Low bandwidth desktop and video streaming for collaborative tiled display environments

Jason Kimball *, Tom Wypych, Falko Kuester

Center of Interdisciplinary Science for Art, Architecture and Archaeology (CISA3), Qualcomm Institute, Calit2, University of California, San Diego, United States

HIGHLIGHTS

- H.264 video streaming of desktop content to high resolution tiled display systems.
- Low bandwidth and low latency streaming outperform existing raw RGB techniques.
- Enables streaming full HD resolution desktop content from wireless laptops.
- Removes the dependence on 10 Gbps networks in collaborative tiled display systems.
- Demonstration system delivers 1080P30 desktop content under 10 Mbps, 100 ms latency.

ARTICLE INFO

Article history:
Received 20 July 2011
Received in revised form
16 July 2015
Accepted 18 July 2015
Available online xxxx

Keywords:
Real-time video streaming
Tiled display environments
Collaborative visualization
H.264 video compression

ABSTRACT

High-resolution display environments built on networked, multi-tile displays have emerged as an enabling tool for collaborative, distributed visualization work. They provide a means to present, compare, and correlate data in a broad range of formats and coming from a multitude of different sources. Visualization of these distributed data resources may be achieved from a variety of clustered processing and display resources for local rendering and may be streamed on demand and in real-time from remotely rendered content. The latter is particularly important when multiple users want to concurrently share content from their personal devices to further augment the shared workspace. This paper presents a high-quality video streaming technique allowing remotely generated content to be acquired and streamed to multi-tile display environments from a range of sources and over a heterogeneous wide area network.

The presented technique uses video compression to reduce the entropy and therefore required bandwidth of the video stream. Compressed video delivery poses a series of challenges for display on tiled video walls which are addressed in this paper. These include delivery to the display wall from a variety of devices and localities with synchronized playback, seamless mobility as users move and resize the video streams across the tiled display wall, and low latency video encoding, decoding, and display necessary for interactive applications. The presented technique is able to deliver 1080p resolution, multimedia rich content with bandwidth requirements below 10 Mbps and low enough latency for constant interactivity. A case study is provided, comparing uncompressed and compressed streaming techniques, with performance evaluations for bandwidth use, total latency, maximum frame rate, and visual quality.

1. Introduction

Tiled display environments offer high resolution display resources at a scale unparalleled by other display technologies. These higher resolution display surfaces allow for larger datasets to be visualized on larger workspaces and support new modes of collaboration among users. Several techniques currently exist for filling the high pixel count with certain types of data, but challenges still exist for displaying many types of content on tiled display environments, such as a wide range of desktop applications and tools that researchers are already familiar with, and sharing content from multiple sources which are external to the display environment.

While projects like the OptiPortal research initiative [1] that are developing cost-effective tiled display wall technology from commodity components aim to increase access to tiled display walls, wide spread adoption of tiled display environments will

* Corresponding author.
E-mail addresses: jkimball@eng.ucsd.edu (J. Kimball), twypych@ucsd.edu (T. Wypych), fkuester@ucsd.edu (F. Kuester).

http://dx.doi.org/10.1016/j.future.2015.07.009
0167-739X/

Please cite this article in press as: J. Kimball, et al., Low bandwidth desktop and video streaming for collaborative tiled display environments, Future Generation Computer Systems (2015), http://dx.doi.org/10.1016/j.future.2015.07.009

not occur until users can access and visualize their data in a way that they are familiar with. Despite the benefits higher resolution display walls provide, an interim solution is necessary until these tiled display walls support all of the visualization functionality researchers need.

Until now a popular solution has been to stream remote content directly from rendering nodes in an uncompressed format that requires high bandwidth interconnects. The high bandwidth required for this streaming limits the types of devices that can provide content, as well as the total resolution and update rate of the content streamed.

To address the issue of high speed network dependence we present a new framework for streaming remote content to tiled display environments using low latency H.264 video compression. The use of video compression on a video stream significantly reduces the required bandwidth and network resources which provides several improvements over uncompressed streaming. Real-time video streaming capabilities become accessible to a new class of bandwidth constrained devices such as wireless laptops. Also, the number of concurrent video streams that can be sent over existing high speed network infrastructure is increased, allowing collaboration with multiple concurrent content streams on a network with a 1 Gbps bandwidth or significantly less.

This paper describes the implementation of a complete end-to-end system for streaming desktop content to tiled display walls as outlined in Fig. 1. It also addresses the challenges of streaming compressed video to a tiled display wall including acquisition of a range of source content, extremely low-latency video encoding, efficient network transport over heterogeneous wide area networks, video decompression, and visualization on a tiled display system. Also addressed is the issue of mobility, allowing the users to dynamically reposition and resize each video stream anywhere on the tiled display wall without interruption.

The presented framework provides support for the acquisition and display of a variety of desktop and video content, allowing users the freedom to visualize and collaborate on tiled display walls with conventional desktop applications, HD video cameras, video game consoles, and almost any device which outputs to an HDMI or DVI interface.

In the following sections we describe the components of the system that make this possible and present a performance analysis between streaming of uncompressed RGB pixel streams, single frame compressed streams, and H.264 compressed streams. We are able to demonstrate that using the system outlined in this paper, H.264 compressed video streams provide a higher frame throughput, lower end-to-end latency, and require significantly less bandwidth than uncompressed RGB streams while providing a higher quality and lower bandwidth usage than competing real-time single frame compression approaches.

2. Related work

This paper bridges contributions from two disjoint areas of research: displaying content on tiled display walls, and streaming desktop content for collaboration.

2.1. Scalable tiled display wall applications

Several existing middleware systems provide access to different content in tiled display environments. Chromium [2] and DMX [3] are the most non-invasive middleware in terms of allowing generic applications to be run. They both operate by intercepting graphical API calls of an application running on a dedicated head node and rerouting them to a distributed display application. These approaches restrict scalability because content has to be distributed from one computer running the application to the display wall, limiting the amount of content to that which can be processed by the head node. Furthermore, Chromium only works with OpenGL applications, and can only run one application at a time, and DMX only works with X11 environments and has limited support for hardware accelerated graphics such as OpenGL. While both of these approaches allow easy access to certain subsets of existing applications, both are limited in scalability and neither are a solution for collaborative visualization as neither approach can bring together content from multiple source computers.

