Technische Universität Braunschweig

Institut für Betriebssysteme und Rechnerverbund

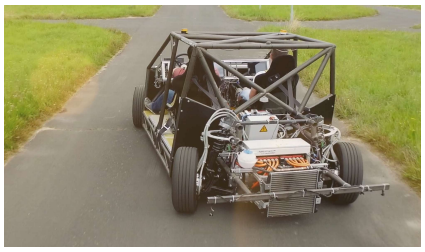
Byzantine Agreement Service for Cooperative Embedded Systems

Wenbo Xu, Martin Wegner, Lars Wolf, Rüdiger Kapitza

The 3rd International Workshop on Safety and Security of Intelligent Vehicles (SSIV) co-located with DSN, Denver, 26.June. 2017

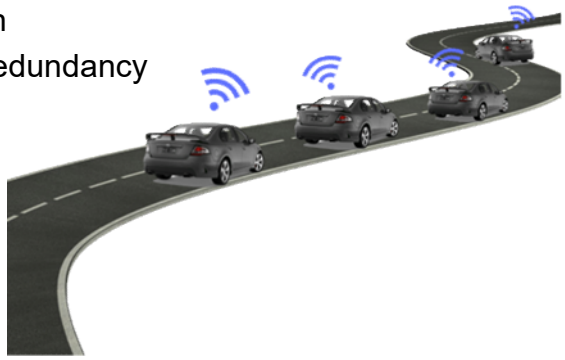
Controlling Concurrent Change

- Application domain: automotive and space
- Facilitate independent software/hardware updates on embedded system platforms
- Safety, security, availability
- www.ccc-project.org



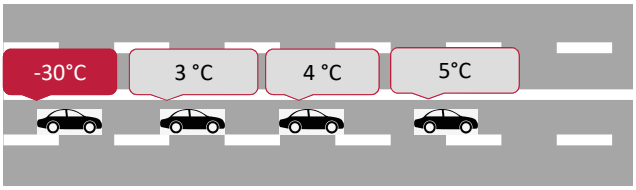
Motivation

- CCC second phase: distributed changes
- Cooperative agents coordination
  - Vehicle platooning, unmanned aerial vehicle swarms, robots etc.
  - Self-perception, self-action
- Joint view and action
- Share information, redundancy



Motivation

- Cooperation=efficiency? Redundancy=reliability?
- Nodes might behave differently
  - Objective factors: position, environment, configuration
  - Faulty behaviors: transient sw/hw error, communication failure, malicious attack



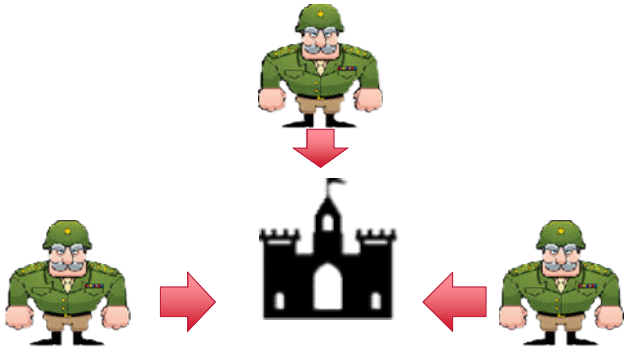
Motivation

- Popular approach: convergence based
- Less explored: distributed agreement algorithm
  - Explicit termination criterion (undecided ➔ decided)
  - Guarantee of exact consensus
  - Well tailored for small groups

Outline

- An introduction to Byzantine fault tolerance
- Making an agreement
- Trusted subsystem
- Evaluation
- Conclusion and future work

Byzantine General Problem [2]



- Honest generals and traitors
- Byzantine faults: not only crash

[2] Lamport, L.; Shostak, R.; Pease, M. (1982). "The Byzantine Generals Problem" (PDF). ACM Transactions on Programming Languages and Systems. 4 (3): 382–401.

