Video-helper

Abhishek Mangla,
Faramarz Munshi,
Yuzhou Guo

Motivation & Goal

Goal: Video playback in the presence of overwhelming network congestion!

Technical contributions:

- “Parallel” downloading (faster)
- Application of meshes for synchronization of group view
- Decreasing network congestion near server

Most Important Related Work

- YuTube: A Scalable Distributed Video-streaming System
  - Idea of using “Meshes” for synchronized group views of global state.
  - Use mesh to map public IP and port to the machine’s private IP behind the router, get the incoming video port, and available disk space (in bytes).

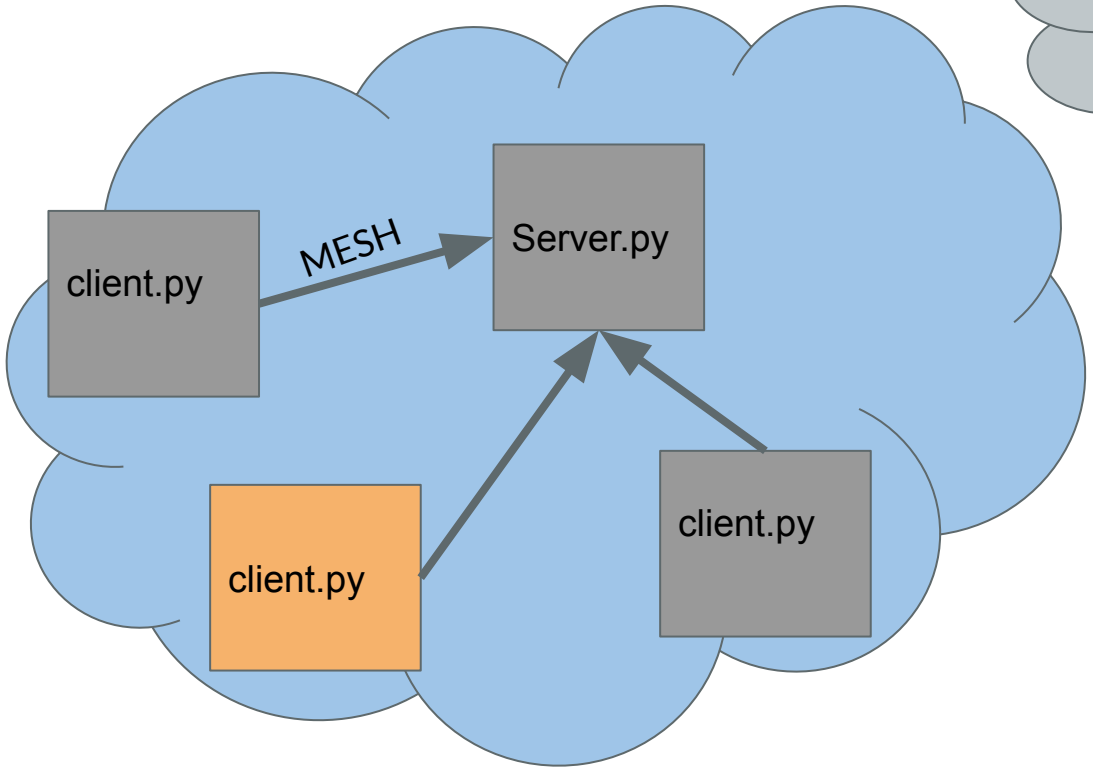
Example of our mesh:

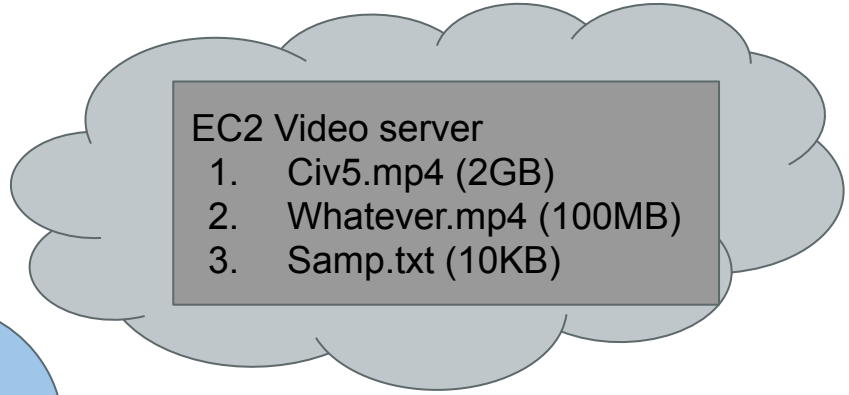
```
{"98.164.242.11-64680":
        {
                "disk": 15762563072,
                "private_ip": "192.168.0.203",
                "sender_vid_port": 64681
                “server_rtt”: 0.102353
        },
}
```

