Gharai, L. and Perkins, C. (2006) Holographic and 3D teleconferencing and visualization: implications for terabit networked applications. In, *25th IEEE International Conference on Computer Communications, 23-29 April 2006*, pages pp. 1-5, Barcelona, Spain.

Holographic and 3D Teleconferencing and Visualization: Implications for Terabit Networked Applications

Ladan Gharai
Information Sciences Institute
University of Southern California
3811 N. Fairfax Drive #200
Arlington VA 22203, USA

Colin Perkins
Department of Computing Science
University of Glasgow
17 Lilybank Gardens
Glasgow G12 8QQ, UK

Abstract— We discuss the evolution of teleconferencing and networked visualization applications to support 3-dimensional display technologies. The implications of a continuation of Moore's law, coupled with constraints on device clock rate due to power consumption, suggest that future end system and network architectures will become increasingly parallel in nature. Given this, we review trends in programming language design that will ease development of highly concurrent systems, and suggest how network transport protocols might evolve to support them.

I. INTRODUCTION

The natural evolution of teleconferencing and visualization systems is towards ever higher fidelity media, from the high resolution flat displays of current systems to the autostereoscopic and holographic displays that will drive future 3-dimensional virtual environments and visualization systems. We draw on our experience developing UltraGrid [1] – the high-end in current teleconferencing systems – to discuss how advances in media capture and display devices, microprocessor architecture and networks will affect the design of future systems. We contend that, in order to fully utilize terabit networks and multicore processors, a paradigm shift from straight-line to concurrent application design and transport protocols is needed.

We begin, in Section II, with a brief background on how autostereoscopic 3D applications operate, motivating the need for additional parallelism in transport protocols and for changes in programming languages. We discuss the direction of changes needed in transport protocols and programming languages in sections III and IV respectively, and draw our general conclusions in section V.

II. BACKGROUND

Current high-end teleconferencing and real-time networked visualization systems use high definition TV (HDTV) video formats at 60 frames per second, with exceptional picture quality, for an uncompressed data rate somewhat over one gigabit per second. This can be transported uncompressed over high-end networks [1], but is typically compressed to save network capacity. Compression by approximately a factor of 5 is commonly used in low-latency environments, while a factor of 50 is more usual when latency is not an issue, and can be achieved with acceptable quality.

Moving to 3-dimensional images dramatically increases these requirements: it has been suggested that a holographic display with equivalent spatial resolution to current HDTV displays requires 400 megapixels for horizontal parallax only viewing [2], a compressed data rate on the order of 80 Gbps if trends in compression carry over to 3-dimensional imaging[1]. Accommodating vertical parallax movement will likely increase this by at least another order of magnitude.

In addition to bandwidth requirements, such 3-dimensional teleconferencing systems will also require significant amounts of computational power to support their image processing and compression. We expect to see continued progression of Moore's law, and the consequent evolution of end system and network hardware, in parallel to the evolution in media formats. We see no reason why transistor density will not continue to increase, but it has become increasingly clear that clock rates cannot grow according to past trends since power consumption has become a limiting factor. Accordingly, we expect to see the increased transistor budget being used to develop increasingly parallel systems, in terms of multicore processors and with the integration of multiple functions onto single devices (such as crypto or network offload, or digital signal processing support).

It is also likely that emerging network architectures will be constructed from parallel components and links, with only limited aggregation. This is, once again, due to physical limitations in the conversion between optical and electrical domains at high rates, since power consumption and cooling are limiting factors in the electrical domain. Therefore, while it is possible to construct 40 Gbps links, establishing four 10 Gbps links is more cost effective; a trend likely to continue at higher data rates.

