Careful Protocols or How to Use Highly Reliable Networks

David Mosberger Larry L. Peterson

Department of Computer Science
University of Arizona
Tucson, AZ 85721

{davidm,llp}@cs.arizona.edu

Abstract

The motivation for the work presented in this paper stems from the observation that optical token-ring networks have bit-error rates that are low enough to be negligible for all but the most demanding applications. We define the notion of careful protocols that attempt to benefit from the reliability of such networks. Although it might seem trivial to implement protocols in the presence of a reliable network, a closer look reveals that this is not at all true. In essence, while protocols based on unreliable networks have to worry recovering from lost packets, careful protocols have to worry about flow-control.

This is work in progress, and as such, incomplete. However, first results appear to show that it might be worthwhile to use careful protocols over networks with high reliability.

1 Introduction

Communication protocols for workstations have traditionally been designed based on the assumption that the physical connection is unreliable. That is, a packet sent over the network could be duplicated, lost, damaged, or arbitrarily delayed. Traditional protocols often take advantage of this assumption by dropping packets whenever they encounter resource shortages. In other words, unreliability is a self-fulfilling prophecy.

However, many current networking technologies provide service with outstanding reliability. For example, on token-ring based networks, there are no out-of-order packets, as long as the ring operates correctly. With the advent of optical interconnections, bit-error rates became so small that for most practical purposes they can be ignored.

Protocol designers sometimes exploit this reliability by implementing so-called "optimistic" protocols. While they make optimistic assumptions about the reliability of the network, they still implement reliability in the case the optimistic assumptions are violated. So far, however, little work has been done to take the next logical step to trust *completely* in physical connection.[1]

[1] The interconnect is often trusted in an multicomputer, but to the best of our knowledge, not on clusters of workstations connected by local area networks.

This research investigates what the influence of reliable physical connections would have on protocol design. It is not clear at all whether specialized protocols could achieve considerably better networking performance compared to traditional protocols. For brevity, we call protocols that have been designed under the assumption of a reliable physical connection *careful* protocols.

The next section first defines careful protocols and discusses their potential benefits and disadvantages compared to traditional protocols. Then we discuss related work in Section 3. Section 4 presents research issues we have identified. The final section briefly discusses the work that has been done so far.

2 Careful Protocols

A *careful protocol* is defined to be a protocol offering a reliable service, provided that the lower-level protocol is reliable. It is careful in the sense that it *preserves* the reliability of the lower-level protocol in the service it provides to the higher-level protocol. A protocol is said to be *reliable* if the probability of lost, damaged, duplicated, or out-of-order delivered packets is negligibly small.

The definition of "negligibly small" is intentionally left unspecified, as it depends on the nature of the application using a careful protocol. For the kind of applications we will discuss later on, a protocol could be considered reliable even if fails on the order of once a week.

2.1 Advantages

The advantage of careful protocols shows if a whole protocol stack solely consists of careful protocols. In such a stack, no retransmission scheme is required. Neither does it have to deal with duplicate or out-of-order packets, thus simplifying the protocols considerably. The absence of a retransmission scheme makes sender-side buffering optional, which is beneficial, both in terms of lower memory demands and in terms of throughput. The latter is true because packets have to be copied less often, as they are discarded immediately after transmission.

A careful protocol stack has to be built on top of a physical connection guaranteeing ordered delivery and very low loss-rates. For example, optical rings like FDDI have these properties. Interconnects of multi-computers usually provide similar guarantees. Even if the physical connection is not sufficiently reliable, it might be worthwhile to add a protocol that would ensure it, e.g., by adding error correction. For example, satellite connections often use forward error correction to improve on bit-error rate. That is, guaranteed delivery could be implemented at the very lowest level. As there would be no need for end-to-end retransmission, the advantages would be similar to a network that is physically reliable.

To illustrate the potential advantages of careful protocol stack we give two examples. Suppose a cluster of workstations is connected over a reliable high-speed, low-latency network. In such an environment, protocol and copy overheads are likely to dominate end-to-end latency. The simplicity of careful protocols would help reduce protocol overheads considerably. Furthermore, if the data being sent is volatile, a conventional protocol would need to make a backup copy of it, in the case retransmission is needed. This copy can often be avoided in careful protocols, which leads to improved latency and throughput.

Conversely, in a high-latency environment, like a hypothetical[2] reliable, high-speed satellite link, the latency incurred by the protocol stack is negligible. However, if we consider a 1 Gbps connection with a 600 ms delay, then we can see that the bandwidth delay product is roughly 72 MB. Thus, a sender would have to buffer at least 144 MB in order to keep the pipe full! With a careful protocol stack, no sender side buffering at all would be needed.

2.2 Difficulties in Realizing Careful Protocols

The difficulty in designing and implementing careful protocols is in assuring that each incoming packet will find the resources it needs. A well-known technique for ensuring this in point-to-point communication is the sliding window. While the technique is well-understood, there is a question of exactly what resource a sliding-window controls. TCP, for example, controls the amount of buffer space expressed in bytes. This choice is not necessarily appropriate or even sufficient for a careful protocol, because the remaining buffer space is just one of many resources an incoming packet usually occupies[3]. Another problem is deadlock avoidance. A careful protocol has to guarantee that any given packet will be transmitted eventually (provided that the lower-level protocol also gives this guarantee).

