Privacy-Preserving P2P Data Sharing with OneSwarm

Tomas Isdal, Michael Piatek,
Arvind Krishnamurthy, Thomas Anderson

University of Washington

http://oneswarm.cs.washington.edu/

Data sharing

Data sharing

1. Private

Data sharing

1. Private

2. Public (not sensitive)

Data sharing

1. Private

2. Public (not sensitive)

3. Public but *without attribution*

Data sharing

1. Private

2. Public (not sensitive)

3. Public but *without attribution*

Data sharing

This talk

1. Private

2. Public (not sensitive)

3. Public but *without attribution*

Providers

- Sharing provided by:
  - *Centralized* data-centers
  - *Peer-to-peer* networks

Providers

- Sharing provided by:
  - *Centralized* data-centers
  - *Peer-to-peer* networks

P2P privacy

- No centralized control / trust
  - No need to *sign over your data* to a centralized service
  - No no need to tell a centralized service *what you share or download*

P2P privacy

- No centralized control / trust
  - No need to *sign over your data* to a centralized service
  - No no need to tell a centralized service *what you share or download*

- Disadvantage of common P2P systems: *anyone can monitor your behavior!*

Previous solutions

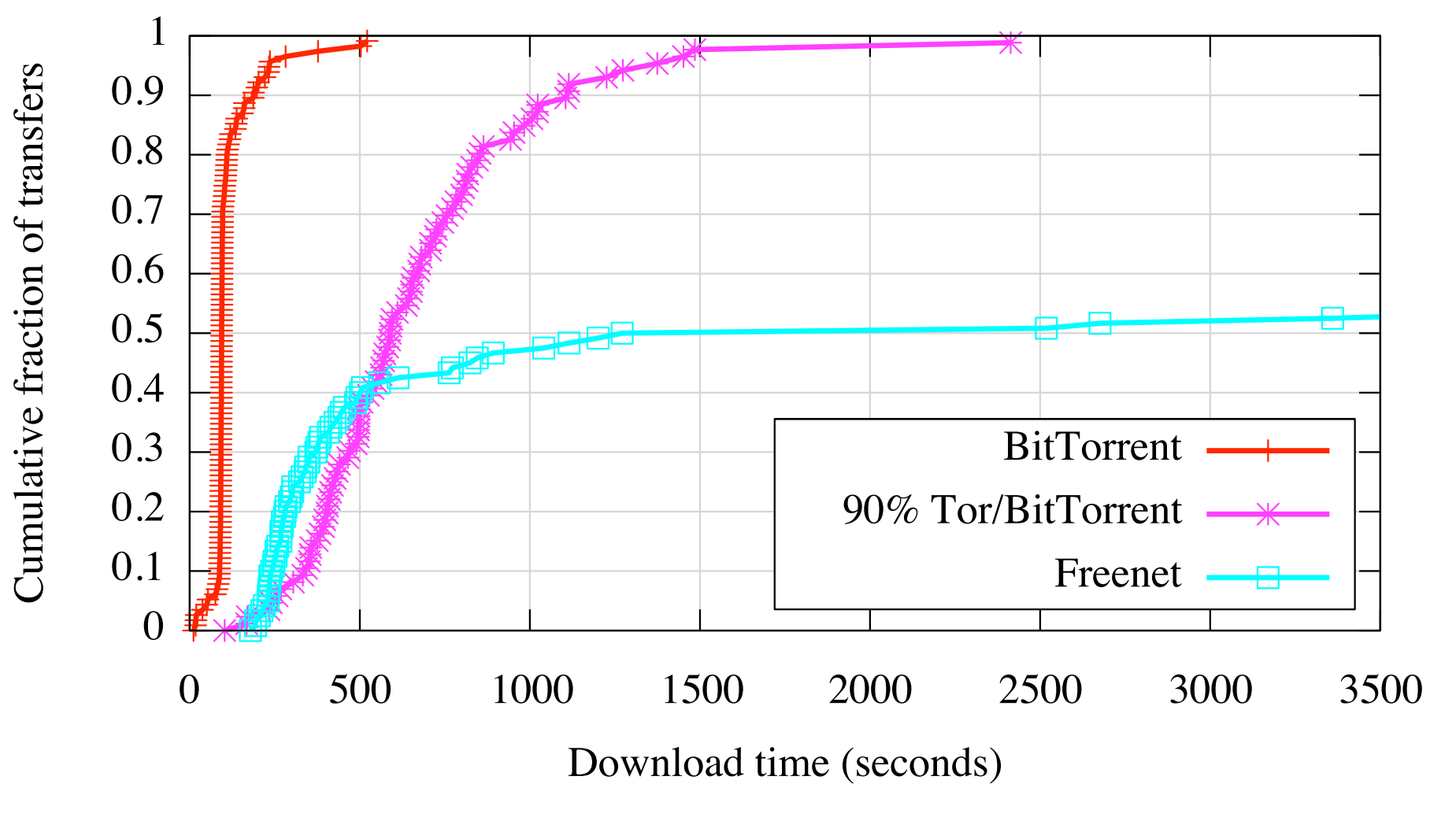
Previous solutions

- Solution 1: Put *Tor* in front of BitTorrent
  - Problems:
    - Requires a fraction of clients to be public
    - Discouraged for *performance* reasons

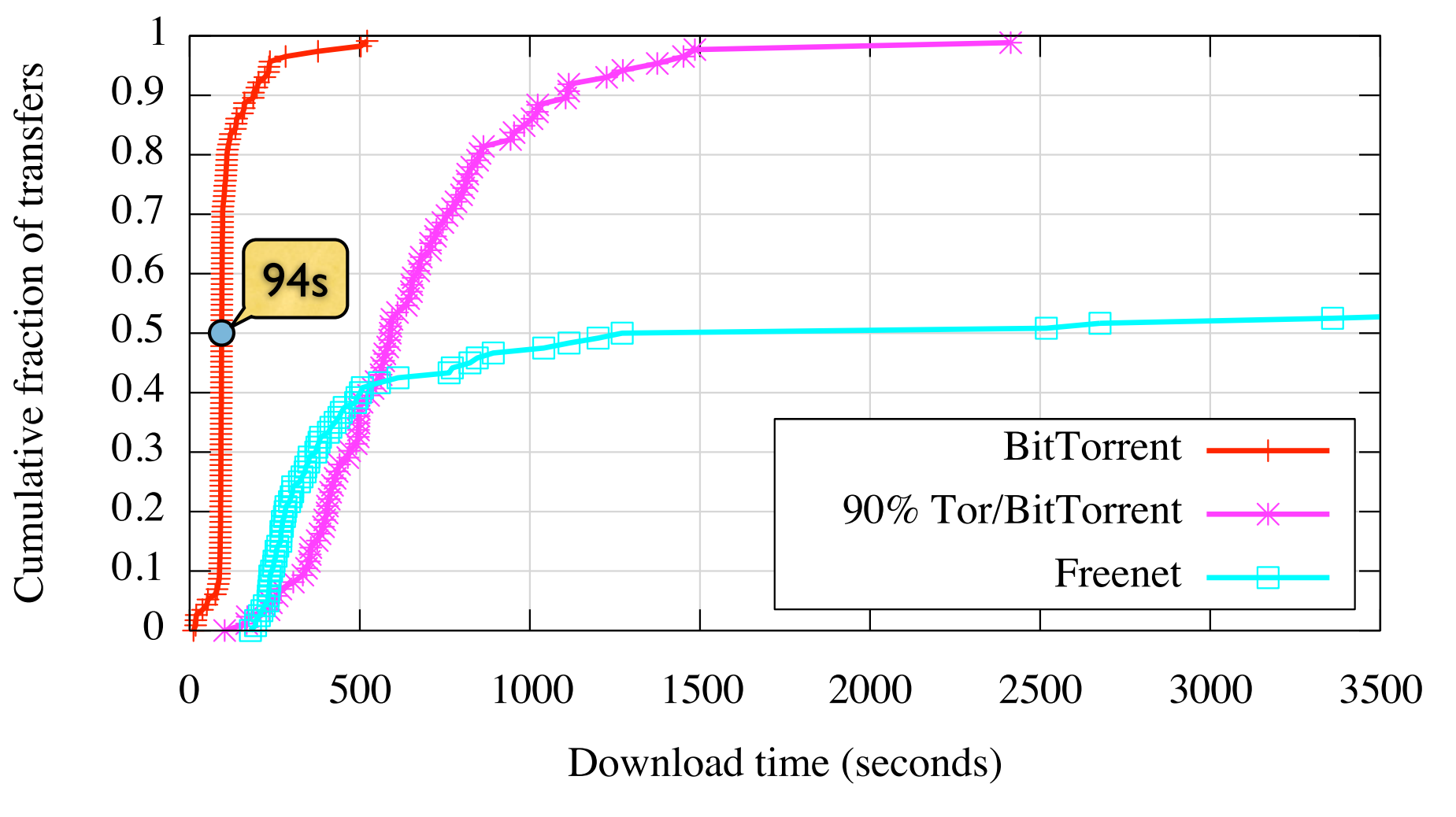
Previous solutions

- Solution 1: Put *Tor* in front of BitTorrent
  - Problems:
    - Requires a fraction of clients to be public
    - Discouraged for *performance* reasons
- Solution 2: *Freenet*
  - Problems:
    - Poor bulk data *performance*
    - Requires users to store other peoples data

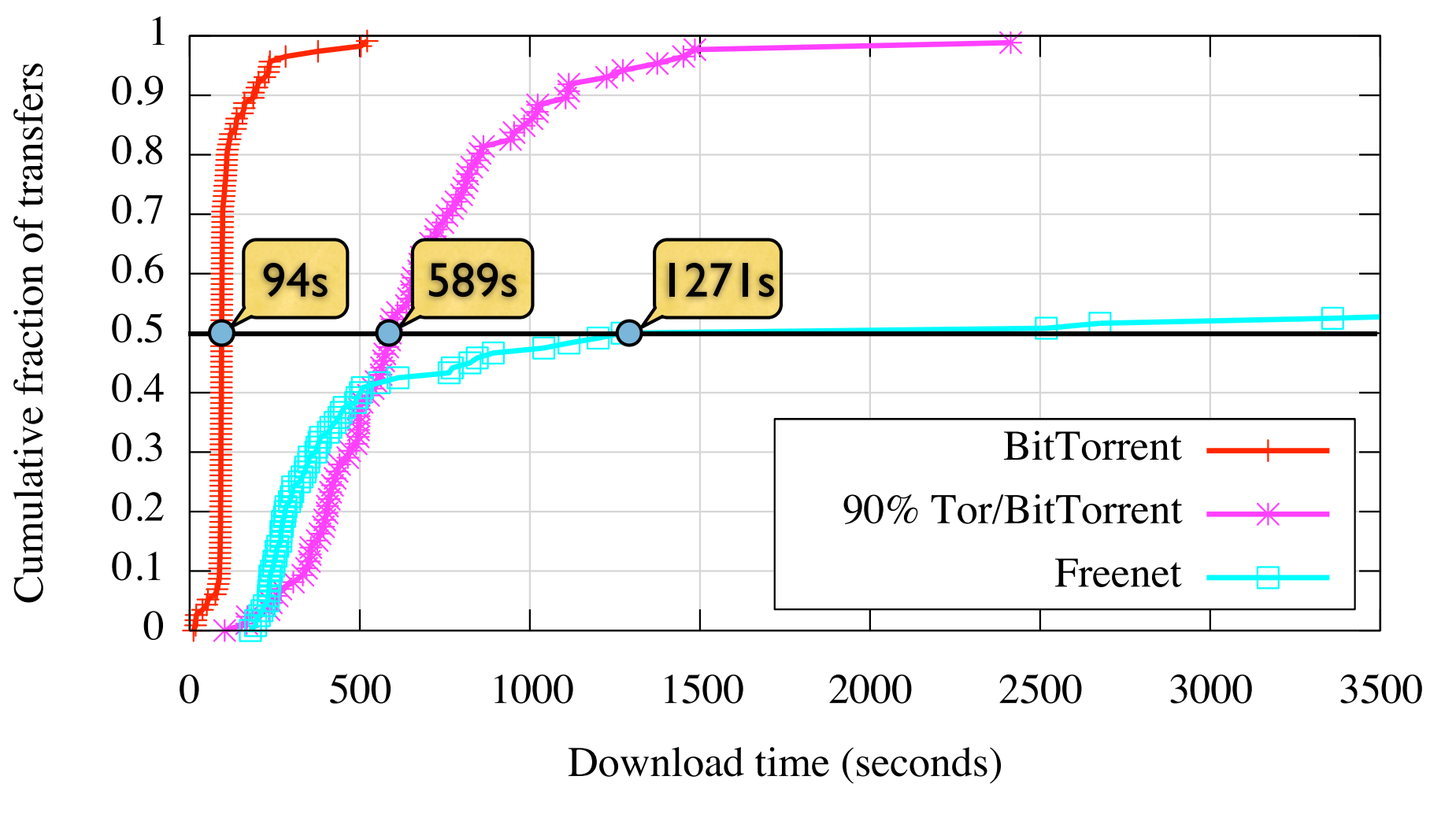
Existing systems



Existing systems



Existing systems



Performance cost for privacy is significant!

OneSwarm

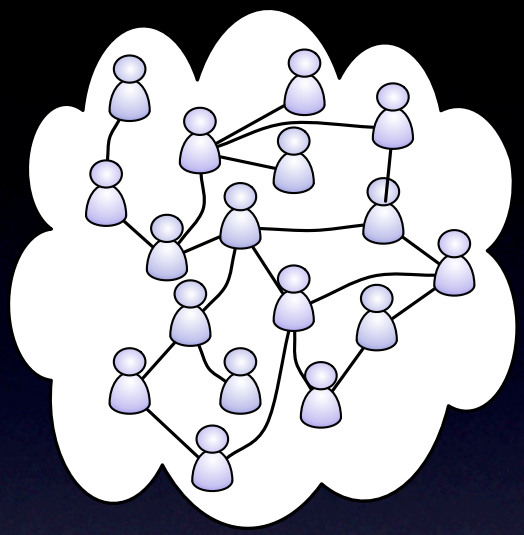
Goal: *Preserve user privacy* while maintaining *high performance*

OneSwarm

Goal: *Preserve user privacy* while maintaining *high performance*

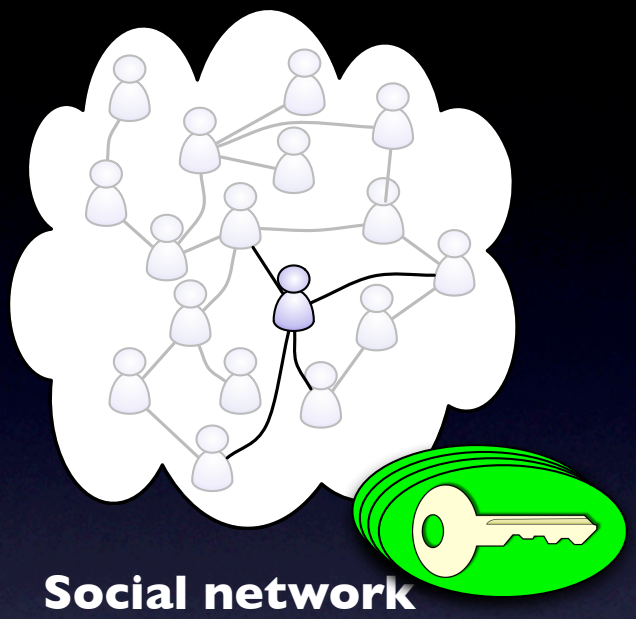
- Client *released* in March 2009
- Autoupdate requests from *hundreds of thousands of unique users*
- Flexible protocol gives users control of *privacy/performance* tradeoff

