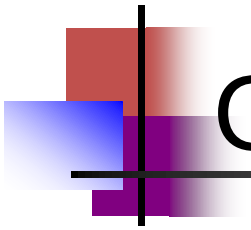
Design and Implementation of Load Reduction System for Mitigating Flash Crowds on Web Server

Kenji Yokota[1] **Takuya Asaka**[2]
Tatsuro Takahashi[1]
[1]Kyoto University, Japan
[2]Tokyo Metropolitan University, Japan



Outline

- Background
- Existing approach
- Proposed system
- Experiments
- Conclusion and future work

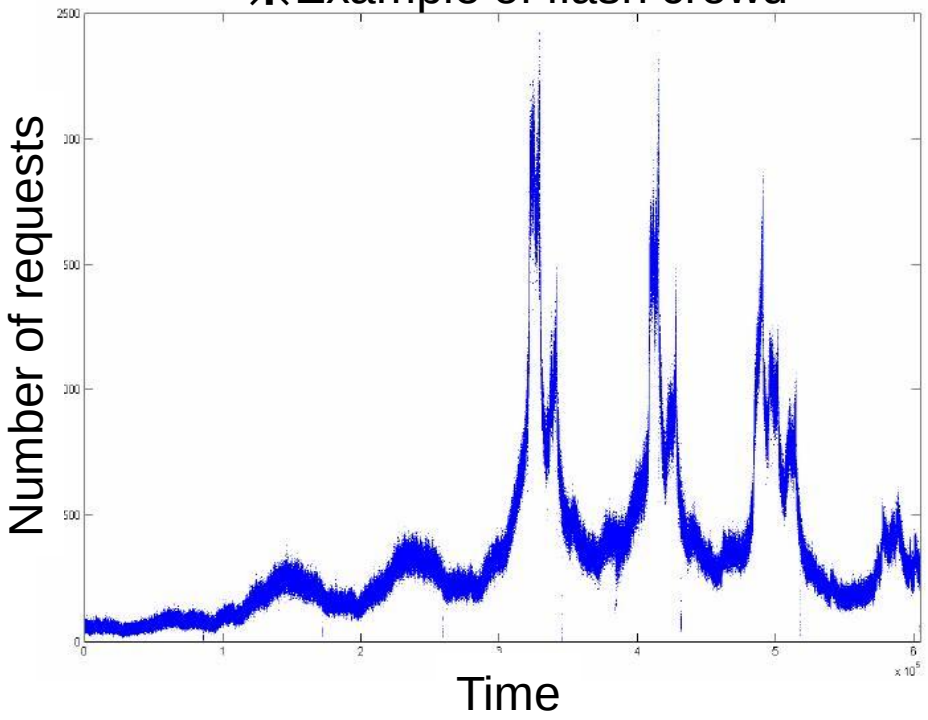
Background

～Flash crowd～

Sharp increase of traffic at a given web server within a short time

- Response rate decreases or a web server may crash as the load increases
- Excessive traffic decrease throughput

※Example of flash crowd



Our goal

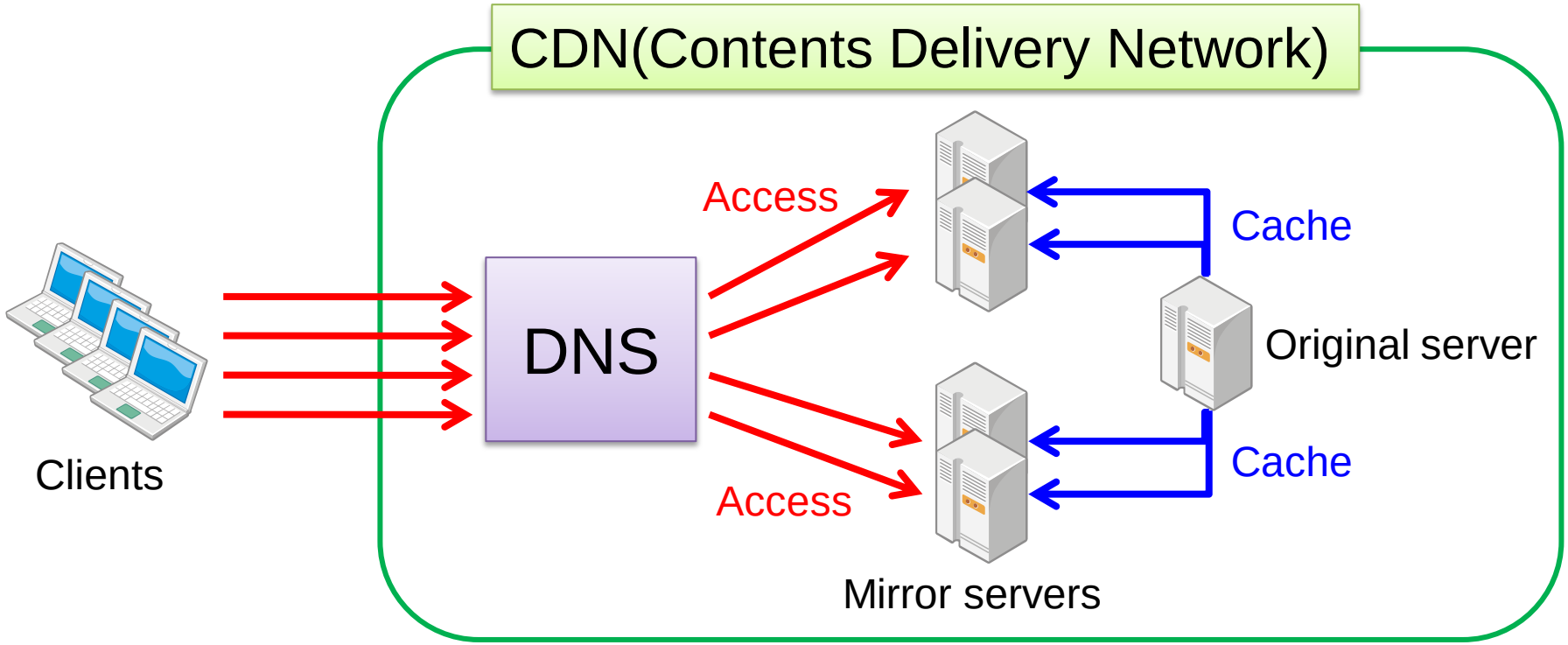
- Provide content stably even when flash crowds occur
- Reduce the load without enhancing the performance of the web server by using cooperation between Internet users

※A workload characterization study of the 1998 World Cup Web site (IEEE Network 2000)

