Peer-to-Peer Systems and Security IN2194

Freenet

Freenet - Overview

- Freenet Design Goals
  - General Freenet information (all versions)
  - Freenet 0.5 specifics
- Freenet “Darknet” (0.7, 0.7.5)
  - Rationale
  - Routing Algorithm
  - Security Improvements
  - Structuring the Network
- Freenet Attack
  - Idea
  - Implementation
  - Results

Freenet Design Goals

- Distributed data store
- Privacy
  - Disseminators
  - Consumers
  - Holders
- Censorship resistance
- Availability and reliability
- Scalable, efficient
- Attack resistance

Freenet General Overview

- P2P Network
  - System made up of volunteers
  - Peers offer resources in return for services
- Cross platform
  - Java based, runs on anything with a Java VM
  - Peers communicate over UDP (> 0.7)
- Enables users to share data privately
- Over 10 years old
- Over 2 million downloads

Freenet Applications

- Freesites
  - Internal Freenet websites
  - Freenet equivalent of WWW
  - FProxy – freesite browser
  - jSite - Freesite creator
- Frost
  - Message board/chat system
  - Feature rich, used for file sharing
- Thaw
  - Convenient access to Freenet FS API
  - GUI filesharing upload/download/search
- Freemail
  - Email between Freenet users
  - Uses normal email client

→ All applications are usable ONLY on Freenet network

Freenet

- Key based storage and routing
  - Peers and data identified by GUID keys
  - DHT api: insert, retrieve, update
- Unstructured network (Freenet 0.5)
  - No *default* organization among nodes
  - Routing essentially random
  - Nodes have static connections
- Storage
  - LRU eviction policy
  - Popular data stays around

Freenet Data Storage/Retrieval

- Data identified by GUID
- GUID's are hashes of
  - CHK – Content-hash Key
    - SHA-1 Hash of actual file to be stored
    - Low level identifier for static block
  - SSK – Signed-subspace Key
    - $H(H(K_{pub}) + H(S))$ signed by K_{priv}
    - H = Hashing function
    - K_{pub} = public key
    - K_{priv} = private key
- CHK
  - Allows files/file parts to be located
  - Cannot be updated
- SSK
  - Typical used for indexing of CHK's
  - Create arbitrary trees of data (for large files)

Freenet 0.7

- Totally rewritten version of Freenet
- Focus is on privacy AND efficiency
- Main version in use today
- Data (storage identification) and applications the same
- Topology and routing new

Freenet 0.7 - Basics

- Overlay based on cyclic address space of size 2^{32}
- Nodes have a constant set of connections (F2F)
- All data identified by key (modulo 2^{32})
- Data assumed to be stored at closest node
- Routing uses depth first traversal in order of proximity to key
- Friend-to-friend (F2F) networks (``darknets'')
  - Makes Freenet a “restricted route” network
  - Applications in other domains

Freenet – Small World

- Small world network assumption
  - F2F “darknet” should be similar to social networks
  - Provided network “friends” are real world friends
- Sparsely connected graph
  - There exists a short path (O(log N)) between any pair of nodes
  - Common real world phenomenon (Milgram, Watts & Strogatz)
  - PGP web of trust, actor/movie connections
- Freenet's routing algorithm attempts to find short paths
  - Uses locations of nodes to determine proximity to target
  - Uses swapping of locations to structure topology

Freenet – Location Swapping

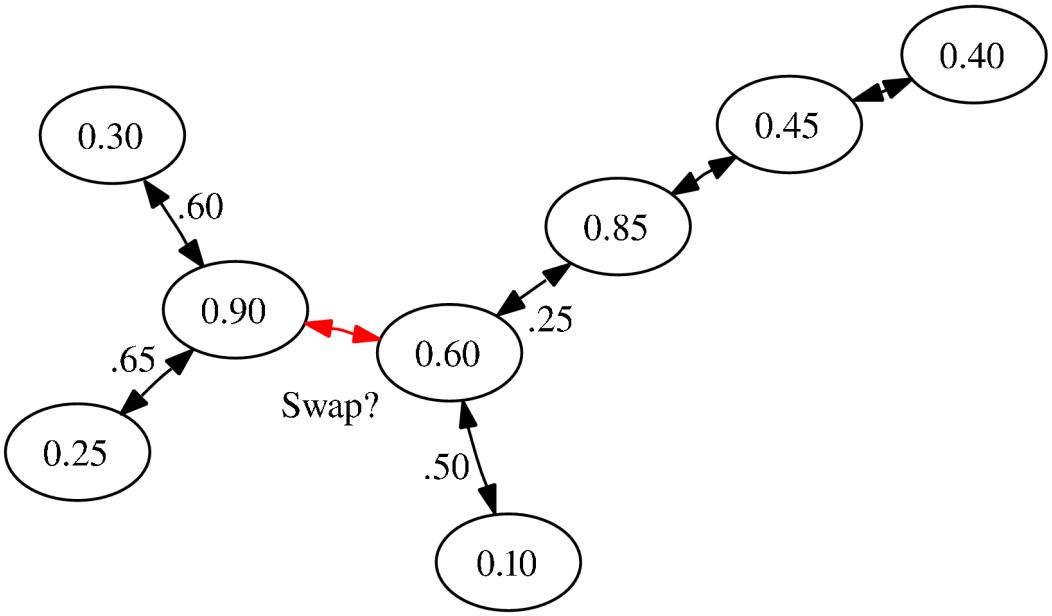
- **Location Swapping**
  - **Nodes swap locations to improve routing performance**
  - **Each connected pair of nodes (a,b) computes:**
-

$$P_{a,b} := \frac{\prod_{(a,o)\in E} |L_a - L_o| \cdot \prod_{(b,p)\in E} |L_b - L_p|}{\prod_{(a,o)\in E} |L_b - L_o| \cdot \prod_{(b,p)\in E} |L_a - L_p|}$$

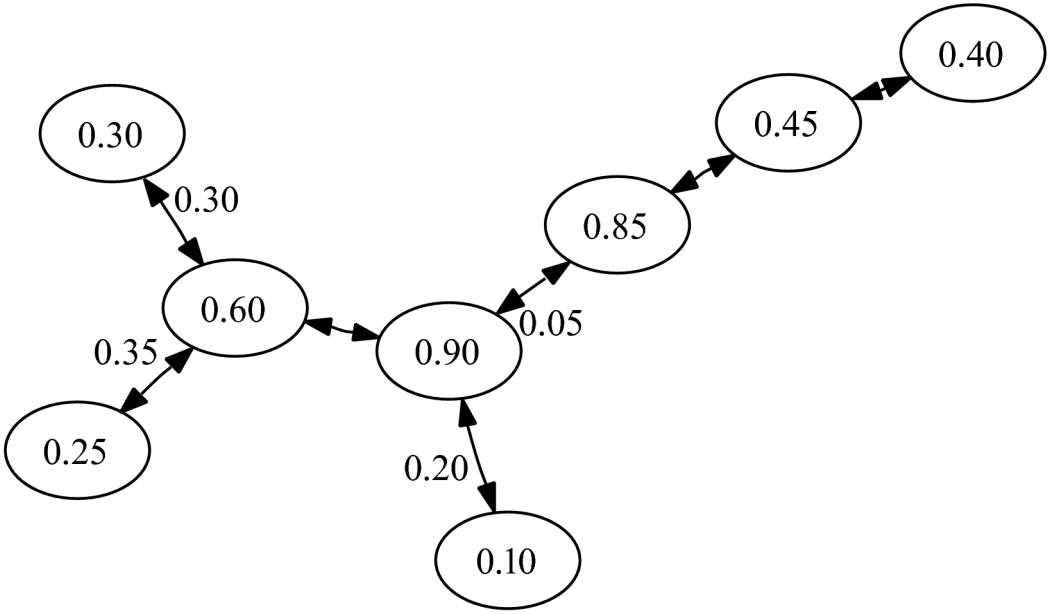
If $P_{a,b} \geq 1$ the nodes swap locations

Otherwise they swap with probability $P_{a,b}$

Freenet – Swap Example



Freenet – Swap Example



Freenet - Routing of GET Requests

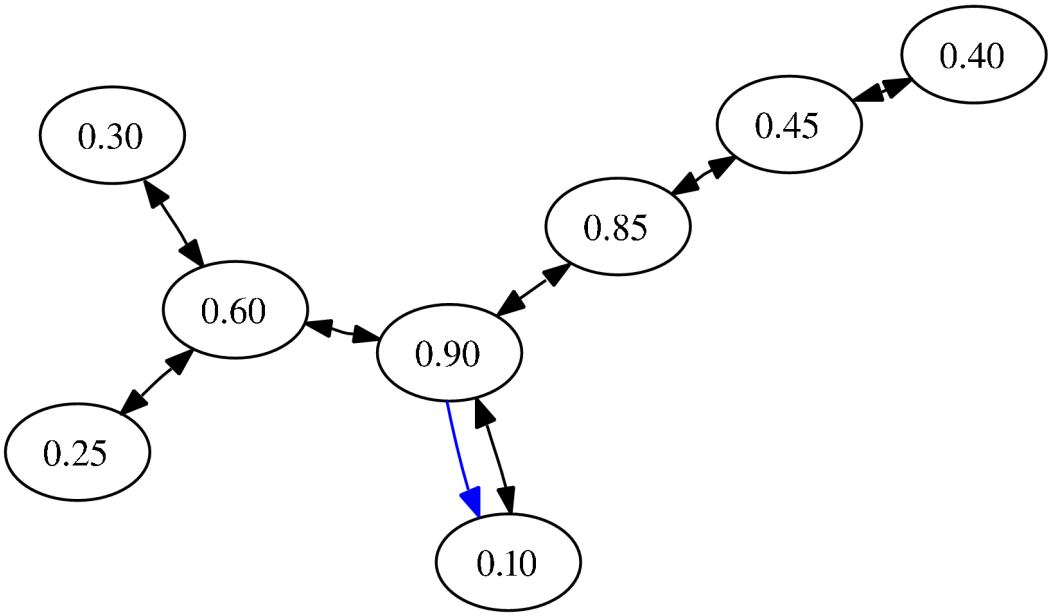
- GET requests are routed based on peer locations and key:
  - Client initiates GET request
  - Request routed to neighbor with closest location to key
  - If data not found, request is forwarded to neighbors in order of proximity to the key

- Forwarding stops when data found, hops-to-live reaches zero or identical request was recently forwarded (to avoid circular routing)

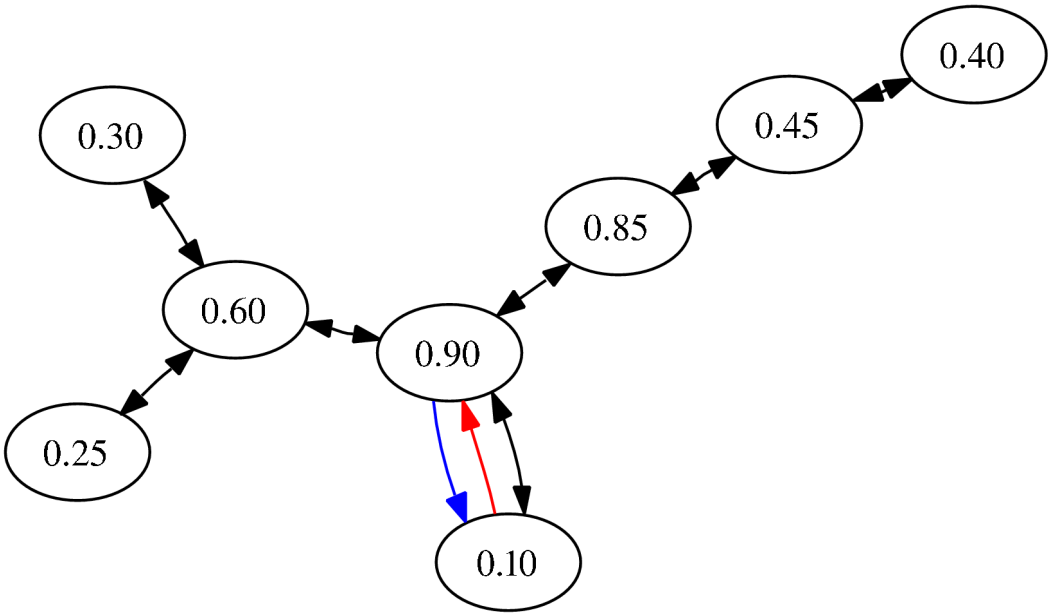
→ Depth-first routing in order of proximity to key.

