Reliable Broadcast vs Silent Churn

Davide Frey

Timothé Albouy, , Michel Raynal, François Taïani

Cryptocurrency

No need for a Blockchain

- [GKMPS19] Rachid Guerraoui, Petr Kuznetsov, Matteo Monti, Matej Pavlovic, Dragos-Adrian Seredinschi: “The Consensus Number of a Cryptocurrency”. PODC 2019: 307-316
- [AFRT20] Alex Auvolat, Davide Frey, Michel Raynal, François Taïani: “Money Transfer Made Simple: a Specification, a Generic Algorithm, and its Proof”. Bull. EATCS 132 (2020)
- [CGKKMPPS20] Daniel Collins, Rachid Guerraoui, Jovan Komatovic, Petr Kuznetsov, Matteo Monti, Matej Pavlovic, Yvonne-Anne Pignolet, Dragos-Adrian Seredinschi, Andrei Tonkikh, Athanasios Xygkis: “Online Payments by Merely Broadcasting Messages”. DSN 2020: 26-38
- [BDS20] Mathieu Baudet, George Danezis, and Alberto Sonnino. 2020. FastPay: High-Performance Byzantine Fault Tolerant Settlement. In Proceedings of the 2nd ACM Conference on Advances in Financial Technologies (AFT '20). Association for Computing Machinery, New York, NY, USA, 163–177.

All You Need is Broadcast

- Byzantine Reliable Broadcast
  - Formally introduced
    - 1984 Toueg (PODC 84)
    - 1985 Bracha & Toueg (JACM 85)
    - 1987 Bracha (I&C 87)
  - Ensure that
    - Correct processes: deliver the same set of messages
    - This set includes all the messages they br-broadcast

Byzantine Reliable Broadcast

- **Validity:** If a correct process delivers a message *m* from a correct process p_i then p_i broadcast *m*

- **Integrity:** No correct process delivers a message *m* more than once

- **No-Duplicity:** No two correct processes deliver distinct messages from p_i

- **Local Delivery:** If a correct process p_i broadcasts *m* then at least one correct process eventually delivers it

- **Global Delivery:** If a correct process delivers a message *m* from p_i then all correct processes deliver *m* from p_i

Introducing Silent Churn

- Typical work on distributed algorithms
  - join/leave operation informing other processes
  - announced disconnections / connections
  - Not what happens in real systems
- Silent Churn
  - Nodes can join or leave silently
  - Reflects the behavior of peer-to-peer systems
  - Model Silent Churn using Message Adversary
  - No attendance lists
  - Process ignore the online/offline state of other processes

Message Adversary

- A message adversary is a (constrained) daemon that, at the network level, eliminates messages sent by processes

- Introduced in the context of synchronous networks by Santoro
  - Santoro N. and Widmayer P., “Time is not a healer”. (STACS’89), Springer LNCS 349, pp. 304-316 (1989)
  - Santoro N. and Widmayer P., “Agreement in synchronous networks with ubiquitous faults”. Theoretical Computer Science, 384(2-3): 232-249 (2007)

Message Adversary Definition

- Broadcast an (implementation) message

```
broadcast(v) {
        for (i in 1 .. n) {
            send (v) to $p_i$
        }
}
```

- For each such broadcast, the message adversary is allowed to suppress up to *d copies of v*

- Remarks
  - Byzantine processes do not necessarily use the broadcast macro
  - d=0 <-> no message adversary

.

Modeling Silent Churn with a Message Adversary

- set D of d’ <= d processes
- adversary suppresses all the messages sent to the processes in *D*,
    - making them input disconnected
- size and content of *D* can vary over time as long as d’ < d
- Message adversary only constrains process inputs
    - Model is perfect with event-based algorithms
    - Open question as to what happens with general broadcast algorithms

SCB, Silent Churn Broadcast, i.e. BRB with Msg Adv/Silent Churn

- **Validity:** If a correct process delivers a message *m* from a correct process p_i then p_i broadcast *m*
- **Integrity:** No correct process delivers a message *m* more than once
- **No-Duplicity:** No two correct processes deliver distinct messages from p_i
- **Local Delivery:** If a correct process p_i broadcasts *m* then at least one correct process eventually delivers it
- **Global Delivery:** If a correct process delivers a message *m* from p_i then at most *d* correct processes do not deliver *m* from p_i