Existing approach 1

Server mirroring

Place a number of mirror servers on the Internet and distributes accesses to the target web server over mirror servers

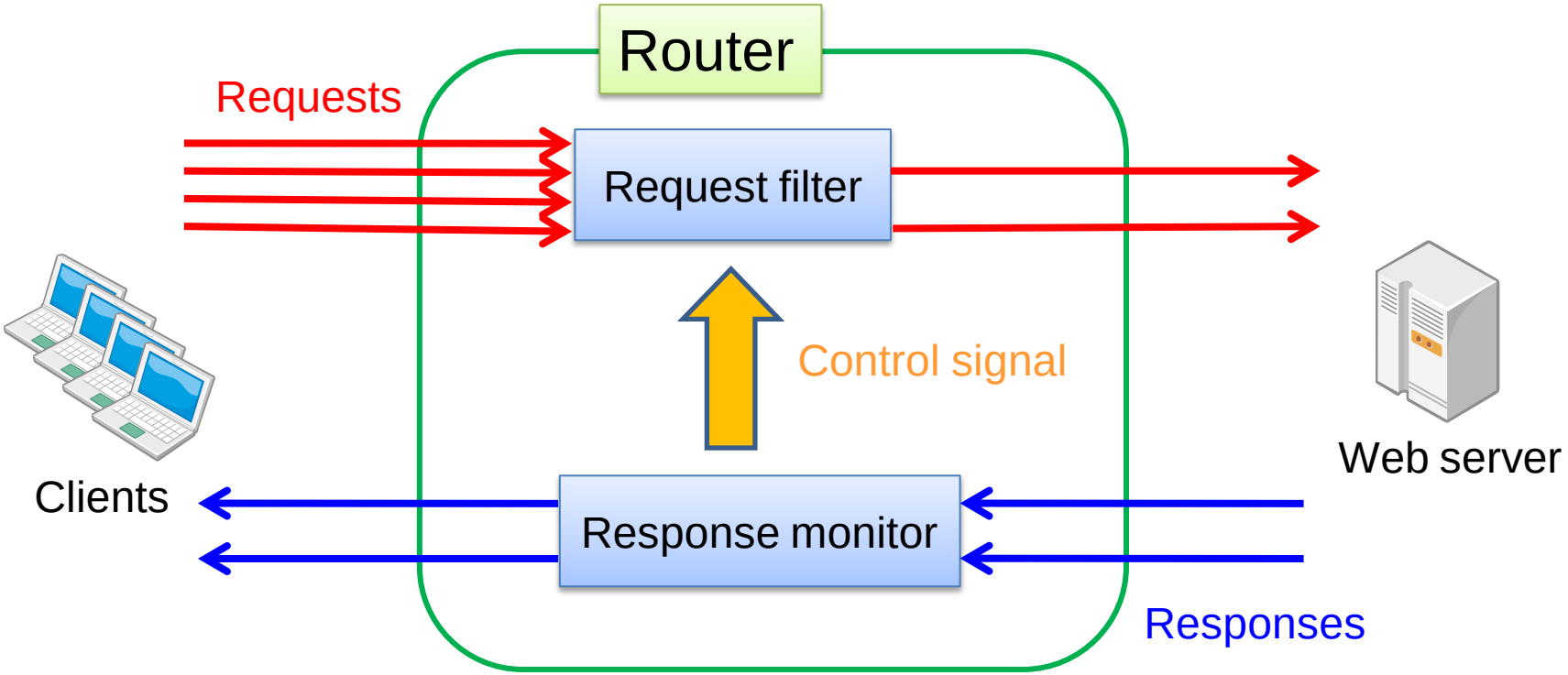


Difficult to know how many mirror servers are needed for flash crowd

Existing approach 2

Admission control

Detects flash crowds and controls the admitted request rate for preventing excessive accesses to the web server

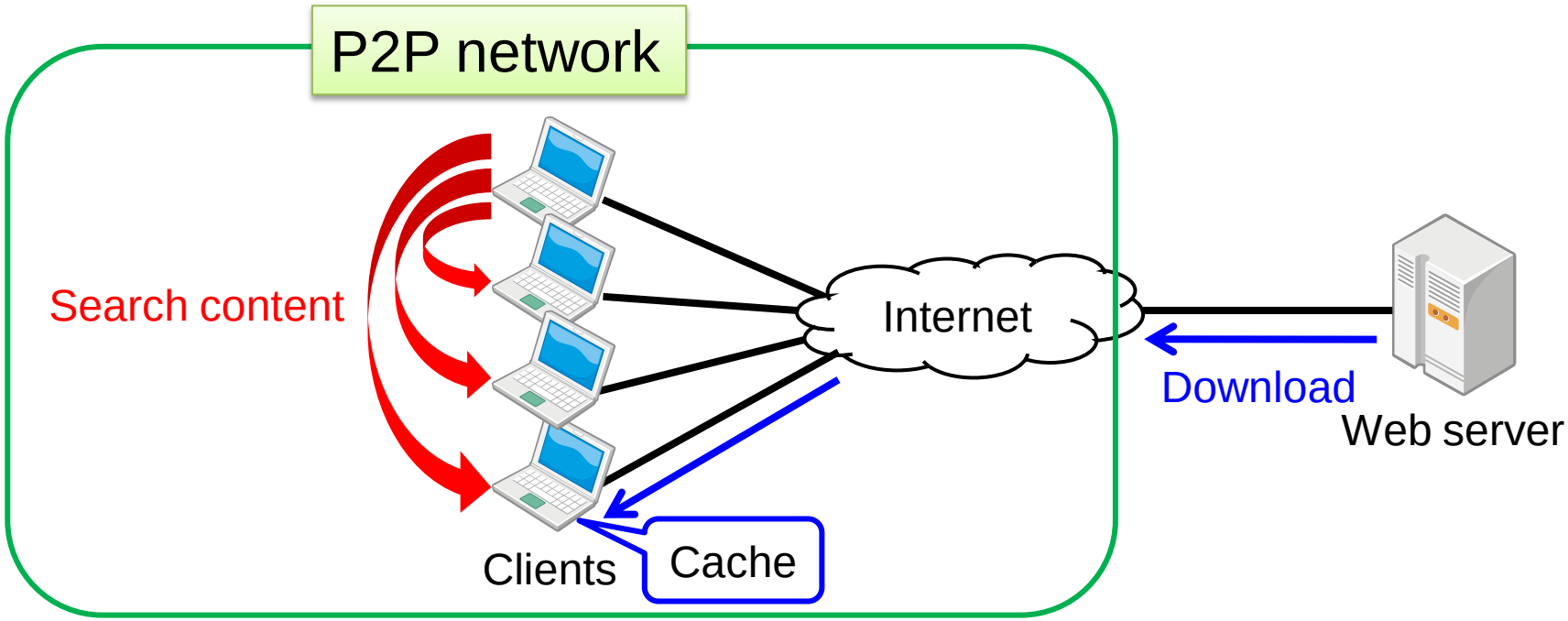


Cannot enjoy a satisfactory level of service when flash crowds occur

Existing approach 3

Peer-to-peer (P2P)

