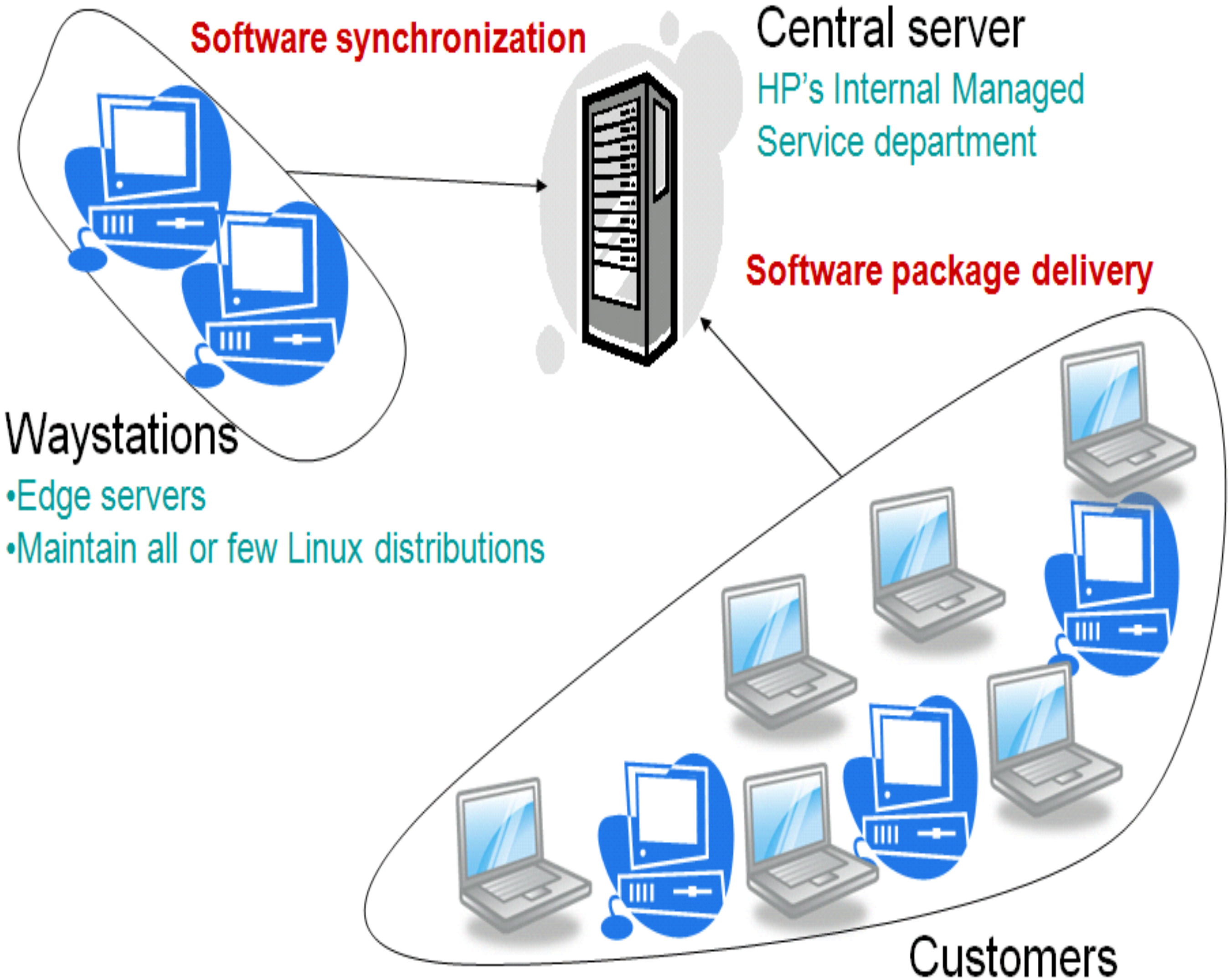
A P2P-Based Architecture for Secure Software Delivery Using Volunteer Assistance

Purvi Shah, Jehan-François Pâris
University of Houston

Jeffrey Morgan, John Schettino, Chandrasekar Venkatraman
Hewlett-Packard Laboratories

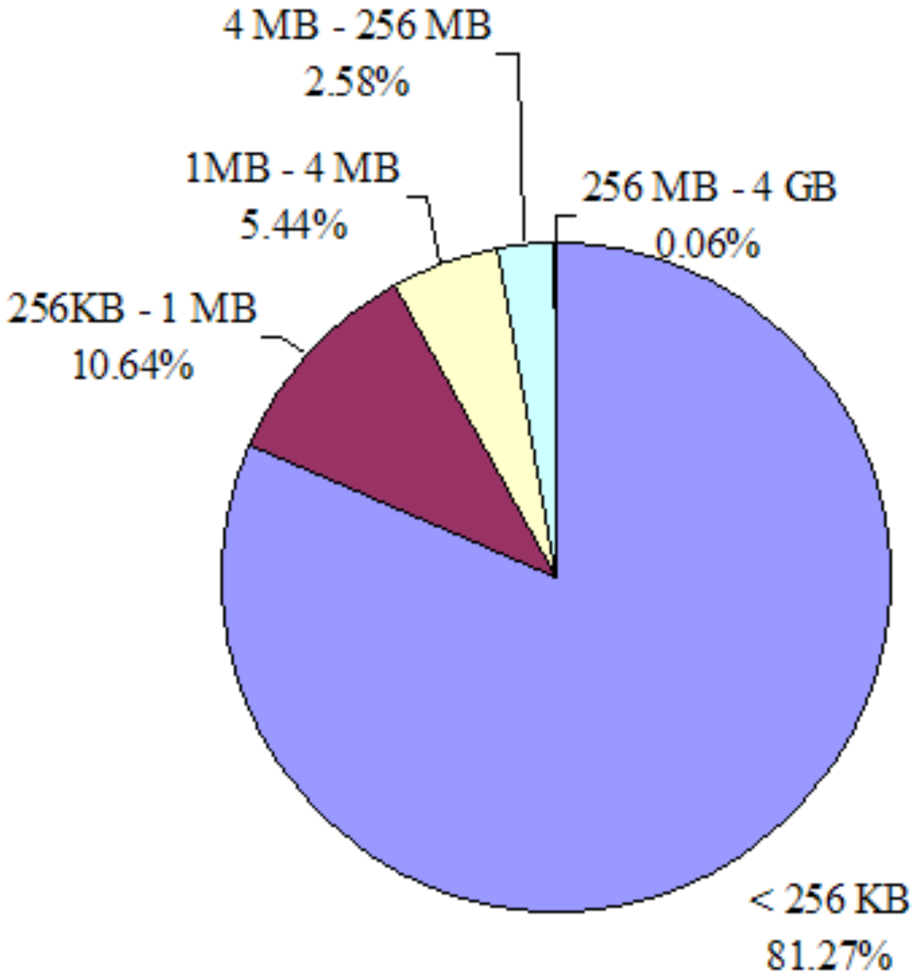
CS@UH

The Problem: Fast and Cheap Software Delivery



The Software Repository (I)

