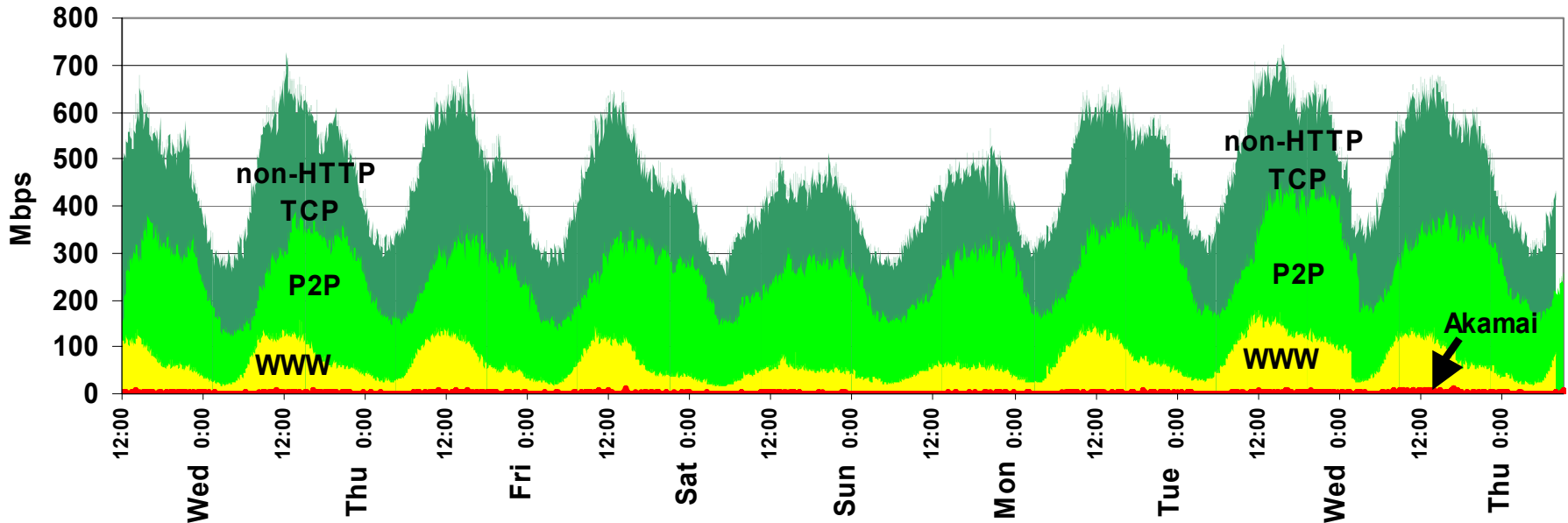




An “important” topic

- ... or at least, it gets a lot of press
 - Recording industry claims that p2p downloads are killing profits!
 - Used to be mostly file sharing, but now online radio feeds (RSS feeds) are a big deal too
 - U. Wash. study showed that 80% of their network bandwidth was spent on music/video downloads!
 - DVDs are largest, and accounted for the lion's share
 - A great many objects were downloaded *many* times
 - Strangely, many downloads took months to complete...
 - Most went to a tiny handful of machines in dorm rooms

Where has all the bandwidth gone?



Breakdown of UW TCP bandwidth into HTTP Components (May 2002)

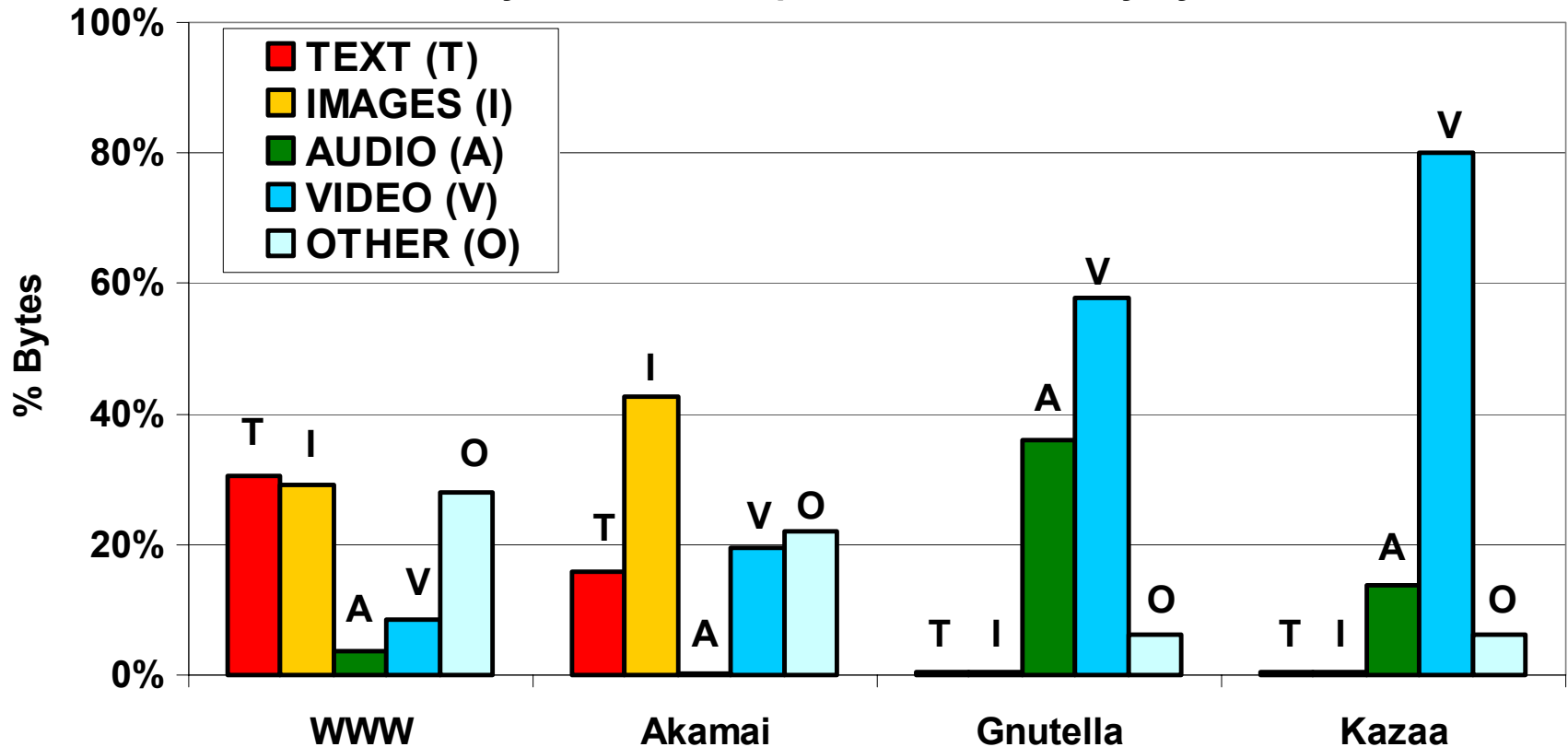
- WWW = 14% of TCP traffic; P2P = 43% of TCP traffic
- P2P dominates WWW in bandwidth consumed!!

