Controlling video bit rate and video encode quality

Introduction

Controlling video bit rate and adjusting or controlling the quality of encoded video are two separate but related functions that the video codec can attempt to manage in order to achieve better quality of service.

Controlling video bit rate

Every channel has a bit rate limit. However, in many situations, the channel's bit rate limit is sufficiently above the codec bit rate requirement that the effects of the limited bit rate can be ignored. One just assumes, or makes it a requirement, that the channel will accept bits for transmission at whatever rate the codec wishes to send them. This is the default assumption for both audio and video channels in openH323.

When the channel bit rate limit is, in fact, close to or below the codec sending bit rate, the usual consequence is lost or dropped packets. That is, packets are sent from the source end point but never arrive at the destination end point. The channel is limiting bit rate by dropping packets. For both audio and video, the effect of dropped packets is to degrade audio or video quality. Depending on the particular encoding audio codec, the effect of a few dropped packets can range from insignificant and unnoticed to very significant with very noticeable degrading of audio quality. For video, the effect is similar depending on the particular encoding codec. However, video information is generally not as important for effective communication as audio. Consequently, the effect of dropped video packets can usually be ignored whereas something must be done to minimize dropped audio packets.

Why control the video bit rate if dropped video packets are not especially important? The problem is that audio and video packets will usually pass through the same router when using an IP based channel. If this total bit rate exceeds the IP channel bit rate limit and the router drops packets, it usually does not distinguish between audio and video packets. Consequently, audio packets can be dropped because the video bit rate exceeds the router bit rate limit. Under this condition, the degraded audio quality can essentially make the channel unusable for communication.

The solution to this problem is to manage the total bit rate and keep it from exceeding some maximum. Since video data generally uses a much larger fraction of the IP based channel capacity than audio data, the total bit rate can be managed effectively by controlling only the video channel bit rate.

The easiest way to control video channel bit rate is to insert a variable delay between transmitting each packet. The delay time is calculated to ensure that the next packet will not be transmitted until sufficient time has elapsed that the average bit rate (bits in packet / time before next packet is transmitted) does not exceed the bit rate control target for the channel. This is the approach taken when video bit rate control is enabled in openH323. The channel is limiting bit rate by inserting additional time for transmission. Note that if one is clever about it, useful work can still be done during the time interval between packets.

If nothing else is done, the immediate consequence of controlling the video bit rate at less than the channel capacity is a drop in video frame rate. Depending on how low the target video bit rate has been set, the frame rate could easily drop below 1 frame per second. Users, however, will generally prefer a higher frame rate, even if the video quality is degraded to achieve it. This leads naturally into the following discussion about reducing video quality in order to gain a faster frame rate.

Controlling video encode quality

As discussed previously, the need for controlling video encode quality arises when the video bit rate is limited or controlled in a way other than simply dropping packets. If there is no video bit rate limit, there really is no need to control video encode quality. The encoder can run at maximum encode quality all the time.

There are three parameters that control video quality in the H261 codec, frame rate of delivered video, background fill rate, and quantization level used during encode (q, varies from 1 to 31). With a fixed video bit rate, it is not

possible to vary these three parameters arbitrarily. Instead, one must find a set of values which results in a video bit rate which both satisfies the target video bit rate requirement and which also maximizes the perceived video quality of the result. This can be thought of as a constrained optimization problem, but note that the solution only applies to one particular frame of video input. In general, the optimum encoding parameters will change with every video frame.

The OpenH323 H261codec uses the following strategy in attempting to achieve maximum perceived video quality. This strategy is simple and no doubt can be improved. Note that this strategy is only active if video bit rate control is active.

1. The video codec measures and keeps track of the average bit rate achieved while transmitting. This ensures that the codec becomes aware of changes in the actual video bit rate through the channel, no matter what causes these changes. It could be due to active bit rate control by the codec or it could be due to some other reason.
2. The video codec calculates a target value for frame size in bits per frame based on the user requested video frame rate and the measured average video channel bit rate.
3. Before encoding a new frame, the codec calculates the difference between the size of the previous frame and the target frame size. This difference is used to adjust the encode quality up or down in attempt to achieve the target frame size for the next encoded frame.

After encoding, the actual frame size will hardly ever be equal to the target frame size. Transmitting this frame causes a short term variation in the video frame rate. In summary, the encode strategy

1. accepts short term variations in the frame rate of delivered video,
2. adjusts video encode quality in an attempt to keep the frame rate close to a user set target frame rate,
3. does not adjust the background fill rate.

