IQ-Services: Network-Aware Middleware for Interactive Large-Data Applications

[Extended Abstract]

Zhongtang Cai, Greg Eisenhauer, Qi He,
Vibhore Kumar, Karsten Schwan, Matthew Wolf
{ztcai,eisen,qhe,vibhore,schwan,mwolf}@cc.gatech.edu
College of Computing
Georgia Institute of Technology

ABSTRACT

IQ-Services are application-specific, resource-aware code modules executed by data transport middleware. They constitute a 'thin' layer between application components and the underlying computational and communication resources that implements the data manipulations necessary to permit wide-area collaborations to proceed smoothly, despite dynamic resource variations. IQ-Services interact with the application and resource layers via dynamic performance attributes, and end-to-end implementations of such attributes also permit clients to interact with data providers. The joint middleware/resource and provider/consumer interactions implemented with performance attributes may be used to realize effective methods for managing the data flows in the large-data, distributed Grid applications targeted by our research. Experimental results in this paper demonstrate substantial performance improvements attained by coordinating network-level with service-level adaptations of the data being transported and by permitting end users to dynamically deploy and use application-specific services for manipulating data in ways suitable for their current needs.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design Network—*communications*; C.2.2 [**Computer-Communication Networks**]: Network Protocols—*applications, protocol architecture*; C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network monitoring*; C.2.5 [**Computer-Communication Networks**]: Local and Wide-Area Networks—*high-speed, Internet*; C.3 [**Special-Purpose and Application-Based Systems**]: Real-time and embedded systems; C.4 [**Performance of Systems**]: Design Studies, Performance Attributes

General Terms

design, measurement, performance

Keywords

middleware, network-aware, coordinated adaptation, adaptive transport, wide-area collaboration, large-data transfer, distributed Grid application

2nd Workshop on Middleware for Grid Computing Toronto, Canada

1. INTRODUCTION

Distributed applications that 'stress' wide area networks include telepresence, remote collaboration and visualization, remote instrument access and control, and real-time monitoring and surveillance. Problems arise both from their large data volumes and from their interactive nature, the latter requiring data to be transferred with low latency and high predictability. Factors contributing to these problems include the heterogeneity and dynamics of underlying networks, the lack of support for QoS at the network level, and TCP's end-to-end congestion control that results in bursty network traffic coupled with the delivery of unstable QoS over time [26]. In addition, it is difficult to measure available network bandwidth, especially when collaborators or remote users must utilize shared substrates like the Internet.

Our research is developing middleware-based methods of providing to interactive wide area applications the data they need at acceptable levels of quality. The idea of the *IQ-Services resource-aware middleware* is to have middleware execute application-defined services for data filtering, transformation, and scheduling, and to continuously adjust such services' behaviors in accordance with current resource availability and client needs. A key, new aspect of IQ-Services is that they can be installed at runtime, initiated by applications, and used whenever or wherever appropriate. This paper's specific focus is on network behavior, which is captured with dynamic bandwidth measurement [16, 19] techniques. An example is a new IQ-Service, a data downsampling filter installed by a visualization client, which uses cutting planes

[1]This work is supported in part by the NSF ANIR program and by the DOE MICS High Performance Networks program.

in order to reduce the amount of data being transmitted to maintain acceptable data rates despite variations in available network bandwidth [15]. As a result and in contrast to adaptive applications or application-specific adaptation frameworks like those developed for multimedia systems [29, 8], IQ-Services are useful for a wide range of applications and techniques. Other examples of in-stream data filtering explored in our own past work include data reordering, prioritization, and downsampling [14], the use of application-specific [28] or general [27] methods for data compression, and data transformations that implement tradeoffs in the amounts of processing vs. data volumes in the overlay networks middleware uses for service execution [5].

