Latency in IP Video Networks

Tackling Delays from Scene to Screen

Table of contents

1 Introduction

In IP video networks, scene content must be electronically processed, prepared as data packets for transmission over the network, and electronically processed again for display on a screen. These processing steps allow resolutions and image enhancements outrunning any analog system – with virtually unlimited distance to travel. But unlike an analog video signal which travels at near light speed, they require time, gradually inducing delays which sum up to a certain latency.

In some applications, latency can be annoying or even become a safety issue.

An IP video transmission is processed through a chain of components starting from the scene, which is the camera sensor, to the screen, which is a monitor device. Further components are image processing pipeline and encoder, the network which consists of links of certain bandwidth, and network infrastructure like switches or routers. Each of these add a little to the overall latency. Though we cannot avoid this technical fact, we can at least ensure not to waste valuable processing or transmission time needlessly.

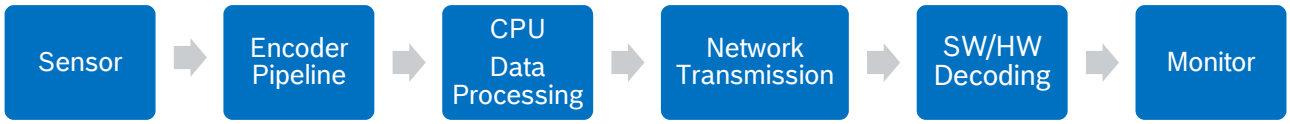

Figure 1: IP Video processing chain

2 Analysing the chain

At the level we look at, the IP video processing chain is at least similar for different platforms or systems.

The first three components typically reside inside an IP camera, which is in the focus of this document. In a video encoder, the sensor is replaced by the analog video input.

In the following, the characteristics for each of the components are analyzed.

You will get tips to achieve correct measurements and, where applicable, also optimization hints.

2.1 Sensor

Sensor

With a frame rate of 30 fps, for example, reading out the sensor takes 33 ms to capture a full image. This is caused by the progressive scan used by most CMOS sensors that scan rapidly through the scene rather than take a snapshot. If the last row would be used as reference the read-out delay would nearly be zero milliseconds.

To come close to zero, arrange the camera to capture the reference stamping from the bottom of the image. By measuring delay in relation to the last sensor row you get the true latency without adding the progressive scan time.