Byzantine General Problem

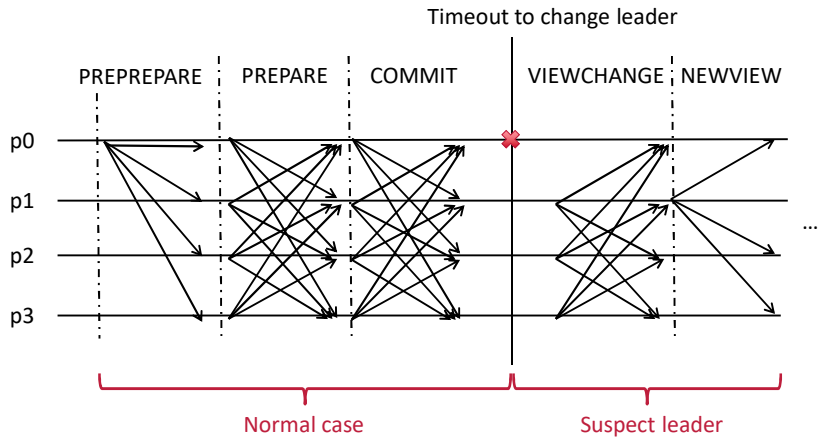
n=3
f=1

???

n≥3f+1 to tolerate f traitors

!>

PBFT Algorithm [3]



[3] Miguel Castro and Barbara Liskov. "Practical Byzantine fault tolerance and proactive recovery". In: *ACM Trans- actions on Computer Systems* 20.4 (2002), pp. 398–461.

PBFT Algorithm

- Mainly designed for state machine replication
  - Powerful machines in data center
  - Simply takes leader's proposal, seldom involves local non-deterministic values (sensor value) *

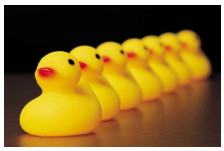
* Briefly discussed in PBFT paper

Outline

- An introduction to Byzantine fault tolerance
- Making an agreement
- Trusted subsystem
- Evaluation
- Open questions

Agree … on what?

- Follow-command: only leader proposes
  ➔Can directly use PBFT algorithm



- Consensus: everyone has its own value
  - Discrete: binary decision, leader election, etc.
  - Continuous: sensor value

Fault model: who and what can go wrong

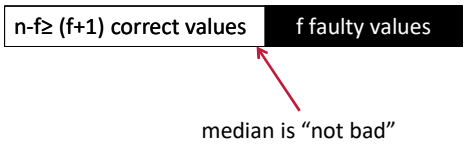
- A group of nodes (n=4 as an example)
- f<n/3 nodes can be faulty (f=1 in the example)
  - Typical Byzantine faults (crash, bit-flip, package loss, malicious behavior etc.)
  - Wrong sensor value

  ➔Even an “intended-to-be-honest” node might be fooled by its mal-functional sensor

  ➔Validity issue: cannot take just a value from single node

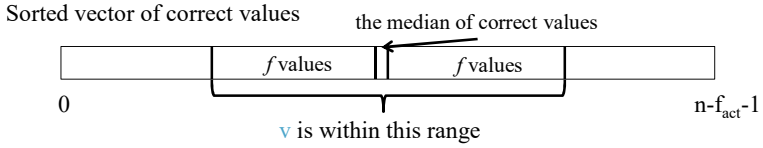
Validity of sensor values

- Value exchange before agreement
  - Broadcast own value
  - Leader collects as many values as possible (at most n-f)
  - Sort and choose the median
- When n ≥ 3f+1



Validity of sensor values

- Collect any (n-f) values, choose the median
- Median validity [8]: the chosen value v is close to the median of all correct values (unknown to algorithm)



$$Sorted\_Correct[index_{med} - f] \leq v \leq Sorted\_Correct[index_{med} + f]$$

- Already a tight bound in asynchronous system.