Source: Hank Levy. See

http://www.cs.washington.edu/research/networking/websys/pubs/osdi_2002/osdi.pdf

Object type for different systems

Byte Breakdown per Content Delivery System



Source: Hank Levy. See

http://www.cs.washington.edu/research/networking/websys/pubs/osdi_2002/osdi.pdf



Today: An Overview

- Today we'll look at the area as a whole
 - Origins: Illegal file sharing
 - Early academic work: "Distributed hash tables"
 - Subsequent spread of field into many other areas: steganographic storage, erasure codes, gossip protocols and epidemic data dissemination, etc
- In upcoming lectures we'll look at details of some research systems



An old idea...

- If you think about it, most of the protocols we've discussed are "peer to peer" in a broad sense
 - Pretty much everything Lamport was interested in uses direct client-to-client communication
 - Group communication systems often do have servers, but not all need them...
- But the term really has a stronger meaning
 - Denotes systems where the "data that matters" is passed among cooperating client systems
 - And there may be huge numbers of clients
 - Evokes image of resistance fighters working to overthrow an evil IP empire



Attributes of p2p systems

- They can be enormous
 - We often talk about hundreds of thousands or millions of client nodes, coming and going rapidly
 - If there are servers, they are small in number and have limited roles
- These clients are everywhere
 - Even in Kenya or Nepal... places with lousy network connectivity
 - Often behind firewalls or NAT boxes
 - Some are supercomputers. But many are slow



The issue with NAT boxes

- When a system uses firewalls or NAT boxes
 - Client systems *inside* the network can usually talk to servers outside it
 - The NAT knows about the TCP 3-way handshake and “creates a tunnel” on the fly
 - It remaps the (IP address, port) pair as packets pass by, so it looks as if the NAT (not the client) is making the connection and receiving the replies...
 - But connectivity from outside to inside is blocked
 - In fact, because client IP address is mapped, the client simply can’t be addressed other than through the NAT!

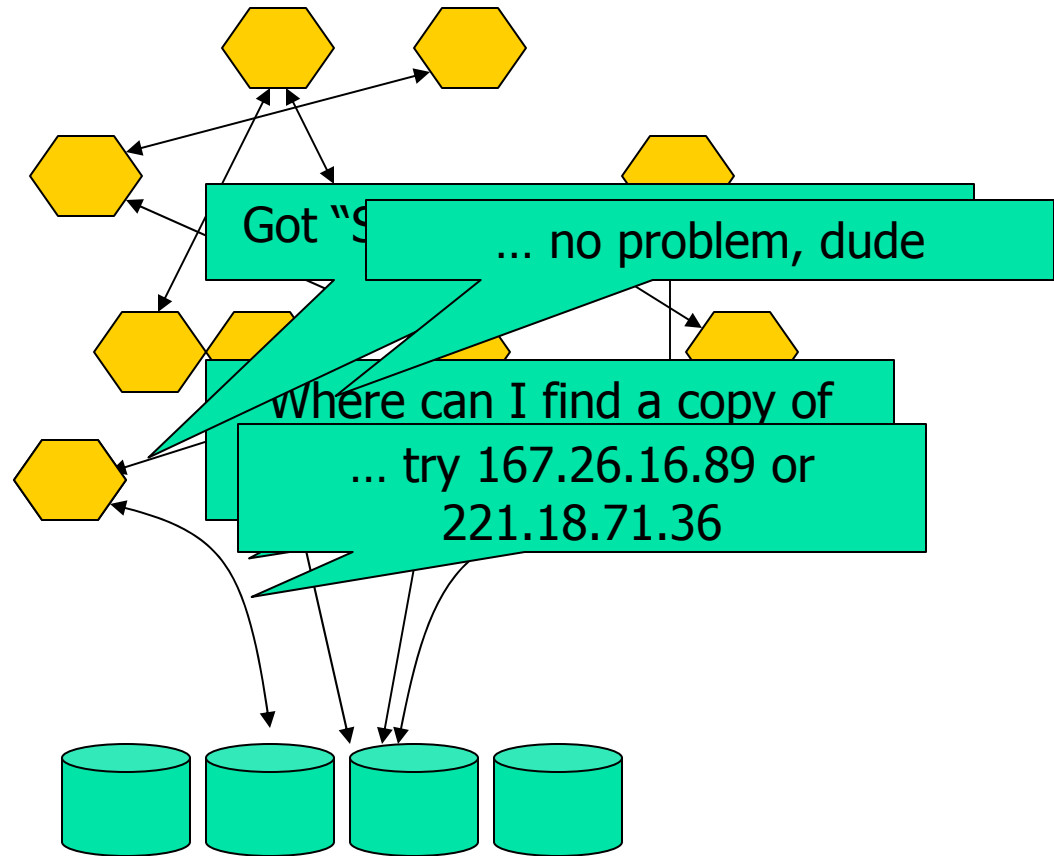


The first peer-to-peer system

- The term, and the intuition, emerged from the Napster file sharing service
 - In fact Napster *has* a set of servers
 - But these just keep a directory on behalf of clients and orchestrate publicity inserts
 - Servers build the web pages users see
 - Actual music and DVD downloads are done from client to client