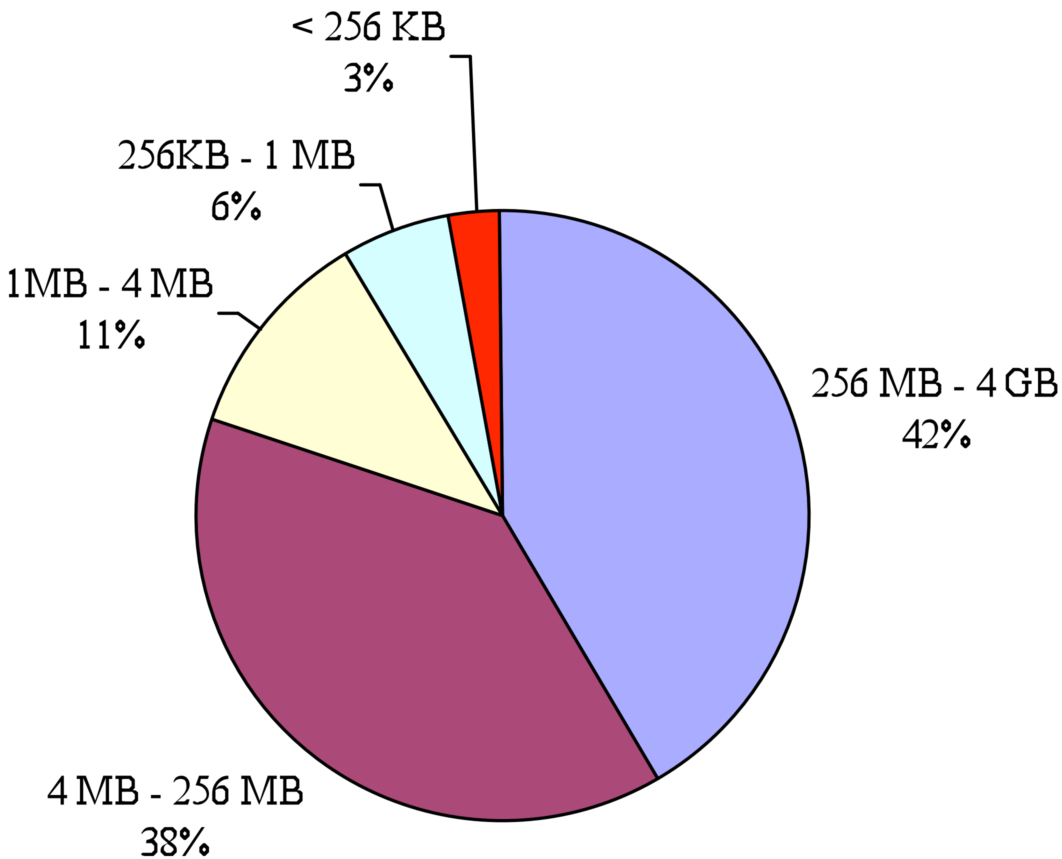
- ~ 2.2 million files
- More than 90% of files fall below the 1 MB mark



CS@UH

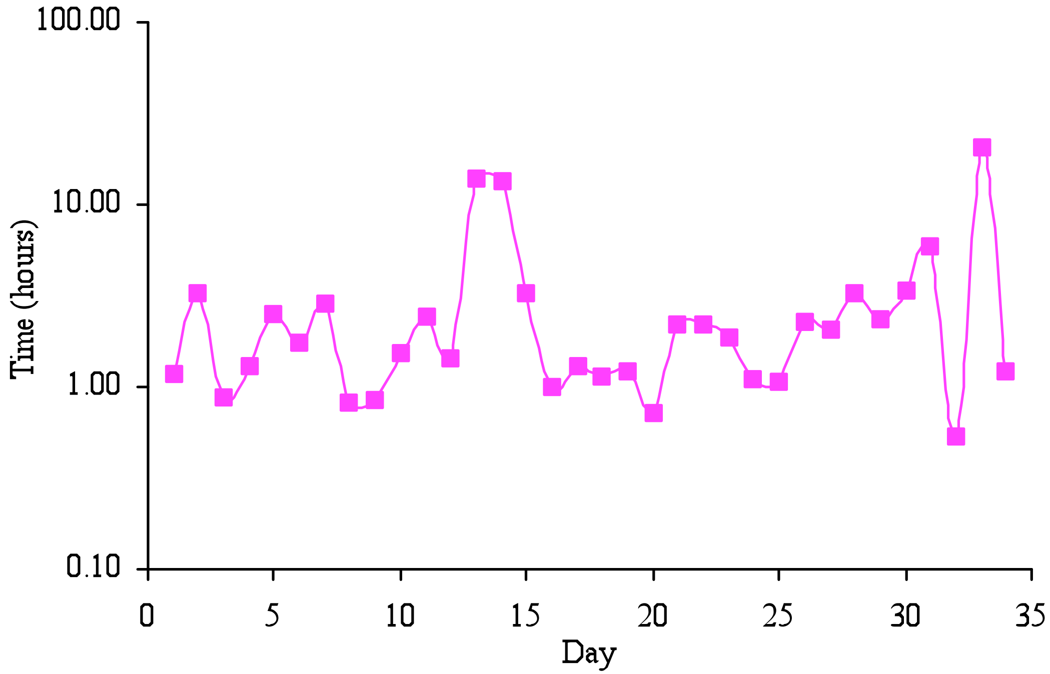
The Software Repository (II)

