A remote video streamer

Live Video Streaming with interactive user controls

Protocol Used at Transport Layer - UDP

1) Chosen due to **simplicity** of learning the API.
2) It is a **fast.**
3) **Connectionless** and **unreliable** protocol.
4) The loss of reliability is **not perceivable** by humans and this provides the much needed benefit of speed, which is not provided by TCP.
5) Basic peer-to-peer messages for video content and information about play/pause, and type of filter needed.
6) Helps to understand working at the transport layer.

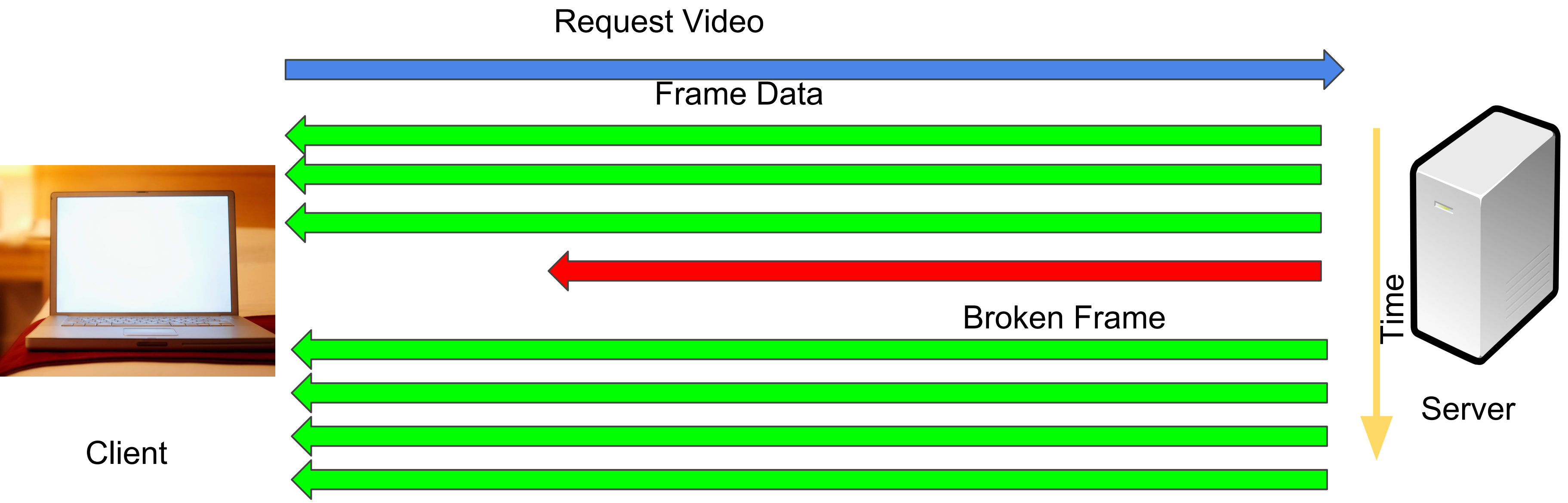
The working of this model

1) Protocol at network layer : **Internet Protocol v4**
2) Client connects to a valid IP:Port which corresponds to the server. This IP: Port combination is needed initially to initiate connection
3) Server providing video streaming service on that port. In subsequent receives, the server obtains the IP:Port of the client.
4) The server opens the video using VideoCapture in OpenCV, and captures the frame in a Mat object. The data of that matrix is then send over the IP protocol, using the UDP protocol at the transport layer.

