CE 817 - Advanced Network Security
Denial of Service Attacks II

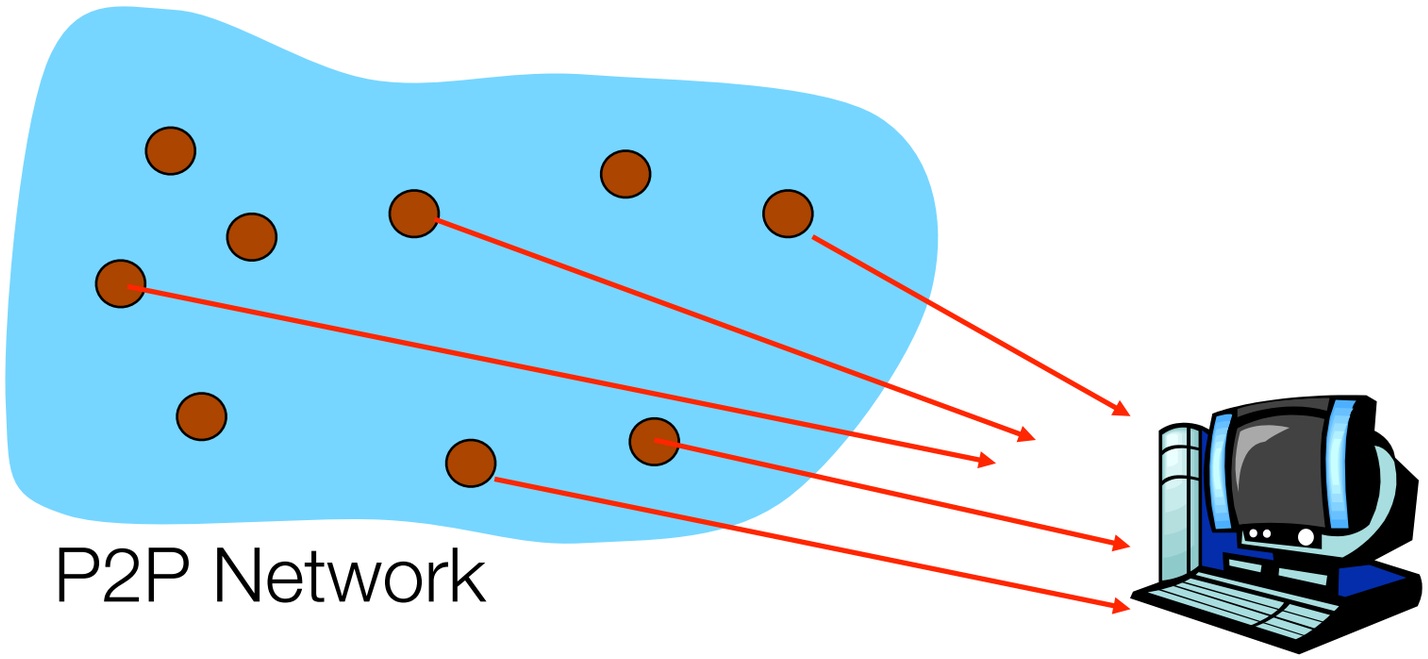
Lecture 8

Mehdi Kharrazi
Department of Computer Engineering
Sharif University of Technology

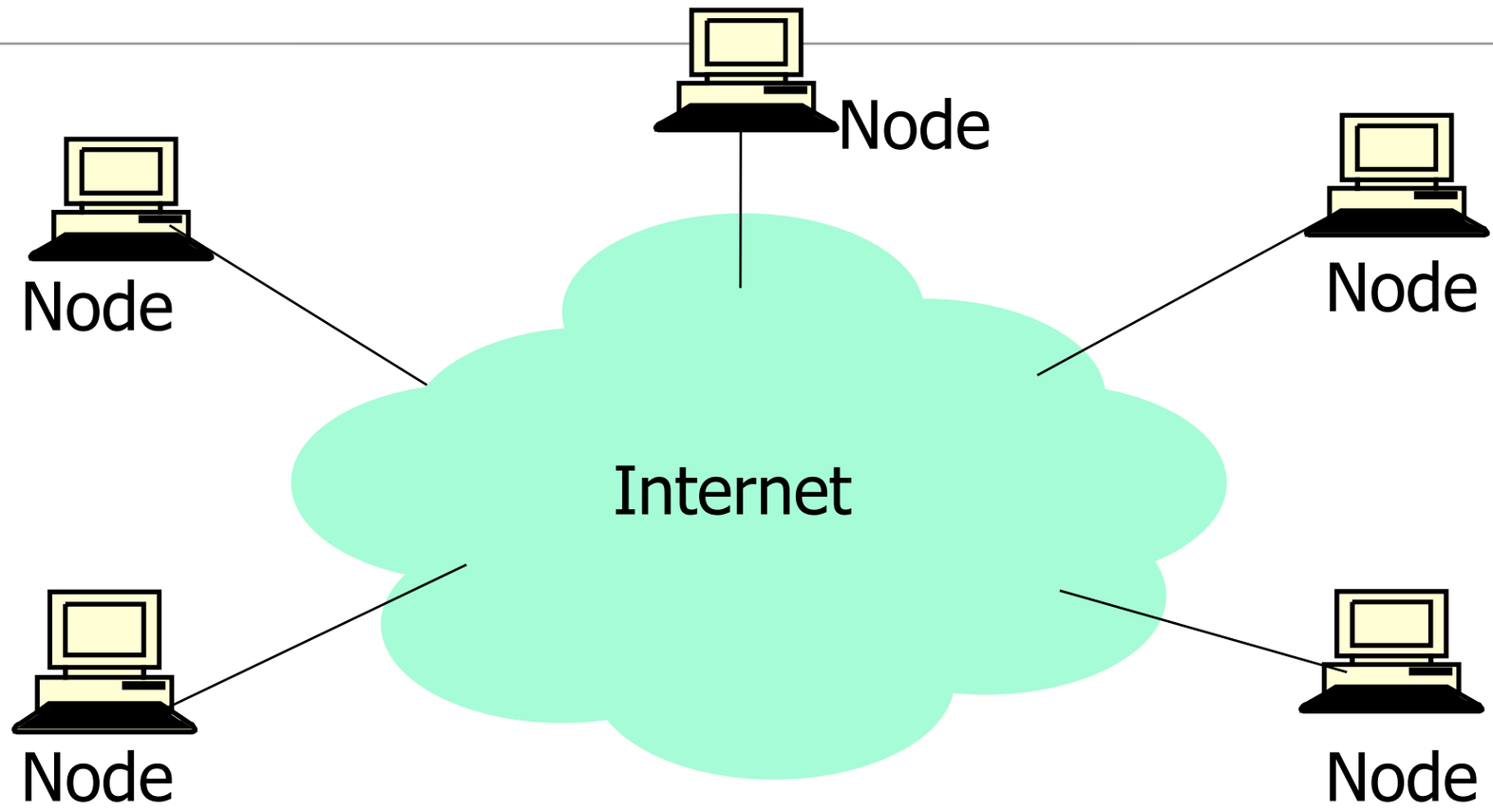
Acknowledgments: Some of the slides are fully or partially obtained from other sources. Reference is noted on the bottom of each slide, when the content is fully obtained from another source. Otherwise a full list of references is provided on the last slide.

Exploiting P2P Systems for DDoS Attacks

- SYN-Cookies can be used to stop SYN attacks
- How about if we open a full connection?



What is a P2P System?



- A distributed system architecture:
  - No centralized control
  - Nodes are symmetric in function
- Large number of unreliable nodes

The Promise of P2P Computing

- High capacity through parallelism:
  - Many disks
  - Many network connections
  - Many CPUs
- Reliability:
  - Many replicas
  - Geographic distribution
- Automatic configuration
- Useful in public and proprietary settings

Quick Review of DHT (Distributed Hash Table)

What Is a DHT?

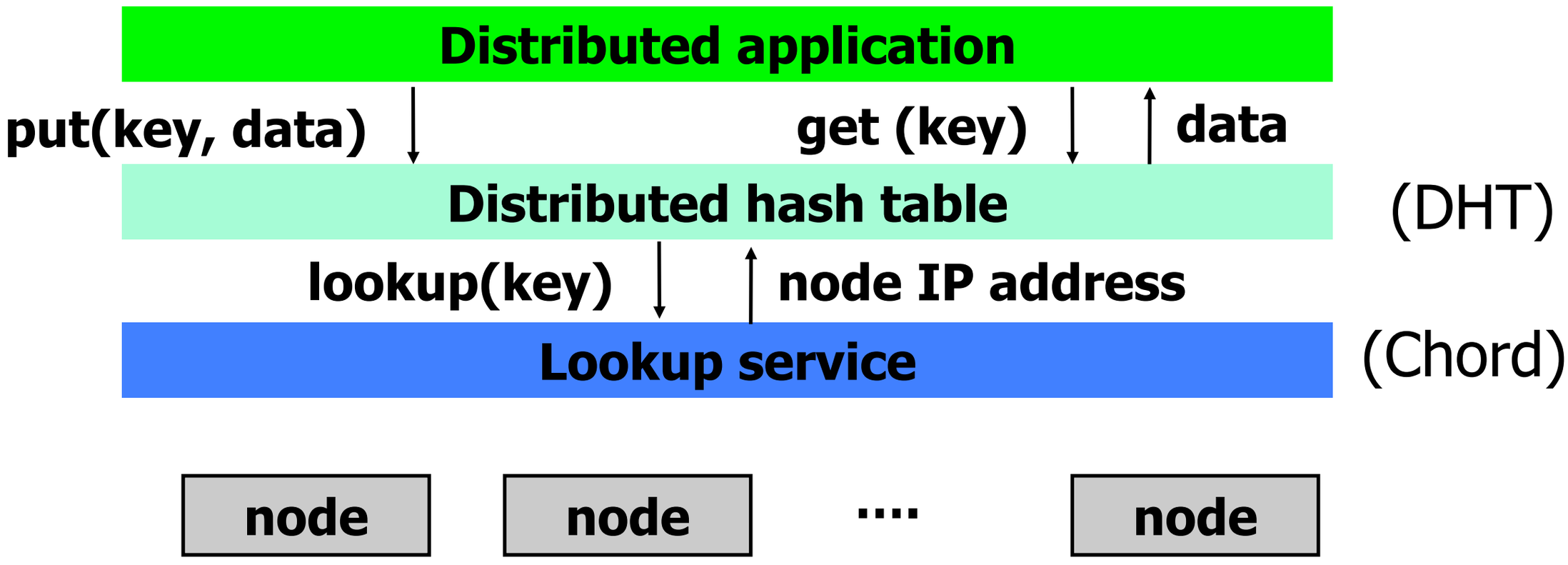
- Single-node hash table:
  - key = Hash(name)
  - put(key, value)
  - get(key) -> value
- How do I do this across millions of hosts on the Internet?
  - Distributed Hash Table

What Is a DHT?

- Distributed Hash Table:
- key = Hash(data)
- lookup(key) -> IP address (Chord)
- send-RPC(IP address, PUT, key, value)
- send-RPC(IP address, GET, key) -> value

- Possibly a first step towards truly large-scale distributed systems
  - a data block in a global file system

DHT Factoring



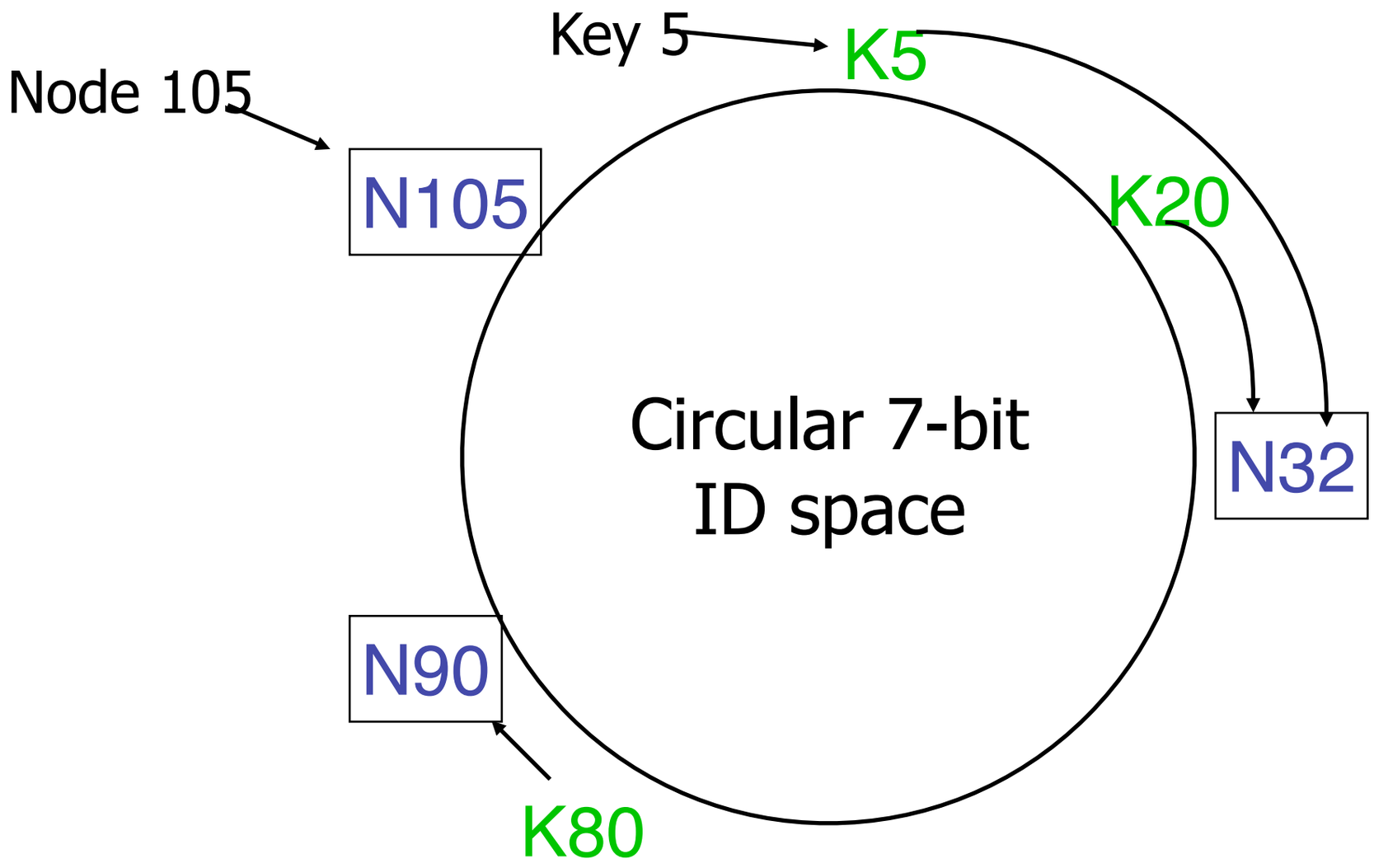
- Application may be distributed over many nodes
- DHT distributes data storage over many nodes

Chord IDs

- Key identifier = SHA-1(data)
- Node identifier = SHA-1(IP address)
- SHA-1 distributes both uniformly

