HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Internet Content Distribution

Chapter 2: Server-Side Techniques

Jussi Kangasharju

Chapter Outline

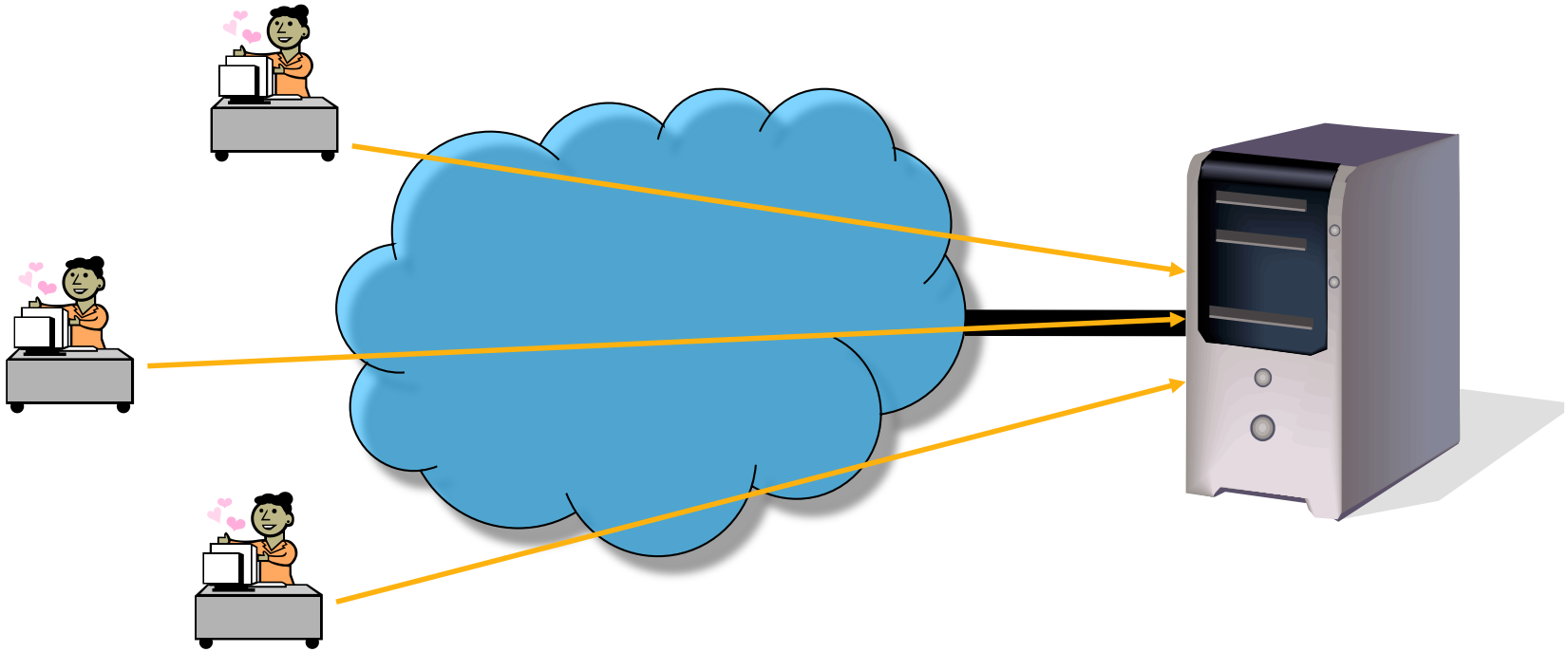
- Server-side techniques for content distribution
- Goals
- Mirrors
- Server farms
  - Surrogates
  - DNS load balancing
- Parallel downloading

Why Server-Side Techniques?

- Server-side techniques are aimed at helping the content provider to lower her costs
- Costs can be:
  - Costs of running a server
  - Costs of a network connection
- Typically, it is easy to upgrade network connection
  - Easy = it only takes money
- Upgrading servers is feasible only up to a point
  - Processors do not have infinite speeds
  - Not possible to put enough memory to handle thousands of simultaneous clients

