Lecture 11

Friday, 13 October 2023 08:35

important concepts from last time:

- coupling
- availability
- cohesion
- quality attribute
- reliability

fault tolerance is a quality attribute of a system that describes its ability to deal with faults
graceful degradation of system

fault tolerance is preventing faults from becoming (system) failures

failure: not meeting its specification

error: system state that may lead to failure

fault: cause of an error

faults can originate from internally (within the system) or externally (environment)

faults may have several reasons
- accidental
- malicious

fault duration can be
- transient → difficult to identify, since it does not always appear in particular, when debugging
- intermittent
- permanent

fault/failure models

fault metric: for measuring fault tolerance

a t-resilient system can tolerate up to and including t faulty components

things which can go wrong in interacting processes:

failure types	
crash	server halts
omission	server fails to respond/receive/send
timing	response outside specified time interval
response	server response (value/control flow) incorrect
arbitrary ⇒ i.e. byzantine	arbitrary response at arbitrary time

halting types:

fail-stop	crash failures, reliably detectable
fail-noisy	crash failures, eventually reliably detectable
fail-silent	omission/crash failures: client cannot tell what went wrong
fail-safe	arbitrary, yet benign failures → cannot do any harm
fail-arbitrary	arbitrary, with malicious failures

message loss = lack of delivery

fair-loss delivery: if correct process p infinitely often sends a message m to correct process q, then q delivers m an ∞ number of times

reliable delivery: if correct process p sends m to correct process q, then q eventually delivers m

message duplication

finite duplication: if correct p sends m finitely many times, correct q
q cannot deliver m infinitely

stubborn delivery: if correct p sends m once to correct q, q delivers m an ∞ number of times

no duplication: no message m is delivered more than once

message creation

no creation: if q delivers m with sender p, then m was previously sent to q by p

authenticity: if q and p are correct, and m is delivered to q with sender p, then p previously sent m to q.

UDP guarantees FLD
TCP guarantees RD

a unicast service is reliable if it satisfies the requirements for a perfect link: RD + ND + NC
- RD → validity: any message is eventually delivered
- ND + NC → integrity: message identical to sent message, no duplication

process resilience:
- flat group: all nodes run same process
- hierarchical group: coordinator + workers (could be server + clients)

consensus: each correct process executes the same commands in the same order

a group communication (multicast) is reliable if it satisfies
- validity: if correct process multicasts m, it will eventually deliver m
- integrity: message is delivered at most once, and if delivered, sender belongs to group of m and m was previously sent multicast by s
- agreement: if correct process delivers m, all correct processes will eventually deliver m

consensus problem: set of correct processes make the same decision i.e. have agreement

consensus guarantees:
- termination: every correct process eventually decides some value
- validity: if process decides v, then v was proposed by some process
- integrity: no process decides twice
- either of
  - agreement: no two correct processes decide differently
  - uniform agreement: no two processes decide differently

fault tolerance tactics

- fault masking
  - error detection
    - acceptance test
    - exceptions
    - watchdog timer
    - ping/echo
    - heartbeat
  - error diagnosis
  - error/fault containment
  - error recovery
    - forward error recovery: find new error-free state to continue — predictable in terms of time and memory ⇒ applicable in real-time systems
    - backward error recovery: restore previous error-free state — e.g. checkpointing, suitable for handling transient failures

- redundancy
  - hot spare: updates are synchronous with primary
  - warm spare: spares are updated periodically
  - cold spare: off-line, not updated, needs to be brought online
  - for failure masking
    - add extra bits — error correction
    - time — repeat failing operations
    - physical redundancy — multiple hardware/software components
  - active: all replicas performing calculations (updates) in parallel
  - passive: replica only activated if previous replica fails

fault prevention:
- sound development methodologies — coding standards, test procedures, design patterns
- temporary removal from service — e.g. occasional restart
- transactions: ACID
  - atomic — commit/abort & rollback, indivisible
  - consistent — maintain system invariants
  - isolated — concurrent states appear serialized, transient states not observable
  - durable — committed transaction is persistent, i.e. system failure cannot undo result of transaction

availability + reliability improvements mainly comprise
- extending uptime
- restricting downtime