Helping Hand or Hidden Hurdle: Proxy-assisted HTTP-based Adaptive Streaming Performance

Vengatanathan Krishnamoorthi[1], **Niklas Carlsson[1]**, Derek Eager[2], Anirban Mahanti[3], Nahid Shahmehri[1]

[1] Linköping University, Sweden

[2] University of Saskatchewan, Canada

[3] NICTA, Australia

LiU EXPANDING REALITY

Proc. MASCOTS, San Francisco, USA, August 15, 2013

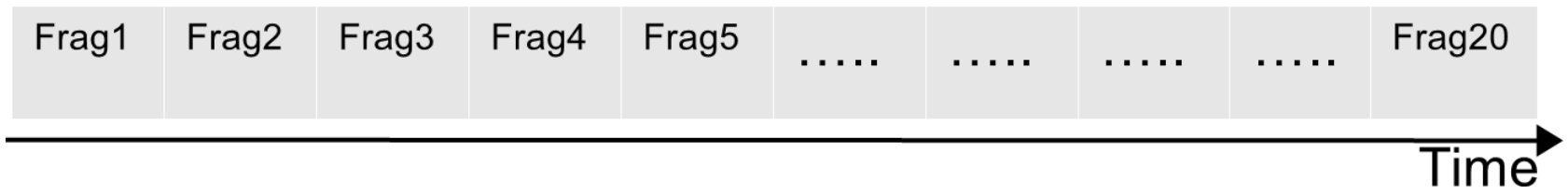
Video streaming

40→60%

of Internet traffic

- Content delivery over the Internet is massive …
  - Consume significant resources
  - How to make scalable and efficient?

HTTP-based streaming



- HTTP-based streaming
  - Allows easy caching, NAT/firewall traversal, etc.
  - Use of TCP provides natural bandwidth adaptation
  - Split into fragments, download sequentially
  - Some support for interactive VoD

HTTP-based adaptive streaming (HAS)

Quality

Frag1	Frag2	Frag3	Frag4	Frag5	.....	.....	.....	.....	@1300 Kbit/s
Frag1	Frag2	Frag3	Frag4	Frag5	.....	.....	.....	.....	@850 Kbit/s
Frag1	Frag2	Frag3	Frag4	Frag5	.....	.....	.....	.....	@500 Kbit/s
Frag1	Frag2	Frag3	Frag4	Frag5	.....	.....	.....	.....	@250 Kbit/s

Time

- HTTP-based adaptive streaming
  - Multiple encodings of each fragment (defined in manifest file)
  - Clients adapt quality encoding based on (buffer and network) conditions

Problem: Proxy-assisted HAS

Clients' want

- High playback quality
- Small stall times
- Few buffer interruptions
- Few quality switches

Problem: Proxy-assisted HAS

Clients' want

- High playback quality
- Small stall times
- Few buffer interruptions
- Few quality switches

HAS is increasingly responsible for larger traffic volumes

Network and service providers may consider integrating HAS-aware proxy policies

Problem: Proxy-assisted HAS

Clients' want

- High playback quality
- Small stall times
- Few buffer interruptions
- Few quality switches

Network providers' want

- High QoE of customers/clients

Problem: Proxy-assisted HAS

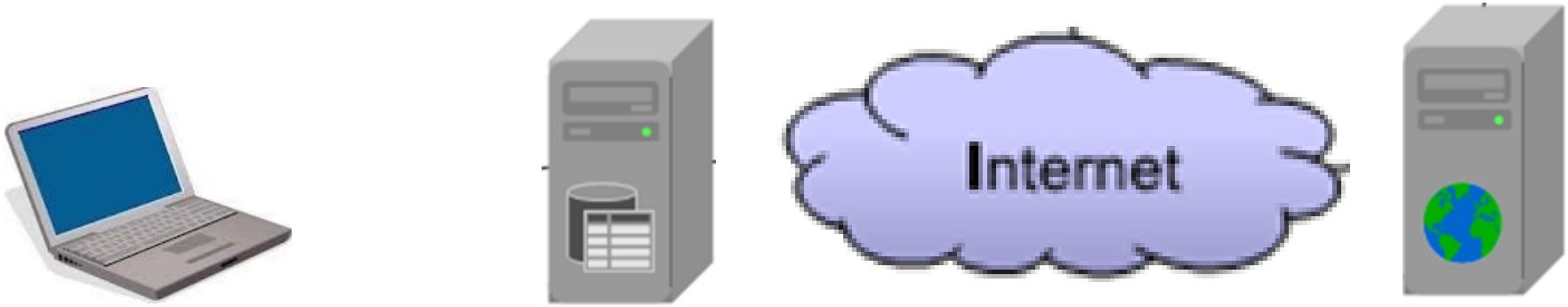
Clients' want

- High playback quality
- Small stall times
- Few buffer interruptions
- Few quality switches

Network providers' want

- High QoE of customers/clients
- Low bandwidth usage
- High hit rate

Problem: Proxy-assisted HAS



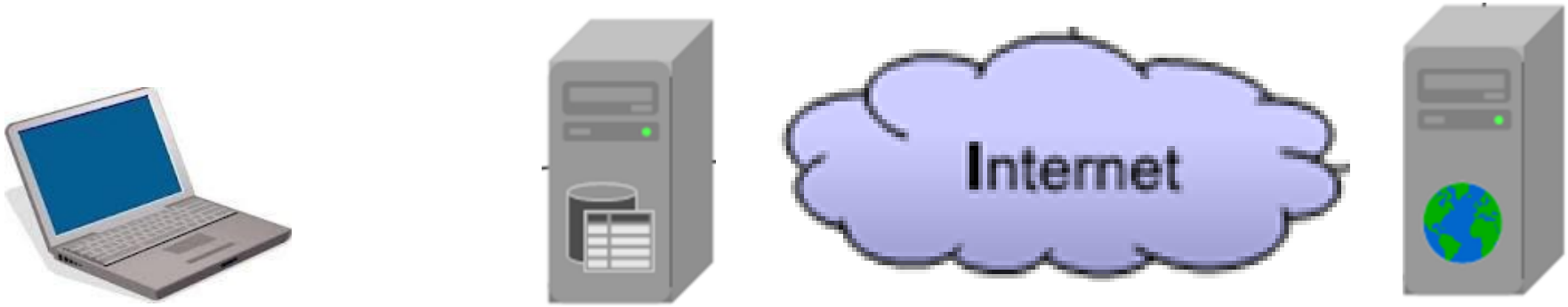
Clients' want

- High playback quality
- Small stall times
- Few buffer interruptions
- Few quality switches

Network providers' want

- High QoE of customers/clients
- Low bandwidth usage
- High hit rate

Problem: Proxy-assisted HAS



In this paper …

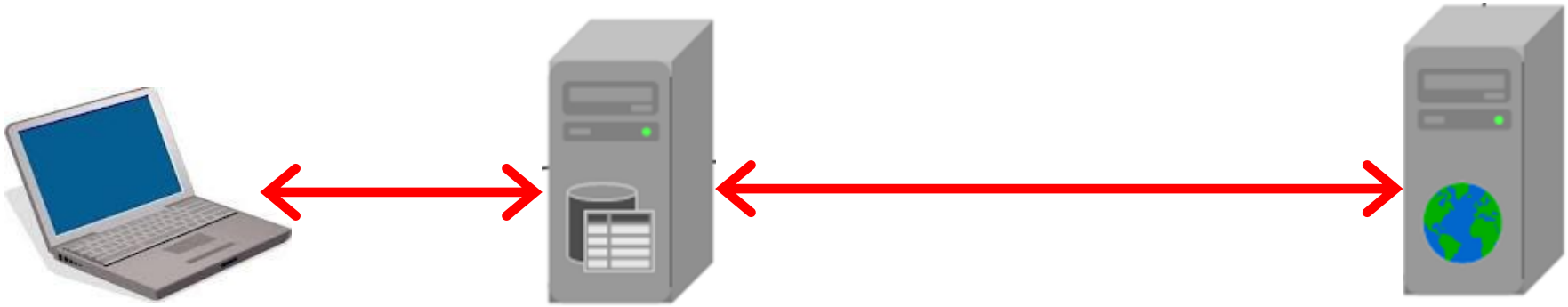
- Evaluation of proxy-assisted HAS policies