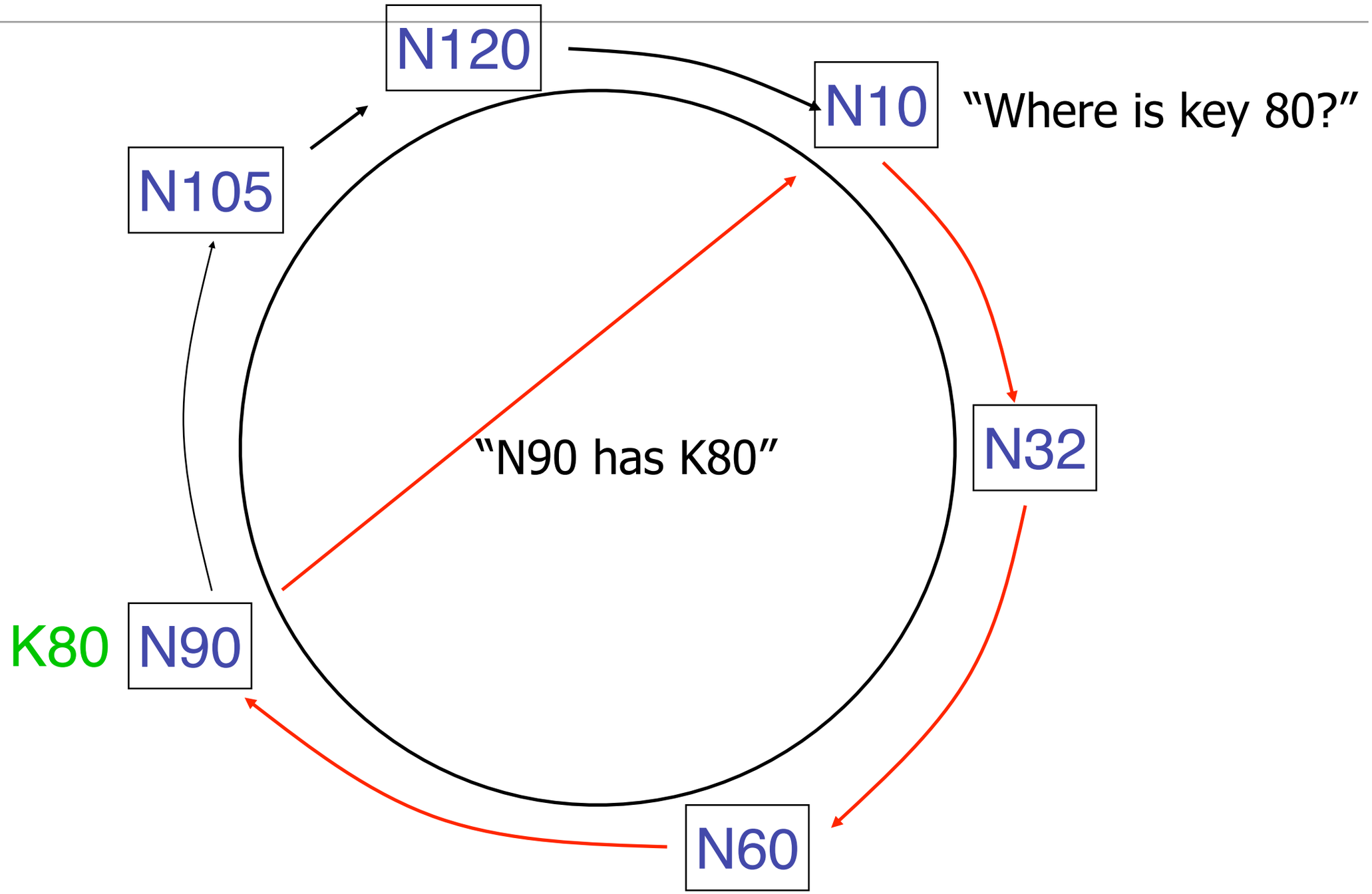
- How to map key IDs to node IDs?

Consistent Hashing [Karger 97]



A key is stored at its successor: node with next higher ID

Basic Lookup



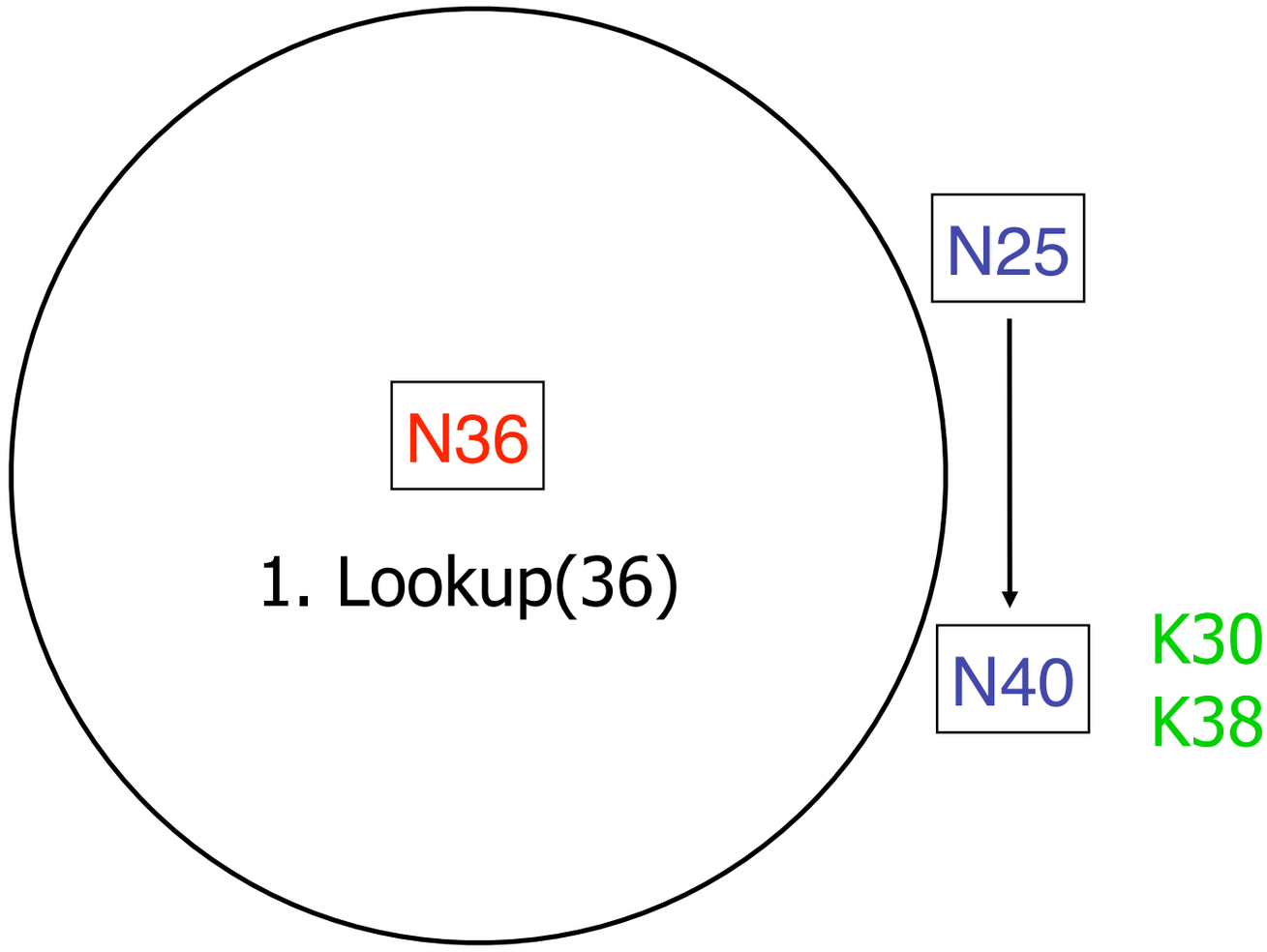
Simple lookup algorithm

- Lookup(my-id, key-id)

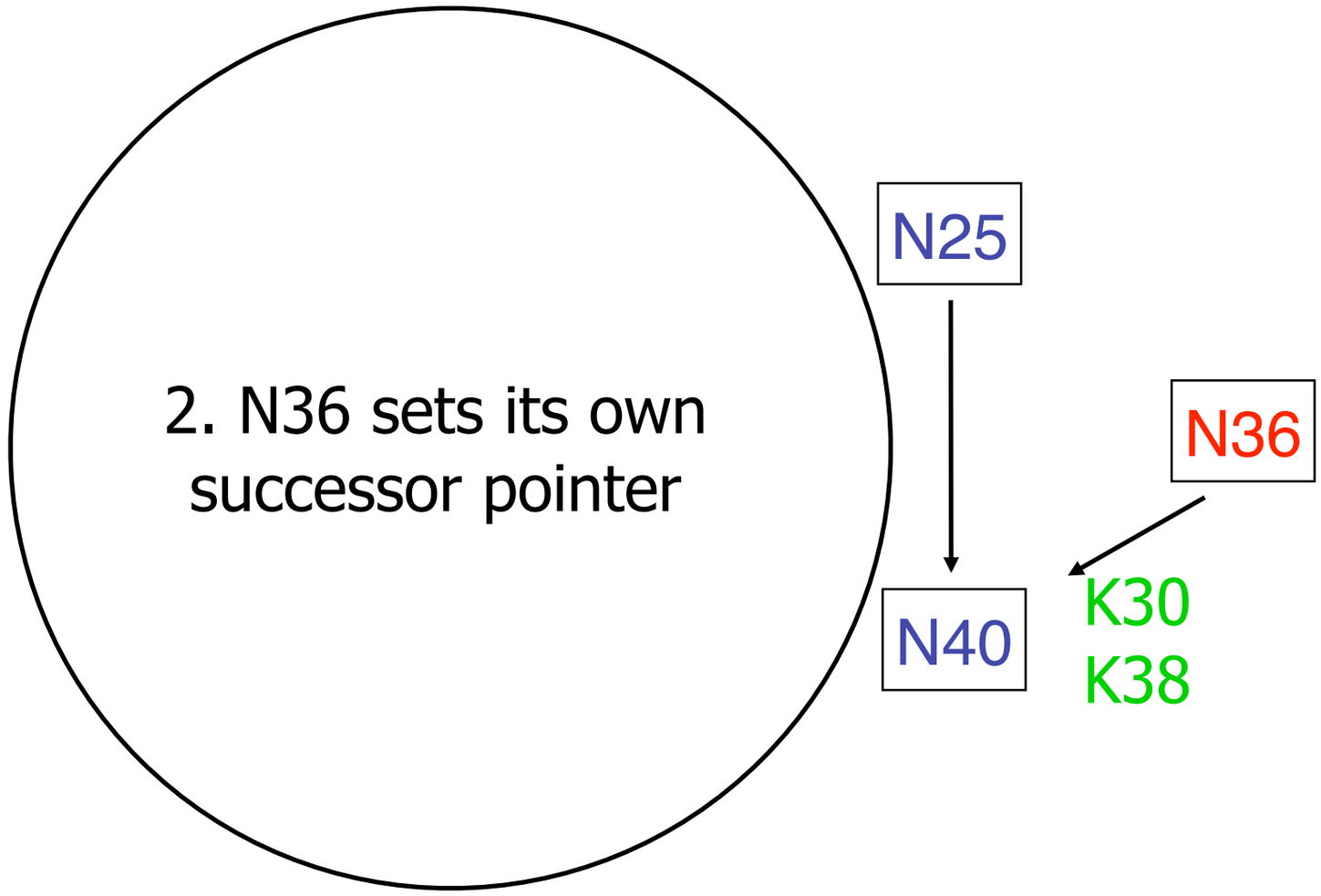
```
n = my successor
if my-id < n < key-id
        call Lookup(id) on node n   // next hop
else
        return my successor         // done
```

- Correctness depends only on successors
- There are optimizations to make lookup faster, but we will skip it

