# Decentralized first-come first-serve naming

Efficient and secure in the anytrust model (assuming any one of the quorum servers is correct). Bad peers can stall the assignment of names, but they cannot reassign names or make them disappear.

For simplicity, time is split to discrete units, *ticks*. Every tick servers receive transactions from clients. After each tick servers synchronize with each other the transactions received during that tick and when they reach a consensus, they publish new name assignments.

Let $L$ be the log of valid transactions received during a tick by our server, it contains entries in the form (transaction, exact receive timestamp). All messages are tagged per message type and tick, but not per receiver.

- Send `|1|`: $signed(L)$ to every other server
- For every other server $i$:
  - Receive `|1|` -> $(L_i, \sigma_i)$
  - Verify that $\sigma_i$ is a valid signature by server $i$ on $L_i$
- To every other server $j$:
  - Send `|2|`: $(i, \sigma_i)$ for all servers $i$
- For every other server $j$:
  - Receive from server $j$ `|2|` -> $(i, \sigma_{ij})$ for all servers $i$
  - For every other server $i$, verify that $\sigma_{ij} == \sigma_i$, otherwise barf and present $\sigma_i$ and $\sigma_{ij}$ as evidence of protocol violation by either $i$ or $j$. Server $j$ can prove its innocence by presenting the transaction log it received from $i$ so others can verify that this signature was indeed generated by $i$.

Every server must have received the same signature (and therefore the same set $L_i$) from all other servers. We can conclude that every server sent the same transactions with the same timestamps to everybody.

- Let $C_\nu$ be the list of entries that appeared in some $L_i$ and if commited would affect the status of name $\nu$, in the order of increasing timestamps.
- Let $T_\nu$ be the list of transactions from entries of $C_\nu$, with consecutive duplicates omitted.

About any transaction $x$ that appears more than once in $T_\nu$, we know that

(a) Multiple servers must have reported seeing that transaction, first at time $t$ and last at time $t'$.

(b) At least one server must have reported at least one transaction $y$ that would affect name $\nu$ and $y$ was reported to have happened between $t$ and $t'$.

(c) As either one of the servers that reported $t$ and $t'$ could be lying, we do not know whether we should execute first $x$ and then $y$, or first $y$ and then $x$.

- Remove from each $T_\nu$ all instances of any transaction $x$ that occurs multiple times, and also remove all transactions that were bracketed by instances of $x$. For example, `abcebdef` becomes `af`.

For all $T_\nu$, we know there are no transactions left about the ordering which nodes made contradictory claims. We can process these transactions without any node disagreeing.

- Create a copy $D'$ of the name assignment database $D$ and run all remaining transactions from all $T_\nu$ on it.
- Send to all servers "I will publish this database iff you all do" `|3|`: $signed(hash(D'))$
- Receive from all nodes and verify signatures |3| -> $hash(D'')$ and check that $hash(D'') == hash(D')$, otherwise barf.
- Send to all servers "I publish this database" `|4|`: $signed(hash(D'))$
- Receive from all nodes `|4|` -> $\sigma_\mathbf{i}$, ignore incorrect signatures
- Publish $D'$ together with all the `|4|` signatures.