Implementation Experience with OMG's SCIOP Mapping

Gautam Thaker, Patrick Lardieri, Chuck Winters,
Ed Mulholland, Jason Cohen, Keith O'Hara and Gaurav Naik

Lockheed Martin Advanced Technology Laboratories
3 Executive Campus, 6th Floor
Cherry Hill, NJ 08002
{gthaker, plardier, cwinters, emulholl,
jcohen kohara, gnaik}@atl.lmco.com

Abstract. Longevity of distributed computing middleware standards, such as CORBA, depend on their ability to support a range of applications by providing low overhead access in a uniform manner to a large variety of platforms and network capabilities. OMG's recent adaptation of Stream Control Transmission Protocol (SCTP) mapping is another instance of this trend. Applications can obtain all the benefits of this emerging protocol via a standard compliant, distributed object model. This paper reports on integration of SCTP with Adaptive Communications Framework (ACE) [2] and The Ace ORB (TAO). [3] By exploiting network path multiplexing capability of SCTP we demonstrate that CORBA applications can bound the *maximum* latencies they suffer under stressful network failures to under 50 msec.

1 Introduction

Middleware and network infrastructure that support many distributed, real-time, embedded systems, such as command and control systems aboard naval ships, must support stringent quality of service (QoS) requirements. These QoS requirements generally span a broad spectrum, from timeliness to fault tolerance to security. Here we report on our ongoing efforts to significantly enhance network link failure tolerance for Distributed, Real-time, Embedded (DRE) systems by exploiting the emerging Stream Control Transmission Protocol (SCTP), a new transport protocol standard from IETF [1]. SCTP provides numerous new capabilities that promise to provide significant QoS application benefits. Ideally these new capabilities become available to applications as familiar abstractions offered by object oriented middleware frameworks and standards. We report on integration of SCTP with Adaptive Communications Framework (ACE) [2] and The Ace ORB (TAO) [3]. This integration was greatly facilitated by design of ACE and TAO that anticipate addition of such newer protocols under existing abstractions. The end result is that applications achieve tightly bounded latency values even in presence of repeated network failures and resto-

rations. Furthermore, where SCTP protocol is available ACE/TAO applications need not even be recompiled to obtain these benefits.

2 Summary of SCTP and SCIOP

Stream Control Transmission Protocol was originally developed in the telecommunications industry and has become a full IETF standard. SCTP, like TCP, is a connection oriented, transport layer protocol. However, SCTP differs in following ways from TCP:

- Network path multiplexing. Rather than forming a connection, SCTP forms an association between two processes. An association may consist of multiple, distinct network paths and multiplexing of traffic among these paths is automatically handled by SCTP.
- Reliability and ordering parameter configuration. Unlike TCP, SCTP provides fine grained control over reliability and ordering parameters. Among the parameters that are directly controllable are retransmission timeouts, acknowledgement delays, heartbeat intervals, maximum number of retransmissions, as well as others. Used carefully these have the potential to permit much greater control over network behavior, something that is of fundamental importance in DRE systems.
- Message (as well as byte stream) service. Unlike TCP that is strictly byte oriented and leaves it up to higher levels to provide message framing, SCTP natively supports messages.
- Connection multiplexing (multiple streams). Multiple streams can be multiplexed over a single association. This can be used to overcome familiar "head of queue blocking" problems.
- Security enhancements for better denial of service protection. SCTP uses a 4-way (rather than TCP's 3-way) handshake in connection establishment that is not susceptible to traditional DoS attacks.
- Support for very large windows for high bandwidth-delay product links.

Currently numerous implementations of SCTP are available on Linux, Sun, and various BSD variant operating systems. We have concentrated on Linux based implementations for our project due to its greater rate of development in support of real-time issues. On Linux there are at least two kernel level implementations available; one from OpenSS7 as a patch against 2.4.18 kernel and another from LKSCTP project that is integrated in 2.5.x developmental series and is thus in the 2.6-test kernels. Of these we have focused our efforts on OpenSS7's implementation since it seemed to have better support for multi-homing, both at the time we started our project in March 2002 and as of this writing, February 2003. In addition, OpenSS7's implementation closely follows existing BSD style socket API and this has facilitated its rapid integration into ACE. Nevertheless, we support both these implementations under ACE and TAO.[1]

---

[1] OpenSS7 support was released in ACE 5.3.3 and TAO 1.3.3. LKSCTP is in the CVS repository and will be released with ACE 5.3.4 and TAO 1.3.4.