Two Main Results

- SCB impossible if n<=3t + 2d

- SCB algorithm with signatures

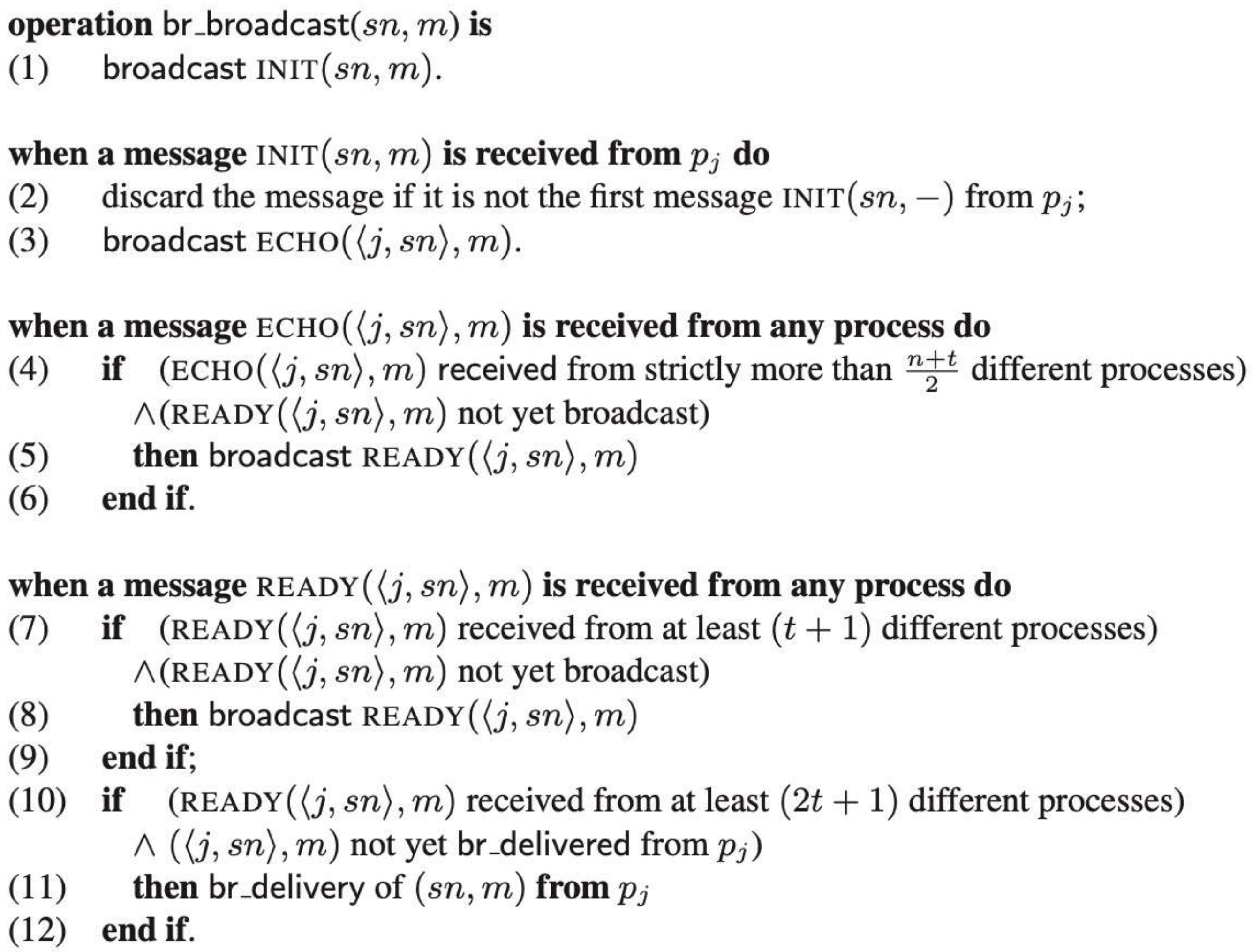
SCB Impossible if $n<=3t+2d$

- Extension of the well known result for BRB
  - Same as BRB when $d=0$
  - Same as unreliable fair channels when $t=0$
- Holds for event-driven protocols
  - only send implementation messages in response to
    - broadcast operation
    - receipt of implementation messages
- Does it hold with spontaneous messages?
  - we conjecture it does
  - maybe…

