

# Attack-Resistant Peer-To-Peer Networks

Jared Saia

University of New Mexico

Joint work with: Amos Fiat, Anna Karlin, Steve Gribble and  
Stefan Saroiu

# What is Peer-to-Peer(P2P)?

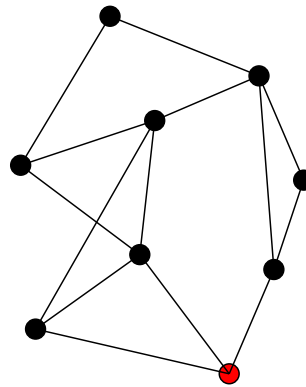
Distributed network where each machine acts as both a server and a client.

Example applications:

- Data-sharing (e.g. Napster, Gnutella, Kazaa and Morpheus)
- Computation (e.g. SETI@home, FOLDING@home DataSynapse, NetBatch)
- Distributed storage (e.g. FARSITE)
- Internet infrastructure systems (e.g. Internet Indirection Infrastructure)
- Collaboration (e.g. Groove Networks)

## Gnutella: A Typical P2P System

*Overlay network*: Link from peer  $x$  to peer  $y$  in overlay network means  $x$  knows the IP-address of  $y$



Gnutella Protocol:

- A new peer decides on its own which other peers to link to in the overlay network and what content to store
- Search requests are broadcast to **all** peers within some fixed number of hops in the overlay network

# P2P Design

Focus on providing efficient access to content across network so:

- Topology of overlay matters
- Where content is stored matters
- Search protocol matters

Gnutella's design decisions give

- Poor Performance - due to search broadcasts
- Poor Reliability - due to ad hoc topology and content storage

Design decisions have big implications for performance *and* reliability.

# Our Research

## Current P2P

- Many deployed systems have poor performance (efficiency and scalability)
- No systems have both good performance and attack-resistance

## Our Research: P2P systems which have

- Attack-Resistance
- Good Performance

# Why Attack-Resistance?

Content is vulnerable to

- Attack by malicious agents
- Censorship by states and corporations
- Loss due to system faults

Key Point:

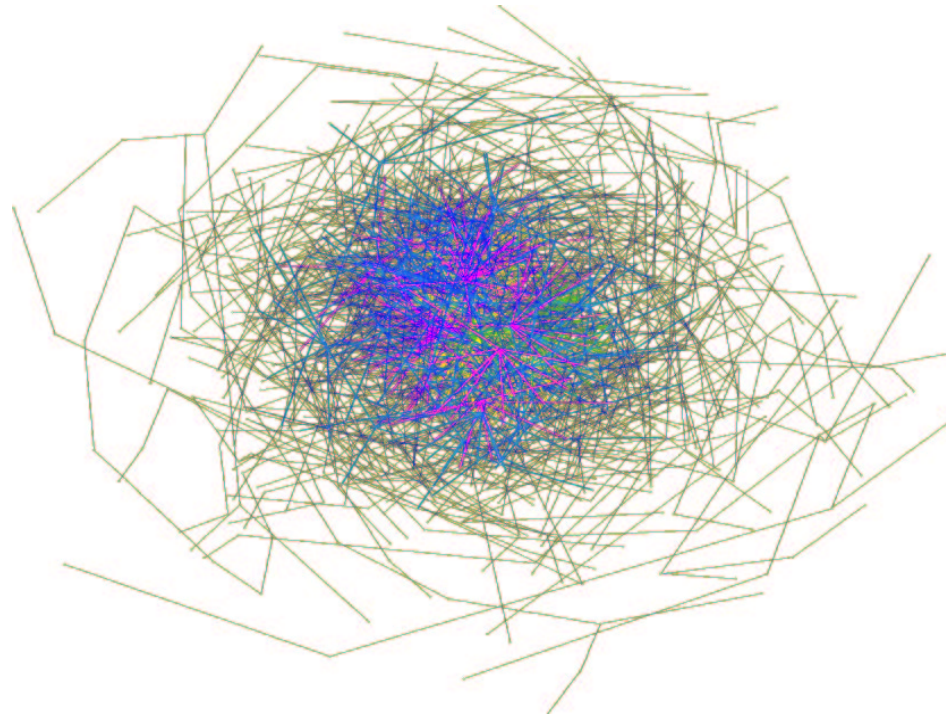
- Single peer has limited technical and legal resources

## Current P2P is **not** Attack-Resistant

Examples:

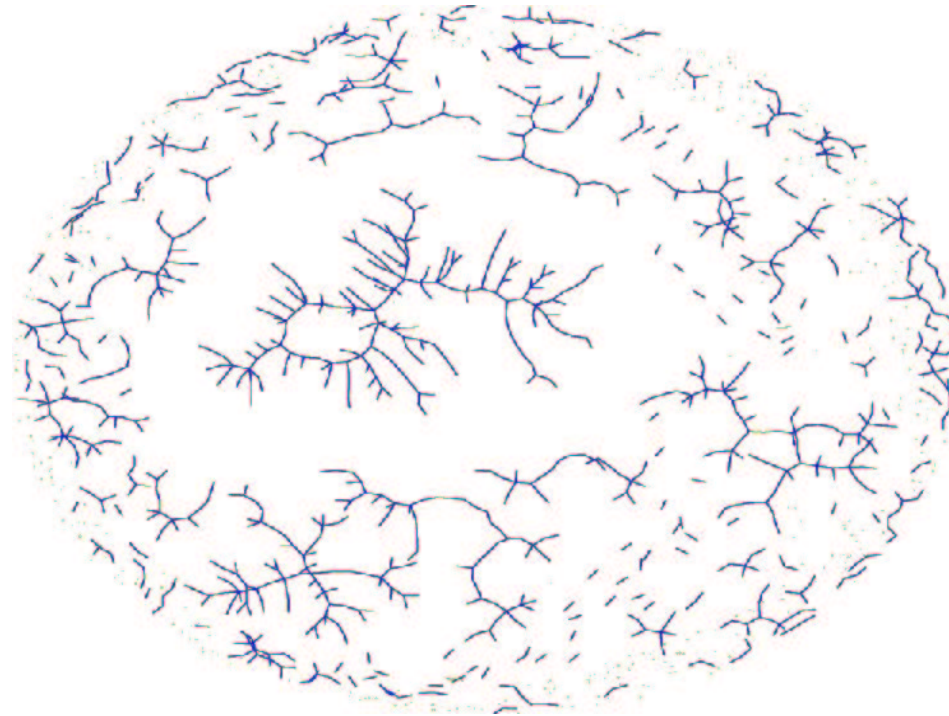
- Napster shut down by attacks on central server
- Flatplanet has launched successful spam attack on Gnutella
- Removal of a few peers will shatter Gnutella[SGG, 2002]

# Gnutella is Vulnerable



Snapshot of Gnutella February, 2001 (1800 peers)[SGG]

# Gnutella is Vulnerable



Same Network (1800 peers) after deleting 63 peers[SGG]

# Can we do better?

Reasons for hope:

- P2p networks have many, many peers
- Lots of flexibility in design of network

# Our Contributions

- First P2P networks provably robust to massive targeted attack by
  - *Fail-stop faults* - Deletion Resistant Network (DRN)
  - *Byzantine faults* - Control Resistant Network (CRN)
- Both networks are scalable and efficient in terms of time and space

## Attack-Resistant Property (DRN)

After deletion of *any*  $2/3$  fraction of the peers,  
99% of the remaining peers can access 99% of the data items.

# Performance

This would be simple if we didn't care about performance!

Naive System:

- Topology is fully connected
- Data Items are stored everywhere
- Insertion by broadcast to all nodes

# Performance

This would be simple if we didn't care about performance!

Naive System:

- Topology is fully connected
- Data Items are stored everywhere
- Insertion by broadcast to all nodes

Our System:

- Pick a robust topology with small maximum degree
- Pick a data replication strategy that balances availability with storage overhead
- Pick a routing protocol that is efficient but redundant enough

## Example Result (DRN)

After deletion of *any*  $2/3$  fraction of the peers, 99% of the remaining peers can access 99% of the data items.

Resource Bounds ( $n$  peers,  $O(n)$  data items):

- $O(\log n)$  storage per peer
- Search takes  $O(\log n)$  time
- Search takes  $O(\log^2 n)$  messages

## Related Work

Network	Storage Per Peer	Search Time	Search Messages	Deletion Resistant?	Control Resistant?
CRN[FS]	$O(\log n)$	$O(\log n)$	$O(\log^2 n)$	Yes	Yes
DRN[FS]	$O(\log n)$	$O(\log n)$	$O(\log^2 n)$	Yes	No
Chord[SMK+]	$O(\log n)$	$O(\log n)$	$O(\log n)$	No	No
CAN[RFH+]	$O(\log n)$	$O(\log n)$	$O(\log n)$	No	No
Tapestry[KBC+]	$O(\log n)$	$O(\log n)$	$O(\log n)$	No	No

Deletion Resistant? = Resistant to Deletion Attacks?

Control Resistant? = Resistant to Control Attacks?

# Outline

- Description of DRN
- Theorems for DRN and CRN and overview of proofs
- Conclusion and Future Work

# DRN Description

Best Resource Bounds:

- $O(\log n)$  search time
- $O(\log^2 n)$  messages per search
- $O(\log n)$  storage per peer

This Talk:

- $O(\log n)$  search time
- $O(\log^3 n)$  messages per search
- $O(\log^2 n)$  storage per peer

# Designing a P2P System

Must Specify:

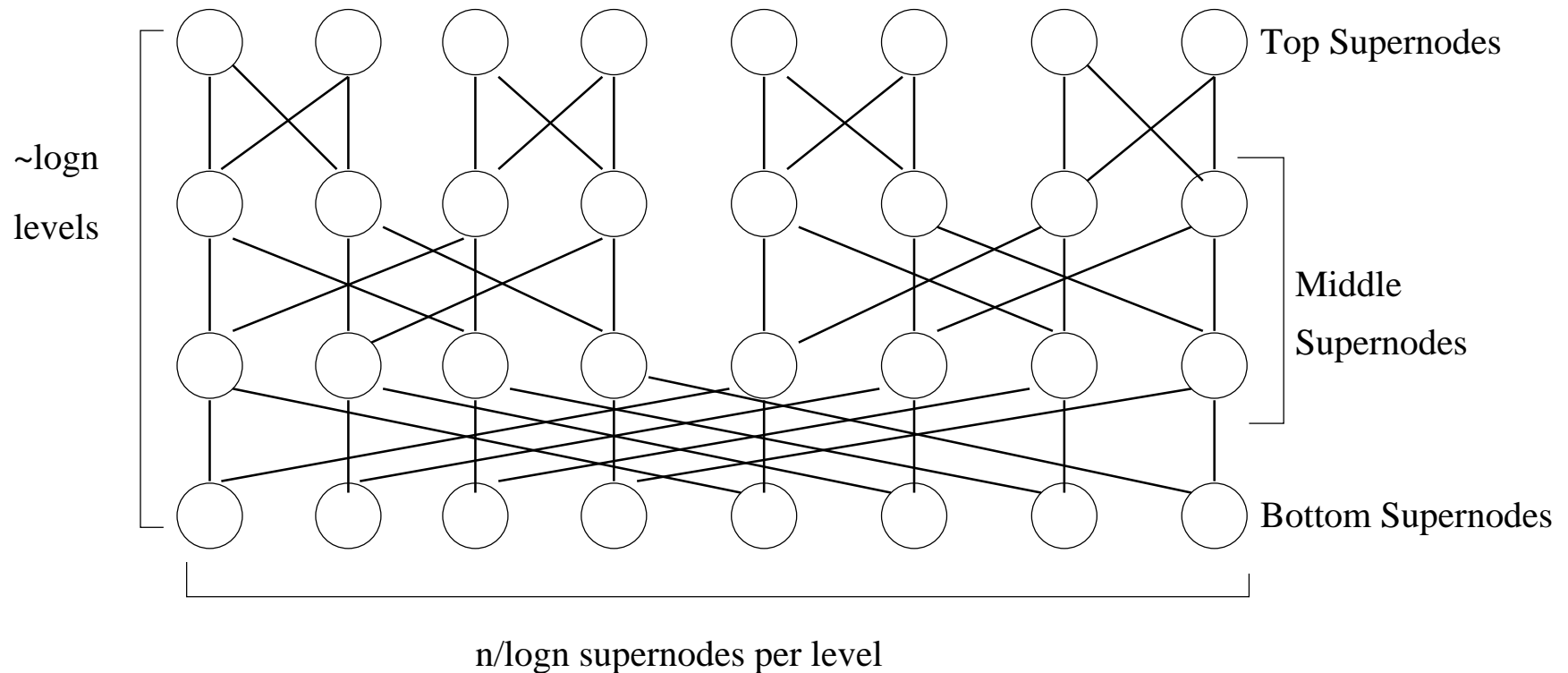
- Overlay topology
- Where to store content
- Protocol for accessing content

# DRN Topology

Topology based on the butterfly network (constant degree version of hypercube)

- Each vertex of butterfly is called a *supernode*
- Each supernode represents a set of peers
- Each peer is in **multiple** supernodes

