

# Real Time Classification Mechanism for the Causes of Data Loss and its Integration into a High Performance Data Transfer System for Grid Computing

Phillip M. Dickens

Department of Computer Science, University of Maine, Orono, Maine 04469  
[dickens@umcs.maine.edu](mailto:dickens@umcs.maine.edu)

**Abstract.** The importance of high-performance communication to the success of Grid applications makes it critical to develop communication protocols that can take full advantage of the underlying bandwidth when system conditions permit, can back-off in response to observed (or predicted) contention within the network, and can accurately distinguish between these two situations. Achieving this goal requires the development of classification mechanisms that are both accurate and efficient enough to execute in real time. In this paper, we discuss one such classifier that is based on the analysis of the patterns of packet loss and the application of Bayesian statistics. We describe two different analysis techniques that we apply to such patterns, one based on complexity theory and one based on a simple measure of the distance between successive packet losses. In addition, we discuss the integration of the classification mechanism into the control structures of an existing high-performance data transfer system for computational Grids. We present empirical results showing that the classifier is extremely accurate, efficient enough to execute in real time, and that utilizing the information it provides can have a tremendous impact on the performance of a large-scale data transfer.

**Keywords:** Communication protocols; high-performance networks; Classification Mechanisms; Grid Computing

## 1 Introduction

Computational Grids create powerful distributed computing systems by connecting geographically distributed computational/storage facilities via high-performance networks. Such systems can aggregate tremendous computational power on a single large-scale problem, enabling scientific discovery in areas that were heretofore impossible to explore. Critical to the success of such large-scale Grid applications is a high-performance networking infrastructure that can efficiently move extreme-scale data sets between nodes on the Grid. However, even though advances in networking technologies have significantly increased the bandwidth available to Grid applications, actually obtaining a large percentage of such bandwidth has turned out to be a difficult issue.

One problem is that TCP, the transport protocol of choice for most wide-area data transfers, was not designed for and does not perform well in the high-bandwidth, high-delay networks typical of computational Grids. This has led to significant research activity aimed at modifying TCP itself to make it compatible with this new network environment (e.g., Highspeed TCP [15]), as well as systems that monitor the end-to-end network to diagnose and fix performance problems (e.g., [2, 22]). An alternative strategy has been the development of application-level protocols that can largely circumvent the performance issues of TCP. This includes, for example, UDP-based protocols (e.g., FOBS[10], UDT[16] ), and approaches that spawn multiple TCP streams for a single data flow (e.g., GridFTP [4]).

UDP-based protocols can be attractive for two reasons: First, some applications require a smooth transfer rate that can be difficult to obtain with TCP. Second, such protocols are well-suited for high-bandwidth, high-delay network environment and are able to obtain a significant percentage of the underlying bandwidth. However, because UDP-based protocols execute at the application level, the protocol developer must provide a mechanism to detect and respond fairly to competing traffic flows. Also, application-level protocols can lose data packets for any number of reasons unrelated to network congestion. This second issue can result in very poor performance if the control mechanisms interpret such loss as growing network contention and, in response, trigger very aggressive congestion control actions.

This research is developing a classification mechanism that can be used by UDP-based protocols to distinguish between data loss caused by network contention from loss caused by factors outside of the network domain. In particular, we focus on distinguishing between network contention and contention for CPU resources. This distinction is important because contention for CPU cycles can be a major contributor to packet loss in UDP-based protocols. This happens, for example, when the receiver's socket-buffer becomes full, additional data bound for the receiver arrives at the host, and the receiver is switched out and thus unavailable to pull such packets off of the network. The receiver could be switched out for any number of reasons including preemption by a higher priority system process, interrupt processing, paging activity, and multi-tasking. This last point is particularly important in a Grid environment where resource availability, including the CPU cycles allocated to a particular application, can fluctuate significantly during the execution of a long-running application.

To illustrate the importance of this issue, consider a data transfer with a sending rate of one gigabit per second and a packet size of 1024 bytes. Given these parameters, a packet will arrive at the receiving host around every 7.9 micro-seconds, which is approximately the amount of time required to perform a context switch on the TeraGrid systems [3] used in this research (as measured by Lmbench [21]). Thus the receiver does not need to be switched-out long before packets can begin to get dropped. We have observed, for example, tens to hundreds of packets being dropped when the operating system creates three to four new processes.