The overwhelming success of IP protocol can be attributed to its ability to support a large number of applications and services above the network layer and simultaneously support a large number of different data links and physical lower layers. In the field of distributed object computing, CORBA has the potential to provide a similar capability by supporting many applications and services on a wide array of network transport protocols. OMG's adaptation of GIOP SCTP protocol mapping (known as SCIOP) [6] continues this trend. In addition to defining client and server roles in forming an association, the SCIOP specification defines the SCTP Inter ORB Reference (SCIOR) profile and enumerates the protocol properties that must be configurable from a CORBA application in a conformant implementation.

3 DRE Application System Model

Naval shipboard command and control systems are our motivation for this work. The computing environment of these ships consist of several hundred compute nodes richly interconnected by high performance switches. Reliability and damage tolerance requirements dictate that compute nodes are spatially distributed and are inter-connected by multiple, independent network paths. Figure 1 shows a logical diagram of such a system. To keep the figure readable, only a few nodes are shown and only two of the nodes are shown to be multi-homed; however, in actual system most nodes will have two to four redundant network connections. We wish to use SCTP's support for network path multiplexing to exploit this rich interconnect redundancy and to provide the application with a high degree of network path loss tolerance.

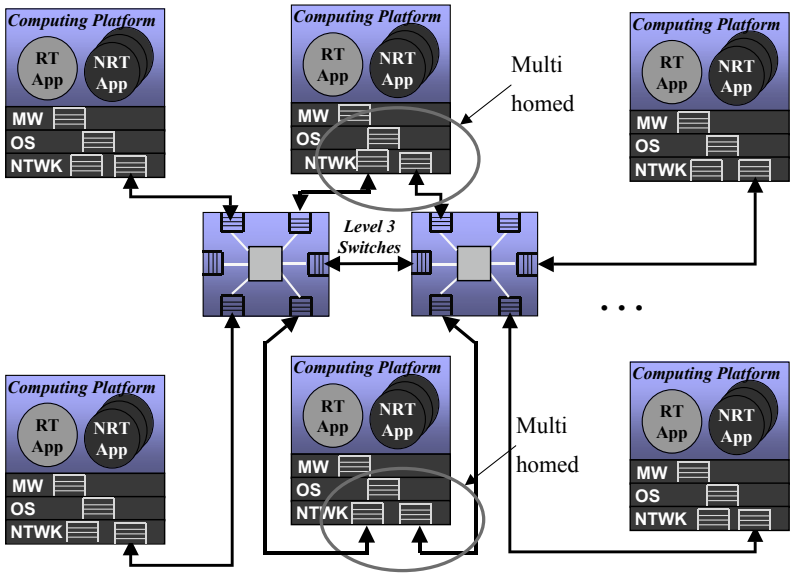

Figure 1. Expected Shipboard Deployment of Interconnect Computer Nodes

4 SCTP Support in ACE

SCTP supports two types of network service: SOCK_STREAM and SOCK_SEQPACKET. To integrate SCTP's SOCK_STREAM transport service into ACE we made a small modification to the current SOCK_STREAM wrapper facade. We added a protocol parameter to one constructor and one connect method of the ACE_SOCK_Connector class. After this modification the ACE SOCK_STREAM wrapper facade worked properly over both TCP and SCTP.

To integrate SCTP's SOCK_SEQPACKET transport service into ACE we created a new wrapper facade, SOCK_SEQPACK. We closely emulated the current SOCK_STREAM wrapper facade to develop our new SOCK_SEQPACK wrapper facade. Figure 2 depicts the classes that implement this new wrapper facade. Also indicated are those methods that have a substantial change from their SOCK_STREAM wrapper façade counterparts.

To enable the user to fully exploit the network path multiplexing features of SCTP we created a new subclass of ACE_INET_Addr called ACE_INET_Multihomed_Addr. This class enables applications to specify restricted subsets of network interfaces for inclusion on SCTP associations on the client and server side. This is also depicted in Figure 2.