Given these related trends, we expect emerging network architectures and future multimedia applications to have highly parallel structures. Indeed, applications such as the capture, encoding and transport of 3-dimensional moving images are inherently parallel. Figure 1 shows the operation of a time-multiplexed 3-dimensional system: multiple cameras capture a scene from different view points, the camera outputs are

---

[1]Early results with compression of 3-dimensional images [3] achieve compression ratios of 26 to 1 and 15 to 1 depending on the algorithm chosen, consistent with typical compression ratios for 2-dimensional images
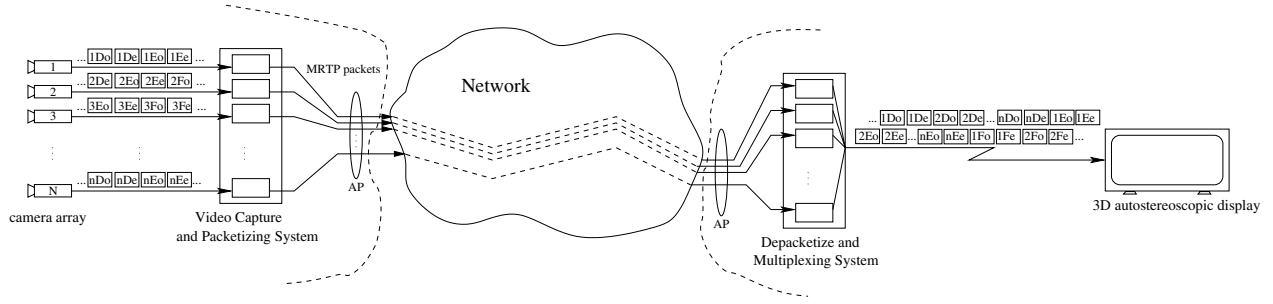
Fig. 1. The operation of a networked time-multiplexed 3D system. Each of the interlaced cameras (1—N) capture a scene from different view points. The camera outputs are labelled by: camera number (1—N), a scene identifier (..., D, E, F, ...), and an even or odd (e or o) field indicator. These fields are captured and packetized by the capture system, and transported over the network via a multi-stream real-time transport protocol (MRTP). On the receiving side, even and odd fields of each scene from all the cameras are demultiplexed, decompressed, and displayed by the autostereoscopic display, creating a 3D image. Site boundaries and aggregation points (AP) for network traffic are shown.

packetized, compressed and transported as parallel streams to a receiver where the images are decompressed, and even and odd fields from each camera are time-multiplexed and displayed in rapid succession, providing a multi-view point 3-dimensional image [4] on an autostereoscopic display. A key point to note is that while there is some inevitable aggregation of flows at site boundaries, no single network or end system element can process all the generated data.

To support development of 3-dimensional conferencing and visualization environments we must therefore fully utilize the parallelism inherent in multicore chips and high speed networks. Existing transport protocols are mostly single stream transport, with no support for parallelism, and concurrent programming is a difficult and labour intensive task: both need to change if we are to readily and effectively develop future teleconferencing systems. We expound further on the direction of these changes next.

III. MULTI-STREAM REAL-TIME TRANSPORT

Research and development on transport protocols for high speed networks has largely focused on single stream transport in networks with large bandwidth delay products. For example, much has been done to improve the response of TCP to congestion, either by changing the way it reacts to packet loss (e.g. [5]), or by making use of variations in network transit time as an additional source of information on congestion (e.g. [6]). Also, several non-TCP protocols have been developed (e.g. [7–9]) with congestion response optimised for the needs of high performance bulk data transfer. While there are some significant differences in the details of their operation, TCP and the other protocols considered are conceptually similar at the fundamental level: they transport a single sequential stream of data objects in a single transport flow, which is somehow optimised for use on a single high speed link.

We contend that such protocols are not sufficient to fully utilize terabit networks and support future high-end media applications, where data is generated in parallel, and where the transport medium is architected in parallel too. We believe these applications would benefit from a multi-stream real-time transport protocol, to provide support for parallel data transfer, in terms of data partitioning and naming, synchronisation and coordination of multiple flows, and congestion control and adaptation of parallel flows.

A. Data Partitioning and Naming

When considering a future teleconferencing system such as that illustrated in Figure 1, we observe that data flows through the system, and is processed, as a sequence of parallel streams. The data for each stream will be compressed and packetised as a sequence of meaningful application data units (ADUs) to enable robust network transport [10], and each ADU must be uniquely named to enable correct reconstruction of the data.