# DRN Topology

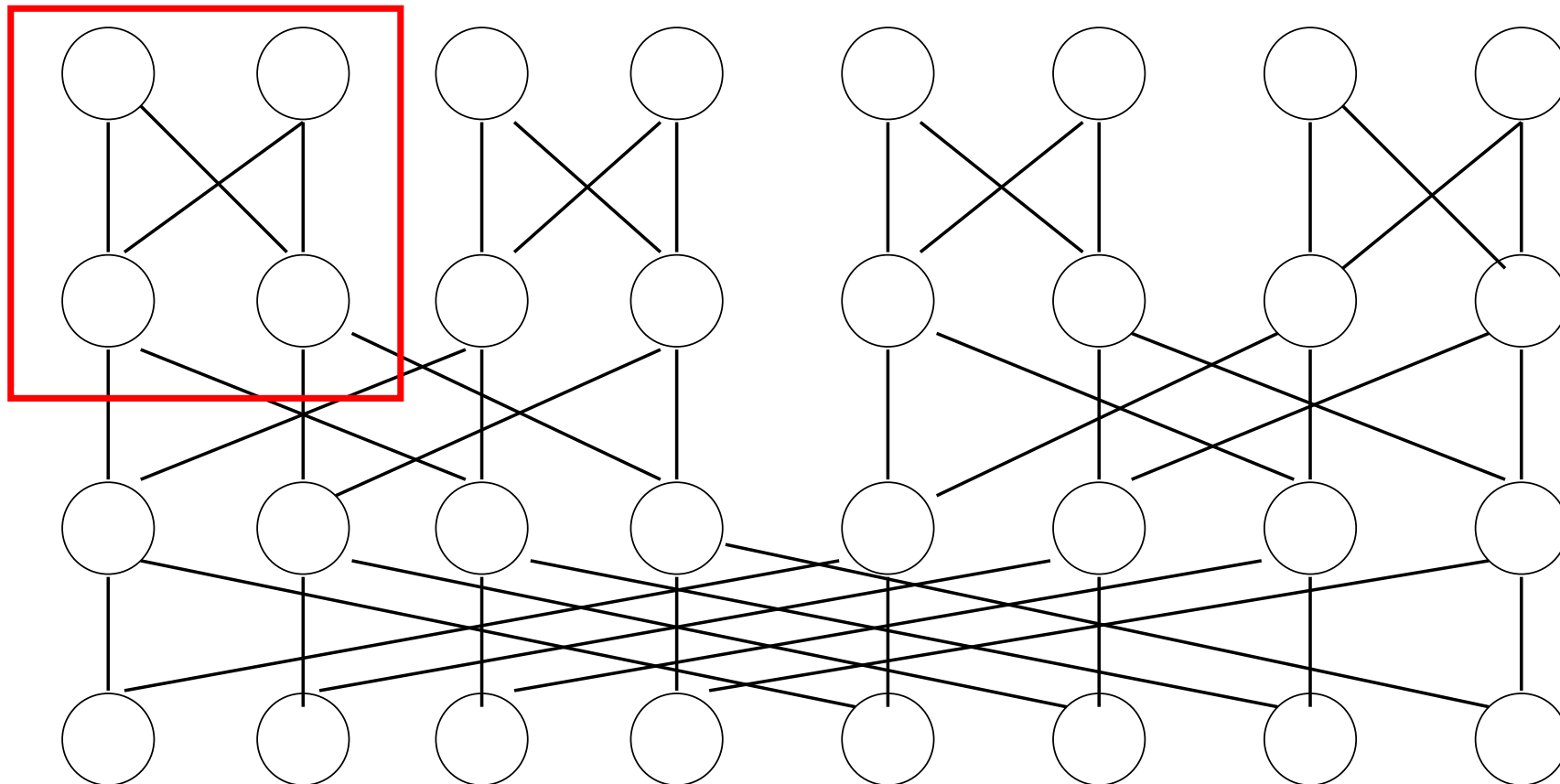


Unique path from every top supernode to every bottom supernode.

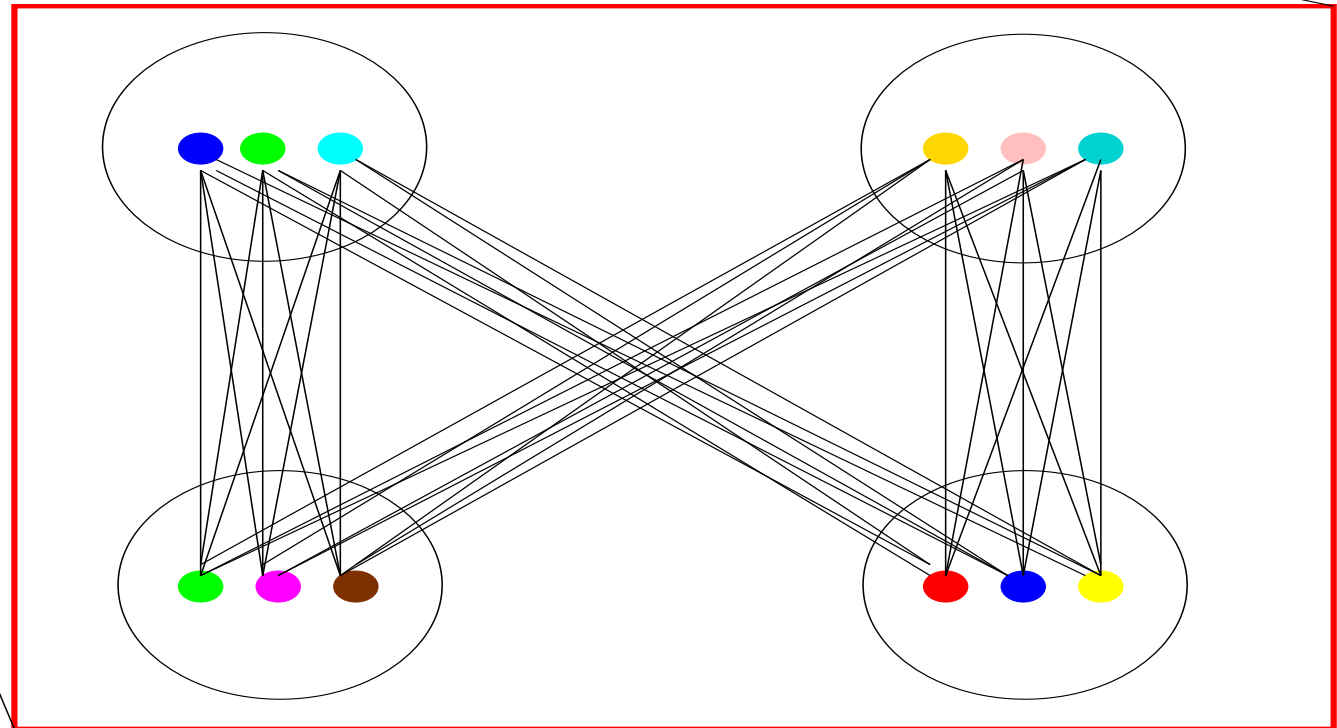
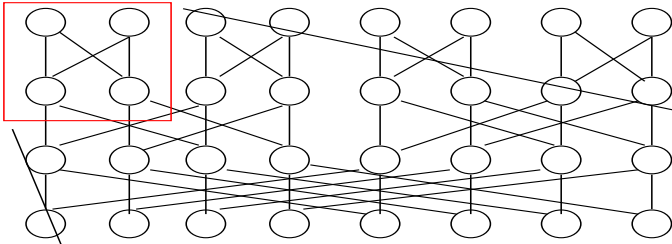
# DRN Topology

- $n$  peers,  $n$  supernodes
- Each peer participates in  $C \log n$  supernodes chosen randomly from set of all supernodes
- Supernode  $X$  connected to supernode  $Y$  in butterfly means **all** peers in  $X$  connected to **all** peers in  $Y$ .

