Application Layer
Part 4

Mark Allman
Case / ICSI

EECS 325/425
Fall 2018

"I don't really want to stop the show,
But I thought that you might like to know,
That the singer's going to sing a song,
And he wants you all to sing along."

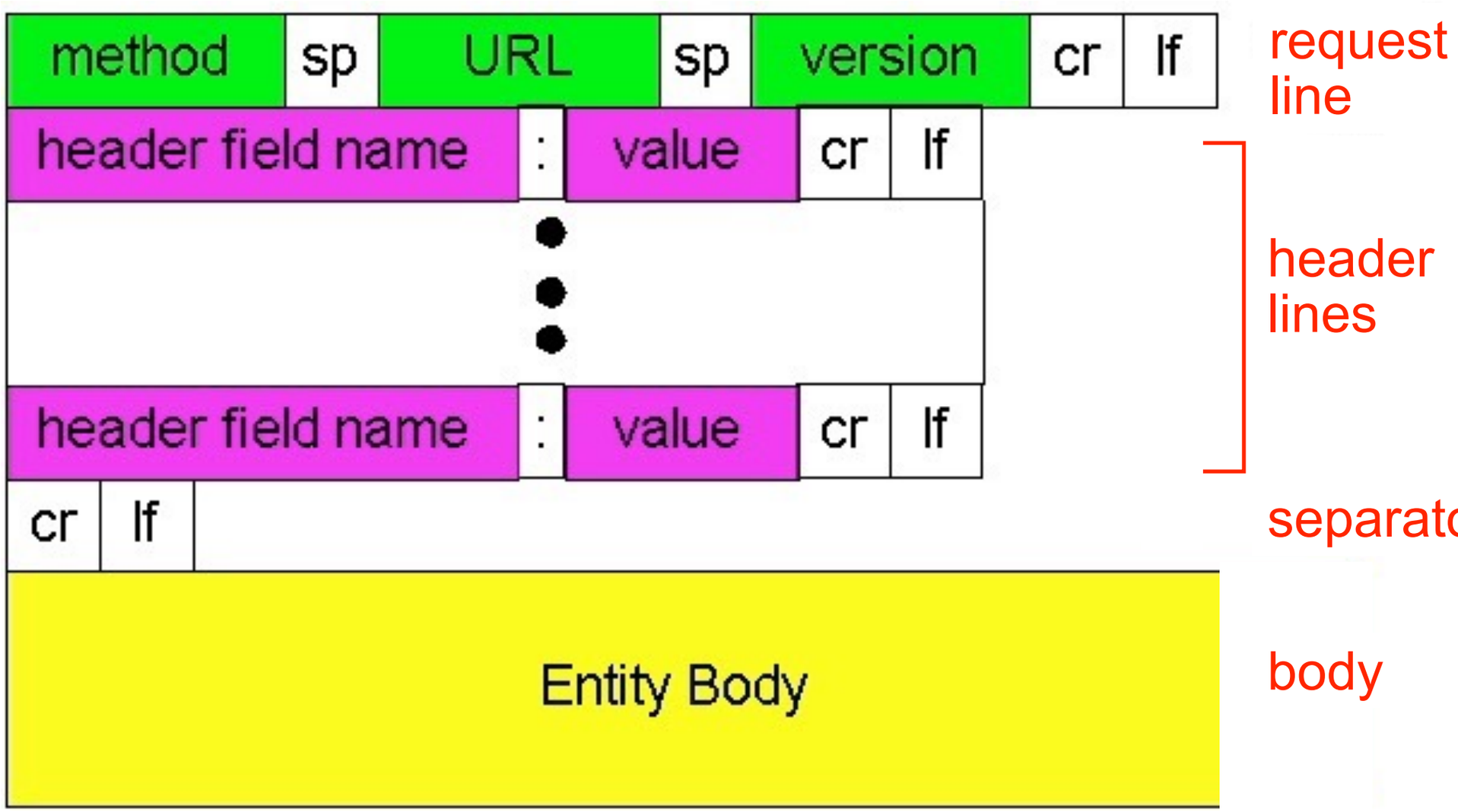
Many of these slides are more-or-less directly from the slide set developed by Jim Kurose and Keith Ross for their book “Computer Networking: A Top Down Approach, 5th edition”.

The slides have been lightly adapted for Mark Allman’s EECS 325/425 Computer Networks class at Case Western Reserve University.

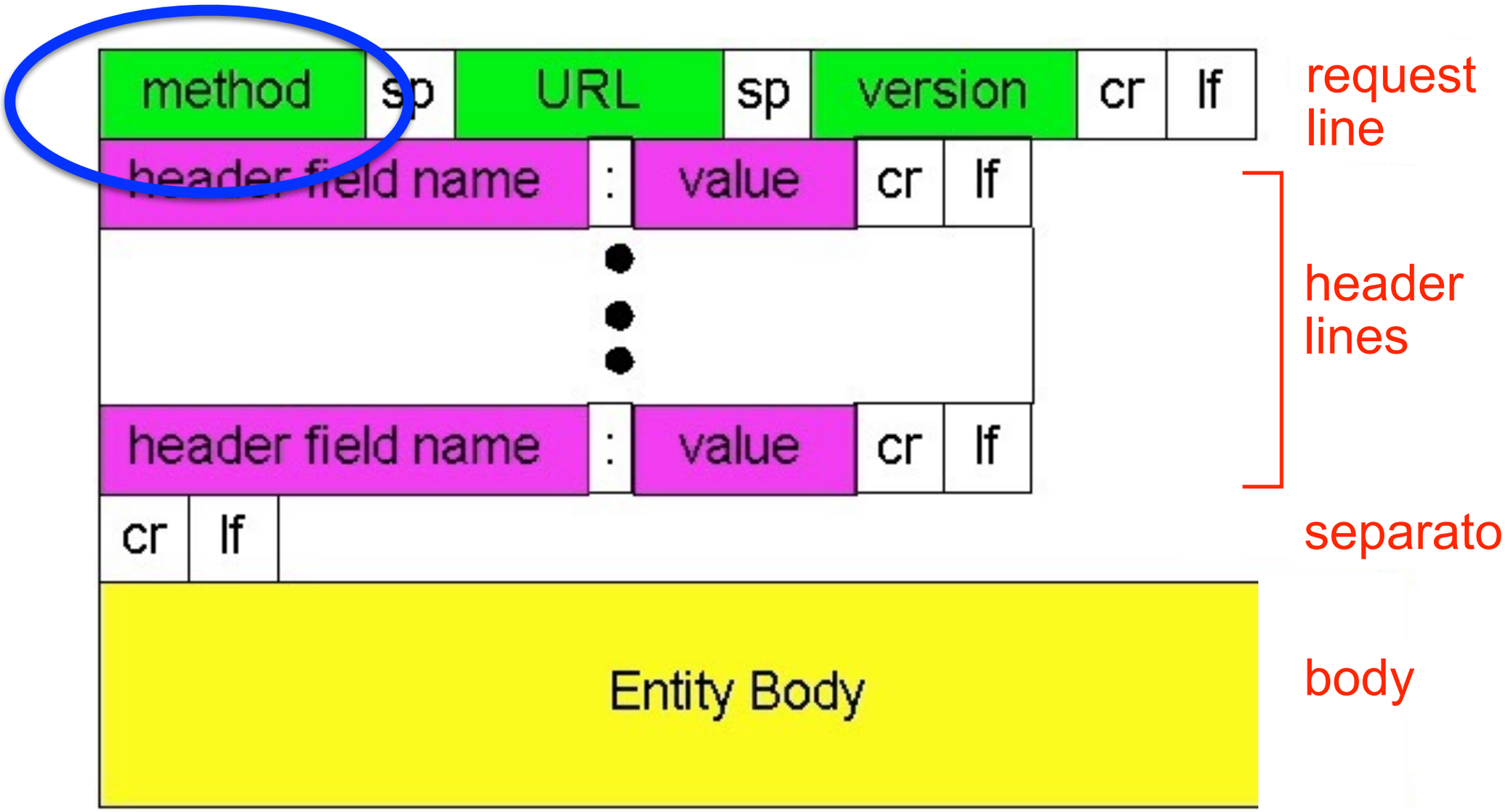
Beyond Basic HTTP

- HTTP is a *big* protocol / ecosystem with many extensions and variants
- Let's consider a few examples (not exhaustive in any way)

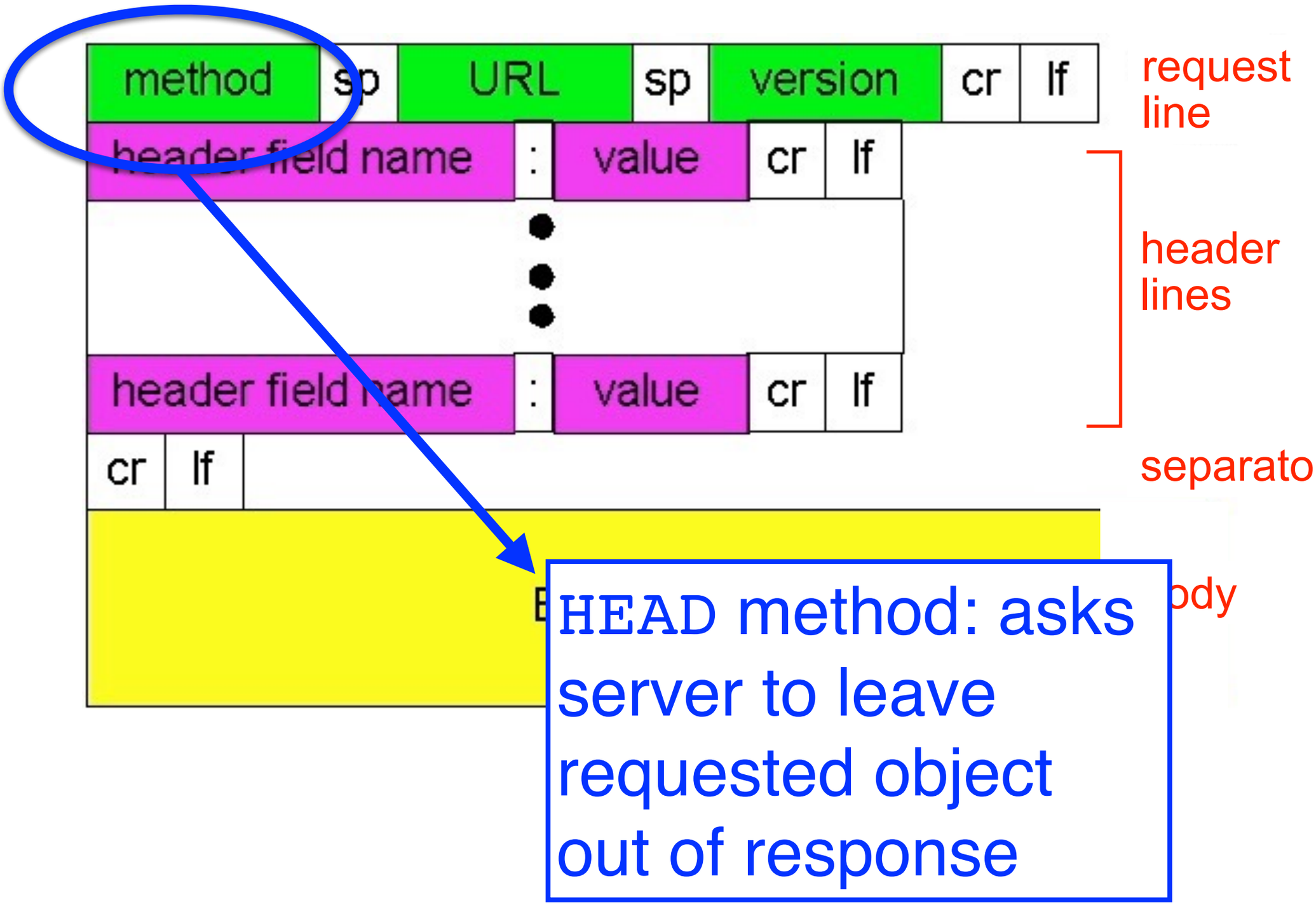
HTTP request message: general format



HTTP request message: general format



HTTP request message: general format



HTTP response message

```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02 GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-1\r\n
\r\n
data data data data data ...
```

HTTP response message

```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02 GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-1\r\n
\r\n
data data data data data data ...
```

Conditional GET

- Goal: don't send object if cache has up-to-date cached version
- cache: specify date of cached copy in HTTP request

  If-modified-since: <date>

- server: response contains no object if cached copy is up-to-date:

  HTTP/1.0 304 Not Modified

Conditional GET

- Goal: don't send object if cache has up-to-date cached version
- cache: specify date of cached copy in HTTP request

  If-modified-since: <date>

- server: response contains no object if cached copy is up-to-date:

  HTTP/1.0 304 Not Modified

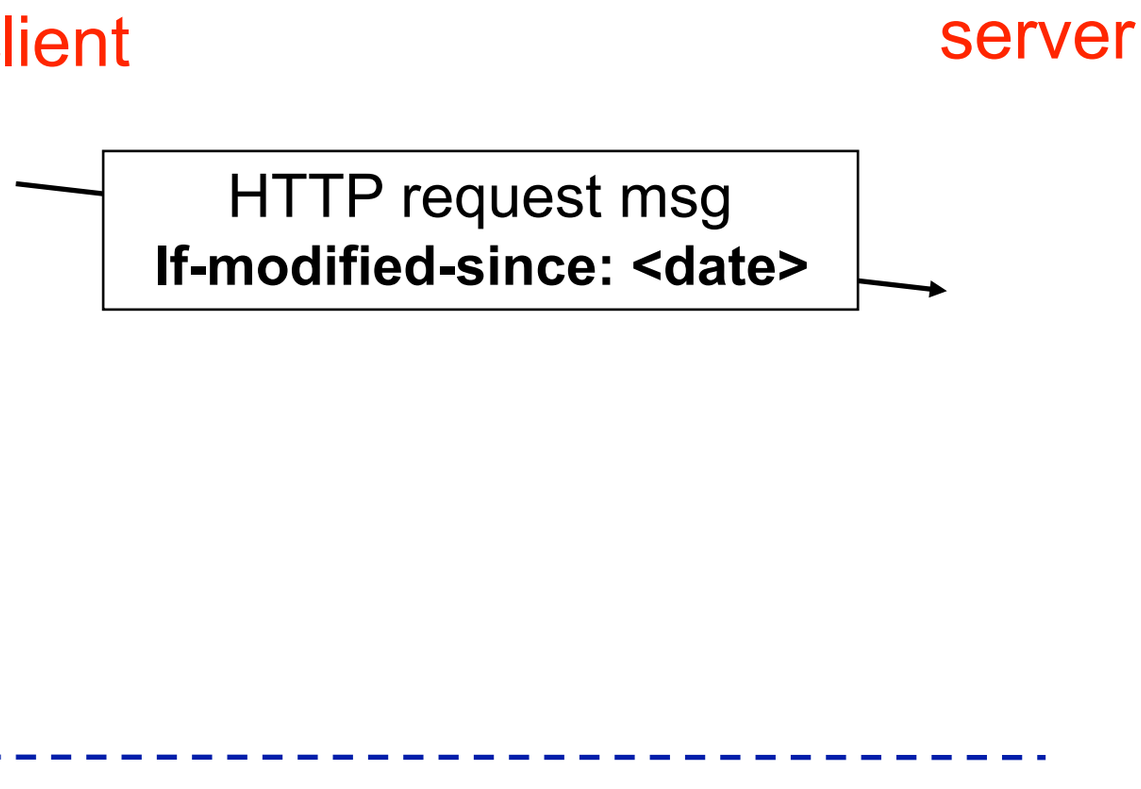
Conditional GET

- Goal: don't send object if cache has up-to-date cached version
- cache: specify date of cached copy in HTTP request