OneSwarm sketch

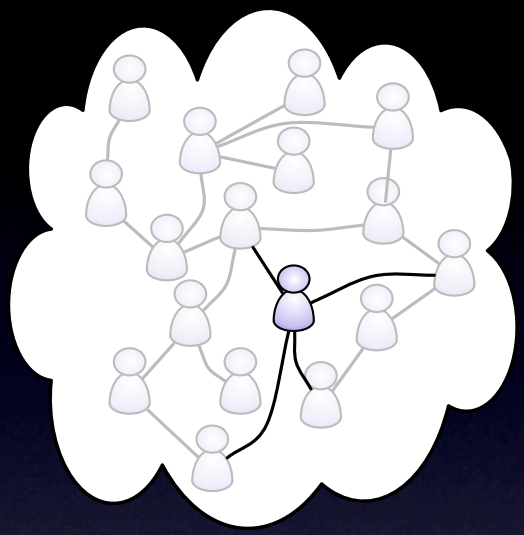


Social network

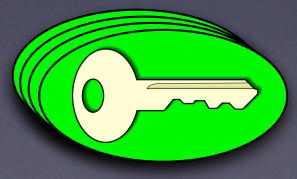
OneSwarm sketch



OneSwarm sketch

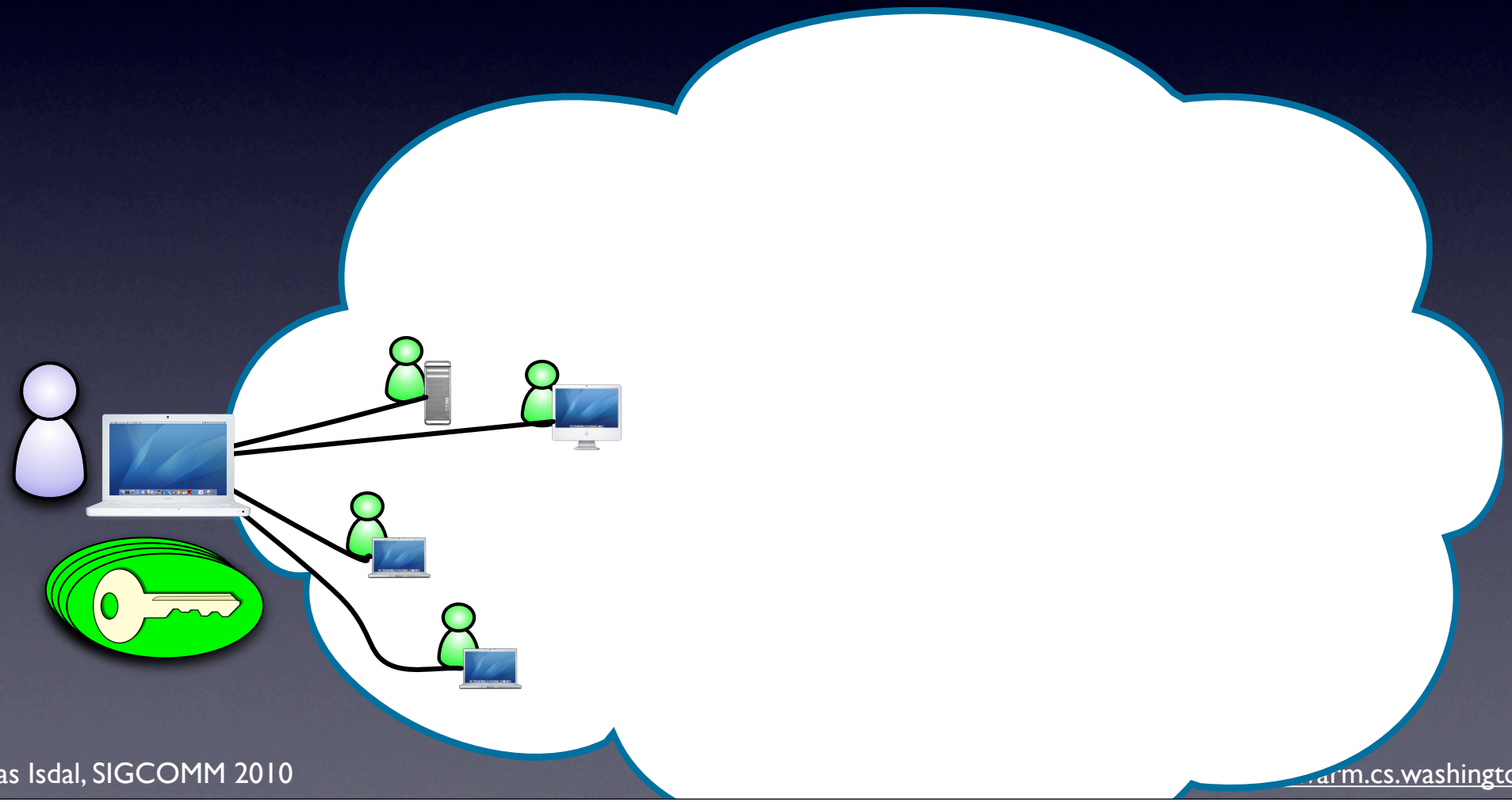


Social network



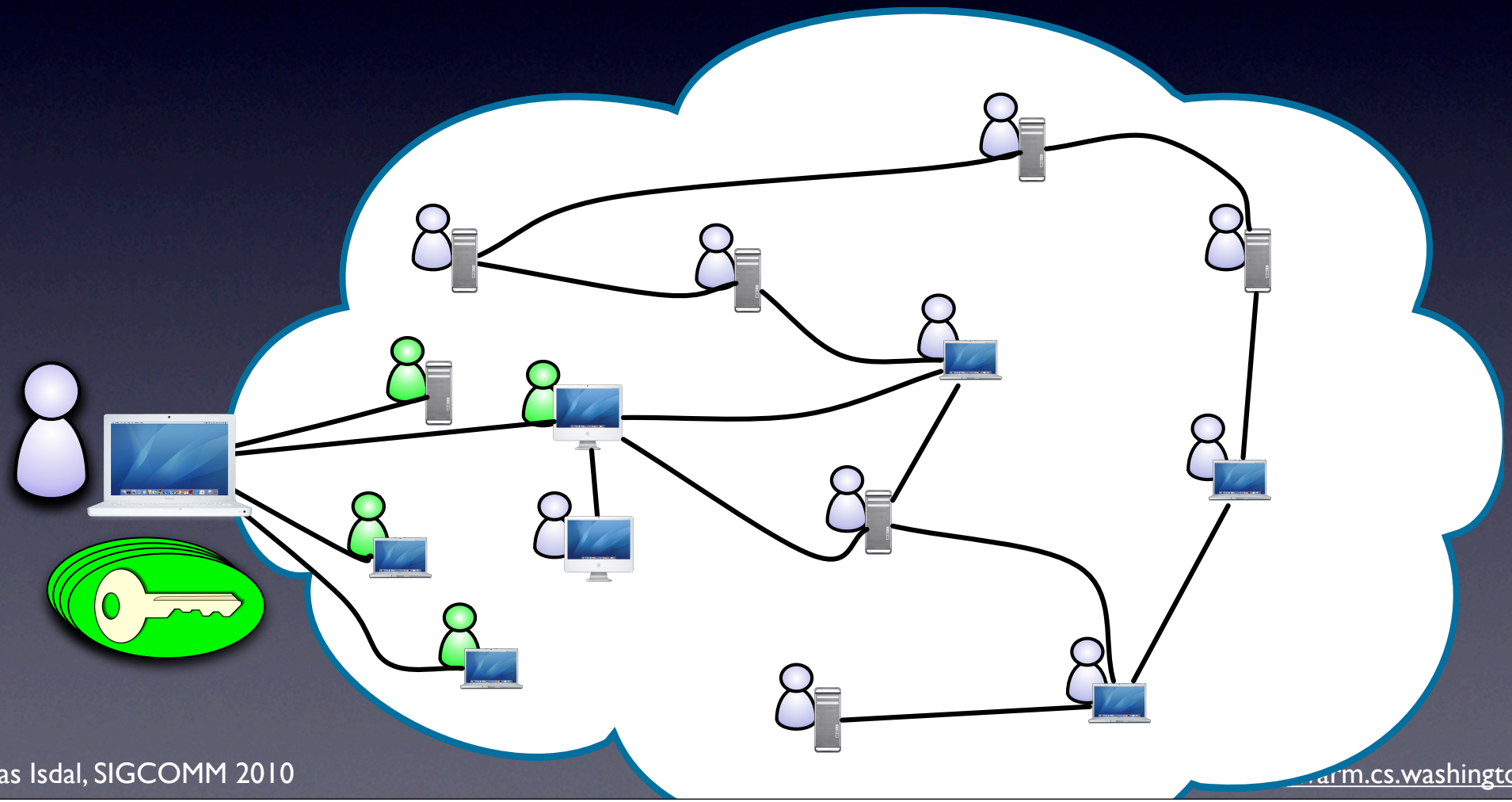
OneSwarm sketch

1. Import keys
2. Connect



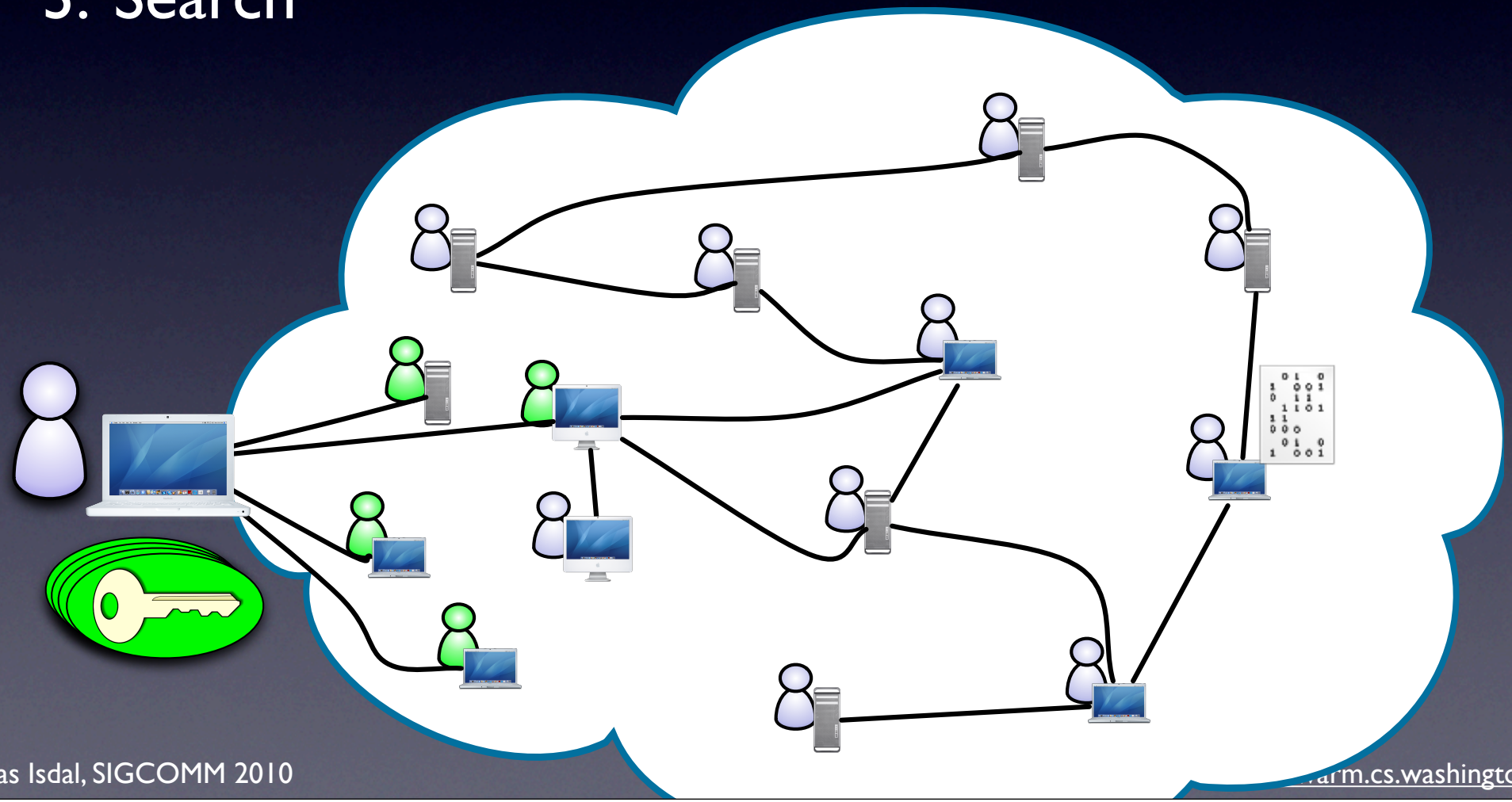
OneSwarm sketch

1. Import keys
2. Connect



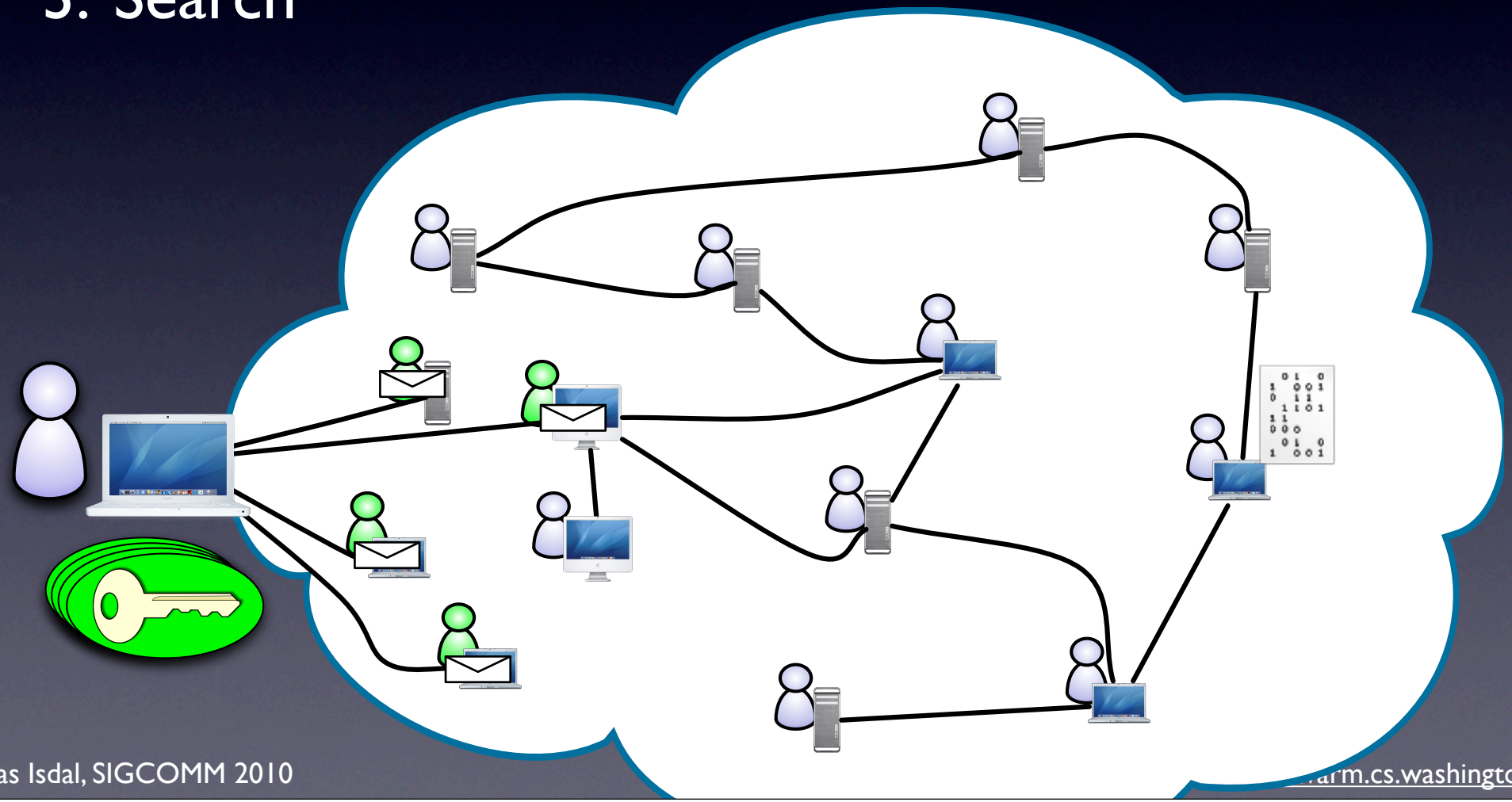
OneSwarm sketch

1. Import keys
2. Connect
3. Search



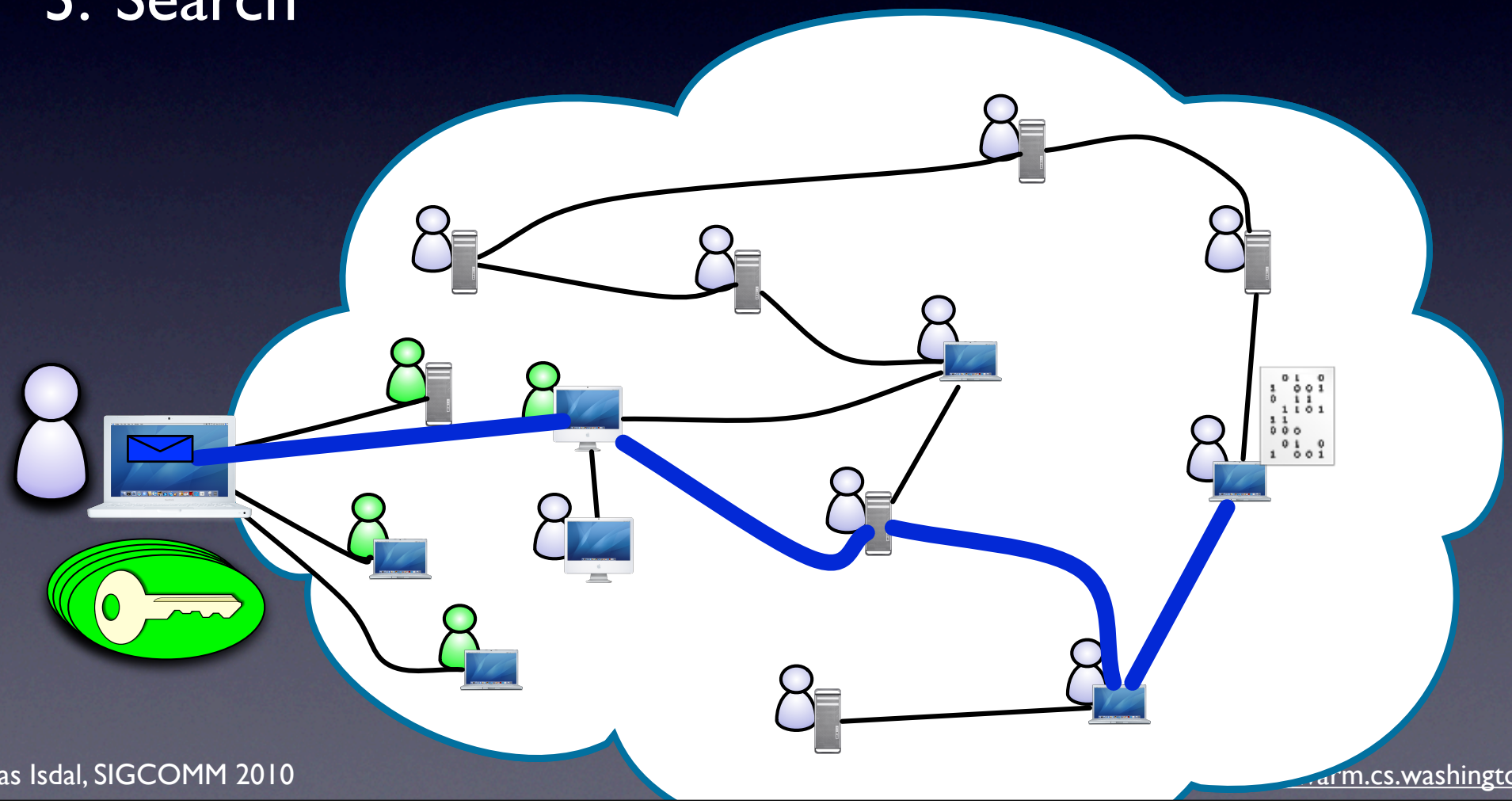
OneSwarm sketch

1. Import keys
2. Connect
3. Search



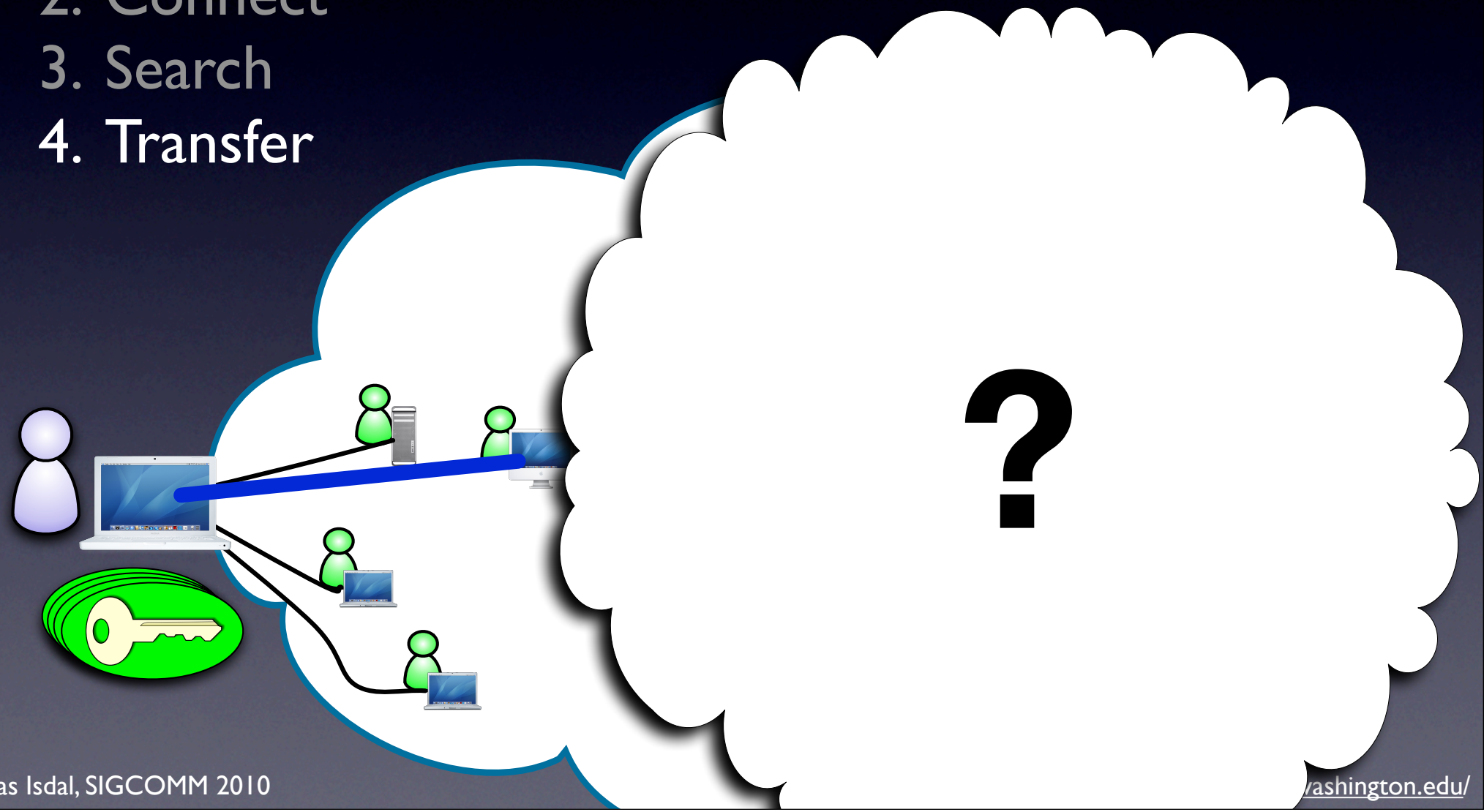
OneSwarm sketch

1. Import keys
2. Connect
3. Search



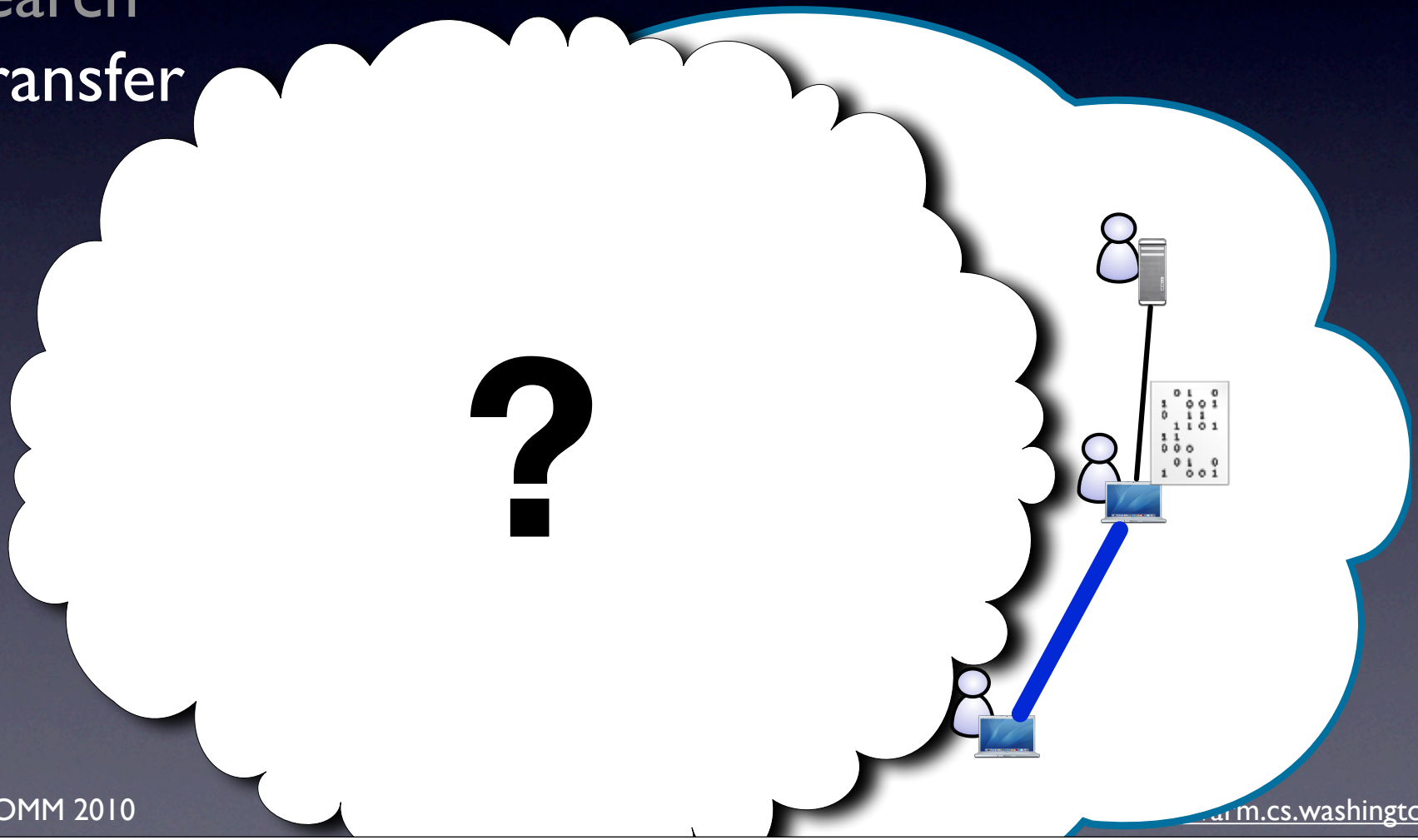
Receiver view

1. Import keys
2. Connect
3. Search
4. Transfer



