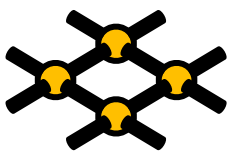
Updates on UltraGrid Platform

Petr Holub

<hopet@ics.muni.cz>

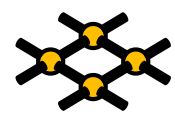
Laboratory of Advanced Networking Technologies

CESNET, Czech Republic



ResearchChannel/Internet2 Working Group
Internet2 Fall Member Meeting

New Orleans, Louisiana, 2008–10–13



Talk Overview

UltraGrid – Production Status

- 1.5 Gbps uncompressed video-only stream
  250 Mbps DXT-compressed video-only stream
- Linux support
  - DVS Centaurus capture card
  - compression options: uncompressed, DXT
  - display options: SDL, GL, SAGE
- MacOS X support
  - various cards
  - compression options: uncompressed, DXT
  - display options: SDL, GL, AJA Kona3
  - 2× 4-core processors are ideal
    - allows even for small (1500 B) packets with DXT
  - 2K 4:2:2 support
    - 2048×1080 @ 24 psf, 24 p
    - 4:2:2 YUV / 10 b per color channel $\Longrightarrow$ 1,2 Gbps

Development Team

- Petr Holub <hopet@ics.muni.cz>
- Miloš Liška <xliska@fi.muni.cz>
- Jiří Matela <matela@ics.muni.cz>
- Martin Beneš <martinbenesh@gmail.com>
- Alumni:
  - Lukáš Hejtmánek <xhejtman@ics.muni.cz>
  - Ian Wesley-Smith <iwsmith@cct.lsu.edu>

iHDTV Compatibility Mode for UltraGrid

- Analysis of iHDTV code format
  - done on iHDTV SVN trunk code
  - UltraGrid doesn't support audio
  - UYVY for 8-bit, v210 for 10-bit video
  - 8-bit seems to be supported only for *n*-way
  - per-stream frame offset (we have the same idea for new version of UltraGrid stream format)
  - frames are taken as "progressive" and split into top and bottom half when sending over two links
- UltraGrid now supports
  - v210 in 10-bit mode
  - sending over one or two links
  - two-way sending only

iHDTV Compatibility Mode for UltraGrid

- iHDTV really needs code reworking to make it sustainable in our view
  - the code is unnecessarily obfuscated (e.g., single letter variables),
  - better structured programming and minimizing `gotos`,
  - code documentation is missing.
- Goal is to have a new release including iHDTV compatibility by SuperComputing|o8.

iHDTV → UltraGrid

- Rendering through AJA Kona3, as we don't support v420 in software displays as of now.
- Audio packets are dropped by UltraGrid now.

```
uv -i -d kona
uv -i -d kona 11.22.33.44
uv -i -d kona 11.22.33.44 11.22.33.55
```

UltraGrid → iHDTV

- Video captured from QuickTime that gives us v210 data in 10-bit depth
- Centaurus uses a different format that would require transformation
- No audio packets are sent
- Tested using iHDTV rendering through AJA Xena

```
uv -i -t quicktime -p 11.22.33.44
uv -i -t quicktime -p 11.22.33.44 11.22.33.55
```

DXT Compression

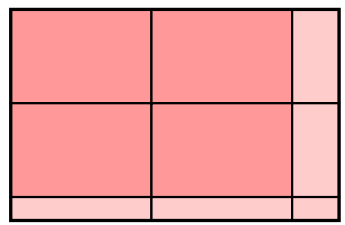
- DXT compression
  - color-indexing-based compression
  - fixed compression ratio:
    - 8:1 or 4:1 data reduction in RGBA
    - 6:1 or 3:1 data reduction in RGB
  - `http://www.cesnet.cz/doc/techzpravy/2008/dxt-compression-for-ultragrid/`
  - posterization (banding) of long “slow” gradients
- CPU-based compression
  - needs about 4 cores
- GPU-based decompression/rendering
  - very low requirements
- 250 Mbps HD video stream

JPEG2K Compression

- Relevant advantages
  - lossless and lossy compression
  - no blocking (unless too small tiles are used)
  - superior quality
  - support for resolution levels
  - simple downscaling by omitting finest wavelet scale (so called "multiple component transform")
- Opted for implementation from scratch
  - Jasper was too modular to be rewritten for GPU and also more complex than we need

