ROYAL INSTITUTE OF TECHNOLOGY

Systems Issues in P2P

Slides by Jim Dowling

P2P in practice

- Many existing P2P protocols are elegant in theory but ugly in practice
- Why is Kademlia widely deployed on the open Internet, but not Chord? [d]

Node Heterogeneity

Systems Issues in P2P

- Today we will concentrate on three different systems issues that are important in building real-world P2P systems

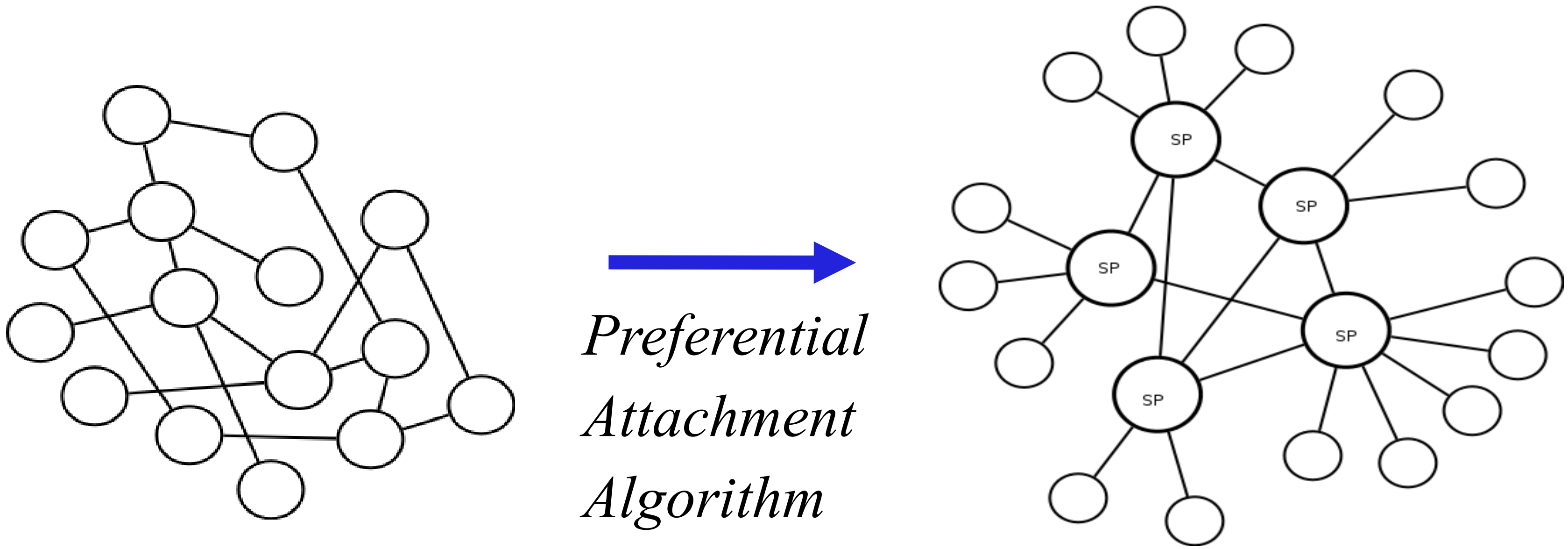
1. Node heterogeneity
2. Overcoming limited direct connectivity on the Internet
   - Network Address Translation Gateways and Firewalls
3. Secure gossiping protocols

Gossiping in Distributed Systems

- "Gossiping is the endless process of ***randomly*** choosing two members and subsequently letting these two exchange Information" [Kermarrec/Van Steen, Gossiping in distributed systems]

Scale-Free Networks [Barabasi]

- New nodes preferentially create links to those nodes with a higher number of links (positive feedback).
- *Symmetry breaking* from a random network.
  - Nodes now can use information encoded in the topology to send search requests to hubs.



Random Topology

Scale-Free Topology

Hetrogeneity

- Real-World P2P systems for the open Internet are heterogeneous
  - Peer resources (Bandwidth, CPU, Memory)
  - Peer session-time

- Use Peers with better “characteristics” to provide services to other peers in the system

All Peers are not Created Equal

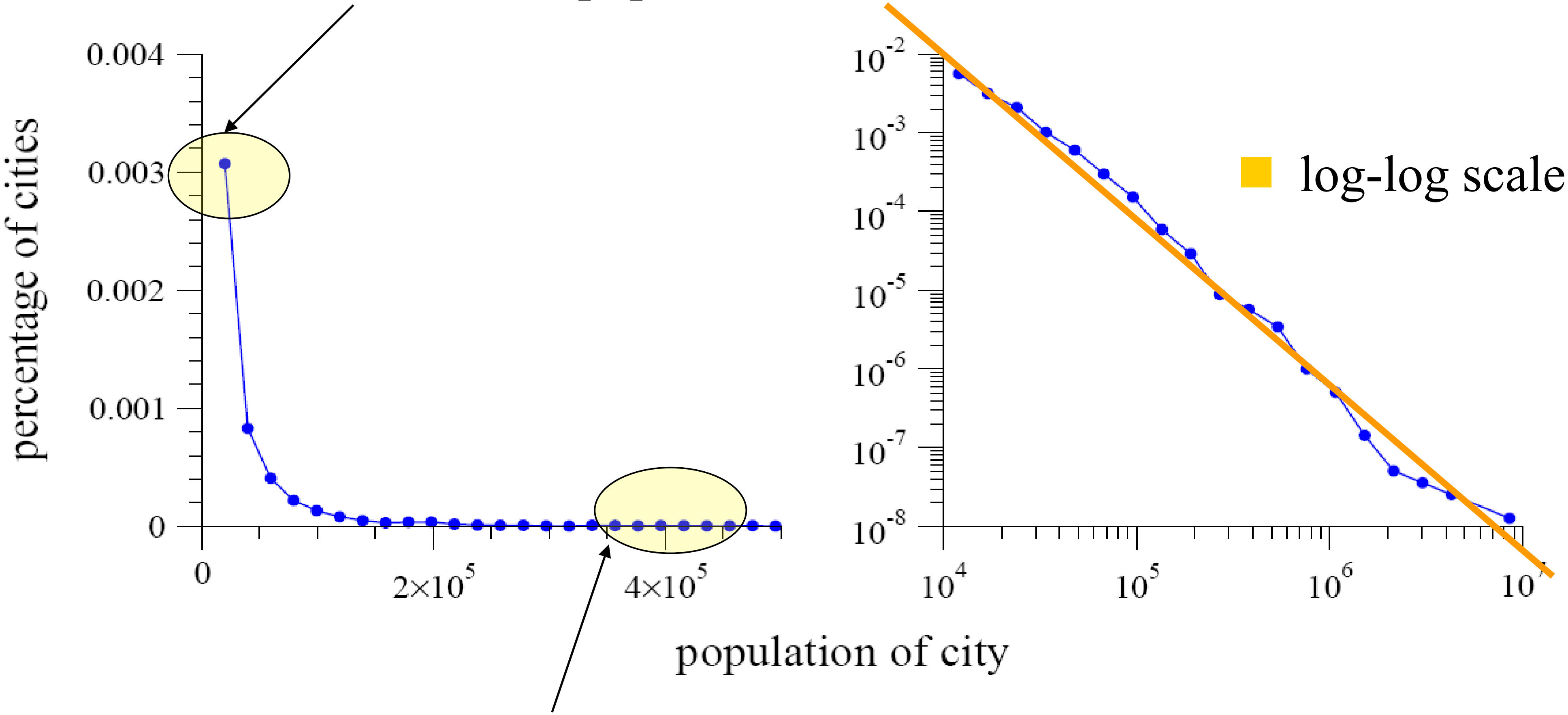
- Peers have heterogeneity with respect to:
  - Available Bandwidth
  - Average Session Time
  - Open IP address (vs. NAT-bound)
  - Latency
  - CPU/Memory

Peer Heterogeneity and Power Laws

- What type of heterogeneity is found in peers over different characteristics, such as bandwidth, session-time, etc?
- Measurements of P2P systems showed all sorts of power-law like relationships

Power Law Example

Lots of cities with a small population



Small number of cities with high population

Power Laws

A power law distribution satisfies:

$$Pr(X \geq x) \approx Cx^{-\alpha}$$

normalization constant
(probabilities over all x must sum to 1)

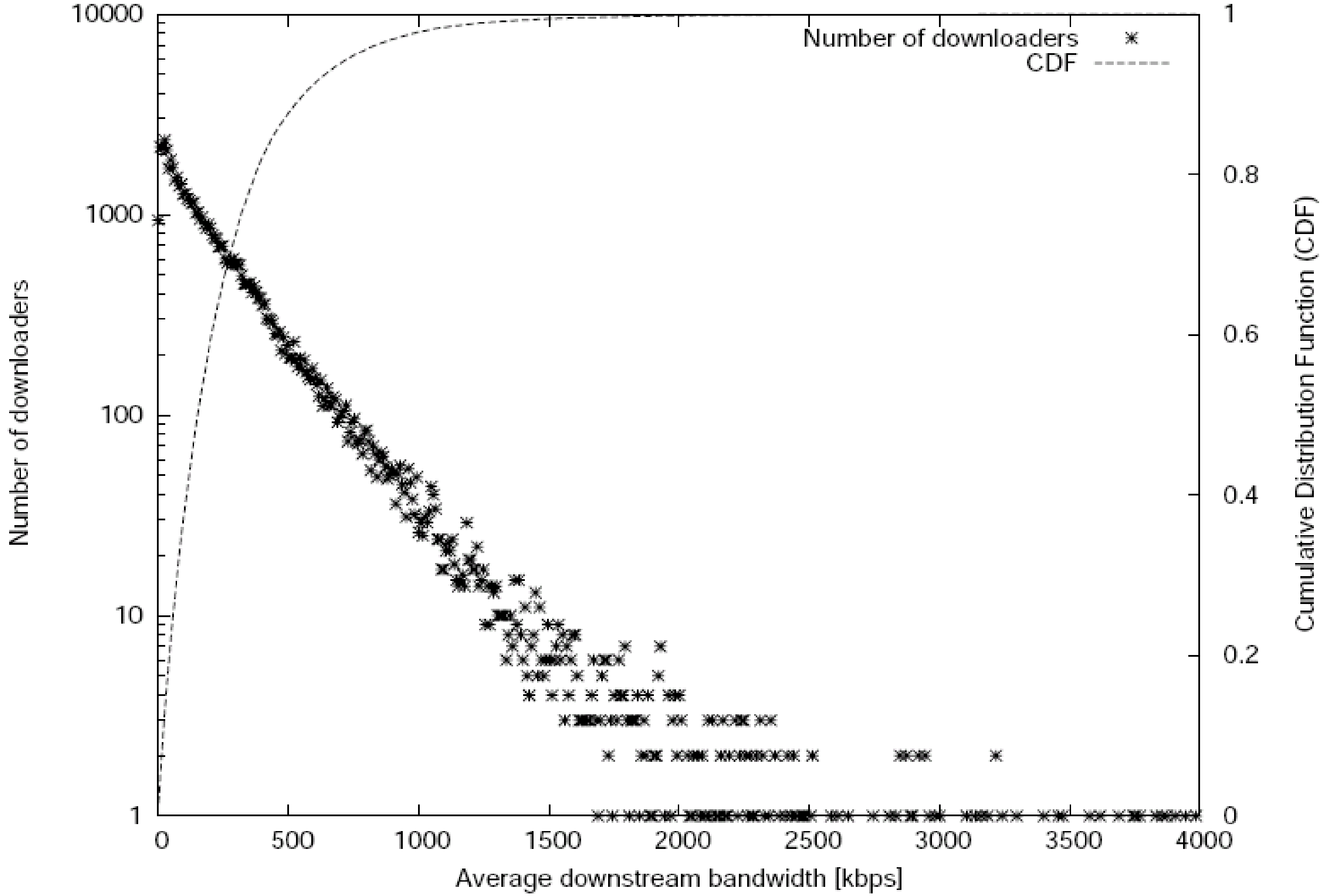
power law exponent α

Log-Log cumulative distribution function (CDF) is exactly linear:

