Client-centered Load Distribution (C^2LD)
A Mechanism for Constructing Responsive Web Services

Vittorio Ghini, Fabio Panzieri, Marco Roccetti
Dipartimento di Scienze dell'Informazione,
Università di Bologna

- a downloading mechanism, devoted to replicated Web services
- implemented at the browser site
- requires fragments of documents from different replicas, dynamically
- provides the user with timely responses and high availability
- experiments validate the effectiveness

Web Services

- based on
  - HTTP protocol
  - client server architecture
- QoS requirements (user viewpoint): Responsiveness, i.e.
  - availability

    percentage of served requests
  - timeliness

    User Response Time (URT)
- Introduction of redundancy by replicating Web servers
  - each request is served by a replica server

Replicated Web Servers

- Locally replicated Web Servers
  - more servers into a cluster
  - a gateway distributes the requests
  - increasing of responses per second
  - no variation of transmission delay

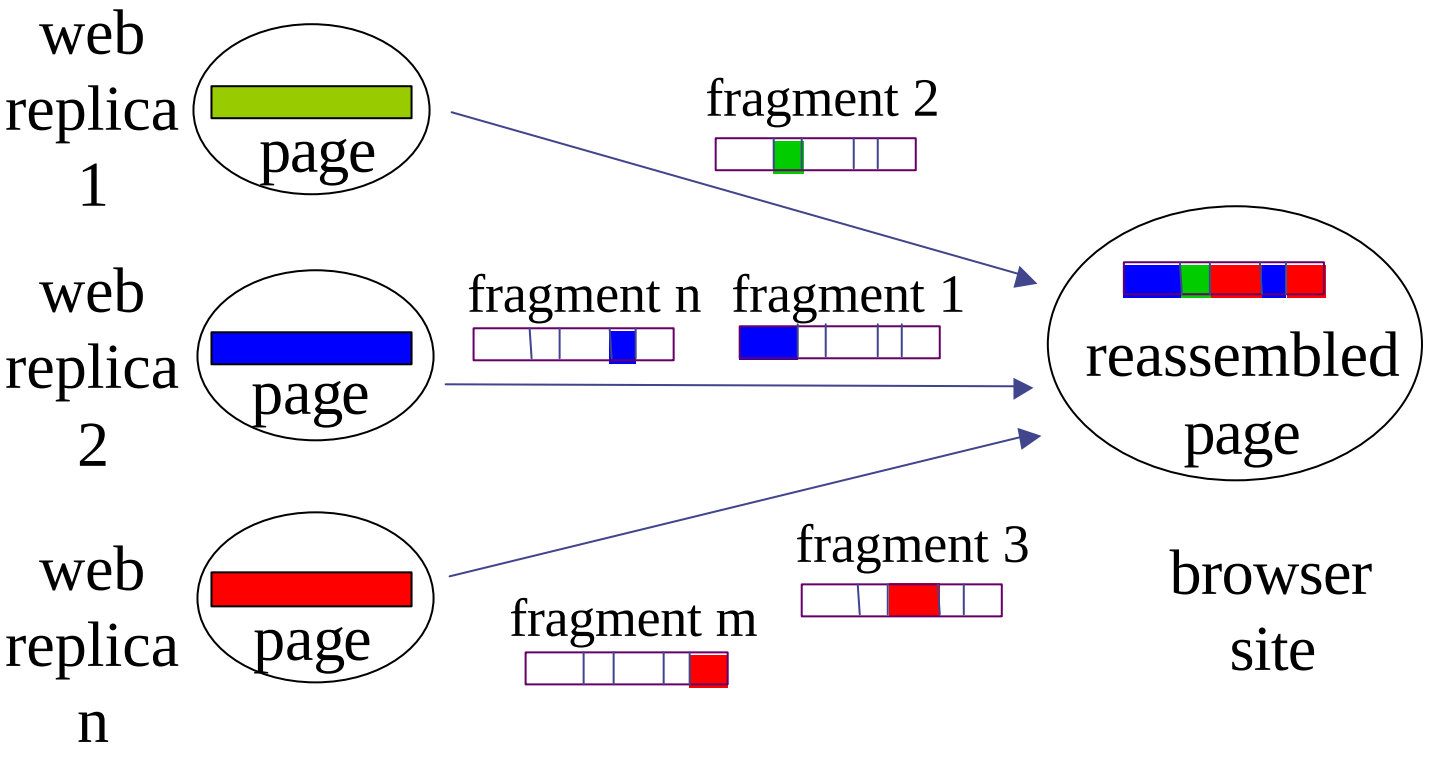
- Replicas distributed across Internet
  - different transmission delays between browser and servers
  - choice of most convenient replica for each browser request
  - mapping between hostname (into URL) and replica IP address
  - choice criteria such as round-robin, or QoS based
  - each request depend on the chosen replica only.