Fig. 1. An example of 4 live video streams on a 32 screen tiled display wall. Researchers and a remote collaborator compare a collection of images from previous San Diego County fires with video streams of a real-time wind flow visualization, current fire locations map on a tablet, and live panoramic images from various San Diego County locations on a laptop.

SAGE is a pixel streaming middleware which takes raw RGB input from a source application and streams it to a tiled display wall [4]. It can display simultaneous content from multiple sources and those sources can be subdivided across a rendering cluster in order to parallelize applications which are data, CPU, or graphics intensive. In this way, SAGE improves upon scalability by allowing rendering to be parallelized and enables collaborative input by displaying streams from multiple sources. Because SAGE streams raw RGB pixel data, bandwidth usage can be very high. Applications streaming high resolution content to multiple displays require 10 Gbps interconnects and networking hardware with even higher bisection bandwidth as the aggregate bandwidth to all of the displays can well exceed 10 Gbps. Furthermore, applications must be recompiled to take use of the SAGE Application Interface Library (SAIL).

Careful segmentation of the SAGE stream is required in order to conserve the bandwidth of data sent to each display node and computational effort is required to segment the video stream each frame. SAGE facilitates the segmentation of the rendered stream geometry to match the output geometry on the tiled display wall. This requires communication between the rendered source and the display nodes whenever users want to move or resize content on the display wall, resulting in mobility latency. In this paper's approach, the video bandwidth is much lower and can be distributed to all nodes, so there is no delay when moving or resizing content.

For collaboration between multiple SAGE display walls, Renambot et al. [5] implement a network utility, called SageBridge, which dynamically re-segments the rendered streams when streaming to multiple display walls. Jeong et al. [6] improve on the collaborative display of SAGE streams by introducing macro-block segmentation as a replacement for per-pixel image segmentation across multiple displays. This simplifies the task of segmenting and distributing a large stream, reducing the load on the SageBridge machines when collaborating between a large number of display walls. CGLX [7] is

Please cite this article in press as: J. Kimball, et al., Low bandwidth desktop and video streaming for collaborative tiled display environments, Future Generation Computer Systems (2015), http://dx.doi.org/10.1016/j.future.2015.07.009

another scalable tiled display wall middleware which uses a distributed rendering approach to improve scalability. It uses the display nodes as the rendering nodes, and all rendering is done on computers connected directly to the display devices. This allows very scalable applications as each display node only loads and processes the data it needs for its display. However, like SAGE, applications must be rebuilt to use the CGLX middleware. Based on this approach, applications for high resolution image exploration [8,9] and HD video playback from files [10] have been designed, which use significantly less network bandwidth for display than previous approaches. We build upon this approach and leverage the distributed computational power of the display nodes to receive, decode, and display compressed video streams using lower bandwidth than uncompressed approaches. Like Chromium and DXM, CGLX is only able to run one application at a time. However, by integrating support for collaborative video streaming directly into CGLX, support for collaborative video stream content is provided to all CGLX applications.

2.2. Streaming desktop content

While applications must be specifically adapted to use the SAGE Application Interface Library, requiring them to manage their own display pixels in order to provide them to the library, SAGE does provide a specially designed VNC client which connects with SAIL, allowing SAGE to pull content from any VNC server running applications of interest. Stodle et al. [11] also demonstrate using VNC to distribute a laptop's display to multiple users over a local network for collaboration.

While VNC works well for remotely accessing traditional desktop applications, where only small parts of the application windows change at a time, more multimedia intensive applications can consume large amounts of bandwidth and computational resources, resulting in latency and incomplete screen updates. Nieh et al. [12] perform a thorough performance analysis on several thin client solutions and demonstrate that even given sufficient bandwidth, those thin clients are unable to provide high quality video display due to inherent update latencies.

Several solutions attempt to reduce the latency and improve the update throughput of thin clients. Tan-Atichat and Pasquale [13] address VNC's low framerate performance in high-latency networks by increasing the pull request rate from the client to improve the framerate that the server delivers. While this does increase the framerate, it also increases the network bandwidth used and cannot improve the framerate in bandwidth restricted networks. The fact that they stream uncompressed pixels limits the maximum resolution and framerate, and requires connections faster than 1 Gbps for desktop-resolution streaming, as demonstrated by Holub et al. [14].

Barrato et al. [15] intercept X11 display calls and transmit drawing commands instead of the rendered RGB pixels themselves. They also add an application to assist in video playback by sending video data instead of the decompressed frames. For other types of multimedia as well as rendered content such as scientific visualization applications, their approach will not provide an improvement in bandwidth usage or latency.

2.3. Compressed streaming

To address the bandwidth usage of streaming RGB pixels, many solutions adapted a real-time image compression format called DXT compression. DXT was originally designed for compressing textures used in video games and other graphics applications. It has become a popular choice for real-time image compression because of the universal support for DXT decompression by graphics hardware as well as highly optimized libraries available that can do real-time DXT encoding of HD-resolution images at high frame rates. UltraGrid [16] uses DXT compression to deliver 1080i 60 Hz video at 250 Mbps instead of the 1.5 Gbps uncompressed video would require. Support for DXT has also been added to SAGE [17,6]. However, DXT compression only offers a fixed compression ratio of six-to-one whereas tiled display walls are often tens to hundreds of times the resolution of a normal desktop. DXT compression also produces significant visual artifacts, in part because the color space is restricted to 16-bit colors. Section 8 demonstrates the poor image reproduction quality of this compression approach.