  If-modified-since: <date>

- server: response contains no object if cached copy is up-to-date:

  HTTP/1.0 304 Not Modified



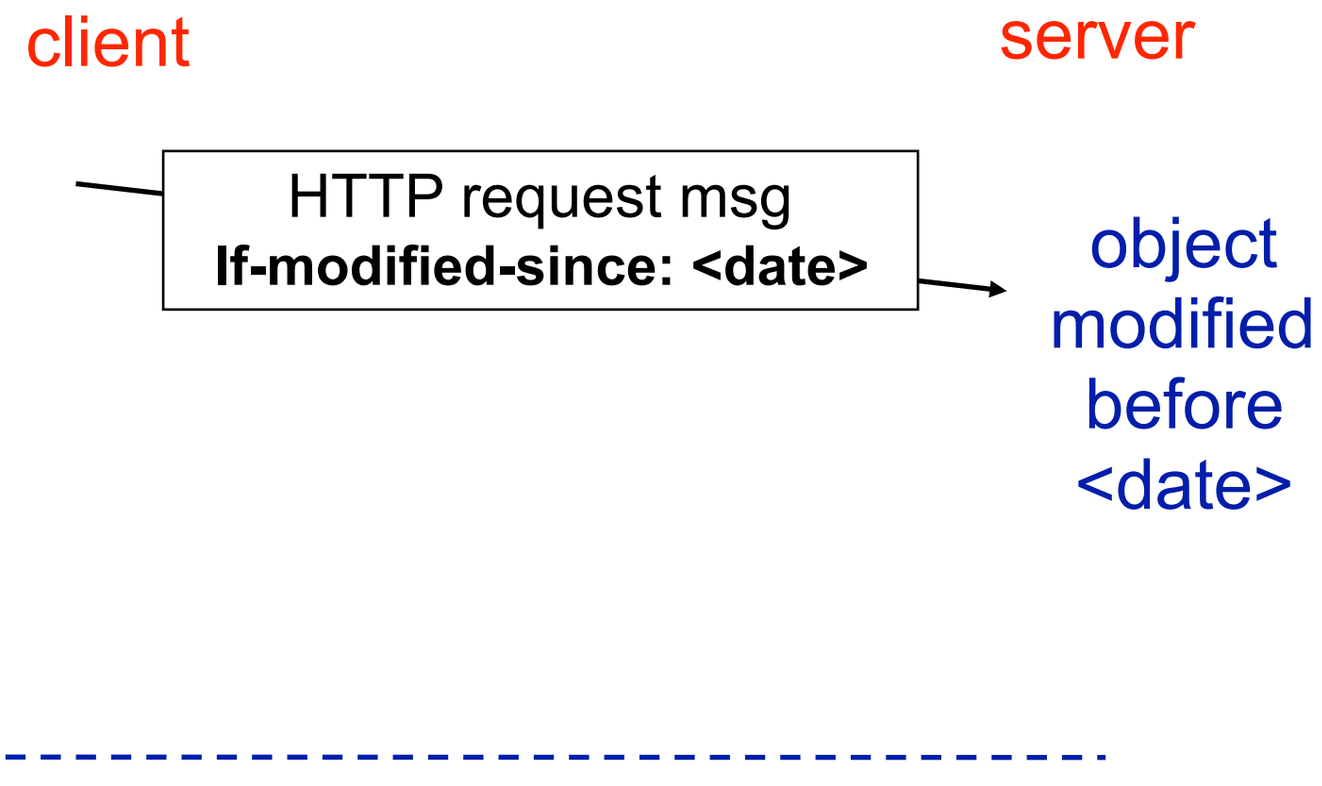
Conditional GET

- Goal: don't send object if cache has up-to-date cached version
- cache: specify date of cached copy in HTTP request

  If-modified-since: <date>

- server: response contains no object if cached copy is up-to-date:

  HTTP/1.0 304 Not Modified



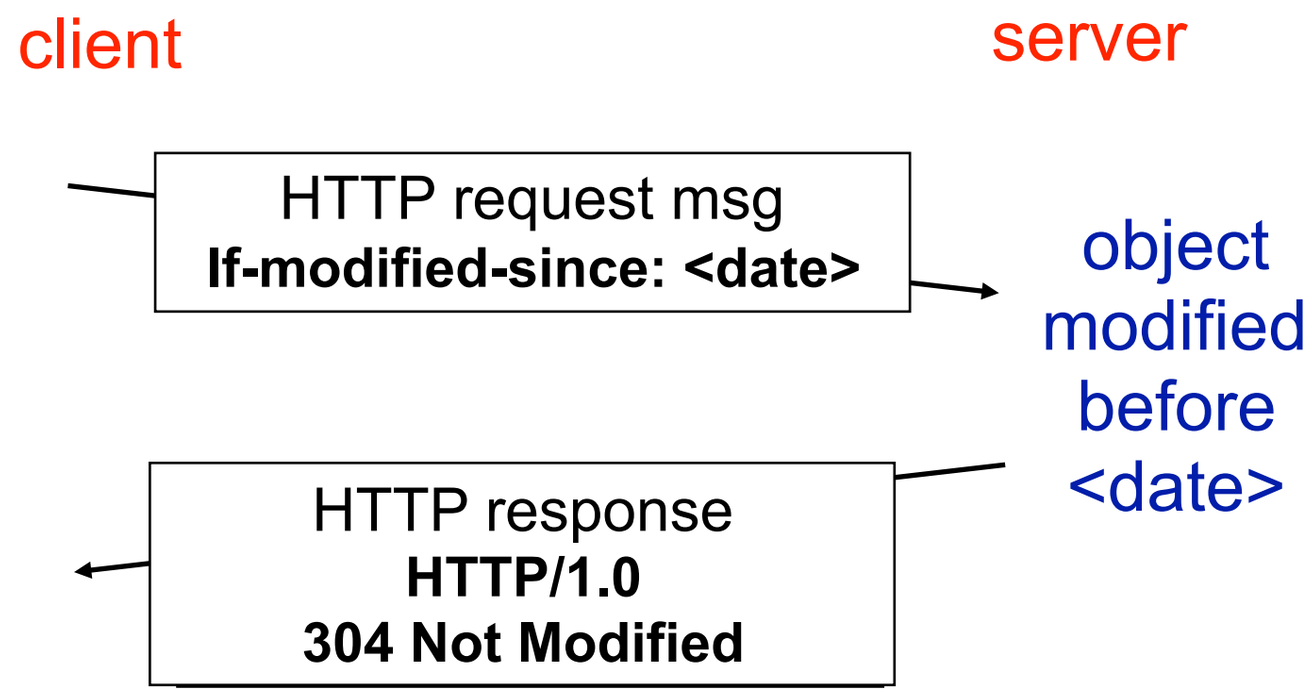
Conditional GET

- Goal: don't send object if cache has up-to-date cached version
- cache: specify date of cached copy in HTTP request

  If-modified-since: <date>

- server: response contains no object if cached copy is up-to-date:

  HTTP/1.0 304 Not Modified



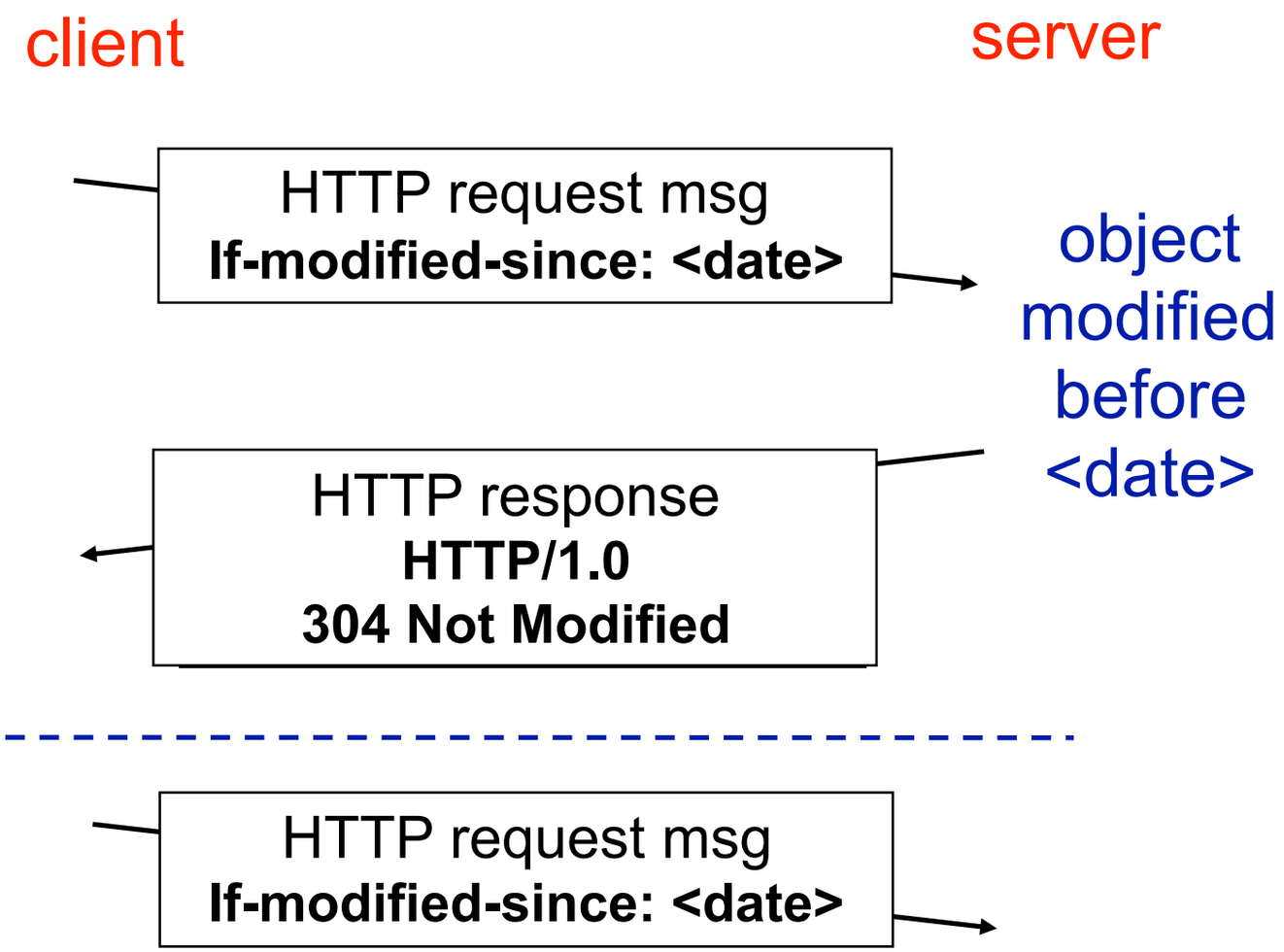
Conditional GET

- Goal: don't send object if cache has up-to-date cached version
- cache: specify date of cached copy in HTTP request

  If-modified-since: <date>

- server: response contains no object if cached copy is up-to-date:

  HTTP/1.0 304 Not Modified



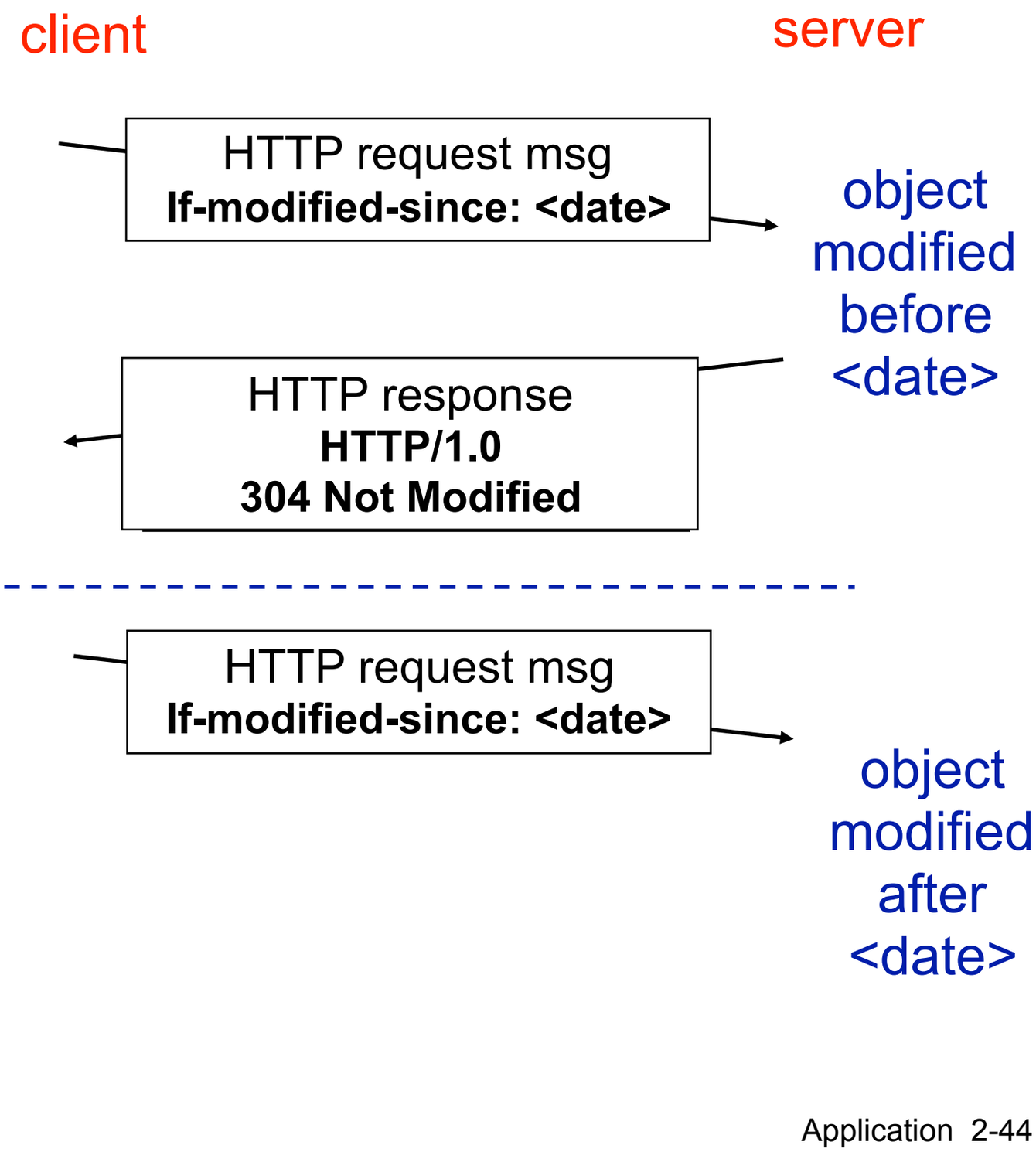
Conditional GET

- Goal: don't send object if cache has up-to-date cached version
- cache: specify date of cached copy in HTTP request

  If-modified-since: <date>

- server: response contains no object if cached copy is up-to-date:

  HTTP/1.0 304 Not Modified



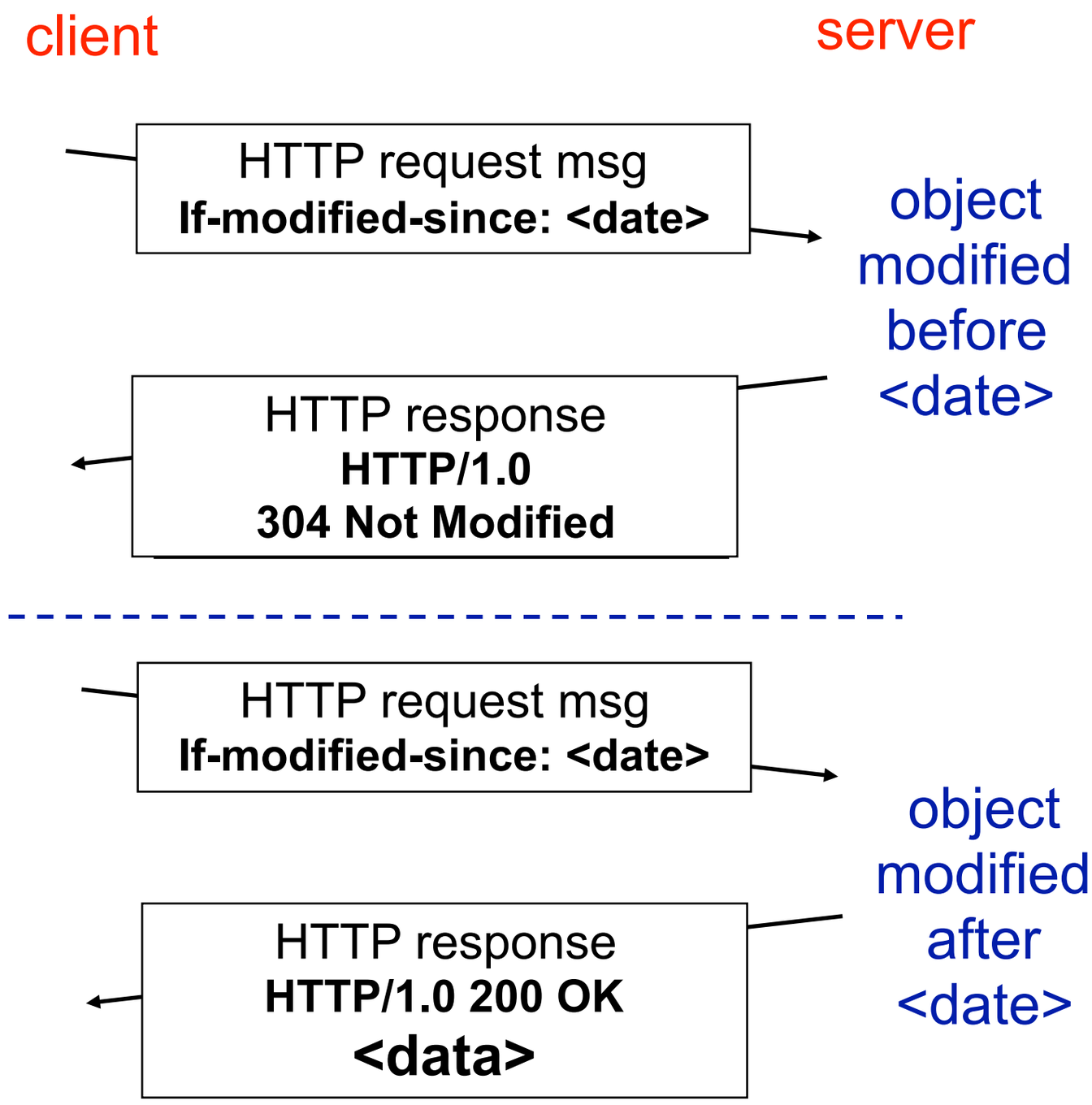
Conditional GET

- Goal: don't send object if cache has up-to-date cached version
- cache: specify date of cached copy in HTTP request

  If-modified-since: <date>

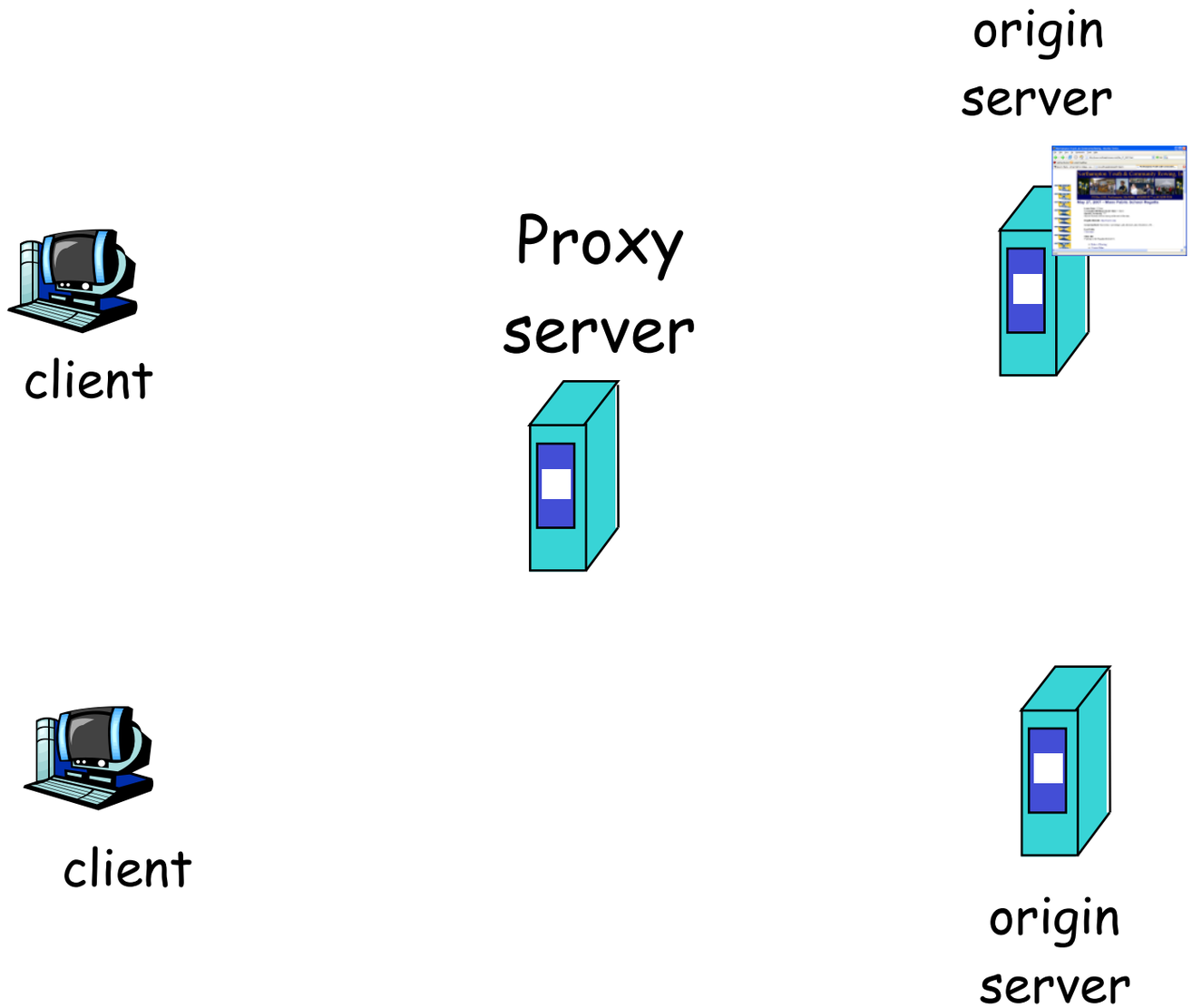
- server: response contains no object if cached copy is up-to-date:

  HTTP/1.0 304 Not Modified



Web caches (proxy server)

Goal: satisfy client request without involving origin server



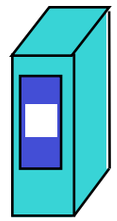
Web caches (proxy server)

Goal: satisfy client request without involving origin server

- user sets browser: Web accesses via cache

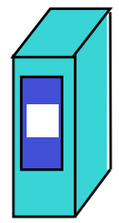
client

Proxy server



origin server

client



origin server

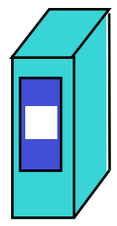
Web caches (proxy server)

Goal: satisfy client request without involving origin server

- user sets browser: Web accesses via cache
- browser sends all HTTP requests to cache

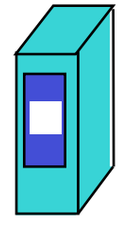
client

Proxy server



origin server

client



origin server

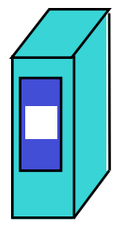
Web caches (proxy server)

Goal: satisfy client request without involving origin server

- user sets browser: Web accesses via cache
- browser sends all HTTP requests to cache
  - object in cache: cache returns object
  - else cache requests object from origin server, then returns object to client

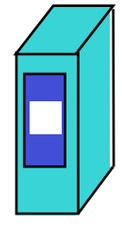
client

Proxy server



origin server

client

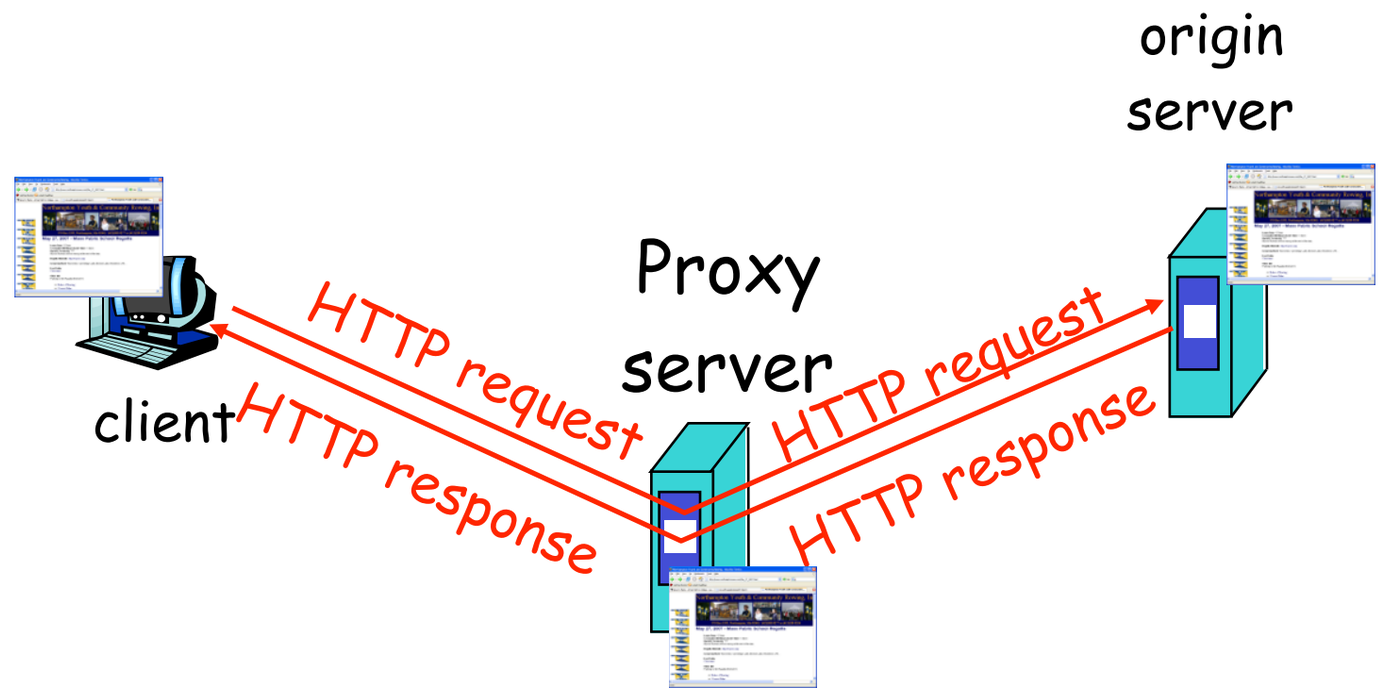


origin server

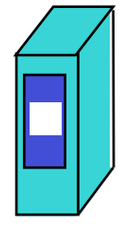
Web caches (proxy server)

Goal: satisfy client request without involving origin server

- user sets browser: Web accesses via cache
- browser sends all HTTP requests to cache
  - object in cache: cache returns object
  - else cache requests object from origin server, then returns object to client



client

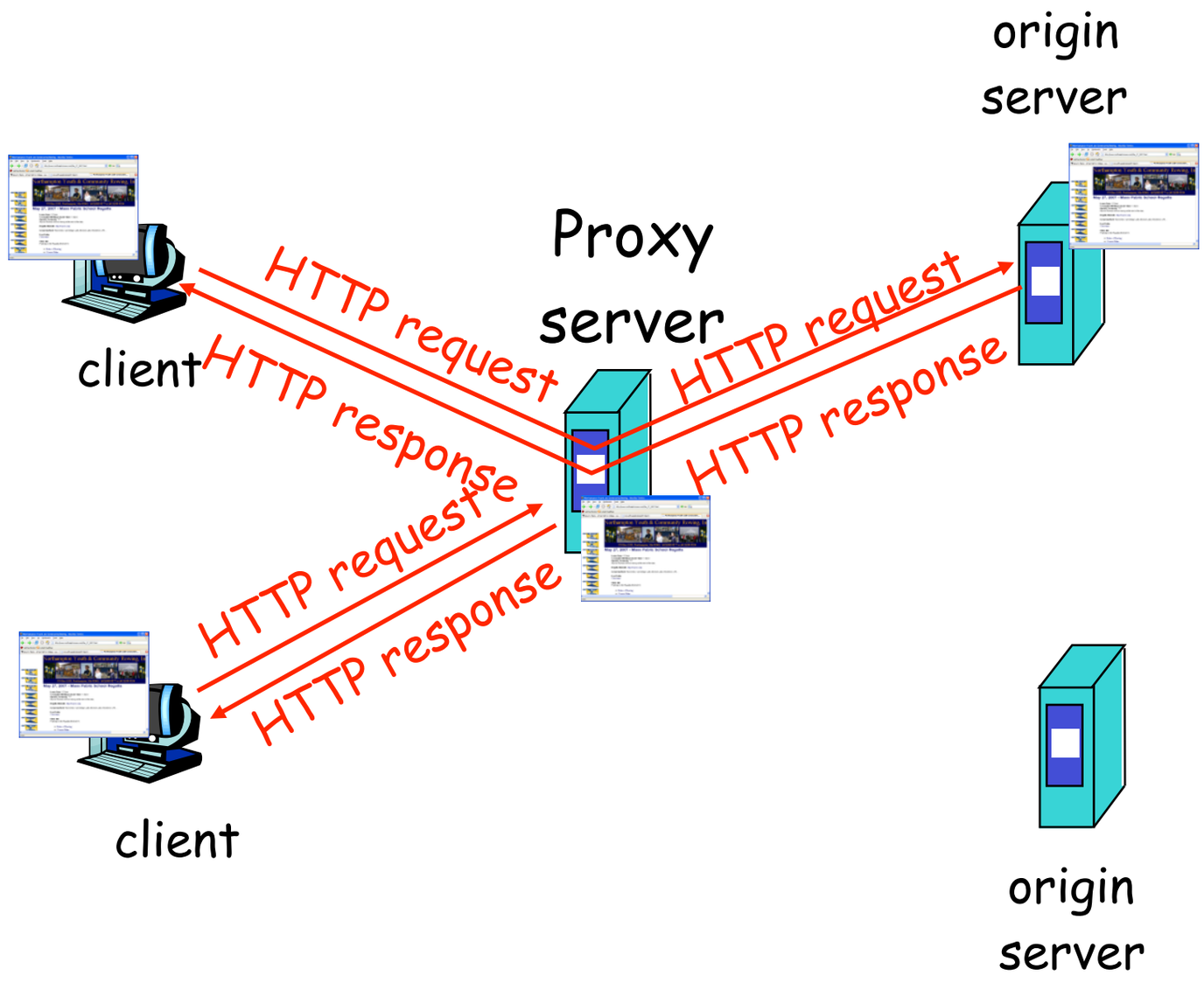


origin
server

Web caches (proxy server)

Goal: satisfy client request without involving origin server

- user sets browser: Web accesses via cache
- browser sends all HTTP requests to cache
  - object in cache: cache returns object
  - else cache requests object from origin server, then returns object to client



More about Web caching

- cache acts as both client and server
- typically cache is installed by ISP (university, company, residential ISP)

More about Web caching

- cache acts as both client and server
- typically cache is installed by ISP (university, company, residential ISP)

why Web caching?