$$\ln \Pr(X \geq x) \approx C - \alpha \ln x$$

FYI: Zipf and Pareto are similar to the power law distribution

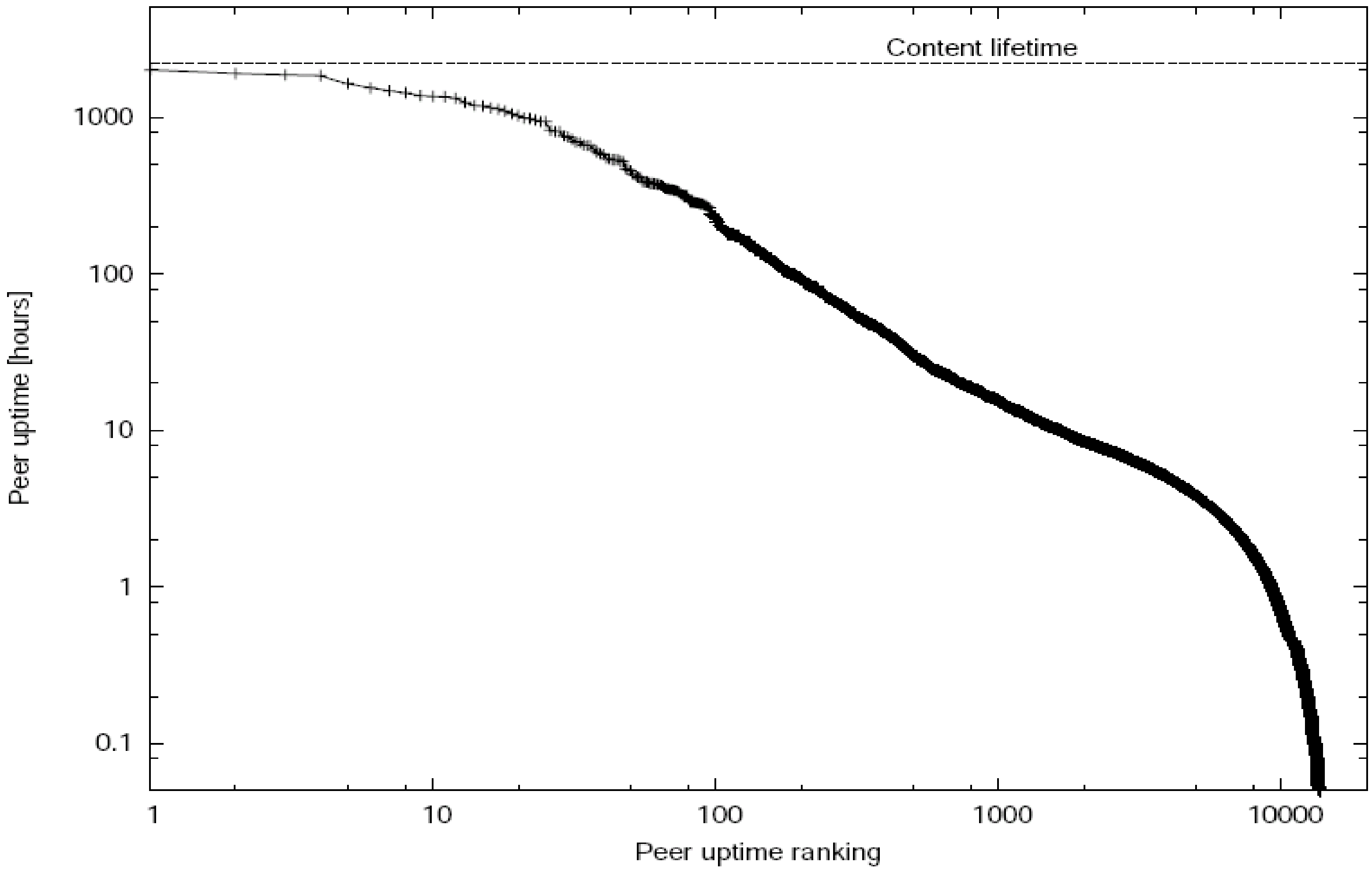
Bittorrent Download Speed Distribution



Plot of the download speeds of 54,845 peers over 2 week period

Poulse et al., "The Bittorrent P2P File-sharing System: Measurements and Analysis", IPTPS '06

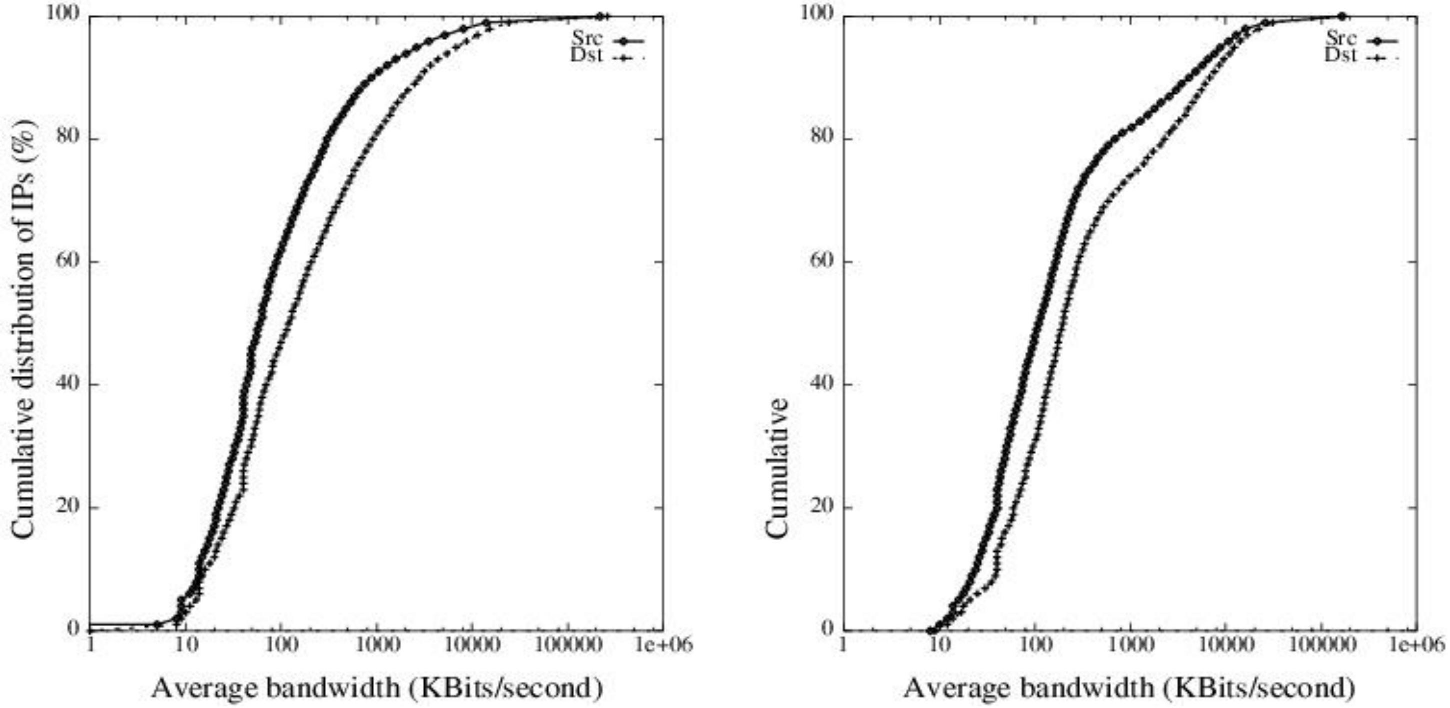
Bittorrent, Heavy-Tailed, for Session Time



Log-Log plot of the uptime distribution of the 53,833 peers

Poulse et al., “The Bittorrent P2P File-sharing System: Measurements and Analysis”, IPTPS '06

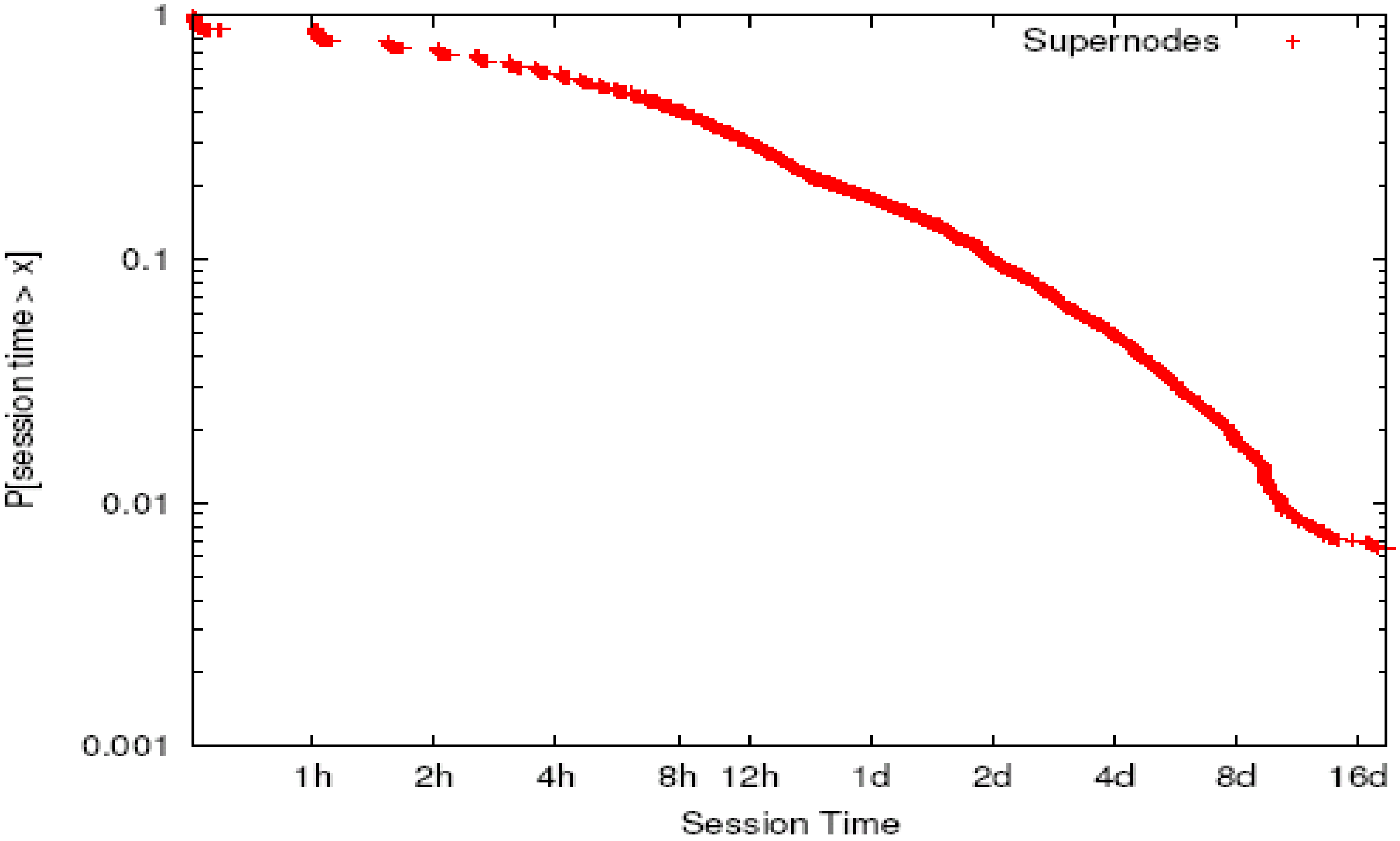
Peer Bandwidth Distribution



- FastTrack: 33% IP addresses have mean downstream b/w 56Kbps or less; 50% have mean upstream b/w 56Kbps or less
- Direct Connect: 20% IP addresses have mean downstream b/w 56Kbps or less; 33% have mean upstream b/w 56Kbps or less

Sen and Wang, Analyzing peer-to-peer traffic across large networks, IEEE/ACM TON, 2004

Super-Peers in Skype: session Times are heavy-tailed



Super-peer session times in Skype
(Loglog plot of the Cumulative Distribution Function)

Guha et al., "An Experimental Study of the Skype Peer-to-Peer VoIP System"

