Cooperative Architectures and Algorithms for Discovery and Transcoding of Multi-version Content

Claudia Canali
University of Parma

Valeria Cardellini
University of Rome
"Tor Vergata"

Michele Colajanni
University of Modena and Reggio Emilia

Riccardo Lancellotti
University of Rome
"Tor Vergata"

Philip S. Yu
IBM T. J. Watson Research Center

Heterogeneous environment

- Clients have different capabilities:
  - Display
  - CPU power
  - Network

Heterogeneous environment

- Clients have different capabilities:
  - Display
  - CPU power
  - Network

↓

Transcoding

(content adaptation)

Content adaptation: where?

- *Server*
- *Client*
- Intermediate-based content adaptation
  - Firewall, proxy, gateway already exists
- *Problem:* transcoding computationally expensive
- *Typical solutions:*
  - Caching $\rightarrow$ less transcoding required
  - Replication $\rightarrow$ load sharing

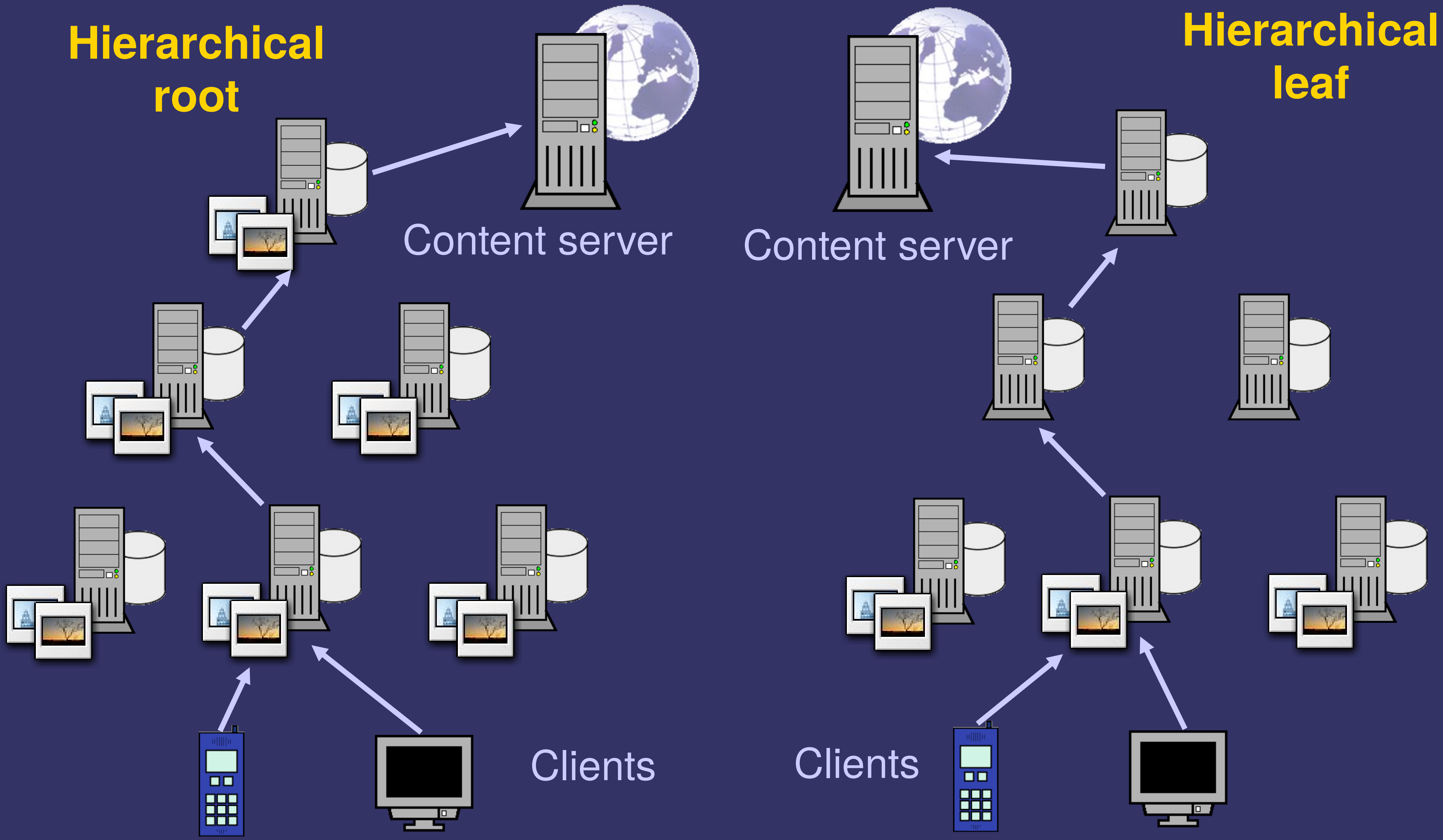
Distributed content adaptation

- *Caching issue:*
  - Multiple versions $\rightarrow$ working set size increased
- *Replication issue:*
  - Transcoding computationally expensive $\rightarrow$ need effective load sharing
- *Contribution:*
  - *Analysis of cooperation schemes applied to content adaptation architecture*

