SCIOP Implementation in a Real-time ORB Using an Extensible Transport Framework

OMG Real-time Workshop

July 17, 2003

Patrick Lardieri    Gautam Thaker
Chuck Winters    Keith O’Hara
Jason Cohen    Gaurav Naik
Edward Mulholland
Navy’s Next Generation Vision

**FUTURE – Total Ship Computing (TSCE)**
- 1,000s of computer nodes connected by standard/COTS middleware on distributed switched backplane
- N-version redundancy (no single failure point)
- Virtually unlimited growth capability
- Software replicated on many CPUs/nodes
- Essentially invulnerable to battle damage

**TSCE Design Goals**
- Use COTS Infrastructure Technology
- Enable Plug-n-Play Component Architecture

**TSCE Benefits**
- Improve Performance
- Increase Extensibility
- Break Apart Application Stovepipes

**TODAY - Adjunct Processing (Aegis Baselines 5P3, 6)**
- UYK-43s w/COTS processors
- Point-to-point interconnection
- Display LAN
- Limited growth capability
- Vulnerable to damage

**UNDER DEVELOPMENT – Networked Processing (Aegis Baseline 7)**
- Open HW + operating system (COTS/industry standards)
- Distributed LAN interconnects
- Redundancy plus reconfigurability
- Significant growth capability
- Software distribution possible
- Vulnerable to large scale damage
- Highly constrained by legacy stovepipe systems
Notional Shipboard Deployment

Lockheed Martin Advanced Technology Laboratories
**Key Engineering Challenge**

- **Bounded time recovery from system failures**
- **Via encapsulated, adaptive capabilities within**
  - Networks
  - Computing Platforms
- **Transport Protocols**
  - Hardware Capabilities
    - Infrastructure Middleware
    - Distribution Middleware
    - Common Services

Complete solution requires overlapping and coordinated capabilities across the layers.

SCTP enables applications to immediately recover from a network fault while other mechanisms (e.g. HSRP) work to heal the network at a slower rate.

- **Enabling composition of RT & FT systems from reusable application components**

SCTP and SCIOP can help address this challenge.
Lockheed Martin Advanced Technology Laboratories

Problem Overview

Stream Control Transport Protocol (SCTP)

- Developed by the telecommunications industry for robust switch control
- Provides
  - Connection oriented byte and message stream service
  - Connection multiplexing (multiple streams)
  - Network path multiplexing
  - Reliability and ordering parameter configuration
  - Multiple types of service
    - SOCK_SEQPACKET
    - SOCK_STREAM
    - SOCK_RDM
Problem Overview (cont.)

- **SCTP Inter-Orb Protocol (SCIOP)**
  - An extension to GIOP that leverages the features of SCTP (OMG standardization Completed May 2003)
  - A primary goal, make CORBA objects resilient to network failures

- **LM ATL Goal**
  - Develop an SCTP pluggable protocol for TAO that conforms to the OMG SCIOP standard
    - OMG TC Document mars/2003-05-03
  - Demonstrate bounded time recovery of CORBA object interactions after a network failure
Design Approach

- Initial Design leveraged OpenSS7 SCTP implementation for Linux – Recently extended support to LKSCTP implementation
  - Kernel module providing IPPROTO_SCTP
  - Supports SOCK_SEQPACKET, SOCK_STREAM, and SOCK_RDM

- Develop New ACE Wrapper Façade
  - Delivers a SOCK_SEQPACKET service
    - SOCK_SEQPACK_Acceptor
    - SOCK_SEQPACK_Connector
    - SOCK_SEQPACK_Association

- Develop New TAO Pluggable Protocol
  - Delivers an SCIOP service
Preserves existing Berkley Unix networking API

- Implementation of `bind(…)` and `connect(…)` accept multiple `sockaddr_in` structures
  - `bind(sock_fd, (struct sockaddr *) addr_list, addr_list_size*sizeof(struct sockaddr_in))`

- Implementation of `accept(…), getsockname(…) and getpeername(…) return multiple `sockaddr_in` structures
  - `accept(sock_fd, (struct sockaddr *) peer_list, MAX_NUM_ADDRS * sizeof (struct sockaddr_in))`

- New “socket options” and “sendmsg(…) flags” to implement multiple streams
  - `SCTP_ISTREAMS, SCTP_OSTREAMS, SCTP_SID`
LKSCTP Design

- **API based on IETF SCTP Sockets Draft**
- **Uses existing Berkley API for single-homed associations**
- **New bindx API for multi-homed associations**
  - `sockaddr_storage (RFC2553) for holding addresses`
  - *Traditional bind(…) for primary address, and sctp_bindx(…) for secondaries*
    - `sctp_bindx(int sd, struct sockaddr_storage *addrs, int addrcnt, int flags);`
  - `sctp_getpaddrs/getladdrs to get local/peer addresses`
    - `Sctp_getpaddrs(int sd, sctp_assoc_t id, struct sockaddr_storage **addrs);`
# OpenSS7 and LKSCTP SOCKET Types

