FreeNet

A Distributed Anonymous Information Storage and Retrieval System

Presented By Xiao Wei-Cheng

2004.04.06

Outline

- Introduction
- Architecture
  - Keys and searching files
  - Inserting, Storing, Retrieving, Managing files
  - Adding nodes
- Security
- Performance Analysis

Introduction (1/2)

- Decentralization
- Privacy
- Sharing of Storage Space
- Location–independent file system
- Retrieving, Inserting, Storing files

Introduction (2/2)

- Design Goals
  - Anonymity for producers and consumers
  - Deniability for storers of information
  - Resistance to attempts to deny access of information
  - Efficient dynamic storage and routing of information
  - Decentalization of ALL network functions

Architecture

- Freenet is like a peer–to–peer network
- Files are named by keys
- Each node has its own datastore and routing table
- Routing is driven by 'key'
- Each request has a unique ID
  - prevent loop

Keys and Searching Files – KSK (1/4)

- Keyword–signed key (KSK) is derived from Descriptive Text String (DTS)
- public/private key pair
- The file key is yielded by hashing the public part
- The file is encrypted with DTS
- The private part is used to sign the file
- Problem – Different files have the same DTS

Keys and Searching Files – SSK (2/4)

- Signed–subspace key
- Personal namespace is enabled, and generated randomly
- Files key = hash(XOR(hash(namespace), hash(DTS)))
- File is encrypted with DTS as KSK
- Private key is needed when storing the file

Keys and Searching Files–CHK (3/4)

- Content–hash key
- File key is derived by hashing the file content
- Files are encrypted by randomly–generated keys
- CHK is usually conjucted with SSK
  - Indirect file
  - Version updating
  - file splitting

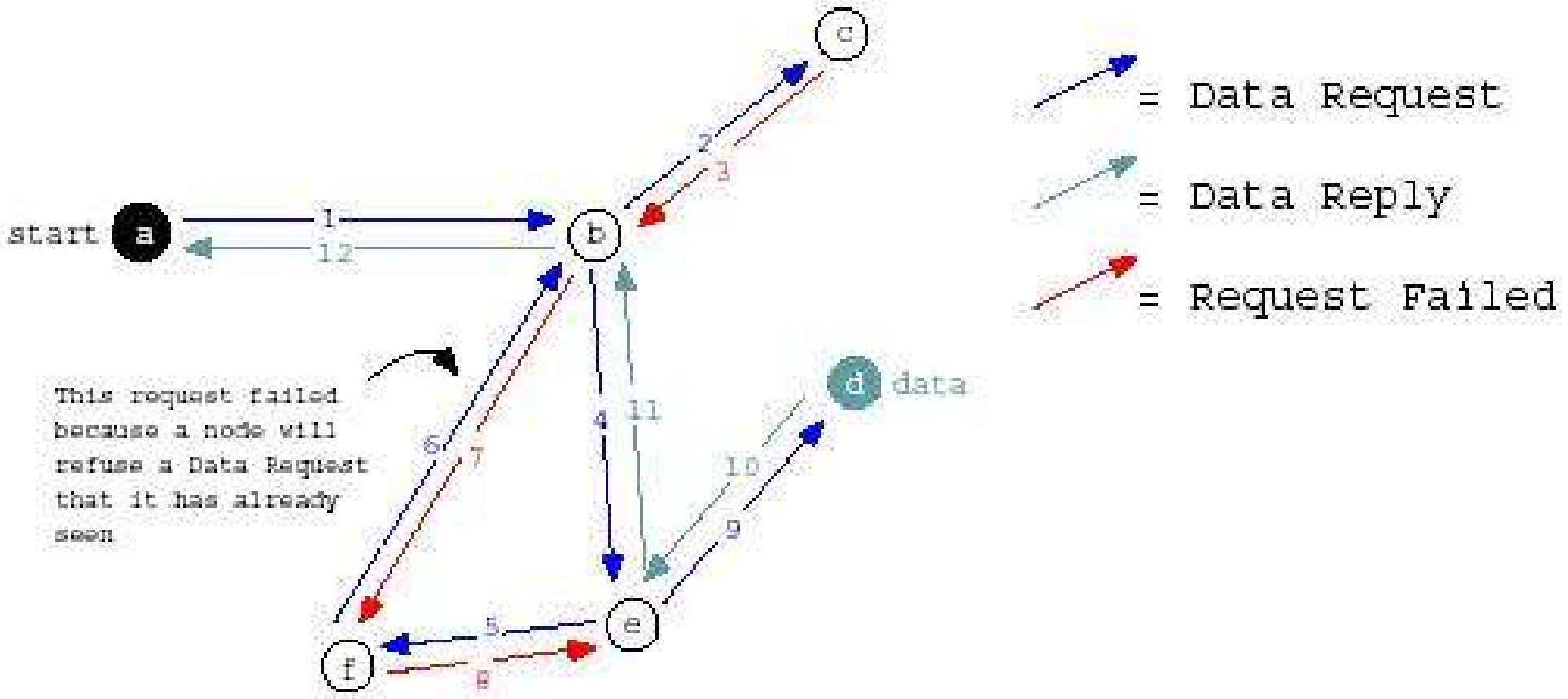
Keys and Searching Files (4/4)