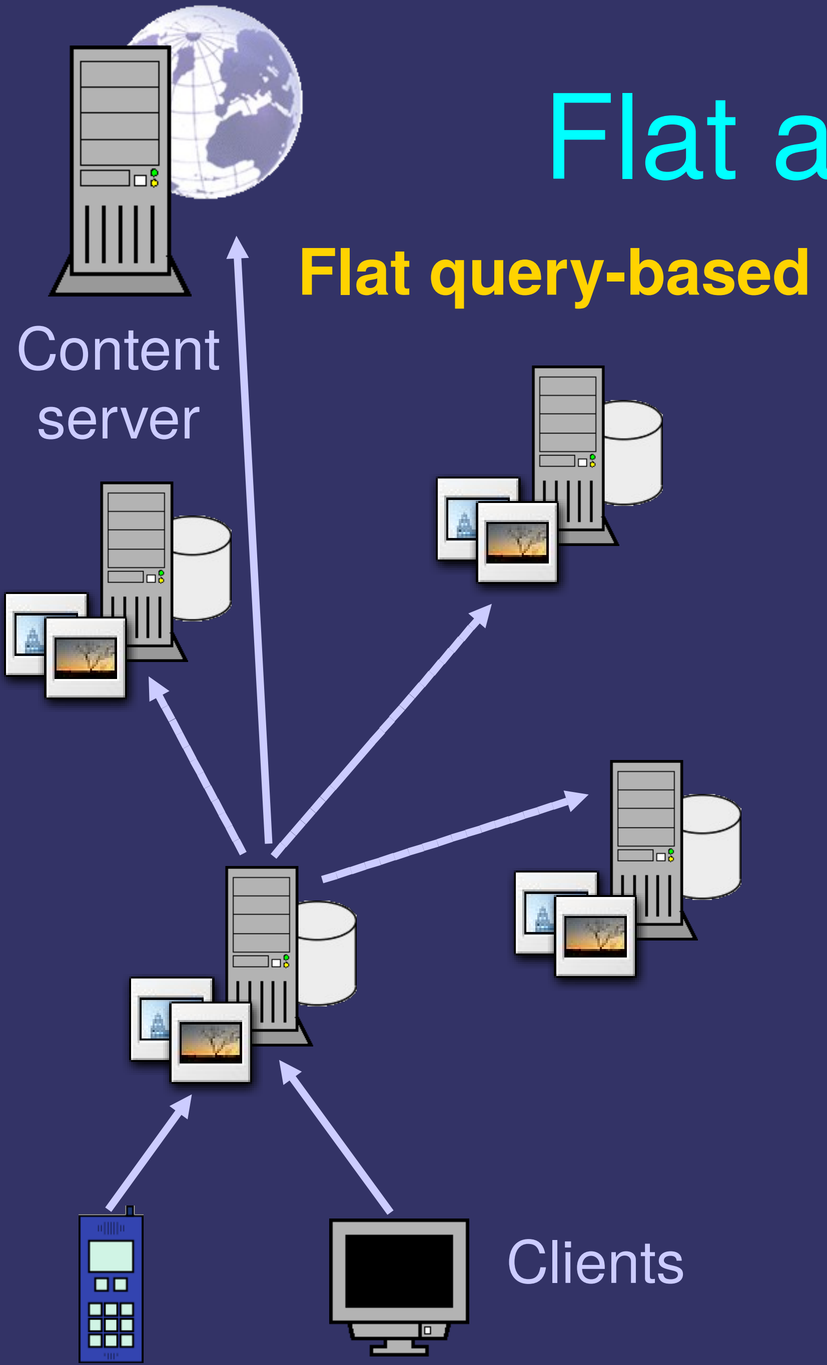
Topologies and schemes

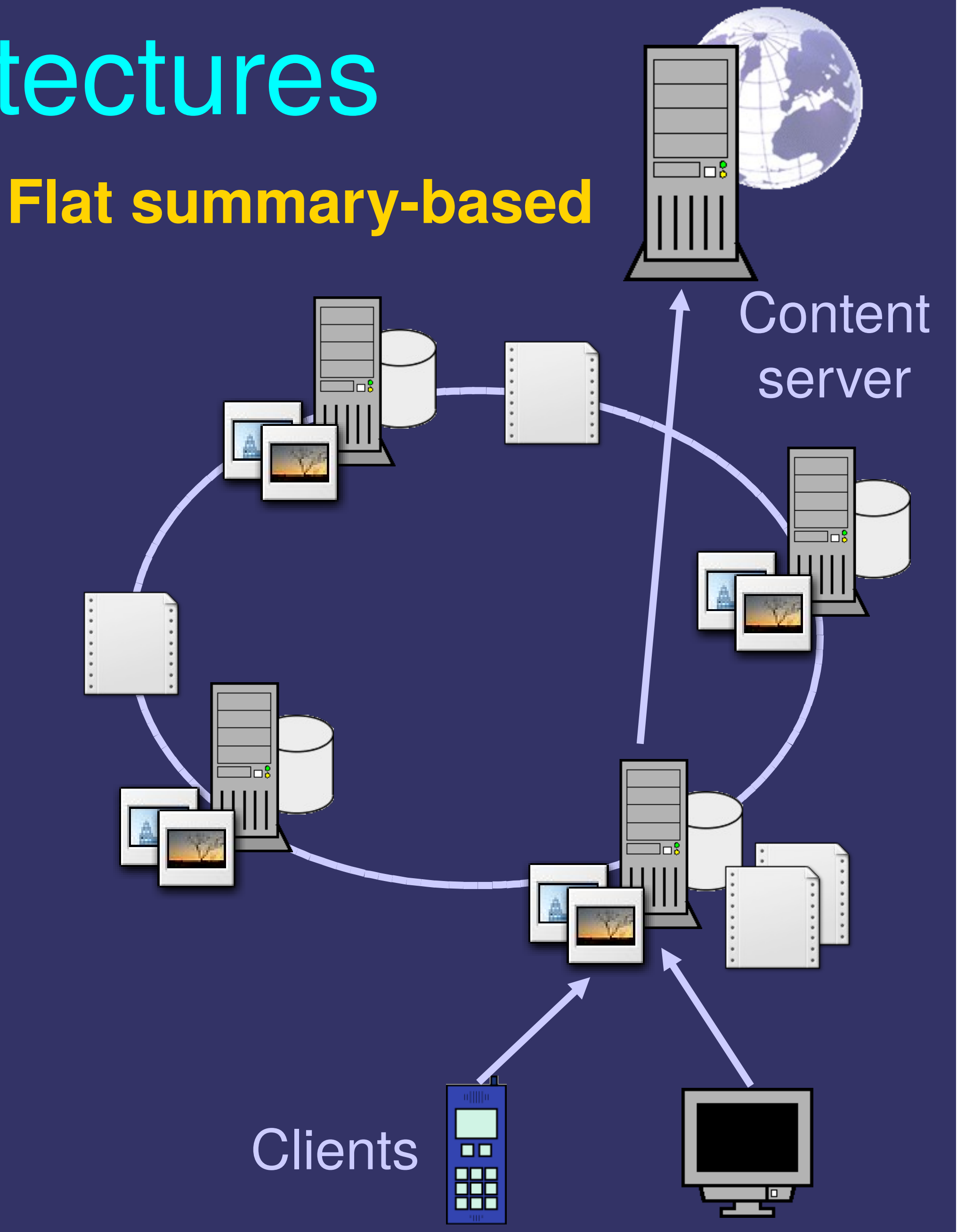
- Non cooperative (No coop)
- Hierarchical root (Hierarchical root)
- Hierarchical leaf (Hierarchical leaf)
- Flat query-based (Flat-query)
- Flat summary-based (Flat-summary)

