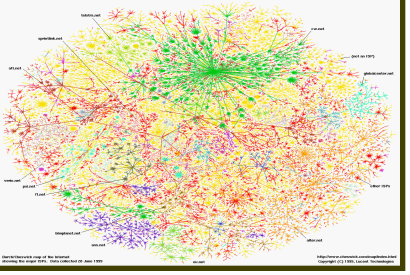
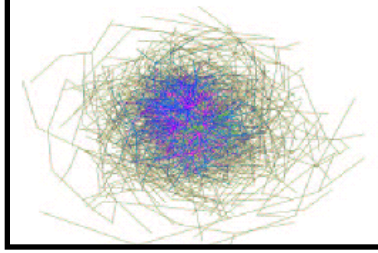


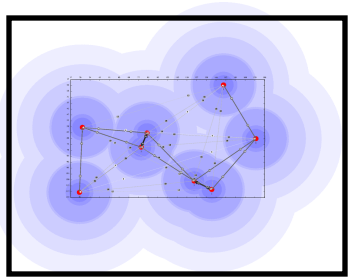


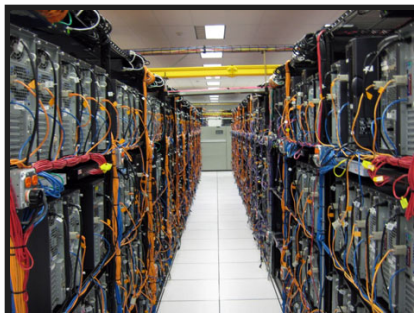
# CS525
# Advanced Distributed Systems
# Spring 2015

Indranil Gupta (Indy)

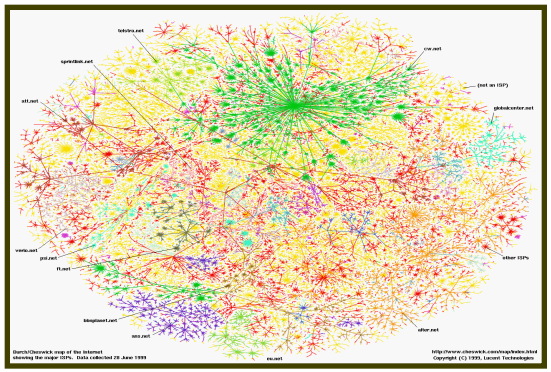
Lecture 1

January 20, 2015

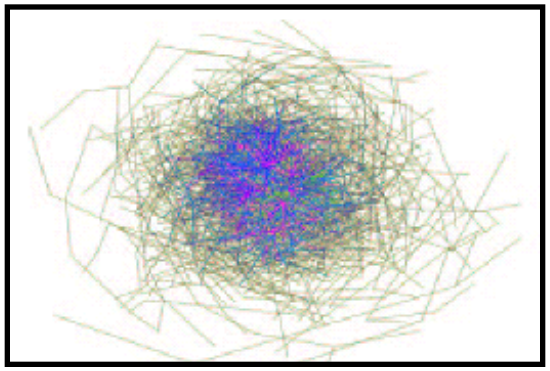




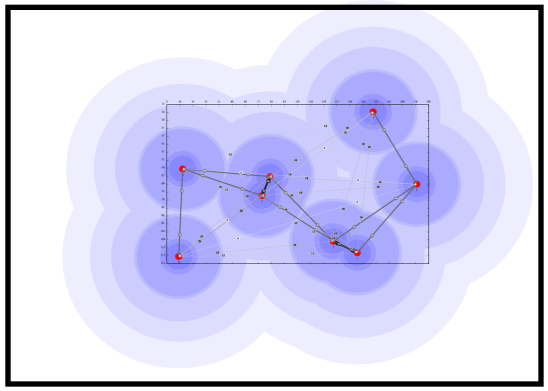
# What is a Distributed System?
## (examples)



The Internet



Gnutella peer to peer system



A Sensor Network

Datacenter/Cloud

# Can you name some examples of Operating Systems?

# Can you name some examples of Operating Systems?

…

Linux WinXP Vista Unix FreeBSD Mac OSX

2K Aegis Scout Hydra Mach SPIN

OS/2 Express Flux Hope Spring

AntaresOS EOS LOS SQOS LittleOS TINOS

PalmOS WinCE TinyOS

…

# What is an Operating System?

# What is an Operating System?

- User interface to hardware (device driver)
- Provides abstractions (processes, file system)
- Resource manager (scheduler)
- Means of communication (networking)
- …

# FOLDOC definition

- The low-level software which handles the interface to peripheral hardware, schedules tasks, allocates storage, and presents a default interface to the user when no application program is running.
- The OS may be split into a kernel which is always present and various system programs which use facilities provided by the kernel to perform higher-level house-keeping tasks, often acting as servers in a client-server relationship.
- Some would include a graphical user interface and window system as part of the OS, others would not. The operating system loader, BIOS, or other firmware required at boot time or when installing the operating system would generally not be considered part of the operating system, though this distinction is unclear in the case of a roamable operating system such as RISC OS.
- The facilities an operating system provides and its general design philosophy exert an extremely strong influence on programming style and on the technical cultures that grow up around the machines on which it runs.

# Can you name some examples of Distributed Systems?

# Can you name some examples of Distributed Systems?

- Client-server (e.g., NFS)
- The Internet
- The Web
- A sensor network
- DNS
- BitTorrent (peer to peer overlay)
- Datacenters
- Hadoop

# What is a Distributed System?

# FOLDOC definition

A collection of (probably heterogeneous) automata whose distribution is transparent to the user so that the system appears as one local machine. This is in contrast to a network, where the user is aware that there are several machines, and their location, storage replication, load balancing and functionality is not transparent. Distributed systems usually use some kind of client-server organization.

# Textbook definitions

- A distributed system is a collection of independent computers that appear to the users of the system as a single computer.

  [Andrew Tanenbaum]

- A distributed system is several computers doing something together. Thus, a distributed system has three primary characteristics: multiple computers, interconnections, and shared state.

  [Michael Schroeder]

# Unsatisfactory

- Why are these definitions short?
- Why do these definitions look inadequate to us?
- Because we are interested in the insides of a distributed system
  - algorithmics
  - design and implementation
  - maintenance
  - study

I shall not today attempt further to define the kinds of material I understand to be embraced within that shorthand description; and perhaps I could never succeed in intelligibly doing so. But I know it when I see it…

[Potter Stewart, Associate Justice, US Supreme Court (talking about his interpretation of a technical term laid down in the law, case Jacobellis versus Ohio 1964) ]

# A working definition for us

*A distributed system is a collection of entities, each of which is autonomous, programmable, asynchronous and failure-prone, and which communicate through an unreliable communication medium.*

- Our interest in distributed systems involves
  - algorithmics, design and implementation, maintenance, study
- Entity=a process on a device (PC, PDA, mote)
- Communication Medium=Wired or wireless network

# A range of interesting problems for Distributed System designers

- 
- P2P systems [Gnutella, Kazaa, BitTorrent]
- Cloud Infrastructures [AWS, Azure, GCE]
- Cloud Storage [Key-value stores, NoSQL, BigTable]
- Cloud Programming [MapReduce, Pig, Hive, Storm, Pregel]
- Coordination [Paxos]
- Routing [Sensor Networks, Internet]
- 

