TOWARDS COORDINATED DATA MANAGEMENT FOR HIGH-PERFORMANCE DISTRIBUTED MULTIMEDIA CONTENT ANALYSIS

Pierpaolo Giacomin and Alessandro Bassi
Hitachi Sopia Antipolis Laboratory
Immeuble Le Thélème, 1503 Route de Dolines
06560 Valbonne, France
giacomin@few.vu.nl
alessandro.bassi@hitachi-eu.com

Frank J. Seinstra and Thilo Kielmann
Vrije Universiteit, Computer Systems Group
Dept. Computer Science, De Boelelaan 1081a
1081 HV Amsterdam, The Netherlands
fjseins@cs.vu.nl
kielmann@cs.vu.nl

Abstract

In a few years, access to the content of multimedia data will be a problem of phenomenal proportions, as digital cameras may produce high data rates, and multimedia archives steadily run into petabytes of storage space. As a consequence, in the field of large-scale distributed Multimedia Content Analysis (MMCA) there is an urgent need for the coordinated handling of vast amounts of distributed and replicated data collections.

Any sustainable solution to the management problem of distributed (multimedia) data in Grids has to follow a layered approach, starting from a new generation of network controllers and, through a novel middleware, up to an interface to applications.

In this paper we report on our first steps of our bigger plan in this direction. Specifically, we introduce a new transport protocol layer (called HI-TP), which is capable of taking advantage of the storage capabilities of network devices, such as routers. The paper presents the protocol definition, and discusses initial performance results for basic transmission functionality. Results indicate that our protocol is capable of achieving higher throughput than traditional transport protocols, even in networks with significant latency.

Keywords: Multimedia, Grid, Network, Storage, Coupling

1. Introduction

Multimedia data is rapidly gaining importance along with recent deployment of publicly accessible digital television archives, surveillance cameras in public locations, and automatic comparison of forensic video evidence. Consequently, for emerging problems in multimedia content analysis (MMCA), Grid architectures are rapidly becoming indispensable.

Any sustainable solution to the management problem of distributed (multimedia) data in Grids has to follow a layered approach, starting from a new generation of network controllers and, through a novel middleware, up to an interface to applications. To achieve the goals of scalability, security, versatility, location independence, and delivering a Quality-of-Experience driven performance over current network infrastructures, we are developing a solution where 'intelligence' in a globally scalable system is put in the middleware, where the lower level deals with local problems only, while all other properties are taken care of at the upper layers.

Today, network devices such as routers have transparent storage capabilities: when IP packets reach a router, they are stored in a queue, waiting to be processed. An *active network caching* scheme can take advantage of such storage, and buffer streams to be released only at a later stage. To achieve this, the network nodes should expose in some way the underlying storage, thus separating the basic forwarding functions from higher control capabilities. Furthermore, a middleware layer should provide interfaces to these advanced data management properties, to be available to Grid services and applications. Research and development in this direction, including the design and standardization of new protocols, is one of the core businesses of the Hitachi Sophia Antipolis lab.

To transparently make available the newly developed protocols to application developers, an integration effort is taking place with the advanced Grid programming tools developed by the Computer Systems group at Vrije Universiteit: (1) Ibis Grid communication system and environment [17]and (2) Java version of the Grid Application Toolkit (GAT) [1]. The group at the Vrije Universiteit also is engaged in a strong collaboration with MultimediaN, a Dutch national consortium integrating over 120 researchers from leading Dutch institutes in the field of multimedia technology [15]. In this collaboration, Ibis and JavaGAT are currently being integrated with a user transparent cluster programming framework for multimedia content analysis, called Parallel-Horus [10–12]. The work described in this paper complements these ongoing efforts, and will benefit from the availability of many existing state-of-the-art multimedia applications, and large-scale multimedia data sets.

In this paper we report on our first steps towards a solution for the coordinated management of vast amounts of multimedia data, for real-time and off-line content analysis.

This paper is organized as follows. Section 2 presents the rationale of active network caching, and signals the need for a new transport protocol. In Section 3 we present an overview of state-of-the-art transport protocols, and indicate problems and drawbacks with respect to our purposes. Section 4 presents a new transport protocol, which is capable of taking advantage of the storage capabilities of network devices, such as routers referred to as HI-TP. Section 5 describes the testbed applied in our measurements, and discusses our initial performance results for basic transmission functionality. In Section 6 we give a brief overview of the set of tools in which our newly developed transport protocol will be integrated. Concluding remarks are given in Section 7.