JPEG2K Compression—Process

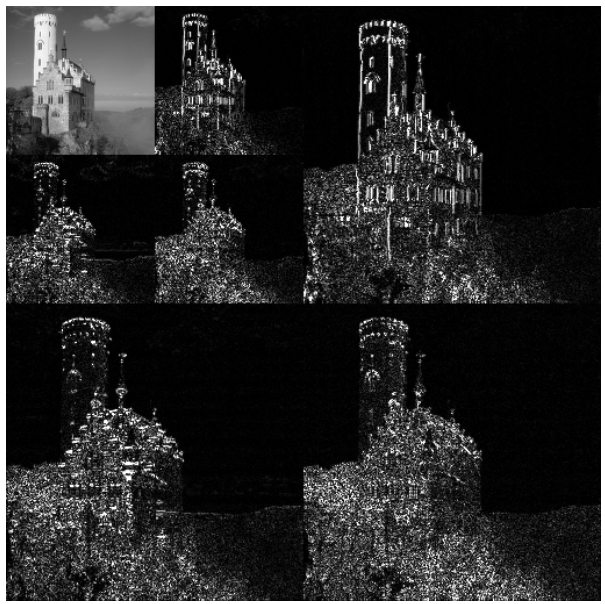
- Color component transformation
  - to YC_BC_Rcolor space through Irreversible Color Transform (ICT)
  - to modified YUV through Reversible Color Transform (RCT)
- Tiling
  - tile can be chosen of (almost) arbitrary size
  - decreases memory usage
  - application may decode specific tiles only
  - introduces risk of blocking
  - deteriorates quality by decreasing PSNR



CESNET

JPEG2K Compression—Process

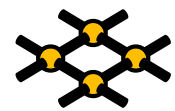
- Wavelet transform
  - irreversible Cohen-Daubechies-Feauveau 9/7 wavelet transform
    - uses floats and thus introduces quantization noise depending on the precision of the decoder
  - reversible Cohen-Daubechies-Feauveau (or LeGall) 5/3 wavelet transform
    - uses integer coefficients $\Longrightarrow$ doesn't require rounding
  - implementation either using lifting scheme or convolution
  - we use lifting-based CGF 5/3 to be able to implement lossless JPEG2000



http://en.wikipedia.org/wiki/Image:Jpeg2000_2-level_wavelet_transform-lichtenstein.png

JPEG2K Compression—Process

- Quantization
  - lossy part of the process resulting in a set of integers
  - can be set to lossless behavior when desired
- Coding
  - sub-bands (approximation scales) $\longrightarrow$ rectangular precincts $\longrightarrow$ code blocks of the same size (except edges)
  - Embedded Block Coding with Optimal Truncation (EBCOT) on each code block for each bit plane
  - bits selected by the previous coding passes go through binary MQ-coder
  - resulting bitstream is split into packets (some packet may get dropped to save bandwidth), then further organized into layers
  - search for optimum packet length in order to minimize image distortion



CESNET

JPEG2K Compression—Where We Are

- CUDA-based GPU implementation, available as a library
- Implemented by Jiří Matela
- Summer internship at EVL

1. color transforms on GPU – DONE
   (not necessary, we have YC_BC_R already)
2. tiling – DONE
   (one tile per HD whole frame)
3. DWT based on CDF 5/3 and lifting scheme on GPU – DONE
   …we're at 2 ms latency here without optimizations
4. quantization – NOT DONE
5. coding – NOT DONE

End-to-End Latency Measurements

Configuration	Latency [ms]
Linux, Centaurus, no compression	90±8
Linux, Centaurus II, no compression	85±8
Linux, Centaurus, DXT compression	130±8
Linux, Centaurus II, DXT compression	95±8
MacOS X, DeckLink Pro HD, no compression	148±8
MacOS X, DeckLink Pro HD, DXT compression	178±8

