Sub-Second Lookups on a Large-Scale Kademlia-Based Overlay

Raul Jimenez, Flutra Osmani, Björn Knutsson

KTH, Sweden

Distributed Hash Table (DHT)

- Scalable
- Robust to churn
- Performance

Distributed Hash Table (DHT)

- Scalable
- Robust to churn
- Performance
  - 100-300 ms [Davek-04]
  - <200 ms [Rhea-05]
  - 450 ms [Li-05]
  - 250 ms [Kaune-08]

Distributed Hash Table (DHT)

- Scalable
- Robust to churn
- Performance [simulators & small-scale]
  - 100-300 ms [Davek-04]
  - <200 ms [Rhea-05]
  - 450 ms [Li-05]
  - 250 ms [Kaune-08]

Large-Scale DHT Overlays

Large-Scale DHT Overlays

> 1 million nodes

Large-Scale DHT Overlays

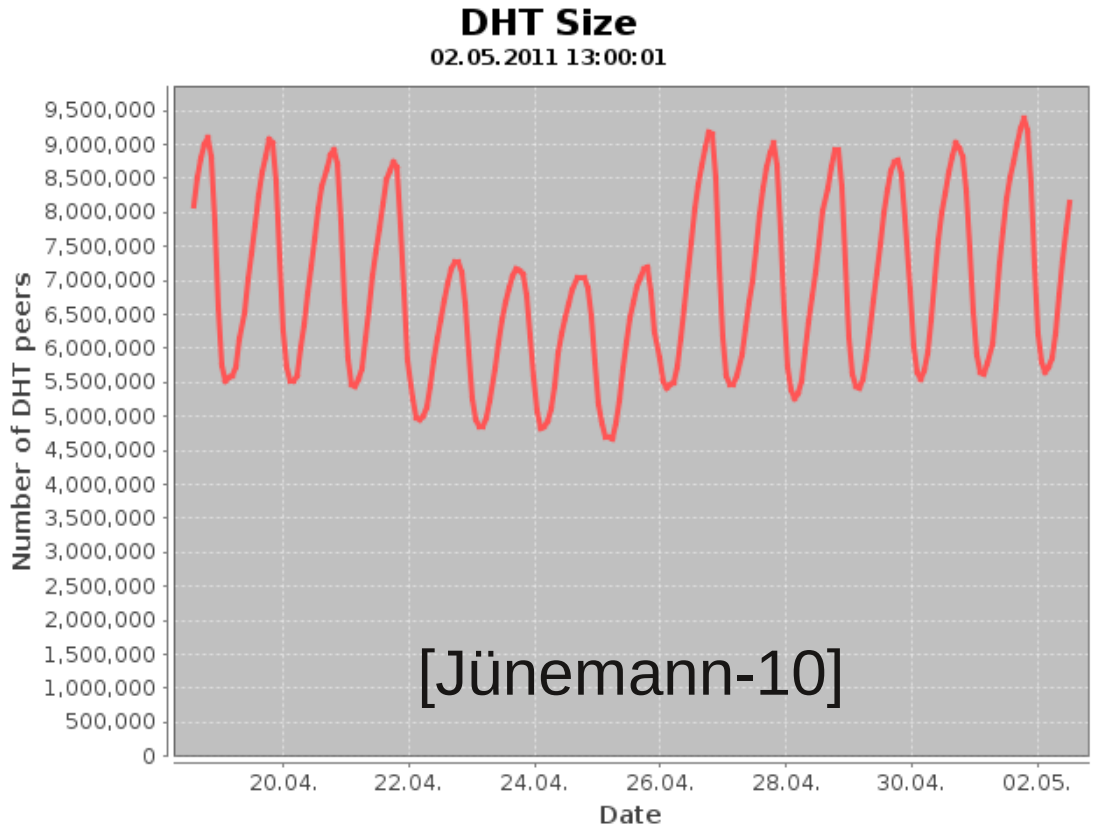
- **KAD** (eMule)
  - **2 s** [Stutzbach-06]
  - **1.5 s** [Steiner-09]
- **Azureus DHT** (Azureus/Vuze)
  - **2 min** [Crosby-07]
  - **13 s** [Falkner-07]
- **Mainline DHT** (BitTorrent)
  - **1 min** [Crosby-07]

Large-Scale DHT Overlays

- **KAD** (eMule)
  - **2 s** [Stutzbach-06]
  - **1.5 s** [Steiner-09]
- **Azureus DHT** (Azureus/Vuze)
  - **2 min** [Crosby-07]
  - **13 s** [Falkner-07]
- **Mainline DHT** (BitTorrent)
  - **1 min** [Crosby-07]