Sender view

1. Import keys
2. Connect
3. Search
4. Transfer



Outline

- Protocol sketch
- *Threat model*
- Design challenges
- Evaluation

Threat model

OneSwarm goal: Protect the users from revealing who is sharing / downloading which items

Threat model

OneSwarm goal: Protect the users from revealing who is sharing / downloading which items

- *The Attacker:*
  - Controls a *limited number* of overlay nodes
  - Can do *anything* on nodes it controls
    - inject/sniff traffic, correlate data
- *Cannot:* Sniff / inject traffic into *arbitrary network links*

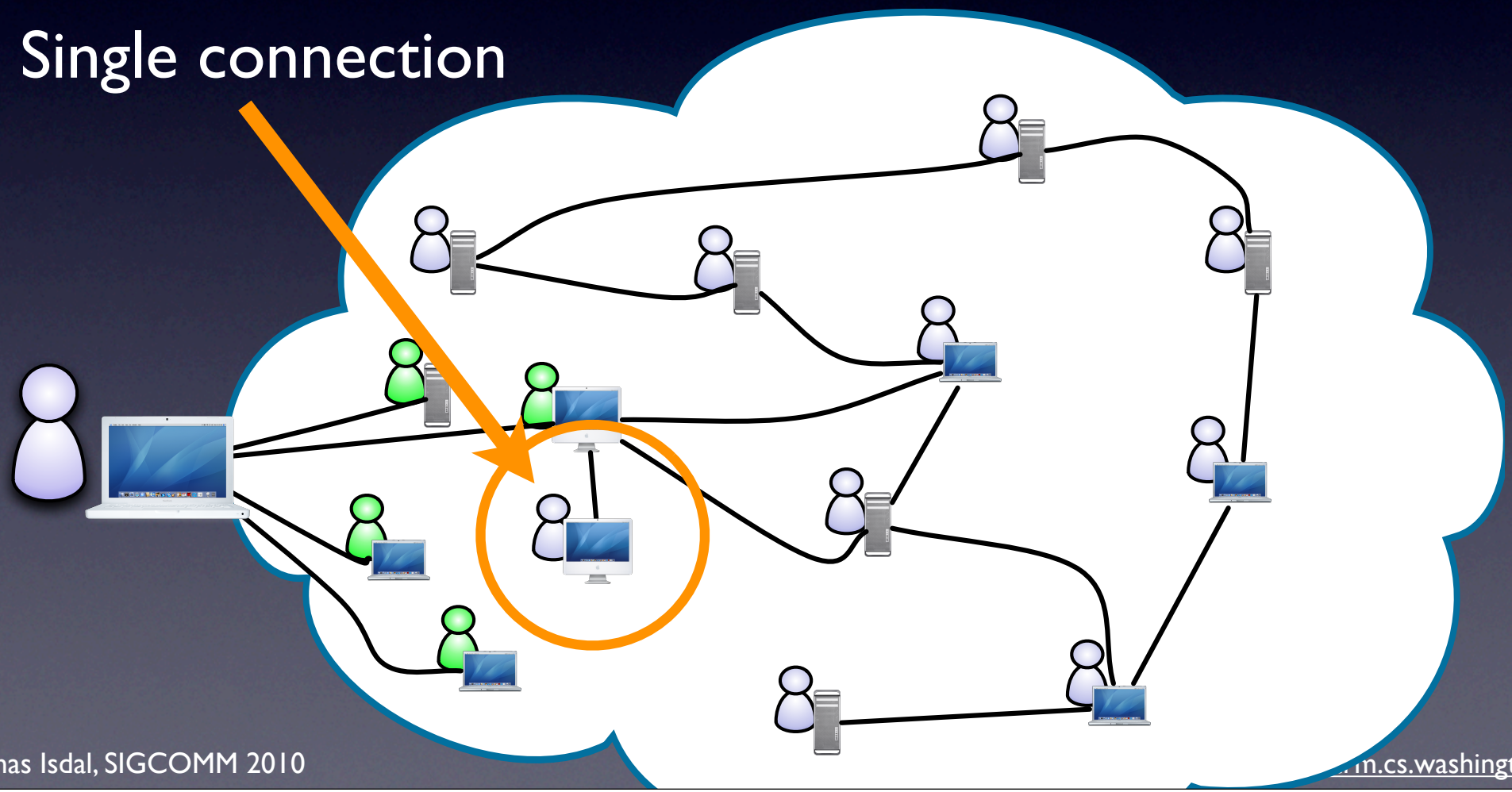
Design challenges

- *Locating mobile peers*
- Robustness despite sparse social networks
- Search without revealing src and dst
- Maintain performance despite long paths