- applies for Jumbo frames

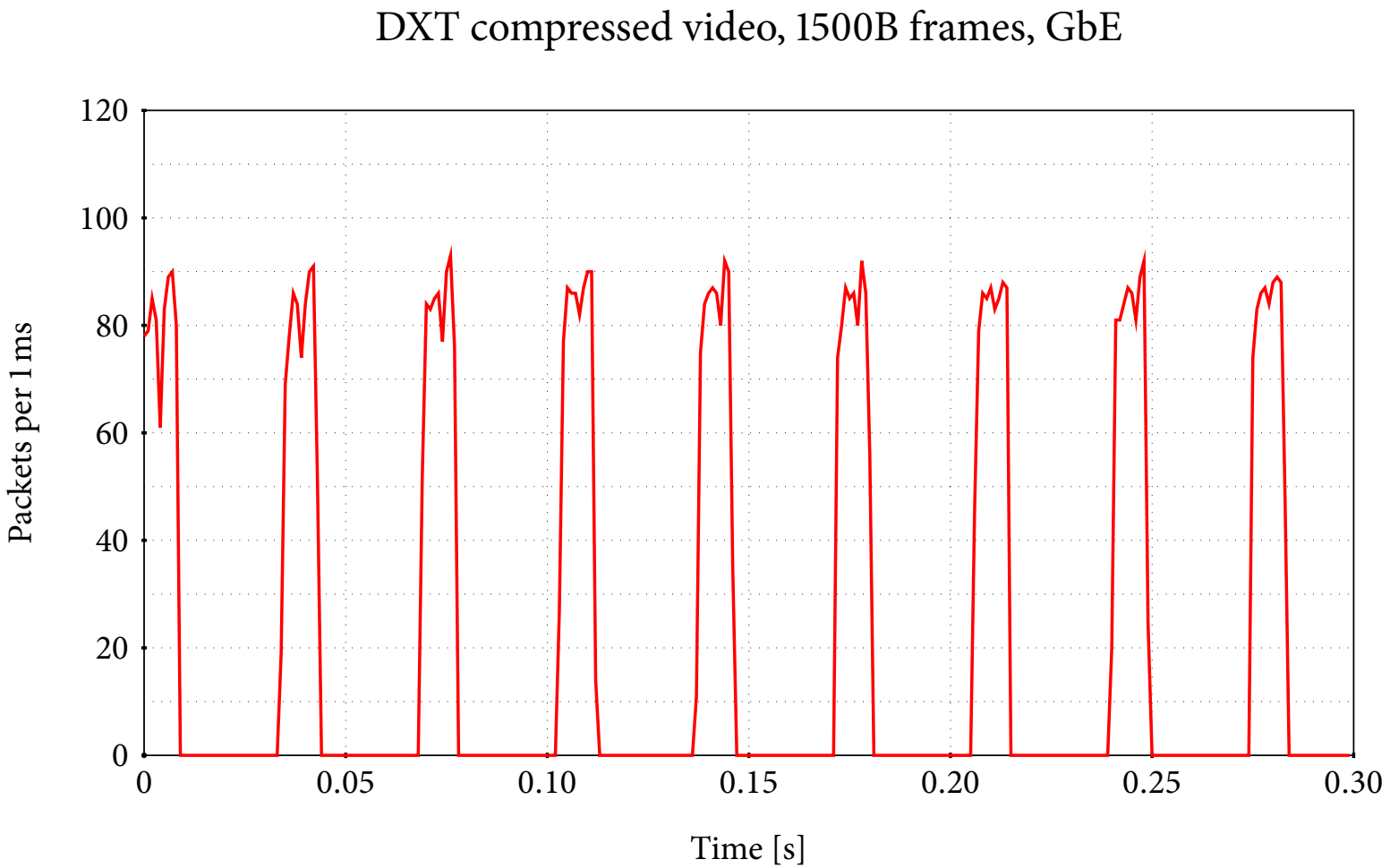
Sender CPU Load Measurements

Configuration	CPU load
Linux, no compression, Jumbo frames	26 %
Linux, DXT compression, Jumbo frames	332 %
Linux, DXT compression, 1500B frames	350 %
MacOS X, no compression, Jumbo frames	46 %
MacOS X, DXT compression, Jumbo frames	324 %
MacOS X, DXT compression, 1500B frames	350 %