Freenet – GET Request (1/6)

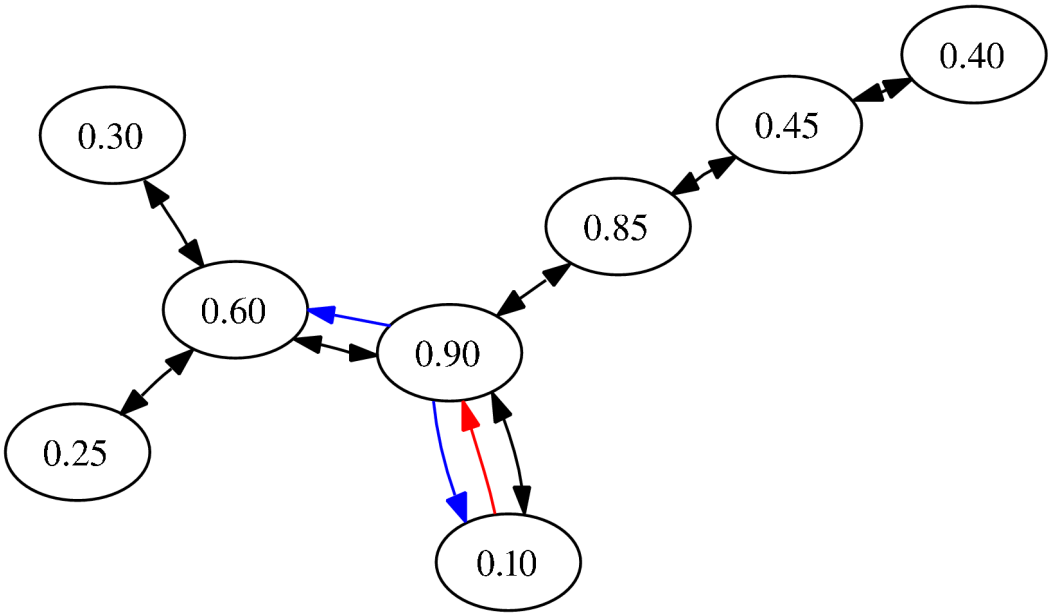
Node .90 searches for data with key .2 stored at peer .25



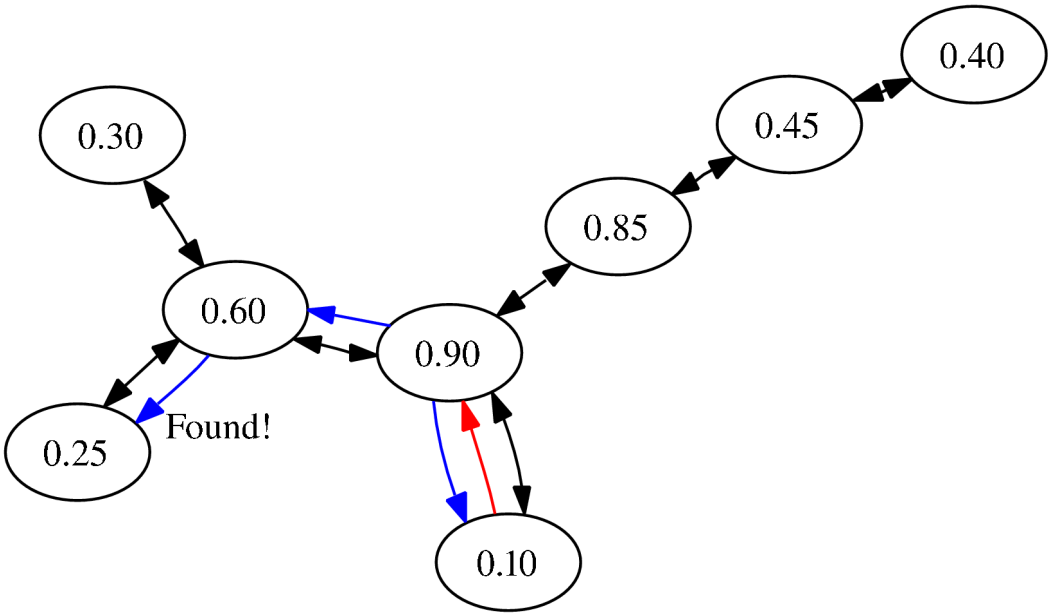
Freenet – GET Request (2/6)



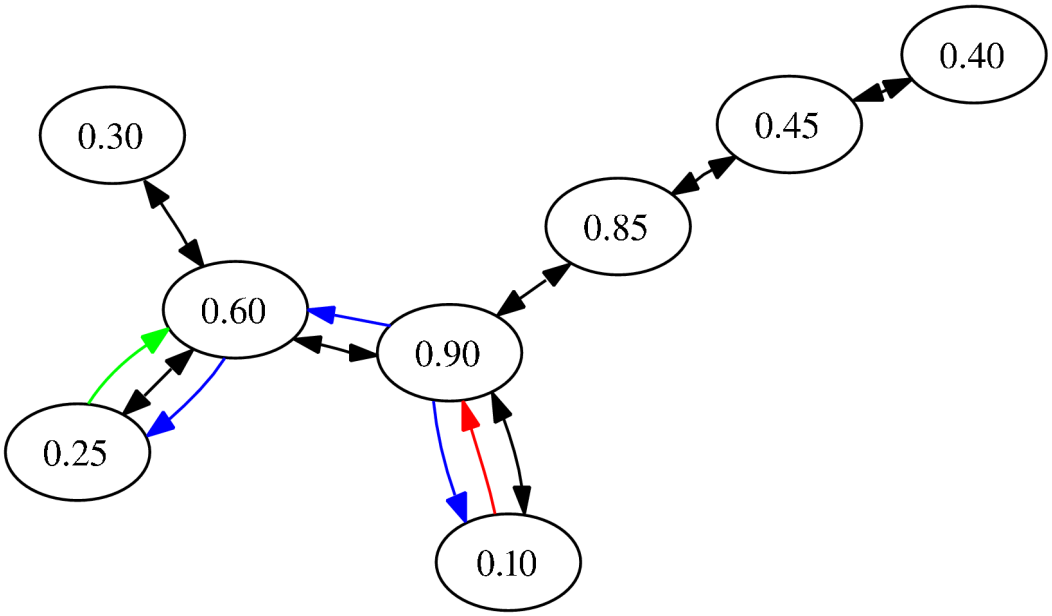
Freenet – GET Request (3/6)



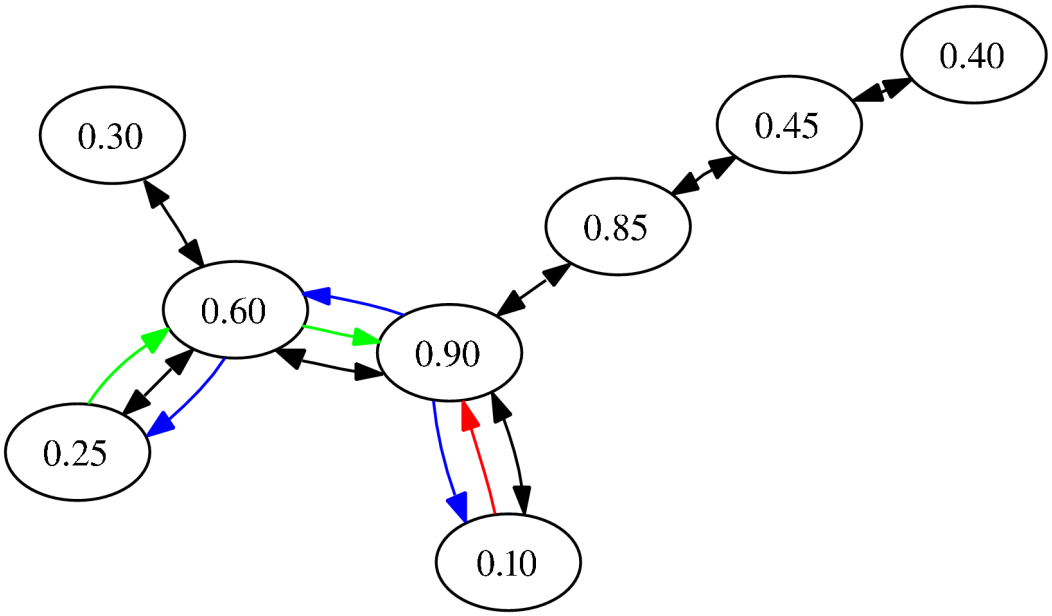
Freenet – GET Request (4/6)



Freenet – GET Request (5/6)



Freenet – GET Request (6/6)

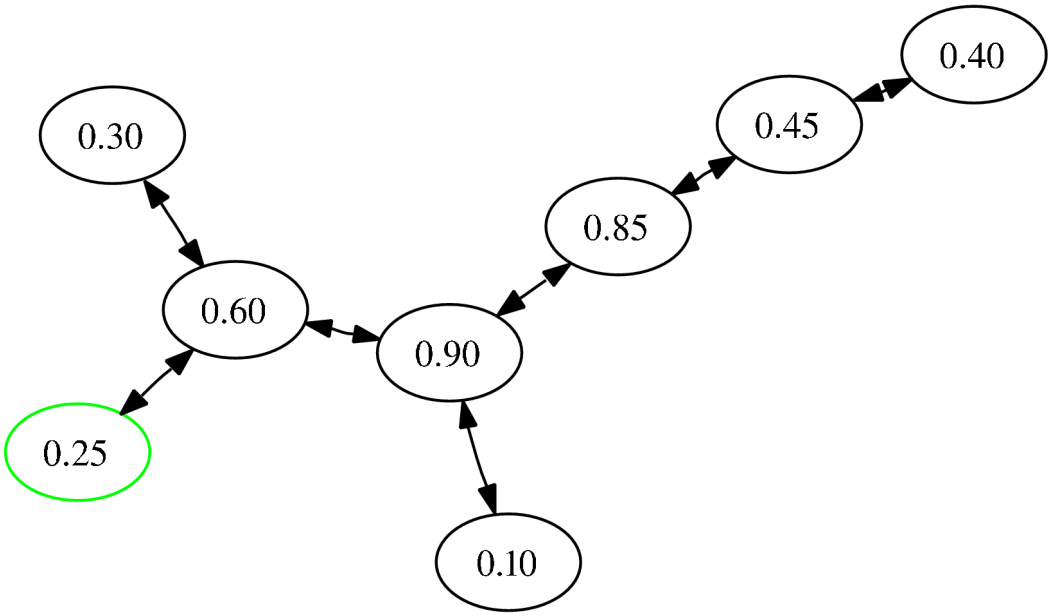


Freenet PUT Request

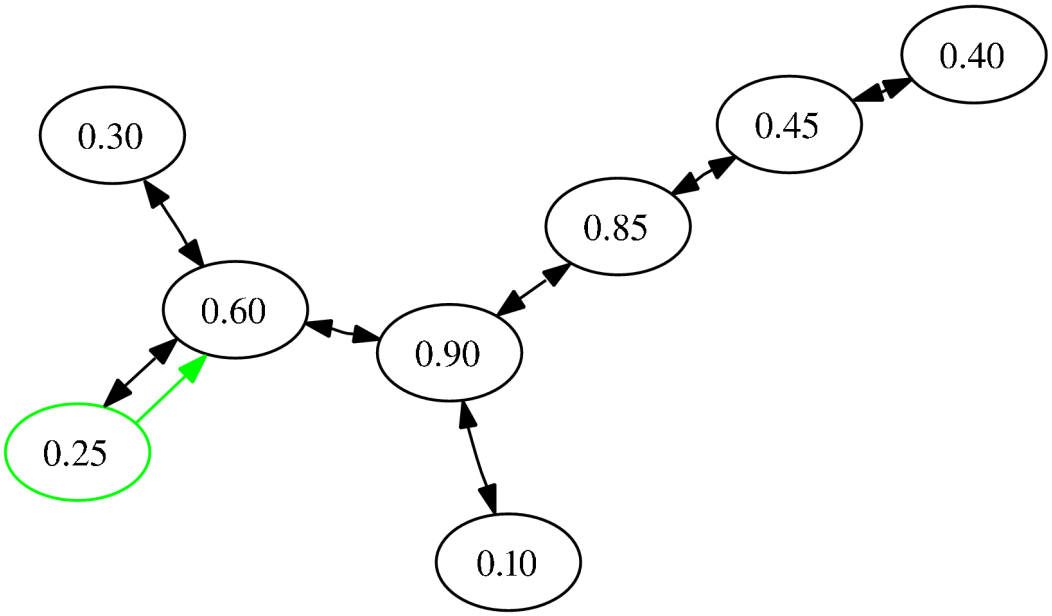
- PUT requests are routed the same as GET requests:
  - Client initiates PUT requests
  - Request routed to neighbor closest to the key
  - If receiver has any peer whose location is closer to the key, request is forwarded
  - If not, the node resets the hops-to-live to the maximum and sends the put request to all of its' neighbors
  - Routing continues until hops-to-live reaches zero (or node has seen request already)
  - Once item is inserted at a node, it resends the request out to all known peers (replication)