Mobile peers

- Peers publish IP:port in DHT
- *Allow revocation* of location updates:
  - Publish IP:port *separately* for each peer
  - Encrypt and sign updates
  - Removing a peer:
    - ➡ Disallow incoming connections
    - ➡ *Stop location updates* for that peer

Sparse social networks



Relying on social networks

- Many users have a *sparse social network*
  - ➡ Poor reliability
  - ➡ Poor performance
  - ➡ Privacy issues
- Early adopters suffer

Early adopters

http://oneswarm.cs.washington.edu/

Early adopters

Solution: *Community Servers*

- Source of *untrusted* peers

http://oneswarm.cs.washington.edu/

Untrusted peers

- *Long term* relationships with *few* untrusted peers
- *Delay* responses to foil timing attacks
- *Probabilistic forwarding* of queries
- *Deterministic* random behavior to limit information leakage from repeated queries

Untrusted peers

- *Long term* relationships with *few* untrusted peers
- *Delay* responses to foil timing attacks
- *Probabilistic forwarding* of queries
- *Deterministic* random behavior to limit information leakage from repeated queries

Peers can be *trusted* / *untrusted* per data object

Design challenges

- Locating mobile peers
- Robustness despite sparse social networks
- *Search without revealing src and dst*
- Maintain performance despite long paths

Flood with delay

Goal: locate data without revealing source or destination

- *Flood* searches into the overlay
- Only forward searches using *spare capacity*
- Searches are *delayed* at each hop
  - ➡ Require users send *cancel* message on incoming response to *stop flood* propagation

Flood with delay

Goal: locate data without revealing source or destination

- *Flood* searches into the overlay
- Only forward searches using *spare capacity*
- Searches are *delayed* at each hop
  - ➡ Require users send *cancel* message on incoming response to *stop flood* propagation

Property: No guarantee that all objects can be found by all users at all times

Response delay

- *Hash*: 160bit content hash
- *Text*: return items matching "britney s"
- *Delay response* towards *untrusted* peers:
  - Random reply delay *seeded by hash*
  - *Multiple replies* if multiple matches

Design challenges

- Locating mobile peers
- *Robustness despite sparse social networks*
- *Search without revealing src and dst*
- *Maintain performance despite long paths*

Multi-path

- End-to-end capacity *limited by slowest link*
- Paths *fail if any node* on the path *disconnects*

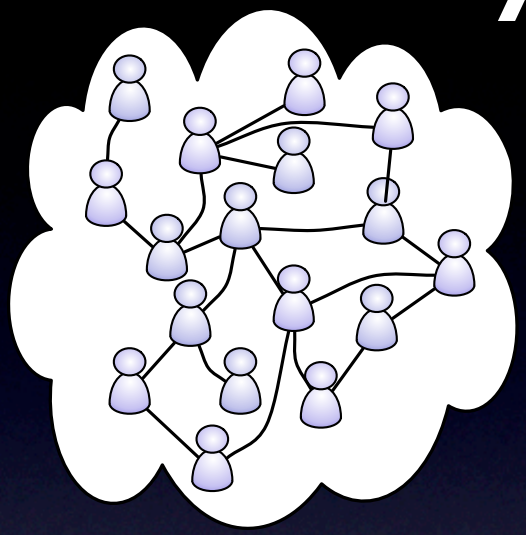
➡ *Use multiple paths to the same source, send cancel after "n" responses (default=20)*

- Eliminate "*find fastest path*" problem
- Distribute forwarding load
- Resilience to path failures

Data transfer

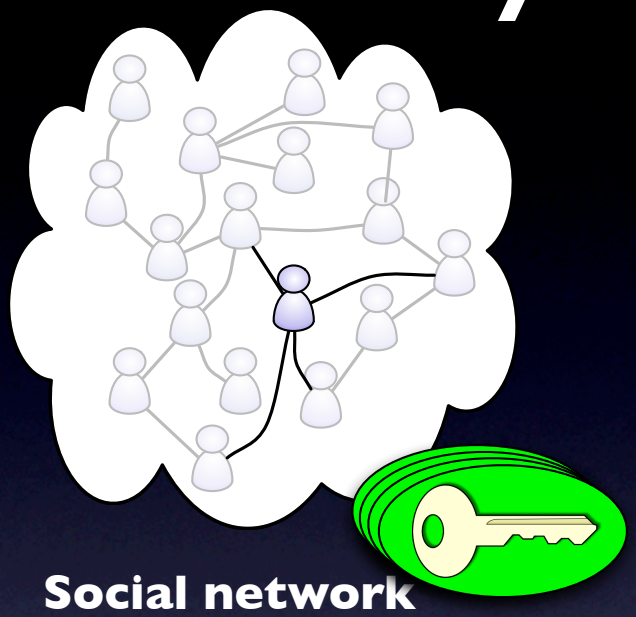
- BitTorrent used in the overlay as basis for the data transfer protocol
  - *Multiple sources* handled naturally
  - *Peer* in BitTorrent→*Path* in OneSwarm
  - *Swarming* downloads

System overview

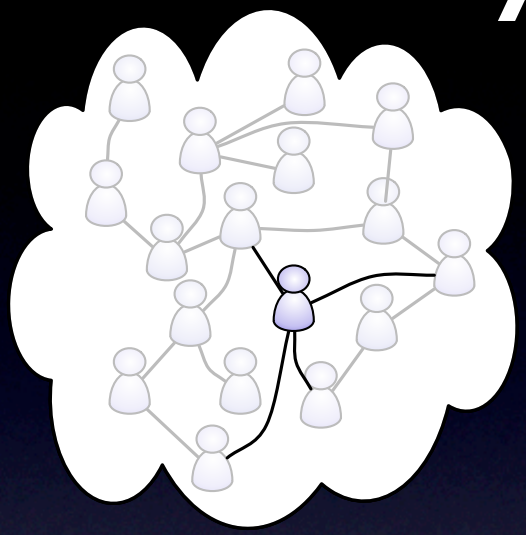


Social network

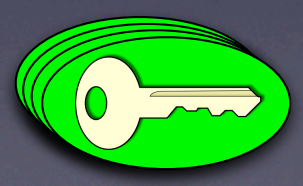
System overview



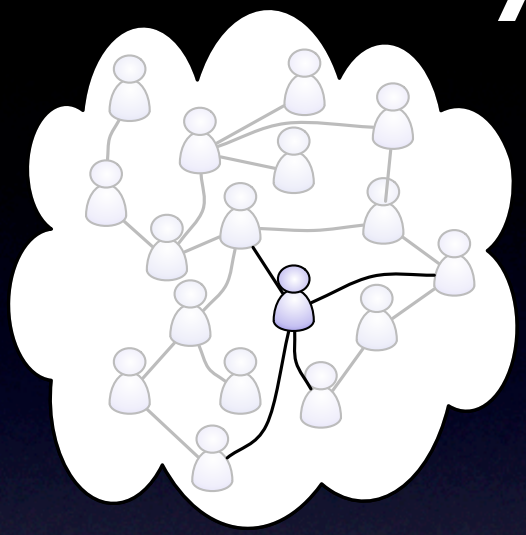
System overview



Social network

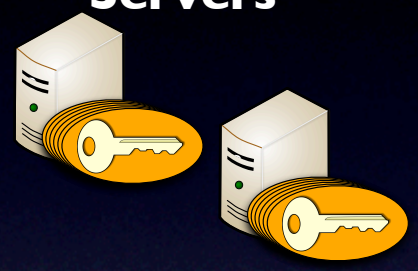


System overview

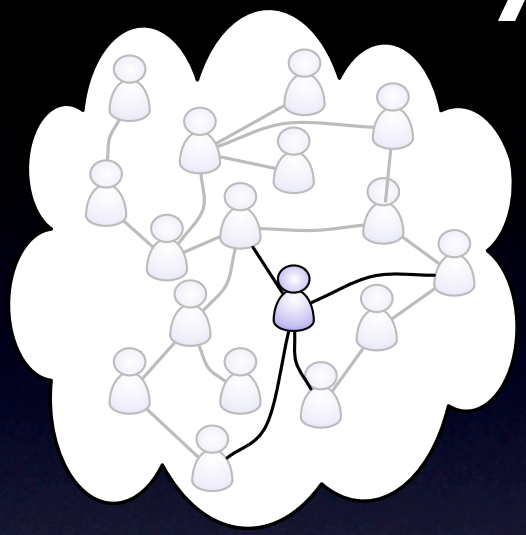


Social network

Community Servers

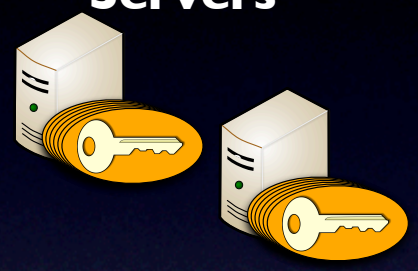


System overview



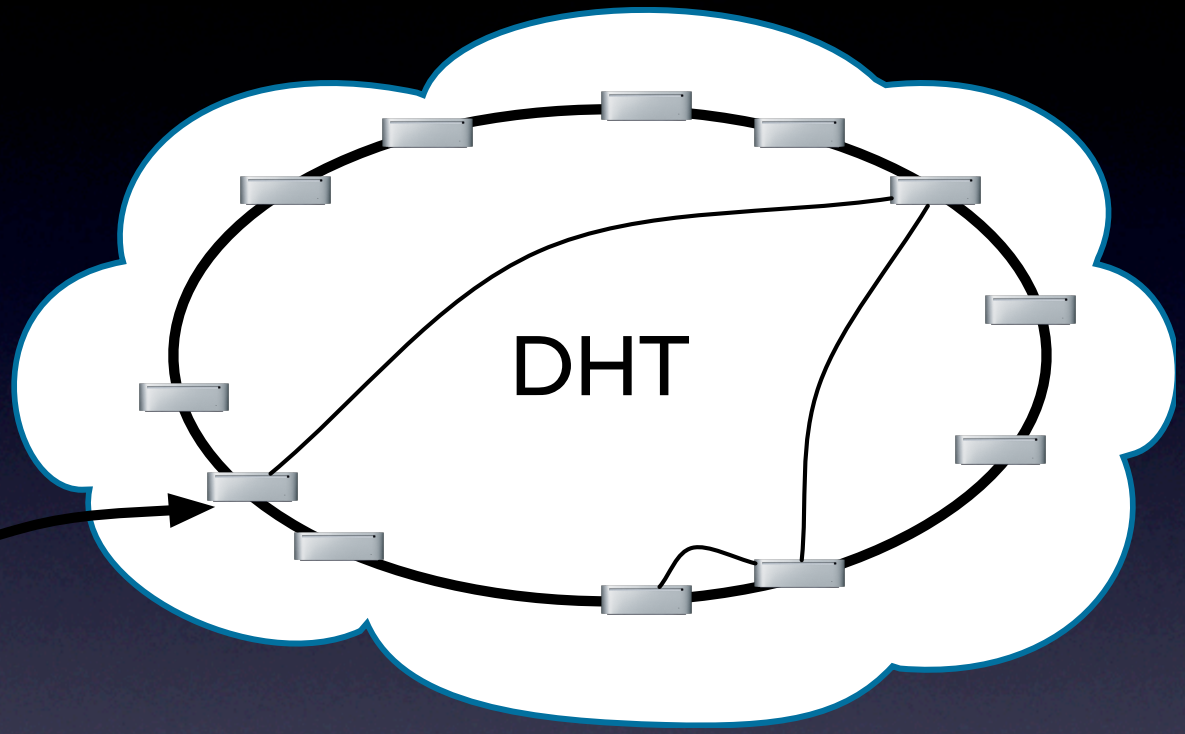
Social network

Community Servers



System overview

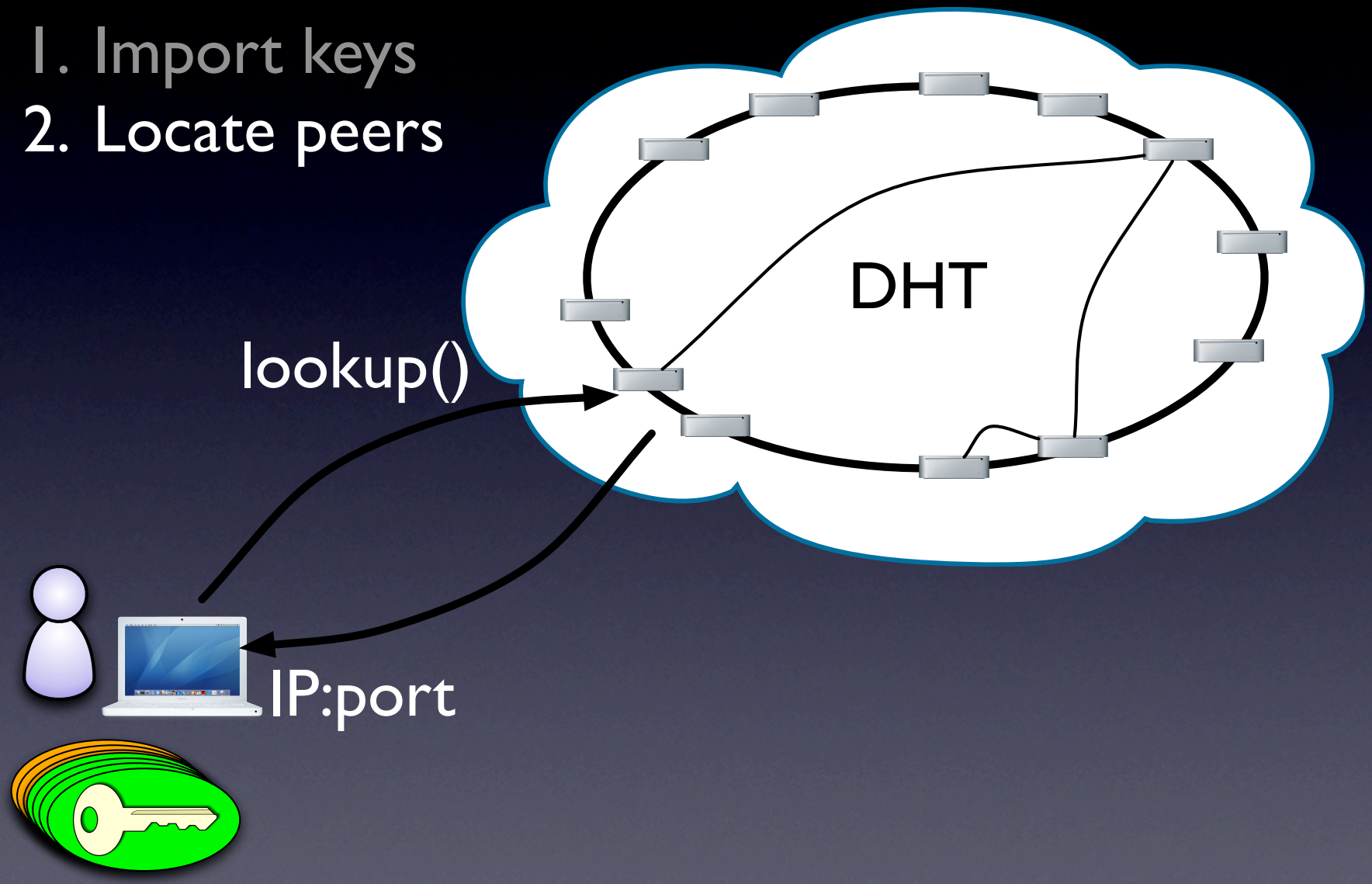
1. Import keys
2. Locate peers



lookup()