Towards an SCB Algorithm

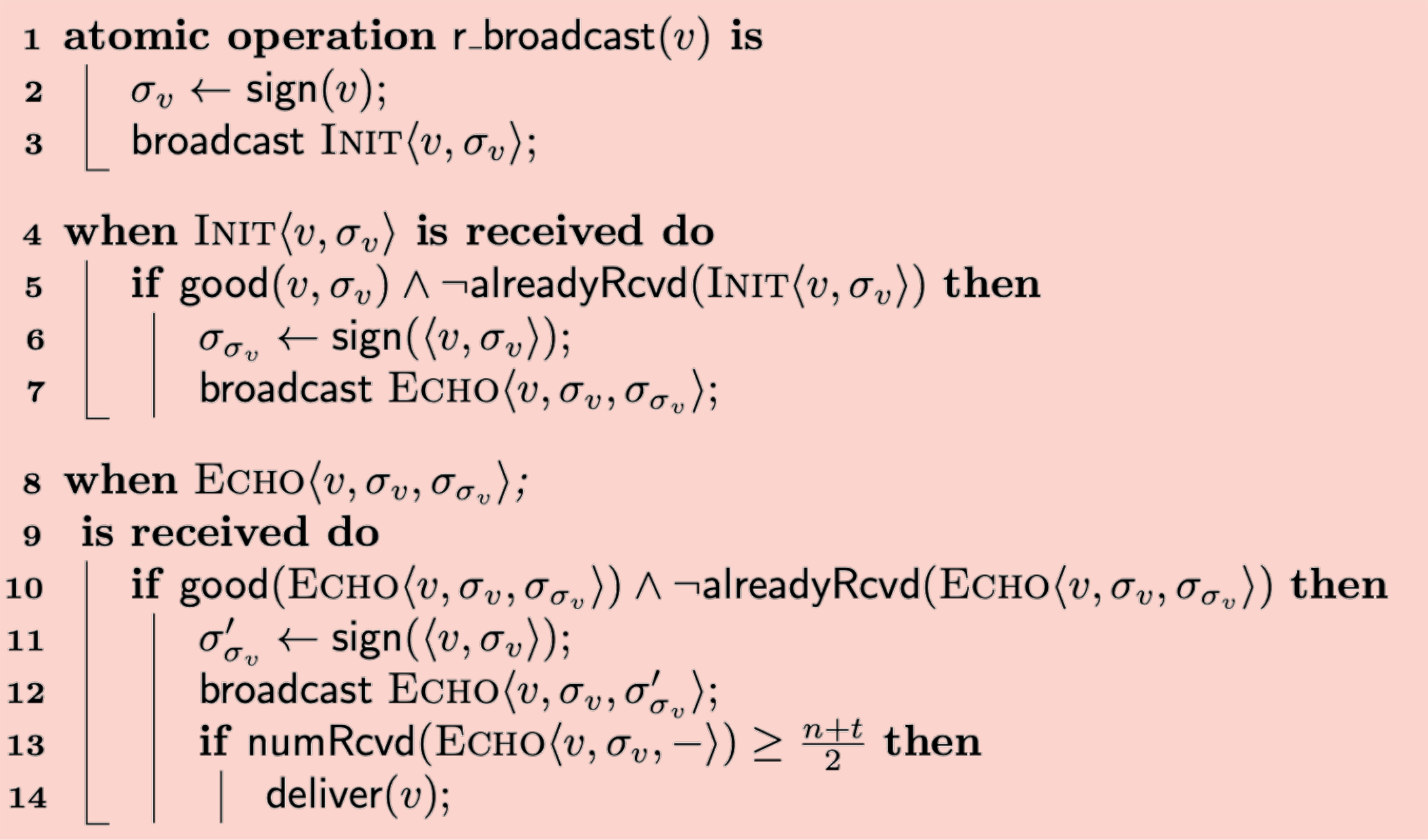
- Signature Free BRB (Bracha’s Algorithm)
- Signature-Based BRB
- Signature-Based SCB (Timothe’s Algorithm)
- Signature-Free SCB?

Signature-Free BRB (Bracha)



```
operation br_broadcast($sn, m$) is
    broadcast INIT($sn, m$).

when a message INIT($sn, m$) is received from $p_j$ do
    discard the message if it is not the first message INIT($sn, -$) from $p_j$;
    broadcast ECHO($\langle j, sn \rangle, m$).

when a message ECHO($\langle j, sn \rangle, m$) is received from any process do
    if (ECHO($\langle j, sn \rangle, m$) received from strictly more than $\frac{n+t}{2}$ different processes)
        $\wedge$(READY($\langle j, sn \rangle, m$) not yet broadcast)
        then broadcast READY($\langle j, sn \rangle, m$)
    end if.

when a message READY($\langle j, sn \rangle, m$) is received from any process do
    if (READY($\langle j, sn \rangle, m$) received from at least $(t+1)$ different processes)
        $\wedge$(READY($\langle j, sn \rangle, m$) not yet broadcast)
        then broadcast READY($\langle j, sn \rangle, m$)
    end if;
    if (READY($\langle j, sn \rangle, m$) received from at least $(2t+1)$ different processes)
        $\wedge$ ($\langle j, sn \rangle, m$) not yet br_delivered from $p_j$)
        then br_delivery of $(sn, m)$ from $p_j$
    end if.
```

Signature-Based BRB (Timothé)