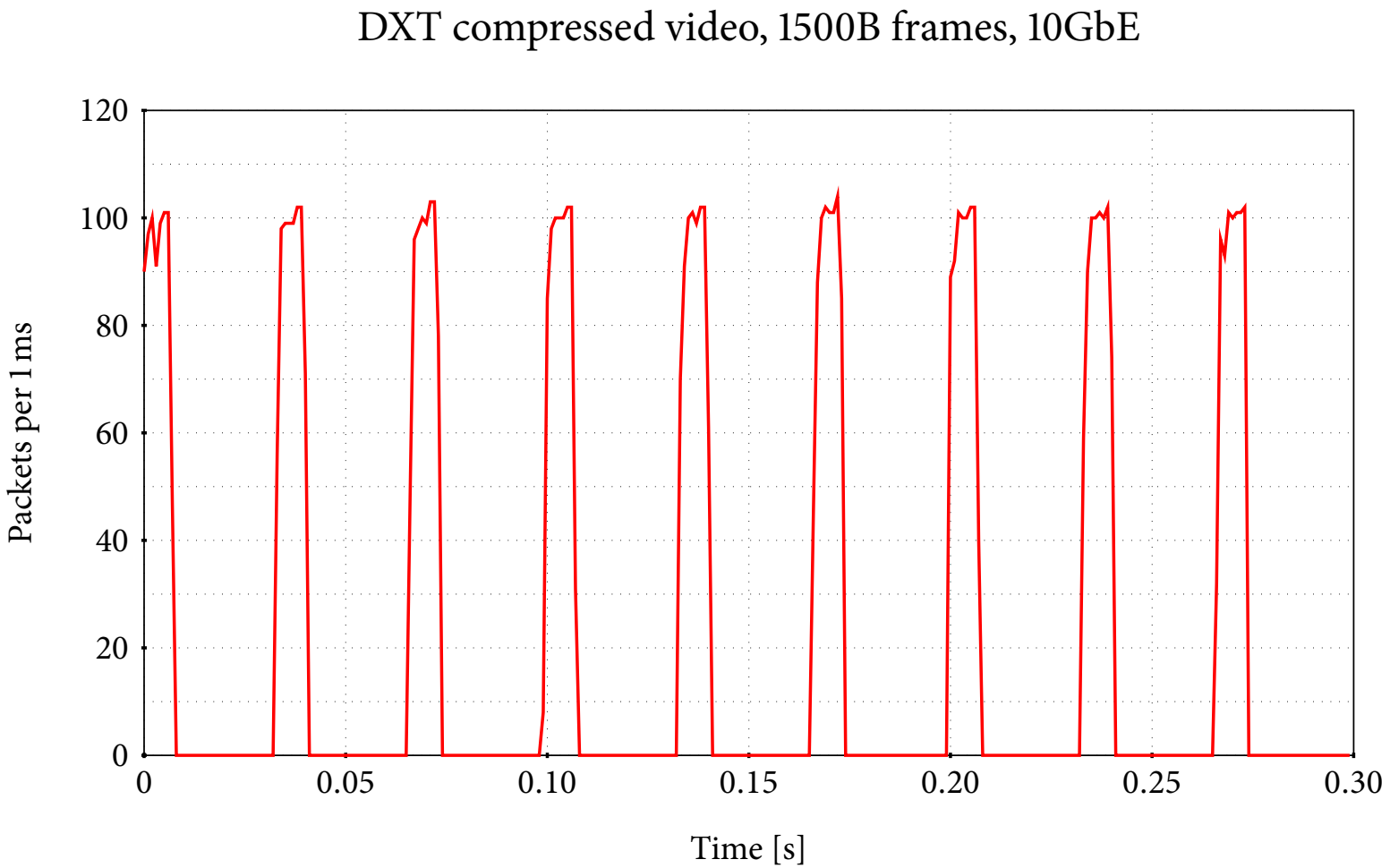
Receiver CPU Load Measurements

Configuration	CPU load
Linux, no compression, Jumbo frames	123 %
Linux, DXT compression, Jumbo frames	17 %
Linux, DXT compression, 1500B frames	24 %
MacOS X, no compression, Jumbo frames	160 %
MacOS X, DXT compression, Jumbo frames	9 %
MacOS X, DXT compression, 1500B frames	29 %