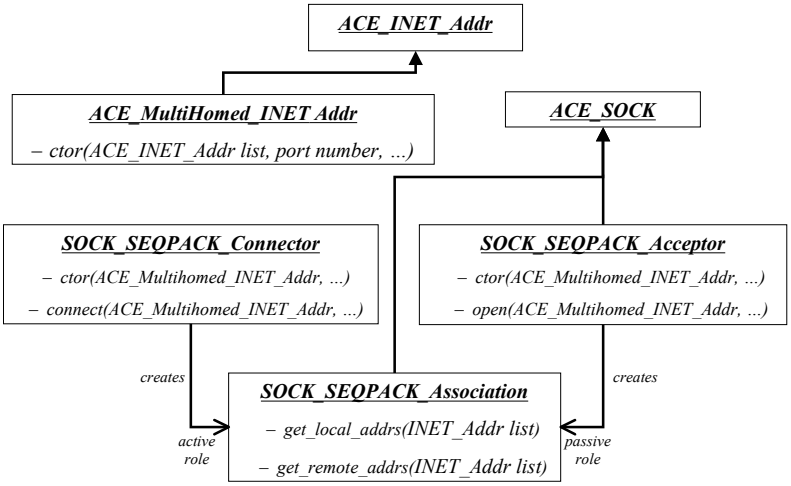

Figure 2. Wrapper-Façade Implemented on OpenSS7 API and LKSCTP, which is Supported by Current ACE::OS Methods

All SCTP options can be read and written from the current socket options methods provided by ACE_SOCK. Finally, although Figure 2 shows that our SOCK_SEQPACK wrapper facade support SCTP stream multiplexing capability, this has not been implemented in ACE or exploited in TAO. The SCIOP mapping [6] suggests that streams *may* be used by RTCORBA for transport level priority banding. Todate we have not completed the necessary design and implementation work to exploit SCTP streams.

After completion of support for OpenSS7's implementation of SCTP on the Linux platform another implementation with slightly different API and semantics emerged. This was the LKSCTP implementation that is in the 2.5.x series Linux kernel. While OpenSS7 supports both SOCK_STREAM and SOCK_SEQPACKET as connection oriented service, under LKSCTP only SOCK_STREAM is connection oriented. Thus, when ACE is built for LKSCTP the SOCK_SEQPACKET wrapper façade is mapped to SOCK_STREAM. The result is that all differences between these two implementations are abstracted and hidden under ACE wrapper façade classes and applications (including TAO) run without any changes.

5 SCTP Support in TAO

The Object Management Group has standardized a GIOP protocol mapping for SCTP that is referred to as SCIOP. The original RFP was issued in September 2000 and after the customary process of independent submissions and consolidation, a unified submission was adopted. Our approach for integration of SCTP in TAO closely followed our approach with ACE—reusing possible exiting patterns in TAO as we had done at the ACE level. We have emulated the use of TAO's pluggable protocol frameware [5] for IIOP to develop SCIOP.

OMG's proposed standard also includes a format for SCIOR (SCTP Inter-operable Object Reference). We have also enhanced TAO to generate proper SCIORs. We have modified the TAO program, catior, which deciphers IORs and pretty prints them, to also properly handle SCIORs. Compared to IIOP, the principal difference is that rather than having a single IP address and a port, an address list and a port are encoded in the IOR.

6 Performance Test Methodology

Our adaptation of SCTP to ACE and TAO must demonstrate real QoS benefits to applications to warrant its use. Towards this end we undertook extensive performance evaluation. Our test methodology is based on defining metrics that we use to judge the efficacy of our approach and implementation. Thus, we focused on the following parameters:

- Maximum and Mean end-to-end latencies under:
  - Normal (no failures) condition.
  - In presence of following types of network failures
    (1) Random 1%-5% packets losses.
    (2) Single link failure.
    (3) Repeated, rapid link failures and restorations.

Goal: 50 msec maximum latency under all test conditions. (This metric for SCTP is compared with the same tests performed with TCP.)

Our objective is to achieve this performance goal with minimal to no application level impact. Indeed, due to TAO's ability to dynamically load protocol factories, the application does not even have to be recompiled to receive these benefits.