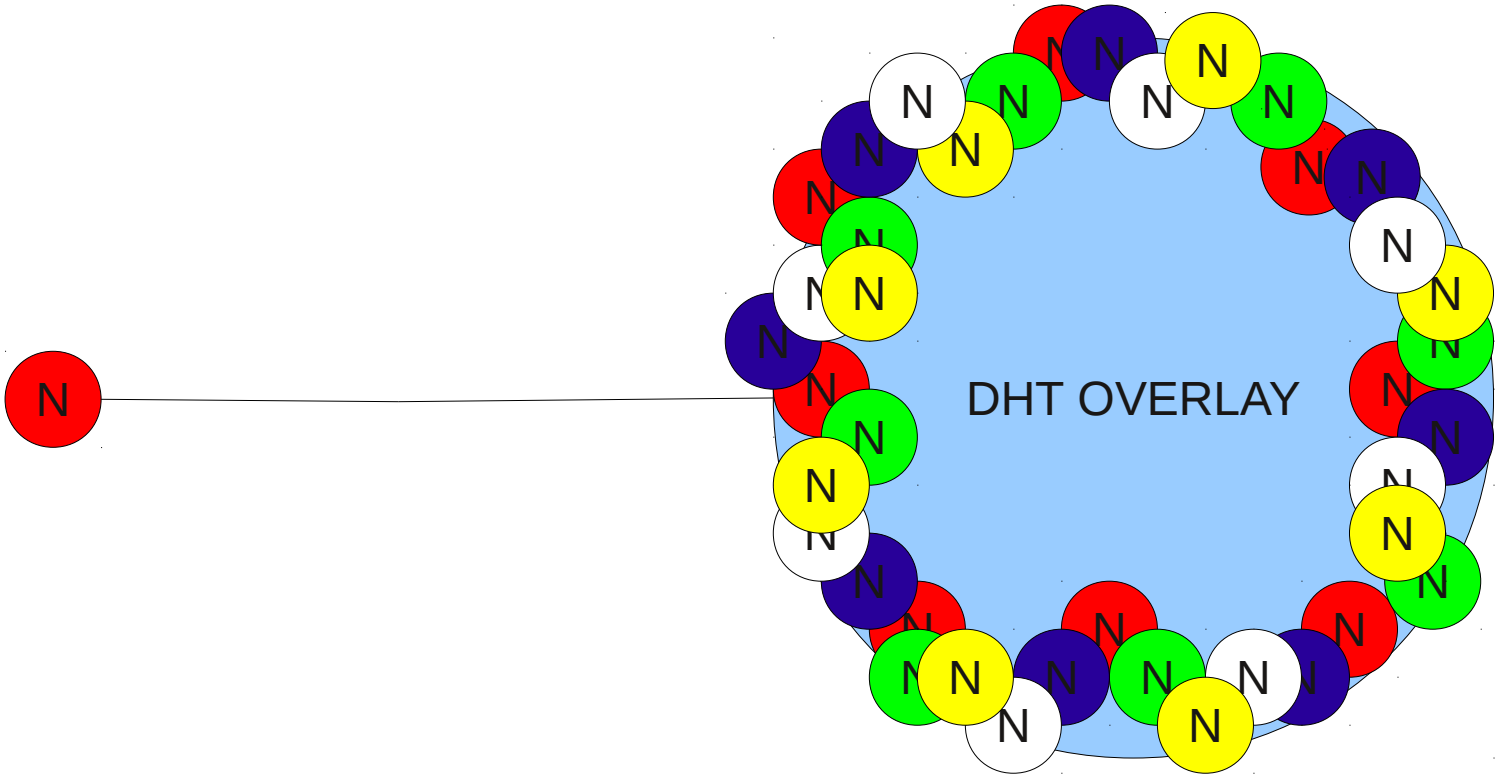
Mainline DHT

- Peer discovery mechanism for BitTorrent
- Video Streaming

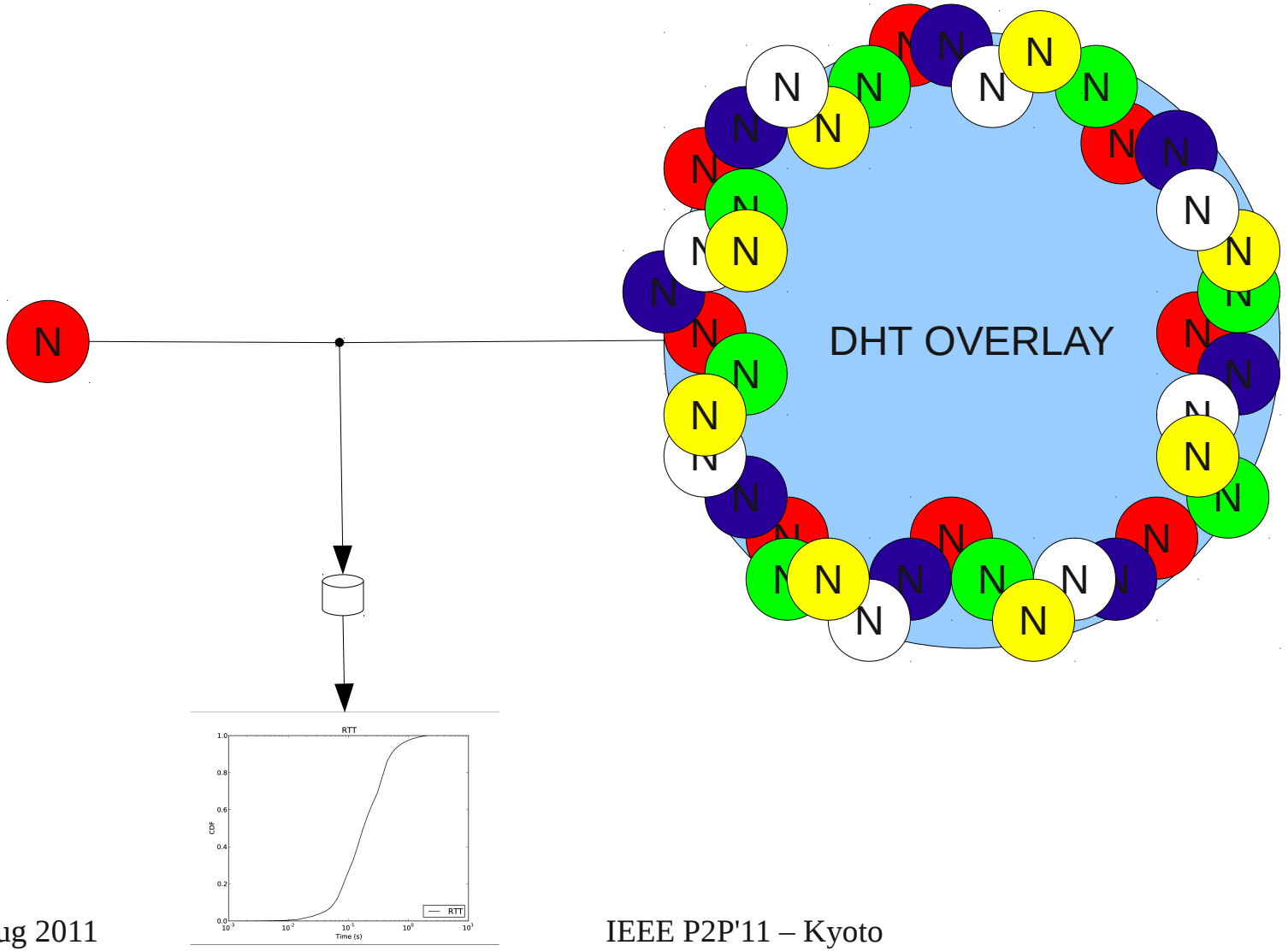


- Large-Scale overlay
  - 5M+ nodes
  - Heterogeneus:
    µTorrent, litbtorrent, transmission, ...

Measuring performance



Measuring performance



µTorrent's performance

- 60% of Mainline DHT nodes
- Median lookup latency: **~650 ms**
- But …
  - **> 1 s 27% of lookups**
  - **High (and “bursty”) maintenance traffic**