- ~ 2.86 TB
- More than 90% of the space is occupied by files larger than 1 MB



CS@UH

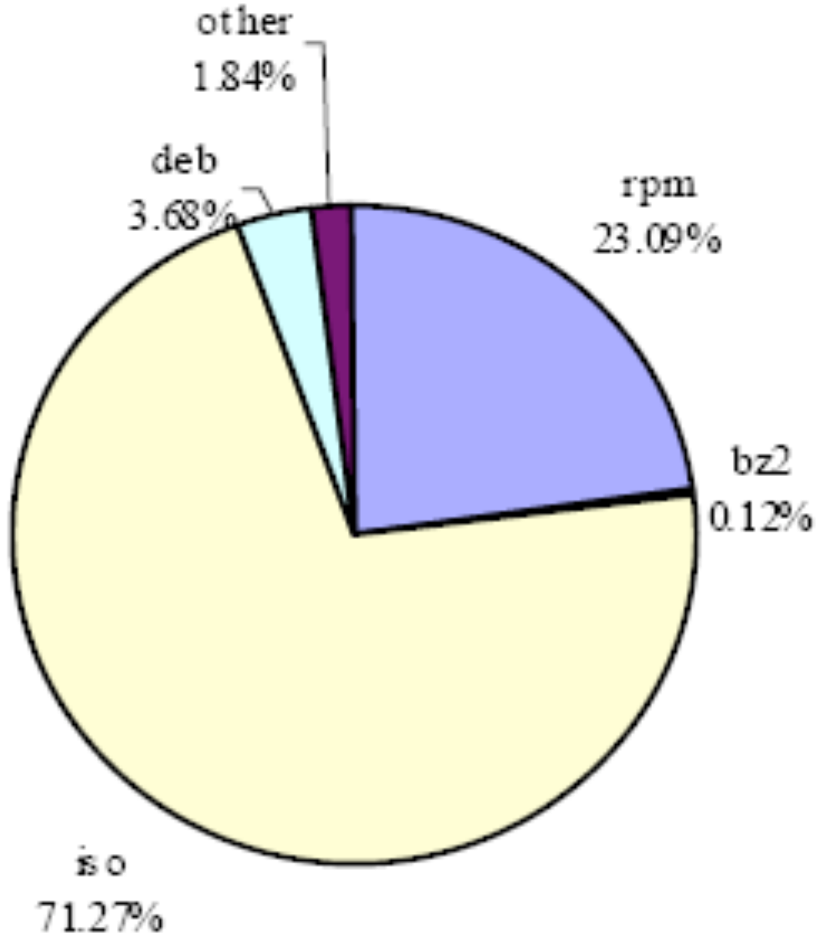
The Edge Servers



- Spend an average of 1.81 hours per day synchronizing their software distribution

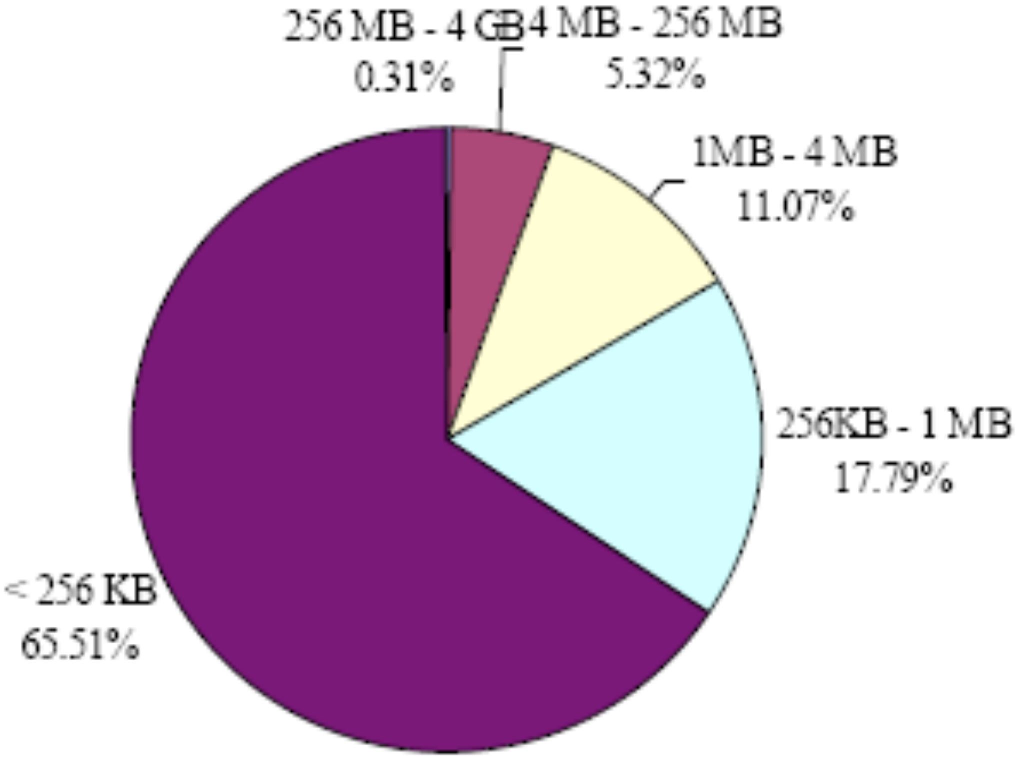
The Workload (I)

- Large image downloads represent 71% of the workload
- Update packages 23% of same