De Winter et al. [18] demonstrate using H.264 video encoding to deliver desktop multimedia content to thin clients with both a better compression ratio and higher quality. These efforts focus on several H.264 encoding options and how those affect bandwidth usage and latency. They measure a variety of content including office applications, 3D video games, web browsing, and video playback, and demonstrate a significant bandwidth savings compared to other thin client solutions such as VNC, X11 forwarding, and FreeNX. Kimball et al. [19] also utilize H.264 video compression for a multimedia centric remote control and streaming application. Their research focuses on reducing bandwidth and latency to allow training simulations to be run and streamed remotely from across the country. While the compression latency they achieve is low enough for real-time interaction, neither of these address the requirements to transmit and display the encoded video on tiled display walls.

This paper's proposed approach is able to stream high resolution desktop content and video to tiled display walls using much less bandwidth than previous approaches. This allows for a higher frame throughput of video data with lower latencies, for tens to hundreds of video streams to fit through a single gigabit connection, and for devices on wireless networks to stream desktop and video data to tiled display environments. We address the issue of mobility and synchronization as the video streams are moved across tile display boundaries and our solution is able to deliver high-resolution video (1080p 30 Hz) using under 10 Mbps bandwidth.

3. System overview

The streaming framework consists of two major components: the video streaming application which acquires, compresses, and sends the video stream, and the display framework which runs as part of the cglX middleware on a tiled display wall. The streaming application runs on a desktop computer. It has options to select the input source, such as local screen capture, or a list of hardware capture devices, such as a camera or a capture card. After selecting the source device a target bitrate is entered for the video encoder, which can be adjusted to optimize for the network bandwidth available or the type of content being captured. Finally, a network address is entered to match the configuration of the receiving display wall.

The visualization framework for receiving and displaying video streams is integrated with the cglX tiled display visualization middleware. By default, each running application will accept and display incoming video streams on the configured multicast address and port. Multiple streams from multiple sources can connect using the same address and port and can join and leave at any time. As each video stream is received, it is displayed on the screen in tandem with the running application content. The user is then free to resize and reposition it anywhere on the display wall. Applications that want more control over the display of video streams can explicitly disable and enable them using an API call. Furthermore, applications can request direct access to the video texture to draw it in a custom way. In this case, the video texture is

Please cite this article in press as: J. Kimball, et al., Low bandwidth desktop and video streaming for collaborative tiled display environments, Future Generation Computer Systems (2015), http://dx.doi.org/10.1016/j.future.2015.07.009

still automatically updated and globally synchronized each frame by cglX.

The presented system enables video streaming to tiled display environments from a wide range of devices, including many environments which do not have sufficient bandwidth to support uncompressed streaming. While video compression is at the heart of the bandwidth reduction, several other important components are integral to this flexible video acquisition, streaming, and display system, as shown in Fig. 2.

The *Source Management* component acquires video stream source frames from a number of sources including desktop capture, hardware devices, and video files. A modular driver API allows users to implement drivers for custom sources such as propriety hardware devices.

The *Encoding Engine* tunes the video compression for real-time, low latency entropy reduction on commodity desktop processors. Video encoding is computationally expensive, and software based video encoders for HD-resolution content are typically used in an offline mode due to the time it takes to encode high-quality video. While hardware based video encoders are used in a myriad of consumer devices for real-time video streaming, they lack the flexibility to read from a large variety of sources that this framework provides. Careful experimentation and tuning have lead to a set of encoding parameters that reduce encoding latency and facilitate real-time compression.

The *Network Transport* provides a system for efficient network transmission of the compressed video packets to all of the display nodes as well as a tool for application level routing across heterogeneous, wide area networks. A custom, light-weight protocol, similar to RTP (Real Time Protocol) was designed to be able to stream from one-to-many using UDP multicast to efficiently deliver to all of the tiled display wall nodes. An application called *VideoRouter* was written to also facilitate streaming across heterogeneous network boundaries when direct multicast transport to the display wall is not suitable.

Finally *Decoding Engine* provides a mechanism for efficient video decoding, color conversion, and synchronized display on tiled display environments. A custom shader application converts the decoded video back to RGB color on the GPU to further reduce latency. Decoding of multiple video streams is parallelized, and the current display timestamp for each video is broadcast by the middleware's synchronization node when it changes. Users can freely move and resize each of the video streams anywhere on the display wall without delay in tandem with other display content from any application running on the cglX middleware.

4. Source management

The source management component is the interface to all of the hardware and software acquisition devices which provide the live source frames for each video stream. It communicates with each acquisition source by way of a custom interface driver. Each driver advertises the availability of video frames at a specific resolution, color format, and sampling rate. The driver obtains source image frames as they become available and uses a callback based signaling mechanism to notify the encoding engine when each new video frame is available. The use of drivers serves as an abstraction layer by providing a uniform interface to the subsequent components in the streaming architecture. It also allows modular video sources to be added or removed as appropriate from a number of differently equipped physical machine configurations based on available hardware and operating system architecture.

A variety of input device drivers were implemented to demonstrate the utility and flexibility of the streaming framework, including capture from hardware devices and a software-based desktop capture driver. For hardware device support, a driver wrapping the

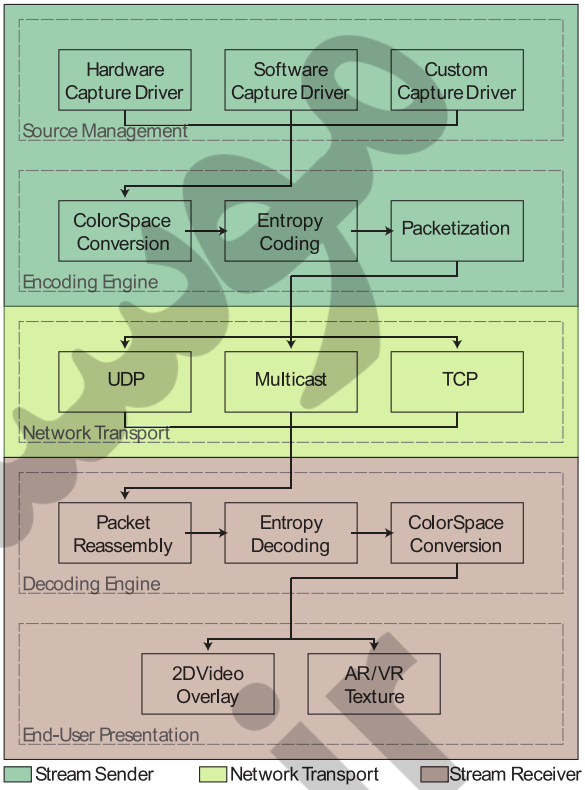


Fig. 2. The streaming system architecture.

Microsoft DirectShow Capture API was used. This covers a wide range of cameras and video capture hardware. Video capture cards can be used to capture the video output of another computer. In addition to capture from hardware devices, a software capture driver was developed for instances where video capture hardware is not available. This driver uses native screen capture API calls to collect the entire desktop or a subregion, such as one specific display when multiple monitors are attached. Fullscreen desktop images can be acquired at a rate above 20 frames per second on a moderately equipped workstation despite the system overhead involved in the acquisition of the desktop bitmap. For hardware devices, input video frames are delivered as fast as 60 Hz.

5. Encoding engine

The encoding engine performs the entropy reduction process for the input image sequence data and provides compressed frame data to the network transport engine. Image frames are received from the source management component in their native color format and are converted into encoder-standard YUV420 colorspace. Then video encoding is performed using the FFMPEG and libx264 video codec library with support for the H.264 video codec.

Low latency real-time compression is achieved by selecting encoding options which eliminate multi-frame dependence, eliminating any multi-frame buffering in the encoder. The first option is to disable bi-directional frames, which require a following frame before they can be resolved. Second, lookahead rate control is disabled. By default, lookahead rate control provides accurate bitrate allocation decision making to the x264 encoder, but induces a latency equal to the number of frames in the lookahead group as input frames must be buffered. Disabling the lookahead rate control and instead using a conventional hysteresis based scanning technique can be achieved without any frame buffering. Finally, using multiple threads for parallel encoding improves encoding throughput but adds an extra frame of latency for

Please cite this article in press as: J. Kimball, et al., Low bandwidth desktop and video streaming for collaborative tiled display environments, Future Generation Computer Systems (2015), http://dx.doi.org/10.1016/j.future.2015.07.009

each additional encoding thread. Instead, a slice based threading model is used, which partitions each frame into multiple image slices and performs parallel encoding on each slice. While this achieves less effective entropy coding because optimizations cannot be performed across slice boundaries, it reduces single frame encoding time without requiring extra frame buffering.

A further optimization to reduce the maximum per-frame encoding time is to enable *intra-frame refresh*. Normally a group of pictures contains an intra-frame and multiple inter-frames. Each intra-frame contains a single image which is compressed spatially but not temporally. These frames serve to refresh the reference for the inter-frames that follow. Whenever there is a change in scene or high motion this refresh makes the subsequent inter-frames more efficient. However, intra-frame size is generally significantly larger than inter-frames and their encoding takes longer too. Because of this, there are spikes in both encoding time and bandwidth usage. In order to smooth out these spikes, *intra-frame refresh* periodically refreshes columns of macro-blocks instead of whole frames. This spreads the intra-frame over multiple frames and results in a more uniform time and bandwidth distribution and reduces the maximum encoding cost.

Once the frame dependent latencies have been eliminated, the single frame encoding cost is dependent on several encoding parameters which affect the quality and compression efficiency at the cost of extra computation. We refer the reader to De Winter et al. [18] for a discussion of the effects these parameters have. In general, the right parameter values will depend on the processing power and network bandwidth available and may need to be tuned to provide the appropriate speed and quality trade-off.

6. Network transport

It is important to consider how compressed video is distributed to the display nodes, both from a local network and from a remote location, as to how it effects latency and bandwidth usage. Streaming approaches that send only the visible portions to each display node can avoid data duplication. However, segmenting a video into subregions is not feasible when using a compressed video stream. Instead, a full copy of the compressed video is delivered to each node. This has the advantage that each node receives the same data, which can be done efficiently by using UDP multicast to transmit the video stream from the source to all of the display nodes. Instead of the source having to send data packets to each node individually, multicast allows the networking hardware to duplicate each packet of video data to all nodes.

The use of UDP multicast removes the necessity for any of the receiving nodes to communicate directly with the video stream source as the video is moved between different display nodes. Nodes that do not need to display the video can still receive the video stream without putting additional strain on the network, allowing nodes to buffer frames for when the video is moved from one display to another. In this way, the streaming is display location agnostic, a strong advantage over previous methods which incur computational overhead and update latency when repositioning the video stream.

For streaming from wide area networks, using UDP multicast may not be possible due to network policies or high packet loss rate. In these situations, other network protocols can be used to send video packets to a bridging node on the display cluster network. A simple bridging application called *VideoRouter* was written to accept either TCP or UDP video streams and retransmit them to the display nodes via multicast. A custom streaming protocol similar to RTP (Real Time Protocol) was created to assist in network transport, wide area network routing using VideoRouter, and in display synchronization. Each compressed video stream is packetized and encapsulated with a delivery header. The delivery header contains the packet count, frame number, presentation timestamp, dimensions of the video stream, and a unique 32-bit identifier. The packet count and frame number allow the packets to be ordered correctly by the receiving nodes. The presentation timestamp is used to synchronize the update of each sequential video frame across all of the display nodes. The unique identifier is used for stream specific routing by VideoRouter and also to differentiate multiple video streams simultaneously received on the same address by the display nodes. Furthermore, it can be used by cglX applications to indicate contextual information about each video stream.

It is important to understand the limits of this network strategy as the number of video streams and display tiles grows large and the bandwidth approaches the limits of the network. Using 10 Mbps video streams and a 10 Gbps network, 1000 video streams can be distributed simultaneously via multicast to all display nodes. In this situation, the input and output bitrates match. However if multiple incoming ports supporting 10 Gbps are used, the bottleneck will be the multicast output. Using the subscription features of multicast to only subscribe to the subset of video streams each display node currently needs can reduce the output bottleneck. However, the total multicast bisection bandwidth on a switch may not be larger than the output of a single port. In this case, switching to a strictly UDP distribution method may increase the total number of video streams that can be delivered to the display nodes. Adapting the *VideoRouter* utility to use UDP distribution could facilitate this change without any modification to the streaming client, however switching to a UDP distribution method would introduce latency when video is moved to new tiles. In practice, this method handles 10 s of simultaneous video streams on display walls with many 10 s of displays. Further research into network distribution methods may be required for display walls with 1000 s of display screens and 1000 s of video streams.