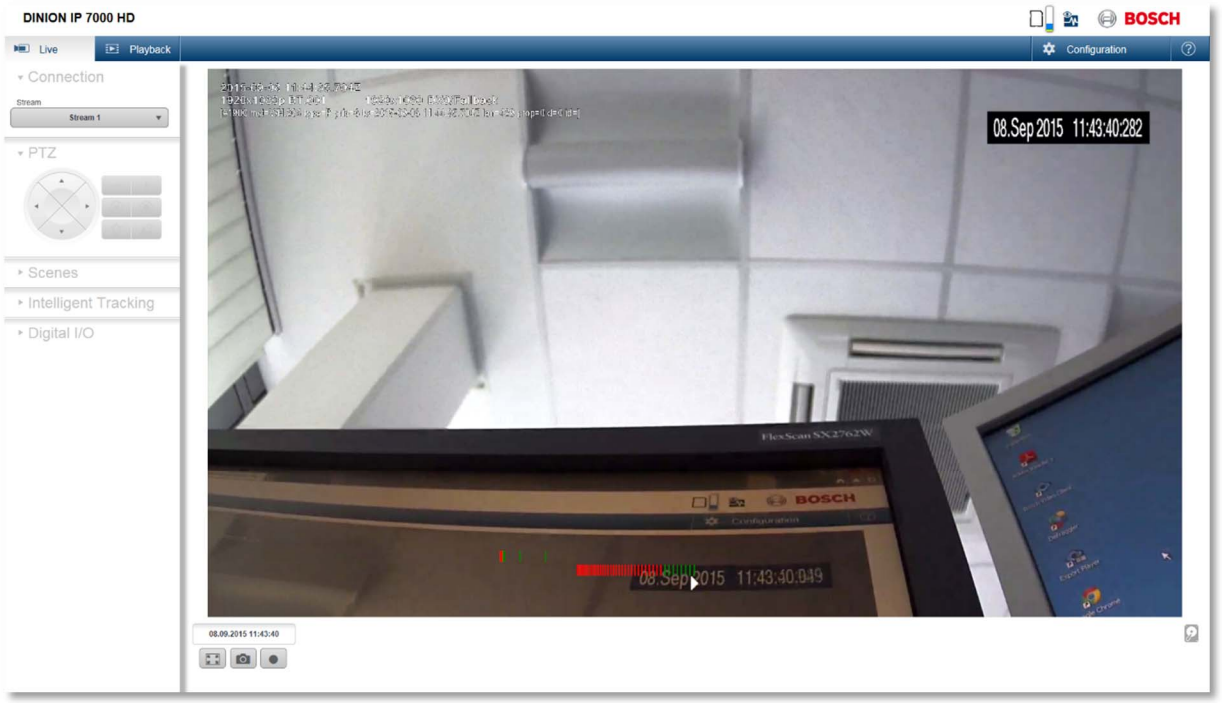

Figure 2: Place the stamping to be captured to the bottom of the image

To avoid a smearing effect on the timestamp set the exposure time to a minimum, e.g. 1/240 s. With long exposure time you might capture two timestamps of sequentially decoded images, making the timestamp unreadable.

2.2 Encoder Pipeline

Encoder Pipeline

The encoder pipeline does image processing (CBIT), scaling and the encoding itself. Depending on the encoder pipeline mode, our platforms have a delay of a defined number of frames to process an image (see Appendix A for details). Higher frame rate reduces both, sensor read-out and encoder pipeline delay.

Use highest possible frame rate like 60 fps or 30 fps, also to match with typical monitor refresh rate. Typically, in an IP video system, there is no technical reason to stick with 25 fps or 50 fps other than the need to deliver PAL video at the analog output.

The encoder pipeline delay depends on the GOP (group of pictures) structure. The GOP is a set of images led by an I frame, which provides the reference image, followed by a number of P- and/or B frames, which provide the changes compared to the last reference image. B-frames allow bit rate reduction but increase delay as they compare to prevailing and succeeding frames which need to be decoded first.