C^2LD: Client-centered Load Distribution

- a downloading mechanism that:
  - make use of replicated Web servers
  - works on browser (or proxy) side
  - don't bind a browser's request, for a web page, to a replica only
  - splits the browser's requests into more sub requests for page fragments
  - involves all replica servers in the retrieval of a given page
  - requires one or more fragments to each replica
  - requires larger fragments to fast replicas

C²LD approach: Fragment Requests

- C²LD splits the browser's requests into more sub requests for fragments.
- Each fragment is required to a replica
- When a fragment is received from a replica, another fragment is required to that replica



C^2LD approach: Adaptivity

- C^2LD adapts dynamically its fragment sub requests to the replicas, depending on both network and replicas performances.
- C^2LD monitors periodically the performances, and select those replicas that can provide the fragments.
- Larger fragments are required to the faster replicas, in order to receive entirely the fragments at the end of the monitoring period.
- For each sub requests an internal timeout is set. If this timeout expires before the requested fragment is received, the fragment is required to another replica.

C²LD analytical model (1)

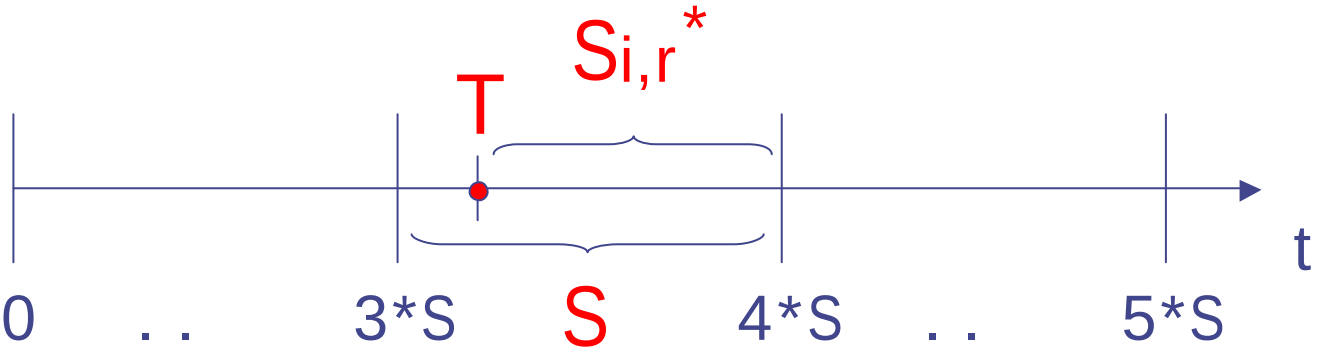
- User Specified Deadline USD: the extent of time a user is willing to wait for a page
- Monitoring period S: C²LD expect to receive the fragments at the end of S
- given:

i index of the replica server

r index of the sub request to replica i

T the instant in which C²LD receives entirely the response for the sub request r-1, from replica i

$\mathbf{S_{i,r}^*}$ the remaining time before the end of the current monitoring period S



- at the instant T, C²LD requires a new fragment, to the replica i, trying to receive the response at the end of the current period S

C²LD analytical model (2)

- C²LD assumes that the replica i responds to the new sub request r, providing the Data Rate $DR_{i,r}$, as experimentally measured during the previous sub request r-1.