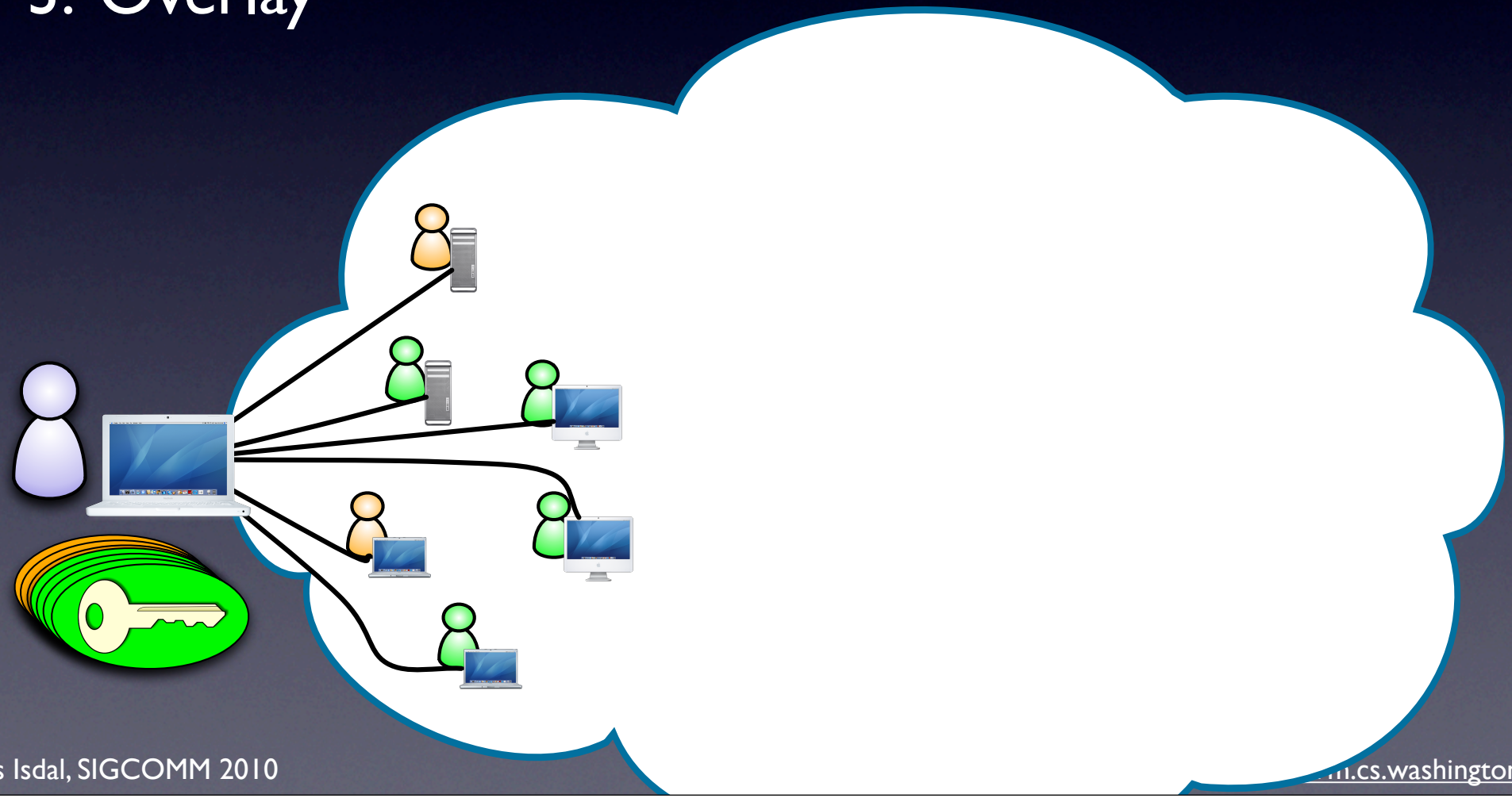
System overview

1. Import keys
2. Locate peers



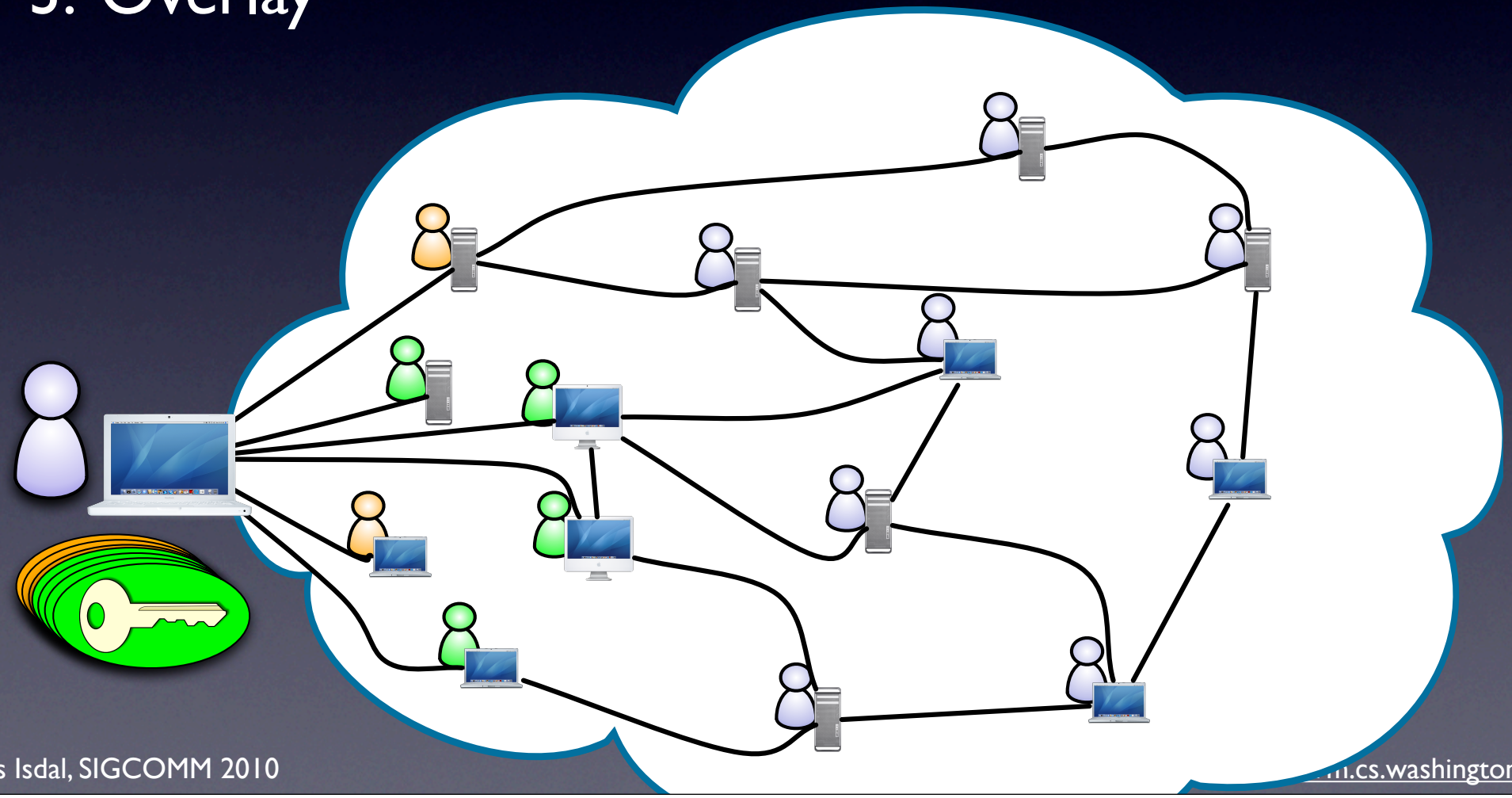
System overview

1. Import keys
2. Locate peers
3. Overlay



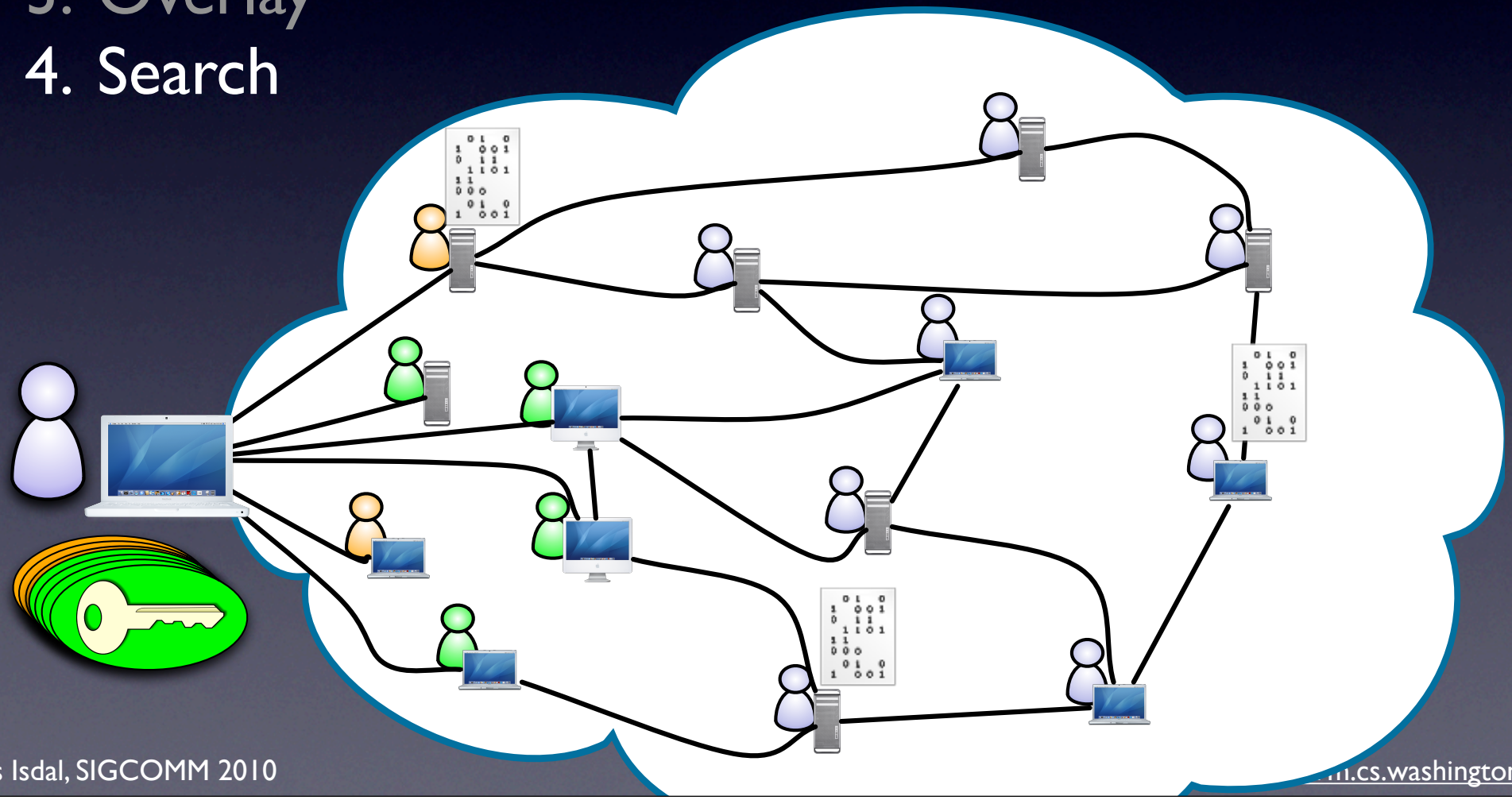
System overview

1. Import keys
2. Locate peers
3. Overlay



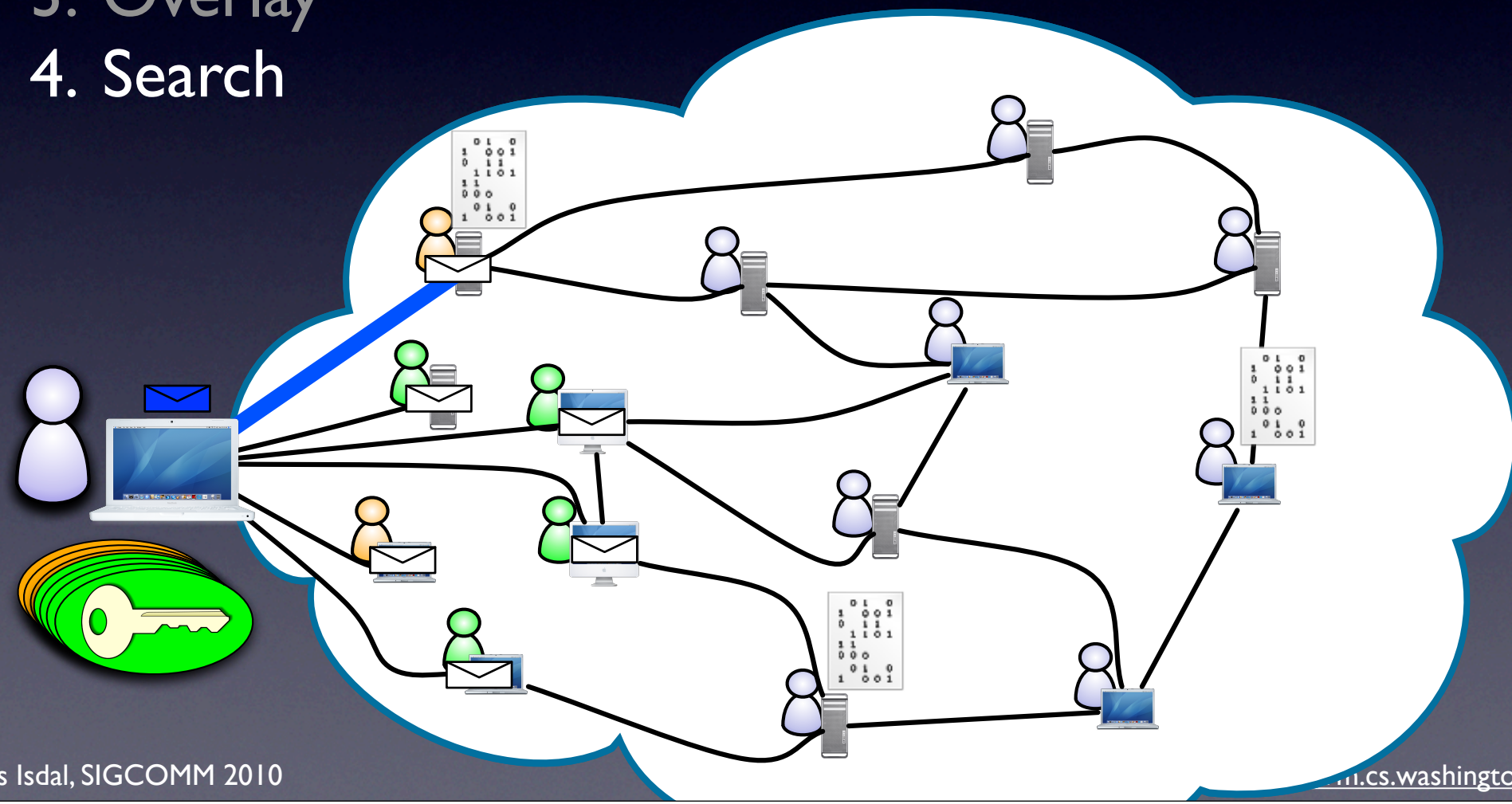
System overview

1. Import keys
2. Locate peers
3. Overlay
4. Search



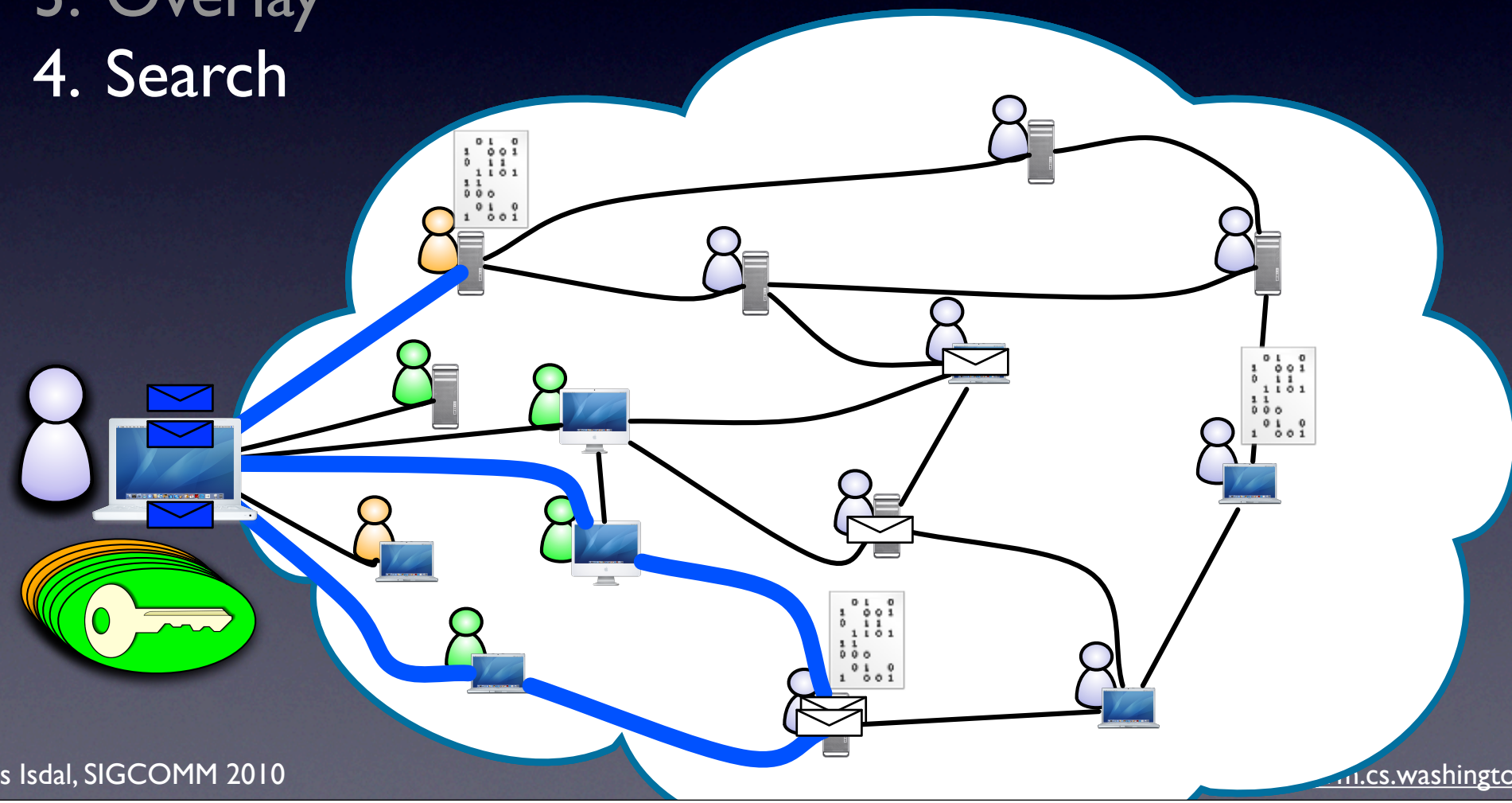
System overview

1. Import keys
2. Locate peers
3. Overlay
4. Search



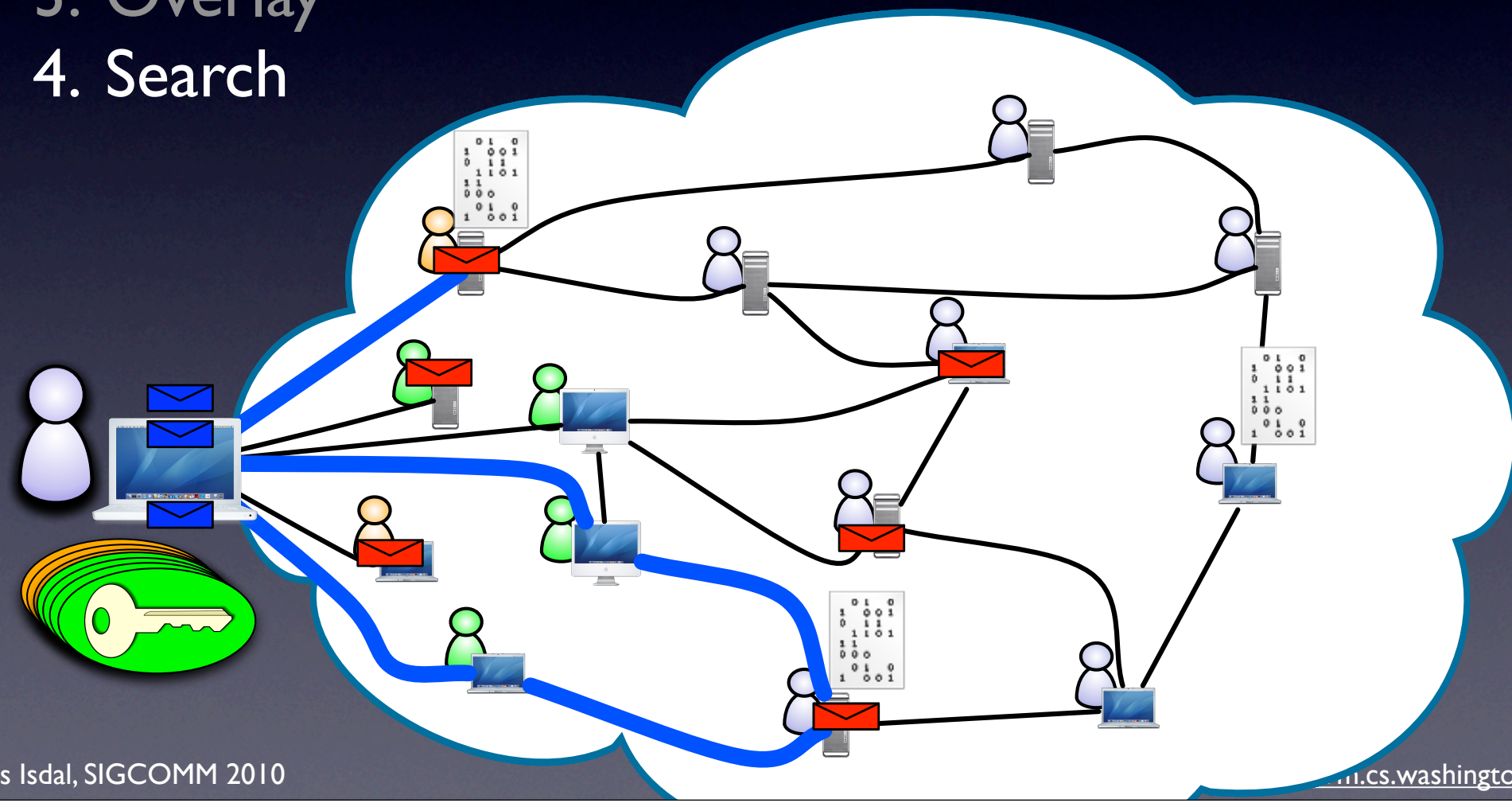
System overview

1. Import keys
2. Locate peers
3. Overlay
4. Search



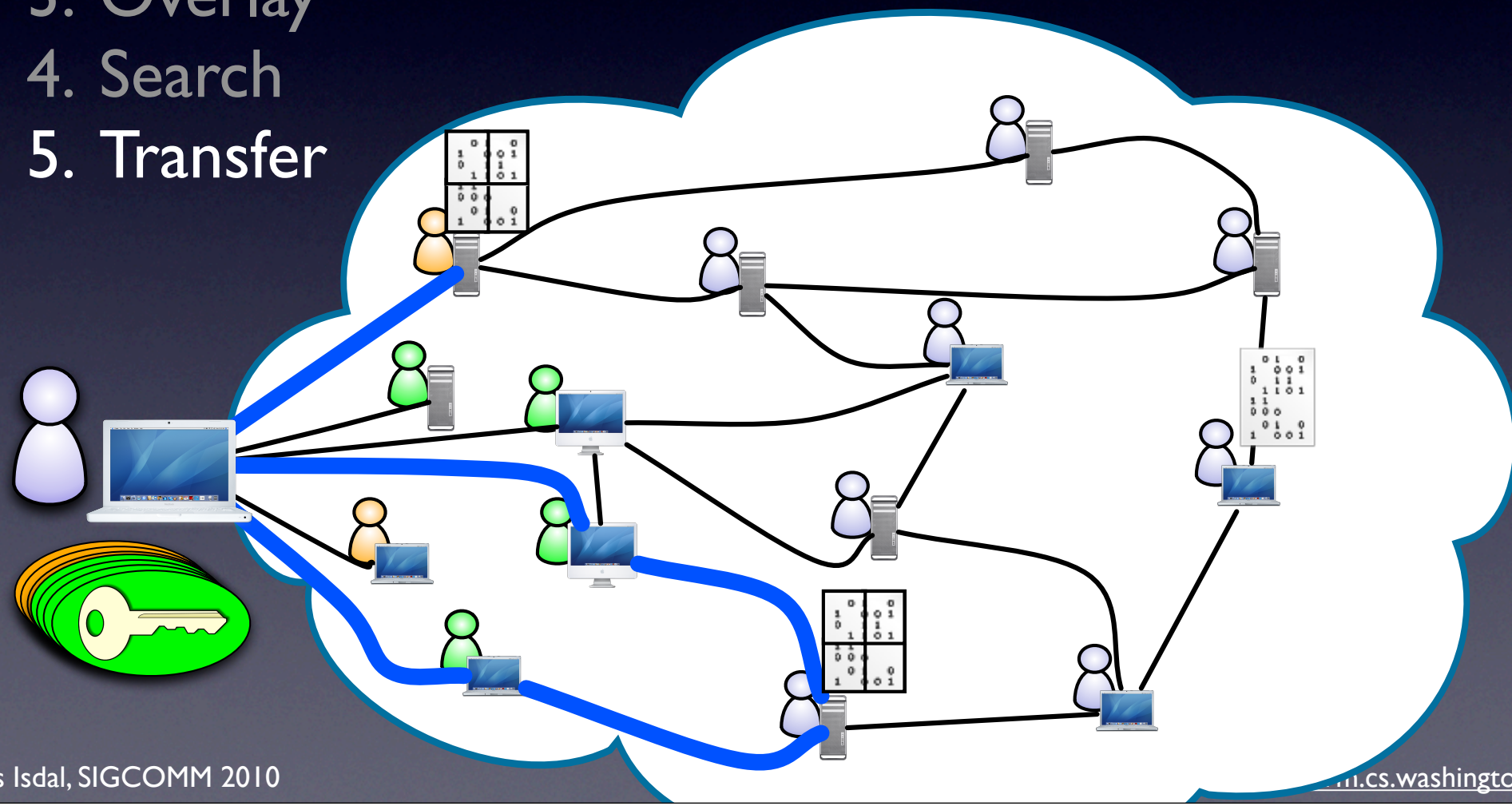
System overview

1. Import keys
2. Locate peers
