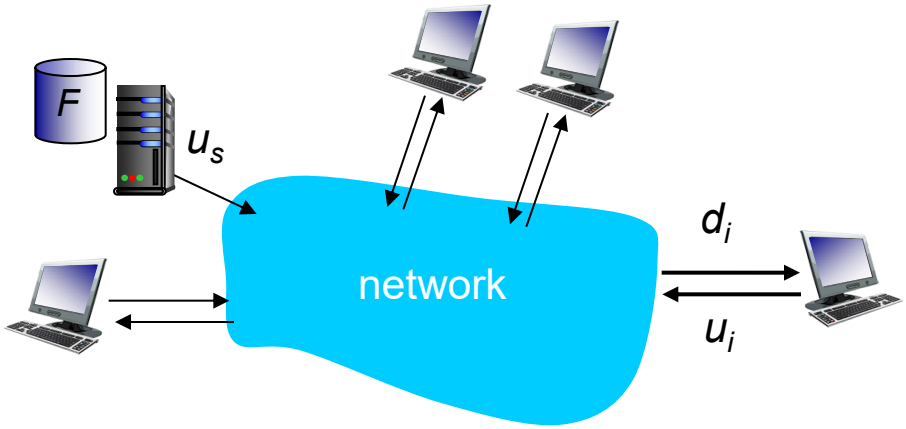
File distribution time: client-server

- *server transmission:* must sequentially send (upload) *N* file copies:
  - time to send one copy: F/u_s
  - time to send N copies: NF/u_s
- *client:* each client must download file copy
  - d_{min} = min client download rate
  - min client download time: F/d_{min}



time to distribute F to N clients using client-server approach

$$D_{c\text{-}s} \geq max\{NF/u_s, F/d_{min}\}$$

increases linearly in N

Problem 1

- The "Fluid Model": Instead of thinking about data as millions of individual, discrete packets, the fluid model simplifies this by imagining data flowing like **water through a set of pipes**.
- This model is powerful because it allows us to ignore the complex details of individual data packets and focus on the most important factor: **the bottleneck** that limits the speed of the entire operation

Problem 1 a

a) The Server is the Bottleneck ($u_s/N < d_{min}$)

the server's total upload speed (u_s), when divided among all N clients, is less than what even the slowest client (d_{min}) can handle.

Intuition: The server is the bottleneck. Its total capacity is what dictates the overall time. The server's total "work" is to send N copies of the file, for a total of N * F bits of data.

Distribution Scheme & Time:

1. The server uses its full capacity u_s.
2. It divides this capacity equally among all N clients, sending to each at a rate of u_s /N. The condition confirms that every client can handle this rate.

Problem 1 a

a) The time it takes for a single client to receive the file is:

$$\text{Time} = \frac{\text{File Size}}{\text{Rate}} = \frac{F}{u_s/N} = \frac{NF}{u_s}$$

Since all clients receive the file in parallel and finish at the same time, the overall distribution time is also NF/u_s

Problem 1 b

b) The Slowest Client is the Bottleneck ($u_s/N > d_{min}$)

This means the server is fast enough to give each client a share of bandwidth that is more than what the slowest client can handle.

Intuition: The slowest client is the bottleneck. The entire distribution process cannot finish until this "weakest link" has received the complete file. The absolute fastest this client can download the file is at its maximum rate, d_{min}

Problem 1 b

b) Distribution Scheme & Time:

1. The time for the entire distribution will be dictated by the time it takes the slowest client to finish.

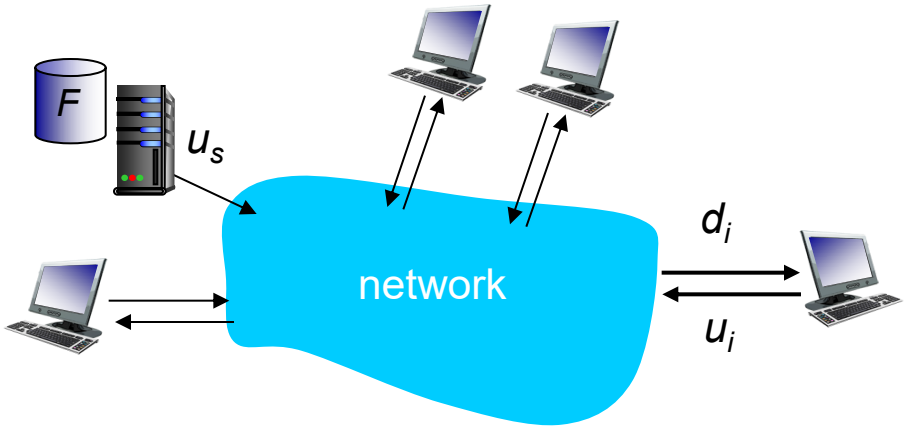
2. This client needs to download a file of size F at its maximum possible speed, d_{min}.Therefore, the minimum possible distribution time is F/ d_{min}.

3. To achieve this, the server simply needs to send data to each client at a rate of d_{min}.

4. Is this possible? The total aggregate rate required would be N* d_{min}. From the condition $u_s/N > d_{min}$, we can rearrange it to see that $u_s > N * d_{min}$. This confirms the server has more than enough upload capacity to do this. Since all clients receive the file in this amount of time, the overall distribution time is F/ d_{min}.

