Grid-Specific Network Enhancements: A Research Gap?

Michael Welzl, Muhammad Murtaza Yousaf

Institute of Computer Science, University of Innsbruck, Austria
{Michael.Welzl, Murtaza.Yousaf}@uibk.ac.at

Abstract

The Internet communication infrastructure (the TCP/IP protocol stack) is designed for broad use; as such, it does not take the specific properties and requirements of Grid applications into account. This one-size-fits-all approach is far from being optimal – general network mechanisms, while useful for the Grid, cannot be as efficient as customized solutions. In this paper, we list properties that make a Grid somewhat unusual from a network perspective, and explain network enhancement ideas that are based upon them. In doing so, we hope to create some awareness for what we consider to be a research gap.

1. Introduction

Grid computing has been defined as "coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations" – in practice, this usually means that a computation intensive application is distributed across a number of computers for the sake of speeding it up. This step from local clusters to massively distributed applications that use the Internet as its underlying means of communication entails a requirement for efficient network usage, but the Internet is a general-purpose network that was not built to provide maximum efficiency to the Grid. While there are special protocols for special applications in the protocol suite (for example, the "Simple Mail Transfer Protocol (SMTP)", the "HyperText Transfer Protocol (HTTP)" the "File Transfer Protocol" (FTP), and a large number of protocols for real-time multimedia transmission), there is no such support for the Grid, where efficiency is much needed.

Yet, a Grid has properties that make it quite unique from a networking perspective:

1. A set of nodes contributes to a common cause (i.e. it is reasonable to require them to carry out work for each other if this leads to a benefit for the whole system), and they can be expected to remain available for a long time.
2. Traffic typically consists of short, sporadic method invocations (Grid Service calls) and bulk data transfers, which may be huge. The distinction between these two types of data transfer is usually clear to the programmer of a Grid application, who explicitly invokes a file transfer (e.g. by calling the "GridFTP" service) when she or he sends a large amount of data.
3. File transfers are "pushed" and not "pulled": in a standard FTP session, it is more common for users to initiate file downloads rather than uploads. In a Grid scenario, a scheduler decides where application parts go; then, it transfers the required data to all the places where calculations are carried out. This means that there may be more than one receiver for a large file, which renders reliable multicast especially useful in a Grid setting.

4. In order to enable efficient scheduling, performance predictions are needed (e.g. "how long will it take to transfer this file?").
5. A Grid application may be able to specify its communication processes in advance – e.g. a workflow based application may be able to state "a 1-Gigabyte-file will be transferred from node A to node B; then, node B will calculate something – this lasts at least 30 seconds – and then, the results will be transferred back." In general, Grid applications have a larger probability of showing some workflow aspects than more traditional applications. Thus it seems to be required to have a networking-related analysis being performed automatically, which utilizes workflow information and other heuristics.
6. A node may know (in advance) when another node will send something.

Due to their heterogeneous nature, certain Grids may only satisfy some but not all of these properties; nevertheless, this list captures some essential network specifics that are typical for many of them. Interestingly, there are few other applications that share some of these properties – an example may be P2P file sharing tools, but here, the communication model is usually "pull" as opposed to "push", and nodes come and go at a much shorter time scale.

Some of the properties above are quite unusual indeed – for instance, the ability to specify a communication flow in advance is rare: a web browser can never know how long a communication break is going to be, as this depends on the user. FTP downloads simply start and end. The same is true for real-time multimedia applications, which typically play until a user decides to quit (e.g. stop watching the video or hang up the phone).

In what follows, we will present some ideas which make use of the properties listed above for enhancing the communication capabilities of Grid applications. The suggested network improvements are Grid specific, i.e. they would only be applicable for Grid applications and similar ones that share the properties above – this concerns certain Web Service based applications, P2P applications and applications that are built on top of distributed object systems such as CORBA, for instance. We will conclude with an overview of related work.

2. Network measurement

Due to its distributed nature, a Grid application can adversely influence the network performance of itself if some hosts send too much. Similarly, Grid application 1 can disturb Grid application 2 if they share the same hosts. This problem is shown in fig. 1, where the two hosts A and B are connected to host C via a single link (the broken line in the figure). If both A and B send at a high rate, they can reduce the throughput of each other, thereby degrading the performance of the application. If this fact was known, a Grid scheduler could try to circumvent this problem by relocating parts appropriately – but in practice, these instances remain uninformed as there is no means available to detect such a problem.