This paper discusses the development of a classification mechanism for the causes of data loss that is both very accurate and highly efficient. Also, we show how it is integrated into the control structures of an existing UDP-based data transfer system, and provide experimental results showing that the use of the classifier can result in significant performance gains. The classification mechanism is based on the

analysis of what we term *packet-loss signatures*, which show the distribution (or pattern) of those packets that successfully traversed the end-to-end transmission path and those that did not. These signatures are essentially large selective-acknowledgment packets that are collected by the receiver and delivered to the sender upon request. We chose the name “packet-loss signatures” based on previous studies showing that different causes of data loss have different “signatures” [12]. In this paper, we briefly describe how the signatures are analyzed and used by the classifier, and direct the interested reader to this same paper for a detailed discussion of the approach.

The major contribution of this paper is showing how a classification system can be developed, integrated into the control mechanisms of a data transfer system, and used to increase performance. This paper should be of interest to a large segment of the Grid community given the interest in and importance of exploring new approaches by which data transfers can be made more intelligent and efficient.

The rest of the paper is organized as follows. In Section 2, we discuss related work. In Section 3, we describe FOBS, the data transfer system in which the classification mechanism is implemented. We provide an overview of the classification algorithms in Section 4. In Section 5, we discuss the experimental design and provide the experimental results in Section 6. We provide our conclusions in Section 7.

## 2 Related Work

The issue of distinguishing between causes of data loss has received significant attention within the context of TCP for hybrid wired/wireless networks (e.g., [5, 6, 8]). The idea is to distinguish between losses caused by network congestion and losses caused by errors in the wireless link, and to trigger TCP’s aggressive congestion control mechanisms only in the case of congestion-induced losses. This ability to classify the root cause of data loss, and to respond accordingly, has been shown to improve the performance of TCP in this network environment [5, 20]. These classification schemes are based largely on simple statistics on observed round-trip times, observed throughput, or the inter-arrival time between ACK packets [7, 20]. Debate remains, however, as to how well techniques based on such simple statistics can classify loss [20]. Another approach being pursued is the use of Hidden Markov Models where the states are characterized by the mean and standard deviation of the distribution of round-trip times [20].

Our research has similar goals, although we are developing a finer-grained classification system to distinguish between network contention and contention for CPU resources. Another major difference is that the analysis of packet-loss signatures appears to be a more robust classifier than (for example) statistics on round-trip times, and could be substituted for such statistics within the mathematical frameworks established in these related works.

Also related are efforts such as Web100 [22] and Pathdiag [1], that provide sophisticated monitoring systems and tools with which performance issues in TCP networks can be diagnosed and fixed. The goal of these systems is to provide ordinary users, i.e., those without significant networking expertise, with high-performance

networking in a completely transparent manner. A major difference between our work and these related projects is the timescale at which each operates. In particular, these projects are iterative in nature, with possible consultation with network administrators between iterations. Our classification mechanism performs on a much smaller timescale, where it very quickly computes the probability that the cause of data loss was within the network or outside of the network. However, it is unable to diagnose performance problems such as inadequate buffer sizes, under-configured network paths, or problems with the software stack as these related works can provide. Thus while the goal of providing high-performance networking are shared, the problems being addressed are quite different. In fact, such work is orthogonal to our efforts in that any improvements to the networking infrastructure such projects can provide would also benefit the performance of our data transfer system.

Research into other application-level alternatives to TCP is also related (e.g.,[17]). However, projects such as this do not attempt to determine the root cause(s) of packet loss that is a major focus of this research.

### **3 Data Transfer System**

The test-bed for this research is FOBS<sup>1</sup>: a high-performance data transfer system for computational Grids [10]. FOBS is a UDP-based data transfer system that provides reliability through a selective-acknowledgment and retransmission mechanism. It is precisely the information contained within the selective-acknowledgment packets that is collected and analyzed by the classification mechanism. FOBS can be executed as a window-based protocol where all packets within the current transmission window are put onto the network at a constant sending rate. It can also be used as a rate-based system, where a constant sending rate is used until a congestion event occurs, at which point a new sending rate is determined based on the long-term loss rate.

