A Reliable Network Measurement and Prediction Architecture for Grid Scheduling

Muhammad Murtaza Yousaf, Michael Welzl

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

Abstract

For efficient use of geographically distributed resources in a grid, the selection of the optimum site is highly important for an autonomic grid scheduler. We propose a reliable network measurement and prediction architecture that helps its clients with their scheduling decisions by informing them about the minimum time that it will take to transfer certain amount of data. This is achieved by calculating TCP throughput under ideal conditions. Our prediction is based on the "packet pair" measurement method; we introduce a receiver side passive capacity estimation technique which additionally calculates a reliable lower bound of the Round Trip Time. Passive operation is feasible in a grid, where large file transfers between nodes are frequent, and it ensures non-intrusiveness of our architecture; active measurements are only initiated when they are needed by clients.

1. Introduction

Grid Computing promises a super-computing like performance at a low cost, which means that a large number of users should have the opportunity to solve their problems on a widely accessible platform. In order to meet the claimed performance, the grid research community needs to design methodologies which can exploit this computation power. As resources in a grid span over several administrative domains and heterogeneous environments, a comprehensive software infrastructure is needed for Grid Computing [1].

Among others, one major issue in fully utilizing the grid capabilities is assignment of appropriate resources for a job in such a way that minimizes the execution time of a particular set of tasks. For a good allocation, which is crucial for efficient execution and use of grid resources, a scheduler must have comprehensive and up-to-date knowledge of available resources. Choosing them just on the basis of a job requirement is not enough. As these resources can be available on many execution sites, it is important to know the time it takes to transfer a certain amount of data between them – this could help a scheduler to select the optimum site. A system to estimate and predict network path characteristics can help a scheduler a great deal. Ideally, these estimates should neither be old nor based on some weak techniques; rather, they must be very precise and reliable.

There are many reasons to measure network traffic [2], including service monitoring, network planning, cost recovery etc. - but this is problematic because the Internet is highly dynamic in nature, and the measurements are most valuable when the relevant network properties remain steady. However, some path properties are relatively stable

[3]: the capacity of a path tends to be constant until a routing change or link upgrade occurs [4], as compared to available bandwidth which varies with time and shows high variability in a wide range of timescales, hence making it hard to measure.

Most data-transfer applications and about 90% of the Internet traffic use the TCP protocol [5], which makes the throughput of a TCP transfer the most important performance metric of a path. It depends on many factors and path properties, and a precise estimate of these characteristics, particularly path capacity can yield a reliable upper bound on TCP throughput. This corresponds with the minimum time that will be needed to transfer a file. Methods to detect the capacity are difficult in a normal environment but a grid is more supportive for accurate measurements: the nodes in a grid are available most of the time and we can monitor large flows among all of its sites.

The rest of this paper is structured as follows. Section 2 has an architecture description of our proposed system. We describe our measurement technique in Section 3. How to predict the minimum transfer delay is illustrated in Section 4. We summarize related work in Section 5, and our conclusions and future work are given in Section 6.

2. Architecture Description

2.1 Design Considerations

We identified the following requirements for our measurement and prediction system: the dynamic nature of the network requires frequent measurements in order to provide a precise and up-to-date estimation of network characteristics. As measurements are taken quite often, the measurement overhead should be as small as possible. The system must be scalable because it should not limit the size of the grid. Last but not least, it must be available all the time.

2.2 Architecture

Our system will consist of five basic components, a *Monitor*, an *Information Manager*, an *Information Base*, an *Active Prober*, and a *Client*. A high level architecture is shown in Figure 1.