IQ-Services target high performance applications, and their attainment of high performance is in part because we separate the IQ-Service middleware layer from other functionality required for dynamic adaptation, such as online performance monitoring. Instead, IQ-Services use dynamic *performance attributes* to 'link' the services being adapted with the monitoring and decision-making facilities that drive adaptation processes. In our implementation of IQ-Services with the IQ-ECho middleware [12], for instance, monitoring is performed by the underlying communication protocol. This protocol, termed IQ-RUDP [14], is a reliable UDP(RUDP)-based protocol that implements instrumentation less general than but akin to the instrumented Linux protocol stacks developed in the Web100 project [19]. Per message monitoring information includes current RTT (Round Trip Time) and loss rate. Beyond its ability to emit monitoring information, IQ-RUDP also implements user-level callbacks. The resulting composite functionality extends that of MIT's communication congestion manager [4], in that IQ-Services and IQ-RUDP do not limit service adaptations to the application level. Specifically, additional protocol-level adaptations use application-provided performance attributes like desired loss tolerance or even per message attributes like 'priority markups', thereby enabling fine-grain, rapid reactions to changes in network state. Finally, performance attributes have other interesting uses, including (1) to carry information relevant to network communications, (2) to capture system-level information (e.g., provided by the DProc OS-level mechanisms described in [3]) , and (3) to indicate client-level service desires, as demonstrated in this paper with a client that specifies currently desired cutting plane positions for high end remote data visualization.

This paper's contributions include (1) experimental validations of our own earlier simulation- and emulation-based results [14], in which we had shown that coordinated application/middleware-, and network-level adaptations can outperform application-only adaptation methods; (2) an overview of the middleware architecture that enables network-aware service adaptation ; and (3) a demonstration of the architecture's generality by applying it to grid applications other than those written by our group, specifically, by creating and evaluating a network-aware version of GridFTP, termed *IQ-GridFTP*. IQ-GridFTP dynamically adjusts file contents during transfer (via user-supplied data manipulation services) in order to maintain high transfer rates under varying network conditions.

2. DATA-INTENSIVE, DISTRIBUTED APPLICATIONS

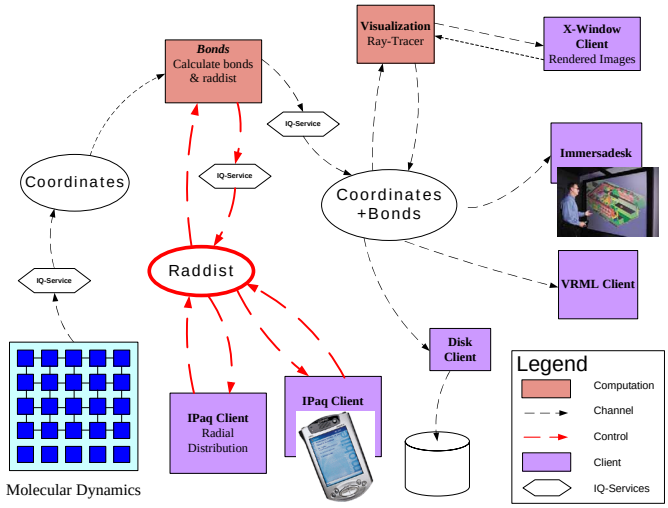

Figure 1: SmartPointer System Overview

There are multiple, ongoing, wide-area collaborations in scientific computing, ranging from high profile efforts like Ligo [18, 13] and DOE's Terascale Supernova Initiative (TSI) [9] to adhoc collaborations between specific sets of researchers. Collaboration tools like Access Grid [2] and grid portal efforts are evidence of the importance of remote collaboration to the broader CS and scientific communities. Our research uses an interactive high performance application, termed SmartPointer [28], to evaluate the utility of IQ-Services. This application permits remote users to collaboratively view and manipulate data produced by a molecular dynamics simulation [28]. Users can specialize data by dynamically deploying suitable middleware-level transformation and filtering services, whose behaviors are adjusted at runtime in accordance with available network resources.

2.1 Real-time Collaboration

Figure 1 shows a prototypical, distributed collaborative visualization [1, 28]. It implements a many-to-many dataflow, and the heterogeneous underlying computing/communication platform ranges from low end, ill-connected clients to high end, well-connected server machines. Collaborative environments like these pose problems to developers both because of the heterogeneous platforms and dynamic user behaviors, and because of inherent capabilities expected by end users. One issue is the coordination of data delivery to multiple collaborators, since collaboration activities may be compromised if the information for one collaborator consistently lags that of the others. The resulting soft real-time constraints on data delivery imply a need for consistent frame rates (to insure data freshness), perhaps in preference over always receiving data at the highest levels of resolution. Another issue is fairness across multiple connections, which is particularly important when large data transfers share network links with remote control loops. Sample loops in our application include those that control a remote data filter and those that implement dynamic annotations on clients' displays.