Searches and downloads content on a P2P network without accessing the web server

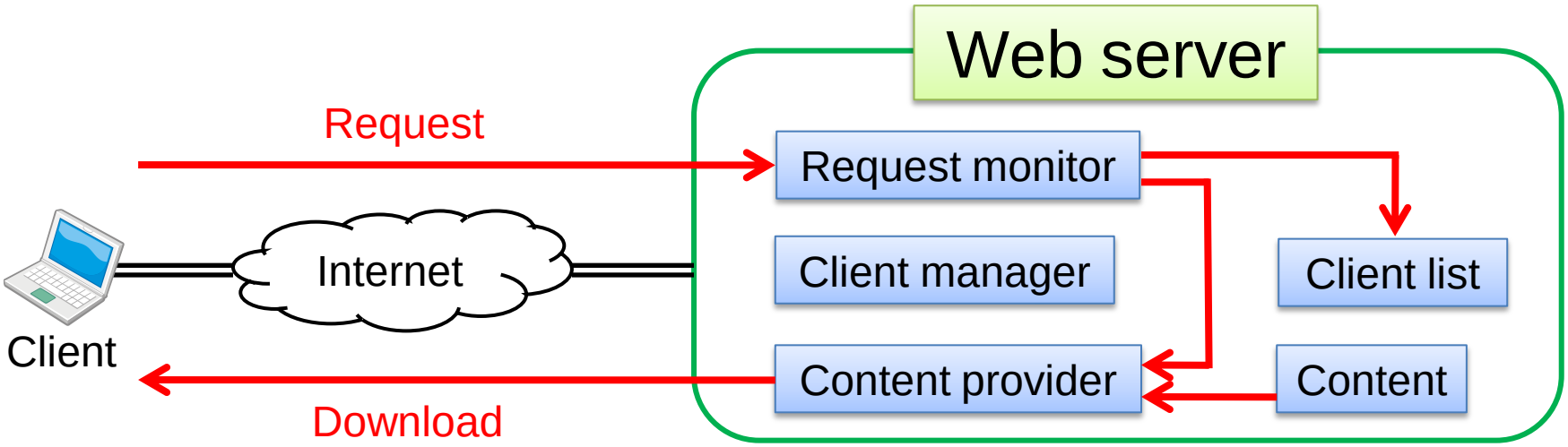


Doesn't know who has the desired content, and the overhead of message to construct the P2P system is large

Proposed system

Web server detects flash crowds by monitoring the number of requests and changes the content delivery system when flash crowds occur

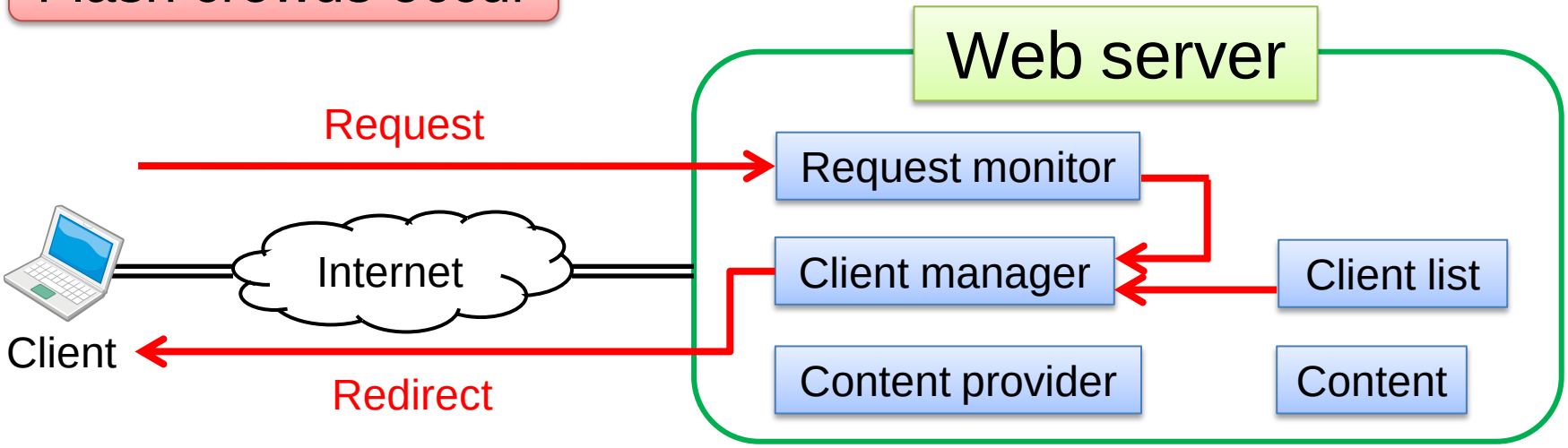
No flash crowds



- Client downloads content from the web server
- Web server stores client's IP address in the client list