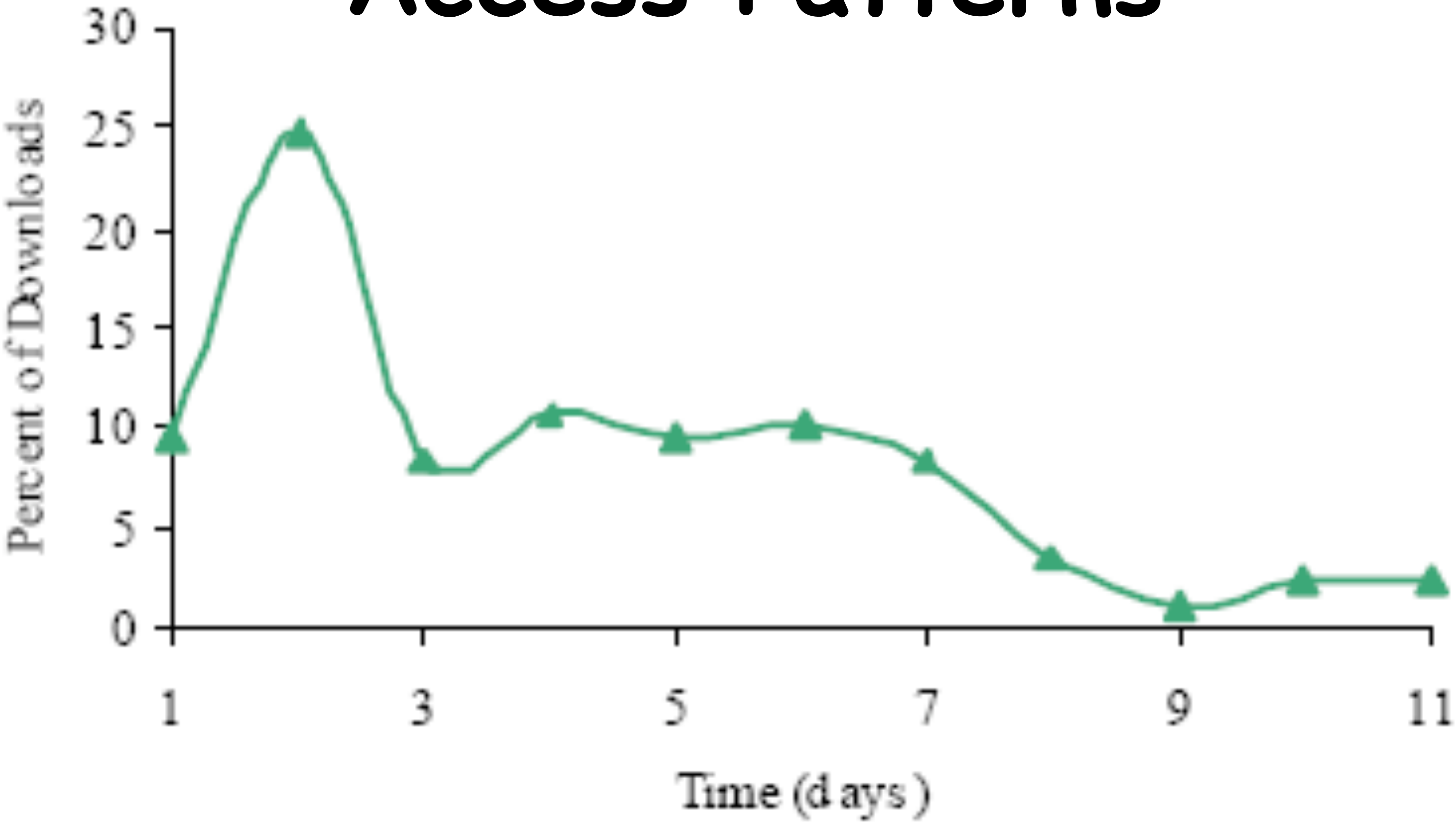


The Workload (II)

- Most of the transfers involve small files



Access Patterns



- Most download requests occur within two to three days of package release

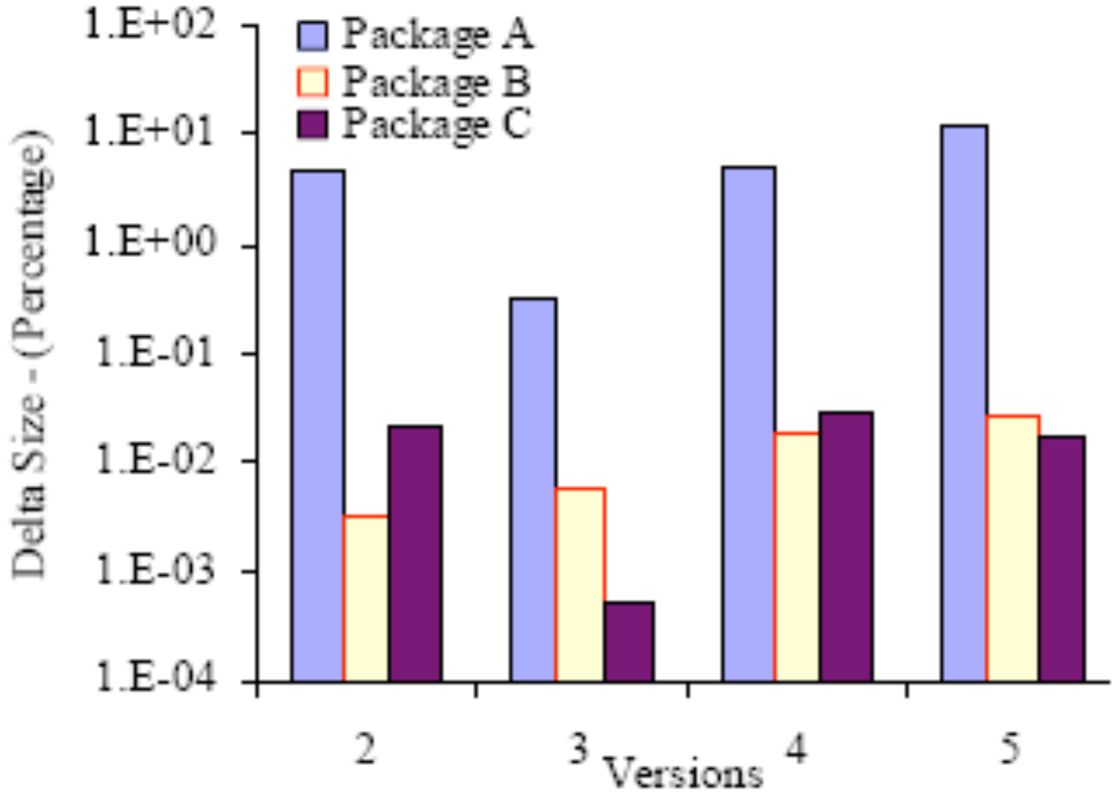
Similarities (I)

- 17 % of the files larger than 1MB are identical and only differ in name
- 17 percent of these identical files are source-code packages for more than one Linux distribution. They comprised
  - Comprise up to 85 GB of the disk space
  - Accounted for 30 percent of space occupied by identical files