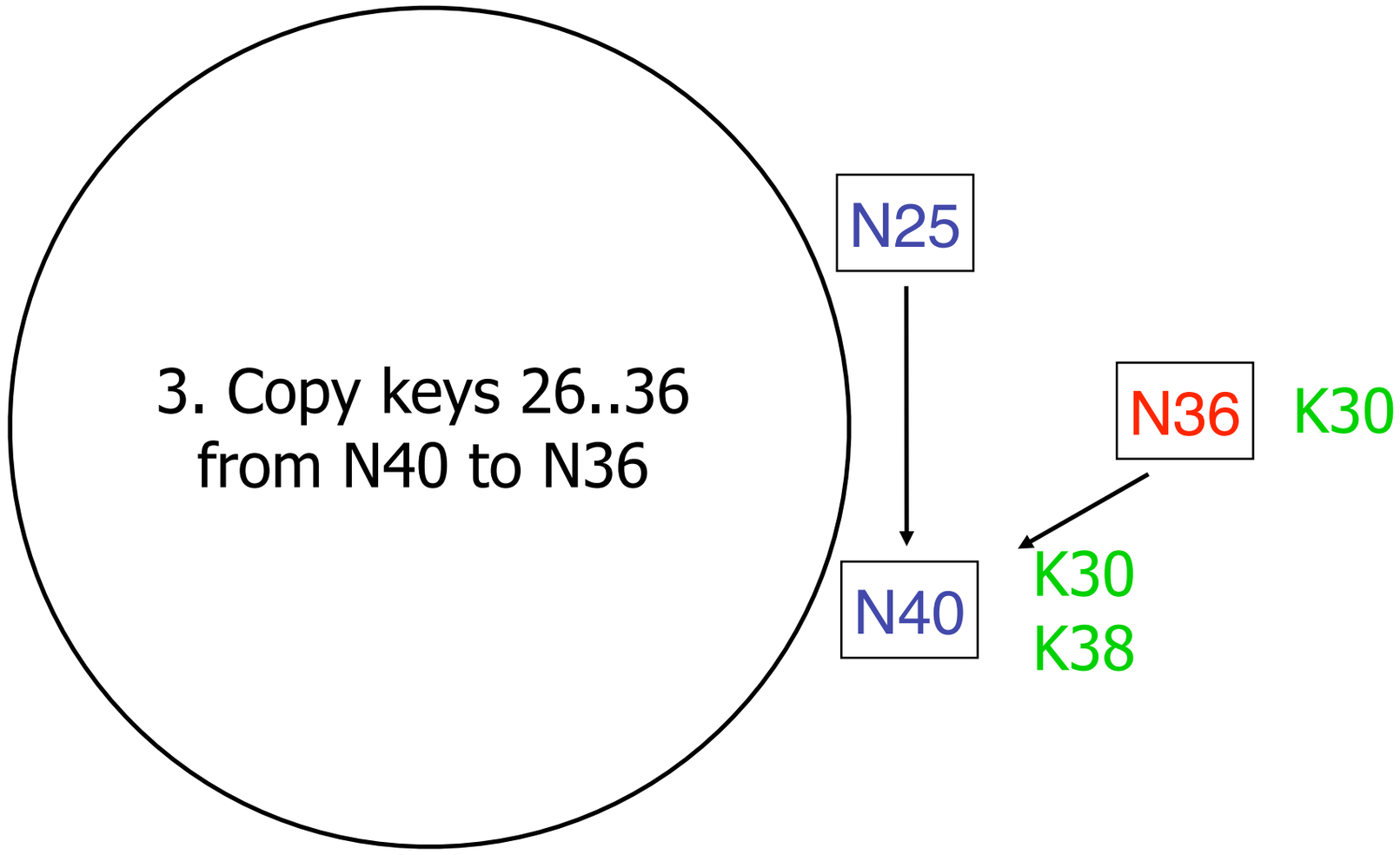
Joining: Linked List Insert



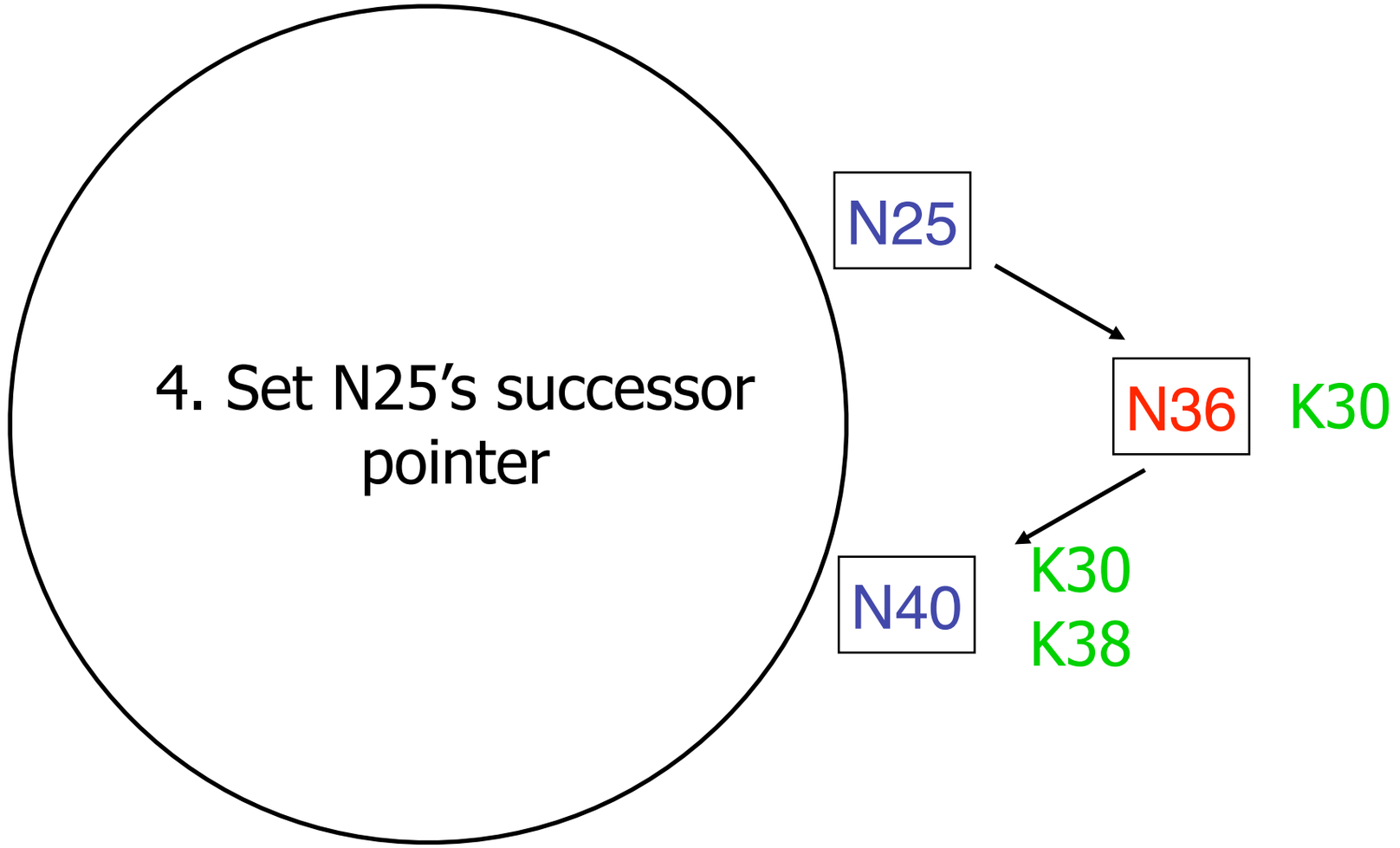
Join (2)



Join (3)



Join (4)



N25 checks if N40 has a predecessor
Predecessor pointer allows link to new host

Back to the Attack

Overnet (eMule): DHT

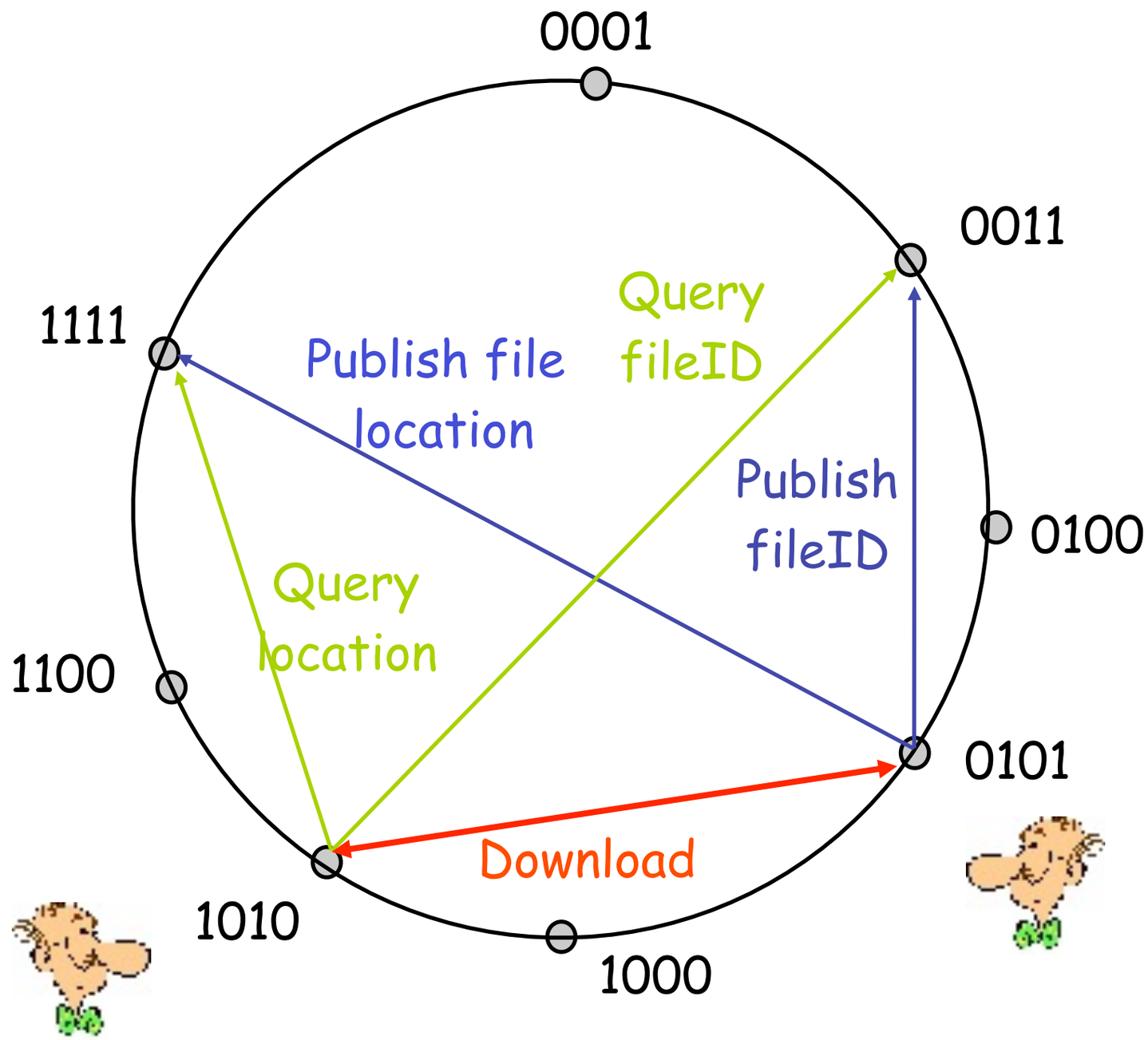
- DHT: (key, value) record stored in node closest to key

- Overnet records: (hash_title, version_id)
  (version_id, location)

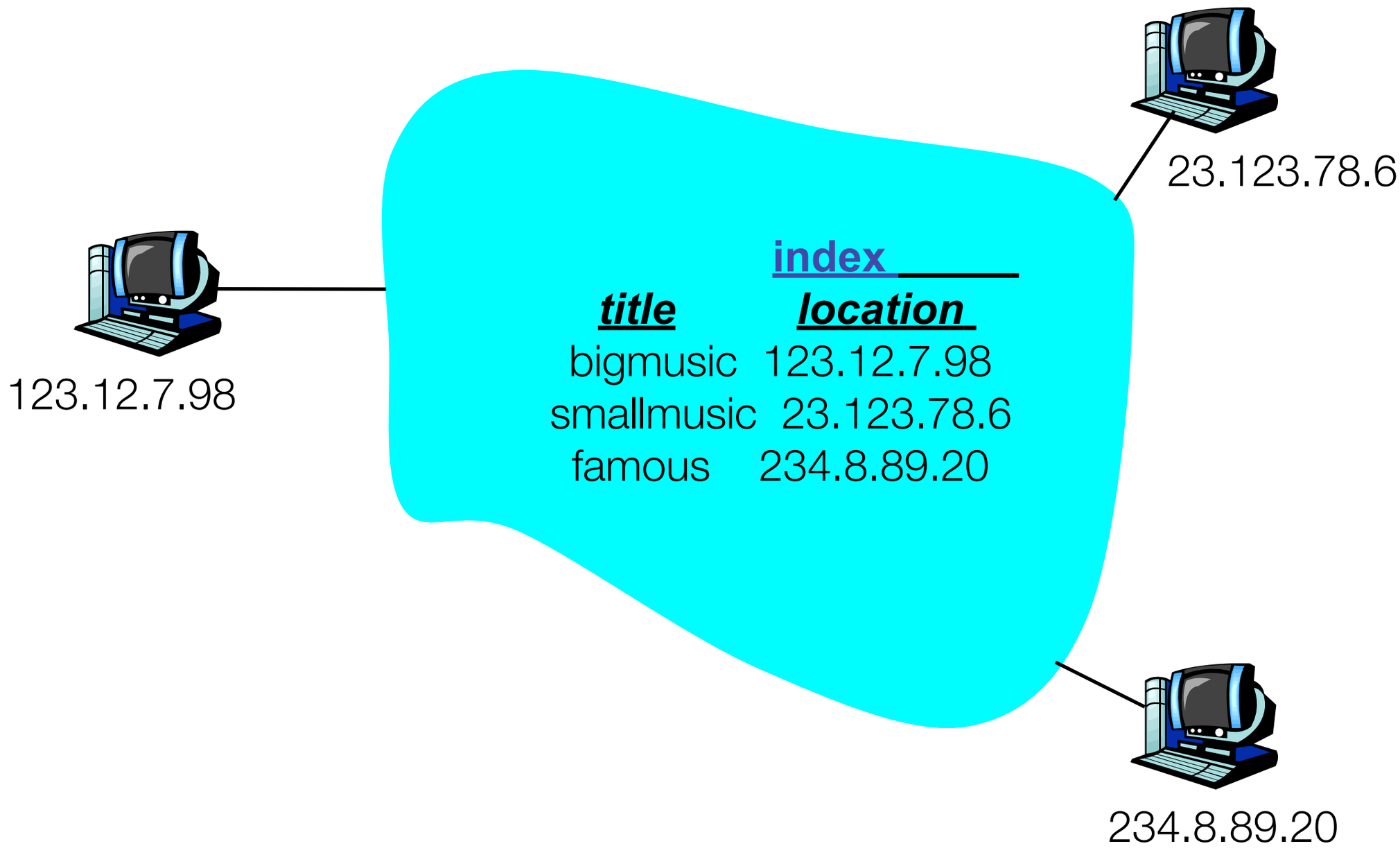
- First search hash_title, get version_id and metadata

- Then search version_id, get location

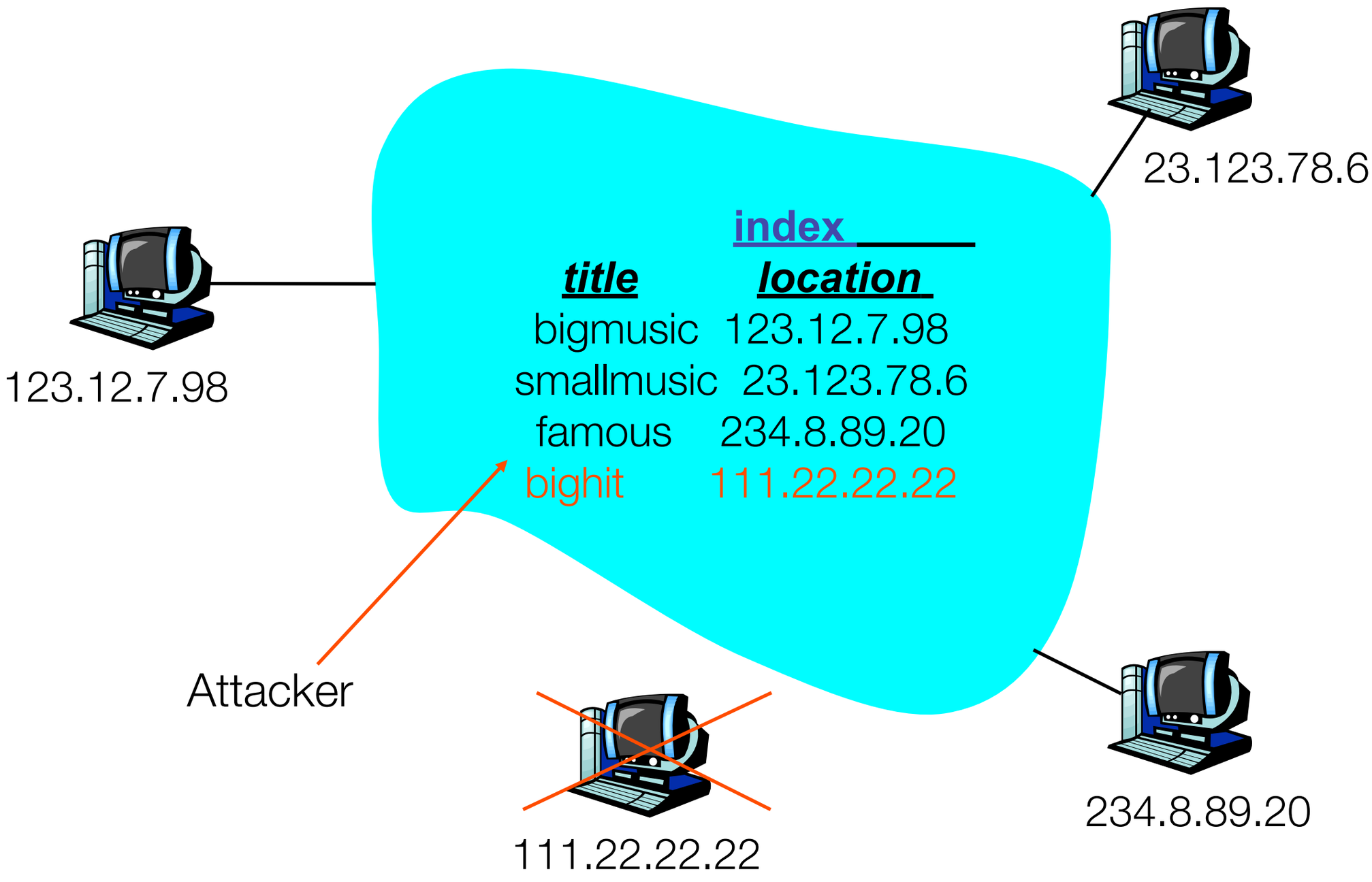
Overnet: DHT


0001
0011
1111
Publish file location
Query fileID
Publish fileID
0100
Query location
1100
0101
Download
1010
1000