<table>
<thead>
<tr>
<th></th>
<th>TCP Style</th>
<th>UDP Style</th>
<th>TCP Compat.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TCP Style</strong></td>
<td>reliable, connection oriented, msg based</td>
<td>reliable, connectionless, msg based</td>
<td>reliable, connection-oriented, byte stream</td>
</tr>
<tr>
<td><strong>OpenSS7</strong></td>
<td><strong>SOCK_SEQPACKET</strong></td>
<td><strong>SOCK_RDM</strong></td>
<td><strong>SOCK_STREAM</strong></td>
</tr>
<tr>
<td><strong>LKSCTP</strong></td>
<td><strong>SOCK_STREAM</strong></td>
<td><strong>SOCK_SEQPACKET</strong></td>
<td>None</td>
</tr>
</tbody>
</table>
ACE SCTP Wrapper-Facade Design

- Create new ACE wrapper-façade for SCTP
  - ACE_SOCK_SEQPACK_*
  - Support multiple protocol versions
    - For LKSCTP use POSIX SOCK_STREAM
    - For OPENSS7 use POSIX SOCK_SEQPACK
  - Add lightweight stream support (TBD)
- Use ACE_SOCK_* wrapper-façade for TCP as a template
- Enhance with explicit support for address control on multihomed machines
- Enhance SOCK_* wrapper-façade to also work with SCTP

ACE SOCK

ACE_INET_Addr

ACE_Multihomed_INET_Addr
  - ctor(INET_Addr list, port number)

SOCK_SEQPACK_Connector
  - ctor(Multihomed_INET_Addr)
  - connect(Multihomed_INET_Addr)

SOCK_SEQPACK_Acceptor
  - ctor(Multihomed_INET_Addr)
  - open(Multihomed_INET_Addr)

SOCK_SEQPACK_Association
  - get_local_addrs(INET_Addr list)
  - get_remote_addrs(INET_Addr list)

Lockheed Martin Advanced Technology Laboratories
Use IIOP_* Pluggable Protocol Implementation as a template
  » Primarily substituted SCIOP for IIOP in all implementation files
  » Enabled by
    ● Pattern oriented design of pluggable protocol framework
    ● Nearly identical semantics between ACE_SOCK_* and ACE_SOCK_SEQPACK_* wrapper-facades
  » Used ACE_SOCK_SEQPACK wrapper-façade as PEER_Acceptor and PEER_Connector

Fully implemented the Stream Control Interoperable Object Reference (SCIOR)
  » Example on following slide

SCTP Protocol Properties (TBD)
  » Use TCP Protocol Properties as a template

Does not support SCTP Streams
  » Substantial design effort
decoding an IOR:
   The Byte Order: Little Endian
   The Type Id:
   "IDL:ORBPerfTest/SIISyncLatency:1.0"
   Number of Profiles in IOR:      1
   Profile number: 1
   SCIOP Version: 1.0
   Addresses: 3
   ▪ Host Name: utica
   ▪ Host Name: utica-b
   ▪ Host Name: utica-a
   Port Number: 32768
   Max Streams: 1
   Object Key len: 27
   Object Key as hex:
   14 01 0f 00 52 53 54 32 32 5d 3e e0 c3 01 00 00
   00 00 00 01 00 00 00 01 00 00 00
   The Object Key as string:
   ....RST22]>...............
Testing Methodology

• Metrics
  » Maximum & Mean Recovery Time
    • Random packets losses and Single link failure
    • Goal: 50 millisecond maximum
  » Recovery Time Stability
    • Change in mean and maximum recovery time over large numbers of repeated failures and recoveries
    • Goal: no growth in maximum recovery time
  » Application Design Impact
    • Degree to which application code must be changed to benefit from recovery features
    • Goal: No application code change for recovery from network failures

• Measure under
  » Normal and failure conditions
  » TCP, SCTP, IIOP and SCIOP
Experimental Approach: Random Losses & Link Failures

1. Random, 1%-5% packet loss on both the links.
   - A kernel module is loaded into Linux that does this packet dropping.

2. Up and down cycles of two links have relative phases to assure that at least one link is up all the time. Furthermore, no two link state transitions occur closer than 1 second.
# Testbed Configuration

## Bert
- Dual 350 MHz P-II
- 2 100 Mb/s Ethernet Cards
- Linux 2.4.18 Kernel (UniProc)
- OpenSS7 SCTP Module 0.2.10b
- N % Packet Loss Module

## Ernie
- Dual 350 MHz P-II
- 2 100 Mb/s Ethernet Cards
- Linux 2.4.18 Kernel ( UniProc)
- OpenSS7 SCTP Module 0.2.10b
- N % Packet Loss Module

### Test Software Runs
- As root
- In SCHED_FIFO
- On Unload Machine

### Key SCTP Parameters
- Set to maximize failure performance (more on following slide)

### Similar tests uploaded and executed on Emulab
- [www.emulab.net](http://www.emulab.net)

### Results available at
# Key SCTP Parameters

<table>
<thead>
<tr>
<th>parameter</th>
<th>IETF</th>
<th>Openss7</th>
<th>LKSCTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>assoc_max_retrans</td>
<td>10</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>heartbeat_ivtl</td>
<td>30s</td>
<td>1s</td>
<td>1s</td>
</tr>
<tr>
<td>init_retries</td>
<td>8</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>max_path_retrans</td>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>rto_initial</td>
<td>30s</td>
<td>0ms</td>
<td>1s</td>
</tr>
<tr>
<td>rto_max</td>
<td>60s</td>
<td>0ms</td>
<td>1s</td>
</tr>
<tr>
<td>rto_min</td>
<td>1s</td>
<td>0ms</td>
<td>1s</td>
</tr>
</tbody>
</table>