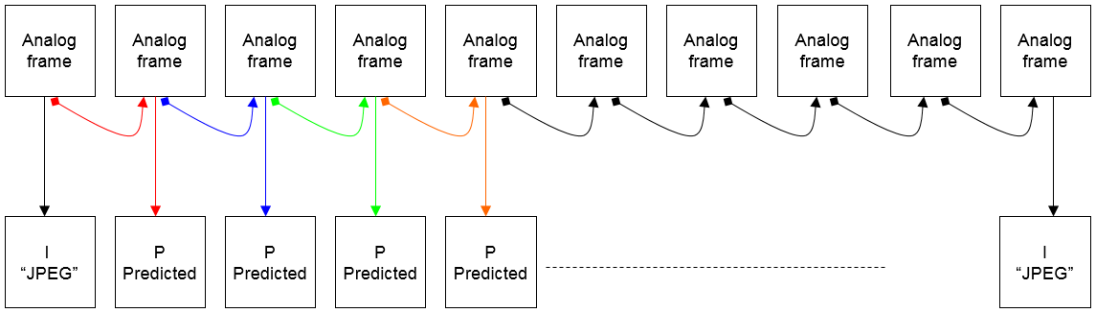

Figure 3: Group of pictures, IP structure, differential encoding

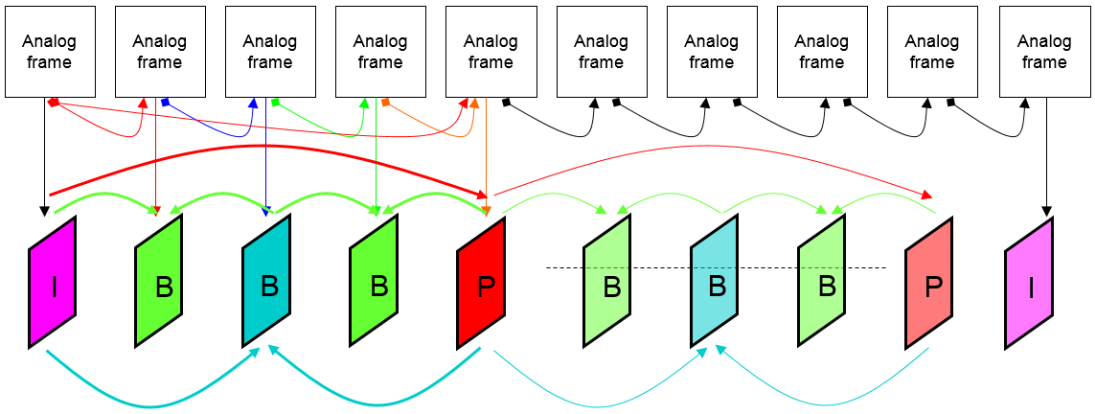

Figure 4: Group of pictures, IBBBP structure, bi-directional predictive coding

Set GOP structure to use I- and P-frames only.

The platforms CPP4 and CPP6 use different encoder architecture and thus have a different delay:

- CPP4 has a constant delay of 3 or 4 frames depending on the output resolution.
- CPP6 uses different pipeline modes depending on input resolution, analogue video output and dewarping, causing the encoder pipeline delay to vary between 1 and 4 frames plus 1 frame jitter.

See **Appendix A** for details on the various modes.

Out-of-the-box settings for the encoder pipeline focus on optimized image quality, not on low delay. So it is necessary to optimize the encoder settings for low delay, if required.

Quality settings in H.264 algorithm are defined by ‘Quantization Parameters’, short QP. The lower the QP, the higher is the encoded image quality.

The QP setting defines the best possible image quality if the maximum bit rate allows. Setting the QP to a value below visual recognition of image quality difference only increases bit rate.

Typically, I-frames and P-frames are encoded with different QP. The leading value is the P-frame QP, while the difference to the I-frame is described as I/P-frame delta QP.

The I/P-frame delta QP standard setting -6 improves image quality but generates large I-frames which – depending on the network characteristics and conditions – may lead to considerable jitter, visible as jerky video. So-called video smoothing in the decoder may compensate, but this leads to additional delay as the decoder must buffer a number of frames.

To reduce the jitter caused by big I frames, set I/P frame delta QP to a higher value, for example 0 (zero). But be aware that it leads to a pumping effect in the video due to the image quality difference.

With the minimum P-frame QP value you control the maximum size of I- and P-frames can be controlled. A high value leads to low bit rate and low image quality but also reduces the delay. For example, a minimum P-frame QP value of 27 +/- 3 can be a setting to improve delay. But it may also lead to a lower picture quality.

If the encoder settings will only be used for a live-viewing stream, that is no recording is required, then set I-frame distance to auto (infinite). Thus, the encoder produces I frames only sporadically and on request, allowing an intelligent decoder to adjust to the lower transmission delay of P-frames.

Besides the QP settings, set the maximum and target bit rate to an identical value to avoid too much jitter in the frame sizes.
This leads to less peaks in the delay but lower quality during motion.

2.3 CPU and Data Processing

CPU
Data Processing

An IP camera often is a complex computing system, loaded with various tasks like recording, analytics, script handling, alarm responses, exports, JPEG requests, or multiple streams generation. Delay might increase if the camera's CPU is (over-)loaded with other tasks.

Figure 5: Performance indicator on Web page

The CPU requires computing power to process the bit stream and serve the network interface.

If tasks of same or higher priority keep the CPU from processing encoded data towards transmission, data will be kept in buffers, increasing delay.

Make sure that a camera requiring low latency is not overloaded with tasks. CPU load should consistently stay below an average of 70%.

2.4 Network Transmission

Network Transmission

The delay caused by the network transmission is the time needed for the transportation of the IP data through the complete network path. It depends on frame size and connection speed.

The frame size can be reduced adjusting the respective encoder settings.