CS@UH

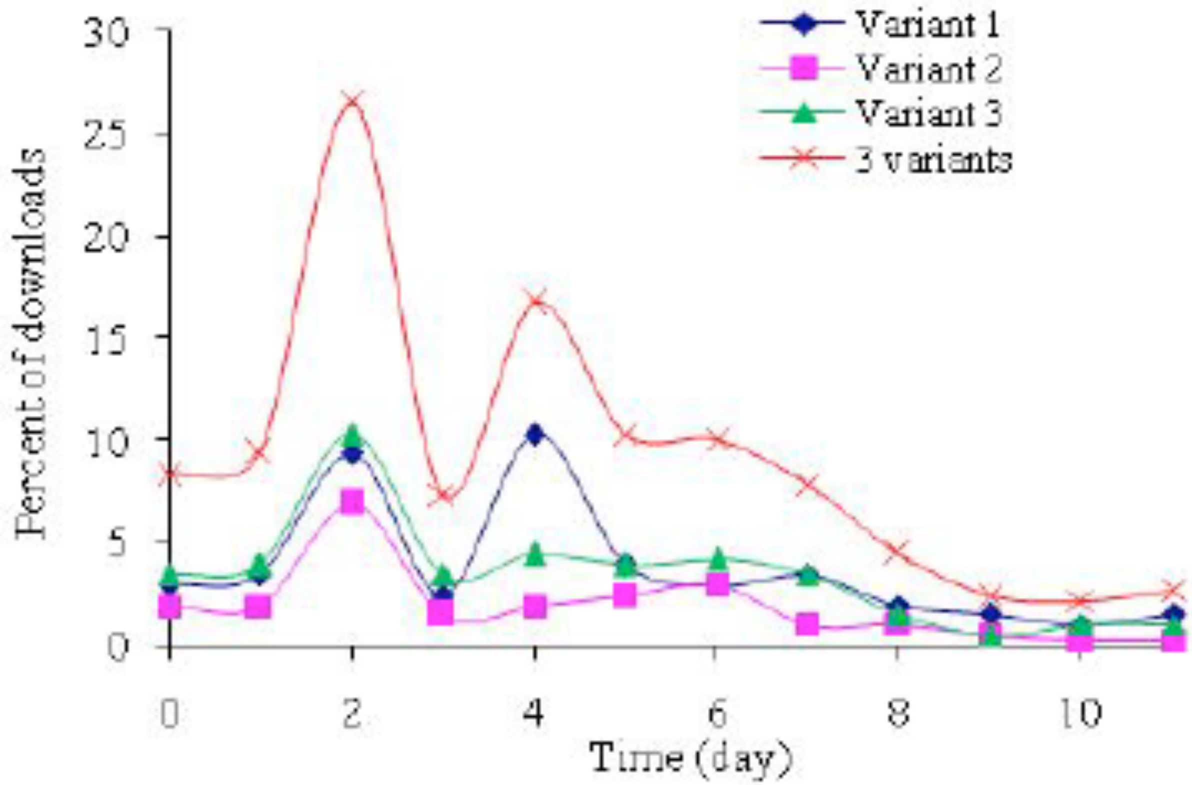
Similarities (II)

- Considerable similarity exists among variants of the same package for various distributions



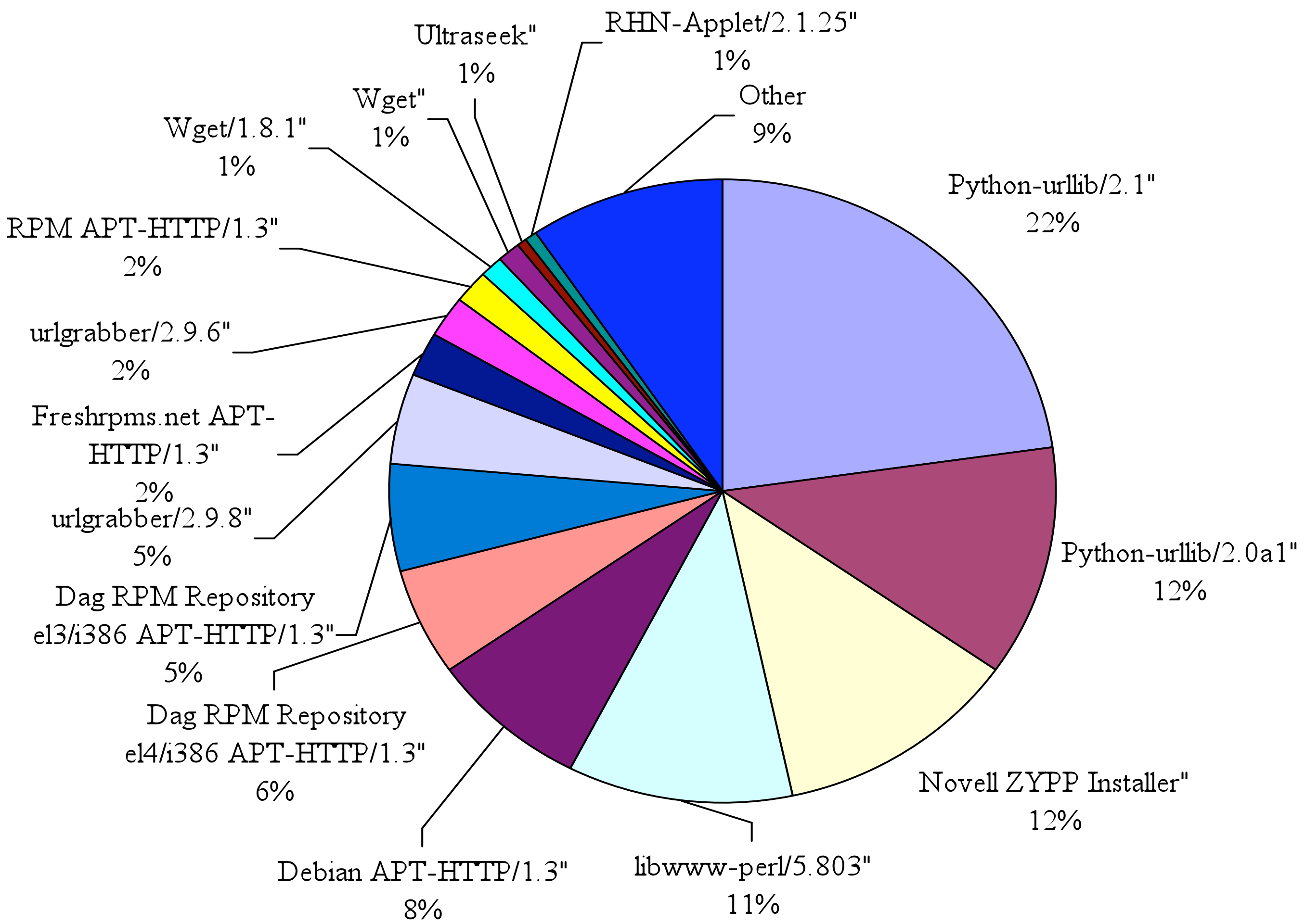
Similarities (III)

- Downloads of variants of the same package are strongly correlated



Heterogeneity

- Clients use very different tools for their updates
- Only the top 15 tools!