More about Web caching

- cache acts as both client and server
- typically cache is installed by ISP (university, company, residential ISP)

why Web caching?

- reduce response time for client request

More about Web caching

- cache acts as both client and server
- typically cache is installed by ISP (university, company, residential ISP)

why Web caching?

- reduce response time for client request
- reduce traffic on an institution's access link.

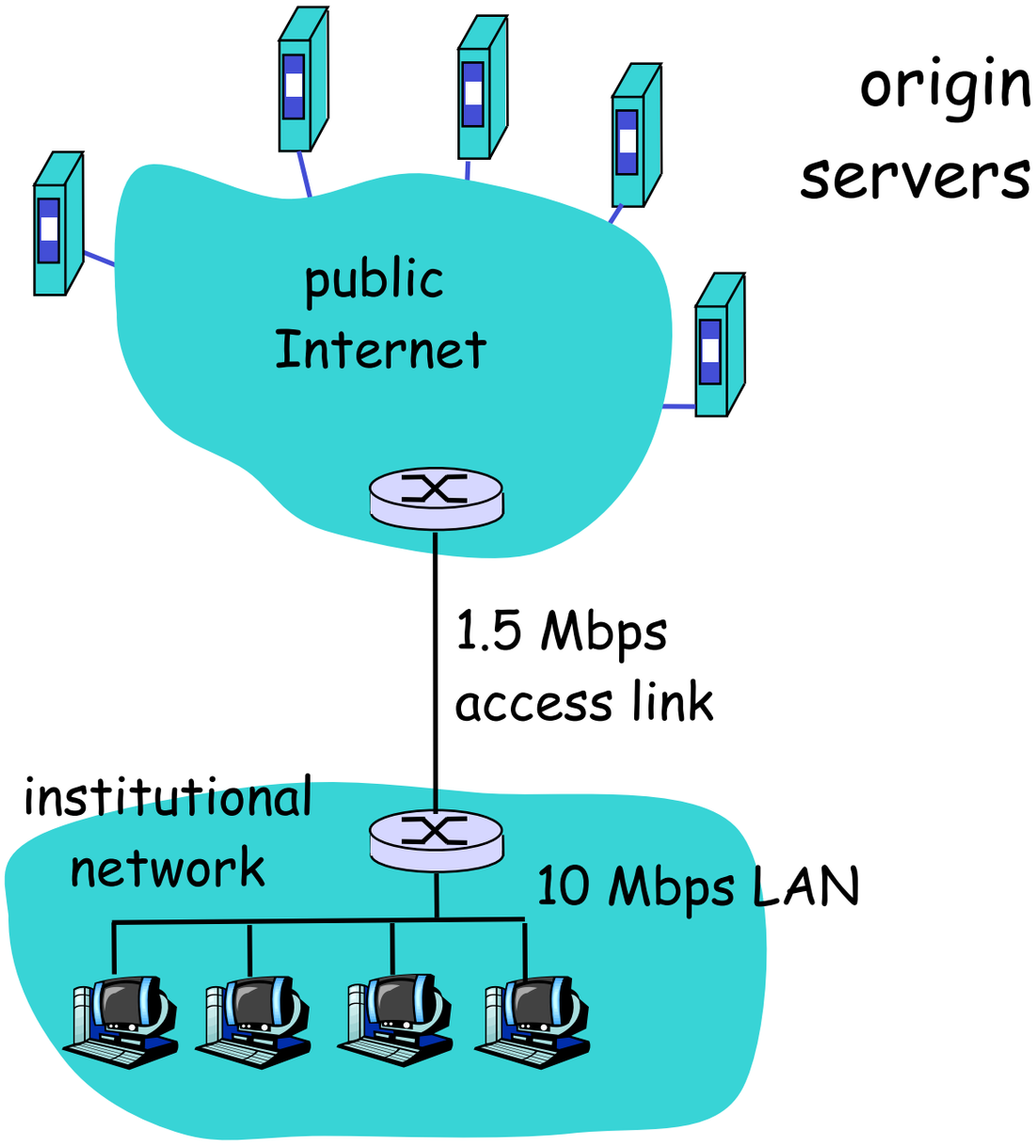
More about Web caching

- cache acts as both client and server
- typically cache is installed by ISP (university, company, residential ISP)

why Web caching?

- reduce response time for client request
- reduce traffic on an institution's access link.
- Internet dense with caches: enables "poor" content providers to effectively deliver content

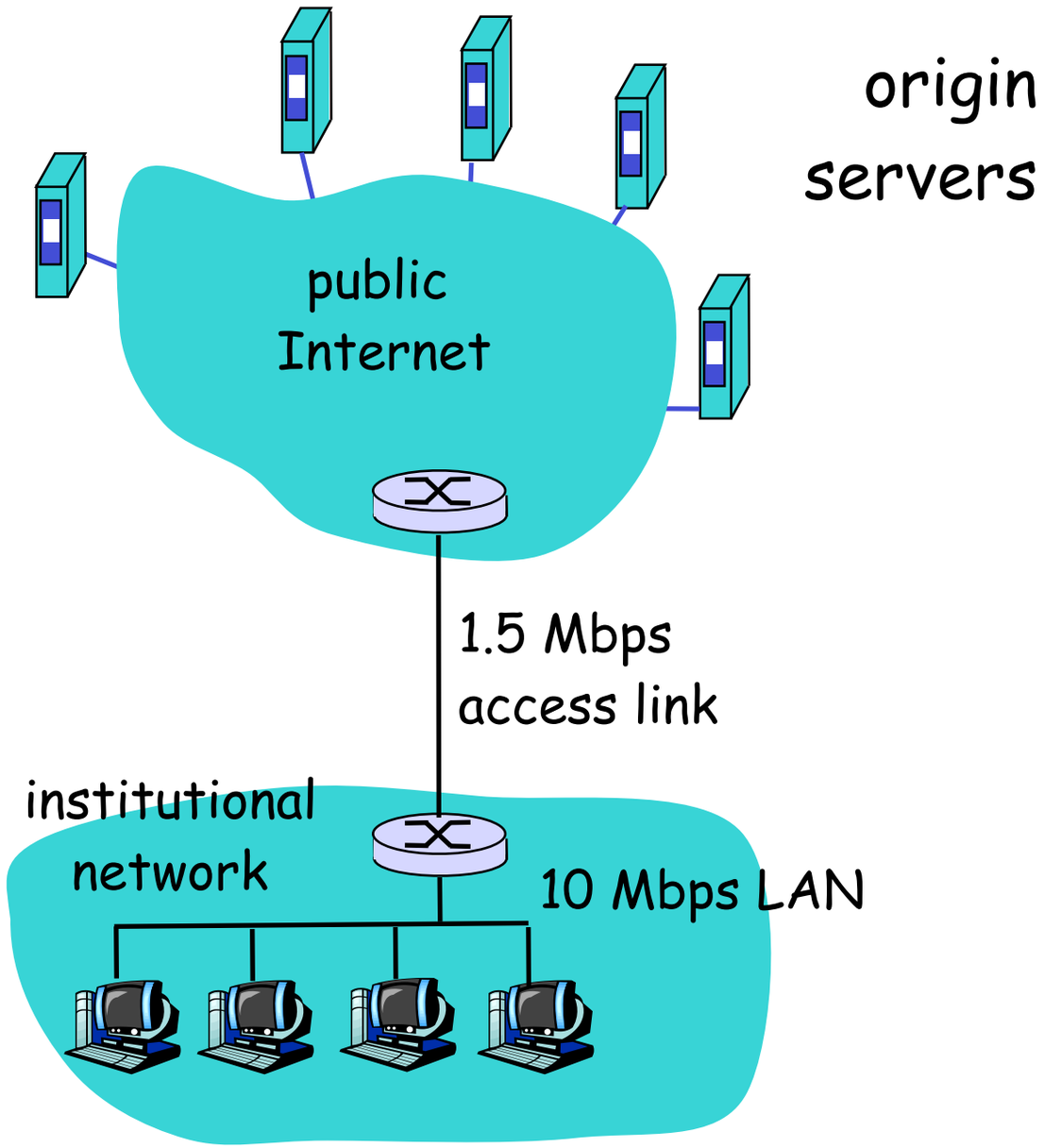
Caching example



Caching example

assumptions

- average object size = 100,000 bits
- avg. request rate from institution's browsers to origin servers = 15/sec
- delay from institutional router to any origin server and back to router = 2 sec

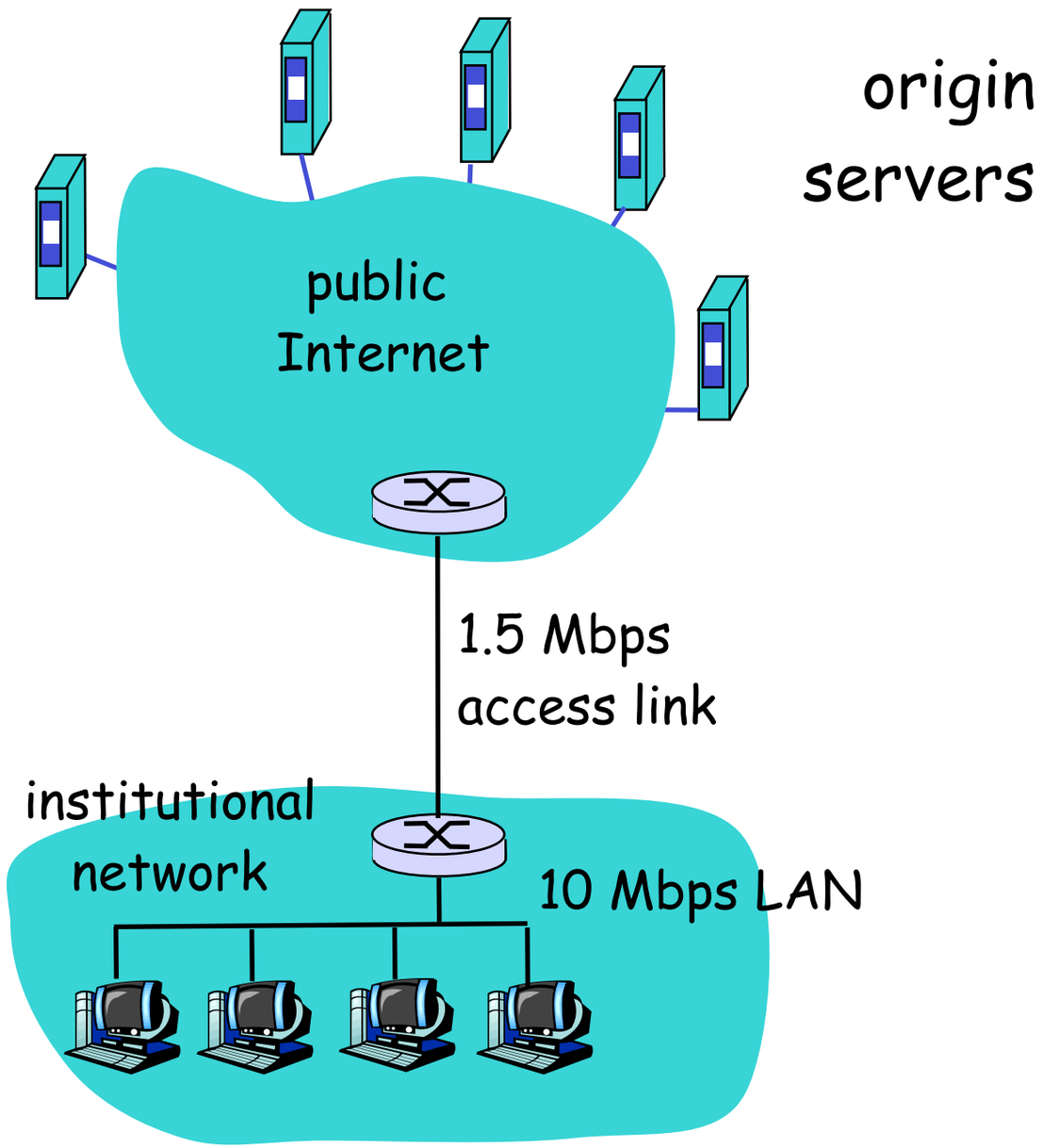


Caching example

assumptions

- average object size = 100,000 bits
- avg. request rate from institution's browsers to origin servers = 15/sec
- delay from institutional router to any origin server and back to router = 2 sec

consequences



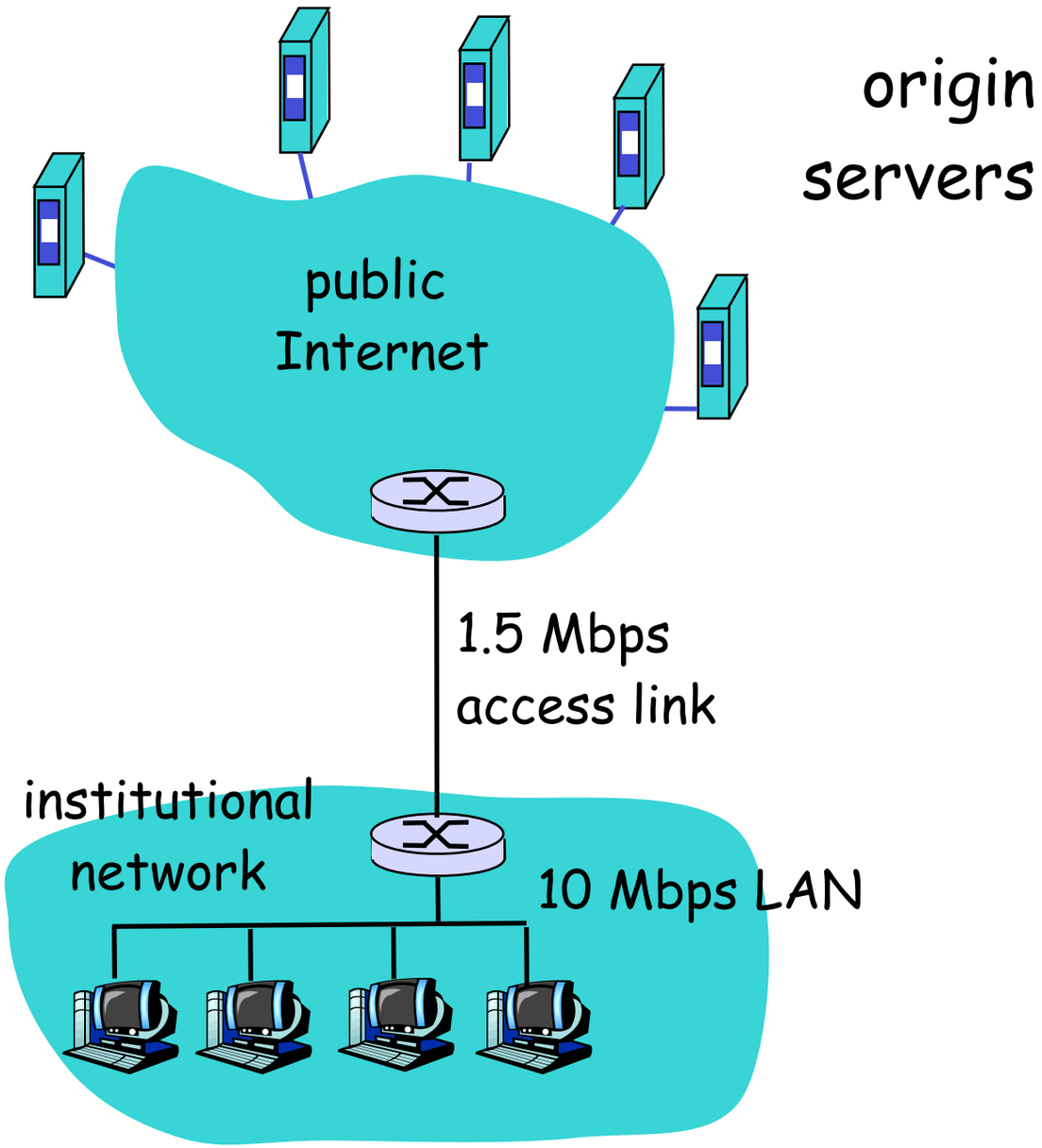
Caching example

assumptions

- average object size = 100,000 bits
- avg. request rate from institution's browsers to origin servers = 15/sec
- delay from institutional router to any origin server and back to router = 2 sec