Napster

Having obtained a top-level page listing peers with copies of music or other content desired, a client can download the files directly from the peer



Data center builds the pages users see when they access Napster



Quick aside

- Should “intellectual property” be free?
 - Topic of much debate right now
 - Lessig: “East Code vs West Code”
 - East Code is a term for “laws on the books”
 - West Code is a term for software
 - His point?
 - We need to evolve a balance between what we demand (law), what we can implement (code), and what will promote the general welfare
 - What regime gives the most benefit for the most people?



Why did Napster go this route?

- When service launched, developers hoped to work around legal limits on sharing media
 - They reasoned: let client systems advertise “stuff”
 - If some of that stuff happens to be music, that’s the responsibility of the person who does it
 - The directory system “helps clients advertise wares” but doesn’t “endorse” the sharing of protected intellectual property. Client who chooses to do so is violating the law
 - They make their money on advertising they insert
- Judges saw it differently...
 - “Napster’s clear purpose is to facilitate theft of IP...”



Characteristics of big populations

- With huge numbers of users
 - Surprisingly many “come and go” on short time scales
 - One study: mean residence time in Freenet was just a few seconds... and many clients were never heard of again!
 - British telcom reassigns IP addresses for all its networked users every few hours!



List of (technical) issues with Napster

- Many clients just aren't accessible
 - Firewalls can limit incoming connections to clients
 - Many client systems come and go (churn)
 - Round trip times to Nepal are slow...
 - Slow "upload" speeds are common connections
- Clients might withdraw a file unexpectedly
 - E.g. if low on disk space, or if they download something on top of a song they aren't listening to anymore



More (technical) issues with Napster

- Industry has been attacking the service... and not just in court of law
 - Denial of service assaults on core servers
 - Some clients lie about content (e.g. serve Frank Sinatra in response to download for Eminem)
 - Hacking Napster “clients” to run the protocol in various broken (disruptive) ways
 - And trying to figure out who is serving which files, in order to sue those people

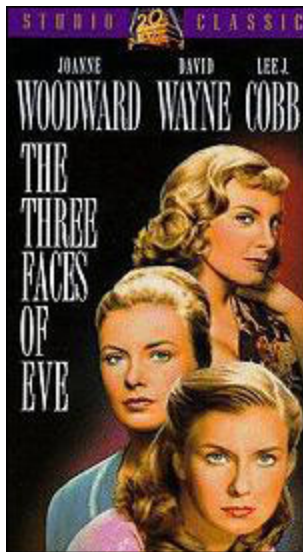


What problems are “fundamental”?

- If we assume clients serve up the same stuff people download, the number of sources for a *less popular item* will be very small
- Under assumption that churn is a constant, these less popular items will generally not be accessible.
- But experiments show that clients fall into two categories:
 - Well-connected clients that hang around
 - Poorly-connected clients that also churn
 - ... this confuses the question

What problems are fundamental?

- One can have, some claim, as many electronic personas as one has the time and energy to create. – *Judith S. Donath*.
- So-called “Sybil attack....”
 - Attacker buys a high performance computer cluster
 - It registers *many times* with Napster using a variety of IP addresses (maybe 10’s of thousands of times)
 - Thinking these are real, Napster lists them in download pages. Real clients get poor service or even get snared
 - Studies show that no p2p system can easily defend against Sybil attacks!





Refined Napster structure

- Early Napster just listed anything. Later:
 - Enhanced directory servers to probe clients, track their health. Uses an automated reporting of download problems to trim “bad sources” from list
 - Ranks data sources to preferentially list clients who...
 - Have been up for a long time, and
 - Seem to have fast connections, and
 - Appear to be “close” to the client doing the download (uses notion of “Internet distance”)
 - Implement parallel downloads and even an experimental method for doing “striped” downloads (first block from source A, second from source B, third from C, etc)
 - Leverages asymmetric download/uplink speeds