Freenet – PUT Request (1/4)

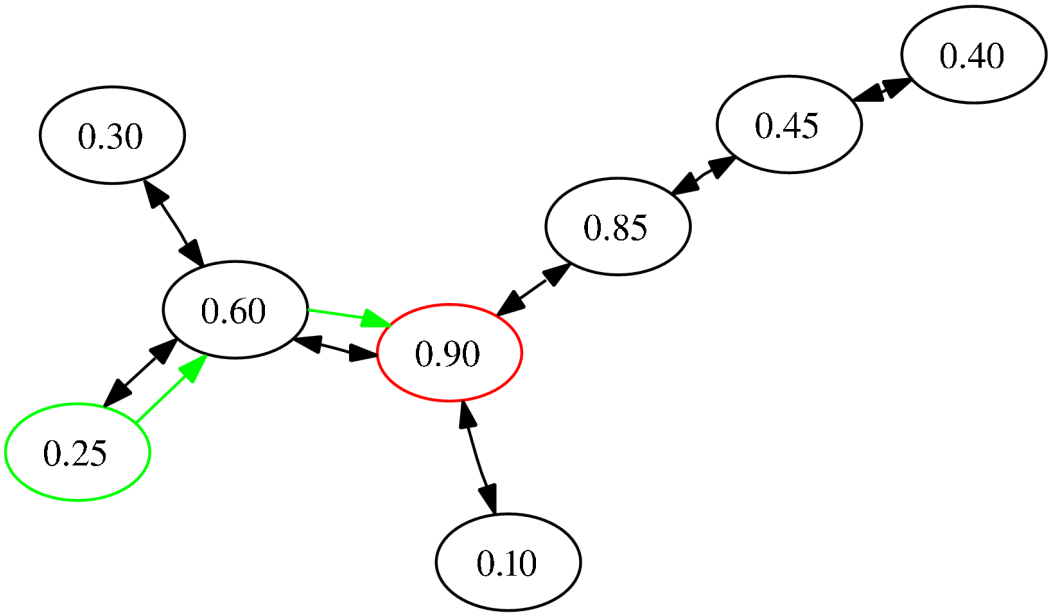
Node .25 inserting data identified by key .93



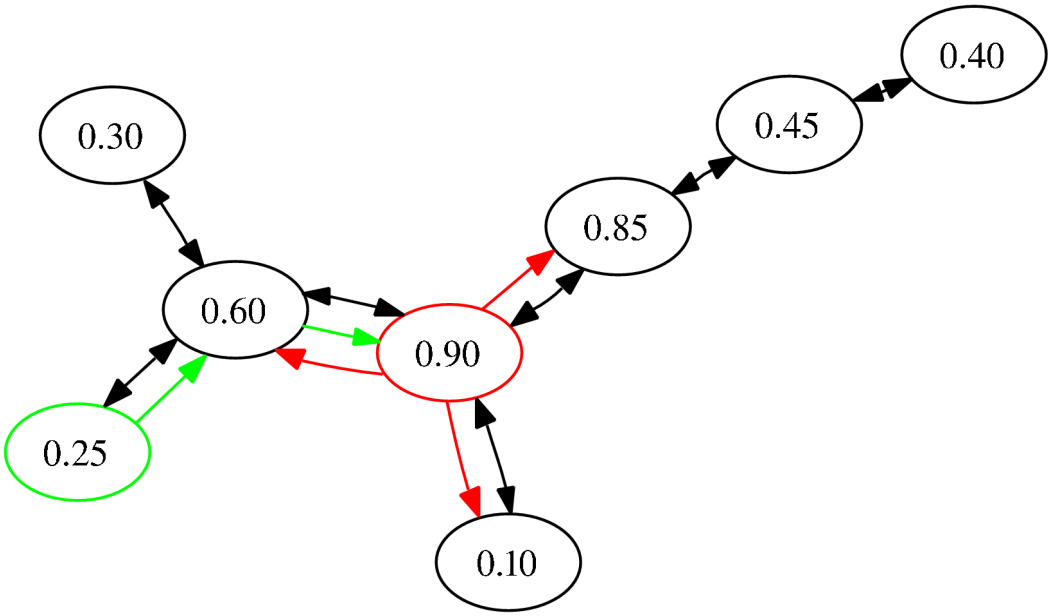
Freenet – PUT Request (2/4)



Freenet – PUT Request (3/4)



Freenet – PUT Request (4/4)



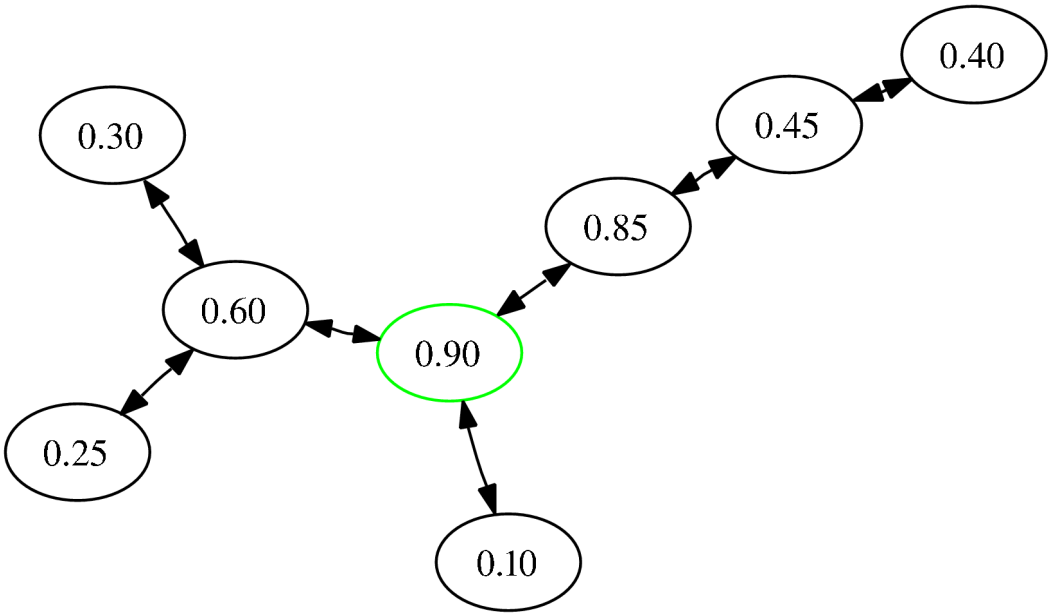
Freenet – Attack Idea

- Freenet relies on a balanced distribution of node locations for data storage
- Reducing the spread of locations causes imbalance in storage responsibilities
- Peers cannot verify locations in swap protocol, including location(s) they may receive
- Use swap protocol to reduce spread of locations!

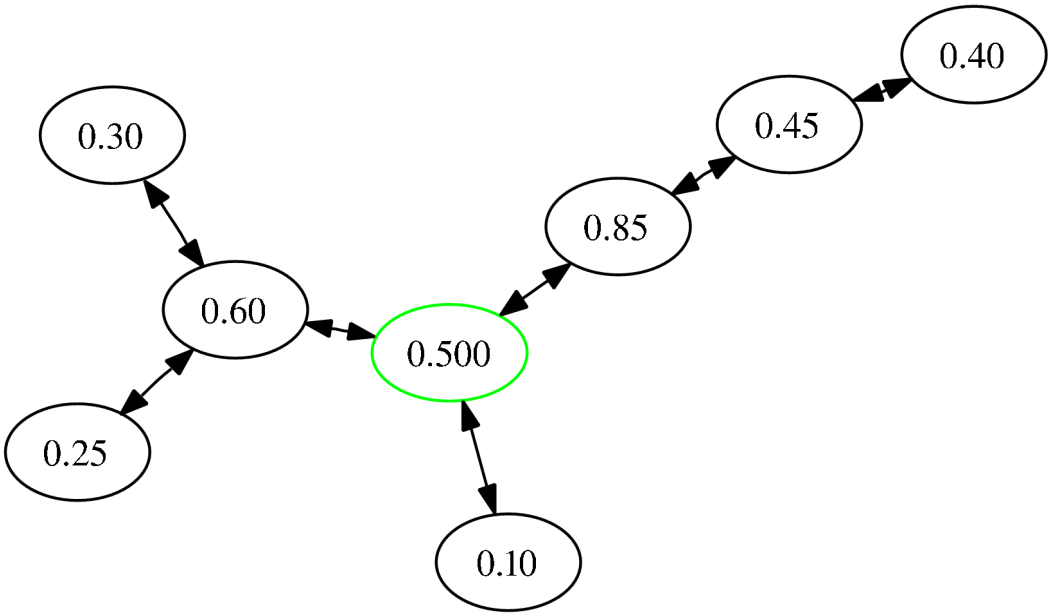
Freenet – Attack Details

- Initialize malicious nodes with a specific location
- If a node swaps with the malicious node, the malicious node resets to the initial location (or one very close to it)
- This removes the ``good'' node location and replaces it with one of the malicious nodes choosing
- Each time any node swaps with the malicious node, another location is removed and replaced with a ``bad'' location
- Bad location(s) spread to other nodes through normal swapping behavior
- Over time, the attacker creates large clusters of nodes around a few locations

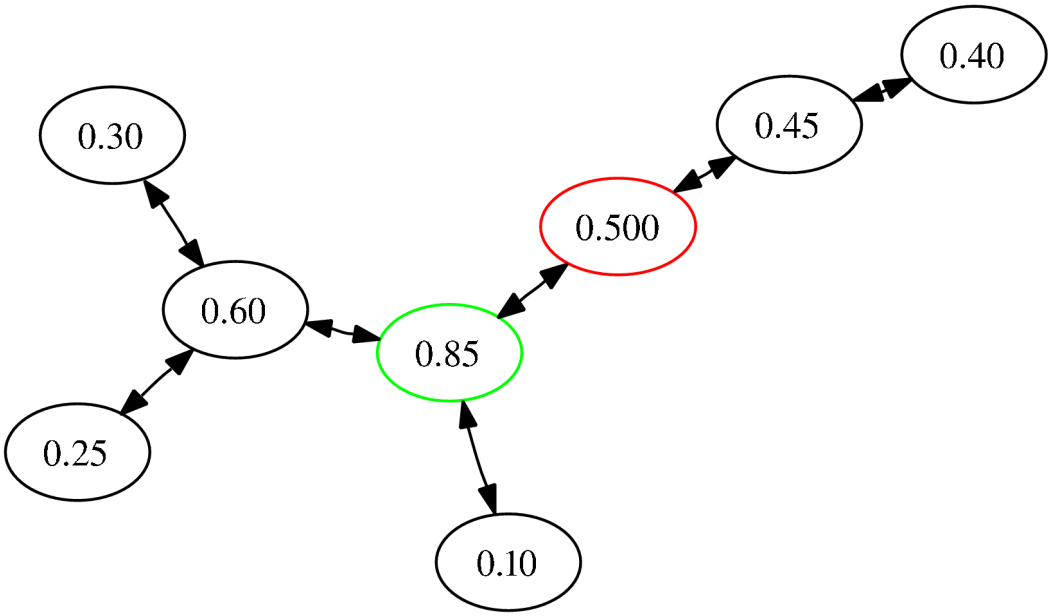
Freenet – Attack Example (1/11)



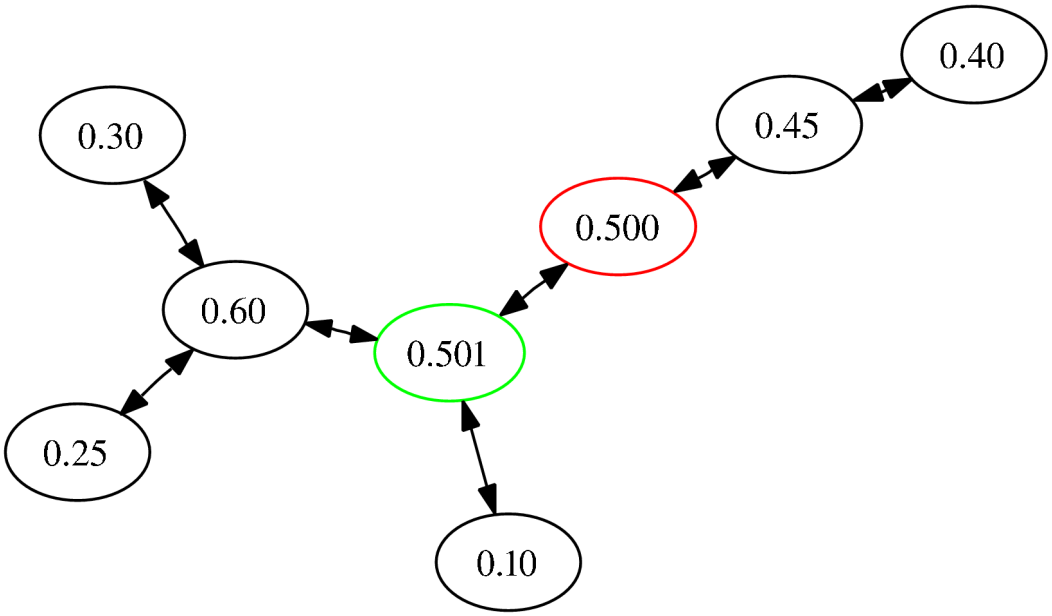
Freenet – Attack Example (2/11)



