Survivability Through Customization and Adaptability: The Cactus Approach

Matti A. Hiltunen, Richard D. Schlichting*, Carlos A. Ugarte, and Gary T. Wong

Department of Computer Science
The University of Arizona

*Current address:
AT&T Research, Florham Park, NJ

http://www.cs.arizona.edu/cactus/
Work supported in part by DARPA under grant N66001-97-C-8518.

Work done in collaboration with Gregg Townsend and current and former graduate students.
Introduction

Survivable systems:

- Complete missions in time despite failures and attacks.
- Build on security, fault tolerance, safety, etc.
- Require techniques to protect, detect, react, and recover.

Cactus: A framework for constructing configurable and adaptive distributed services and protocols.

Theme: Application of Cactus and its techniques to issues in survivability.
Fundamental techniques:
- Fine-grain customization through configurability.
- Dynamic adaptation.

Advantages:
- Customizable cost versus protection.
- Customization for scale.
- Artificial diversity through configuration.
- New survivability techniques can be easily added.
- Dynamic adaptation to attacks and changes in survivability requirements.
Presentation Outline

- The Cactus Approach.
- Cactus Survivability Mechanisms.
- Services - Current and Planned.
- Status
- Conclusions.
The Cactus Approach

CHANGED USER REQUIREMENTS

APPLICATION

CACTUS

Reliability
Timeliness
Security

Availability
Consistency
Performance

SERVICE X

SERVICE Y

OS & NETWORK

Memory

FAILURES

CPU

Network Bandwidth

CHANGES IN AVAILABLE RESOURCES

INTRUSIONS
Cactus Model

Composite/Traditional Protocol

Customizable API

Composite protocol

Messages

Shared data structures

Hash tables etc.

Micro-protocols

Reliability

TotalOrder

Privacy

Events

Msg from above

Msg from below

Site failure

Msg timeout

Event handlers

Composite/Traditional Protocol

Messages/Method invocations

QoS requests/Notifications

Customizable API
Configurability in Cactus

Configurability through independence between micro-protocols provided by the Cactus mechanisms:

Events:
- Dynamic binding of event handlers to events.
- Flexible: parameter passing, synchrony, ordering.

Shared session and protocol variables.

Cactus messages.
- Msg header = dynamic set of named msg attributes.
- Attribute scope: LOCAL, PEER, STACK.
- Coordination mechanism for sending msg to next protocol.
Adaptability in Cactus

Idea: Adaptation by dynamically reconfiguring a service.

Examples:

- FT: changing multicast algorithm to accommodate a change in failure model assumption.
- Security: increasing level of encryption to counteract an intruder.

Cactus mechanisms:

- Activation/deactivation of micro-protocols through event handler binding.
- Dynamic code loading + activation of new code.
Coordination issues:
- When to activate/deactivate a micro-protocol.
- Distributed coordination of adaptation.

Goal: smooth adaptation.

Approach: a multiphase transition from old to new micro-protocol.
Cactus Survivability Mechanisms

Fault tolerance and security

Fundamental properties for survivable systems.

Different FT and security mechanisms can be implemented as micro-protocols.

FT: retransmission, atomicity, checksums, object/process replication, message logging.


Implementation options:

1. Integration with an existing configurable service (e.g., communication or file service) or

2. Separate fault-tolerance or security service.
Artificial diversity

Harder for an intruder to apply same attack method on different installations of a system/service.

Configurability a mechanism for artificial diversity:

- Natural diversity through customization for user requirements and characteristics of the execution environment.

- Additional diversity by providing alternative micro-protocols with same service property (e.g., different encryption algorithms for privacy).
Adaptability

Survivable systems exhibit adaptive behavior:

- Automated reactions to intrusions, system state restoration.
- Service upgrades to handle new attacks.
- Dynamic change of security level when intrusion suspected.

Current work: performance and fault-tolerance adaptations.
Future work: security and real-time adaptations.

Challenges:

- Adaptation mechanisms must be intrusion tolerant (e.g., message authentication, Byzantine methods).
- Adaptation must happen in bounded time.
Transparent survivability

Legacy and off-the-shelf applications not often survivable enough
⇒ transparent enhancement of survivability necessary.

Replacement of underlying communication/OS services or
transparent insertion of middleware services:

- Linux loadable kernel modules (Cactus comm. protocols).
- Interception of signals on Linux and Solaris (Cactus DSM service).
- Smart stubs, interceptors, and DSI on CORBA allow insertion of Cactus services between an application and the ORB.
Current Cactus Services

A number of distributed services implemented to validate and demonstrate the Cactus approach.

Examples:

- RTD Channels (real-time, reliability, ordering).
- Group membership (ordering, consistency).
- Distributed shared memory (consistency, replacement, etc).
- Secure communication service.
- Group RPC.
- System monitoring service.

Focus on basic attributes rather than adaptation.
Secure Communication Service

Increasing emphasis on customizing communication security (e.g., IPSec, SSL, TLS).

SecComm: customizable secure communication service implemented using Cactus:

- Multiple basic security MPs for privacy, integrity, authenticity, non-repudiation, replay prevention, key distribution, etc.