Meanwhile, p2p took off

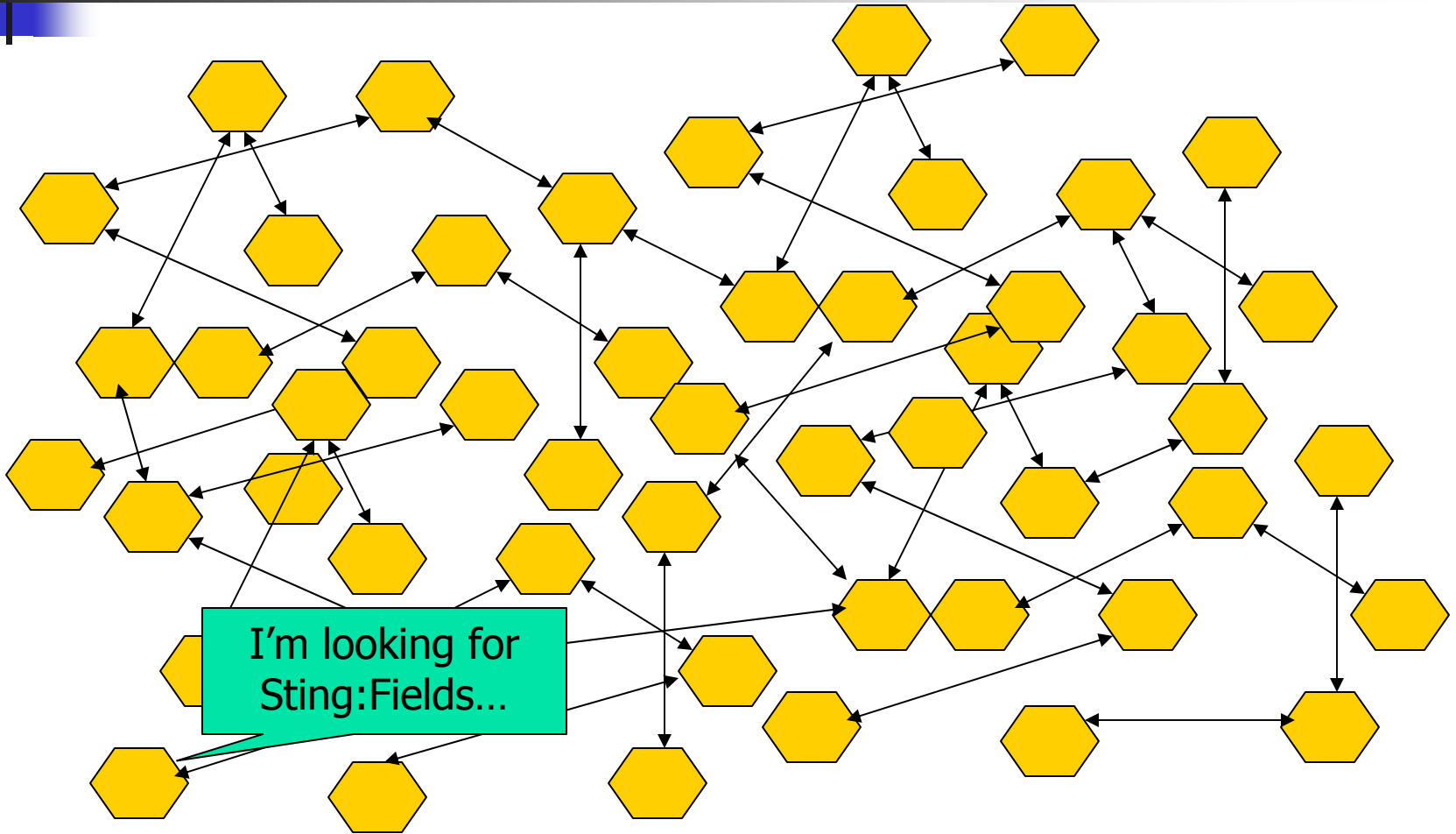
- By the time Napster was ruled illegal, it had 15 million users. 5 million of them joined in just a few months!
- With Napster out of business, a vacuum arose
 - Some users teamed up to define an open standard called “Gnutella” and to develop many protocol implementations
 - Gnutella eliminates the server
 - Judge singled it out in deciding that Napster was illegal
 - Also, a true peer-to-peer network seems harder to defeat than one that is only partly peer-to-peer
 - Credo: “All information should be free”



How Gnutella works

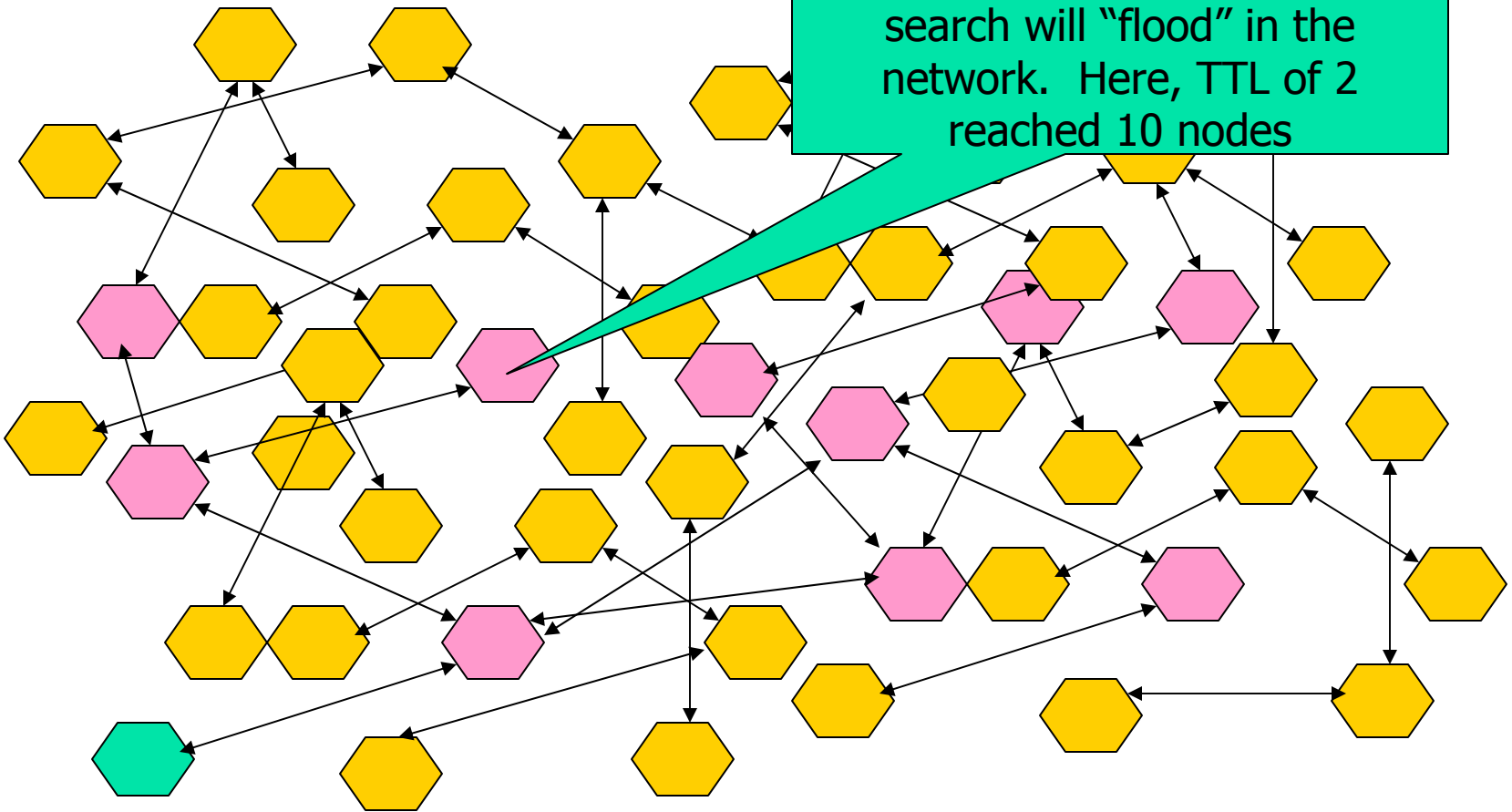
- Rough outline
 - User joins the network using a broadcast with increasing TTL values
 - “Is anyone out there?”
 - Links itself to the first Gnutella node to respond
 - To find content, protocol searches in a similar way
 - Broadcasts “I’m looking for Eminem:WhackHer”
 - Keeps increasing TTL value... eventually gives up if no system respond
 - Hopefully, popular content will turn up nearby

