| BULETINUL Universităţii Petrol – Gaze din Ploieşti | Vol. LXI No. 2/2009 | 27 - 32 | Seria Matematică - Informatică - Fizică |
|---|---|---|---|

# A Solution for Improving Communication in Desktop Grid Systems

Zoran Constantinescu[*], Monica Vlădoiu[**]

[*] ZealSoft Ltd., str. Tg. Neamţ, nr. 11, Bucharest
e-mail: zoran@unde.ro

[**] Petroleum –Gas University of Ploieşti, Informatics Department, Bd. Bucureşti, 39, Ploieşti
e-mail: mmvladoiu@acm.org

## Abstract

*In this paper we present our solution for improving the quality of communication in our particular desktop grid system, QADPZ, which consists of a dual communication mechanism. Thus, in order to improve the efficiency of communication, QADPZ support two different protocols (UDP and TCP/IP). To deal with the unreliability of UDP, an additional, more reliable level of communication has been added to the system, which works similarly with the real life postal service, delivering and receiving high-level message and implements a reliable, confirmation-based message exchange protocol. We point out that an unreliable transport mechanism that deals with packet loss gracefully and does not exhibit extreme losses can compete well with other more reliable transport protocols.*

**Key words:** *distributed computing, desktop grid computing, communication, TCP/IP, UDP*

## Introduction

Grid and desktop grid computing systems are largely recognized as being very successful in cost effective computation of fully partitionable computations [1, 8, 9, 12, 16, 29]. However, desktop grids cannot be applied to general parallel computations as long as communication is restricted to the master-slave paradigm of parallelism and communication and current parallel computational infrastructures, which for the most part rely on synchronous algorithms, executing in a fully reliable resource environment. *The requirements for desktop grids which can effectively execute iterative parallel computations requiring communication are anonymous, scalable and fault-tolerant communication among the hosts of a scalable desktop grid systems and fault-tolerant computational algorithms, which are insensitive to heterogeneity in processing power of hosts and communication speeds among hosts* [6].

In this paper we present our solution for improving the quality of communication in our particular desktop grid system, QADPZ, which consists of a dual communication mechanism. Thus, in order to improve the efficiency of communication, QADPZ support two different protocols (UDP and TCP/IP). To deal with the unreliability of UDP, an additional, more reliable level of communication, based on message confirmation, has been added to the system. This reliable communication layer works similarly with the real life postal service, delivering and receiving high-level messages.

QADPZ is an open source system for heterogeneous desktop grid computing [21]. QADPZ allows users from an organization-wide LAN or from the Internet to share their computing resources. QADPZ's users can submit to the system compute-intensive applications that are then automatically planned for execution. The scheduling is performed based on application's requirements, both hardware and software. Users are allowed to monitor and control the execution of their applications. Each application consists of one or more tasks. Applications can be independent, when the composing tasks do not need any interaction, or parallel, when the tasks communicate with each other during the computation [3]. QADPZ implements a conceptual model that is based on the master-worker paradigm and that has been improved in several directions: pull vs. push work-units, pipelining of work-units, more work-units sent at a time, adaptive number of workers, adaptive time-out interval for work-units, and multithreading [16].

QADPZ provides for low-level optimizations, such as on-the-fly compression and encryption for communication. QADPZ's user can select from various algorithms, depending on the application, which improves both the communication overhead and the privacy of data. The system goes further and provides an experimental, adaptive compression algorithm, which can transparently pick different algorithms to improve the application [3].

## Communication of QADPZ

The various components of the QADPZ system (master, slaves, clients) communicate with one-another using IP connections over a LAN or WAN, depending on the deployment of the components. Single-stream TCP performance on the WAN is often disappointing [4, 20]. Even with aggressive tuning of the TCP window size, buffer sizes, and chunking of transfers, typical performance is still a fraction of the available bandwidth on the WAN on OC-12 or faster links [5, 11, 15]. While there are a number of factors involved, the behavior of the TCP congestion avoidance algorithm has been implicated as a leading cause of this performance deficit [5, 7, 20].