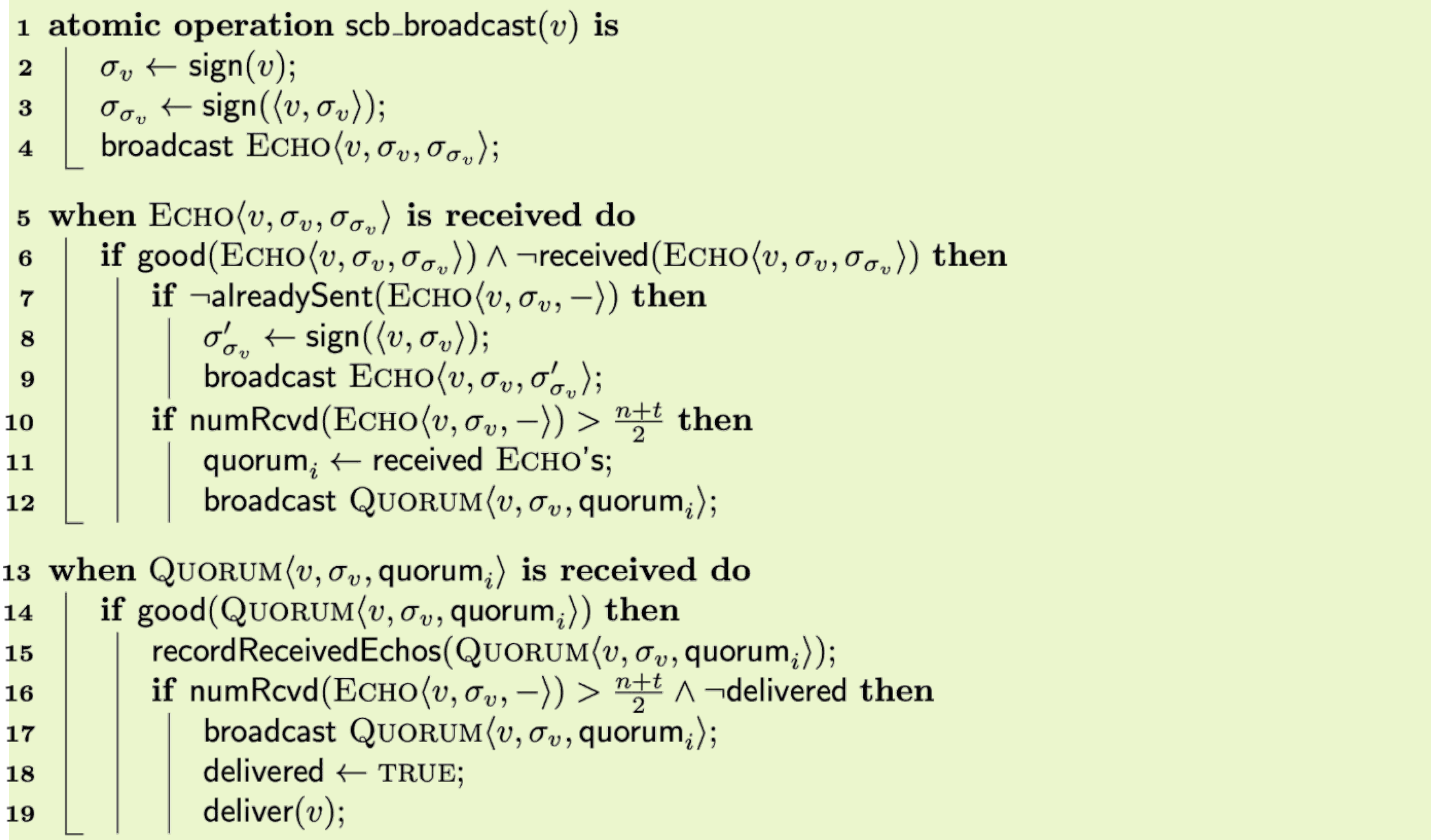
```
atomic operation r_broadcast($v$) is
    $\sigma_v \leftarrow$ sign($v$);
    broadcast INIT$\langle v, \sigma_v \rangle$;

when INIT$\langle v, \sigma_v \rangle$ is received do
    if good($v, \sigma_v$) $\wedge \neg$alreadyRcvd(INIT$\langle v, \sigma_v \rangle$) then
        $\sigma_{\sigma_v} \leftarrow$ sign($\langle v, \sigma_v \rangle$);
        broadcast ECHO$\langle v, \sigma_v, \sigma_{\sigma_v} \rangle$;

when ECHO$\langle v, \sigma_v, \sigma_{\sigma_v} \rangle$;
 is received do
    if good(ECHO$\langle v, \sigma_v, \sigma_{\sigma_v} \rangle$) $\wedge \neg$alreadyRcvd(ECHO$\langle v, \sigma_v, \sigma_{\sigma_v} \rangle$) then
        $\sigma'_{\sigma_v} \leftarrow$ sign($\langle v, \sigma_v \rangle$);
        broadcast ECHO$\langle v, \sigma_v, \sigma'_{\sigma_v} \rangle$;
        if numRcvd(ECHO$\langle v, \sigma_v, - \rangle$) $\geq \frac{n+t}{2}$ then
            deliver($v$);
```

If we add Message Adversary

- Safety is retained

- Liveness is lost

Signature-Based SCB (Timothé)