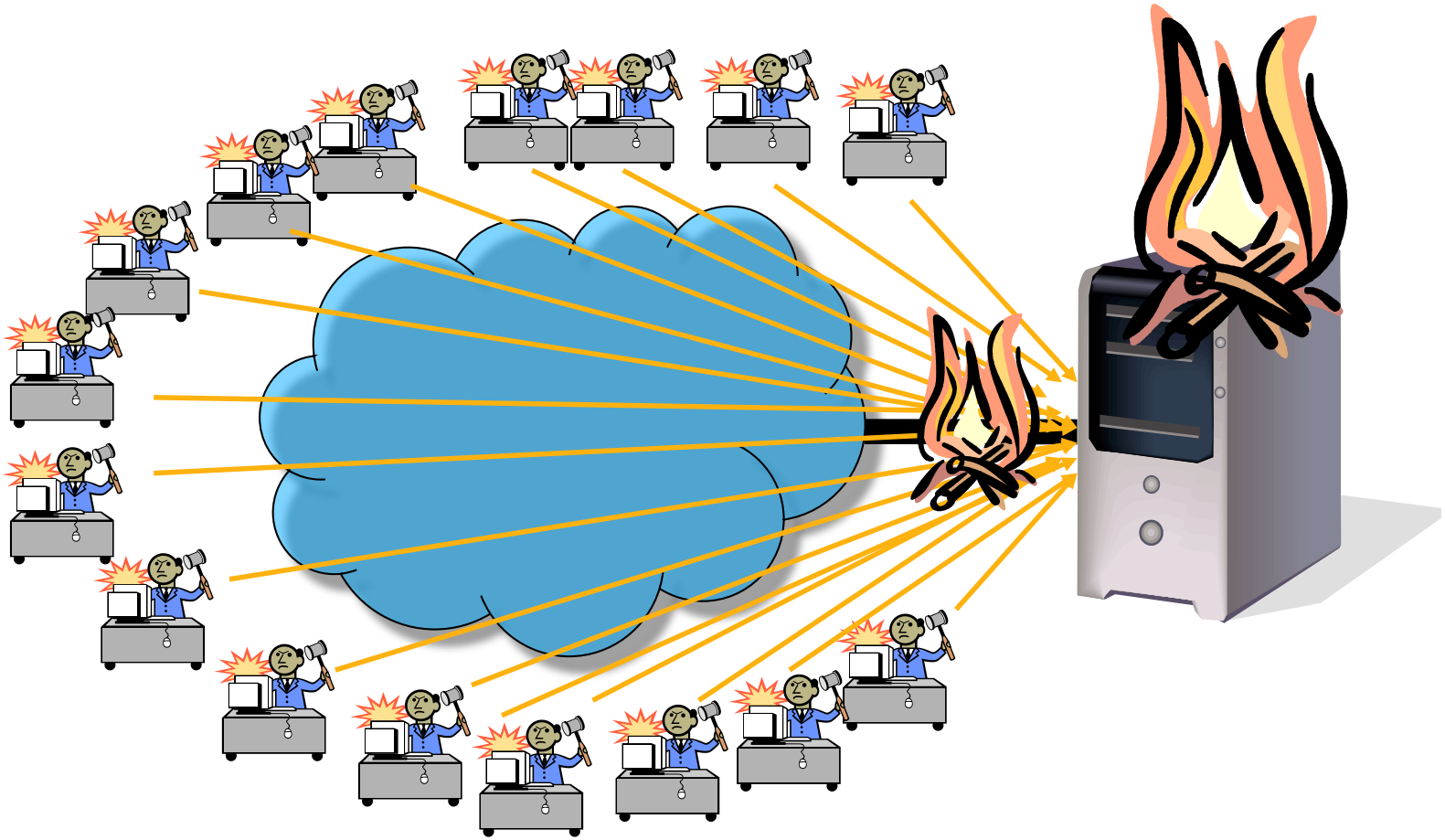
Problems on the Server-Side

- What happens when we do not have enough capacity?
- With a small number of users all works well



Problems on the Server-Side

- Problems start when we have lots of users



Problem in Short

- Problem is that we cannot handle the traffic
- Two main aspects

1. Not enough server capacity
2. Not enough network capacity

- Both can be alleviated (or solved) with money
- Buying enough network bandwidth is possible, but extremely expensive
  - Possible to buy tens of Gbps (in theory at least)
- However, a single server has a maximum capacity
  - One CPU is only “so fast”
  - Can only add X GB of memory (limited by hardware/OS)
  - Network cards go only up to certain speed
- First bottleneck is the server

Solution

- If one server cannot handle all the traffic, we'll install several servers
- Total capacity is the sum of the individual capacities
- Such an arrangement is called server farm
- Questions:

1. How many servers do I install?
2. Where do I install them?
3. How do I get the users to use those servers?

- We focus more on questions 2 and 3
- Answer to question 1 is more a business decision

Server Farms

- Typically server farms are hosted in a single data center
- This means all the servers share the same network connection to the Internet
  - Means: Must still spend lots of money on that
- Advantages:
  - Easier to manage, since all servers are in the same place
  - Increased service capacity
- Disadvantages:
  - Still need big pipe to Internet
  - If the network path from the user to the data center is the problem, then the user will not see many benefits
- How about distributing the server farm?

Mirror Servers

- We can take servers from a server farm and install them in different geographical locations
- Traditionally this has been called mirroring
- Mirror servers are an old technology
  - Already used for FTP servers in 1980's
  - Still in popular use, especially for open software downloads
    - For example, SourceForge
- Idea behind a mirror server is to copy the content from the origin server and offer it on a different server
- Users access the content from the "different server"
  - For example because it's closer to them (or cheaper)
  - In the old days main goal of mirroring was to reduce international bandwidth costs (e.g., ftp.funet.fi)

Mirror Servers

- Advantages:
  - Easy to collect lots of data, one mirror can mirror several origin servers
  - Can be installed close to users
  - Teaches users about networking (hopefully :-)
- Disadvantages:
  - Users must use mirrors for us to get any benefits
  - Typically no automatic mirror selection
  - Content on mirror might be out-of-date
- Biggest problem with mirrors:
  How to get users to use them?
- Existing solutions:
  - Manual selection from a list
  - Automatic selection
  - Parallel download from several mirrors (also used in P2P networks)

Manual Selection of a Mirror

- Manual selection means that the user has to select the mirror somehow manually
  - Type a different URL, pick mirror from list, click on an extra link…
- List of mirrors must somehow be available
  - These days typically on a website
- User picks mirror and uses it
  - Typically you have to choose it every time you download
  - Automatic selection of mirror by server becoming common
- Sufficient procedure if:
  1. Users understand what they are doing
  2. Selection does not happen too often
- Otherwise too confusing or annoying

Automatic Selection of a Mirror

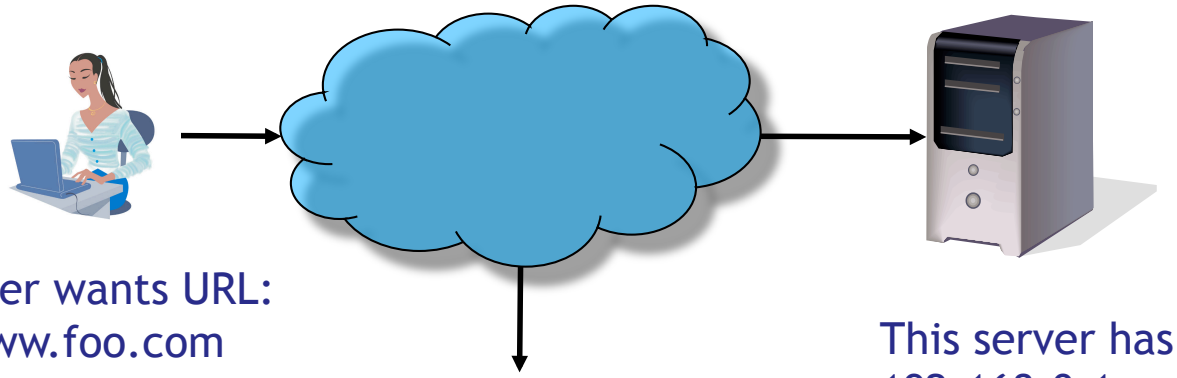
- Two main techniques currently in use
- Note: They are currently used for co-located server farms, not so much for real mirrors
  - But both techniques would work for geographically distributed mirror servers

1. Surrogate servers
2. DNS load balancing

- Main goal and current use of both is to balance load on a server farm
- Only real difference is that DNS load balancing is visible to clients, surrogates are not (necessarily)