3. Overlay
4. Search



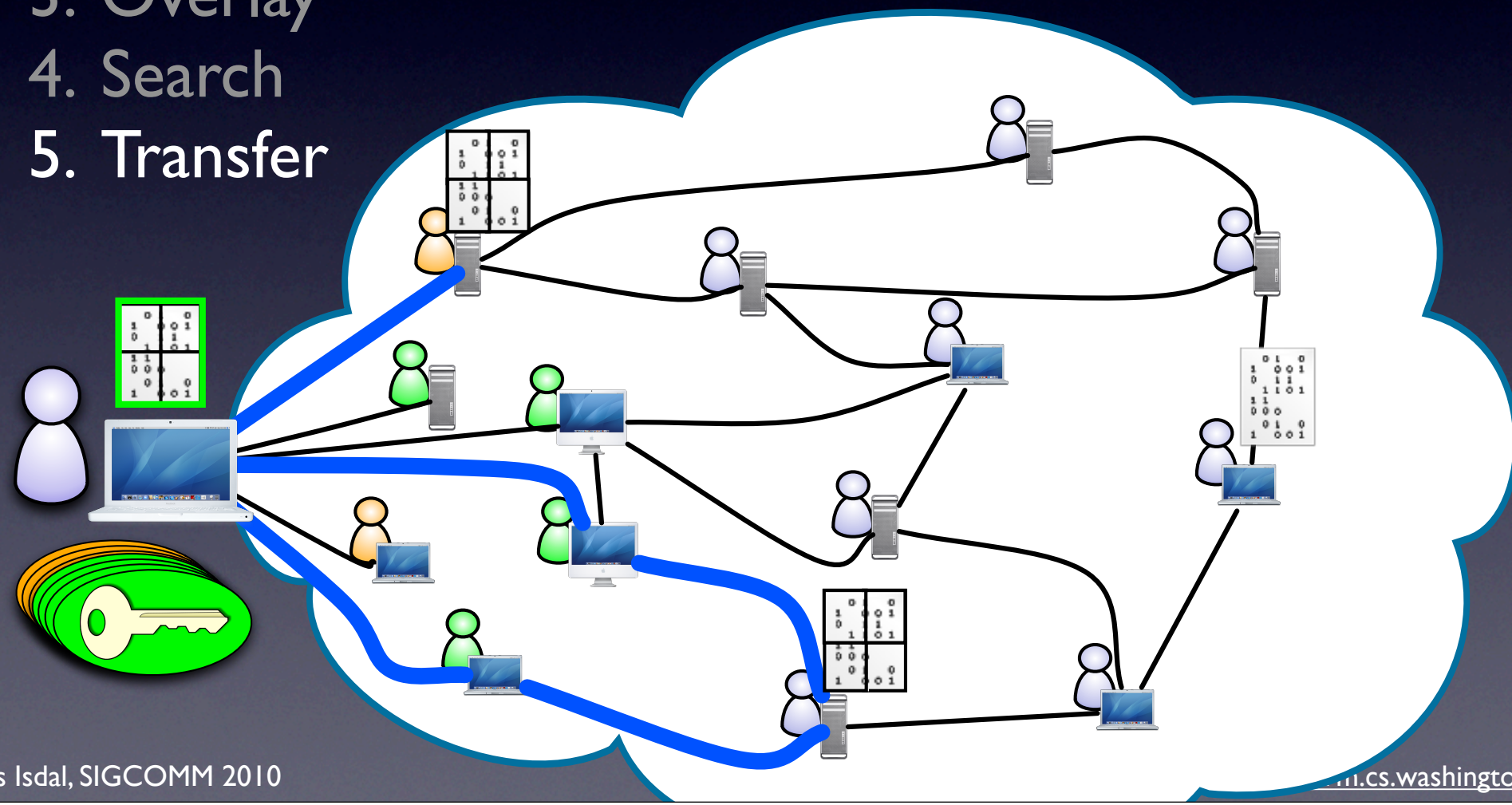
System overview

1. Import keys
2. Locate peers
3. Overlay
4. Search
5. Transfer



System overview

1. Import keys
2. Locate peers
3. Overlay
4. Search
5. Transfer



Outline

- Protocol sketch
- Threat model
- Design challenges
- *Evaluation*

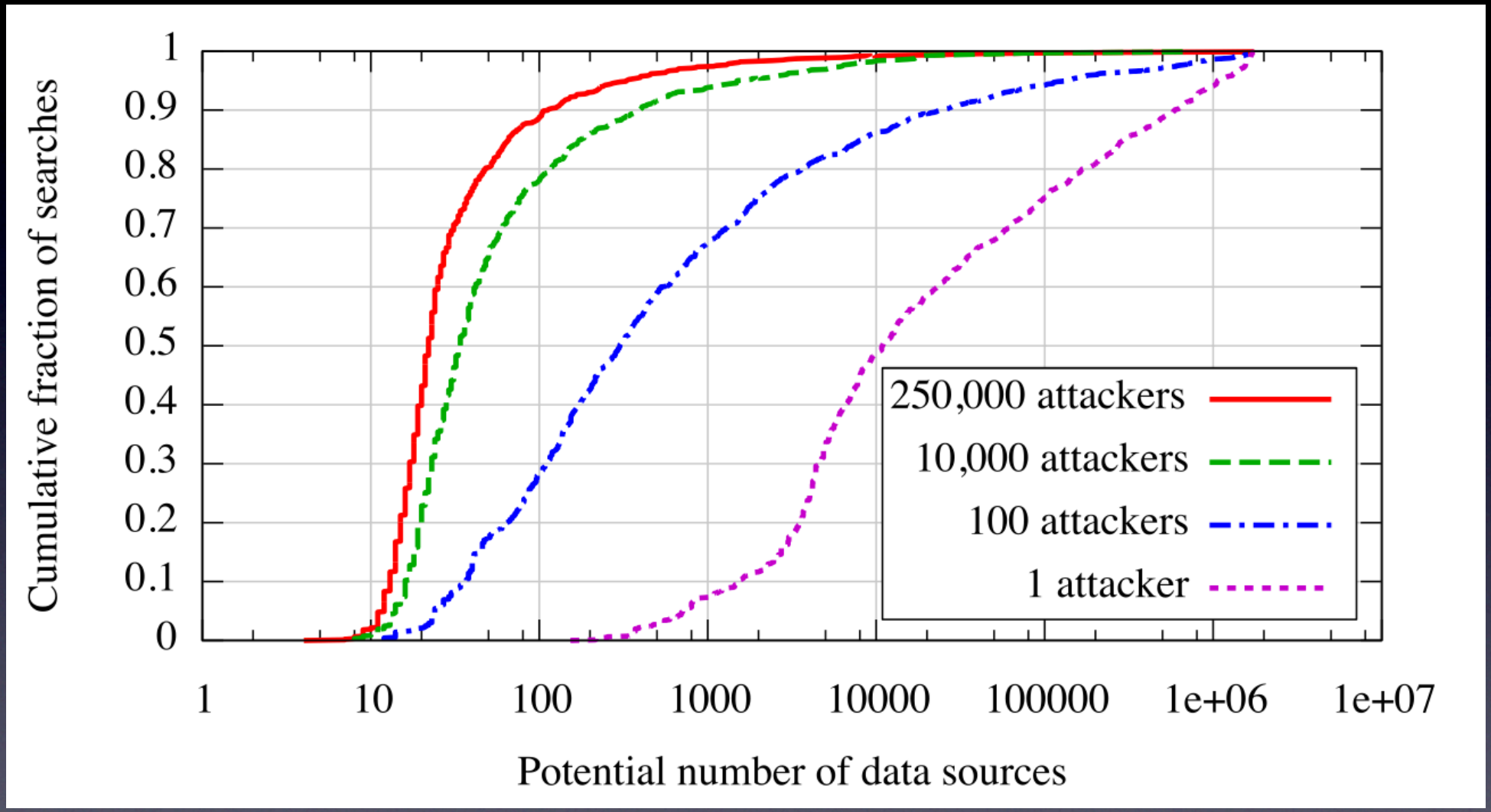
Timing attack

- Combination of social graph and untrusted peers
- Attackers search for content in the overlay
  - *Monitor delay* between search and response
- How many users *could* have sent the reply within the recorded time?
- To quantify attacker effectiveness we use:
  - Connectivity oracle
  - Link latency oracle

Timing attack setup

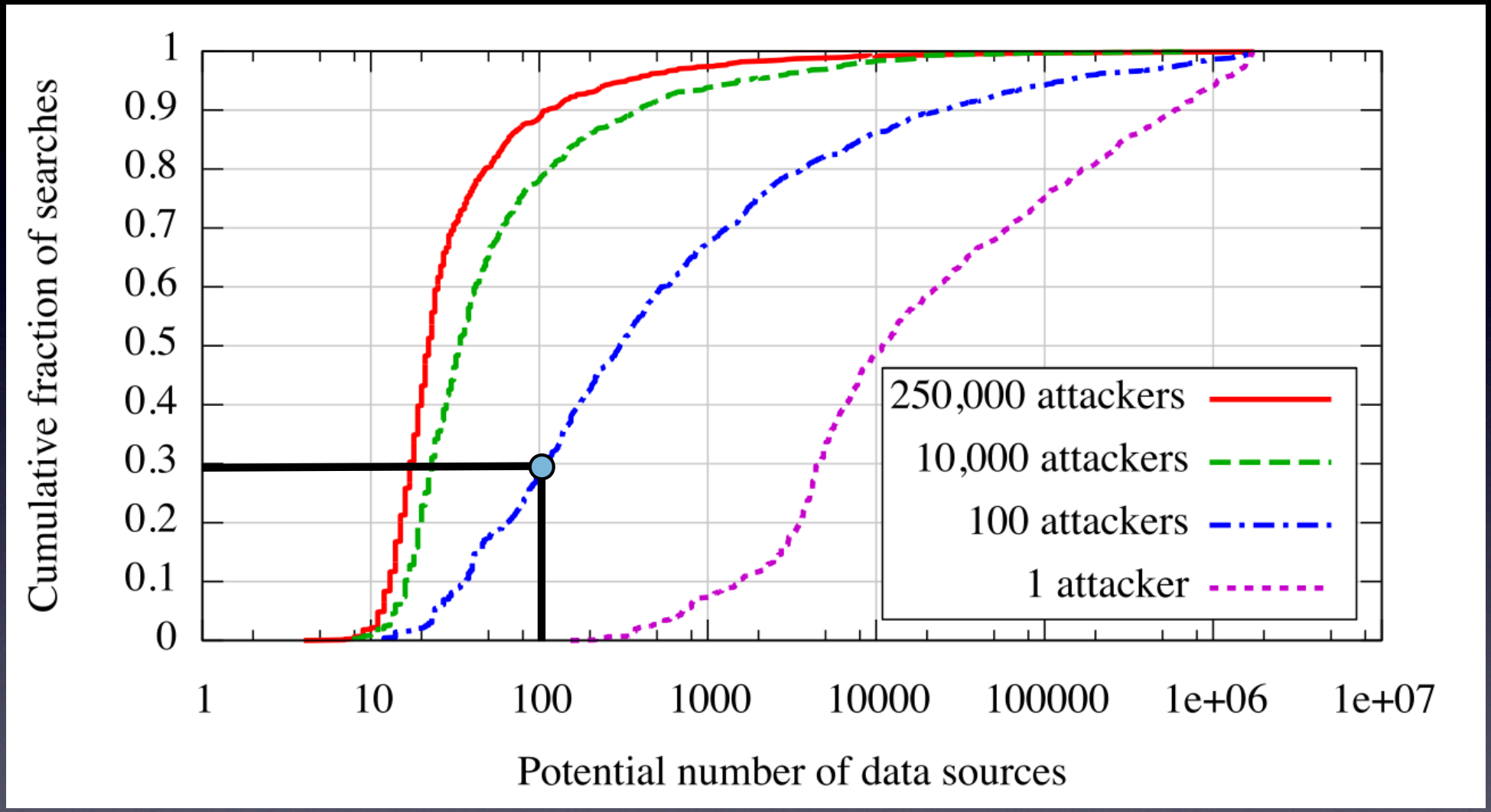