Most Important Related Work

- Distributed Video Streaming Over Internet
  - Receive-driven protocol for multi streaming
  - **Control Packet**: Sent by receiver to synchronize over multi senders, essential for Partition Algorithm
  - **Congestion Control**: Rate control by specifying the sending rate of multiple senders in order to reduce jitter
  - **Partition Algorithm**: Decide which sender to send at a specific time based on faster sender gets the turn

Current Architecture



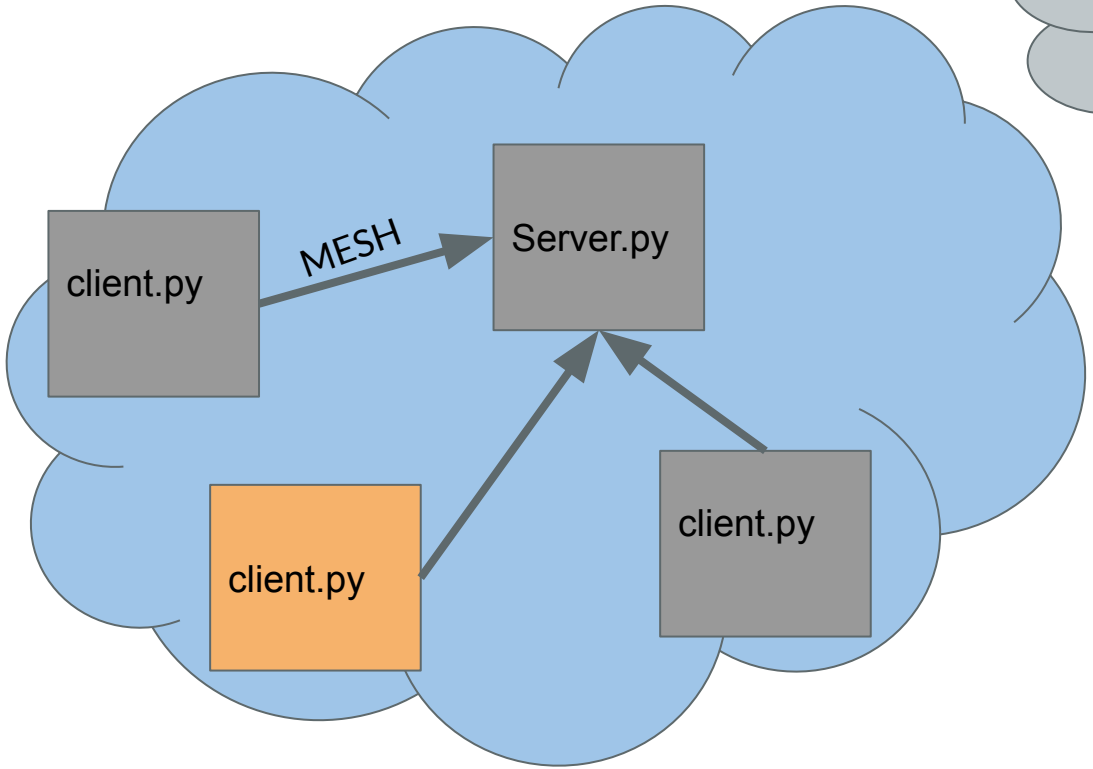