Proposed system

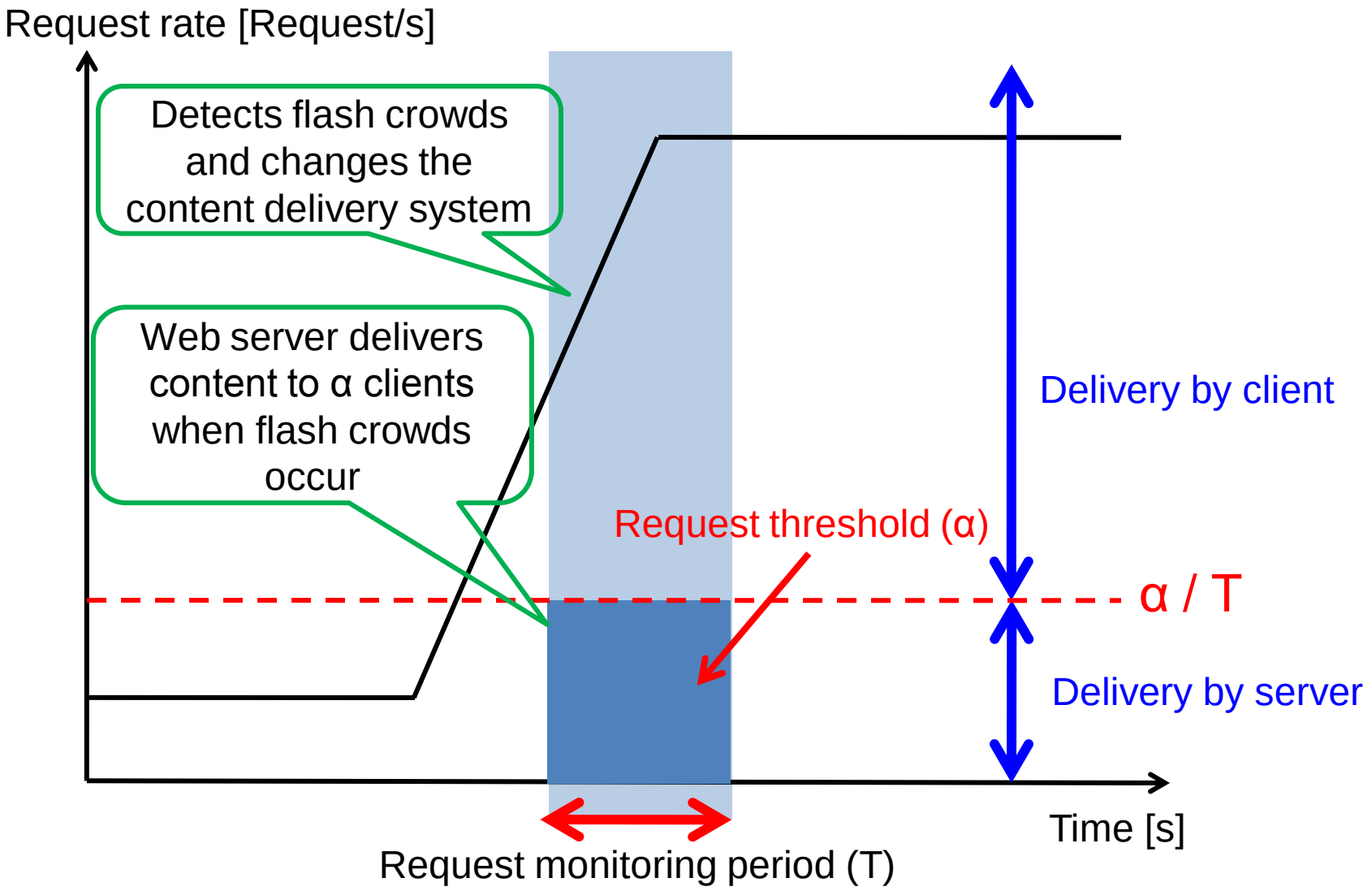
Flash crowds occur



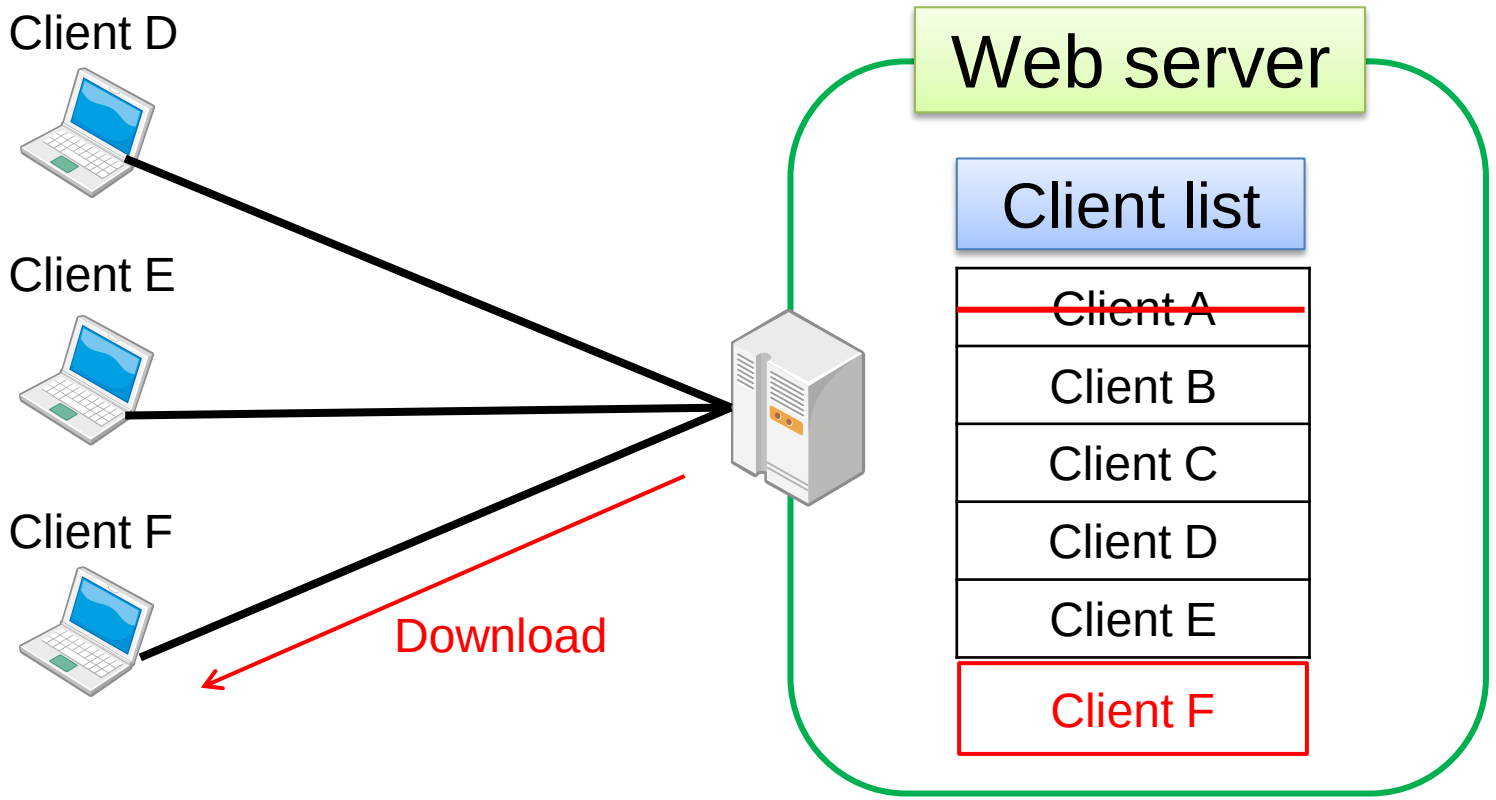
- Web server doesn't provide content to all clients
- Sends a redirect message in order to change the destination address
- Client sends a request to another client that holds content in the cache