Hierarchical architectures



Flat architectures

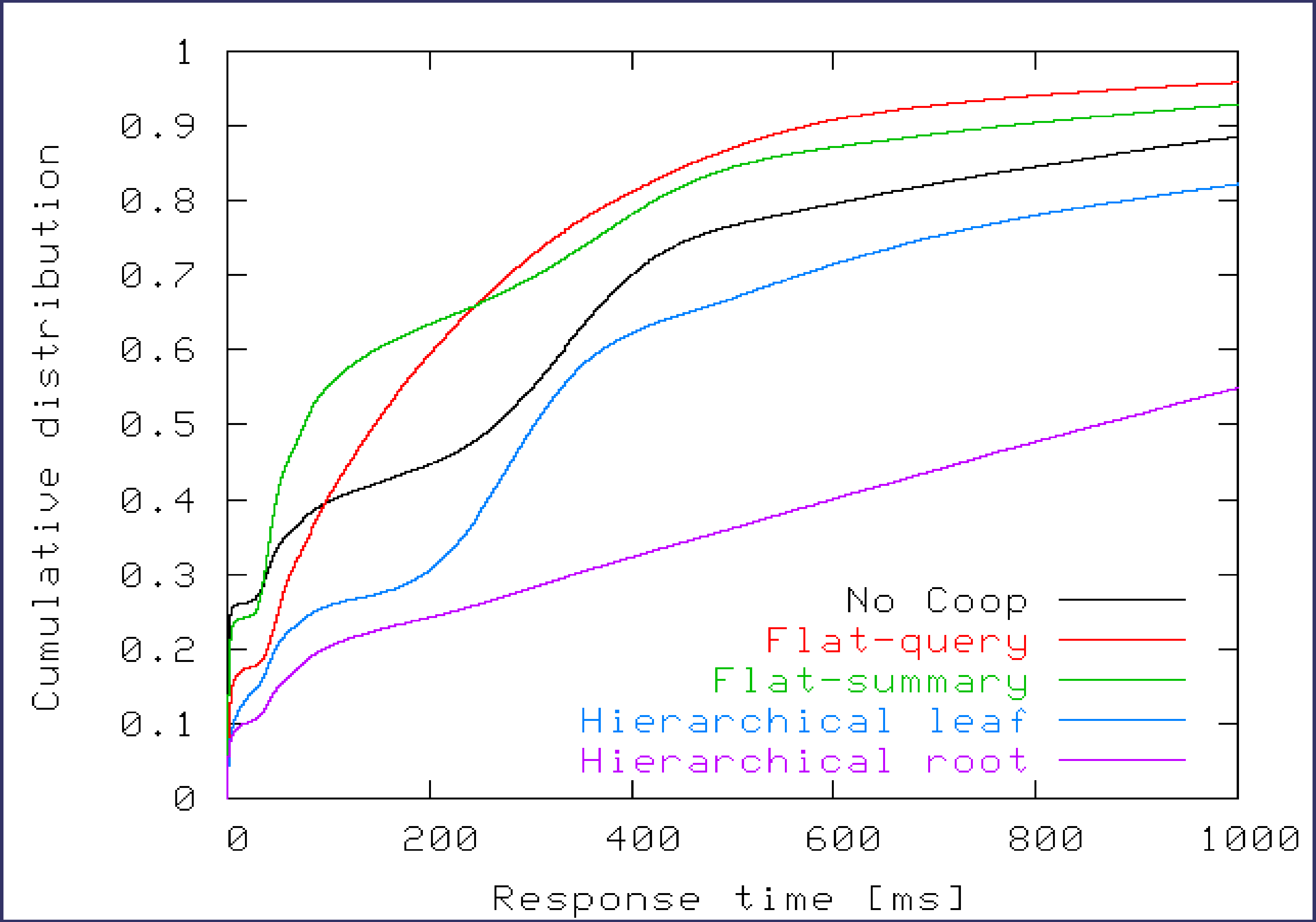




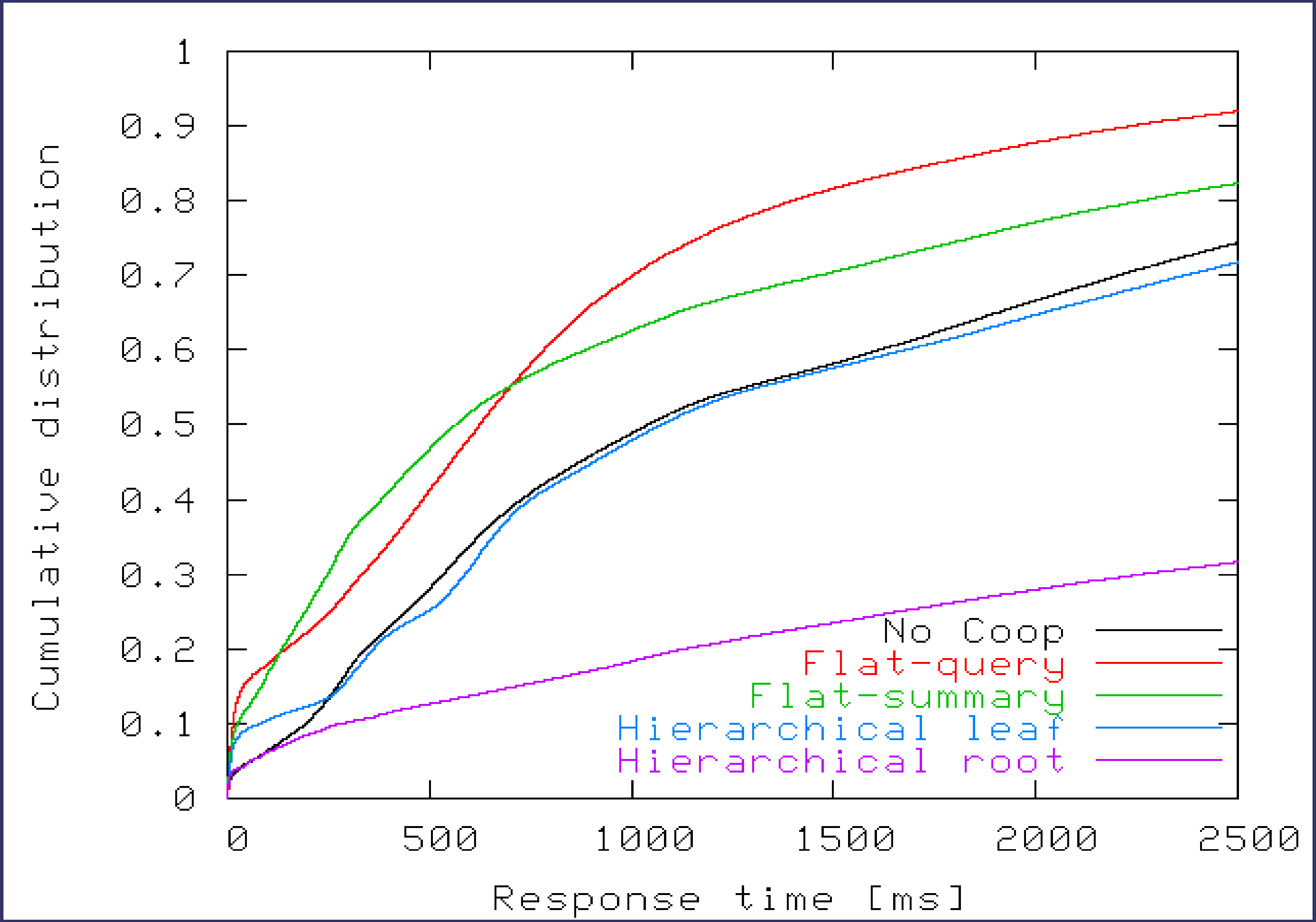
Workload models

- Two working sets
  - Light trans-load (resources from IRCache logs)
  - Heavy trans-load (*multimedia* working set)
- Syntetically generated traces

Architecture comparison (light trans-load)



Architecture comparison (heavy trans-load)



Summary of experiments

	Response time			
	Light trans-load [sec]		Heavy trans-load [sec]	
Cooperation scheme	Median	90-percentile	Median	90-percentile
Flat quey-based	0,11	0,64	0,62	2,24
Flat summary-basd	0,07	0,78	0,56	3,76
Hierarchical root	0,86	2,82	5,52	14,57
Hierarchical leaf	0,3	1,74	1,07	5,11