Self-organized "overlay" network

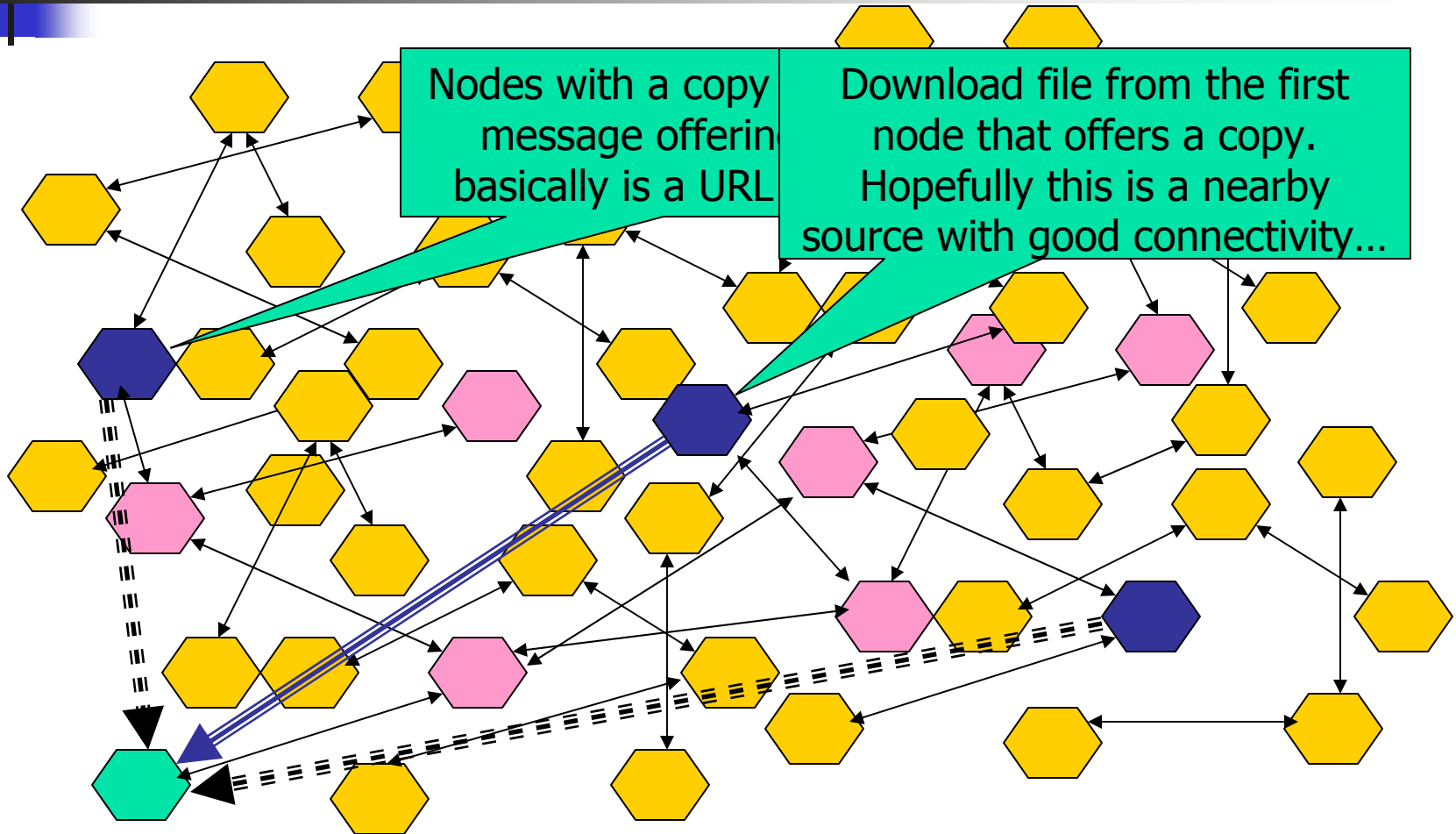


Self-organized "overlay" network

TTL determines how far the search will "flood" in the network. Here, TTL of 2 reached 10 nodes



Self-organized "overlay" network





Gnutella has “issues”

- In experimental studies of the system
 - Very high rates of join requests and queries are sometimes observed
 - Departures (churn) found to disrupt the Gnutella communication graph
 - Requests for rare *or misspelled* content turn into world-wide broadcasts
 - Rare is... um... rare. Misspellings are common.