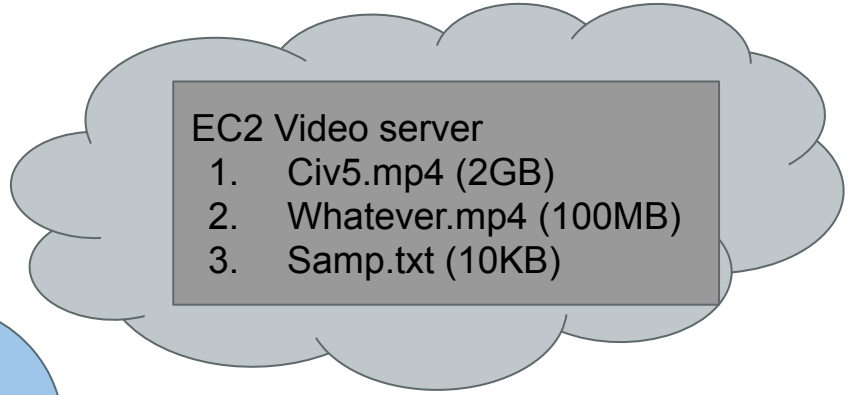
Each client sends “connect” request to server.

Then each client sends a “mesh” of its machine attributes:

```
{"98.164.242.11-64680":
        {
                "disk": 15762563072,
                "private_ip": "192.168.0.203",
                "sender_vid_port": 64681
                “server_rtt”: 0.102353
        },
}
```

Current Architecture



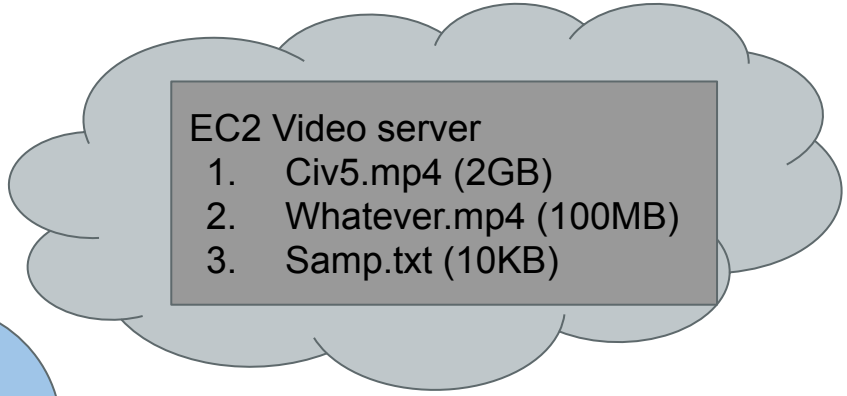


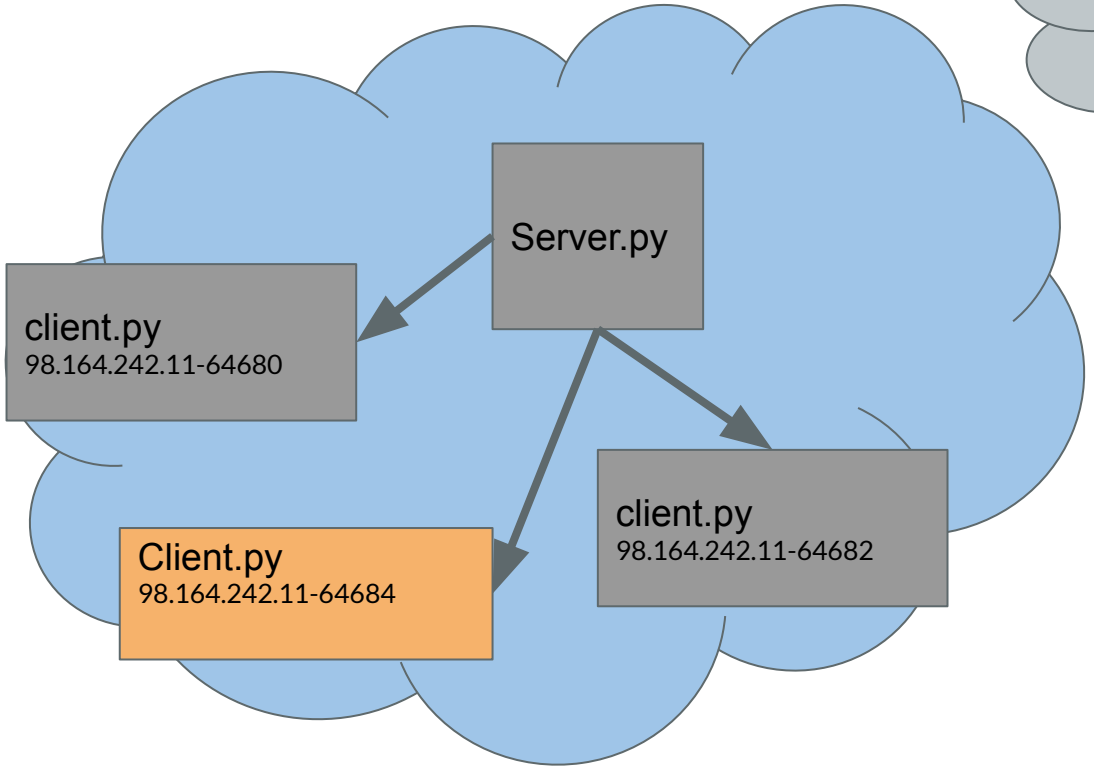
Each client sends “connect” request to server.

