Delivering live video streams to YouTube over IPv6

The live web cam which is located on the Koningsplein in Amsterdam has been running for a decade in various shapes and forms.

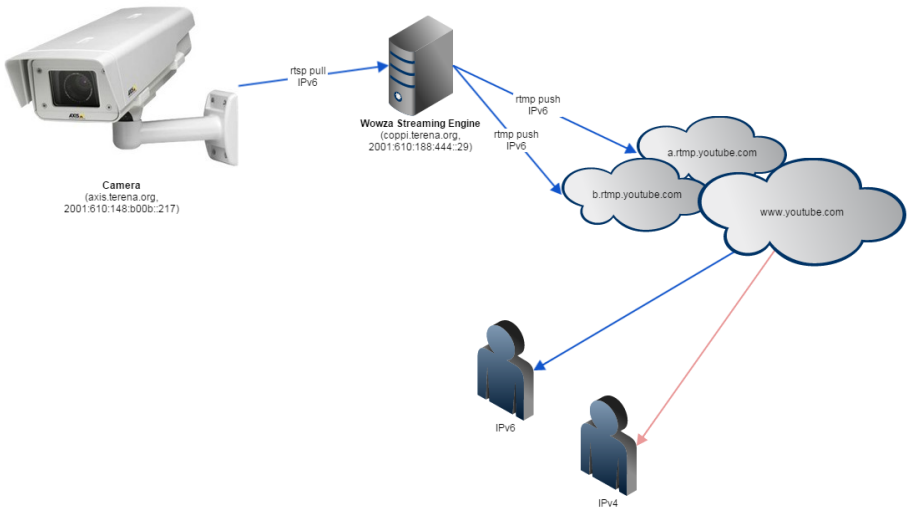
In all of these set-ups we used to run the actual content delivery systems ourselves, from our own networks.

The latest incarnation is a cloud-based set-up which uses Wowza Streaming Engine to relay a video stream to YouTube and takes advantage of the new Youtube Live features to deliver the content to users.

This set-up has several benefits over the systems that we used to run locally:

- **More scalable.** YouTube can distribute video much better than any locally run system. In our case we had a limit of 1GB/s, so anything over a few hundred visitors would mean the video stream got jerky and dropped out. This used to happen when someone posted a link to the stream on a popular forum such as GeenStijl, thereby DoSing the video stream 😉.
- **Multiple bit rates.** We used to do transcoding into various lower bit rate streams, so that users on slow connections (mobile phones) could also watch it. This required an expensive plugin, which needed careful configuration, and which put a significant CPU burden on the streaming server. Google does this now, so no plugin needed, no CPU load, and less complex configuration.
- **Youtube works on many different devices.** Many mobile devices have dedicated apps for YouTube, which makes sure it plays well on those devices.
- **Less support issues for end users.** YouTube is the defacto standard for viewing video content on the Internet, so end users will have few issues with it.
- **Users can post comments.** While it is possible for users to comment on the stream, we had to disable it because eventually the comments filled up with trolling messages.
- **VCR capability.** You can actually skip back in time by 4 hours, which is handy if you missed some action 🙂.
- **Production side can be purely IPv6** (despite the general rule "What an engineer *can* do, is rarely what he *should* do "). Both the camera and the relaying system do not need to be publicly available. The camera just needs to communicate with the relaying system, which in turn needs to talk to YouTube. Since YouTube is dual stack, this means that both the camera and the relaying system can be purely IPv6.

Overview



Configuration of the camera

The camera is an Axis Q1755, which is mounted outside our office on the 4th floor of Singel 468 in Amsterdam.

The streaming server will request an RTSP stream, which is done over TCP port 554. So make sure the port isn't firewalled off.

Important settings:

- Create a dedicated user on the Axis (System Options -> Security -> Users) that has only viewer permissions, and assign a password to it. Since it is a machine, you can use `pwgen -s 64 1`.
- Make sure the RTSP server is enabled (System Options -> Network -> TCP/IP -> Advanced). The default port is 554.
- Audio setting (Video & Audio -> Audio Settings): Half-duplex, AAC encoding, 16 KHz, 64 kbit/s.
- Enable audio (Video & Audio -> Video Stream -> Audio). This sounds obvious, but if you deselect this, Google will report "NO DATA".

Configuration of the streaming server

The server is a VMware VM called **coppi.terena.org**, which runs Ubuntu 14.04 LTS and is located in the SURFnet datacenter in Amsterdam.

It runs Wowza Streaming Engine software (WSE). We have a perpetual license for the software. The management interface is HTTPS/Apache.

Follow the instructions on http://www.wowza.com/forums/content.php?217 and install WSE.

This howto explains how to configure WSE for YouTube live streams: https://software.intel.com/en-us/articles/intel-inde-media-pack-for-android-tutorials-video-streaming-from-device-to-youtube

I had some trouble getting WSE to work properly in an IPv6-only environment, turns out you have to force Java to explicitly use IPv6.

See http://www.wowza.com/forums/showthread.php?37909-PushPublishRTMP-doesn-t-work-with-IPv6 on how to do that.

The RTMP push out to YouTube is done to two host names (a.rtmp.youtube.com and b.rtmp.youtube.com). Initially I had put these hostnames in my ip6tables configuration, but eventually the cam went offline. Turns out that the IP addresses are not stable - which is perfectly valid. But because the ip6tables rules only get loaded and thus resolved at boot time, eventually the rtmp.youtube.com host names would start resolving to something else, and the outgoing stream would get blocked by the ip6tables script. Since there is no such thing as "Google IP ACL" I had to open up outgoing TCP port 1755 to the entire Internet. Not a big deal, but something to keep in mind.

Gotchas

- YouTube Live is only be available on channels that have at least 100 subscribers. I'm guessing that this is done by YouTube to prevent random users from clogging up their network with sleeping cat webcam streams without there being any viewers for it. Either way, the TERENA YouTube channel already had 100+ subscribers so we were all good to go.
- YouTube Live is not available to users in Germany: http://en.wikipedia.org/wiki/Blocking_of_YouTube_videos_in_Germany. This is a political issue that we can't solve.