Tolerating Byzantine processes in distributed systems: using wormholes to reduce the number of replicas

Miguel Correia
joint work with Nuno F. Neves and Paulo Veríssimo
Faculdade de Ciências, Universidade de Lisboa
LASIGE / Navigators group

TheNIS – 2nd Information Security Workshop
IST, Lisboa, July 17th 2007

Navigators group

- Group leader: Paulo Veríssimo
- Currently 9 PhDs (6 faculty, 3 post-docs), 7 PhD students, 3 MsC students, 3 junior researchers
- Projects: 2 EC STREP (CRUTIAL, HIDENETS), 1 EC NoE (ReSIST), 1 EC CA (ESFORS), 1 ESA, 5 FCT
- CMU-PT partnership – dual degree MsC in Security and PhD in Informatics
- Research Lines
  - Fault and Intrusion Tolerance in Open Distributed Systems
  - Timeliness and Adaptation in Dependable Systems
Outline

• Intrusion Tolerance – motivation
• Hybrid system models and Wormholes
• State machine replication
• 2f+1 atomic multicast
• Consensus
• Conclusions

Intrusion Tolerance – motivation
Motivation for I-T

- Every year thousands of new vulnerabilities appear, zillions of attacks and intrusions
  - Doing the best we know/can, using security best practices etc. is not enough
- Systems with very high societal importance are becoming “online”
  - Critical infrastructures: gas, water, electr.,…
  - Controlled by computers indirectly connected to the Internet

Intrusion Tolerance

- To apply the Fault Tolerance paradigm in the domain of Security
- *Do the best we know to protect systems* (prevention)
- *…but vulnerabilities still remain…*
- *Tolerate intrusions that still occur* (tolerance)
**I-T: an example**

I-T Distributed Service

- **Redundancy**
- **Diversity**

**Servers (N)**

NFS, DNS, on-line CA, Web server, etc.

**Request**  
**Reply**

**0-Day vulnerability**

**Clients**

---

**Hybrid system models and Wormholes**
Homogeneous system models

- In Fault and Intrusion Tolerance the system model is usually homogeneous, e.g.:
  - Asynchronous (no bounds on delays)
  - Byzantine (or arbitrary) faults

Hybrid system models

- We proposed and are interested in hybrid system models. For instance:
  - Asynchronous/Byzantine as before (red) +
  - Secure wormhole (green)
Question 1: reasonable model?

- Yes, it models several current systems:
  - PCs with Trusted Platform Modules
    - [https://www.trustedcomputinggroup.org/](https://www.trustedcomputinggroup.org/)
  - PCs with SmartCards
  - DIY: PCs with virtual machines (Xen, VMWare)
  - DIY: PCs with hardware appliances

Question 2: why model?

- Why not do research about PCs + SmartCards or TPMs or…?
- Science vs. engineering; we want:
  - Expressive models of reality
  - Sound theoretical basis for proofs of correctness
  - Enablers of concepts for building new algorithms
- For practical minds: we can do things that cannot be done with SmartCards or TPMs…
  - See rest of the talk
Question 3: model what?

- Not necessarily “insecure system + secure subsystem”
- Some of us have been working with “untimely system + timely subsystem”
  A. Casimiro, P. Veríssimo, Timely Computing Base

- on hybrid models and wormholes:
  P. Veríssimo, “Travelling through Wormholes: a new look at Distributed Systems Models”
  ACM SIGACT News 2006

State machine replication
SMR basics

SMR is a mechanism to implement any deterministic service.

A server or client is said to be faulty if it deviates from its correct behaviour, e.g., because there is an intrusion or it crashes.

SMR definition

- Servers are state machines:
  - state variables, commands
- All correct servers follow the same history of states iff:
  - Initial state: all servers start in the same state
  - Agreement: all servers execute the same commands
  - Total order: all servers execute the commands in the same order
  - Determinism: the same command executed in the same initial state generates the same final state
**I-T Atomic Multicast**

- There is a maximum number $f$ of servers that can be faulty for the system to remain correct.
- With an *homogeneous system model* (asynchronous Byzantine):
  - Minimum: $N=3f+1$ servers
  - 4 to tolerate 1 faulty, 7 to tolerate 2 faulty, ...
- With a *hybrid system model* (secure wormhole in servers; not in clients):
  - Minimum: $N=2f+1$ servers
  - 3 to tolerate 1 faulty, 5 to tolerate 2 faulty, ...
  - This reduction has a huge impact on the system costs due to the need for diversity.