Index Poisoning



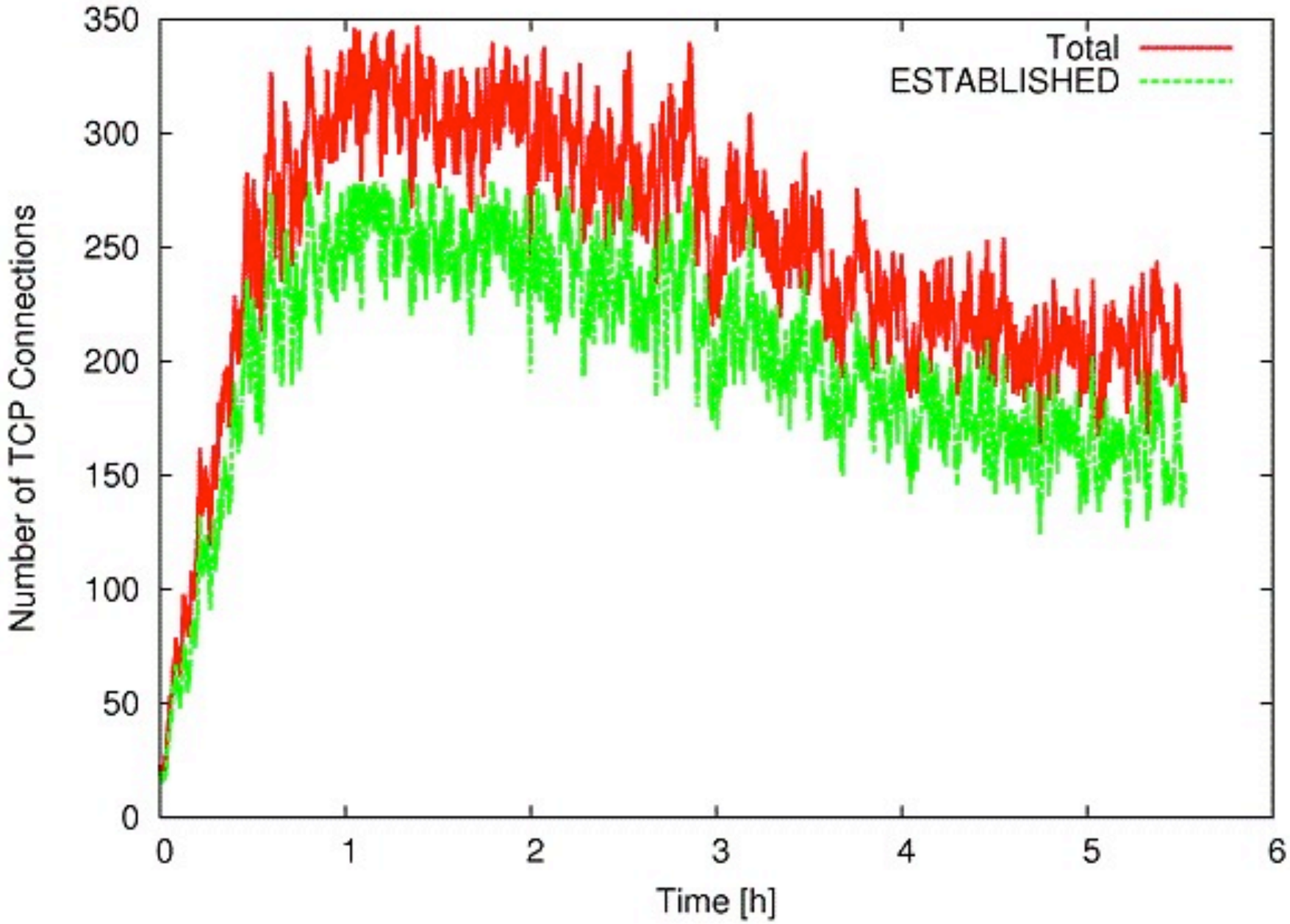
Index Poisoning



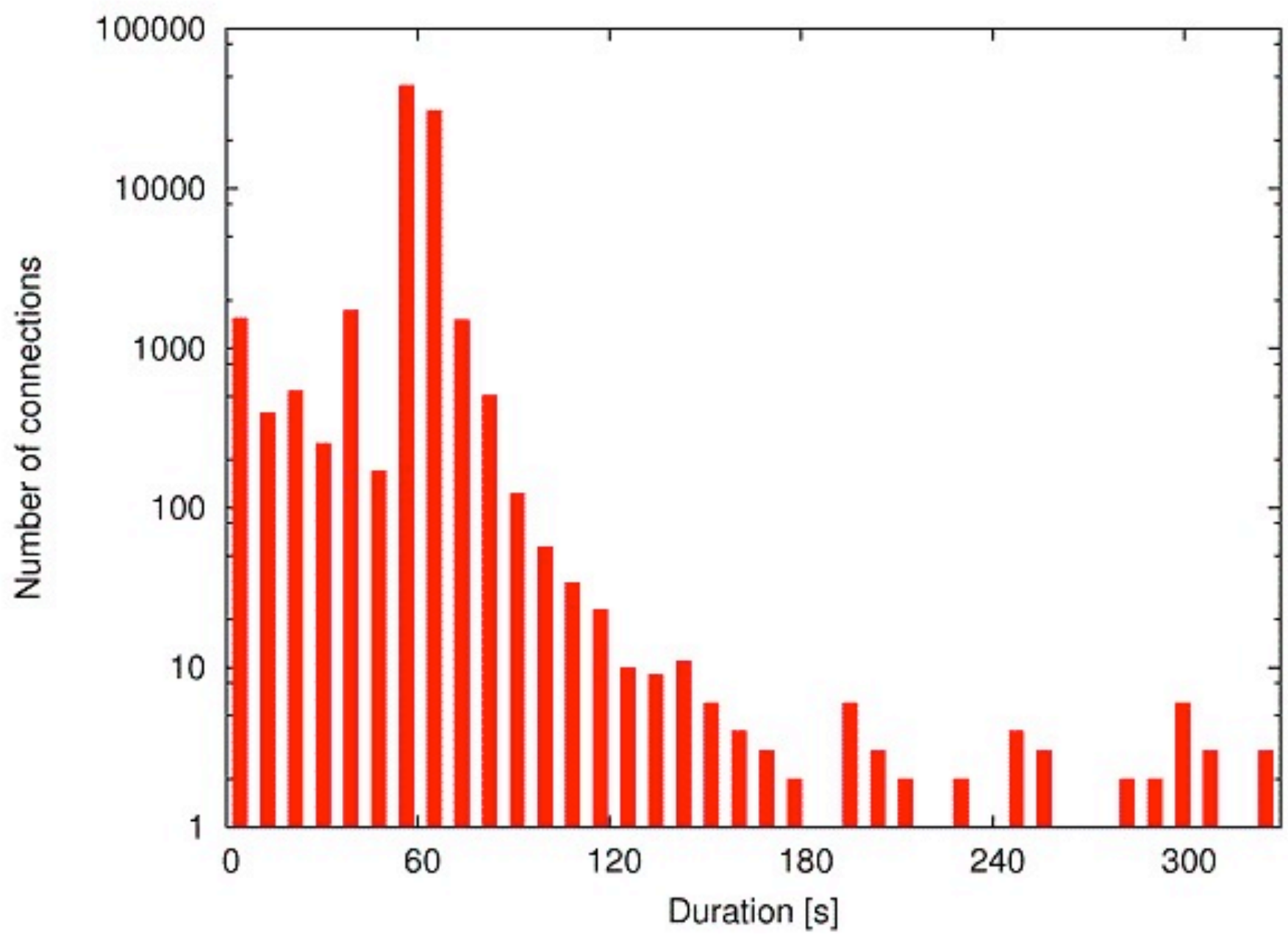
Poison distributed index

- Poison distributed index
  - target_IP = 128.238.X.X
  - Popular_title = music hit
  - Advertise records:
    (hash_title, ID_music_hit)
    - (ID_music_hit, target_IP)
- Users attempt to download popular title
  - Generates fully-open TCP connection's from user peers to target
  - Nastier than syn-flood DDoS attack
- Poisoned index only for 45 minutes, to limit effect on other users using the P2P network

Connections at target



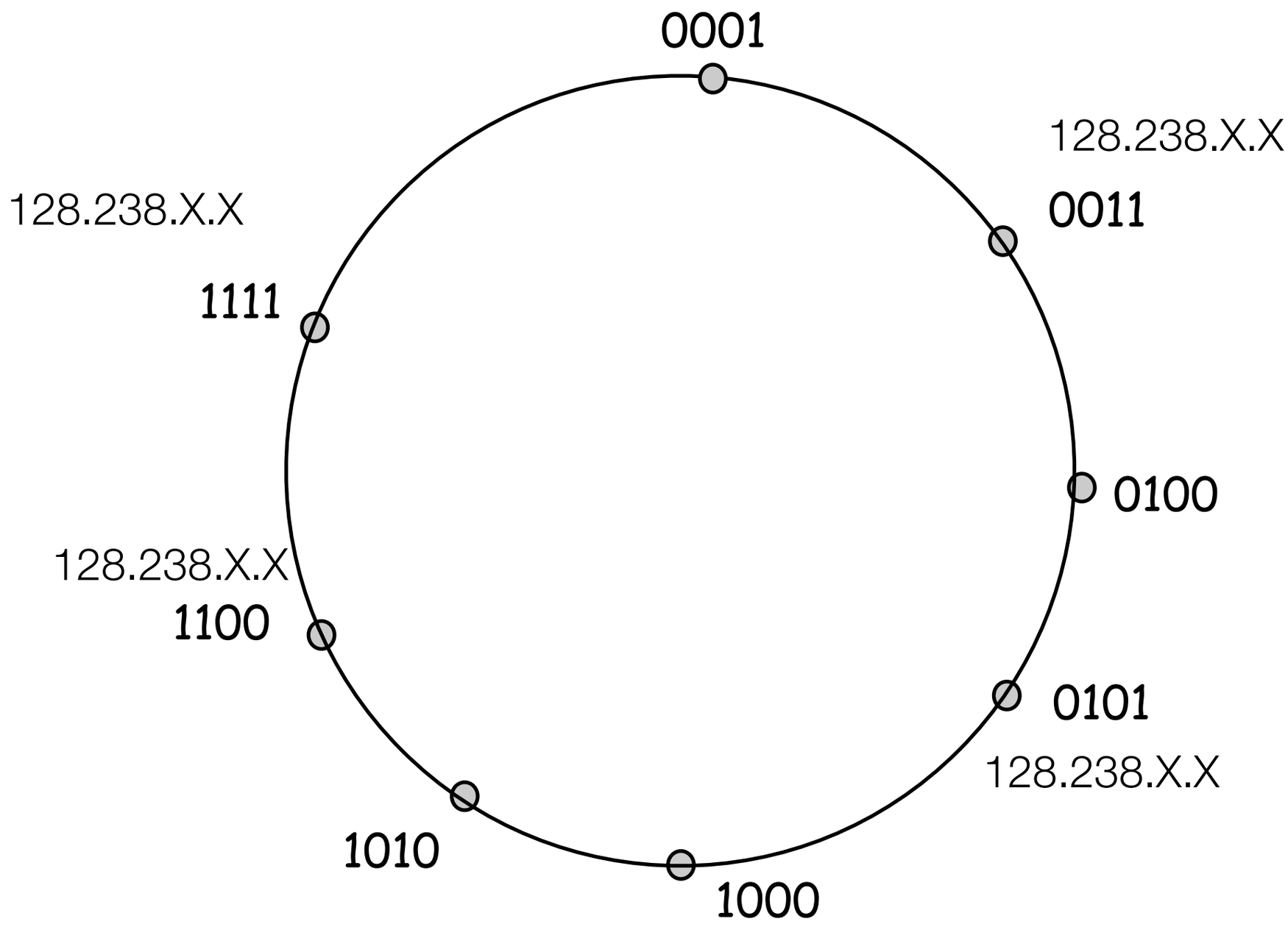
Connection Durations



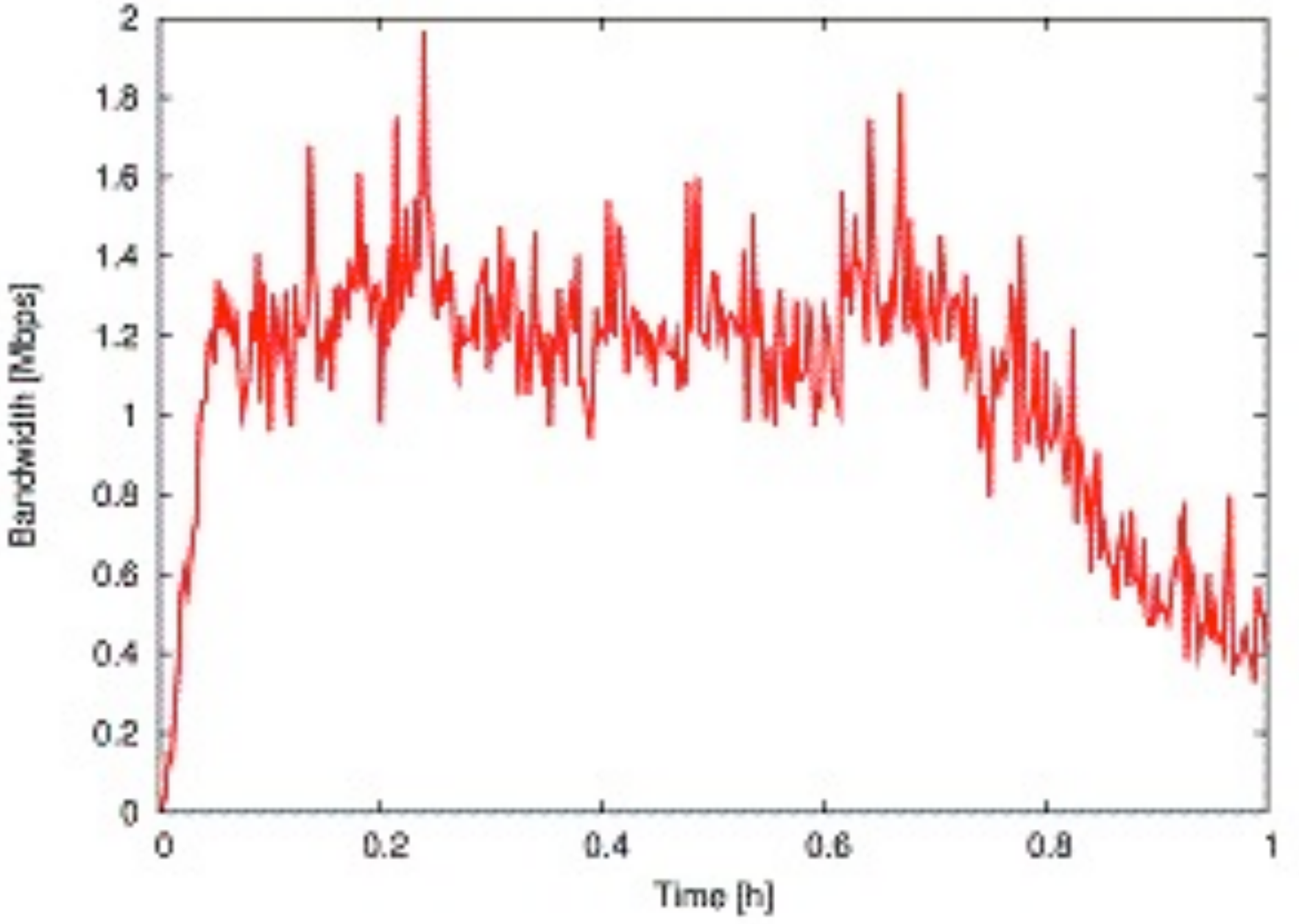
Poison overlay routing tables

- Poison overlay routing tables
  - Target = 123.238.X.X
  - Advertise existence to many nodes:
    (node_ID, target_IP)
    for many node_IDs
    - Many nodes will absorb advertisement
- Query, publish, overlay messages arrive at poisoned node
  - Some are directed at target