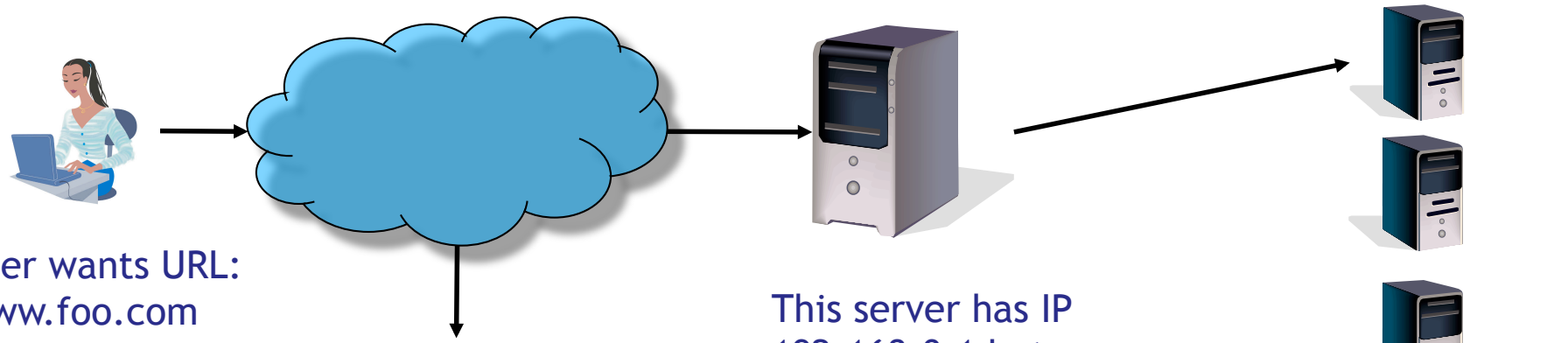
Surrogate Servers

- Surrogates sometimes also called server-side proxies
  - Dictionary definition of surrogate explains where the name comes from
- Traditionally web sites work as follows:



Surrogate Servers

- Surrogate is put in front of the server farm and receives all client requests
- Surrogate decides to which content server to forward the request
- Content server processes the requests and sends reply to surrogate
- Client receives reply from surrogate



Surrogates: Pros and Cons

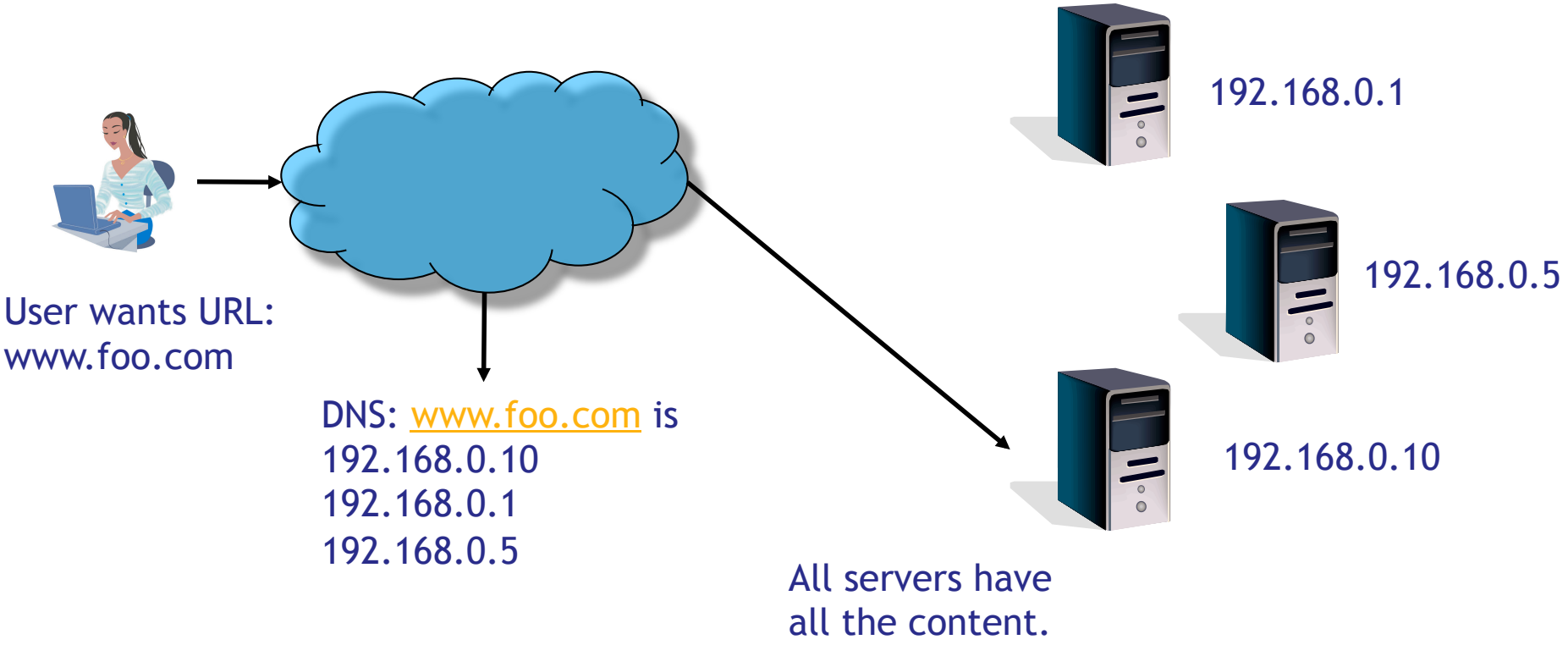
- Advantages of surrogates:
  - Totally invisible to client, no need to modify clients
  - Allows for fine grained load balancing because surrogate sees actual HTTP requests
    - Note: Not used in practice, but theoretically possible
    - Also, see below about L4 switches
  - Can build a cache into surrogate --> Less load on content servers
- Disadvantages of surrogates:
  - Surrogate can become performance bottleneck since all requests must go through the surrogate
    - Even if an L4 switch is used, processing is more complicated than in a normal router
  - Extra hardware to buy and maintain

Surrogates: Practical Details

- Surrogate can be implemented with a web proxy or with an L4 switch
- Web proxy:
  - Real web proxy, has to parse HTTP request
  - Can easily become a bottleneck, since HTTP processing is not “cheap” (compared to layer 3 or 4 processing)
- L4 switch:
  - L4 stands for Level 4 of the OSI model, i.e., transport
  - Simply a redirector based on the port number in TCP header
  - Much more common on client side
- Summary: Surrogates not widely used in practice