CS@UH

The Challenge

- Provide more efficient delivery of software updates
  - Taking advantage of package similarity
  - Being able to handle request surges
  - Without modifying the client interfaces

Our Approach

- Use ***volunteer nodes***
- Combine P2P technology with ***rsync*** (currently synchronization tool)
- New tool: ***PRsync***
  - Chop files into chunks (similar to ***rsync***)
  - Uses ***BitTorrent*** to disseminate files by exchanging chunks between peers

CS@UH

Volunteer Nodes

- Various departments manage around forty ***edge nodes***
  - *Maintain partial or complete mirrors of the software repository for specific sets of machines*
- Proposed to involve them in the software delivery processed
  - "Volunteered" by their departments

CS@UH

rsync (I)

- UNIX application that synchronizes files and directories
  - *Currently used protocol and tool*
- ***Receiver***
  - Splits its copy of each file into fixed-size non-overlapping chunks
  - Computes two checksums of each chunk
  - Sends these checksums to the sender.

rsync (II)

- ***Sender***
  - Compares checksums it has received with own checksums
    - *Use of special "rolling checksums" speeds up that step*
  - Identifies file parts that need to be sent
  - Sends these parts along with assembly instructions

CS@UH

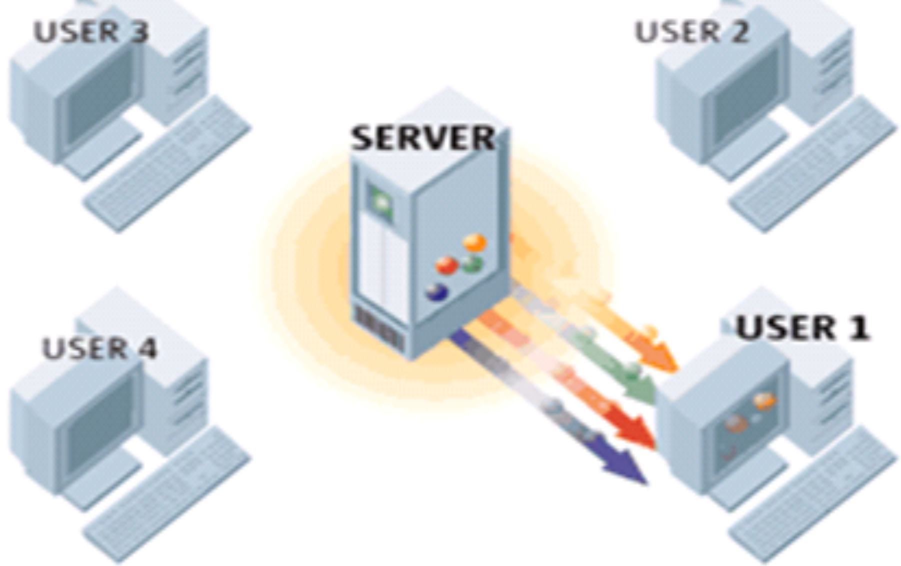
BitTorrent

- Most popular P2P protocol
- Used to distribute data among the volunteer nodes
- Clients are not involved in that process
  - *Would have required a total redesign of their client interface*

The early days of peer-to-peer file-sharing involved one user downloading a file from another user.

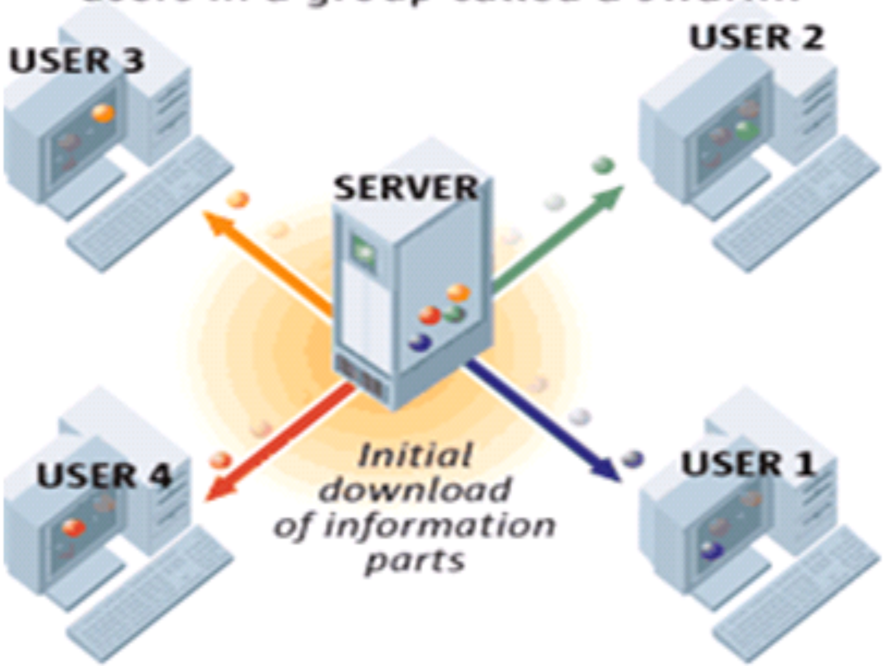
That's all changed with BitTorrent, which breaks files into small chunks and allows users to simultaneously upload and download them.

PEER-TO-PEER FILE-SHARING

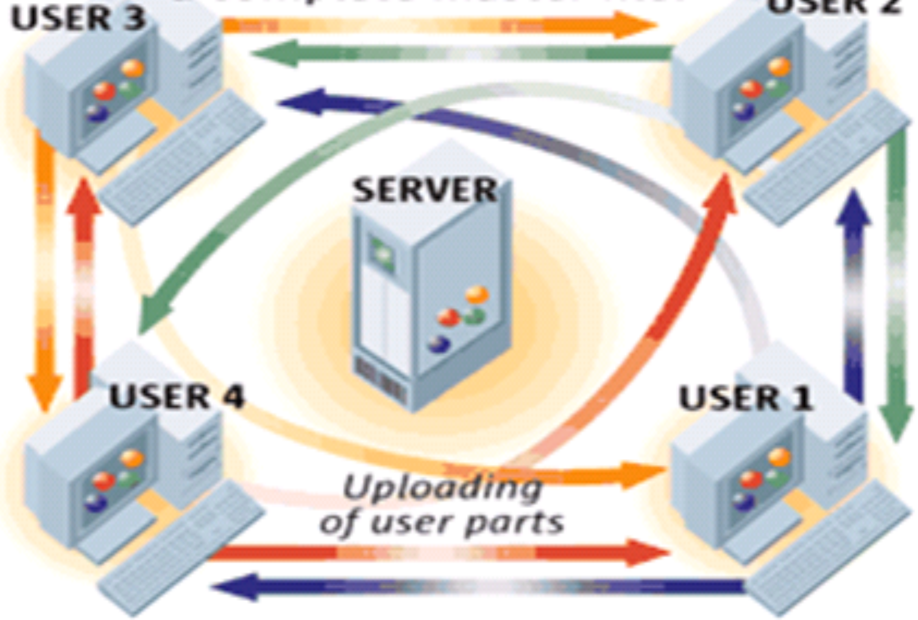


BITTORRENT

❶ The master file is broken into smaller parts and downloaded to users in a group called a swarm.