# DRN Topology



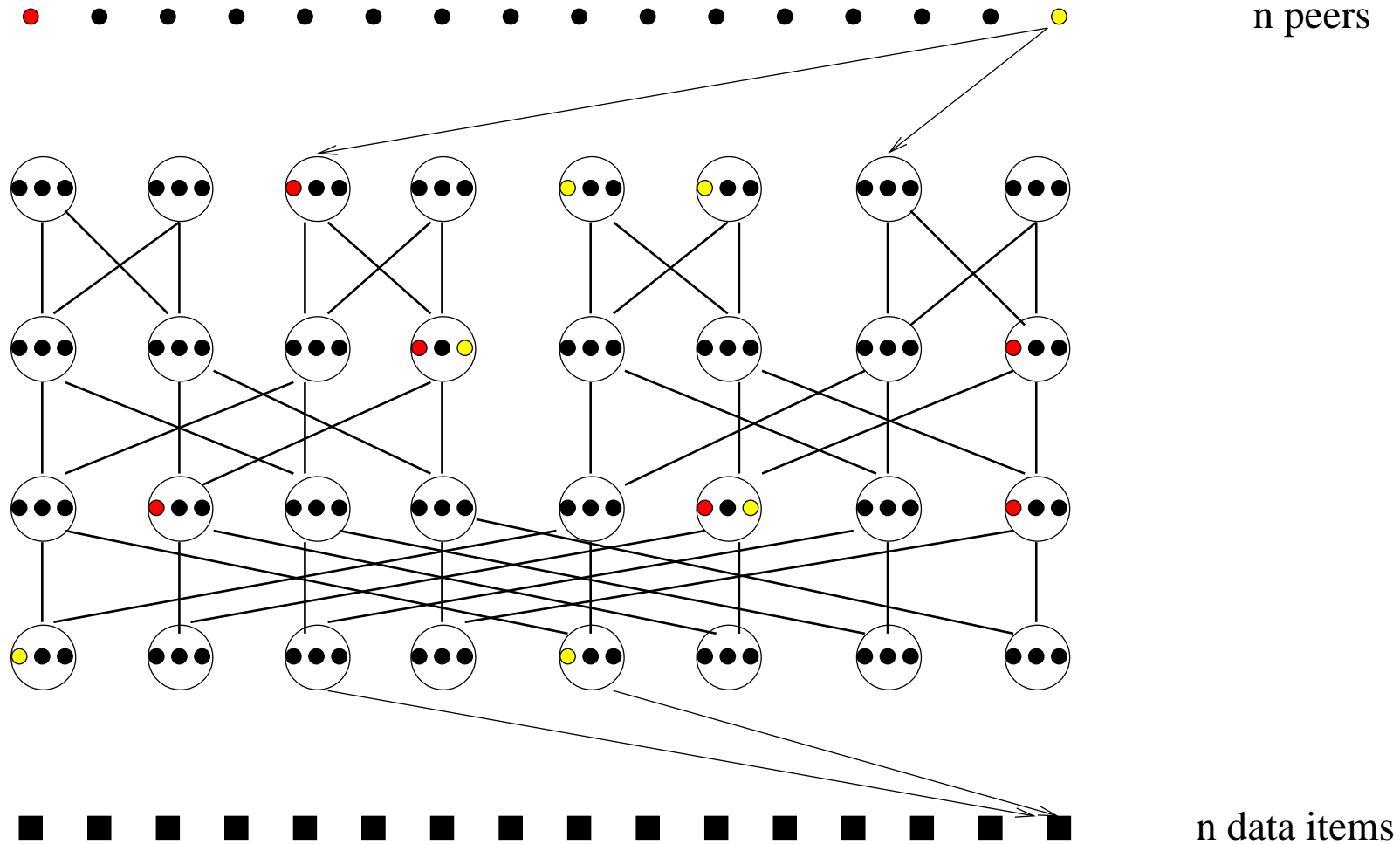
# DRN Topology



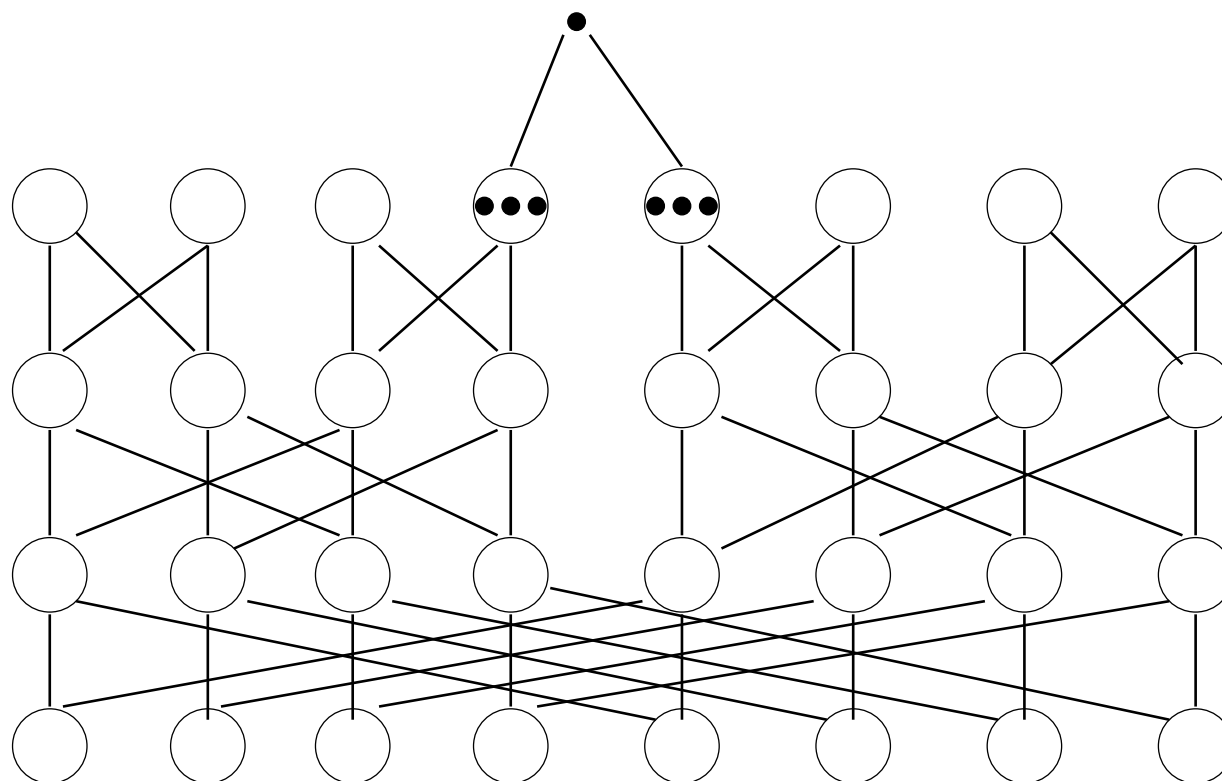
# DRN Topology

- Each peer connected to all peers of  $T$  random top supernodes
- Each data item is stored on all peers in  $B$  random bottom supernodes
- Each peer participates in  $C \log n$  supernodes chosen randomly from set of all supernodes
- $T, B$  and  $C$  depend on fault tolerant parameters