A key issue is that there is no single coordination point in the system. This makes it difficult to use a simple sequence number to identify ADUs. Instead, a more complex name must be used, taking into account to what stream an ADU belongs, as well as the position of the ADU within the stream. This requirement for structured ADU names is in contrast to many existing protocols. For example, RTP [11] uses a single sequence number as the ADU name, and does not impose any additional naming structure.

While data naming may initially be viewed as a minor implementation detail, it is, in fact, key to the performance of the system. For example, use of a single sequence number to identify ADUs will require synchronisation of the parallel data flows, in order to co-ordinate which parts of the sequence number space are assigned to which flow, whereas naming ADUs in terms of flow and position within that flow does not require such synchronisation. This makes RTP, for example, inefficient when compression and transmission of a single logical stream must be performed in parallel, since there is contention for the RTP sequence number space.

The naming of data elements becomes even more critical in systems where computational elements must process data from several flows. For example, consider a system that calculates the difference between images from adjacent viewpoints as part of a compression algorithm for 3-dimensional video. This will need to name not just flows and frames within those flows, but complex objects within a compressed frame from another flow, and this must be possible without requiring costly

synchronisation operations. In a data-flow architecture of this type, processing elements must be able to predictively name objects in other streams, ensuring the continual parallel flow of data. All this argues for a transport protocol that supports rich ADU names, the closest example being, perhaps, the naming scheme developed to support the Scalable Reliable Multicast (SRM) protocol [12], which was designed to name data that is subject to concurrent modification.

B. Synchronization and Coordination of Parallel Flows

Systems designed for interactive use require low end-to-end latency. That is, the total time from initial media capture at the source to final media playout at the destination must be low, and must be predictable [13]. In order to achieve this, we require both low latency on each path through the system and network, and tight synchronization between the parallel flows.

The typical approach to flow synchronization in networked multimedia systems has been to add some buffering delay at the receiver, to delay other streams to match the maximum observed latency. For example, if audio and video data for a streaming movie is sent separately, but the video consistently arrives later than the audio, then additional buffering is introduced at the receiver to delay the audio such that lip synchronization is achieved.

This approach is not ideal for the systems we envisage, since buffering stalls a decoding and rendering pipeline, and hence other parts of the data-flow system depending on that pipeline, and imposes large memory and synchronization costs. It is clear that a better solution would be to provide feedback to the sender, such that it can adjust the encoding of the media to move the playout point[2] and so regain synchronization.

One approach to such coordination is described in [14]. This uses a shim header, which is inserted into packets at aggregation points on the edge of the sender and receiver sites, to monitor the network between those sites. This data is retrieved and stored at the de-aggregation point, and queried by receivers to monitor performance across flows. This lets receivers compare reception quality in their parallel sub-flow with the reception quality of other flows, and hence send adjustment requests to the senders.

This approach suffices in systems with relatively low bandwidth demands, that use parallelism primarily to increase processing performance. In such systems the aggregation point can keep up with the data flows, and generate meaningful statistics. As the number and rate of the parallel flows increases, however, we believe such designs will become increasingly impractical, since it will not be realistic for a single device to process all packets. Accordingly, a distributed coordination function will be used to summarize reception quality, with receivers independently making the decision to adjust media playout based on asynchronous transfer of summary reception quality data from other receivers. This

more closely follows the loosely coupled RTCP [11] or Mbus [15] models, than the centralised coordination approach.

C. Congestion Control and Rate Adaptation of Parallel Flows

Congestion control and adaptation of real-time interactive media applications in best effort IP environments is a difficult problem for single flows, and more so for multiple parallel flows. This is due, in large part, to the dominance of the TCP transport protocol in the Internet, limiting new congestion control algorithms to mechanisms that are compatible with TCP, such that they do not disrupt existing network traffic. There are three key areas where the compatibility requirement has proven problematic for real-time multimedia applications:

1) A TCP flow has a distinctive transmission rate profile, following the additive increase multiplicative decrease (AIMD) model. This does not match the transmission profile of existing networked multimedia applications, which are naturally bursty on a different time scales due to the presence of intra- and inter-coded video frames, and cannot be expected to match the rate profile of 3-dimensional video compression systems either.[3]

2) The notion of fair sharing in TCP translates to (roughly) equal use of capacity between flows, thereby making it difficult for flows with different steady state data rates to co-exist. For example, if a 30 Mbps application-limited flow shares a link with a 4 Gbps video flow, they will likely disrupt each other, even if there is sufficient link capacity for both.