Then each client sends a “mesh” of its machine attributes:

```
{"98.164.242.11-64680":
        {
                "disk": 15762563072,
                "private_ip": "192.168.0.203",
                "sender_vid_port": 64681
                “server_rtt”: 0.102353
        },
}
```

Current Architecture





The server broadcast its view of the "mesh" of each machine attributes. This is the ground truth.

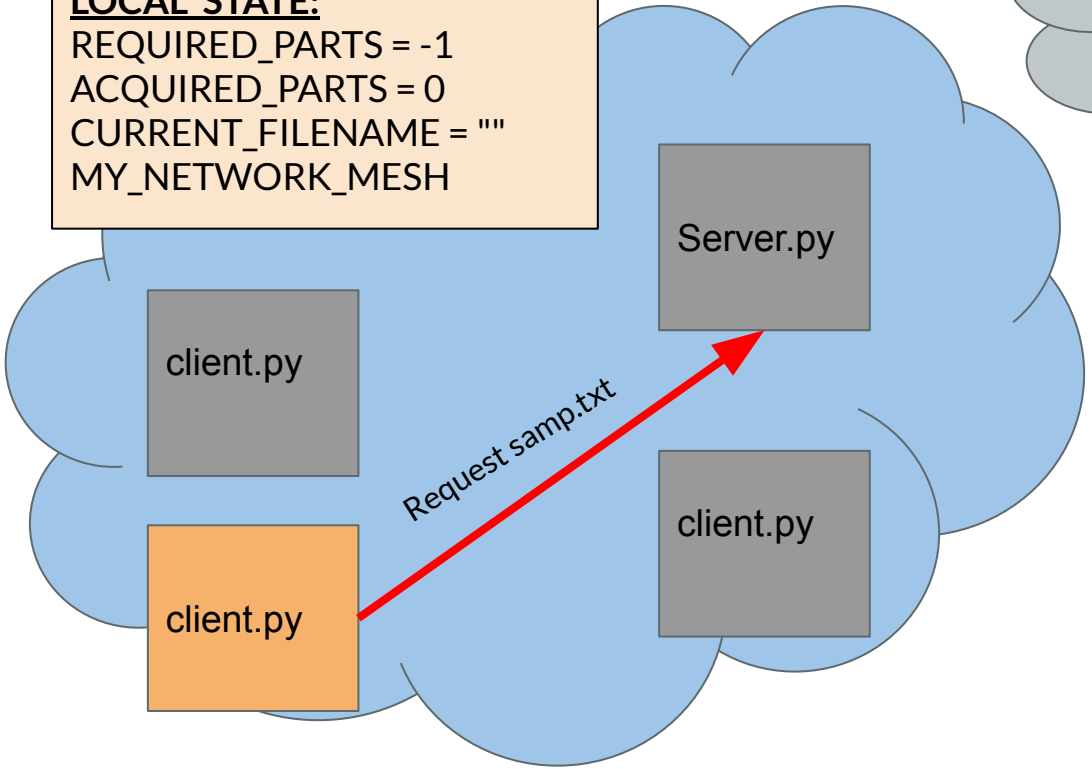
{

"98.164.242.11-64680": {. . .},
"98.164.242.11-64682": {. . .},
"98.164.242.11-64684": {. . .},