Berkeley, MIT research in p2p

- Universities were first to view p2p as an interesting research area
 - CAN: “Content addressable network” proposed by Berkeley
 - Chord: MIT “distributed hash table”
- Both systems separate the “indexing” problem from actual storage

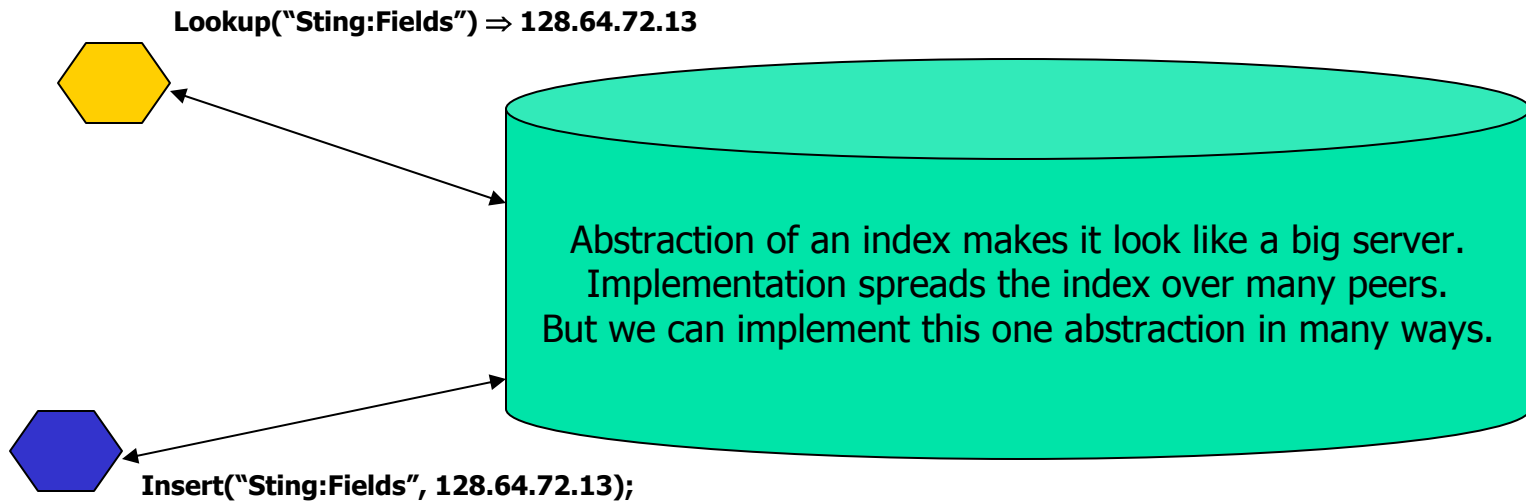


Distributed hash tables (DHTs)

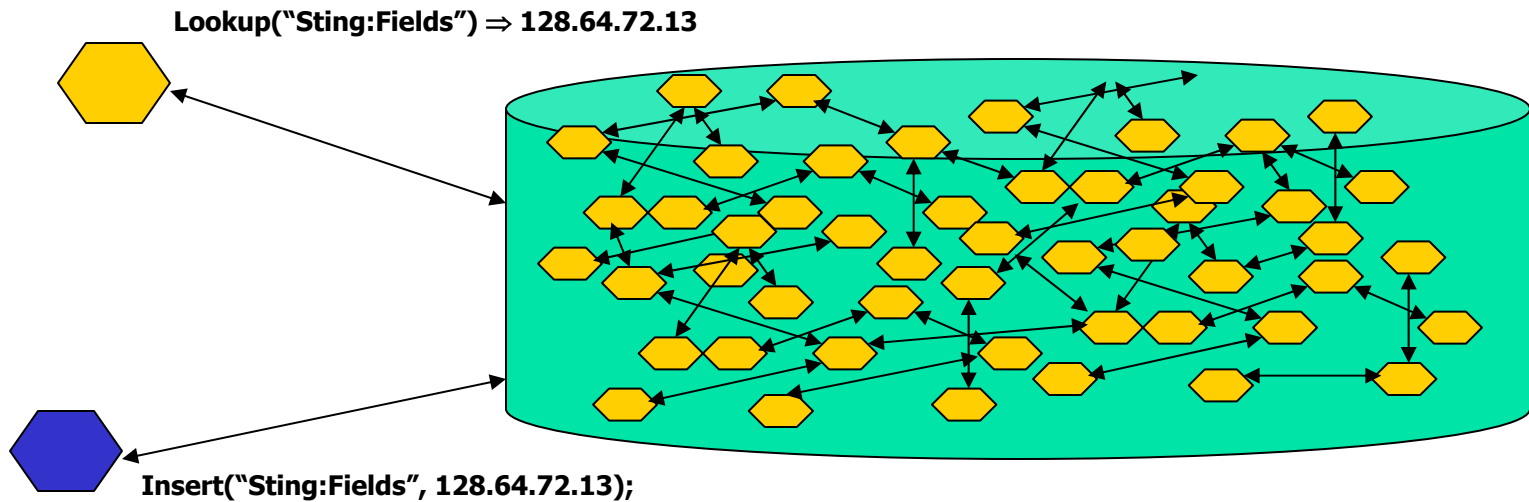
- Idea is to support a simple index with API:
 - Insert(key, value) – saves (key,value) tuple
 - Lookup(key) – looks up key and returns value
- Implement it in a p2p network, not a server...
 - Exactly *how* we implement it varies
 - Normally, each p2p client has just part of the tuples, hence must route query to the right place



Distributed indexing



Distributed indexing





Some details

- Keep in mind
 - There are lots of protocols that can solve this problem: *the protocol used is not part of the problem statement*
 - Some DHTs allow updates (e.g. if data moves, or nodes crash). Others are write once.
 - Most DHTs allow many tuples with the same key and can return the whole list, or a random subset of size k , etc



So what can we insert?

- Normally, we want to keep the values small... like an IP address
 - So the (key,value) pairs might tell us *where to look for something* but probably not *the actual thing*
 - Value could be (and often is) a URL
- Once we have the DHT running we can use it to build a p2p file system



DHTs: Area quickly took off

- Can, Chord: DHTs, already mentioned
- Pastry: From Rice and MSR, uses “Plaxton trees” (a kind of lookup tree)
- Tapestry: Berkeley (similar to Pastry)
- Kelips, Beehive: Cornell (use replication to get much faster responses)

... and too many more to list!



Representative research topics

- Can we make a DHT...
 - ... “resilient” to churn?
 - ... hide content and guarantee anonymity?
 - ... secure and robust against attack?
 - ... support high quality parallel striped downloads?
- Can we use a DHT...
 - To support scalable content distribution (IP multicast isn't popular with ISPs)?
 - To implement a new style of Internet addressing (i.e. replace IP routing or multicast)?



Are there legitimate uses of p2p file systems?

- One thought: corporations might want to index “everything in their file store” or to archive stuff
- Digital libraries might use p2p to avoid keeping extra copies of special or extremely big objects
- Risk of “bit rot” is a big concern
 - Suppose some huge set of PCs collaborates to preserve important documents
 - Might also encrypt them – various options exist...
 - How many replicas needed to avoid risk that “rare events” will destroy all copies simultaneously?
 - A topic of study in Oceanstore and at UCSD



Are there legitimate uses of p2p file systems?

- p2p could be a great way to legally share information within a team of collaborators at work, or some other “interest group”
 - Think of these as little groups superimposed on a massive p2p network using the same technology
 - Idea would be: “We help each other out”
- Some argue that p2p systems could be valuable in resisting repressive political regimes
 - Like “coffee house” meetings in pre-revolutionary Russia
 - Can repressive regimes survive if they can’t control the flow of information?



Spyware: The real thing

- Imagine a *popular* p2p system that
 - Encrypts content: need key to make sense of it
 - Achieves a high degree of anonymity
 - Pretty much everyone helps to serve each request, but nobody actually has a copy of the whole file on their drive – e.g. I have a few bits, you have a few bits
 - Real sources and nodes accessing content concealed from intruders
 - Robust against disruptive attack
- Needs to be popular: Spies hide in crowds



Philosophical debate

- Is technology “political”?
 - Here we have a technology invented to
 - Rip off IP from owners
 - Conceal crime from law enforcement
 - Pretty much unstoppable without incredibly intrusive oversight mechanisms
 - What’s the story here? Are we all anarchists?
- Some people believe technology is negative, some positive, some neutral
 - What about p2p technology?
 - Are we allowed to answer “all of the above”?



p2p outside of file sharing

- Key idea was that p2p systems could “gossip” about replicated data
 - Now and then, each node picks some “peer” (at random, more or less)
 - Sends it a snapshot of its own data
 - Called “push gossip”
 - Or asks for a snapshot of the peer’s data
 - “Pull” gossip
 - Or both: a push-pull interaction

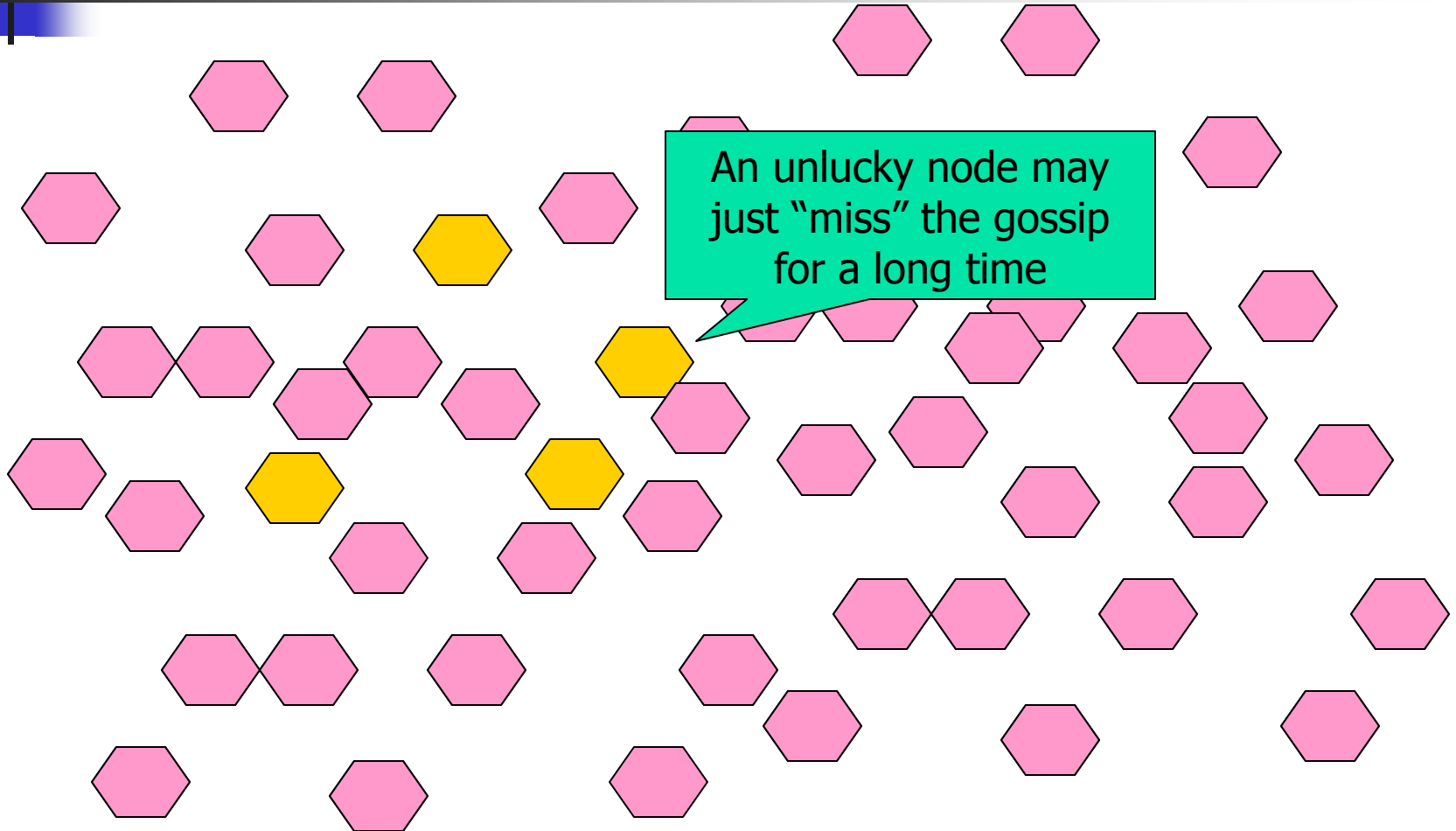


Gossip “epidemics”

- [t=0] Suppose that I know something
- [t=1] I pick you... Now two of us know it.
- [t=2] We each pick ... now 4 know it...
- Information spread: exponential rate.
 - Due to re-infection (gossip to an infected node) spreads as 1.8^k after k rounds
 - But in $O(\log(N))$ time, N nodes are infected

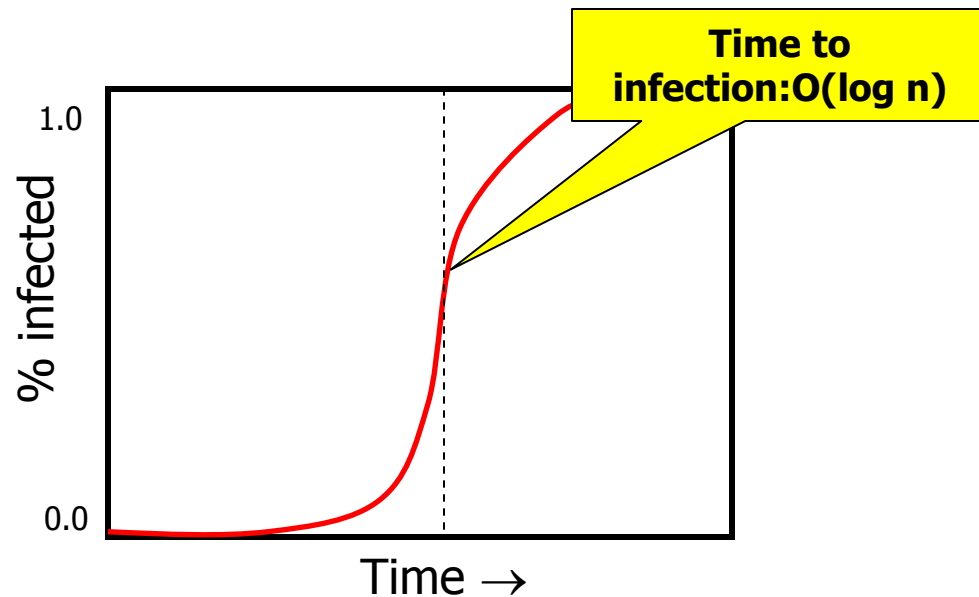


Gossip epidemics



Gossip scales very nicely

- Participants' loads independent of size
- Network load linear in system size
- Data spreads in $\log(\text{system size})$ time





Facts about gossip epidemics

- Extremely robust
 - Data travels on exponentially many paths!
 - Hard to even slow it down...
 - Suppose 50% of our packets are simply lost...
 - ... we'll need 1 additional round: a trivial delay!
 - Push-pull works best. For push-only/pull-only a few nodes can remain uninfected for a *long time*
- Later we'll see that many optimizations are needed in practice... but the approach works!



Uses of gossip epidemics

- To robustly multicast data
 - Slow, but very sure of getting through
- To repair inconsistency in replicas
- To support “all to all” monitoring and distributed management
- For distributed data mining and discovery



A contemporary perspective

- p2p computing has many pros and many cons, and for most purposes the cons outweigh the pros
 - A “hard to control” technology
 - Firewalls cause many annoyances
 - Rather slow to propagate updates
- But at the same time
 - Incredibly robust against disruption



Contemporary response?

- So... use p2p techniques, but mostly
 - In data centers or LANs where there are no firewalls
 - In uses where slow update times aren't an issue
- Often means that we need to marry p2p mechanism to a more "urgent" protocol like our multicast protocols



Peek ahead

- We'll look at several p2p technologies
 - Chord, Pastry, Kelips: three DHTs
 - Bimodal Multicast: Uses gossip in a multicast protocol to get superior scalability
 - Astrolabe: Uses gossip to implement a scalable monitoring, management and control infrastructure (also great for data mining)