This gap could be filled by a new measurement tool as follows:

- It monitors the traffic that each Grid application running on a host generates. Should it detect a certain special traffic pattern (e.g., a short burst of high-rate UDP traffic), it informs all measurement instances at hosts that receive traffic which may be influenced by the pattern. These instances then try to identify the pattern in their received traffic, estimate any potential negative impact on their received data stream

and feed back the information to the original measurement instance, which in turn informs the scheduler at the sender.

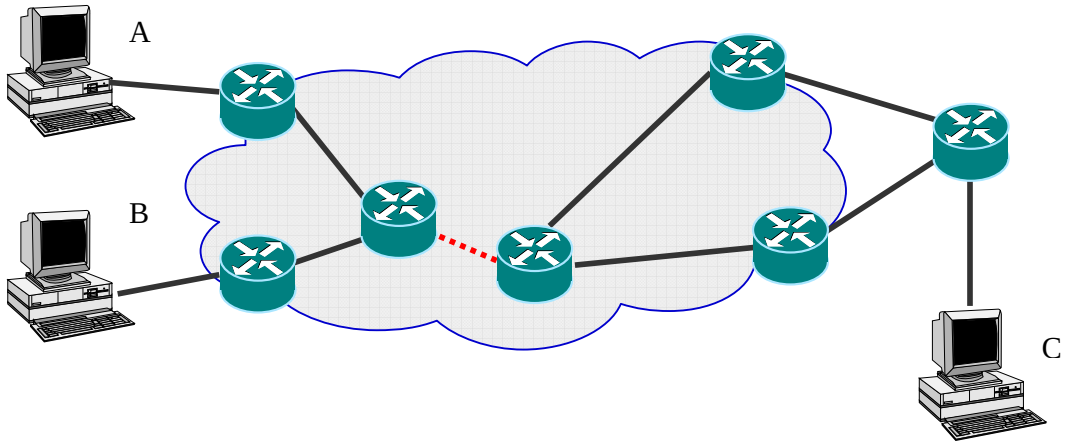

Figure 1: A Grid application hindering itself

- If no such "special" traffic pattern is generated by an application (e.g., the application only maintains long-lasting TCP connections), a measurement instance generates a traffic pattern that will leave a "signature" in the traffic aggregate. To limit overhead, this could be done using regular payload, e.g. by influencing TCP congestion control in a way that is transparent to the application.

Our envisioned measurement tool operates at the network layer. It is, in general, concerned with all the traffic that a host generates. Its goal is to identify or create special patterns within the traffic in order to notice negative effects that were caused by a Grid application. The only interaction with other layers is feedback to the Grid scheduler after the measurement. The tool builds upon existing network measurement techniques: the concept of identifying a traffic signature that may be generated by the tool itself is borrowed from a similar idea for backtracing "Denial-of-Service (DoS)" attacks, where designated hosts generate traffic streams – if the impact of their traffic is seen in the received traffic pattern, the hosts share a link with the attacker [2] [3].

Detection of shared bottlenecks has additional benefits in a Grid setting, where schedulers need to know the time it takes to transfer a certain amount of data between sites in advance in order to choose the ideal one. Theoretically, this can be attained by means of a prediction system such as the "Network Weather Service (NWS)" [8], but any such prediction will fail miserably if it is independently used by computers A and B for the connection A-C and B-C, if they then decide to transmit their files at the same time. Predictions in a Grid must therefore make use of knowledge about shared bottlenecks.

Obtaining reliable predictions is also a difficult function that can make use of Grid peculiarities. Since the goal is often to calculate the time that it will take to transfer a file (which can be exceedingly large, especially in a Grid), short-term predictions may not be very valuable. Internet traffic fluctuations can occur at very short time scales; using these data to

predict bulk data transfers is a mismatch in terms of time scales. Path changes, however, are rather rare occasions in the Internet, and so it may be much more reasonable to determine the bottleneck capacity and use this value as a basis. This can be done with the "packet-pair" measurement method – a mechanism to estimate the bottleneck capacity, which may not necessarily be the same as the bottleneck link bandwidth due to special queuing, but it can be interpreted as an upper limit for the bandwidth along a path [6] [7].

Packet pair requires a long series of measurements in order to yield a reliable result, rendering it useless for mechanisms that operate in the order of round-trip times (RTT), such as congestion control. Similarly, passive measurements (which have the advantage of not placing any burden onto the network) can only be made when there is traffic that can be monitored, i.e. such tools may need to be active for a long time before they yield a useful result. Despite the potential benefits, both packet pair and passive measurements are therefore somewhat unattractive for network operators, who sometimes need data fast.