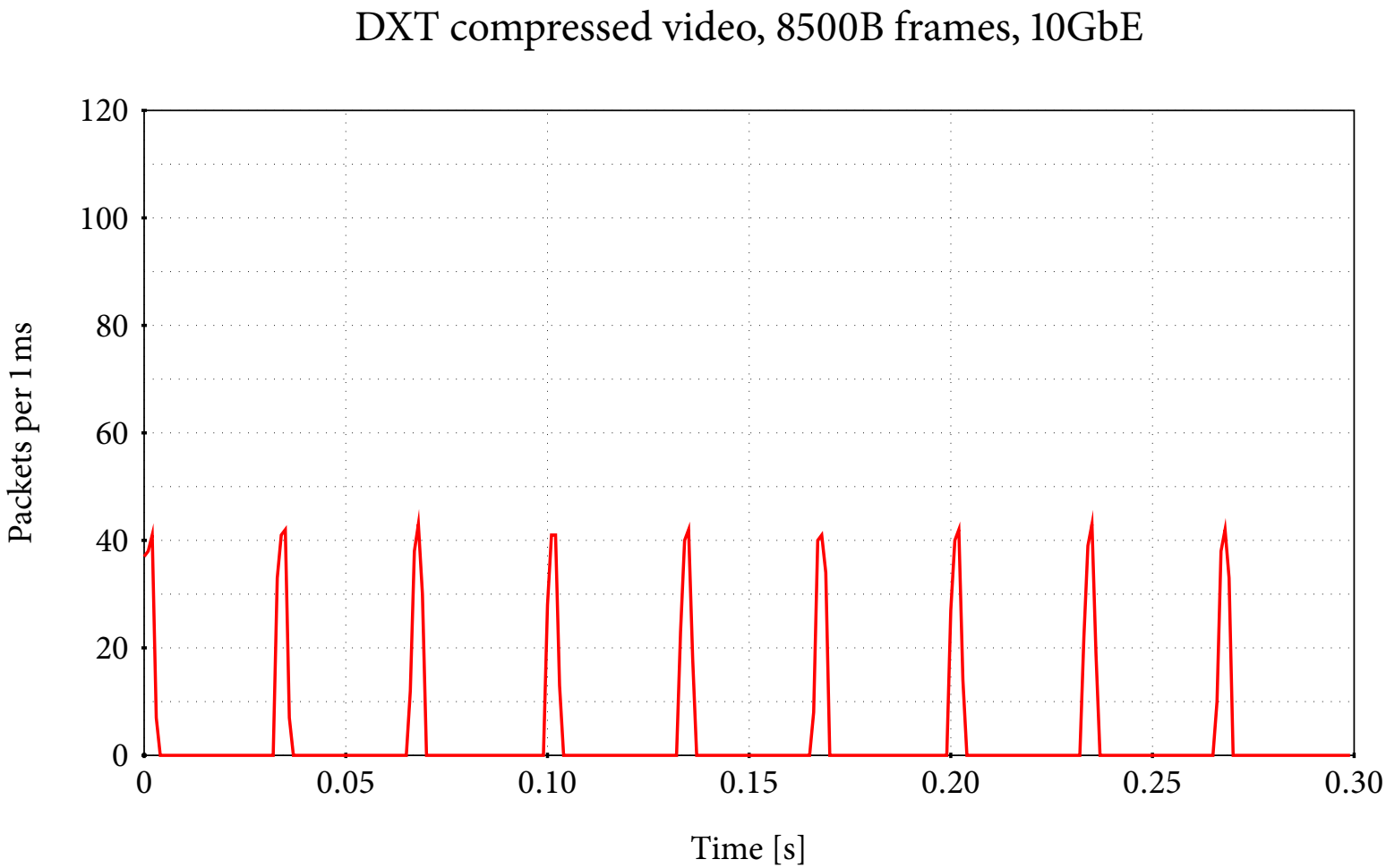
Bandwidth Profiles



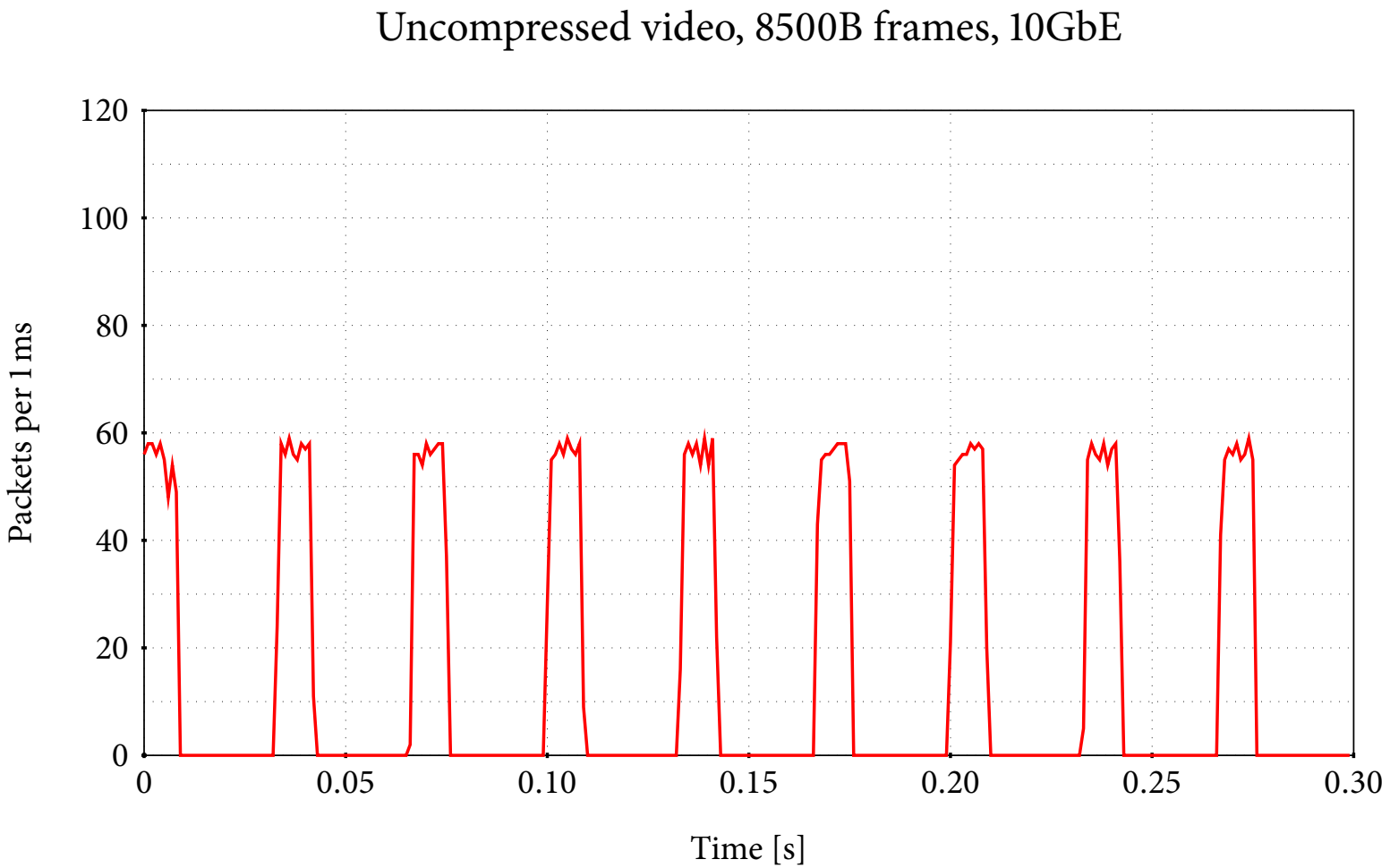
Bandwidth Profiles



Bandwidth Profiles



Bandwidth Profiles

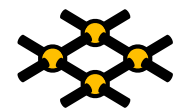


Future Development – Brno Team

- Modularization of UltraGrid
  - input format description
  - transmission format description
  - display output description
- New general transmission format
  - frame number + memory buffer offset in octets
  - will work for any intra-frame compression
  - doesn't allow direct screen addressing from packet header
    - but compression usually disables this either
- Finish 2K 4:4:4 support
  - actually RGB transmission

Future Development – for Others

- Audio
  - long-term issue
  - HD-SDI input (maybe also separate input)
  - HD-SDI playback
  - Linux ALSA and MacOS X playback
  - ideally up to 8 channels (AJA HD10AM)
  - out-of-band $\times$ inband transmission
- Mac Mini fixes
  - OpenGL 1.4 is only supported on Intel GMA 950
  - we get a bus error when binding the DXT texture in the GL



CESNET

Future Development

Thank you for your attention!

Q?/A!

<hopet@ics.muni.cz>
https://ultragrid.sitola.cz/
(→ https://www.sitola.cz/igrid/index.php/UltraGrid)