File distribution time: P2P

- *server transmission:* must upload at least one copy
  - time to send one copy: F/u_s
- *client:* each client must download file copy
  - min client download time: F/d_{min}
- *clients:* as aggregate must download NF bits
  - max upload rate (limting max download rate) is $u_s + \Sigma u_i$



time to distribute F to N clients using P2P approach

$$D_{P2P} \geq max\{F/u_s, F/d_{min}, NF/(u_s + \Sigma u_i)\}$$

increases linearly in N …

… but so does this, as each peer brings service capacity

Problem 2

For calculating the minimum distribution time for client-server distribution, we use the following formula:

$$D_{c\text{-}s} > max\{NF/u_{s,}, F/d_{min}\}$$

Similarly, for calculating the minimum distribution time for P2P distribution, we use the following formula:

$$D_{P2P} > max\{F/u_{s,}, F/d_{min,}, NF/(u_s + \Sigma u_i)\}$$

Where, F = 15 Gbits = **15,000 Mbits**

u_s = 30 Mbps

$d_{min} = d_i$ = 2 Mbps

u = 700 Kbps = **0.7 Mbps**

N = 100 peers

Problem 2a

For $$D_{c\text{-}s} >= max\{NF/u_{s,}, F/d_{min}\}$$

1. Time for server to upload N copies (NF/u_s):

   100×15,000 Mb / 30 Mbps = 50,000 sec

2. Time for the slowest client to download (F/d_{min}):

   15,000 Mb / 2 Mbps = 7,500 sec

The minimum distribution time is the maximum of these two values.

$D_{c\text{-}s} >= \max\{50000 \text{ sec}, 7500 \text{ sec}\} = 50{,}000$ sec

The correct answer is

C) 50000 sec.

Problem 2b

For $D_{P2P} > max\{F/u_{s,}, F/d_{min,}, NF/(u_s + \Sigma u_i)\}$

1. Time for server to upload 1 copy (F/u_s):
   15,000 Mb / 30 Mbps = 500 sec
2. Time for the slowest client to download (F/d_{min}):
   15,000 Mb / 2 Mbps = 7500 sec
3. Time for the entire system to distribute N copies ($NF/(u_s + N * u)$):
   The total upload capacity is the server's plus all N peers':
   30 Mbps + (100 * 0.7 Mbps) = 30 + 70 = 100 Mbps.
   100×15,000 Mb / 100 Mbps = 15,000 sec

$D_{P2P} >$ = max{500 sec,7,500 sec,15,000 sec}=15,000 sec

The correct answer is

B) 15000 sec.

Problem 3a

- Yes, Bob's claim is possible, but it would be very slow. He can acquire the file through a BitTorrent feature called **optimistic unchoking**
- To encourage sharing, BitTorrent peers use a "tit-for-tat" system. By default, a peer "chokes" you, meaning it will not upload any file pieces to you. To get data, you must be "unchoked" by that peer.
- There are two primary ways a peer unchokes you:
  - Reciprocation (Tit-for-Tat): A peer constantly monitors who is sending it data at the highest rates. It will reward these top partners by unchoking them, allowing them to download in return. Since Bob never uploads, he will almost never be unchoked this way.
  - **Optimistic Unchoking**: In addition to its top partners, a peer will also unchoke one other random peer for a short period. It does this "optimistically," hoping to discover a new, better partner. This mechanism gives brand-new peers a chance to get their first pieces and start participating. **By staying connected long enough, Bob can slowly accumulate all the necessary file pieces from these temporary, random acts of sharing from many different peers.**

Problem 3b

- Yes, Bob's claim is absolutely possible. Using multiple computers is a well-known way to exploit the system and make free-riding much more efficient. This is a form of what's known as a **Sybil attack** in peer-to-peer networks.

- In essence, Bob creates the illusion of being many different users, allowing him to unfairly multiply his chances of receiving data from the peers who are genuinely sharing.

Problem 4

DASH (Dynamic Adaptive Streaming over HTTP) is the technology that services like Netflix and YouTube use to deliver video smoothly.

- Instead of sending one giant video file, the server stores the video in multiple versions of different qualities (e.g., 480p, 1080p, 4K). Each version is broken into small, consecutive "chunks" of a few seconds.
- Your player (on your phone or TV) intelligently requests these chunks one by one, dynamically adapting to your internet speed. If your connection is fast, it requests 4K chunks; if it slows down, it seamlessly switches to 1080p chunks to avoid buffering.

Problem 4.a

a. Mixed Audio and Video

- In this scenario, the audio and video for each quality level are combined into a single file.
- The problem states there's a **strict one-to-one matching** of quality levels.
- This means:
  - The highest quality video (V1) is paired only with the highest quality audio (A1) to create one file.
  - The next-highest quality video (V2) is paired only with the next-highest quality audio (A2)
  - ....and so on.
- This creates a simple list of N distinct files:

Problem 4.b

b. Separate Audio and Video

In this modern approach, the server stores the audio and video streams separately, and the client's player requests one of each and syncs them up. It provides **maximum flexibility**.

The server needs to have all versions available for the client to choose from:

- N separate video files: one for each of the N quality levels.
- N separate audio files: one for each of the N quality levels.

The total number of files the server must store is the sum of the video files and the audio files.

Total Files = (Number of Video Files) + (Number of Audio Files) = N + N = 2NThe correct answer is B) 2N.