Eventual Consistency

• Many systems: one or few updaters and many readers
  – No write-write conflicts
  – How fast should the updates be made available to other read-only processes?

• Examples:
  – DNS: single naming authority per domain
    • Only naming authority allowed updates
  – NIS: user information database in Unix systems
    • System admins update database
    • Only user updates are changes to password

Eventual Consistency

• Eventual consistency: in absence of updates, all replicas converge towards identical copies
  – Only requirement: an update should eventually propagate to all replicas
  – Cheap to implement
  – Things work fine so long as user accesses same replica
  – What if they don’t:
### Client-Centric Consistency Models

- Client-centric consistency provides guarantees for a single client concerning the consistency of accesses to a data store by that client
  - Four different consistency models
- **Monotonic reads**
  - Once read, subsequent reads on that data item return same or more recent values
- **Monotonic writes**
  - A write must be propagated to all replicas before a successive write by the same process
  - Resembles FIFO consistency (writes from same process are processed in same order)
- **Read your writes**
  - read(x) always returns write(x) by that process
- **Writes follow reads**
  - write(x) following read(x) will take place on same or more recent version of x

### Epidemic Protocols

- Implement eventual consistency
- Based on theory of epidemics (*spreading infectious diseases*)
  - Upon an update, try to “infect” other replicas as quickly as possible
  - Pair-wise exchange of updates (*like pair-wise spreading of a disease*)
- Terminology:
  - Infected store: store with an update it is willing to spread
  - Susceptible store: store that is not yet updated
Spreading Updates

• Anti-entropy
  – Server $P$ picks a server $Q$ at random and exchanges updates
  – Three possibilities: only push, only pull, both push and pull
  – Claim: A pure push-based approach does not help spread updates quickly (Why?)

• Rumor spreading (aka gossiping)
  – Upon receiving an update, $P$ tries to push to $Q$
  – If $Q$ already received the update, stop spreading with probability $1/k$
  – Does not guarantee that all replicas receive updates
    • $s = e^{-(k+1)/(1-s)}$, $s$ is the fraction of replicas that will remain susceptible
    • Example: when $k=4$, $s$ is less than 0.007

Removing Data

• Deletion of data items is hard in epidemic protocols
• Example: server deletes data item $x$
  – No state information is preserved
    • Can’t distinguish between a deleted copy and no copy!
• Solution: death certificates
  – Treat deletes as updates and spread a death certificate
    • Keep a record of the deletion
    • Need eventually clean up dormant death certificates