Connection speed may not always be a 1:1 relation to the slowest link, as transmission may also be delayed by packet buffering in switches congested due to high traffic load. If flow control is used, in case of network congestion, the data will be held back in the camera, increasing latency.

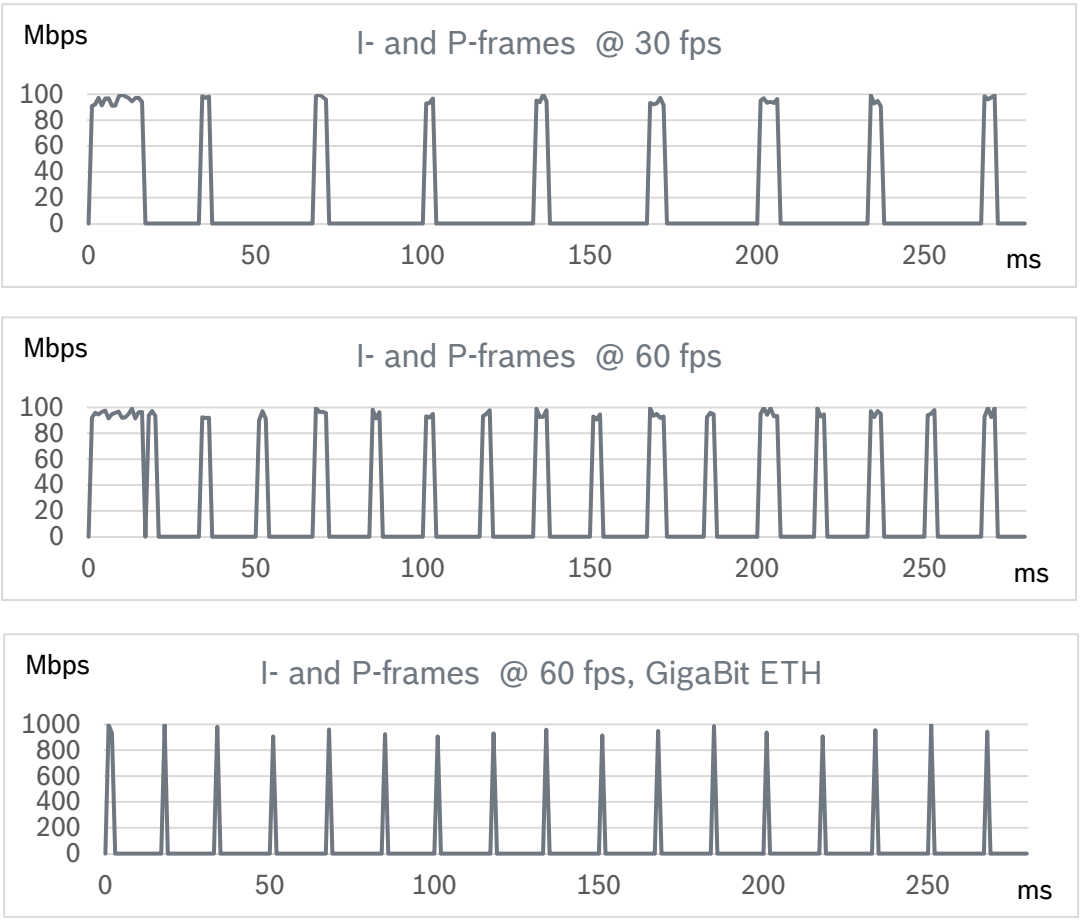

Figure 6: Data transmission of frames of various size and frame rate

Use the best possible network connection to the camera, for example 100 Mbps, and ensure sufficient transport capability within the network.

2.5 SW/HW Decoding

SW/HW Decoding

CPU decoding time depends on resolution, frame size, CPU performance and the software decoder.

Big I frames can cause considerable jitter in decoding time. As described in the encoder section above, you can reduce the I frame size by adjusting the value for I/P frame delta QP.

CPU performance on the decoding side is less of a problem with modern PCs, but the tool used for software decoding is. Many software decoders, like VLC, maintain smooth video with extensive buffering, inducing high delay, which is contradictive to minimizing latency.

