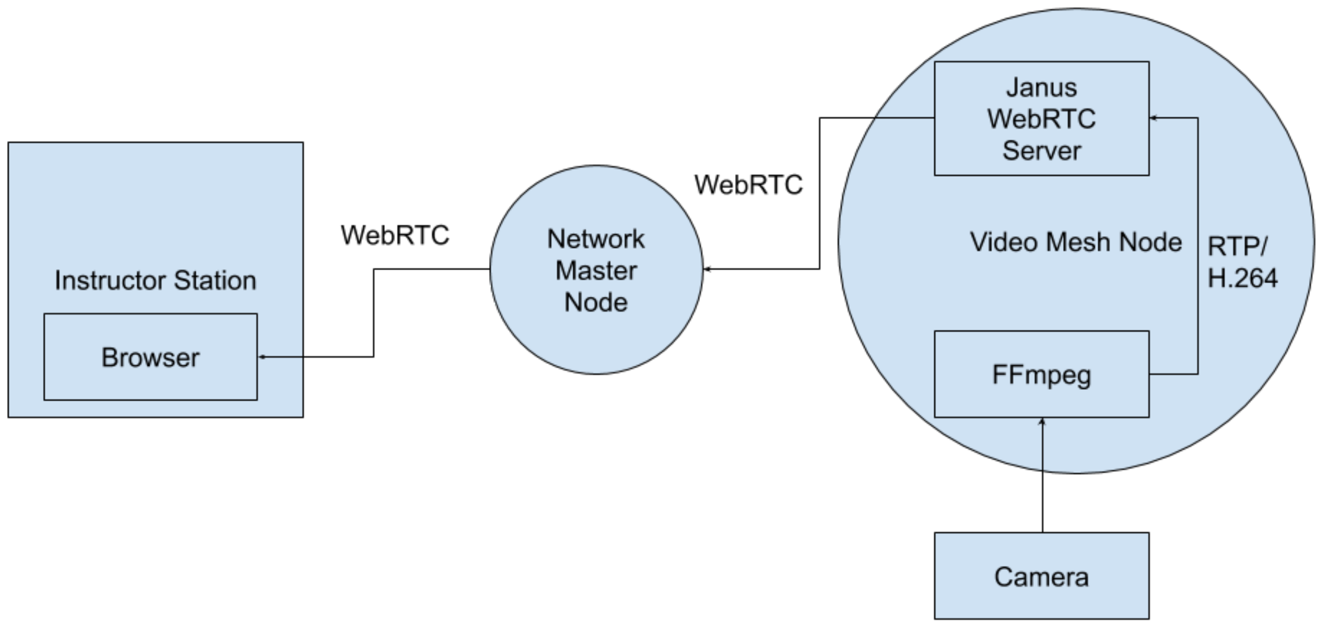
Last edited by **rthedens** 5 minutes ago

Video Streaming Design

Background

The video streams in the mesh network are captured from the Raspberry Pi Camera (V2.1) by an ffmpeg command and streamed across the network to the network master node, which serves the video stream to the instructor station within the user application. More specifically, the ffmpeg command running on the Pi uses Video4Linux2 (v4l2) to capture the video stream from the Pi camera. The ffmpeg command encodes the stream to the H.264 codec and streams it via the RTP protocol to an instance of the Janus WebRTC Gateway that is running on the same Pi. The Janus WebRTC Gateway provides the stream to the user application (via WebRTC). The user application webpage for each camera node will display the node's video feed.



Video Streaming Software Design

This section explains the design of the Video Streaming Software, found in the repository directory CameraNode. Each file is explained in alphabetical order (excluding the README).

build.sh:

- This script will run the command `docker build -t camera_node .` which builds a Docker container with the name "camera_node" using the Dockerfile in the current directory.

commands.sh:

- This file is copied into the Docker container as part of the build process and executed when the Docker container is run.
- The file contains two main commands which are run in the background, as they need to run simultaneously.
    - The first command runs Janus. The -F options specifies the configuration folder. As compiled, Janus will run the streaming plugin using the copied-in config file.
    - The ffmpeg command starts the video stream.
        - The -f option acquires the stream directly using the v4l2 driver for the Pi Camera
        - The -input_format specifies the image encoding format. yuyv422 is the raw format for Pi Camera output.
        - The -video_size specifies the resolution of the video. hvga corresponds to 480x320. Note that ffmpeg does not support arbitrary resolutions. See (https://ffmpeg.org/ffmpeg-utils.html) for details. Not all the listed resolutions will work either, so it may take some guessing to find the ones that do. 4cif and cif are known to work as well.
        - -framerate specifies the framerate.
        - -i /dev/video0 specifies the video device to read from (which will be the Pi Camera). As long as there is only one camera device attached to the Pi, this won't be a problem. If this needs to be changed, one options is here (https://www.raspberrypi.org/forums/viewtopic.php?t=70274)
        - -c:v specifies the codec/encoder to use. h264_omx is nice because it uses the onboard GPU to do the video encoding, which is fast/efficient.
        - -profile:v baseline chooses a specific H.264 profile. This profile is purported to be better for low bit-rate applications, which is generally a good thing considering that the mesh network has limited bandwidth.
        - -b:v 320k specifies a 320 kbps bit rate for the stream. This parameter, along with framerate and resolution, needs to be tuned to match the available bandwidth of the network. Note that ffmpeg will automatically compress the video when you set the bitrate too low for the resolution and framerate, which can cause poor quality video. Note also that when the bitrate is set too high for the current network (i.e. there is not enough bandwidth), the video will become choppy and freeze until enough bandwidth is available.
        - -bf 0 is related to B frames. These are not supported by the baseline profile and are turned off (hence the 0)
        - -flags:v +global_header causes the use of global packet headers instead of individual packet headers
        - -bsf:v "dump_extra=freq=keyframe" is related to the above. Note that removing this option and/or the above one will break the stream. See (https://ffmpeg.org/ffmpeg-bitstream-filters.html) for some vague description of what these do.
        - -max_delay 0 disables the max delay feature. Setting the max delay is related to a reordering feature for UDP video packets. Since we really don't want to waste time/effort reordering the video packets, it is disabled by setting it to 0.
        - -an specifies that no audio should be captured. This isn't necessarily important, since the camera doesn't have an attached microphone.
        - -f rtp rtp://127.0.0.1:8004/ specifies that the output should use the RTP protocol and use the loopback interface on port 8004. This delivers the video stream to Janus, which listens on port 8004.
    - There is a third command, /bin/bash, which opens the bash shell in the foreground. This can be helpful for debugging inside the container.

Dockerfile:

- The Dockerfile contains the instructions needed to build the Docker container for video streaming.
    - First, balenalib/raspberry-pi-debian:buster is used as the base image. This is an image freely available on Docker Hub
    - Updates/upgrades to existing packages are performed via `apt-get`
    - A large number of packages are installed via `apt-get`. These are required dependencies for Janus, as well as the ffmpeg and picamera packages
    - The libsrtp package is a Janus dependency. Janus requires a version (2.3.0) that must be built from source. The `--enable-openssl` option is crucial when building this package, or else Janus will give errors.
    - Janus WebRTC Gateway must be built from source as well. Janus contains many features we do not use, such as websockets and data channels. We also specify the /opt/janus prefix, which dictates where the Janus binary and config files will be placed.
    - Within the CameraNode directory there are two resource files that must be transferred into the container.
        - The first to copy in is the janus.plugin.streaming.jcfg file. This defines the configuration for the streaming plugin in Janus.
        - The second to copy in is the commands.sh file. This gives the commands that are run within the container.
    - usermod -a -G video adds the current user (root) to the video group, while keeping root in the root group.
    - modprobe bcm2835-v4l2 loads the driver module for the Pi Camera
    - The CMD statement runs the commands.sh script inside the Docker container

janus.plugin.streaming.jcfg:

- This file defines the configuration for the streaming plugin in Janus. Note that multiple streams would be supported, with different id numbers. However, we only need one stream per node, since each node has a single camera.
    - The type must be rtp, matching the protocol used in ffmpeg
    - The id and description are arbitrary, as only one stream is sent from each node (Janus is also capable of serving many WebRTC streams simultaneously, but we do not use it that way). The id will be referenced in the user application code.
    - We specify video with no audio, on port 8004 (also referenced in the user application)
    - videopt specifies an RTP payload format. 96 is the first available dynamically-defined payload type. videortpmap specifies H.264/90000, which

    indicates that the H.264 codec is used. See (https://tools.ietf.org/html/rfc6184) for more info on H.264 over RTP, but note that this follows the sample h264 stream config file provided by Meetecho (makes of Janus) here (https://github.com/meetecho/janus-gateway/blob/master/conf/janus.plugin.streaming.jcfg.sample.in).
  - videofmtp is a tricky one. It has to do with the Session Description Protocol (SDP) that is used with RTP to define parameters of the stream. The arguments here follow the sample streaming jcfg file provided by Meetecho.

start.sh:

- This file runs the Docker container with the command `docker run -it --network=host --device=dev/vchiq --device=/dev/video0 camera_node`
  - -it enables interactive mode, in case you want to use the bash shell in the container
  - --network=host gives the Docker container access to its host's network interface(s) (i.e. the physical Raspberry Pi). This allows the stream to travel across the network.
  - --device=/dev/vchiq allows access to the GPU device
  - --device=/dev/video0 allows access to the Pi Camera
  - camera_node is the name of the container

User Application Elements

This section explains how the video stream is acquired and displayed by the User Application.

The video stream is picked up by the user application, which displays it in a webpage. There are some files that are closely tied to Janus. These are (in NetworkMasterNode/UserApplication/src/app/network-manager/component/common/live-video-player) live-video-player.component.ts, live-video-player.component.html, janus.js, adapter.min.js, and (in UserApplication/src/app/network-manager/component/pages/node-detail) node-detail.component.html.

- live-video-player.component.ts contains a method startVideoStream that invokes the Janus JavaScript API to acquire the video stream. The function of this code is similar to that found in the Janus streaming plugin demo web app. This page (https://janus.conf.meetecho.com/docs/JS.html) contains documentation for this process. Generally speaking, the following is the process for the video negotiation. Note that the communications are done with the Janus WebRTC Gateway server application, which is a third-party application that is configured by the user but whose communication pattern is already determined (not designed by anyone on this team).
  - First, we have to initialize the Janus library
  - The callback that follows creates a new Janus object connected to the CameraNode via the streamUrl
  - If the connection succeeds, the streaming plugin is attached, and a request to watch the stream at streamId is sent.
  - When a response is received, the streaming plugin requests the start of the stream.
  - When the remote stream is received, we get the video track and display it.
- live-video-player.component.html contains the video element that is displayed in the user application. If the autoplay option is removed, the user will have to right click and select "play" on the video element in the browser to be able to see the video.
- janus.js is the Janus JavaScript library that facilitates access to the WebRTC video stream from the user application. Its functions are called by the code in live-video-player.component.ts.
- adapter.min.js is required by janus.js. It is a "shim" that makes it easier for janus.js to use WebRTC features. The code is found on GitHub at (https://github.com/webrtc/adapter).
- node-detail.component.html is the page that contains the video element. It provides the hard-coded stream ID (universal to all camera nodes) and camera node stream URL (using the camera node's IP address) to the element so the video can be displayed.

References

Here are some references that may be useful:

- Meetecho Janus Homepage: https://janus.conf.meetecho.com/docs/

  - JavaScript API Documentation: https://janus.conf.meetecho.com/docs/JS.html
  - Streaming Plugin Documentation: https://janus.conf.meetecho.com/docs/streaming.html
  - Janus Gateway on Github: https://github.com/meetecho/janus-gateway
    - See the demo pages here (including the streaming demo): https://github.com/meetecho/janus-gateway/tree/master/html
- FFmpeg Homepage: https://ffmpeg.org/

  - ffmpeg command main documentation: https://ffmpeg.org/ffmpeg.html
  - FFmpeg's documentation can be difficult to find. Use Google as well (but remember that "-" tells Google to find results without the following word, which is annoying for command line options)
- Docker: https://www.docker.com/

- Docker Engine installation instructions (Debian is most relevant): https://docs.docker.com/engine/install/