- Problem – How to get the file key ?
  - Through Web Server
  - Lightweight indirect files
    - Multiple indirect files may have the same key
  - Still an open problem

Retrieving File (1/2)

- A Request.Data message is sent, with transaction ID, hops–to–live, depth, and search key
- A Send.Data message and the desired file will be sent back after successful request
- In the nodes on the path, file is cached, and routing table is updated
- A Reply.NotFound message would be sent back if failed
- Files with similar keys would be cached in some group of nodes

Retrieving File (2/2)



Example of retrieving a file (DFS)

Storing and Inserting File

- A Request.Insert message is sent
- If inserting sucesses, a Reply.Insert message is sent back, and a Send.Insert is then sent by the requestor
- If failed, a Send.Data message with the existing data or a Reply.NotFound message is sent back
- In the nodes on the path, file is cached, and routing table is updated
- New nodes can use inserts to announce their existence

Managing File

- Storages have finite capacity
- LRU algorithm is used to manage files
- Entries in routing table are deleted only when the routing table is full

Adding Nodes

- All nodes have to be consistent in deciding the new node key (Address Resolution Key)
- The new node sends its address and hash(rand()) out first
- Nodes in the path send hash(rand() XOR prehash) to the next one
- The final hash value becomes the key of the new node

Security (1/3)

- Anonymity of sender, receiver, and the key
- Key anonymity is impossible since routing depends on the key
- For malicious nodes, sender anonymity is preserved beyond suspicion

System	Attacker	Sender anonymity	Key anonymity
Basic Freenet	local eavesdropper	exposed	exposed
	collaborating nodes	beyond suspicion	exposed
Freenet + pre-routing	local eavesdropper	exposed	beyond suspicion
	collaborating nodes	beyond suspicion	exposed

Security (2/3)

- Freenet + pre-routing
  - For key anonymity and sender anonymity
  - Messages are encrypted by a succession of public keys, and pre-routed first
  - After pre-routing, the message is injected into the normal Freenet network
- The data source field can be resetted in the path
- A hops-to-live of 1 doesn't reveal an endpoint
  - Finite probability

Security (3/3)