Super-Peer Definition

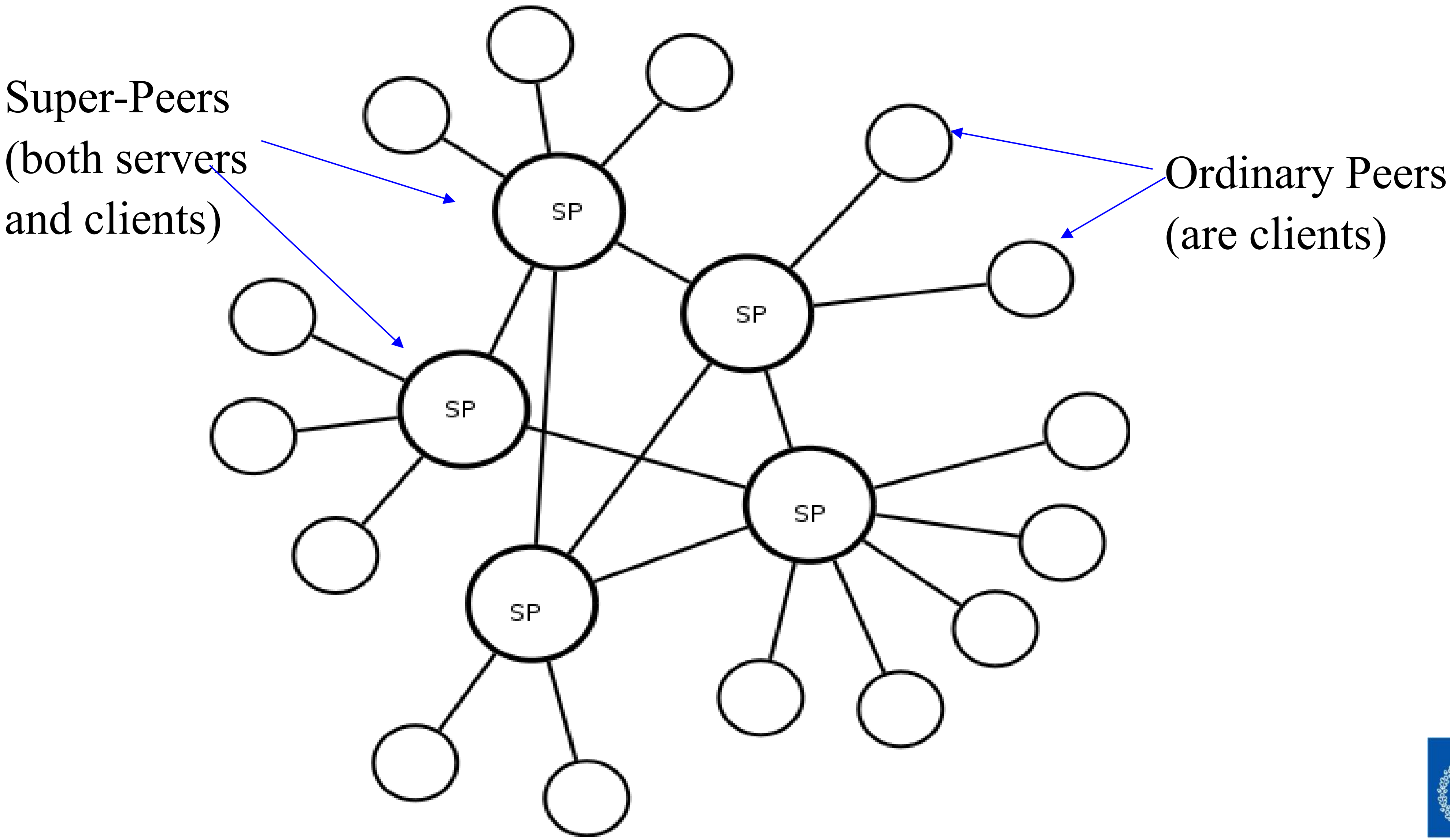
- Super-peers have *high utility* relative to non super-peers, where higher utility peers are "better" at providing super-peer service(s).
  - Measured peer utility can be used to rank peers to enable the best peers to be promoted to super-peers.

Spare Bandwidth/CPU; Open IP Address; etc

Super-Peer P2P Networks

- Exploit heterogeneity in P2P Networks by using higher utility peers to provide services
- Super-Peers provide redundant instances of System Services giving a P2P system:
  - Scalability
  - Load balancing
  - Fail-over
  - Robust to node failures, message loss

Super-Peer Architecture



Services provided by Super-Peers

- File Indexing/Retrieval
  - Fast-Track, Kazaa, E-Donkey
- Voice Over IP (VoIP)
  - Skype uses super-peers to setup and route calls
- Framework for building Super-Peer Systems
  - Sun's JXTA framework

Super-Peer (SP) Design Issues

- Ordinary peer to super-peer connections
- Intra-super-peer overlay network
- Super-peer promotion

Ordinary Peer to SP Connections

- Redundancy / Performance
  - =1 active SP connection per ordinary peer
    - Suitable for TCP traffic
  - >1 active SP connection per ordinary peer
    - Requires session management for P2P routing
- Fairness allocating Ordinary Peers to SPs
  - Don’t overuse the SP’s resources

Intra-Super-Peer Overlay Network

- Random Overlay Network
  - Random walk and gossiping or flooding
- DHT Overlay Network
  - Good for Identifier-based Routing
- Gradient Overlay Network
  - Good for SP discovery using gradient search
- Hierarchical : Skype, low latency but less robust.

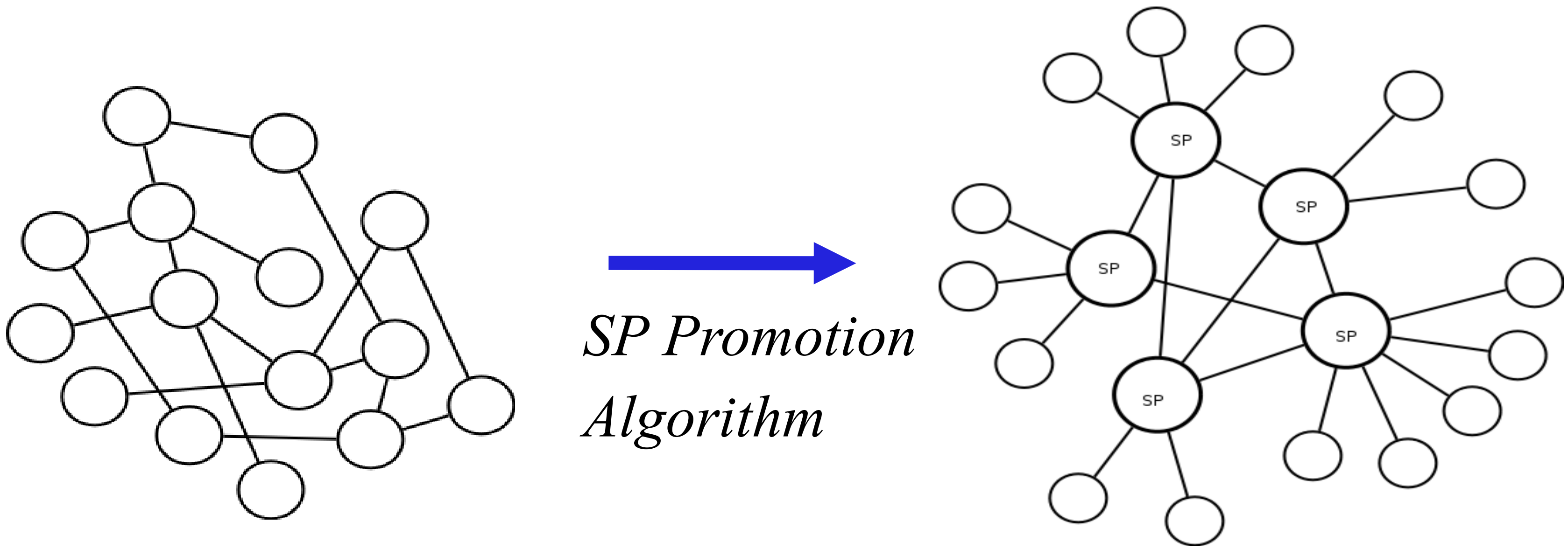
Super-Peer Promotion

- Peer Utility is Service Dependent:
  - What level of “utility” is required for a peer to become a super-peer?

- Options:
  - 1. Promote all peers whose utility exceeds a well-known utility level (uses local knowledge)
  - 2. Promote the top 'X' percent of peers with highest utility (requires global knowledge)

Super-Peer Promotion Decision Problem

- Local Decision > Centralised Decision
- Session-start or Runtime > Bootstrap Time
- Fairness to Super-Peers vs. System Availability



Random Topology

Super-Peer Topology

Super-Peer Promotion in Skype

- If the peer has an open IP address, and its measured available bandwidth exceeds a threshold, it is promoted to be a super-peer.
- At peer bootstrap-time, Skype runs the Simple Traversal of UDP through NATs (STUN) protocol between the Peer and a Server

Guha et al., “An Experimental Study of the Skype Peer-to-Peer VoIP System”

Overcoming Limited Direct Connectivity in IP

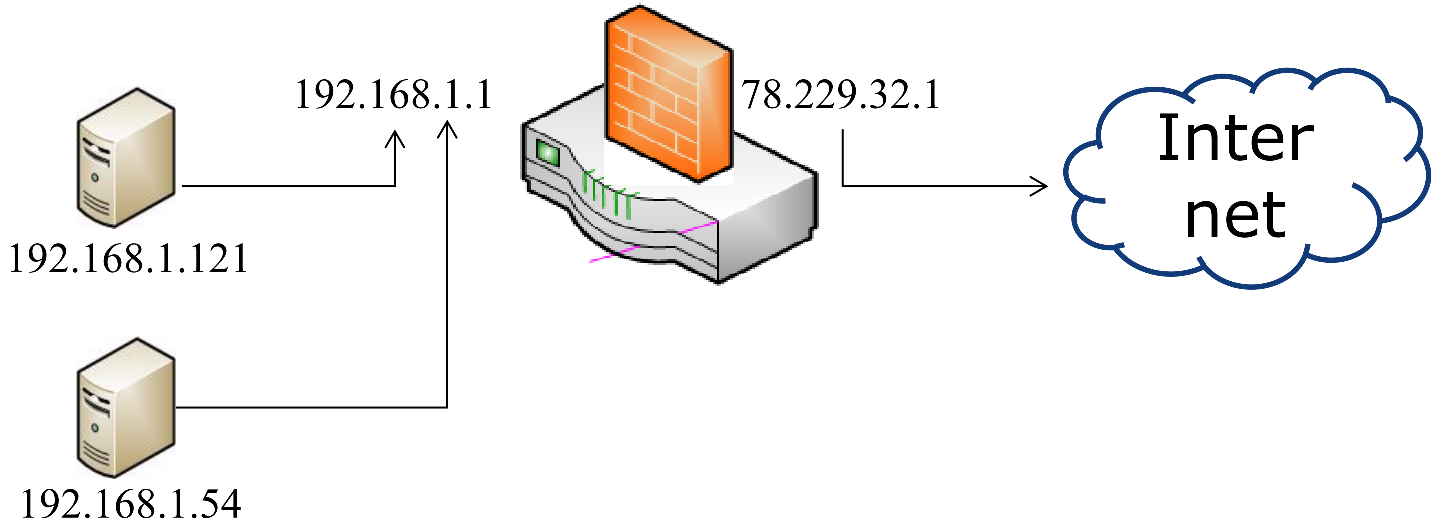
ROYAL INSTITUTE
OF TECHNOLOGY

Direct Connectivity on the Internet

- Naive assumption: any node can establish a direct connection to any other node on the Internet.
- For any given P2P system, roughly 80-90% of the time this is not true!
- NATs and firewalls get in the way!
- It's getting both better (UPnP) and worse (decreasing number of available IP addresses) atm.
- IPV6 will not make this problem just go away.