Load on the web server is reduced when flash crowds occur because it only redirects requests and it doesn't provide content to all clients

Monitoring of requests on web server

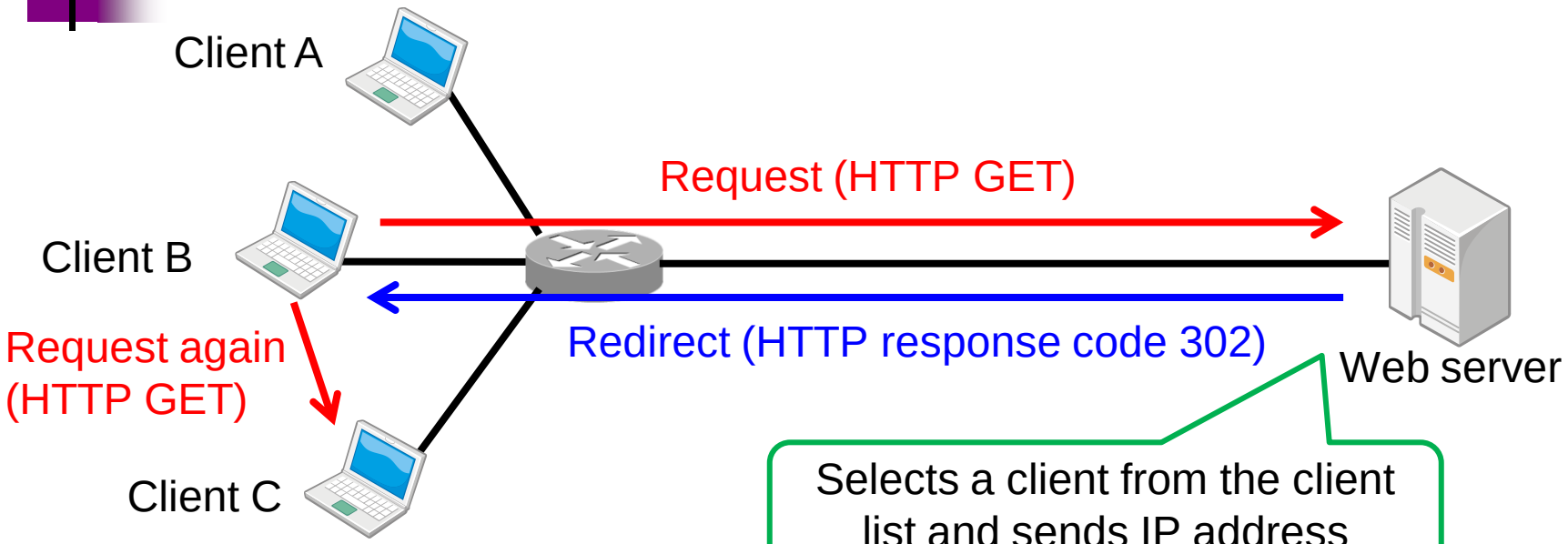


Behavior of client list



- First-In First-Out (FIFO) cache: The web server deletes the oldest client on the client list if a new client downloads content
- When T is 1 s, α is 20 requests, and the capacity of the client list is 100, a client remains on the client list for 5 s.
- A client receives a request from other clients within 5 s.