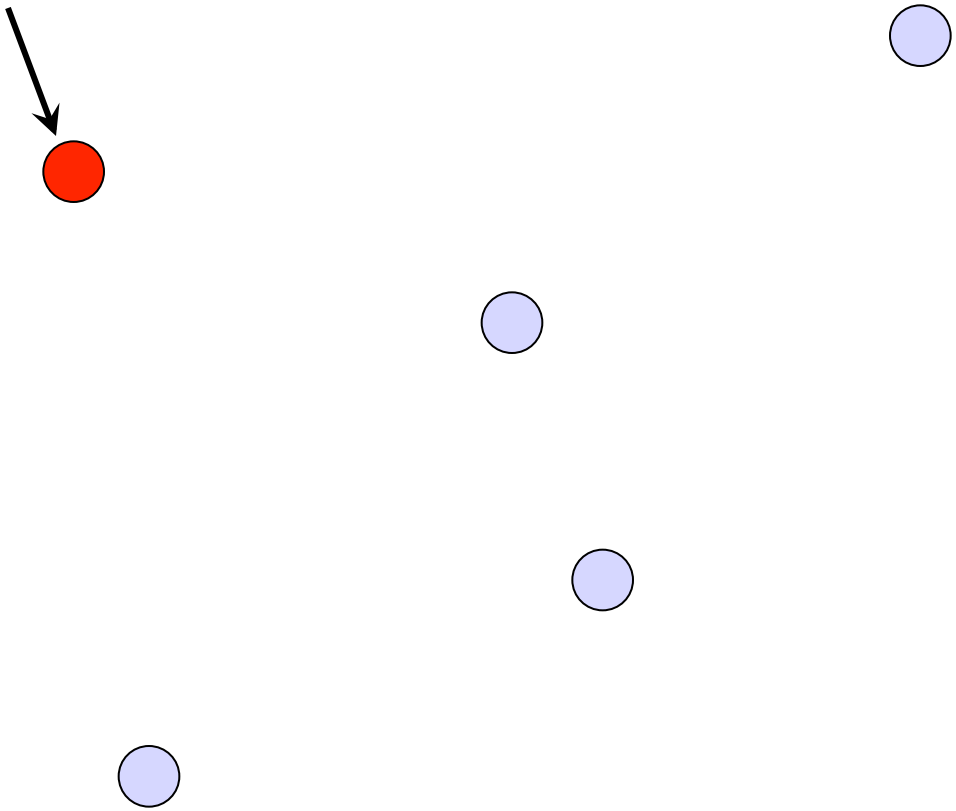
# A range of challenges

- 
- Failures: no longer the exception, but rather a norm
- Scalability: 1000s of machines, Terabytes of data
- Asynchrony: clock skew and clock drift
- Security: of data, users, computations, etc.
- 

# Multicast

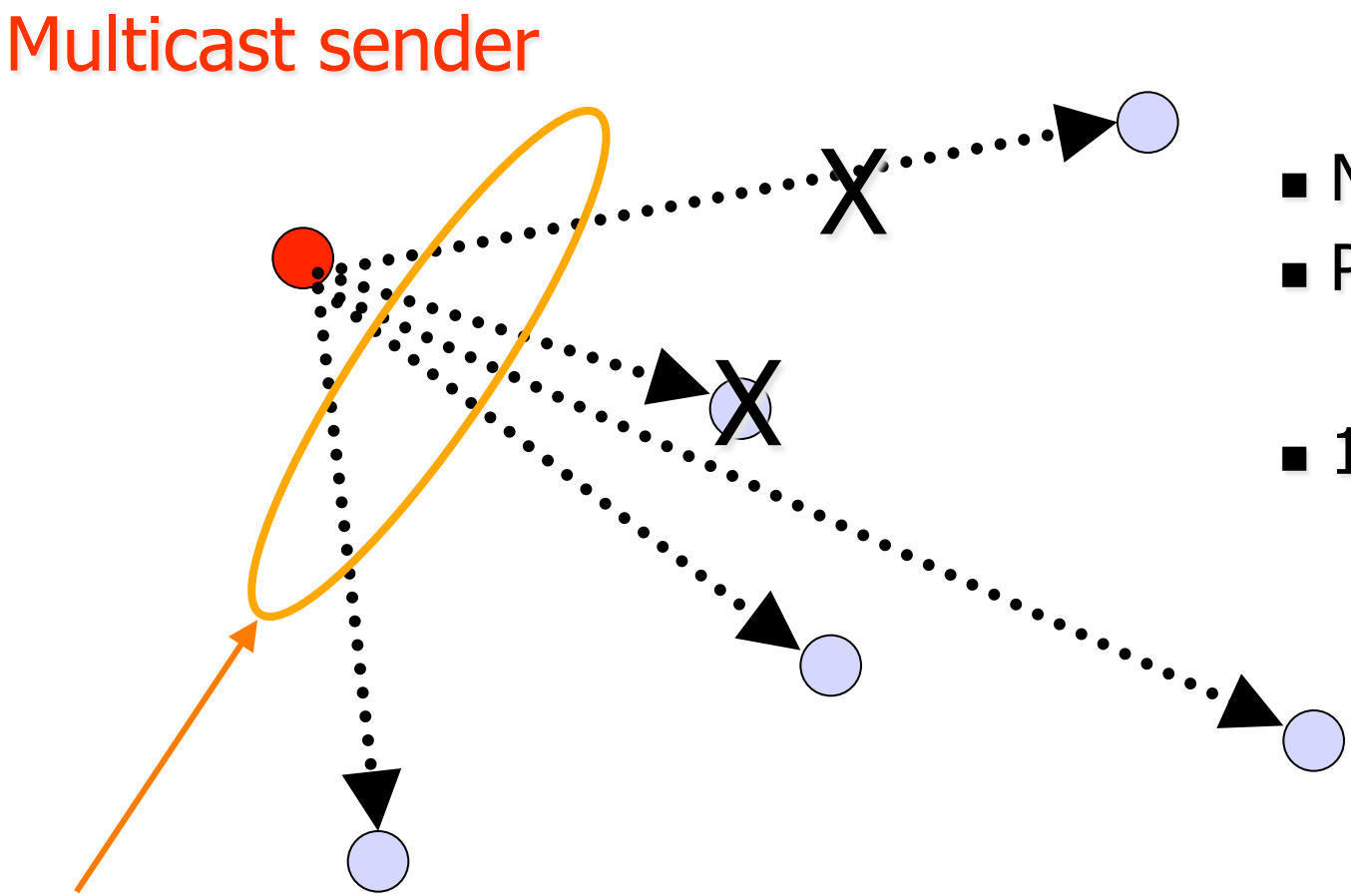
# Multicast

Node with a piece of information to be communicated to everyone



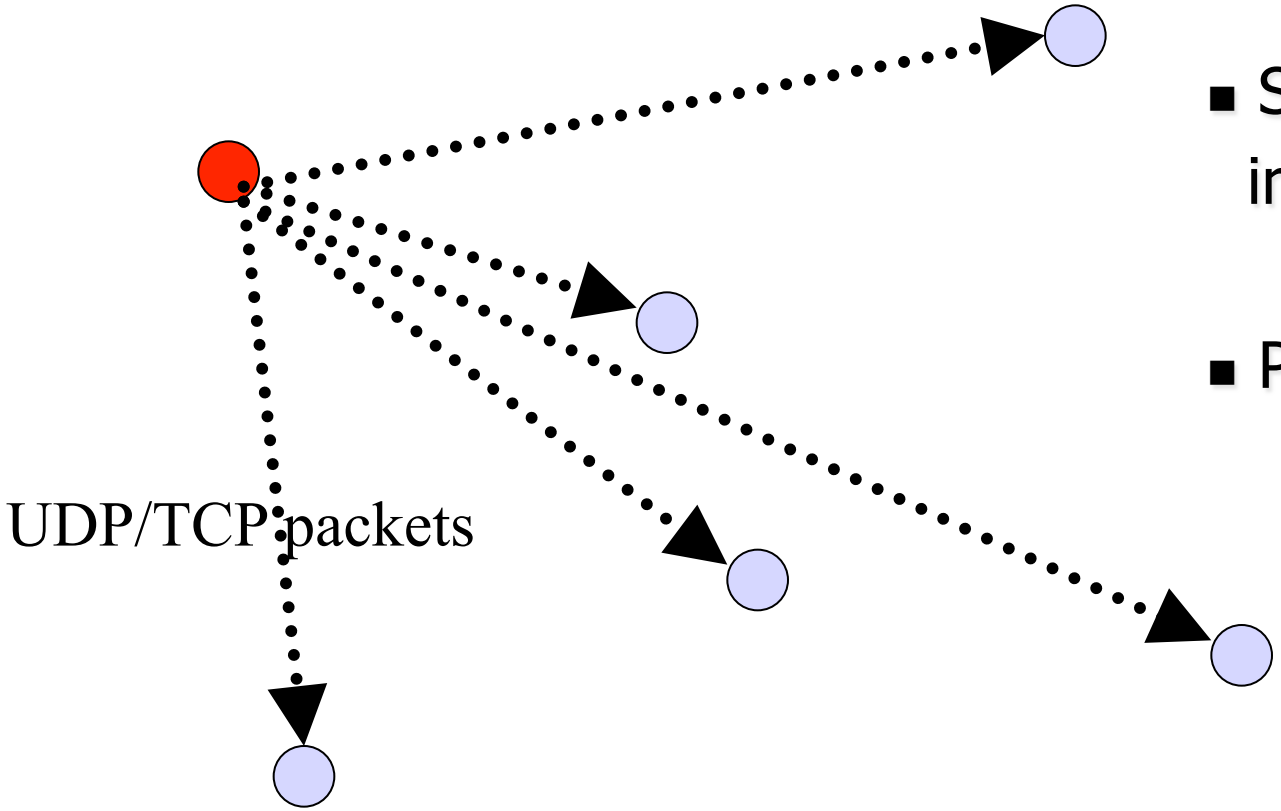
Distributed Group of "Nodes"= Processes at Internet-based hosts