}

Current Architecture

LOCAL STATE:
REQUIRED_PARTS = -1
ACQUIRED_PARTS = 0
CURRENT_FILENAME = ""
MY_NETWORK_MESH



EC2 Video server
1. Civ5.mp4 (2GB)
2. Whatever.mp4 (100MB)
3. Samp.txt (10KB)

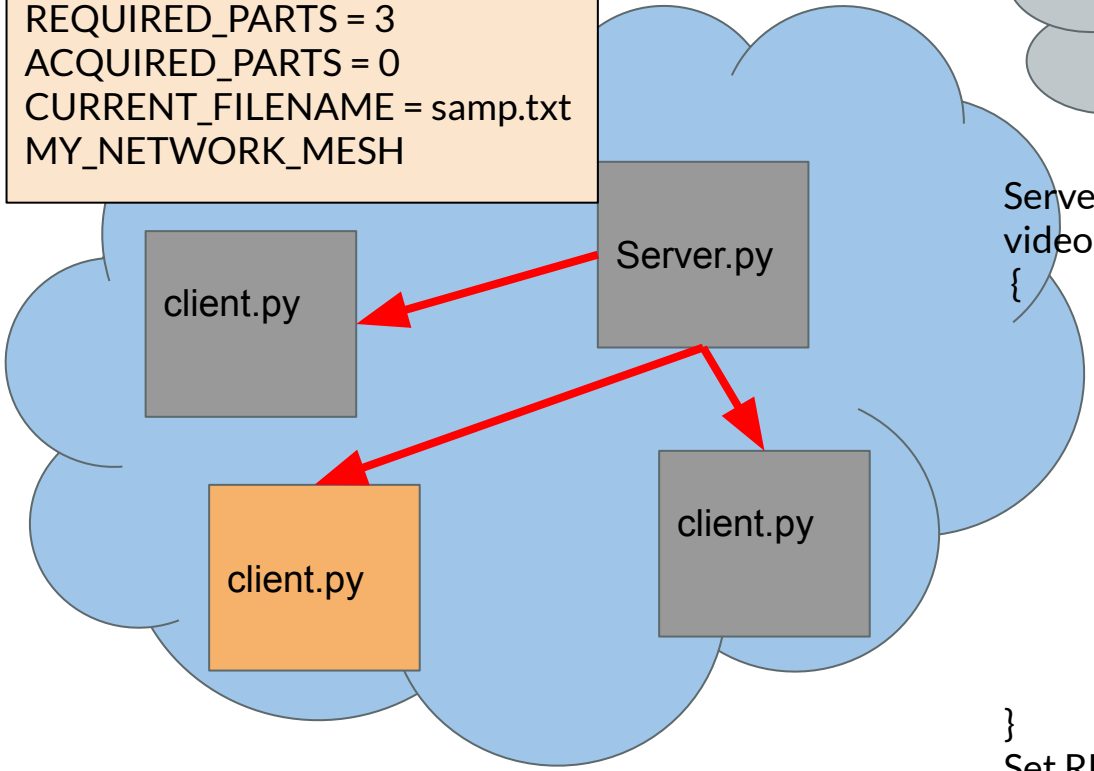
A client send request of a video to the server by specifying filename

CURRENT_FILENAME set.