NAT Devices

- NAT devices differ in many application-observable aspects.
- NAT port mappings,
- Traffic filtering,
- NAT binding timeouts,
- ICMP handling,
- Queuing,
- Hair pinning,
- Buffer sizes

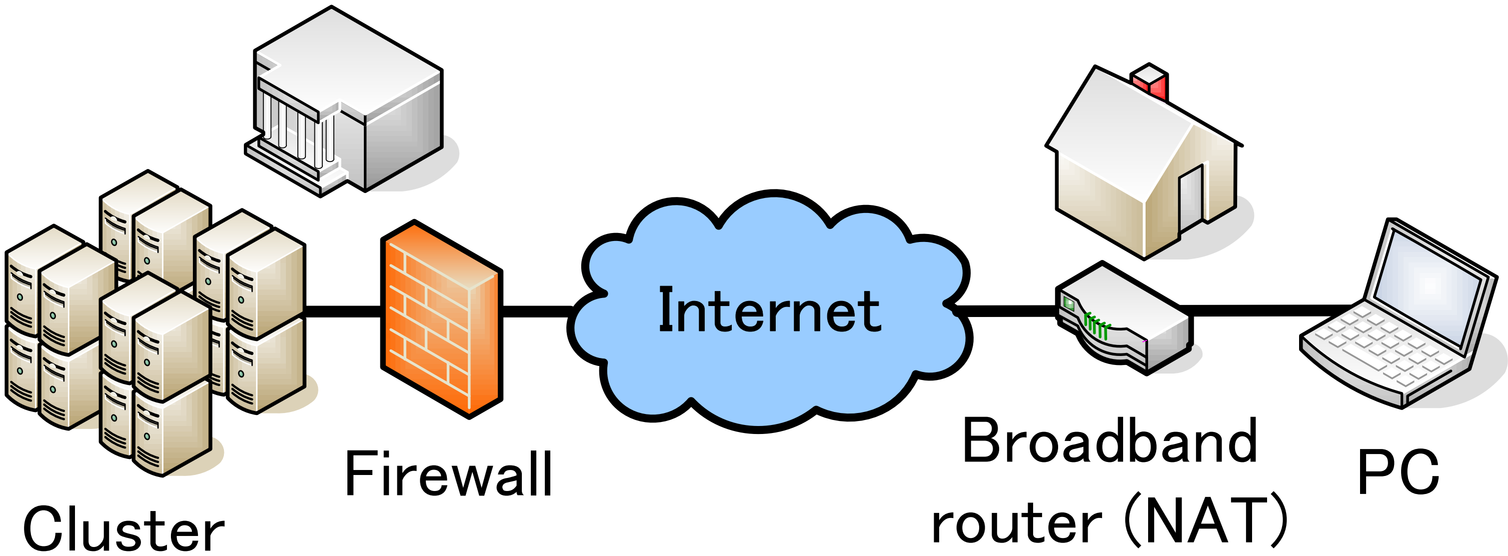


IETF NAT Behavioral Requirements standards not adopted yet by manufacturers.

NAT Type Classification

- BEHAVE RFC [1] defines NAT behaviour as a set of policies:
  - Port Allocation
  - Port Mapping
  - Port Filtering
  - NAT Binding Timeout

~~OLD NAT MODEL~~
~~Symmetric~~
~~Port-Restricted~~
~~Partial-Cone~~
~~Full-Cone~~



NAT Port Allocation Policy

NAT with Public IP = 124.29.31.1

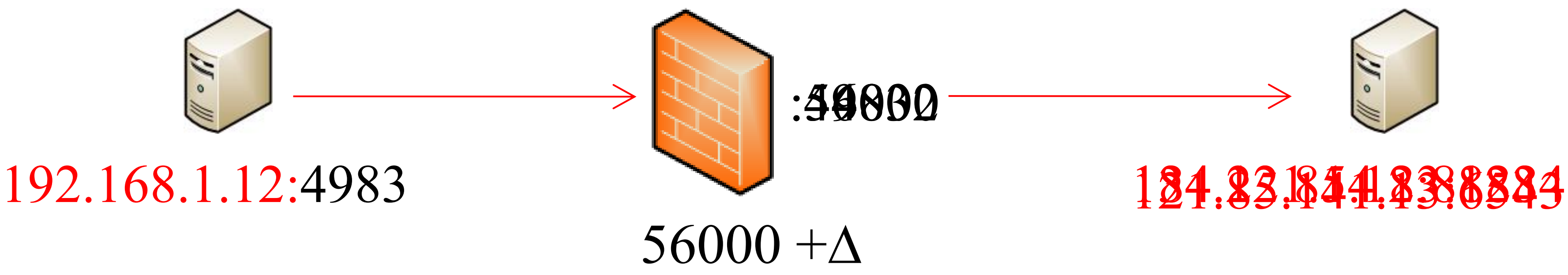
Source IP:port	NAT Port	Destination IP:port
192.168.1.12:4983	4983	134.229.81.12:8888
192.168.1.12:4983	56000	121.85.141.13:6543
192.168.1.12:4983	54832	184.121.54.83:1234

Port Allocation Policy

Preservation

Contiguity

Random



Port Mapping Policy

Source IP:port	NAT Port	Destination IP:port
192.168.1.12:4983	4983	134.22.81.12:8888 134.22.81.12:6543 184.121.54.8:1234

Endpoint Independent Mapping
(Preservation)

Source IP:port	NAT Port	Destination IP:port
192.168.1.12:4983	56000	134.22.81.12:8888 134.22.81.12:6543
192.168.1.12:4983	56001	184.121.54.8:1234

Host Dependent Mapping
(Contiguity)

Source IP:port	NAT Port	Destination IP:port
192.168.1.12:4983	13545	134.22.81.12:8888
192.168.1.12:4983	45352	134.22.81.12:6543
192.168.1.12:4983	6957	184.121.54.8:1234

Port Dependent Mapping
(Random)

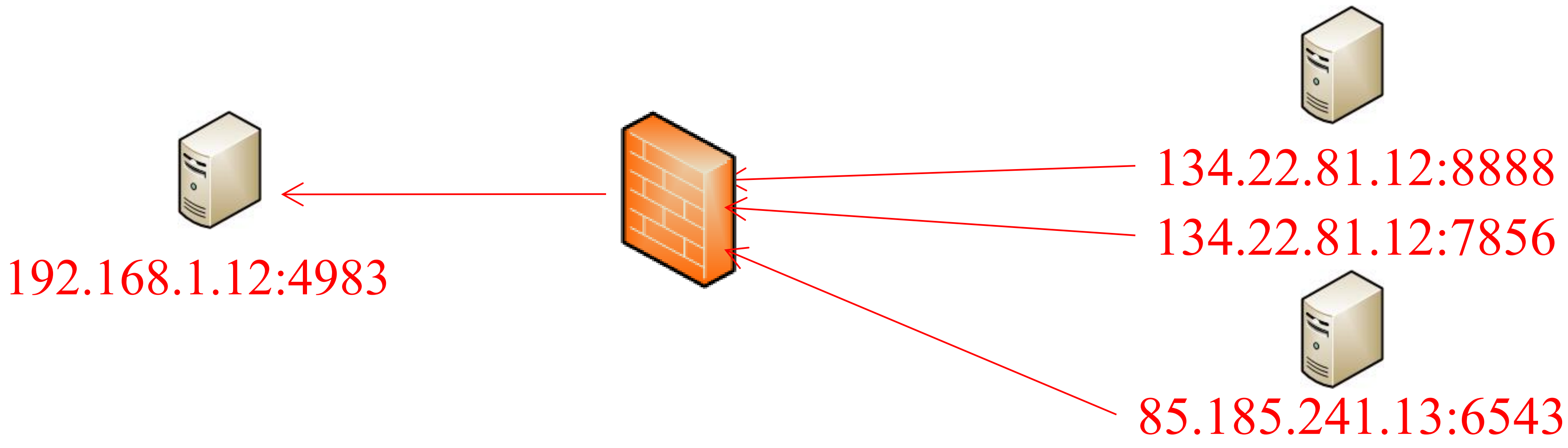
KTH VETENSKAP OCH KONST
ROYAL INSTITUTE OF TECHNOLOGY

NAT Port Filtering Policy

Source IP:port	NAT Port	Destination IP:port
192.168.1.12:4983	4983	134.229.81.12:8888

Port Filtering Policy

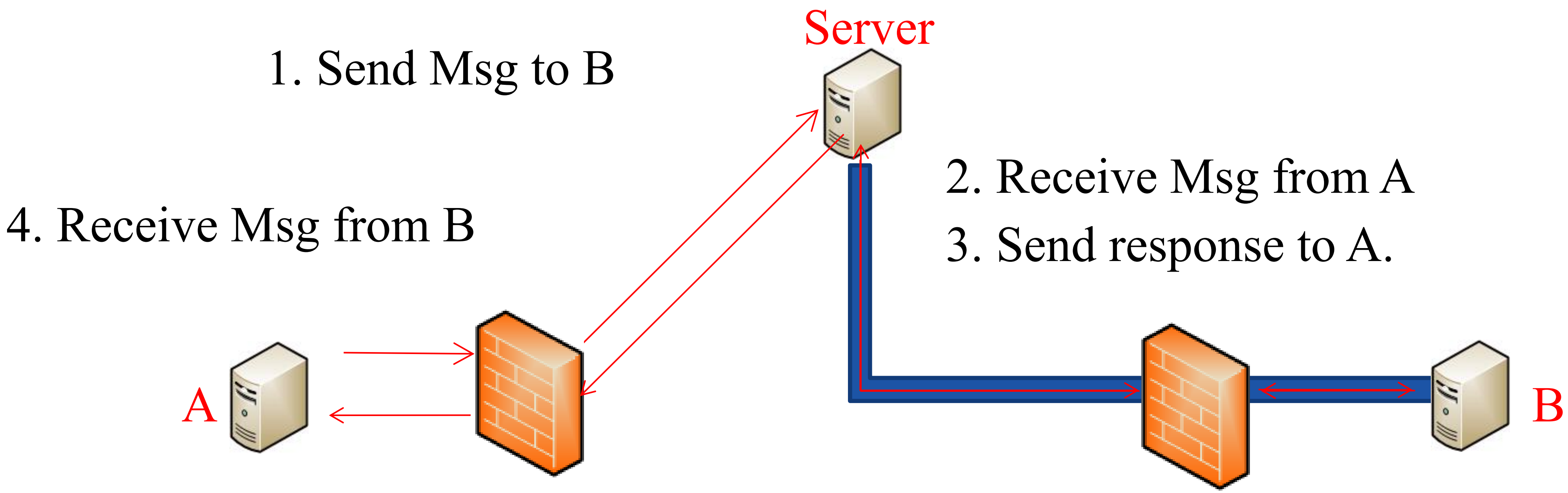
EI	**HD**	**PD**	Incoming Packet
Y	Y	Y	134.229.81.12:8888
Y	Y	N	134.229.81.12:7856
Y	N	N	85.185.241.13:6543



EI =Endpoint Independent; HD=Host Dependent; PD=Port Dependent

Relaying

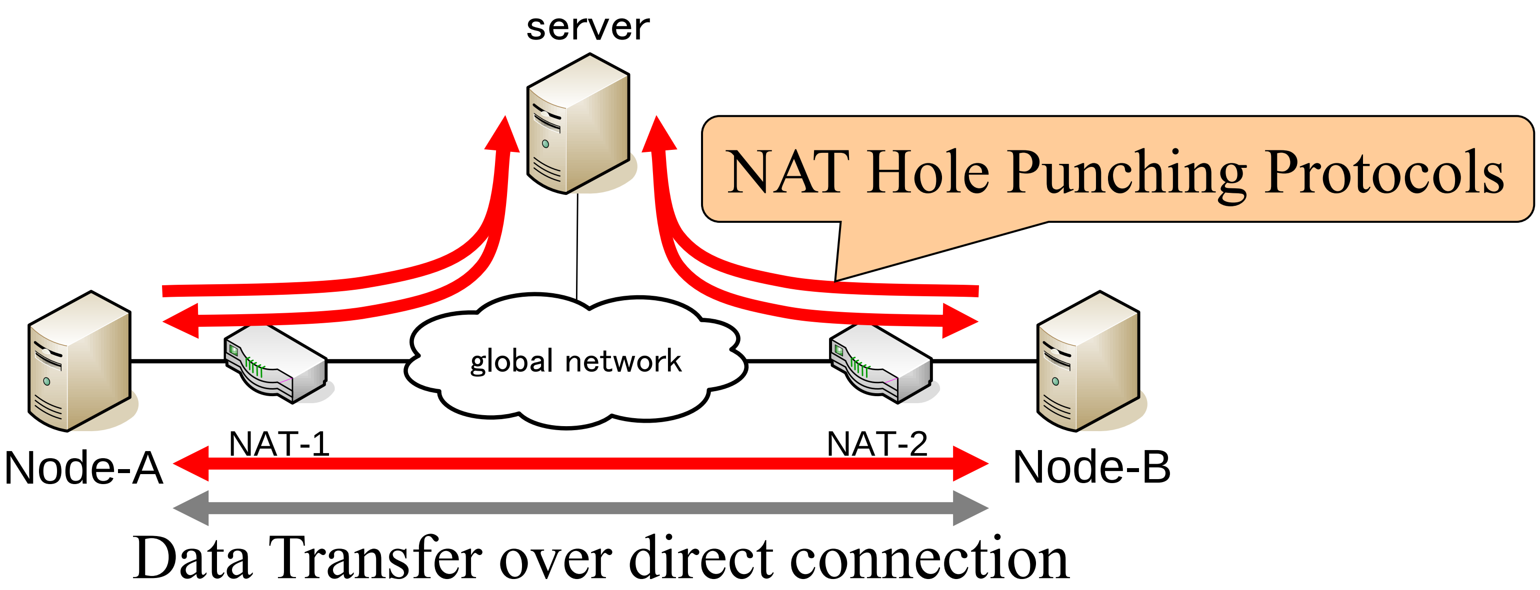
- Relaying of P2P traffic requires that a node behind a NAT has a valid port mapping in its NAT for a Server. This can be achieved using an open TCP connection or heartbeating over UDP.
- When node A wants to communicate with node B, it send a message to the Server that routes the message to B via its existing connection to B.