# Fault-tolerance and Scalability



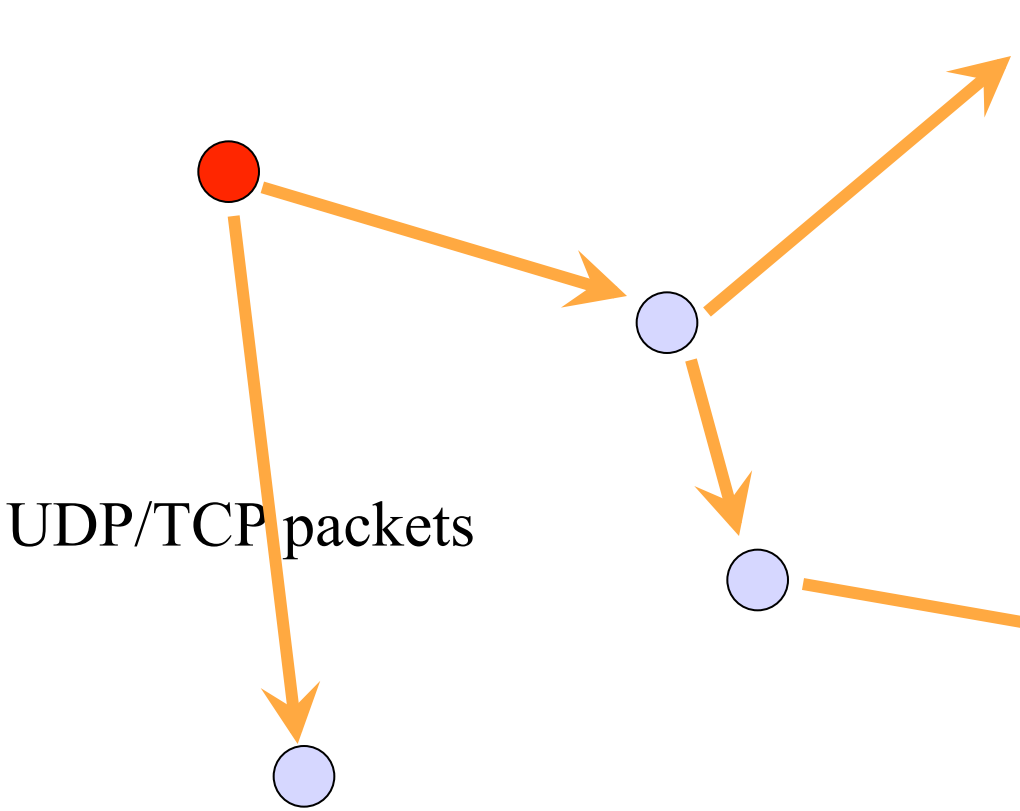

- Nodes may crash
- Packets may be dropped
- 1000's of nodes

# Centralized




- Simplest implementation
- Problems?

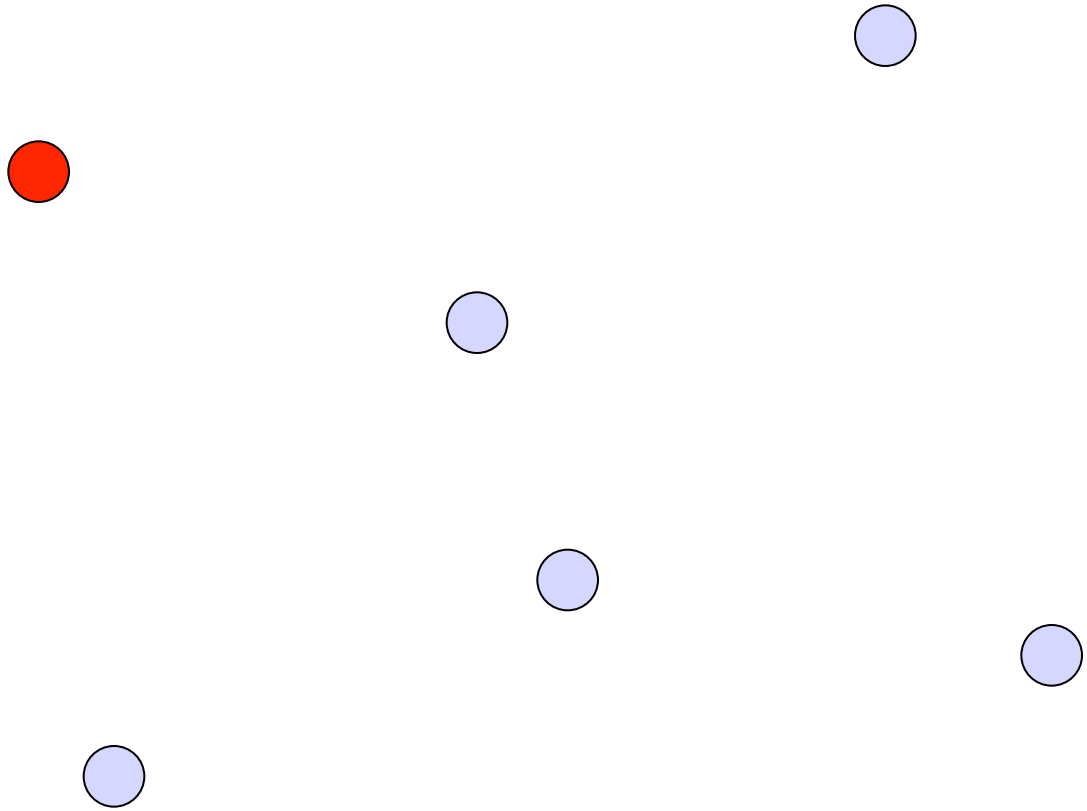
# Tree-Based




- e.g., IPmulticast, SRM RMTP, TRAM,TMTP
- Lower load per node
- Tree setup and maintenance
- Problems?