consequences

- utilization on LAN = 15%



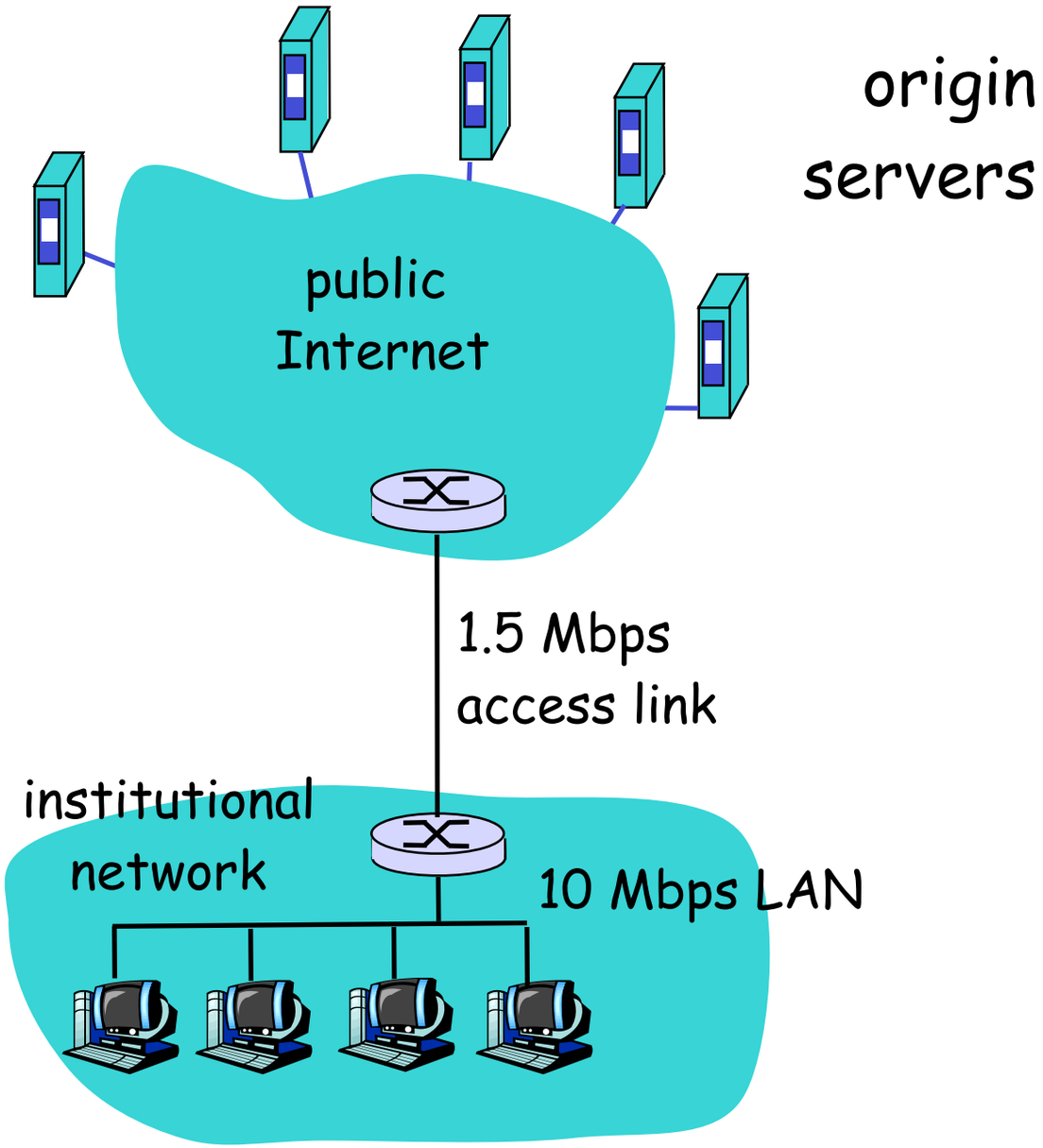
Caching example

assumptions

- average object size = 100,000 bits
- avg. request rate from institution's browsers to origin servers = 15/sec
- delay from institutional router to any origin server and back to router = 2 sec

consequences

- utilization on LAN = 15%
- utilization on access link = 100%



Caching example

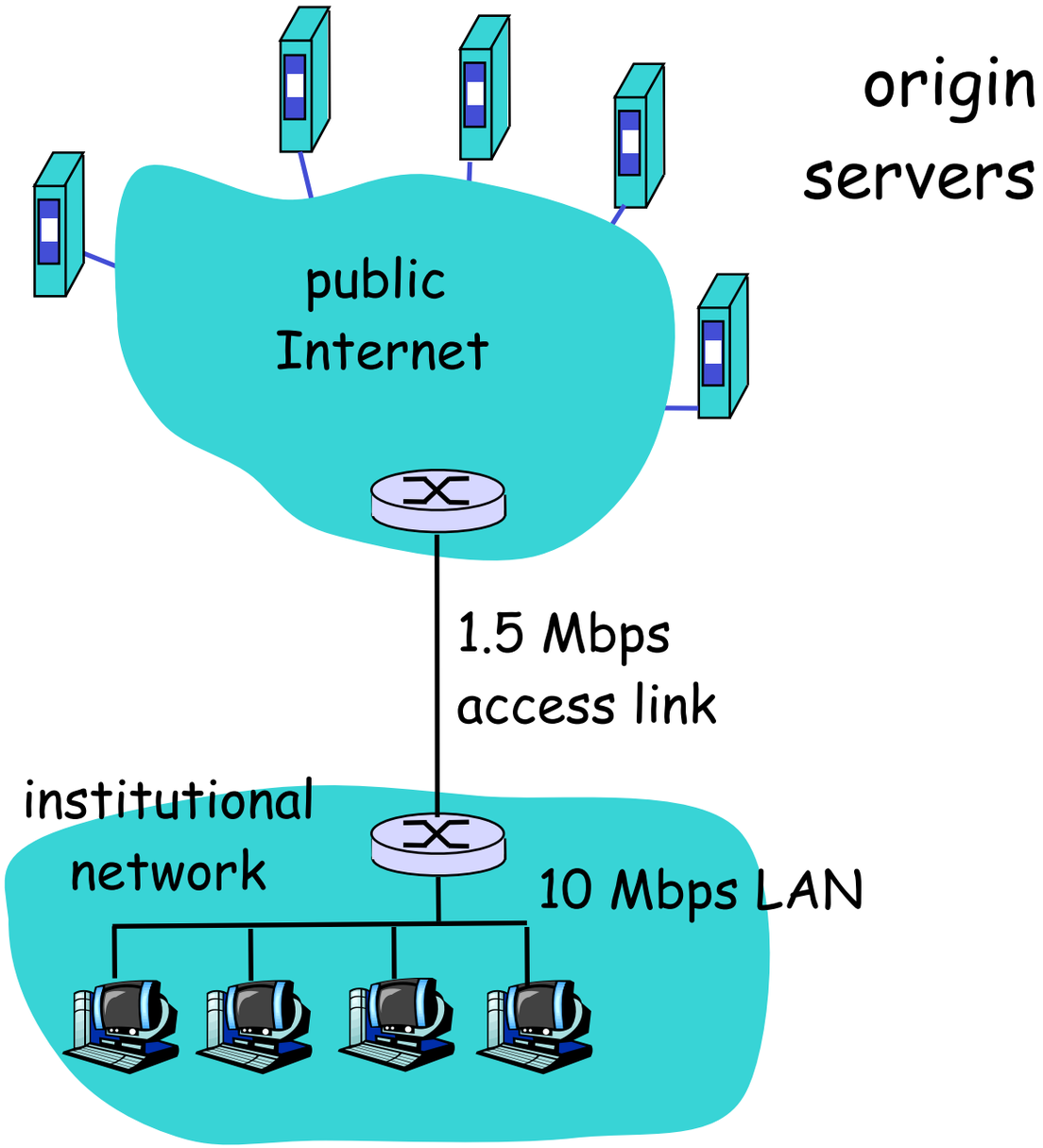
assumptions

- average object size = 100,000 bits
- avg. request rate from institution's browsers to origin servers = 15/sec
- delay from institutional router to any origin server and back to router = 2 sec

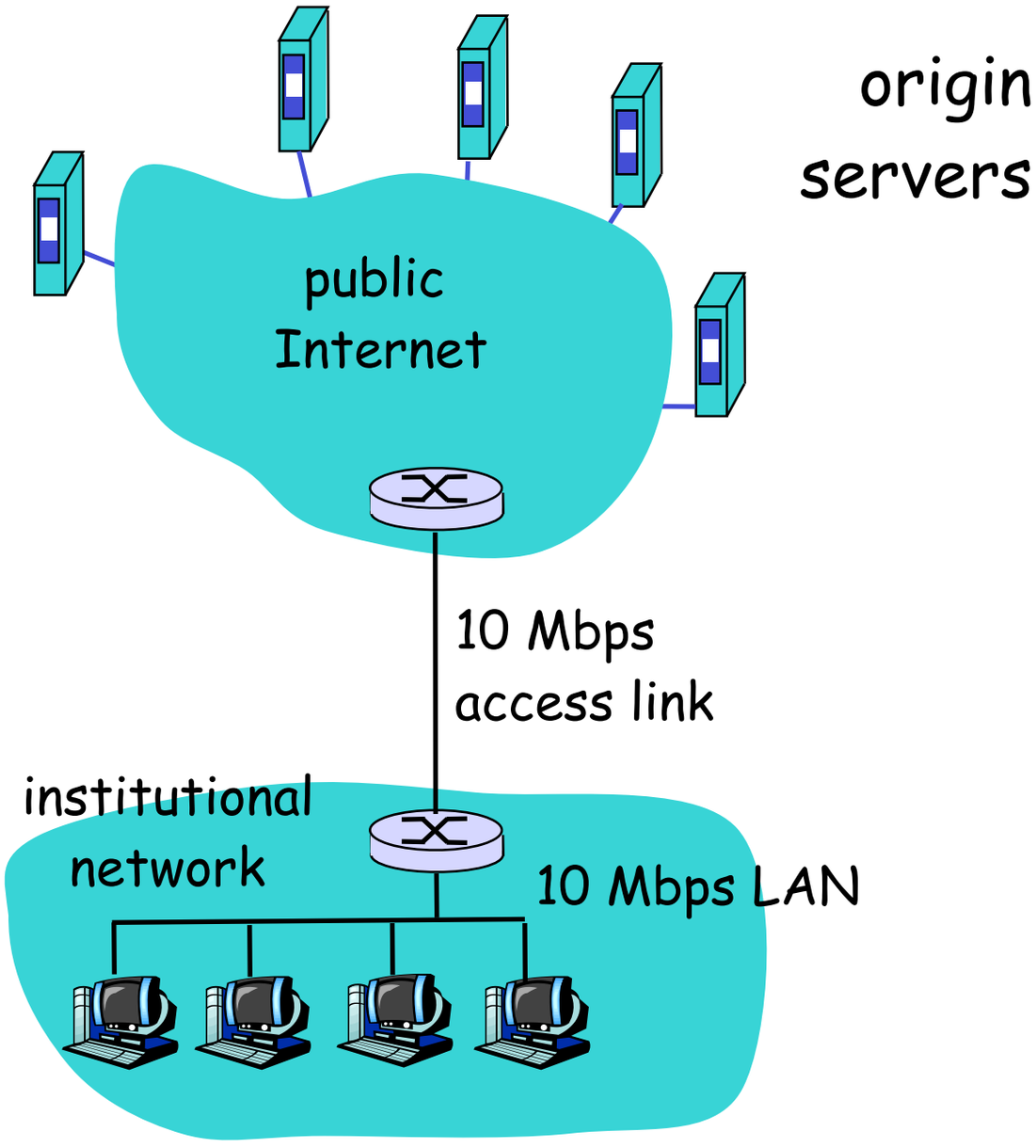
consequences

- utilization on LAN = 15%
- utilization on access link = 100%
- total delay = Internet delay + access delay + LAN delay

  = 2 sec + seconds + milliseconds



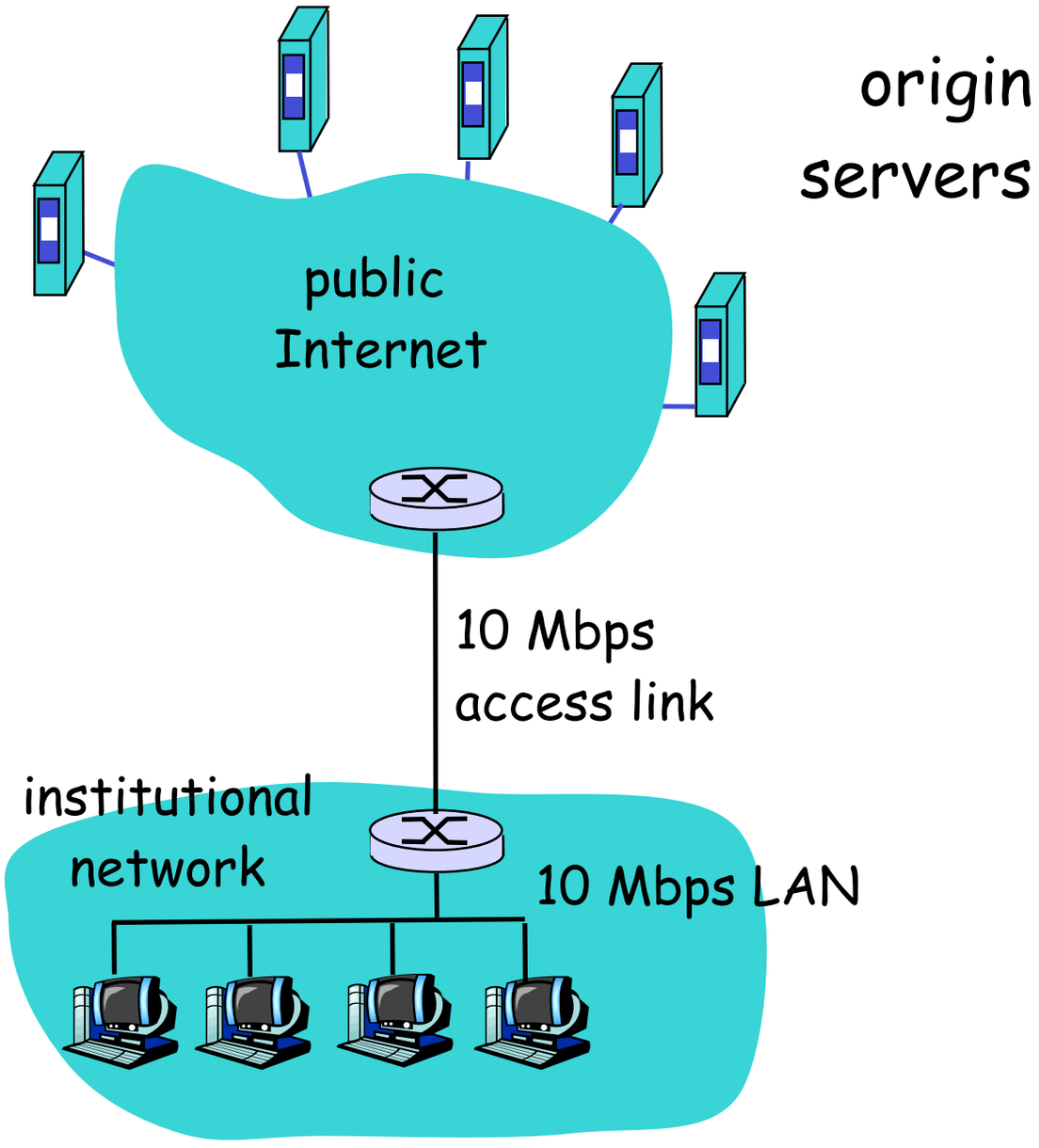
Caching example (cont)