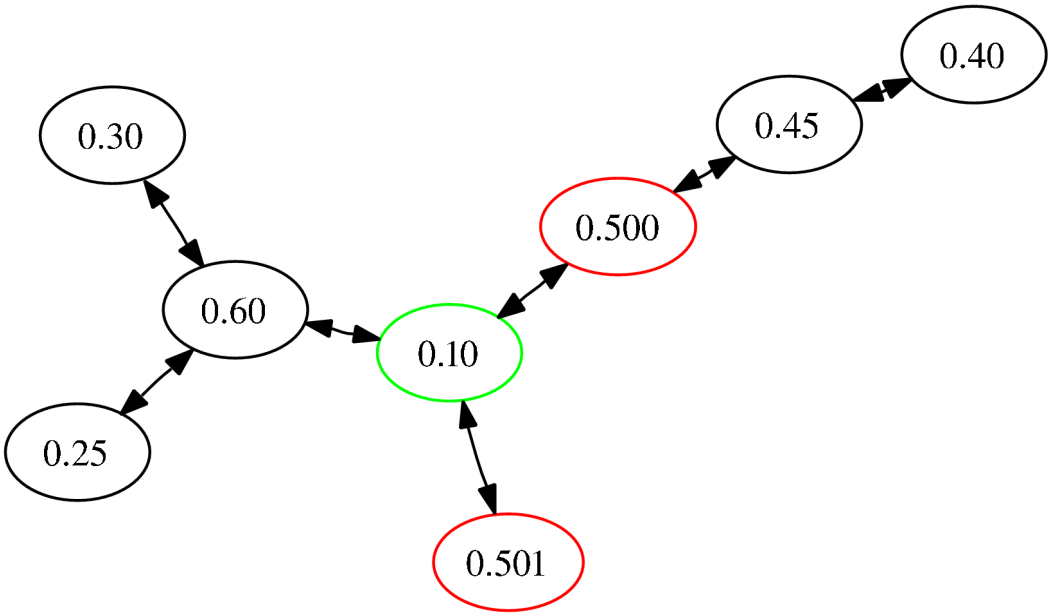
Freenet – Attack Example (3/11)



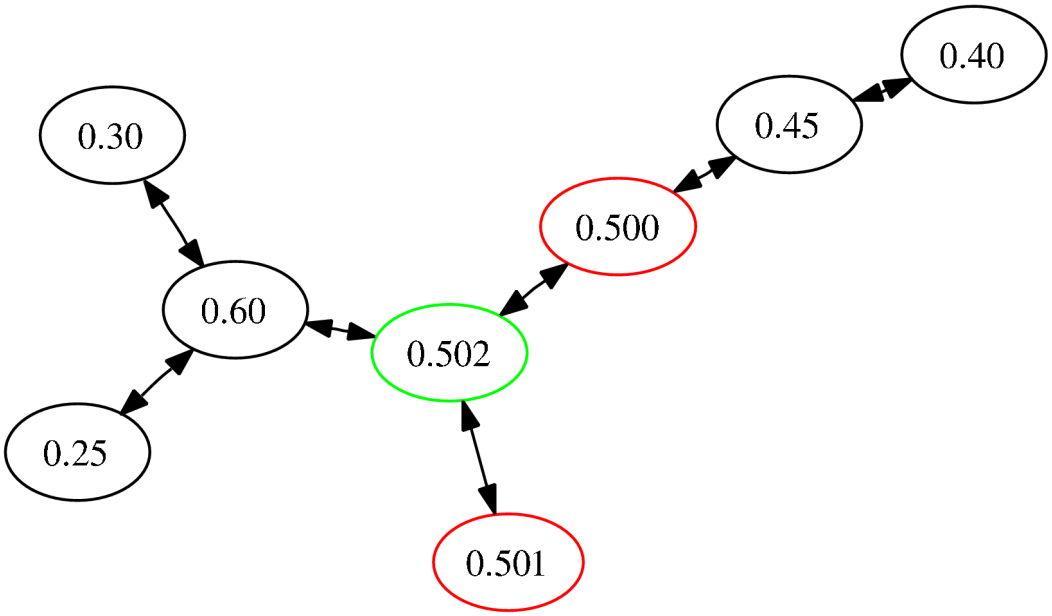
Freenet – Attack Example (4/11)



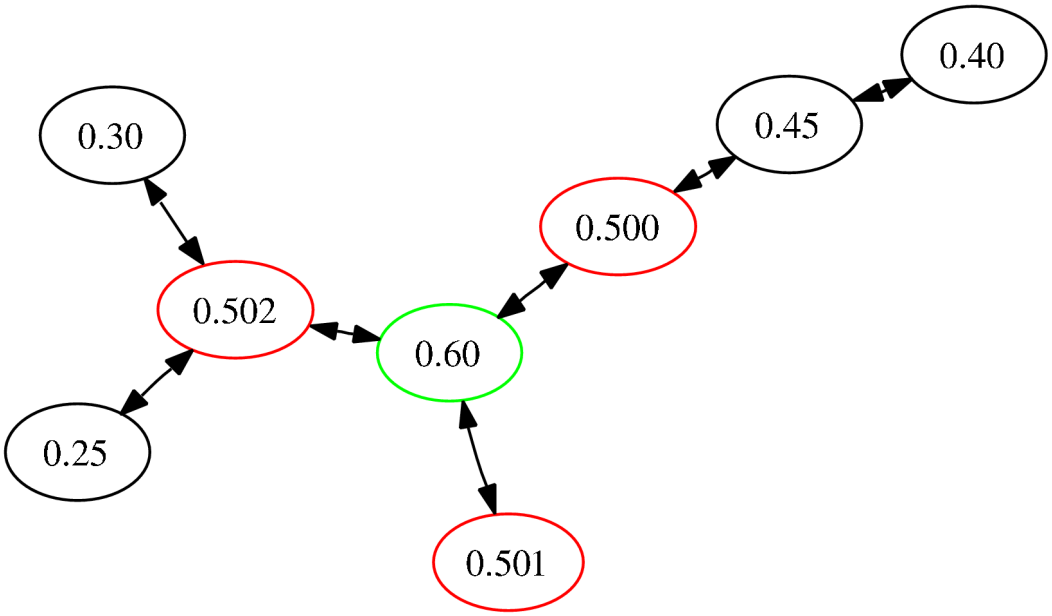
Freenet – Attack Example (5/11)



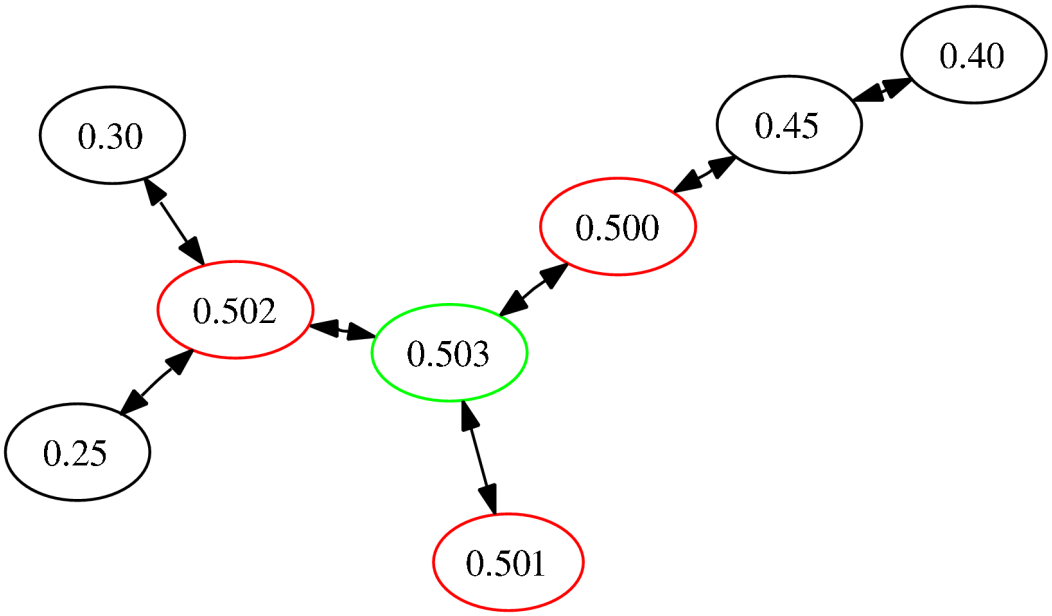
Freenet – Attack Example (6/11)



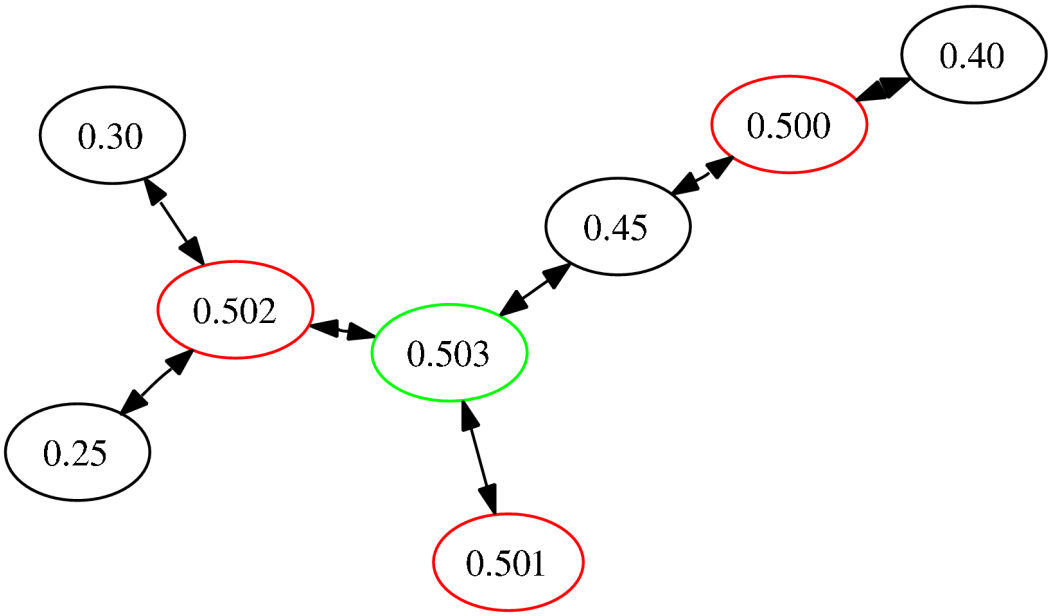
Freenet – Attack Example (7/11)



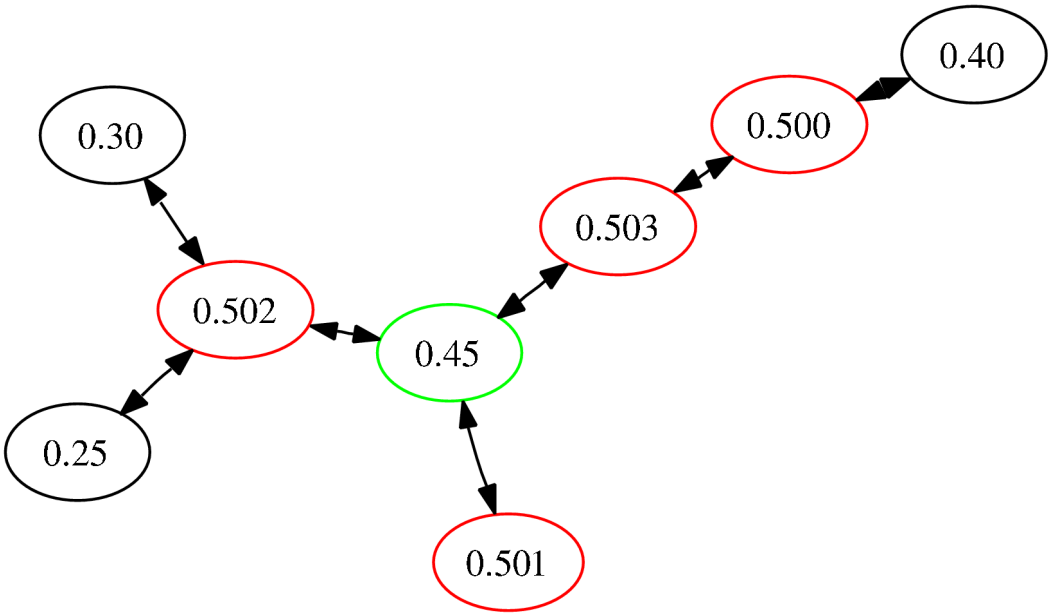
Freenet – Attack Example (8/11)