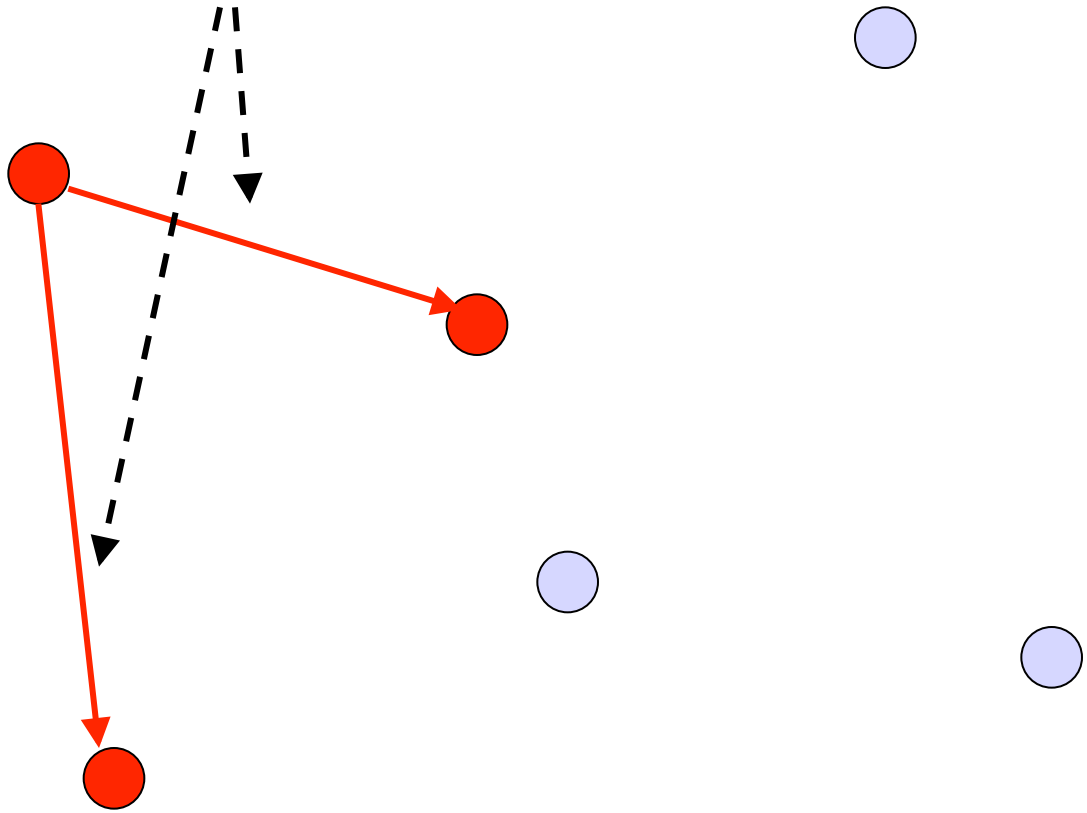
# A Third Approach

Multicast sender



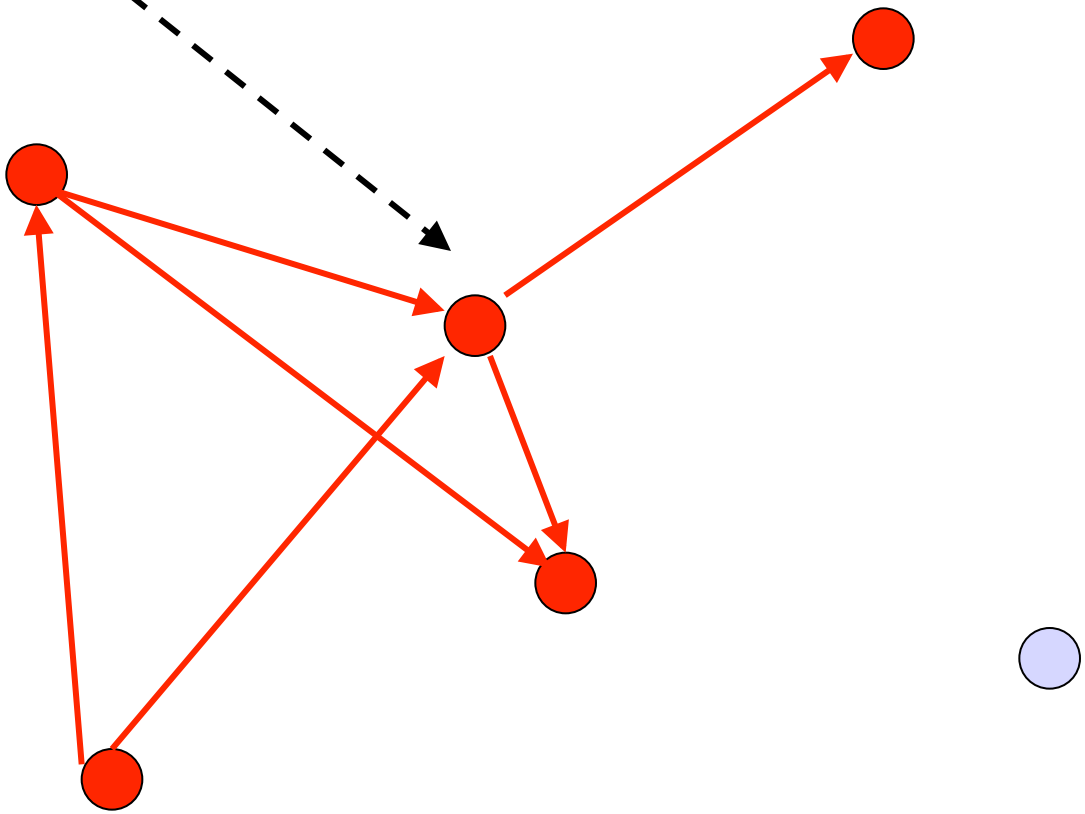
Periodically, transmit to *b* random targets

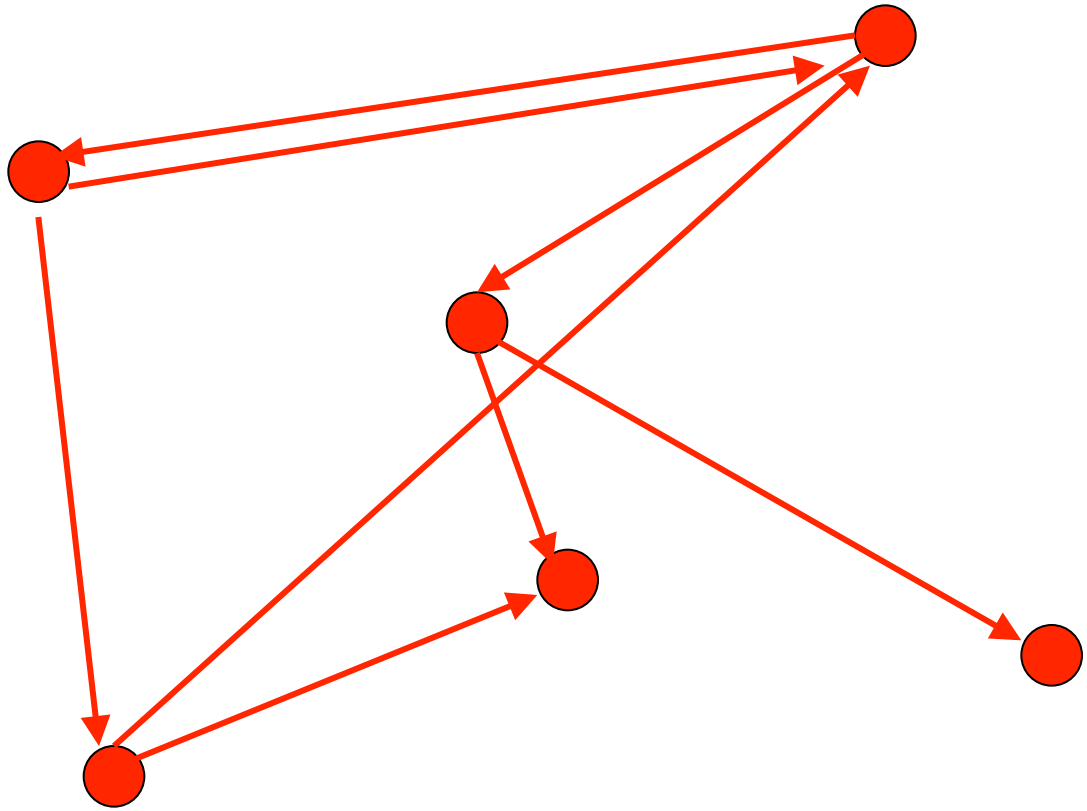
Gossip messages (UDP)