Can we do better?

(while keeping backward-compatibility)

Our MainlineDHT nodes

- **Modular architecture**
  - **Lookup** module
  - **Routing table management** module

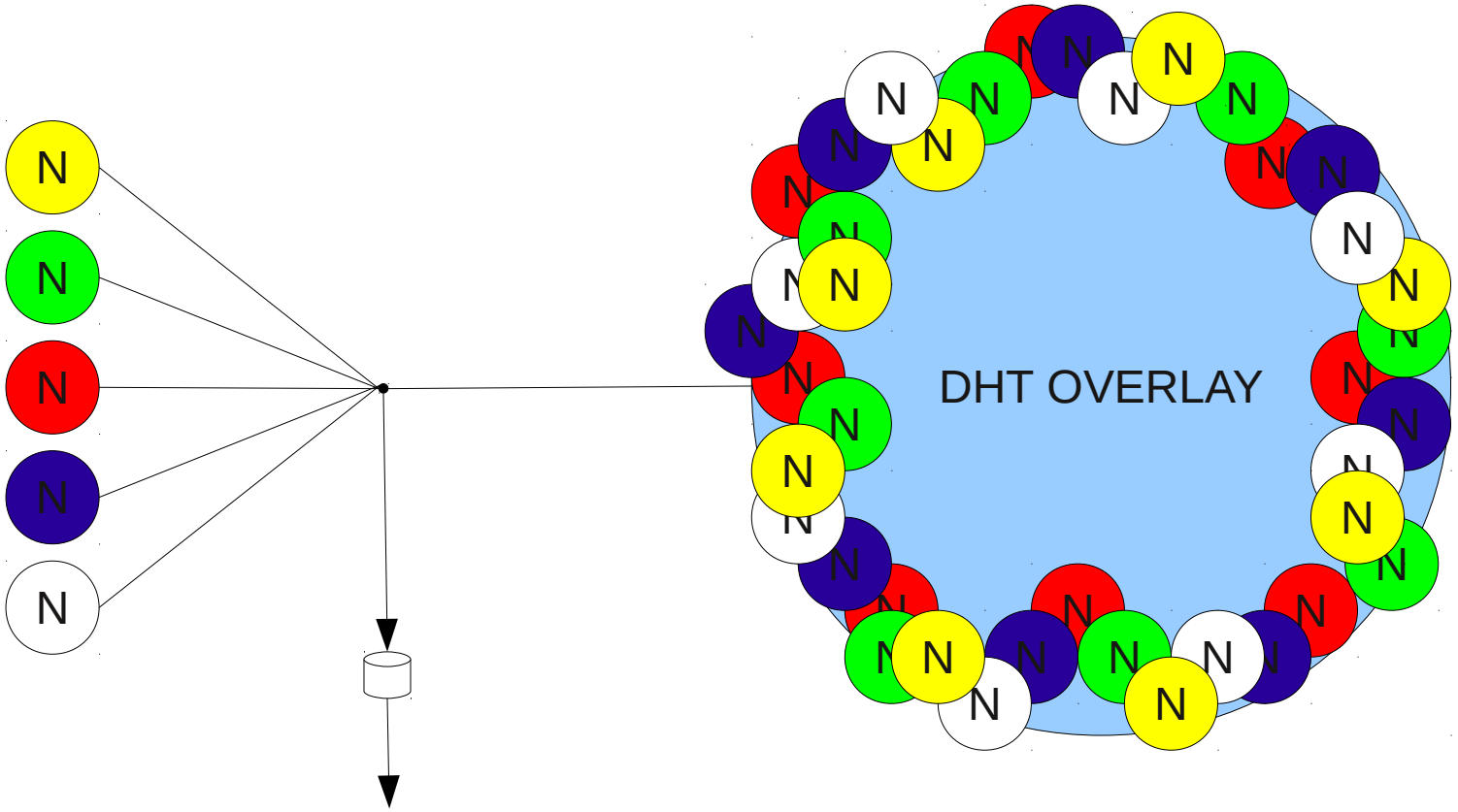
Lookup Parameters

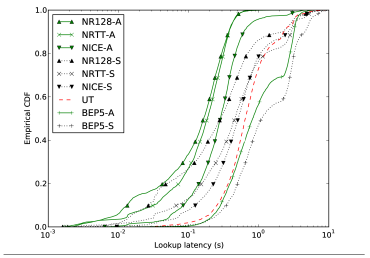
- **Standard** lookup module
  - **[same as µTorrent]**
- **Aggressive** lookup module
  - **[+] fast lookups [-] higher lookup cost**
  - [Stutzbach-06] [Steiner-09]