[8] Stolz, David, and Roger Wattenhofer. "Byzantine Agreement with Median Validity." *LIPIcs-Leibniz International Proceedings in Informatics*. Vol. 46. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

Validity of sensor values

n=4, f=1, f_{act}=1

Node	A	B	C	D
Measured temperature	3	4	5	X Byzantine!

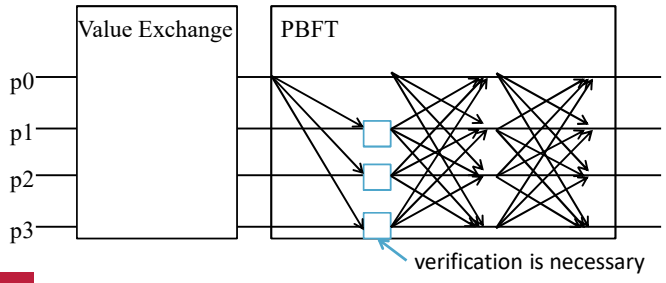
Some possible cases:

C is slow	-30	3	4
A is slow	-30	4	5
B is slow	3	5	+40
C is slow	3	3.5	4

3 ≤ valid value ≤ 5

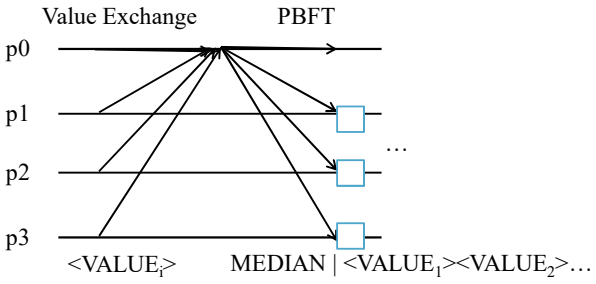
Agree on sensor value

- Value exchange phase + Agreement algorithm (PBFT)
- Requirement:
  - Leader proposes the median of (n-f) values …
  - … with a certificate, so that followers can verify



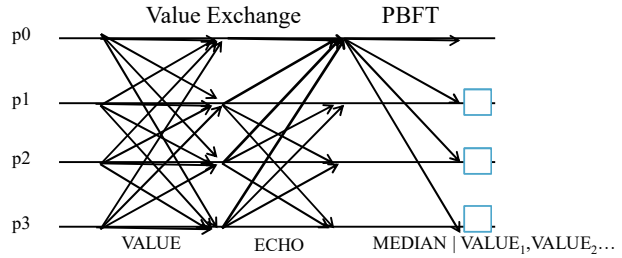
Sensor value agreement: solution 1

- With digital signature is trivial



Sensor value agreement: solution 2

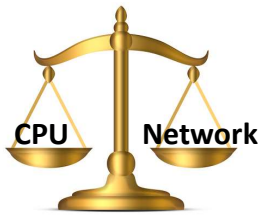
- Get rid of signature [4]:
  - Leader receives (2f+1) echoes of a value -> put it into certificate
  - Leader collects (n-f) values in certificate -> propose
  - Follower receives (f+1) corresponding echoes for every value in certificate -> accepts



[4] Milosevic, Zarko et al. "Unifying Byzantine consensus algorithms with weak interactive consistency." *International Conference On Principles Of Distributed Systems*. Springer Berlin Heidelberg, 2009.

Sensor value agreement

- The use of signature
  - ✓ One communication round less
  - ✗ Computationally expensive (for embedded systems)

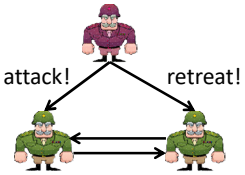


Outline

- An introduction to Byzantine fault tolerance
- Making an agreement
- Trusted subsystem
- Evaluation
- Open questions

Background: recap

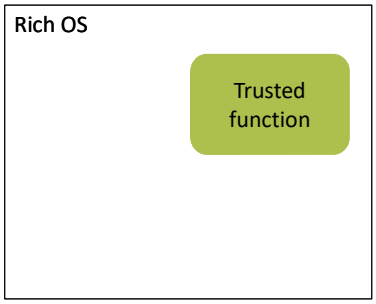
- Why 3f+1 nodes required?