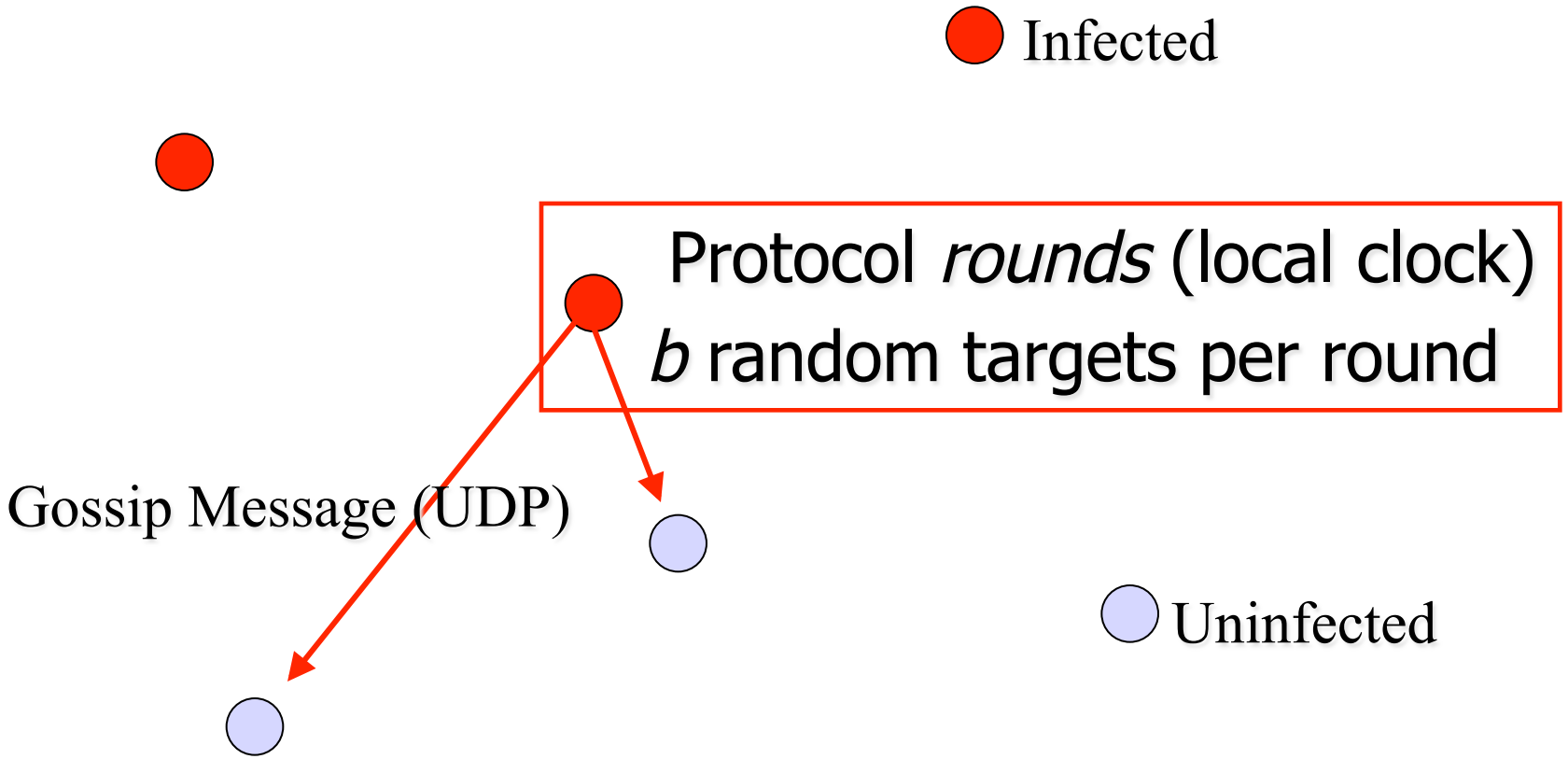
Other nodes do same after receiving multicast

Gossip messages (UDP)





# “Epidemic” Multicast (or “Gossip”)



# Properties

Claim that this simple protocol

- Is lightweight in large groups
- Spreads a multicast quickly
- Is highly fault-tolerant

# Analysis

From old mathematical branch of *Epidemiology* [Bailey 75]

- Population of $(n+1)$ individuals mixing homogeneously
- Contact rate between any individual pair is $\beta$
- At any time, each individual is either uninfected (numbering $x$) or infected (numbering $y$)
- Then, $x_0 = n, y_0 = 1$

  and at all times $x + y = n + 1$
- Infected–uninfected contact turns latter infected, and it stays infected

# Analysis (contd.)

- Continuous time process
- Then

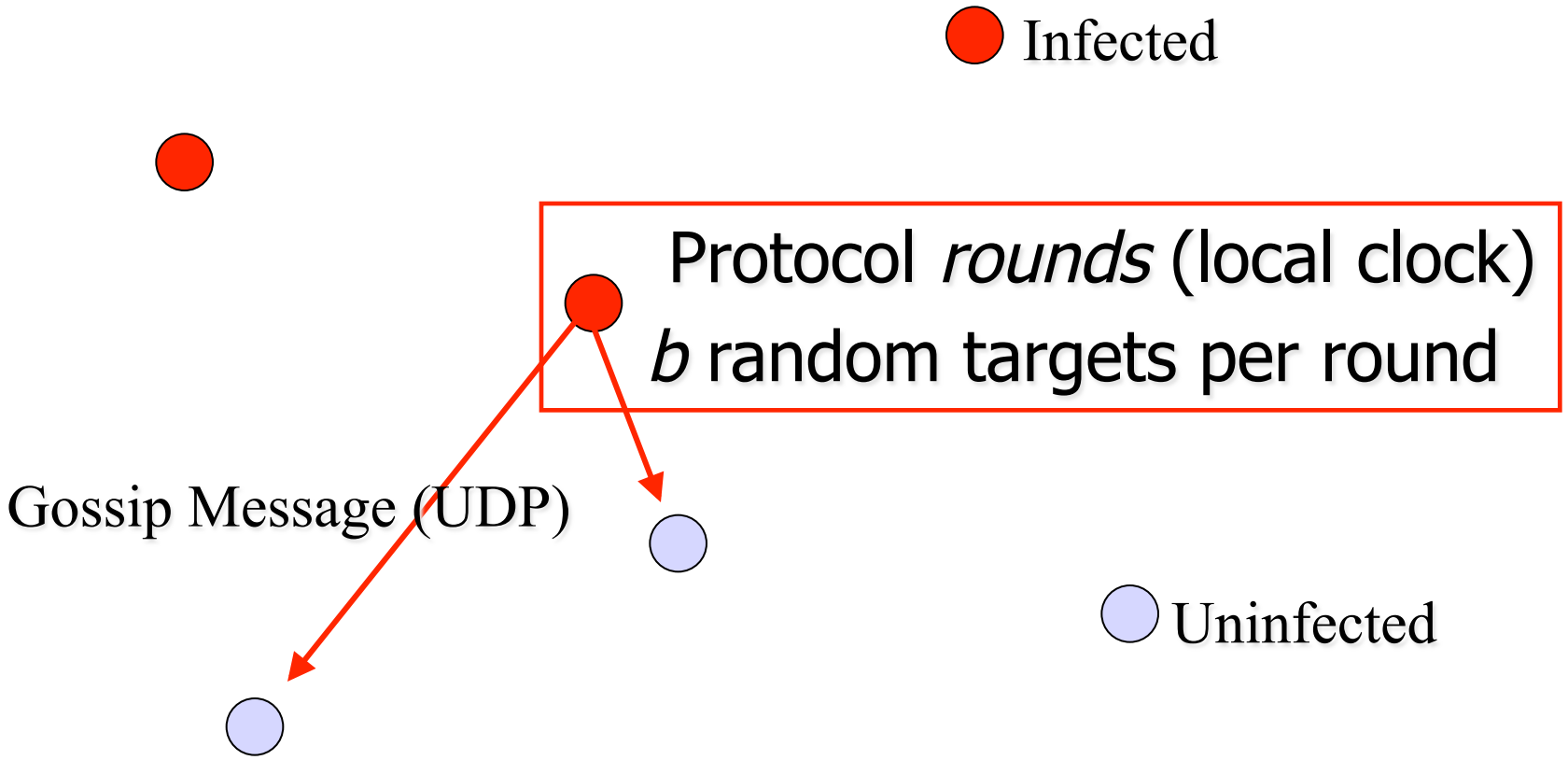
$$\frac{dx}{dt} = -\beta xy \qquad \text{(why?)}$$

with solution

$$x = \frac{n(n+1)}{n + e^{\beta(n+1)t}}, y = \frac{(n+1)}{1 + ne^{-\beta(n+1)t}}$$

(correct? can you derive it?)

# Epidemic Multicast



# Epidemic Multicast Analysis

$$\beta = \frac{b}{n}$$ (why?)

Substituting, at time $t=clog(n)$, num. infected is

$$y \approx (n+1) - \frac{1}{n^{cb-2}}$$

(correct? can you derive it?)

# Analysis (contd.)

- Set $c,b$ to be small numbers independent of $n$
- Within $clog(n)$ rounds, [**low latency**]
  - all but $\frac{1}{n^{cb-2}}$ number of nodes receive the multicast [**reliability**]
  - each node has transmitted no more than $cblog(n)$ gossip messages [**lightweight**]

# Fault-tolerance

- Packet loss
  - 50% packet loss: analyze with *b* replaced with *b/2*
  - To achieve same reliability as 0% packet loss, takes twice as many rounds
- Node failure
  - 50% of nodes fail: analyze with *n* replaced with *n/2* and *b* replaced with *b/2*
  - Same as above

# Fault-tolerance

- With failures, is it possible that the epidemic might die out quickly?
- Possible, but improbable:
  - Once a few nodes are infected, with high probability, the epidemic will not die out
  - So the analysis we saw in the previous slides is actually behavior *with high probability*

  [Galey and Dani 98]
- Think: why do rumors spread so fast? why do infectious diseases cascade quickly into epidemics? why does a virus or worm spread rapidly?