$$DR_{i,r} = \frac{PS_{i,r-1}}{URT_{i,r-1}}$$ expected data rate

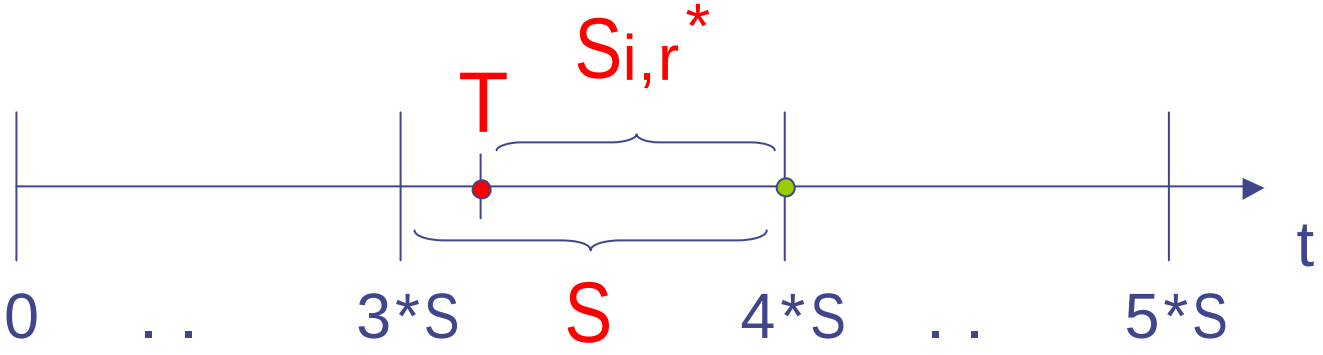
where

$PS_{i,r-1}$ size of fragment required with the sub request r-1

$URT_{i,r-1}$ response time of request r-1

- The size of fragment to be requested to the replica i, with the sub request r, is