TCP congestion avoidance algorithms assume that any packet loss is due to congestion, which we define to be over subscription of bandwidth on any switch or link along the path the packet stream takes on the WAN. However, it is increasingly the case that packet loss is caused by events that are unrelated to congestion. The sensitivity of TCP to loss is further exacerbated as the bandwidth of the network is increased, so solutions to remedy poor TCP performance will be more and more important to distributed computing applications on the WAN [10]. Simulations of the TCP protocol performed by Floyd show a high sensitivity to loss, but also demonstrate the fact that - from a control theory standpoint - the TCP congestion avoidance algorithm results in periodic fluctuations that resonate with the deterministic control mechanisms of switching fabric in a highly non-linear and unstable fashion [5]. The default "taildropping" behavior of packet switch input-queues leads to significant degradation in performance. The conclusion is that TCP congestion control, while adequate for the 10-megabit networks it was originally designed for, fails completely on today's multiple gigabit networks and multiple efforts are underway to work around these problems [2, 7, 10].

Loss-tolerant UDP-based protocols will play an increasingly important role in high throughput network applications of the near future [7, 13, 14]. With custom tools, it is possible to find out that UDP packet loss rates are consistently low until you reach a critical limit, which is the available bandwidth of the slowest/most congested link in the network path. At the critical point, when the slowest link in the path across the WAN becomes congested, the loss rate climbs almost linearly in proportion to the increase in output rate. It is interesting to note that frame loss rates, even at low data rates, exhibit some background loss. This is counterintuitive,

as switches should be less congested when exposed to the slower stream, based on the models of network loss assumed by TCP congestion avoidance algorithm.

In QADPZ, messages exchanged between entities in the system are in XML format, in accordance with a strictly defined communication protocol between client and master, and between master and slave (the QADPZ protocol). For our current implementation, the low level communication is based on both TCP (Transmission Control Protocol) and UDP (User Datagram Protocol). We have chosen to implement both protocols to be able to make performance measurements and comparisons. Nevertheless, for the above reasons, UDP is our first option for the low-level communication protocol.

As shown previously, the UDP is an unreliable communication protocol, in which packets are not guaranteed to arrive and if they do, they may arrive out of order [14]. We have outcome this by adding a new layer that ensures reliable communication with UDP. Our higher-level communication abstraction implements a reliable, confirmation-based message exchange protocol. Messages are represented in an XML format for easier extensibility and interconnection with other potential systems.

The advantage of UDP over TCP/IP is that UDP is fast, reducing the connection setup and tear-down overhead, and is connectionless, making the scalability of the system much easier. The higher-level protocol is message-based, and the messages exchanged between the components of the system are of a small size. Also, messages are exchanged only for control purposes.

Because of the unreliable nature of the UDP protocol, an additional, more reliable level of communication is needed (see Figure 1). This is based on message confirmation. Each message contains a sequence number, and each time it is sent, it is followed by an acknowledgment from the receiver. Each sent or received message is accounted, together with the corresponding acknowledgment, and in case of not receiving an acknowledgment, the message is resent a few more times. An acknowledgment and a normal message can be combined into one message to reduce the network traffic.

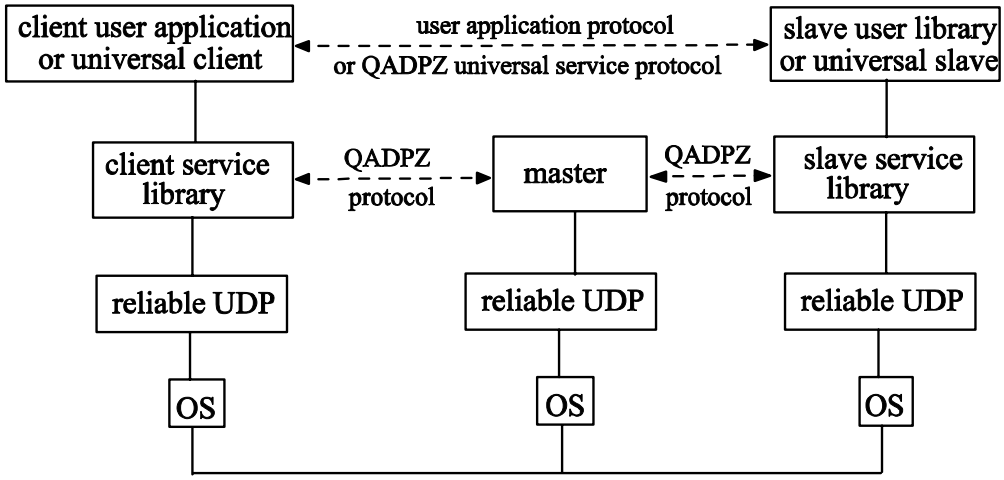


**Fig. 1.** QADPZ communication layers

The abstraction used in our reliable communication layer (described in Figure 2) is similar in functionality with the real life postal service. It delivers and receives high-level messages. We use two types of high-level messages: an XML format to control messages between entities in the system, and a plain binary format, used for any kind of data transfer (for example, when slaves communicate between each other).