NAT Hole Punching Strategies

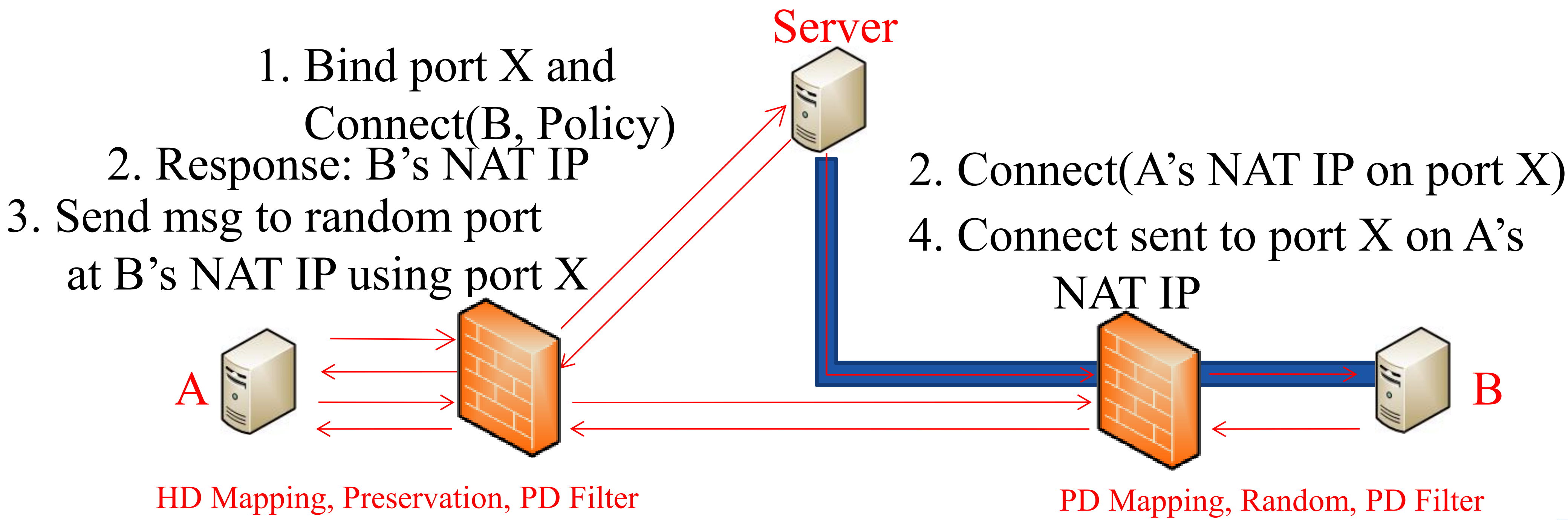
- Connection reversal
  - From public node to a private node
- Simple Hole-Punching
  - Endpoint-Independent filtering and/or mapping required
- Port-prediction using Preservation
- Port-prediction using Contiguity

lower chance of success



Hole-Punching using NAT Combinations

- It is the combination of NAT types of 2 nodes that is important when connecting two nodes behind NATs.
- In the example below, two nodes connect using 'Port-prediction using Preservation'.

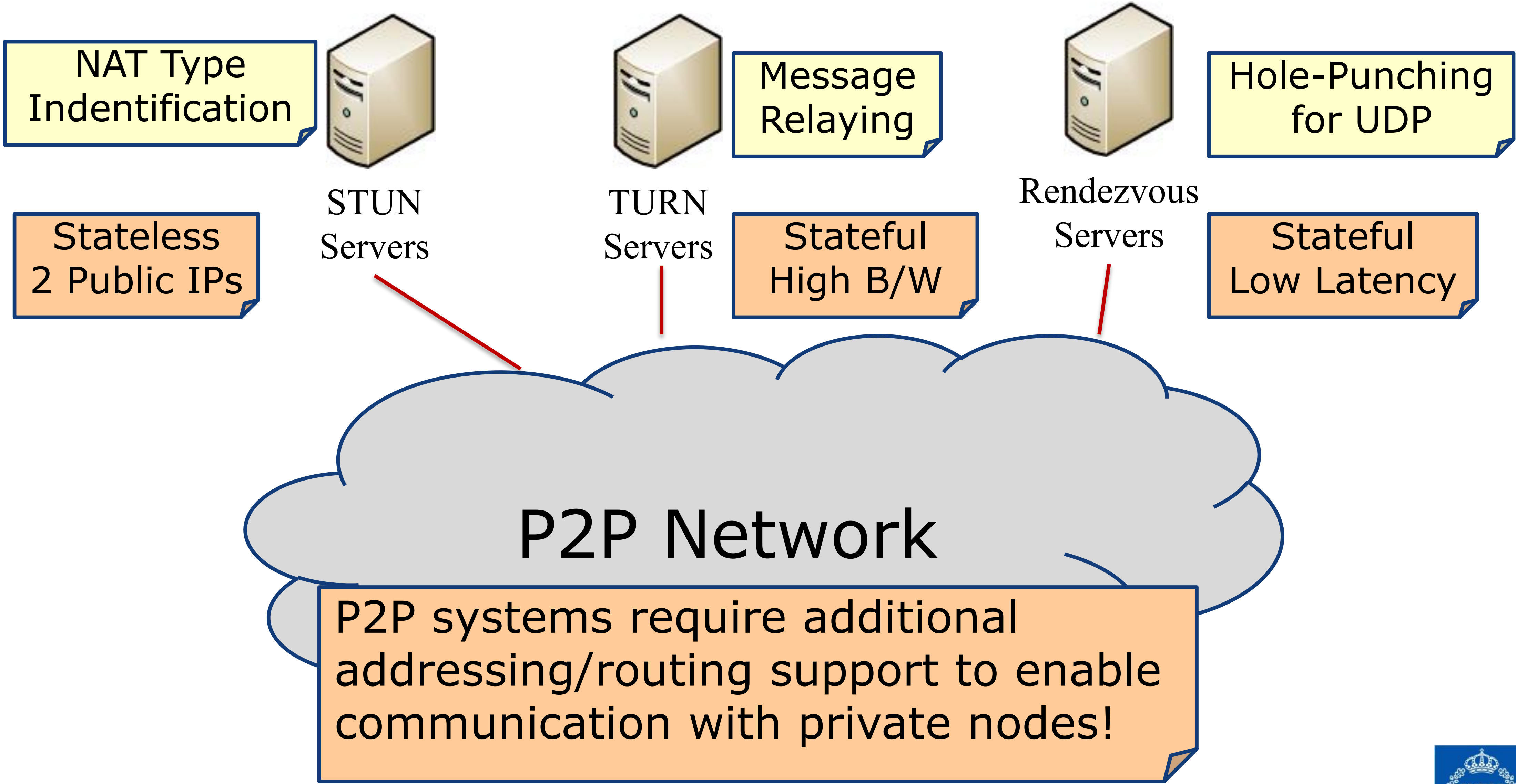


To Relay or Hole-Punch P2P Traffic?

	Management and control traffic	**Application-level Data transfer**
Requirement	Reliable, Low latency	High throughput (large data volume)
Mechanism	Relay	Hole-punch
Challenges	Fairly distribute traffic over relay-nodes	Improve success rate. Reduce connection latency.

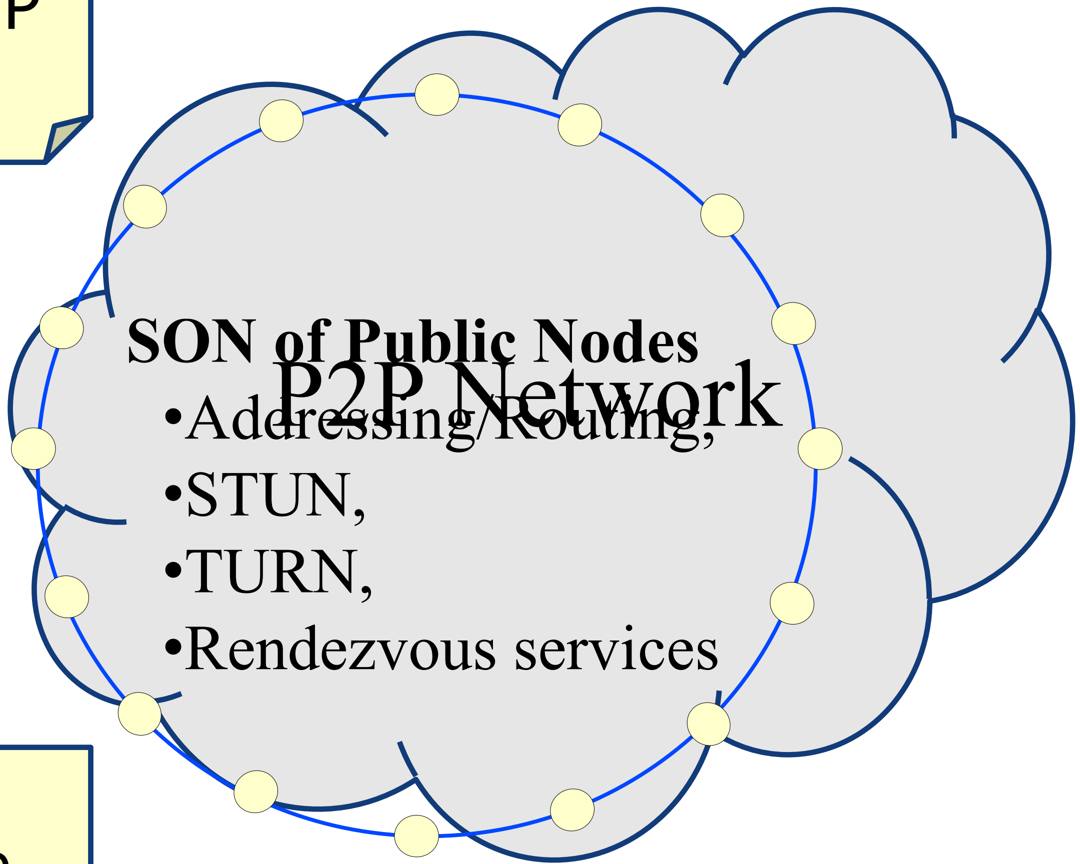
ROYAL INSTITUTE OF TECHNOLOGY

Existing P2P NAT Infrastructures



Distributed NAT Infrastructure [Usurp]

Public Nodes have an Open IP address or support the UPnP IGD profile.