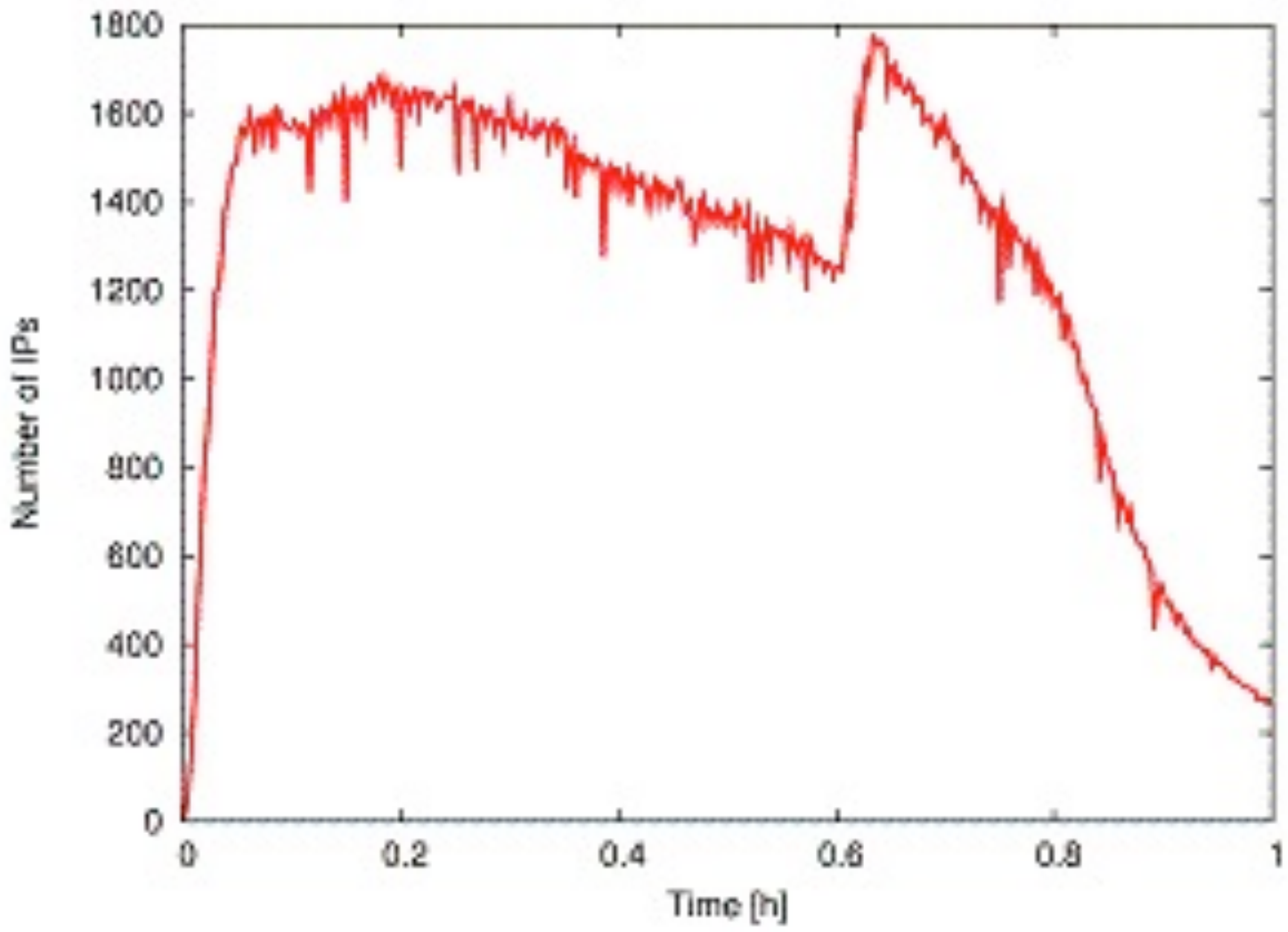
Overnet: Route Poisening



Bandwidth generated at target (UDP Traffic)



of attack sources



Low-Rate TCP-Targeted Denial of Service Attacks (The Shrew vs. the Mice and Elephants), A. Kuzmanovic and E. Knightly, in Proceedings of ACM SIGCOMM 2003, Karlsruhe, Germany, August 2003.

Background

- Traditional view of DoS attacks
  - Attacker consumes resources and denies service to legitimate users
    - Ex. traffic floods, DDoS
    - Result: TCP backs off
  - Observe: statistical anomalies that are relatively easily detectable
    - Due to attacker's high rate

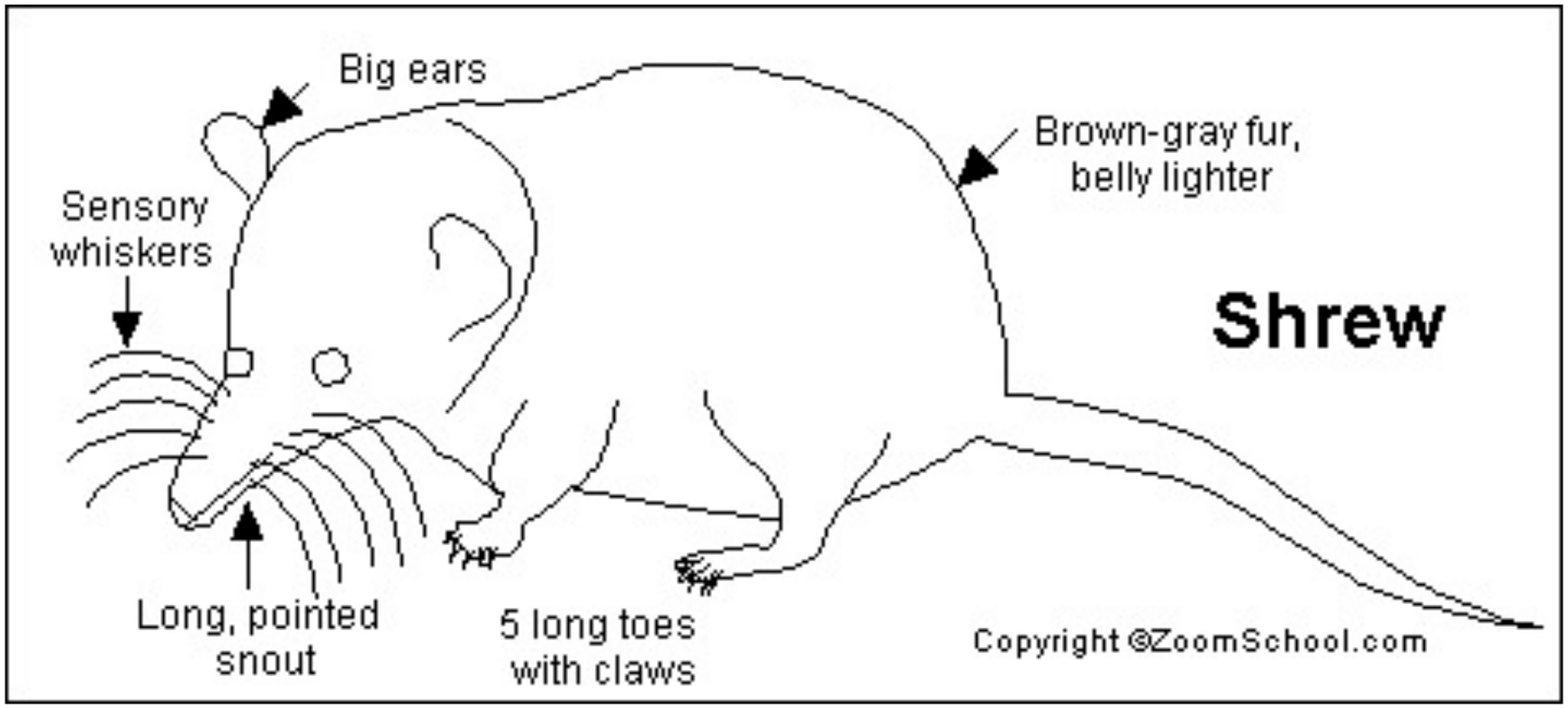
Thesis: TCP is Vulnerable to Low-rate Attacks

- Shrew: low-rate TCP-targeted attacks
- Elude detection by counter-DoS mechanisms
- Able to severely deny service to legitimate users

- Goals
  - Analyze TCP mechanisms that can be exploited by DoS attackers
  - Explore TCP frequency response to Shrews
  - Evaluate detection mechanisms
  - Analyze effectiveness of randomization strategies

- Methodology: modeling, simulations, Internet experiments

Shrew



- Very small but aggressive mammal that ferociously attacks and kills much larger animals with a venomous bite
- Reviewer 3: “only some shrews are venomous and the amount of venom in even the venomous species is very mild.”

TCP: a Dual Time-Scale Perspective

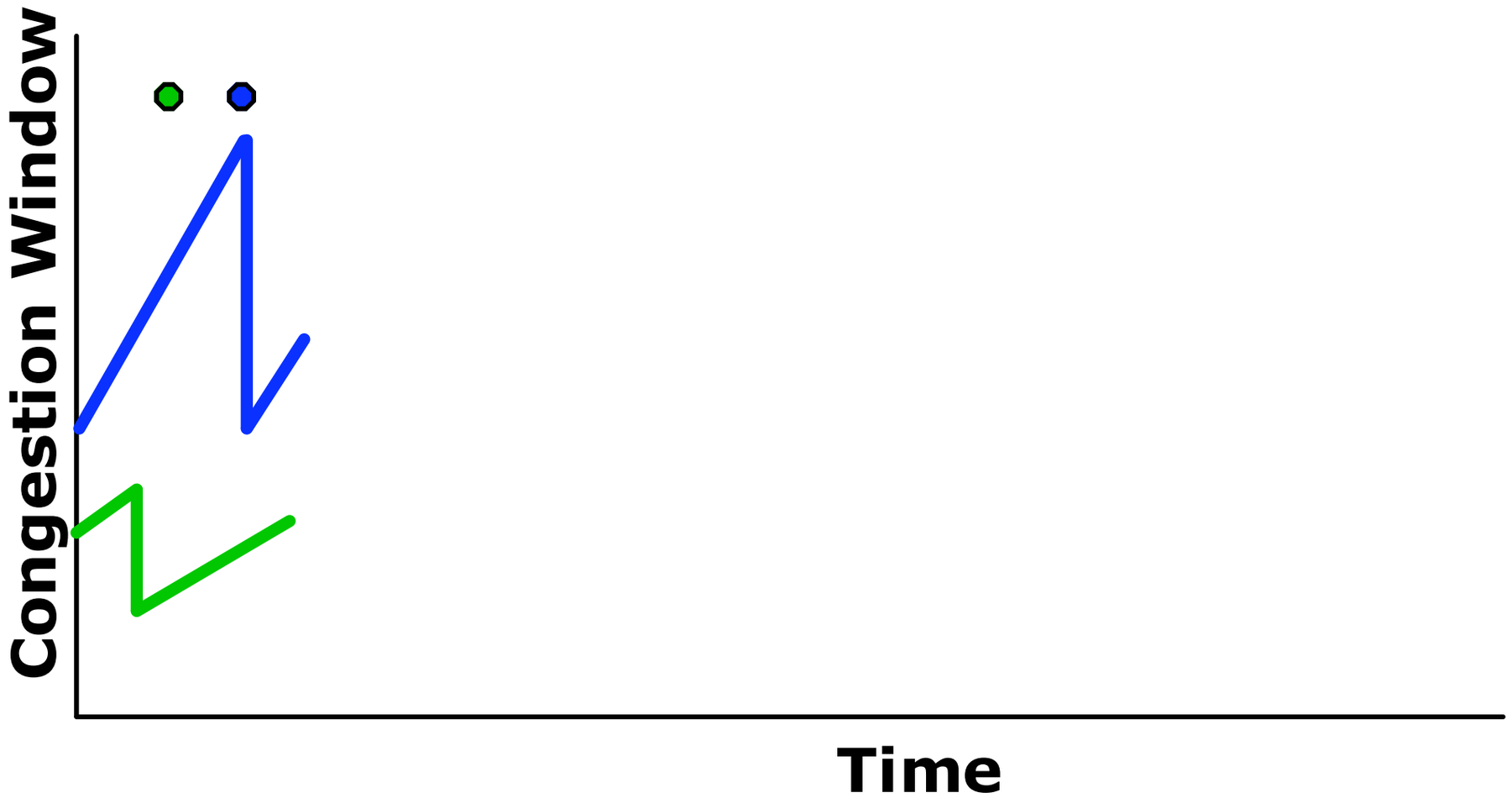
- Two time-scales fundamentally required
  - RTT time-scales (~10-100 ms)
    - AIMD control
  - RTO time-scales (RTO=SRTT+4*RTTVAR)
    - Avoid congestion collapse

- RTO must be lower bounded to avoid spurious retransmissions
  - [AllPax99] and RFC2988 recommends minRTO = 1 sec

Slow RTO time-scale mechanisms are a **key source of vulnerability** to low rate attacks