7 Test Measurements

Figure 3 shows the testbed we used for evaluating our SCTP implementations in both ACE and TAO. For random packet losses we deploy a custom Linux kernel module that provides uniformly distributed, random packet drops. Typically we test with a 1% packet loss, which represents a high loss channel, but we have also tested with packet loss rates as high as 5%. For single link failures we simply disconnect the RJ-45 ethernet connector at the network interface card. For repeated, rapid link failures and restorations we have constructed a Network Failure Test Appliance shown in the Figure 3. This appliance uses a programmable IC and four solid state relays to achieve any desired pattern of link failures and restorations. Tests that we conduct with the Network Failure Test Appliance represent conditions that are more severe than are likely to be encountered in practice.

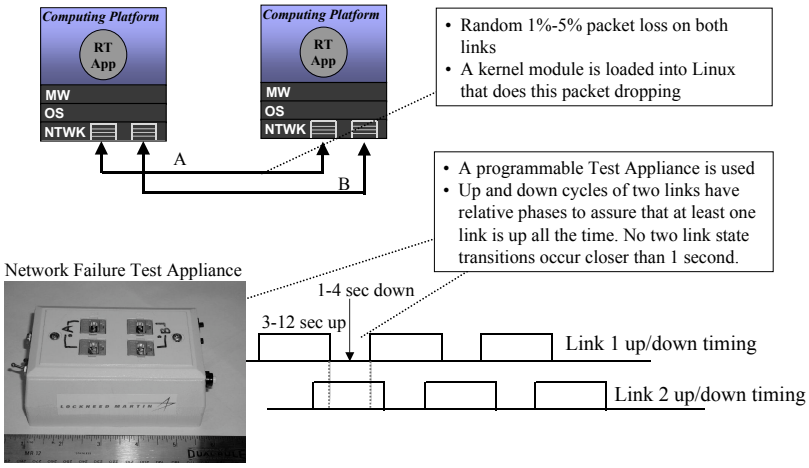

Figure 3. Experimental Testbed for Failure Testing

Table 1 details the experiments in our test plan. As shown, while we have successfully concluded most of the tests, additional work remains for some of the tests where we note "Issues." Specific cases are further discussed below. There are a total of six test cases, two at the socket level, two at the ACE level and two at the TAO level.[2] First we examine the mean roundtrip latency values on our testbed for all six cases. This is shown in Figure 4. Due to current instability in the SCTP stack for message sizes greater than 1 MTU we have limited most of our tests to messages of 1024

[2] At each level there are two tests, one based on TCP and one based on SCTP.

Table 1. Current Status of Experiments

	No Failure	Induced Packet Loss (1%)*	Single Link Failure	Repeated Link Failure
TCP (SOCK_STREAM)	Done/OK	Done/OK	Done/OK	Done. OpenSS7 is OK. LKSCTP has some issues
STCP (SOCK_SEQPACK)	Done/OK	Done/OK	Done/OK	
ACE_SOCK_STREAM (TCP)	Done/OK	Done/OK	Done/OK	
ACE_SOCK_SEQPACK (SCTP)	Done/OK	Done/OK	Done/OK	
TAO_IIOP	Done/OK	Done/OK	Done/OK	
TAO_SCIOP	Done/OK	Done/OK	Done/OK	