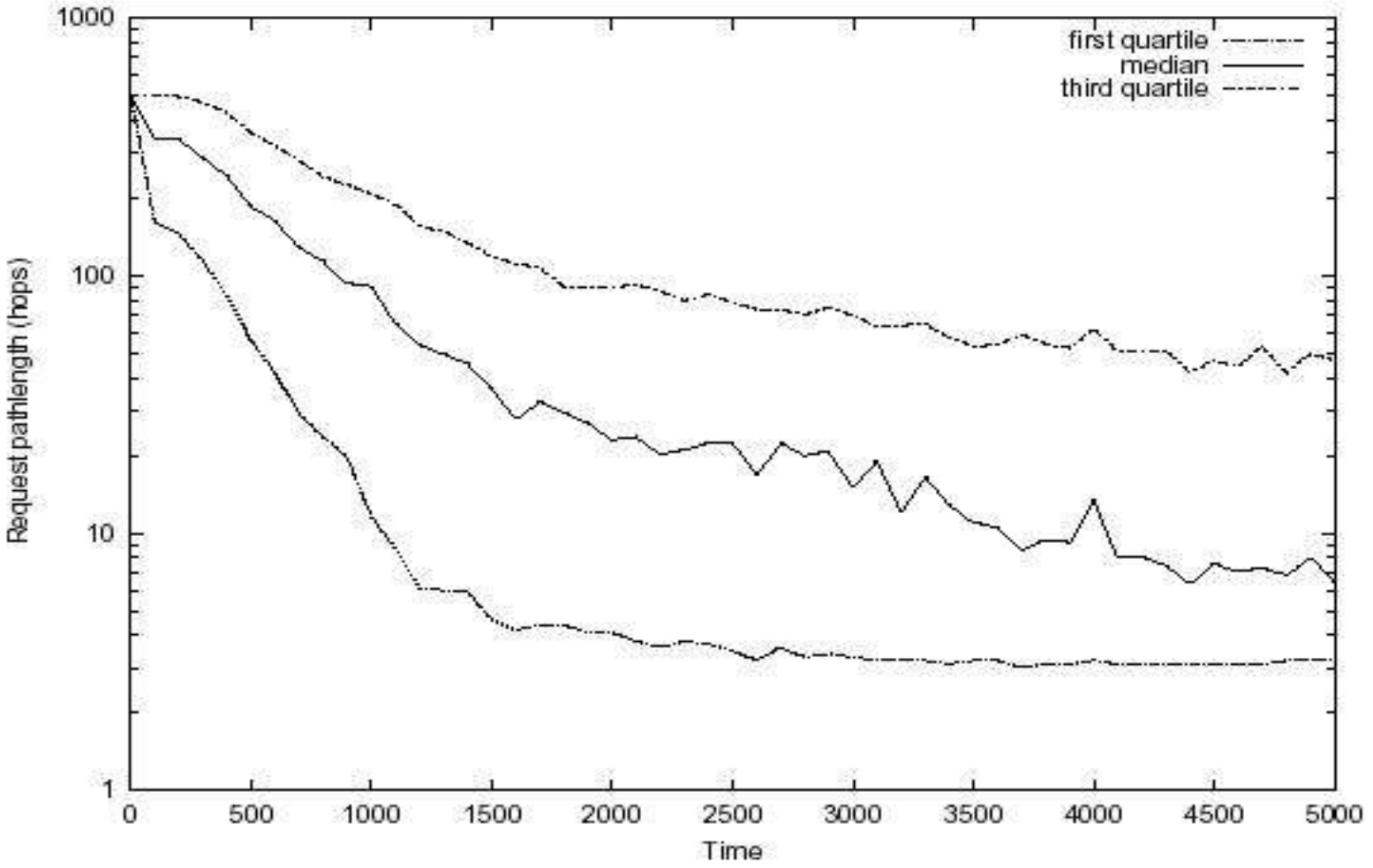
- Modification of requested files by malicious nodes
  - Not feasible under CHK or SSK
- Displace existing files by malicious nodes
  - Not feasible under CHK or SSK
- Prevent DoS attack
  - Use 2 part of datastore
    - Established files
    - New files

Performance Analysis (1/5)

- Network for simulation
  - 1000 nodes
  - Datastore size of 50 items per node
  - Routing table size of 250 addresses per node

Performance Analysis (2/5)