2. Logistical Networking: Active Network Caching

As stated in the introduction, network devices such as routers have transparent storage capabilities: when IP packets reach a router, they are stored in a queue, waiting to be processed. An active, intelligent *network caching* scheme can take advantage of such storage, and buffer streams to be released only at a later stage. To achieve this, the network nodes should expose in some way the underlying storage, thus separating the basic forwarding functions from higher control capabilities.

Our work in this direction is a natural follow-up to the Internet Backplane Protocol (IBP), which aims to exploit a unified view of communication that creates a strong synergy between storage and networking [7]. IBP allows data to be stored at one location while en route from sender to receiver, adding the ability to control data movement temporally as well as spatially. This generalized notion of data movement is called *logistical networking*, drawing an analogy with systems of warehouses and distribution channels. Logistical networking can improve application performance by allowing files to be staged near where they will be used, data to be collected near their source, or content to be replaced close to its users.

Our current work, as part of the Intelligent Network Caching Architecture (INCA [2]), extends these ideas. In particular, we believe that any envisaged solution to logistical networking must be comprehensive, and must follow a layered approach, starting from a new generation of network equipment controllers and, through a novel middleware, up to the interface to applications.

The INCA system aims at *exposing* and *expanding* the storage capabilities of network equipment, such as routers. In particular, our goal is to implement an active network caching scheme that can take advantage of network storage, to buffer streams that are to be released only at a later stage. As an example, the INCA architecture aims to support files which can be replicated and scattered over the network, or at least in a defined network domain. To achieve this, we need to rethink the current architecture of network nodes, allowing them

to expose their underlying storage. In other words, the network equipment should move towards a modular, or distributed, architecture, separating the basic forwarding functions from higher control capabilities.

This paper focuses on the lowest layer of transmission protocols, in particular at the IP level. At this level we need new mechanisms and protocols to make data storage available *inside* the network boundaries. This particular storage should have very basic properties and very simple failure modes, in order to allow for maximum scalability of the framework. The middleware above, which is not the focus of this paper, would act as the transport layer in networking; it would provide all those properties that allow the framework to be useful to services and applications.

The following section gives a brief overview of state-of-the-art network transmission protocols, and indicates problems and drawbacks with respect to our purpose of efficient and reliable network caching. In the subsequent section we introduce our newly developed transport protocol, referred to as HI-TP (High-volume INCA Transport Protocol).

3. State-of-the-Art Network Transmission Protocols

The Transmission Control Protocol (TCP) is one of the core protocols of the Internet protocol suite [9]. TCP provides reliable, in-order delivery of data streams, making it suitable for applications like file transfer and e-mail. Despite its importance, TCP has many drawbacks. The most significant problem for our purposes is that TCP does not perform well on broadband high latency networks. This is because the maximal bandwidth usable by TCP is an inverse function of the latency.

An alternative to TCP is the User Datagram Protocol (UDP), which is yet another core protocol of the Internet protocol suite [8]. Using UDP, networked computers can send short messages (known as datagrams) to one another. UDP does not guarantee reliability or ordering in the way that TCP does. Datagrams may arrive out of order, appear duplicated, or go missing without notice. Avoiding the overhead of checking whether every packet actually arrived makes UDP faster and more efficient, at least for applications that do not need guaranteed delivery. Despite this increased performance potential, the lack of reliability and congestion control makes UDP not suitable for our purposes.

The problems of TCP and UDP have been acknowledged in the field, and have led to many extended and adapted data transfer protocols. One important research effort in this direction is UDT (or UDP-based Data Transfer Protocol), developed at the National Center for Data Mining (NCDM) at the University of Illinois at Chicago [4]. UDT is designed to effectively utilize the rapidly emerging high-speed wide area optical networks. It is built on top of UDP with reliability control and congestion control. A significant result of UDT is that

it was capable of reaching 711Mb/s (peak 844Mb/s) disk to disk data transfer between the United States and Russia. This result still represents the highest recorded information transfer between these countries.