2.2 Large-Data Transfers via IQ-GridFTP

IQ-GridFTP extends the Globus GridFTP implementa-

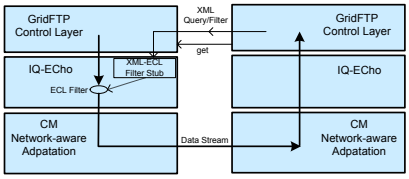

Figure 2: IQ-GridFTP Overview

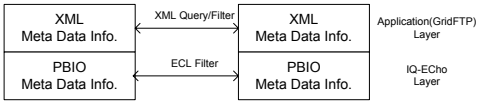

Figure 3: IQ-GridFTP MetaData Mapping

tion [11] by realizing its goal of partial file transfers, that is, to transfer both entire files and also certain regions of files. The GridFTP standard represents this functionality with ERET (Extended Retrieve) and ESTO (Extended Store) file access instructions, which include an offset and size parameter to fetch relevant file portions. In the future, we expect file manipulations and transfers to be able to utilize the DFDL (Data Format Description Language) currently under development. This language will provide the high-level descriptions of file contents needed for realizing finer grain content filtering, down to the level of individual file data attributes.

Our intent is to utilize partial file transfers to limit network usage from a server to specific clients, by deploying customized filters at the server-resident data source. Figure 2 demonstrates such functionality by placing a client-specific XML-based, SQL-like query with the server, which then ensures that only relevant file portions and/or file attribute data are transferred. This exploits the fact that data sharing in scientific applications means transferring data on the order of GigaBytes, from one location to another, even when all the data attributes and rows might not be needed or when certain attributes can be combined to reduce the number of bytes being transferred per row [22]. Finally, as with the real-time collaborations outlined above, the amounts of data transmitted by filters can also be adjusted to match available network bandwidth.

3. IQ-SERVICES: SOFTWARE ARCHITECTURE

3.1 Architecture Overview

IQ-Services are application-specific code modules associated with data transport middleware. Figure 4 shows a prototypical real-time collaboration in which application-level data is distributed to remote collaborators and is manipulated by computational components like data clustering or data analysis [22]. The role of IQ-Services in this scenario is illustrated by the additional data filters and selectors associated as 'handlers' with the publish/subscribe event channels used for data distribution. As evident from the figure, IQ-Services do not replace application-level computations like data analysis or clustering. Instead, they form a 'thin', efficient layer of application-provided functionality that is placed 'into' middleware by applications, the purpose of which is to allow middleware to manipulate data on its path from providers to consumers and to do so in conjunction with information about network behavior provided by communication protocols. In the IQ-ECho publish/subscribe infrastructure used in our implementation, this layer is comprised of dynamically created and deployed 'event handlers'.

IQ-Services may be associated with clients, servers, or intermediate nodes, thereby forming an overlay network. Typical configurations of overlays used in interactive scientific applications are described in [22, 7]. Overlays should be created and changed during an application's execution, due to dynamic changes in end user behavior or when indicated by substantial changes in network or machine resources. IQ-ECho supports both the runtime creation of overlay nodes and the installation of services on those nodes, using dynamic binary code generation techniques [12].

Performance attributes implement IQ-Service/monitoring interactions, which in this paper, are focused on service/protocol information exchanges. Such attributes traverse multiple layers of the protocol stack (1) to provide to the protocol layer certain application-level information like packet priorities or importance to communication protocols, and (2) to provide network-level monitoring information like available bandwidth or current packet loss to middleware-level services. Attributes may also be used to implement end-to-end performance-relevant interactions between data providers and consumers. Examples include a client's use of performance attributes to set parameters in a server's data filter, and a server-side instruction of the middleware handler to upsample the data sent since additional network bandwidth is now available. End-to-end and cross-layer interactions via performance attributes are depicted by the solid, vertical arrow in Figure 5.

Figure 5 also shows how IQ-ECho maps an application-level message submitted to an event channel to a message sent to the underlying protocol stack and communication socket used by a specific source-sink pair: after event submission, the event is mapped to a lower-level facility called the 'Communication Manager', which then sends the event to one of multiple communication protocols. This permits IQ-ECho to run on top of multiple transport protocols, including the standard TCP protocol and an instrumented version of RUDP developed in our research, termed IQ-RUDP. The latter protocol employs performance attributes (1) to provide feedback to the application-level handler also shown in the figure, (2) implements *callbacks* to applications along with the conditions under which they should be triggered, (3) provides prioritized reliability control for each application-level message, and (4) participates in the implementation of adaptations that are coordinated between the application and the transport (see [14] for additional detail on this protocol). In comparison, for TCP, network measurement capabilities are directly associated with communication services, as depicted in Figure 5 by the 'measurement' methods layered between event management and the communication manager. By placing such methods directly 'into' the communication stream, measurements can be performed using the application's native communications, rather than with additional packet trains generated for such purposes. This is useful for the streaming data applications targeted by our work, but should be complemented by active bandwidth measurements for applications with intermittent or bursty communication behaviors.

