# Project 3 Overview

15-441 Spring 2010

Recitation #9

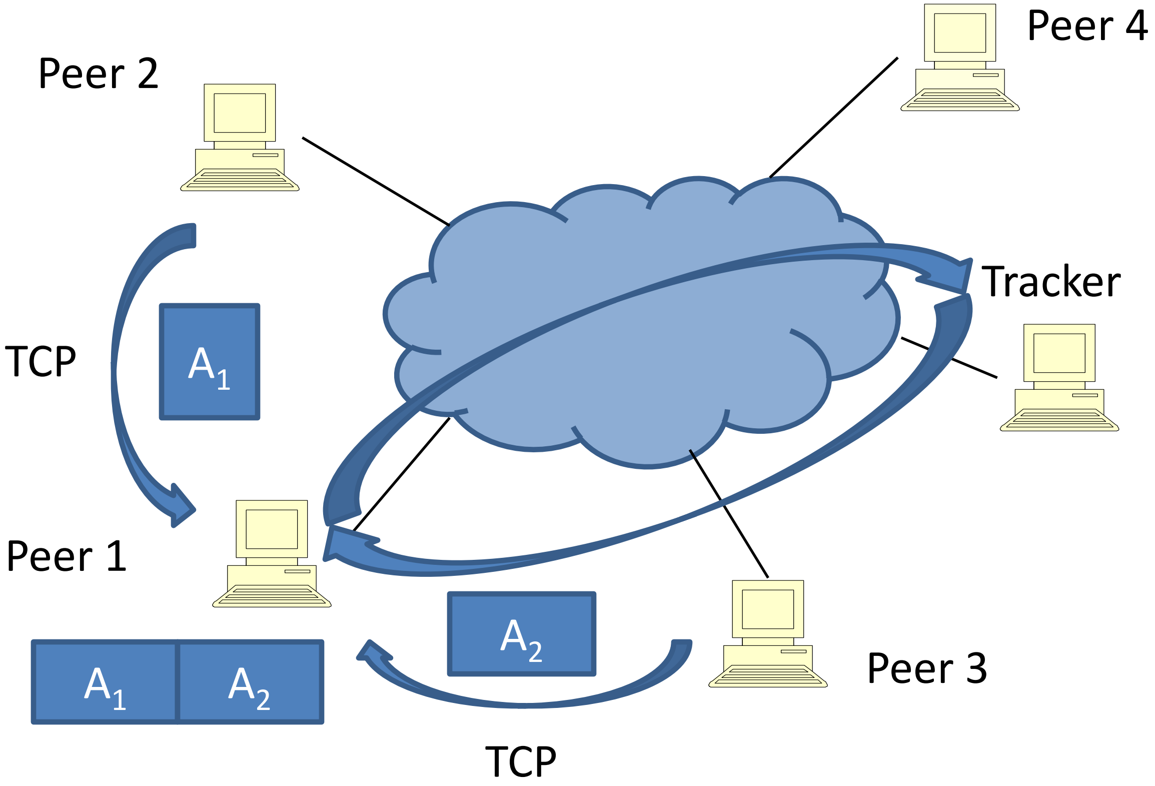
# So Far ..

- IRC Server (Project 1)
  - Focus: IRC protocol, select()
- IRC Routing (Project 2)
  - Focus: Routing (OSPF)
  - Working as a team

# Now what?

- Project 3
  - A bit of BitTorrent
  - Flow Control
  - Congestion Control
  - Again, work in teams
    - Choose fast and mail your andrew ids (id1 id2) it to kaushik@cs.cmu.edu with the subject ``15441 GROUP”