Complications and Open issues

Frame rate control by video grabber

The strategy for controlling video encode quality makes the implicit assumption that decreasing the number of bits per frame will result in an increase in the frame rate. This assumption does not hold true if the frame rate is limited or controlled by some other mechanism. Specifically, the frame rate could be controlled by the video frame grabber.

If the user requested video frame send rate is set higher than the frame rate from the video grabber, the video codec will reduce encode quality in an attempt to decrease the size of each encoded frame and increase the transmitted frame rate. However, the average bit rate achieved while transmitting decreases but the frame rate stays the same (because it is controlled by the video grabber). The codec again decreases the encode quality, the encoded frame size decreases, the average bit rate decreases some more, but the frame rate stays the same. This process continues until the encode quality cannot be decreased further and poor quality video frames are transmitted at the grabber determined frame rate. There is plenty of capacity available in the video channel but it is not being used because the codec is measuring and adjusting to a low average channel bit rated due to the control exerted by the video grabber.

The problem can be illustrated by viewing one of the moving test patterns and setting a video send rate above 10 frames per second. The test pattern generator limits the frame rate to 10 FPS.

The solution is to adjust the way the video codec measures and keeps track of the average bit rate achieved while transmitting so that it ignores any time that passes while the encode thread is blocked by the video grabbing function. This is done by recording a time stamp before and after the call to grab a video frame (including time to pre-encode the frame) and subtracting the elapsed time from the total time to transmit that frame. That way, the measured video bit rate measurement is not affected by the time spent while blocked.

Effective averaging time for bit rate control

As described earlier, video bit rate control is accomplished by inserting a variable delay between transmission of each video packet. The delay time is calculated to ensure that the next packet will not be sent until sufficient time has elapsed that the packet bit rate (bits in packet / time before next packet is transmitted) does not exceed the bit rate control target for the channel. This approach effectively controls the average bit rate over the time period required for transmission of one video packet, or the packet average bit rate. The actual bit rate carried by the channel is most likely much higher during the early part of the packet transmission period followed by a zero bit rate for the later part of the transmission period.

For time periods longer that the packet transmission period, the average bit rate will always be equal to or less than the packet average bit rate.

Usually it is desirable to control average bit rate using a time period longer than the packet transmission time.

Implementation details

Control variables and default behavior

1. targetFrameTimeMs - contains the target time interval between video frames, in milliseconds, used when video encode quality control is active. The reciprocal of frame rate is used so that frame rates below 1 frame per second can be requested without requiring a floating point data type. This variable is initialized to 167 ms corresponding to 6 frames per second.
2. bitRateHighLimit - contains the maximum transmitting bit rate, in kbps where 1 kbps = 1024 bits per second, used when video bit rate control is active. This variable is initialized to 2048 kbps.
3. videoBitRateControlModes - contains the currently active video control modes. This variable is initialized to none. Video bit rate control and adaptive video quality control can be enabled or disabled independently.
4. videoQuality - contains the current video quality parameter which can take values between 1 (best video quality) and 31 (worst quality). This variable is initialized to 9. When video encode quality control is active, the value will adjusted between videoQMin and videoQMax by the quality control algorithm.
5. videoQMin - contains the minimum value allowed for videoQuality. This variable is initialized to 1.
6. videoQMax - contains the maximum value allowed for videoQuality. This variable is initialized to 24.

Use with ohPhone

The following command line switches have either been added to ohphone or are used differently from before:

```
  --videotxminquality n   : Select video quality lower limit,(def 1). 1(best)<=n<=31
                            A value of 4 works best for NetMeeting
  --videosendfps n        : Target minimum number of video frames sent per sec 0.001<6(def)<30
  --videobitrate n        : Enable bitrate control.   16< 256(def) <2048 kbit/s (net bw)
```

For example, the following command line,

```
ohphone -nlttt -o trace.txt --videodevice fake --videoinput 2 --videoreceive sdl --videobitrate
32 --videosize large --videotest
```

shows a large bouncing squares test pattern with a 32 kbps bit rate limit. The video quality is degraded (higher q number) in an attempt to maintain at least 6 video frames per second.

To run ohphone with netmeeting, use the switch command line "`--videotxminquality n`". This will ensure that video quality never falls below 4. Netmeeting has problems displaying video at qualities better (lower) than 4.

```
ohphone -fTnl --videodevice 0 --videoreceive sdl --videotxminquality n --videobitrate 320 --
videosize large
```