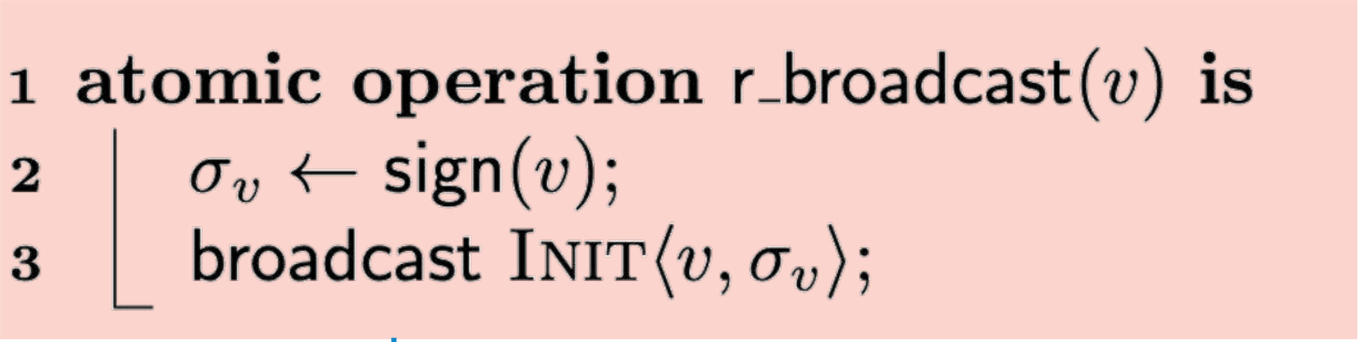


```
atomic operation scb_broadcast($v$) is
    $\sigma_v \leftarrow$ sign($v$);
    $\sigma_{\sigma_v} \leftarrow$ sign($\langle v, \sigma_v \rangle$);
    broadcast $\text{ECHO}\langle v, \sigma_v, \sigma_{\sigma_v} \rangle$;

when $\text{ECHO}\langle v, \sigma_v, \sigma_{\sigma_v} \rangle$ is received do
    if good($\text{ECHO}\langle v, \sigma_v, \sigma_{\sigma_v} \rangle$) $\wedge \neg$received($\text{ECHO}\langle v, \sigma_v, \sigma_{\sigma_v} \rangle$) then
        if $\neg$alreadySent($\text{ECHO}\langle v, \sigma_v, - \rangle$) then
            $\sigma'_{\sigma_v} \leftarrow$ sign($\langle v, \sigma_v \rangle$);
            broadcast $\text{ECHO}\langle v, \sigma_v, \sigma'_{\sigma_v} \rangle$;
        if numRcvd($\text{ECHO}\langle v, \sigma_v, - \rangle$) $> \frac{n+t}{2}$ then
            quorum$_i \leftarrow$ received ECHO's;
            broadcast $\text{QUORUM}\langle v, \sigma_v, \text{quorum}_i \rangle$;

when $\text{QUORUM}\langle v, \sigma_v, \text{quorum}_i \rangle$ is received do
    if good($\text{QUORUM}\langle v, \sigma_v, \text{quorum}_i \rangle$) then
        recordReceivedEchos($\text{QUORUM}\langle v, \sigma_v, \text{quorum}_i \rangle$);
        if numRcvd($\text{ECHO}\langle v, \sigma_v, - \rangle$) $> \frac{n+t}{2} \wedge \neg$delivered then
            broadcast $\text{QUORUM}\langle v, \sigma_v, \text{quorum}_i \rangle$;
            delivered $\leftarrow$ TRUE;
            deliver($v$);
```

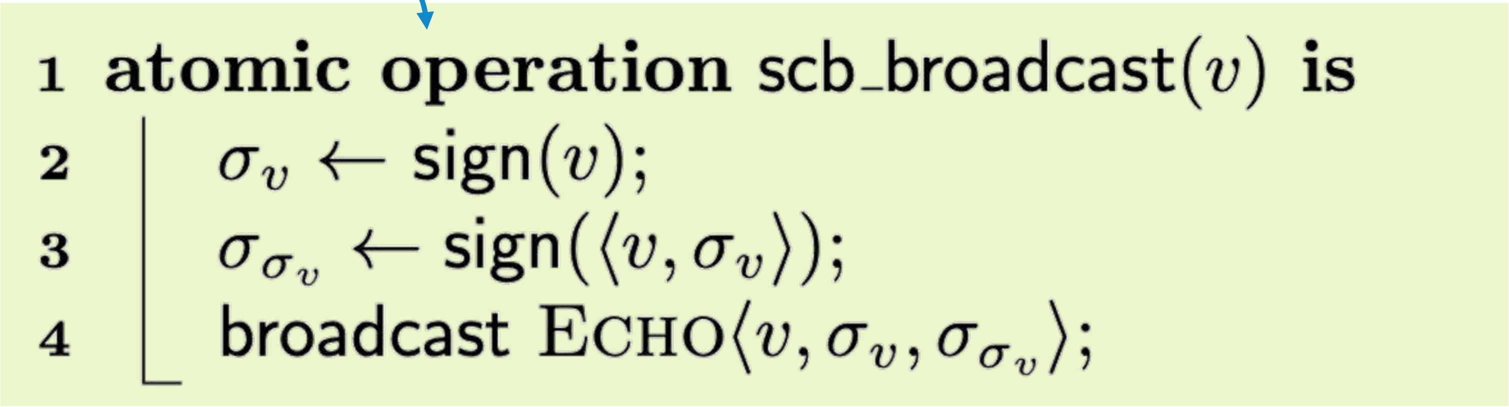
Signature-Based SCB

- Two main changes
  - Collapse ECHO and INIT
    - (OR send ECHO without INIT)
  - Additional Communication Round
    - QUORUM message
    - make sure everyone reaches quorum of ECHOs