# DRN Topology

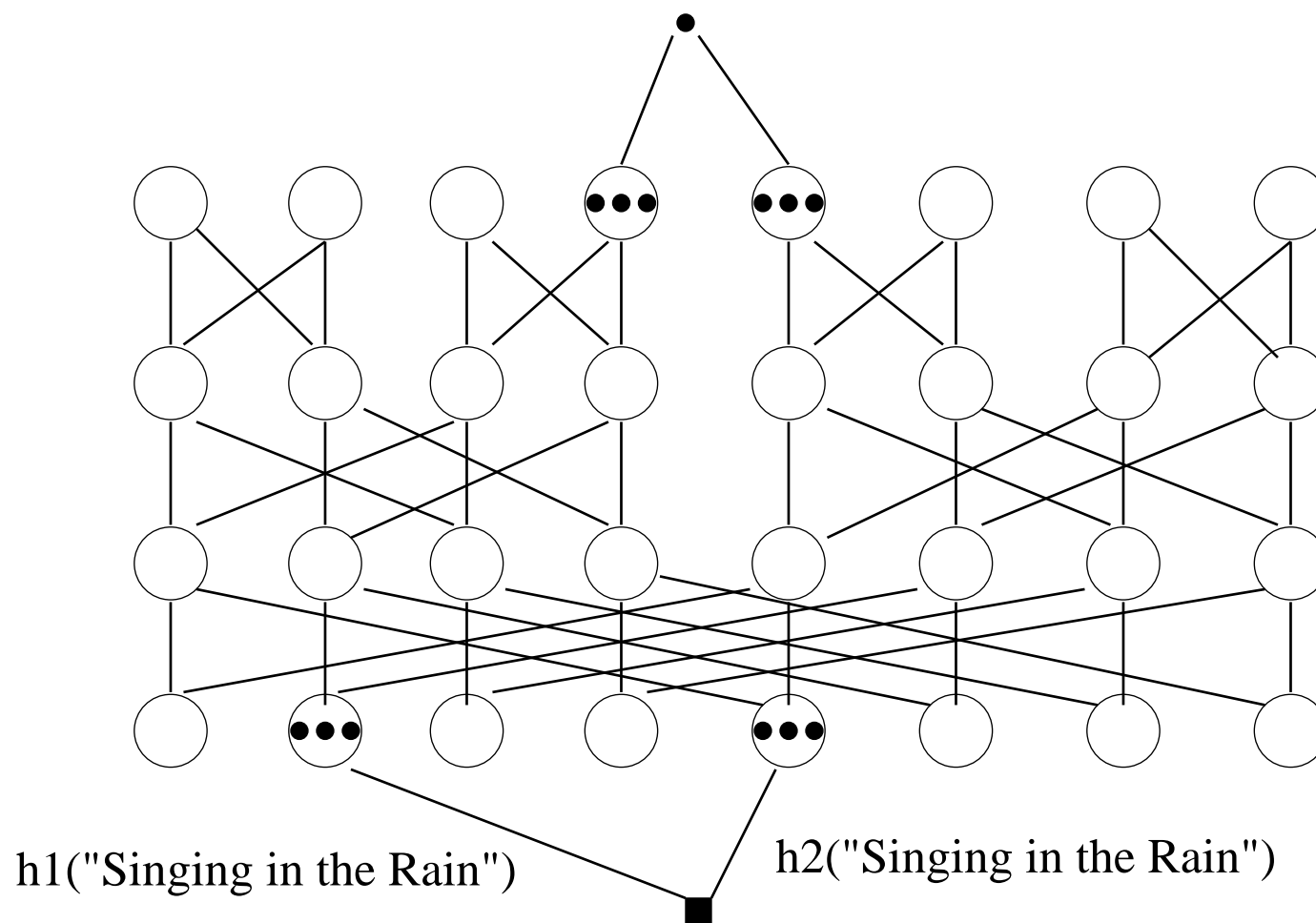


# DRN Searches



■ "Singing in the Rain"

# DRN Searches





## DRN Searches from $v$ for $d$

- Hash “title” to get  $B$  target bottom supernodes
- Send request to peers in all top supernodes  $v$  connects to
- In parallel, for each path between a top supernode  $t$  which  $v$  connects to and a bottom supernode  $b$  where  $d$  is stored do:
  - Send request from  $t$  to  $b$  in the butterfly:
    - \* Each peer sends request down to all peers in the supernode below it
  - return  $d$  from  $b$  to  $t$  along same path
    - \* Each peer passes up content to all peers in the supernode above it

# Outline

- Description of DRN
- Theorems for DRN and CRN and overview of proofs
- Conclusion and Future Work

## Deletion Resistant Network (DRN)

**Theorem 1** For any fixed  $\alpha < 1$ ,  $\epsilon > 0$ , there is a DRN over  $n$  peers accessing  $O(n)$  data items with the property that:

After deletion of *any* set of  $\alpha n$  peers, a  $(1 - \epsilon)$  fraction of the remaining peers can access a  $(1 - \epsilon)$  fraction of the original data items.

Example: After deletion of *any*  $2/3$  fraction of the peers, 99% of the remaining peers can access 99% of the data items.

## Proof Sketch

**Theorem 1** *For any fixed  $\alpha < 1$ ,  $\epsilon > 0$ , there is a DRN network for  $n$  peers, accessing  $O(n)$  data items such that:*

*After deletion of **any** set of  $\alpha n$  peers, a  $(1 - \epsilon)$  fraction of the remaining peers can access a  $(1 - \epsilon)$  fraction of the original data items.*

- Critically rely on random assignment of:
  - peers to supernodes
  - peers to top supernodes
  - data items to bottom supernodes
- Use the Probabilistic Method

# Proof Sketch

*Definitions:*

- A supernode is *good* if it has one live peer.
- A path is *good* if it contains all good supernodes

*Lemma 1:* A good path enables secure communication

# Proof Sketch

*Definitions:*

- A supernode is *good* if less than half its peers are controlled by adversary.