DNS Load Balancing

- DNS load balancing uses DNS to send clients to different content servers
- Reply to DNS query for server name results in several IP addresses
- Client picks one of them and sends request to that server



DNS Load Balancing Details

- Basic idea: Redirect each client to a different content server by giving different DNS answers
  - Same idea as DNS redirection (Chapter 4), but goals different
- DNS server of content provider decides which server handles the clients request
- Typically some kind of round-robin algorithm
  - But any kind of complicated load balancing is possible
- Clients typically receive a list of several IP addresses for the given hostname
- Client can choose any of the received addresses, but most current DNS client implementations pick the first
- Allow only short caching times for replies
  - Clients must refresh DNS lookups --> Adapt load balancing

DNS Load Balancing: Pros and Cons

- Advantages:
  - Easy to implement, DNS lookups are mandatory anyway
  - No additional hardware needed
  - Can in principle use any load balancing algorithm
- Disadvantages:
  - Client can keep on using the “wrong” server
    - Unlikely to happen, though, since this is controlled by OS, not user
  - No fine-grained control over load balancing
    - Granularity: This client goes to that server for X amount of time
    - Note: Client = Any browser behind same DNS server!
    - Not so much a problem for server-side load balancing, but a bigger issue for client DNS redirection (Chapter 4)

Comparison

Surrogates

- Allows for fine-grained load balancing
  - Even per request!
- Typically must process up to application level
  - Large effort
- Not widely used

DNS load balancing

- Extremely widely used by all major websites
  - Currently trend is to use CDN
  - CDNs use kind of DNS load balancing
- Not much additional processing needed on top of DNS request processing
- Relatively coarse-grained
  - But not much of a problem in practice (statistics!)

Parallel Downloads

- Let's get back to mirrors
- DNS load balancing could be used to select mirrors
- Other alternative was manual selection
- Question: Why select at all?
- Or rather, why not select them all?
- Motivation behind parallel downloads is to eliminate the need for mirror selection
- Main benefit is increased download speed
- Results in the following from Rodriguez & Biersack, *Dynamic Parallel-Access to Replicated Content in the Internet*, IEEE/ACM Transactions on Networking, Aug. 2002

What Are Parallel Downloads?

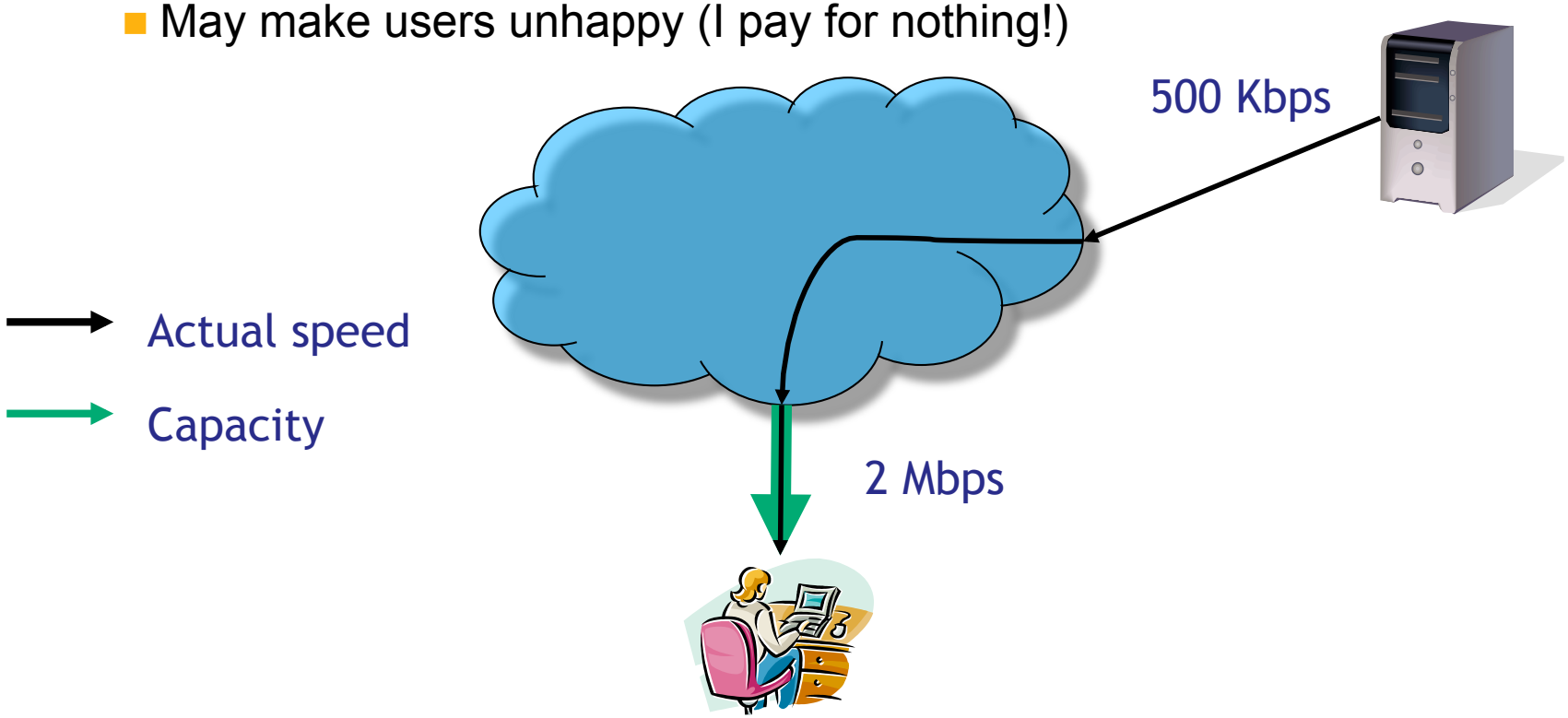
- Client downloads different parts of the file from different sources at the same time
- Not used for web content
- Widely used in P2P file sharing networks
  - All modern file sharing networks use parallel downloads
- Two assumptions for efficiency:

1. File to be downloaded is relatively large
   - Several hundred KB and larger
2. Paths from client to the sources are bottleneck-disjoint
   - See below