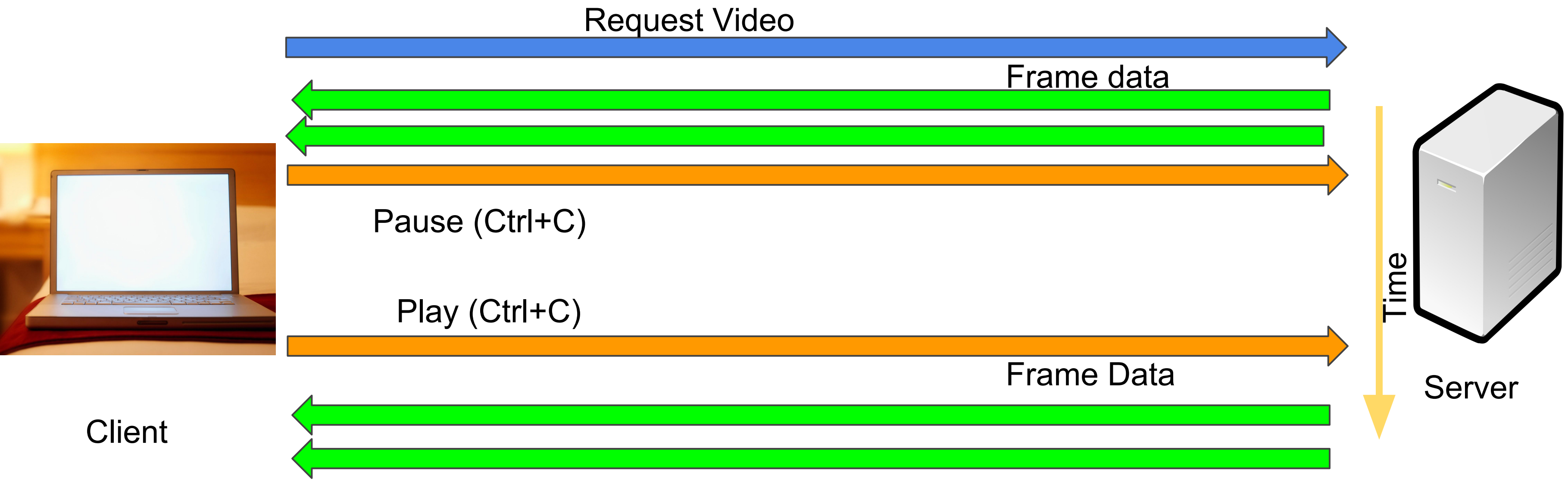
The working of this model

1) The frame is processed by an appropriate filter before sending. This is in accordance with the added feature of filter selection by client.
2) Play/Pause feature. This is implemented using unix signals. Similar, procedure is followed for filter selection.

Basic diagram of live video streaming in our implementation

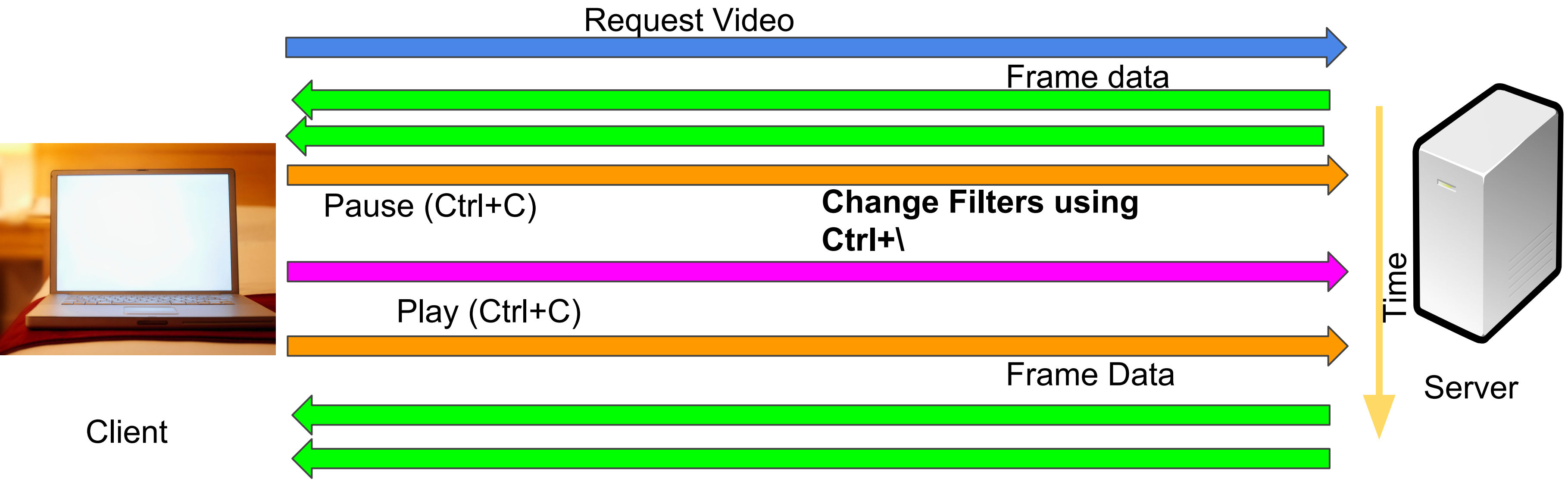


Basic keyboard interaction for Play/Pause



NOTE: The user can use Ctrl+C as a **toggle** to Play/Pause. The signal is send to the server to stop sending data packets.

Basic user interaction for Filter Change



NOTE: The user can use Ctrl+\ to move across four filter types in order. The effect of the filter changes can be seen after the next play/pause.

A feature we can’t demonstrate!

The simple video transfer done can be extended to multicasting and broadcasting for use in video conferences, wherein multiple clients can be connected to each other. This is implemented in our design, with few more lines of code.

Later work

The current implementation deals with only one operating system, namely Ubuntu. We can extend it to include mobile devices and other operating systems.

We have focused only on video transmission right now. Not using UDP but we can include audio too. It will require use of a different protocol.