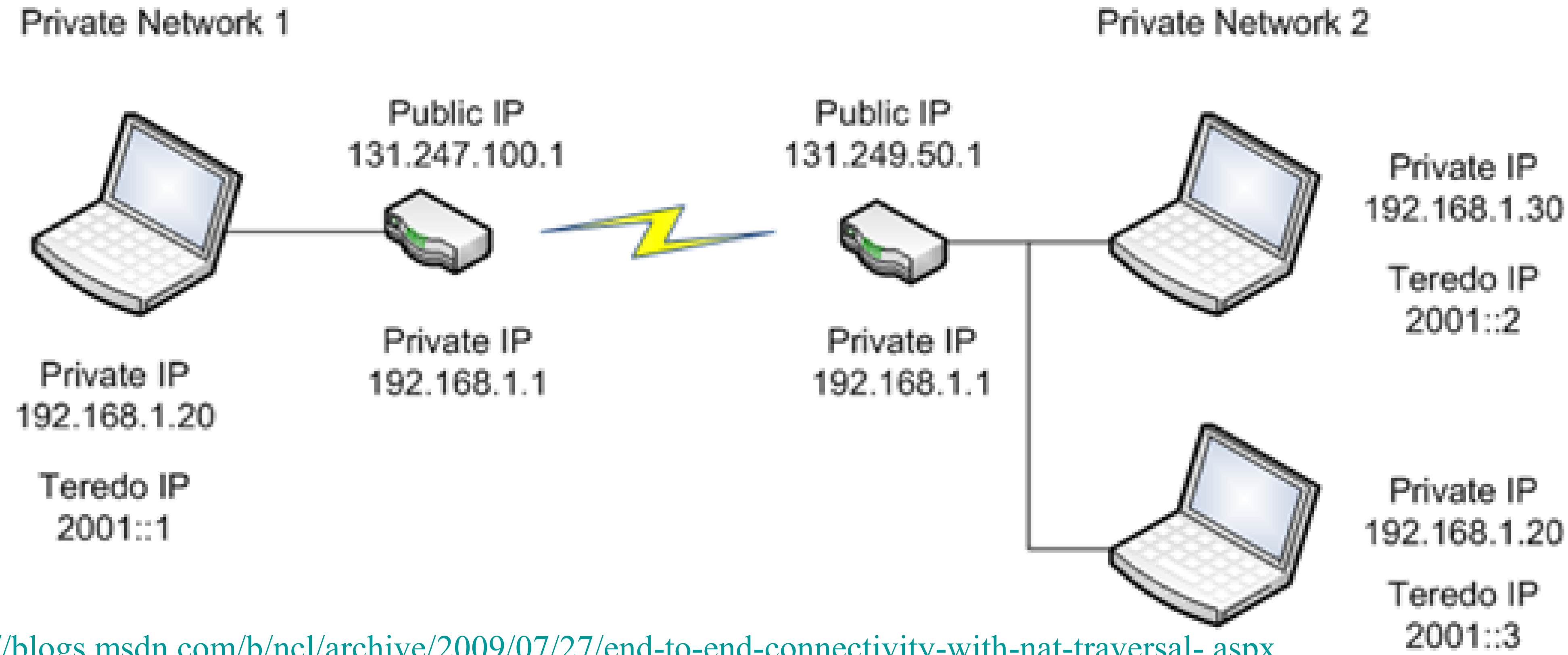


Private Nodes are behind NATs/firewalls and become clients of public nodes.

KTH VETENSKAP OCH KONST
ROYAL INSTITUTE OF TECHNOLOGY

Enabling NAT Traversal by Configuration

- Explicit port forwarding in home routers
  - Requires sophisticated users
- UPnP Internet Gateway Device (IGD)
  - Devices that support UPnP IGD can act as public nodes
- Teredo IPV6 Tunneling



http://blogs.msdn.com/b/ncl/archive/2009/07/27/end-to-end-connectivity-with-nat-traversal-.aspx

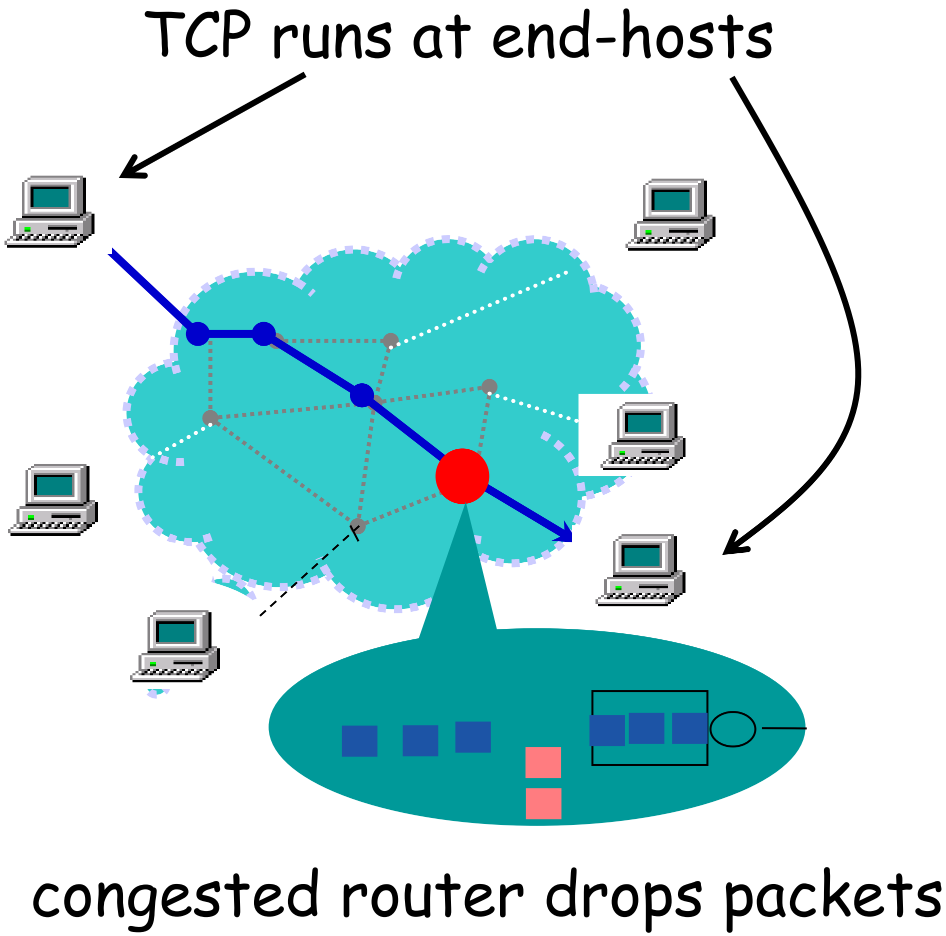
Congestion Control for P2P Systems

Congestion Control in P2P systems

- TCP has very low NAT-traversal success rates in real-world P2P systems (compared to UDP)
  - NAT-traversal techniques such as STUNT are not widely deployed.
  - UDP enables the utilization of more peers upload bandwidth.
- P2P systems based on UDP have to consider congestion control in sending/receiving data over the network.
- Congestion control algorithms have to consider inefficiency and congestion collapse
  - "self-interest" vs. "social welfare"

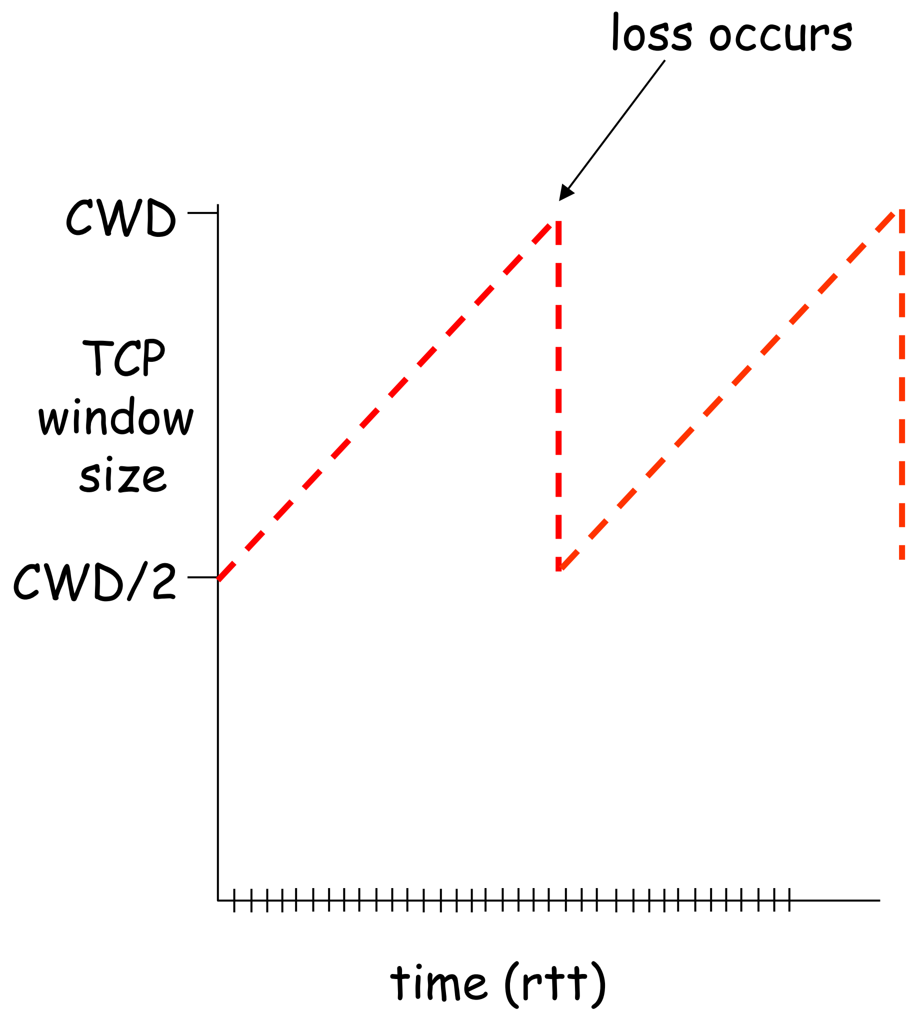
TCP Congestion Control Behavior

- Congestion control:
  - decrease sending rate when loss detected, increase when no loss
- Routers
  - discard packets (tail-drop) when congestion occurs
- TCP is slow to ramp up even if spare bandwidth is huge (slow start)
  - Increases by 1 segment/RTT
  - Can do better on modern networks



Generic TCP Behavior

- Increase congestion window size by one segment (1500 bytes) per RTT
- Halve CWD size on detection of loss, CWD <– CWD /2
- If there is a timeout due to missed ACKs reset the CWD size to 1, *CWD* <– 1
- Relationship between network throughput and loss is shown on the right.



LEDBAT

- When UDP is used to build P2P systems, you need to implement your own congestion control algorithm.

- LEDBAT is a congestion control algorithm that uses *delay-based congestion control* (not loss-based as in TCP) to control amount of traffic sent over a link
  - If the packet delay over a link exceeds a threshold value (default 100ms), then decrease sending rate
- LEDBAT 'backs off' to TCP
  - It should not cause a congestion collapse of the Internet!
  - It can parasitically use your bandwidth and back-off when you want to use TCP applications.

KTH VETENSKAP OCH KONST

ROYAL INSTITUTE OF TECHNOLOGY

Secure Gossiping

How secure are gossiping algorithms?

- How can they be exploited by malicious nodes (*attackers*)? [d]

- Example:
  For peer-sampling services (PSS), can the sampling process can be biased toward a specific group of nodes instead of being random?

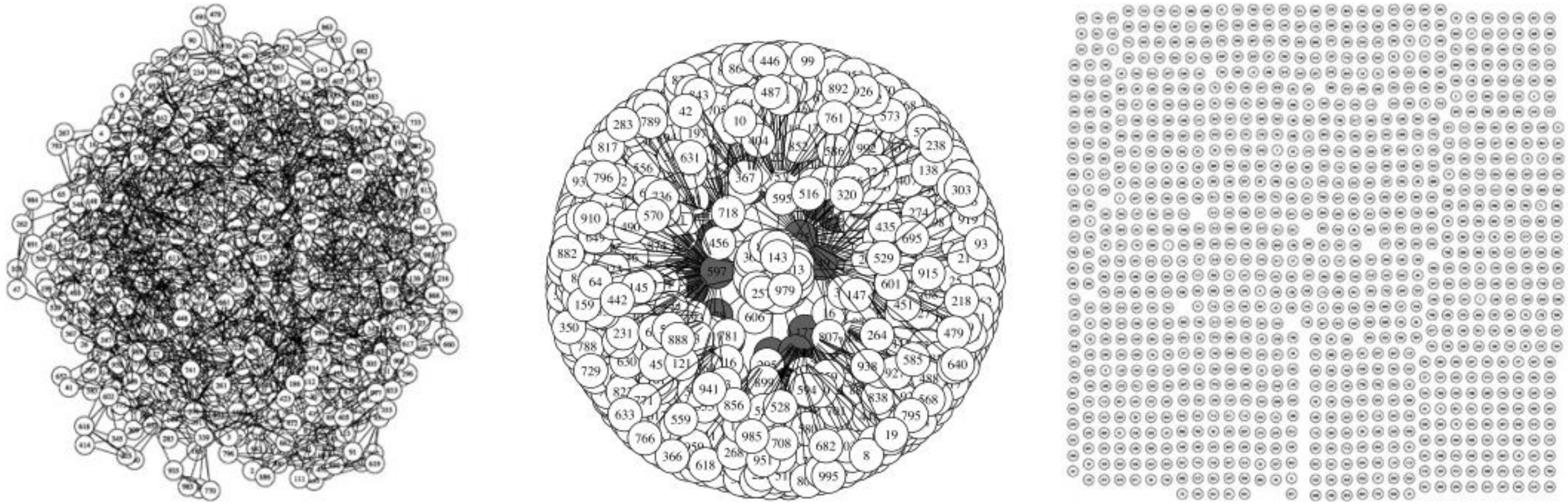
- What about P2P systems that have quality-of-service (QoS) requirements – e.g., media streaming that is vulnerable to QoS fluctuations?
  - Proactive rather than reactive solutions.