FOBS is multi-threaded to take advantage of nodes with multiple processors or processors with multiple cores. In such cases, the classification mechanism can execute as a separate thread that runs concurrently with the ongoing transfer. In fact, we have observed that when the data sender is executing on a dedicated node with dual-processors, there is no additional cost incurred by executing the classifier (that is, the data transfer rate is unchanged when it is executed).

#### **3.1 Congestion Control**

An important design goal for FOBS is that it competes fairly with other network flows. Toward this end, FOBS uses a modified version of the TCP Friendly Rate Control (TFRC) protocol [18] for its congestion control. TFRC is equation-based, where the sending rate is computed as a function of the steady-state loss rate. The primary difference is that FOBS replaces the TFRC response function with that derived for Highspeed TCP [15], a more aggressive version of TCP for high-performance network environments with very low loss rates. The use of this more aggressive congestion control mechanism is completely appropriate given that FOBS

---

<sup>1</sup> Fast Object-Based Data Transfer System

is designed for the well-provisioned, high-bandwidth, high-delay networks associated with computational Grids, and is not intended for the Internet1 environment.

We do not discuss the derivation of the HighSpeed TCP response function here, and direct the interested reader to [15] for a complete analysis. For our purposes, it is sufficient to note that a parameter termed `Low_Window` is defined, which sets the lower bound on the congestion window at which the HighSpeed TCP response function will be used. That is, if the current congestion window is greater than `Low_Window`, then the HighSpeed response function will be used to determine the size of the next congestion window in the event of packet loss. This response function is defined as:

$$(1) \quad w = 0.12 / p^{0.835}.$$

Otherwise, the standard TCP response function will be used<sup>2</sup>:

$$(2) \quad w = 1.2 / \sqrt{p}$$

where  $w$  is the size of the next congestion window and  $p$  is the loss rate.

### 3.1.1 Integration of Classifier into FOBS

While the algorithms used by the classification mechanism are somewhat complex, the way the information is used by the controller is relatively simple. When a congestion event occurs, the classifier is queried to determine the cause of such loss. The classifier then assigns a probability to the event that the data loss was caused by contention for CPU resources (and thus one minus this probability that the cause of loss was network related). If the probability exceeds a certain threshold that the cause of data loss was CPU related (currently set at 95%), then no corrective action is taken and the loss rate is not modified. Otherwise, the cumulative loss rate is updated appropriately, and Equation (1) or (2) is invoked to determine the new sending rate (depending upon the size of the current congestion window).

## 4 Classification Mechanism

Having discussed how the results of the classification mechanism are used in FOBS, we briefly discuss how these probabilities are computed. The interested reader is directed to [12, 13] for a complete discussion of the statistical analysis.

### 4.1 Classification Metrics

The classification mechanism is based on the application of Bayesian statistics, which centers on how the value of certain metrics can be used to identify a cause of

---

<sup>2</sup> As noted by the authors, this equation assumes a loss rate that is less than  $10^{-2}$  where the effects of TCP retransmit timeouts can be largely ignored.

packet loss. Assume there are two causes of data loss: network contention and CPU contention. The idea is to find a metric that has very different statistical properties under the two causes of data loss, the greater the difference the more accurate the classification.

Our research has identified two excellent metrics upon which the classification mechanism is based, both of which are derived from the packet-loss signatures. The first metric is the *complexity* of the packet-loss signatures that is derived using techniques from *symbolic dynamics*. In symbolic dynamics[19], the packet-loss signature is viewed as a sequence of symbols drawn from a finite discrete set, which in our case is two symbols: 1 and 0. One diagnostic that quantifies the amount of structure in the sequence is *complexity*. There are numerous ways to quantify complexity. In this discussion, we have chosen the approach of d'Alessandro and Politi [9] which has been applied with success to quantify the complexity and predictability of time series of hourly precipitation data [14]. The approach of d'Alessandro and Politi is to view the stream of 1s and 0s as a language and focus on subsequences (or *words*) of length  $n$  in the limit of increasing values of  $n$  (i.e., increasing word length). First-order complexity, denoted by  $C^1$ , is a measure of the richness of the language's vocabulary and represents the asymptotic growth rate of the number of admissible words of fixed length  $n$  occurring within the string as  $n$  becomes large. The number of admissible words of length  $n$ , denoted by  $Na(n)$ , is simply a count of the number of distinct words of length  $n$  found in the given sequence. The first-order complexity ( $C^1$ ) is defined as

