Survivable Network Systems

Bingrui
Hank

DCSL

Outline

- Overview of survivability
- Survivability Analysis of networked system
- Survivability through customization and adaptability : the Cactus Approach

DCSL

Overview of Survivability

- Pervasive societal dependency on network systems
  - Faults/vulnerabilities are inevitable in terms of current system sophistication level
  - Malicious attackers are around
- Survivability
  - Capacity of a system to provide essential services in the face of disruptions

DCSL

Problems behind survivability

- What are the essential services ?
  - Identify/defeine essential services
  - By user categories ? Or by business criticality ? Or ?
- What network services & computer systems are required for supporting these essential services ?
- How can a minimum essential service level be guaranteed ?
- What mechanisms need to allow the recovery following a disruption ?

DCSL

Concerns on dealing with these problems

- System complexity
- Open to multiple types of threats due to reliance on public/open infrastructures & technologies
- Non-specialist or poorly trained administrators/users

DCSL

The solutions

- The philosophy in behind :
  - Keep essential services running even during successful disruptions
  - Non-essential services are to be recovered after disruptions have been dealt with
  - Essential services may be further stratified into a number of levels
  - Scenario-dependent dynamic essential services variations

DCSL

The solutions

- Robust systems
  - Fault tolerance
  - Secure operating environments
- Secure end-to-end networking
- Design, Analysis & Integration
- System validation & verification
- Established research avenues

DCSL

Survivability Analysis of Network Specifications

Idea: Inject fault/intrusion events into a given network specification and visualize effects in the form of *scenario graphs*.

Objective: To provide valuable info to assist system architect during design phase.

DCSL

Uses a technique called *model checking*.

This technique is used for formally verifying properties about specifications of finite-state concurrent systems.

Advantage: Extremely large state-spaces can be traversed quickly.

DCSL

They model the nodes in the network as finite-state machines.

Each node has a set of input/output channels and each channel has a finite queue.

Input messages to a node are put in a queue and may cause the node to change state and produce an output message.

State transitions can be user-defined or non-deterministic.

Survivability properties are expressed in the temporal logic *Computation Tree Logic* (CTL).

DCSL

Scenario graphs

User can specify a property of the network, and model checker will produce a scenario graph that encapsulates network behaviors that lead from initial state to a state that violates the property.

Types of scenario graphs:

1. Fault scenario graph
2. Transaction success/failure graph

DCSL

Survivability analysis

Symbolic analysis:

- Assign symbolic probabilities (e.g. high, low) to events.
- Designer can ask for all scenarios that have at least 1 event with “high” likelihood of occurrence.

Reliability analysis:

- Use numeric probabilities.

DCSL

Survivability through Customization and Adaptability : The Cactus Approach

- A framework for constructing highly customizable and dynamically adaptable middleware services for networked systems.
- http://www.cs.arizona.edu/cactus/
- Use fine-grain customization & dynamic adaptation as key enabling technologies

DCSL

- Fine-grain customization
  - Customized tradeoffs between QoS attributes including performance, security, reliability, and survivability
  - Support software diversity
- Dynamic adaptation
  - Allow services to change behavior at runtime as a reaction to incoming disruptions

DCSL

The Cactus model

- A subsystem is constructed out of services
  - Ex: a communication subsystem which has two services ‘secure communication service’ and ‘system monitoring service’
- A service is seen as a composite-protocol
- Different properties and functions of a service are implemented as ‘micro-protocols’.

DCSL

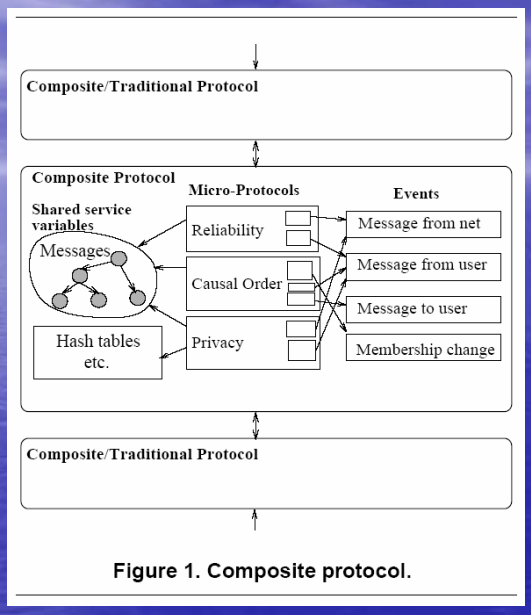

Figure 1. Composite protocol.