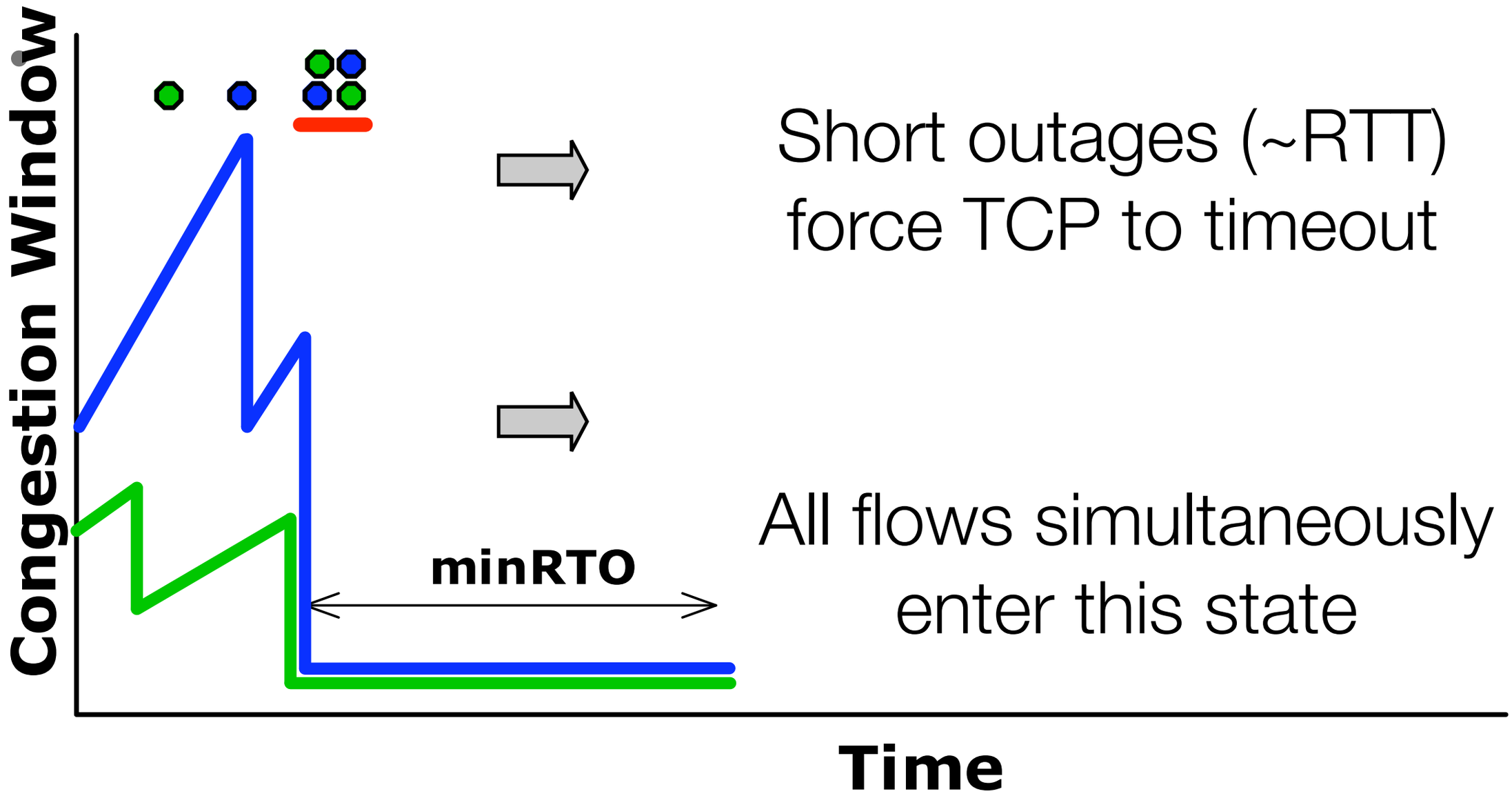
TCP Timeline

- Timeline of TCP congestion window
  - AIMD control



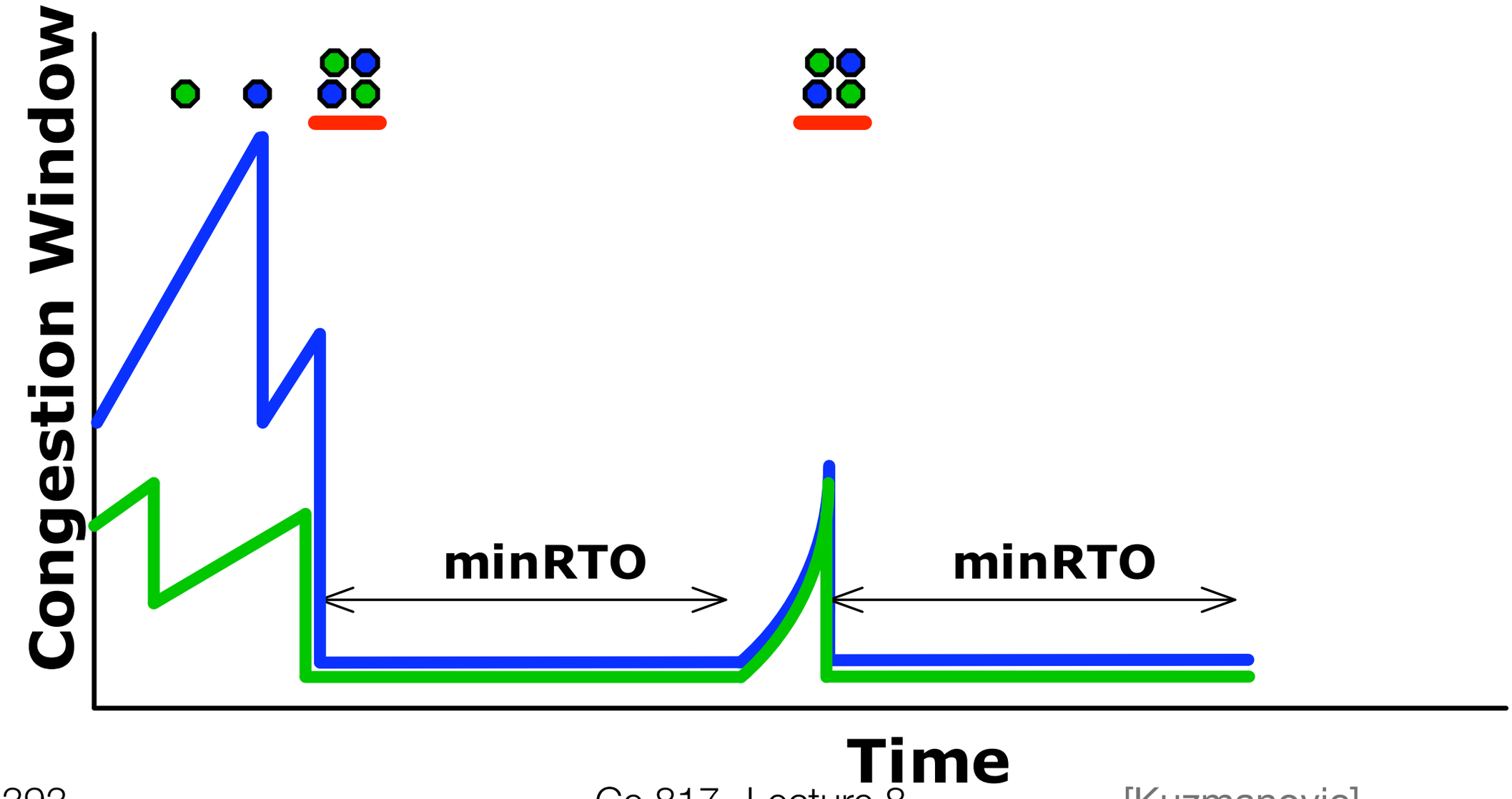
The Shrew Attack (1/3)

- Pulse-induced outage – multiple losses force TCP to enter RTO mechanism



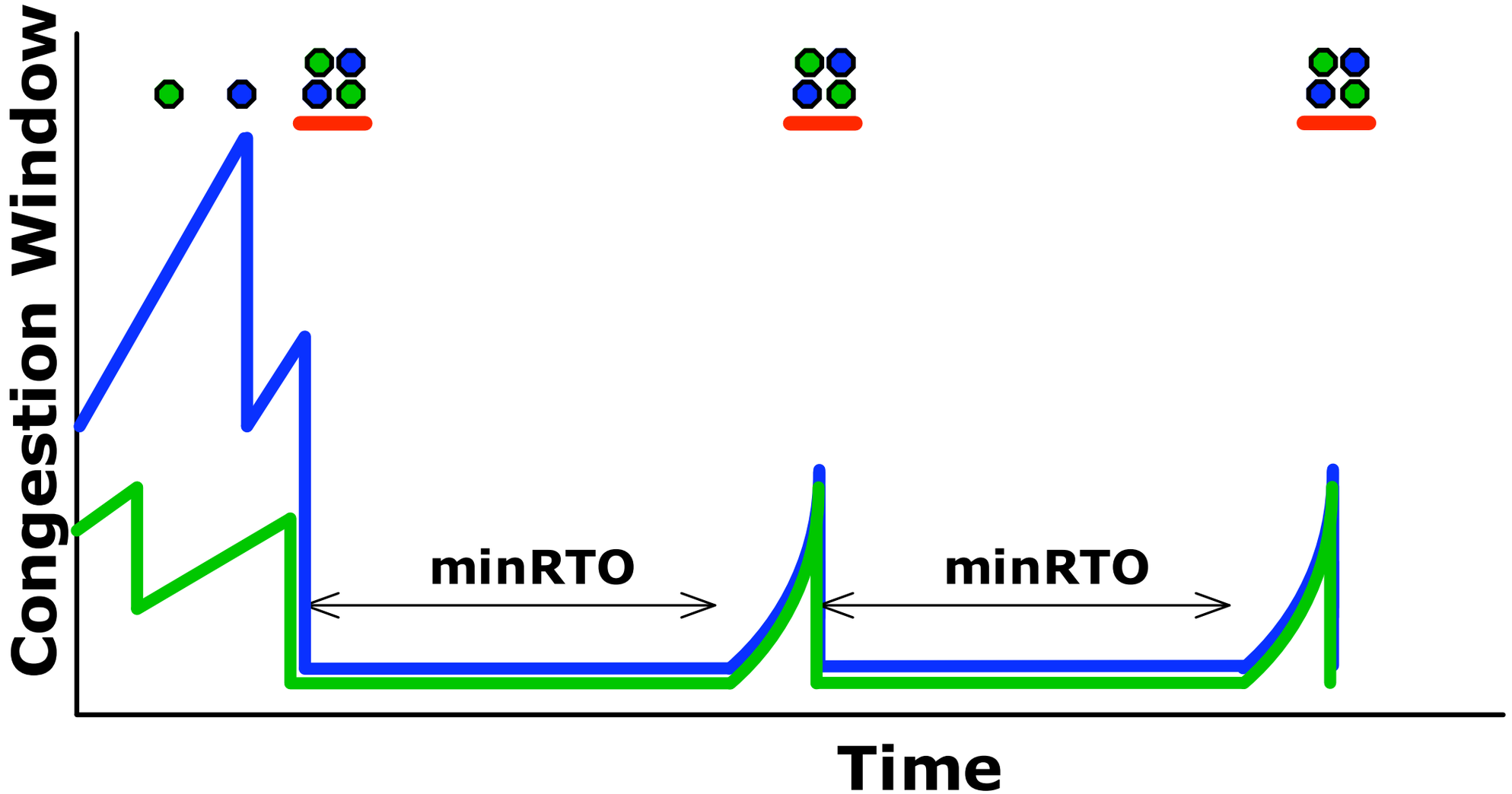
The Shrew Attack (2/3)

- When flows attempt to simultaneously exit timeout and enter slow-start…
- Shrew pulses again and forces flows synchronously back into timeout state



The Shrew Attack (3/3)

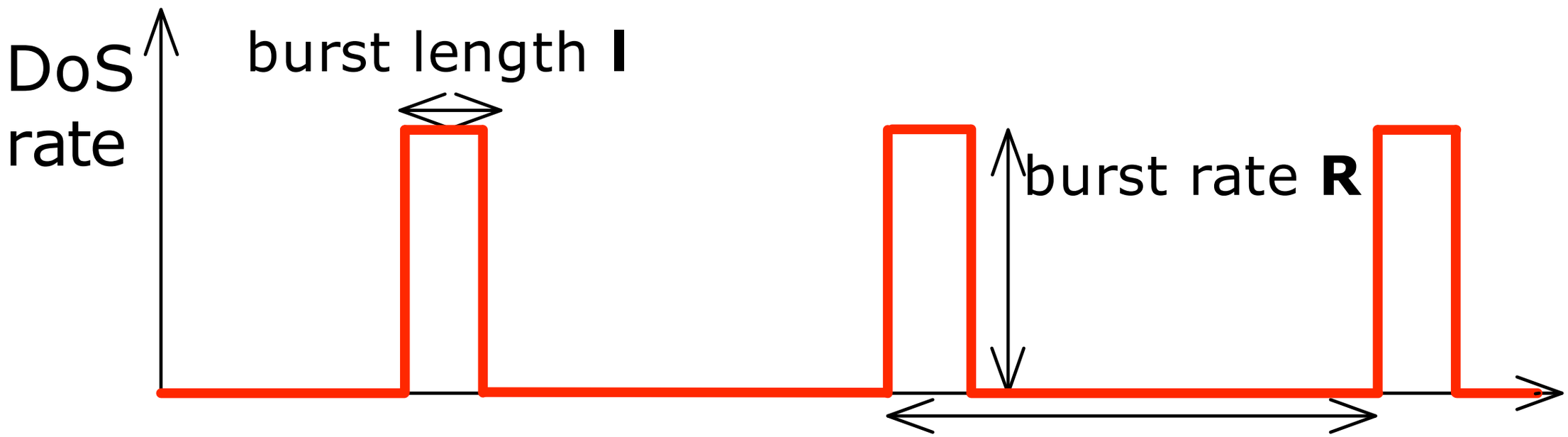
- Shrew periodically repeats pulse
  - RTT-time-scale outages inter-spaced on minRTO periods can deny service to TCP
  - Flows synchronize their state to the Shrew



Shrew Principles

- Shrews exploit protocol homogeneity and determinism
  - Protocols react in a pre-defined way
  - Tradeoff of vulnerability vs. predictability
- Periodic outages synchronize TCP flow states and deny their service
- Slow time scale protocol mechanisms enable low-rate attacks
  - Outages at RTO scale, pulses at RTT scale imply low average rate