Another reliable data delivery protocol, that proxy TCP connections over UDP, is Airhook [3]. Unlike TCP, Airhook gracefully handles intermittent, unreliable, or delayed networks. Other features include session recovery, congestion control, and delivery status notification. In particular, Airhook continuously transmits small status packets to keep both endpoints aware of connection status; lost data packets are transmitted immediately when their absence is reported.

Despite the impressive results obtained with UTP and Airhook, both protocols can not exploit network storage capabilities. As there is not a straightforward manner to integrate such capabilities (efficiently) within existing protocols, it is essential to define a new transport protocol for our purposes. Our proposed protocol will be introduced in the following section.

3.1 Related Approaches to Wide-Area File Transmission

Apart from low level transport protocols, there are several related efforts that must be mentioned here, and that must be compared in our evaluations. First, there is the distributed file system implementation created by Sun Microsystems, called NFS. Over the years, NFS has been the only satisfactory solution for a Network Attached Storage (NAS). Although NFS is much more than a file transfer system, it is relevant to compare NFS with our efforts, in particular due the recent trend to use NAS/SAN storage in Grid environments.

Second, the 'scp' application, part of the OpenSSH package, even if it is partially out of the scope of this paper, as it relies on SSL and TCP, is worthwhile to incorporate 'scp' in this paper's evaluation due to the fact that it is such a commonly used tool.

Finally, 'Sendfile()' is an operating system kernel primitive that aims to send a file in a 'zero copy' manner, using as little CPU time as possible. It is available in recent kernel editions of major operating systems. Sendfile() is particularly useful in our evaluations, as it allows us to send a file over TCP without any kind of overhead.

4. HI-TP: High-volume INCA Transport Protocol

In this section we introduce our new transport protocol, which we refer to as HI-TP, or High-volume INCA Transport Protocol. The aim of HI-TP is to transfer very large volumes of data in the most efficient way. The protocol is connectionless, such that it can deal with multiple peers easily and get easily proxied.

Table 1. HI-TP header

0	1-15	16	17-31
mo	offset	mr	rid
ml	length	checksum	
payload. . .			

Table 2. Simplified HI-TP header

0-15	16-31
offset	rid
lenght	checksum
mblock	Mblock
payload. . .	

A natural way to optimize data transport is to minimize overhead. To this end, HI-TP tries to identify the minimal amount of information needed to transfer a byte-sequence across the Internet successfully. In addition, HI-TP is made generic and extensible to allow it to deal with underlying heterogeneous network infrastructures and overlying applications. For these reasons, we have chosen the scheme in table 1 as the packet header of our prototype protocol implementation. A very important role is played by the most significant bits (msb) of the fields rid, checksum and offset. A more detailed explanation of the header fields is available in [2]. We believe that, using this strategy, we are able to keep the protocol extremely expandable. Each extension field is placed after the checksum in the same order as the basic fields. Each extension field of the same type is placed contiguously.

All HI-TP packets carry from a sender, identified by its IP address in the field Source Address of the IP protocol header, to a receiver, identified by his IP address in the field Destination Address of the IP protocol header, the amount of data specified by the length field of the HI-TP. When the packets arrive at the receiver side the data are stored in a buffer identified by the rid at the offset specified by the HI-TP header.

4.1 A Simplified Header

A simplified header, which is more targeted to current network infrastructures, could be considered as an alternative. For such a simplified header definition, we apply the following basic assumptions: (1) 64 KB is sufficiently large as a single packet size, (2) a peer will never have to handle more than 65536 incoming memory buffers at a time, and (3) a single dataset is limited to 256 TB. Under these assumptions our HI-TP header would look as in table 2. This simplified header definition incorporates two new field definitions: **m block**, minor block, and **M block**, major block. Not very different from offset they allows one to address blocks of 64 KB and 4 GB respectively.

With this simplified definition, the implementation performance is expected to increase significantly due to the fixed header size, and (thus) a reduced complexity of packet analysis. Still it is important to note that, even though a 256 TB addressing space is not perceived as a limit today, this may (and

probably will) change in the future. Hereinafter we will refer to this simplified version as HI-TP.

5. Performance Evaluation

The following presents the results obtained with our new protocol using a basic sender-receiver scenario.