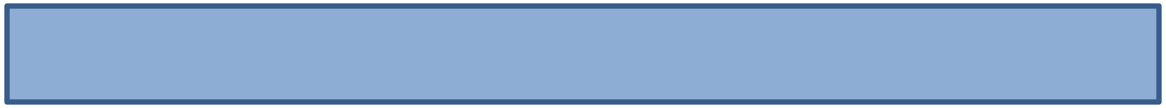
# BitTorrent



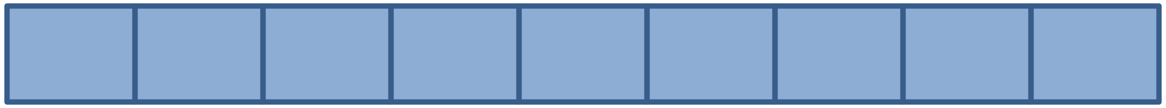
- Get the torrent file (has tracker and metadata for the file)
- Contact the tracker
- Tracker tells the peer which peers to contact for the different chunks
- Use TCP to get the chunks, use checksum to see if error-free and merge

# Chunking and hashing

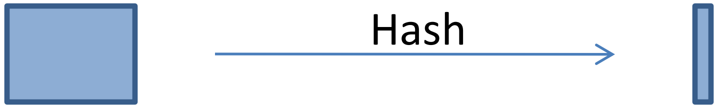
- Original file



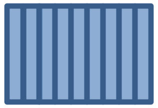
- Chunks



- Hashing



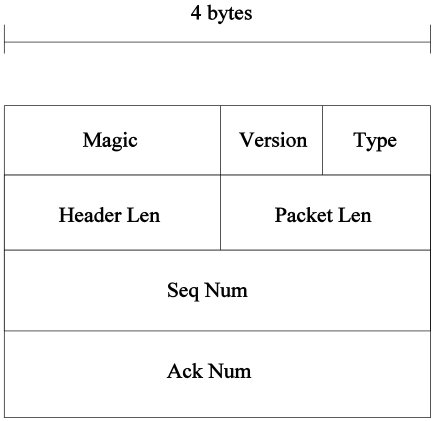
- Torrent



# This project

- Open chunks file, read the list of chunk hashes
  - All file data accessed from a single master data file
  - Peers are configured to tell them what chunks they own
- Send WHOHAS <list>
  - Instead of the tracker, flood the network
- Receive IHAVE <list> from different peers
- GET <chunk-hash> for each chunk from some peer
  - Instead of TCP, use UDP
  - Implement congestion control
- Check the hash(chunk_recvd) with chunk_hash
- Merge all the received chunks

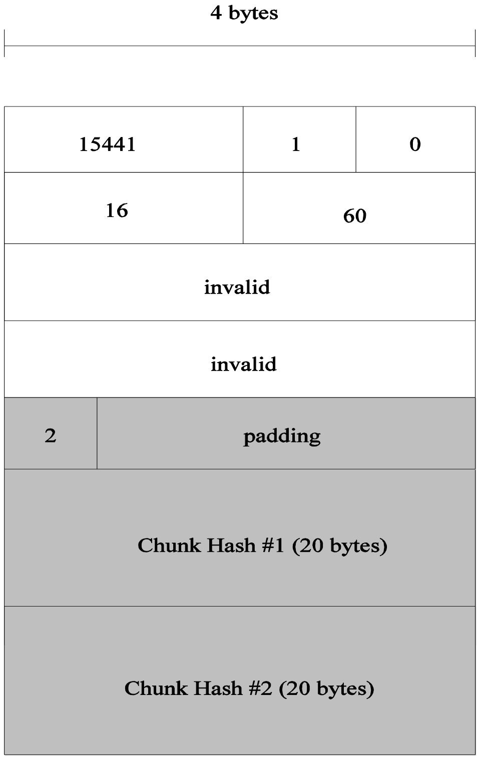
# Packet formats



4 bytes

| | | |
|---|---|---|
| Magic | Version | Type |
| Header Len | Packet Len | |
| Seq Num | | |
| Ack Num | | |