$$PS_{i,r} = DR_{i,r} \cdot S_{i,r}^{*}$$

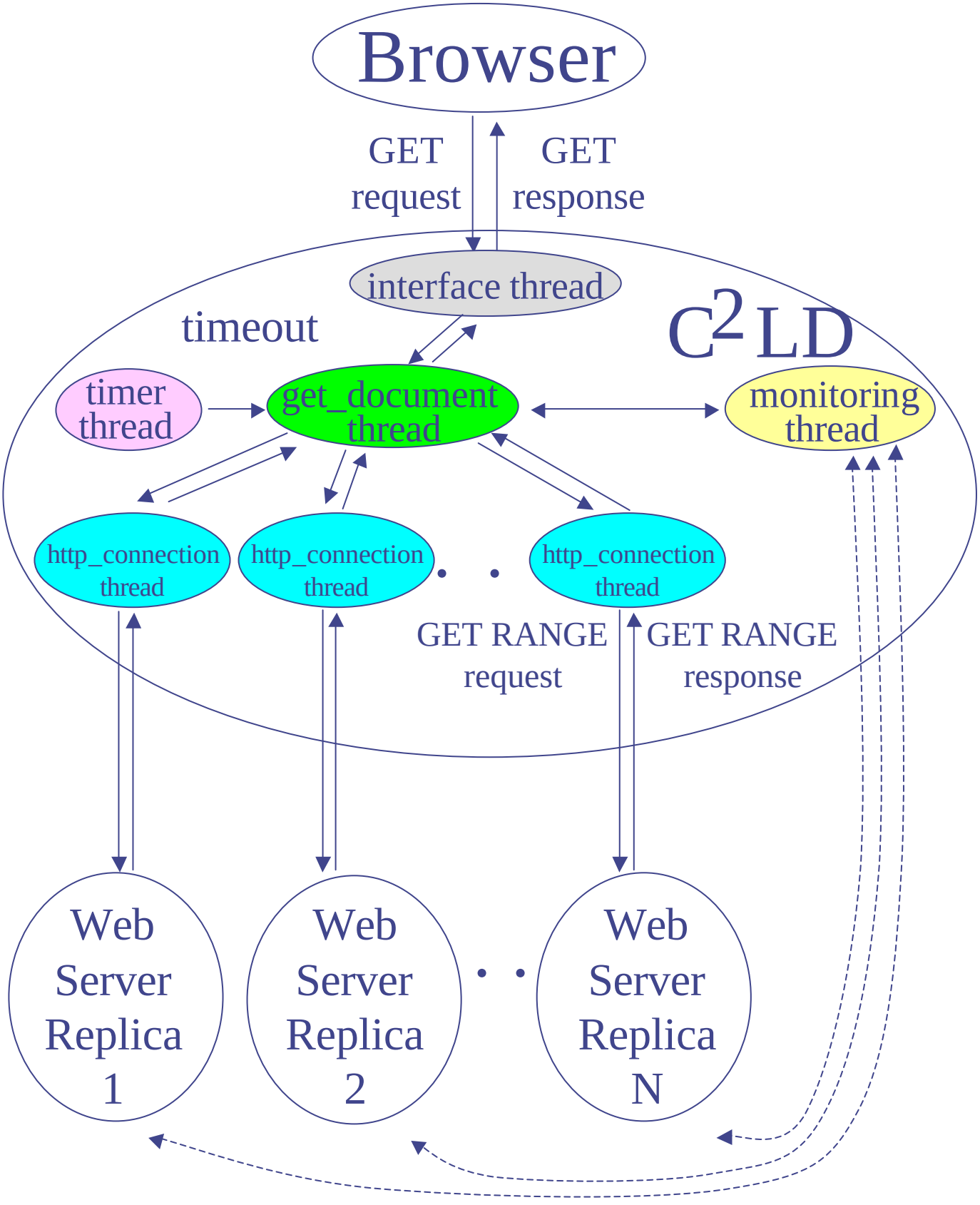


C^2LD Implementation

given the URL required by the browser, the C^2LD:

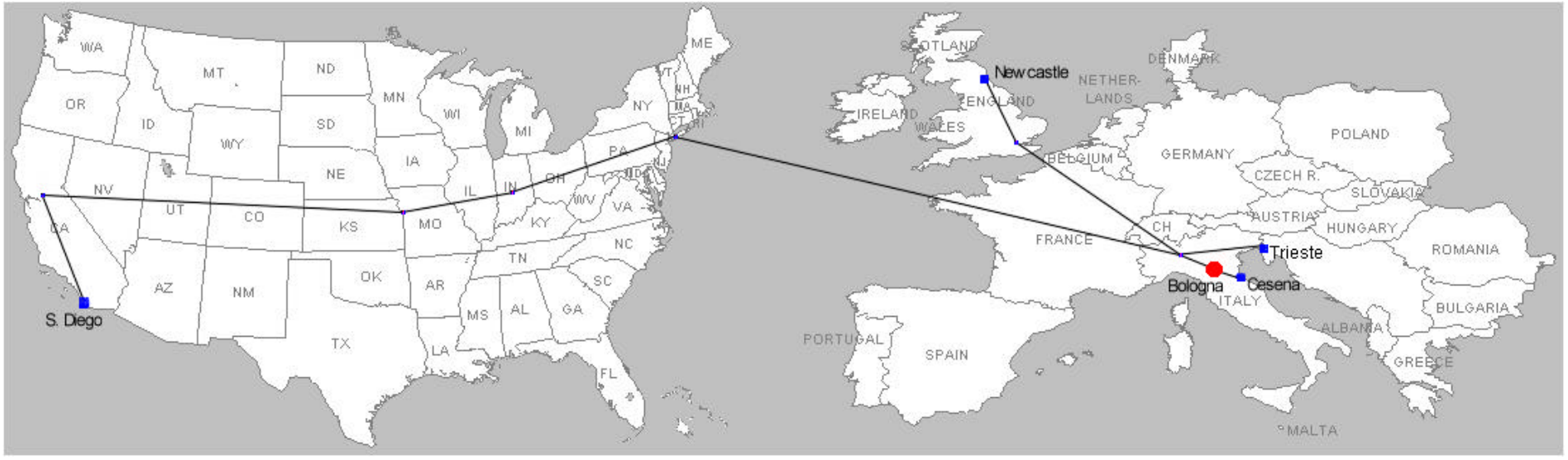
- asks the DNS for the IP addresses of all replicas
- sends HEAD request to each replica, asking for page size,
- starts a loop :
  - receives response from a given replica i
  - if page is not completely downloaded
    - $DR_{i,r} = PS_{i,r-1} / URT_{i,r-1}$ computes expected Data Rate
    - $PS_{i,r} = DR_{i,r} \cdot S_{i,r}^{*}$ computes fragment size for the next sub request to the replica i
    - sends GET sub requests to the replica i
  - else
    - return