We have performed our measurements on a testbed system comprising of three standard desktop PCs running Debian GNU/Linux "etch". The first machine, i.e. the 'sender', is a DELL Dimension 4400 with a Pentium 4 1.7 Mhz processor and 512 MB of RAM. The second machine, i.e. the 'receiver', is a DELL Dimension 4500 with a Pentium 4 2.4 Mhz processor and 512 MB of RAM. The traffic between these two hosts, or 'peers', is entirely routed through the third machine, hereafter referred to as the 'gateway/router'. In the initial stages of our measurements the gateway/router machine was a DELL Dimension L667. Due to an unfortunate malfunction of this particular machine, it has been replaced by a DELL PowerEdge 600SC. Importantly, the replacement of the gateway/router machine had no measurable effect on performance. We have performed measurements using several Fast Ethernet network cards, without noticing significant differences in the performance behavior of our testbed.

The gateway/router machine is of essential importance in our testbed system, as it is used to introduce controlled packet loss, latency, and packet reordering in a transparent way. These variations in network and transmission behavior have been incorporated using the 'netem' network emulation suite, which consist of a kernel component, and an user space extension to iproute2. Hence, all applied software components are either open source, or developed by ourselves.

5.1 Theoretical Scenarios vs. Real-world Scenarios

As it is quite difficult, if not impossible, to realistically compare protocols acting on different layers, it is important to realize that — for our purposes — it is sufficient to simply transfer a static buffer and to see under which protocol the highest throughput is obtained. For this reason, we generated our own static buffer, consisting of the first 100 Mbytes of the Linux kernel 2.6.18 tar archive. This buffer has been transmitted under varying circumstances to analyze the behavior of our system in comparison with other protocols.

In our evaluation we have followed two alternative approaches. In the first approach we have studied the behavior of our protocol under variations in emulated network latency (ranging from no additional emulated latency up to 110 ms of emulated latency), and variations in packet loss (in the range of 0-2%).

In the second approach we have tried to create a real-world scenario with all sorts of 'network perturbations' occurring concurrently: i.e., variable latency

and packet loss. We have decided to ignore issues such as packet duplication and packet corruption. Even though these issues are essential for testing a protocol, if no bugs are present in the protocol implementation, packet duplication has no significant effect on performance. Similarly, packet corruption has the same effect as packet loss, and thus can be ignored safely.

5.2 Measurements

In our first measurement (see Figure 1), we compare HI-TP, TCP and UDP under variations in emulated latency, but without any emulated packet loss or emulated packet reordering. Essentially, we should also add a small value representing real network latency.

In Figure 1 we can immediately see that TCP throughput is significantly and negatively influenced by increased latency, which is a well-known property of TCP. UDP, in contrast, performs much better under increased latency. If latency becomes significantly large, however, UDP also suffers from a significant decrease in throughput. In the graph it is clear that throughput obtained with our newly defined HI-TP protocol hardly suffers from variations in latency at all.

In our second measurement (see Figure 2) we compare the throughput performance of HI-TP, UDT, NFS, TCP Sendfile, and TCP Sendfile on top of Airhook using a link without latency (or less than 1 ms, as stated above), and no packet reordering, under variations in packet loss ranging from 0% to 2%. As can be seen in Figure 2, in this emulated scenario HI-TP again performs better than all other protocols. However, we can also observe that HI-TP is affected by packet loss in the same relative amount as all other protocols, with the exception of TCP Sendfile on top of Airhook. At this time of writing the exact reasons for this behavior of Airhook are unknown. This will be investigated further in the near future.

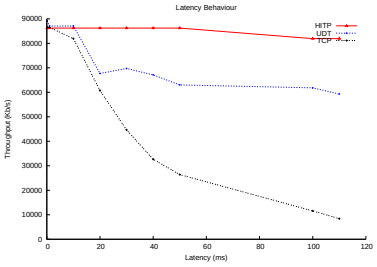

Figure 1. Theory: Throughput under latency varying from 0ms to 115ms

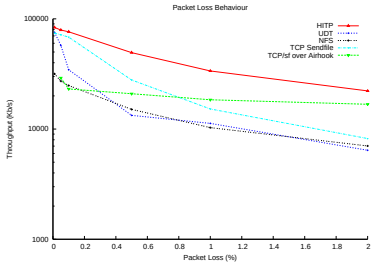

Figure 2. Theory: Throughput under packet loss varying from 0% to 2%

In our third evaluation we reconstruct a real world scenario, as can be expected on a link between Amsterdam and Paris. On such a link, the latency for http traffic is around 29 ms, with a variation of +/- 1 ms. As a consequence,