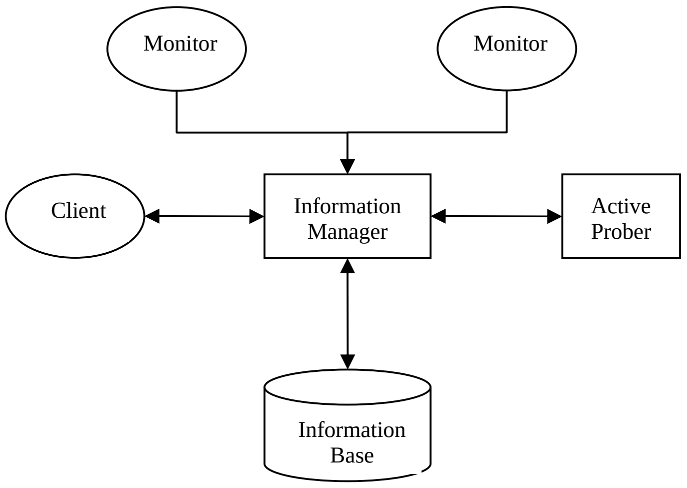

Figure 1: High level architecture of the system

The key component in the whole system is the *Monitor* which has sub components: a *Sniffer*, a *Filter*, and an *Information Extractor*. The *Sniffer* monitors and captures traffic from a network interface by using the packet capturing library "libpcap". The captured packets[1] from the *Sniffer* may belong to many flows; they are all passed to the *Filter* that separates packets which belong to a particular stream. Another task of the *Filter* is to remove cross traffic effects from a particular stream because ultimately we need potential packet pairs. The *Information Extractor* (IE) receives a filtered stream and generates reliable estimates by using the measurement method explained in the next section. This whole scenario is shown in Figure 2.

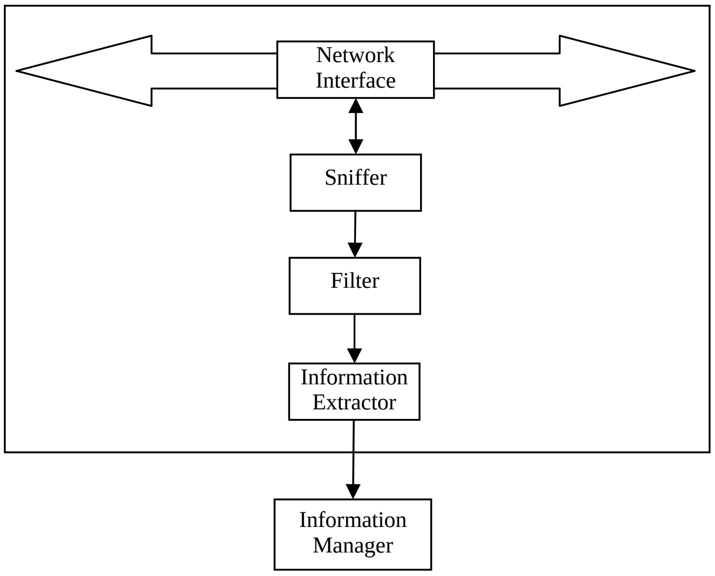

Figure 2: Internal architecture of the Monitor

The *Information Manager* (IM) stores and retrieves information from the *Information Base* (IB). The information in the IB is stored with a timestamp and fresh information is considered more reliable. Another task of the IM is to invoke the *Active Prober* if:

- Information is too old for a particular path to meet the required level of reliability.
- A network path is lightly loaded and traffic generated by the Active Prober will not severely affect the regular traffic.
- Some information for a particular path is missing.
- For a particular path, client behavior is such that it needs more reliable information.

The *Active Prober* injects some traffic for the measurement purpose and generates estimates as required.

The next component of our system is the *Information Base*, placed at a central location, to store measurements. The *Information Base* is quite similar to the *Persistent State* component of Network Weather Service NWS [8]. We are therefore considering to integrate our system with NWS. Interaction with the *Client* will be

[1] When we talk about "captured packets", we refer to packet related information as one would obtain with tcpdump.

provided according to the Grid Monitoring Architecture (GMA) [9] .