*1% packet loss is imposed on both links

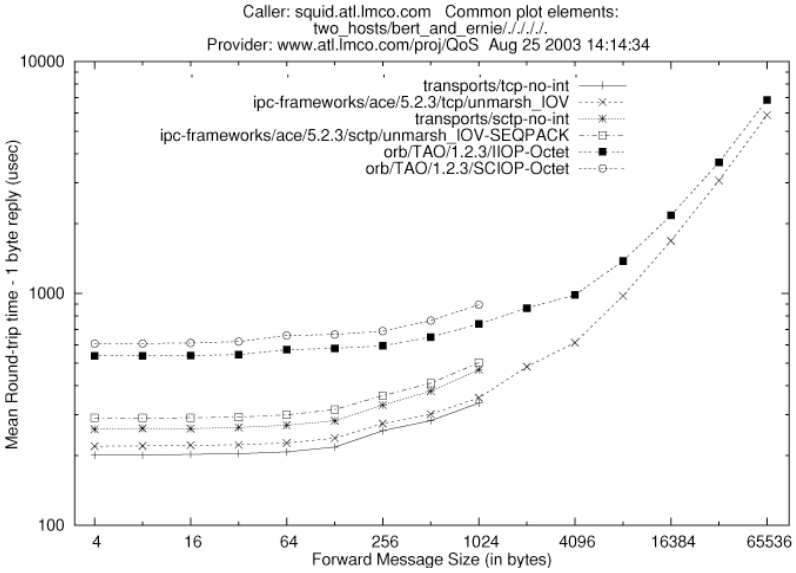

Figure 4. Mean Roundtrip Latencies

bytes or less. At all three levels, for small message sizes, our SCTP-based test is between 60-70 microseconds slower than the corresponding test using TCP. The fact that this difference remains unchanged as we move up the levels of abstractions tells us that our SCTP adaptations at the ACE and TAO levels are as efficient as existing TCP based adaptations.

Next, we examine the mean and maximum latencies when the 1% random packet loss module is enabled and compare this to the base case when there are no packet losses. Figures 5, 6, and 7 show these results for socket, ACE and ORB levels, respectively. In these graphs for each message size (shown on the x-axis) we show vertical lines that note the [min, max] latency range. In addition, a small symbol marks the mean value of observed data. Note that the y-axis is in logarithmic scale. In all tests the sample size is 1,000,000 roundtrip invocations.

The results show that under base tests where there are no packet losses, we experience only modest maximum latency of less than 1 msec. However, when 1% random losses are introduced, maximum latencies for TCP increases by about three orders of magnitude to well over 1 second.[3] However, SCTP maximum latencies do not increase beyond 30 msec which, we note, is below our original goal of keeping such latencies below 50 msec. This pattern of results is observed across the socket (Figure 5), ACE (Figure 6), or ORB (Figure 7) levels, which confirms that the maximum latency values are entirely controlled by the transport (and OS) issues.

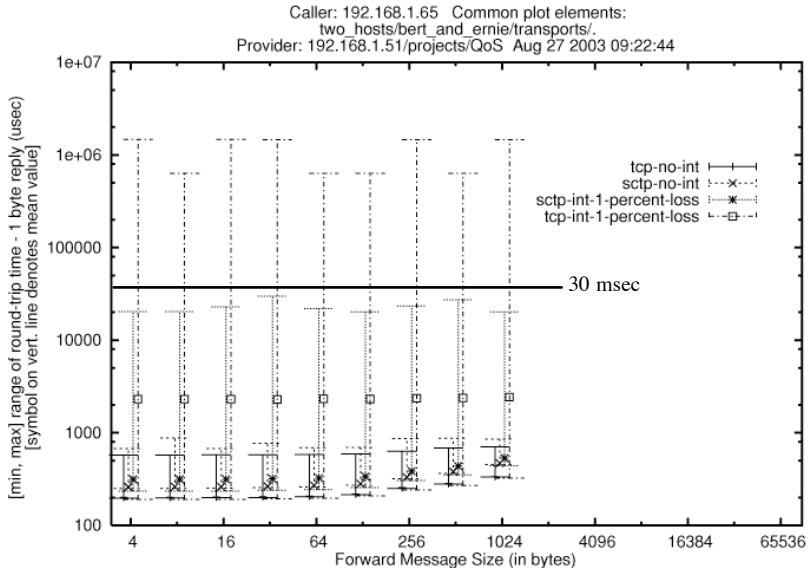

Figure 5. Comparing TCP and SCTP at Socket Level

Finally we discuss the repeated, rapid link failures and restoration test case. We program the Network Failure Test Appliance to follow a pattern of 4 seconds down and 12 seconds up. The two links, A and B (see Figure 3), follow this pattern with an 8 second phase offset. This assures that at least one link is up all the time and that no two link state transitions occur closer than 4 seconds apart. This test, as expected, puts severe stress on the system. We conducted approximately 1,200 experiments with each experiment consisting of 10,000 roundtrip invocations. Maximum latency for each experiment is show in Figure 8. Following our observations with the 1% packet loss tests we would expect to record no maximum values greater than 30 msec. Indeed, most samples fall at or below 30 msec. Thus, even under unrealistically severe patterns of network path failure and restoration, we observe a nicely bounded application performance.

---

[3] Due to extensively studied and well documented nature of TCP protocol this result is not surprising. Our goal is to compare how SCTP fares in comparison.

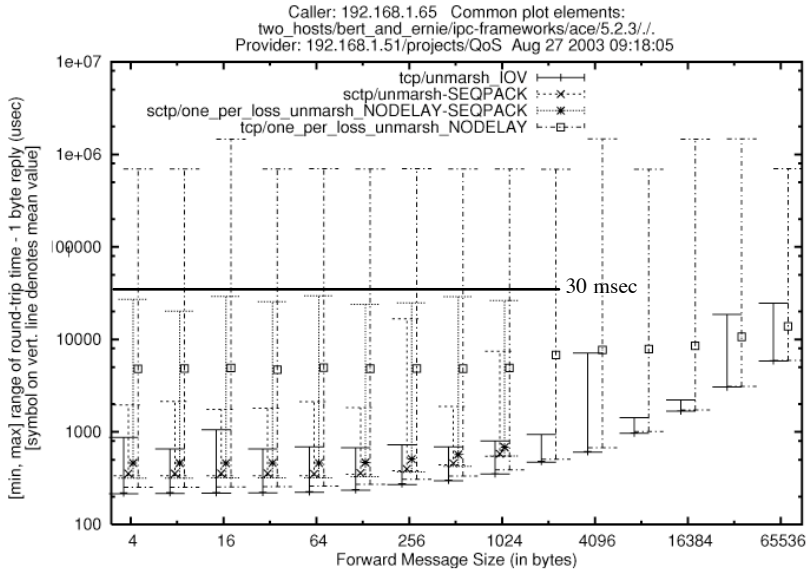

Figure 6. Comparing TCP (SOCK_STREAM) and SCTP (SOCK_SEQPACK) at ACE Level

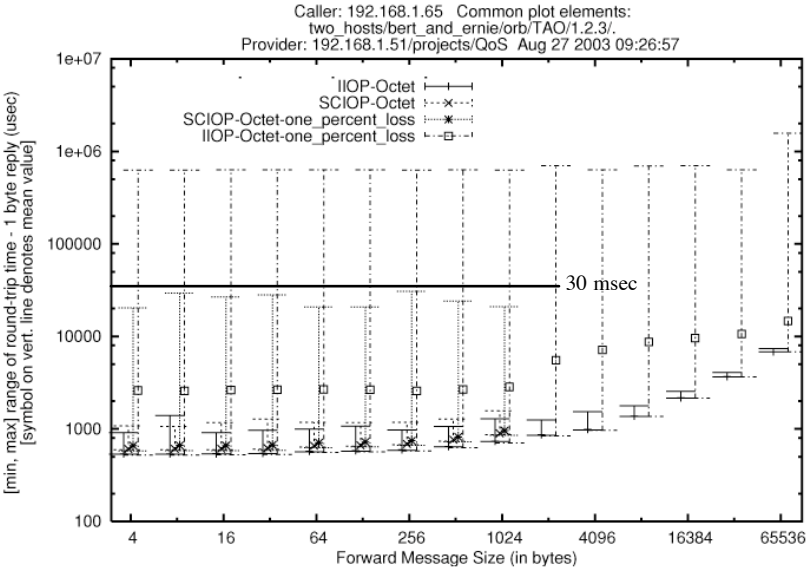

Figure 7. Comparing TCP (IIOP) and SCTP (SCIOP) at ORB Level

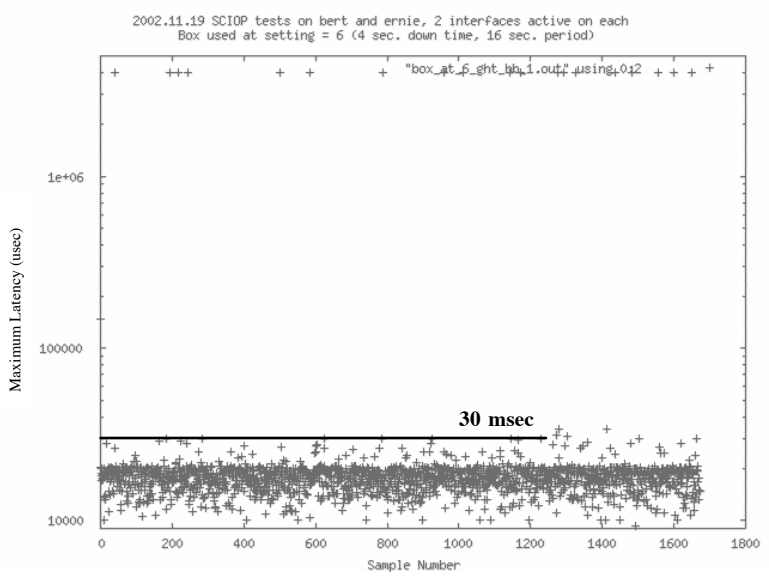

Figure 8. Repeated Link Failure Test

8 Summary

We have successfully integrated SCTP in ACE and TAO by exploiting existing, well documented patterns and frameworks. Resulting systems show significantly improved ability to tolerate network link failures.

As of this writing we are experiencing significant stability and performance problems with the LKSCTP protocol stack. We are working with the developer community to overcome these difficulties. There is every reason to believe that both the OpenSS7 and LKSCTP stack will provide similar performance over the long run as they are on the same kernel. Table 2 summarizes the key protocol property settings. The IETF column has the recommended default values. These are too conservative for LAN environment for shipboard command and control. The other two columns show the aggressive settings that we use. Note that LKSCTP does not permit RTO (retransmission timeout) to be less than 1 second, which is too coarse for our requirements.[4] (Value of 0 msec under OpenSS7 reverts to a minimum of "1 tick" or 10 msec in 2.4.18 Linux kernel).

[4] LKSCTP community is in he process of moving to msec granularity RTO values.

Table 2. Protocol Parameeer Settings

Parameter	IETF	Openss7	LKSCTP
assoc_max_retrans	10	25	25
heartbeat_ivtl	30	1	1
init_retries	8	25	25
max_path_retrans	5	0	1
rto_initial	30s	0ms	1s
rto_max	60s	0ms	1s
rto_min	1s	0ms	1s

9. Future Work

Future work includes completing support for SCTP streams and proper integration of differentiated services with streams support. We are also working on providing support for SCIOP protocol properties. For experiments described in this paper we set various protocol properties by the /proc file system. Another possibility is to enhance some of the CORBA services, such as Audio/Video streams or RT/FT event channel, to exploit SCTP.

Acknowledgments

Primary sponsor for this projects was the U.S. Defense Advanced Research Projects Agency under the Program Composition for Embedded Systems Project, Contract #F33615-01-C-1847.

In addition, we would like to acknowledge detailed comments and suggestions from the reviewers, especially reviewer number three.

References

1. Stewart,, R., et al. “Stream Control Transmission Protocol,” RFC 2960, October 2000
2. Schmidt, D.C., Huston, S.D., “Mastering Complexity with ACE and Patterns.” Addison Wesley Professional, C++ Network Programming, Vol. 1-2
3. Schmidt, D.C., Levine D., and Mungee, S., “The Design and Performance of Real-Time Object Request Brokers,” Computer Communications, Vol. 21, No. 4, April 1998
4. Schmidt, D.C., “Acceptor and Connector: Design Patterns for Initializing Communication Services,” Pattern Languages of Program Design (R. Martin, F. Buschmann, and D. Riehle, eds.), Reading, MA: Addison-Wesley, 1997
5. O'Ryan, C., Kuhns, F., Schmidt, D.C., Othman, O., and Parsons, J., “The Design and Performance of a Pluggable Protocols Framework for Real-time Distributed Object Com-

puting Middleware," Proceedings of the IFIP/ACM Middleware 2000 Conference, Palli-sades, New York, April 3-7, 2000
6. Borland Software Corp., Nokia, Objective Interface Systems, Inc. GIOPSCTP Protocol Mapping, mars/03-05-01 and mars/03-05-02 Object Management Group, May 2003 http://doc.omg.org/mars/03-05-01,http://doc.omg.org/mars/03-05-02