packet reordering is a realistic and common phenomenon. Hence, for real-world scenarios we must evaluate how the different protocols react to packet reordering as well. Figure 3 shows that UDT seems to perform better under at least a certain amount of packet loss (i.e., around 0.5%). This is because an increased packet loss results in an increased number of acknowledge messages, and hence and increased sensitivity to packet reordering. Figure 3 also clearly shows that HI-TP is not at all influenced by packet reordering. Essentially, this is because the HI-TP buffer in the receiver is addressed directly from the packet.

In our final measurements (see Figure 4) we compare the absolute time for a file transfer as obtained in user space for UDT, scp, and HI-TP. In this measurement 25% of the packets (with a correlation of 50%) will be sent immediately, while all others will be delayed by 10 ms. Also, the scp measurements have been performed without any latency or packet reordering introduced, just to use the optimal scp case as a reference, for a real user-perspective comparison. In Figure 4 we can again observe that HI-TP performs best, and that it is not at all affected by packet reordering. More importantly, end-users are not at all bothered by any such troubles on the underlying network.

Based on all our measurements we conclude that HI-TP is only marginally influenced by increased network latency, and not at all affected by packet reordering. HI-TP is still vulnerable to packet loss, but this is a condition which is rather unusual in high-speed optical Grid network links.

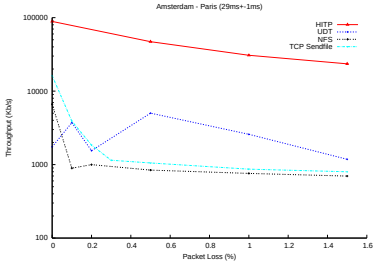

Figure 3. Real world: a file transferred from Amsterdam to Paris

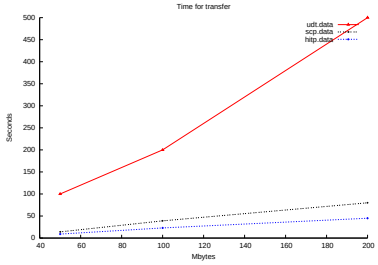

Figure 4. Time for transferring a file: scp in a LAN environment against HI-TP and UDT under packet reordering conditions

6. Future work

The HI-TP protocol initially is intended to be applied in a set of tools that allows straightforward development of high-performance distributed multimedia applications, which are to be executed on a very large scale. To this end, the Vrije Universiteit is developing a set of advanced Grid programming tools, each of which will be described briefly in the following. This section concludes with a brief description of a set of target applications.

6.1 Java-GAT

Today, Grid programmers generally implement their applications against a Grid middleware API that abstracts away part of the complexities of the underlying hardware. This is a daunting task, because Grid APIs change frequently, are generally low-level, unstable, and incomplete [6]. The Java Grid Application Toolkit (JavaGAT [16]), developed at the Vrije Universiteit, solves the above problems in an integrated manner. JavaGAT offers high-level primitives for access to the Grid, independent of the Grid middleware that implements this functionality. JavaGAT integrates multiple middleware systems into a single coherent system. It dynamically forwards calls on the JavaGAT API to one or more middlewares that implement the requested functionality. If a Grid operation fails, it will automatically dispatch the API call to an alternative Grid middleware. Using JavaGAT, Grid applications can, among other functionality, transparently access remote data and spawn off jobs within Grid installations. JavaGAT further provides support for monitoring, steering, resource brokering, and storing of application-specific data. The importance of JavaGAT is illustrated by the fact that, based on these efforts, the Open Grid Forum is now standardizing the next generation Grid programming toolkit: the Simple API for Grid Applications (SAGA).

JavaGAT is freely available at: https://gforge.cs.vu.nl/projects/javagat/.

6.2 Ibis

Grid systems, by nature, are open world and faulty, meaning that resources can be added and removed at will, and crash at any moment. Thus, to ensure robust Grid execution, application programmers must be handed the basic functionality to allow their applications to be made malleable, such that processors can be added and removed at application run-time.

