SANGOMA

A History of Video in Asterisk

Joshua C. Colp
Asterisk Technical Lead
Sangoma Technologies

Hi! I’m Josh! (or file/jcolp)
Twitter: @joshnet

SANGOMA

15+ Year Developer
Creator of many things

SANGOMA

Now let us dive into video...

Video in Asterisk

- How media works
- The addition of video in the beginning
- Timestamps
- Re-ordering / loss
- Format attributes
- Video streams
- Video conferencing
- Mid-call re-negotiation

Media and The Outside World

- Channel driver (such as chan_pjsip) responsible for setting up media in protocol specific way.
- Channel driver negotiates media according to configuration (allow / disallow) and sets this information on the channel
- Provides protocol translation from the Asterisk representation to the protocol specific method

Media and The Outside World

- Channel driver provides and consumes media using what are called frames.
- Frames contain all the information about a segment of media. The media itself, how much there is, and other information.
- Asterisk itself only deals with media in the form of these frames.

Media Through The Core (Into Asterisk)

- Channel driver reads in protocol specific manner the media and converts it into an Asterisk frame.
- Channel driver notifies Asterisk that it has new media and provides it as the frame.
- Asterisk reads the frame and returns it to what is currently handling the channel, or stores it away if there are other frames pending.
- Up to the consumer to do what may be needed with it.

Media Through The Core (Out of Asterisk)

- Asterisk calls into channel driver providing the frame with media.
- Channel driver converts the frame into the protocol specific representation.
- Channel driver sends the media using the protocol specific manner.

But how do you know what codec the media is?

How Media Codecs Were Represented Originally

- A codec was a bitmask value and nothing more.
- Frames had this set on them to indicate what codec was in use.
- Everything revolved around this value.
- Channel driver converted to/from the bitmask value.
- Frame comes in, frames goes out.

Video Appears

- Video became another bitmask value with no additional information.
- Special logic added to differentiate audio and video for media path.
- Worked essentially the same as audio, just with no transcoding.
- Few codecs supported (H.263, H.263+, H.264)
- Video worked beautifully!

Sounds too easy...

Video Is Hard (Attributes)

- Video codecs ideally don't work in a "this is H264" way
- Attributes and parameters are negotiated to better convey what is capable and desired
- This allows higher quality video to occur
- In SDP this is done using fmtp lines
- Asterisk originally did not have support for this

Video Is Hard (Attributes)

- Codec representation was changed to numeric values, no more bitmask
- A higher level concept called a format was added which included attributes and the codec numerical value
- This worked, but proved slow due to having to copy all the information constantly
- A new way was needed

Video Is Hard (Attributes)

- A format remained but an immutable version was used, removing the need to copy except when changing
- This proved substantially faster and is the method used today
- What's this mean? 1 ulaw format is used, 1 alaw format, etc
- It is only when attributes are used that a new format is created once when negotiated - and then used for call

Video Is Hard (Attributes)

- Passing formats around means on outgoing calls you know what attributes were asked for in the incoming leg
- Format attribute modules provide the translation and understanding of attributes for each format
- SDP parsing/generation is also provided
- End result is a better video quality stream

Well that wasn’t THAT bad…

Video Is Hard (Timestamps)

- In practice once you start taking video seriously timestamps are important, you have to pass them through
- They allow the receiver to know how to properly play out the video
- They also allow a video frame to span multiple RTP packets which is used for large sizes

Video Is Hard (Timestamps)

- Timestamps were not originally preserved for video, but instead were re-generated
- This caused issues where re-generation resulted in an incorrect value
- Video stream suffered as a result

Video Is Hard (Let's Do Timestamps)

- Support for setting/using timestamps for video frames was added to RTP
- Core preserves this information end-to-end, and granularity of timestamp ensures it is correct on both sides
- End result an even better video experience

Er, is there more?