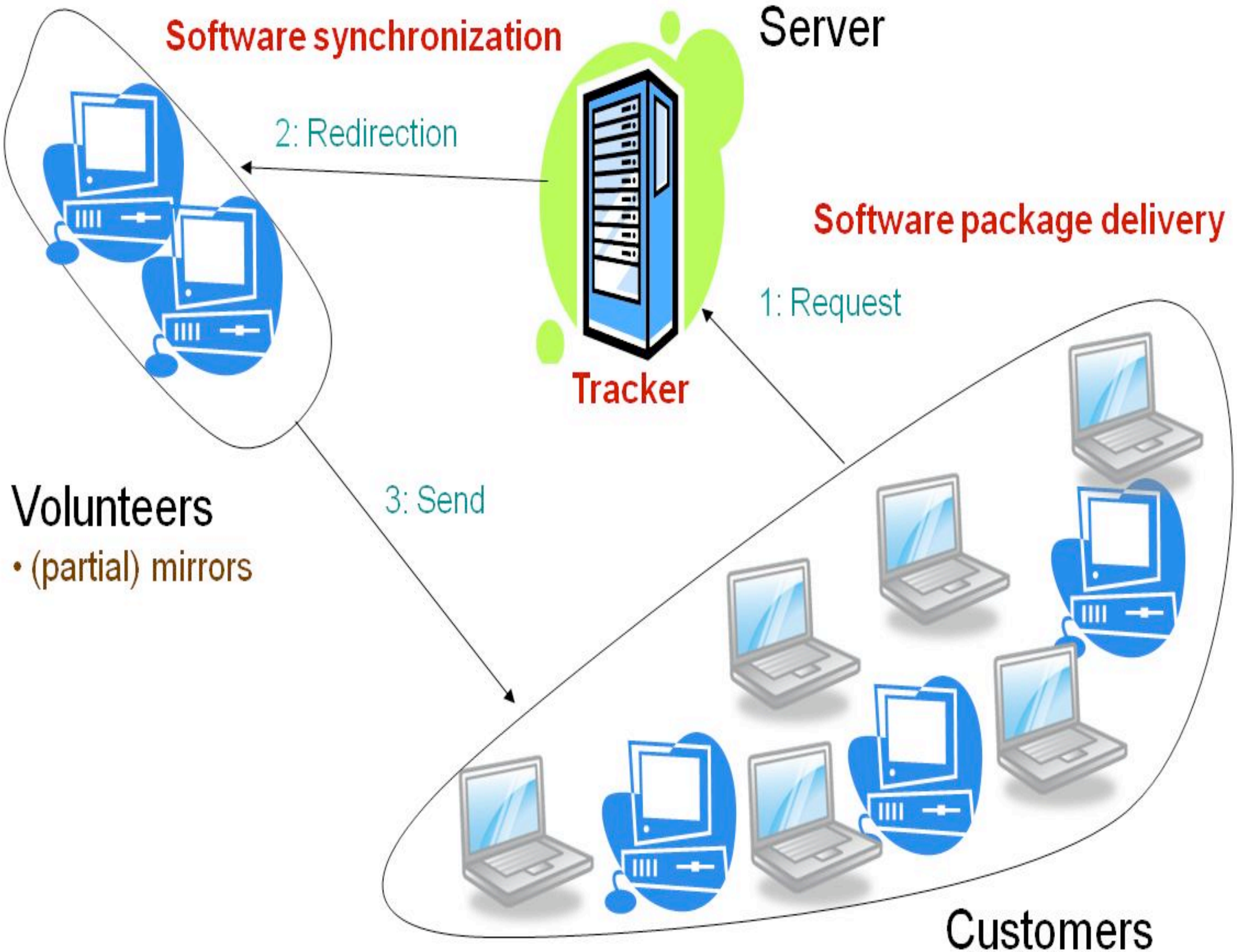
❷ As the parts are being downloaded, they are simultaneously being uploaded to other users. Each user gets a complete master file.



Solution Overview

- ***Volunteer peers***
  - Act as partial or complete mirrors
  - Synchronize their contents using **PRsync**
- ***Clients***
  - Send requests to central server
  - Get redirected to peer selected by central server
    - Using information collected by ***BitTorrent tracker***

CS@UH


Software synchronization
Server
2: Redirection
Software package delivery
1: Request
Tracker
Volunteers
• (partial) mirrors
3: Send
Customers

Key Differences from CDNs

- Customer is redirected to volunteer only when volunteer has the requested data
- Decision made by the server aided by tracker
  - Volunteers do not have to decide which customers to serve
  - Full decentralization would make security and optimization hard when using volunteers
  - Opportunity to make intelligent decisions at the tracker

Next Issue

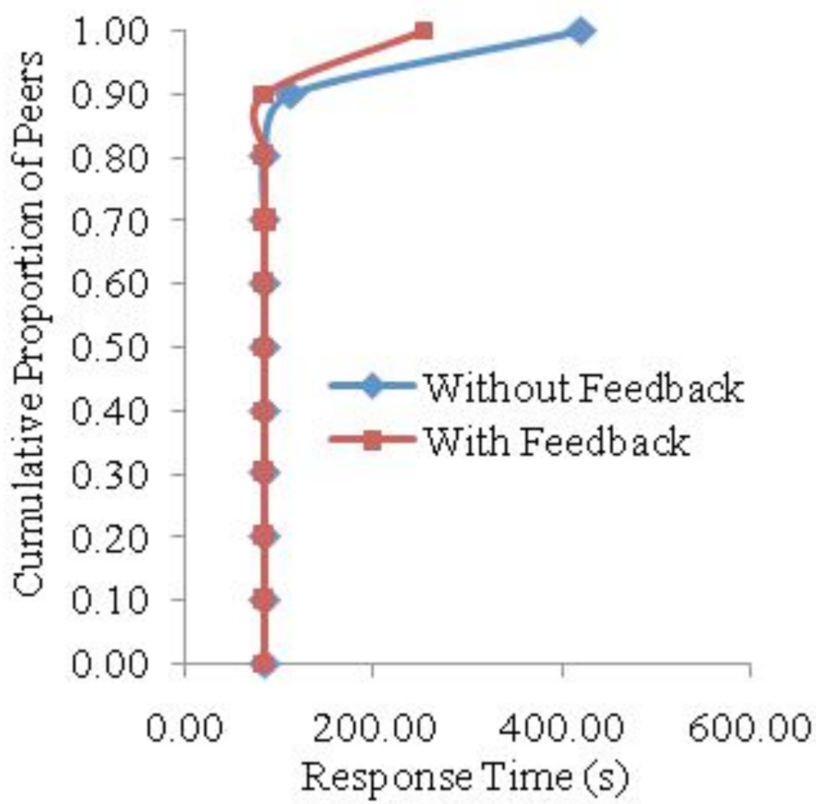
- Balancing the load among the volunteer nodes
- Must consider both components of volunteer load
  - Client requests
  - Synchronization traffic among volunteer nodes

