8. The Trouble with Distributed Systems

- Faults and Partial Failures
  - partial failure
    - nondeterministic
  - Cloud Computing and Supercomputing
    - high-performance computing (HPC)
    - cloud computing
      - ⚠️ thousands of nodes => something is always broken
      - 👍 we need to build a reliable system from unreliable components
- Unreliable Networks
  - ⚠️ you can't understand why you didn't get response
    - 👍 The usual way of handling this issue is a timeout
  - Network Faults in Practice
    - Network partitions
      - or netsplit
        - network fault
    - Not necessarily need to tolerate it
      - Just showing error to user can be a good strategy
  - Detecting Faults
  - Timeouts and Unbounded Delays
    - ⚠️ Hard question: how long to wait?
    - unbounded delays
      - No upper limits for delays
    - Network congestion and queueing
      - A lot of queues everywhere
      - TCP Versus UDP
        - UDP does not perform flow control and does not retransmit lost packets
        - 👍 UDP is good when delayed data is worthless
      - 👍 Good practice to define timeouts dynamically considering round trips and some variabilities (jitter)
        - And TCP is doing this already. And Cassandra and Akka
  - Synchronous vs Asynchronous Networks
    - Sync networks such as ISDN reserve 16 bits for every established call in every packet which is sent every 250microseconds
      - 1 circuit-switched network
        - no queues
        - bounded delay
    - Can we not simply make network delays predictable?
      - Ethernet and IP
        - 2 packet-switched protocols
          - Queues
          - Unbounded delays
          - Better for bursty traffic
      - 3 There were attempts to use hybrid approach
        - Asynchronous Transfer Mode (ATM)
        - InfiniBand
      - 👍 Emulating hybrid over TCP
        - quality of service (QoS, prioritization and scheduling of packets)
        - admission control (rate-limiting senders)
        - Latency and Resource Utilization
          - Latency can be guaranteed in case of static partitioning of resources. Which reduces utilisation
- Unreliable Clocks
  - Monotonic Versus Time-of-Day Clocks
    - Time-of-day clocks
      - Aka "wall-clock time"
      - Returns number of seconds from epoch (1-Jan-1970) not counting leap seconds
      - May be jumped back during sync with NTP server
      - 👎 Unsuitable for measuring elapsed time!
    - Monotonic clocks
      - 👍 Difference is good
      - ⚠️ Absolute value is meaningless!
        - Can be time from start of computer
        - No sense to compare times from two computers
      - NTP can ask clock rate to speedup or slow down by up to 0.05%
        - But cannot jump it forward or backward
  - Clock Synchronization and Accuracy
    - Can depend on temperature
    - Drifts can be = 17 seconds / day
  - Relying on Synchronized Clocks
    - Timestamps for ordering events
      - Last write wins (LWW) strategy may rely on timestamps and cause problem - not applying later changes in multi-leader environment
      - ❓ Q: What to use for Logs then?
    - Clock readings have a confidence interval
    - Synchronized clocks for global snapshots
  - Process Pauses
    - Response time guarantees
      - hard real-time systems
        - For controlling aircrafts, cars, etc
      - Is real-time really real?
      - real-time operating system (RTOS)
    - Limiting the impact of garbage collection
      - 👍 Emerging idea - runtime to tell that we need to GC soon and stop directing requests to this node - so it can GC w/o impacting users
      - 👍 Regular restart of nodes can also limit needs in GC for long-lived objects
- Knowledge, Truth, and Lies
  - 👍 The Truth Is Defined by the Majority
    - The leader and the lock
      - The leader or lock may no longer be valid / can be expired
    - Fencing tokens
  - ⚠️ Byzantine Faults
    - When you cannot trust nodes - i.e. node can lie
    - The Byzantine Generals Problem
    - Byzantine fault-tolerant
      - If system can operate in such environment
      - Examples
        - Aerospace environment
          - Systems can be damaged by radiation
        - Multiple organisations environment
          - e.g. Bitcoin
      - Weak forms of lying
        - Checksums
        - Validate user inputs
        - Multiple NTP servers
  - System Model and Reality
    - Timing assumption models
      - Synchronous model
        - Bounded delays, pauses, clock errors
      - Partially synchronous model
        - Most of the time bounded, but not always
        - 👍 Most useful model
      - Asynchronous model
        - No clocks and any timing assumptions
    - Node failures models
      - Crash-stop faults
        - Node can crash and it will gone forever
      - Crash-recovery faults
        - Node after crash may recover after some time
        - 👍 Most useful model
      - Byzantine (arbitrary) faults
    - Correctness of an algorithm
      - Properties
        - Uniqueness
          - 👍 Each request for fencing token returns new one
        - Monotonic sequence
          - 👍 Increasing of fence tokens
        - Availability
          - Node will eventually get response with fencing token
    - Safety and Liveness
      - Safety
        - nothing bad happens
      - Liveness
        - Often includes word "eventually"
        - something good eventually happens
    - Mapping system models to the real world