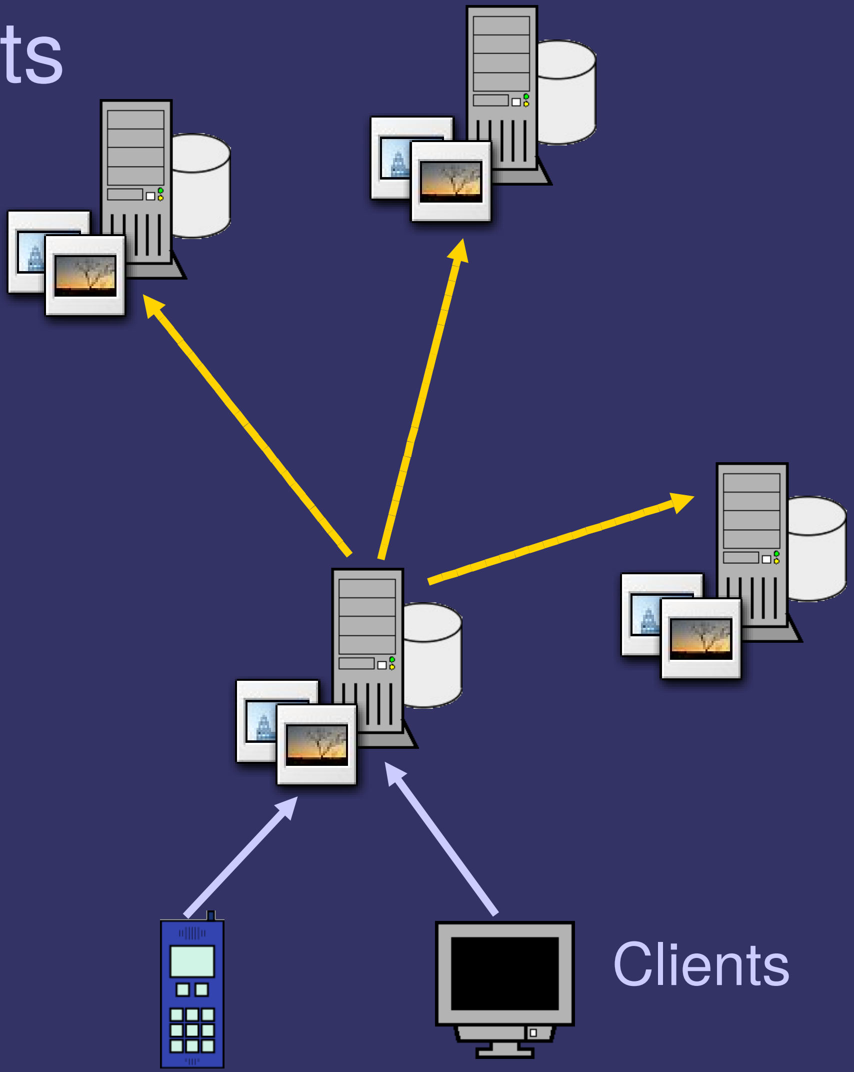
- Hierarchical root $\rightarrow$ congestion on the root node
- Flat architectures better than hierarchical
- Performance gain can be improved
- Issues related to load partially addressed
  - First solution: Load sharing algorithms

Load sharing algorithms

- Flat query-based architecture
- Choices in case of useful hit
- *Load-aware algorithm*
  - Local load-aware
  - Threshold
- *Load-blind algorithms*
  - Blind-active
  - Blind-lazy

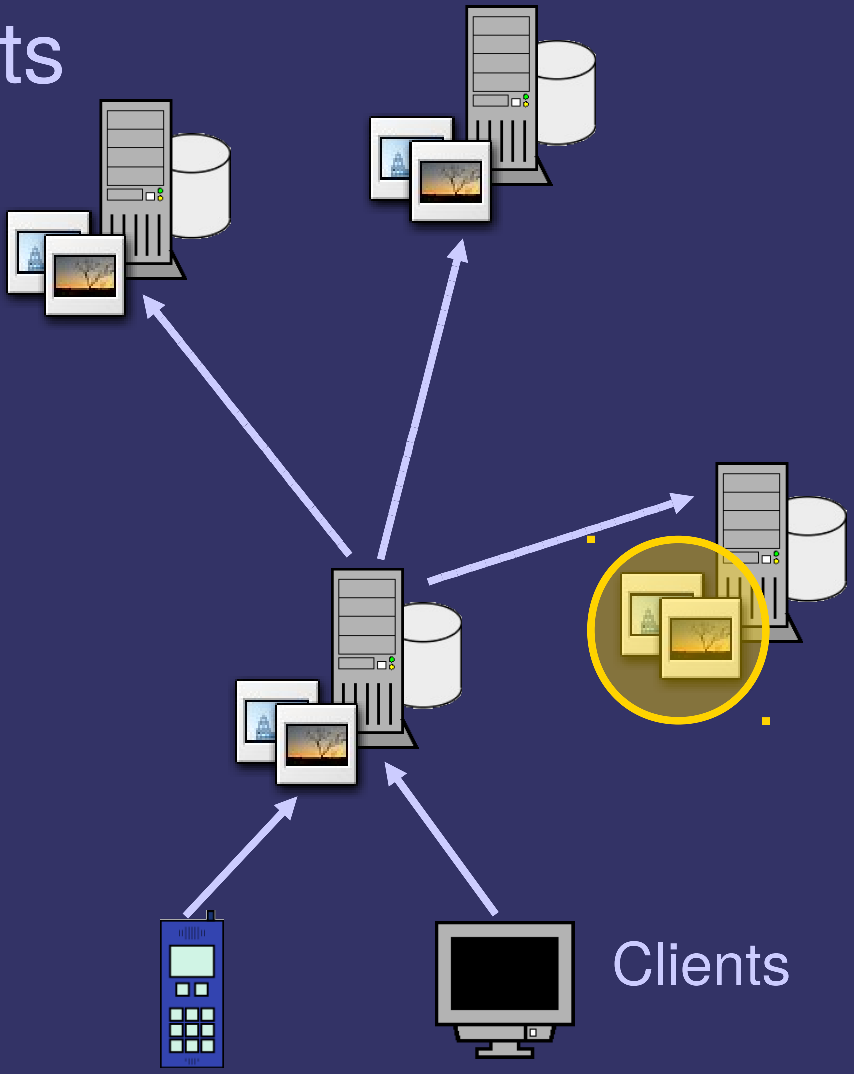
Load-aware algorithm

- Operate on useful hits
    - Local
    - Remote
- Threshold based algorithm
    - Load $\geq$ Thr $\rightarrow$ avoid transcoding
    - Load $<$ Thr $\rightarrow$ accept transcoding