Ibis [17], developed at the Vrije Universiteit, is a platform-independent Grid programming environment that combines flexible treatment of dynamically available resources with highly efficient object-based communication. Given his results it follows that Ibis provides a stable, efficient, and platform-independent communication layer that supports the open, dynamic, and faulty nature of real-world Grid systems.

Ibis is freely available at: http://www.cs.vu.nl/ibis/.

6.3 Parallel-Horus

Whereas JavaGAT and Ibis provide the basic functionality to allow supercomputing applications to execute efficiently and transparently in a Grid system, it is still up to the programmer to identify the available parallelism in a problem at hand. For the application programmer — generally a domain-expert with

limited or no expertise in the field of supercomputing — this is often an insurmountable problem. Clearly, there is a need for programming models that either alleviate the burden somewhat or, preferably, shield the user from all intrinsic complexities of parallelization.

Parallel-Horus [11, 10, 12]is a cluster programming library that allows its users to implement parallel multimedia applications as fully sequential programs, using a carefully designed set of building block operations. These algorithmic patterns of parallel execution cover the bulk of all commonly applied multimedia functionality, while hiding all complexities of parallelization behind a familiar sequential API (identical to that of an existing sequential multimedia computing library: Horus [5]). Notably, Parallel-Horus was applied in recent international benchmark evaluations for content-based video retrieval, and played a crucial role in achieving top-ranking results [13–14]. A proof-of-concept implementation of Parallel-Horus is freely available at http://www.science.uva.nl/˜fjseins/ParHorusCode/.

6.4 Integration of Tools

The Parallel-Horus programming model is being extended for use in Grid systems [10]. To this end, an innovative runtime system (RTS) is being developed, based on the notion of a SuperServer — i.e., a lightweight server implementation that allows for uploading of data as well as program codes (i.e., transferring Java byte codes by way of dynamic class loading over the network). With JavaGAT's transparent job submission functionality, the RTS starts a pool of SuperServers on behalf of a coordinating (client) application, hidden from the user. The pool of SuperServers is then applied by the RTS on behalf of the client application on a need-be basis, by uploading objects (data) and related program codes. Importantly, in this way — and in contrast to the common use of (web) services in Grid computing — it is the application programmer who implements the actual program code executed by a SuperServer. In case the uploaded code represents a (sequential) sequence of Parallel-Horus calls, the SuperServer, when running on a cluster, executes these calls transparently in data parallel fashion. By transparently outsourcing the running of Parallel-Horus code to multiple SuperServers in a fully asynchronous manner, we arrive at a 'wall-socket' computing solution: *transparent task parallel execution of data parallel services*.

6.5 Target Applications

The research group at the Vrije Universiteit closely collaborates with the University of Amsterdam on urgent MMCA problems, classified in two groups: real-time analysis and off-line categorization. Realistic examples from both groups include (1) the comparison of objects and individuals in video streams

obtained from surveillance cameras, as researched in close cooperation with the Dutch Forensic Institute (NFI), (2) iris-scan based identification and automatic fingerprint checks (e.g. to be performed at international airports), also with the NFI, (3) the international NIST TRECVID evaluation [14], i.e. the yearly bench-mark evaluation for approaches to finding semantic concepts in archives of TV news broadcasts from (a.o.) ABC and CNN, (4) interactive access to Petabytes of current and historic TV broadcasts, as researched in close collaboration with the Dutch Institute for Sound and Vision (Beeld&Geluid). These examples represent urgent MMCA problems that will serve as targets in our future research.

7. Conclusions

In this paper we have presented and validated a new transport protocol (HI-TP) conceived for active network caching, initially intended for coordinated data management in large scale distributed multimedia computing applications. To this end we have presented the rationale of logistical networking and active network caching, and indicated the need for a new data transmission protocol. In our overview of state-of-the-art transport protocols we have indicated problems and drawbacks with respect to our purposes, mostly from an end-user application and integration perspective. We have presented the new protocol, referred to as HI-TP, or High-volume INCA Transport Protocol, focusing on its ability of addressing, potentially, unlimited buffers in a simple manner. We have performed an initial set of measurements using our new protocol on a simple testbed environment, and compared our results with those obtained with other protocols. Our main conclusions are that HI-TP performs better than existing protocols, under many realistic variations in transmission and network behavior. Most importantly, HI-TP is only marginally influenced by increased network latency, and not at all affected by packet reordering. HI-TP is still vulnerable to packet loss, but this is a condition which is rather unusual in high-speed optical Grid network links. Finally, we have given a brief overview of the set of tools in which our newly developed transport protocol will be integrated in the near future.

