Disposition 9: Synchronous Agreement

Mathias Ravn Tversted

January 12, 2020

Table of contents

Synchronous Broadcast

Synchronous Broadcast: The Goal

With synchronous broadcast, we are trying to solve an agreement problem. We are looking for the following properties

- **Agreement**: All honest parties make the same decision
- **Validity**: The decision must be sensible in some sensible
- **Termination**: If all parties start running the protocol, then all honest parties must end up with some decision

Synchronous Agreement

And we are looking at the following agreement problems:
Broadcast: The sender S sends a single message. All receivers a message or NoMsg and agree on an output. If S is honest, then only the message can be output as coming from S. If S is honest, no one outputs NoMsg.

Byzantine Agreement: There are n parties $P_1, \ldots, P_n$. Each has bit b_i as input. They output a common decision bit d. All parties should agree on d. If all parties have the same input, they should all agree.

Definition of broadcast

There are n parties. $P_1, \ldots, P_n$. One sends message m to all the other parties. We are looking for *agreement*, *validity*, *termination*.

Dolev-Strong protocol

Dolev-Strong

The Dolev-Strong is essentially just a normal round-based protocol with signatures, but with a twist: A party will only accept a forwarded message if it has been signed by the person sending it. It requires a new signature for every round.

Dolev-Strong Protocol: This Time It's Protocol

- **Initialise**: Activate ToyPKI and learn people's public keys. Init $\mathit{Relayed}(\mathit{bid}, m) = \emptyset$
- **Broadcast**: On $\mathit{Input}(\mathit{bid}, P_i, m)$. Set $\mathit{Relayed}(\mathit{bid}, m) = \top$. Compute and add signature and send $(\mathit{bid}, P_i, m, \mathit{sigset})$
- **Relay**: On input $(\mathit{bid}, P_i, ?)$ in round r, if not relayed, then verify signatures ($r - 1$ signatures in round r). One of these has to be from the original sender. Add own signature, set $\mathit{Relayed}(\mathit{bid}, m) = \top$ and broadcast
- **Output**: In round $n + 2$, if there is only one message such that $\mathit{Relayed}(\mathit{bid}, m) = \top$ then output that. Else output NoMsg on Cast.

Properties of Dolev-Strong

- **Validity**: When P_j outputs m, it does if iff it saw P_i sign it, and P_i only sends m if it is honest.
- **Agreement**: Let P_j, P_k be honest, then should have $Relayed_j(bid, m) = Relayed_k(bid, m)$ in round $n + 2$. There is agreement because there are $n + 2$ rounds, so we cannot try to avoid some of the honest parties. It's complicated. ????????
- **Termination**: It terminates in round $n + 2$.

Synchronous Broadcast from Authenticated Channels

Emulating Signatures

We want to be able to emulate signatures with authenticated channels, because distribution of keys and setting up that whole infrastructure could be difficult. Instead, we will seek to emulate signatures with authenticated channels. We can then run Dolev-Strong on top of that.

EmuSig Protocol

We will use the following tools to implement the EmuSig.

- **HasSeen**: Set of parties claiming to have seen message
- **SentBy**: Either has seen party send a message, or $t + 1$ have claimed to have seen it. Keeps track of who has sent what
- **SignedBy**: If it sure party has sent the message

EmuSig: This Time It's Protocol

The actual protocol goes as follows

- **Init**: Set $\mathit{HasSeen}, \mathit{SignedBy}, \mathit{SentBy} = \emptyset$
- **Sign**: P_i wants to sign m. It sets $\mathit{SignedBy}(P_i, m) = \top$ and sends it.
- **See**: If receiving (p_i, m) add to HasSeen
- **Sent I**: If Receiving (p_i, m) from P_i itself, add to SentBy
- **Sent II**: If more than $t + 1$ has seen (P_i, m), add to $\mathit{SignedBy}$

EmuSig: This Time It's Protocol, Electric Boogie-loo

- **Signed**: If $|HasSeen| \geq n - t$ then add it to *SignedBy*
- **Transfer I**: If $SignedBy(P_i, m) = \top$ then send $(Transfer, P_i, m)$ to everyone

Lower-Bound on broadcast