- *Synthetic* OneSwarm overlay
- Based on social graph of *1.7 million* last.fm users
- Attackers are connected using a *public community server*
- Last.fm users with less than 26 friends use same community server to augment their connectivity graph

Timing uncertainty



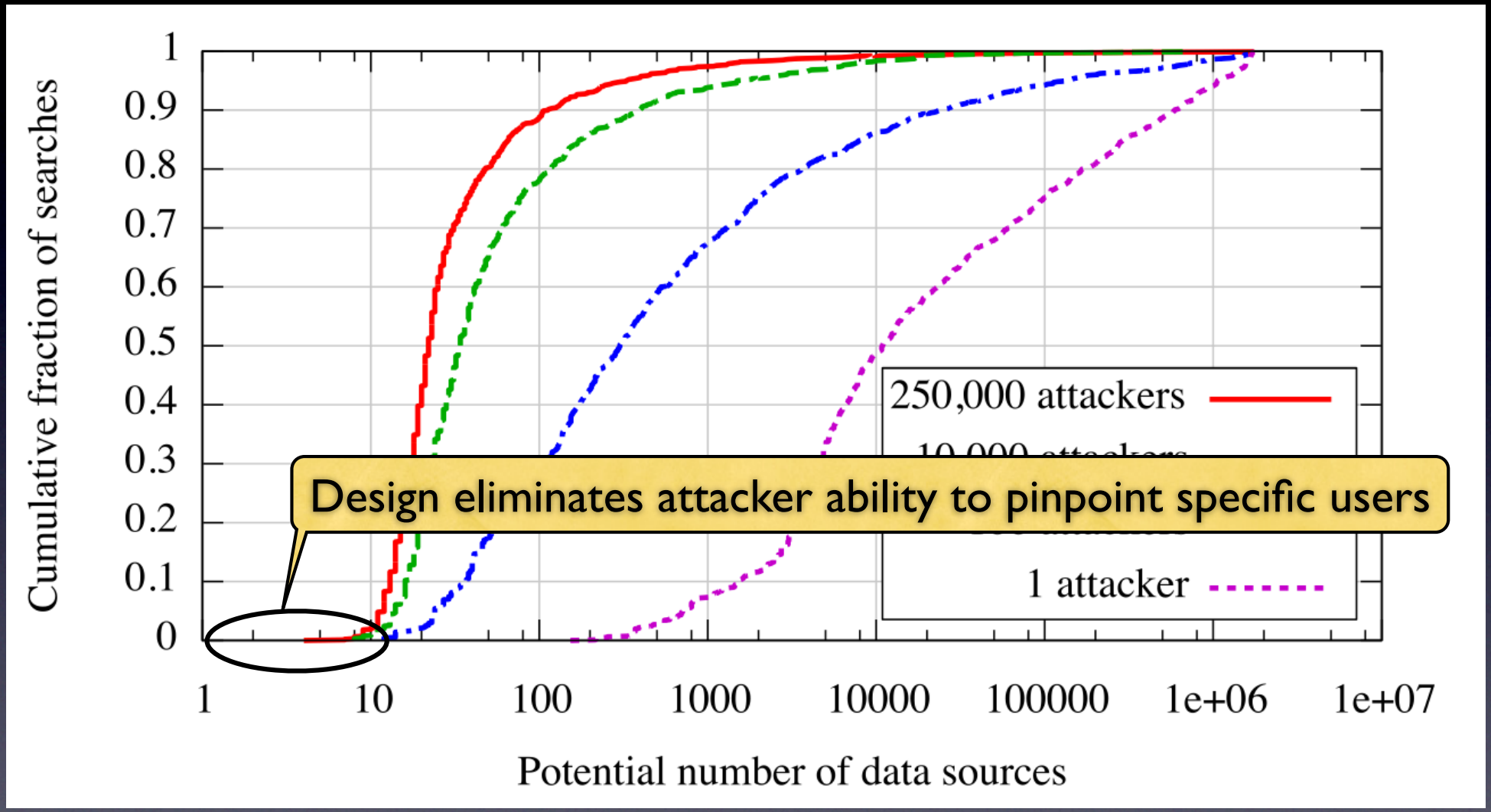
Combination of trusted/untrusted adds uncertainty

Timing uncertainty



Combination of trusted/untrusted adds uncertainty

Timing uncertainty



Combination of trusted/untrusted adds uncertainty

Performance

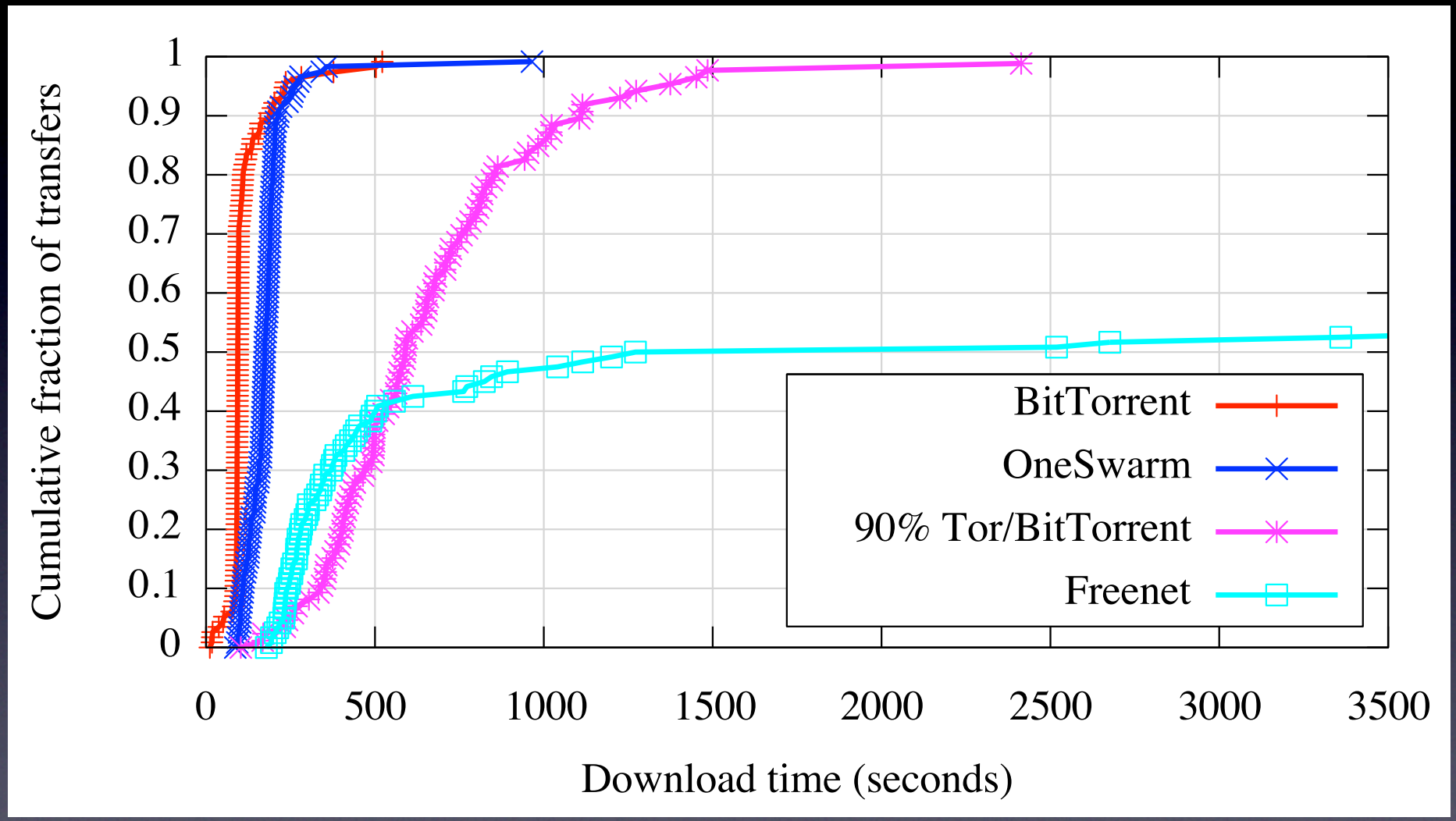
- *Comparison* with existing systems
  - OneSwarm (using community server)
  - BitTorrent
  - BitTorrent over TOR
  - Freenet
- 20MB file from 120 PlanetLab nodes