Collapse ECHO and INIT



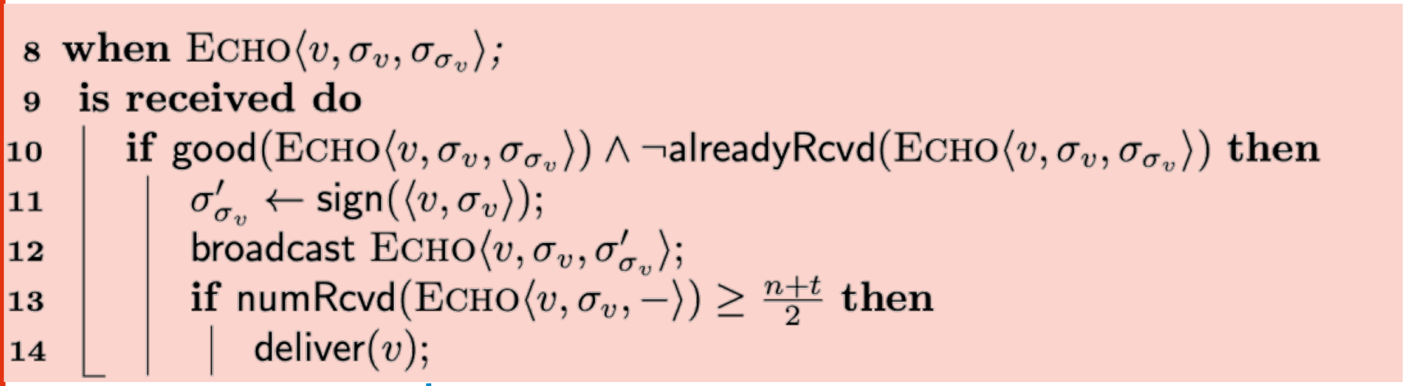
```
atomic operation r_broadcast($v$) is
    $\sigma_v \leftarrow$ sign($v$);
    broadcast INIT$\langle v, \sigma_v \rangle$;
```



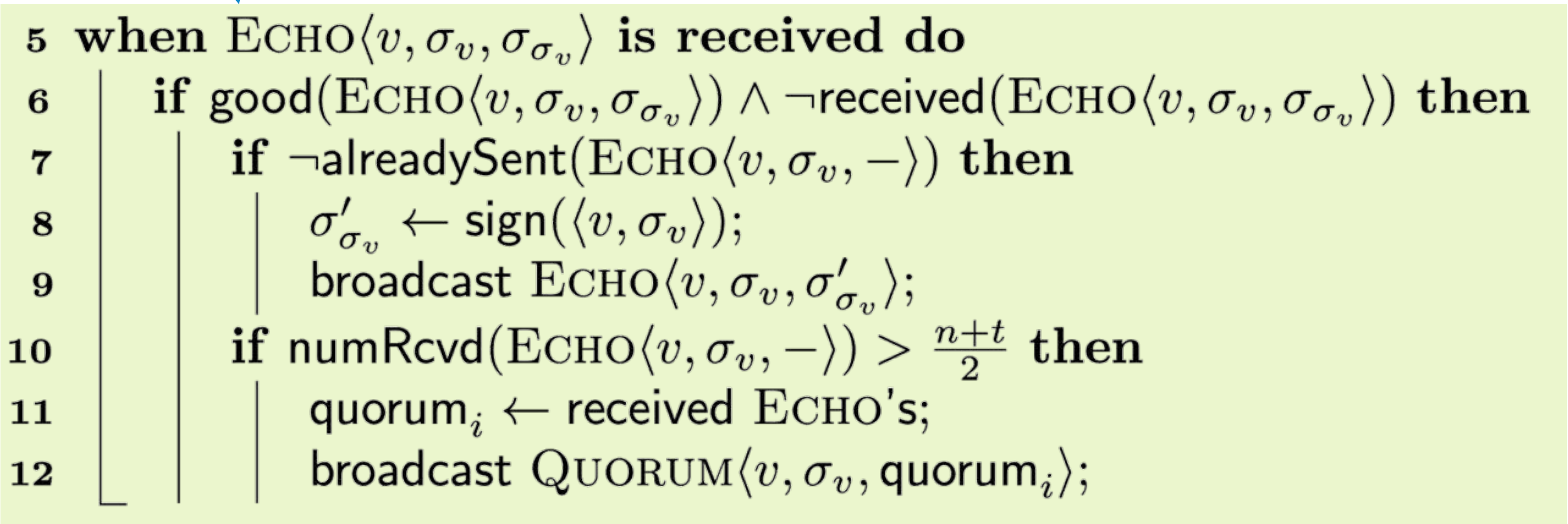
```
atomic operation scb_broadcast($v$) is
    $\sigma_v \leftarrow$ sign($v$);
    $\sigma_{\sigma_v} \leftarrow$ sign($\langle v, \sigma_v \rangle$);
    broadcast ECHO$\langle v, \sigma_v, \sigma_{\sigma_v} \rangle$;
```

