Clouding Computing Week-2

Building Blocks for Distributed Systems
Grid Computing

Gossip Protocol
- multicast problem
  - node with information and nodes that want to receive the information
- multiclass protocol (application level does not deal with the underlying network)
  - fault-tolerance and scalability
  - nodes may crash
  - packets may be dropped
  - 1000s of nodes
- centralized
  - sender goes through loop and sends UDP/TCP packets
  - overhead on the sender is very high and very high latency
  - what if the sender fails?
- Tree-based
  - spanning tree among the nodes
  - use spanning tree to disseminate multicasts
  - use acknowledgements or negative acknowledgements to repair multicasts not received
  - SRM (scalable reliable multicast)
    - use NAKs
    - but use random delays to avoid NAK storms, and using exponential back off
  - RMTP (reliable multicast transport protocol)
    - use ACKS
    - but only sent to designated receivers, which the re-transmit missing multicasts
- Third Approach
  - multicast sender
  - periodically, transmit to b random targets
  - once a node obtains the gossip it then sends out periodically, transmit to b random targets
    - nodes may receive multiple duplicate messages
    - all nodes receive the gossip message
    - the increase in overhead involved is not that much
- epidemic multicast or gossip
  - when node receives gossip message becomes infected
  - then begins sending out a random to targets
- push gossip
  - once you have a multicast message, you start gossiping about it
  - multiple messages
    - gossip a random subset of the messages
- pull gossip
  - periodically poll a few randomly selected processes for new multicast messages that you haven't received
  - get those messages
- hybrid variant

Gossip Analysis

push protocol
- lightweight in large groups
- gossip converges

epidemic multicast
- B = b/n
  - b targets
  - n possible targets per round
- substituting, at time t=clog(n), the number of infected is
  - y ~~ (n+1) - (1/n^(cb-2))

- within in clog(n) rounds, log latency

Gossip Implementations
- NTTP Inter-server protocol
  - each client uploads and downloads news posts from a new server
  - server retains news posts for a while, transmits them lazily, deletes them after a while

———-
Group membership

group based systems
- clouds/datacenters
- replicated servers
- distributed databases

crash-stop/Fail-stop process failures

group membership service
- membership list that have not failed
  - applications access memberships
  - membership protocol updates the membership list
    - failures, leaves, and joins
    - unreliable communication
- two sub protocols
  - complete list all the time (strongly consistent)
  - almost - complete list (weakly consistent)
  - partial random list (other systems)
  - two protocols
    - dissemination
      - process joins and leaves
    - failure detector

Failure Detectors
- p_j crashes
  - common case rather than exception

- completeness = each failure is detected (eventually by a non faulty process) (guaranteed)
- accuracy = there is no mistaken detection (partial/probabilistic guarantee)
- speed (between the time of failure and when the time of the failure is detected)
  - time to first detection of failure
- scale (no bottlenecks/single failure point)
  - equal load on each member
  - network message load

centralized heart beating
- heart beat sent periodically
- if heart beat not received from p_i within timeout, mark pi as failed

ring heart beating
- every process sends heart beats to its left and right neighbors
- if p_j times out while waiting for p_i marks p_j failed
- unpredictable on multiple failures

All-to-all heart beating
- each process p_i sends out heart beats to all the other process in the system
- if p_j times out waiting for p_i marks p_i as failed
- equal load per member
- if have on process that is slow might end up marking all the other processes as failed
  - improve this by using more robust ways of sending messages rather than direct messages

Gossip-style heart beating (sends to a subset at random)
- protocol
  - nodes periodically gossip their membership list
  - on receipt, the local membership list is updated
- if heart beat has not increased for more than T-fail seconds the member is considered failed
- and after t_cleanup seconds it will delete the member from the list

What happens if gossip period T_gossip is decreased?

A single heartbeat takes O(log(n)) time to propagate

What happens to P_mistake as T-fail and T_cleanup is increased
- Tradeoff: False positive rate vs detection time vs bandwidth

Which is the best failure detector
- completeness (guarantee always)
- accuracy (probability pm(t))
- speed (T time units)
- scale
  - network message load

Another probabilistic failure detector
swim failure detector
- using pinging instead of heartbeating
  - random p_j ping
  - random k ping_req

Dissemination options
- multicast
  - unreliable
  - multiple simultaneous multicasts
- point to point
  - expensive
- zero extra messages: piggyback on failure detector messages
  - infection-style dissemination

Infection-style dissemination
- epidemic style dissemination
- after lambda log(N) protocol periods N ^ (2k-2) processes would not have heard about an update
- maintain a buffer of recently joined/evicted processes
  - piggyback from this buffer
  - prefer recent updates
- buffer elements are garbage collected after a while
- after lambda log (N) protocol periods; this defines weak consistency

Suspicion mechanism
- false detections
  - perturbed processes
  - packet losses, from congestion
- indirect pinging may not solve the problem
  - correlated message losses near pinged host
- key: suspect a process before declaring it as failed in the group

———

Grid applications
- running jobs or multiple jobs across multiple resource centers (grid)

Grid Infrastructure
- 2 level scheduling infrastructure
  - Intra-site protocol
    - internal allocation and scheduling
  - across sites run a global protocol (globus protocol)
    - which job scheduled at each site
- HTCondor
  - high-throughput computing system form U
  - belongs to a class of cycle-scavenging systems

Such systems
- run on a lot of workstations
- when workstation is free, ask site's central server for tasks or globus for tasks
- if user hits a keystroke or mouse click, stop task
  - either kill task or ask server to reschedule task
- can also run on dedicated machines

Internal structure of intra protocol is invisible to global protocol

Globus
- alliance
- standardized several things, especially software
- Globus Toolkit

Grid Computing
- focuses on computation intensive computing
- though often federated, architecture and key concepts have a lot in common with that of clouds
- are grids/hpc converging towards clouds