Problem: Proxy-assisted HAS

In this paper …

- Evaluation of proxy-assisted HAS policies

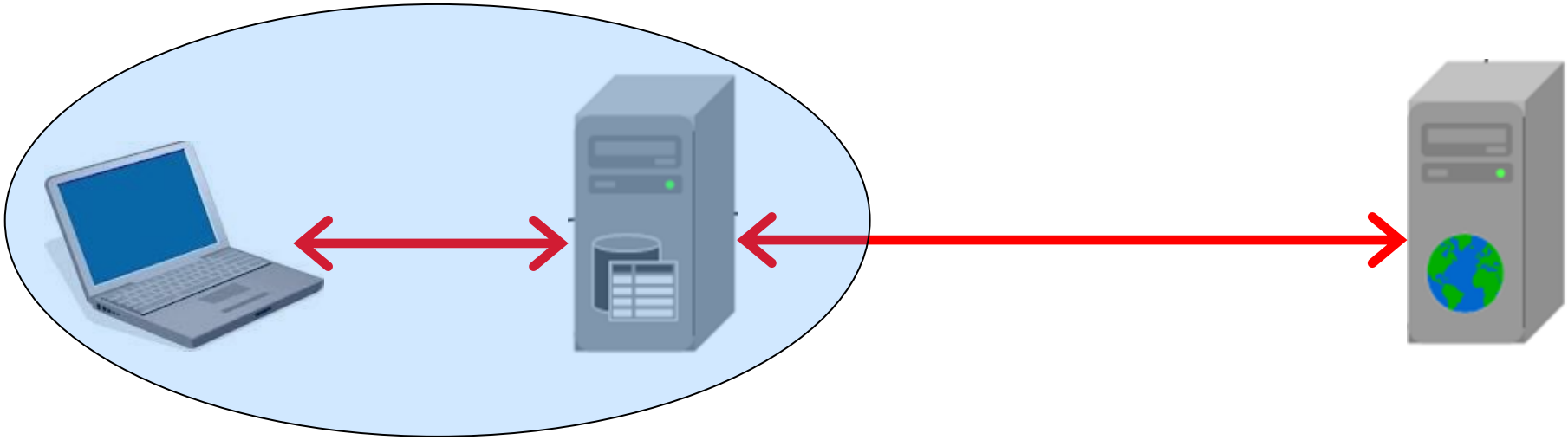
Problem: Proxy-assisted HAS



In this paper …

- Evaluation of proxy-assisted HAS policies

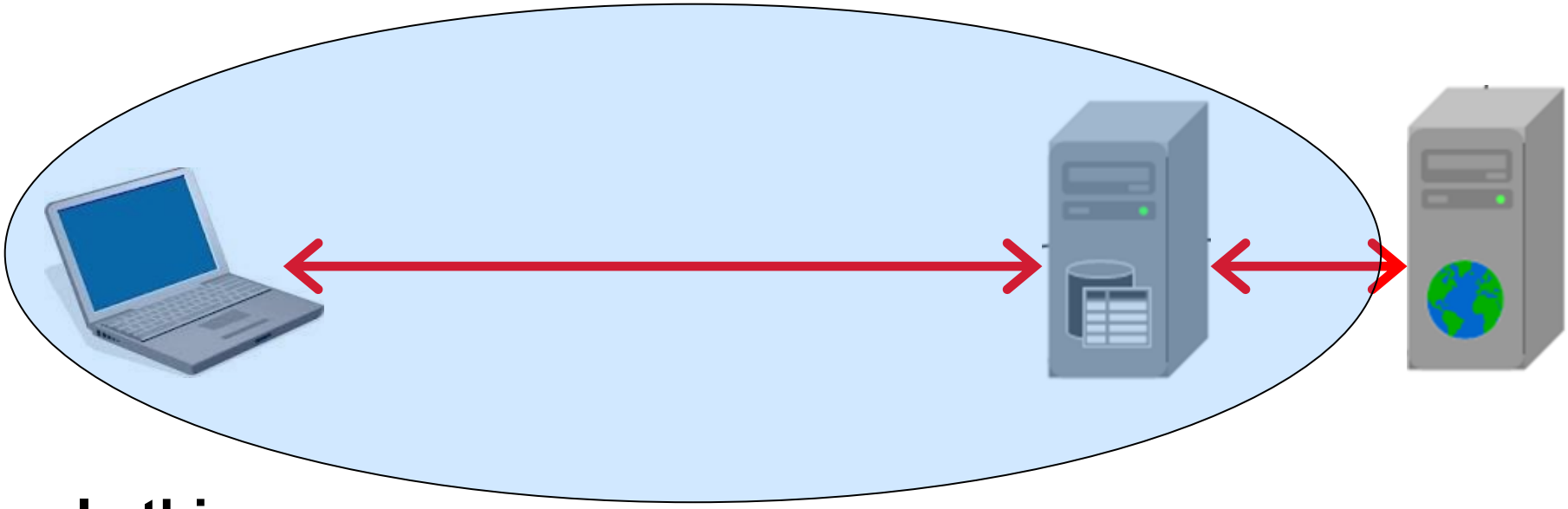
Problem: Proxy-assisted HAS



In this paper …

- Evaluation of proxy-assisted HAS policies

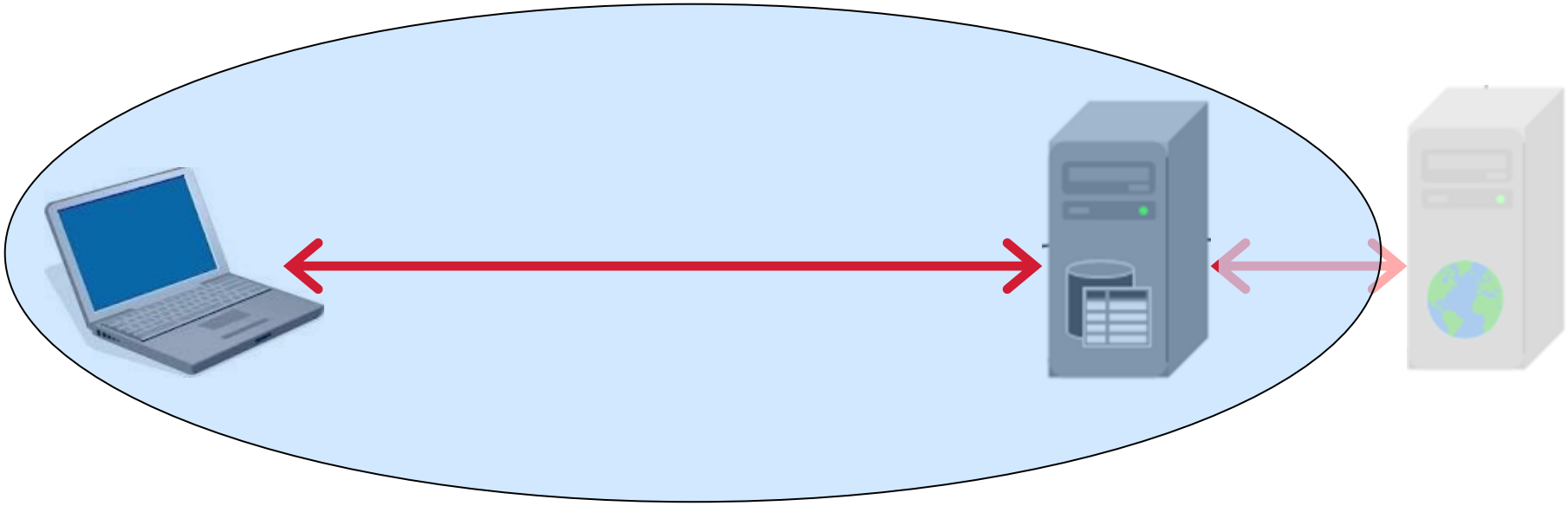
Problem: Proxy-assisted HAS



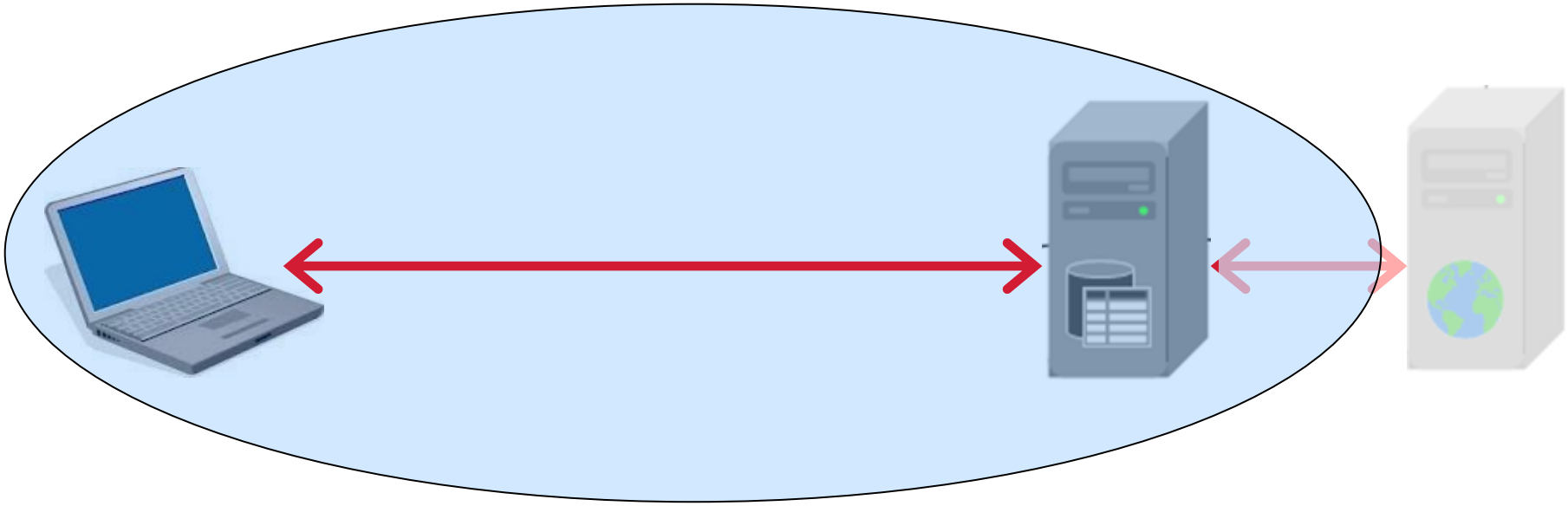
In this paper …

- Evaluation of proxy-assisted HAS policies

Example: Default “best-effort”



Example: Default “best-effort”



Quality

1,4	2,4	3,4	4,4	5,4	6,4	7,4	…			…	@1300 Kbit/s
1,3	2,3	3,3	4,3	5,3	6,3	7,3	…			…	@850 Kbit/s
1,2	2,2	3,2	4,2	5,2	6,2	7,2	…			…	@500 Kbit/s
1,1	2,1	3,1	4,1	5,1	6,1	7,1	…			…	@250 Kbit/s

Time

Client

Quality

1,4	2,4	3,4	4,4	5,4	6,4	7,4	…			…	@1300 Kbit/s
1,3	2,3	3,3	4,3	5,3	6,3	7,3	…			…	@850 Kbit/s
1,2	2,2	3,2	4,2	5,2	6,2	7,2	…			…	@500 Kbit/s
1,1	2,1	3,1	4,1	5,1	6,1	7,1	…			…	@250 Kbit/s

Time

Proxy

Example: Default “best-effort”

Quality

1,4	2,4	3,4	4,4	5,4	6,4	7,4	…		…	@1300 Kbit/s
1,3	2,3	3,3	4,3	5,3	6,3	7,3	…		…	@850 Kbit/s
1,2	2,2	3,2	4,2	5,2	6,2	7,2	…		…	@500 Kbit/s
1,1	2,1	3,1	4,1	5,1	6,1	7,1	…		…	@250 Kbit/s

Time

Client 1

Quality

1,4	2,4	3,4	4,4	5,4	6,4	7,4	…		…	@1300 Kbit/s
1,3	2,3	3,3	4,3	5,3	6,3	7,3	…		…	@850 Kbit/s
1,2	2,2	3,2	4,2	5,2	6,2	7,2	…		…	@500 Kbit/s
1,1	2,1	3,1	4,1	5,1	6,1	7,1	…		…	@250 Kbit/s

Time

Proxy before

Quality

1,4	2,4	3,4	4,4	5,4	6,4	7,4	…		…	@1300 Kbit/s
1,3	2,3	3,3	4,3	5,3	6,3	7,3	…		…	@850 Kbit/s
1,2	2,2	3,2	4,2	5,2	6,2	7,2	…		…	@500 Kbit/s
1,1	2,1	3,1	4,1	5,1	6,1	7,1	…		…	@250 Kbit/s

Time

Proxy after

Example: Default “best-effort”

Quality

1,4	2,4	3,4	4,4	5,4	6,4	7,4	…		…	@1300 Kbit/s
1,3	2,3	3,3	4,3	5,3	6,3	7,3	…		…	@850 Kbit/s
1,2	2,2	3,2	4,2	5,2	6,2	7,2	…		…	@500 Kbit/s
1,1	2,1	3,1	4,1	5,1	6,1	7,1	…		…	@250 Kbit/s

Time

Client 1

Quality

1,4	2,4	3,4	4,4	5,4	6,4	7,4	…		…	@1300 Kbit/s
1,3	2,3	3,3	4,3	5,3	6,3	7,3	…		…	@850 Kbit/s
1,2	2,2	3,2	4,2	5,2	6,2	7,2	…		…	@500 Kbit/s
1,1	2,1	3,1	4,1	5,1	6,1	7,1	…		…	@250 Kbit/s

Time

Proxy before

Quality

1,4	2,4	3,4	4,4	5,4	6,4	7,4	…		…	@1300 Kbit/s
1,3	2,3	3,3	4,3	5,3	6,3	7,3	…		…	@850 Kbit/s
1,2	2,2	3,2	4,2	5,2	6,2	7,2	…		…	@500 Kbit/s
1,1	2,1	3,1	4,1	5,1	6,1	7,1	…		…	@250 Kbit/s

Time

Proxy after

Example: Default “best-effort”

Quality

1,4	2,4	3,4	4,4	5,4	6,4	7,4	…			…	@1300 Kbit/s
1,3	2,3	3,3	4,3	5,3	6,3	7,3	…			…	@850 Kbit/s
1,2	2,2	3,2	4,2	5,2	6,2	7,2	…			…	@500 Kbit/s
1,1	2,1	3,1	4,1	5,1	6,1	7,1	…			…	@250 Kbit/s

Time

Client 2

Quality

1,4	2,4	3,4	4,4	5,4	6,4	7,4	…			…	@1300 Kbit/s
1,3	2,3	3,3	4,3	5,3	6,3	7,3	…			…	@850 Kbit/s
1,2	2,2	3,2	4,2	5,2	6,2	7,2	…			…	@500 Kbit/s
1,1	2,1	3,1	4,1	5,1	6,1	7,1	…			…	@250 Kbit/s

Time

Proxy before

Quality

1,4	2,4	3,4	4,4	5,4	6,4	7,4	…			…	@1300 Kbit/s
1,3	2,3	3,3	4,3	5,3	6,3	7,3	…			…	@850 Kbit/s
1,2	2,2	3,2	4,2	5,2	6,2	7,2	…			…	@500 Kbit/s
1,1	2,1	3,1	4,1	5,1	6,1	7,1	…			…	@250 Kbit/s

Time

Proxy after

Example: Default “best-effort”

Quality

1,4	2,4	3,4	4,4	5,4	6,4	7,4	…		…	@1300 Kbit/s
1,3	2,3	3,3	4,3	5,3	6,3	7,3	…		…	@850 Kbit/s
1,2	2,2	3,2	4,2	5,2	6,2	7,2	…		…	@500 Kbit/s
1,1	2,1	3,1	4,1	5,1	6,1	7,1	…		…	@250 Kbit/s

Time

Client 2

Quality

1,4	2,4	3,4	4,4	5,4	6,4	7,4	…		…	@1300 Kbit/s
1,3	2,3	3,3	4,3	5,3	6,3	7,3	…		…	@850 Kbit/s
1,2	2,2	3,2	4,2	5,2	6,2	7,2	…		…	@500 Kbit/s
1,1	2,1	3,1	4,1	5,1	6,1	7,1	…		…	@250 Kbit/s

Time

Proxy before

Quality

1,4	2,4	3,4	4,4	5,4	6,4	7,4	…		…	@1300 Kbit/s
1,3	2,3	3,3	4,3	5,3	6,3	7,3	…		…	@850 Kbit/s
1,2	2,2	3,2	4,2	5,2	6,2	7,2	…		…	@500 Kbit/s
1,1	2,1	3,1	4,1	5,1	6,1	7,1	…		…	@250 Kbit/s

Time

Proxy after

Example: Default “best-effort”

Quality

1,4	2,4	3,4	4,4	5,4	6,4	7,4	…			…	@1300 Kbit/s
1,3	2,3	3,3	4,3	5,3	6,3	7,3	…			…	@850 Kbit/s
1,2	2,2	3,2	4,2	5,2	6,2	7,2	…			…	@500 Kbit/s
1,1	2,1	3,1	4,1	5,1	6,1	7,1	…			…	@250 Kbit/s

Time

Client 3

Quality

1,4	2,4	3,4	4,4	5,4	6,4	7,4	…			…	@1300 Kbit/s
1,3	2,3	3,3	4,3	5,3	6,3	7,3	…			…	@850 Kbit/s
1,2	2,2	3,2	4,2	5,2	6,2	7,2	…			…	@500 Kbit/s
1,1	2,1	3,1	4,1	5,1	6,1	7,1	…			…	@250 Kbit/s

Time

Proxy before

Quality

1,4	2,4	3,4	4,4	5,4	6,4	7,4	…			…	@1300 Kbit/s
1,3	2,3	3,3	4,3	5,3	6,3	7,3	…			…	@850 Kbit/s
1,2	2,2	3,2	4,2	5,2	6,2	7,2	…			…	@500 Kbit/s
1,1	2,1	3,1	4,1	5,1	6,1	7,1	…			…	@250 Kbit/s

Time

Proxy after

Example: Default “best-effort”

Quality

1,4	2,4	3,4	4,4	5,4	6,4	7,4	…			…	@1300 Kbit/s
1,3	2,3	3,3	4,3	5,3	6,3	7,3	…			…	@850 Kbit/s
1,2	2,2	3,2	4,2	5,2	6,2	7,2	…			…	@500 Kbit/s
1,1	2,1	3,1	4,1	5,1	6,1	7,1	…			…	@250 Kbit/s

Time

Client 3

Quality

1,4	2,4	3,4	4,4	5,4	6,4	7,4	…			…	@1300 Kbit/s
1,3	2,3	3,3	4,3	5,3	6,3	7,3	…			…	@850 Kbit/s
1,2	2,2	3,2	4,2	5,2	6,2	7,2	…			…	@500 Kbit/s
1,1	2,1	3,1	4,1	5,1	6,1	7,1	…			…	@250 Kbit/s

Time

Proxy before

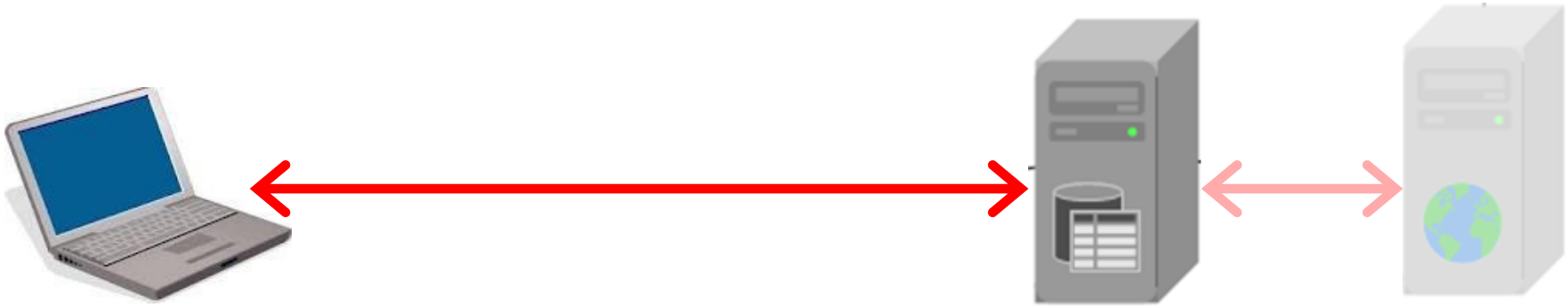
Quality

1,4	2,4	3,4	4,4	5,4	6,4	7,4	…			…	@1300 Kbit/s
1,3	2,3	3,3	4,3	5,3	6,3	7,3	…			…	@850 Kbit/s
1,2	2,2	3,2	4,2	5,2	6,2	7,2	…			…	@500 Kbit/s
1,1	2,1	3,1	4,1	5,1	6,1	7,1	…			…	@250 Kbit/s

Time

Proxy after

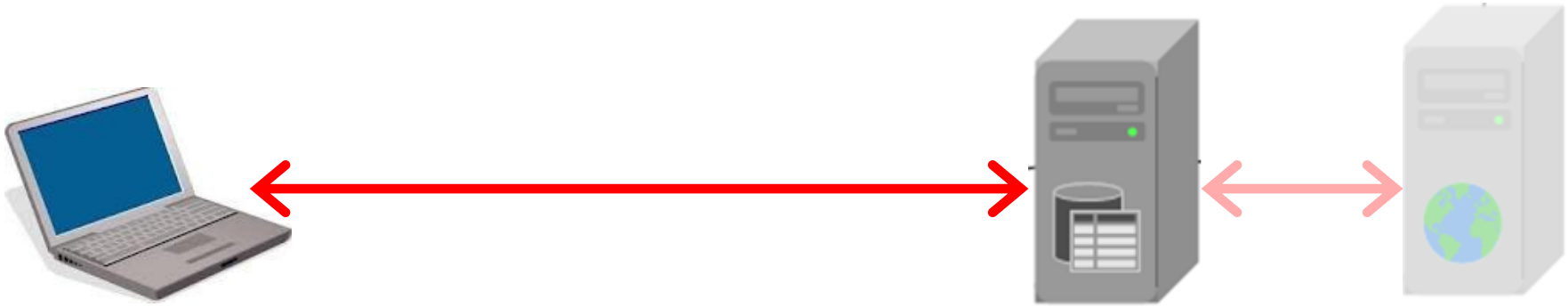
Policies and policy classes



Baseline policies

- Empty cache
- Full cache (preload all versions)
- Best effort (default, as previous example)