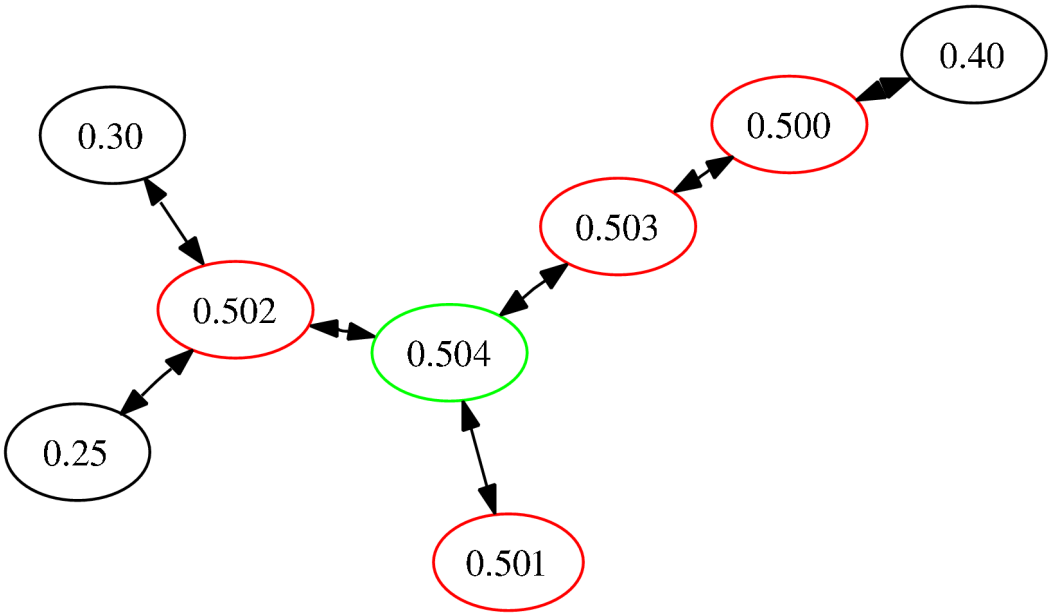
Freenet – Attack Example (9/11)



Freenet – Attack Example (10/11)



Freenet – Attack Example (11/11)



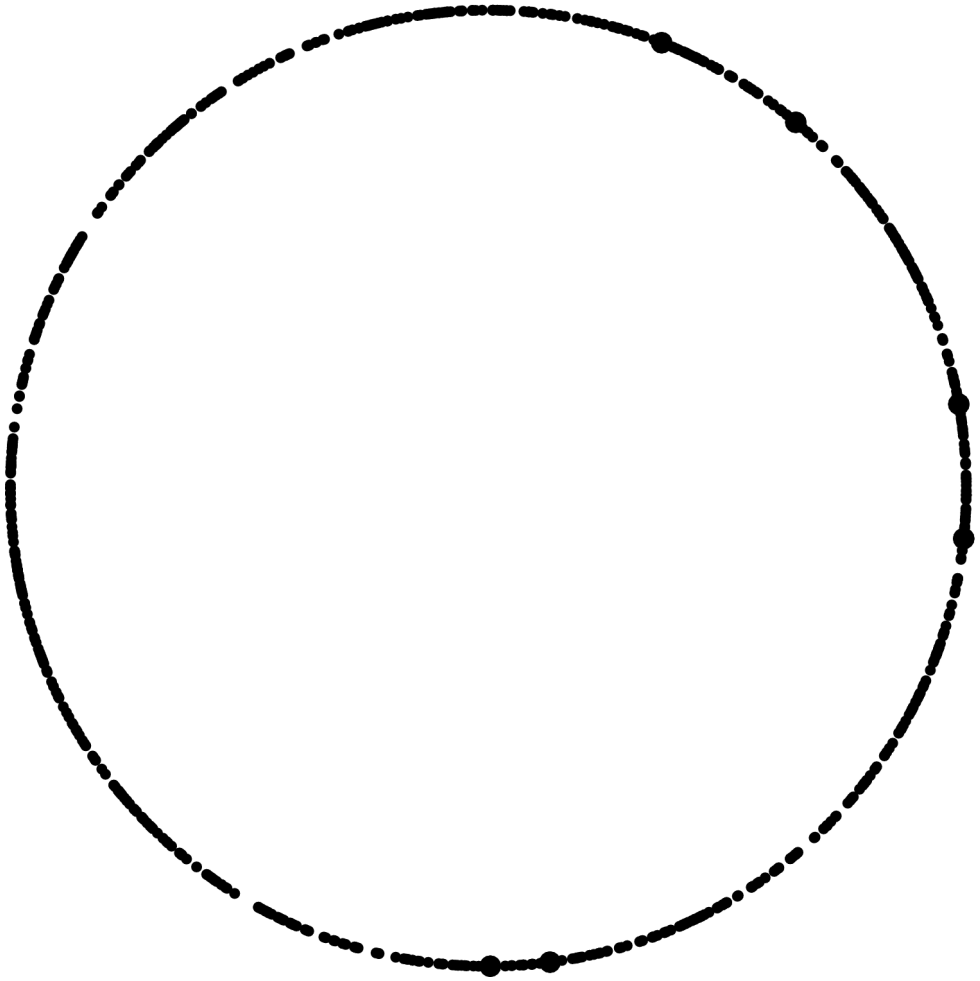
Freenet – Attack Implementation

- Malicious node uses Freenet 0.7 codebase with minor modifications
- Attacker does not violate the protocol in a detectable manner
- Malicious nodes behave as if they had a large group of friends
- Given enough time, a single malicous node can spread bad locations to most nodes
- Using multiple locations for clustering increases the speed of penetration

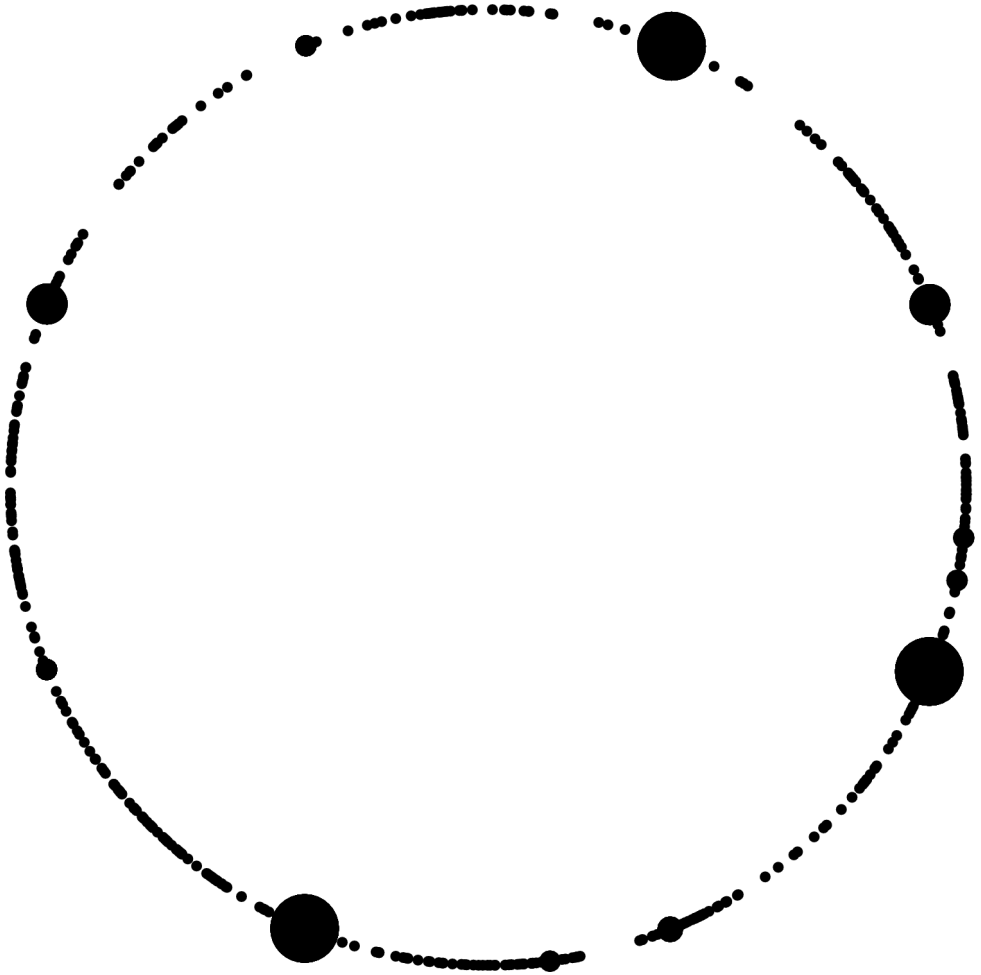
Freenet Attack – Experimental Setup

- Created testbed with 800 real Freenet nodes
- Main topology corresponds to Watts & Strogatz small world networks
- Instrumentation captures path lengths and node locations
- Content is always placed at node with closest location
- Nodes have bounded storage space
- Trials run in iterations of 90s and 45s, respectively

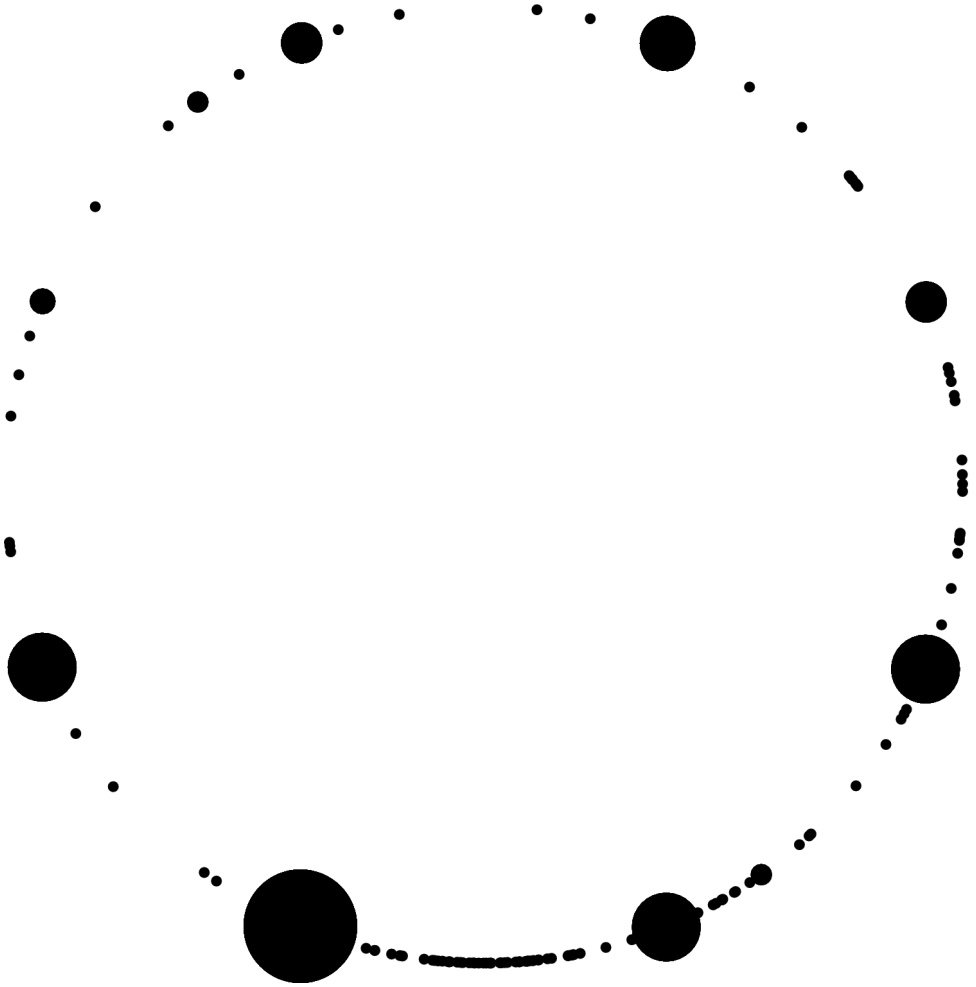
Freenet Attack – Dispersion Example (1/4)