For instrumented protocols like IQ-RUDP, the Dynamic Adaptation Layer (DAL) shown in Figure 5 layer moder-

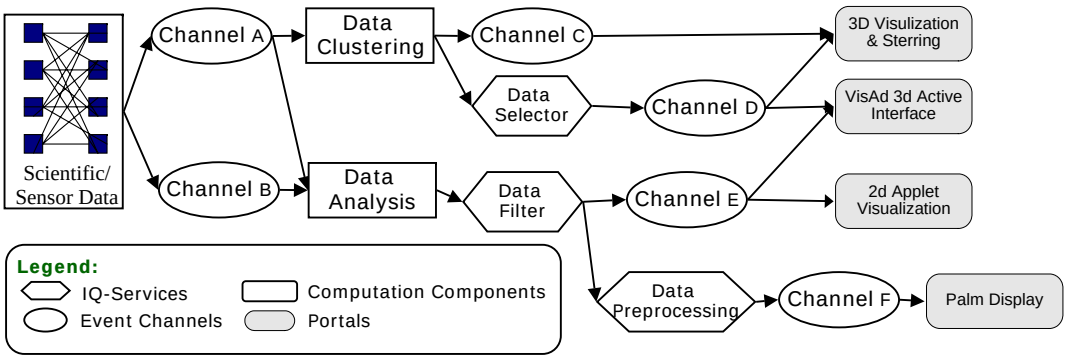

Figure 4: Typical Application Scenario

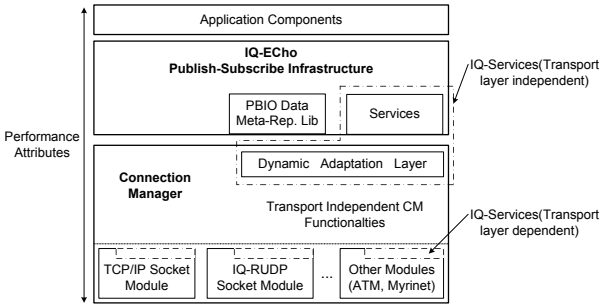

Figure 5: System Architecture Overview

ates between protocol- and application-level semantics. The DAL can also implement service-specific performance models. Model outputs exported via performance attributes then 'drive' adaptation methods realized in the DAL, which in turn drive the actual adaptations performed by IQ-Services. Current methods implemented in our work include packet reliability control and coordination across multiple, concurrent transport-level connections. Performance 'models' utilized in our work employ (1) linear bandwidth regression and (2) an end-to-end method for measuring available bandwidth. Finally, the DAL can use additional threads to execute adaptation methods and/or periodically run network measurement tools like Pathload [16] or Netlets [23].

We have developed and deployed a variety of adaptation methods in the DAL. They include (1) coordinated packet Reliability control, which permits applications to define different levels of packet reliability requirements, which are then handled in a coordinated fashion by the combined actions of the IQ-Service/DAL/IQ-RUDP layers, (2) congestion avoidance that uses application-level methods to control the traffic volume imposed on the underlying network, and (3) connection coordination implementing various schemes for imposing traffic on different or multiple network connections used for data transmission. An example of (3) is a collaboration-friendly 'fairness' scheme that improves synchronization among collaborators, by reducing packet rates for flows favored by the network.

4. EXPERIMENTAL EVALUATION

Experimental results demonstrate the ability of the IQ-Services architecture and implementation to dynamically manage traffic behavior in response to network resource availability:

- demonstrations that coordinated network- and middleware-level adaptation can substantially improve the performance of wide area applications;
- sample adaptations performed for interactive high performance applications, focusing on online collaboration via large data sets; and
- comparisons of results attained with different network measurement methods.

Experimentation is performed across wide-area network links and on the NetLab network emulation facility(an Emulab site). Some measurements involve the use of IQ-ECho to transport large-scale interactive data across a 10Gbps link connecting Georgia Tech with machines located at Oak Ridge National Laboratories. The goal is to demonstrate the feasibility of wide-area collaborations like those required in DOE's Supernova Initiative, where multiple remote collaborators seek access to output data produced at rates approximating 1GB/sec for future large-scale simulation scenarios.