Behavior of proposed system



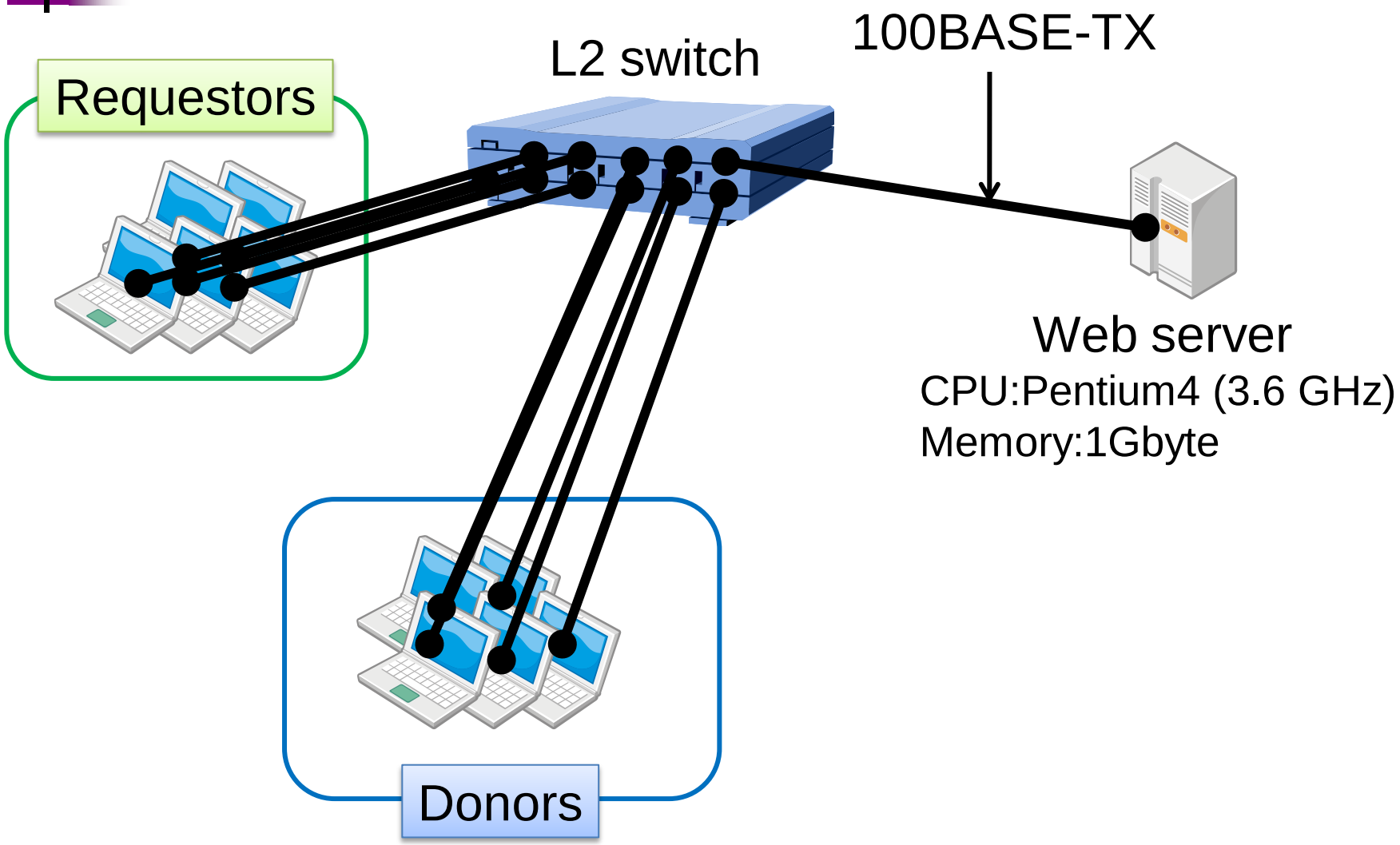
Web browser

- Caches the web page which the user views
- Automatically transfers the cache by the demand of another client

Advantage: Doesn't need to estimate the size of flash crowds and can handle various sizes of flash crowds

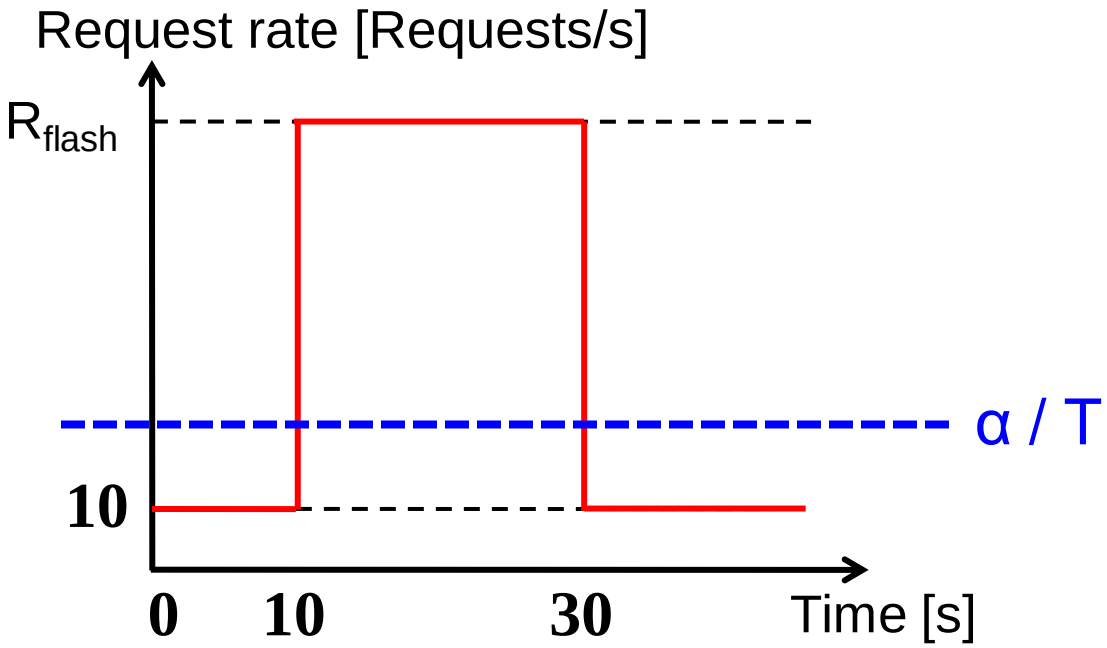
Disadvantage: Users have to change their web browsers, but the proposed system can work by add on for existing web browsers