- Equivocation: send different values to different recipients
- Can we restrict the ability of Byzantine nodes?

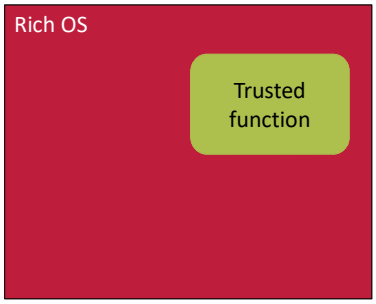
Trusted subsystem

- Assume a special part of system cannot go wrong
  - …except for crash faults
  - Hardware support in modern CPUs (ARM TrustZone, Intel SGX)



Trusted subsystem

- Assume a special part of system cannot go wrong
  - …except for crash faults
  - Hardware support in modern CPUs (ARM TrustZone, Intel SGX)



Trusted subsystem

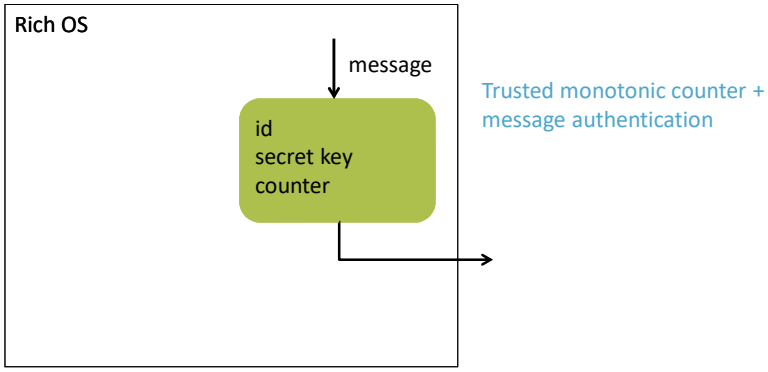
- As simple as possible
- …but powerful enough

- Trusted monotonic counter + message authentication
- Examples are MinBFT, CheapBFT, Hybster [5,6,7] etc.

[5] Giuliana Santos Veronese et al. “Efficient Byzantine Fault-Tolerance”. In: *IEEE Trans. Computers* 62.1 (2013), pp. 16–30.
[6] Rüdiger Kapitza et al. “CheapBFT: Resource-efficient Byzantine Fault Tolerance”. In: *Proceedings of the EuroSys 2012 Conference*. Ed. by European Chapter of ACM SIGOPS. Switzerland, 2012, pp. 295–308.
[7] Behl, Johannes et al. "Hybrids on Steroids: SGX-Based High Performance BFT." Proceedings of the Twelfth European Conference on Computer Systems. ACM, 2017.

Trusted subsystem

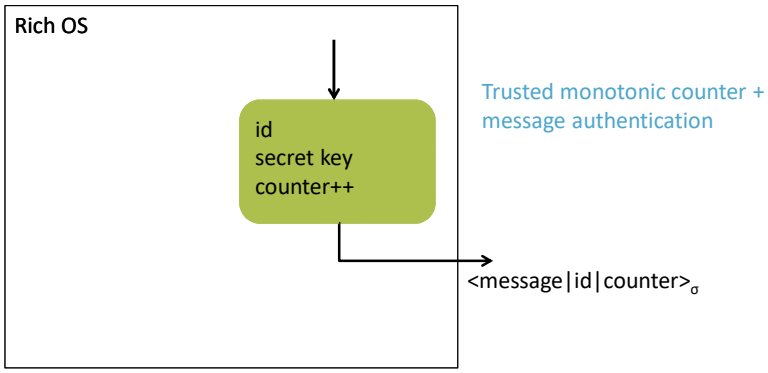
- All broadcast messages are certified by trusted counter



Trusted subsystem

- All broadcast messages are certified by trusted counter
- After certifying a message, counter increases by 1

➔Cannot use the same counter to certify two different messages

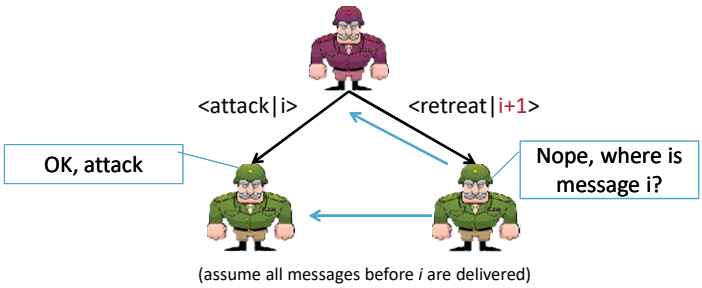


Trusted subsystem

- All outgoing messages are certified by trusted counter
- After certifying a message, counter increases by 1

➔Cannot use the same counter to certify two different messages

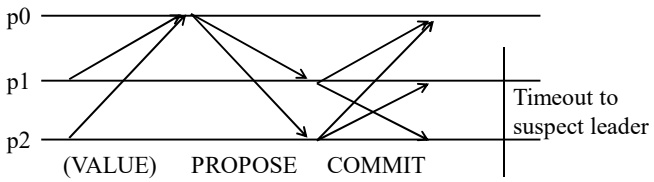
- Recipient knows the expected counter and can verify



(assume all messages before *i* are delivered)

Trusted subsystem

- Cannot equivocate anymore ➔ hybrid fault model
  - Only 2f+1 nodes needed
  - One less communication round



Trusted subsystem

- Cannot equivocate anymore ➔ hybrid fault model
  - Only 2f+1 nodes needed
  - One less communication round
- More benefit: avoid public-private key encryption

  $<message|id|counter>_{\sigma}$
  - Cannot use other's id to certificate
  - Key won't be leaked even to host OS
  - All trusted subsystems can share one secret key for broadcast

  ➔Assume static group, and keys are correctly distributed

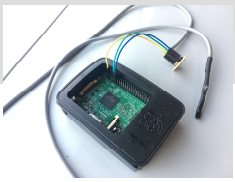
Trusted subsystem: a remark of sensor value validity

- Validity issue: required n≥3f+1
  - Still (3f+1) value providers, but only (2f+1) voters, the others are passive learners
  - Or assume known sensor permissible error

Outline

- An introduction to Byzantine fault tolerance
- Making an agreement
- Trusted subsystem
- Evaluation
- Open questions

Experiment setting



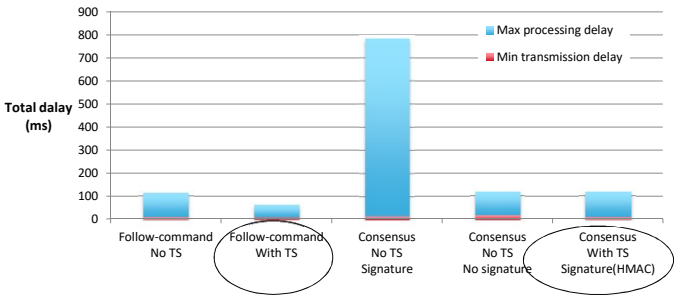
- 4 RaspberryPis
  - already support for ARM TrustZone…
  - …but as a preliminary work, we use a software simulation
  - Inject 2ms extra delay for each encryption operation [6]
- Ad-hoc wireless communication, ping ≈ 8ms
- Two agreement scenarios:
  - Follow-command: only leader proposes something
  - Consensus: everyone has its own value

  ➔Thermometer, reading time ≈ 850ms, not counted for agreement time

[6] Rüdiger Kapitza et al. "CheapBFT: Resource-efficient Byzantine Fault Tolerance". In: *Proceedings of the EuroSys 2012 Conference*. Ed. by European Chapter of ACM SIGOPS. Switzerland, 2012, pp. 295–308.

Experiment result

- Digital signature results in much higher delay
- Trusted subsystem is efficient in both scenarios



Outline

- An introduction to Byzantine fault tolerance
- Making an agreement
- Trusted subsystem
- Evaluation
- Conclusion and future work

Conclusion and future work

- Provide agreement-as-a-service for embedded systems
- Sensor value agreement with value validity
- Using trusted subsystem to decrease overhead
  - Reduce from 3f+1 to 2f+1 nodes, and one less message round
  - MAC instead of digital signature
- Next steps:
  - Implement trusted subsystem based on ARM TrustZone
  - Network stack support in target application
  - Real-time requirement
  - Compare & combine with convergence based approaches

Conclusion and future work

- Provide agreement-as-a-service for embedded systems
- Sensor value agreement with value validity
- Using trusted subsystem to decrease overhead
  - Reduce from 3f+1 to 2f+1 nodes, and one less message round

Thank you for your attention!

- Next steps:
  - More lightweight code base targeted for embedded system
  - Implement trusted subsystem based on ARM TrustZone
  - Network stack support
  - Real-time requirement
  - Comparison & combine with convergence based approaches

References

[1] W. Ren, R. W. Beard and E. M. Atkins, "Information consensus in multivehicle cooperative control," in *IEEE Control Systems*, vol. 27, no. 2, pp. 71-82, April 2007.

[2] Lamport, L.; Shostak, R.; Pease, M. (1982). "The Byzantine Generals Problem" (PDF). ACM Transactions on Programming Languages and Systems. 4 (3): 382–401.

[3] Miguel Castro and Barbara Liskov. “Practical Byzantine fault tolerance and proactive recovery”. In: *ACM Trans- actions on Computer Systems* 20.4 (2002), pp. 398–461.

[4] Milosevic, Zarko et al. "Unifying Byzantine consensus algorithms with weak interactive consistency." *International Conference On Principles Of Distributed Systems*. Springer Berlin Heidelberg, 2009.

[5] Giuliana Santos Veronese et al. “Efficient Byzantine Fault-Tolerance”. In: *IEEE Trans. Computers* 62.1 (2013), pp. 16–30.

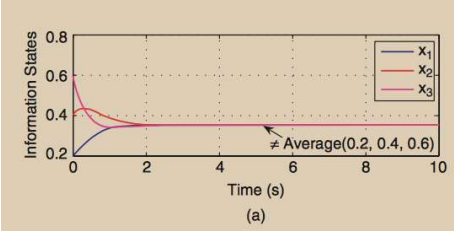
[6] Rüdiger Kapitza et al. “CheapBFT: Resource-efficient Byzantine Fault Tolerance”. In: *Proceedings of the EuroSys 2012 Conference*. Ed. by European Chapter of ACM SIGOPS. Switzerland, 2012, pp. 295–308.

[7] Behl, Johannes et al. "Hybrids on Steroids: SGX-Based High Performance BFT." Proceedings of the Twelfth European Conference on Computer Systems. ACM, 2017.

[8] Stolz, David, and Roger Wattenhofer. "Byzantine Agreement with Median Validity." *LIPIcs-Leibniz International Proceedings in Informatics*. Vol. 46. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

Motivation

- Popular approach: convergence based

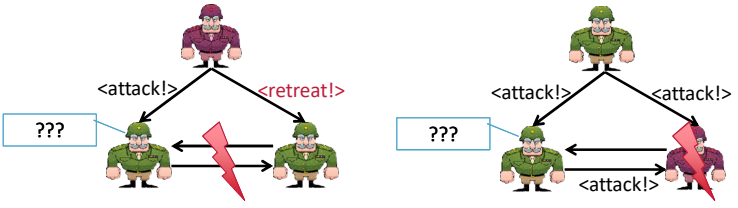


[1] W. Ren, R. W. Beard and E. M. Atkins, "Information consensus in multivehicle cooperative control," in *IEEE Control Systems*, vol. 27, no. 2, pp. 71-82, April 2007.