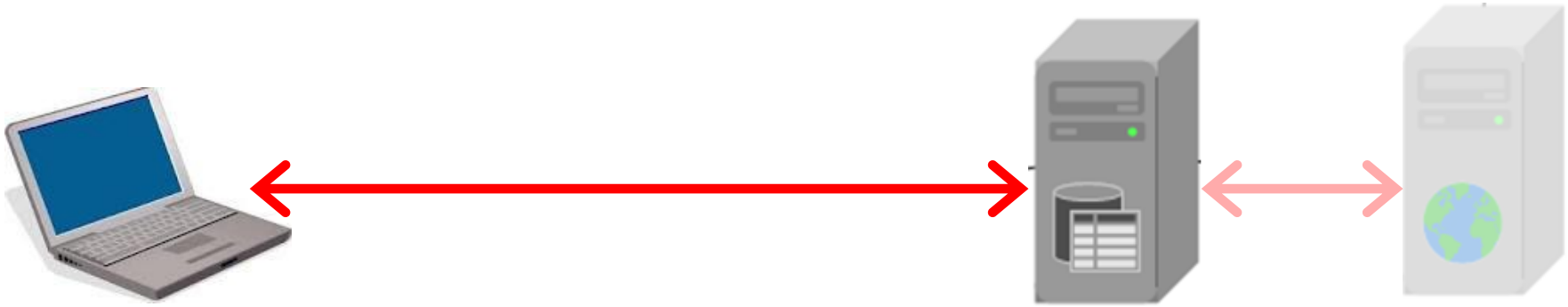
Policies and policy classes



Quality and content-aware prefetching policies

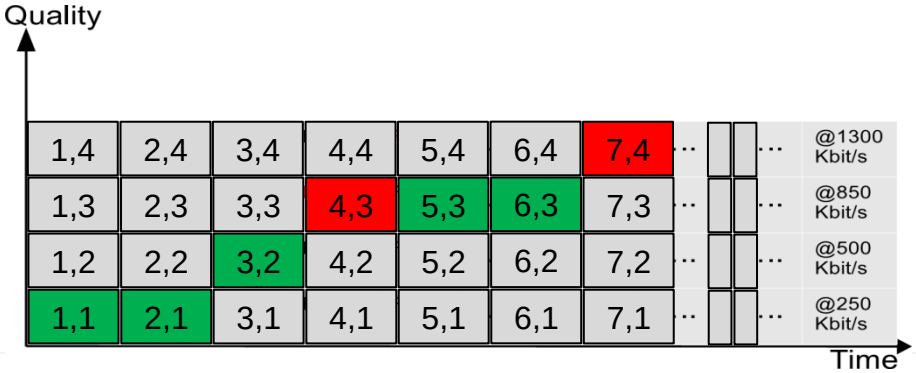
- 1-ahead
- N-ahead
- Priority-based

Policies and policy classes

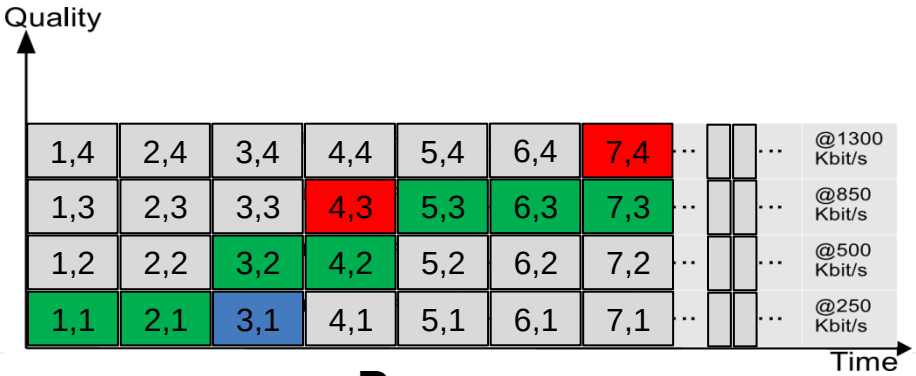


Quality and content-aware prefetching policies

- 1-ahead
- N-ahead
- Priority-based

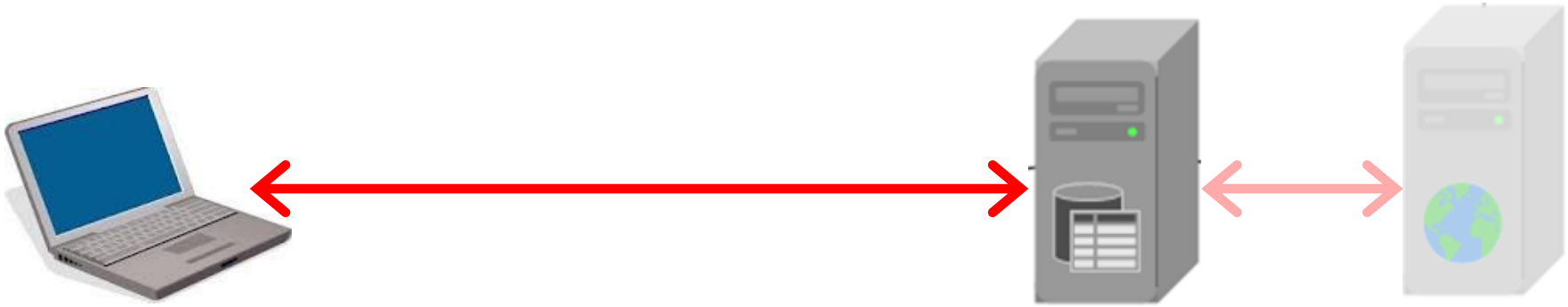


Client 2



Proxy

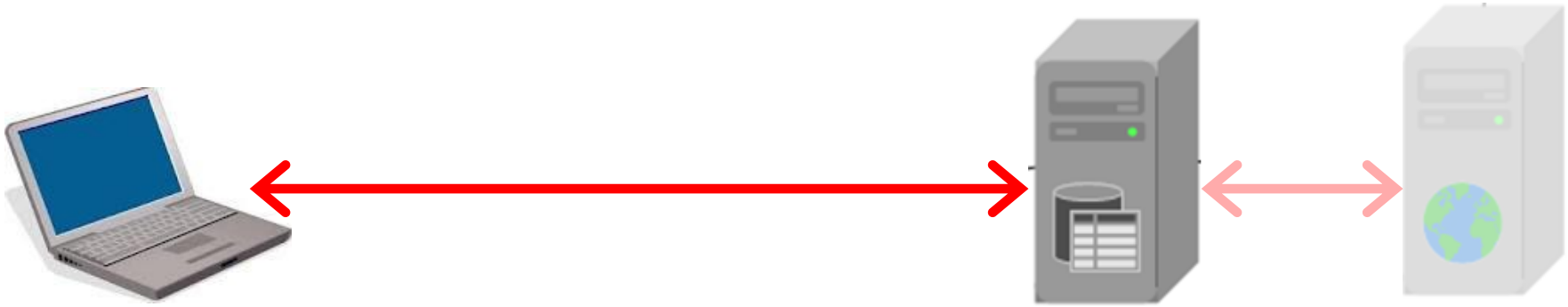
Policies and policy classes



Quality and content-aware prefetching policies

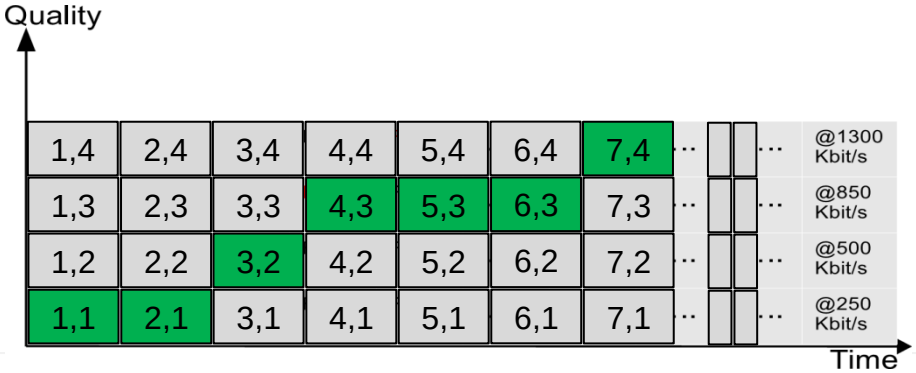
- 1-ahead
- N-ahead
- Priority-based (based on likely switches)
  - **If** client switches to a higher encoding *and* it is not the first time that the client is requesting this quality, **then** prefetch: (i) current quality, (ii) one quality level below, (iii) one quality level above, and (iv) no prefetching.
  - **Else** prefetch: (i) current quality, (ii) one quality level above, (iii) one quality level below and (iv) no prefetching.

Policies and policy classes

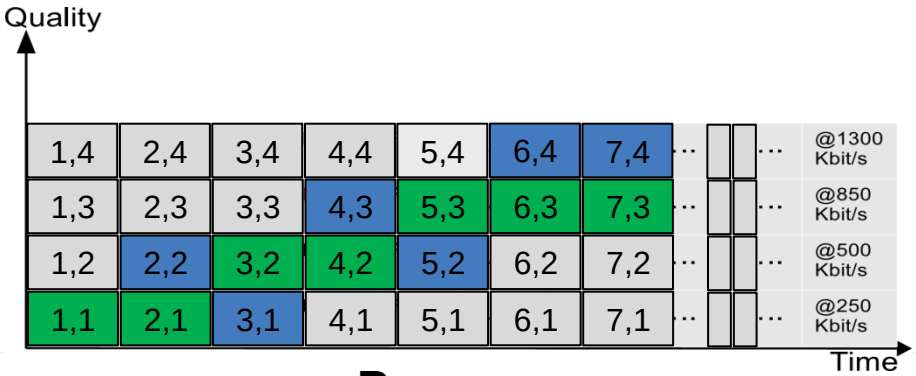


Quality and content-aware prefetching policies

- 1-ahead
- N-ahead
- Priority-based (based on likely switches)

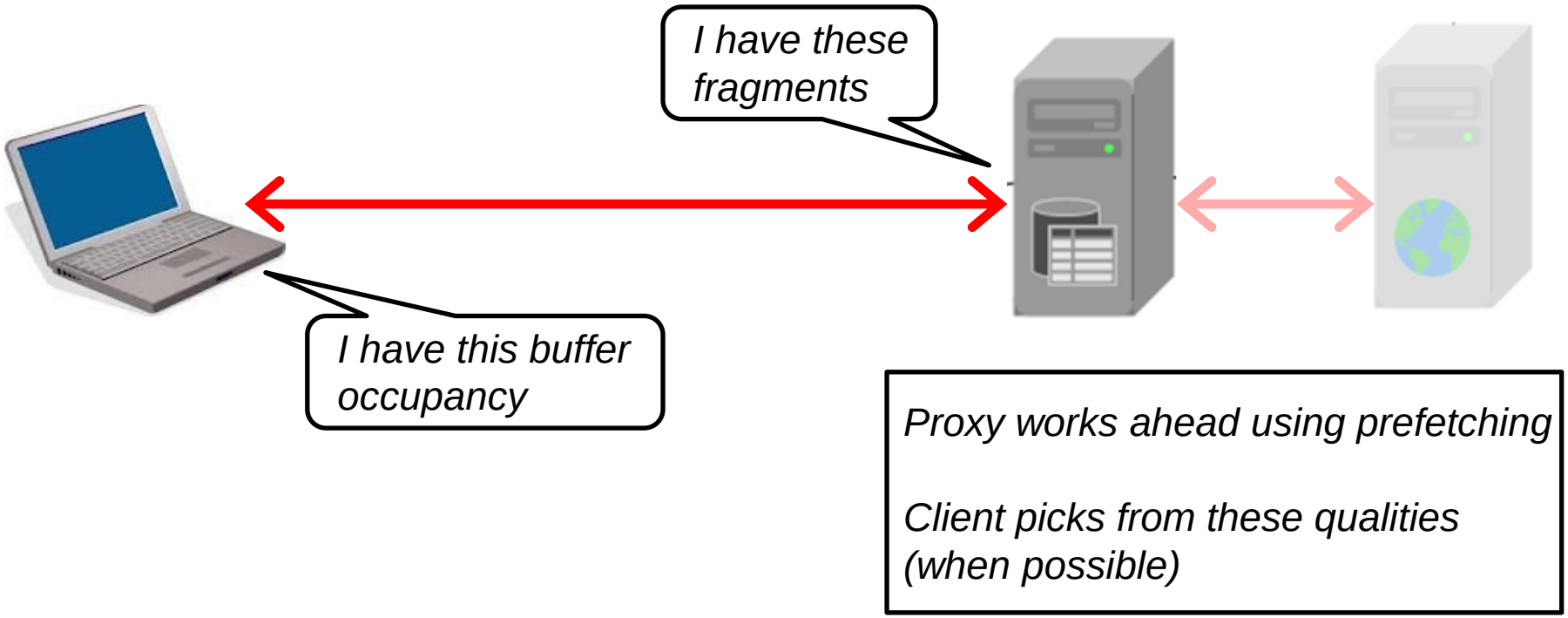


Client 2



Proxy

Policies and policy classes



Client-proxy cooperation policies

- Buffer oblivious (priority-based prefetching)
- Buffer aware (conservative quality during low buffer conditions)

Policy overview

Baseline policies

- Empty cache
- Full cache (preload all versions)
- Best effort (default, as previous example)

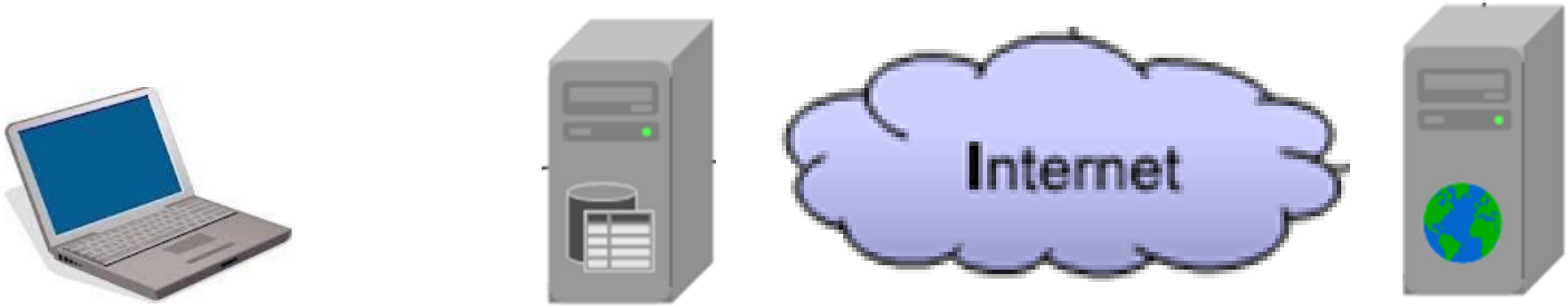
Quality and content-aware prefetching policies

- 1-ahead
- N-ahead
- Priority-based (based on likely switches)

Client-proxy cooperation policies

- Buffer oblivious (priority-based prefetching)
- Buffer aware (conservative quality during low buffer conditions)

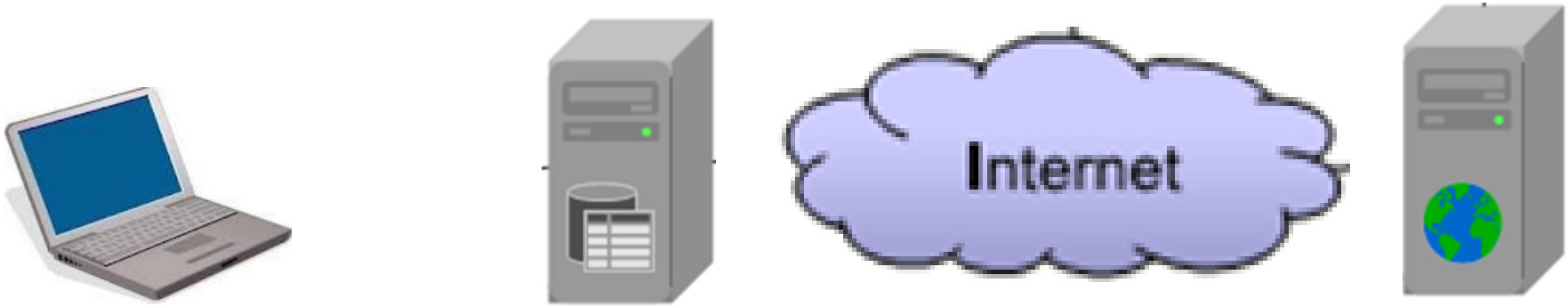
Evaluation and Instrumentation



In this paper …

- Evaluation of proxy-assisted HAS policies

Instrumentation



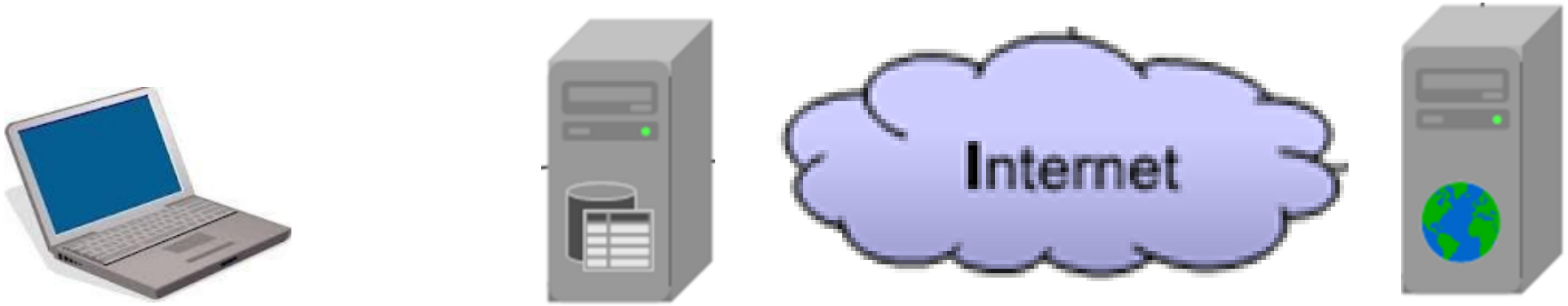
In this paper …

- Evaluation of proxy-assisted HAS policies

… for this we need instrumentation of …

- **Clients:** Measure performance/service
- **Proxy:** Implementing policies and measure performance
- **Network:** Capture network conditions (bottlenecks, bandwidths, delays, packet losses, etc.)