This is different in a Grid, where nodes are available for a long time, long-term predictions are needed and it can be expected that every node will obtain some traffic every once in a while (in particular, when the "low-priority-pushing" scheme which we will explain in the next section is in use). In this scenario, packet pair is attractive because we can deduce facts with it that do not change too often, and passive measurements are attractive because they can constantly monitor what is going on in a non-intrusive way. Luckily, these two measurement methods can even be combined: it has been shown that, due to the self-clocking effect described in [9] and the delayed ACK mechanism [10], implemented in many TCP stacks today, every TCP sender sends about 50% of all packets as packet pairs [11]. In general, TCP probes for the available bandwidth in the network, and therefore, many things about a path can be learned from monitoring its behavior.

3. Connection handling and congestion control

The fact that SOAP and the program on top do not have the notion of a "session" or a "connection" requires TCP connections to be set up and torn down for each call to a Grid Service. This problem was described in [12]; we conducted measurements that have shown that a simple "Hello World" Grid Service call with Globus Toolkit version 3 causes some 60 packets to go back and forth and requires at least 6 round-trip times. On the other hand, we found that MPI implementations, where the notion of a connection is implicitly available because the number of communicating processes is known from the beginning, typically reuse existing connections.

Even if SOAP calls would maintain a single connection, Grid communication would be a poor match for the standard window-based flow and congestion control algorithms in TCP: a pair of Grid sites may only use the network sporadically – but interrupting a TCP connection causes the congestion window to decay [13]. TCP uses a standard congestion control mechanism for each and every type of application – this is not a good choice for the Grid.

Since Grid traffic is sometimes generated by applications “on their own” (i.e. not directly invoked by humans), its pattern may be predictable (property 5 in our list). This includes "taking a break". One might be tempted to ask the question: Is it fair that my Grid application, which runs for 5 minutes and only uses the network sporadically, achieves much less throughput than an ftp download which utilizes the network for the whole 5 minutes?

It would, for instance, be possible to “reward” an application for phases of transience with “aggresssion points” that would be taken from it during phases of data transmission. Methods to be more aggressive than TCP without necessarily endangering the stability of the Internet are described in [14] and [15].

Being more aggressive than TCP is not the only option; actually, *low-priority* data transfer can also have some merit in a Grid. Since TCP (almost, if ECN is used) increases its rate until a packet is dropped, a mechanism that reduces its rate in response to increasing delay (as caused by a growing bottleneck queue) will generally be less aggressive than TCP and only use the excess capacity while largely remaining non-intrusive. Such schemes are described in [16] and [17]; in a Grid, they could be used for a "low-priority push" action – with such a service, a scheduler could send files to sites where it is anticipated that they will be needed in the future. Replication (having several Grid nodes attempt the same calculation at the same time) could also be supported with such a service; since it would be non-intrusive, its impact onto the network would be negligible.

Finally, Grid file transfers can have deadlines and associated priorities. Once again, consider the scenario shown in figure 1; this time, let us assume that A wants to transfer a large file to C, and A knows that B will also start a file transfer to C in 5 minutes. Then, it would be very beneficial to get the file across within the 5 minutes – being allocated an amount of bandwidth that enables the application to transfer the file in time rather than, say, 10 minutes would be very important for the application. Being allocated enough bandwidth to transfer the file within 10 rather than, say, 20 minutes might be much less useful.

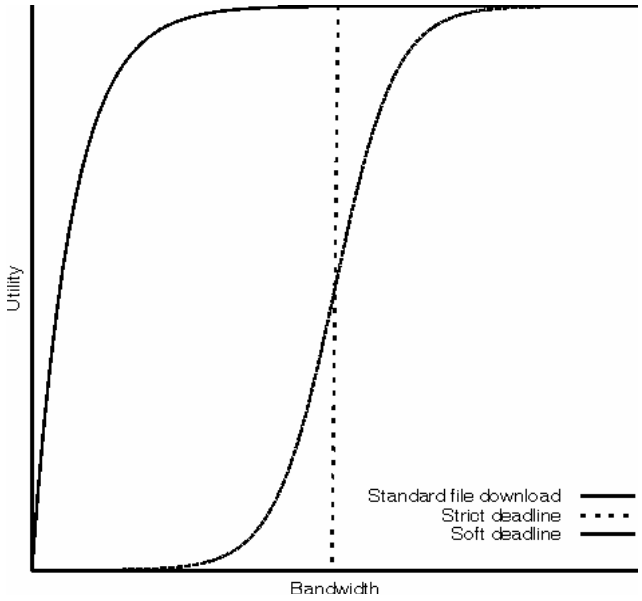

Figure 2: Utility functions for Grid file transfers

The value that an amount of bandwidth has to an application can be specified by means of a utility function [18]. As shown in figure 2, standard file downloads are known to have a strictly concave utility function; a system that would optimize the rates such that all these strictly concave (logarithmic) utility functions are maximized would converge to proportional fairness [19]. Some applications have unusual utility functions. Take VoIP, for example: it