- Good scalability in large groups
- E.g. data fusion, speed control of platooning

Byzantine General Problems

- Asynchronous system



Byzantine fault tolerance

- Synchronous vs asynchronous system model
- Known results:
  - Synchronous system + message signature: n ≥ 2f+1 nodes
  - FLP impossibility: No consensus can be achieved in asynchronous system, even if only one node can crash.

  ➔Work around: algorithm runs without assuming synchrony, but has different guarantees under two conditions.
  - (Partially-) Asynchronous system: n ≥ 3f+1 nodes

Validity of sensor values

- Co
- Pr
- Th

n=4, f=1, f_{act}=0

Node	A	B	C	D
Measured temperature	3	4	5	6

Some possible cases:

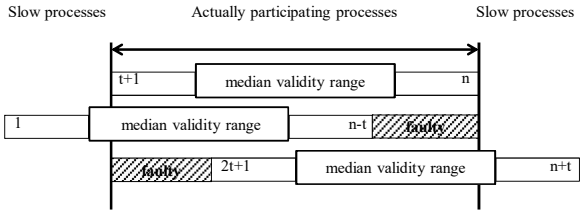
C is slow	3	4	6
A is slow	4	5	6
B is slow	3	5	6

↑

3 4 ≤ valid value ≤ 5

An explain of tight bound

- Run the same algorithm 3 times
- The same value should be decided



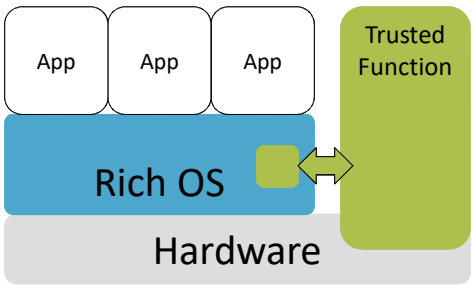
n=6, f=1

	2	3	4	5	6	x(crash)
1 (slow)	2	3	4	5	6 (faulty)	
	2 (faulty)	3	4	5	6	7 (slow)

median validity range

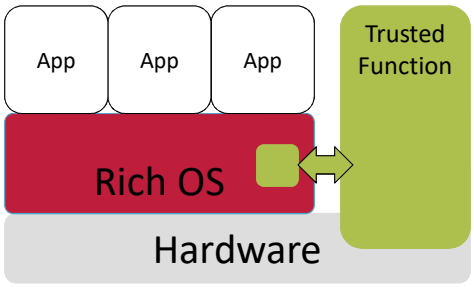
Trusted subsystem

- Assume a part of system cannot go wrong
  - …except for crash faults
  - Enabled by trusted hardware in modern CPUs (ARM TrustZone, Intel SGX)



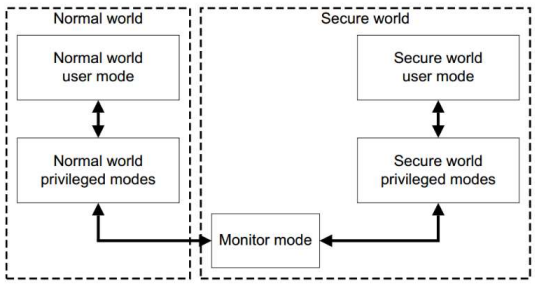
Trusted subsystem

- Assume a part of system cannot go wrong
  - …except for crash faults
  - Enabled by trusted hardware in modern CPUs (ARM TrustZone, Intel SGX)



Trusted subsystem: ARM TrustZone

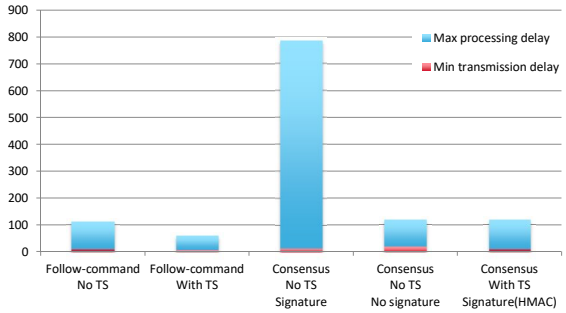
- Tight control of access to secure world (secure monitor call, hardware exceptions)
- Remote attestation: is the function correctly installed on the trusted subsystem?



ARM Security Technology: Building a Secure System using TrustZone Technology, Fig. 3-1

Experiment result

Delay(ms)	Follow-command	Consensus signature-based	Consensus signature-free
No trusted module	113	786	118
With trusted module	60	119 (HMAC)	-



Existing core OS support: Inadequate

- IP multicast just doesn't work...
  - Amazon AWS disables IPMC and tunnels over TCP
- TCP is the main option, but it has some issues:
  - No support for reliable transfer to multiple receivers
  - Uncoordinated model for breaking connections on failure
  - Byte stream model is mismatched to RDMA

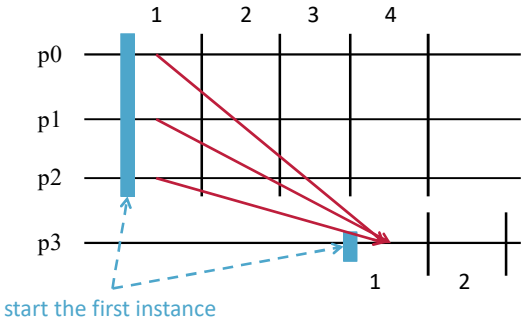
Ken Birman: Evolution of fault tolerance, SOSP History Day 2015

Timing issue

- When periodically agreeing, how to know a message belongs to the current period?
  - Replay attack; Delay attack;
- Timestampped message
  - Reasonable in vehicular communication
  - But not favorable by all people

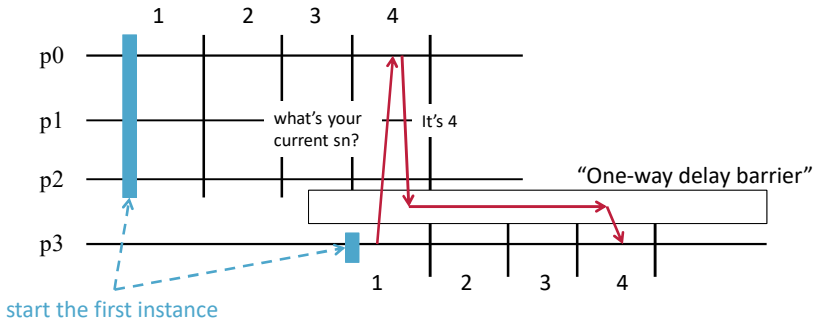
Timing issue

- Sequence number? How to determine the first period?



Timing issue

- Sequence number? How to determine the first period?
- Ask the others for a calibration?



Difference with State Machine Replication

- Limited computation resource & energy
- Communication
  - TCP/IP not suitable. New network stack support needed
  - Broadcast support
  - Network issue, handled in which layer?
- Not require high throughput. Latency more important
- ~~Checkpoint~~, ~~Ordered execution~~

Byzantine fault tolerance

- Synchronous vs asynchronous system model:
  - Synchronous system: message transmission and processing delay has a known upper bound. ➔can detect crashed node

  - Asynchronous system: no upper bound of delay

Difference with State Machine Replication: Philosophy

- SMR:
  - Require fault tolerance -> Replication -> Agreement
  - Agreement serves for fault tolerance, so itself must be FT
  - Can inherit the fault model
- Cooperative system
  - Agreement is an intrinsic service
  - Safety requirement, fault model reasonable?
  - ➢ Maybe that's why convergence approaches dominate?