$$(3) \quad C^1 = \lim_{n \rightarrow \infty} (\log_2 Na(n)) / n .$$

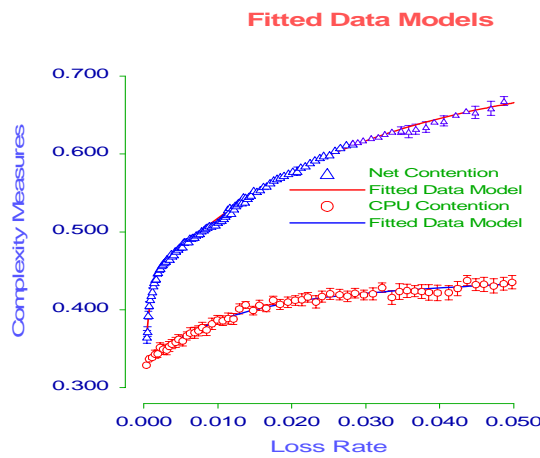
The first-order complexity metric characterizes the level of randomness or periodicity in a string of symbols. A string consisting of only one symbol will have one admissible word for each value of  $n$ , and will thus have a value of  $C^1 = 0$ . A purely random string will, in the limit, have a value of  $C^1 = 1$ . A string that is comprised of a periodic sequence, or one comprising only a few periodic sequences, will tend to have low values of  $C^1$ . We have developed simple empirical models that relate complexity measures to the different causes of data loss as a function of the loss rate, and it is these models that are used by the classification mechanism.

While the calculation of complexity measures is simple and efficient, the size of the words that can be examined in real time (without negatively affecting performance) is somewhat limited. In the experiments reported here, the maximum word size was set to  $n = 17$ , which was sufficient for discerning the basic structure of the signatures (i.e., either random or periodic) when the loss rate was greater than approximately 0.0004. However, for significantly lower loss rates, the dropped packets (and the corresponding 0s in the packet-loss signatures) were too far apart to be detected with a word size of 17. While increasing the word size can help, it cannot be increased enough to detect the randomness in the string at very low loss rates. Thus, complexity measures are unable to serve as a classification metric at very low loss rates.

To address this issue, we developed another metric based on the distance between two consecutive dropped packets. The idea is that the fundamental structure of the packet-loss signatures will not be significantly different at very low loss rates, and thus data loss caused by CPU contention will still be largely contiguous in the signature, and loss caused by NIC contention will still be random. To develop this metric, we define “success” as two consecutive packet drops (i.e., two consecutive 0s in the signature). We then performed a large number of experiments to learn the proportion of successes for each cause of data loss at very low loss rates. These proportions were then used as parameters to a beta distribution (that provides the probability of a given proportion of successes), in hopes that the statistical properties of the distribution would be very different under different causes of packet loss. The beta distribution takes two parameters, **a** and **b**, and has the following density function, where  $p$  is the proportion of successes.

$$(4) \quad p^{a-1} (1-p)^{b-1}$$

Figure (1) shows the considerable difference in the statistical properties of the complexity metric under both causes of data loss. Figure (1) further shows the empirical data models derived in association with complexity measures. This figure demonstrates quite clearly the power of this metric in distinguishing between causes of data loss. Due to space constraints, we do not show the differences of the statistical properties for the beta distribution.



**Figure 1. This figure shows the mean complexity measures at each data bin, and 95% confidence intervals around the mean, for each cause of data loss. Also, it shows how the data lay along the fitted data model.**

## 5. Experimental Design

All experiments were conducted on the TeraGrid [3]: a high-performance computational Grid that connects various supercomputing facilities via networks

operating at 40 gigabits per second. The two facilities used in these experiments were the San Diego Supercomputing Center (SDSC), and the National Center for Supercomputing Applications (NCSA, located at the University of Illinois, Urbana). The host platform at each facility was an IA-64 Linux cluster where each compute node consisted of dual 1.5 GHz Intel Itanium2 processors. The operating system at both facilities was Linux 2.4.21SMP. Each compute node had a gigabit Ethernet connection to the TeraGrid network.