Routing Table Management

- Avoid nodes behind NAT / firewall [Jimenez-09]
  [+] more reliable neighbors
- Continuous refresh
  [+] lower maintenance traffic & no bursts
- Prefer **low-latency** neighbors
- Enlarge **most frequently used** buckets

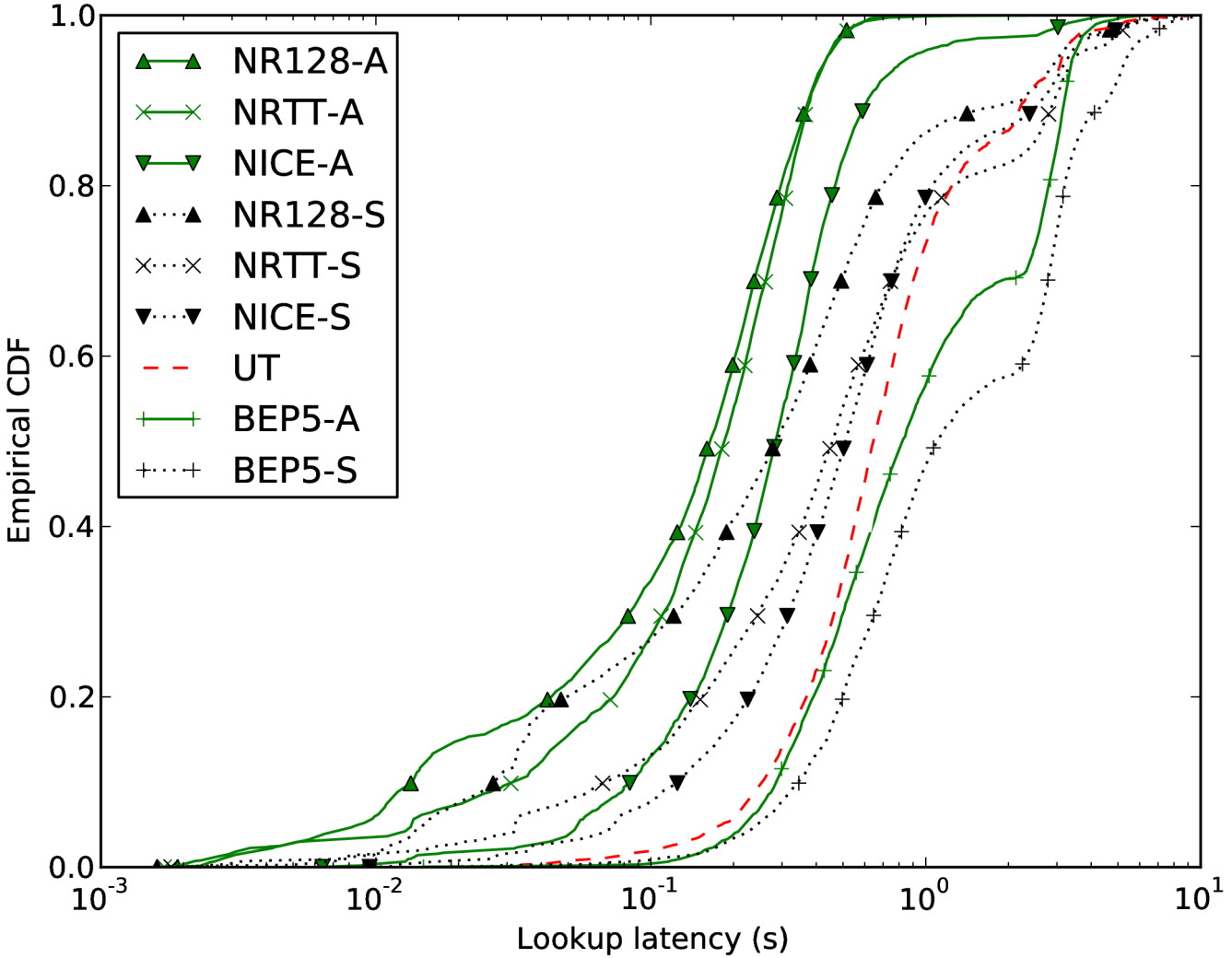
Measuring performance



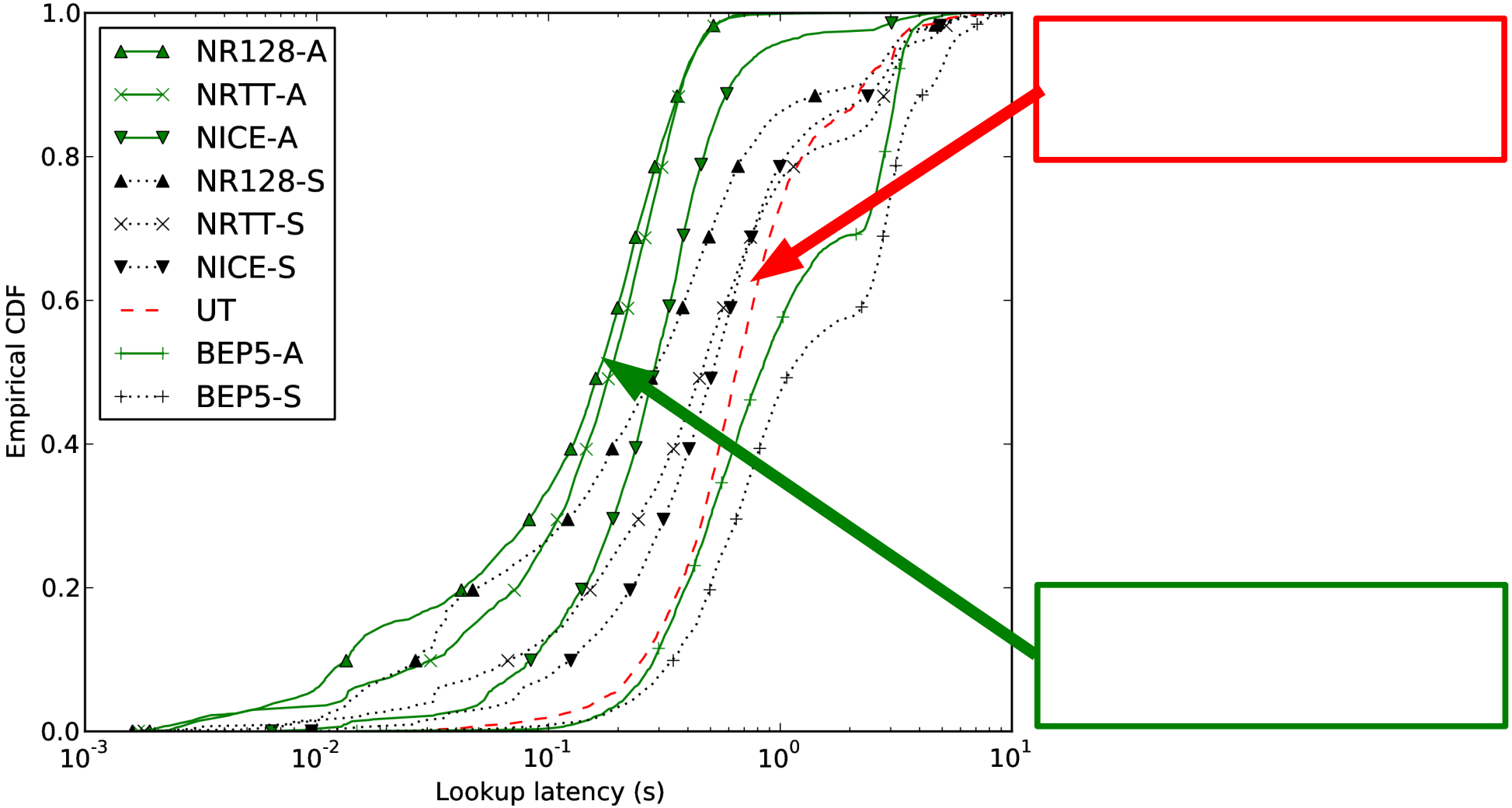


- µTorrent + 8 nodes
- 3078 lookups/node
- 80+ hours, 11+ GB
- **Very consistent results**