Creating Outages in the Network

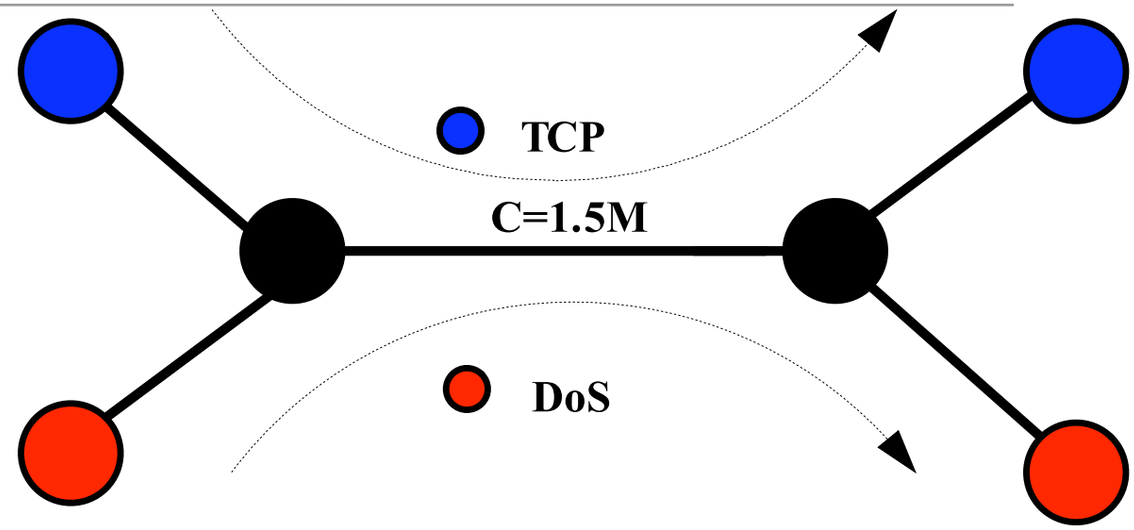


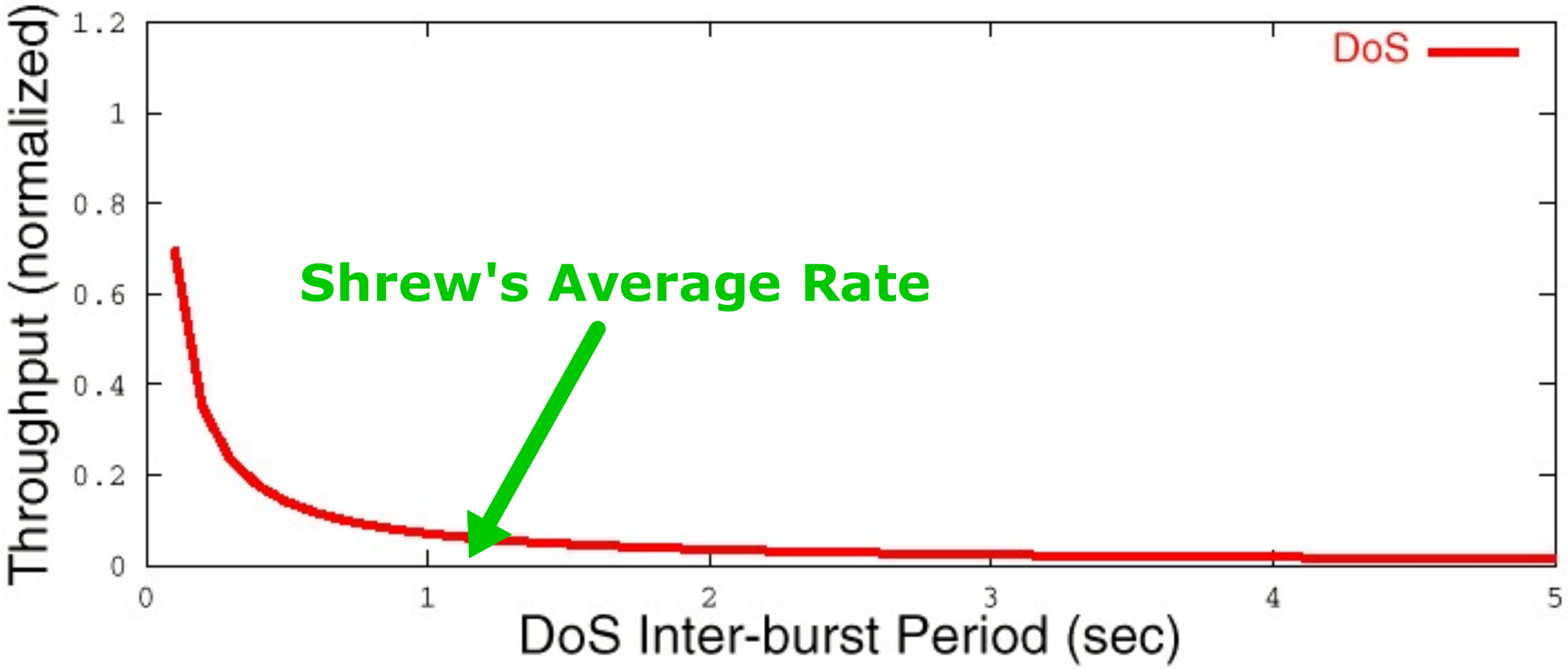
- Shrew: square-wave stream (l~RTT, T~minRTO)
- Low-rate “TCP friendly” DoS ⇒ hard to detect
  - Counter-DOS mechanisms tuned for high rate attacks
  - Detecting Shrews may have unacceptably many false alarms (due to legitimate bursty flows)

Simulation and Internet experiments

The Shrew in Action

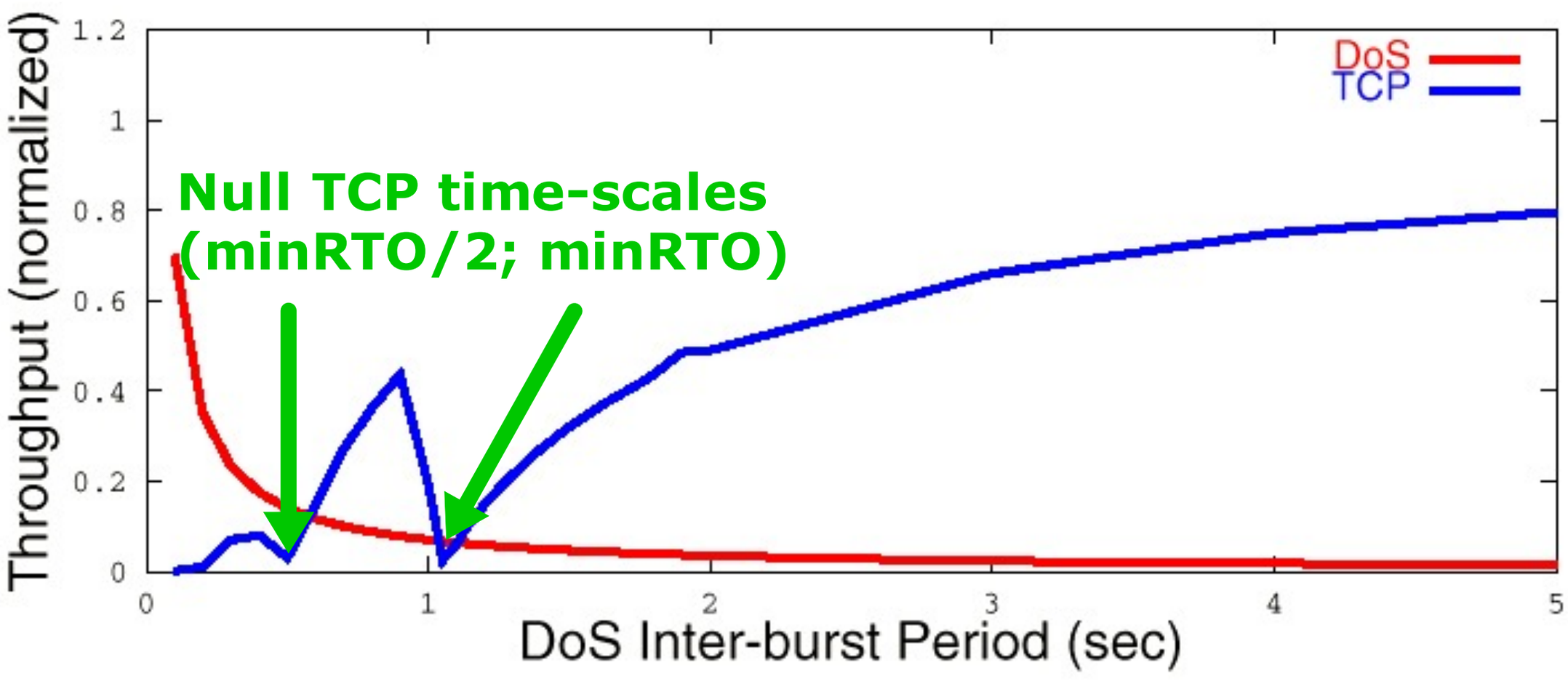
- How much is TCP throughput degraded?
- DoS stream:
  - R=C=1.5Mb/s; l=70ms (~TCP RTT)





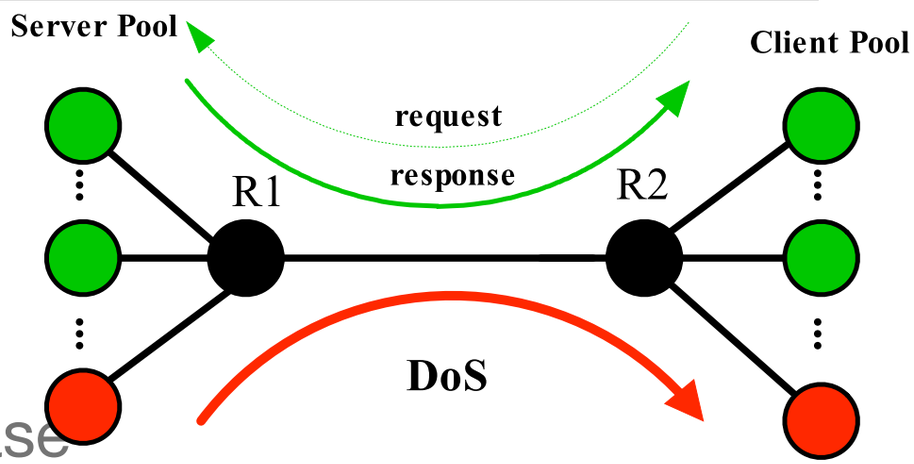
The Shrew in Action

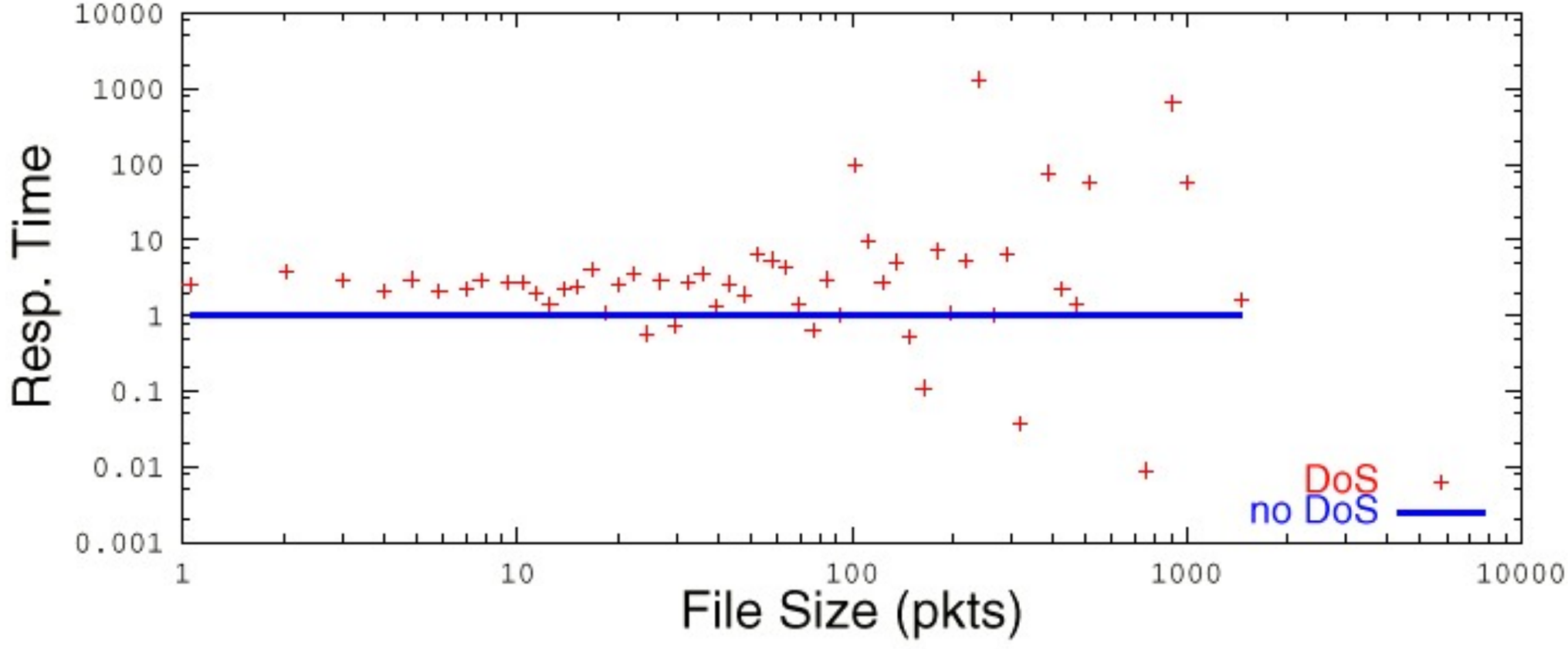
- Shrews induce null frequency near RTO
- Shrew has low average rate ≈ .08C
- Analytical model accurately predicts degradation



Shrews vs. Short-lived TCP Traffic

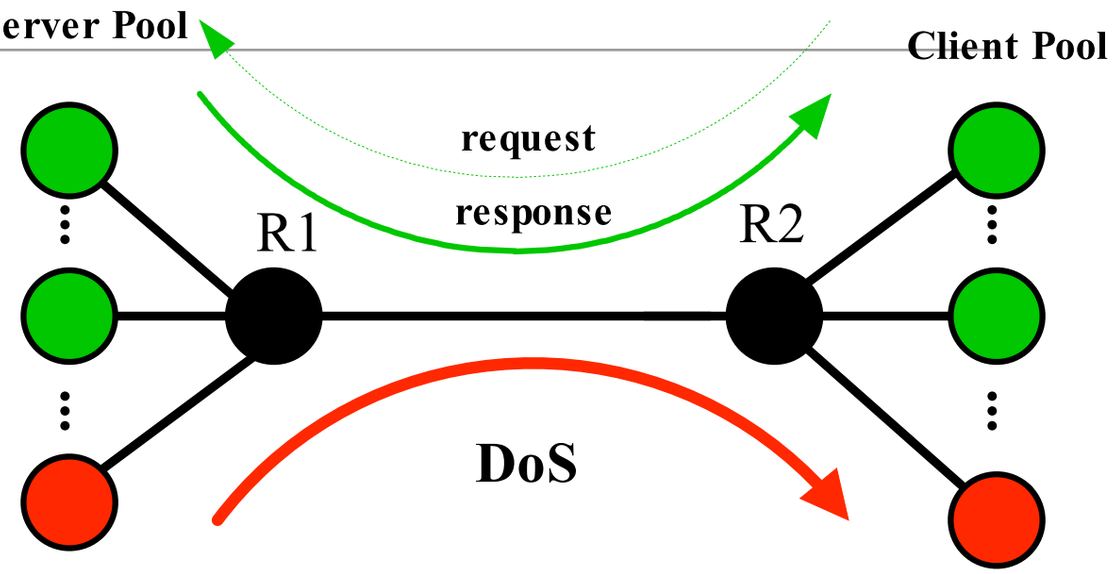
- Scenario: Web browsing [FGHW99]
- Average damage to
  - a mouse (<100pkts) = 400% delay increase
  - an elephant (>100pkts) =24500% delay increase