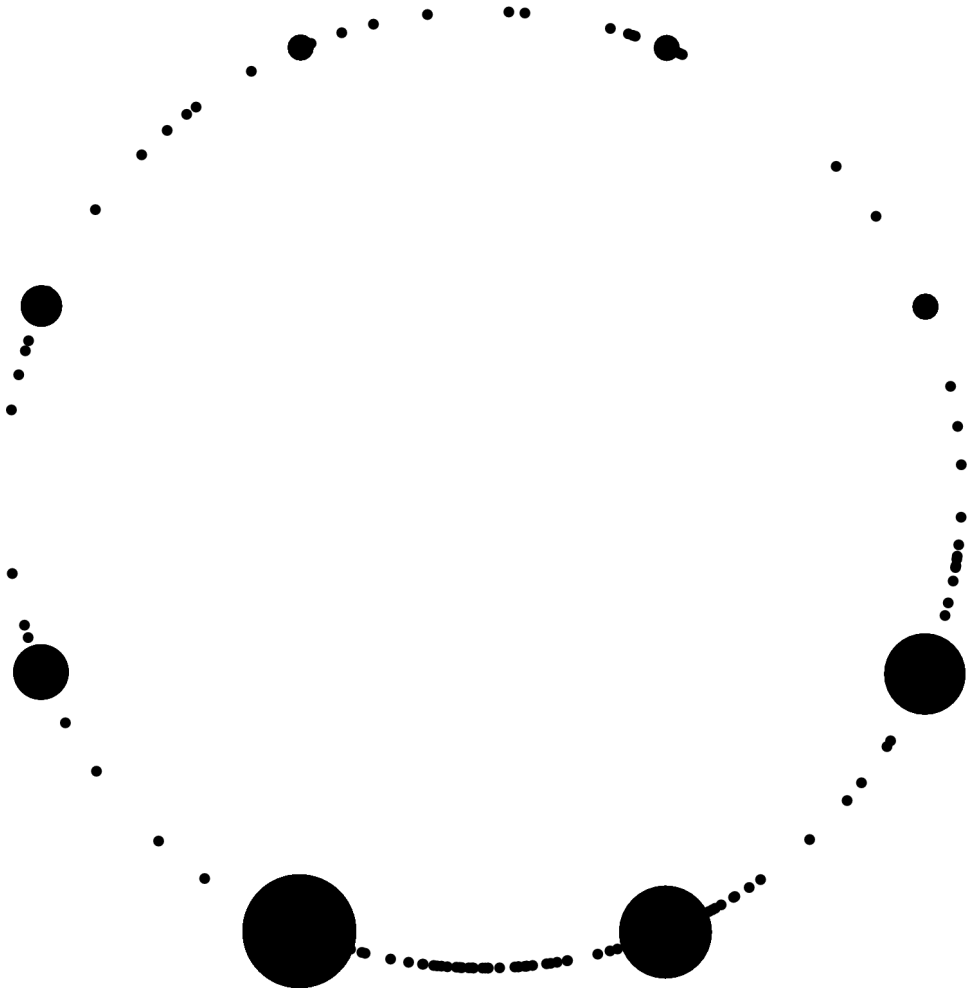
Freenet Attack – Dispersion Example (2/4)



Freenet Attack – Dispersion Example (3/4)



Freenet Attack – Dispersion Example (4/4)



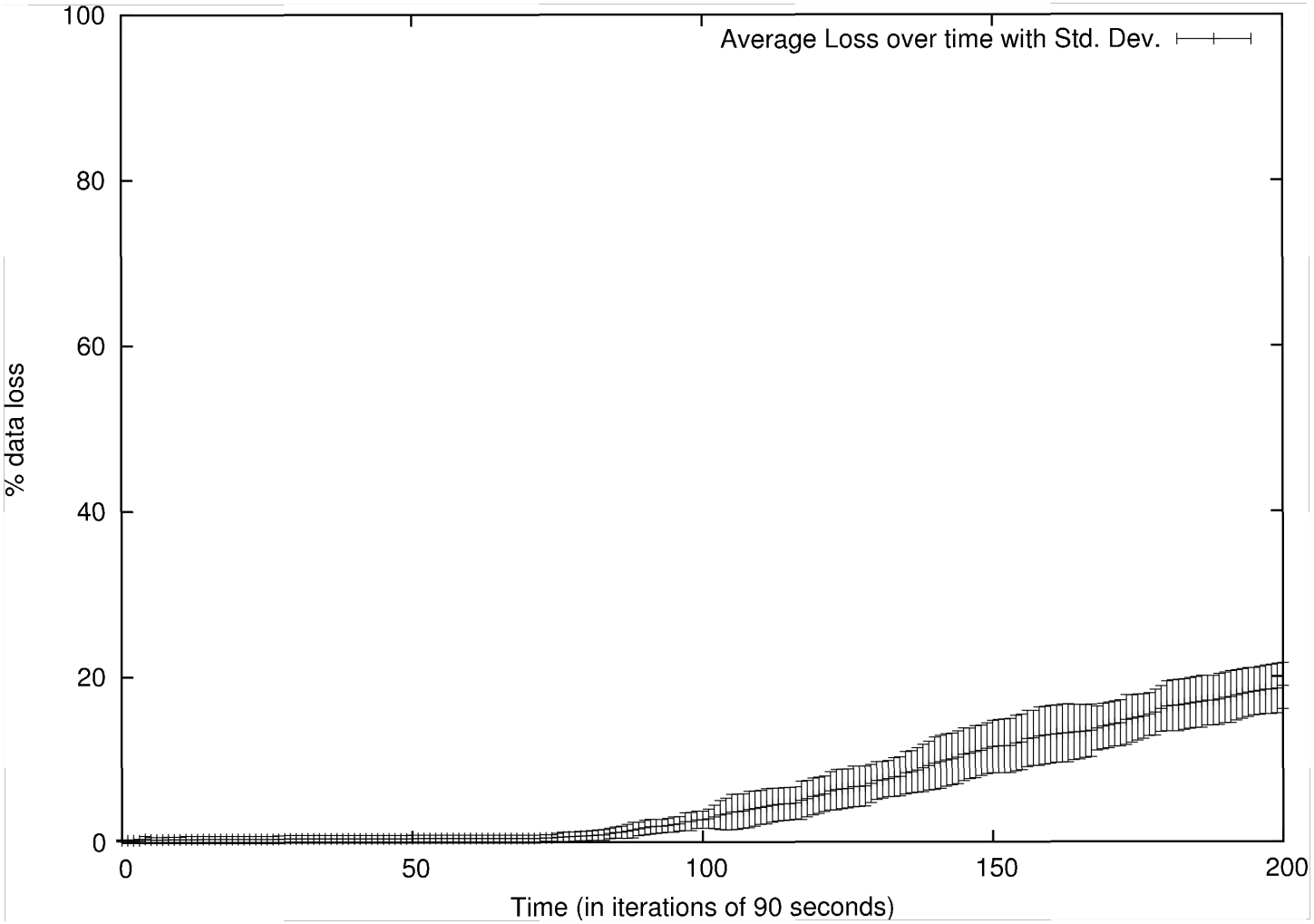
Freenet Attack - Effects

- Data Loss
  - Diversity of locations reduced
  - Peers on “edges” of clusters responsible for data in “gaps”
  - Those peers run out of storage space
  - Data is dropped

- Routing
  - Similarly, nodes on “edges” are contacted for routing more often
  - Increase in bandwidth on those peers
  - Reduces load balancing of network

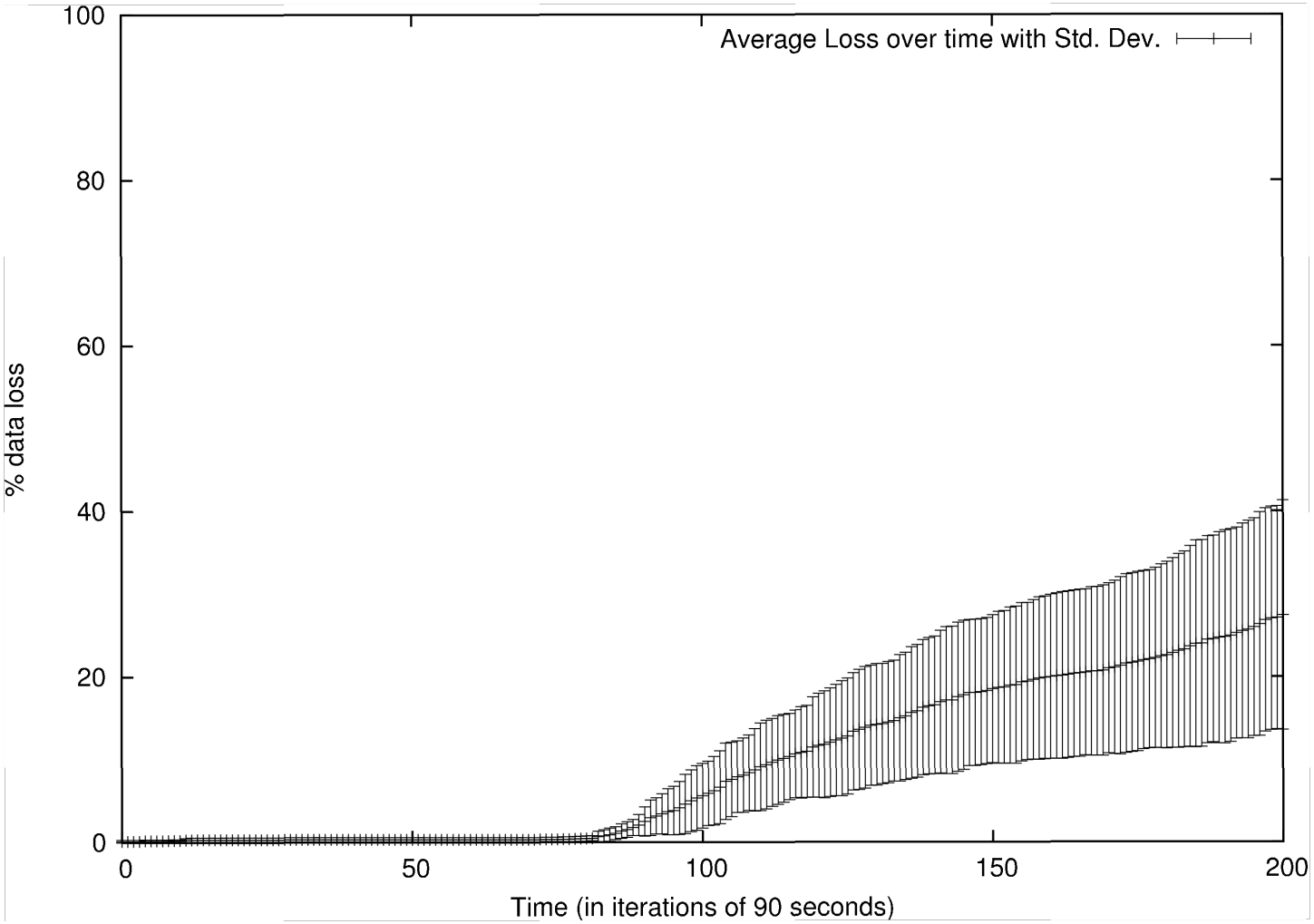
Freenet Attack – Data Loss Example (1/3)