# So,…

- Is this all theory and a bunch of equations?
- Or are there implementations yet?

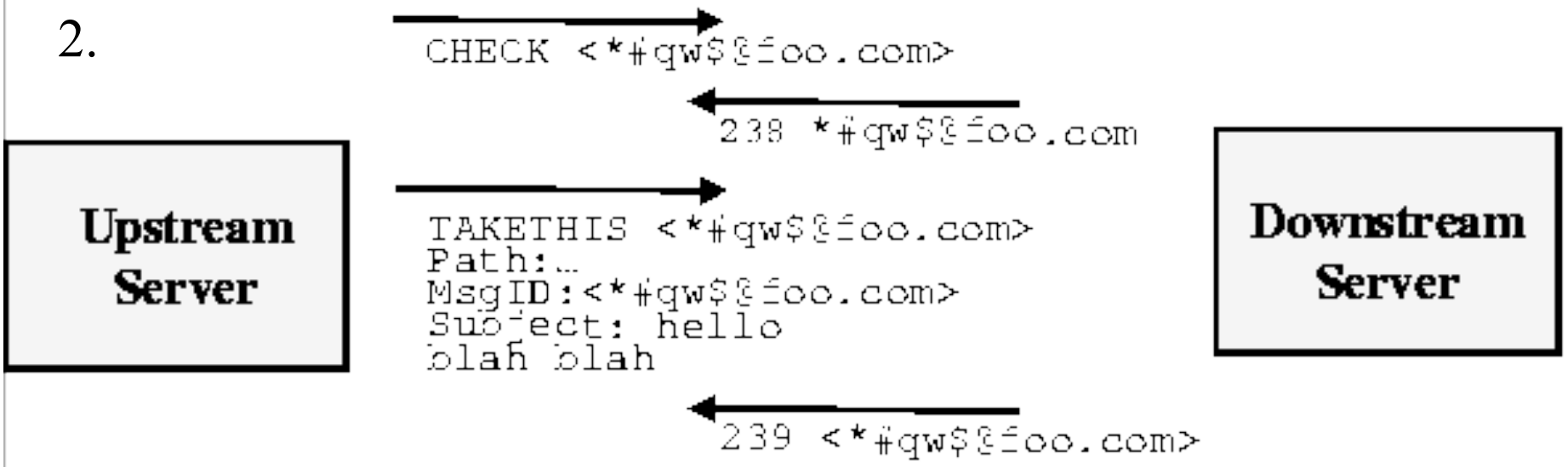
# Some implementations

- Clearinghouse and Bayou projects: email and database transactions [PODC ‘87]
- refDBMS system [Usenix ‘94]
- Bimodal Multicast [ACM TOCS ‘99]
- Sensor networks [Li Li et al, Infocom ’02, and PBBF, ICDCS ‘05]
- Usenet NNTP (Network News Transport Protocol) ! [‘79]
- AWS EC2 and S3 Cloud (rumored). [’00s]

# NNTP Inter-server Protocol

1. Each client uploads and downloads news posts from a news server

2.

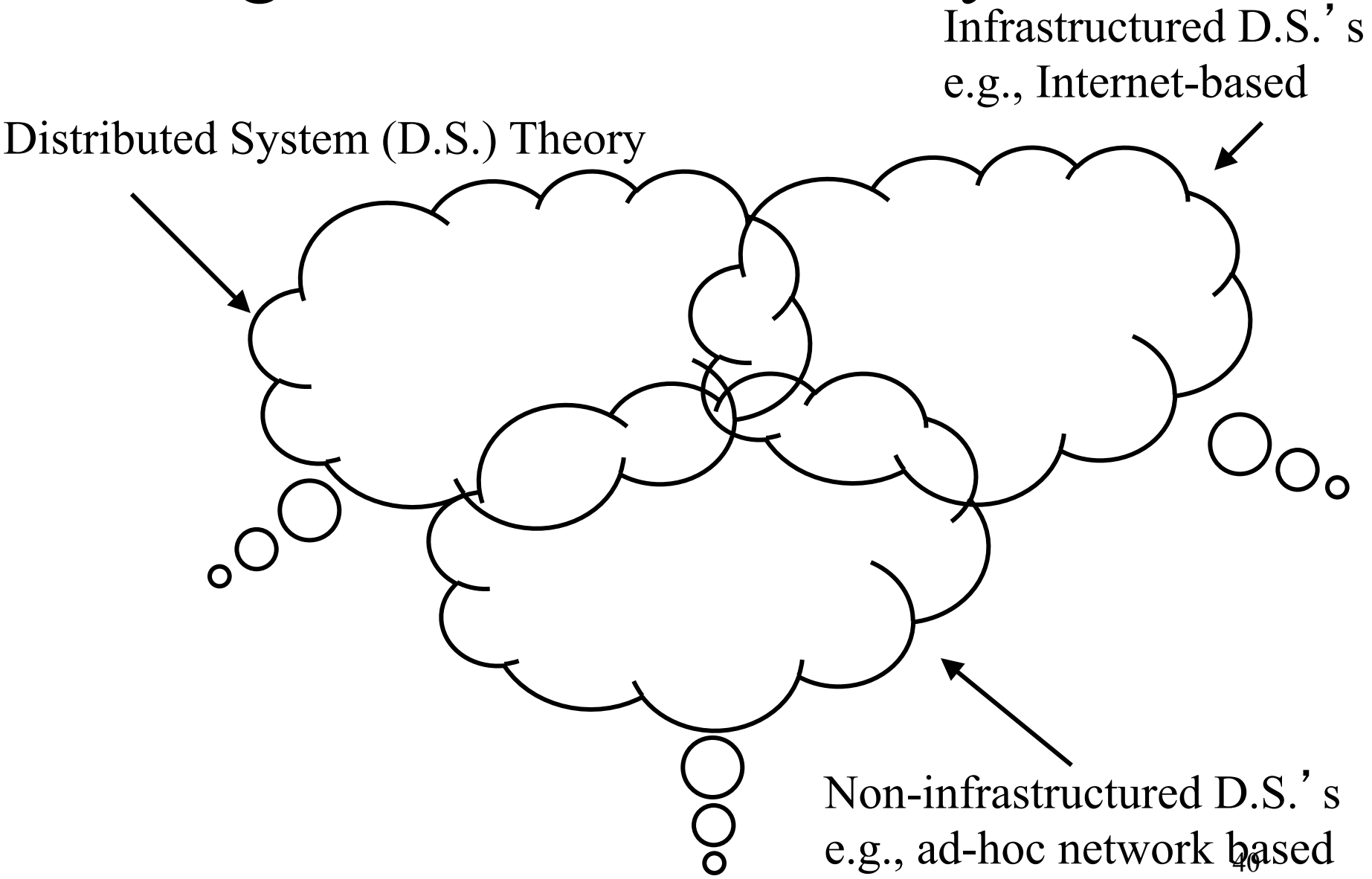



Server retains news posts for a while,
transmits them lazily, deletes them after a while