In most cases, you achieve a lower delay if the decoding software uses hardware accelerators for decoding, lifting off decoding load from the CPU to the GPU.

Also operating systems have an impact as the video needs to pass through all OS display layers.

2.6 Monitor

Monitor

Monitors need to display a video signal they receive. This signal is scanned for processing and often buffered for post-processing. There is a high variation of this internal processing delay, depending on the settings and quality of a monitor.

Check the datasheet or benchmarks of the monitor for the delay introduced by the monitor.

Use a special monitor with a very low delay. The higher the monitor refresh rate, the lower the delay typically is. Gaming monitors generally prove best results.

3 Conclusion

It should be clear from the above that low latency does not come on no cost.

Reducing image quality, and thus bit rate, is an easy first try - but for sure is not the only solution and might not be the preferred one. Network infrastructure, decoding and display equipment take their share, too, and investment into high-quality equipment will pay back as well.

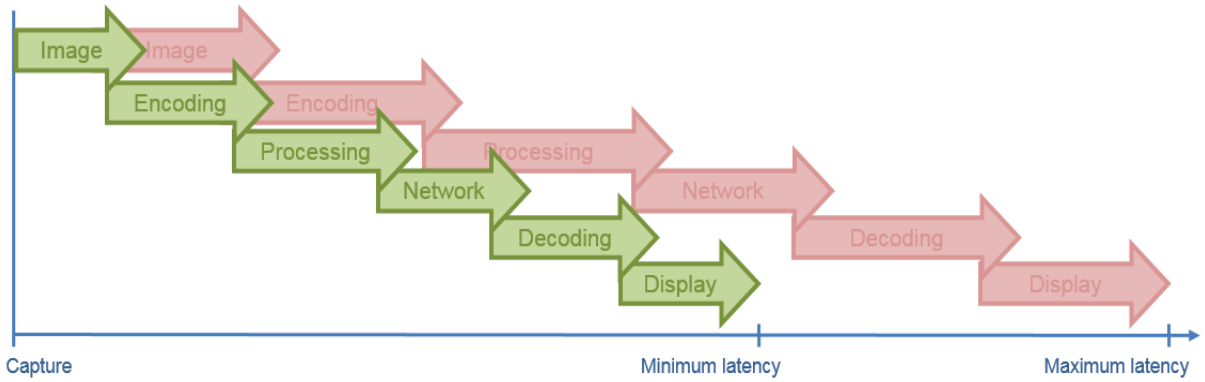

Figure 7: Delays in every processing step sum up to overall latency

There are many possibilities to do things right for achieving low latency but they don't come for free and need to be addressed properly.

4 Appendix A

4.1 CPP4

Mode	Image processing	Encoding time	If not main scaler (low resolution)
Up to 1080p	2 frames	1 frame	+ 1 frame
> 1920w	3 frames	1 frame	+ 1 frame

4.2 CPP6

Mode	Image processing	Encoding time	If not main scaler (low resolution)
Up to 2716w	<< 1 frame	1 frame + 1 frame jitter	+ 1 frame
Up to 2716w with PAL VOUT	1 frame	1 frame + 1 frame jitter	+ 1 frame
> 2716w	1 frame	1 frame + 1 frame jitter	+ 1 frame
panoramic 360° in dewarping mode	2 frames	1 frame + 1 frame jitter	+ 1 frame
panoramic 180° in dewarping mode	3 frames	1 frame + 1 frame jitter	+ 1 frame

4.3 CPP7

Mode	Image processing	Encoding time	If not main scaler (low resolution)
Starlight mode	2 frames	1 frame	+ 1 frame
HDR mode	3 frames	1 frame	+ 1 frame

4.4 Calculated camera latency

Calculation includes encoder pipeline and data processing which is typically less than 20 ms.

Sensor capture and network transmission are excluded.

All calculations refer to stream 1.