DCSL

Customization in Cactus

- Cactus promotes the independence between micro-protocols
- As a result, micro-protocols typically do not need to know about one another
  - Facilitate construction of customized variants of a service by configuring together the desired set of micro-protocols

DCSL

Adaptability in Cactus

- The event mechanism enables micro-protocols to be activated/deactivated by simply binding/unbinding their event handlers
- Service variables make it easy to transfer state from old to new micro-protocols
- Methods of adaptation
  - Changing the execution parameters of a micro-protocol
  - Switch the set of micro-protocols used to implement a service

DCSL

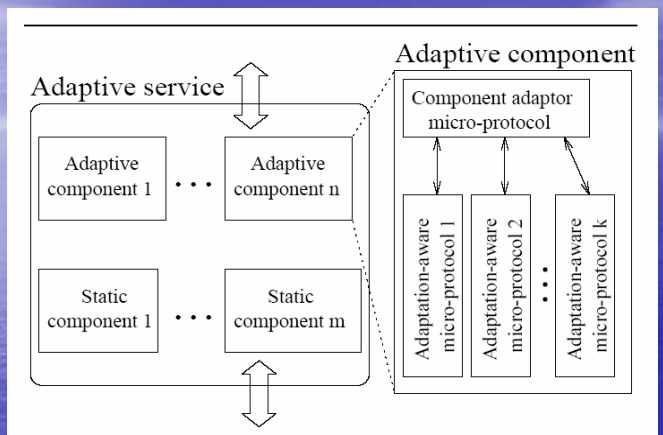

Figure 3. The structure of an adaptive service.

DCSL

Survivable Cactus Services

- Secure communication service
  - Fine-grain customization of a range of security attrubites including privacy, authenticity, message integrity, replay prevention and non-repudiation
  - Adaptability (future work) : increase the level of security when requested by user or when an intrusion attempt is detected

DCSL

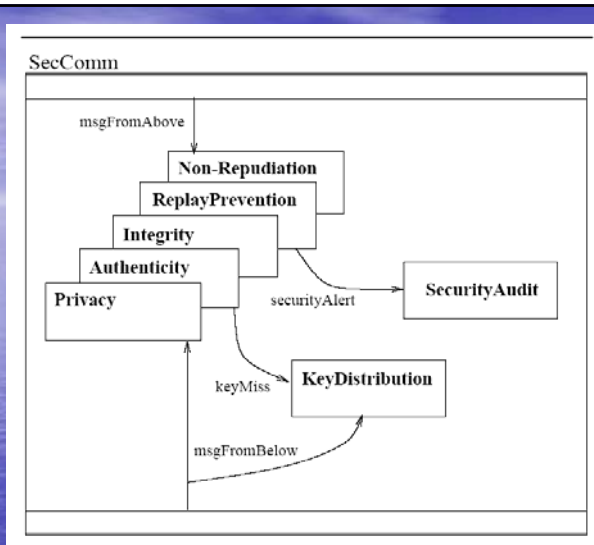

Figure 4. Micro-protocol classes and their interactions.

DCSL

- GroupRPC
  - Allows a client to execute a procedure call on one or more remote servers

Failure Model	Clients	Servers	Time
None	1	1	3.6
	2	1	5.9
Crash	1	2	6.2
	2	2	13.4
Send omission and late timing	1	2	6.6
	2	2	13.8
Receive omission and early timing	1	3	10.5
Byzantine	1	3	18924

Table 1. Average response time (in ms)

DCSL

- System monitoring service
  - A centralized global monitor that provides the user a GUI for monitoring and controllig
  - A number of local monitors executing on each of the machines being monitored
  - Each aspect of the system behavior is monitored and reported by a separate micro-protocol

DCSL

Conclusion

- Cactus
  - Fine-grained customization allows survivability versus performance trade-offs
  - Adaptability allows services to react to disruptions and facilitates their evolution when new disruptions become available
- Back to slide 4

DCSL