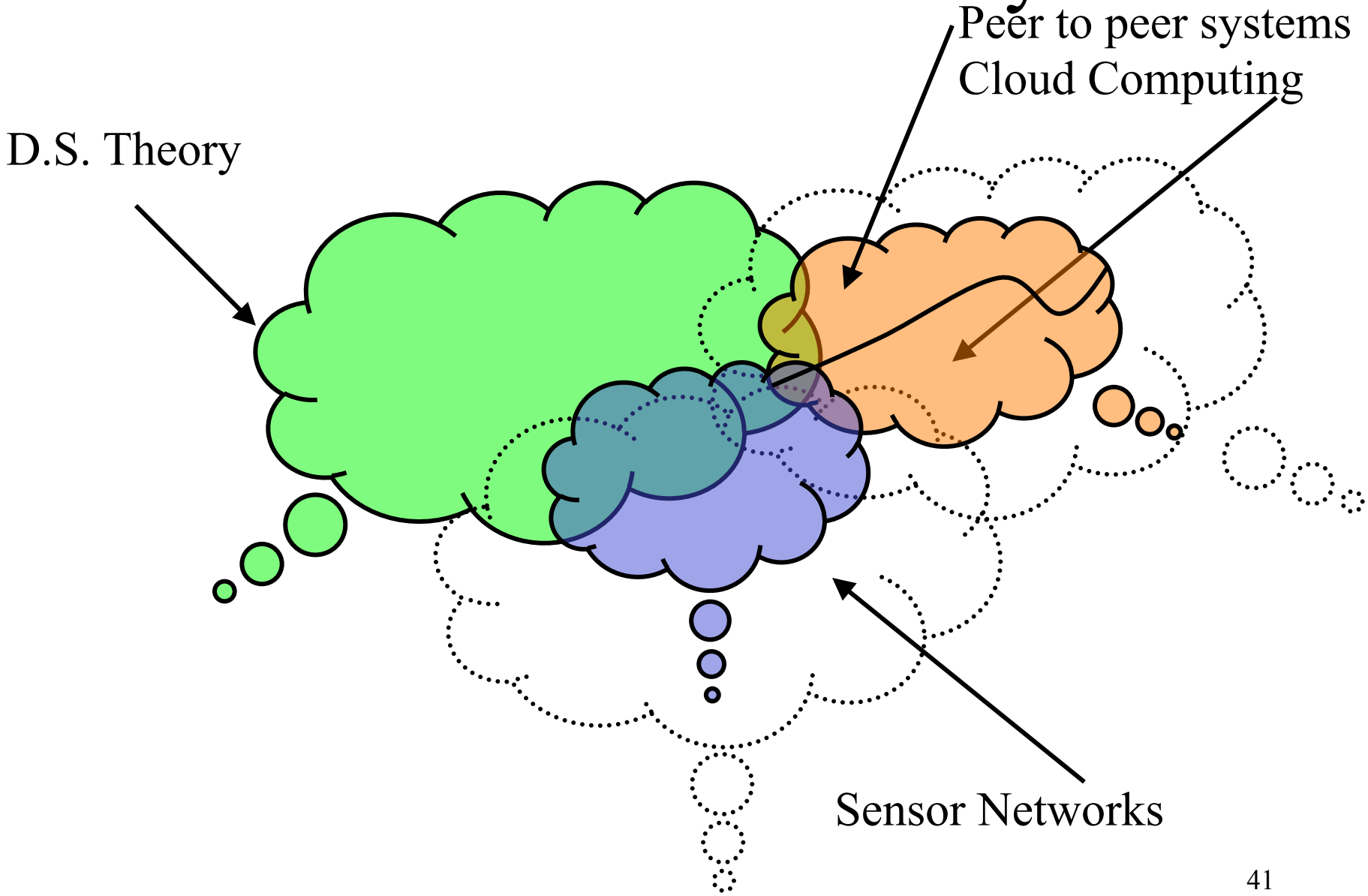
We'll cover some of these other implementations during the course