4.1 Coordinating the Transfer of Two IQ-RUDP Connections

In this experiment, the DAL layer coordinates message transmissions across two IQ-RUDP connections used by a single application. The experiment emulates two remote clients that receive the same visualization data stream. Since these two clients are engaged in a real-time collaboration, their desire is to receive and display the same data at the same time. The goal of DAL-based coordination in that case is to improve synchronization between the two clients, thereby reducing buffering costs and increasing the 'freshness' of information received by each client [17].

The experiment uses two connections that share a bottleneck link but have different RTTs and therefore, different throughput. Based on the common TCP throughput model[1], the throughput of a TCP connection will be inversely proportional to its RTT. To better synchronize the two connections, the DAL layer performs rate adjustment based on the RTT information of each connection, made available by the IQ-RUDP layer. Suppose the DAL performs rate control for every G packets sent on the faster connection, and the RTTs of the two flows are RTT_1 and RTT_2, respectively. If $RTT_1 < RTT_2$, the DAL simply adjusts the rate of connection 1 by adding a delay of $RTT_2 - RTT_1$

[1] $T = \frac{1.22}{RTT * \sqrt{b*p}}$ [21]

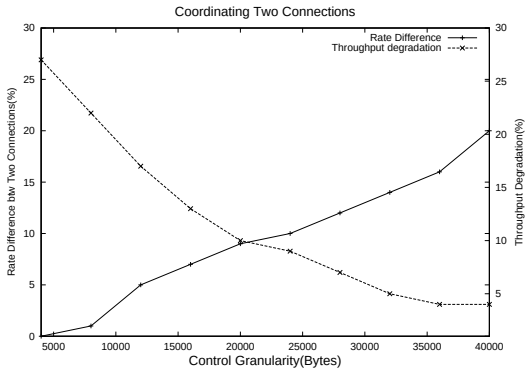

Figure 6: Coordination between Connections

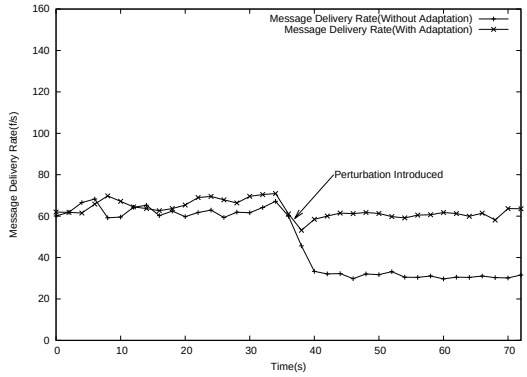

Figure 7: Adaptive Downsampling(ORNL link)

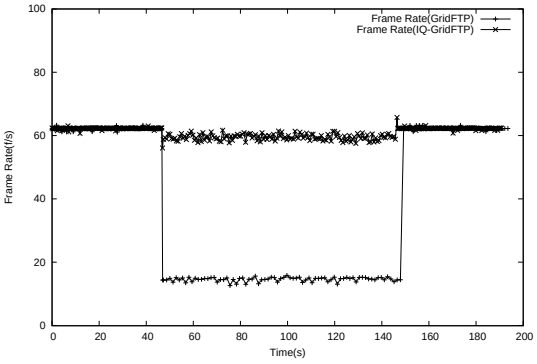

Figure 8: IQ-GridFTP Adaptation(60Mbps Cross Traffic)

to its packets. With this adjustment, the connections 1 and 2 then experience similar RTTs. For the results presented in Figure 6, the parameters without rate adjustment are: $RTT_1 = 2.4ms$, $RTT_2 = 6.6ms$, and loss rate $p = 0.6\% - 1.2\%$.

This experiment has two interesting outcomes. First, it demonstrates the viability of application-level, network-aware control over multiple connections. Second, it evaluates 'granularity' issues concerning such control. The purpose of the latter is to better understand potential control limitations due to application constraints like delays in control due to threads scheduling or delays due to differences in the sizes of application-level messages being sent. In Figure 6, the X-value is the size of the data unit G over which a single rate adjustment is effective. The left Y-value is the receiving rate difference between the two receivers. The right Y-value is the throughput degradation of the faster connection. The cross-over point at packet sizes of about 20KBytes represents a reasonable point, where the faster connection's throughput is degraded by about 10% and both connections experience a rate difference of no more than 10%. It is also evident that finer grain control achieves better synchronization.