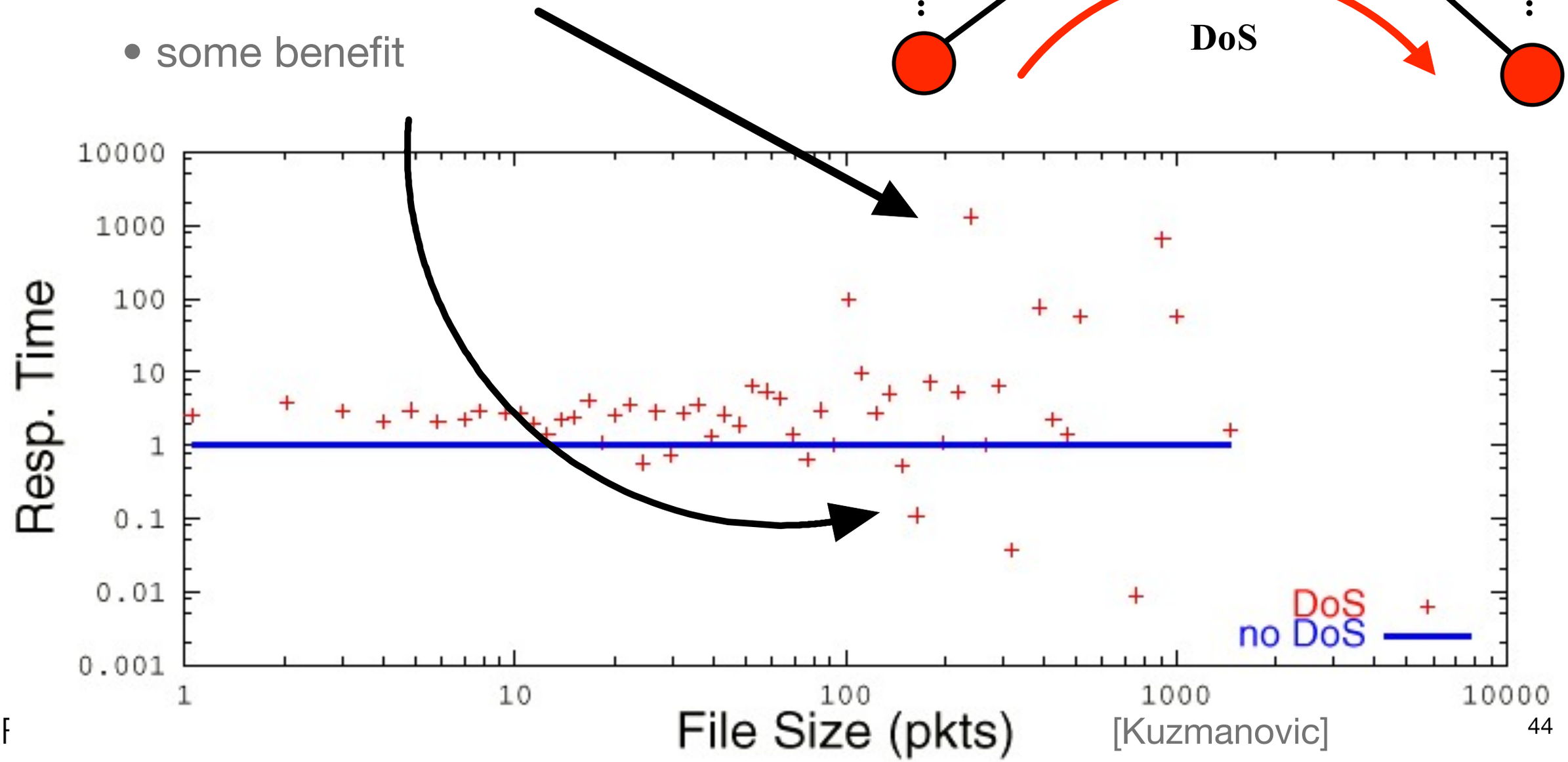




Shrews vs. Short-lived TCP Traffic

- Scenario: Web browsing
  - Larger files more vulnerable
    - most suffer
    - some benefit

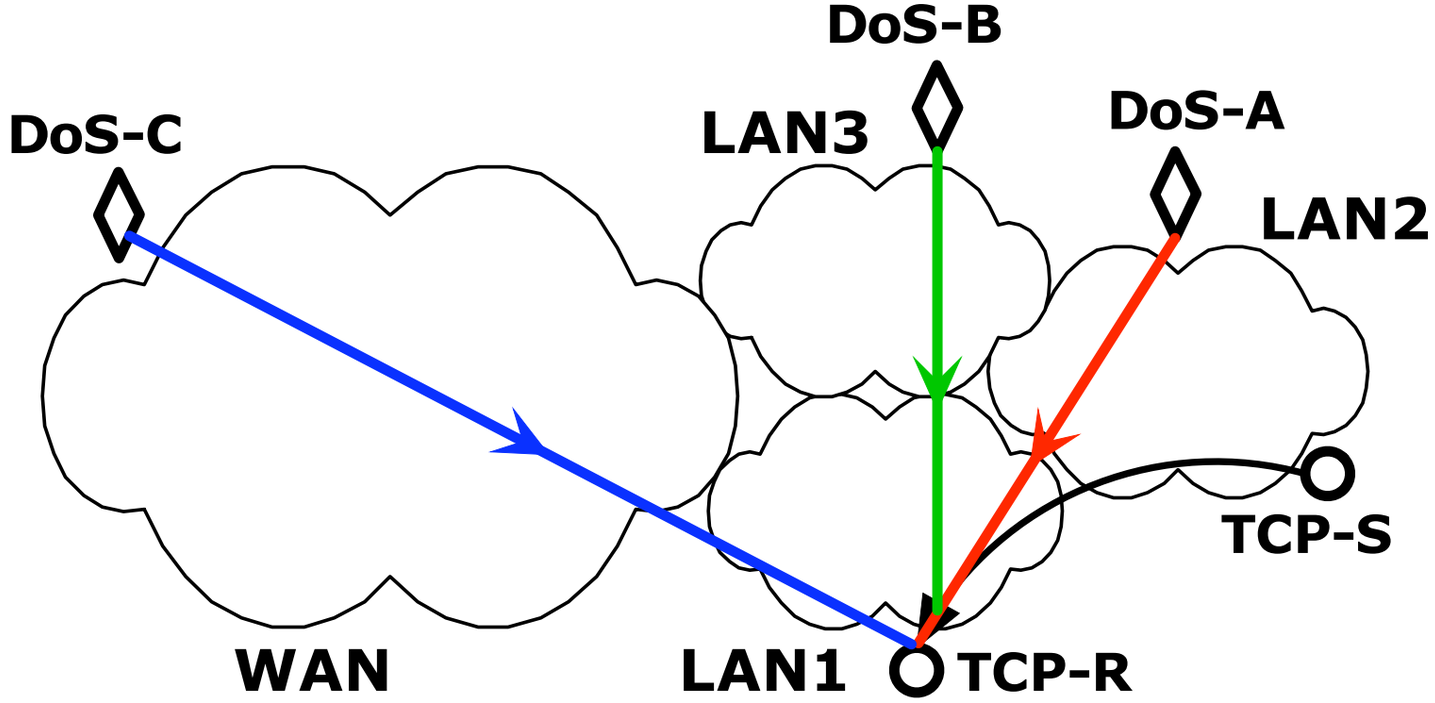




[Kuzmanovic]

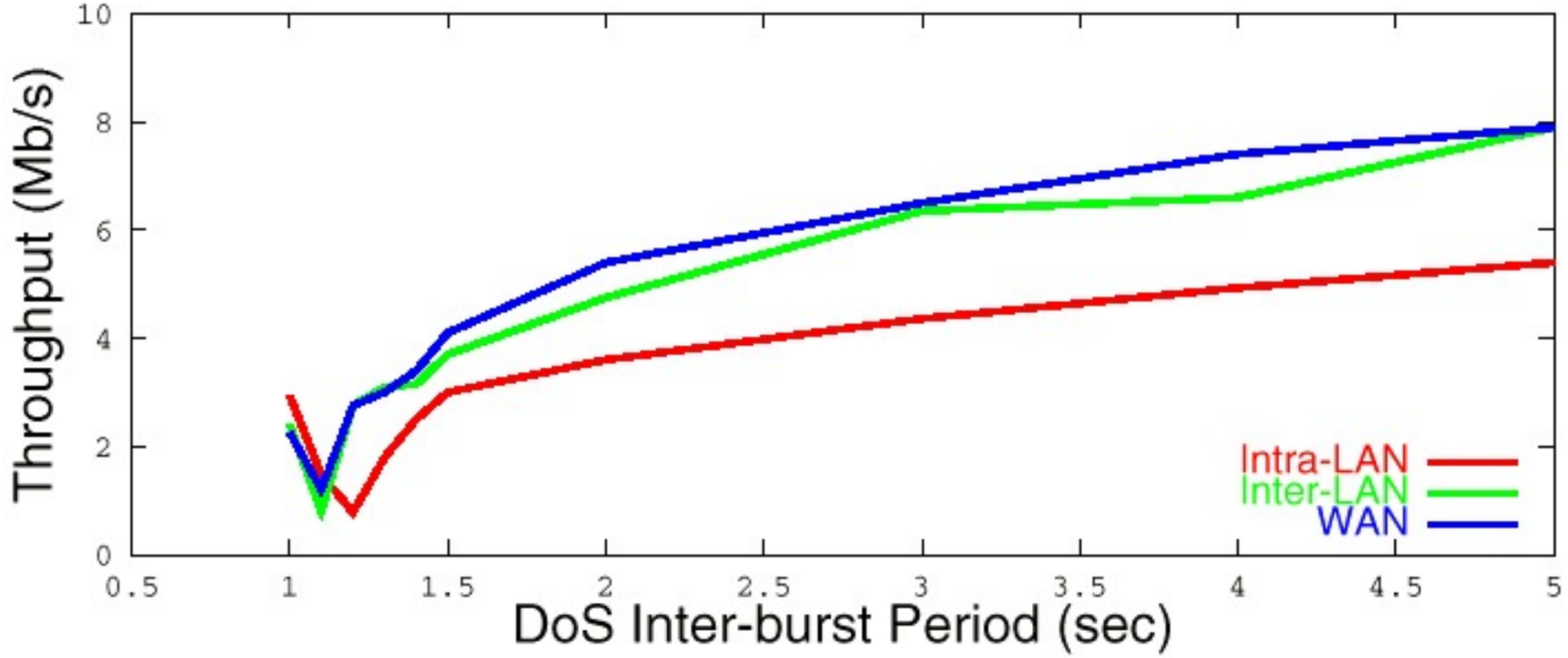
Internet Experiments: Scenario

- Scenario: victim on a lightly loaded 10 Mb/sec LAN
- Attacker on same LAN, nearby LAN, or over WAN
- WAN path:
  - EPFL→ETH, 8 hops (10/100/OC-12)



Internet Experiments: Results

- Shrew average rate: 909 kb/sec
  - R = 10 Mb/sec, l = 100 msec, T = 1.1 sec
- TCP throughput
  - 9.8 Mb/sec without Shrew
  - 1.2 Mb/sec with Shrew, 87.8% degradation



Detecting Shrews

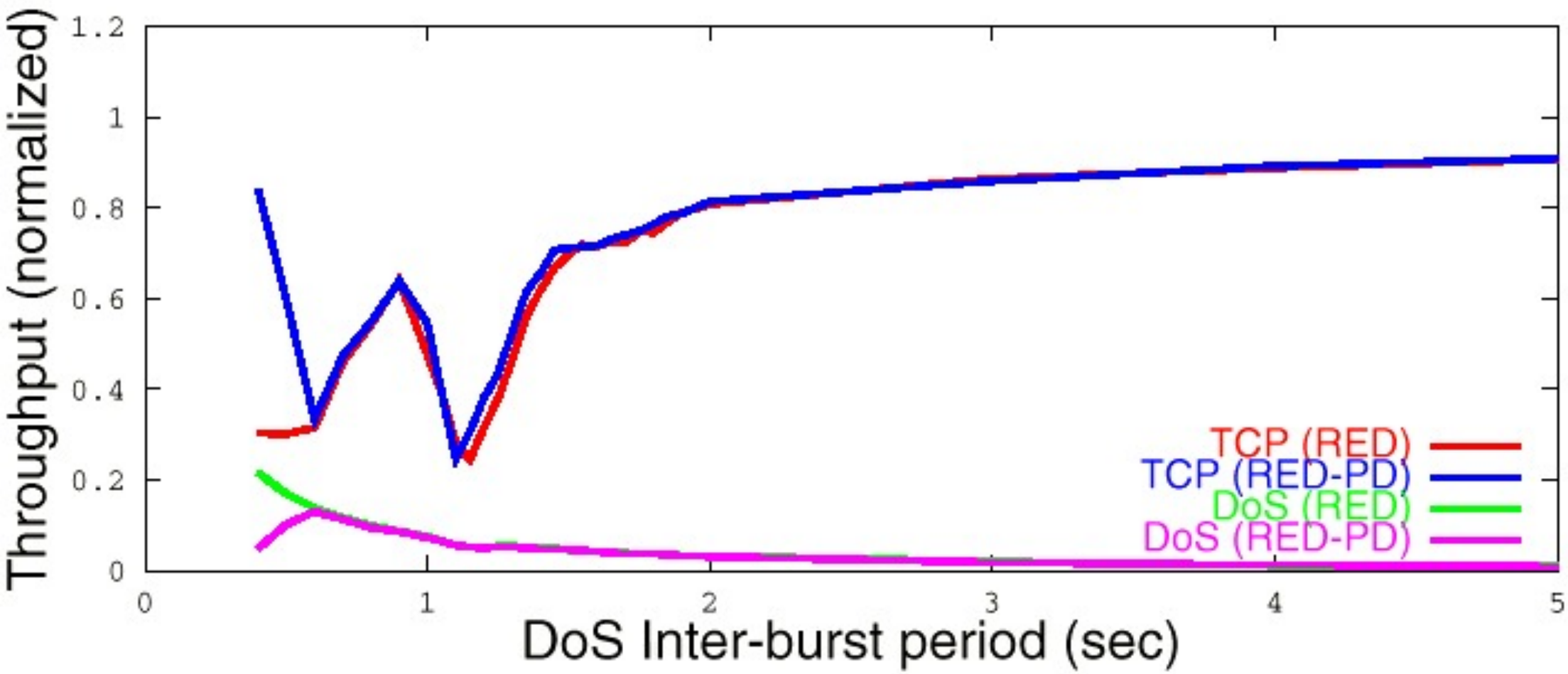
- Shrews have low average rate, yet send high-rate bursts on short time-scales

- Key questions
  - Can algorithms intended to find high-rate attacks detect Shrews?
  - Can we tune the algorithms to detect Shrews without having too many false alarms?

- A number of schemes can detect malicious flows
  - E.g., RED-PD:
    - use the packet drop history to detect high-bandwidth flows and preferentially drop packets from these flows

Router-Assisted Mechanisms

- Scenario: 9 TCP Sack flows with RED and RED-PD
- RED-PD only detects Shrews with unnecessarily high rate
- Reducing RED-PD measurement time scale results in excessive false positives



End-point minRTO Randomization

- Observe
  - Shrews exploit protocol homogeneity and determinism
- Question
  - Can minRTO randomization alleviate threat of Shrews?
- TCP flows' approach
  - Randomize the minRTO = uniform(a,b)

- Shrews' counter approach
  - Given flows randomize minRTO, the optimal Shrew pulses at time-scale T=b
    - Wait for all flows to recover and then pulse again

Conclusions

- Shrew principles
  - Exploit slow-time-scale protocol homogeneity and determinism

- Real-world vulnerability to Shrew attacks
  - Internet experiment: 87.8% throughput loss without detection

- Shrews are difficult to detect
  - Low average rate and “TCP friendly”
  - Cannot filter short bursts
  - Fundamental mismatch of attack/defense timescales

Acknowledgments/References

- [Ross] Exploiting P2P Systems for DDoS Attacks, Naoum Naoumov, Keith W. Ross, Polytechnic University.
- [Karp06] Distributed Hash Tables: Chord, Brad Karp, CS 4C38 / Z25, 25th January, 2006, UCL Computer Science.
- [Kuzmanovic] Low-Rate TCP-Targeted Denial of Service Attacks (The Shrew vs. the Mice and Elephants), A. Kuzmanovic and E. Knightly, in Proceedings of ACM SIGCOMM 2003, Karlsruhe, Germany, August 2003.

Administrivia

- Reminder of the rules:
  - There will be a zero tolerance policy for cheating/copying HWs. The first time you are caught, you will receive a zero for the task at hand. If you are caught for a second time, you will fail the course.
  - Providing your assignment to someone else is considered cheating on your behalf.