Video Is Hard (Sequence Numbers)

- Sequence numbers in RTP increment for each RTP packet sent
- Allow receiver to re-order packets and to detect loss
- Very important for video as media has to be decoded in order
- Was not originally passed through

Video Is Hard (Let's Preserve Sequence Numbers)

- Support for setting/using sequence numbers for video frames was added to RTP
- Core preserves this information end-to-end
- End result an even better video experience

Video is a never ending journey of improvement

Video Streams (Don’t Cross The Streams)

- Streams are a representation of a flow of media, they may be unidirectional or bidirectional
- Asterisk originally only conceptually had 1 audio and 1 video stream
- This conceptual aspect was due to the work done originally to separate out the flow for audio and video
- No actual code representation of these streams

Video Streams (Okay, Let's Add Streams)

- Asterisk 16 added an actual defined API for streams and a representation
- Check out "core show channel" to see stream information
- Channel drivers must be updated to provide and be aware of streams if they want to use them
- Asterisk core transparently allows old channel drivers to work using old APIs by providing representation for the conceptual streams

Video Streams (What's A Stream To Us?)

- Named entity
- Has a state (send and receive, send only, receive only, etc)
- Formats
- Stream attributes
- Can be part of a group of streams, this is called a topology
- Being part of a topology also bestows a number to the stream, which allows efficient usage

Video Streams (What Supports Them?)

- PJSIP channel driver and Asterisk RTP implementation
- ConfBridge dialplan application
- Dial dialplan application
- Simple and RTP bridge modules

What things can be done with streams, though?

Video Conferencing (Outbound Re-negotiation)

- Asterisk 16 added support for SFU video conferencing to the ConfBridge dialplan application
- SFU video conferencing is an outgoing stream to each channel for each participant
- To support SFU video conferencing we needed the ability to change/add/remove streams on a channel
- The stream API added a defined mechanism to do this but the channel driver does have to support it

Video Conferencing (Outbound Re-negotiation)

- Applications (such as ConfBridge) can request, asynchronously, that a new set of streams be negotiated on a channel
- Application receives notification if this occurs and what the negotiation result was
- It is up to the application to then use this information as it needs

Video Conferencing (Outbound Re-negotiation)

- ConfBridge uses this to request new outgoing streams when a participant joins, or to remove a stream when a participant leaves
- When negotiation completes ConfBridge maps streams so that video flows, from each participant to every other participant

What if I want to add video later, or do screen share while showing my webcam?

Video Conferencing (Inbound Re-negotiation)

- Asterisk 18 added the ability for an endpoint to re-negotiate its streams with Asterisk
- An endpoint can add, remove, or modify the streams of a channel
- Asterisk application receives notification that this has happened and can react as it needs

Video Conferencing (Inbound Re-negotiation)

- ConfBridge compares the new streams to the old ones to determine which have been removed/added/changed
- Other participants are re-negotiated as appropriate based on what has changed
- This removes the limitation of a single video source from a channel.
- Any number of video sources can come from a participant with this, and they can be added/removed as needed

That’s cool, but what about normal calls?

Two-Party Call (The Beginning)

- In the beginning streams could only be negotiated at the start of a call
- You had to start a call with both audio and video, and could not change your mind during the call
- Trying to add video mid-call for example would fail with no video

Two-Party Call (Change It Up)

- In Asterisk 18 alongside the ConfBridge changes we now support mid-call re-negotiation on two-party calls
- When in a two-party call if a party adds video then the other side is re-negotiated, the same applies if video is removed
- Multiple video sources are supported, such as webcam and screen share just like ConfBridge

Some cool mentions

Asterisk 18 Video Improvements

- Local channel support for streams, and thus re-negotiation
- Performance improvements for passing video around

Video is cool, but hard

This Isn’t Your Asterisk 1.4

- Video in the Asterisk of today is vastly different to how it was and has come a long way
- A lot of work (some I haven’t even mentioned) has gone into improving video, but always more to do
- Keep watch on releases for potentially more video improvements in the future

Thanks!
Follow me @joshnet on Twitter

SANGOMA