3. Measurement Method

3.1 Packet Pair

The main idea behind the *Information Extractor* is to apply the packet pair [10] method on the traces and estimate the path capacity. Packet pair has been studied extensively; among others, it was used by Bolot [10], Carter and Crovella [11], Paxson [12], and Lai and Baker [13] to measure the bottleneck bandwidth. The concept of packet pair is, if two packets are sent back-to-back such that both packets are queued together at the bottleneck link, then the difference in their arrival times is determined by the bottleneck link. Specifically the time difference between two packets is equal to the time that the router at the end of the bottleneck link spent receiving the second packet after the first one was received. As the spacing can only be changed by a slower link, it will remain the same through to the receiver. This concept is presented in Figure 3.

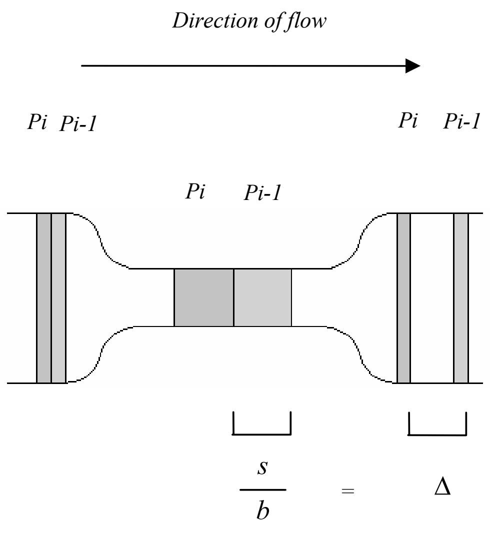

Figure 3: Packet Pair (*Pi-1 and Pi*), before and after bottleneck

If Δ is the arrival time difference of packet pair (*Pi and Pi-1*) at the receiver, *s* is the size of the first packet, and *b* is the bandwidth of the bottleneck, then the path capacity (the bandwidth of the bottleneck) can be calculated by the Equation 1. A formal proof of this equation is given in [14].

$$b = \frac{s}{\Delta} \tag{1}$$

The above explanation assumes an ideal packet pair situation, but cross traffic can cause congestion leading to a high or low estimate. This can happen as follows:

- If, after the bottleneck, some packets are queued before the first one of a packet pair, then the second packet of the pair will come closer to the first one. In this case, the gap caused by the bottleneck link will be reduced which will result in a high estimate of the bottleneck capacity. A higher estimate is however not a severe problem for us, as we are looking for a reliable maximum of the capacity – in other words, this error will cause our system to say something like "it will take at least 3 seconds to transfer this file" instead of saying "it will take at least 5 seconds to transfer this file". While it would be better to answer "5 seconds" if this is possible, our "3 seconds" answer would still be correct.

- If some packets are queued between the two packets, then they will increase the gap of the packet pair, which will result in a low estimate of bottleneck capacity. This is however unlikely to occur all the time in a long lasting stream – therefore, this error will probably always be filtered out when we take the maximum of the packet pair capacity estimates.

3.2 Detection of Packet Pairs

To ensure that both packets are sent close enough to be queued at bottleneck, traffic has been sent actively by many measurement tools [4]. To avoid the overhead caused by active probing our technique is based on passive monitoring at the receiver side.

Based on the self-clocking effect described in [6] and the delayed ACK mechanism [15] (the receiver only acknowledges every second packet, and an ACK for the first packet is only generated if a TCP timer expires), implemented in many TCP stacks today, it was shown that every TCP sender sends about 50% of all packets as packet pairs [7].

We have designed a TCP receiver window (*rwnd*) based technique to detect potential packet pairs. If the receiver implements the delayed ACK mechanism, then for a delayed ACK we can expect a pair of packets (or even a longer series of packets) to be sent back-to-back. Potential packet pairs are caused by a delayed ACK, but from a continuous stream it is not easy to decide about this pair. We have used the *rwnd* size property to identify a pair of packets triggered by a delayed ACK. A sender can send at most the amount of data which is allowed by the *rwnd*. This window size is sent by the receiver every time it ACKs an incoming data packet. So, if we receive data segments which are beyond the window range of an ACK, then these must have been caused by the next ACK. This is shown in Figure 4.