Platform	Resolution	Frame Rate	Calculated camera latency
CPP4	2MP, 1920x1080	30 fps	< 120 ms
CPP4	1MP, 1280x720	60 fps	< 67 ms
CPP4	SD, 704x480	30 fps	< 150 ms
CPP4	5MP, 2992x1680	12 fps	< 350 ms
CPP6	2MP, 1920x1080	30 fps	< 90 ms
CPP6	2MP, 1920x1080	60 fps	< 55 ms
CPP6 with PAL VOUT	2MP, 1920x1080	25 fps	< 140 ms
CPP6	4K/8MP, 3840x2160	30 fps	< 120 ms
CPP6	12MP, 4000x3000	20 fps	< 170 ms
CPP6 panoramic 360° in dewarping mode	2MP, 1920x1080	20 fps	< 220 ms
CPP6 panoramic 180° in dewarping mode	2MP, 1920x1080	20 fps	< 270 ms
CPP7 starlight	2MP, 1920x1080	30 fps	< 120 ms
CPP7 starlight	2MP, 1920x1080	60 fps	< 67 ms
CPP7 starlight	SD, 704x480	30 fps	< 150 ms
CPP7 HDR	2MP, 1920x1080	30 fps	< 150 ms
CPP7 HDR	2MP, 1920x1080	60 fps	< 85 ms
CPP7 HDR	SD, 704x480	30 fps	< 185 ms

5 Appendix B - Measurements

5.1 Measurement methods

The most simple, and easy to reproduce method to measure scene-to-screen delay is to use a milliseconds timestamp on the camera and film the camera's decoded stream from a monitor.

A more sophisticated method is to blink into a camera sensor with a LED and interpret the resulting change on the IP packet layer. This requires a bit more equipment and some experience to do the interpretation right. The advantage of this method is that network, decoding and monitor delay is excluded. The result is just the delay of the camera itself until the IP traffic is delivered to the network.

But as a user is typically interested in what will be seen at the end, the first method is sufficient for most cases and thus was also used in the examples below.

Since release of firmware 6.20 and the suitable MPEG-ActiveX 5.90 – when installed – the video smoothing behavior for the software decoder used in the browser can be modified via the camera web page.

For latency measurements it is recommended to switch off any buffering in the software decoder by selecting "Unbuffered" latency mode.