- First assumption makes parallel downloads unsuitable for web content

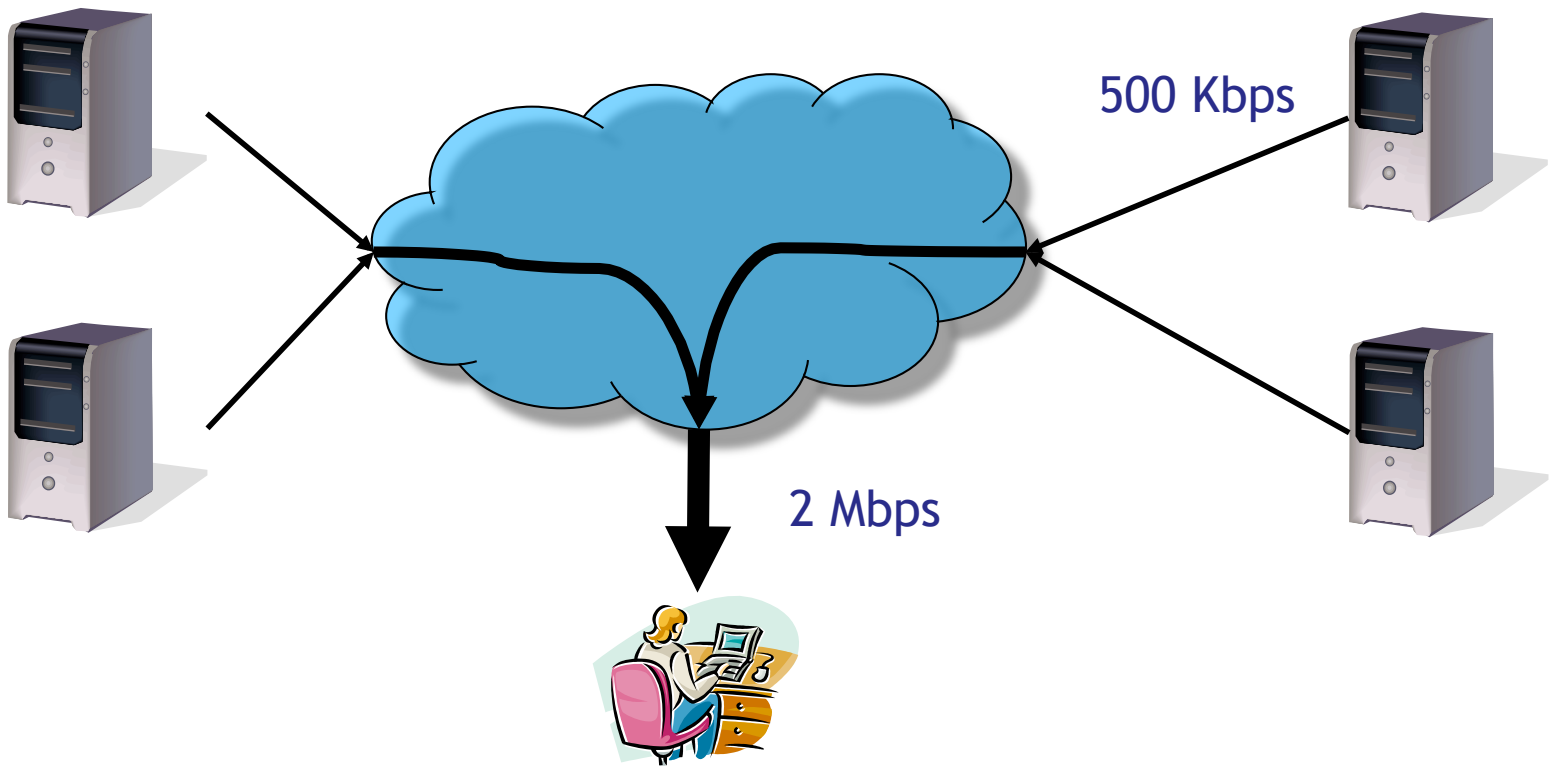
How Does Parallel Download Work?

- Downloading from a single server, user is limited by that server's upload bandwidth
- In the case below, user cannot use her full bandwidth
  - May make users unhappy (I pay for nothing!)



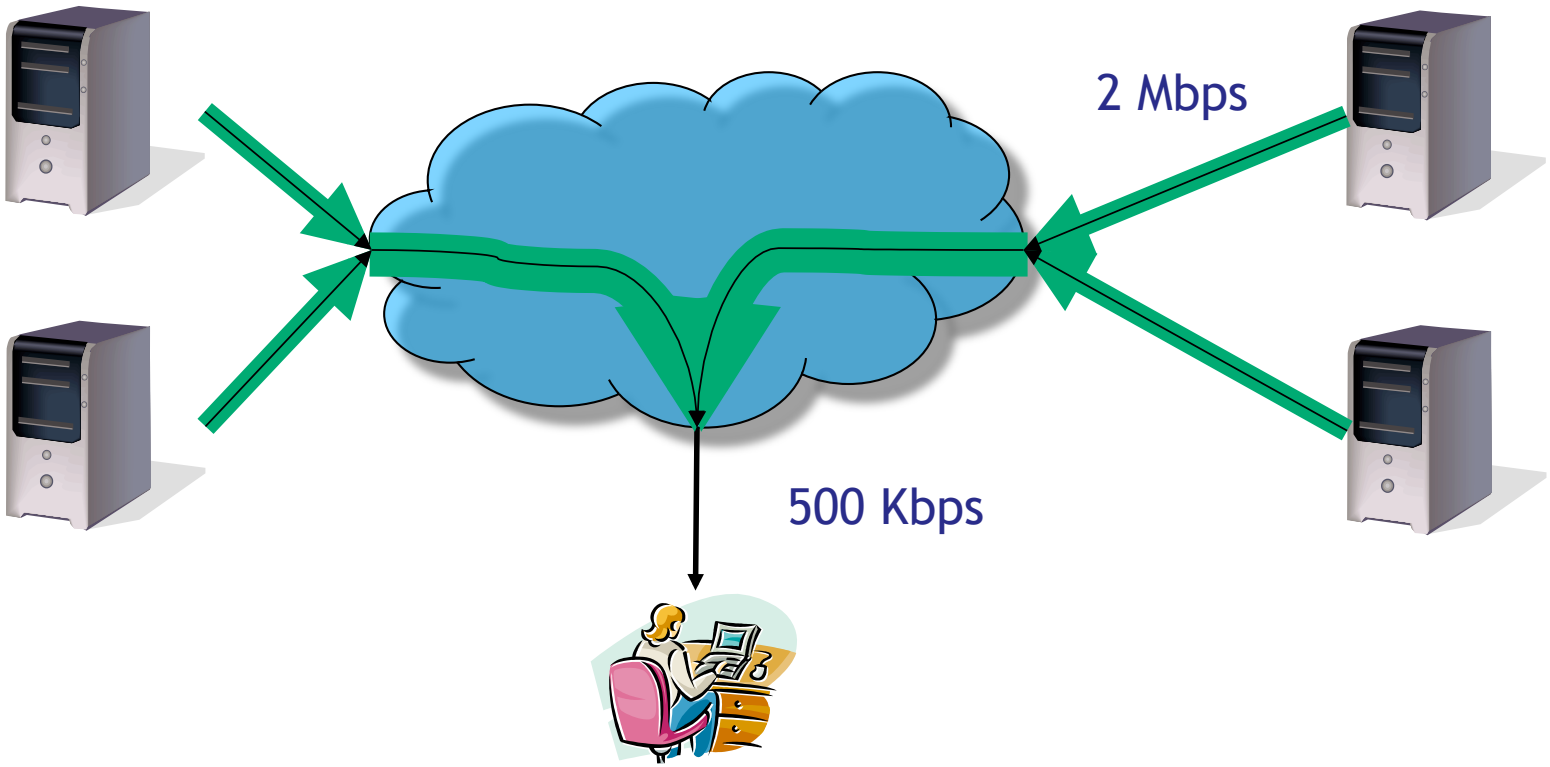
How Does Parallel Download Work?