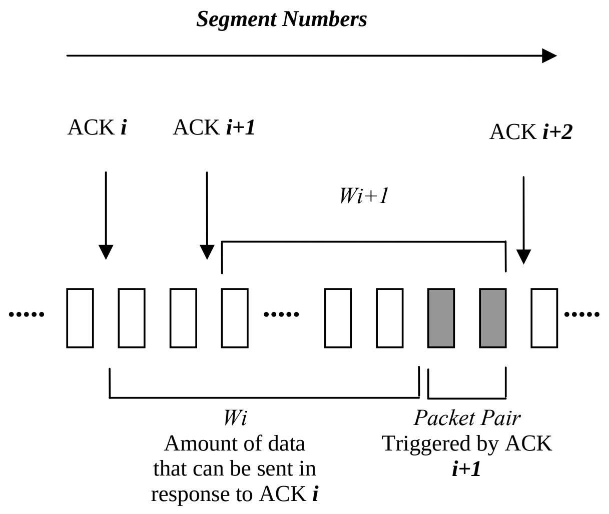

Figure 4: Association between Packet Pair and the ACK which caused it.

However, simply waiting for the next ACK and assuming the corresponding packet series to be a pair is not enough: the receiver can, for instance, change its window size with every ACK, and if it is decreased, then the above assumption may not hold. This is shown in Figure 5.

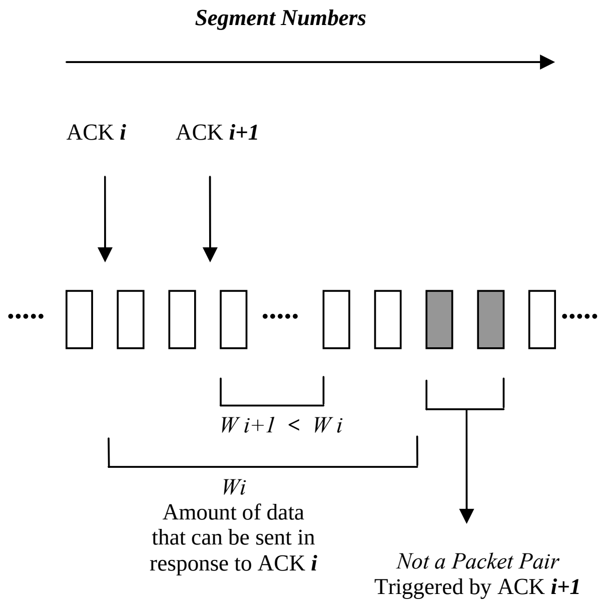

Figure 5: Wrong association between Packet Pair and an ACK

Therefore, if a series of packets is *only* within the range of a *single* ACK, then this series can reliably be considered as having been caused by that particular ACK. This is also indicated by ACK*i+2* in Figure 4 – if that ACK had been sent before receiving the pair, we could not be sure that the pair was caused by ACK*i+1* alone (and therefore actually sent as a pair).

If we observe a flow from start, then the slow start mechanism [16] also helps us in finding potential packet pairs. In the slow start phase, if the initial window size is two times the maximum segment size (MSS), then after the handshake, an initial burst of packet is always sent back-to-back [17].

Another facet of our method is that it can also be used to estimate the Round-Trip Time (RTT). According to Figure 4, the time difference between ACKi+1 and the shaded pair is precisely an RTT. The authors of [18] suggested a similar approach to estimate an *upper bound* of the RTT, the delay between ACK*i* (instead of ACK*i+1*) and the packet pair in Figure 4, which can result in higher estimates if the situation explained in Figure 5 occurs.

The reliability of our estimates will increase with the number of packet pairs we identify and the large flows in grid will help us to find a large number of packet pairs, resulting in more reliable estimates. Currently we are applying this technique at the receiver side but the grid environment will help us to carry out measurements at the sender side too, which could give us more accurate results.