only requires a very small amount of bandwidth; if it gets less, the program becomes useless, and getting much more usually does not change much. The utility function of such an application would be similar to a file transfer in a Grid with strict deadlines (e.g., when the destination node is known to be available until 7 pm, and the source data must therefore be transmitted until 6:30 pm for processing). The bandwidth requirements of A in our example from figure 1 could perhaps be represented by the "soft deadline" utility function in figure 2. Then, the goal would be to allow Grid applications to specify their respective utility functions and maximize them all – something similar has been described with prioritization inside the network in [20], but it could also be possible to reach the same goal in a distributed fashion.

4. Conclusion

To use a Grid at its full potential, the underlying network must also be a managed resource, just like computing and storage. As such, it should be managed in an intelligent way by an autonomic Grid middleware. In this paper, we claimed that not much has been done in this aspect; we presented ideas that are based on specific Grid properties and requirements, while most network enhancements that have so far been used in and described for a Grid context are actually much broader in scope. Consider the following examples:

The Transmission Control Protocol (TCP) was found to perform poorly in high-bandwidth, high-delay networks [21]. For Grids, where changing the network stack in the operating system may not be feasible and quick solutions are often desired, many application level communication protocols for better performance have been proposed – e.g. RUDP [22], SABUL [23], GridFTP [24], FOBS [25], PSockets [26], and UDT [27]. These protocols are based on approaches such as: improvements to TCP, new algorithms at the application layer, and employing parallel TCP connections. Although high speed links are common in Grids, special Grid scenarios were not considered while designing these protocols; this concerns, for example, the nature and requirements of end points, the type of traffic, and advance knowledge of the communication process (in case of workflow based applications in a Grid).

Quality of Service (QoS) and security issues are challenging for Grids. In order to negotiate and obtain QoS on data transfers, the authors of [28] designed a data transfer service based on a WS-Agreement model [29] (a set of protocols under development by Global Grid Forum for the negotiation, creation, and management of services in a grid). This data transfer service defines agreements which guarantee that a file transfer can be completed under a specified time within a certain confidence level. It relies on the prediction of available bandwidth. This prediction is based on the history of data transfers, and due to the highly dynamic nature of a Grid, it cannot be very reliable.

A network measurements and monitoring system for the Grid is described in [30]. This system is derived from Simple Network Management Protocol (SNMP) and it collects and stores the network performance information to provide services to consumer; other than our proposed measurement approach, it does not make use of the special properties of a Grid.

By pointing out these examples, we do not intend to downplay the quality of these developments; rather, our goal is to make the community aware of a research gap. By describing our ideas for filling it, we hope that we can motivate colleagues to follow us in our exploration of Grid-specific network mechanisms.

References

[1] I. Foster. The Grid: Blueprint for a New Computing Infrastructure. Morgan-Kaufman, 2004 (2nd edition).

[2] Stefan Savage, David Wetherall, Anna Karlin and Tom Anderson. Network Support for IP Traceback. ACM/IEEE Transactions on Networking, 9(3), June 2001.

[3] Hal Burch and Bill Cheswick. Tracing Anonymous Packets to Their Approximate Source. USENIX LISA 2000, December 3-8, 2000, New Orleans, Louisiana, USA.

[4] Bill Cheswick, Hal Burch, and Steve Branigan. Mapping and Visualizing the Internet. USENIX 2000 Technical Conference, June 18-23, 2000, San Diego, California, USA.

[5] B. Huffaker, D. Plummer, D. Moore, and K. Claffy. Topology discovery by active probing. Symposium on Applications and the Internet (SAINT) 2002.

[6] Jean-Chrysostome Bolot. End-to-End Packet Delay and Loss Behavior in the Internet. ACM SIGCOMM 1993, pp. 289-298. also in Computer Communication Review 23 (4), Oct. 1992.

[7] Vern Paxson. End-to-End Internet Packet Dynamics. IEEE/ACM Transactions on Networking 7(4), pp. 1-16.

[8] R. Wolski, L. Miller, G. Obertelli, M. Swany. Performance Information Services for Computational Grids. In Resource Management for Grid Computing, Nabrzyski, J., Schopf, J., and Weglarz, J., editors, Kluwer Publishers, Fall, 2003.