There are multiple real-life scenarios in which this 'coordination' solution is of interest. One is the co-existence of control and data connections in distributed applications, where a single, large data flow should not be able to unduly delay control flows concerning such data. Another is the co-existence of related flows, such as video and visualization data streams in real-time collaboration.

4.2 Adaptive Downsampling in Congested Networks

In real-time collaboration, it may not be viable to further reduce the rate at which data is provided to end users, since that may violate timeliness guarantee or may cause different end users to get 'out of sync' with respect to the data they are jointly viewing. A solution is to deploy application-level data filters as IQ-Services to downsample the actual data being sent prior to submitting it to the network transport. IQ-Services can regulate the traffic imposed on the underlying network by 'pacing' application-level messages to effectively reduce congestion and maintain better message delivery rates. Furthermore, clients can deploy exactly the filters they wish when and if they need them, and filters can be controlled to deliver the best quality data permitted by current network conditions. This is in stark contrast to earlier work on multimedia data services that offered only a few built-in, rigid data downsampling methods [10].

Experimental results(Figure 7) demonstrate the necessity and effectiveness of IQ-Services-level adaptation through dynamic data downsampling. Here, large cross-traffic (250Mbps) is injected as a controlled perturbation into the link from **isleroyale** at Georgia Tech to **cruise** at ORNL. The network bottleneck is at Georgia Tech's edge router. When permitting the client to characterize the subset of data most important to it, the client installs an IQ-Service data filter at the server side when congestion occurs, and henceforth receives only 'essential' (i.e., as defined by the deployed filter) data at satisfactory speeds. The specific data downsampler used in these experiments removes data relating to visual objects that are not in the user's immediate field of view. That is, the client transfers the current position and viewpoint of the user to the filter (i.e., using attributes), at the server side these values are used to determine what data set the user is currently watching, and that information is then used to transfer appropriately downsampled data to the client. The result is a consequent reduction in the network bandwidth used for data transmission, thereby speeding up data transmission.

This experiment demonstrates the utility of network-initiated data downsampling for maintaining high data rates for limited network bandwidths. By giving end users the ability to define their own IQ-Services for implementing data downsampling, similar methods can be applied to other applications, as shown by our past work on real-time and multimedia applications [24, 25] , with services that implement general rather than application-specific compression methods [27], and by the next sections results attained with GridFTP.

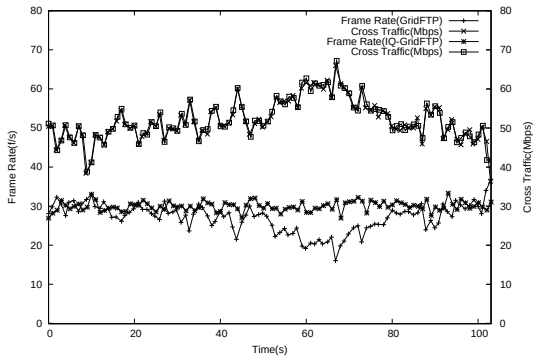

Figure 9: IQ-GridFTP Adaptation(Dynamic Cross Traffic)

4.3 Experimental Results with IQ-GridFTP

Our implementations of IQ-GridFTP replaces the transport layer of GridFTP with IQ-ECho, thereby making it easy for end users to deploy desired IQ-Services 'into' FTP transfers. Such filters may be deployed statically, or via dynamic linking methods, or written with E-Code, a highly portable subset of C, which is dynamically executed at the source. XML is used to represent the structure of data contained in the files transported by IQ-GridFTP. Specifically, whenever a client needs to do a partial retrieve on a structured file, say weather.dat, a format description file, say weather.dat.xml (we assume that such a file exists for large structured files), is first fetched from the source. Next, the requesting client creates an IQ-ECho structured data channel using the information from the xml file, and then generates E-Code for the filter using the information supplied. Using IQ-ECho's facilities for dynamic deployment, the IQ-Service that realizes this filter can dynamically: (1) select the specific file attributes that need to be transferred, (2) select the rows required, and (3) perform data reduction operation like averaging, etc., at the data source, thereby reducing the data volumes transmitted.