4. Predicting Minimum Transfer Delay

If W is the TCP window size and p is the packet loss ratio, then from [29], we know that:

$$W = \sqrt{\frac{8}{3p}} \tag{2}$$

$$\Rightarrow \quad p = \frac{8}{3W^2} \tag{3}$$

Therefore, we can write the well-known TCP steady-state throughput equation [30]:

$$T = \frac{s}{R\sqrt{\frac{2p}{3}} + t_{RTO}(3\sqrt{\frac{3p}{8}})p(1+32p^2)} \tag{4}$$

as follows, assuming the ideal case of no background traffic:

$$T = \frac{s}{\frac{4R}{3W} + t_{RTO}(\frac{8}{W^3})(1+32(\frac{64}{9W^4}))} \tag{5}$$

Setting $t_{RTO} = 4xRTT$ as a simplification [31], and replacing W with the ideal window size $C\ x\ RTT$ (where C is the bottleneck capacity), we can derive the minimum time TCP will need to transfer a file of size F as:

$$t = F\frac{s}{\frac{4R}{3(CxRTT)} + \frac{32}{C^3(RTT)^2}(1+32(\frac{64}{9(CxRTT)^4}))} \tag{6}$$

Where s is the packet size that is used by the sender. This is the final result that clients will obtain from the *Information Manager*.

5. Related Work

Nettimer [19] is a tool which uses passive packet pair monitoring and works at the receiver side. It uses the basic formula in equation (1) to calculate the path capacity having dispersion of packet pairs and applies a density function on the samples to filter good samples from bad ones. In case of two hosts (when it is possible to deploy measurement software both at sender and receiver), the authors used “Receiver Based Packet Pair (RBPP)” [20] with their filtering density function; in case of one host they used “Sender Based Packet Pair (SBPP)” [20], if the host is a sender and “Receiver Only Packet Pair (ROPP)” [21] if the host is a receiver.

In RBPP the pattern of data packet arrivals at the receiver is analyzed by using knowledge of the pattern by which data packets were originally sent. So, it is assumed that the receiver has full timing information available. SBPP works by using arrival times of ACKs instead of arrival times of packets.

ROPP is similar to RBPP but it takes timing measurements only from the receiver. The results show that ROPP achieves accuracy within 1% of RBPP but it is easy to deploy as compared to RBPP.

The technique presented in [22] is also receiver based and uses passive monitoring; here, packet triplets are considered rather than packet pairs. The authors of [22] trigger packet triplets from the receiver side by modifying the window size and negotiating MSS. As compared to this, we don't have to change the *rwnd* and we can also take longer series of packets into account. In [23], passive monitoring is used but modifications at the sender side are suggested.

For passive measurement of RTT, the authors of [24] introduced a method on the basis of changes in the sender's congestion window. According to their method, the measurement point must accurately predict the type of congestion control used. The three way handshake during the TCP connection set up and predictable behavior of slow start has also been used for RTT estimation [17]. A statistical method, which associates a data segment with an ACK segment that triggered it [25], also provides an RTT estimate. Another method which associates data segments with the acknowledgments that triggered them is given in [26] but here, the TCP timestamp option was used for this association. The authors of [26] also proposed a second method, which infers the RTT by observing the repeating patterns of segment clusters.

6. Conclusion and Future Work

We presented a measurement system to estimate network path characteristics in a grid. The key component of the system, the *Monitor*, works at the receiver side of a TCP connection. Most of the time, our system works in a passive mode, but it can switch to an active mode according to requirements. Our measurement technique is based on the packet pair mechanism; we presented a methodology to identify packet pairs at the receiver side based on the relationship of ACKs and corresponding data packets. Our technique is also helpful in the estimation of RTT.

Currently we are working on a preliminary version of the *Information Extractor* component of the *Monitor* for a reliable estimate of the path capacity. After an accurate and reliable estimate of bottleneck bandwidth, our target is the detection of shared bottlenecks because this knowledge can enhance the usefulness of our predictions.

For experimental results we plan to use our system in Planet Lab [28], and our ultimate target is to deploy it on Austrian Grid [27].

References

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

[2] J. Nevil Brownlee. Internet Traffic Measurement: An Overview. A background paper for the ICAIS seminar. Miyazaki, Japan, March 99.

