

Towards Recoverable Hybrid Byzantine Consensus

Hans P. Reiser¹, Rüdiger Kapitza²

¹University of Lisboa, Portugal

²University of Erlangen-Nürnberg, Germany

September 22, 2009

Overview

1 Background

- Why? When? Where?

2 Towards Recoverable Hybrid Byzantine Consensus

- Wanted: Recovery
- Recovery in existing BFT algorithms
- Recoverable BFT: A State Model

3 Provocative Questions and Conclusions

Why?

Security threats

- NVD, Bugtraq, etc: Countless vulnerabilities
- Viruses, botnets, cyber warfare: Countless attacks

Pervasive IT systems

- Everything (incl. critical infrastructures) connected to Internet
- High security requirements no longer limited to traditional critical infrastructures

Why?

Security threats

- NVD, Bugtraq, etc: Countless vulnerabilities
- Viruses, botnets, cyber warfare: Countless attacks

Pervasive IT systems

- Everything (incl. critical infrastructures) connected to Internet
- High security requirements no longer limited to traditional critical infrastructures

Current best practices cannot avoid all faults/intrusions

New approaches are needed. Intrusion tolerance might be one key building block for more secure, more dependable systems.

When?

I do not know. Hoping for interesting discussions :-)

When?

I do not know. Hoping for interesting discussions :-)

Marketing / political issue

- How to convince people to pay for intrusion tolerance?
- Quantifying the benefit?
Intrusions harder to predict than traditional faults. . .

When?

I do not know. Hoping for interesting discussions :-)

Marketing / political issue

- How to convince people to pay for intrusion tolerance?
- Quantifying the benefit?
Intrusions harder to predict than traditional faults...

Do we still need further improvements? new research directions?

- Cheaper BFT? (Rüdiger's $f + 1$ talk)
- Missing functionality? (e.g., *node recovery*)
- ...

Where?

Where to use BFT algorithms in practice?

- Use it to build intrusion-tolerant systems
- ... wherever we will find vulnerabilities & attacks (i.e., almost everywhere)
- ... wherever we can afford the cost

Where?

Where to use BFT algorithms in practice?

- Use it to build intrusion-tolerant systems
- ... wherever we will find vulnerabilities & attacks (i.e., almost everywhere)
- ... wherever we can afford the cost

(don't forget diversity)

Where?

Where to use BFT algorithms in practice?

- Use it to build intrusion-tolerant systems
- ... wherever we will find vulnerabilities & attacks (i.e., almost everywhere)
- ... wherever we can afford the cost

*(don't forget diversity
... and determinism)*

Overview

1 Background

- Why? When? Where?

2 Towards Recoverable Hybrid Byzantine Consensus

- **Wanted: Recovery**
- Recovery in existing BFT algorithms
- Recoverable BFT: A State Model

3 Provocative Questions and Conclusions

Wanted: Recovery

Using BFT for building intrusion-tolerant systems

- Function correctly in spite of malicious intrusions
- Capability of reorganizing itself autonomously

Wanted: Recovery

Using BFT for building intrusion-tolerant systems

- Function correctly in spite of malicious intrusions
- Capability of reorganizing itself autonomously

Limitation of simple BFT algorithms

- Sooner or later, attackers might compromise more nodes than the system can tolerate
- Intrusions usually are hard to detect

Wanted: Recovery

Using BFT for building intrusion-tolerant systems

- Function correctly in spite of malicious intrusions
- Capability of reorganizing itself autonomously

Limitation of simple BFT algorithms

- Sooner or later, attackers might compromise more nodes than the system can tolerate
- Intrusions usually are hard to detect

Wanted: Proactive Recovery

Replicas should proactively be refreshed periodically, in addition to reactively repairing detected faults.

Overview

1 Background

- Why? When? Where?

2 Towards Recoverable Hybrid Byzantine Consensus

- Wanted: Recovery
- Recovery in existing BFT algorithms
- Recoverable BFT: A State Model

3 Provocative Questions and Conclusions

Recovery in existing BFT algorithms

PBFT (Castro et al.): Explicit proactive recovery support

Prerequisites for proactive recovery

- *Tamper-free device* that periodically triggers recoveries
- *Trusted component* that stores private key and creates signatures
- Means for avoiding *message replay* after recovery
- Recovering non-faulty replica *must not lose state* and
- Recovering faulty replica *must not spread wrong information*

Recovery in existing BFT algorithms

<i>Algorithm</i>	<i>TCB</i>	<i>Recovery support</i>
PBFT	yes	yes
Q/U	no	no
HQ	no	no
BFT2F	no	no
Zyzyva	no	no
A2M	yes	maybe
MinBFT	yes	no

Table: Recovery support in BFT algorithms

Implications of prerequisites

Tamper-free device that triggers recoveries

Trusted component that stores private key and creates signatures

Means for avoiding *message replay* after recovery

Recovering non-faulty replica *must not lose state*

Recovering faulty replica *must not spread wrong information*

Implications of prerequisites

Tamper-free device that triggers recoveries

- Easy: External tamper-free box

Trusted component that stores private key and creates signatures

Means for avoiding *message replay* after recovery

Recovering non-faulty replica *must not lose state*

Recovering faulty replica *must not spread wrong information*

Implications of prerequisites

Tamper-free device that triggers recoveries

- Easy: External tamper-free box

Trusted component that stores private key and creates signatures

- Easy: trusted box + minor implementation changes

Means for avoiding *message replay* after recovery

Recovering non-faulty replica *must not lose state*

Recovering faulty replica *must not spread wrong information*

Implications of prerequisites

Tamper-free device that triggers recoveries

- Easy: External tamper-free box

Trusted component that stores private key and creates signatures

- Easy: trusted box + minor implementation changes

Means for avoiding *message replay* after recovery

- Session keys, changes to message format, message filtering?