Network composition in experiments

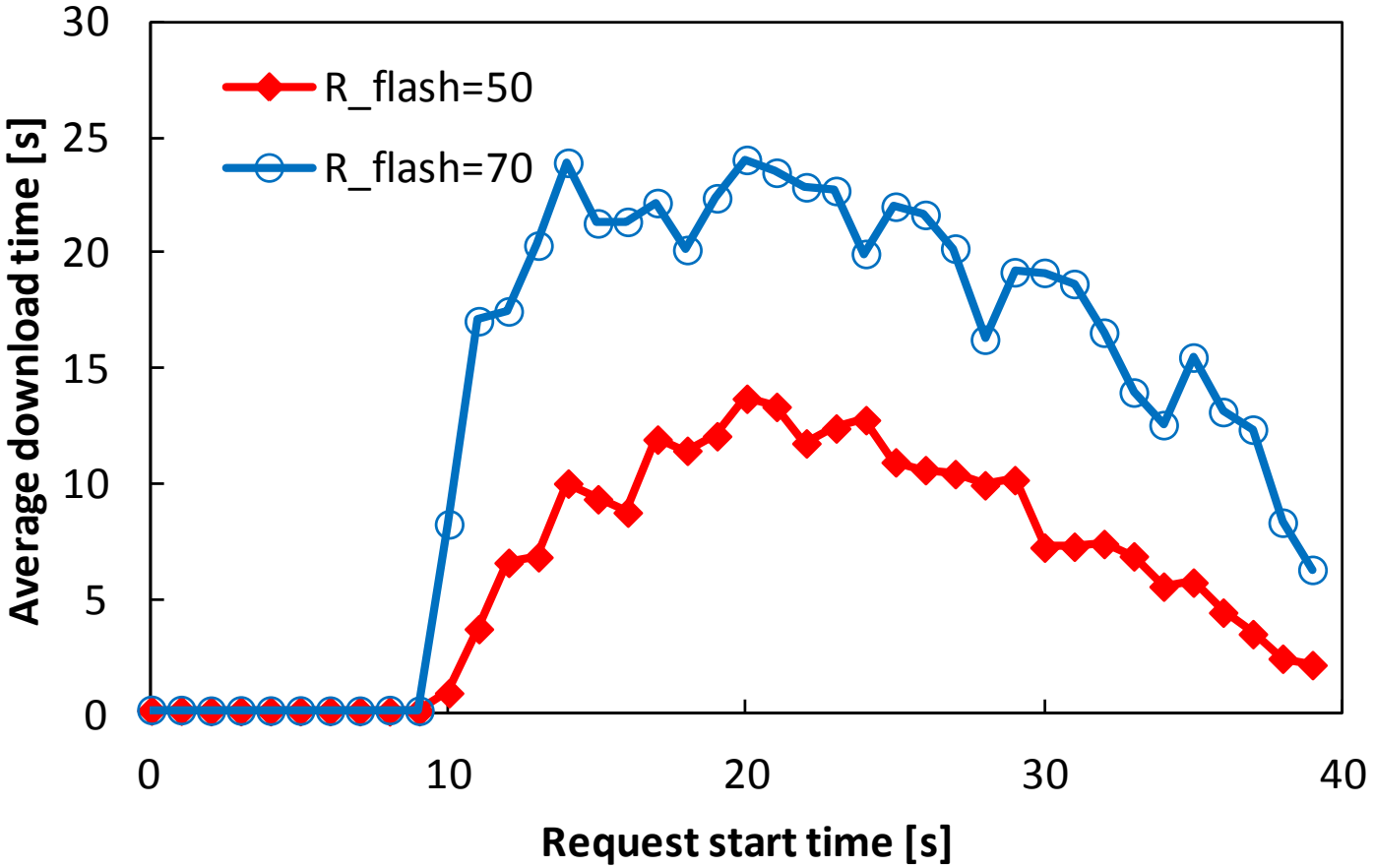


Experimental parameters

Parameters	Values
Rate of flash traffic (R_{flash})	50 or 70 [Requests/s]
Size of web page	230 [kbytes]
Request monitoring period (T)	1 [s]
Request threshold (α)	20 [Requests]

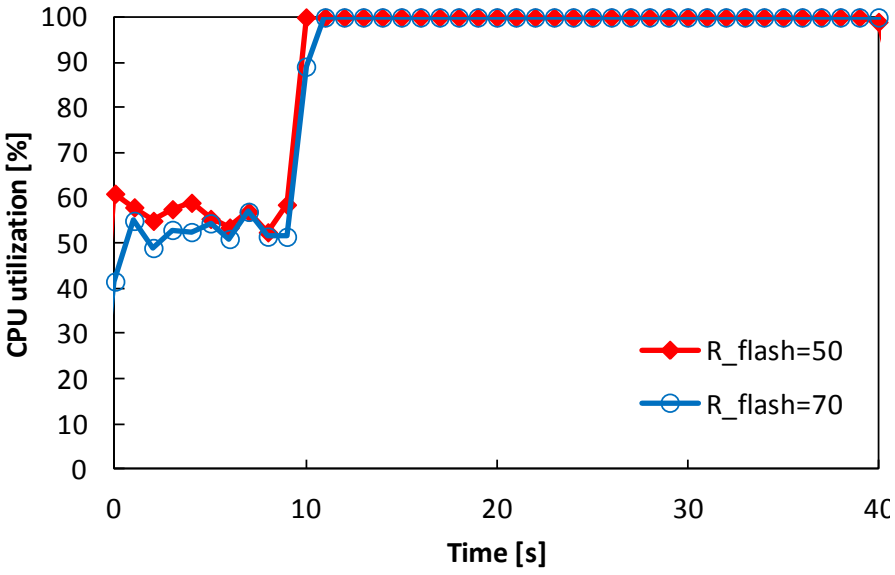


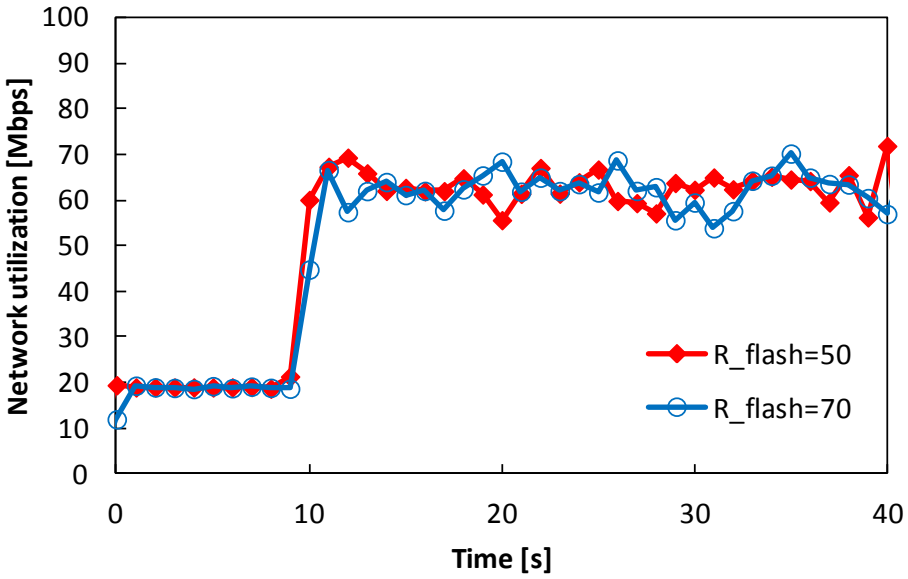
Download time (Conventional system)



- Average download time is short in normal conditions, but it increases greatly when flash crowds occur
- Average download time doubles when the request rate increases 50 to 70