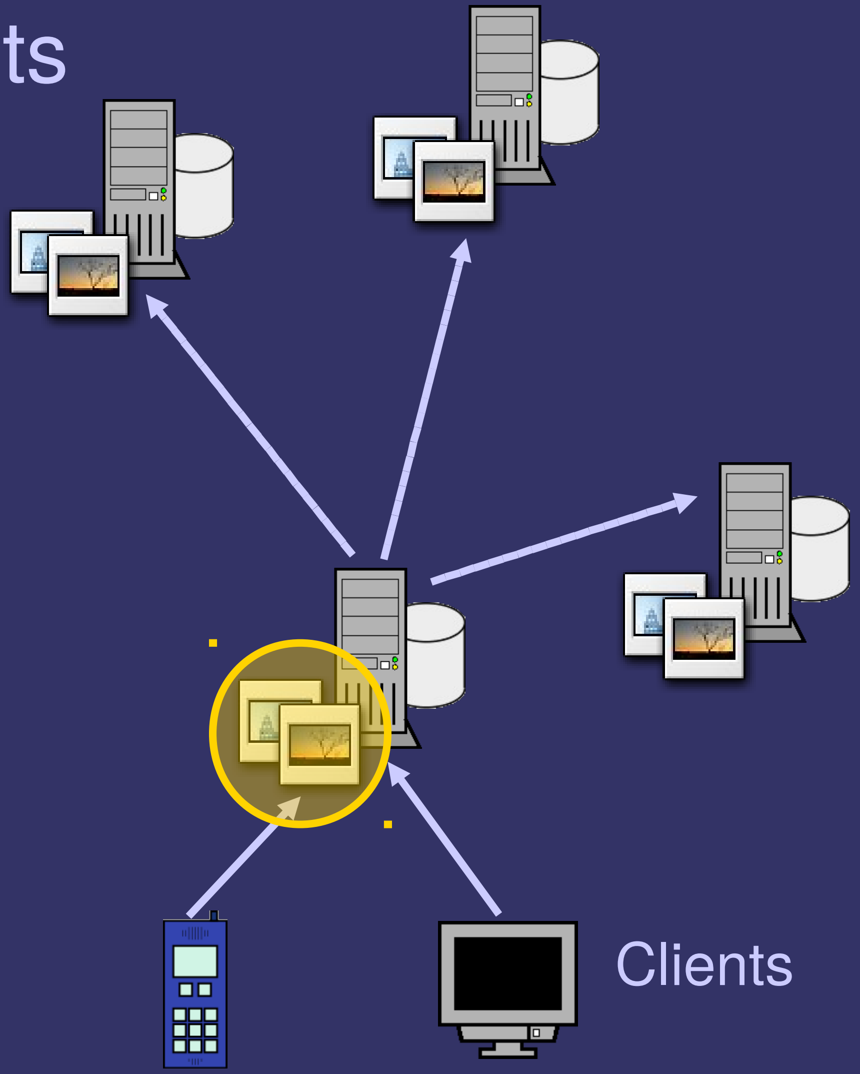
Load-aware algorithm

- Operate on useful hits
  - Local
  - Remote
- Threshold based algorithm
  - Load $\geq$ Thr $\rightarrow$ avoid transcoding
  - Load $<$ Thr $\rightarrow$ accept transcoding



Load-aware algorithm

- Operate on useful hits
  - Local
  - Remote
- Threshold based algorithm
  - Load ≥ Thr → avoid transcoding
  - Load < Thr → accept transcoding



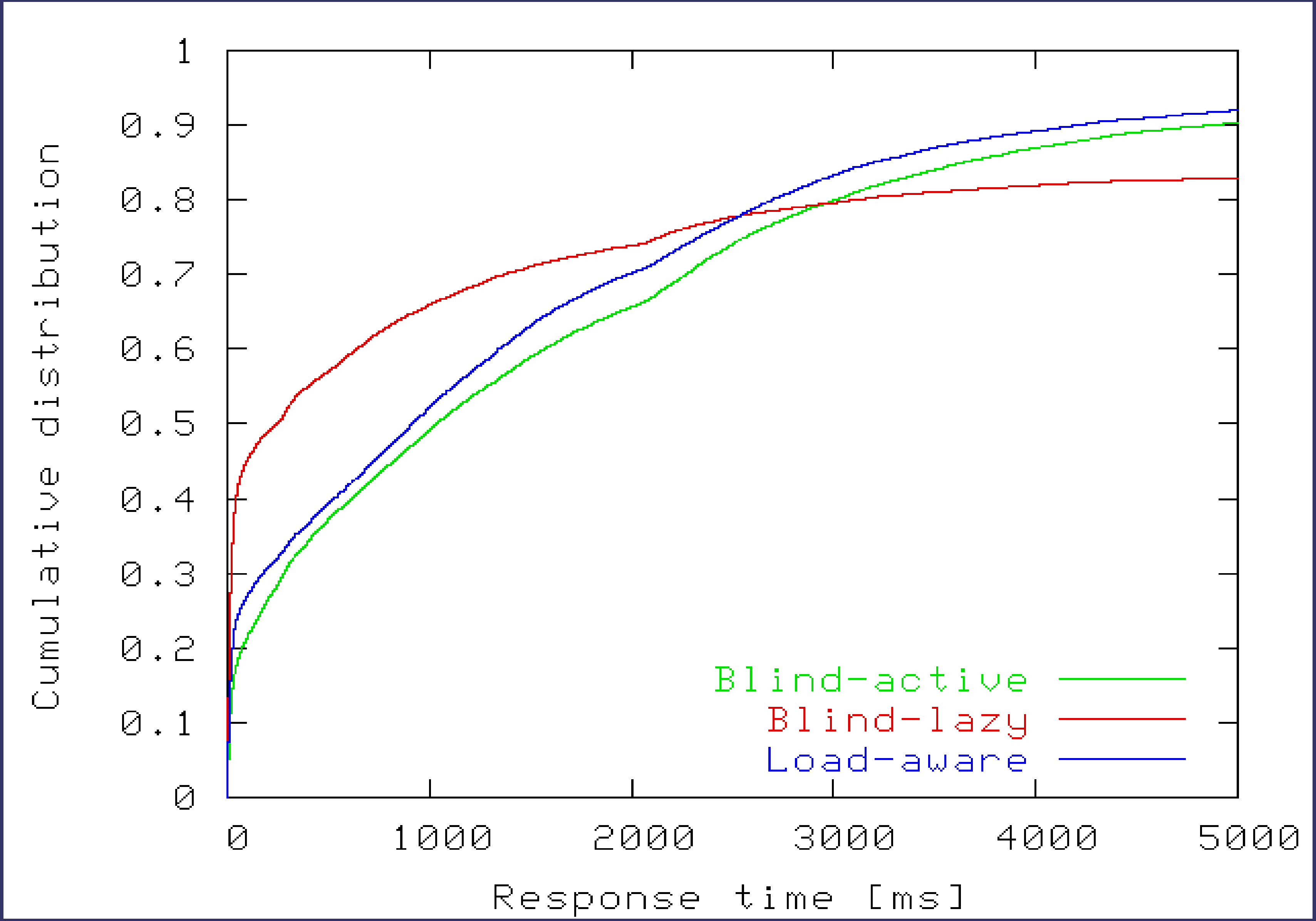
Load-blind algorithms

- Extreme threshold value $\rightarrow$ load blind
  - Thr=0.0 $\rightarrow$ Blind-lazy never content adaptation
  - Thr>1.0 $\rightarrow$ Blind-active always content adaptation

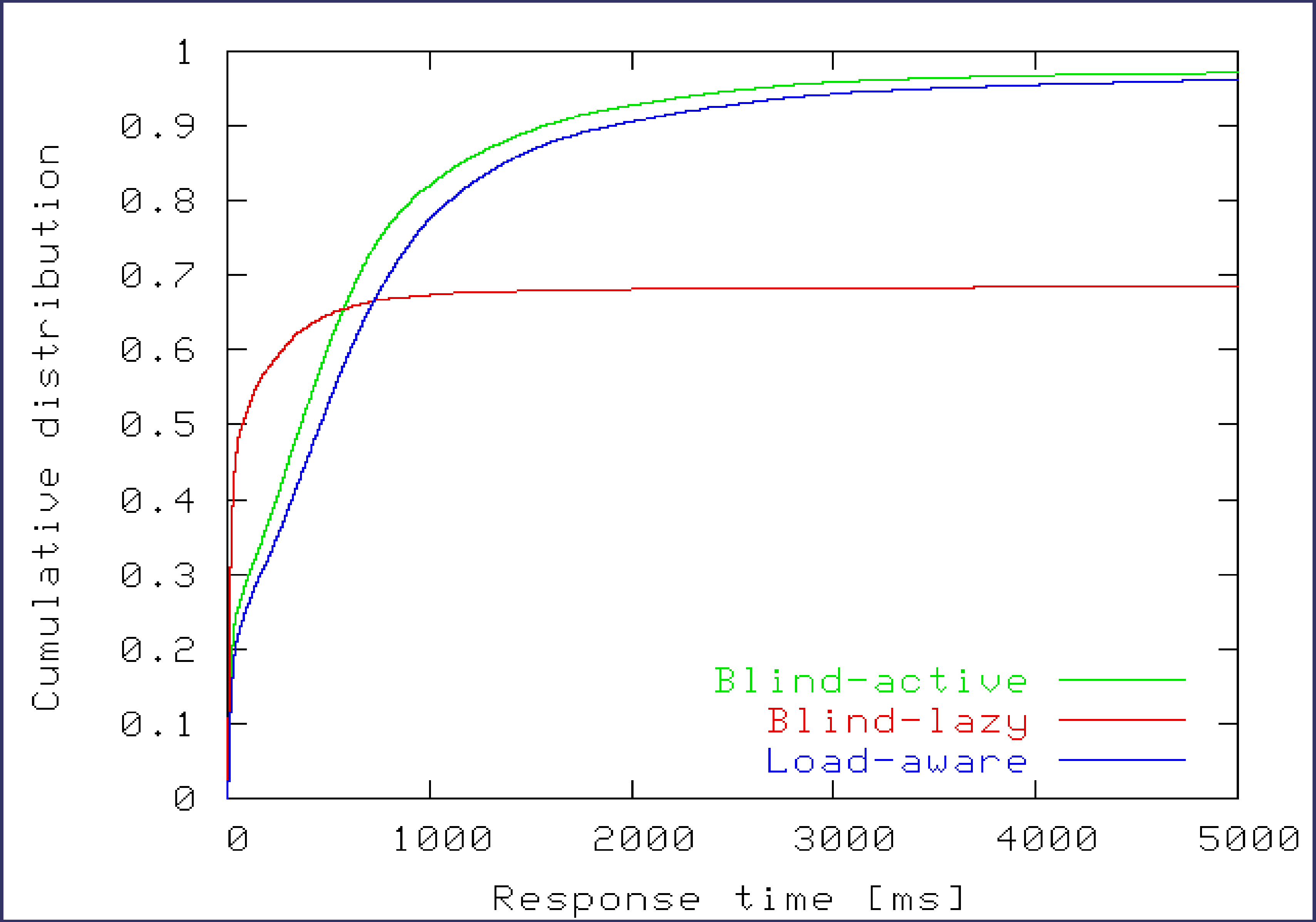
Workload

- Heavy trans-load working set
- Two request distribution
    - Bimodal 10%-90%
    - Uniform

Experimental results (bimodal workload)



Experimental results (uniform workload)

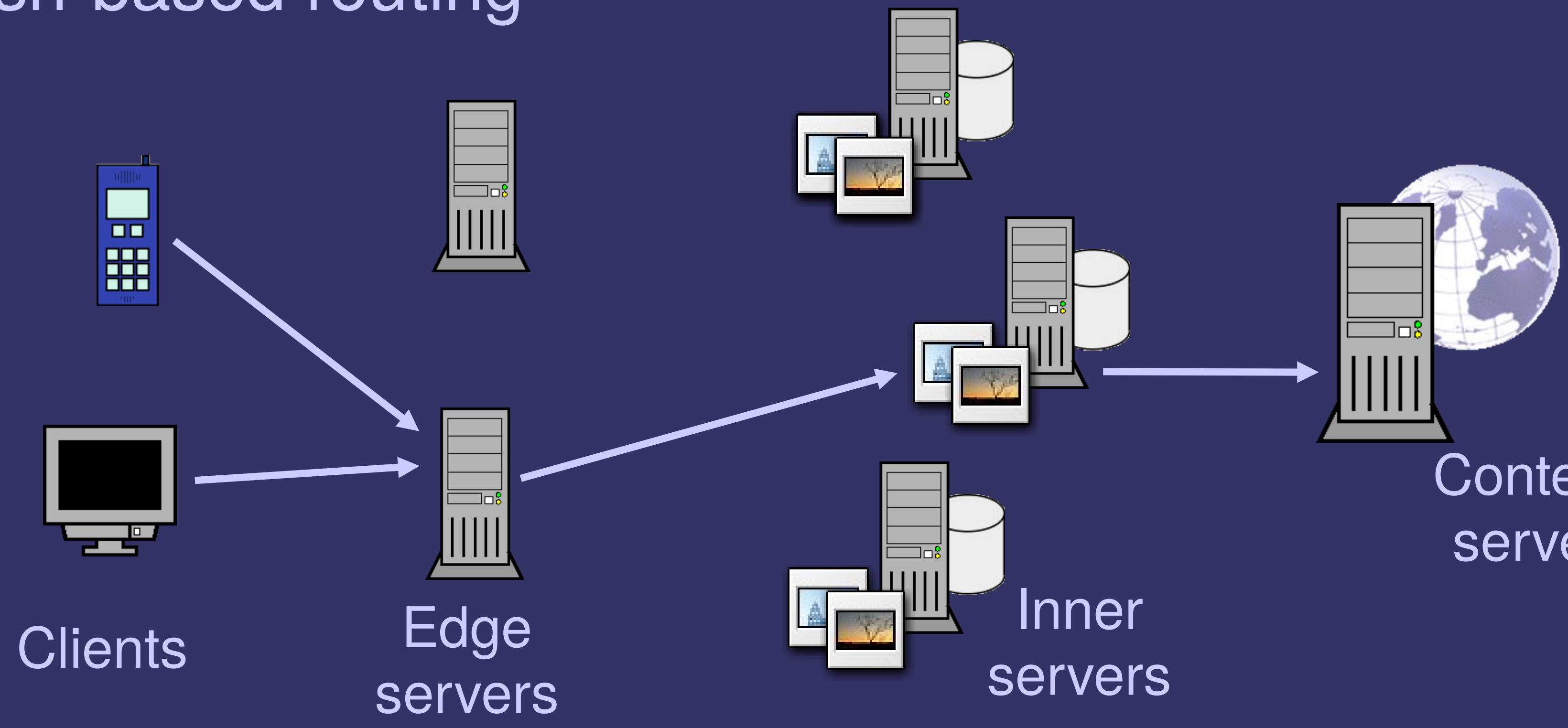


Summary of experiments

- Blind-lazy
  - Best median response time, worst 90-percentile
- Blind-active
  - Best with uniform workload
- Load-aware
  - Best with heavily skewed workload
- Limited performace gain
- Load awareness do not address working set growth issues

Two-level architecture (future work)

- Issues related to load partially addressed
  - First solution: Load sharing algorithms
  - Second solution: Two-level architecture (new architecture)
- Hash-based routing



Two-level architecture (future work)

- Address both issues
  - *Increased working set size*
    - Avoid duplicates $\rightarrow$ efficient cache usage
  - *Transcoding computational load*
    - Hash function $\rightarrow$ load sharing
    - Higher cache hit rate $\rightarrow$ reduced load
- Preliminary results (90-percentile resp. time):
  - 2x Flat query-based
  - 5x No cooperation

Cooperative Architectures and Algorithms for Discovery and Transcoding of Multi-version Content

For more information:

http://weblab.ing.unimo.it/research/trans_caching.shtml