- Meta security MPs: use basic security MPs to construct more complex protocols, e.g., alternating encryption.

- MPs simple $\Rightarrow$ easy to add custom security MPs.

- Arbitrary number and combinations of the micro-protocols allowed $\Rightarrow$ arbitrarily high security at arbitrarily high cost.
MP classes and event interactions

- Meta Security
- Basic security MPs
  - Non-Repudiation
  - Replay Prevention
  - Integrity
  - Authenticity
  - Privacy
- Security Audit
- Key Distribution

Events:
- msgFromAbove
- securityAlert
- keyMiss
- dataMsgFromBelow
- keyMsgFromBelow
SecComm performance

Implemented on a cluster of Pentium PCs (133 MHz) running MK 7.3 OS from Open Group/RI connected by a 10 Mbps Ethernet.

<table>
<thead>
<tr>
<th>micro-protocols</th>
<th>roundtrip</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>3.59 ms</td>
</tr>
<tr>
<td>XOR</td>
<td>3.82 ms</td>
</tr>
<tr>
<td>DES</td>
<td>6.75 ms</td>
</tr>
<tr>
<td>DES, XOR</td>
<td>6.96 ms</td>
</tr>
<tr>
<td>DES, XOR, Blowfish</td>
<td>8.98 ms</td>
</tr>
<tr>
<td>MD5</td>
<td>4.01 ms</td>
</tr>
<tr>
<td>SHA</td>
<td>3.99 ms</td>
</tr>
<tr>
<td>MD5, SHA</td>
<td>4.36 ms</td>
</tr>
</tbody>
</table>

Package size: 100 bytes.

Key lengths:
- DES 56,
- Blowfish 448,
- XOR 64

Average IP roundtrip time:
- 3.03 ms.
Group RPC

Replicated RPC to a group of servers.

Novel feature: customizable failure model (crash, send/receive omission, late/early timing, value, Byzantine).

Other properties: synchronous/asynchronous, FIFO/total order, atomicity.

11 micro-protocols, dozens of configurations.

<table>
<thead>
<tr>
<th>clients</th>
<th>servers</th>
<th>failure model</th>
<th>fifo</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>none</td>
<td>3.3 ms</td>
<td>3.6 ms</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>crash</td>
<td>4.2 ms</td>
<td>6.2 ms</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>send omission</td>
<td>4.1 ms</td>
<td>6.6 ms</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>rec. omission</td>
<td>5.3 ms</td>
<td>10.5 ms</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>byzantine</td>
<td>2181 ms</td>
<td>18924 ms</td>
</tr>
</tbody>
</table>
System Monitoring Service

A dynamically configurable distributed system monitor for NT and Linux implemented using Cactus/J 2.0 (Java).

Each aspect of system monitoring implemented as a separate micro-protocol:

- Users.
- Processes: CPU/memory usage, etc.
- Processor: Available memory, context switches, etc.

Micro-protocols can be loaded/unloaded at runtime.

New features could be easily added, e.g., monitoring for survivability.
Services for Survivability

Cactus mechanisms could be used to construct services specifically geared for survivability.

Intrusion detection.

- Extensible: new data collection and analysis micro-protocols.
- Customizable coverage versus performance/resource utilization/inconvenience.
- Customization for current mode of operation.
- Short term adaptation to detected threats.
Survivable data storage.

- Confidentiality: cryptography, data fragmentation.
- Integrity for intrusion detection.
- Replication for availability.
- Checkpointing, change logging for recovery.

Access control and authentication.

- Customized authentication => diversity.
- Adaptive authentication.
### Status: Prototypes and Services

<table>
<thead>
<tr>
<th>QoS Attributes</th>
<th>Example services</th>
<th>System level</th>
<th>Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security</td>
<td>SecComm</td>
<td>Application</td>
<td>DistMonServ</td>
</tr>
<tr>
<td>Fault tolerance</td>
<td>GroupRPC</td>
<td>“Middleware”</td>
<td>ConfDSM</td>
</tr>
<tr>
<td>Consistency</td>
<td>ConfDSM</td>
<td>Protocols</td>
<td>ConfCORBA</td>
</tr>
<tr>
<td>Real time</td>
<td>RTD Channel</td>
<td></td>
<td>GroupRPC</td>
</tr>
</tbody>
</table>

Colors:
- completed, released,
- in progress, planned.

<table>
<thead>
<tr>
<th></th>
<th>Solaris</th>
<th>MK</th>
<th>NT</th>
<th>Linux</th>
<th>In-kernel Linux</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cactus 1.0</td>
<td>C++</td>
<td>C</td>
<td></td>
<td>C++</td>
<td></td>
</tr>
<tr>
<td>Cactus 2.0</td>
<td>Java</td>
<td>C</td>
<td>Java</td>
<td>Java C C++</td>
<td>C</td>
</tr>
</tbody>
</table>
Conclusions

Configurability and adaptability supported by the Cactus platform important mechanisms for survivable systems.

- Customization of tradeoffs.
- Extensibility to introduce new survivability techniques.
- Adaptation to attacks.

Cactus framework and example services available through Cactus home page: http://www.cs.arizona.edu/cactus/