3) There is no direct support for parallel multi-stream transport, and it is unclear how best to map parallel application layer flows to the underlying network paths, which may experience different levels of congestion, such that they avoid mutual disruption. There is a need to exchange state between parallel flows, to ensure the aggregate is TCP-Friendly, even if the individual flows are not, but this is not supported in the current TCP-compatible network which does not recognise the concept of aggregate flows (and end-system approaches, such as the Congestion Manager [17] suffer from the communication overheads described in Section III-B).

As a result of these issues, there is considerable interest in new network architectures, to avoid sharing capacity with best effort TCP/IP traffic. Many of these are based on the use of circuit-based paths, since research and development on dynamic provisioning of lambda paths over heterogeneous networks [18] is making circuit-based connections more accessible. In a fully dynamic environment, circuits can be provisioned within seconds on the time scale of minutes or hours. While provisioned paths do not require congestion control, they do benefit from rate control. Rate control adjusts an application's sending rate to the provisioned paths bandwidth and to potential receiver limitations. A multi-stream protocol

---

[2]For example, a flow could be more heavily compressed such that it is running below the bottleneck bandwidth. This would allow queuing delays in the network to dissipate, reducing the network latency and bringing forward the playout point for the flow.

[3]Proposals such as TFRC [16] aim to send at a smoother rate, matching the average throughput of a TCP flow. However, this changes, but does not solve, the rate mismatch problem.

must be able to adjust the application's send rate over multiple flows in a manner that benefits the application and best utilizes the underlying paths.

It is unclear as yet if future terabit networks will support connection, connectionless or a hybrid of both transports. The ideal vision is seamless integration of the two technologies and widespread availability of both. Best effort IP networks provide ubiquitous but non-guaranteed service, whereas connection-oriented transport provides costly but guaranteed service. In order for best effort transport to remain dominant and relevant in future terabit networks, it is imperative that we develop congestion control mechanisms that can support multi-stream real-time applications in these environments.

D. Discussion

To fully utilize future terabit networks and deploy the next generation of interactive multimedia applications, we need the support of multi-stream real-time transport protocols, as it is clear that our current single stream paradigm will not suffice. Transport protocol design and deployment is a complicated and time consuming task. However, we note that high-end multimedia applications can play the role of catalyst in protocol design. For example, our experiences with UltraGrid demonstrated the utility of high end media application in testing out different congestion control schemes on real-world networks. We believe 3-dimensional applications can play a similar role for future terabit networks, providing us with a real-time application to experiment with transport protocols and gain a better understanding of packet dynamics and characteristics of terabit networks.

IV. Programming Language Support

A major challenge in future teleconferencing applications will be software development. As we have shown, high-dimensional video capture, compression and network transport are inherently parallel processes, but existing programming languages and their supporting libraries provide only limited support for concurrency. These limitations make it difficult to take advantage of parallelism, by forcing programmers to work with low-level synchronization and threading constructs which are tedious to use and error-prone. It is clear that new programming languages and tools are needed to make effective use of parallelism, to raise the level of abstraction and support automatic extraction of implicit concurrency.

There are two promising trends in this area: languages which use linear type theory to enforce thread-safe access to data structures for packet processing, and functional array processing languages which support automatic extraction of parallelism from large scale matrix and vector computations, such as those needed to implement video compression. We now consider these in turn.

A. Use of Linear Types to Manage Concurrency

The theory of linear types allows the construction of programming languages that provably enforce certain state-dependent properties of programs. One recent example of this is PacLang [19], a language developed to ease program development on network processors such as the Intel IXP2400.

PacLang was developed based on the observation that as packets flow through a router, only a single component of the system has access to the packet at any particular time instant. Such a constraint can be formalized in the type system of a programming language such that the language constrains data to be accessible by only a single thread of a multi-threaded system at any time, ensuring unique ownership as objects pass between threads. The result is a language which appears somewhat unusual to those familiar only with more mainstream languages, but which is readily learnt, and which provides strong guarantees that allow code to be optimized to execute in parallel on a distributed memory system.

We observe that data flow through networked multimedia applications is conceptually similar to the flow of packets through a router. In both cases processing follows a linear sequence of operations, transforming the data as it passes through different parts of the system, with a clear hand-off between different pipelined operations. The details are clearly very different: media capture, compression, packetization and transmission are more computationally demanding, and require more extensive transformation of the data than does IP packet forwarding, but these are differences of degree, rather than fundamentals. Both applications involve pipelined processing of a high rate stream of data from source to sink, with some intermediate transformation. Both can benefit from parallel execution and a provably clean hand-off of data between different components in the system. Accordingly, we expect the concepts of PacLang to be readily applicable to a video processing language.

B. Functional Array Processing Languages

Another promising area of research is in the development of modern array processing languages such as single assignment C [20] or J [21]. These languages use functional programming techniques to specify array processing operations with a high level of abstraction, but with a syntax usable by programmers used to mainstream languages.

Array processing languages are well suited to the implementation of video compression since they allow operations on multi-dimensional arrays of pixels to be expressed with a high level of abstraction. In addition, concepts of functional programming and immutable data structures are applicable to compression algorithms, since mutable state is not required when transforming an input media stream to a compressed output format. Indeed, many video coding algorithms are most simply described using a purely functional style, and have been implemented in low-level imperative languages such as C only for reasons of efficiency on uniprocessor systems.

With the widespread adoption of multicore processors, it is becoming more important that algorithms are parallelizable, than it is that they are efficient on uniprocessor systems. A key feature of array processing and purely functional languages is that, in addition to a clear and abstract representation of the compression algorithm, they provide a number of opportunities

for automatic extraction of parallelism, due to the data flow and side effect free nature of the resulting code. As a result, we expect such languages will become more important, since they will allow easier exploitation of the latent performance of future hardware.

C. Discussion

We believe the combination of linear types and functional array processing languages has the potential to greatly ease the programming of massively parallel hardware for networked multimedia systems. It should be possible to devise new programming languages that enforce unique ownership of data across threads of execution, and that provide opportunities for automatic extraction of parallelism. The resulting systems should scale much better than do existing imperative programming languages, should provide a higher degree of confidence that the resulting code is correct, and should allow algorithms to be expressed in a concise high-level manner. Such domain-specific languages can be embedded within existing general purpose languages to ease implementation of media processing, and to allow such operations to run on future parallel hardware with relative ease.

The transition to such languages will not, of course, be trivial. Indeed, we expect it to be a significant challenge to make these concepts accessible to a wide audience. There are many prior systems which make use of some of the ideas we discuss, yet have not found use outside particular niches. There is a reason to be optimistic however: as discussed in Section II, we expect future computing hardware to become increasingly parallel, and increasingly difficult to program to full effect using traditional imperative languages. Both are reasons to consider alternatives.

V. Conclusions and Future Directions

In conclusion, we believe that future microprocessor designs will emphasise increased parallelism over increased clock rates, and that future network architectures will provide for increased bandwidth through greater link-level parallelism, not just by providing faster links. These predictions are based on trends in semiconductor performance and power consumption, as we describe in Section II.

To take advantage of these changes will require a corresponding shift in the design of network protocols and networked multimedia applications. As we discuss in Section III, we expect to see the development of network protocols that provide richer data naming, synchronization and congestion control primitives. We suggest ways in which this can be done, drawing on lessons learnt from existing real-time transfer and reliable multicast and coordination protocols. We also discuss, in Section IV, directions in which programming languages can develop to support automatic extraction of parallelism; essential to make full use of future systems.

Network protocol, language and application development are slow processes, and deployment is a labour intensive and iterative task. To make effective use of future terabit networks, and to support emerging applications, we must initiate development of needed protocols and tools need soon, as the current single-stream protocol suite will not suffice.

VI. Acknowledgements

References

[1] L. Gharai, T. Lehman, A. Saurin, and C. S. Perkins, "Experiences with high definition interactive video conferencing," in *Proc. IEEE International Conference on Multimedia and Expo*, Toronto, July 2006.

[2] C. Slinger, C. Cameron, and M. Stanley, "Computer-generated holography as a generic display technology," *IEEE Computer*, vol. 38, no. 8, August 2005.

[3] Z. Yang, C. Y., Z. Anwar, R. Bocchino, N. Kiyanclar, K. Nahrstedt, R. H. Campbell, and W. Yurcik, "Real-time 3D video compression for tele-immersive environments," in *Proceedings of Multimedia Computing and Networking 2006*, San Jose, CA, USA, January 2005.

[4] N. A. Dodgson, J. R. Moore, and L. S. R., "Time-multiplexed autostereoscopic camera system," in *Proc. SPIE Symposium on Stereoscopic Displays and Applications VIII*, San Jose, CA, February 1997.

[5] S. Floyd, "High speed TCP for large congestion windows," Internet Engineering Task Force, December 2003, RFC 3649.

[6] C. Jin, D. Wei, and S. Low, "FAST TCP: motivation, architecture, algorithms, performance," in *Proceedings of IEEE Infocom 2004*, Hong Kong, March 2004.

[7] R. L. Grossman, M. Mazzucco, H. Sivakumar, Y. Pan, and Q. Zhang, "Simple available bandwidth utilization library for high-speed wide area networks," *Journal of Supercomputing*, vol. 34, no. 3, 2005.

[8] Y. Gu, X. Hong, and R. L. Grossman, "Experiences in design and implementation of a high performance transport protocol," in *Proc. 2004 ACM/IEEE Conference on Supercomputing*, Pittsburgh, PA, USA, November 2004.

[9] E. He, J. Leigh, O. Yu, and A. T. DeFanti, "Reliable blast UDP: Predictable high performance bulk data transfer," in *Proc. IEEE International Conference on Cluster Computing*, Chicago, IL, USA, September 2002.

[10] D. D. Clark and D. L. Tennenhouse, "Architectural considerations for a new generation of protocols," in *Proceedings of ACM SIGCOMM'90*, Philadelphia, PA, USA, September 1990.

[11] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: a transport protocol for real-time applications," Internet Engineering Task Force, July 2003, RFC 3550.

[12] S. Raman and S. McCanne, "Scalable data naming for application level framing in reliable multicast," in *Proceedings of ACM Multimedia'98*, Bristol, UK, September 1998.

[13] International Telecommunication Union, "One-way transmission time," Recommendation G.114, May 2003.

[14] D. E. Ott and K. Mayer-Patel, "Coordinated multi-streaming for 3D tele-immersion," in *Proc. 12th ACM International Conference on Multimedia*, New York, NY, USA, October 2004.

[15] J. Ott, D. Kutscher, and C. S. Perkins, "The message bus: A platform for component-based conferencing applications," in *Proc. CSCW 2000 workshop on Component-based Groupware*, Philadelphia, PA, USA, December 2000.

[16] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *Proceedings of ACM SIGCOMM 2000*, Stockholm, Sweden, August 2000.

[17] H. Balakrishnan, H. Rahul, and S. Seshan, "An integrated congestion management architecture for Internet hosts," in *Proceedings of ACM SIGCOMM'99*, Cambridge, MA, USA, September 1999.

[18] T. Lehman, J. Sobieski, and B. Jabari, "DRAGON: a framework for service provisioning in heterogenous grid networks," *IEEE Communications*, vol. 44, no. 3, March 2006.

[19] R. Ennals, R. Sharp, and A. Mycroft, "Linear Types for Packet Processing (extended version)," Cambridge University Computer Laboratroy, Technical Report TR-578, 2004.

[20] S. B. Scholz, "Single Assignment C – Efficient Support for High-level Array Operations in a Functional Setting," *Journal of Functional Programming*, vol. 13, no. 6, 2003.

[21] Jsoftware, Inc., "The J programming language," Software available online, http://www.jsoftware.com/.