The PostOffice module supports both blocking and non-blocking delivery of messages. Messages have a source and a destination address. Received messages can be kept by the

PostOffice as long as needed, the upper layers in the system having the possibility to retrieve only certain messages, based on the sender's address. This layer provides also support for encryption of messages, providing a certain amount of security to the system. Another feature provided is the possibility of using compression, this way reducing the size of the messages by using some more CPU power.

The PostOffice module is using the capabilities of the UDPConfirm layer for sending and receiving messages. This layer is responsible for resending any previously sent and unconfirmed messages. A separate thread is checking periodically these messages. It provides both blocking and non-blocking message sending. The lowest level module is called UDPSocket, and is responsible for hiding operating system specific function calls, making a more general interface for UDP socket communication.

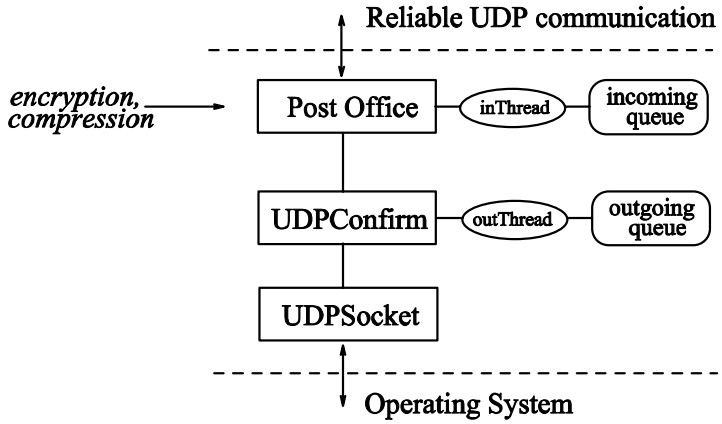


**Fig. 2.** Reliable UDP communication

## Parallel Computing in QADPZ

The Message Passing Interface implemented is based on the previously described reliable UDP communication mechanism, and is shown in Figure 3. It is using the same message based protocol as the one used for the communication between the entities in the system. The PostOffice abstraction provides a way to send/receive messages in a blocking or non-blocking mode. This provides an easy way to implement the different types of MPI_send() and MPI_recv() functions from the MPI standard, and the MPI_wait().

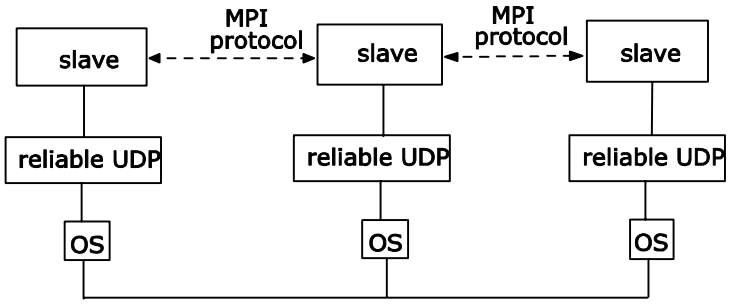


**Fig. 3.** MPI communication

The initialization of the MPI communication between slaves is done with the help of the master node: when executing the MPI_Init() routine, each slave sends a message to the master, specifying the address and port of its UDP socket used for sending and receiving messages. The master waits for this message from all of the slaves, then gives each slave a rank and distributes a list of all MPI nodes to each of the slaves.

Our current implementation does not support collective communication, only the MPI_COMM_WORLD communicator. However, a complete library of the collective communication routines can be written entirely using the point-to-point communication functions and a few auxiliary functions. The implementation provided is limited to a small subset of the MPI. It contains only the most used functions, and is intended only for testing purposes and evaluation of the parallel communication. More complete implementation based on existing libraries is possible, but it has been outside the scope of this work.

## Conclusion

For data transfer and replication, file integrity is paramount. Response time and performance is of comparable importance to file integrity for computationally intensive applications. For instance, visualization tools almost invariably use reliable transport protocols to connect distributed components, since there is a general concern that lifting the guarantee of data integrity would compromise the effectiveness of the data analysis. However, visualization researchers find acceptable other forms of lossy data compression like JPEG, wavelet compression and even data resampling. Acceptance may be due to the fact that degradation in visual quality is well behaved in these cases. Therefore, an unreliable transport mechanism that deals with packet loss gracefully and does not exhibit extreme visual artifacts will compete well with other well-accepted data reduction techniques. Furthermore, when tuned to fit within the available bandwidth of a dedicated network connection, the loss rates for unreliable transport are extremely small - a few tenths of a percent of all packets sent if the packets are paced to stay within the limits of the slowest link in the network path. As shown previously, the UDP is an unreliable communication protocol, in which packets are not guaranteed to arrive and if they do, they may arrive out of order. We have outcomed this by adding a new layer that guarantees reliable communication with UDP. Our higher-level communication abstraction implements a reliable, confirmation-based message exchange protocol.