C²LD Thread Structure



Experimental Measurements

- comparison between C^{2}LD and HTTP (standard GET method) downloading
- 4 replica server: Cesena, Trieste, Newcastle, S.Diego
- 1 client (Bologna)



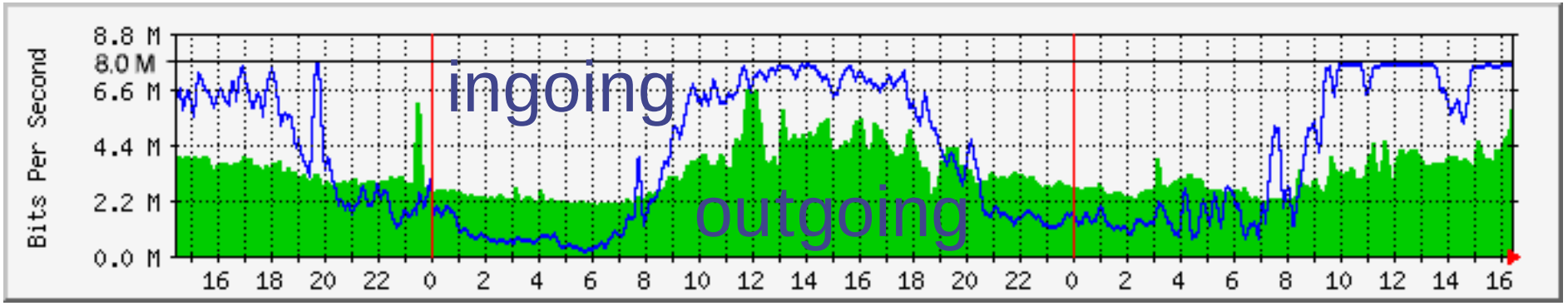
- 11 different Web services (two or more replicas)
- different file size (3 - 1000 Kbytes)

Network Traffic

- Lost Packet and RTT: high packet loss rate with Cesena, high delay with S.Diego

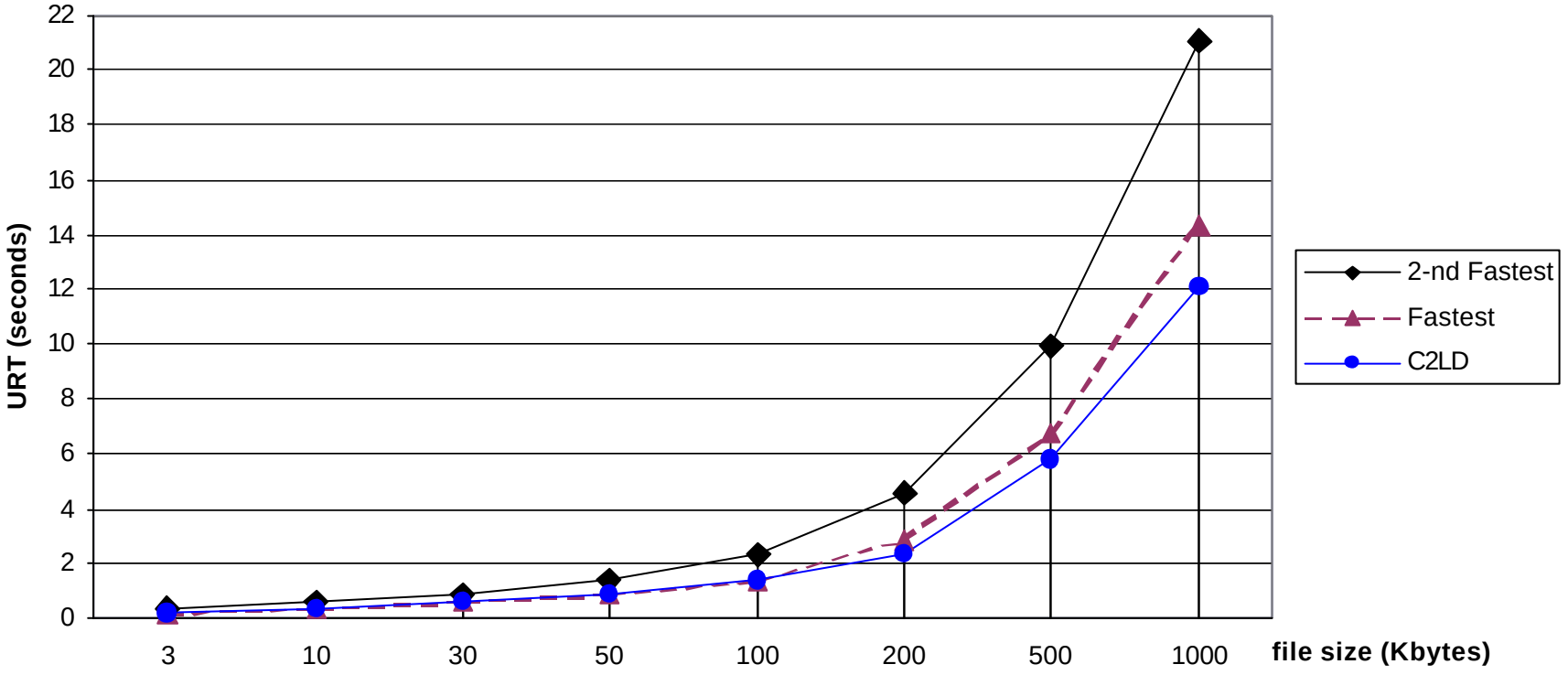
ping from Bologna to :	Cesena	Trieste	Newcastle	S. Diego
Lost Packet	2%-9%	0%	0%	0%-3%
RTT 90% of pkt (msec)	107	95	160	450
RTT minimum (msec)	10	38	59	190