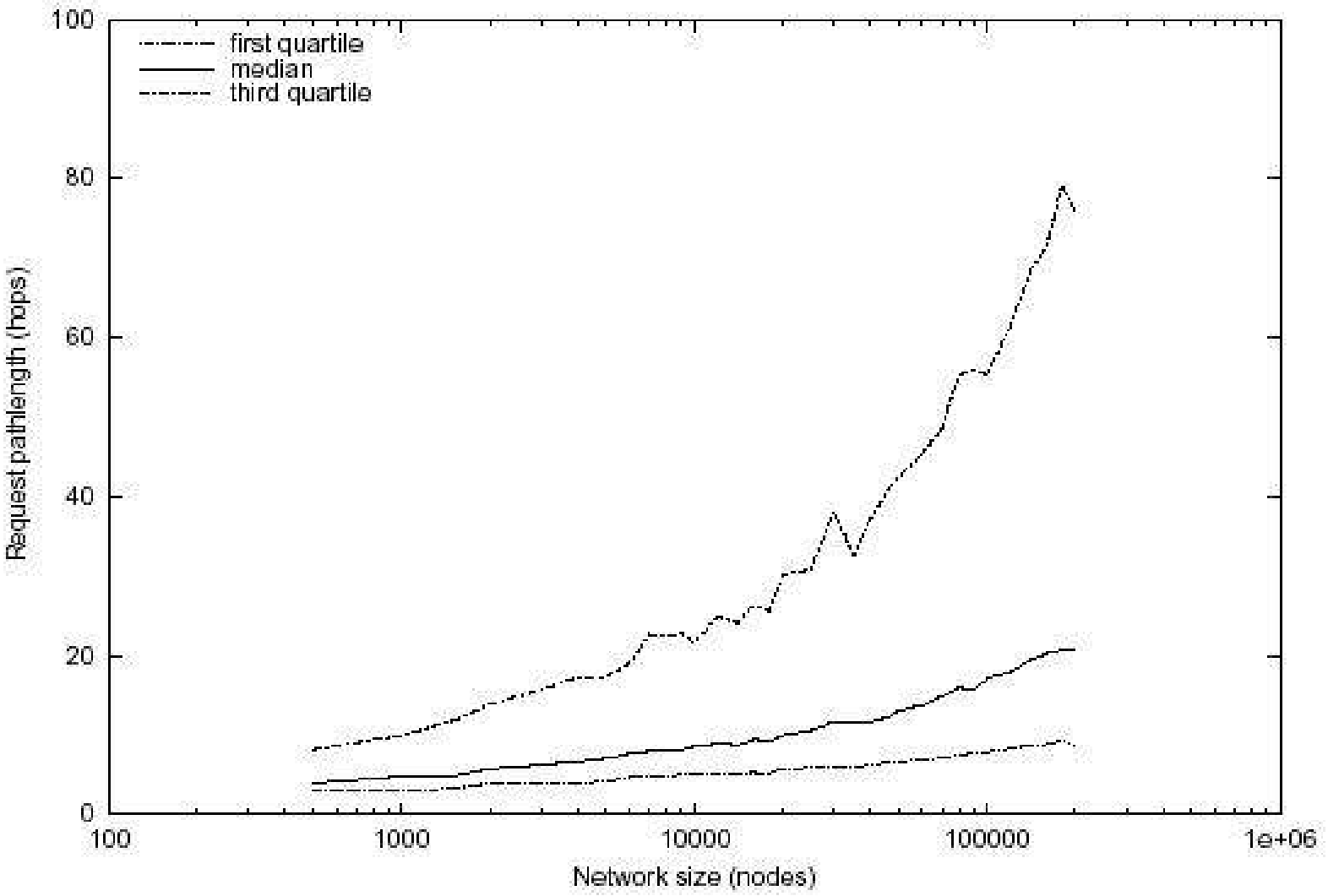
- Network Convergence



300 random requests per probe, hops–to–live = 500, every 100 timestemps

Performance Analysis (3/5)