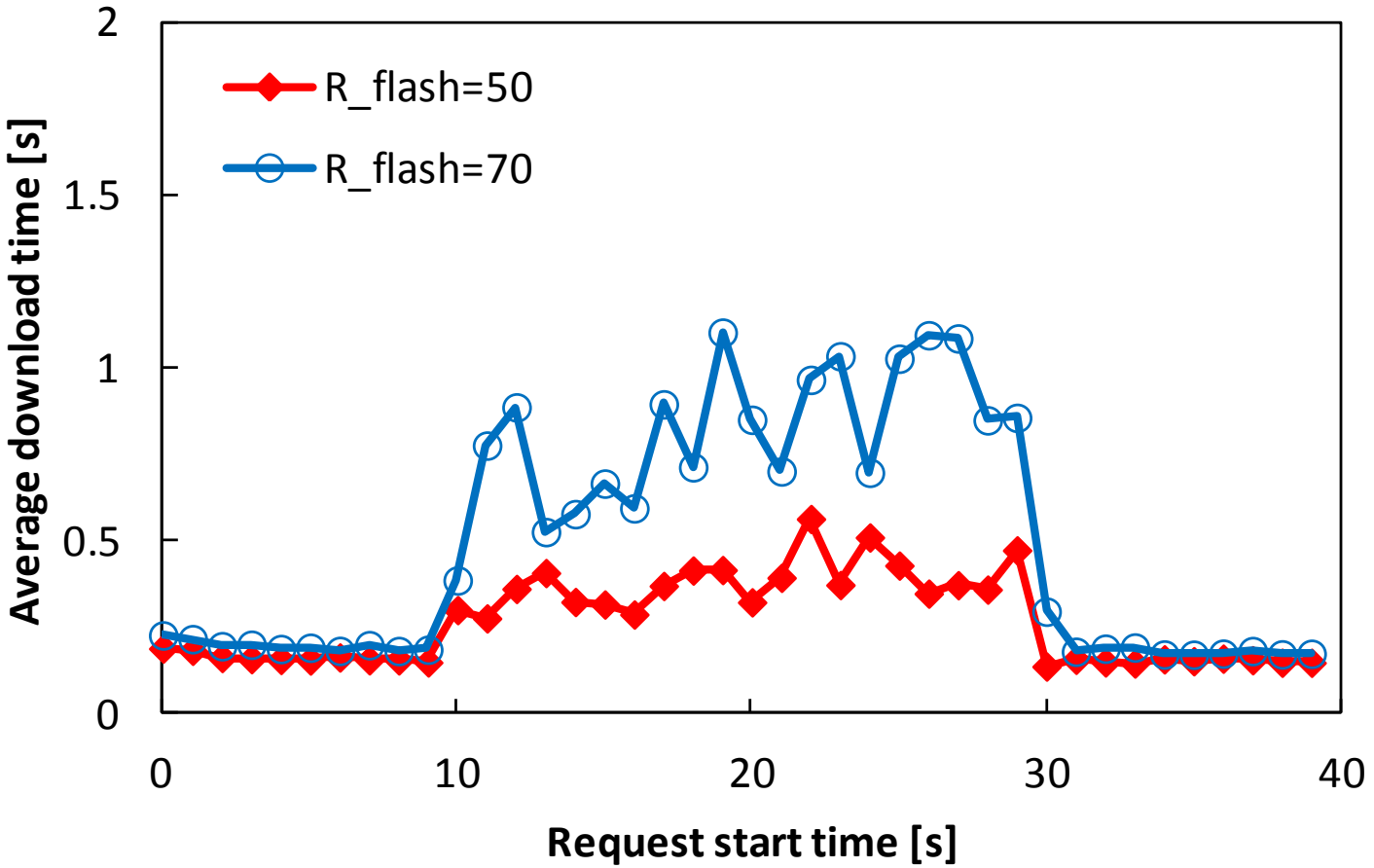
Server load (Conventional system)





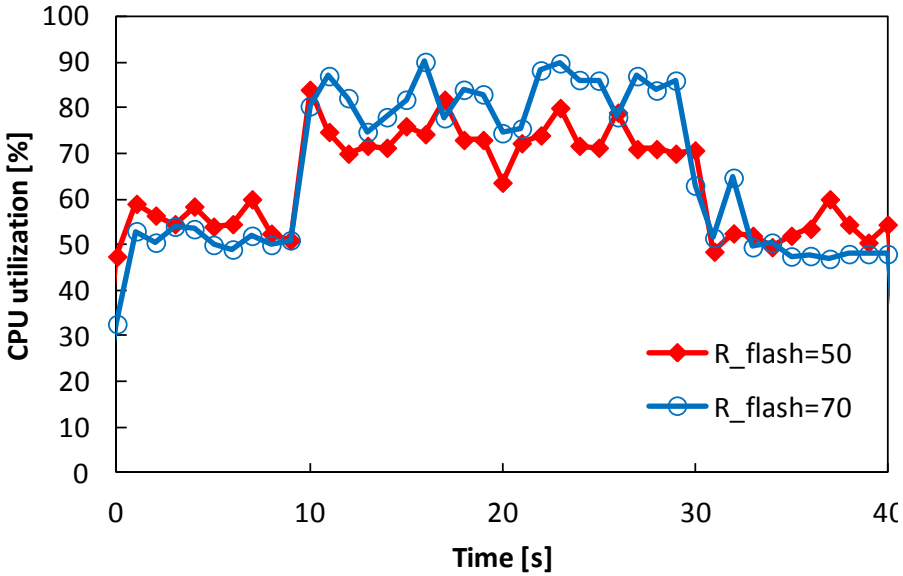
- CPU utilization increases to 100% when flash crowds occur, and it's still high after 30 s
- Network utilization isn't 100 Mbps
- CPU bottleneck causes the increase in the average download time

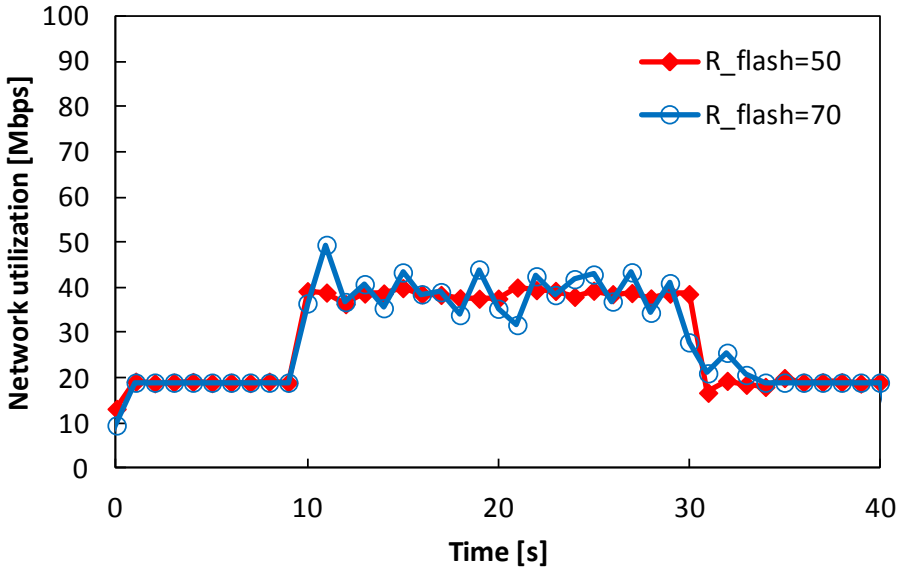
Download time (Proposed system)



- Average download time is significantly reduced with the proposed system when the request rate is both 50 and 70

Server load (Proposed system)





- CPU utilization isn't 100%, and the bottleneck is removed with the proposed system
- CPU utilization increases 10% when the request rate is 70 compared to the case when it's 50 because the web server needs to redirect 20 extra requests/s

Conclusion and future work

- Proposed system can reduce the load on the web server by redirecting download requests to other clients
- Implemented application software of the web server and the web browser to evaluate the efficiency
- Results from experiments prove that the proposed system can reduce the download time when flash crowds occur

Future work

- Evaluate under various network conditions