- Downloading from several servers in parallel, user can fill her download link to capacity



Bottleneck-Disjoint Paths

- If user's access link to the network is the bottleneck, parallel downloads do not help at all
  - Might not hurt either, but parallel download has some overhead

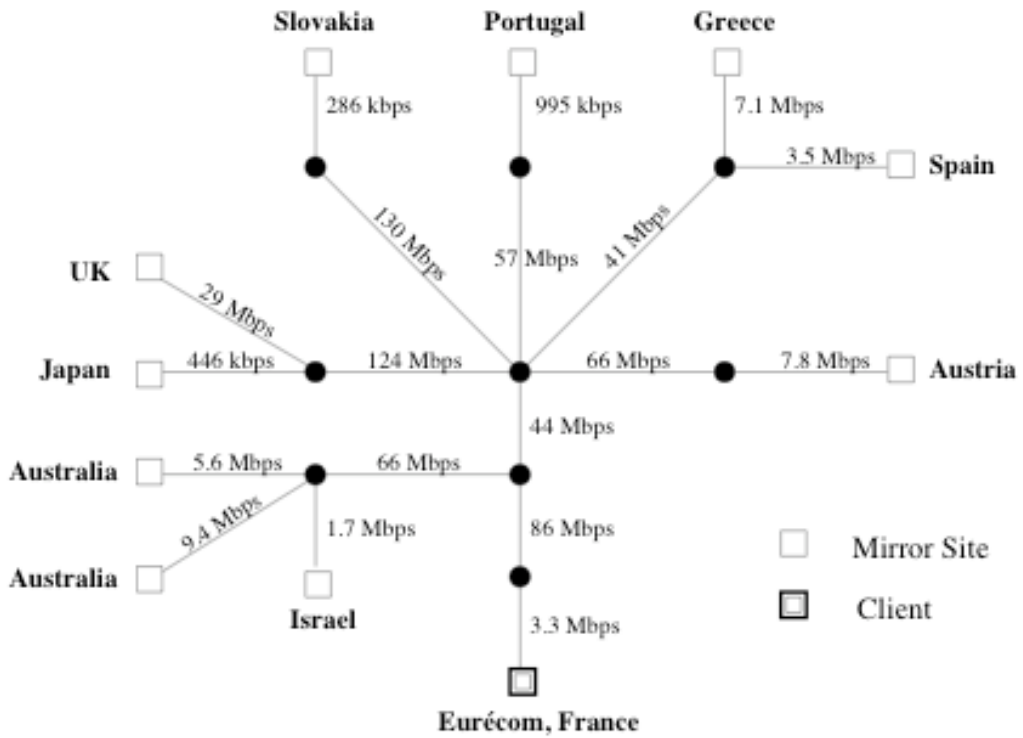


Practical Details

- Two types of parallel download defined:
  - History-based
  - Dynamic
- History-based parallel access:
  - All sources are known and past bandwidths to them are known
  - When client downloads file, it checks past bandwidths
  - Pick the best sources for download
- Dynamic parallel access:
  - Dynamically select best source according to current download speeds
  - This approach popular for P2P networks