Recovering non-faulty replica *must not lose state*

Recovering faulty replica *must not spread wrong information*

Implications of prerequisites

Tamper-free device that triggers recoveries

- Easy: External tamper-free box

Trusted component that stores private key and creates signatures

- Easy: trusted box + minor implementation changes

Means for avoiding *message replay* after recovery

- Session keys, changes to message format, message filtering?

Recovering non-faulty replica *must not lose state*

- Highly intrusive: All relevant state on persistent storage

Recovering faulty replica *must not spread wrong information*

Implications of prerequisites

Tamper-free device that triggers recoveries

- Easy: External tamper-free box

Trusted component that stores private key and creates signatures

- Easy: trusted box + minor implementation changes

Means for avoiding *message replay* after recovery

- Session keys, changes to message format, message filtering?

Recovering non-faulty replica *must not lose state*

- Highly intrusive: All relevant state on persistent storage

Recovering faulty replica *must not spread wrong information*

- Most difficult part: protocol extensions (state validation)

Observations

Observations

- 1 Recovery requires a (\pm complex) trusted component
- 2 Recovery needs to be an integral part of a BFT algorithm
- 3 Recovery is not supported in most BFT algorithms

Overview

1 Background

- Why? When? Where?

2 Towards Recoverable Hybrid Byzantine Consensus

- Wanted: Recovery
- Recovery in existing BFT algorithms
- Recoverable BFT: A State Model

3 Provocative Questions and Conclusions

State Model

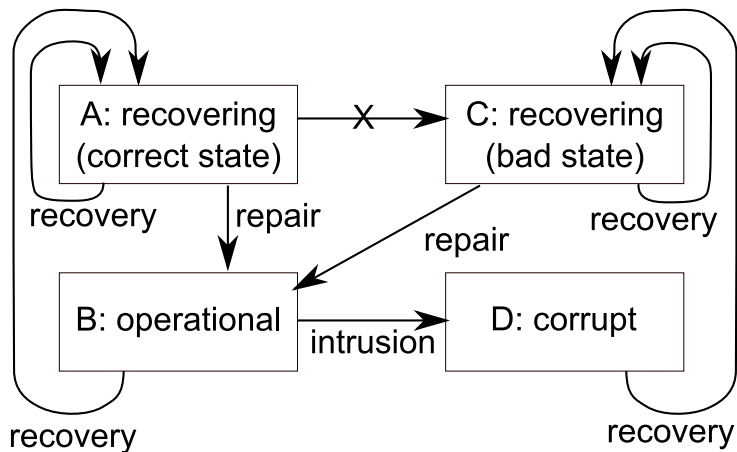
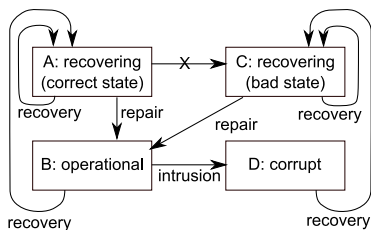


Figure: State transition model of a recoverable BFT algorithm

Recoverable BFT: Challenges



Main challenges:

- Recovery trigger ($* \Rightarrow A, * \Rightarrow C$) should be synchronous
- Recovery operation itself ($A \Rightarrow B, C \Rightarrow B$) probably executes in weaker synchrony model
- Refine system model and verify correctness

Overview

1 Background

- Why? When? Where?

2 Towards Recoverable Hybrid Byzantine Consensus

- Wanted: Recovery
- Recovery in existing BFT algorithms
- Recoverable BFT: A State Model

3 Provocative Questions and Conclusions

Questions

- 1 Is there a place for traditional $3f + 1$ BFT without proactive recovery?

Questions

- 1 Is there a place for traditional $3f + 1$ BFT without proactive recovery?
- 2 Is there a place for traditional $3f + 1$ BFT with proactive recovery?

Questions

- 1 Is there a place for traditional $3f + 1$ BFT without proactive recovery?
- 2 Is there a place for traditional $3f + 1$ BFT with proactive recovery?
 - Proactive recovery in any case requires (rather complex) TCB
 - Simple TCB enables $2f + 1$ -consensus

Questions

- 1 Is there a place for traditional $3f + 1$ BFT without proactive recovery?
- 2 Is there a place for traditional $3f + 1$ BFT with proactive recovery?
 - Proactive recovery in any case requires (rather complex) TCB
 - Simple TCB enables $2f + 1$ -consensus
- 3 Will all practical BFT implementations be $2f + 1$ -BFT with a TCB?
 - What kind of TCB (interface, functionality)?
 - System model?

Conclusions

- We need practical intrusion-tolerant systems
- Long-running systems require proactive recovery
- Most BFT papers do not consider recovery at all
- Recovery needs to be an integral part of BFT systems
- Work in progress
 - Bridge the gap: synchronous trigger vs. asynchronous network
 - Accurately define system model
 - Integrate PR into existing BFT algorithms

Thank you!

Questions?