- Bandwidth usage (Bologna's router): 8 Mbit/s ingoing bandwidth saturated during working hours



Measures: User Response Time

- C^2LD vs. HTTP and Fastest Replica



document size (Kbytes)	50	100	200	500	1000
URT improvement %	.01	5.0	17.3	13.9	15.6

- Average improvement 12.95% (file size>50 KBytes)

Measures: User Response Time

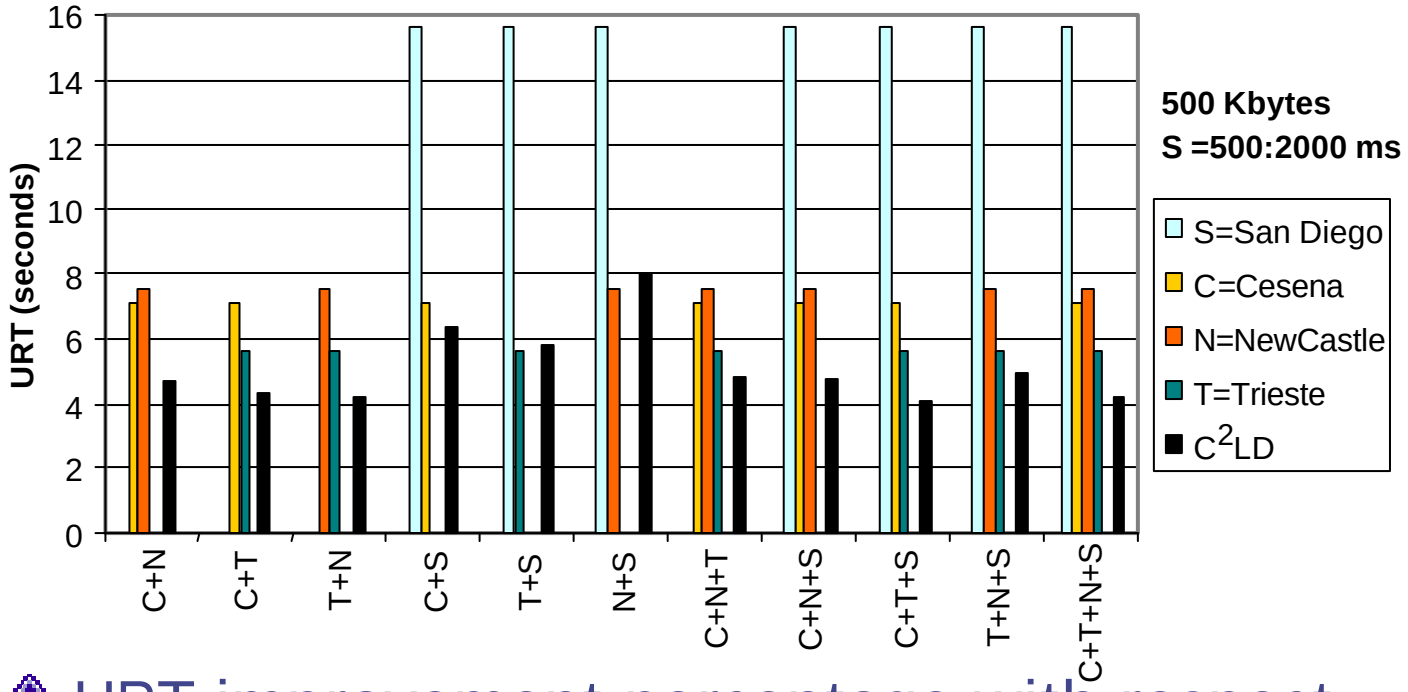
- URT depending on number of replicas

Number of replicas	URT improvement provided by C²LD with respect to the fastest replica and standard HTTP downloading
2	4%
3	17.2%
4	21.5%

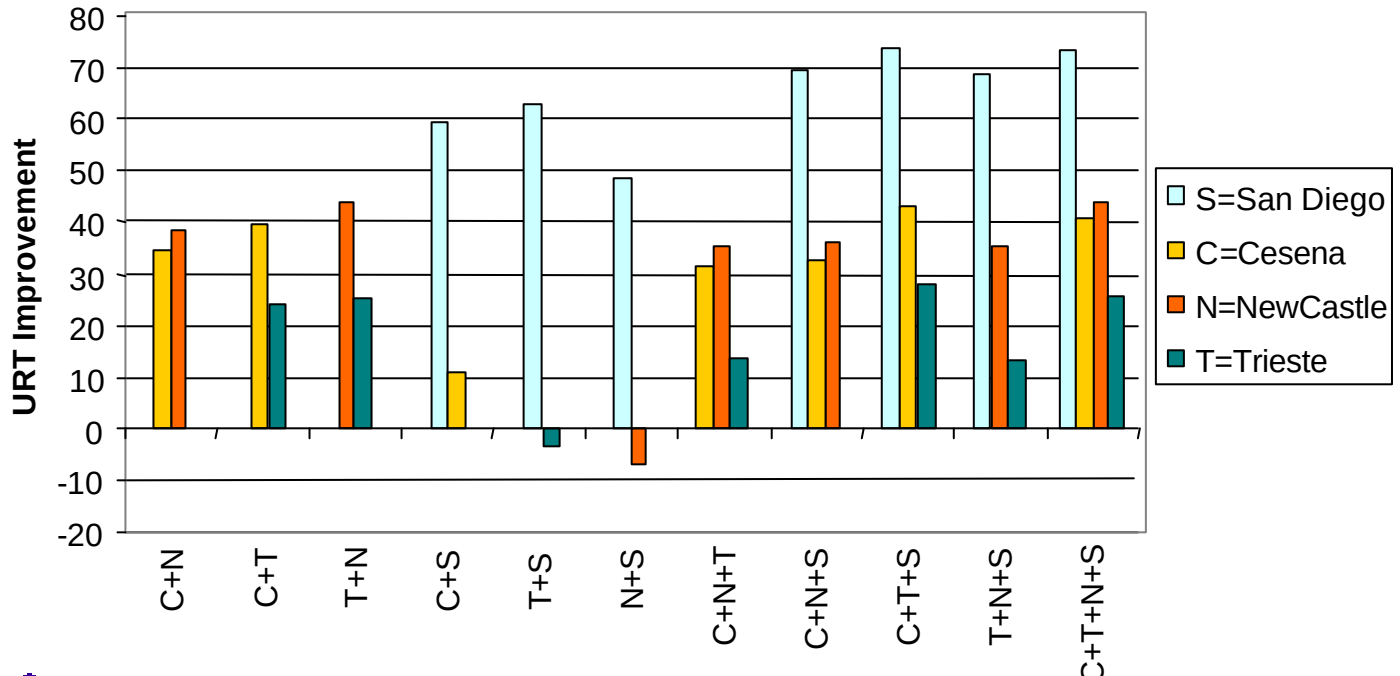
- URT decreases as the number of replicas of the Web service grows