## References

1. Berman, F., Fox, G., Hey, A.J.G. - *Grid computing: making the global infrastructure a reality*, New York, J. Wiley, 2003
2. Clark D.D. et al. - Making the world (of communications) a different place, *Computer Communication Review* (SIGCOMM) 35(3): 91-96, 2005
3. Constantinescu, Z. - *A Desktop Grid Computing Approach for Scientific Computing and Visualization*, PhD Thesis, Norwegian Univ. of Science and Technology, Trondheim, Norway, 2008
4. Edwards, J., Bramante, R. - *Networking Self-Teaching Guide: OSI, TCP/IP, LAN's, MAN's, WAN's, Implementation, Management, and Maintenance*, Wiley, 2009
5. Floyd, S. - A Report on Some Recent Developments in TCP Congestion Control, *IEEE Communications Magazine*, April 2001
6. Foster, I., Kesselman, C. - *The grid: blueprint for a new computing infrastructure*, Boston, Morgan Kaufmann Publishers, 2004
7. Goralski, W. - *The Illustrated Network: How TCP/IP Works in a Modern Network*, 2008
8. Juhasz, Z., Kacsuk, P., Kranzlmuller, D. - *Distributed and Parallel Systems: Cluster and Grid Computing*, New York, Springer, 2004
9. Kacsuk, P., Fahringer, T., Nemeth, Z. - *Distributed and Parallel Systems: From Cluster to Grid Computing,* Springer, 2007
10. Kozierok, C. - *The TCP/IP Guide: A Comprehensive, Illustrated Internet Protocols Reference*, No Starch Press, 2005
11. Loshin, P. - *TCP/IP Clearly Explained*, Morgan Kaufmann, 2003
12. Magoules, F., Pan, J., Tan, K.A, Kumar, A. - *Introduction to Grid Computing*, CRC 2009

13. Medina, A., Allman, M., Floyd, S. - *Measuring the Evolution of Transport Protocols in the Internet*, ACM CCR, April 2005
14. Peterson, L.L., Davie, B.S. - *Computer networks: a systems approach*, Morgan Kaufmann, 2007
15. Scaglia, S. - *The Embedded Internet: TCP/IP Basics, Implementation and Applications*, Addison-Wesley Professional, 2007
16. Silva, V. - *Grid Computing for Developers,* Charles River Media, 2005
17. Tanenbaum, A.S. - *Computer Networks,* Prentice Hall, 2003
18. Vladoiu, M., Constantinescu, Z. - An Extended Master-Worker Model for a Desktop Grid Computing Platform (QADPZ), *3rd Int'l Conference on Software and Data Technologies ICSOFT 2008*, pp. 169-174, 2008
19. Wang, L., Jie, W., Chen, J. - *Grid Computing: Infrastructure, Service, and Applications,* CRC 2009
20. Welzl, M. - *Network Congestion Control: Managing Internet Traffic*, Wiley, 2005
21. *** - *QADPZ,* http://qadpz.sourceforge.net, accessed 2009

## O soluție de îmbunătățire a comunicațiilor în sistemele desktop grid

## Rezumat

*În această lucrare este prezentată soluția noastră de îmbunătățire a calității comunicațiilor în sistemul desktop grid QADPZ, care constă dintr-un mecanism de comunicație dual. Astlfel, pentru a creşte eficiența comunicațiilor, QADPZ oferă support pentru două protocoale diferite (UDP şi TCP/IP). Pentru a depăşi faptul că protocolul UDP are pierderi, am adăugat un nivel adițional de comunicații, mai sigur, care lucrează similar cu serviciul poştal din lumea reală, livrînd şi primind mesaje de nivel înalt şi implementează un protocol de schimb de mesaje bazat pe confirmarea mesajelor. Am subliniat aici că un mecanism de transport nesigur care gestionează elegant pierderea de pachete şi care nu prezintă pierderi extreme, poate concura bine cu alte protocoale mai sigure de transport.*