tackle case when INIT is lost

Add QUORUM Message

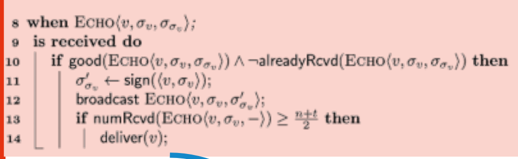


```
when $\textsc{Echo}\langle v, \sigma_v, \sigma_{\sigma_v} \rangle$;
 is received do
    if $\mathsf{good}(\textsc{Echo}\langle v, \sigma_v, \sigma_{\sigma_v} \rangle) \wedge \neg\mathsf{alreadyRcvd}(\textsc{Echo}\langle v, \sigma_v, \sigma_{\sigma_v} \rangle)$ then
        $\sigma'_{\sigma_v} \leftarrow \mathsf{sign}(\langle v, \sigma_v \rangle)$;
        broadcast $\textsc{Echo}\langle v, \sigma_v, \sigma'_{\sigma_v} \rangle$;
        if $\mathsf{numRcvd}(\textsc{Echo}\langle v, \sigma_v, - \rangle) \geq \frac{n+t}{2}$ then
            $\mathsf{deliver}(v)$;
```

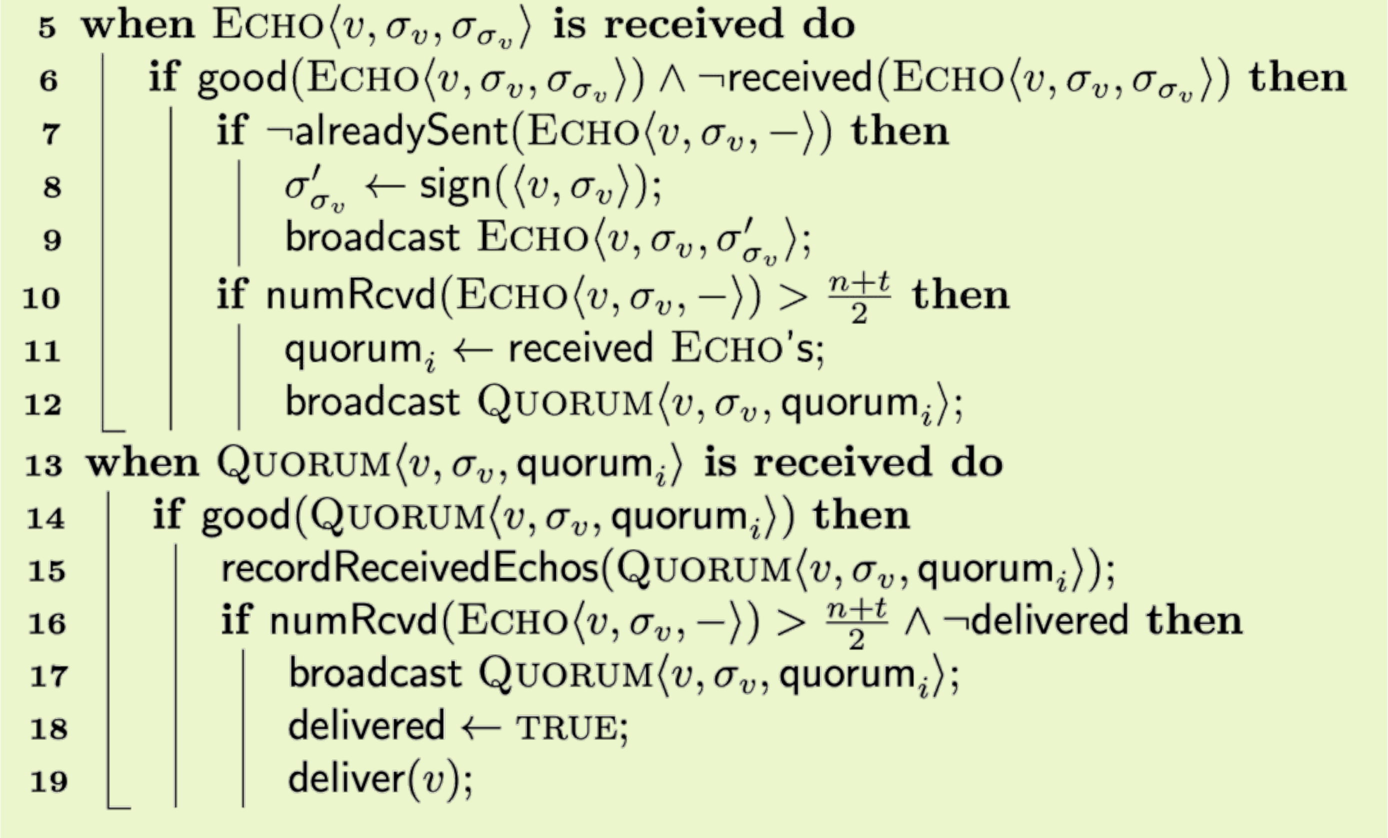


```
when $\textsc{Echo}\langle v, \sigma_v, \sigma_{\sigma_v} \rangle$ is received do
    if $\mathsf{good}(\textsc{Echo}\langle v, \sigma_v, \sigma_{\sigma_v} \rangle) \wedge \neg\mathsf{received}(\textsc{Echo}\langle v, \sigma_v, \sigma_{\sigma_v} \rangle)$ then
        if $\neg\mathsf{alreadySent}(\textsc{Echo}\langle v, \sigma_v, - \rangle)$ then
            $\sigma'_{\sigma_v} \leftarrow \mathsf{sign}(\langle v, \sigma_v \rangle)$;
            broadcast $\textsc{Echo}\langle v, \sigma_v, \sigma'_{\sigma_v} \rangle$;
        if $\mathsf{numRcvd}(\textsc{Echo}\langle v, \sigma_v, - \rangle) > \frac{n+t}{2}$ then
            $\mathsf{quorum}_i \leftarrow$ received $\textsc{Echo}$'s;
            broadcast $\textsc{Quorum}\langle v, \sigma_v, \mathsf{quorum}_i \rangle$;
```