Performance

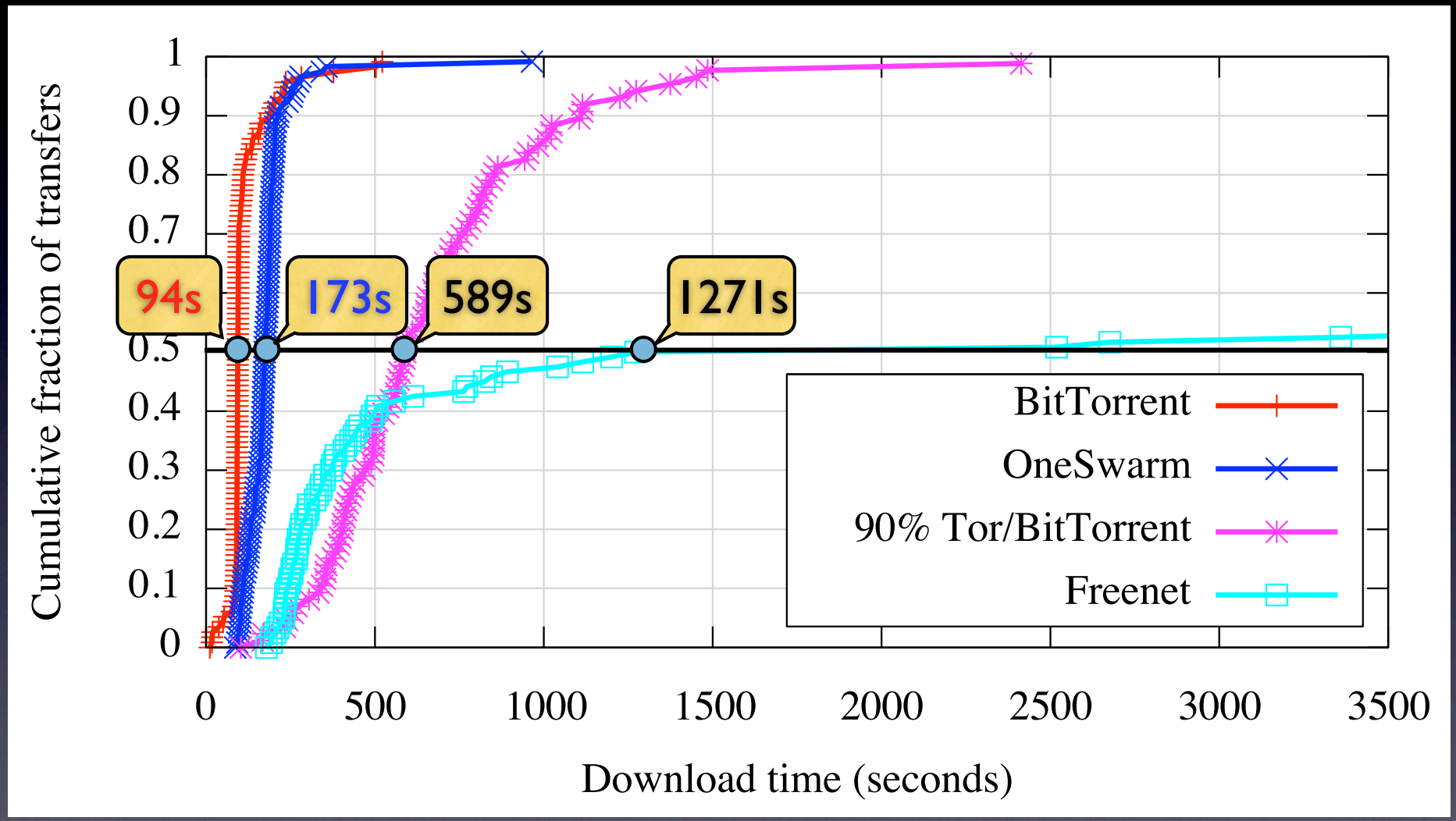
- *Comparison* with existing systems
  - OneSwarm (using community server)
  - BitTorrent
  - BitTorrent over TOR
  - Freenet
- 20MB file from 120 PlanetLab nodes

For OneSwarm: PlanetLab nodes part of public mesh, no two experiment nodes talk directly to each other

Download time

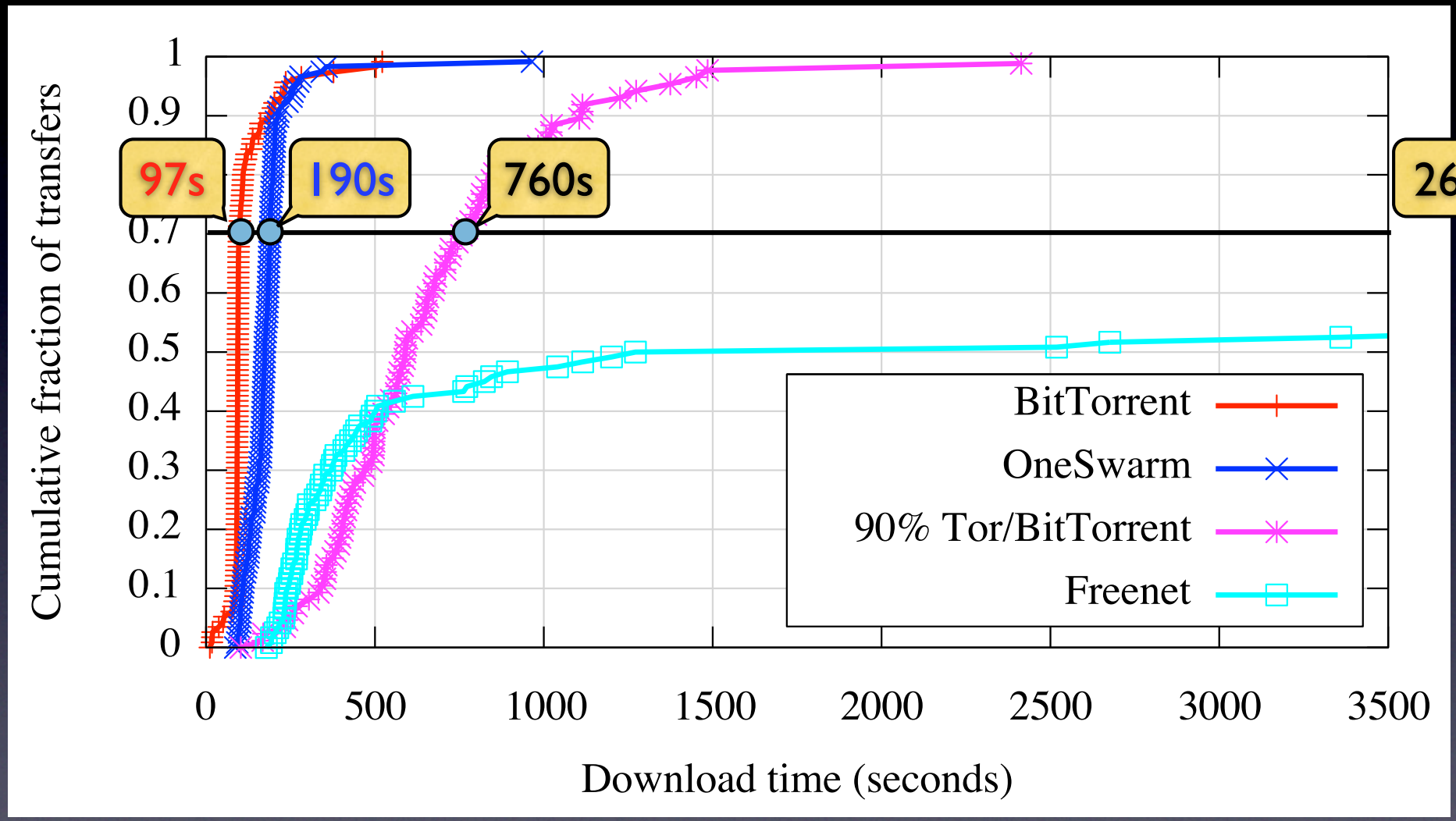


Download time



OneSwarm: 3.4 times faster than TOR, 6.9 times faster than Freenet

Download time

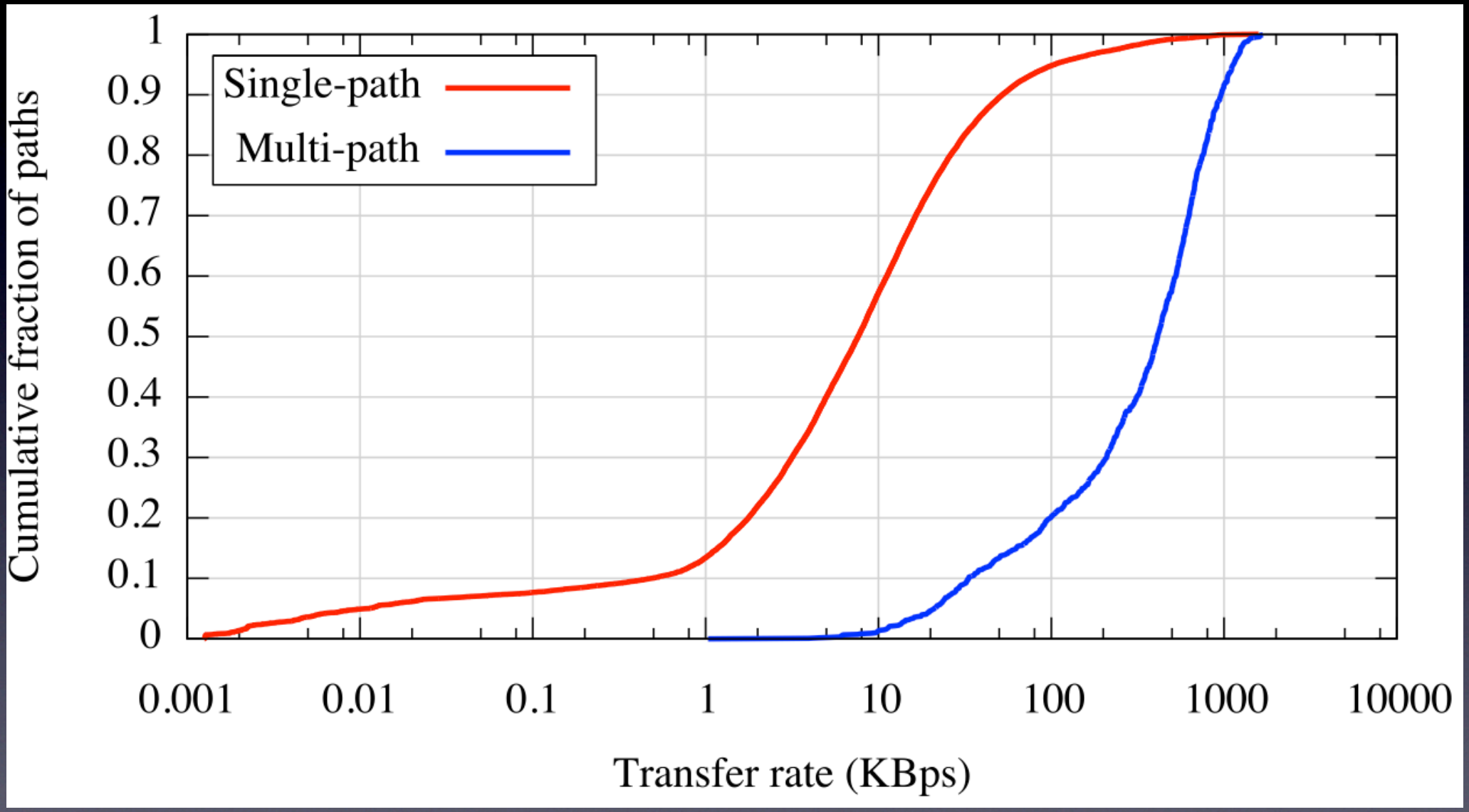


OneSwarm: 3.4 times faster than TOR, 6.9 times faster than Freenet

Multi-path

- 120 Planetlab nodes connected to the public OneSwarm mesh
- Compare throughput for:
  - *All paths* discovered
  - The *shortest path* (first response)

Multi-path transfers



Average: multipath: 457KB/s, shortest path: 29KB/s

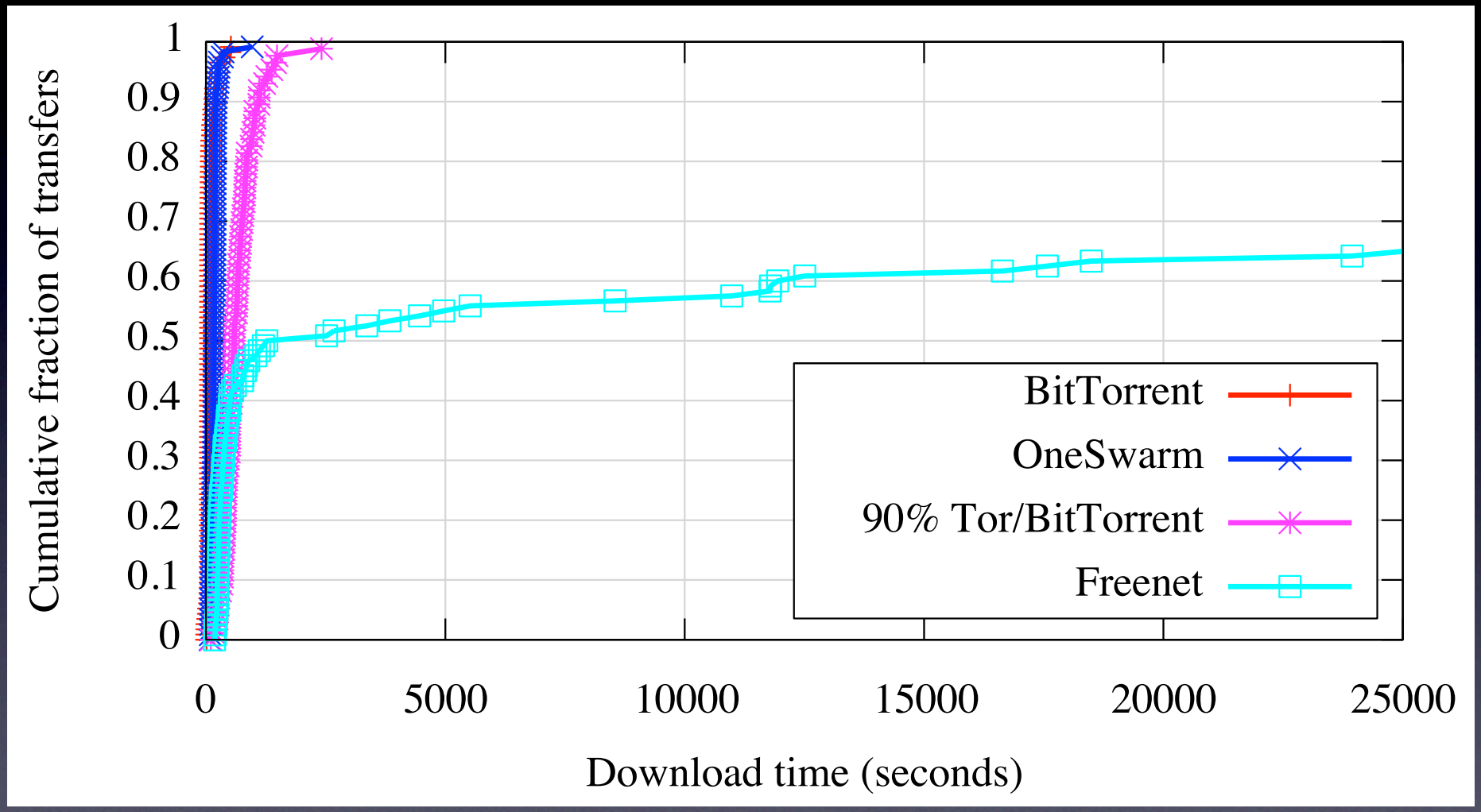
OneSwarm

- Gives users *control* of how they share data
- Maintains good *performance*
- *Trusted/untrusted peer* combination improves robustness
- Join the *100,000s* that have downloaded the OneSwarm client at:

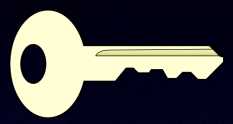
http://oneswarm.cs.washington.edu/

Extra slides

Performance zoom out



External material



- Creative Commons Attribution-Share Alike 3.0 Unported
- http://commons.wikimedia.org/wiki/File:Crypto_key.svg