Caching example (cont)

possible solution

- increase bandwidth of access link to, say, 10 Mbps



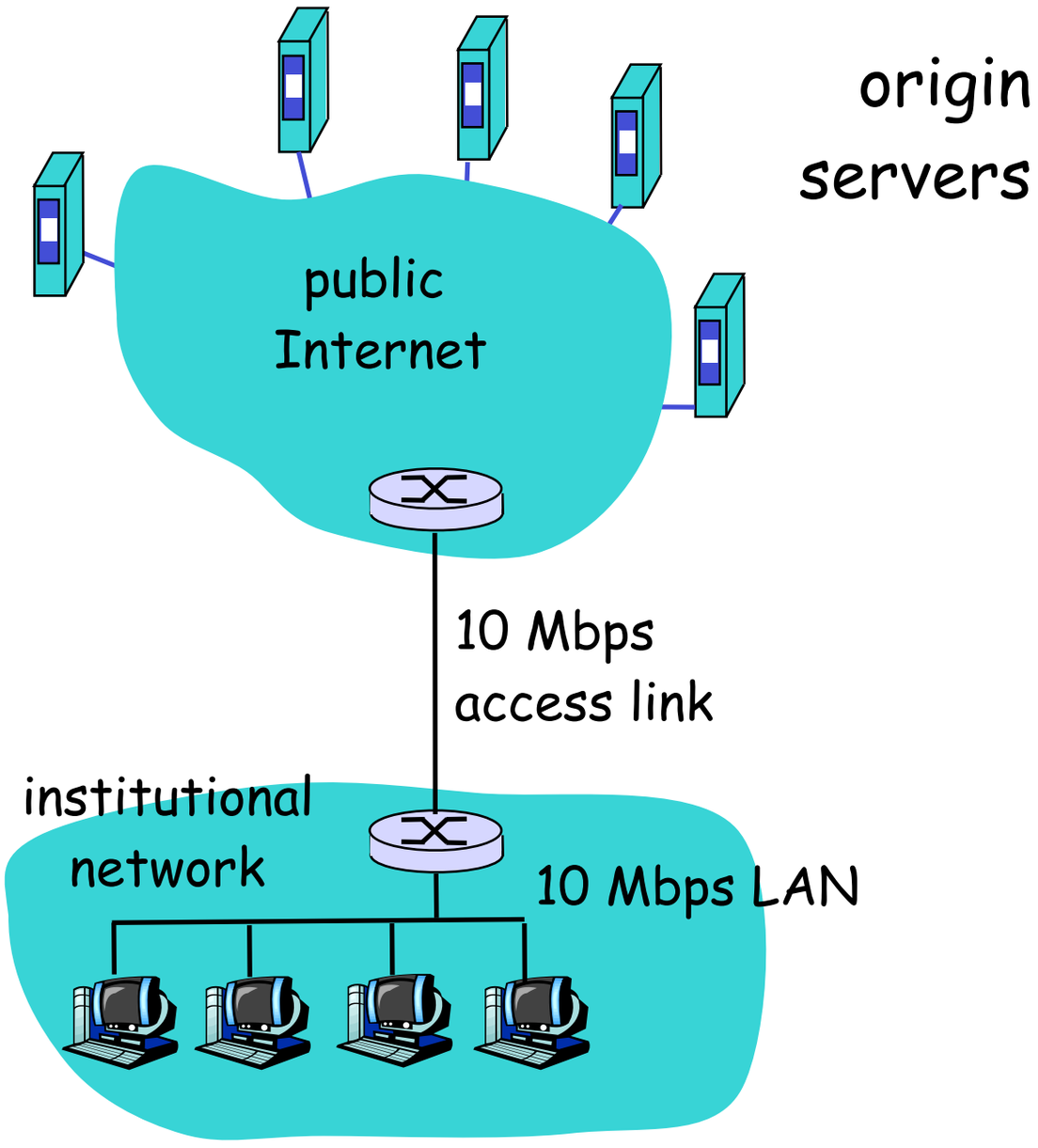
Caching example (cont)

possible solution

- increase bandwidth of access link to, say, 10 Mbps

consequence

- utilization on LAN = 15%



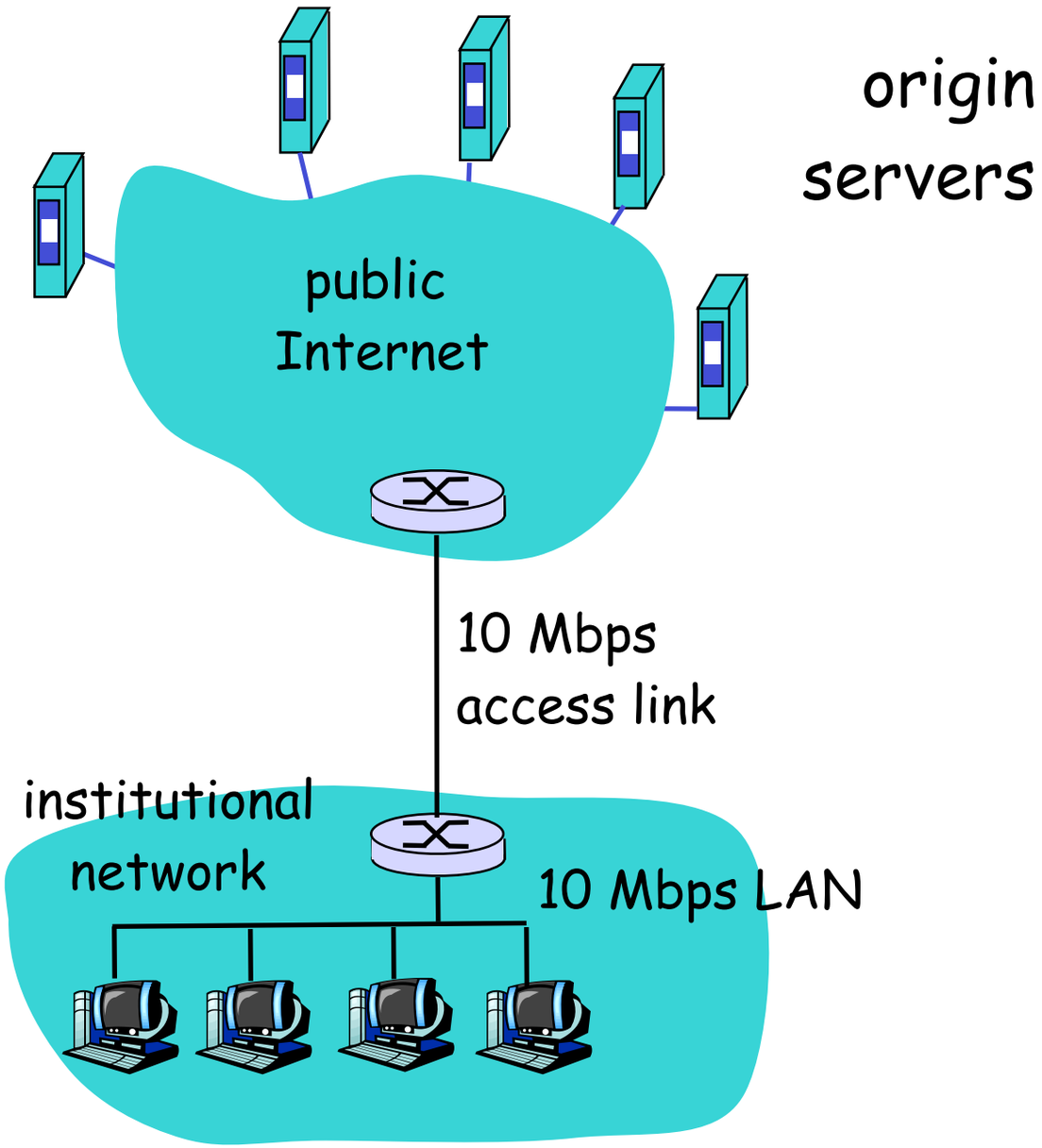
Caching example (cont)

possible solution

- increase bandwidth of access link to, say, 10 Mbps

consequence

- utilization on LAN = 15%
- utilization on access link = 15%



Caching example (cont)

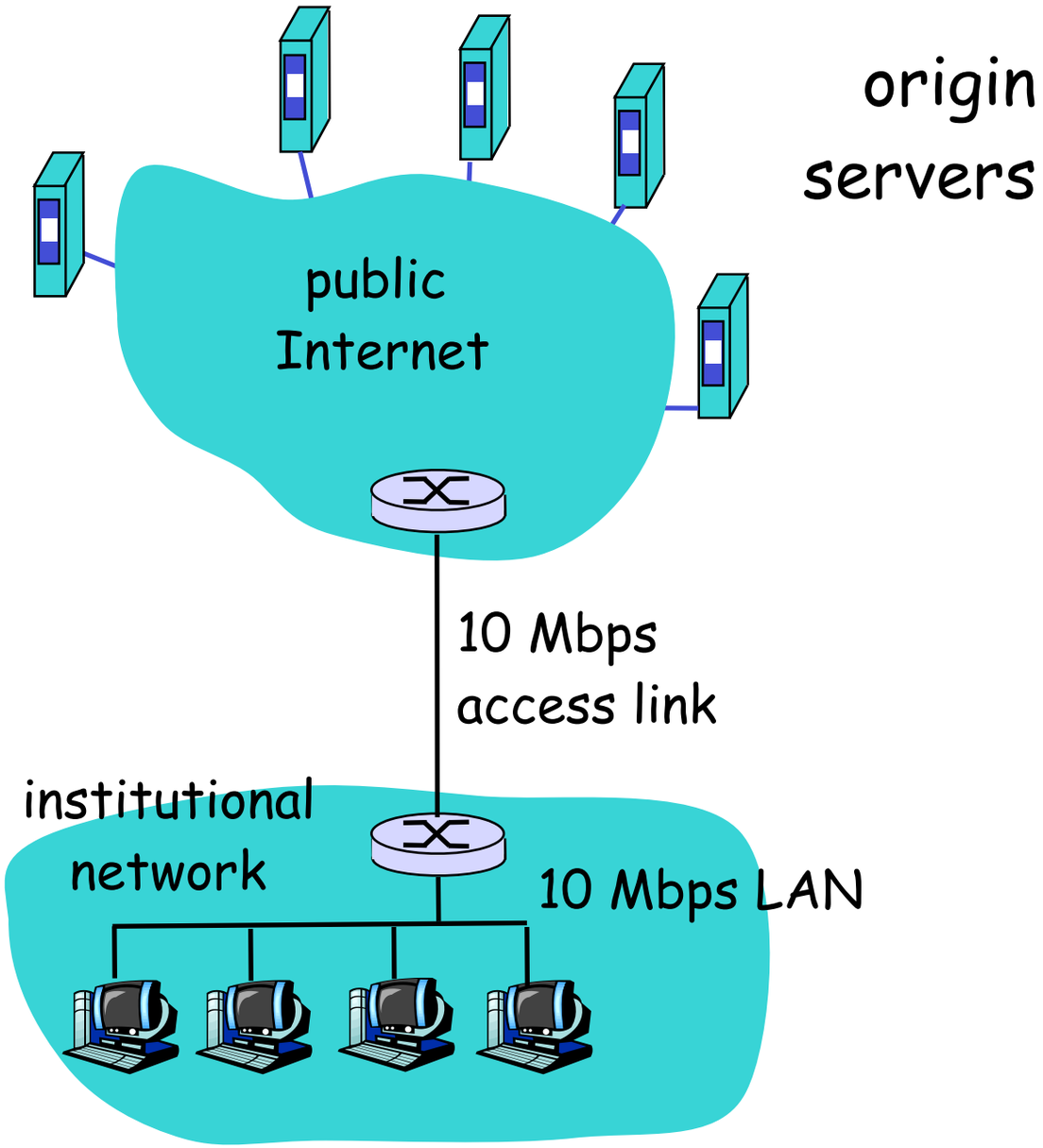
possible solution

- increase bandwidth of access link to, say, 10 Mbps

consequence

- utilization on LAN = 15%
- utilization on access link = 15%
- Total delay = Internet delay + access delay + LAN delay

  = 2 sec + msecs + msecs



Caching example (cont)