We were interested in whether or not the classification mechanism could improve performance in the case where data loss was caused by contention for CPU resources. To test this, we executed one set of transfers where the results of the classification system were used by the controller, and another set where they were not used (and thus all data loss was assumed to be network related). We implemented a background process on the data receiver which periodically caused data to be dropped. The number of packets lost per congestion event was based largely on operating system behavior (e.g., process scheduling), and was thus somewhat out of our control. However, the long-term loss rate in all experiments was on the order of 0.0002. We performed three long data transfers (of about 3 hours each), under each condition (i.e., results of classifier used/not used). The results of interest were the percentage of successful classifications and the throughput achieved in each instance. The classifier used the beta distribution to compute the required probabilities when the loss rate was  $\leq 0.0004$ . Otherwise, the complexity measures were used.

We used the technique of *direct-execution simulation* [11] to determine the throughput achieved in each situation. In this approach, the behavior of the network connection, the behavior of the system in the presence of contention for CPU resources, and the packet-loss signatures generated by such contention, were all obtained by actually executing the data transfer. That is, there was an ongoing data transfer between NCSA and SDSC and a physical background process that created contention for CPU resources. The resulting packet-loss signatures were generated by such contention, and these signatures were analyzed by the classification mechanism in real time (i.e., as the transfer was progressing). Thus all of these aspects of the problem were real.

However, the results of the classification were provided to the simulator, which then determined the new (virtual) sending rate and new (virtual) loss rate by applying the congestion control mechanism described in Section (3.1). The simulator then computed the number of (virtual) seconds that had elapsed since its last invocation, and, based on this and the previous (virtual) sending rate, determined the amount of data that would have been transferred during that time. It is the throughput calculated by the simulation that is presented in the experimental results discussed in the following section.

We chose to use this approach because the physical network connection between the nodes on the TeraGrid was limited by the one gigabit link between the compute nodes and the backbone network. We wanted to study the impact of the classification mechanism with essentially the same parameters as those used to define the HighSpeed TCP response function, which assumed a 10 gigabit per second connection.

## 6. Experimental Results

Total trials	Percentage of inconclusive classifications	Percentage of incorrect classifications	P(CPU CPU)	Av. TP with Classifier	Av. TP without Classifier
265	5.6%	0.7%	94%	8697 mps	168 mps

The results of these experiments are shown in the table above. Column 1 shows that there were 265 congestion events in all six trials combined. Column 2 shows the percentage of the congestion events for which the classification was inconclusive (returning a probability of 50% for each cause of data loss). Such inconclusive classifications generally occurred at very low loss rates (i.e., less than 0.0001). Column 3 shows the percentage of incorrect diagnoses (0.7%, or 2 out of 265), where the classifier diagnosed the loss as being network related when it was in fact CPU related. This again occurred at very low loss rates. Column 4 shows the percentage of times that the classifier diagnosed the loss as being CPU related when this was in fact the cause (94%). Columns 6 and 7 show the average throughput in megabits per second when the results of the classifier were being utilized (column 6), and when they were not (column 7).

As can be seen, the diagnostic abilities of the classification mechanism were quite good, correctly diagnosing the cause of data loss 94% of the time. When it was unsuccessful, it returned an inconclusive rather than incorrect diagnosis in a vast majority of cases (15 out of 17). These results also demonstrate that having a real-time classification mechanism can significantly improve performance when the cause of data loss is contention for CPU resources (in fact, by orders of magnitude). These are all very encouraging results.

## 7 Conclusions

In this paper, we have presented a highly accurate classification mechanism that can distinguish between data loss caused by contention for CPU resources from that caused by network contention at loss rates as low as 0.0001. We have also shown that the classifier can be quite easily integrated into the control structure of FOBS, an existing high-performance data transfer system for computational Grids. We further discussed that the classifier is efficient enough to execute in real time, incurring no reduction in the transfer rate when the data sender is executing on a dedicated dual-processor node. Otherwise, we have observed a performance penalty of approximately 12%. However, the results presented here show that this is a very low price to pay when the cause of data loss is largely CPU related.

One question that this research does not answer is how often data will be lost due to contention for CPU resources. Given the highly dynamic nature of a Grid environment, it is reasonable to think that contention for CPU resources can become problematic during the execution of a long-running application. In such circumstances, the technology described here could be quite useful. However, if data loss were always caused by network contention, then this technology would not be

particularly helpful. Extensive monitoring of long-running Grid applications may help to shed light on this question.