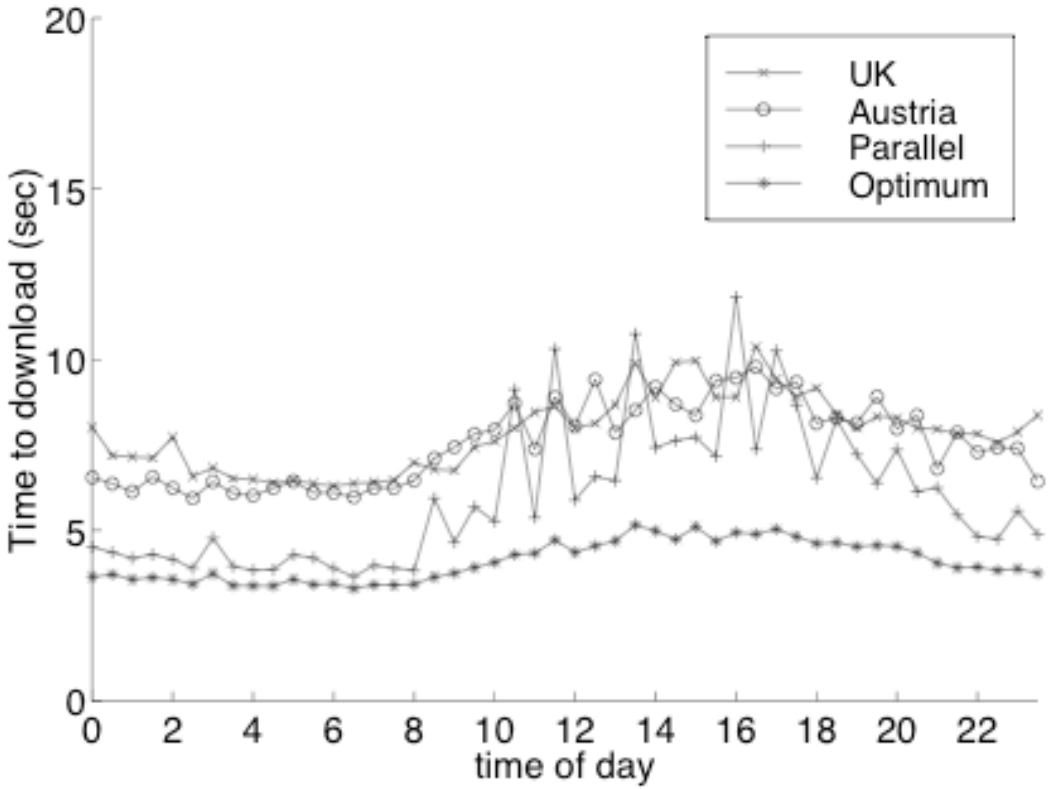
Experiment Setup

- Client in France, sources all over the world
- File size 763 KB (Squid proxy caching software)



History-Based Parallel Access

- History-based parallel access to two servers simultaneously



History-Based Parallel Access

- Optimum calculated after-the-fact
- Similar results obtained for larger sets of servers
- Observations:
  - During night, history-based access achieves good performance
  - During day, often downloading from either single server is faster than parallel!
- Solutions:
  - Different bandwidth estimates for different times of day
    - Complicated
  - Fully dynamic mirror selection

Dynamic Parallel Access

- One client, set of known servers, one file
- File divided into equal-size blocks
- Client requests file as follows

1. Client requests 1 block from each server
2. When server finished uploading, client requests new block from that server
3. When all blocks are there, client reassembles file

- Problems:
  - Servers idle for a while when waiting for new request
  - Not all servers terminate at the same time

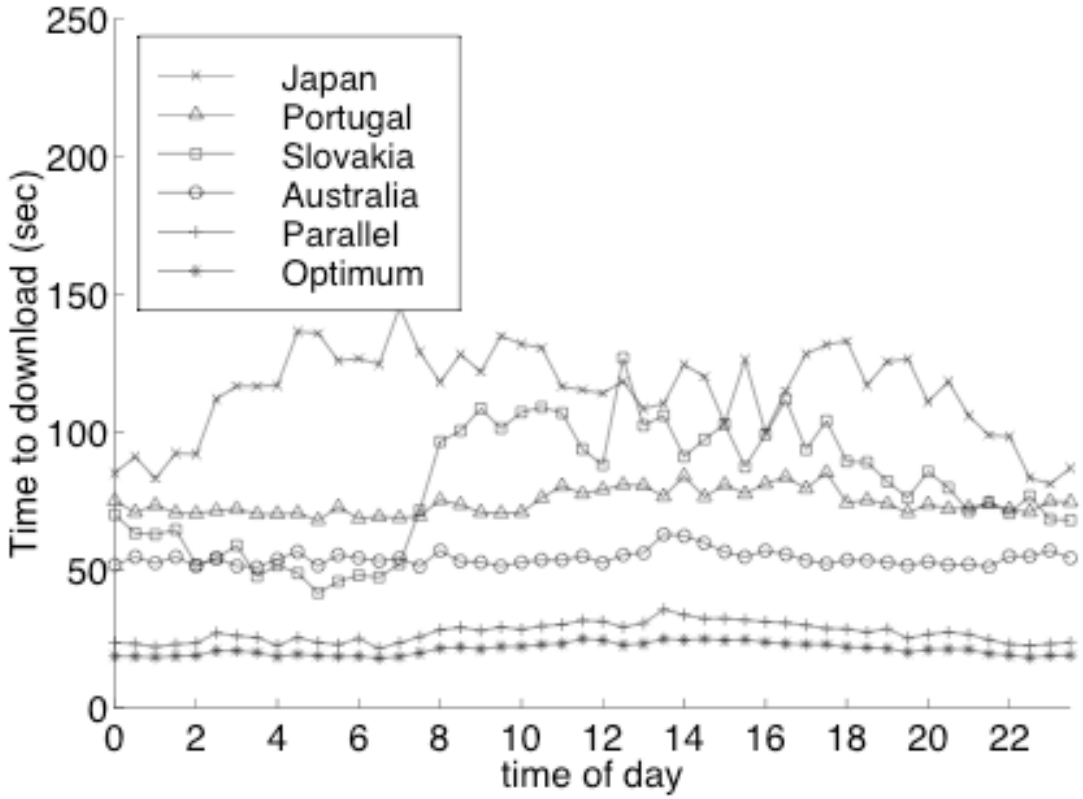
Solutions to Problems

1. Number of blocks should be much larger than number of servers
2. Blocks should be small in size
   - Provides fine-grained balancing of server capabilities
   - Aim is to finish all downloads at the same time
3. Blocks should be large enough to avoid idle times
   - Between two blocks is 1 RTT idle time
   - If blocks are large, idle times are a small fraction of total time
   - Also, possible to pipeline requests to some degree