[3] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker. On the Constancy of Internet Path Properties. Proc. ACM SIGCOMM Internet Measurement Workshop, November

2001.

[4] R. S. Prasad, M. Murray, C. Dovrolis, K. Claffy. Bandwidth estimation: metrics, measurement techniques, and tools. Published in IEEE Network, November – December 2003.

[5] Q. He, C. Dovrolis, M. Ammar. On the Predictability of Large Transfer TCP Throughput. In the Proc. of ACM SIGCOMM. Philadelphia PA, August 2005.

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

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

[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] B. Tierney, R. Aydt, D. Gunter, and et al. A grid monitoring architecture. Technical Report GWD-PERF-16-2, January 2002.

[10] J. C. Bolot. End-to-End Packet Delay and Loss Behavior in the Internet. In Proceedings of ACM SIGCOM, 1993.

[11] R. L. Carter, M. E. Crovella. Measuring Bottleneck Link Speed in Packet-Switched Networks. Technical Report BU-CS-96-006, Boston University, 1996.

[12] V. Paxson. End-to-End Internet Packet Dynamics. In Proceedings of ACM SIGCOMM, 1997.

[13] K. Lai. M. Baker. Measuring Bandwidth. In Proceedings of IEEE INFOCOM, Mar 1999.

[14] K. Lai, M. Baker. Measuring Link Bandwidths Using a Deterministic Model of Packet Delay. In Proceedings of ACM SIGCOMM, August 2000.

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

[16] M. Allman, V. Paxon, W. Stevens. TCP Congestion Control. Apr 1999. IETF RFC 2581.

[17] H. Jiang, C. Dovrolis, Passive Estimation of TCP Round-Trip Times. ACM SIGCOMM Computer Communication Review. July 2002.

[18] W. Feng, M. Gardner, M. Fisk, E. Weigle. Automatic Flow-Control Adaptation for Enhancing Network Performance in Computational Grids. Journal of Grid Computing (inaugural issue). Vol. 1, No. 1, 2003, pp.63-74.

[19] K. Lai, M. Baker. Nettimer: A Tool for Measureing Bottleneck Link Bandwidth. In Proceedings of the 3rd USENIX Symposium on Internet Technologies and

Systems, San Francisco, California. March 2001.

[20] V. Paxson. Measurement and Analysis of End-to-End Internet Dynamics. PhD thesis, University of California, Berkeley, April 1997.

[21] K. Lai, M. Baker. Measuring Bandwidth. In Proceedings of IEEE INFOCOM, March 1999.

[22] C. Barz, M. Frank, P. Martini, M. Pilz. Receiver-Based Path Capacity Estimation for TCP. In Proceedings of KIVS'05, Kaiserslautern, Germany. February/March 2005.

[23] L. Chen, A. Nandan, G. Yang M. Y. Sanadidi, M. Gerla. CapProbe based Passive Capacity Estimation. Technical Report TR040023, UCLA CSD, 2004.

[24] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, D. Towsley. Inferring TCP connection characteristics through passive measurements. In INFOCOM, IEEE 2004.

[25] G. Lu. X. Li. On the correspondency between tcp acknowledgment packet and data packet. In Proceedings of ACM SIGCOMM conference on Internet Measurement, 2003.

[26] B. Veal, K. Li, D. Lowenthal. New Methods for Passive Estimation of TCP Round-Trip Times. In Proceedings of PAM 2005.

[27] The Austrian Grid Consortium. http://www.austriangrid.at.

[28] http://www.planet-lab.org/

[29] M. Hassan, R. Jain. High Performance TCP/IP Networking: Concepts, Issues, and Solutions. Prentice-Hall, 2003, ISBN:0130646342.

[30] M. Welzl. Scalable Performance Signalling and Congestion Avoidance. Springer (originally Kluwer Academic Publishers), August 2003. ISBN 1-4020-7570-7.

[31] S. Floyd, M. Handley, J. Padhye, J. Widmer. Equation-Based Congestion Control for Unicast Applications. SIGCOMM, August 2000.