## References

- [1]. Enabling High Performance Data Transfers. <http://www.psc.edu/networking/projects/tcptune/>
- [2]. Net100: Development of Network Aware Operating Systems. <http://www.csm.ornl.gov/~dunigan/net100/>
- [3]. The Teragrid Project. <http://www.teragrid.org>
- [4]. Allcock, W., et.al. Secure, Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing. In the Proceedings of the *IEEE Mass Storage Conference*, (2001).
- [5]. Balakrishnan, S., Padmanabhan, V., Seshan, S. and Katz, R. A Comparison of Mechanisms for Improving TCP Performance Over Wireless Links. *IEEE/ACM Transactions of Networking*, 5 (6). 756-769.
- [6]. Balakrishnan, S., Seshan, S., Amir, E. and Katz, R., Improving TCP/IP performance over wireless networks. In the Proceedings of the *ACM MOBICON*, (1995). November 1995.
- [7]. Barman, D. and Matta, I., Effectiveness of Loss Labeling in Improving TCP Performance in Wired/Wireless Networks. In the Proceedings of the *ICNP 2002: The 10th IEEE International Conference on Network Protocols*, (Paris, France, 2002). November 2002.
- [8]. Biaz, S. and Vaidya, N., Discriminating Congestion Losses From Wireless Losses using Inter-Arrival Times at the Receiver. In the Proceedings of the *IEEE Symposium ASSET '99*, (Richardson, TX, 1999). March 1999.
- [9]. D'Alessandro, G. and Politi, A. Hierarchical Approach to Complexity with Applications to Dynamical Systems. *Physical Review Letters*, 64 (14). 1609-1612. April 1990.
- [10]. Dickens, P., FOBS: A Lightweight Communication Protocol for Grid Computing. In the Proceedings of the *Europar 2003*, (2003).
- [11]. Dickens, P., Heidelberger, P. and Nicol, D. Parallelized Direct Execution Simulation of Message-Passing Parallel Programs. *IEEE Transactions on Parallel and Distributed Systems*, 7 (10). 1090-1105. October 1996.
- [12]. Dickens, P., Larsen, J. and Nicol, D., Diagnostics for Causes of Packet Loss in a High Performance Data Transfer System. In the Proceedings of the *2004 IPDPS Conference: The 18th International Parallel and Distributed Processing Symposium*, (Santa Fe, New Mexico, 2004).
- [13]. Dickens, P. and Peden, J., Towards a Bayesian Statistical Model for the Classification of Causes of Data Loss. In the Proceedings of the *International Conference on High Performance Computing and Communications*, LNCS 3726.
- [14]. Elsner, J. and Tsonis, A. Complexity and Predictability of Hourly Precipitation. *Journal of the Atmospheric Sciences*, 50 (3). 400-405.
- [15]. Floyd, S. Modifying TCP's Congestion Control for High Speeds. <http://www.aciri.org/floyd>
- [16]. Gu, Y., Hong, X. and Grossman, R.L., Experiences in Design and Implementation of a High Performance Transport Protocol. In the Proceedings of the *SC 2004*, (Pittsburgh, PA). November 6 - 12.
- [17]. Hacker, T., Noble, B. and Athey, B., Improving Throughput and Maintaining Fairness using Parallel TCP. In the Proceedings of the *IEEE INFOCOM '04*, (2004).
- [18]. Handley, M., Floyd, S., Padhye, J. and Widmer, J. [RFC 3448] TCP Friendly Rate Control (TFRC): Protocol Specification.
- [19]. Hao, B.-L. Elementary Symbolic Dynamics and Chaos in Dissipative Systems World Scientific, 1988.
- [20]. Liu, J., Matta, I. and Crovella, M., End-To-End Inference of Loss Nature in a Hybrid Wired/Wireless Environment. In the Proceedings of the *Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt '03)*, (Sophia-Antipolis, France, 2003).
- [21]. LMBench. <http://www.bitmover.com/lmbench/>
- [22]. Mathis, M., Heffner, J. and Reddy, R. Web100: Extended TCP instrumentation for research, education and diagnosis. *ACM Computer Communications Review*, 33 (3). July 2003.