7. Decoding engine

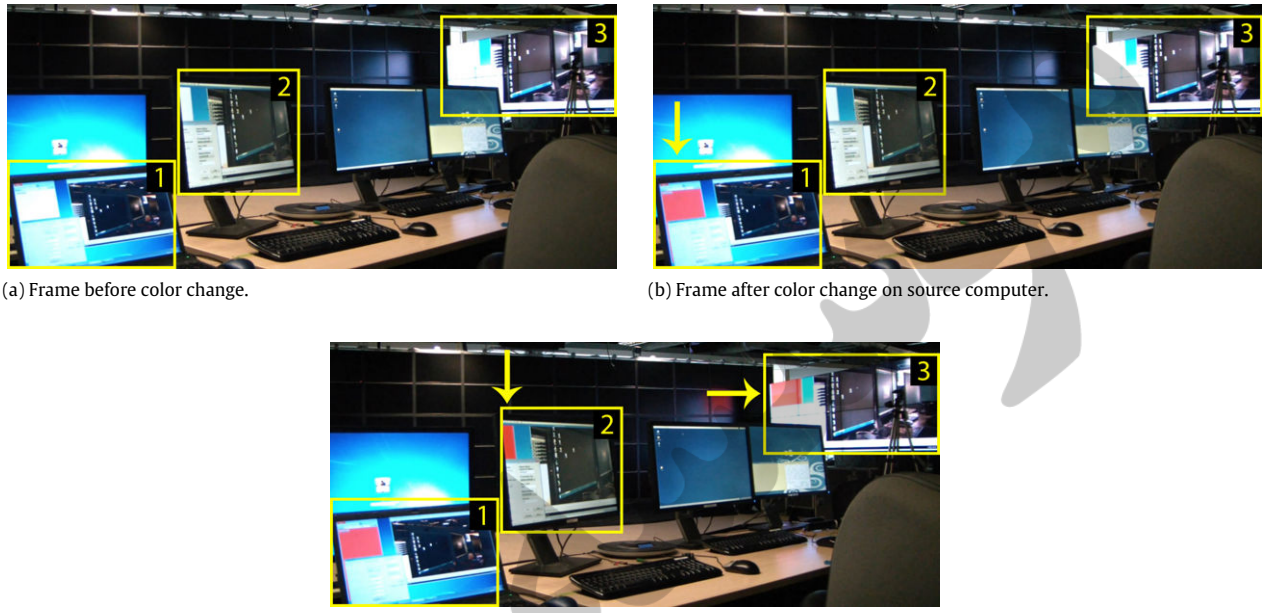
Receiving and presenting the video streams on each of the display nodes involves several steps, including receiving the video packets and reassembling them into video frame data, decoding each video frame, uploading the decoded frames to the GPU, converting from YUV420 to RGB color format, and finally displaying the video frame on the tiled display with synchronized updating. A streamlined process with minimal latency is described.

A receiving and decoding system waits for new video frames to be received on the network, then passes received video frames to a set of decoding threads, one per video stream. All that is needed to start decoding a video stream is the width and height of the video which is included in the header of each intra-frame sent, allowing decoding to start at any time on an already streaming video. Decoding is performed by the FFMPEG library. Each frame gets decoded into a buffer in YUV420 color format. The decoded image buffer, the frame number, presentation time stamp, and channel ID are passed to an upload queue for uploading and display via the GPU.

After being decoded, each frame has to be converted from YUV420 into RGB before it can be displayed. This can be a CPU intensive operation that adds a lot of latency. Instead of being converted to RGB on the CPU and then uploaded to the GPU, the color conversion happens on the GPU. Since the YUV420 color format is half the size of RGB, the upload to the GPU is quicker. Also, the conversion on the GPU is able to utilize the massive parallelism of processing units and memory bandwidth on the GPU, reducing the conversion time.

The YUV420 color format frame buffer is uploaded to a set of special textures in YUV color format consisting of one full

Please cite this article in press as: J. Kimball, et al., Low bandwidth desktop and video streaming for collaborative tiled display environments, Future Generation Computer Systems (2015), http://dx.doi.org/10.1016/j.future.2015.07.009



(a) Frame before color change.

(b) Frame after color change on source computer.

(c) Frame after color change propagates to tiled display.

Fig. 3. In the testing setup a 60 frame per second video camera is used to record the changes in the source computer (1) and measure the time until the changes propagate to the head node (2) and tiled display wall (3). Image (a) shows the source computer and tiled displays with a white indicator window. Image (b) shows the indicator window on the source computer changing to red. Image (c), captured 12 frames later, shows the head node and tiled display now display the red window. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

resolution single channel texture for the luminance and two half resolution single channel textures for the two chrominance channels. Then that texture is rendered into an RGB24 texture using a custom shader program to do the color conversion.

Once the RGB texture has been generated, a signal is raised to notify the application that a new frame is ready. Progression from one frame to the next is synchronized by the head node. A timestap attached to each frame indicates when it should be displayed. When that time is reached, the head node sets a flag and all display nodes switch to the next video frame texture during their next draw cycle.

The actual drawing of the frame's texture can be handled either by the CGLX library, and be drawn as a rectangular window drawn on top of the application, or the application itself can draw the texture. Because it is an RGB texture, the video stream texture is straightforward to integrate with application specific scene geometry. Each video stream is tagged with a unique identifier so custom applications can indicate individual video context.

8. Performance analysis

To qualify our system using H.264 compressed video, we compare its performance to uncompressed RGB streaming with no video compression and single-frame DXT compression. We measure the total latency of the system from generation of a frame on the source computer to display on the tiled display wall. We also report the bandwidth usage for each method, as well as the maximum sustained frame rate of the streams over wireless, gigabit, and 10 gigabit networks.