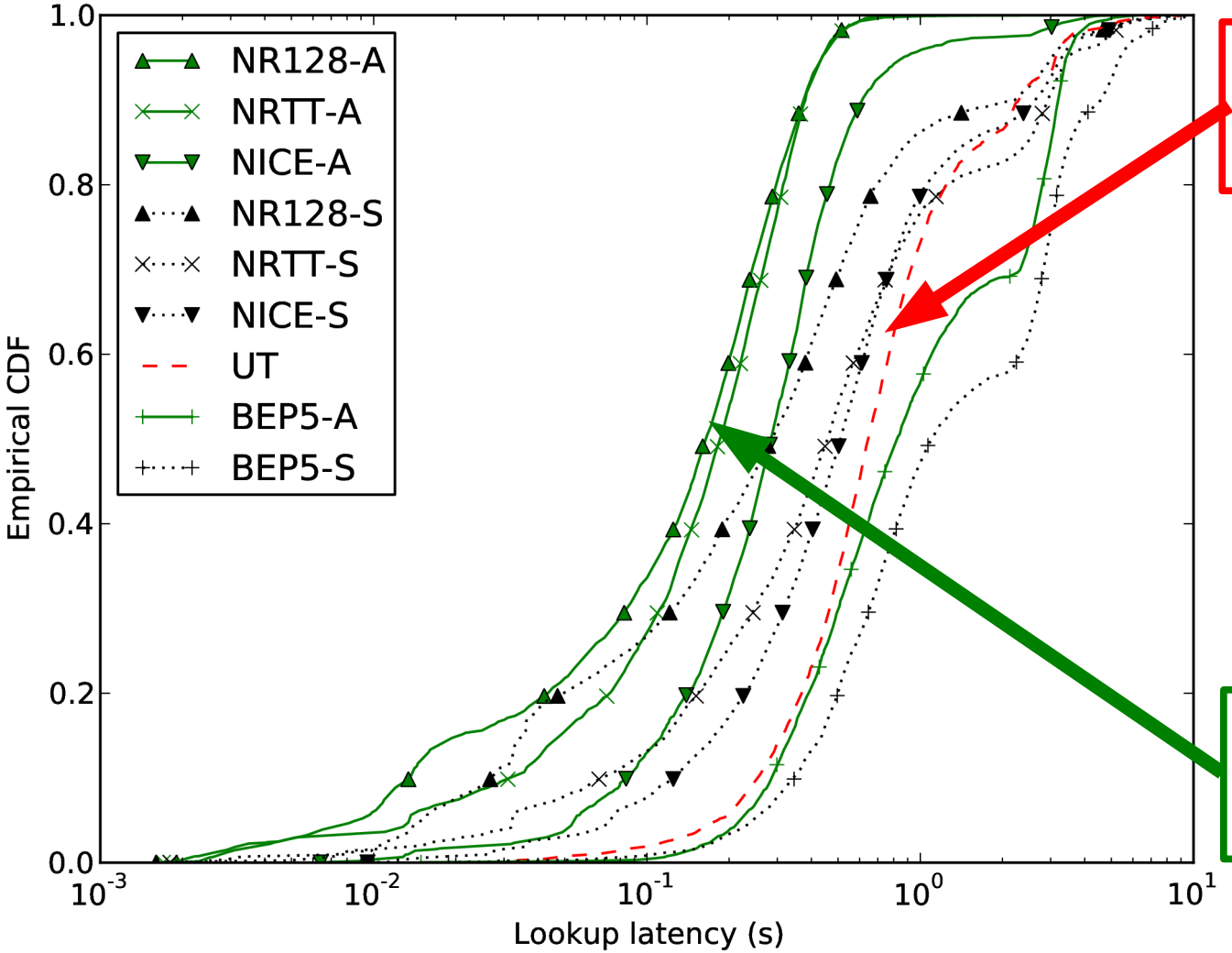
Lookup Latency



Lookup Latency



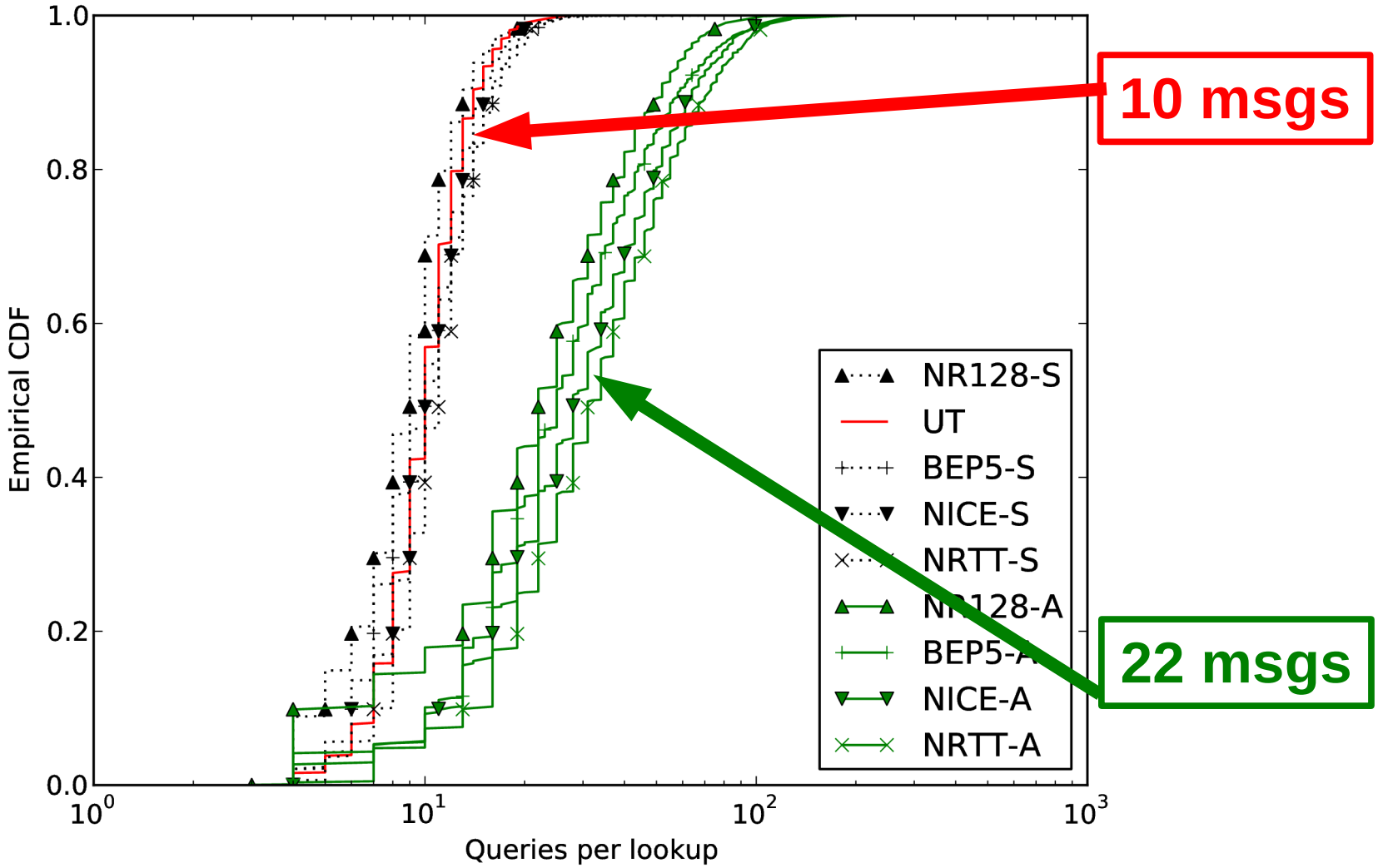
Lookup Latency



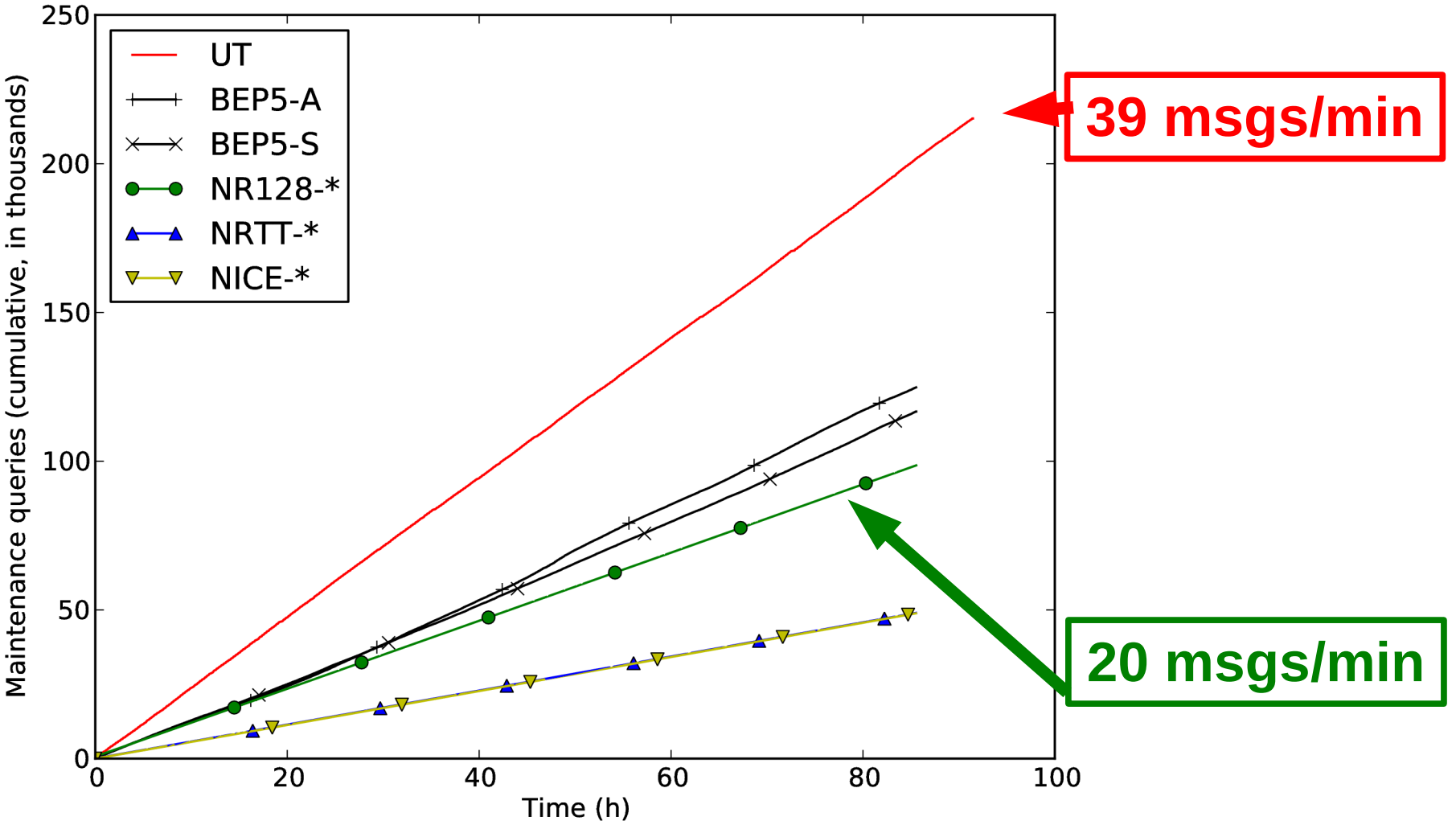
Median: **647 ms**
75th perc.: **~1 s**

Median: **164 ms**
99th perc.:**566 ms**