[9] V. Jacobson, M. Karels. Congestion Avoidance and Control. In Proc. of ACM SIGCOMM, Stanford, August 1988.

[10] R. Braden. Requirements for Internet Hosts – Communication Layers. Oct 1989. IETF RFC 1122.

[11] H. Jiang, C. Dovrolis. The effect of flow capacities on the burstiness of aggregate traffic. In Proc. of PAM, France 2004.

[12] Volker Sander (ed.), William Allcock, Pham CongDuc, Jon Crowcroft, Mark Gaynor, Doan B. Hoang, Inder Monga, Pradeep Padala, Marco Tana, Franco Travostino, Pascal Vicat-Blanc Primet, Michael Welzl. Networking Issues for Grid Infrastructures. Global Grid Forum Document GFD.037 (informational), Grid High Performance Networking Research Group, November 22, 2004.

[13] M. Handley, J. Padhye and S. Floyd. TCP Congestion Window Validation. RFC2861, June 2000.

[14] Philippe Oechslin and Jon Crowcroft. Differentiated End-to-End Internet Services using a Weighted Proportional Fair Sharing TCP. ACM Computer Communication Review (CCR), 1998.

[15] Panos Gevros. Internet Service Differentiation using Transport Options:the case for policy-aware congestion control. RIPQoS Workshop co-located with ACM SIGCOMM 2003, August 2003, to be published in ACM Computer Communication Review.

[16] A. Kuzmanovic and E. W. Knightly. TCP-LP: A Distributed Algorithm for Low Priority Data Transfer. In Proceedings of IEEE INFOCOM 2003, San Francisco, CA, April 2003.

[17] Arun Venkataramani, Ravi Kokku and Mike Dahlin. TCP Nice: A Mechanism for Background Transfers. In Proceedings of OSDI'02, Boston, MA, December 2002.

[18] Scott Shenker. Fundamental Design Issues for the Future Internet. IEEE Journal on Selected Areas in Communications, 13, pp. 1141-1149, 1995.

[19] Frank Kelly. Charging and rate control for elastic traffic. European Transactions on Telecommunications, 8, pp. 33-37. An updated version is available at http://www.statslab.cam.ac.uk/frank/elastic.html

[20] Narayanan Venkitaraman, Jayanth P. Mysore, and Mike Needham. A Core-Stateless Utility Based Rate Allocation Framework. Proceedings of PFHSN 2002 (Protocols for High Speed Networks 2002 - IFIP TC6 WG6.2 / IEEE Comsoc TC on Gigabit Networking), Springer Verlag, Berlin, Germany, 22-24 April 2002.

[21] W. Feng, P. Tinnakornsrisuphap. The Failure of TCP in High-Performance Computational Grids. In Proceedings of Super Computing 2000.

[22] E. He, J. Leigh, O. Yu, T. DeFanti. Reliable Blast UDP : Predictable High Performance Bulk Data Transfer. In Proceedings of the IEEE Cluster Computing, Chicago, Illinois. September 2002.

[23] R. L. Grossman, Y. Gu. SABUL: A Transport Protocol for Grid Computing. J. Grid Computing. 1(4): 377-386 (2003)

[24] W. Allcock, J. Bester, J. Breshahan, A. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, S. Tuecke. Secure, Efficient Data Transport and Replica Management for High-Performance Data_Intensive Computing. In Proceedings of IEEE Mass Storage Conference 2001.

[25] P. Dickens. FOBS : A Lightweight Communication Protocol for Grid Computing. In Proceedings of Europar. 2003.

[26] S. Bailey, R. L. Grossman, H. Sivakumar. PSockets: The Case for Application-level Network Striping for Data Intensive Applications using High Speed Wide Area Networks. In Proceedings of Supercomputing 2000.

[27] R. L. Grossman, Y. Gu, X. Hong, A. Antony, J. Blom, F. Dijkstra, C. de Laat. Experimental Studies Using Application Layer Protocols Based Upon UDP to Support Applications Requiring Very High Volume Data Flows.

[28] H. Zhang, K. Keahey, W. Allock. Providing Data Transfer with QoS as Agreement-Based Service. In Proceedings of International Conference on Services Computing, Shanghai, China. September 2004.

[29] K. Keahey, T. Araki, P. Lane. Agreement-Based Interactions for Experimental Science. In Proceedings of Europar. August 2004.

[30] W. Junfeng, Z. Mingtian. Providing Network Monitoring Service for Grid Computing. In Proceedings of International Conference on Network and Parallel Computing, Wuhan, China. October 2004.