Add QUORUM Message



```
when $\textsc{Echo}\langle v, \sigma_v, \sigma_{\sigma_v}\rangle$;
 is received do
    if good($\textsc{Echo}\langle v, \sigma_v, \sigma_{\sigma_v}\rangle$) $\wedge$ $\neg$alreadyRcvd($\textsc{Echo}\langle v, \sigma_v, \sigma_{\sigma_v}\rangle$) then
        $\sigma'_{\sigma_v} \leftarrow$ sign($\langle v, \sigma_v\rangle$);
        broadcast $\textsc{Echo}\langle v, \sigma_v, \sigma'_{\sigma_v}\rangle$;
        if numRcvd($\textsc{Echo}\langle v, \sigma_v, -\rangle$) $\geq \frac{n+t}{2}$ then
            deliver($v$);
```



```
when $\textsc{Echo}\langle v, \sigma_v, \sigma_{\sigma_v}\rangle$ is received do
    if good($\textsc{Echo}\langle v, \sigma_v, \sigma_{\sigma_v}\rangle$) $\wedge$ $\neg$received($\textsc{Echo}\langle v, \sigma_v, \sigma_{\sigma_v}\rangle$) then
        if $\neg$alreadySent($\textsc{Echo}\langle v, \sigma_v, -\rangle$) then
            $\sigma'_{\sigma_v} \leftarrow$ sign($\langle v, \sigma_v\rangle$);
            broadcast $\textsc{Echo}\langle v, \sigma_v, \sigma'_{\sigma_v}\rangle$;
        if numRcvd($\textsc{Echo}\langle v, \sigma_v, -\rangle$) $> \frac{n+t}{2}$ then
            quorum$_i$ $\leftarrow$ received $\textsc{Echo}$'s;
            broadcast $\textsc{Quorum}\langle v, \sigma_v, \text{quorum}_i\rangle$;
when $\textsc{Quorum}\langle v, \sigma_v, \text{quorum}_i\rangle$ is received do
    if good($\textsc{Quorum}\langle v, \sigma_v, \text{quorum}_i\rangle$) then
        recordReceivedEchos($\textsc{Quorum}\langle v, \sigma_v, \text{quorum}_i\rangle$);
        if numRcvd($\textsc{Echo}\langle v, \sigma_v, -\rangle$) $> \frac{n+t}{2}$ $\wedge$ $\neg$delivered then
            broadcast $\textsc{Quorum}\langle v, \sigma_v, \text{quorum}_i\rangle$;
            delivered $\leftarrow$ TRUE;
            deliver($v$);
```

Signature-Free SCB?

We're working hard for ~~You~~ Science

More Powerful Message Adversaries

- We considered receipt omissions

- How about altered messages?

To Summarize

- ~~Consensus is overrated~~ Reliable Broadcast is Important
- Novel model for silent churn
- Modeled by a message adversary
- Impossibility if $n<3t+2d$ with
- Novel Signature-Based Protocol
- Working on Signature-Free Protocol
- Theory (of Distributed Algorithms) is Fun!