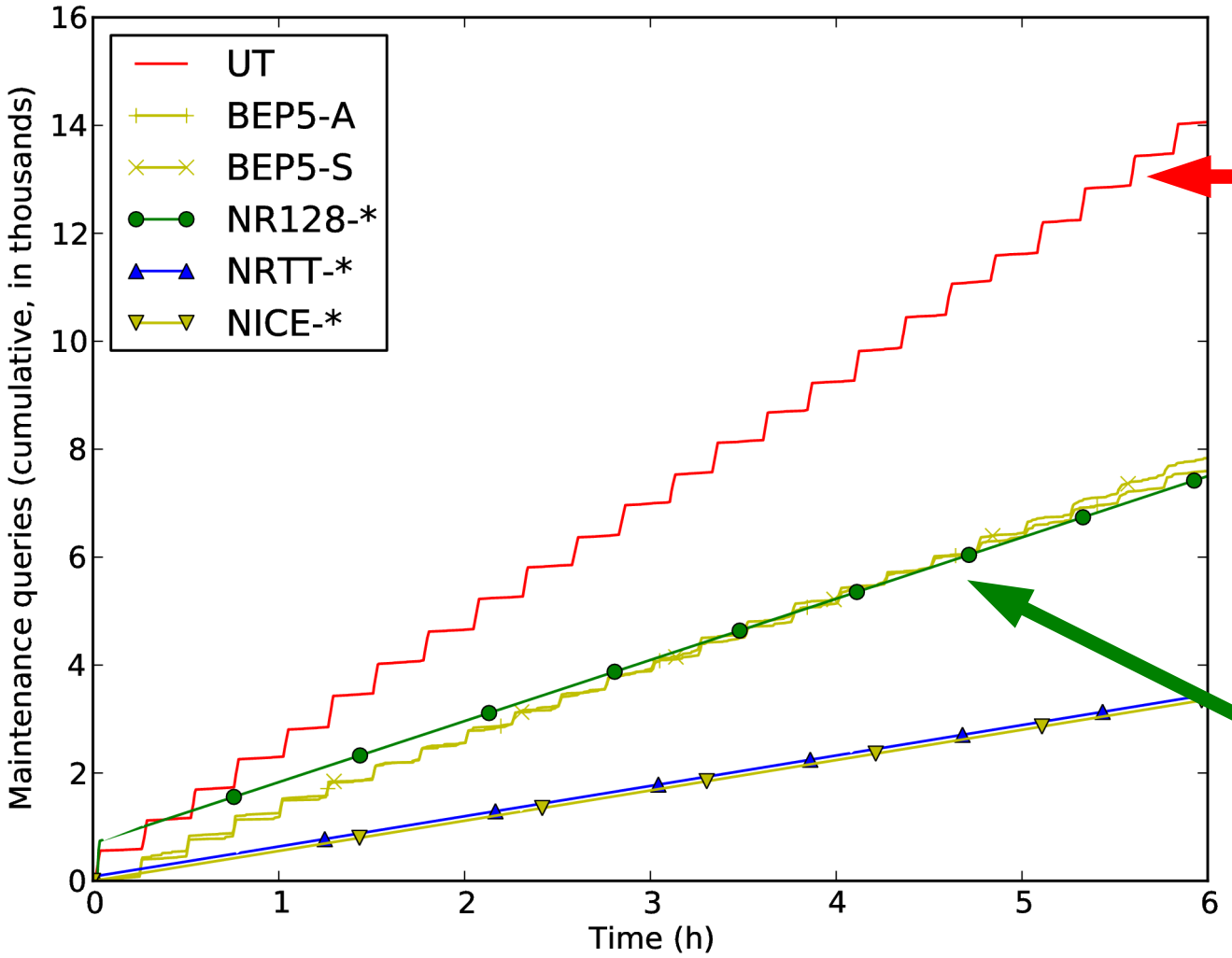
Lookup Cost


1.0
0.8
0.6
0.4
0.2
0.0
Empirical CDF
10^0
10^1
10^2
10^3
Queries per lookup
NR128-S
UT
BEP5-S
NICE-S
NRTT-S
NR128-A
BEP5-A
NICE-A
NRTT-A
10 msgs
22 msgs

Maintenance Cost