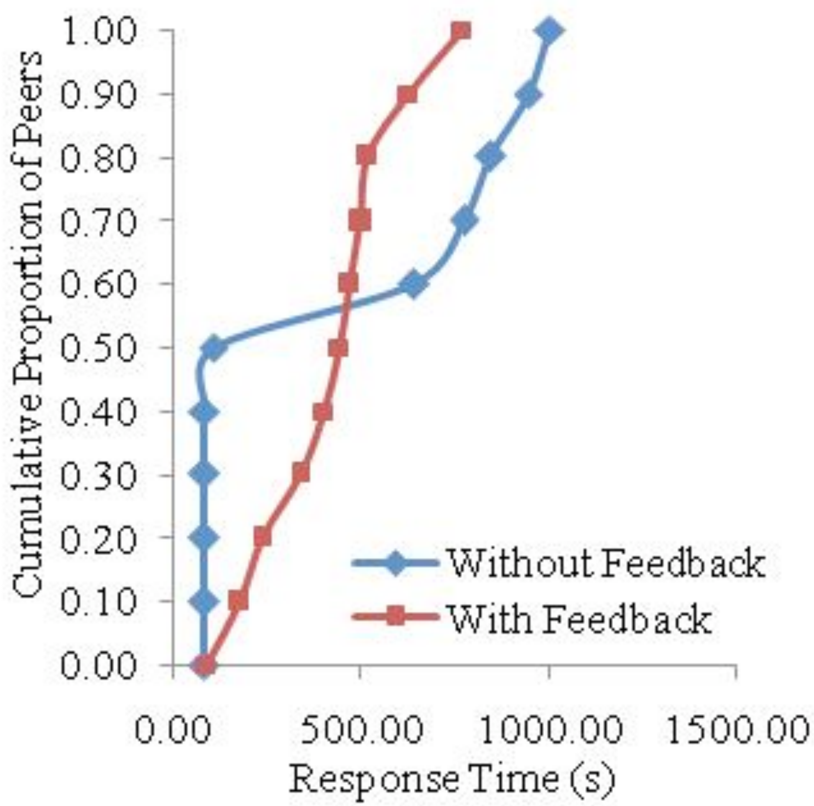
Load-Balancing

- Use a ***tracker-based solution***
- Tracker collects synchronization information
  - What files are on what volunteers
  - What volunteers are currently synchronizing their repository
- Done by piggybacking new information on existing messages
- Use this information to make better load balancing decisions

CS@UH

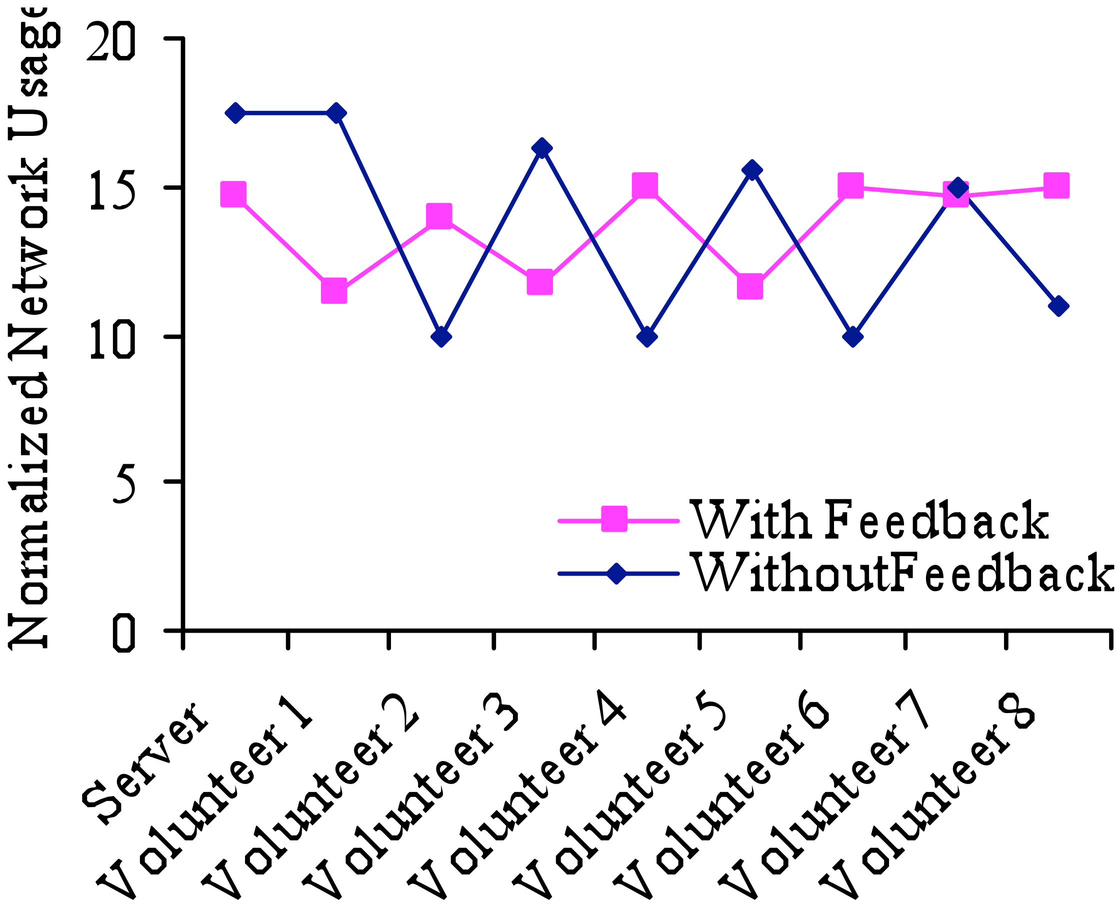
Results (I)





- More even system response times even when workload increases from 20 to 100%

Results (II)



- More equal distribution of workload among volunteers at 100% workload

Conclusions

- Three main aspects of our proposal are
  - Adding volunteer peers to distribute the central server workload
  - Using BitTorrent to synchronize the volunteer nodes (**PRsync**)
  - Using swarm tracker to collect data on volunteer node workloads and achieve more uniform client response times

CS@UH