800 Nodes – 200 iterations – 2 malicious nodes – attack begins at iteration 75



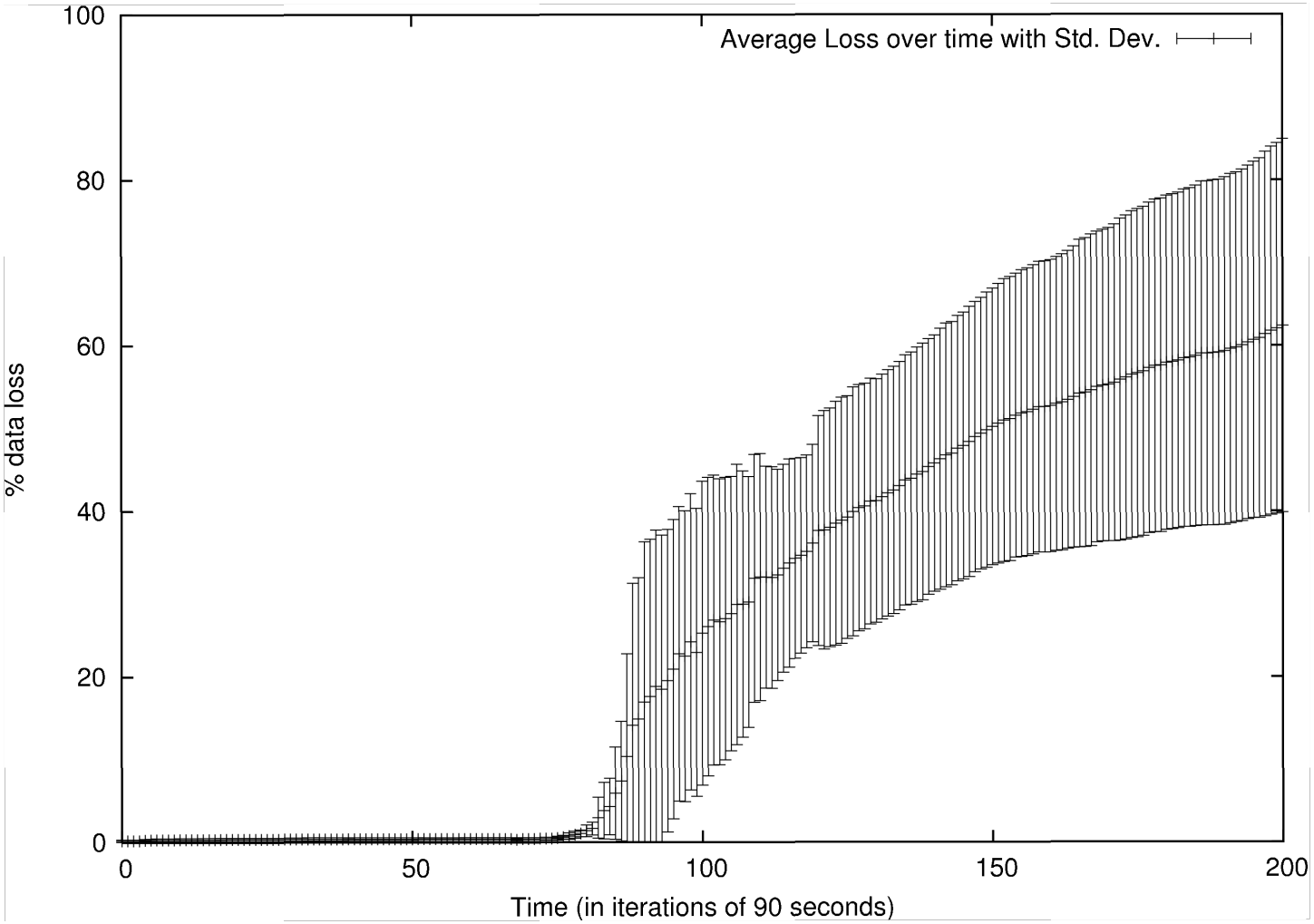
Freenet Attack – Data Loss Example (2/3)

800 Nodes – 200 iterations – 4 malicious nodes – attack begins at iteration 75



Freenet Attack – Data Loss Example (3/3)

800 Nodes – 200 iterations – 8 malicious nodes – attack begins at iteration 75



Freenet Attack - Fixes

- Check how frequently a node swaps similar locations?
  - Requires state, how similar is similar?
- Limit number of swaps with a particular peer?
  - Only swap with peer X times in Y milliseconds
  - Reduces routing performance
- Determine a node is malicious because its' location is *too* close?
  - Depends on network size
  - Defeats security/privacy goals
- Periodically reset all node locations?
  - Choose an interval, and have peers reset to random locations
  - Reduces routing performance (no experiments done)
- Secure multiparty computation for swaps?
  - Requires knowledge of topology
  - Defeats “darknet”
- In F2F networks, you can never be sure about the friends of your friends!

Freenet – Churn

- Leave join churn
  - Nodes are not constantly in the network
  - They leave for some period of time and then come back into the network
- Join leave churn
  - Nodes join the network for a time, then disconnect permanently
  - Causes node clustering
  - Results in load imbalances similar to the described attack (only more slowly)
- Churn clustering
  - P2P networks often have “stable core”
  - Other peers come and go
  - Stable core generally well connected
  - Swapping causes stable core to cluser locations

Freenet Attack/Churn – Chosen Workaround

- Periodic location resets
  - Freenet 0.7 peers reassign themselves locations
  - Interval chosen impacts routing performance
  - Resilience depends on network size
  - This hurts the scalability of the network

- Developers estimate this “fix” works to combat churn based location clustering, but not necessarily an active attack.

- No comprehensive studies have been done on effectiveness.

Freenet – Current State

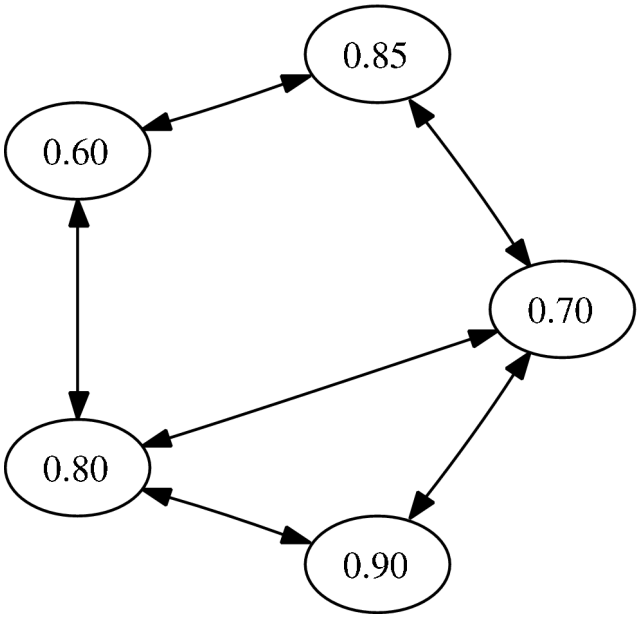
- **Project Development**
  - Currently still active
  - One full time developer
  - Many contributors
  - Frequent Google SoC project

- **Darknet Status**
  - Darknet great for security, difficult for users
  - Current Freenet version can operate in “opennet” mode or “darknet” mode
  - Opennet allows random connections
  - Darknet allows only known friend connections
  - No solid data on users, but most new users forced to use opennet

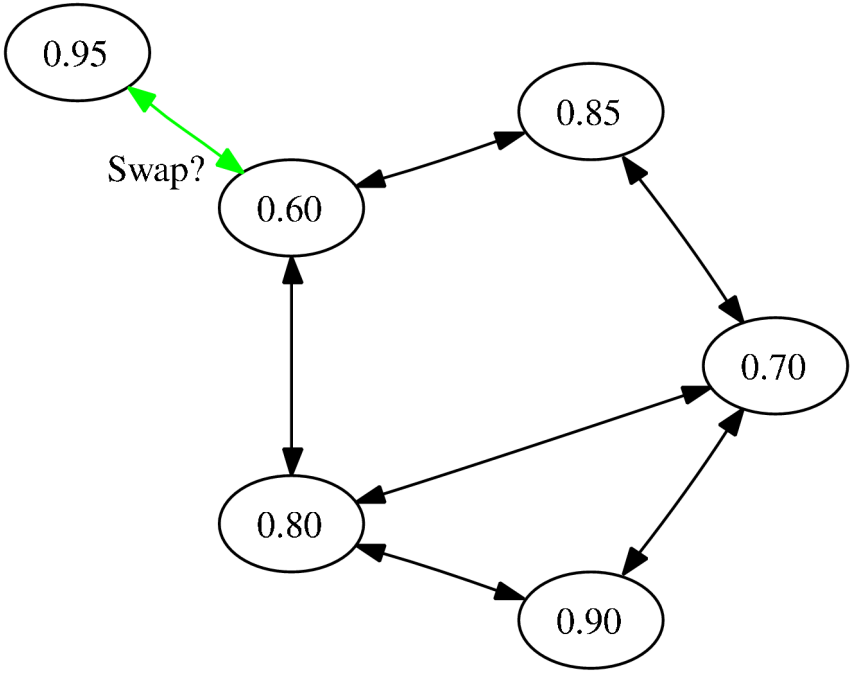
Freenet - Conclusion

- Unique P2P network
  - Typical DHT's used exclusively for file sharing
  - Long lived project
  - Freenet has rich set of applications
  - Large set of Freesites, indexes
  - Split file downloads
- F2F “Darknet”
  - Provides better security
  - Difficult in practice
- Swap attack
  - Reduces performance
  - Never seen in the wild
- Try it out (Freenet, not the attack)!

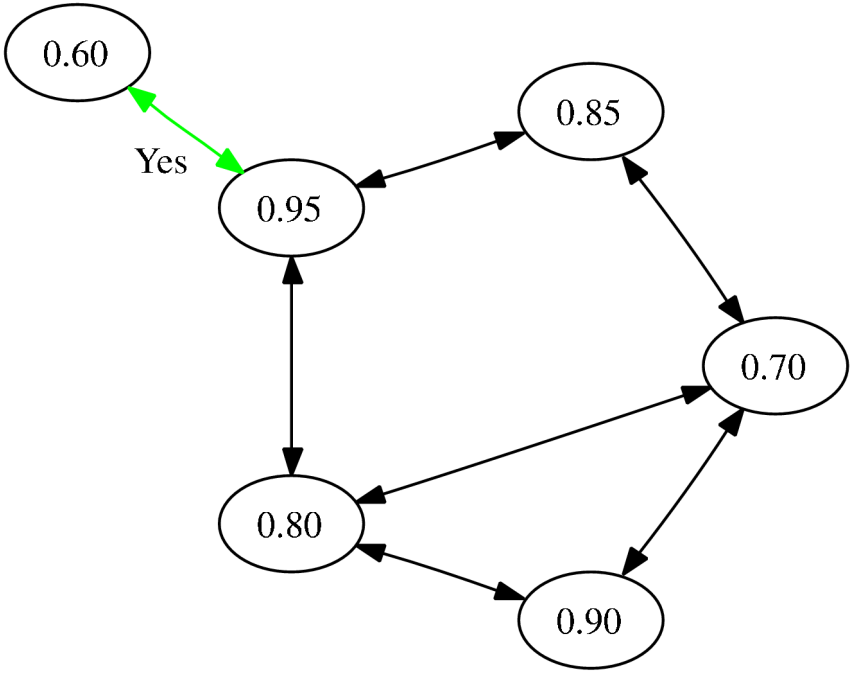
Freenet – Churn Example (1/13)



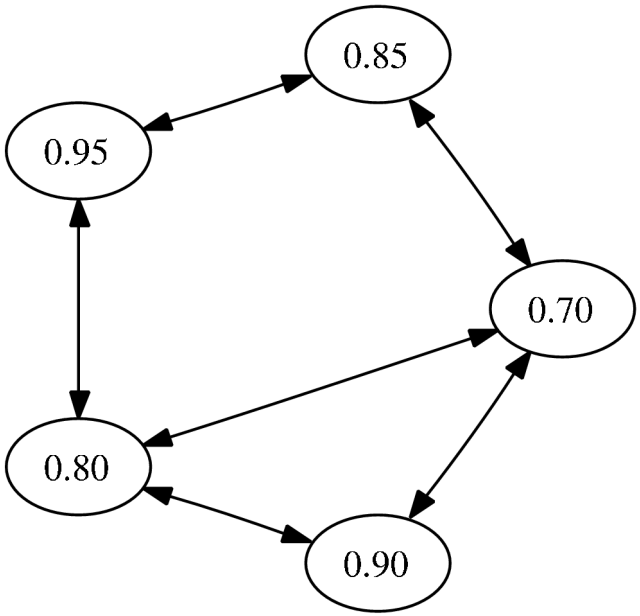
Freenet – Churn Example (2/13)



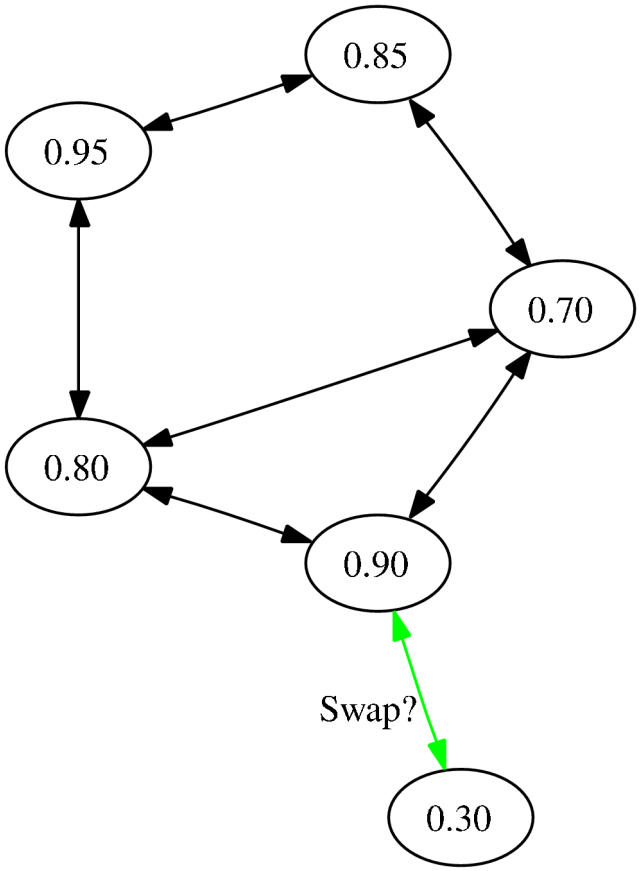
Freenet – Churn Example (3/13)