- However, for solutions 2 and 3, file should be large

Performance

- File size 763 KB, 30 blocks, 4 servers

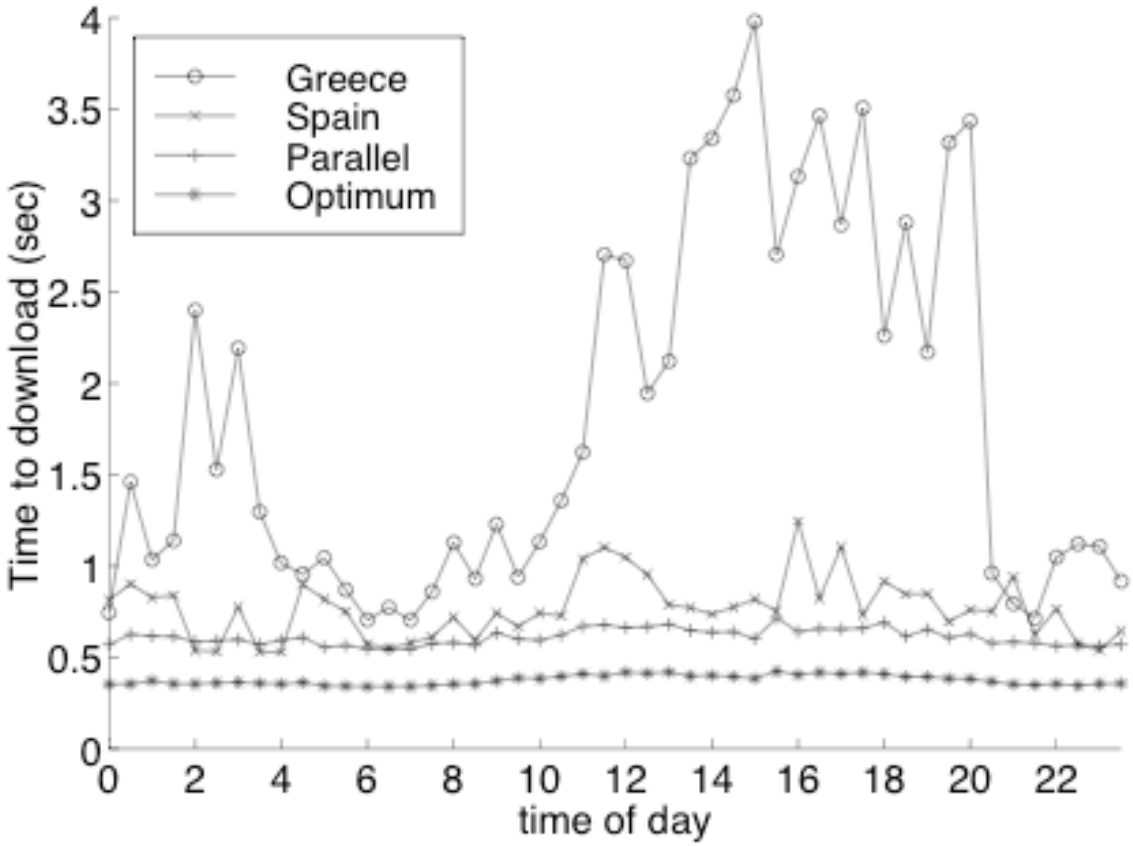


Results

- Servers chosen to minimize common network links
- Parallel downloads are almost equal to optimal
- Time goes from 50 seconds to 20 seconds
- Performance independent of the time of day
- Similar results when some servers are fast and other slow, but:
  - In this case, parallel downloads have only small performance advantage over the fastest single server
  - But: No risk of picking a bad server

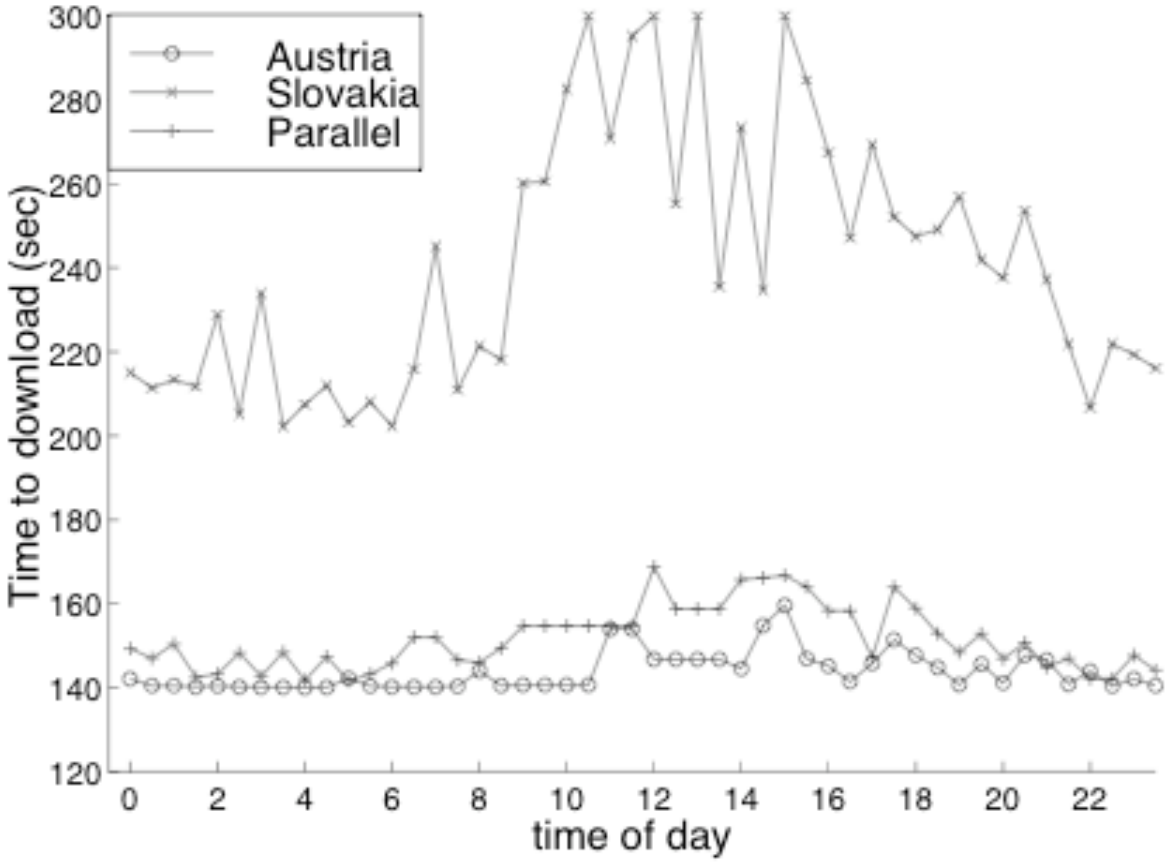
Small Documents

- Document 10 KB, 4 blocks, 2 servers
- Advantage exists, but is quite small



Shared Bottleneck Link

- Modem client, 763 KB, 30 blocks, 2 servers



Results and Summary

- Not much gain from parallel access
- In fact, picking just the better server gives better performance

Summary

- Parallel downloading efficient in heterogeneous cases
- Requires large files and bottleneck-disjoint paths
- Currently widely used in P2P file sharing networks

Chapter Summary

- Server-side techniques for content distribution
- Goals
- Mirrors
- Server farms
  - Surrogates
  - DNS load balancing
- Parallel downloading