Development Manual

This is a group chat application. We used RTPBin heavily in our gstreamer pipeline to handle backend video/audio streaming. Our client is built based on Java MVC Swing framework. The communication between client and server is done by sending messages through TCP sockets. Each client itself is also a server, as other clients can request to connect with it to get the webcam live streaming on their end.

The project is separated into multiple packages. The ui package contains code for View components, the util contains classes that handle backend logics of the program, such as TCP message communication, gstreamer initialization etc.

Video Streaming:

Server pipeline:

rtpbin name=rtpbin \ v4l2src ! videorate ! jpegenc ! rtpjpegpay ! rtpbin.send_rtp_sink_0 \ rtpbin.send_rtp_src_0 ! udpsink port=[] host=[] \ rtpbin.send_rtcp_src_0 ! udpsink port=[] host=[] sync=false async=false \ udpsrc port=[] ! rtpbin.recv_rtcp_sink_0

Client pipeline:

rtpbin name=rtpbin \ udpsrc caps=application/x-rtp,media=video,clock-rate=90000,encoding-name=JPEG port=[] ! rtpbin.recv_rtp_sink_0 \ rtpbin. ! rtpjpegdepay ! jpegdec ! autovideosink \ udpsrc caps=application/x-rtcp port=[] \ rtpbin.recv_rtcp_sink_0 \ rtpbin.send_rtcp_src_0 ! udpsink port=[] host=[] sync=false async=false

Audio/Video Streaming:

Server pipeline:

rtpbin name=rtpbin \ v4l2src ! videofilter ! jpegenc ! rtpjpegpay ! rtpbin.send_rtp_sink_0 \ rtpbin.send_rtp_src_0 ! udpsink port=[] host=[] \ rtpbin.send_rtcp_src_0 ! udpsink port=[] host=[] sync=false async=false \ udpsrc port=[] ! rtpbin.recv_rtcp_sink_0 alsasrc ! alawenc ! rtppcmapay ! rtpbin.send_rtp_sink_1 \ rtpbin.send_rtp_src_1 ! udpsink port=[] host=[] \ rtpbin.send_rtcp_src_1 ! udpsink port=[] host=[] sync=false async=false \ udpsrc port=[] ! rtpbin.recv_rtcp_sink_1

Client pipeline:

rtpbin name=rtpbin \ udpsrc caps=application/x-rtp,media=video,clock-rate=90000,encoding-name=JPEG port=[] ! rtpbin.recv_rtp_sink_0 \ rtpbin. ! rtpjpegdepay ! jpegdec ! autovideosink \ udpsrc caps=application/x-rtcp port=[] \ rtpbin.recv_rtcp_sink_0 \ rtpbin.send_rtcp_src_0 ! udpsink port=[] host=[] sync=false async=false \ udpsrc caps=application/x-rtp,media=audio,clock-rate=8000,encoding-name=PCMA port=[] ! rtpbin.recv_rtp_sink_1 \ rtpbin. ! rtppcmadepay ! alawdec ! autoaudiosink \ udpsrc caps=application/x-rtcp port=[] \ rtpbin.recv_rtcp_sink_1 \ rtpbin.send_rtcp_src_1 ! udpsink port=[] host=[] sync=false async=false

A/V Synchronization:

Used RTPBin which handles A/V synchronization.

Resource Admission and Negotiation:

Method: TCP socket communication

Client sends admission request then server calculates capacity and decides whether to accept or not.

QoS Enforcement (Rate Control):

Main measure of QoS is frame rate, the server would adjust the sent rate by client requests.

Session Control:

Method: TCP socket

Detail:

Client sends control signals to server via TCP connection then server changes the pipeline accordingly

Session Monitor:

Display feedback