We compare the performance of the IQ-GridFTP with basic GridFTP in several ways. First, experiments reported in [6], compare the two GridFTP implementations' effective throughput with and without cross traffic; they demonstrate that IQ-GridFTP achieves throughput similar to that of GridFTP. Second and more interestingly, Figure 8 demonstrates the advantage of utilizing IQ-Services in conjunction with GridFTP. Here, a large number of data files are transferred from an FTP server to a client. In IQ-GridFTP, when the client finds the frame rate to be lower than desired, it creates a data downsampling filter (an image resolution reduction filter when data is being visualized) and deploys it on the server, then uses the filter to reduce data volume, when necessary.

Figure 9 compares the performance of GridFTP and IQ-GridFTP with dynamic cross-traffic derived from an actual trace[2]. Cross-traffic is injected via UDP, thereby reasonably emulating actual network behavior. The data the server transfers to the client is climate data (the size of each record/frame is 172.8K bytes). The client sends a filter to the server to specify whether it wants IQ-GridFTP to adjust data precision when congestion is noticed and if so, what percentage of the data will be adjusted. While the frame rate targeted by the client is 30f/s, the acheived frame rate(averaged over 5 experiments) is improved from 27.34f/s to 30.03f/s, where the normalized standard deviation of frame rate is reduced from 0.11 to 0.04.

Additional capabilities of IQ-GridFTP now under development include per connection adaptations like dynamic window size adjustments, as well as fairness improvements or stream synchronization when multiple connections are used for single, large file transfers. The latter is important for storage systems (e.g., DPSS, HPSS) that utilize parallel data transfers and data striping across multiple servers to improve performance.

[2]Trace file BWY-1063337799-1 from NLANR trace repository(pma.nlanr.net), collected at Columbia University (BWY site) on Sept. 12th, 2003.

5. CONCLUSIONS AND FUTURE WORK

The software architecture of IQ-Services shown in Figure 5 offers developers the ability to insert application-specific, lightweight services into data exchange middleware. In this paper, such 'IQ-Services' dynamically adjust the data sent from a distributed scientific collaboration's data providers to its data consumers [18, 13, 9], to guarantee stable rates of data delivery despite runtime variations in available network resources. IQ-Services are also used to implement adaptive file transfers via an IQ-version of GridFTP. We have also used IQ-Services to create resource-aware communication services that apply general compression methods to data being exchanged across wide area networks [27], and we are currently using them to create adaptive remote graphical displays, for the high end 3D visual depictions required by applications like molecular dynamics. Finally, the IQ-Services software architecture has been shown useful for other communication paradigms, such as the M-by-N data exchanges used in remote storage, monitoring, or visualization systems.

The performance improvements attained by use of IQ-Services can be substantial, including up to 25% improvements in message delivery rates when information sources 'pace' the data offered in conjunction with available network bandwidth, and almost threefold improvements in message rates when a client-specific data downsampling service is used to control the amounts of data sent from data server to client.

Future work will compare service-level adaptations that utilize different network-level techniques for assessing current network bandwidth [23, 20]. In addition, it will utilize overlay networks to combine the lightweight data filtering and downsampling methods used in this paper with heavier-weight methods for data transformation and personalization executed by additional machines interposed into the path between data providers and consumers [5, 22], and/or to utilize alternative or concurrent network and machine paths from data providers to consumers.

Acknowledgments

We acknowledge the help of Constantinos Dovrolis and Nagi Rao in defining the network testbeds used in this paper's experimentation. Nagi Rao also made available the Internet-connected machine used for wide-area measurements, and his network measurement methods have been integrated into the current version of IQ-ECho. Neil Bright and the Utah Emulab team spent many hours to set up Georgia Tech's NetLab facility.

6. REFERENCES

[1] M. Aeschlimann, P. Dinda, L. Kallivokas, J. Lopez, B. Lowekamp, and D. O'Hallaron. Preliminary Report on the Design of a Framework for Distributed Visualization. In *Proceedings of International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, pages 1833–1839, Las Vegas, NV, June 1999.

[2] AG. Access Grid. http://www-fp.mcs.anl.gov/fl/accessgrid.

[3] S. Agarwala, C. Poellabauer, J. Kong, K. Schwan, and M. Wolf. Resource-Aware Stream Management with the Customizable dproc Distributed Monitoring Mechanisms. In *"Proceedings of High Performance Distributed Computing"*, 2003.

[4] H. Balakrishnan, H. Rahul, and S. Seshan. An Integrated Congestion Management Architecture for Internet Hosts. In *Proceedings of ACM SIGCOMM*, 1999.