- A path is *good* if it contains all good supernodes

*Lemma 1:* A good path enables secure communication

*Lemma 2:* After adversarial attack, all but  $\epsilon n / \log n$  supernodes are good

# Proof Sketch

*Definitions:*

- A supernode is *good* if less than half its peers are controlled by adversary.
- A path is *good* if it contains all good supernodes

*Lemma 1:* A good path enables secure communication

*Lemma 2:* After adversarial attack, all but  $\epsilon n / \log n$  supernodes are good

*Lemma 3:* All but an  $\epsilon$  fraction of the paths are good

# Proof Sketch

*Definitions:*

- A supernode is *good* if less than half its peers are controlled by adversary.
- A path is *good* if it contains all good supernodes

*Lemma 1:* A good path enables secure communication

*Lemma 2:* After adversarial attack, all but  $\epsilon n / \log n$  supernodes are good

*Lemma 3:* All but an  $\epsilon$  fraction of the paths are good

These Lemmas Imply:

$(1 - \epsilon)$  fraction of remaining peers can access  $(1 - \epsilon)$  fraction of the data items

## Control Resistant Network (CRN)

**Theorem 2** *For any fixed  $\alpha < 1/2$ ,  $\epsilon > 0$ , there is a CRN network for  $n$  peers, accessing  $O(n)$  data items such that:*

*Even if adversary controls any set of  $\alpha n$  peers,  $(1 - \epsilon)$  fraction of the remaining peers can access  $(1 - \epsilon)$  fraction of the true data items.*

*Even if adversary controls 1/3 of the peers, 99% of the peers can access 99% of the true data items.*

*Adversary has complete knowledge of the system. Knows topology of network and where all data items are stored.*

## CRN is “Spam Resistant”

Assume:

- All true data items are stored in the network
- Adversary takes over 1/3 of the peers
- Adversary uses these peers to send “fake” messages instead of what was requested

Then it’s still the case that

- 99% of the remaining peers can access 99% of the true data items

# Overview of CRN

Key change for searches:

- Each peer only passes along a message if it received that message from a majority of its neighboring peers.