Instrumentation



… for this we need instrumentation of …

- Clients: Measure performance/service
- Proxy: Implementing policies and measure performance
- Network: Capture network conditions (bottlenecks, bandwidths, delays, packet losses, etc.)

Instrumentation

… for this we need instrumentation of …

- Clients: Measure performance/service
- Proxy: Implementing policies and measure performance
- Network: Capture network conditions (bottlenecks, bandwidths, delays, packet losses, etc.)

Instrumentation

Microsoft Silverlight

NETFLIX

… for this we need instrumentation of …

- Clients: Measure performance/service
- Proxy: Implementing policies and measure performance
- Network: Capture network conditions (bottlenecks, bandwidths, delays, packet losses, etc.)

Instrumentation

	Player	Container	Type	Open Source
Microsoft Silverlight				
NETFLIX				

… for this we need instrumentation of …

- Clients: Measure performance/service
- Proxy: Implementing policies and measure performance
- Network: Capture network conditions (bottlenecks, bandwidths, delays, packet losses, etc.)

Instrumentation

	Player	Container	Type	Open Source
Microsoft Silverlight	Microsoft Smooth Streaming	Silverlight	Chunk	✗
NETFLIX				

… for this we need instrumentation of …

- Clients: Measure performance/service
- Proxy: Implementing policies and measure performance
- Network: Capture network conditions (bottlenecks, bandwidths, delays, packet losses, etc.)

Instrumentation

	Player	Container	Type	Open Source
Microsoft Silverlight	Microsoft Smooth Streaming	Silverlight	Chunk	✗
NETFLIX	Netflix player	Silverlight	Range	✗

… for this we need instrumentation of …

- Clients: Measure performance/service
- Proxy: Implementing policies and measure performance
- Network: Capture network conditions (bottlenecks, bandwidths, delays, packet losses, etc.)

Instrumentation

	Player	Container	Type	Open Source
Microsoft Silverlight	Microsoft Smooth Streaming	Silverlight	Chunk	✗
NETFLIX	Netflix player	Silverlight	Range	✗
	Apple HLS	QuickTime	Chunk	✗

… for this we need instrumentation of …

- Clients: Measure performance/service
- Proxy: Implementing policies and measure performance
- Network: Capture network conditions (bottlenecks, bandwidths, delays, packet losses, etc.)

Instrumentation

	Player	Container	Type	Open Source
Microsoft Silverlight	Microsoft Smooth Streaming	Silverlight	Chunk	✗
NETFLIX	Netflix player	Silverlight	Range	✗
	Apple HLS	QuickTime	Chunk	✗
	Adobe HDS	Flash	Chunk	✓

… for this we need instrumentation of …

- Clients: Measure performance/service
- Proxy: Implementing policies and measure performance
- Network: Capture network conditions (bottlenecks, bandwidths, delays, packet losses, etc.)

Instrumentation

Adobe's OSMF v1.6 and v 2.0

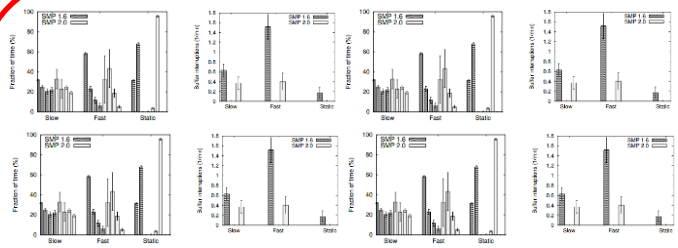
Adobe Flash media server 4.5

… for this we need instrumentation of …

- Clients: Measure performance/service
- Proxy: Implementing policies and measure performance
- Network: Capture network conditions (bottlenecks, bandwidths, delays, packet losses, etc.)

Instrumentation

Adobe's OSMF v1.6 and v 2.0



New player (v 2.0) better for all metrics and scenarios

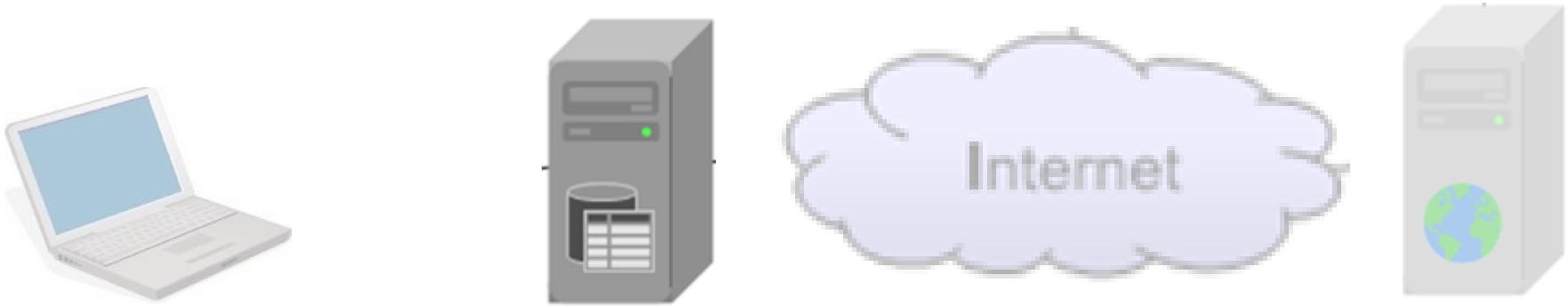
(also looked at buffer size)

Adobe Flash media server 4.5

… for this we need instrumentation of …

- Clients: Measure performance/service
- Proxy: Implementing policies and measure performance
- Network: Capture network conditions (bottlenecks, bandwidths, delays, packet losses, etc.)

Instrumentation



… for this we need instrumentation of …

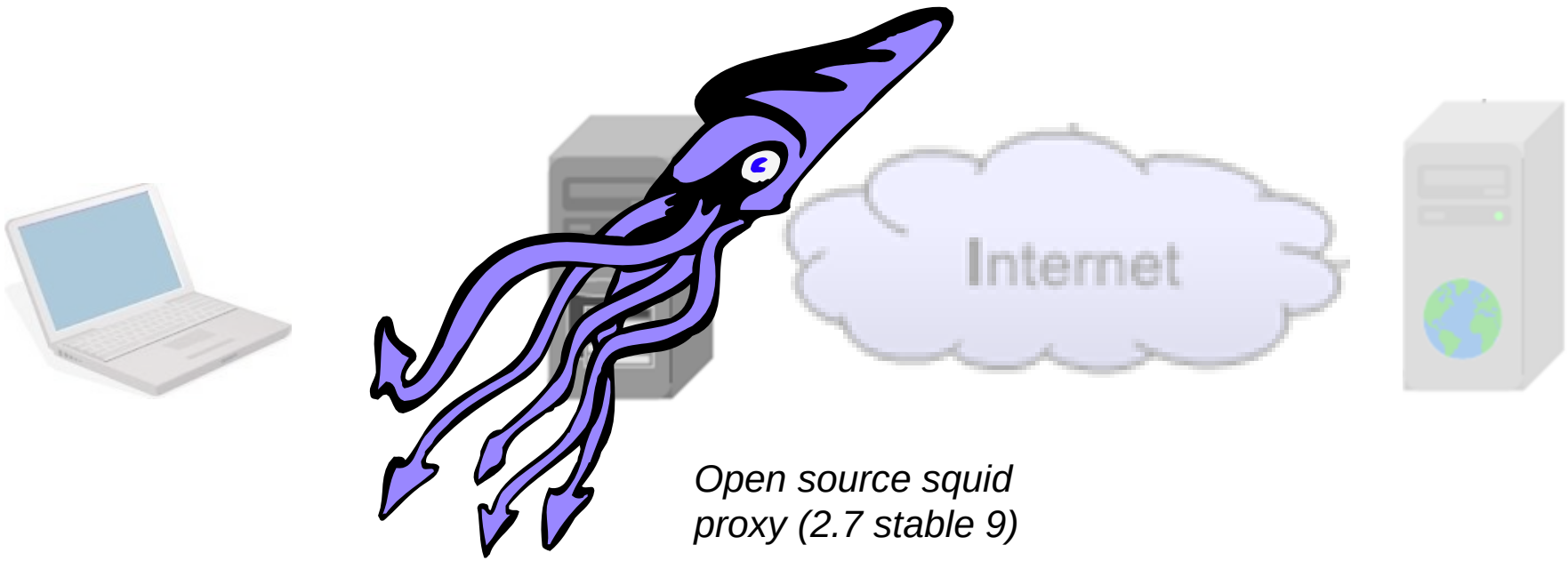
- Clients: Measure performance/service
- Proxy: Implementing policies and measure performance
- Network: Capture network conditions (bottlenecks, bandwidths, delays, packet losses, etc.)

Instrumentation

… for this we need instrumentation of …

- Clients: Measure performance/service
- Proxy: Implementing policies and measure performance
- Network: Capture network conditions (bottlenecks, bandwidths, delays, packet losses, etc.)

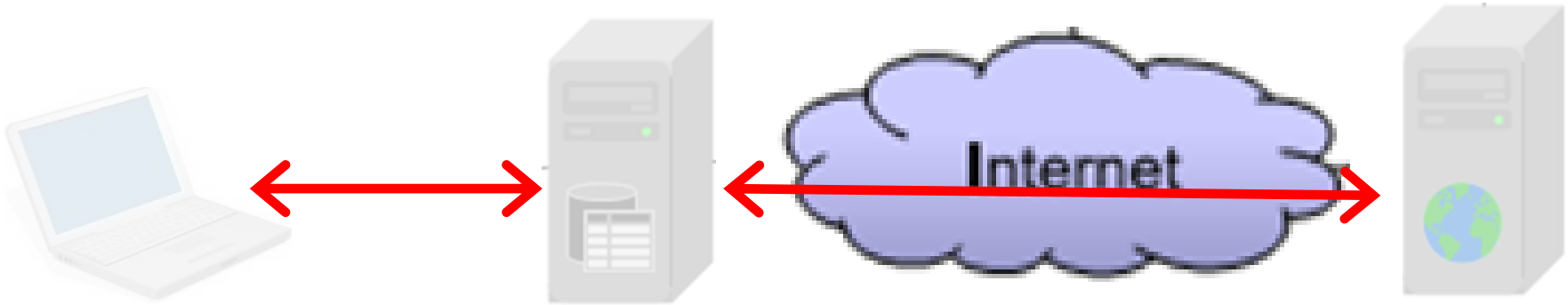
Instrumentation



… for this we need instrumentation of …

- Clients: Measure performance/service
- Proxy: Implementing policies and measure performance
- Network: Capture network conditions (bottlenecks, bandwidths, delays, packet losses, etc.)

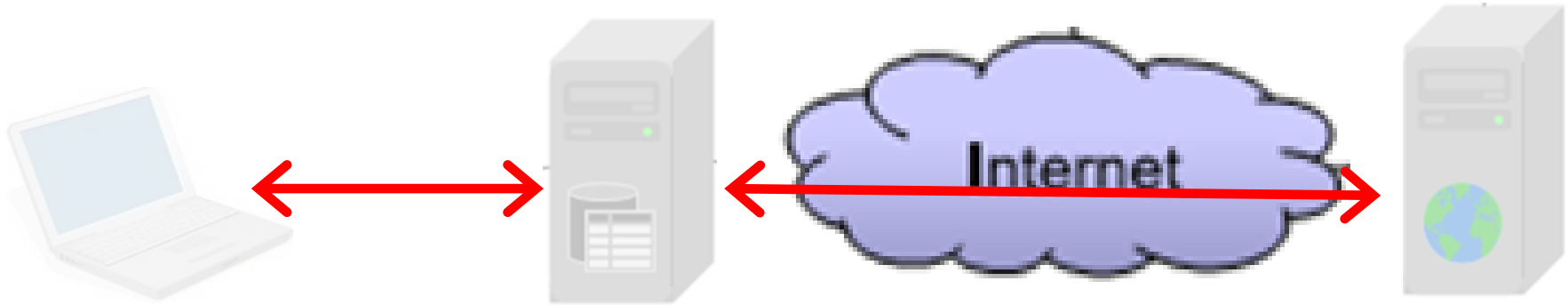
Instrumentation



… for this we need instrumentation of …

- Clients: Measure performance/service
- Proxy: Implementing policies and measure performance
- Network: Capture network conditions (bottlenecks, bandwidths, delays, packet losses, etc.)

Instrumentation

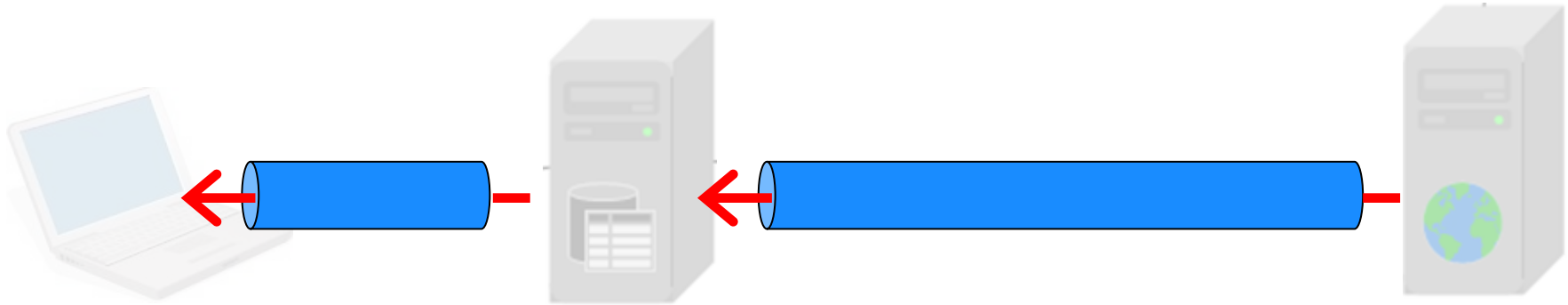


Dummynet to control network conditions
- *delays*
- *packet losses*
- *bandwidth and bottlenecks*

… for this we need instrumentation of …

- Clients: Measure performance/service
- Proxy: Implementing policies and measure performance
- Network: Capture network conditions (bottlenecks, bandwidths, delays, packet losses, etc.)

Instrumentation



Dummynet to control network conditions
- *delays*
- *packet losses*
- *bandwidth and bottlenecks*

… for this we need instrumentation of …

- Clients: Measure performance/service
- Proxy: Implementing policies and measure performance
- Network: Capture network conditions (bottlenecks, bandwidths, delays, packet losses, etc.)

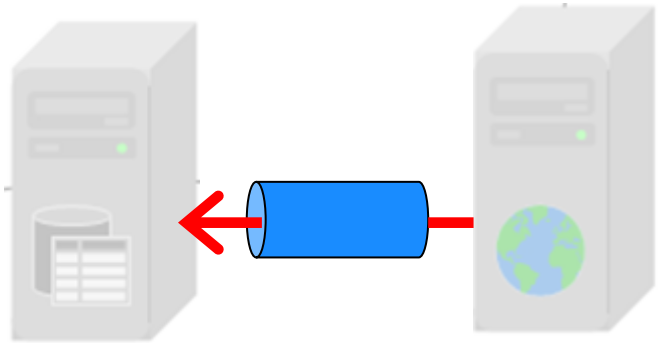
Instrumentation

Dummynet to control network conditions
- *delays*
- *packet losses*
- *bandwidth and bottlenecks*

… for this we need instrumentation of …

- Clients: Measure performance/service
- Proxy: Implementing policies and measure performance
- Network: Capture network conditions (bottlenecks, bandwidths, delays, packet losses, etc.)

