# Disposition 10: Asynchronous Byzantine Agreement

Mathias Ravn Tversted

January 17, 2020

# Table of contents

# Asynchronous Model

# The asynchronous model

With the asynchronous model, we have no notion of clocks.

We have no timeouts

We only have eventual delivery of messages.

# The asynchronous model

- **Agreement**: All honest parties make the same decision
- **Validity**: Decision made must be sensible in some sensible
- **Termination**: If all parties run the protocol, eventually all honest parties will make a decision

Additionally, we say that the protocol does not guarantee liveness until all parties start running it, since people can fail arbitrarily long behind.

# Asynchronous Broadcast

## Async Broadcast from signatures

- We build it with ACast and a simple *ToyPKI*
- We will use signatures. A broadcaster $P_i$ will ask all parties to sign the message
- To broadcast, wait for $n - t$ signatures.
- Receiver outputs only if it has $n - t$ signatures

# Async Broadcast from signatures 2: This Time It's Protocol

- **Request signatures**: To send a message, send it on ACast. We ask that $n - t$ sign that message.
- **Grant signatures**: When receiving a message, add your signature to it and send it back to the receiver (on the flooding network)
- **Collect signatures**: The sender collects $n - t$ signatures.
- **Send signatures**: When sender has collected $n - t$ signatures, broadcast it. When reciving a message signed by $n - t$, output it.

# Agreement

If two honest $P_j, P_k$ output $m_j, m_k$, then we want $m_j = m_k$. If there are $t$ corrupted (possibly including sender $P_i$) then we have the following argument: Each party sees $n - t$ distinct signatures. If all corrupt $t$ parties sign both $m_j, m_k$ (causing them to be output), this gives us at most $2t$ signatures. This gives us a total of $(n - t) + 2t = n + t$ distinct signatures on either $m_j$ or $m_k$.

# Validity

Honest parties output $m$ from $P_i$ if it sees at least $n - t$ signatures. But then it saw at least $(n - t) - t = t + 1$ signatures from honest parties. But then $t + 1 \geq 1$ honest parties signed $m$. And honest parties only sign the $m$ that comes from $P_i$.

# Termination

If $P_i$ is honest, it asks all honest parties to sign $m$. At least $n - t$ honest grant signatures. $P_i$ at some point receives $n - t$ signatures on $m$. it then forwards them so all honest $P_j$ receives $m$ with $n - t$ signatures, and so they output $m$.

# Async Broadcast from Authenticated Channels

Actually implementing it with an authenticated channel can be done with Bracha broadcast. It goes as follows: Assume $P_1$ is the broadcaster.

- $P_1$ gets input $(P_1, bid, m)$ on $ACast_i$ to start. We say it got $(Broadcast, P_1, bid, m)$
- When party outputs $P_n, bid, m$ on $ACast_j$ we say it has output $(Deliver, P_n, bid, m)$
- There are no rounds, only activation rules

# Bracha Broadcast

**Send**: $P_1$: On input (BROADCAST, $P_1$, bid, $m$), send (SEND, $P_1$, bid, $m$) to all parties

**Echo**: $P_i$: On message (SEND, $P_1$, bid, $m$) from $P_1$, send (ECHO, $P_1$, bid, $m$)

**Ready 1**: $P_i$ once message (ECHO, $P_1$, bid, $m$) has been received from $n - t$ parties, send (READY, $P_1$, bid, $m$)

# Bracha Broadcast

**Ready 2**: $P_i$ once message (READY, $P_1$, bid, $m$) has been received from $t+1$ parties, send (READY, $P_1$, bid, $m$) to all parties if not yet done

**Deliver**: Once message (READY, $P_1$, bid, $m$), has been received from $n-t$ parties, output (DELIVER, $P_1$, bid, $m$) and terminate the protocol

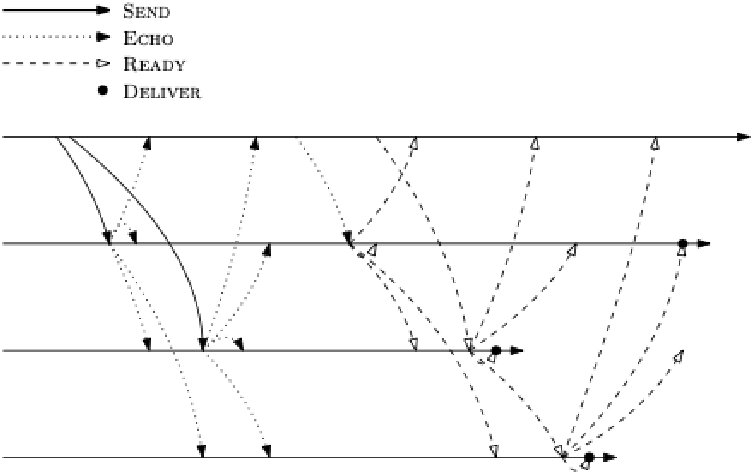


**Figure 9.4** Example execution of Bracha Broadcast with a corrupted $\mathsf{P}_1$.

# About Bracha broadcast

Each of the preceding rules are activation rules. There are no rounds. The rules can be activated in any order (READY 2 before READY 1 for example).

We also wait for $n - t$ messages, and not $n$ messages, because we cannot distinguish between late and never arriving messages. When we have enough information we act.

If you wait for $t + 1$ parties, you are also ensuring you hear from one correct party. We use this to learn that someone honest saw the mesage $m$.

We also need $n > 3t$ to ensure common correct party between two parties. (BEVIS SIDE 210?)