8.1. Test setup

Using a 60 frame per second camera, we are able to measure the end to end latency to within approximately 17 ms. We use a highly visible single frame event, changing the color of a window on the desktop, and measure how long it takes to propagate through the system. To simulate the type of multimedia desktop interactions we are targeting, a mix of 720P HD video and desktop application windows fill the desktop. Fig. 3 shows two frames from a recorded video showing the transition of the color window from green to blue. The tests are performed on a laptop with a 2.4 GHz Intel Core 2 Duo processor using 802.11 N wireless and on a desktop with an Intel Core i7 Extreme 3.33 GHz processor using 1 and 10 gigabit switched ethernet. The test usage scenarios on the laptop include software desktop capture at a resolution of 1440 × 900 and an embedded camera at a resolution of 720 × 480. On the desktop computer, software desktop capture as well as hardware capture using an EMS Imaging XtremeRGB-Ex2 capture card [20] are tested. The card is used to capture the DVI output directly from the video card at a resolution of 1920 × 1200. The desktop software capture is able to provide quicker access to the desktop image, but at lower frame rate, while the hardware capture device can provide a higher frame rate at the cost of added latency. Each of these scenarios is tested using real-time H.264 encoding, real-time DXT compression using FastDXT [21], and uncompressed RGB.

For the tests on the laptop, the video streams are sent via TCP to the VideoRouter application which is running on the tiled display wall's head node. From there it is retransmitted in UDP multicast to all of the wall nodes. On a low-latency network, the use of TCP does not add significant latency or jitter to the video delivery, but was found necessary due to the higher packet-loss rate when using wireless. The tests on the desktop computer are performed using direct UDP multicast to the tiled display wall.

8.2. Results versus uncompressed streaming

The measured values compare the bandwidth usage (Fig. 4), end-to-end latency (Fig. 5), and transmitted frames per second (Fig. 6), and demonstrate that H.264 encoded video out-performs uncompressed RGB in each category. H.264 compressed video is able to deliver both a higher frame rate and a lower end-to-end latency than uncompressed RGB even when a high network bandwidth is available, demonstrating that sending uncompressed video takes a significant amount of processing resources that slow down the pipeline. In addition to the improved video delivery, the bandwidth needed to transmit the H.264 video is around two

Please cite this article in press as: J. Kimball, et al., Low bandwidth desktop and video streaming for collaborative tiled display environments, Future Generation Computer Systems (2015), http://dx.doi.org/10.1016/j.future.2015.07.009

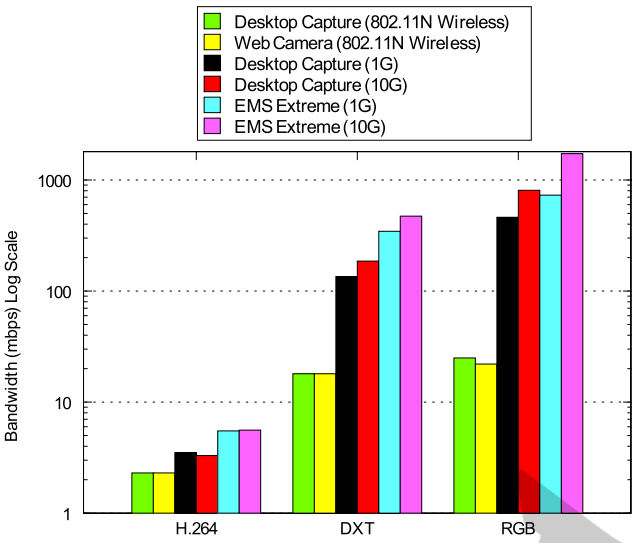


Fig. 4. Bandwidth comparison between streaming methods.

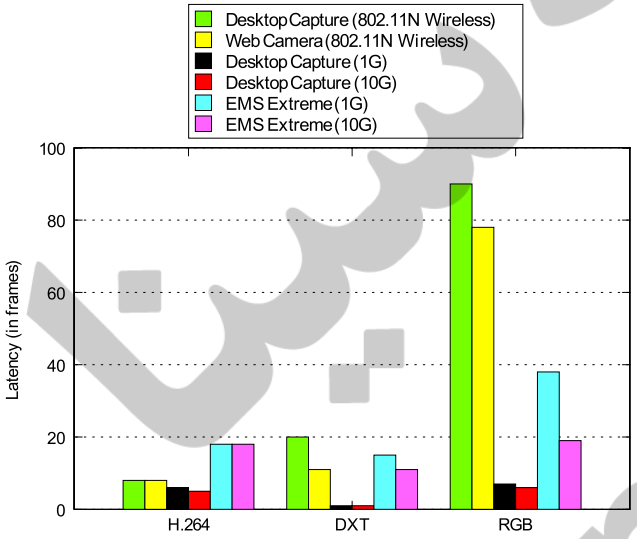


Fig. 5. Latency comparison between streaming methods, as measured by a 60 frame per second video camera.

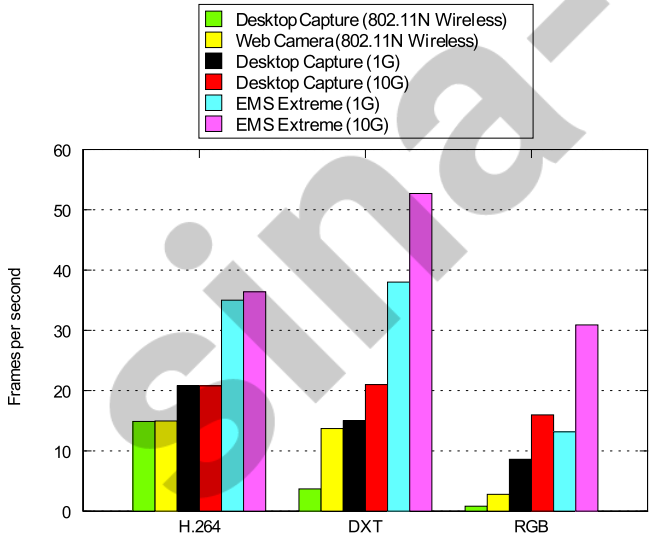


Fig. 6. Sustained frame rate comparison between streaming methods.