- But let's dwell on the big picture of the course

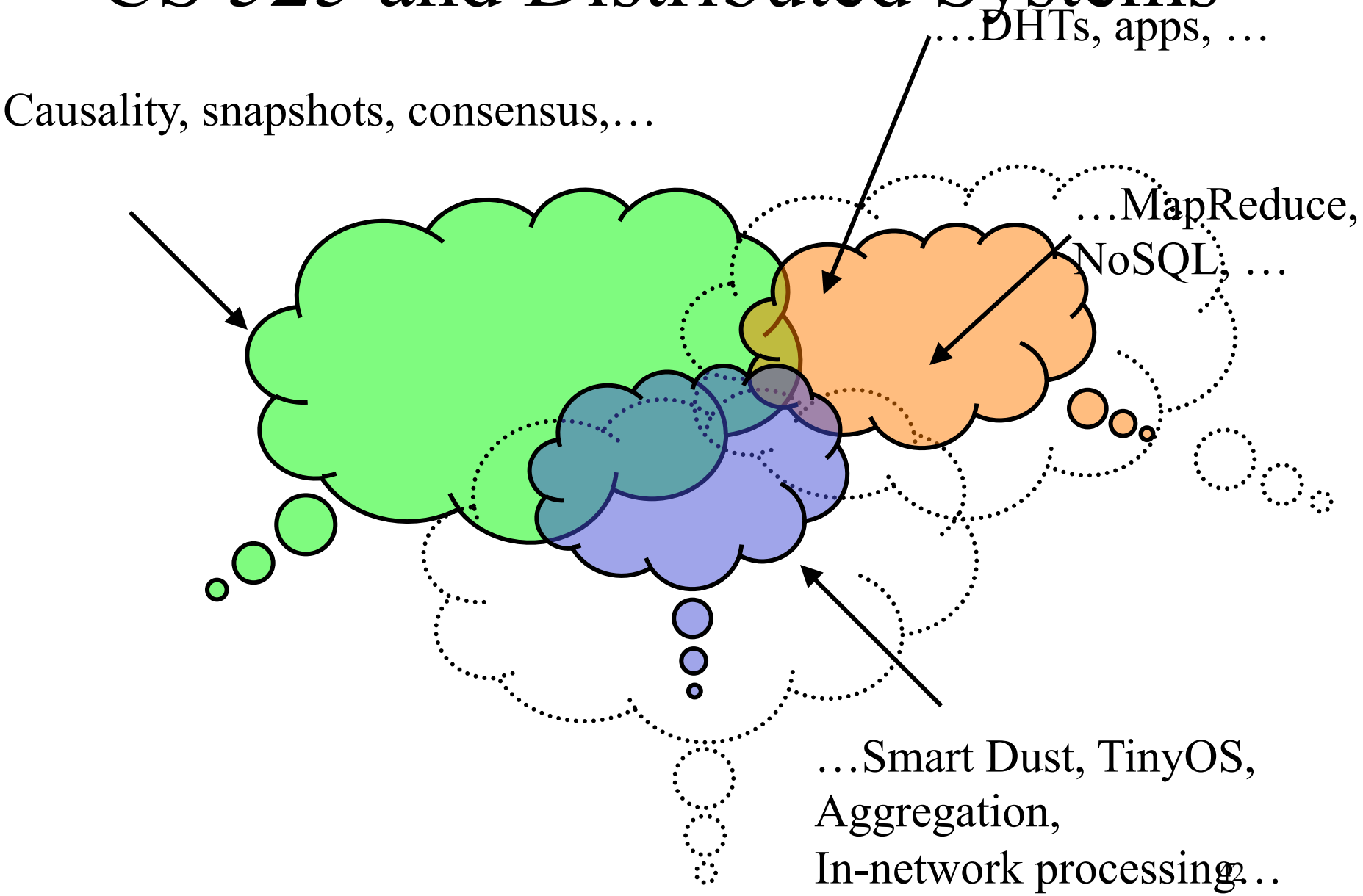
# Angles of Distributed Systems



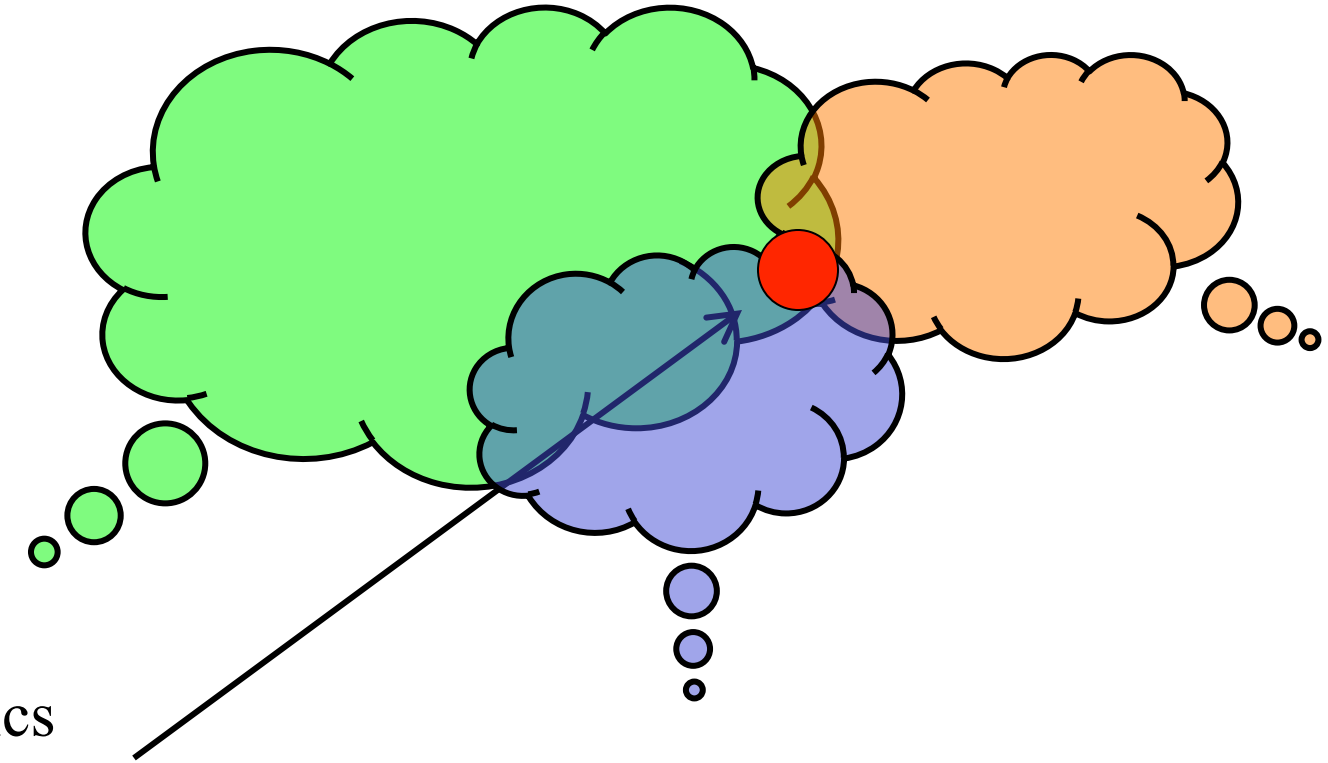
# CS 525 and Distributed Systems



# CS 525 and Distributed Systems



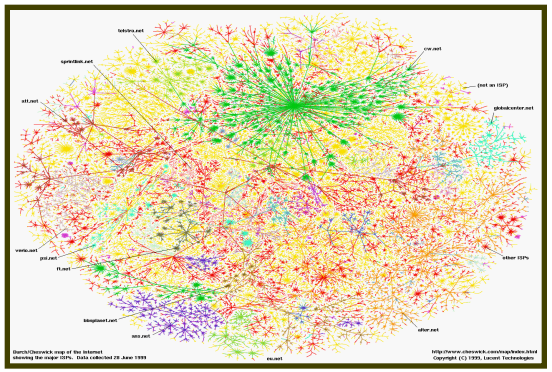
# Interesting: Area Overlaps



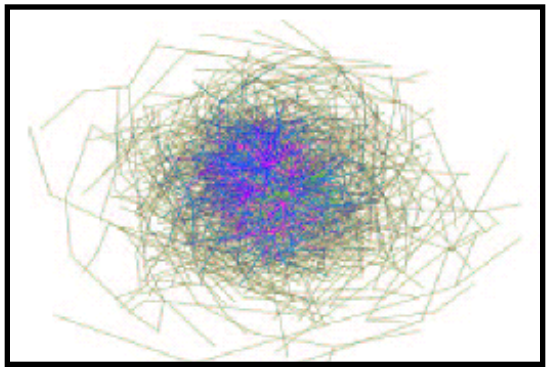
Epidemics
NNTP
Gossip-based ad-hoc routing

# Interesting: Area Overlaps

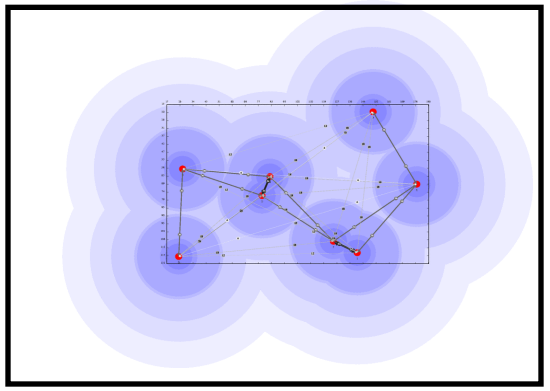
***Do projects that are either entrepreneurial or research***



The Internet



Gnutella peer to peer system



A Sensor Network

Clouds

# Entrepreneurial Project

- Proposes new ideas that can be accommodated into a (your own!) startup
  - Company
  - Or non-profit
- Has to be a marketable product or services to users
  - Need to write a short Business Plan
- You don't actually need to found a company; you need to develop ideas for it. What you do later with it is up to you only.
- EnterpriseWorks incubator at Illinois
- Has to use or leverage concepts from the CS525 class
- To help you get leverage the current and bleeding edge of d.s. research, we will read 2 research papers per class

# Research Project

- Your project has to be related to distributed systems
- Different from your ongoing research projects (ask Indy)
- Must show keen awareness of the current state of the art
- Must solve thoroughly at least one practical research problem
- Must have innovative ideas and originality (algorithms)
- Must build a real system and evaluate it in deployment
- You will write a conference-quality research paper as a part of your project
- We will submit the best papers from this class to top conferences/workshops in the area of distributed systems
  - Past versions of CS525 highly successful in getting papers into conferences and journals (see course website)
- To help you get insight into the current and bleeding edge of d.s. research, we will read 2 research papers per class

# Research vs. Users

- Initial direction =/= Final outcome
  - Apple I and II (Wozniak and Jobs): Research challenge was to minimize cost of chips in the PC. Users loved Apple II and III because it had color and it had flexibility for users to write their own software (until then, every new game was done in hardware!)
  - Flickr (Caterina Fake): Initially were writing "Game Nevernding". Research challenges included scalability. Users loved it because of social network, and tagging. Tagging enabled groups (Squared Circle group), news feeds, and find photos of anything.
  - TiVo (Mike Ramsay): Initially were writing a network server for video content. Research challenges included disk management, n/w management, security. Users were amazed by pausing live TV and being given significant flexibility but without needing to be a "techie".
  - Mosaic (Andreesen) was originally an NSF-funded project at UIUC, and then became a startup
- Be ready to change direction
- Both in research project and entrepreneurial project

# Materials for Course

- All readings available on the course website (you don't need to buy anything)

# Project Buildup

- To ensure semester-wide progress, project is structured into systematic stages:
  - Initial meeting in Feb (+ open office hours)
  - Survey report due late-Feb (proposal + survey)
  - Midterm report due Mar-end (first prototype of system built + initial experimental results)
  - Final report due early May (final version of project and paper)
- Project groups: 2-3 students

# Let's Look at the Course Information Sheet…

- No exams
- Paper Reading
  - Presentations (groups of 2)
  - Reviews (2 papers per lecture, on and after Feb 12th)
  - See instructions on website for presentations and reviews
  - Preferable to do a presentation: you can skip 6 review sessions
  - Without presentation, you have to review all sessions from Feb 12th
- Project
  - You can get access to Google cloud (limited, so use wisely!)
  - Limited access to multiple testbeds: PlanetLab, Emulab, and Cloud Computing Tesbed (CCT)
    - Ask us!
- Class Participation a must (and fun!)
- TA: Mainak Ghosh (mghosh4)
- My office hours: right after lecture/class (3112 SC), until about 5.30 pm
- Please read instructions on course website – you're responsible for following them!

# Things for you to do today

- Look at the course website
- Take the survey on website (10 minutes): https://illinois.edu/fb/sec/2316829
- Follow "Schedule / Papers and Presentations link" and read instructions
  - http://courses.engr.illinois.edu/cs525/
  - Need to sign up for a presentation slot by Jan 29
- Take a look at conference papers arising out of previous versions of this course (CS598IG/CS525)
  - Many CS525 project papers published in conferences and journals

# Prerequisites

- Background in OS'es is required (CS241/ CS423)
- Distributed Systems/Algorithms is recommended
  - If you haven't taken CS425/ECE428 or ECE526, then …
  - … you should highly consider taking the Coursera course on Cloud Computing Concepts (Parts 1 and 2). It's a free course.
  - Starting Feb 2nd: register here: https:// www.coursera.org/course/cloudcomputing

# Next Lecture

- Cloud Computing
  - Take a look at all papers on website for that session
  - Read at least one of those papers completely
  - Try to read all of them completely
  - (no reviews required yet)

# Backup Slides

# Analysis (contd.)

$$\beta = \frac{b}{n} \qquad \text{(why?)}$$

Substituting, at time $t = c\log(n)$

$$y = \frac{n+1}{1+ne^{-\frac{b}{n}(n+1)c\log(n)}} \approx \frac{n+1}{1+\frac{1}{n^{cb-1}}}$$

$$\approx (n+1)\left(1-\frac{1}{n^{cb-1}}\right)$$

$$\approx (n+1)-\frac{1}{n^{cb-2}}$$