When attempting to extend this idea to group communication, a multitude of design and implementation choices arises. For example, if host A knows that it is going to receive mostly from host B then it would make sense for it to advertise a large window to B and small windows to all other hosts. Thus, to achieve optimal performance and resource utilization, it is desirable to advertise windows large enough to ensure that no sender ever has to block due to filling up the window, but small enough to keep the number of unused resources low. As communication patterns are likely to change over time, optimal advertised windows will have to do so, too. Depending on the application, it might be possible to *predict* future communication demands. In general, however, estimating communication demands based on past behavior might be the best one can do. It appears that arbitrarily sophisticated schemes could be developed in the search for optimal window sizes. Of course, the complexity of this scheme competes with the goal of keeping the protocol induced latency smaller than that of traditional protocols.

2.3 Limitations

A strength of traditional protocols that they can achieve receiver resource utilizations arbitrarily close to one. If packets are received faster than they can be processed, the available buffer space will fill up and the receiver will eventually drop incoming packets. This operating point would lead to a low network utilization due to the high number of retransmitted packets. But this illustrates that any operating point between the two extremes of the receiver always having most of the buffers free and the receiver always having most of the buffers in use can be reached with traditional protocols.

Careful protocols, on the other hand, always have to stay on the safe side. A single dropped packet is as severe as a host-crash in a traditional environment. In general, this will make it

[2] Current satellite connections do not appear to provide the required reliability.

[3] In fact, in a predecessors of TCP, both the number of packets and the amount of data was controlled

difficult to achieve high resource utilization. It appears, however, that this is a minor point considering how much resources traditional protocols occupy for sender-side buffering.

A final issue is what should happen in the case disaster strikes and the lower-level protocol violates its reliability promises. Clearly, any fault-tolerating technique such as check-pointing or active redundancy could be employed. If the probability of a lost packet is on the order of the probability that a host crashes, this is actually no different than what would be needed for traditional protocols anyway. That is, even though traditional protocols deal with retransmissions, they usually do not handle host-crashes transparently. So, it is sufficient to have loss probabilities that are small enough to make fault-tolerance techniques reasonably cheap. In fact, we would expect that for many realistic applications, a fault-rate on the order of a fault per week would be acceptable.

3 Related Work

As the interconnect of multicomputers is usually trusted, it is not surprising that their communication protocols are similar to careful protocols. For example, the NX/2 node operating system for the Intel iPSC/860 uses essentially a careful protocol [1]. Of course, the small latency in multicomputer interconnects can enable techniques that are not feasible with LAN's. Thus, while the protocols are similar, they won't be identical.

We are unaware of any previous work attempting to quantify the potential performance benefit of careful protocols versus traditional ones. This is not surprising. So far, there was not really a choice: communication outside of a multicomputer was unreliable and therefore *had* to employ traditional protocols. Similarly, the interconnect of a multicomputer was *designed* to be reliable, so nobody would have considered to employ a traditional protocol. It appears that only with the advent of optical fiber interconnections a technology has arrived that can make a LAN behave more like a multicomputer interconnect than an internet.

4 Research Issues

Careful protocols raise several interesting questions. First and foremost is whether careful protocols can realize their theoretical performance win in practice. It is not clear whether the benefits will be large enough to warrant the complexity of requiring different protocols for different networks. Of course, this question makes sense only if we assume that, for example, multi-computer or local-area networks would employ careful protocol stacks, while internet communication would use traditional protocols. It would also be interesting to investigate the behavior of careful protocol stacks when reliability is implemented by the lowest-level protocol instead of the physical link.

It is less than clear how careful protocols should be designed. For example, for efficiency, it would be desirable to have only one window for a whole stack of protocols. As each layer is likely to manage different resources as well as to experience different resource demands per packet, this is not trivial. It might be that there are simple techniques that could help resolve such problems.

As soon as there are multiple point-to-point connections to a single host, or there is multicast communication, the problem of granting resources optimally becomes apparent. The goal here is to find algorithms that manage the available resources optimally, while being simple enough to be efficiently implementable.

Our approach to answering at least some of the above questions is to study them in the context of distributed, data-parallel programs. Data-parallel programs are interesting applications for careful protocols because communication is relatively well-behaved. Unlike in an environment where arbitrary programs communicate, data-parallel computations are easier to manage because there is essentially only one thread of control.

5 Status

The experimental environment for our work is a cluster of HP Precision Architecture workstations that are connected via a 100 Mbps FDDI ring. Each node in the cluster executes a Mach micro-kernel that has been extended with support for C* data-parallel programs. Currently, there are two protocol suites implementing the communication substrate. The first is a traditional, but *highly* optimistic, protocol stack that can tolerate almost any amount of unreliability in the physical link. The effect of the optimistic assumptions is that while the protocols would operate correctly over a link with high loss-rates, performance would degrade very quickly and become intolerable even with moderate loss rates in the range of a few percents.

The second protocol stack consists of careful protocols. The current implementation is straight-forward in the sense that each point-to-point link is managed independently. Only one window is used for all protocol layers and the window size is expressed in number of packets. Lost packets are not directly detected but would lead to the system locking up, so no damage would be done. However, it would be straight forward to add sequence numbers to detect lost packets directly so that appropriate actions could be taken automatically. We intend to use this stack as a proof of practicality and to establish a performance base-line. Furthermore, we hope that studying its behavior will unveil areas that warrant further research.

At this point, both protocol stacks are operational. Initial performance measurements indicate that careful protocols really can be a win. For example, we are seeing end-to-end latency improvements of nearly 30% (168μs vs. 234μs). Considering that the traditional protocol doesn't have any explicit flow-control, this is rather substantial. Also, throughput was measured to be almost 60% better (89 Mbps vs. 56 Mbps—not including the overhead to produce/consume the packet).

References

[1] R. Berrendorf and J. Helin. Evaluating the basic performance of the Intel iPSC/860 parallel computer. *Concurrency: Practice and Experience*, 4(3):223–240, May 1992.