The Byzantine Generals Problem
Leslie Lamport, Robert Shostak, and Marshall Pease
ACM TOPLAS 1982

Practical Byzantine Fault Tolerance
Miguel Castro and Barbara Liskov
OSDI 1999

Announcements

- Dave is at Hotnets
  - TA (James Hendricks) gave lecture
- Outline requested by 11/30
  - Not graded, but less sympathy if you skip. Goals: (1) Ensure you pass, (2) help you cut down on amount of effort spent
  - Feel free to give James drafts of writeup at any time

A definition

- Byzantine (www.m-w.com):

  1: of, relating to, or characteristic of the ancient city of Byzantium

  …

  4b: intricately involved : labyrinthine <rules of Byzantine complexity>

- Lamport's reason:

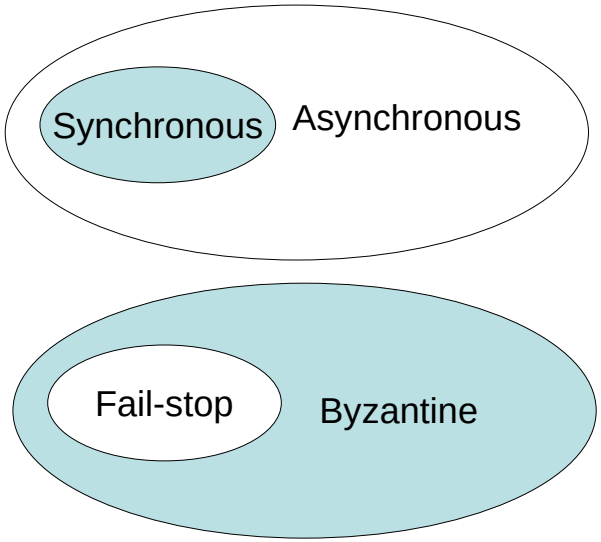
  "I have long felt that, because it was posed as a cute problem about philosophers seated around a table, Dijkstra's dining philosopher's problem received much more attention than it deserves."

  (http://research.microsoft.com/users/lamport/pubs/pubs.html#byz)

Byzantine Generals Problem

- Concerned with (binary) atomic broadcast
  - All correct nodes receive same value
  - If broadcaster correct, correct nodes receive broadcasted value
- Can use broadcast to build consensus protocols (aka, agreement)
  - Consensus: think Byzantine fault-tolerant (BFT) Paxos

Synchronous, Byzantine world



Cool note

Example Byzantine fault-tolerant system:

⇒ Seawolf submarine's control system

Sims, J. T. 1997. *Redundancy Management Software Services for Seawolf Ship Control System*. In Proceedings of the 27th international Symposium on Fault-Tolerant Computing (FTCS '97) (June 25 - 27, 1997). FTCS. IEEE Computer Society, Washington, DC, 390.

But it remains to be seen if commodity distributed systems are willing to pay to have so many replicas in a system

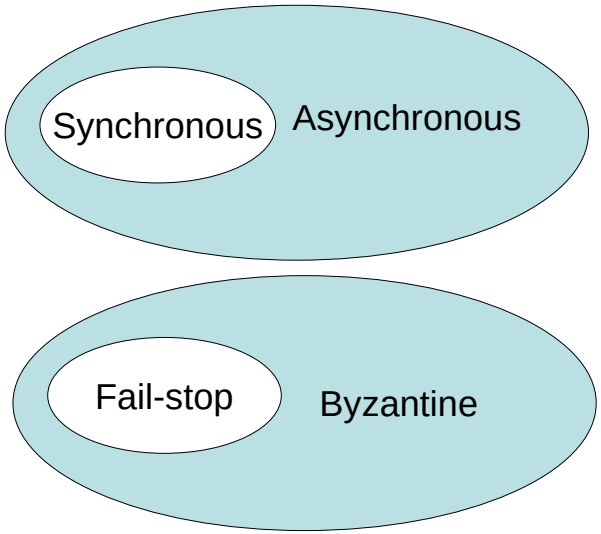
First protocol: no crypto

- Secure point-to-point links, but no crypto allowed
- Protocol OM(m): Recursive, exponential, all-to-all
  - [Try to sketch protocol – see page 388]
  - May be inefficient, but shows 3f+1 bound is tight
  - [Discuss: Understand that this is for synchronous setup without crypto!]
- Need at least 3f+1 to tolerate f faulty!
  - See figures 1 and 2
  - How to fix? Signatures (for example). Or hash commitments, one-time signatures, etc.

Second protocol: With crypto

- Protocol SM(m)
  - [Page 391, but can skip protocol]
  - Given signatures, do m rounds of signing what you think was said. Many messages (don't need as many in absence of faults).
  - Shows possible for **any** # of faults tolerated
  - [Discuss. Understand: Synchronous, lots of messages, but possible.]
- [Skip odd topologies. Note that "signature" can be emulated for random (not malicious) faults.]