KTH VETENSKAP OCH KONST
ROYAL INSTITUTE OF TECHNOLOGY

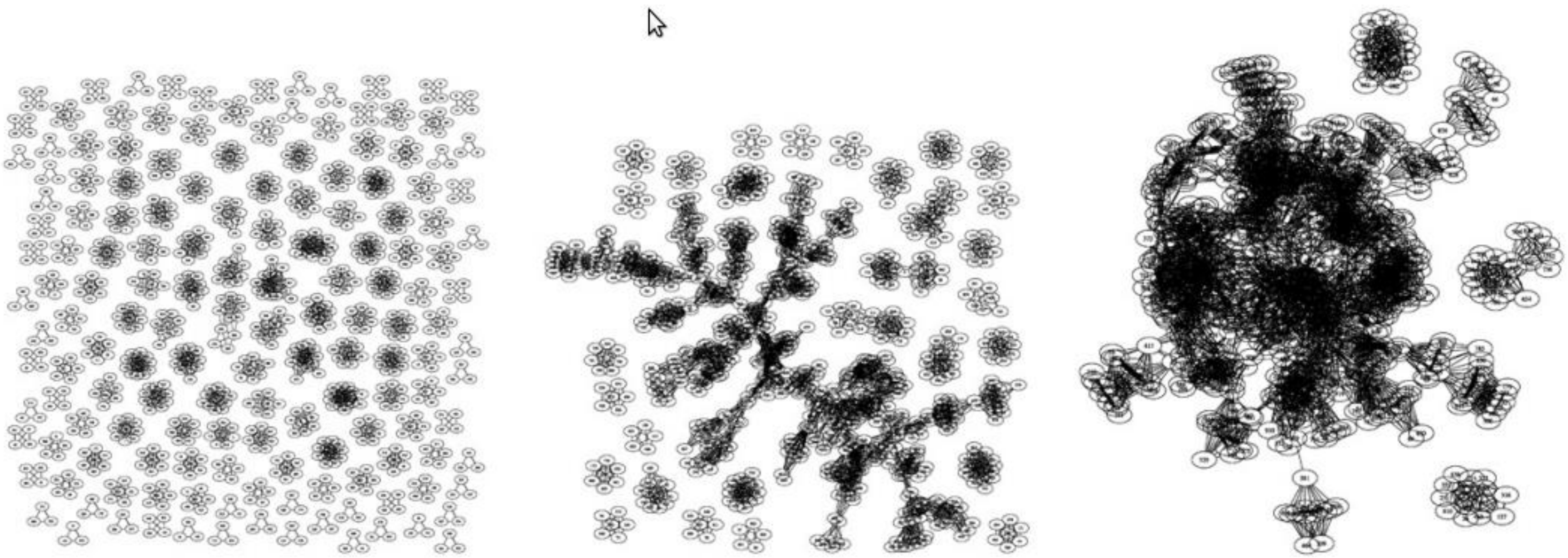
Dummy's guide to attack gossip systems

- Write your own gossip-based client for the protocol you wish to attack.
- Decide on the number **f** of attackers: store a well-known list of your other attackers
- Run the standard gossip protocol with the following exceptions:
  - remove restrictions on the size of your partial view;
  - the message sent to a receiver R is populated with malicious descriptors based on a specific attack strategy;
  - the timestamps of malicious descriptors are manipulated in order to postpone their dropping as late as possible.

How big does ‘f’ have to be to attack? [Jesi]



(a) Healthy (pseudo) random graph (b) Hub topology, $f = 20$ (c) After the attack: $f = 20$



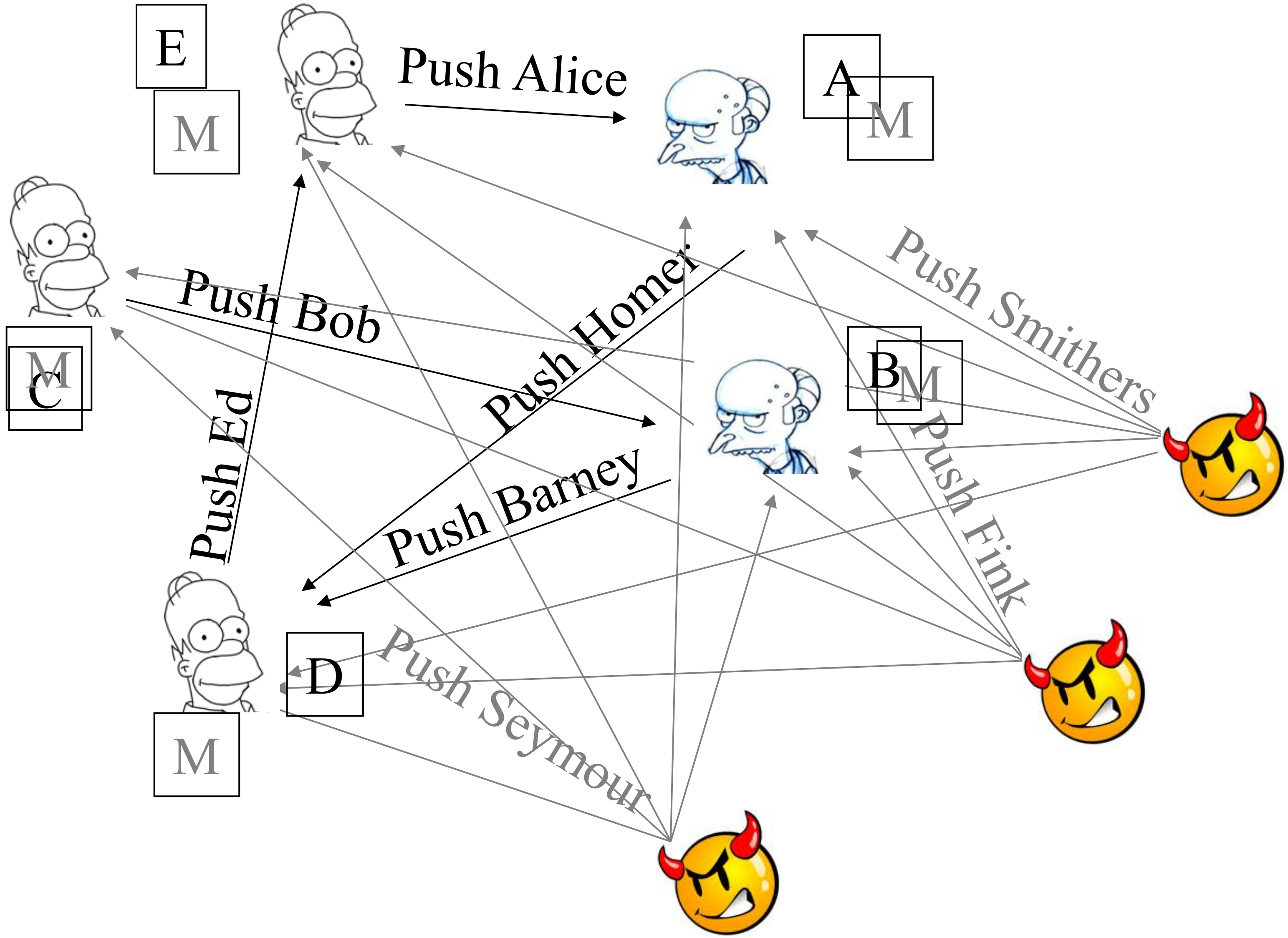
(d) After the attack: $f = 18$ (e) After the attack: $f = 16$ (f) After the attack: $f = 14$

For this PSS, the view size is 20. Even with ‘f’ lower than 20, we can pollute the system.

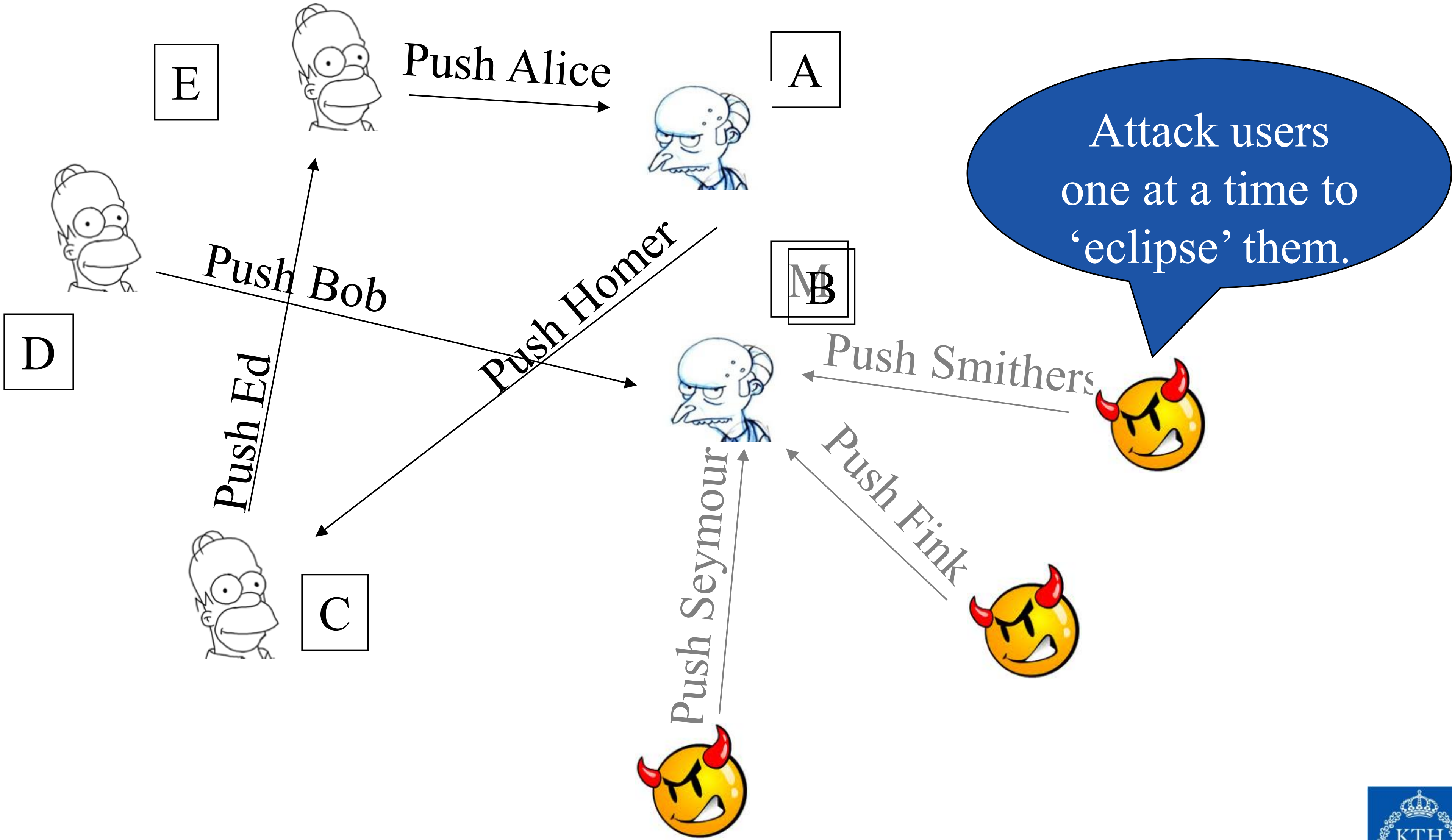
Adversary Attacks in Gossip-Based Systems

- An **attacker** may want to bias samples
  - Isolate nodes, bias statistics, become a hub, etc
- Attacks
  - discard specific node descriptors
  - replay msgs to avoid discarding of node descriptors
  - corrupt messages by modifying their node descriptors
  - forge bogus node descriptors to pollute the network with
  - bias node selection to attack individual nodes
  - flooding attack sends messages faster than gossip rate
- Faulty nodes may also be treated as an attack
  - Byzantine failures are possible

