Eventual Consistency (1)

• Many systems have only one updater and many readers
  – No simultaneous updates (i.e., no write-write conflicts)
  – The question is how fast should the updates be made available to other read-only processes
• Examples
  – DNS: single naming authority per domain, only that naming authority is allowed to update its part of the name space
  – Web: Web pages are updated by a single authority (e.g., webmaster, owner of the page)

Eventual Consistency (2)

• **Eventual consistency** guarantees that if no updates take place for a long time, all replicas will gradually become consistent
  – Only requires that an update should eventually be propagated to all replicas (a relatively high degree of inconsistency)
  – Cheap to implement (use epidemic algorithms)
• Works fine as long as clients always access the same replica
  – What if they don’t? Introduce **client-centric consistency**
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
  - No guarantees are given concerning concurrent accesses by different clients
- Four client-centric consistency models
  - Monotonic reads
  - Monotonic writes
  - Read your writes
  - Writes follow reads

Monotonic Reads (1)

- **Definition**: If a client reads the value of a data item $x$, any successive read operation on $x$ by that client will return the same or a more recent value
- **Example**: reading email from different places
Monotonic Reads (2)

- **Notations**
  - \( x_i \) denotes the version of data item \( x \) at location \( L_i \)
  - \( WS(x_i) \) denotes the set of write operations at location \( L_i \) that leads to version \( x_i \) of \( x \) (at time \( t \)).
  - \( WS(x_i; x_j) \) denotes that operations in \( WS(x_i) \) have been propagated to location \( L_j \) (i.e., \( WS(x_i) \) is part of \( WS(x_j) \))

\[
\begin{array}{c|c|c}
\text{L1:} & WS(x_i) & R(x_1) \\
\hline
\text{L2:} & WS(x_i; x_j) & R(x_2) \\
\end{array}
\]

The read operations performed by a single process \( P \) at two different local copies of the same data store.  
(a) A monotonic-read consistent data store. (b) A data store that does not provide monotonic reads.

Monotonic Writes (1)

- **Definition**: A write operation by a client on a data item \( x \) is completed before any successive write operation on \( x \) by the same client
- **Example**: updating a file at different places
Monotonic Writes (2)

L1: \( W(x_1) \)  
L2: \( WS(x_1) \)  
\( W(x_2) \)  
(a)

L1: \( W(x_1) \)  
L2: \( W(x_2) \)  
(b)

The write operations performed by a single process \( P \) at two different local copies of the same data store. (a) A monotonic-write consistent data store. (b) A data store that does not provide monotonic-write consistency.

Read Your Writes (1)

- **Definition:** The effect of a write operation by a client on data item \( x \) will always be seen by a successive read operation on \( x \) by the same client.

- **Example:** Updating your Web page and guaranteeing that your Web browser shows the newest version instead of its cached copy.
Read Your Writes (2)

(a) A data store that provides read-your-writes consistency. (b) A data store that does not provide read-your-writes consistency.

Writes Follow Reads (1)

- **Definition**: A write operation by a client on a data item x following a read operation on x by the same client is guaranteed to take place on the same or a more recent value of x that was read

- **Example**: reading an article in a newsgroup and posting a reaction
Consistency Protocols

- A consistency protocol implements a specific consistency model
- Two types of consistency protocols that implement sequential consistency
  - Primary-based protocols
  - Replicated-write protocols
Primary-Based Protocols

- In a primary-based protocol, each data item $x$ has a primary copy
  - **Remote-write**: the primary copy is fixed at a remote server, writes are sent to primary server
  - **Local-write**: writes are carried out locally after moving the primary copy to the process where the write is initiated

Remote-Write Protocols

- Reads are carried out locally, writes are sent to primary server
- A write operation blocks until all backup servers have updated their local copy
- Nonblocking approach: primary server returns an ACK as soon as it has updated its local copy
  - Speeds up writes, but there is no guarantee that the write is backed up by other servers
Local-Write Protocols

- Reads and writes are carried out locally
- To update x, a process moves the primary copy of x to its own location
- Multiple successive writes can be carried out locally

Replicated-Write Protocols

- In a replicated-write protocol, writes are performed on multiple replicas simultaneously
  - Active replication
  - Quorum-based protocols
Active Replication

- Reads are performed locally, writes are sent to all replicas
- Writes must be carried out in the same order everywhere; this requires **totally-ordered multicast**
- One way to implement totally-ordered multicast is to use Lamport’s logical clocks, as discussed in Chapter 6
- Another solution is to use a centralized sequencer
  - Each write is forwarded to the sequencer
  - The sequencer assigns a unique sequence number to the write and forwards the write to all replicas
  - Each replica carries out the writes in the order of their sequence number

Quorum-Based Protocol

- All data items are associated with a version number
  - When a data item is modified, its version number is increased
- Writes are executed at a subset of all replicas, called a **write quorum**
- To read a data item, the process must contact a subset of replicas, called a **read quorum**, to find out the newest version of the data item and read it
- Let $N_R = \text{the size of the read quorum}$
  \[ N_R \]
- Let $N_W = \text{the size of the write quorum}$
  \[ N_W \]
- Let $N = \text{the number of replicas}$
- the following conditions must be satisfied:
  \[ N_R + N_W > N \text{ (prevents read-write conflicts)} \]
  \[ N_W > N/2 \text{ (prevents write-write conflicts)} \]
Quorum-Based Protocol: Examples

a) A correct choice of read and write quorum
b) A choice that may lead to write-write conflicts
c) A correct choice, known as ROWA (read-one, write-all)