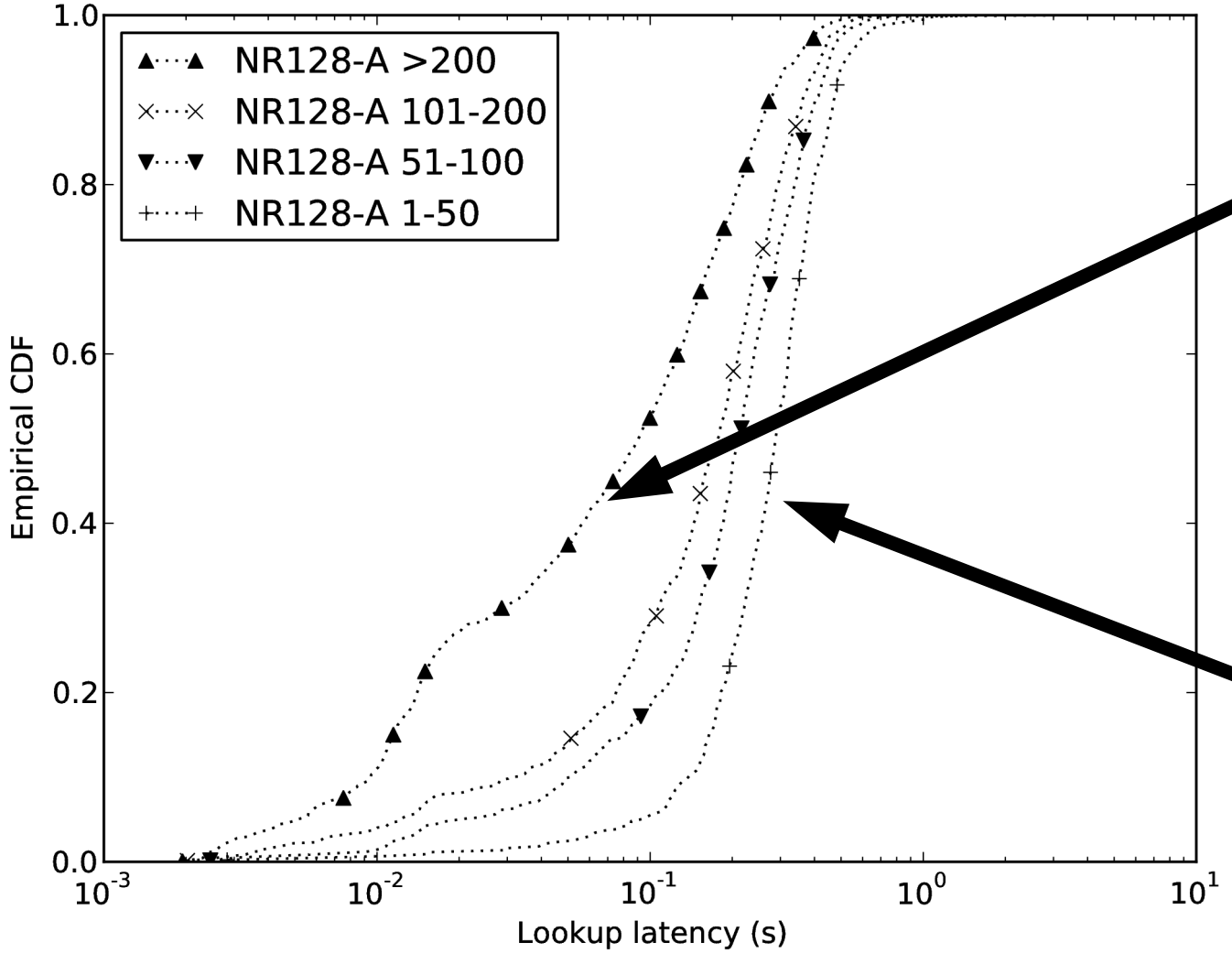
Maintenance Cost [first 6 hours]



~600 msgs
15 min

3-sec slots

Popularity and Latency



Large swarms ~ 100 ms

Small swarms ~ 200 ms

Conclusion

- **Sub-second lookups in a large-scale overlay**
  median < 200 ms; 99th percentile < 600 ms

  - **Modular architecture**: routing and lookup modules
  - **Trade-offs** between **lookup performance**, **lookup cost** and **maintenance cost**
  - **Correlation** between **key popularity** and **lookup latency**

Thank you!