Chain replication is a server protocol designed to give clients the illusion of a consistent, highly-available database [1]. Assuming that server failures can be detected immediately, the protocol can handle arbitrarily many server failures. Some client messages may be dropped and some responses may fail to reach the client, prompting the client to re-send a query or update, but the database remains consistent.

The protocol works by organizing servers in a linear chain. Each server keeps a copy of the database. The last server in the chain receives queries and confirms updates. The first server in the chain performs updates and propogates results to its successor. The consistency invariant is that a server in the chain knows no more than its predecessor.[1]

Strengths

- Clear argument, simple protocol, algorithms to handle failure at any location in the chain (or add a server to the chain).
- Transient outages sound like a real problem for contemporary datastores.

Weaknesses

- Assumes fail-stop servers and an invincible master process.
- Seems to require a lot of extra storage and communication time. Besides recommending that the first server computes the *value* to store and pass that (rather than a thunk) to successors, the authors do not suggest ways to reduce the amount of redundant information or transmit diffs instead of key/value pairs.

References

[1] Robbert van Renesse and Fred Schneider. Chain replication for supporting high throughput and availability. In *OSDI*, 2004.

[1] In the paper: if server i is a predecessor of server j, then $Hist^j \preccurlyeq Hist^i$.