**Notes**

**Abstract**

cyber-physical systems-> communication uncertainties

how to build the first real-time Byzantine reliable broadcast protocol (RTBRB) **tolerating network uncertainties, faults, and attacks**

**RT-ByzCast**

algorithm based on aggregating digital signatures in a sliding time-window

and on empowering processes with self-crashing capabilities to mask and bound losses

**Introduction**

cyber-physical systems (CPS)

uncertainties and faults

hamper the necessary synchronism to meet real-time deadlines

three threats

uncertainty, faults, and attacks

**problems**

real-time reliable broadcast protocols cannot tolerate attacks

existing asynchronous Byzantine solutions do not support realtime

RT-ByzCast algorithm

signed messages over a sliding time-window

algorithm triggers processes suffering such losses to crash themselves

complete shutdown?

the system size increases, the probability of a process crashing itself becomes asymptotic to 0 for any loss rate

contributions

it is impossible to implement RTBRB under traditional distributed computing paradigms

RT-ByzCast

simulation of the reliability and availability

performance evaluation

handle churn

**SYSTEM AND THREAT MODEL**

**A. System Model**

n > 1 processes Π = {p1, p2, ..., pn}

processes pi and pj are connected by links lij and lji.

d delay assumption

any message sent at the beginning of round r, if not omitted, is assumed to be received by the end of round r

**B. Threat Model**

Clocks : non-Byzantine nodes, trusted components

Processes: Byzantine, behavior

**Message** delay > d : omitted

 **referece….**

**Byzantine fault tolerance** <https://www.youtube.com/watch?v=_e4wNoTV3Gw>

<http://www.pmg.csail.mit.edu/papers/bft-tocs.pdf>

<https://people.eecs.berkeley.edu/~luca/cs174/byzantine.pdf>

Definition 1

All non-Byzantine processes that do not crash themselves,

All other processes (Byzantine and non-Byzantine that crashed themselves) are faulty.

maximum burst lengths k

**REAL-TIME BYZANTINE RELIABLE BROADCAST**

RTBRB-broadcast() and RTBRB-deliver()

**AN ALGORITHM IMPLEMENTING RTBRB**

processes monitoring each other

proof-of-life function : processes to periodically exchange heartbeats (and echo received ones) when they do not know of any broadcast

message diffusion and signature aggregation over a timewindow

processes capable of crashing themselves

A message sent by a nonByzantine process p might still fail to reach “enough” processes

R = f(k)? P3

**k**(the anticipated omission degree) R ≥ 2k + 2

**Q1**

R = f(k)? P3

**k**(the anticipated omission degree) R ≥ 2k + 2

Q2

Figure1

2f+1

**Byzantine quorums**

Project Aim:

**Protocol compare**

**Evaluation how fast, how many transitions it can handle**

Significance: Why is the project important?

**How well it can perform in practice**

Relevance:

**Distribute systems**

**How to test**

**Book, paper,**

Objectives

**Christian and maxim**

**How to compare between systems**

[**http://www.dsn.jhu.edu/pub/papers/Prime\_tdsc\_accepted.pdf**](http://www.dsn.jhu.edu/pub/papers/Prime_tdsc_accepted.pdf) **Prime reach good leader and meet deadlines. Monitoring**

**broadcast and consensus**

**software real time and byzantine resistant**

**(1) proof of life; (2) reliable broadast; (3) integrate within omnet; (4) evaluate**