Instrumentation



Dummynet to control network conditions
- *delays*
- *packet losses*
- *bandwidth and bottlenecks*

… for this we need instrumentation of …

- Clients: Measure performance/service
- Proxy: Implementing policies and measure performance
- Network: Capture network conditions (bottlenecks, bandwidths, delays, packet losses, etc.)

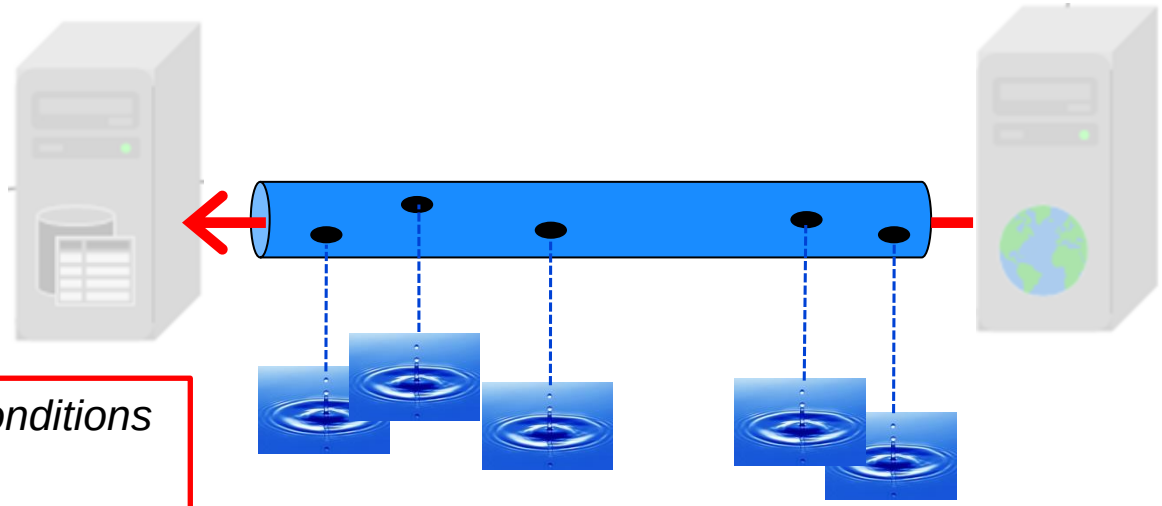
Instrumentation

Dummynet to control network conditions
- *delays*
- *packet losses*
- *bandwidth and bottlenecks*

… for this we need instrumentation of …

- Clients: Measure performance/service
- Proxy: Implementing policies and measure performance
- Network: Capture network conditions (bottlenecks, bandwidths, delays, packet losses, etc.)

Instrumentation



Dummynet to control network conditions
- *delays*
- *packet losses*
- *bandwidth and bottlenecks*

… for this we need instrumentation of …

- Clients: Measure performance/service
- Proxy: Implementing policies and measure performance
- Network: Capture network conditions (bottlenecks, bandwidths, delays, packet losses, etc.)

Instrumentation

Dummynet to control network conditions
- *delays*
- *packet losses*
- *bandwidth and bottlenecks*

… for this we need instrumentation of …

- Clients: Measure performance/service
- Proxy: Implementing policies and measure performance
- Network: Capture network conditions (bottlenecks, bandwidths, delays, packet losses, etc.)

Instrumentation

Dummynet to control network conditions
- *delays*
- *packet losses*
- *bandwidth and bottlenecks*

… for this we need instrumentation of …

- Clients: Measure performance/service
- Proxy: Implementing policies and measure performance
- Network: Capture network conditions (bottlenecks, bandwidths, delays, packet losses, etc.)

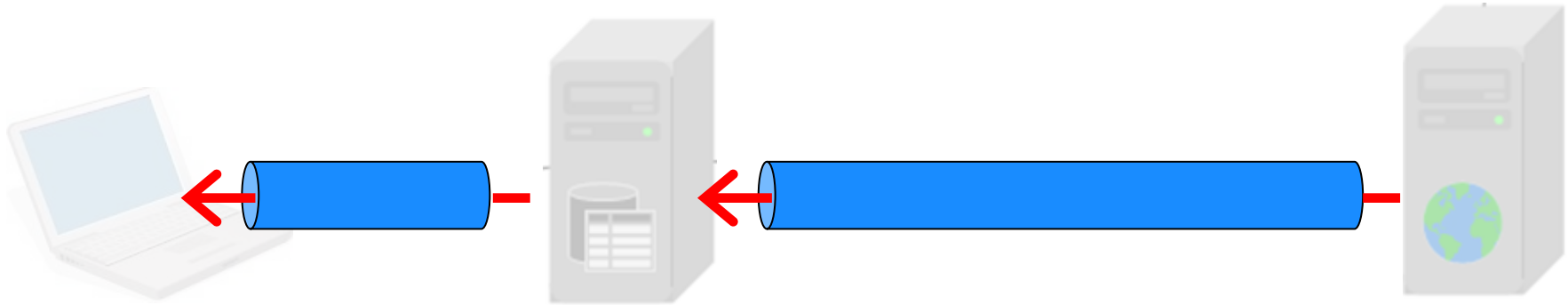
Instrumentation

Dummynet to control network conditions
- *delays*
- *packet losses*
- *bandwidth and bottlenecks*

… for this we need instrumentation of …

- Clients: Measure performance/service
- Proxy: Implementing policies and measure performance
- Network: Capture network conditions (bottlenecks, bandwidths, delays, packet losses, etc.)

Instrumentation

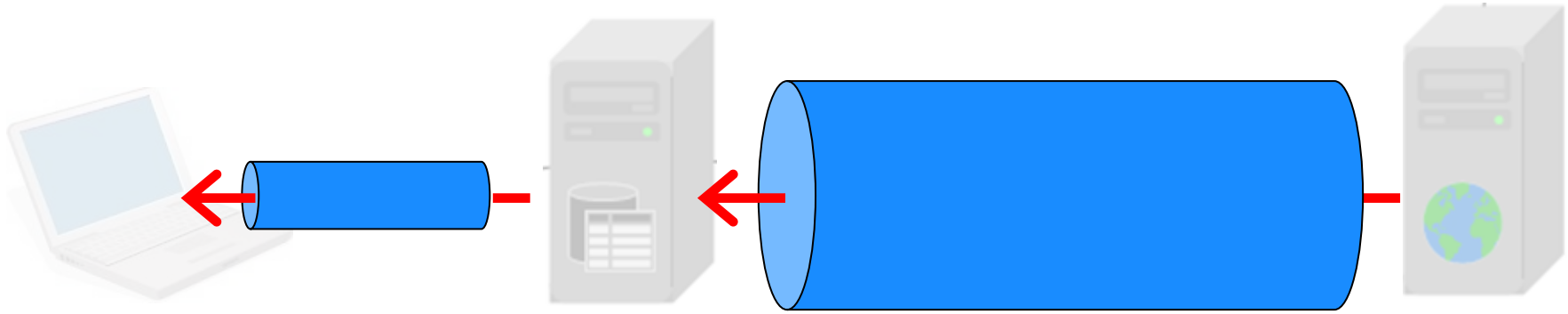


Dummynet to control network conditions
- *delays*
- *packet losses*
- *bandwidth and bottlenecks*

… for this we need instrumentation of …

- Clients: Measure performance/service
- Proxy: Implementing policies and measure performance
- Network: Capture network conditions (bottlenecks, bandwidths, delays, packet losses, etc.)

Instrumentation

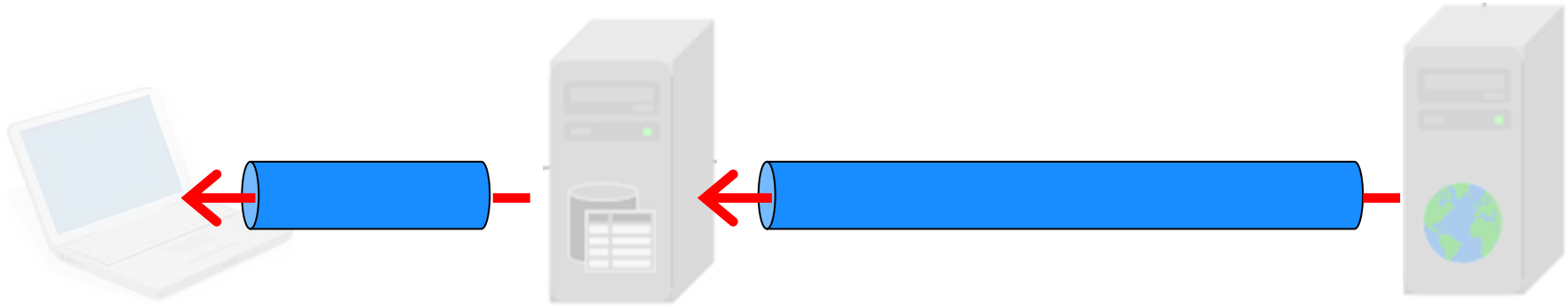


Dummynet to control network conditions
- *delays*
- *packet losses*
- *bandwidth and bottlenecks*

… for this we need instrumentation of …

- Clients: Measure performance/service
- Proxy: Implementing policies and measure performance
- Network: Capture network conditions (bottlenecks, bandwidths, delays, packet losses, etc.)

Instrumentation

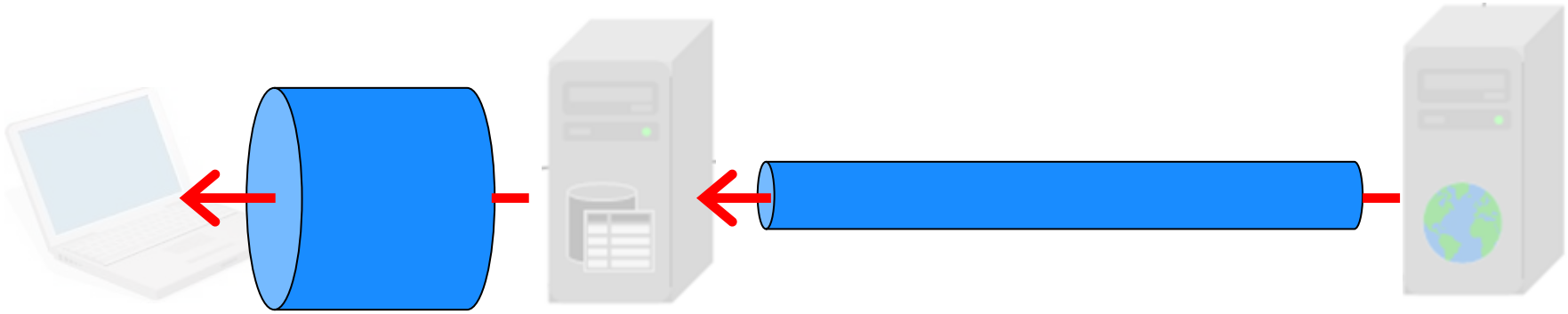


Dummynet to control network conditions
- *delays*
- *packet losses*
- *bandwidth and bottlenecks*

… for this we need instrumentation of …

- Clients: Measure performance/service
- Proxy: Implementing policies and measure performance
- Network: Capture network conditions (bottlenecks, bandwidths, delays, packet losses, etc.)

Instrumentation



Dummynet to control network conditions
- *delays*
- *packet losses*
- *bandwidth and bottlenecks*

… for this we need instrumentation of …

- Clients: Measure performance/service
- Proxy: Implementing policies and measure performance
- Network: Capture network conditions (bottlenecks, bandwidths, delays, packet losses, etc.)

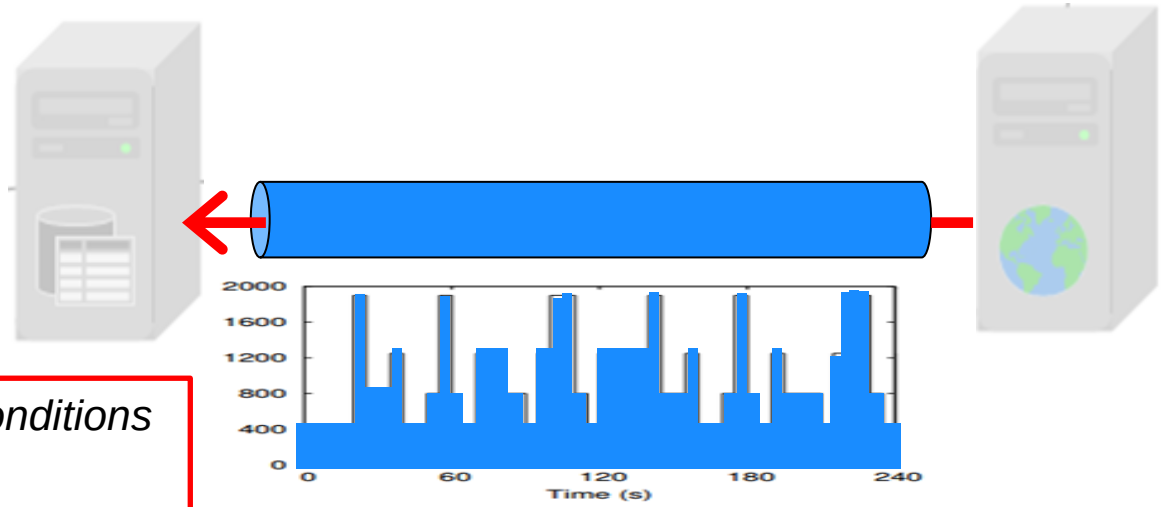
Instrumentation

Dummynet to control network conditions
- *delays*
- *packet losses*
- *bandwidth and bottlenecks*

… for this we need instrumentation of …

- Clients: Measure performance/service
- Proxy: Implementing policies and measure performance
- Network: Capture network conditions (bottlenecks, bandwidths, delays, packet losses, etc.)

Instrumentation

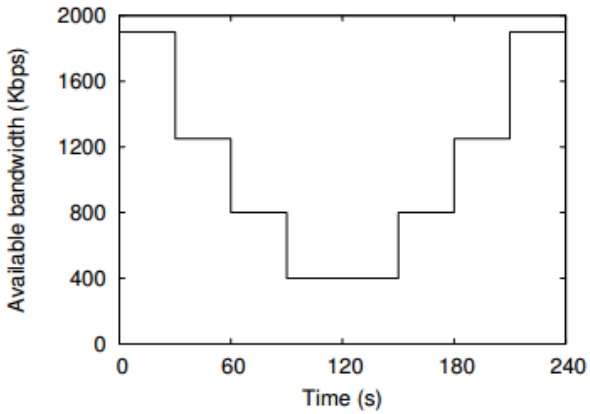


Dummynet to control network conditions
- *delays*
- *packet losses*
- *bandwidth and bottlenecks*

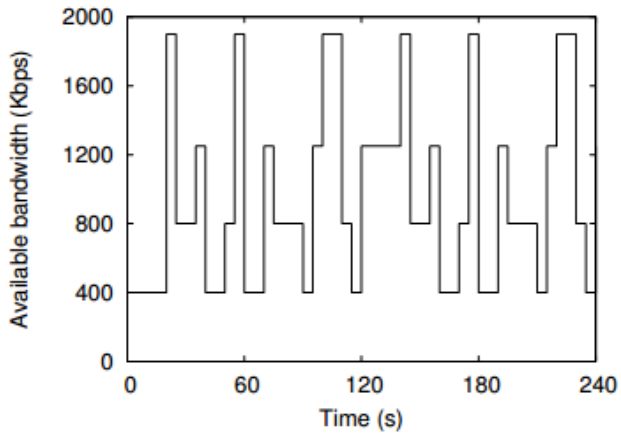
… for this we need instrumentation of …

- Clients: Measure performance/service
- Proxy: Implementing policies and measure performance
- Network: Capture network conditions (bottlenecks, bandwidths, delays, packet losses, etc.)