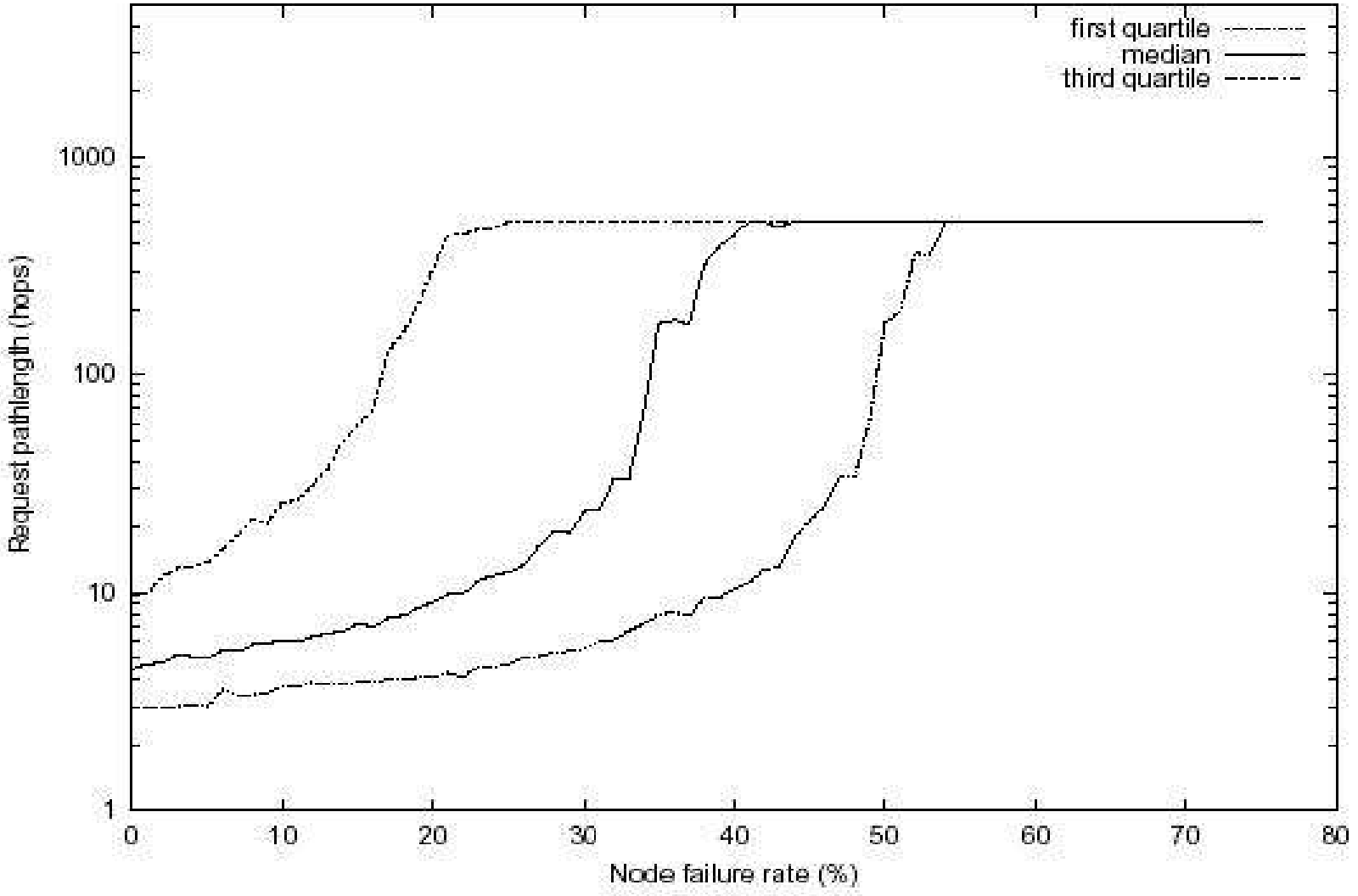
- Scalability



Probing as previous simulation

Performance Analysis (4/5)

- Fault–tolerance – Because of small–world network



Performance Analysis (5/5)

- Small–world network