Key ideas for proof:

- Call a supernode *good* if a majority of its peers are not faulty
- Call a path good if all its supernodes are good
- *Lemma 1*: A good path enables secure communication
- *Lemma 2*: After adversarial attack, all but  $\epsilon n / \log n$  supernodes are good

# Contributions

- First P2P networks provably robust to targeted attack by
  - Fail-stop faults
  - Byzantine faults
- Time and space resource bounds for both networks are competitive with other networks which are not provably attack-resistant

## Future Work

- Use of these ideas in practical systems
- Scalable and attack-resistant computation
  - SETI@home, FOLDING@home, embedded systems
  - Initial results on scalable Byzantine agreement

# Dynamic Attack-Resistance

## The Problem

- DRN is robust only to a static attack
- If all the original peers are attacked, the network fails, even if many new peers have joined

## What do we want?

- Adversary can delete all the original peers, if enough new peers join
- Must have as many new peers join as are deleted

# Dynamic Attack-Resistance

The Result [Saia, Fiat, Gribble, Karlin and Saroiu]

Assume:

- Always storing  $O(n)$  data items
- Each joining peer knows one random peer in network
- In any fixed time interval, more peers join the network than are deleted

What do we get?

- At any time, 99% of the live peers in the network can access 99% of the content

# Data Insertion

## Data Insertion

- Peer performs search and sends data with the search
- Store data at the bottom supernodes in the search
- This insertion fails with small constant probability

# Peer Insertion

## Peer Insertion

- Peer copies links to top supernodes of some other peer
- Takes  $O(\log n)$  time
- Peer does searches from these top supernodes
- This insertion does not increase resiliency of CAN

## Distributed Creation of CRN

Creation requires  $n$  broadcasts or transmission of  $n^2$  messages.

- Each peer hashes its IP-address to get a set of  $C \log n$  supernodes to which it belongs
- Each peer broadcasts a message containing identifiers of these supernodes
- Each peer receives messages from other peers giving supernodes to which they belong
- If some other peer belongs to a neighboring supernode, a link is formed to that peer

# Dynamically Fault-Tolerant CAN

Assumptions:

- Start with a network on  $n$  peers
- Number of items indexed is fixed
- Each joining peer knows one random “good” peer

Definitions:

- An *adversary is limited* if for some  $\gamma > 0, \delta > \gamma$ , at least  $\delta n$  peers join the network in any time interval when adversary deletes  $\gamma n$  peers.
- A CAN is  $\epsilon$ -*robust* at some particular time if all but an  $\epsilon$  fraction of the peers can access all but an  $\epsilon$  fraction of the content.
- A CAN is  $\epsilon$ -*dynamically fault tolerant* if, WHP, the CAN is always  $\epsilon$ -robust during period when a limited adversary deletes number of peers polynomial in  $n$ .

# Dynamically Fault-Tolerant CAN

Result: For any  $\epsilon > 0$ ,  $\gamma < 1$  and  $\delta > \gamma + \epsilon$ , we give a  $\epsilon$ -dynamically fault-tolerant CAN:

- CAN is  $\epsilon$ -robust assuming  $\delta n$  peers added whenever  $\gamma n$  peers deleted.
- Search takes  $O(\log n)$  time and  $O(\log^3 n)$  messages
- Every peer maintains pointers to  $O(\log^3 n)$  other peers
- Every peer stores  $O(\log n)$  data items
- Peer insertion takes  $O(\log n)$  time

## Related Work - Robustness

- Robust File Systems: Publius[WRC], Alon et al.[AKKMS]
- Quorum systems[MRW,MRWW]. A robust way to read and write to a shared variable.
- For strongly robust versions of these systems, search takes  $\Omega(n)$  time.

# The Constants

- Search takes  $\log n$  hops
- Tradeoffs for other constants (e.g. choosing higher constant for storage gives lower constant for messages sent)
- “Typical” values for these constants (i.e. number of messages and storage) are currently in the 100’s.

Reducing the constants (for number of messages sent and storage)

- Proof of Concept
- Currently Have Large Constants to Make Proof Easier
- Decrease in Constants for Expander Graphs Will Decrease our Constants
- In Practice, May Still Get Very Strong Robustness Even With Smaller Constants than Are Required By our Proofs.

## Improving the Bounds

For Theoreticians Only

- $O(\log n)$  pointers per peer
- $O(\log n)$  time per search
- $O(\log^2 n)$  messages per search

To get these bounds, connect supernodes with expander graphs rather than complete graphs.