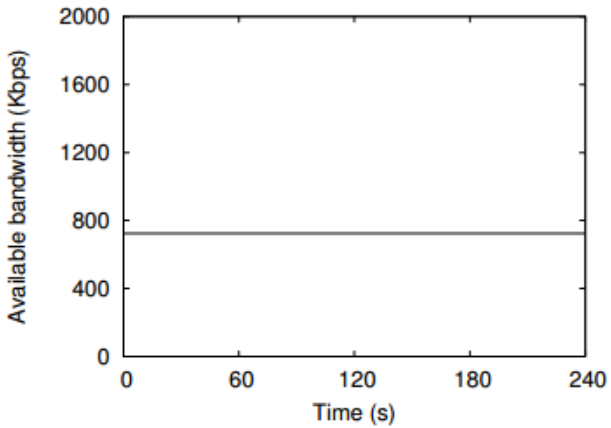
Bandwidth scenarios



Slow variations

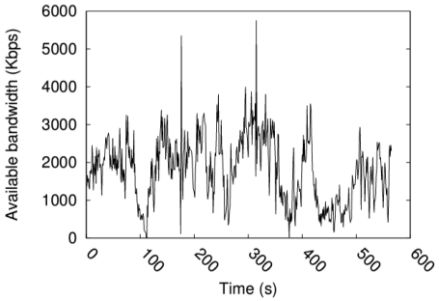


Fast variations

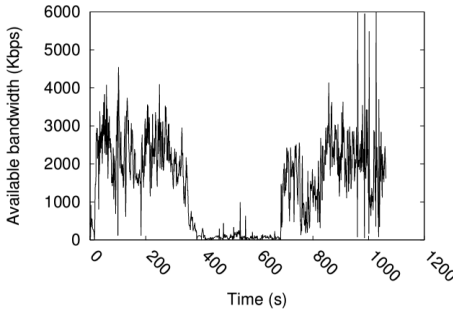


Static

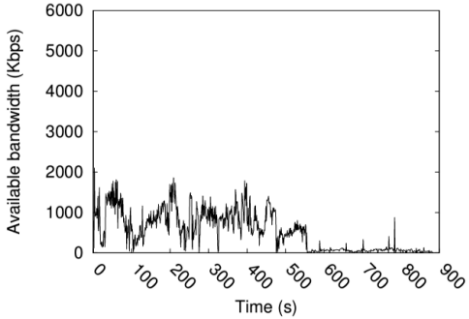
Synthetic traces



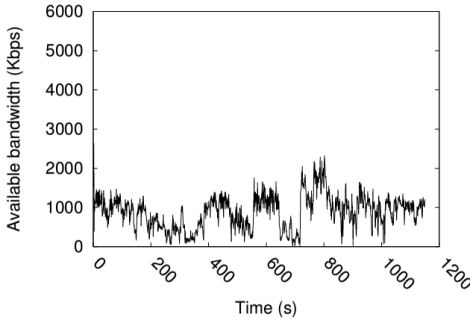
Bus



Ferry



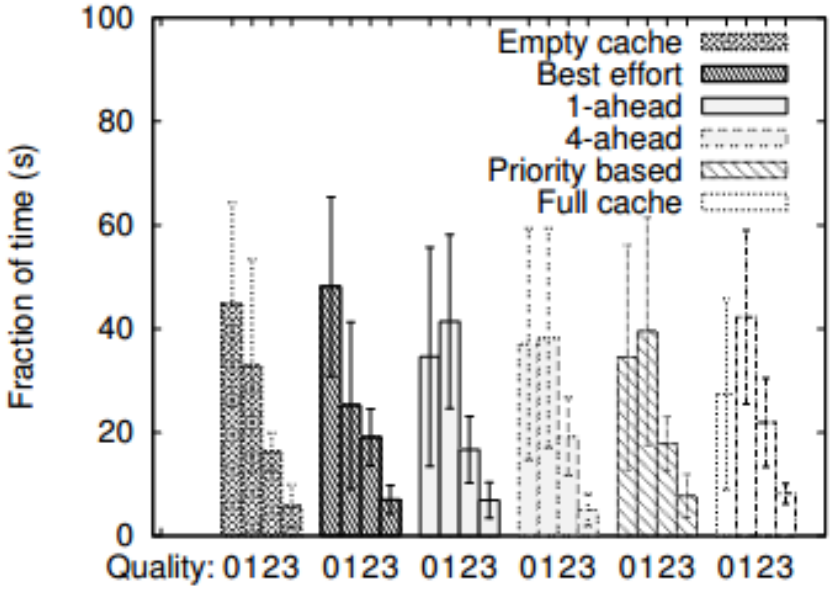
Metro



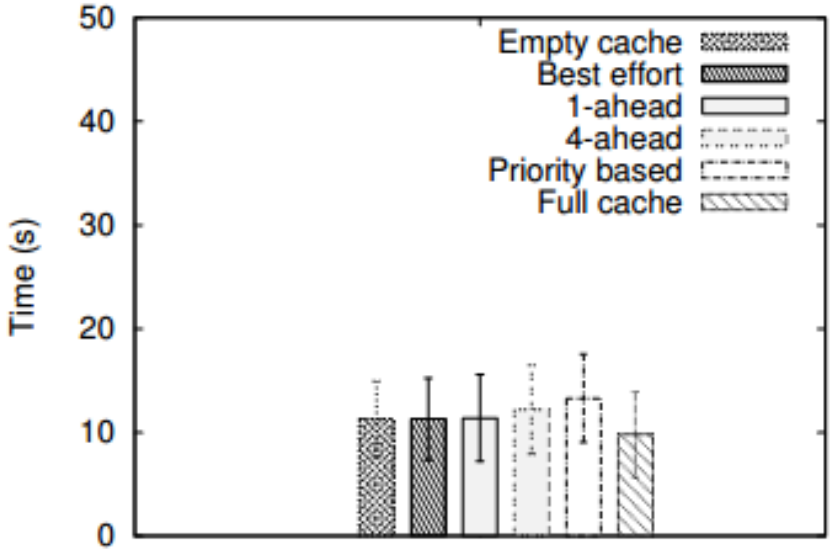
Tram

Real-world traces

Client-proxy bottleneck



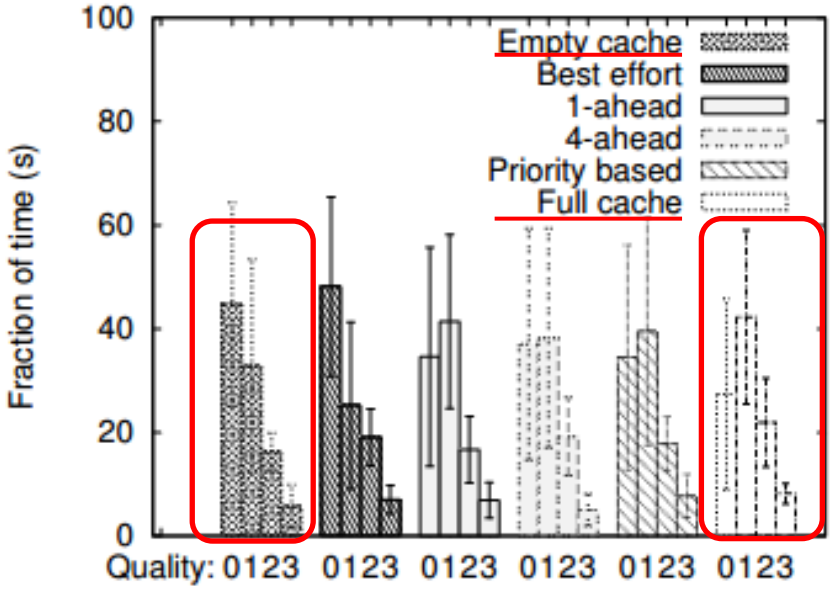
Quality level



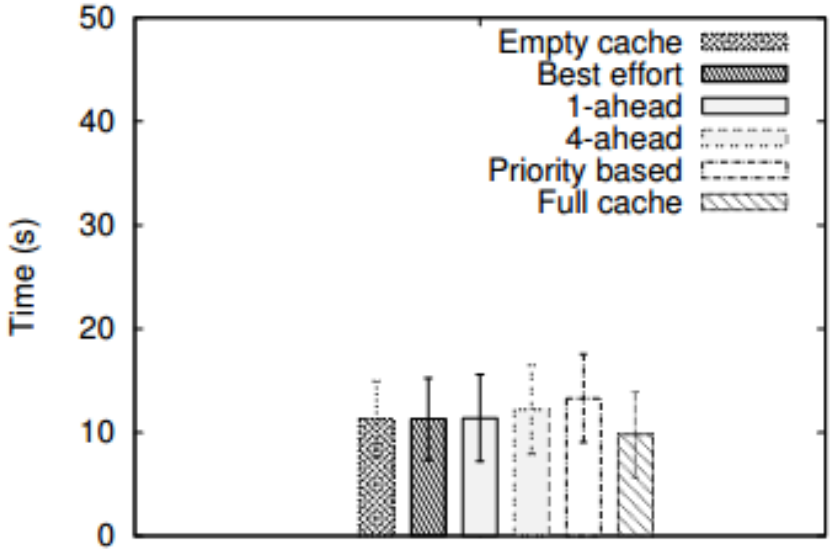
Stall times

- Proxies provide only limited performance advantages under client-proxy bottleneck
- Some performance improvements to prefetching (but penalty to excessive prefetching)

Client-proxy bottleneck



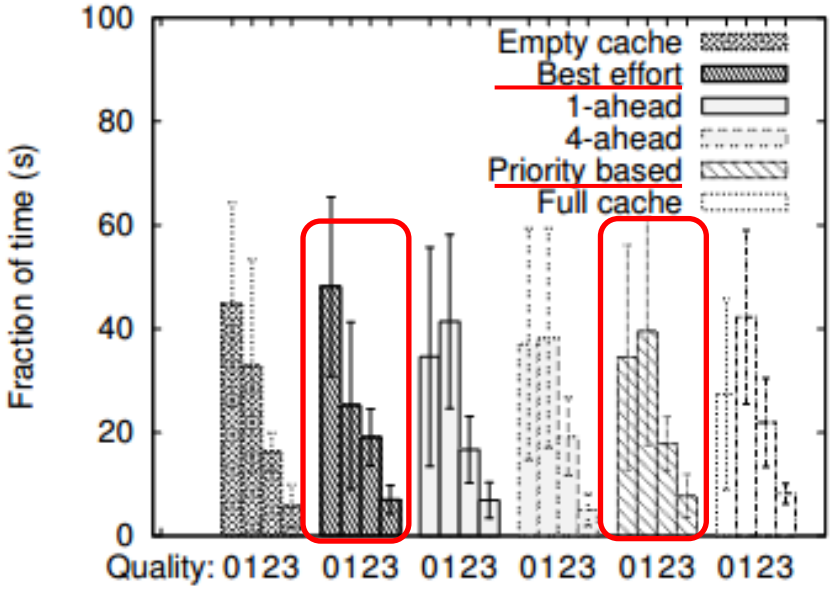
Quality level



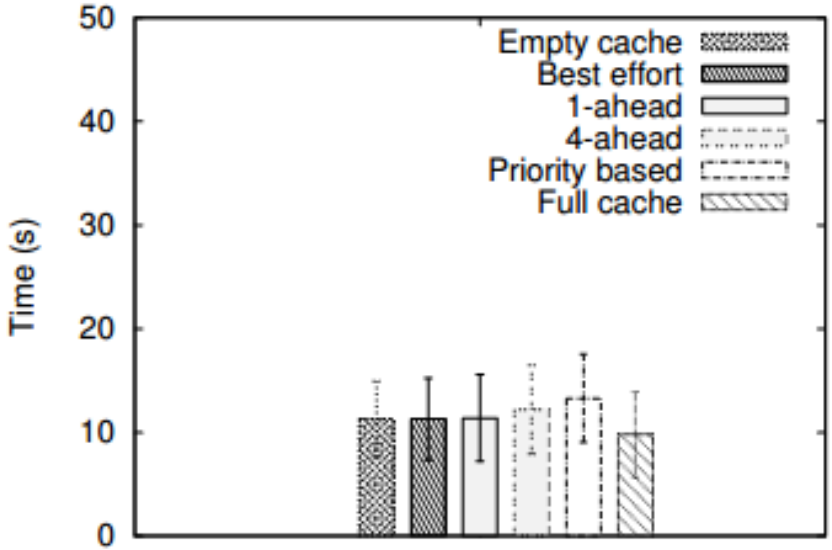
Stall times

- Proxies provide only limited performance advantages under client-proxy bottleneck
- Some performance improvements to prefetching

Client-proxy bottleneck



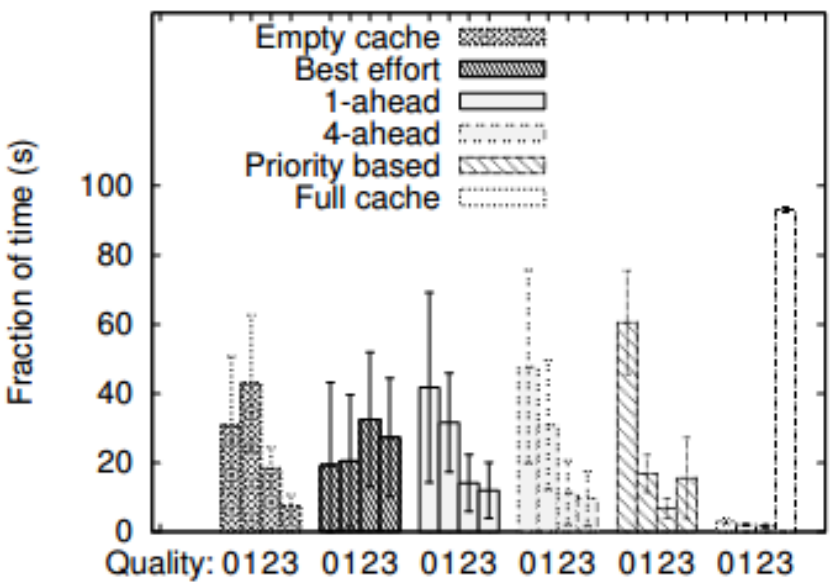
Quality level



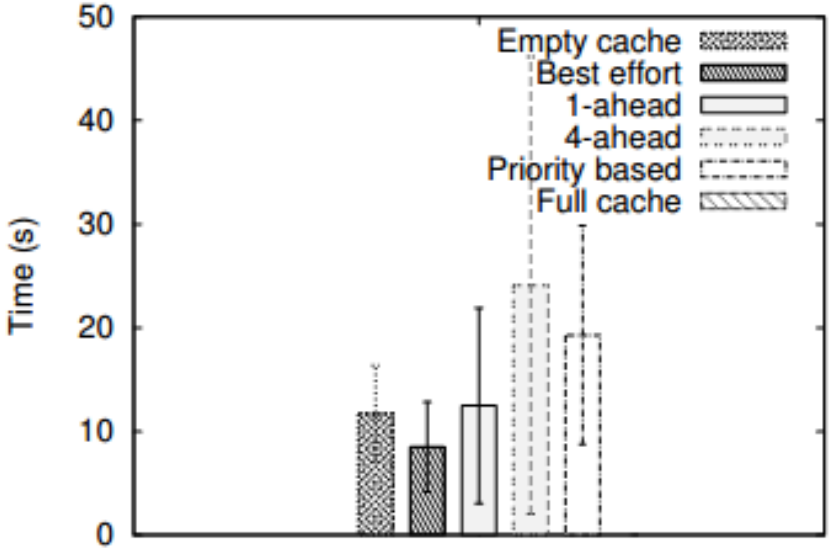
Stall times

- Proxies provide only limited performance advantages under client-proxy bottleneck
- Some performance improvements to prefetching