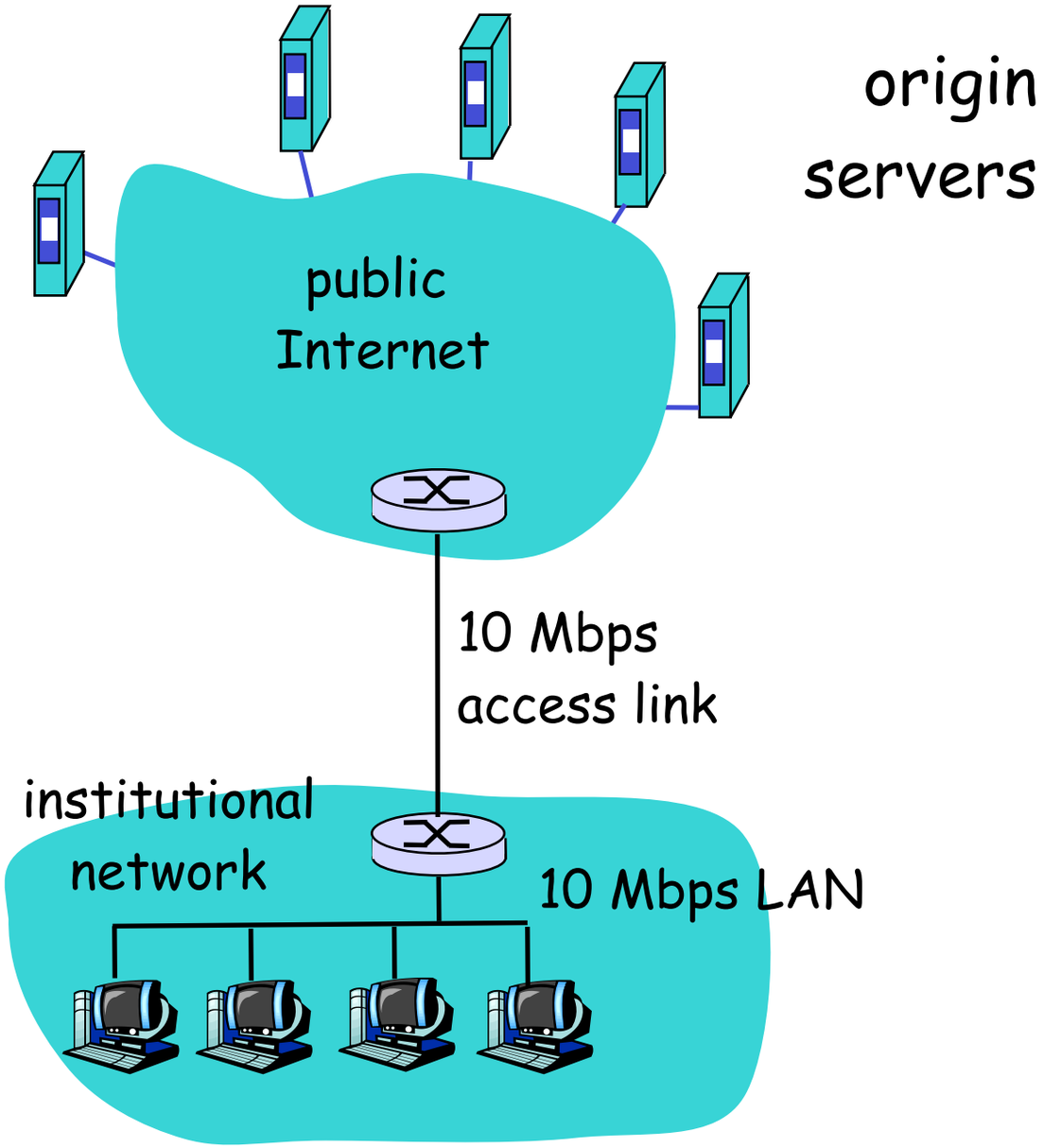
possible solution

- increase bandwidth of access link to, say, 10 Mbps

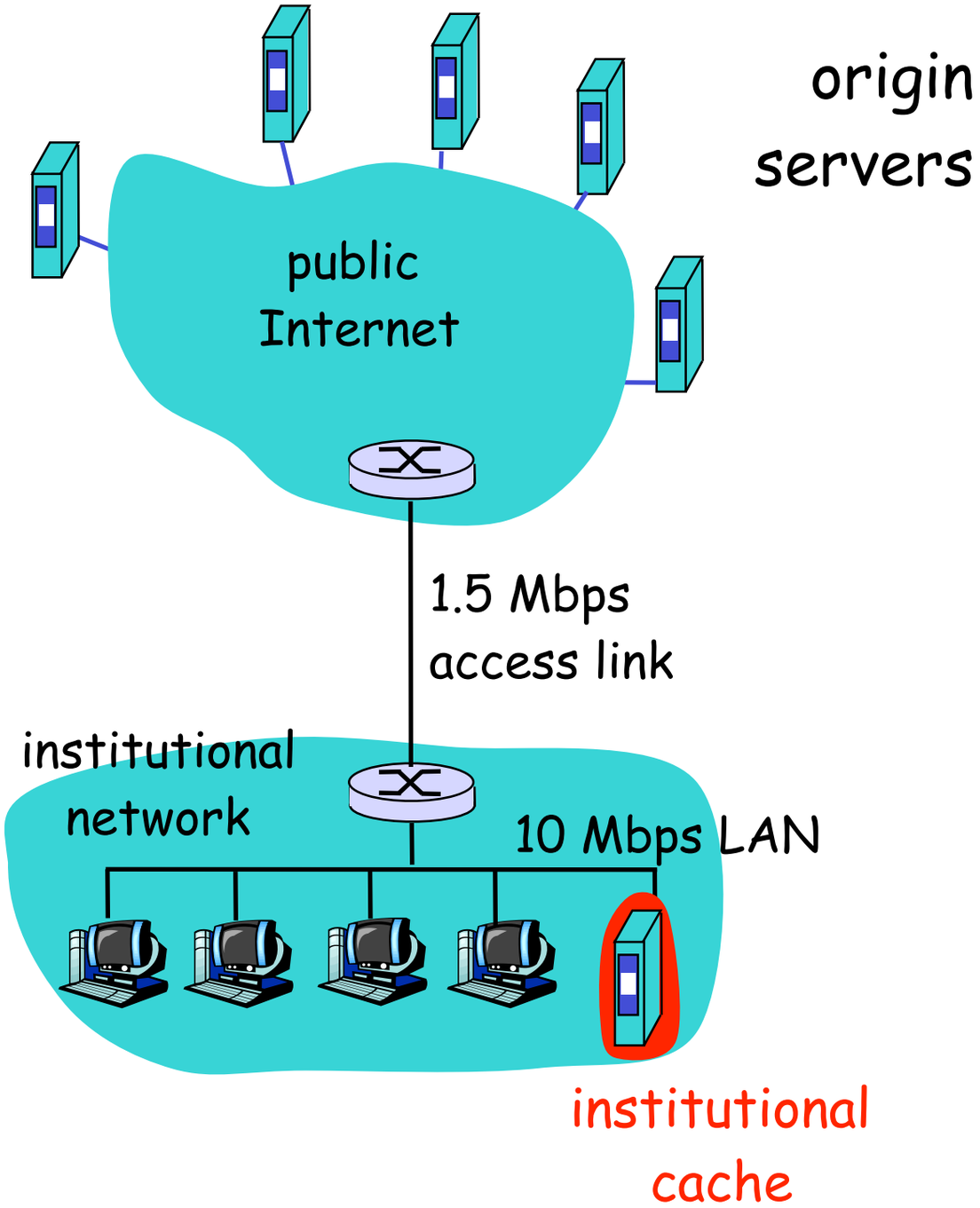
consequence

- utilization on LAN = 15%
- utilization on access link = 15%
- Total delay = Internet delay + access delay + LAN delay

  = 2 sec + msecs + msecs
- often a costly upgrade



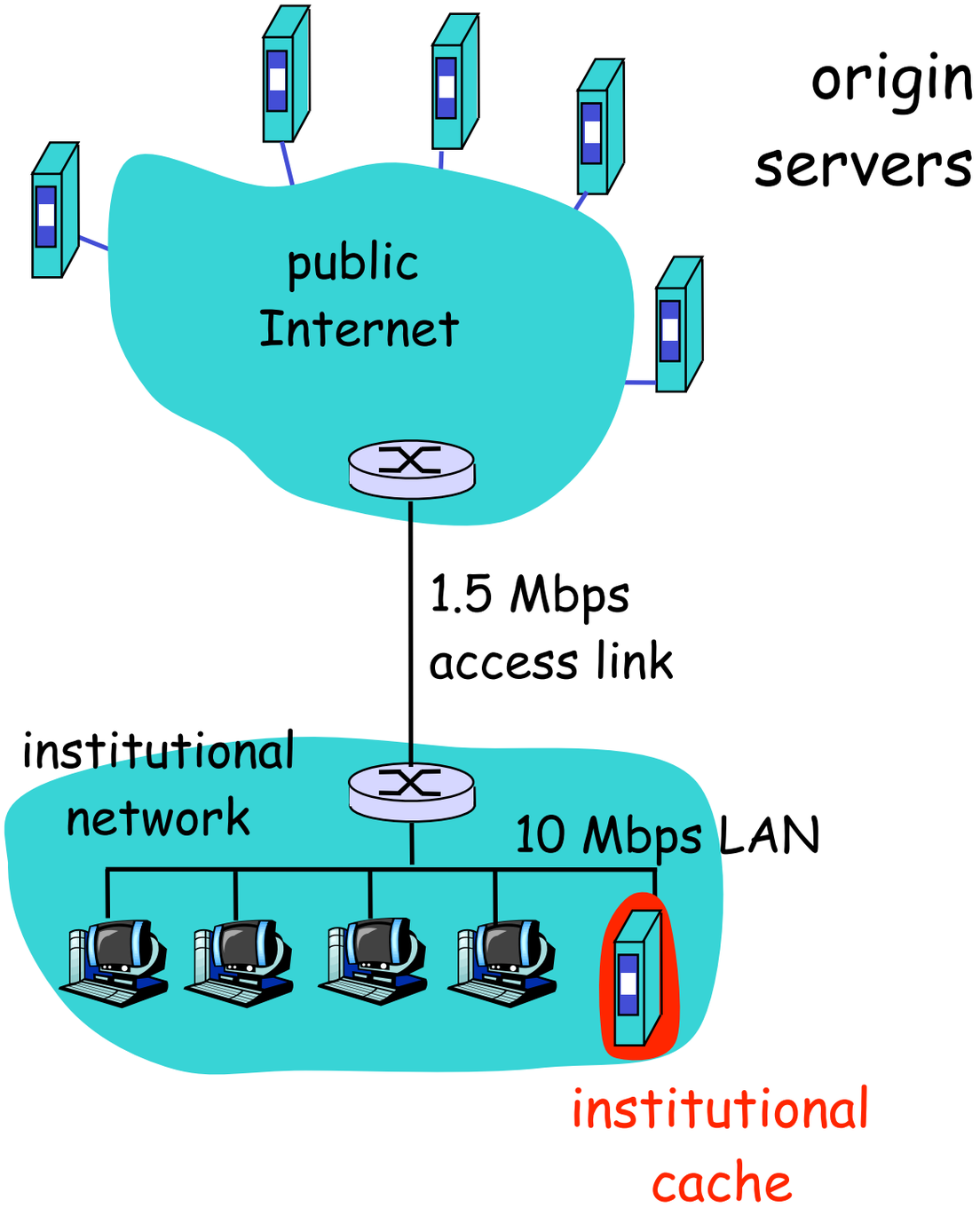
Caching example (cont)



Caching example (cont)

possible solution:

- install cache



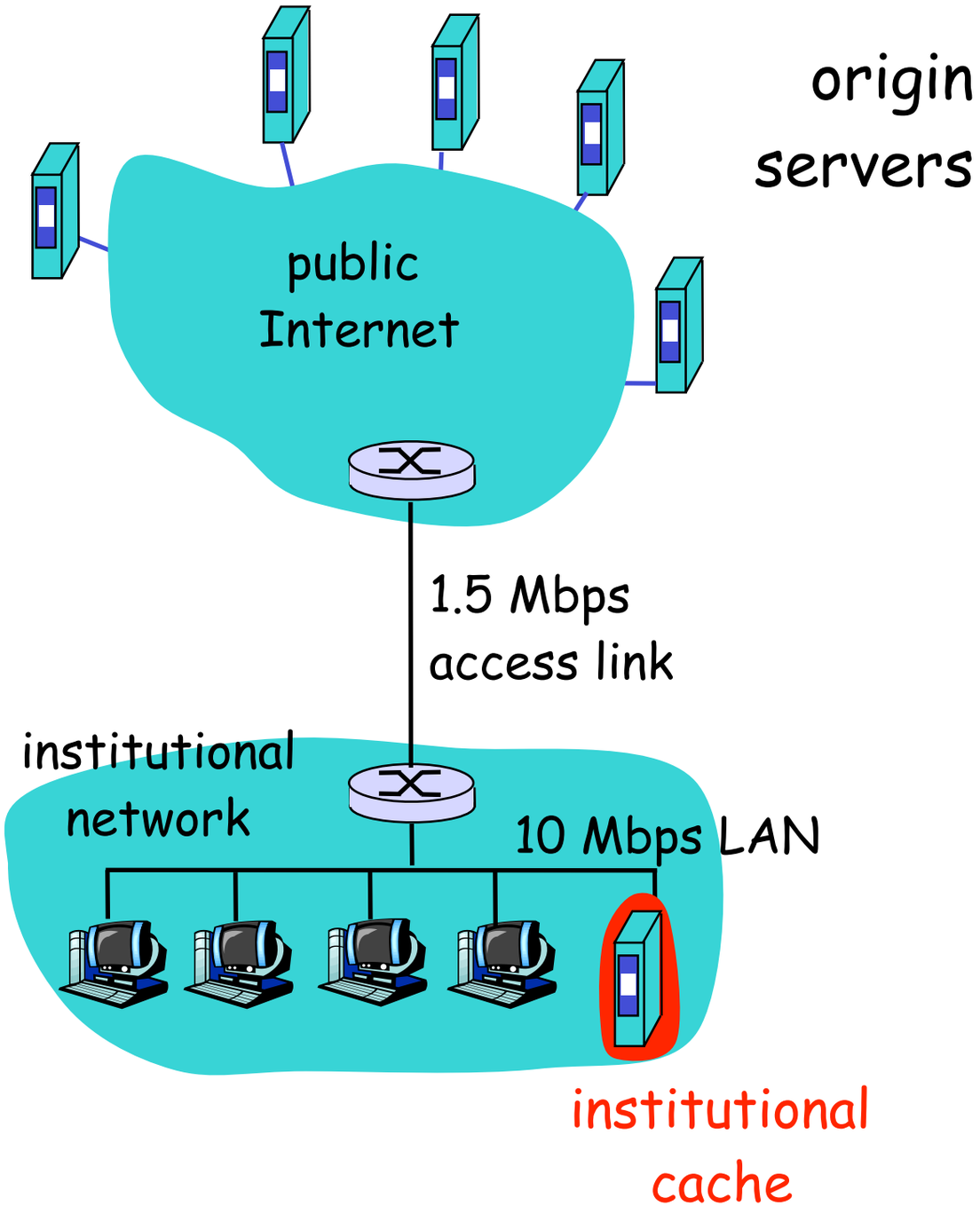
Caching example (cont)

possible solution:

- install cache

consequence

- suppose hit rate is 0.4



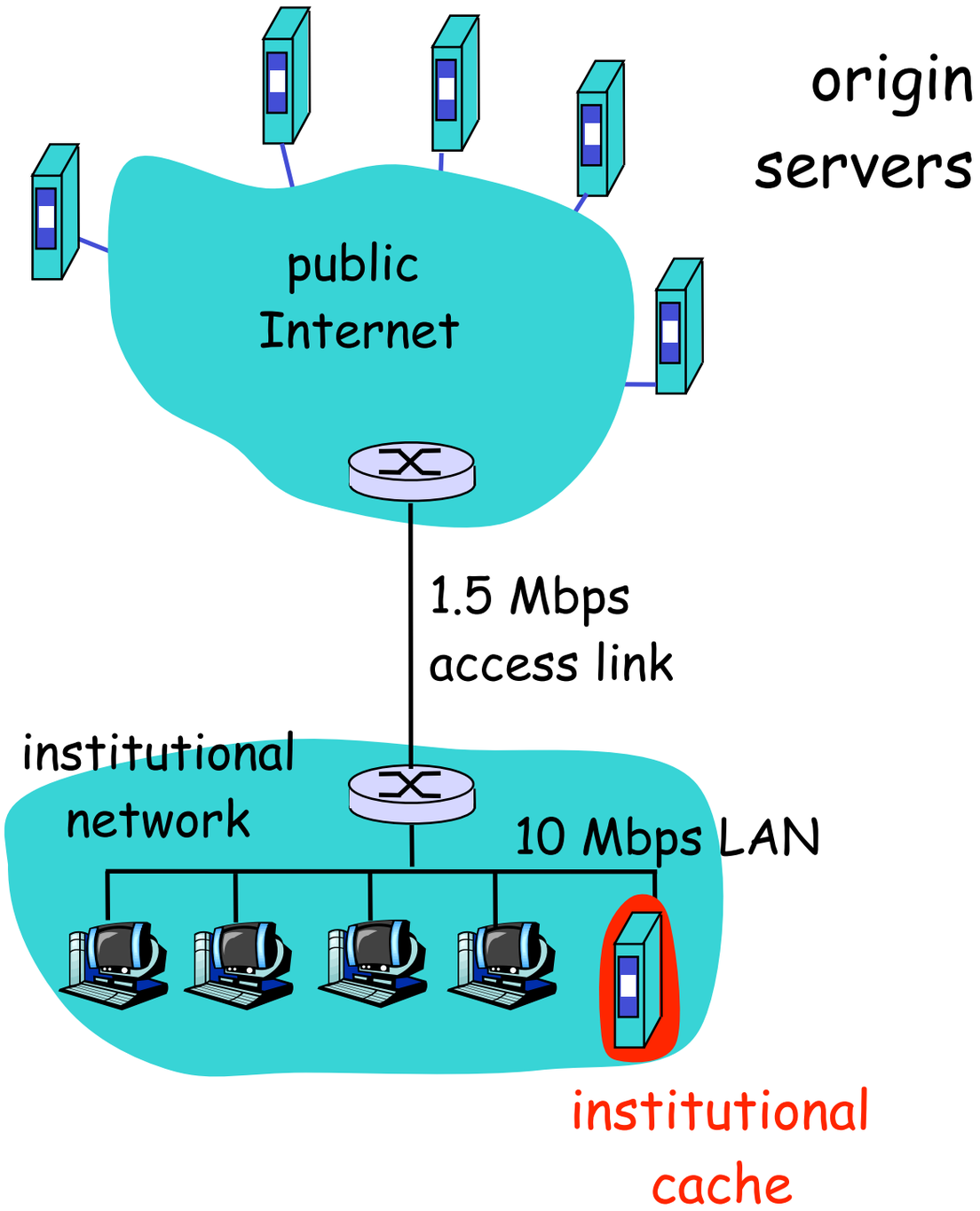
Caching example (cont)

possible solution:

- install cache

consequence

- suppose hit rate is 0.4
  - 40% requests will be satisfied almost immediately
  - 60% requests satisfied by origin server



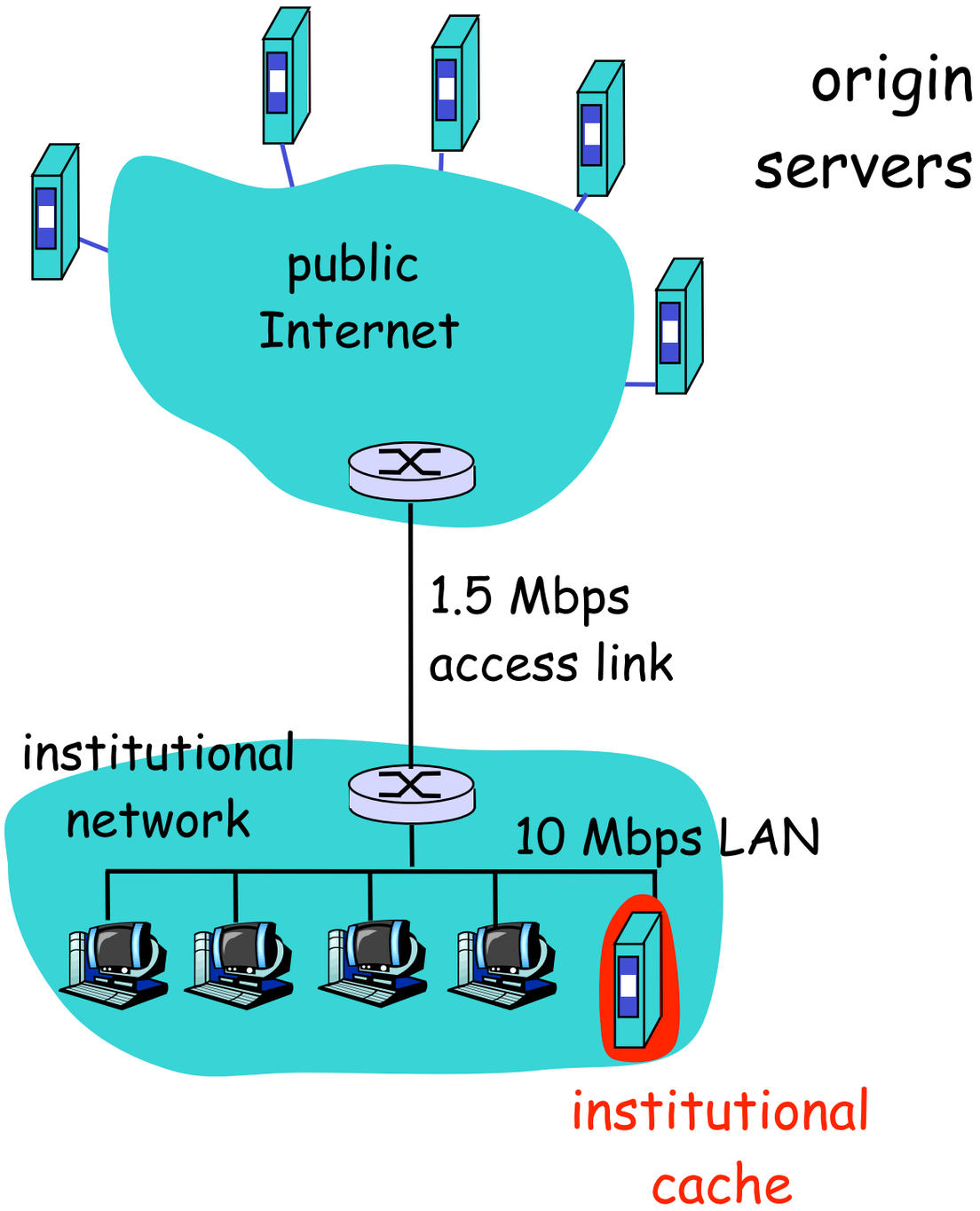
Caching example (cont)

possible solution:

- install cache

consequence

- suppose hit rate is 0.4
  - 40% requests will be satisfied almost immediately
  - 60% requests satisfied by origin server
- utilization of access link reduced to 60%, resulting in negligible delays (say 10 msec)



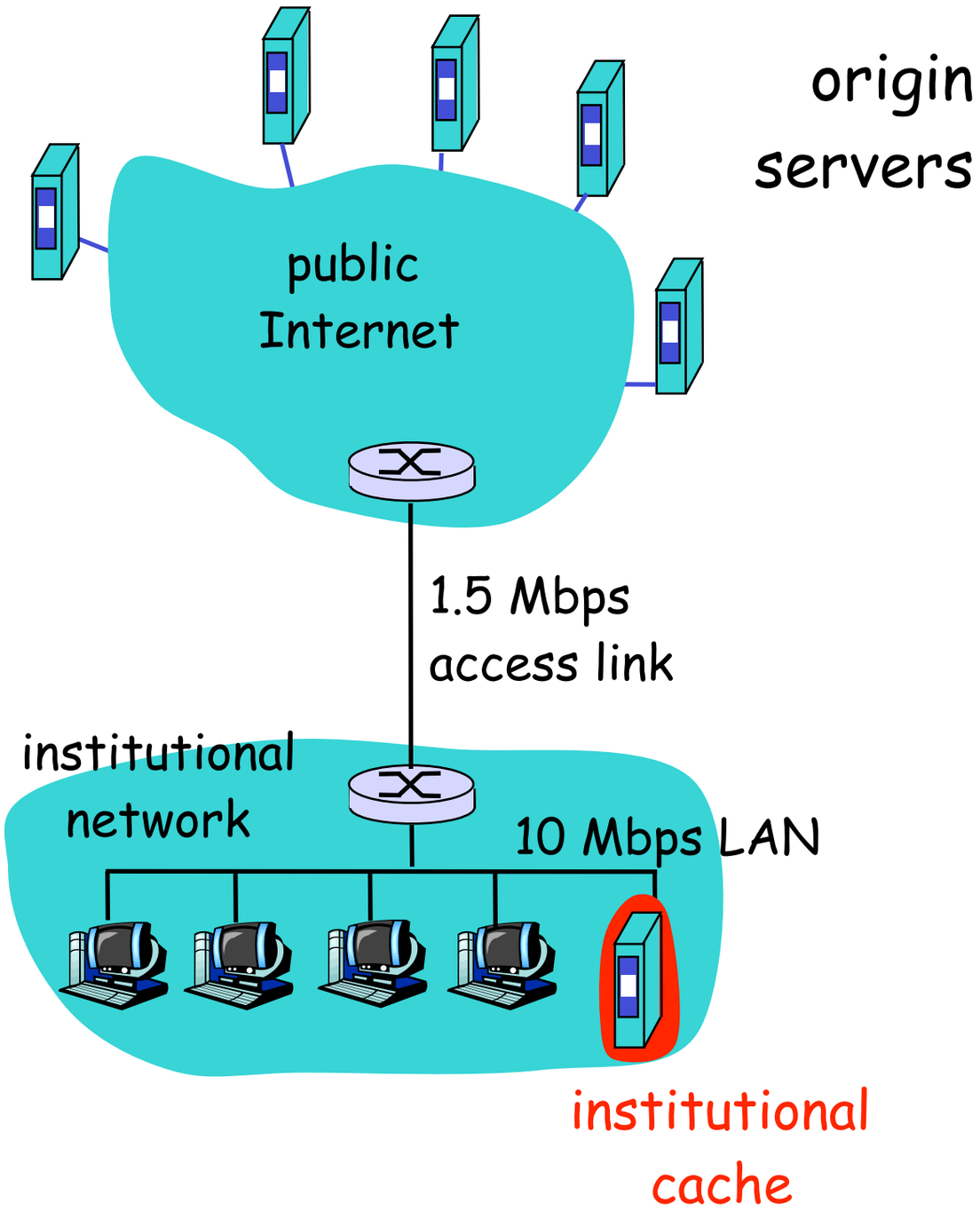
Caching example (cont)

possible solution:

- install cache

consequence

- suppose hit rate is 0.4
  - 40% requests will be satisfied almost immediately
  - 60% requests satisfied by origin server
- utilization of access link reduced to 60%, resulting in negligible delays (say 10 msec)
- total avg delay = Internet delay + access delay + LAN delay = 0.6*2 secs + 0.4*milliseconds < 1.4 secs



HTTP connections

basic HTTP

- one object sent over TCP connection

HTTP connections

basic HTTP

- one object sent over TCP connection

persistent HTTP

- multiple objects can be sent over single TCP connection between client, server.

Non-Persistent HTTP: Response time

Non-Persistent HTTP: Response time

definition of RTT: time for a small packet to travel from client to server and back.

Non-Persistent HTTP: Response time

definition of RTT: time for a small packet to travel from client to server and back.

response time:

- one RTT to initiate TCP connection

Non-Persistent HTTP: Response time

definition of RTT: time for a small packet to travel from client to server and back.

response time:

- one RTT to initiate TCP connection
- one RTT for HTTP request and first few bytes of HTTP response to return

Non-Persistent HTTP: Response time

definition of RTT: time for a small packet to travel from client to server and back.

response time:

- one RTT to initiate TCP connection
- one RTT for HTTP request and first few bytes of HTTP response to return
- file transmission time

total = 2RTT+transmit time

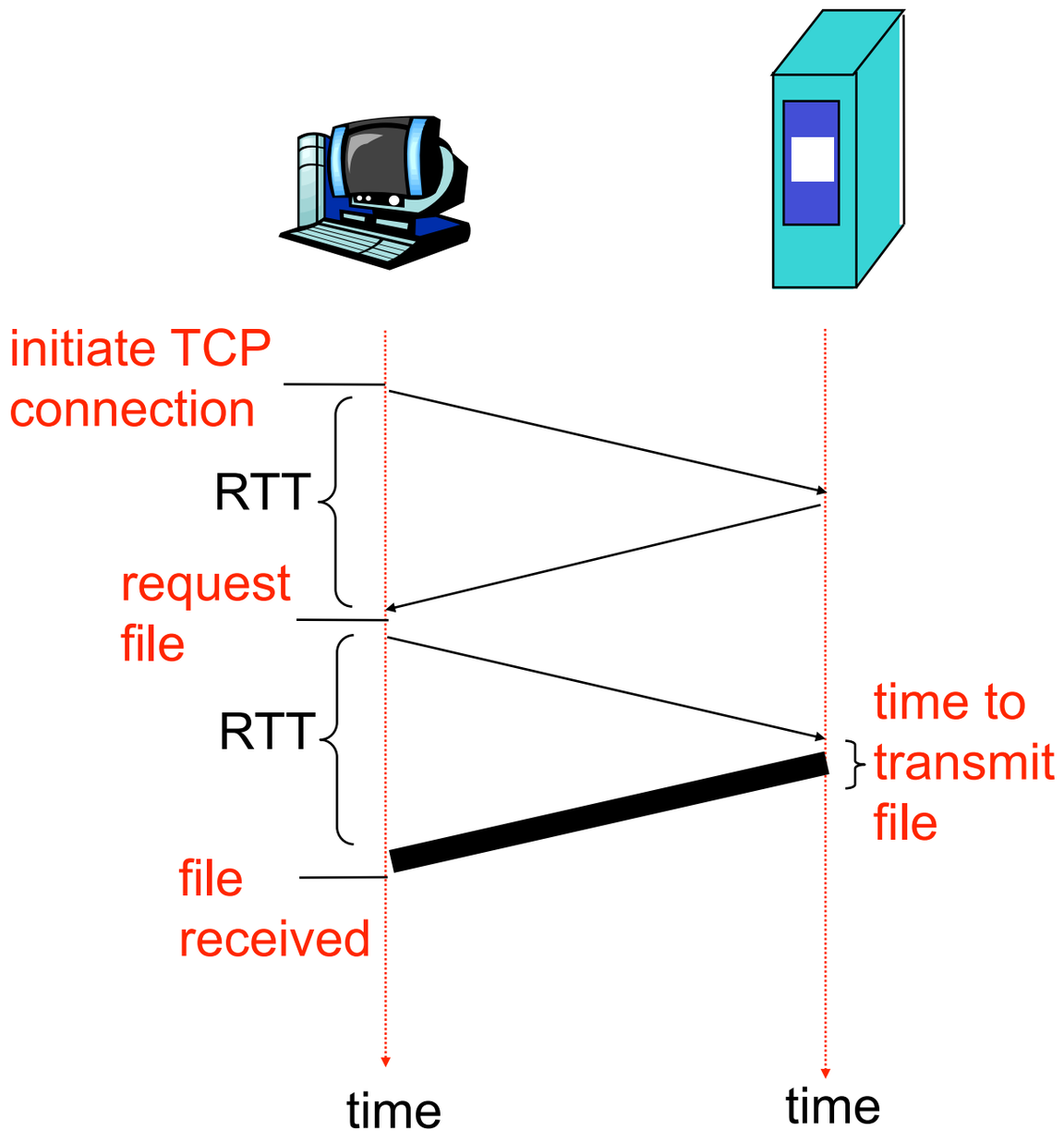
Non-Persistent HTTP: Response time

definition of RTT: time for a small packet to travel from client to server and back.

response time:

- one RTT to initiate TCP connection
- one RTT for HTTP request and first few bytes of HTTP response to return
- file transmission time

total = 2RTT+transmit time



Persistent HTTP

non-persistent HTTP issues:

- requires 2 RTTs per object
- OS overhead for each TCP connection
- browsers often open parallel TCP connections to fetch referenced objects

Persistent HTTP

non-persistent HTTP issues:

- requires 2 RTTs per object
- OS overhead for each TCP connection
- browsers often open parallel TCP connections to fetch referenced objects

persistent HTTP

- server leaves connection open after sending response
- subsequent HTTP messages between same client/server sent over open connection
- client sends requests as soon as it encounters a referenced object
- as little as one RTT for all the referenced objects