[5] F. Bustamante, G. Eisenhauer, P. Widener, K. Schwan, and C. Pu. Active Streams: An Approach to Adaptive Distributed Systems. In *Proceedings of the 8th Workshop in Operating Systems (HotOS-VIII)*, Elmau/Oberbayern, Germany, May 2001.

[6] Z. Cai, G. Eisenhauer, Q. He, V. Kumar, K. Schwan, and M. Wolf. IQ-Services: Network-Aware Middleware for Interactive Large-Data Applications. Technical report, Georgia Tech, Feb. 2004.

[7] P. Chandra, A. Fisher, C. Kosak, and P. Steenkiste. Network Support for Application-Oriented Quality of Service. In *Proceedings of IEEE/IFIP International Workshop on Quality of Service*, May 1998.

[8] L. Chen, K. Reddy, , and G. Agrawal. GATES: A Grid-Based Middleware for Processing Distributed Data Streams. In *Proceedings of High Performance Distributed Computing*, 2004.

[9] DOE-TSI. TeraScale Supernova Initiative. http://www.phy.ornl.gov/tsi.

[10] A. Fox, S. D. Gribble, Y. Chawathe, E. A. Brewer, and P. Gauthier. Cluster-based scalable network services. In *Proceedings of SOSP*, 1997.

[11] Globus. GridFTP. http://www-fp.globus.org/datagrid/gridftp.html.

[12] F. B. Greg Eisenhauer and K. Schwan. Event Services for High Performance Computing. In *Proceedings of High Performance Distributed Computing*, 2000.

[13] GriPhyN. The Grid Physics Network. http://www.griphyn.org.

[14] Q. He and K. Schwan. IQ-RUDP: Coordinating Application Adaptation with Network Transport. In *Proceedings of High Performance Distributed Computing*, July 2002.

[15] C. Isert and K. Schwan. ACDS: Adapting Computational Data Streams for High Performance. In *Proceedings of IPDPS*, May 2000.

[16] M. Jain and C. Dovrolis. End-to-end Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput. In *Proceedings of ACM SIGCOMM*, Aug. 2002.

[17] L. Liu, C. Pu, K. Schwan, and J. Walpole. InfoFilter: Supporting Quality of Service for Fresh Information Delivery. *New Generation Computing Journal*, 18(4), 2000.

[18] LSC. http://www.ligo.org/. LIGO Scientific Collaboration, 2003.

[19] M. Mathis. Web100 and the End-to-End Problem. http://www.web100.org/docs/jtech/.

[20] M.Jain and C. Dovrolis. End-to-End Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput. *IEEE/ACM Transactions in Networking*, Aug., 2003.

[21] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Throughput: A Simple Model and its Empirical Validation. In *Proceedings of ACM SIGCOMM*, 1998.

[22] B. Plale, G. Eisenhauer, K. Schwan, J. Heiner, V. Martin, and J. Vetter. From Interactive Applications to Distributed Laboratories. *IEEE Concurrency*, 6(3), 1998.

[23] N. S. V. Rao, S. Radhakrishnan, and B. Y. Cheol. NetLets: Measurement-based Routing for End-to-End Performance over the Internet. In *Proceedings of International Conference on Networking*, 2001.

[24] D. Rosu and K. Schwan. FARACost: An Adaptation Cost Model Aware of Pending Constraints. In *Proceedings of IEEE RTSS*, Dec. 1999.

[25] L. Sha, X. Liu, and T. Abdelzaher. Queuing Model Based Network Server Performance Control. In *Proceedings of Real-Time Systems Symposium*, Dec. 2002.

[26] P. Tinnakornsrisuphap, W. Feng, and I. Philp. On the Burstiness of the TCP Congestion-Control Mechanism in a Distributed Computing System. In *Proceedings of International Conference on Distributed Computing Systems (ICDCS)*, 2000.

[27] Y. Wiseman and K. Schwan. Efficient End to End Data Exchange Using Configurable Compression. In *Proceedings of ICDCS*, Mar. 2004.

[28] M. Wolf, Z. Cai, W. Huang, and K. Schwan. Smart Pointers: Personalized Scientific Data Portals in Your Hand. In *Proceedings of IEEE/ACM Supercomputing Conference*, Nov. 2002.

[29] J. A. Zinky, D. E. Bakken, and R. E. Schantz. Architectural Support for Quality of Service for CORBA Objects. *Theory and Practice of Object Systems*, 3(1), 1997.