Send filename request + MY_PRIVATE_IP

Current Architecture

LOCAL_STATE:
REQUIRED_PARTS = 3
ACQUIRED_PARTS = 0
CURRENT_FILENAME = samp.txt
MY_NETWORK_MESH



EC2 Video server
1. Civ5.mp4 (2GB)
2. Whatever.mp4 (100MB)
3. Samp.txt (10KB)

Server tells all connected clients what parts of the video to obtain with a VID_REQUEST object:

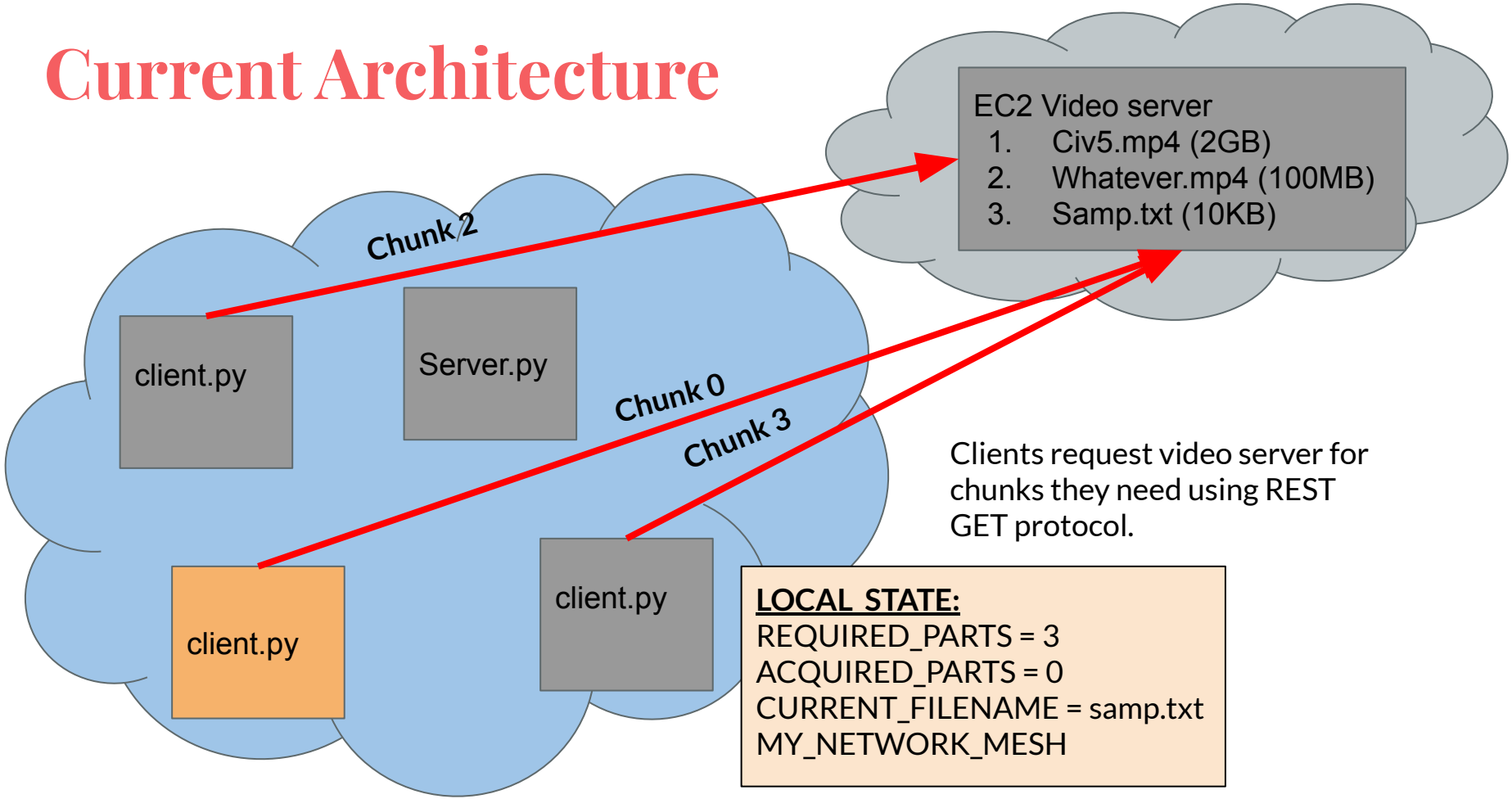
{

"vid_filename": “samp.txt”,
"Requester_ip": 172.0.0.10,
"Requester_port": 23421,
"start": 0,
"length": fair_length | rtt_length | disk_length
"pfilename": samp_0.txt,
"allfilenames":
172.0.0.1→ samp_1.txt,
172.0.0.4→ samp_2.txt,