C²LD vs. Single Replica

- URT comparison, C²LD vs. each replica



- URT improvement percentage with respect to each replica of a given Web service



- There is no gain when adding a very slow replica to the web service

Concluding Remarks

- C^2LD provides access to static documents of replicated Web servers (geographically distributed)
- assumes consistency of all replicas
- works on the browser side
- uses HTTP protocol (no modification required)
- outperforms the HTTP downloading mechanism:
  - availability (fault 0%)
  - timeliness (improvement 4% - 21% for large files)
- Future Works:
  - evaluation of Web servers workload
  - evaluation of network overhead
  - implementation within a proxy server
  - prefetching strategies, and consistency problem

References (1)

[1] R.B. Bunt, D.L. Eager, G.M. Oster, C.L. Williamson, “Achieving Load Balance and Effective Caching in Clustered Web Servers”, Proc. 4th International Web Caching Workshop, San Diego, CA, March 1999.

[2] “Cisco Local Director”, CISCO System Inc., White paper, 1996.

[3] M. Colajanni, P. S. Yu, D. M. Dias, “Analysis of Task Assignment Policies in Scalable Distributed Web-Server Systems”, IEEE Trans. on Parallel and Distributed Systems, Vol. 9, N 6, pp. 585-600, June 1998.

[4] M. Conti, E. Gregori, F. Panzieri, “Load Distribution among Replicated Web Servers: A QoS-based approach”, Proc. 2nd ACM Workshop on Internet Server Performance (WISP'99), Atlanta, GA, , May 1999.

[5] M. Conti, E. Gregori, F. Panzieri, “QoS-based Architectures for Geographically Replicated Web Servers”, Cluster Computing (to appear).

[6] P. Damani, P.E. Chung, Y.Huang, C.Kintala, Y. M. Wang, “ONE-IP: Techniques for Hosting a Service on a Cluster of Machines”, Comp. Net. and ISDN Sys., 29, 1997, pp. 1019-1027.

[7] D. Ingham, S.K. Shrivastava, F. Panzieri, “Constructing Dependable Web Services”, IEEE Internet Computing, Vol. 4, N. 1, January/February 2000, pp. 25 - 33.

[8] A. Iyengar, J. Challenger, D. Dias, P. Dantzig, “High-Performance Web Site Design Techniques”, IEEE Internet Computing, Vol. 4., N. 2, March/April 2000, pp. 17-26.

References (2)

[9] J. Li, H. Kameda, “Load Balancing Problems for Multiclass Jobs in Distributed/Parallel Computer Systems”, IEEE Trans. on Computers, Vol. 47, No. 3, March 1998, pp. 322-332.

[10] E.D. Katz, M. Butler, R. McGrath, “A Scalable HTTP Server: The NCSA Prototype”, Comp. Net. and ISDN Sys., 27 (2), pp.155-164, November 1994.

[11] J. Li, H. Kameda, “Load Balancing Problems for Multiclass Jobs in Distributed/Parallel Computer Systems”, IEEE Trans. on Computers, Vol. 47, No. 3, pp. 322-332, March 1998.

[12] V. Pai, G. Banga, M. Svendsen, P. Druschel, W. Zwaenepoel, E. Nahum, “Locality-aware Request Distribution in Cluster-based Network Servers”, Proc. *ASPLOS- VIII*, San Jose, CA, October 1998.

[13] J. Watts, S. Taylor, “A Practical Approach to Dynamic Load Balancing”, IEEE Trans. on Parallel and Distributed Systems, Vol. 9, NO. 3, March 1998, pp. 235-248.

[14] V. Ghini, F. Panzieri, M. Roccetti “Client-centered Load Distribution: A Mechanism for Constructing Responsive Web Services” Technical Report No. UBLCS-07, Laboratory for Computer Science, University of Bologna, June 2000.