Acknowledgments

This research is carried out under the FP6 *CoreGRID* Network of Excellence funded by the European Commission (Contract IST-2002-004265). This paper's first author is supported via CoreGRID's Industrial Fellowship Programme under grant no. CIFP-4/06.

References

[1] G. Allen, K. Davis, T. Goodale, A. Hutanu, H. Kaiser, T. Kielmann, A. Merzky, R. van Nieuwpoort, A. Reinefeld, F. Schintke, T. Schütt, E. Seidel, and B. Ullmer. The Grid Application Toolkit: Towards Generic and Easy Application Programming Interfaces for the Grid. *Proceedings of the IEEE*, 93(3):534–550, 2005.

[2] A. Bassi, S. Denazis, and P. Giacomin. Towards a Noah's Ark for the Upcoming Data Deluge. In *3rd VLDB Workshop on Data Management in Grids*, Sept. 2007.

[3] D. Egnor. Airhook: Reliable, Efficient Transmission Control for Networks that Suck.

[4] Y. Gu and R. Grossman. UDT: UDP-based Data Transfer for High-Speed Wide Area Networks. *Computer Networks*, 51(7):1777–1799, May 2007.

[5] D. Koelma et al. Horus C++ Reference, Version 1.1. Technical report, ISIS, Faculty of Science, University of Amsterdam, The Netherlands, Jan. 2002.

[6] R. Medeiros, W. Cirne, F. Brasileiro, and J. Sauvé. Faults in Grids: Why are they so bad and What can be done about it? In *Proceedings of the Fourth International Workshop on Grid Computing*, 2003.

[7] J. Plank, A. Bassi, M. Beck, T. Moore, M. Swany, and R. Wolski. Managing Data Storage in the Network. *IEEE Internet Computing*, 6(5):50–58, Sept. 2001.

[8] J. Postel. User Datagram Protocol, Aug. 1980. Internet standard RFC 768.

[9] J. Postel. Transmission Control Protocol, Sept. 1981. Internet standard RFC 793.

[10] F. Seinstra, J. Geusebroek, D. Koelma, C. Snoek, M. Worring, and A. Smeulders. High-Performance Distributed Video Content Analysis with Parallel-Horus. *IEEE Multimedia*, 15(4):64–75, Oct. 2007.

[11] F. Seinstra, D. Koelma, and A. Bagdanov. Finite State Machine-Based Optimization of Data Parallel Regular Domain Problems Applied in Low-Level Image Processing. *IEEE Transactions on Parallel and Distributed Systems*, 15(10):865–877, Oct. 2004.

[12] F. Seinstra, C. Snoek, D. Koelma, J. Geusebroek, and M. Worring. User Transparent Parallel Processing of the 2004 NIST TRECVID Data Set. In *Proceedings of the International Parallel & Distributed Processing Symposium (IPDPS 2005)*, Denver, Colorado, USA, Apr. 2005.

[13] C. Snoek, M. Worring, J. Geusebroek, D. Koelma, F. Seinstra, and A. Smeulders. The MediaMill TRECVID 2005 Semantic Video Search Engine. In *Proceedings of the 3rd TRECVID Workshop*, Gathersburg, USA, Nov. 2005.

[14] C. Snoek, M. Worring, J. Geusebroek, D. Koelma, F. Seinstra, and A. Smeulders. The Semantic Pathfinder: Using an Authoring Metaphor for Generic Multimedia Indexing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1678–1689, 2006.

[15] Stichting MultimediaN. *MultimediaN, Project proposal Bsik*, Feb. 2003. http://www.multimedian.nl/.

[16] R. van Nieuwpoort, T. Kielmann, and H. Bal. User-friendly and Reliable Grid Computing Based on Imperfect Middleware. In *Proceedings of SC'07*, Nov. 2007.

[17] R. van Nieuwpoort, J. Maassen, G. Wrzesinska, R. Hofman, C. Jacobs, T. Kielmann, and H. Bal. Ibis: a Flexible and Efficient Java-based Grid Programming Environment. *Concurrency and Computation: Practice and Experience*, 17(7–8):1079–1107, 2005.