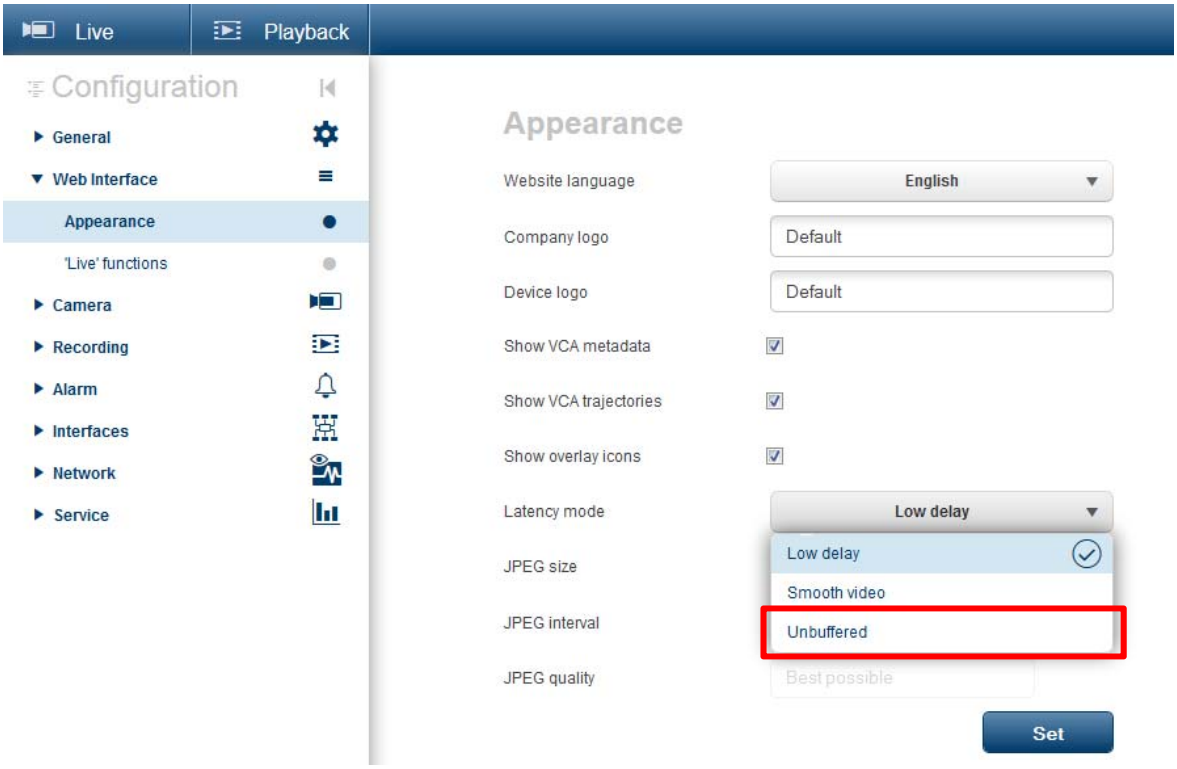

Figure 8: Latency mode setting in browser

5.2 Example measurements

The measurements below have been performed with recent firmware and software decoder versions.

Other prerequisites:

- ► Always stream 1 has been used for measurement to avoid additional delay from scaling.
- ► All camera settings as well as MPEG-ActiveX have been optimized for low latency, according to the advices above.
- ► Monitor delay has been reduced to below relevance by using an ultra-low-delay monitor with 144 Hz refresh rate.

Note: The measurement values are approximate values and may vary depending on the real setup. They may even be improved.

Device under test	DINION HD 1080p (CPP4)	NBN-733 (CPP4)
Firmware version	6.20	6.20
Connection type	UDP	UDP
GOP type	IP	IP
Decoder	MPEG-ActiveX 5.90	MPEG-ActiveX 5.90
Resolution	2MP, 1920x1080	1MP, 1280x720
Frame rate	30 fps	60 fps
Windows 7	**233 ms**	**150 ms**

Device under test	AUTODOME 7000 HD (CPP4)	AUTODOME 7000 HD (CPP4)
Firmware version	6.20	6.20
Connection type	UDP	UDP
GOP type	IP	IP
Decoder	MPEG-ActiveX 5.90	MPEG-ActiveX 5.90
Resolution	2MP, 1920x1080	1MP, 1280x720
Frame rate	30 fps	60 fps
Windows 7	**300 ms**	**166 ms**

Device under test	DINION IP starlight 8000 MP (CPP6)	DINION IP starlight 8000 MP – 5MP (16:9) (CPP6)
Firmware version	6.20	6.20
Connection type	UDP	UDP
GOP type	IP	IP
Decoder	MPEG-ActiveX 5.90	MPEG-ActiveX 5.90
Resolution	2MP, 1920x1080	5MP, 2992x1680
Frame rate	30 fps	30 fps
Windows XP	**133 ms**	**200 ms**

Device under test	DINION IP starlight 8000 MP – 5MP (16:9) (CPP6)	DINION IP starlight 7000 (CPP7)
Firmware version	6.30	6.30
Connection type	UDP	UDP
GOP type	IP	IP
Decoder	MPEG-ActiveX 6.01	MPEG-ActiveX 6.01
Resolution	5MP, 2992x1680	2MP, 1920x1080
Frame rate	30 fps	60 fps
Windows 7 64bit	**167 ms**	**134 ms**

Device under test	DINION IP starlight 7000(CPP7)	DINION IP starlight 7000 HDR(CPP7)
Firmware version	6.30	6.30
Connection type	UDP	UDP
GOP type	IP	IP
Decoder	MPEG-ActiveX 6.01	MPEG-ActiveX 6.01
Resolution	2MP, 1920x1080	2MP, 1920x1080
Frame rate	30 fps	30 fps
Windows 7 64bit	**167 ms**	**200 ms**

Bosch Sicherheitssysteme GmbH
Robert-Bosch-Ring 5
85630 Grasbrunn
Germany
www.boschsecurity.com

Authors: **Konrad Simon, Product Manager IP Video**
Jan Brandstädter, Engineering Technologies and Platforms