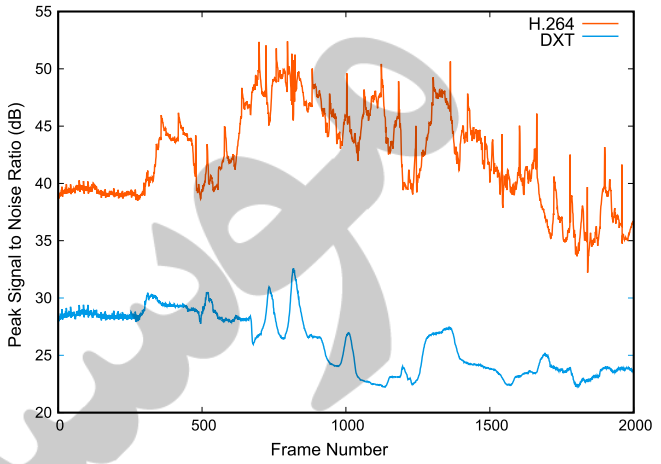


Fig. 7. Peak signal to noise ratio comparison between H.264 (top plot) and DXT encoding (bottom plot).

orders of magnitude less than the uncompressed video. Since the network bandwidth is not a limiting factor, the H.264 tests perform about equally well in both 1 and 10 Gbps networks. Furthermore the performance of H.264 on a laptop using a wireless network is better than the performance of uncompressed RGB both on a laptop and on a desktop using a 1 Gbps network. These results indicate that H.264 compression is a beneficial replacement for uncompressed RGB streaming.

8.3. Results versus DXT compressed streaming

The DXT compressed video performs better than uncompressed RGB but still under-performs H.264 in several aspects. In all tests, DXT uses at least an order of magnitude more bandwidth. This causes a drop in performance over wireless bandwidth, resulting in both lower frame throughput and higher latency. However, when the bandwidth is sufficient, DXT is able to provide a higher frame throughput, and due to its lower computational cost compared to H.264, the latency is lower in both 1 and 10 Gbps tests.

However, the image quality of DXT is inferior to H.264. Fig. 7 shows the peak signal to noise ratio (PSNR) of the compressed images compared to the original versions. The analysis was done using the same test scenario. While PSNR is not as good of a quality assessment as subjective metrics, it is as effective as other objective qualifiers [22]. It is also easy to measure and can serve as a good classifier for quality.

As demonstrated in [23], images with a PSNR above 35 dB can be classified with a low difference mean opinion score (DMOS), or small visually detectable difference compared to the original source, whereas PSNR values below 30 dB are classified with moderate to high levels in visible change and subjectively poor quality. It is demonstrated that H.264 offers a measured PSNR which would be classified as high quality while the measured PSNR for DXT compression would be rated as much lower in quality. Furthermore, with a peak signal to noise ratio often above 40 dB, H.264 can provide image quality rated as very little to no visual difference from uncompressed RGB color.

8.4. Real use example

To demonstrate the usability of our system beyond performance analysis, a real usage scenario was demonstrated as shown in Fig. 1. Four live sources are shown: a laptop sending its full desktop and its camera and a desktop sending a video playing from file and its desktop with video playing in a webpage. These are all displayed on a tiled display wall with full freedom to move and resize each of the streams.

Please cite this article in press as: J. Kimball, et al., Low bandwidth desktop and video streaming for collaborative tiled display environments, Future Generation Computer Systems (2015), http://dx.doi.org/10.1016/j.future.2015.07.009

9. Conclusion

This paper presents an end-to-end system design for streaming of H.264 compressed video to tiled display environments, targeting a wide range of input sources, including software only desktop capture and hardware video capture. The performance of this system is evaluated on commodity desktops with 10 and 1 Gbps network interfaces as well as laptops over a wireless network and compared to other exiting approaching for streaming content to display walls. The results show a higher number of frames delivered per second and lower end-to-end latency compared to uncompressed RGB streaming while requiring two orders of magnitude less bandwidth. Compared to solutions using single image DXT compressed video streams, this approach provides an order of magnitude better compression ratio and maintains a much higher image quality with 10 dB higher PSNR. These results demonstrate an end-to-end system capable of using video compression such as H.264 for low latency, high-quality video streaming to tiled display walls. We leverage the bandwidth savings of video compression to bring orders of magnitude more content from a wider range of source devices than previous methods and to display these streams on tiled display walls with no-latency mobility for user interaction.

Acknowledgments

This work was supported in part by the Army Research Laboratory (ARL) under Cooperative Agreement Number W911NF-10-2-0022, the Office of Naval Research (ONR) under Award N66604-09-C-0428 and the National Science Foundation under IGERT Award DGE-0966375.

References

[1] T.A. DeFanti, J. Leigh, L. Renambot, B. Jeong, A. Verlo, L. Long, M. Brown, D.J. Sandin, V. Vishwanath, Q. Liu, M.J. Katz, P. Papadopoulos, J.P. Keefe, G.R. Hidley, G.L. Dawe, I. Kaufman, B. Glogowski, K.-U. Doerr, R. Singh, J. Girado, J.P. Schulze, F. Kuester, L. Smarr, The optiportal, a scalable visualization, storage, and computing interface device for the optiputer, Future Gener. Comput. Syst. 25 (2) (2009) 114–123.

[2] G. Humphreys, M. Houston, R. Ng, R. Frank, S. Ahern, P.D. Kirchner, J.T. Klosowski, Chromium: a stream-processing framework for interactive rendering on clusters, ACM Trans. Graph. 21 (3) (2002) 693–702.

[3] Distributed multihead x project, http://dmx.sourceforge.net/, 2013, (Online; accessed 22.04.13].

[4] B. Jeong, L. Renambot, R. Jagodic, R. Singh, J. Aguilera, A. Johnson, J. Leigh, High-performance dynamic graphics streaming for scalable adaptive graphics environment, in: ACM/IEEE SC 2006 Conference, SC'06, 2006, Article 108.