Proxy-server bottleneck



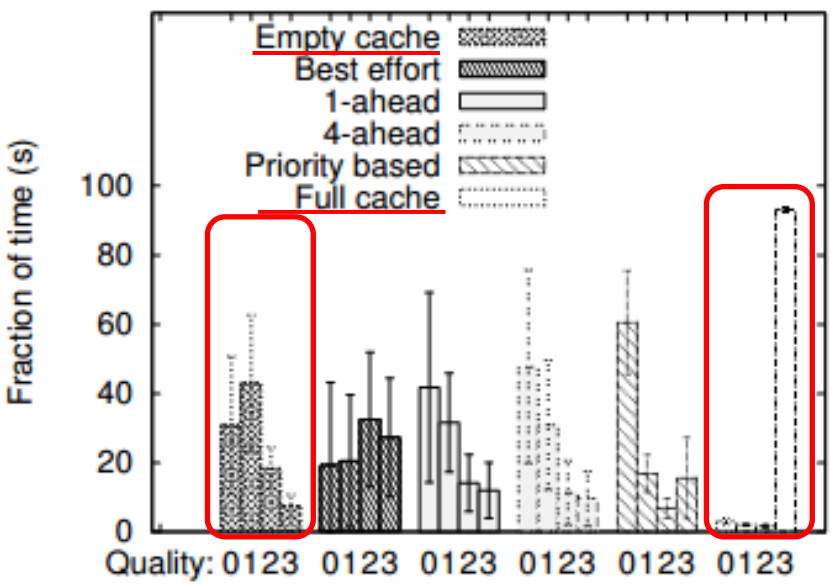
Quality level



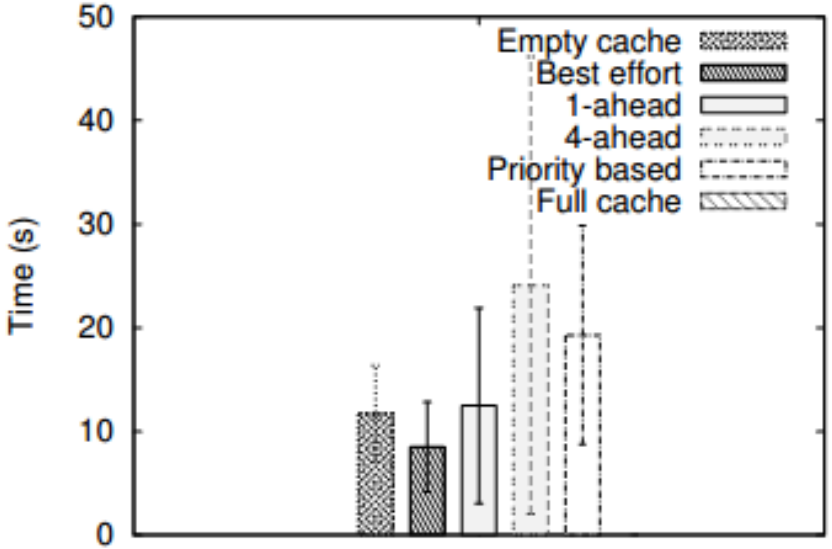
Stall times

- Large performance potential for proxy caching
- Significant performance improvement with the best effort policy
- Naive prefetching results in penalty
- Need for more intelligent prefetching policies (cooperative)

Proxy-server bottleneck



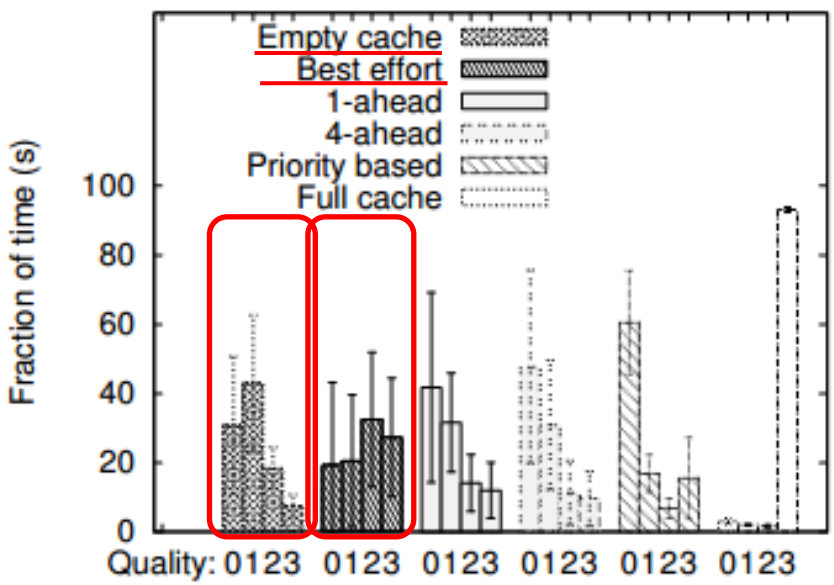
Quality level



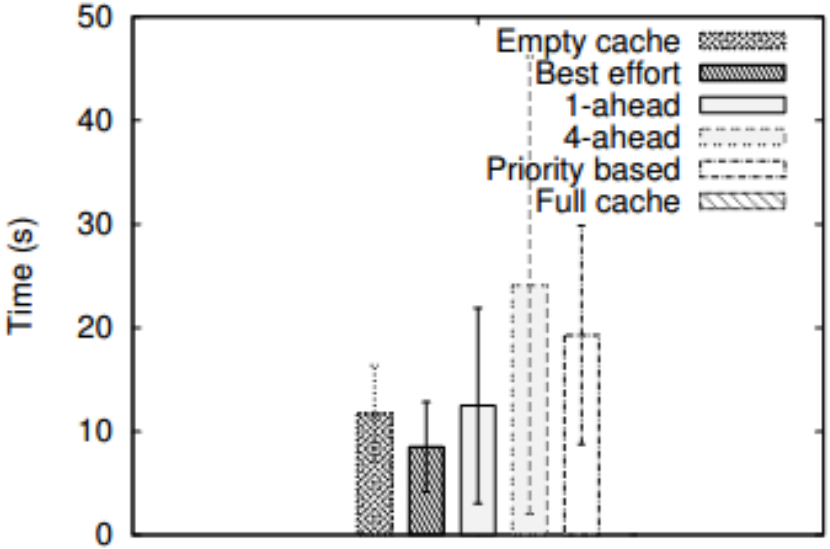
Stall times

- Large performance potential for proxy caching
- Significant performance improvement with the best effort policy
- Naive prefetching results in penalty
- Need for more intelligent prefetching policies (cooperative)

Proxy-server bottleneck



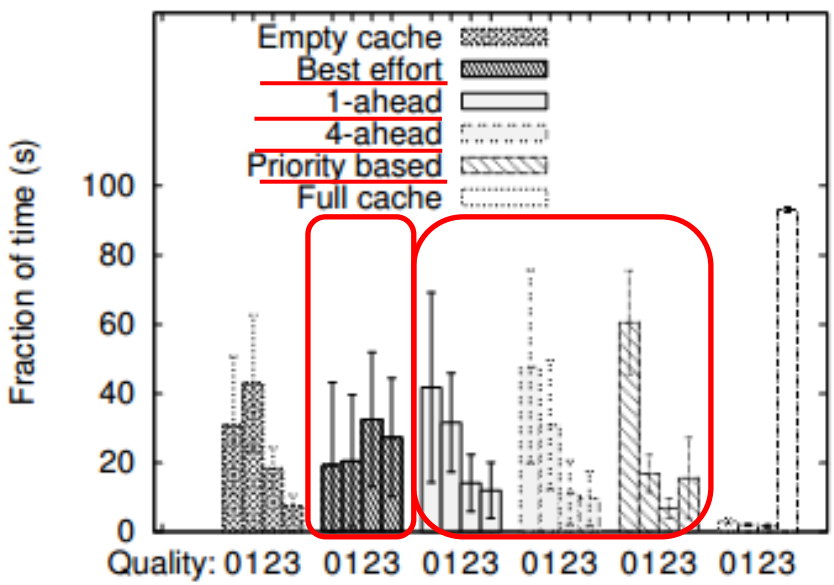
Quality level



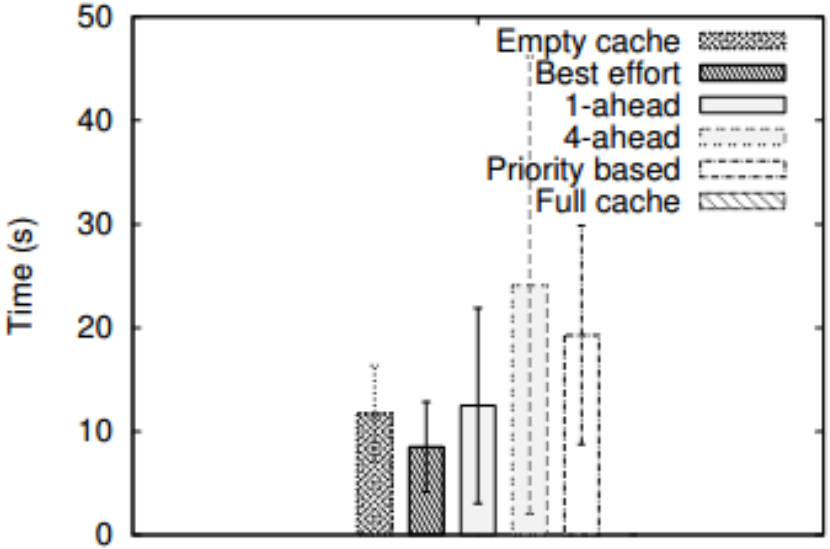
Stall times

- Large performance potential for proxy caching
- Significant performance improvement with the best effort policy
- Naive prefetching results in penalty
- Need for more intelligent prefetching policies (cooperative)

Proxy-server bottleneck



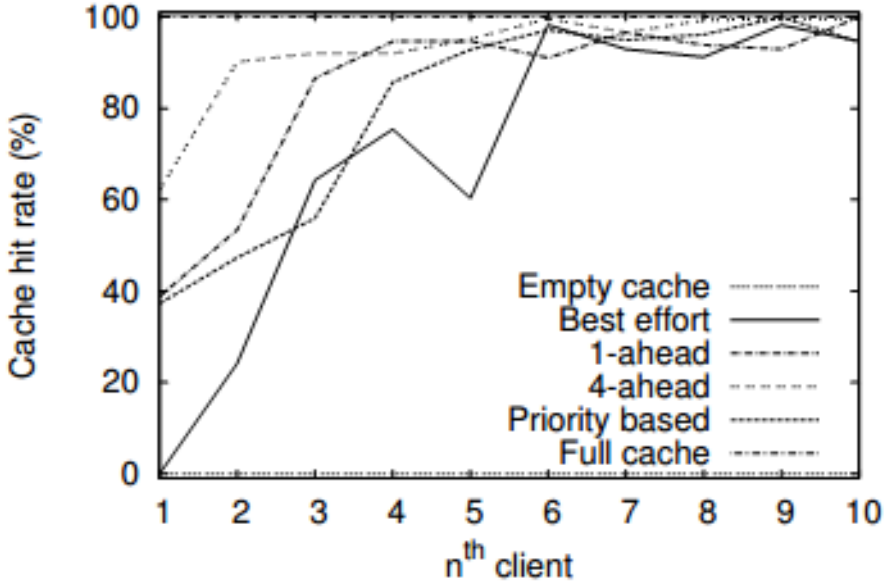
Quality level



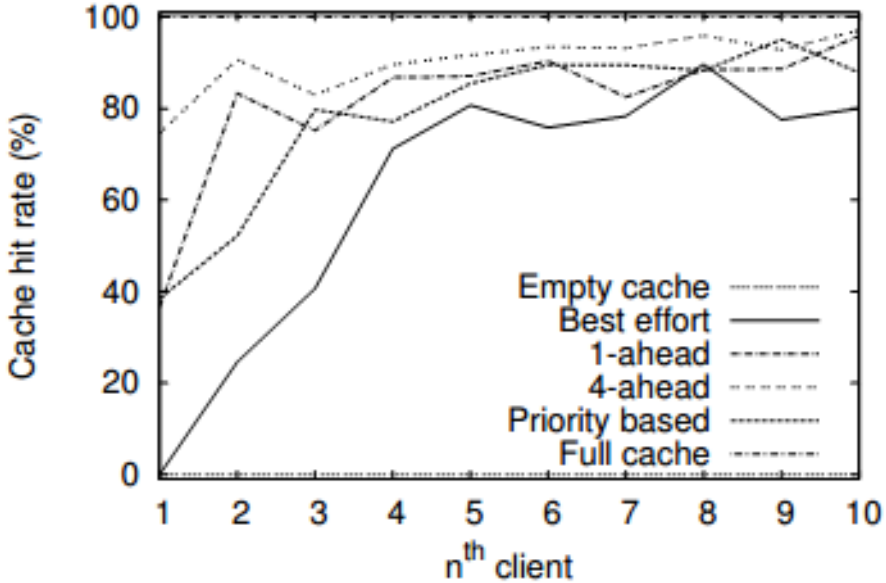
Stall times

- Large performance potential for proxy caching
- Significant performance improvement with the best effort policy
- Naive prefetching results in penalty
- Need for more intelligent prefetching policies (cooperative)

Transient hit rate analysis



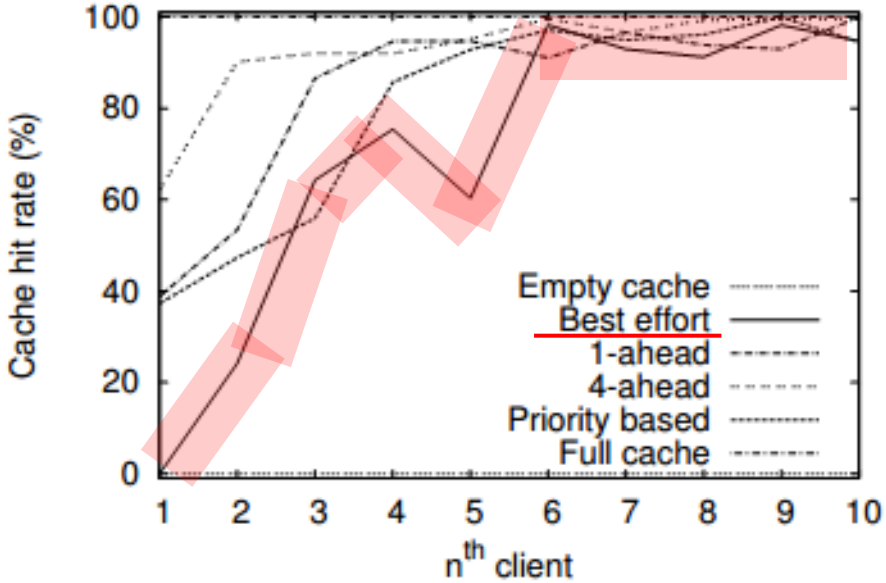
Client-proxy bottleneck



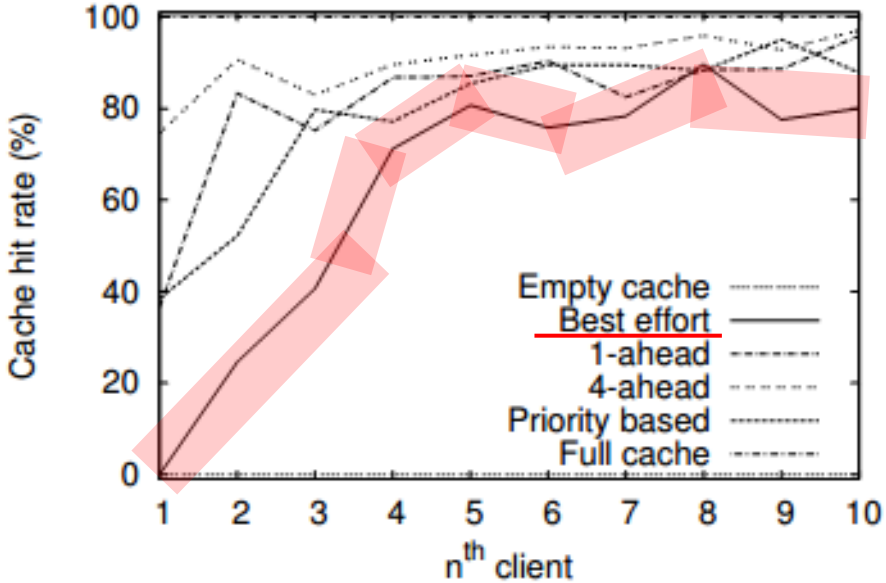
Proxy-server bottleneck

- Best effort has the smallest hit rate
- Prefetching bandwidth benefit future clients
- Hits and high client-proxy bandwidth may cause costly penalties due to limited bandwidth at misses (more variability and lower hit rates)