}

Set REQUIRED_PARTS

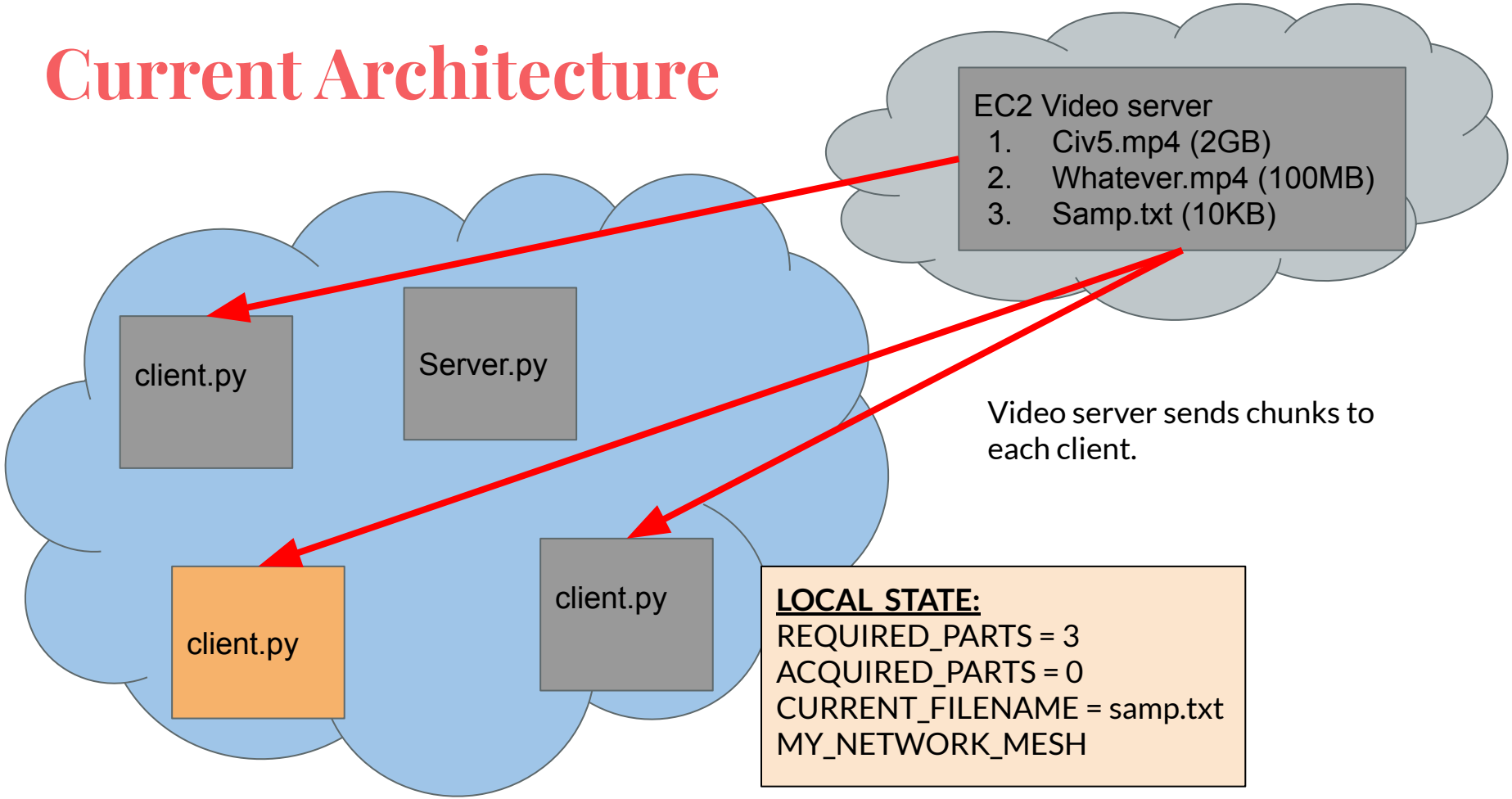
Current Architecture



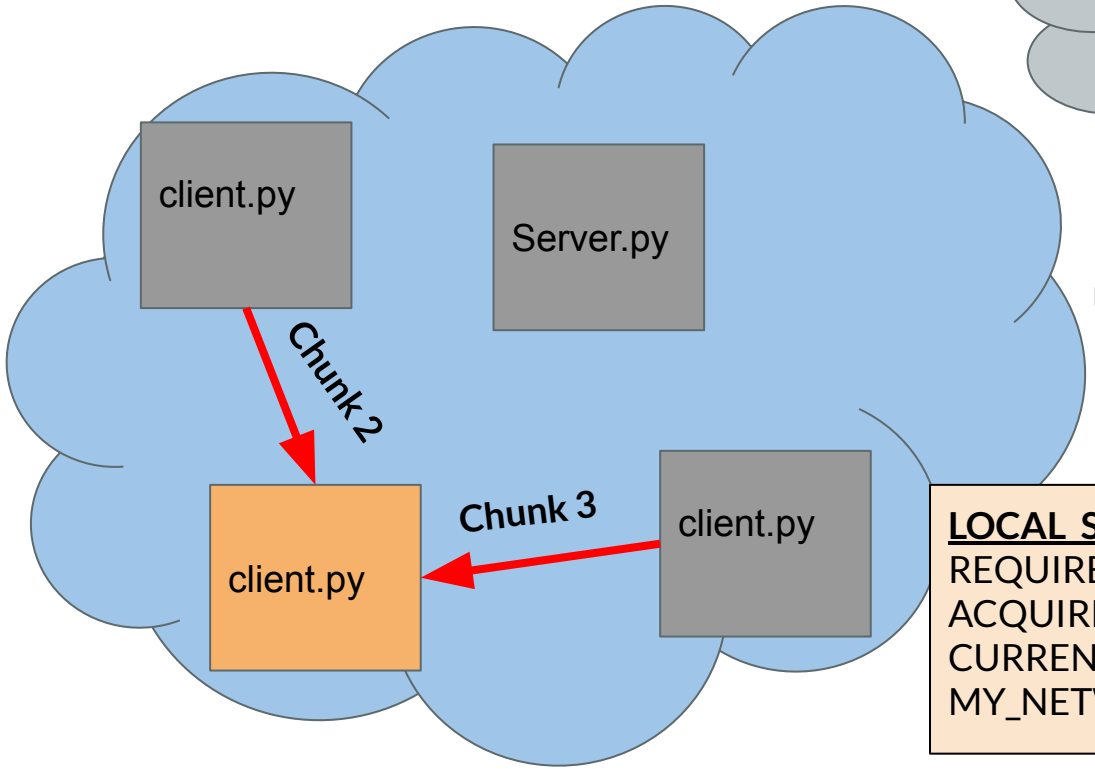
Clients request video server for chunks they need using REST GET protocol.

LOCAL STATE:
REQUIRED_PARTS = 3
ACQUIRED_PARTS = 0
CURRENT_FILENAME = samp.txt
MY_NETWORK_MESH

Current Architecture



Current Architecture



EC2 Video server
1. Civ5.mp4 (2GB)
2. Whatever.mp4 (100MB)
3. Samp.txt (10KB)

Each helper client sends their chunk to original requesting client on to their open video port.

If REQUIRED_PARTS == ACQUIRED_PARTS:
    merge_chunks()

LOCAL STATE:
REQUIRED_PARTS = 3
ACQUIRED_PARTS = 3
CURRENT_FILENAME = samp.txt
MY_NETWORK_MESH

Evaluation plan

- Plan to evaluate in multiple stages for multiple different scenarios:
  - Experiment 1: The Naive Experiments (Fair Partitions) on LAN
    - partition video into equal size of chunks and send to each helpers
  - Experiment 2: The Dynamic Experiments (Unfair Partitions) on LAN
    - partition video dynamically and send to each helpers
    - Partition video based on RTT and available memory
  - Experiment 3: K-Hops Experiment (on WAN)
    - The definition of ‘neighbor’ extends to nodes that is within k-hop distance from the receiver