Practical Byzantine Fault Tolerance: Asynchronous, Byzantine



Practical Byzantine Fault Tolerance

- Why async BFT? BFT:
  - Malicious attacks, software errors
  - Need N-version programming?
  - Faulty client can write garbage data, but can't make system *inconsistent* (violate operational semantics)
- Why async?
  - Faulty network can violate timing assumptions
  - But can also prevent liveness

[For different liveness properties, see, e.g., Cachin, C., Kursawe, K., and Shoup, V. 2000. Random oracles in constantipole: practical asynchronous Byzantine agreement using cryptography (extended abstract). In *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing* (Portland, Oregon, United States, July 16 - 19, 2000). PODC '00. ACM, New York, NY, 123-132.]

Distributed systems

- Async BFT consensus: Need 3f+1 nodes
  - Sketch of proof: Divide 3f nodes into three groups of f, left, middle, right, where middle f are faulty. When left+middle talk, they must reach consensus (right may be crashed). Same for right+middle. Faulty middle can steer partitions to different values!

[See Bracha, G. and Toueg, S. 1985. Asynchronous consensus and broadcast protocols. J. ACM 32, 4 (Oct. 1985), 824-840.]

- FLP impossibility: Async consensus may not terminate
  - Sketch of proof: System starts in "bivalent" state (may decide 0 or 1). At some point, the system is one message away from deciding on 0 or 1. If that message is delayed, another message may move the system away from deciding.
  - Holds even when servers can only crash (not Byzantine)!
  - Hence, protocol cannot always be live (but there exist randomized BFT variants that are probably live)

[See Fischer, M. J., Lynch, N. A., and Paterson, M. S. 1985. Impossibility of distributed consensus with one faulty process. J. ACM 32, 2 (Apr. 1985), 374-382.]

Aside: Linearizability

- Linearizability ("safety" condition) -- two goals:
  - Valid sequential history
  - If completion of E1 precedes invocation of E2 in reality, E1 must precede E2 in history
- Why it's nice: Can reason about distributed system using sequential specification
- [Can give example on board if time]

[See Herlihy, M. P. and Wing, J. M. 1990. Linearizability: a correctness condition for concurrent objects. ACM Trans. Program. Lang. Syst. 12, 3 (Jul. 1990), 463-492.]

Cryptography

- Hash (aka, message digest):
  - Pre-image resistant: given hash(x), hard to find x
  - Second pre-image resistant: given x, hard to find y such that hash(x) = hash(y)
  - Collision resistant: hard to find x,y such that hash(x) = hash(y)
  - Random oracle: hash should be a random map (no structure)
  - Assembly SHA1 on 3 GHZ Pentium D -- ~250 MB/s
  - Brian Gladman, AMD64, SHA1: 9.7, SHA512:13.4 (cycles/byte)
- MACs: ~1 microsecond, > 400 MB/s (700 MB/s?)
- Signatures: ~150 microseconds – 4 milliseconds
  - 150 microseconds: ESIGN (Nippon Telegraph and Telephone)
  - (Compare to Castro's 45 millisecond on PPro 200)

Castro&Liskov use 128-bit AdHash for checkpoints. Broken by Wagner in 2002 for keys less than 1600 bits.Moral of the story: Beware new crypto primitives unless they reduce to older, more trusted primitives!

Basic protocol

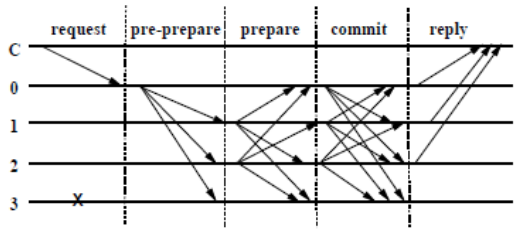


Figure 1: Normal Case Operation

Plus view change protocol, checkpoint protocol.

Question: When is this not live?
Answer: During successive primary timeouts.
(Compare to Q/U)

Recent systems

[Show network patterns]

Q/U: 5f+1, 1 roundtrip (SOSP 2005)

H/Q: 3f+1, 2 roundtrips (OSDI 2006)

Zyzzyva: 3f+1, 3 one-way latencies but need 3f+1 responsive

Evaluation

- Only implemented parts of protocol that mattered for evaluation/analysis
  - Hint: Not a bad idea for 712 projects!
- NFS loopback trick is pretty standard -- good idea for prototyping

- Tolerate 1 fault, use multicast.
- BFT prototype: no disk writes
  - NFS server: disk writes for some operations!
  - Explanation: Replication provides redundancy
  - Is this a fair comparison? How about BFS vs replicated NFS?