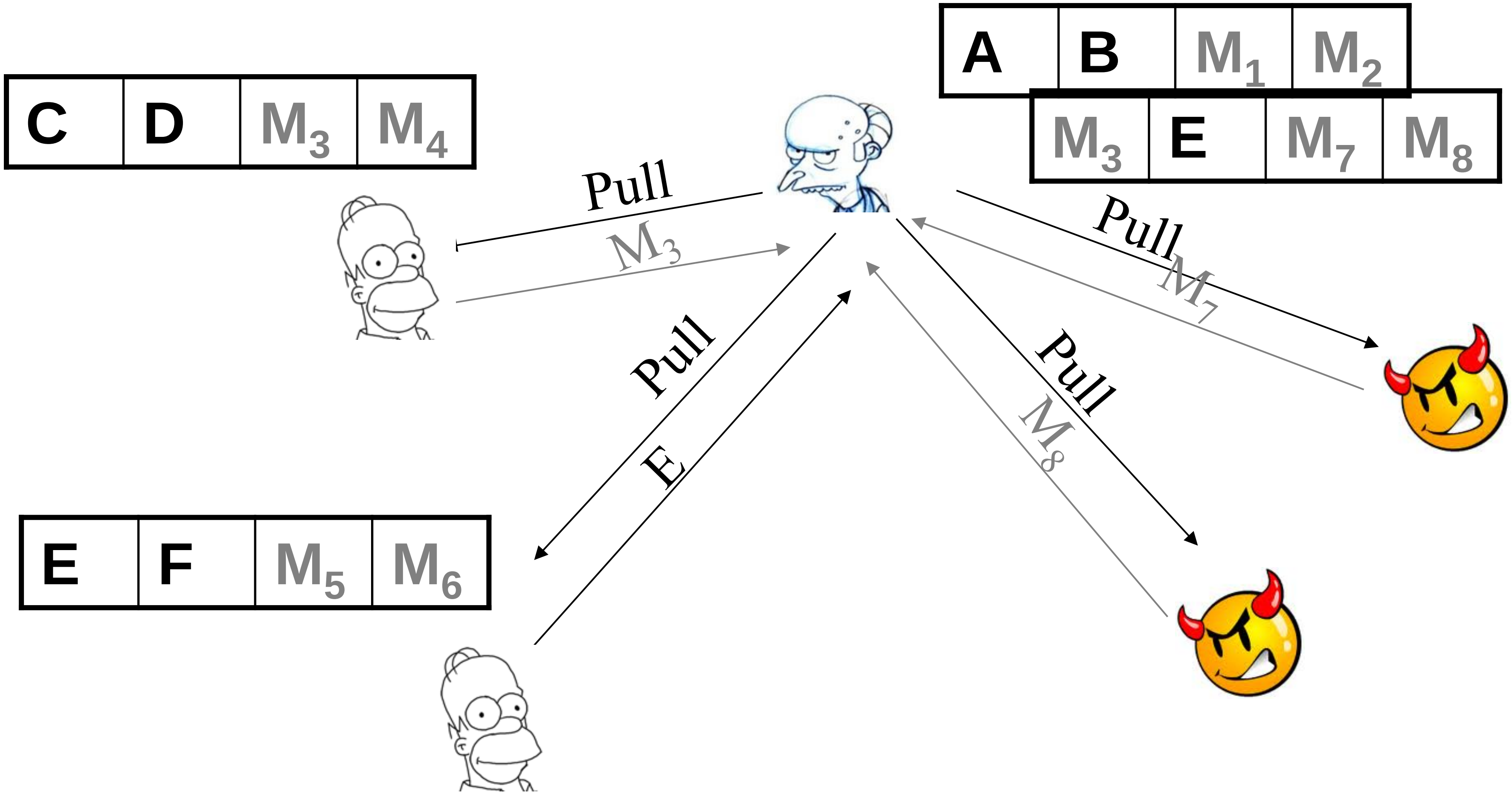
Push Drowning [Brahms]



Eclipse attack [Brahms]



Pull Deterioration [Brahms]



50% faulty ids in views ⇒ 75% faulty ids in views

Denial of Service Attack

- Denial of service (DoS) attacks involve flooding a node with gossip requests, so that the node does not have enough available resources to handle valid gossip requests

- [DRUM] prevents DoS attacks using two main techniques:
  - bound the amount of resources allocated to each gossip operation and
  - direct these operations to random ports

Byzantine-resilient gossip

- Live-streaming gossip-based protocol.
- Synchronous network model
  - Clocks synchronized within Δ seconds of each other
  - Nodes communicate over point-to-point unreliable links
- Limits each IP address to at most one identity
  - Mitigate Sybil attacks
- Nodes are either Byzantine or Altruistic or Rational (**BAR Gossip**)
  - *Altruistic nodes follow the protocol regardless* of costs
  - R*ational nodes follow a strategy that maximizes* their utility
  - *Byzantine nodes behave* arbitrarily
- Nodes have public/private certificates.

ROYAL INSTITUTE OF TECHNOLOGY

BAR Gossip

- Every node has a full static view (not a partial view)
- BAR-Gossip is a sequence of T + Δ-long rounds
  - T is a time interval sufficient to complete the message exchanges
- Nodes periodically execute 2 gossip protocols:
  - initiate **balanced exchange** of non-expired updates with a randomly selected neighbour
  - initiate **optimistic push** of non-expired updates with a randomly selected neighbour
- Signed messages that are internally inconsistent with the protocol amount to proofs of misbehavior
  - Those nodes are evicted from the system

BAR Gossip

- Nodes exchanges 3 pieces of information:
- History exchange
  - A node learns about the updates the other node holds
- Update exchange
  - Each node copies a subset of these updates into a *briefcase* that is sent, encrypted, to the other node
- Key exchange
  - where the parties swap the keys needed to access the updates in the briefcases
- History exchange and update exchange use TCP. Key exchange uses UDP.

Balanced Exchange and Optimistic Push

- *Balanced Exchange* and *Optimistic Push Protocols* are two gossiping algorithms that exchange the same information.

- They differ in what the parties disclose to each other during a *history exchange and in how they determine* the content of their respective *briefcases* during the *update exchange.*

Balanced Exchange

- Each party sends to the other a *history* set H containing the identifiers of all the updates it currently holds, compares the history it has received with its own, and determines the largest number k of updates that can be exchanged on a one-for-one basis

Optimistic Push

- Optimistic Push helps nodes that have fallen behind in the broadcast and that may not have any updates to trade in a Balanced Exchange
- The initiator S forwards to the receiver R two lists: a young list, which contains the IDs of some of the most recent updates S knows, and an old list, which contains the IDs of updates that S is missing and that are about to expire.
- R replies with a want list, which contains the IDs of the updates in the young list that R is missing.
- S and R then exchange briefcases: S's briefcase contains k updates in the want list, while R's briefcase is free to contain junk.

Optimistic Push

- Optimistic Push with two parameters: *pushage and pushsize: the young list consists only of updates* that have been broadcast within the last *pushage* rounds and *pushsize is an upper limit on the number* of updates that the Receiver can place in its *want list.*
  - Larger values of *pushsize* help lagging nodes to catch up faster, but allow nodes to waste bandwidth

Rational Behaviour – Peer Selection

- *Problem:*
  What if a rational node selects more partners per round than prescribed or biases its selections instead of choosing partners uniformly at random?
- *Solution:*
  Restrict choice within balanced exchanges and optimistic pushes

BAR Gossip – Peer Selection

- The sender *S* selects a peer for round *r* by seeding a pseudo-random number generator (PRNG) with the signature *S*(r,BAL), generated using S's private key.
- S then deterministically maps the first number generated by the PNRG into the identity of its gossip receiver R.
- R then verifies that i) the seed is a valid signature, ii) r is the current round, iii) the first number generated by the PRNG when seeded with *S*(r,BAL), maps to R, and iv) this is the first time that S has presented this seed value to R.
- If the tests pass, R accepts the gossip request from S.

Rational nodes follow peer selection protocol

- Peer selection limits the number of connections any node can make to a small constant, preventing Byzantine nodes from abusing the system through the creation of arbitrarily many legitimate connections.
- Each seed contains only the round and type of exchange (Balanced or Optimistic). A node can thus generate only two seeds per round, resulting in two communication partners generated from the deterministic PRNG.
  - nodes keep track of the other nodes that have contacted it in the current round

Rational Behaviour - History Exchanges

- S and R exchange histories, containing 3 messages
  1. S provides a hash of its history and the seed value
  2. R returns its current history
  3. S divulges its actual history to R (R validates with hash)

- Each briefcase message contains the ids of the two parties, the seed uniquely identifying this exchange, the encrypted updates, and an update list stating what the encrypted contents should be.
- Sender signs the *briefcase thereby promising that the encrypted contents* are genuine and match the update list.

Rational Nodes do not over-/under-report

Problem: What if a rational node lies about its history?

- A rational node will not under-report in a balanced exchange
  - Limits the exchange to fewer updates
  - May receive an update that it already holds but did not report
- A rational node over-reports an update by claiming to possess an update that it does not have
  - Goal is to gain more utility in an exchange.
  - However, to do this, it needs to send a briefcase message in which the claimed contents are different from the encrypted contents – a proof of misbehaviour (POM).

Rational nodes do not send garbage

Problem: What if a rational node places fake or garbage data in briefcase messages?

- A rational node does not send invalid key response messages as including updates that do not match the update list in the signed briefcase represent a POM that will lead to the rational node’s eviction.
- Rational nodes never place fake or garbage data in briefcase messages.
- Rational nodes report malformed briefcases to the broadcaster as it is in their interest to do so.

Rational Behaviour - Key Exchange

Problem: What if a rational node chooses not to send the key or sends an invalid key?

- A rational node does not send invalid key response messages.
  - Sending an invalid key will generate a POM
  - Ignoring a partner's key requests saves the cost of sending a symmetric key, but has been shown using the credible threat mechanism and Nash Equilibria to not be in the node's interest.
- Therefore, a rational node eventually responds with a valid key to key request messages.

Other secure gossiping sytems

- Brahms Byzantine-Resilient Gossiping [Brahms]
  - Supports partial views
  - Analysis of the its byzantine robustness

- Secure peer sampling service (SPS) [Jesi10]
  - Identify and blacklist potentially malicious nodes
    - Goal is different to BAR Gossip which prevents attacks
  - Uses certificates to identify nodes
  - Uses *prestige* from social network analysis theory to identify misbehaving nodes
    - Remove misbehaving nodes from the system
  - Prestige is calculated using the in-degree of a node
    - Exploratory gossip msgs used to build up a prestige table
    - A whitelist of nodes believed to be 'good' is also maintained

Summary

- Naive assumptions about P2P network environments can lead to the construction of systems that:
  - do not work due to connectivity problems
  - are vulnerable to attack
  - do not exploit extra capabilities of ‘good’ nodes and/or avoid ‘bad’ nodes
  - do not handle network congestion.

References

- Security and privacy issues in P2P streaming systems: A survey, In Journal of P2P Networked Applications, 2010.
- BAR Gossip, Li et Al, OSDI, 2006.
- BEHAVE RFC, Audet, F., Jennings, C.: Network address translation (nat) behavioral requirements for unicast udp (2007) IETF
- Roverso et al., Natcracker: Nat combinations matter, ICCC, 2009.
- Gummadi, K.P. et al, King: Estimating latency between arbitrary internet end hosts. In SIGCOMM Internet Measurement Workshop (2002)
- Gian Paolo Jesi, Alberto Montresor, and Maarten van Steen. Secure Peer Sampling. Elsevier Computer Networks - Special Issue on Collaborative Peer-to-Peer Systems, 54(12):2086-2098, 2010.

ROYAL INSTITUTE OF TECHNOLOGY

References

- [DRUM] Gal Badishi, Idit Keidar, and Amir Sasson, Exposing and Eliminating Vulnerabilities to Denial of Service Attacks in Secure Gossip-Based Multicast, TDSC, 2006.
- Bortnikov et al, Brahms: Byzantine resilient random membership sampling, Computer Nets, 2009
- Albert-László Barabási, Linked: The new science of networks, 2002.
- Niazi and Dowling, Usurp: Distributed NAT Traversal for Overlay Networks, DAIS 2011.