---

**Trusted Ordering Wormhole**

- The **TOW** is a wormhole that serves specifically to implement a $2f+1$ I-T atomic multicast.
- Provides a single service with two purposes:
  - Says *when* a message can be delivered (which is *when* $f+1$ servers have it)
  - Says the *order* in which it must be delivered
- **API:**
  - `TOW_sent` – “I sent a message”
  - `TOW_received` – “I received a message”
- **Output:**
  - `TOW_decide` – “You can deliver the message, order is $n$”
**2f+1 Atomic multicast w/TOW**

N = 3  \( f = 1 \)

H(M) – a collision-resistant hash function

![Diagram showing multicast process]

- f+1 processes have M1
- order = 1
- message delivery

---

**Performance of I-T SMR**

- In nice runs

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Latency</th>
<th>SignCP</th>
<th>VeriCP</th>
<th>Throughput</th>
<th>SignTot</th>
<th>VeriTot</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Rampart</td>
<td>6</td>
<td>3</td>
<td>2(n-f) + n</td>
<td>4(n-1)</td>
<td>(n-1)</td>
<td>(n-f)(n-f)(n-1)</td>
</tr>
<tr>
<td>2 BFT</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>2(n-1)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3 HQ</td>
<td>4</td>
<td>2</td>
<td>2(n-f)</td>
<td>2n</td>
<td>(n+1)</td>
<td>(n+1)(n-f)</td>
</tr>
<tr>
<td>4 BFT2F</td>
<td>5</td>
<td>2</td>
<td>2f</td>
<td>2n</td>
<td>(n+1)</td>
<td>n(2f+1)</td>
</tr>
<tr>
<td>5 Our alg.</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>2n</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- Bad runs

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Bad run</th>
<th>Consequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Rampart</td>
<td>Long communication delays or faulty coordinator</td>
<td>One or more coordinator elections</td>
</tr>
<tr>
<td>2 BFT</td>
<td>Same as Rampart</td>
<td>Same as Rampart</td>
</tr>
<tr>
<td>3 HQ</td>
<td>Same as Rampart/BFT if there is contention</td>
<td>Change to BFT and run BFT</td>
</tr>
<tr>
<td>4 BFT2F</td>
<td>Same as Rampart/BFT</td>
<td>Not affected (outside the wormhole)</td>
</tr>
<tr>
<td>5 Our alg.</td>
<td>Nothing (outside the wormhole)</td>
<td>Not affected (outside the wormhole)</td>
</tr>
</tbody>
</table>
Consensus

Consensus problem

• “How can some distributed processes achieve agreement on a value despite a number of them being faulty?”
  Important since related to many other distributed problems

• FLP impossibility result [Fischer et al. 85]
  Consensus is impossible to solve deterministically in a completely asynchronous system (with faults)

  For the problem to be solved, this result must be “circumvented” (i.e., system model modified): failure detectors, partial synchrony, randomization, wormholes!
Consensus and atomic multicast

• The 2 problems have been proved to be equivalent in several system models
  - Asynchronous, crash faults, failure detectors
  - Asynchronous, Byzantine, failure detectors
  - Asynchronous, Byzantine, randomization
  - ...

• What about asynchronous Byzantine with TOW?

Consensus and atomic multicast

• Two definitions of Byzantine consensus:
  - Validity 1. If all correct processes propose the same value $v$, then any correct process that decides, decides $v$.
  - Validity 2. If a correct process decides $v$, then $v$ was proposed by some process.
  - Agreement. No two correct processes decide differently.
  - Termination. Every correct process eventually decides.

• It is trivial to use the AM presented to implement consensus with Validity 2
  - Each process atomic multicast its value
  - The decision is the first value delivered

• It is simple to see that it is not possible to use the AM presented to obtain consensus with Validity 1
Conclusions

Conclusions (1)

• First solution for intrusion-tolerant state-machine replication in practical distributed systems with only $2f+1$ replicas
• Interesting impact since each additional replica has a considerable cost
• Circumvents FLP without synchrony assumptions on the asynchronous part of the system
  ➔ all synchrony is encompassed in the TOW
• Good performance:
  ➔ Low time complexity
  ➔ No asymmetric cryptography
  ➔ No leader elections
Conclusions (2)

• This work showed clear benefits of using a hybrid system model and wormholes
• Later: necessity of using wormholes (Paulo Sousa)

Questions?

• Some related publications:

• More info and papers: