Tolerating Byzantine Behavior in Distributed Systems

Miguel Correia
University of Lisboa
LASIGE / Navigators group

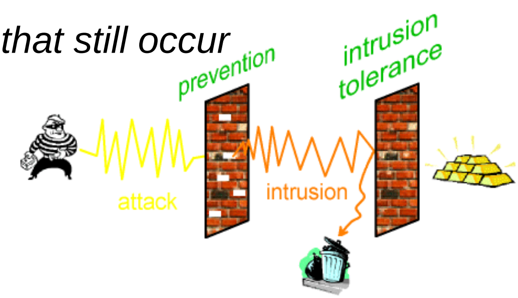
CyLab/CMU, December 2007

Motivation

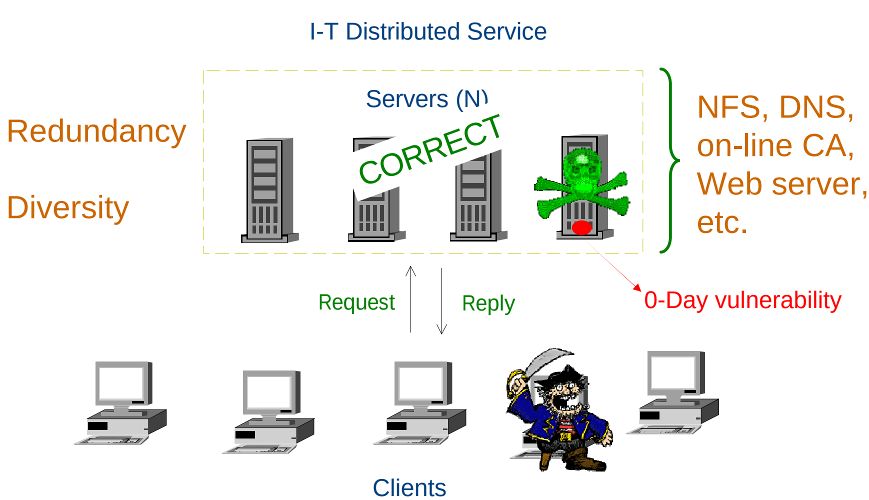
- Every year thousands of new vulnerabilities appear, zillions of attacks and intrusions happen
  - Doing the best we know/can, using security best practices etc. is essential but not enough
- Systems with high societal importance are becoming "online"
  - Critical infrastructures: gas, water, power,…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

Intrusion Tolerance

- (also called Byzantine Fault Tolerance)
- To apply the Fault Tolerance paradigm in the domain of Security
- *Do the best we know to protect systems*
- *…but vulnerabilities still remain…*
- *Tolerate intrusions that still occur*



I-T: an example



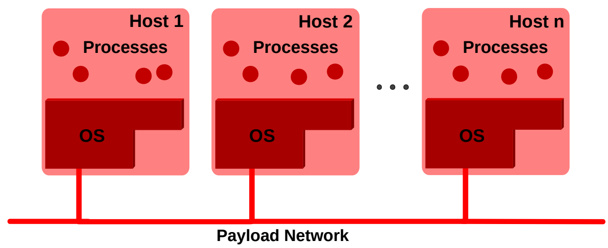
Outline

- Hybrid system models and Wormholes
- I-T State machine replication
- Randomized I-T protocols
- Primary-backup vs decentralized protocols
- Conclusions

Hybrid system models and Wormholes

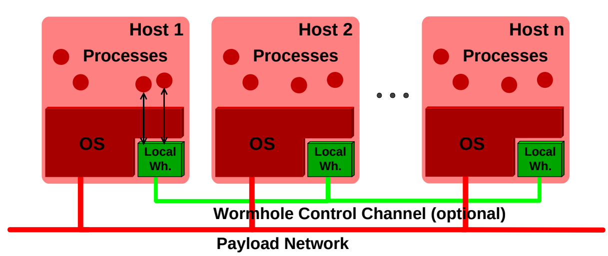
Homogeneous system models

- Most work on I-T assumes an homogeneous system model; typically:
  - Asynchronous (no bounds on delays)
  - Byzantine/arbitrary faults, including attacks/intrusions



Hybrid system models

- We proposed and are interested on *hybrid* system models. For instance:
  - Asynchronous/Byzantine as before (red) +
  - Wormhole that is secure/tamperproof (green)



Question 1: practical?

- Yes, it models several current systems:
- PCs with Trusted Platform Modules (TPM)
  - 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…?
- In our research we want:
  - *Expressive models of real systems*
  - *Sound theoretical basis for proofs of correctness*
  - *Enablers for building new algorithms*
- For practical minds:
  - We don't want to be restricted to what can be done with SmartCards or TPMs…

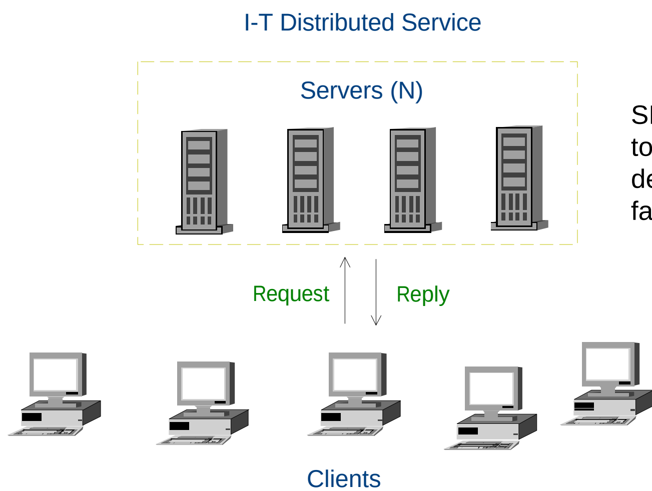
Question 3: model what?

- In this talk:
  - “*insecure system + secure subsystem*”

- But there are other possibilities, e.g.,
  - “*untimely system + timely subsystem*”
  - A. Casimiro, P. Veríssimo, Timely Computing Base

I-T State machine replication

State machine replication basics



SMR is a mechanism to make <u>any</u> deterministic service fault-tolerant

SMR definition

- Servers are state machines:
  - state variables, commands
- Basic idea: to make all servers follow the same sequence of states, i.e., enforce:
  - *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

Atomic multicast protocol

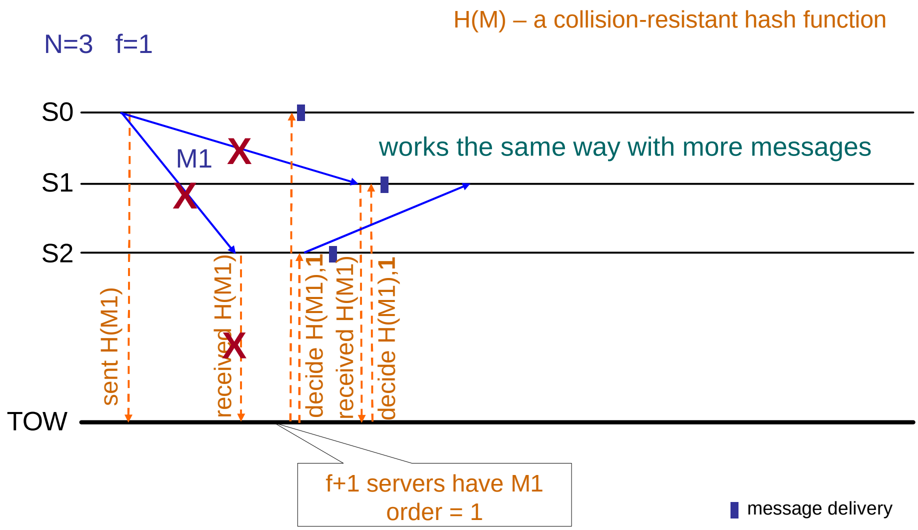
Main Contribution

- 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 servers 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 cost: hw, sw, admin (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



Performance of I-T SMR

- Nice runs

		LATENCY			THROUGHPUT		
	Algorithm	**ComSteps**	**SignCP**	**VerifCP**	**MesgTot**	**SignTot**	**VerifTot**
1	Rampart	8	3	$2(n-f)+n$	$4n \oplus 3(n-1)$	$n \oplus (n-1)$	$(n-f)n \oplus (n-f)(n-1)$
2	BFT	5	0	0	$2n \oplus (n-1)(2n-1)$	0	0
3	HQ	4	2	$2(n-f)$	$4n$	$(n+1)$	$(n+1)(n-f)$
4	BFT2F	5	2	$2f$	$2n \oplus (n-1)(2n-1)$	$(n+1) \oplus 0$	$n(2f+1) \oplus 0$
5	Our alg.	5	0	0	$2n\,[+(n^3+n^2-n)]$	0	0

- Bad runs

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

I-T SMR Research trends

- BFT – Castro and Liskov (OSDI 99)
  - First efficient I-T SMR system
- Increasing speed:
  - FaB Paxos (DSN’05), Q/U (SOSP’05), HQ (OSDI’06), Zyzzyva (SOSP’07)
- Reducing window of vulnerability:
  - BFT-PR (TOCS’02), Sousa et al. (SAC’06)
- Reducing number of replicas:
  - this work (SRDS’04), BFT2F (NSDI’07), A2M-PBFT-EA (SOSP’07)

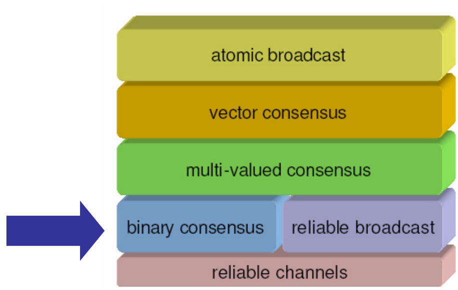
Randomized I-T protocols

Motivation

- Randomized Byzantine FT agreement protocols:
  - Introduced in 1983: Ben-Or (PODC), Rabin (FOCS)
  - Since then many others appeared…
- But from a practical point of view:
  - Ben-Or style protocols ("local coins") → run in an exponential expected number of communication steps
  - Rabin style protocols ("shared coin") → rely on public-key crypto
- DS folklore: work in the area is theoretical; protocols too slow for most applications…
- …but are they really slow?

RITAS

- First, we designed an arguably efficient stack of randomized I-T protocols, RITAS (no wormhole)
  - No signatures, asynchronous, decentralized, n=3f+1



- Then implemented and evaluated their performance…
  - LANs, PlanetLab, wireless (PCs and PDAs)

Local coins vs Shared coin

- Binary consensus protocols evaluated:
  - Bracha's (84), expected n. rounds *O(2^n-f^)*, no crypto
  - ABBA (01), expected n. rounds constant, public-key crypto
- Testbed
  - 10/100/1000 Mbps local-area network (LAN)
  - 11 Dell PowerEdge 850 computers (2.8 GHz, 2 GB RAM)
  - Linux 2.6.11

Shared Coin has always much higher latency

Latency (µs) [1000 Mbps, no faults]

Proposal Distribution		Machines (n): 4	7	10
Uniform	Local	824	2187	4132
	Shared	21590	31315	43633
Corrosive	Local	2453	6172	12075
	Shared	33834	38529	55169
Random	Local	2056	5812	11501
	Shared	24320	36325	49206

Shared Coin is not affected by Byzantine faults

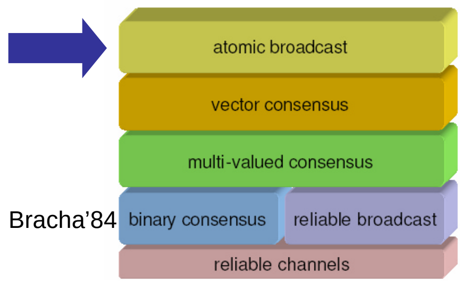
Maximum Throughput (decisions/s)				
		Machines (n)		
Faultload		4	7	10
Failure-free	Local	450	170	80
	Shared	13	9	8
Crash	Local	600	225	110
	Shared	31	25	20
Byzantine	Local	330	87	30
	Shared	16	9	8

Shared Coin is more robust with the Byzantine faultload

Number of Rounds until Decision				
		Machines (n)		
Faultload		4	7	10
Failure-free	Local	1.004	1.005	1.009
	Shared	1.013	1.018	1.010
Crash	Local	1.000	1.000	1
	Shared	1.000	1.000	1.
Byzantine	Local	1.462	1.569	2.289
	Shared	1.016	1.017	1.012

Theoretical expected result is 128 rounds

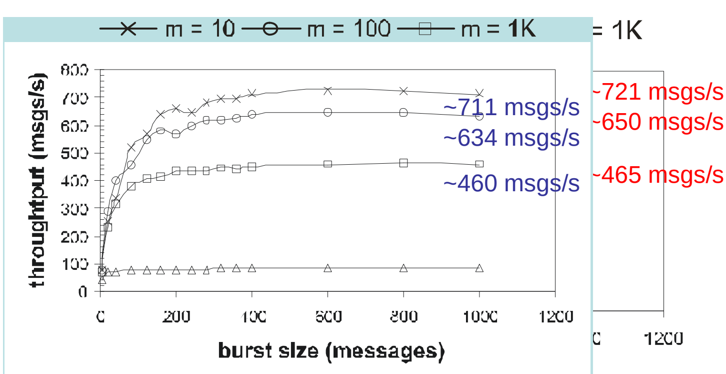
Randomized Atomic Broadcast



- Is it fast/practical?
- Testbed
  - 100 Mbps LAN
  - 4 nodes (Pentium III PCs, 500 MHz, 128 MB RAM)
  - Linux (kernel version 2.6.15)

Throughput

- Byzantine faults – throughput almost not affected



Primary-based vs decentralized protocols

Faster RITAS?

- We wanted RITAS to be faster; best candidate for improvement: Binary Consensus (bottom)
  - Fastest RITAS's BC (Bracha 84): decentralized, n=3f+1, ***$O(n^3)$*** message complexity, no signatures
- Decentralized algorithms that solve asynchronous Byzantine BC can be build with and only with:
  1. More Processes: n = 5f+1, $O(n^2)$ message complexity and no signatures
  2. More Messages: n = 3f+1, $O(o)$ message complexity ($n^2 < o = n^2 f$) and no signatures
  3. Signatures: n = 3f+1, $O(n^2)$ message complexity and using signatures
- To improve RITAS, option 2, message complex. $O(n^2 f)$

State machine replication revisited

- For <u>decentralized</u> consensus algorithms, best:
  - n = 3f+1, *O(o)* message complexity ($n^2 < o = n^2f$), no signatures
- But for a <u>primary-based</u> SMR like BFT:
  - n = 3f+1, $O(n^2)$ message complexity, no signatures
- SMR with n=2f+1:
  - Requires distributed “heavy” wormhole
  - *Decentralized* (but not randomized)
- What about a *primary-based* SMR?
  - n=2f+1 ? “Lighter” wormhole?

Conclusions

Conclusions (1)

- Intrusion tolerance: a new paradigm for more secure distributed systems
- Hybrid system models and Wormholes
  - Model reality as sound basis for proofs of correctness
  - Enablers for building new algorithms…
  - … without getting tied to current devices
- First solution for I-T state-machine replication with only **2f+1** replicas

Conclusions (2)

- Randomized I-T protocols
  - Experimentation contradicted DS folklore
  - Protocols are practical
  - Local coin protocols are fast/practical but scale worse than shared-coin protocols
- Primary-based vs decentralized protocols
  - Primary-based have to recover from faulty leader
  - But decentralized protocols have constraints that do not apply to primary-based

Thank you. Questions?

http://www.di.fc.ul.pt/~mpc/
http://www.navigators.di.fc.ul.pt/

- Some related publications:

  - M Correia, NF Neves, P Veríssimo. How to Tolerate Half Less One Byzantine Nodes in Practical Distributed Systems. IEEE *SRDS* 2004
  - N F Neves, M Correia, P Veríssimo. Solving Vector Consensus with a Wormhole. IEEE TPDS 16-12, Dec. 2005
  - M Correia, N F Neves, L C Lung, P Veríssimo. Low Complexity Byzantine-Resilient Consensus. Distributed Computing, 17-3 Mar. 2005
  - P Veríssimo, Travelling through Wormholes: a new look at Distributed Systems Models. SIGACT News 37-1, 2006
  - M Correia, N F Neves, P Veríssimo. From Consensus to Atomic Broadcast: Time-Free Byzantine-Resistant Protocols without Signatures. Computer Journal 41-1, Jan. 2006
  - H Moniz and N F Neves and M Correia and P Veríssimo. Randomized Intrusion-Tolerant Asynchronous Services. *DSN* 2006
  - A Bessani, M. Correia, H Moniz, N F Neves, P Verissimo. When 3 f +1 is not Enough: Tradeoffs for Decentralized Asynchronous Byzantine Consensus. *DISC* 2007