[5] L. Renambot, B. Jeong, H. Hur, A. Johnson, J. Leigh, Enabling high resolution collaborative visualization in display rich virtual organizations, Future Gener. Comput. Syst. 25 (2) (2009) 161–168.

[6] B. Jeong, J. Leigh, A. Johnson, L. Renambot, M. Brown, R. Jagodic, S. Nam, H. Hur, Ultrascale collaborative visualization using a display-rich global cyberinfrastructure, IEEE Comput. Graph. Appl. 30 (3) (2010) 71–83.

[7] K.-U. Doerr, F. Kuester, CGLX: A scalable, high-performance visualization framework for networked display environments., IEEE Trans. Vis. Comput. Graphics 17 (3) (2010) 320–332.

[8] S. Yamaoka, K.-U. Doerr, F. Kuester, Visualization of high-resolution image collections on large tiled display walls, Future Gener. Comput. Syst. 27 (5) (2011) 498–505.

[9] S. Yamaoka, K. Ponto, K.-U. Doerr, F. Kuester, Interactive image fusion in distributed visualization environments, in: IEEE Aerospace Conference, 2011, pp. 1–7.

[10] K. Ponto, T. Wypych, K. Doerr, S. Yamaoka, J. Kimball, F. Kuester, VideoBlaster: A Distributed, Low-Network Bandwidth Method for Multimedia Playback on Tiled Display Systems, in: 11th IEEE International Symposium on Multimedia, 2009, pp. 201–206.

[11] D. Stødle, J.M. Bjørndalen, O.J. Anshus, The 22 megapixel laptop, in: Proceedings of the 2007 workshop on Emerging Displays Technologies: Images and Beyond: the Future of Displays and Interacton, 2007, pp. 1–8.

[12] J. Nieh, S.J. Yang, N. Novik, Measuring thin-client performance using slow-motion benchmarking, ACM Trans. Comput. Syst. 21 (1) (2003) 87–115.

[13] T. Tan-atichat, J. Pasquale, VNC in High-Latency Environments and Techniques for Improvement, in: IEEE Global Telecommunications Conference, 2010, pp. 1–5.

[14] P. Holub, L. Matyska, M. Liska, L. Hejtmanek, J. Denemark, T. Rebok, A. Hutanu, R. Paruchuri, J. Radil, E. Hladka, High-definition multimedia for multiparty low-latency interactive communication, Future Gener. Comput. Syst. 22 (8) (2006) 856–861.

[15] R.A. Baratto, L.N. Kim, J. Nieh, Thinc: a virtual display architecture for thin-client computing, in: Proceedings of the Twentieth ACM Symposium on Operating Systems Principles, 2005, pp. 277–290.

[16] I. Wesley-Smith, M. Liška, P. Holub, Implementation of dxt Compression for Ultragrid, Tech. Rep., CESNET, 2008.

[17] L. Renambot, B. Jeong, J. Leigh, Real-time compression for high-resolution content, in: Proceedings of the Access Grid Retreat 7.

[18] D. De Winter, P. Simoens, L. Deboosere, F. De Turck, J. Moreau, B. Dhoedt, P. Demeester, A hybrid thin-client protocol for multimedia streaming and interactive gaming applications, in: Proceedings of the 2006 International Workshop on Network and Operating Systems Support for Digital Audio and Video, 2006, pp. 86–92.

[19] J. Kimball, T. Wypych, S. Hoepner, F. Kuester, Media-rich streaming for remote simulation and training, in: Aerospace Conference, 2012, IEEE, 2012, pp. 1–7.

[20] Ems imaging xtremergb-ex2, http://www.ems-imaging.com/, 2011, Online; accessed 30.06.11].

[21] Fastdxt, http://www.evl.uic.edu/cavern/fastdxt/, [Online; accessed 07-June-2011].

[22] A.M. Rohaly, P.J. Corriveau, J.M. Libert, A.A. Webster, V. Baroncini, J. Beerends, J.-L. Blin, L. Contin, T. Hamada, D. Harrison, et al., Video quality experts group: Current results and future directions, in: Visual Communications and Image Processing 2000, International Society for Optics and Photonics, 2000, pp. 742–753.

[23] A.M. Rohaly, J. Libert, P. Corriveau, A. Webster, et al., Final report from the video quality experts group on the validation of objective models of video quality assessment, in: ITU-T Standards Contribution COM, 2000, pp. 9–80.

Jason Kimball, received a Ph.D. in Computer Science from UCSD. His research specializes in high performance computing and interactive visualization, including parallel/cluster computing, GPU programming, and high performance network applications. Previous projects include remote collaborative visualization of 3D biomedical datasets, interactive stereoscopic volume rendering techniques and HD video playback on tiled display environments.

Tom Wypych, received a Ph.D. in Computer Engineering from UCSD. His research encompasses hardware and software design of large-format, high-resolution, high-energy, X-ray computer tomography for non-destructive testing. He is also working on embedded applications on mobile and accelerated processors, implementing real time control systems, dataflow-centric applications, ultra-reliable controls, and media coding applications. Current research is focused on delivering low latency, high-resolution content over low-bandwidth links to mobile and fixed display terminals.

Falko Kuester, is the Calit2 Professor for Visualization and Virtual Reality and an Associate Professor in the Department of Structural Engineering at the Jacobs School of Engineering at the University of California, San Diego. His research is aimed at creating intuitive, collaborative digital workspaces, providing engineers, scientists and artists, with a means to intuitively explore and analyze complex, higher-dimensional data. In support of this research, he is developing new methods for the acquisition, compression, streaming, synchronization, visualization and hands-on analysis of data, as well as the cyberinfrastructure needed for collaborative data analysis and tele-immersion, including the ultra-high-resolution HIPerWall and HIPerSpace visualization environments and associated CGLX middleware.

Please cite this article in press as: J. Kimball, et al., Low bandwidth desktop and video streaming for collaborative tiled display environments, Future Generation Computer Systems (2015), http://dx.doi.org/10.1016/j.future.2015.07.009