**Expected Max Recovery Time**

- For OpenSS7 expect 30 ms recovery time
  - $Rto_{\text{init,min,max}} = 0$ maps to 1 jiffie
  - On Linux jiffie = 10 ms but nanosleep(1) = 20 ms, nanosleep(10) = 30 ms

Most aggressive settings possible
### Experimental Plan Status

<table>
<thead>
<tr>
<th></th>
<th>No Failure</th>
<th>Induced Packet Loss (1%)&lt;sup&gt;1&lt;/sup&gt;</th>
<th>Single Link Failure</th>
<th>Repeated Link Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TCP (SOCK_STREAM)</strong></td>
<td>Done</td>
<td>Done</td>
<td>Done</td>
<td>Done</td>
</tr>
<tr>
<td><strong>SCTP (SOCK_STREAM)</strong></td>
<td>Done</td>
<td>Done</td>
<td>Done</td>
<td>Done</td>
</tr>
<tr>
<td><strong>STCP (SOCK_SEQPACK)</strong></td>
<td>Done</td>
<td>Done</td>
<td>Done</td>
<td>Done</td>
</tr>
<tr>
<td><strong>ACE_SOCK_</strong>*</td>
<td>Done</td>
<td>Done</td>
<td>Done</td>
<td><strong>Partial</strong></td>
</tr>
<tr>
<td><em><em>ACE_SOCK_</em> (SCTP)&lt;sup&gt;2&lt;/sup&gt;</em>*</td>
<td>Done</td>
<td>Done</td>
<td>Done</td>
<td><strong>Partial</strong></td>
</tr>
<tr>
<td><strong>ACE_SOCK_SEQPACK_</strong>*</td>
<td>Done</td>
<td>Done</td>
<td>Done</td>
<td><strong>Partial</strong></td>
</tr>
<tr>
<td><strong>TAO_IlIOP</strong></td>
<td>Done</td>
<td>Done</td>
<td>Done</td>
<td><strong>Partial</strong></td>
</tr>
<tr>
<td><strong>TAO_SCIOP</strong></td>
<td>Done</td>
<td>Done</td>
<td>Done</td>
<td><strong>Partial</strong></td>
</tr>
</tbody>
</table>

<sup>1</sup> 1% packet loss is imposed on both links

<sup>2</sup> socket(AF_INET, SOCK_STREAM, IPPROTO_SCTP);

LKSCTP Tested less extensively than OpenSS7
1% Random Packet Loss Results
Summary
OpenSS7 Experimental Results
TCP vs. SCTP (SEQPACK)

Legend:
- Min
- Max
- Distribution
- Mean

~30 ms
OpenSS7 Experimental Results

ACE_SOCK_* vs. ACE_SOCK_SEQPACK_*

Caller: arachnid.external.lmco.com  Common plot elements:
two hosts/bert and ernie/ipc-frameworks/ace/5.2.3/./

~30 ms
OpenSS7 Experimental Results

TAO_IIOOP vs. TAO_SCIOP

~30 ms
Repeated Link Failure and Recover
Results Summary

Lockheed Martin Advanced Technology Laboratories
OpenSS7 Repeated Link Failures Test

2003.04.03 SCTP tests on bert and ernie, 2 interfaces active on each
Box used at setting = 6 (4 sec. down time, 16 sec. period)
assoc_max_retrans: 25 cookie_inc: 1000 heartbeat_itvl: 1 init_retries: 25 mac_type: 2
max_istreams: 33 max_sack_delay: 0 mem: 0 0 0 path_max_retrans: 0 req_ostreams: 1
rmem: 4096 87380 174760 rto_initial: 0 rto_max: 0 rto_min: 0 throttle_itvl: 50
valid_cookie_life: 60000 wmem: 4096 16384 131072 [min 3227 mean 9205 max 3.6e+04 var 3.6e+06 #]=1422

'\text{max\_output}'' using 0:1

\text{30 msec}
2003 OMG DRE Workshop

Status of SCTP Code Merge in ACE/TAO and Future Plans

- OpenSS7 support is in TAO 1.3.3 Beta Release
  - LKSCTP support will be integrated over summer 2003
- SCTP protocol properties support in progress
  - SCIOP spec pulled back from more ambitious reorganization
- SCTP in wireless, network centric environment
- SCTP and Diff Svc (particularly multiple streams)
- Automate the setting of SCTP protocol parameters based on higher level QoS requirements
  - How is this mapping done?
  - In which CCM configuration files does this end up? (.cad ?)
- SCIOP and RT-CCM (unclear about impact)
- Implications of using SCIOP with FT CORBA (active and semi-active replication, etc.)