Transient hit rate analysis



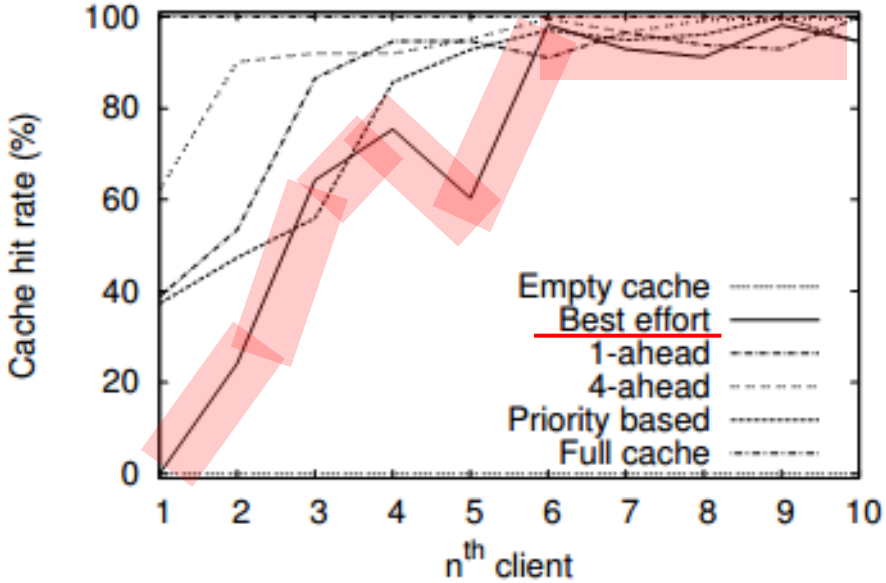
Client-proxy bottleneck



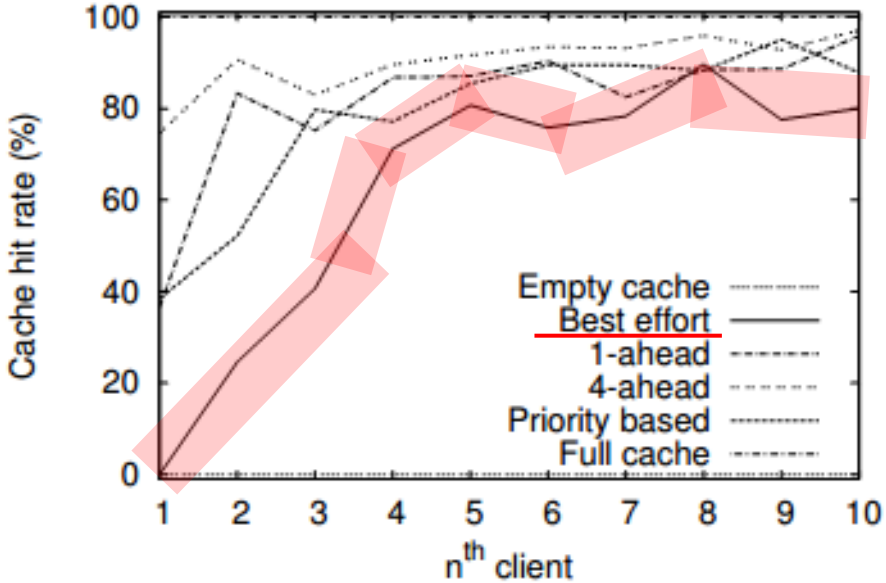
Proxy-server bottleneck

- Best effort has the smallest hit rate
- Prefetching bandwidth benefit future clients
- Hits and high client-proxy bandwidth may cause costly penalties due to limited bandwidth at misses (more variability and lower hit rates)

Transient hit rate analysis



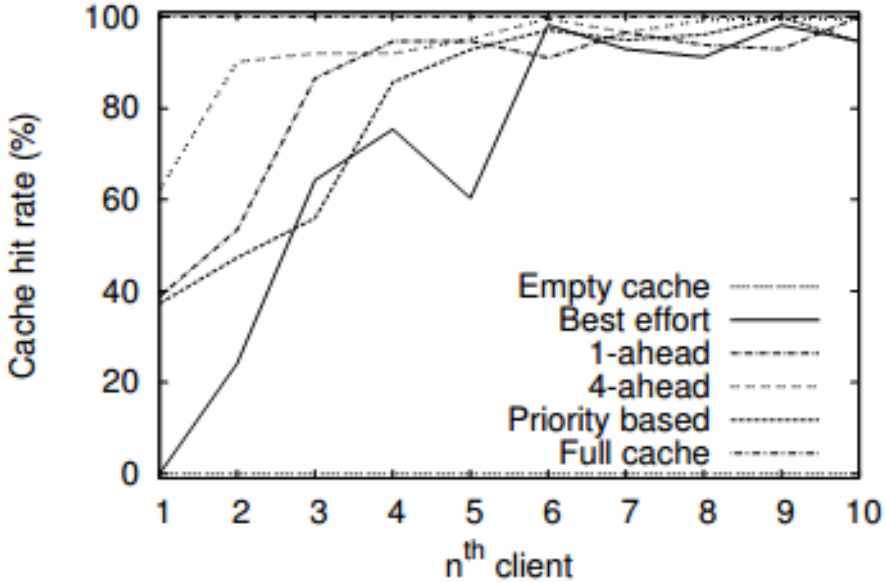
Client-proxy bottleneck



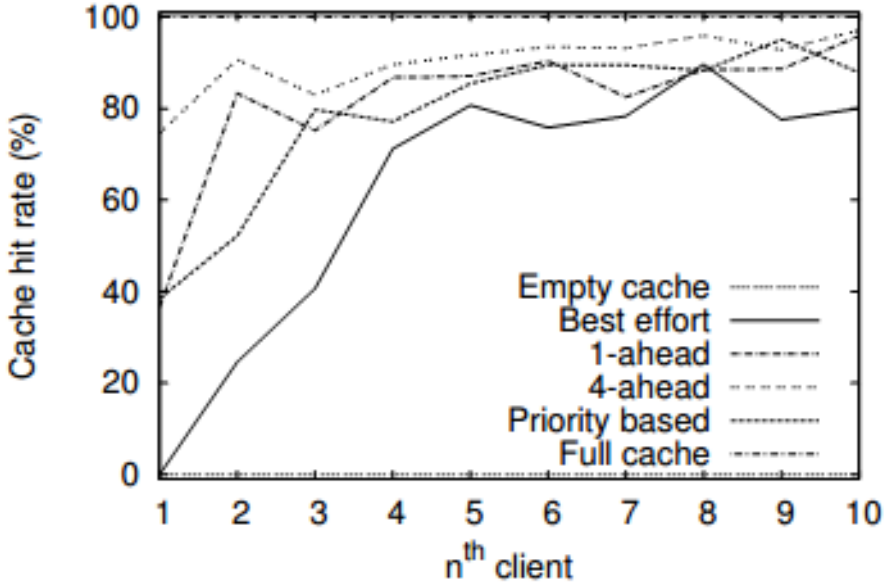
Proxy-server bottleneck

- Best effort has the smallest hit rate
- Prefetching bandwidth benefit future clients
- Low proxy-server bandwidth may cause costly penalties due to slow downloads at misses (more variability and lower hit rates)

Transient hit rate analysis



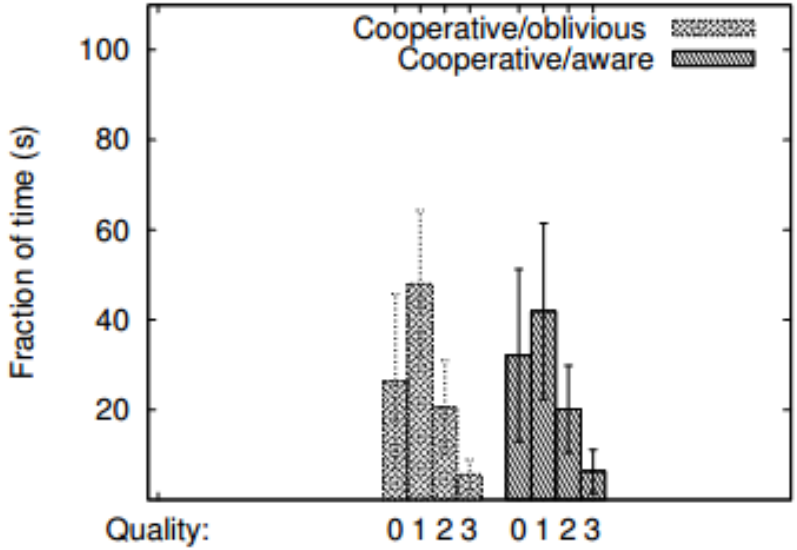
Client-proxy bottleneck



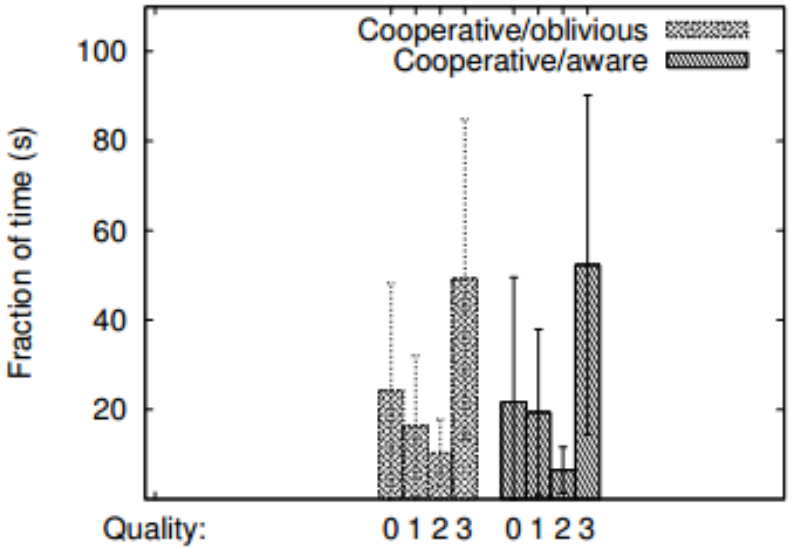
Proxy-server bottleneck

- Best effort has the smallest hit rate
- Prefetching bandwidth benefit future clients
- Low proxy-server bandwidth may cause costly penalties due to slow downloads at misses (more variability and lower hit rates)

Client-proxy cooperation



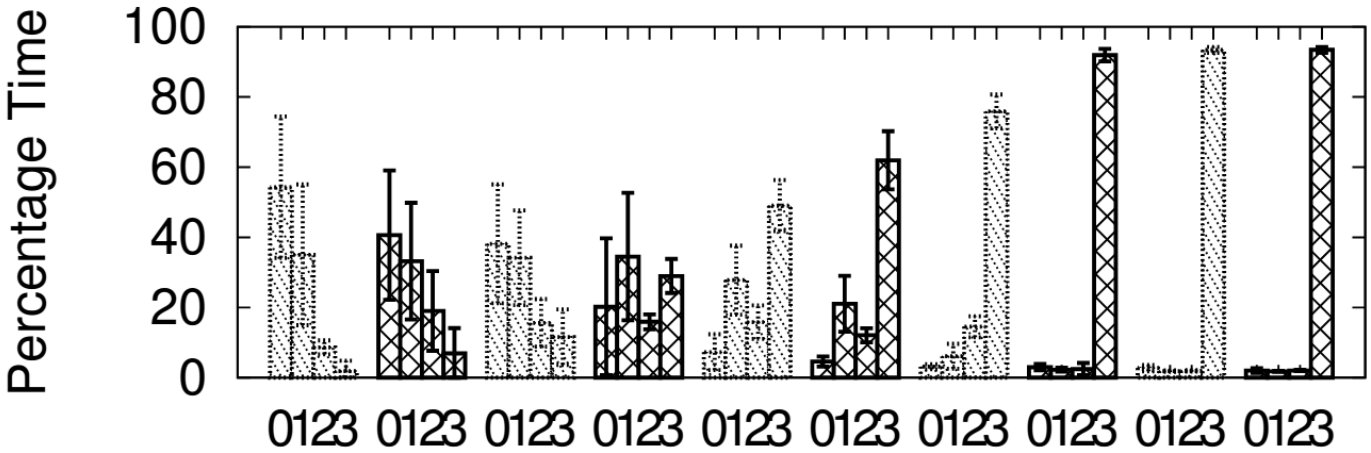
Client-proxy bottleneck



Proxy-server bottleneck

- For client-proxy bottleneck, both policies slightly outperform all baseline and quality-aware prefetching policies
- For proxy-server bottleneck, both policies vastly outperform all baseline and quality-aware prefetching policies

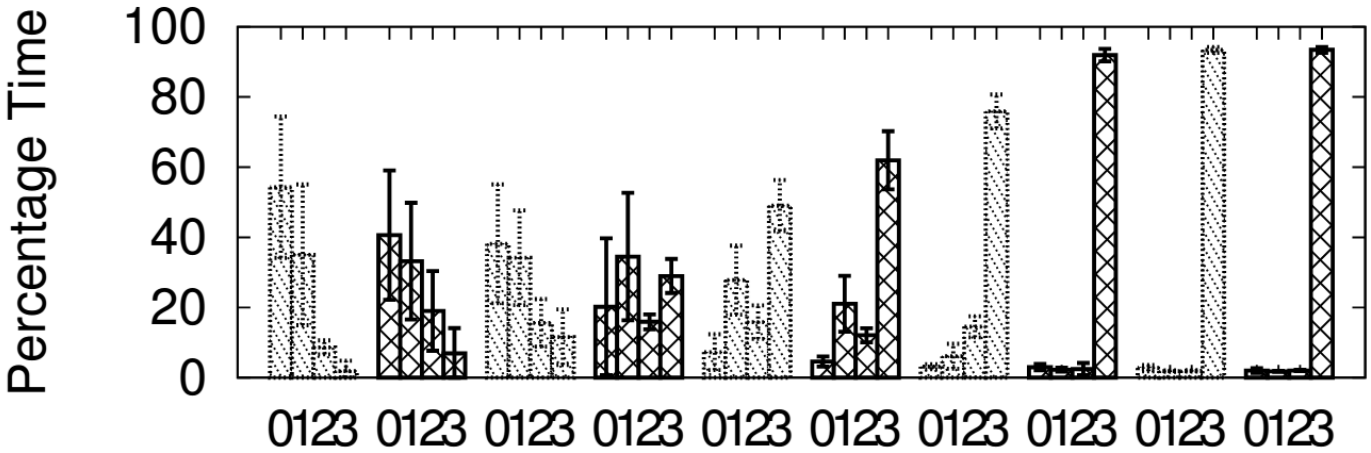
Example trace: Client-proxy cooperation



Quality level of each client (proxy-server bottleneck)

- Cooperative policy quickly adapt cache content such as to best serve the clients, without penalizing early clients

Example trace: Client-proxy cooperation



Quality level of each client (proxy-server bottleneck)

- Cooperative policy quickly adapt cache content such as to best serve the clients, without penalizing early clients

Conclusions

- Performance impact of HAS-aware proxy policies
  - Baseline policies
  - Quality and content-aware prefetching
  - Client-proxy cooperation
- Bottleneck location and network conditions play central roles in which policy choices are most advantageous
  - Large benefits to cooperative policies when proxy-server bottleneck
- Careful proxy design and policy selection very important
- Future work include adaptive policies
  - Bottleneck and their conditions may change

Helping Hand or Hidden Hurdle: Proxy-assisted HTTP-based Adaptive Streaming Performance

Vengatanathan Krishnamoorthi (LiU)
Niklas Carlsson (LiU)
Derek Eager (U of S)
Anirban Mahanti (NICTA)
Nahid Shahmehri (LiU)

Linköping University
expanding reality
www.liu.se