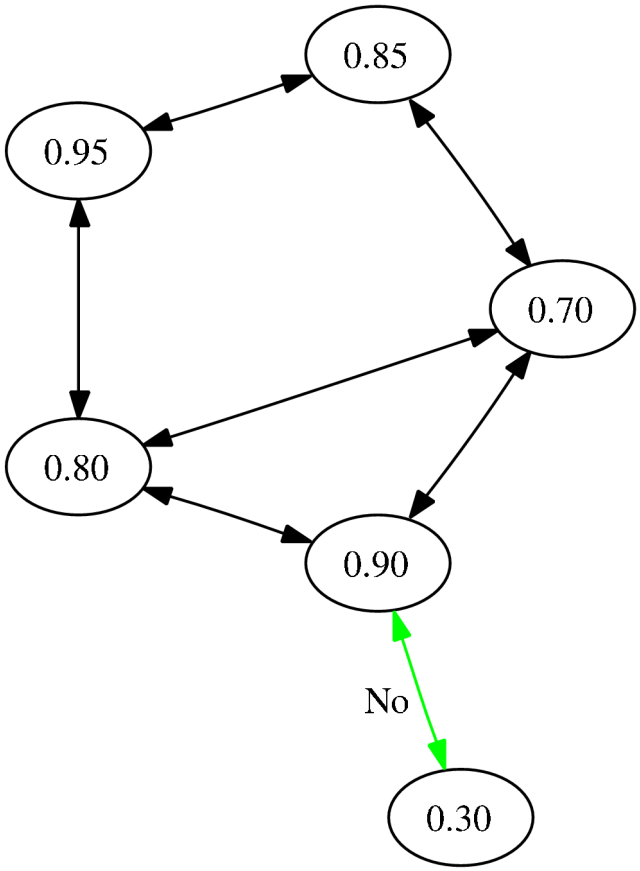
Freenet – Churn Example (4/13)



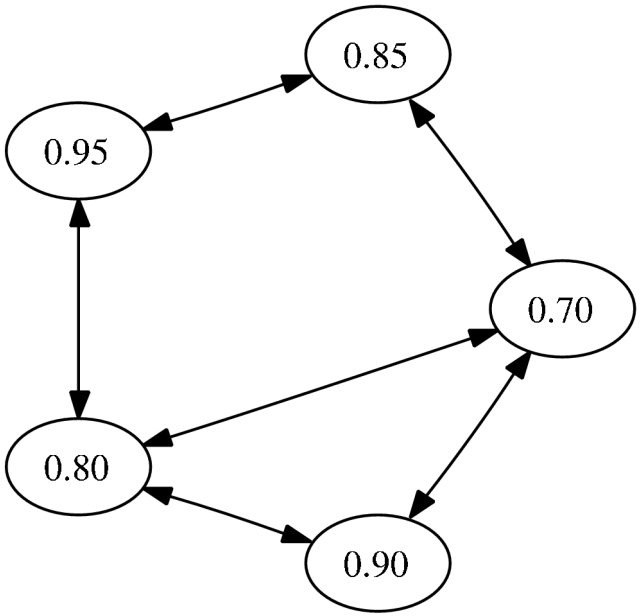
Freenet – Churn Example (5/13)



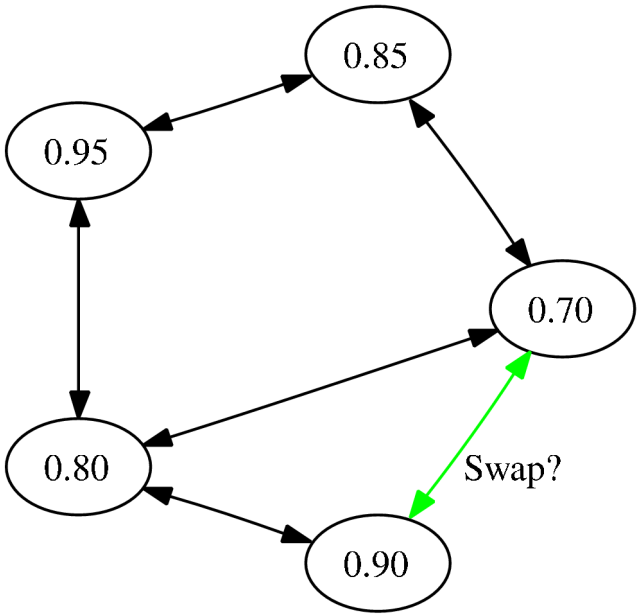
Freenet – Churn Example (6/13)



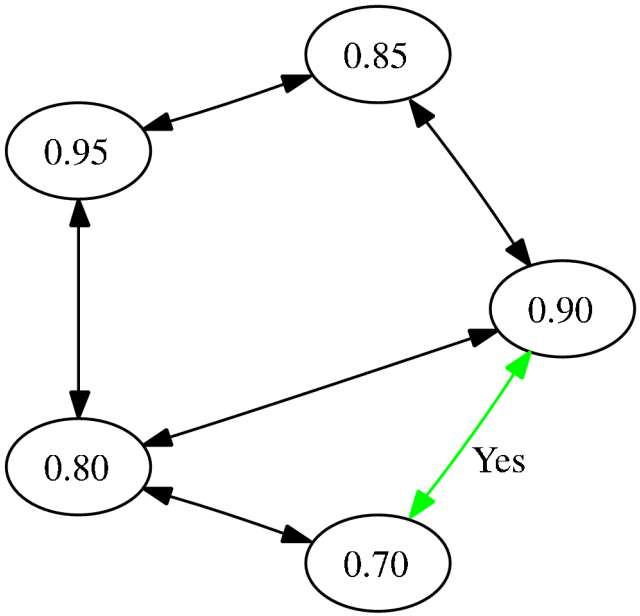
Freenet – Churn Example (7/13)



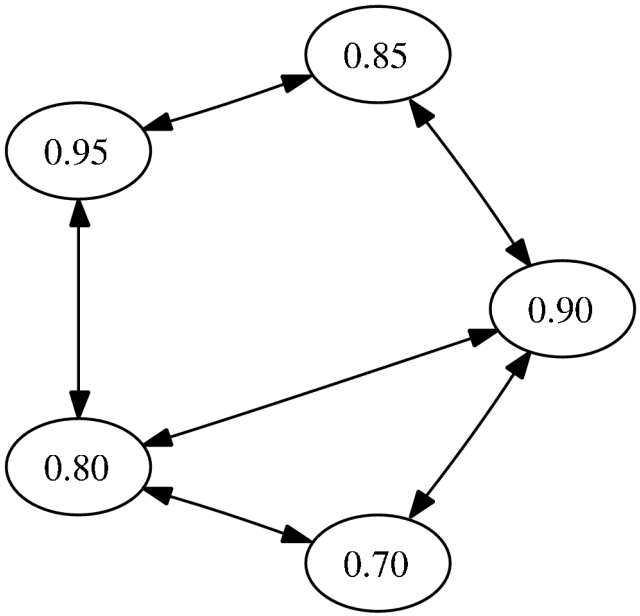
Freenet – Churn Example (8/13)



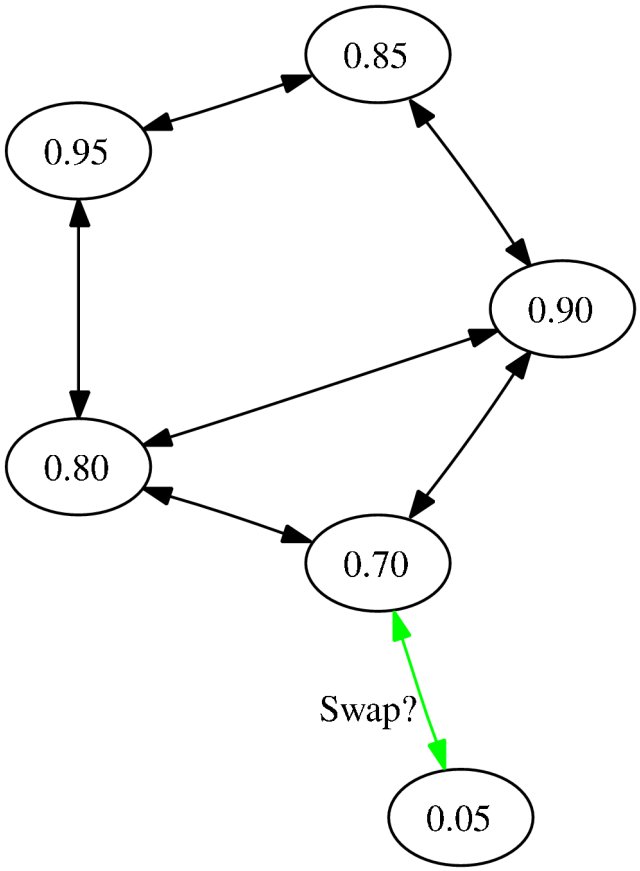
Freenet – Churn Example (9/13)



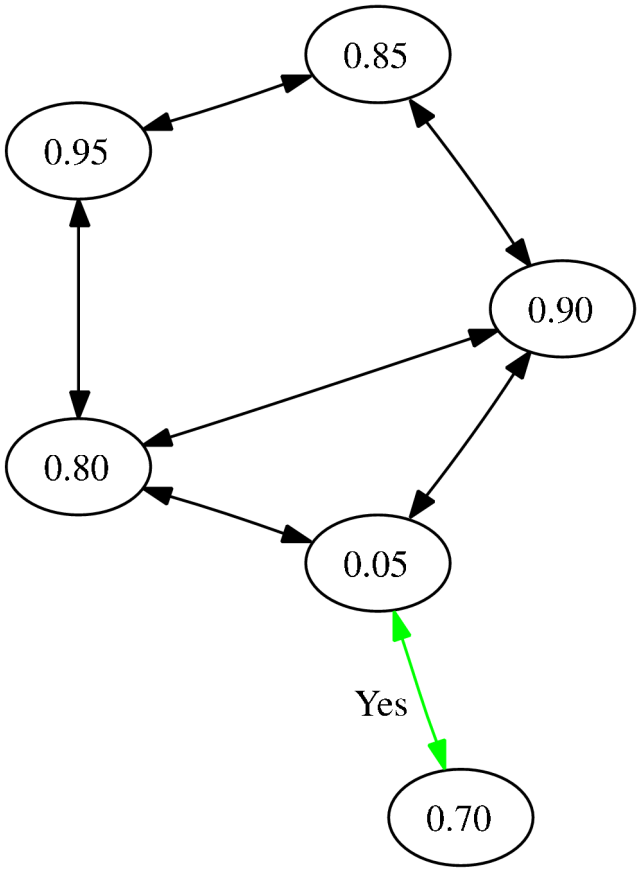
Freenet – Churn Example (10/13)



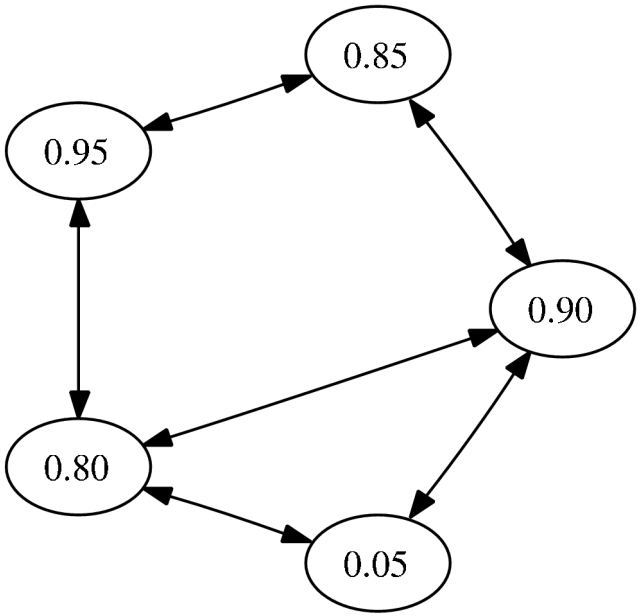
Freenet – Churn Example (11/13)



Freenet – Churn Example (12/13)



Freenet – Churn Example (13/13)



Freenet 0.7 – Churn Simulations

- Created stable core of nodes
- Simulated join-leave churn, let network stabilize
- Ran exactly the native swap code
- Repeat n times
- Revealed drastic convergence to single location
- http://crisp.cs.du.edu/pitchblack/