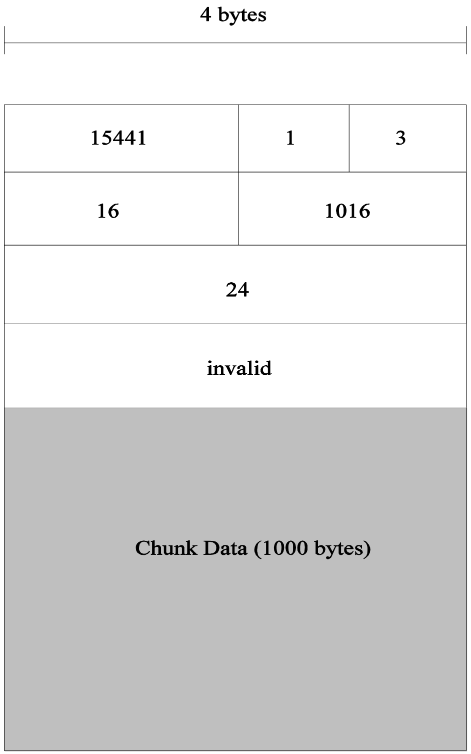
Basic packet header



4 bytes

| | | |
|---|---|---|
| 15441 | 1 | 0 |
| 16 | 60 | |
| invalid | | |
| invalid | | |
| 2 | padding | |
| Chunk Hash #1 (20 bytes) | | |
| Chunk Hash #2 (20 bytes) | | |

WHOHAS request



4 bytes

| | | |
|---|---|---|
| 15441 | 1 | 3 |
| 16 | 1016 | |
| 24 | | |
| invalid | | |
| Chunk Data (1000 bytes) | | |

DATA packet

# Terminology

- Master-data-file
  - All the data in the network is present here
  - A peer can only access the chunk that it owns
  - A peer owns a chunk if chunk-id and hash present in the has-chunk-file
- Master-chunk-file
  - File that lists chunk IDs and hashes for the chunks in the master-data-file
- Has-chunk-file
  - Per-node file containing list of hashes a node owns
  - Chunk IDs in this file match with those in the master-chunk-file
- Get-chunk-file
  - Provided by user; contains the chunk ids and hashes to be downloaded
  - Chunk IDs in this file do not match with those in the master-chunk file; they represent the position of the chunk in the file
- See Section 5 of the project description for an example

# More terminology

- Peer-list-file
  - Contains list of all the peers in the network
  - In this project, each peer knows the identity of every other peer
- Peer-identity
  - Used by the peer to get its IP address and port
- Max-downloads
  - Maximum number of simultaneous connections allowed in each direction

# Provided files

- hupsim.pl
  - Emulates a network topology using topo.map
- spiffy.[c|h]
  - Interfaces with the simulator to send and receive packets (spiffy_sendto(), spiffy_recvfrom())
- sha.[c|h]
  - Generates SHA-1 hash
- input_buffer.[c|h]
  - Handles user input
- debug.[c|h]
  - Helps in debugging
- bt_parse.[c|h]
  - Parses command-line arguments
- peer.c
  - Skeleton peer file (handles some setup and processing)
- nodes.map
  - Peer-list-file
- topo.map
  - Network topology used by hupsim.pl (your code should not read this file)
- make-chunks
  - Creates chunk-file given an input file

# Flow Control and Congestion Control

- Implement sliding window
  - Only sender window (receiver window size need not be communicated in the header)
  - Packet-based sliding window (unlike TCP)
- Implement congestion control (on top of UDP)
  - Only for data packets (chunks)
  - Slow start, congestion avoidance, fast retransmit
  - More on these in the next recitation

# Checkpoints

- Checkpoint 1 (Due: April 9th) – Start now!
  - Simple configuration of 2 hosts
  - Generate WHOHAS queries based on the input
  - Correctly respond with IHAVE
  - Assume no loss in the network
- Checkpoint 2 (Due: April 13th)
  - Send GET request and download entire chunk (2-host configuration)
  - Use simple stop-and-wait (no loss)
- Checkpoint 3 (Due: April 19th)
  - Sliding window
  - Reliability (timeouts and retransmissions)
  - Spiffy router to test your network (with loss)
- Checkpoint 4 (Due: April 23rd)
  - Simple congestion avoidance

# Project advice

- Stay ahead of the checkpoints
- Be an early bird (10 extra points)
  - Submit by April 27th
- Don’t be late
  - Submit by April 30th
- Project 3 contest! (More extra credit)
  - Optimize your system and make it robust
- Be efficient while working as a team
- Read the project write-up more than once!
- See FAQ (coming soon)
- See bboard for related questions
- Post on bboard if nothing above helps