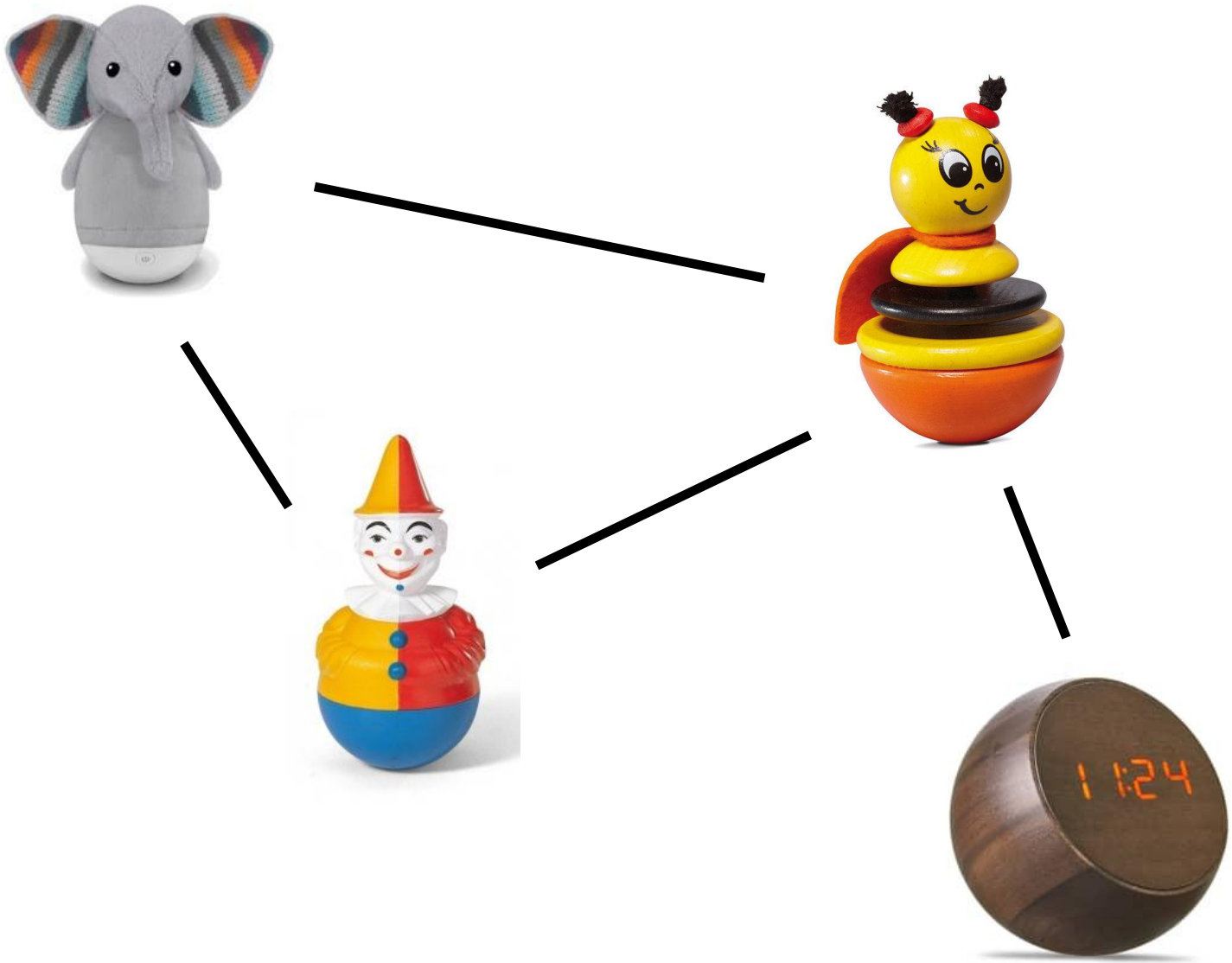
Self-Stabilization & Recovery



Want: Self-stabilizing GCS

Theorem

For any $\mu \geq 2(\theta-1)$, there is a self-stabilizing algorithm s.t.

$$\mathbf{dH/dt \leq dL/dt \leq (1+\mu)dH/dt,}$$

the global and local skew are

$$\mathbf{O((u+\mu d)\, D)} \quad \text{and} \quad \mathbf{O((u+\mu d)\, \log_{\sigma} D)}, \text{ respectively,}$$

where $\sigma = \mu/(\theta-1)$, and the stabilization time is

$$\mathbf{O((d+u/\mu)D)}.$$

Self-stabilizing GCS, Step by Step

Step	Done?	Time
Stabilize BFS tree	✓	O(dD)

Self-stabilizing GCS, Step by Step

Step	Done?	Time
Stabilize BFS tree	✓	O(dD)
Reduce global skew to O((u+μd)D)	✗	O(dD)

Self-stabilizing GCS, Step by Step

Step	Done?	Time
Stabilize BFS tree	✓	O(dD)
Reduce global skew to $O((u+\mu d)D)$	✗	O(dD)
Stabilize estimates of $\delta = O(u+\mu d)$	✓	O(d)

Self-stabilizing GCS, Step by Step

Step	Done?	Time
Stabilize BFS tree	✓	O(dD)
Reduce global skew to O((u+μd)D)	✗	O(dD)
Stabilize estimates of δ = O(u+μd)	✓	O(d)
Maintain global skew of O((u+μd)D)	✓*	0

*Almost same proof, but modify for small times.

Self-stabilizing GCS, Step by Step

Step	Done?	Time
Stabilize BFS tree	✓	O(dD)
Reduce global skew to $O((u+\mu d)D)$	✗	O(dD)
Stabilize estimates of $\delta = O(u+\mu d)$	✓	O(d)
Maintain global skew of $O((u+\mu d)D)$	✓*	0
Stabilize local skew to $O((u+\mu d)\ \log_\sigma D)$	✓*	$O(u/\mu+d)D)$

*Almost same proof, but modify for/discard small times.

Self-stabilizing GCS, Step by Step

Step	Done?	Time
Stabilize BFS tree	✓	O(dD)
Reduce global skew to O((u+μd)D)	✗	O(dD)
Stabilize estimates of δ = O(u+μd)	✓	O(d)
Maintain global skew of O((u+μd)D)	✓*	0
Stabilize local skew to O((u+μd) $\log_\sigma$ D)	✓*	O(u/μ+d)D)
		=> O(u/μ+d)D)

*Almost same proof, but modify for/discard small times.

Breakout Session

Step	Done?	Time
Stabilize BFS tree	✓	O(dD)
Reduce global skew to O((u+μd)D)	**??**	**O(dD)**

catch: this subroutine must modify logical clocks

=> need to avoid messing up the GCS algorithm!

Adding Recovery to Lynch-Welch

Assumption:

There are always n-f correct & synchronized nodes.

These will pulse as for "standard" LW.

Goal:

Nodes should resync after transiently failing!

Algorithm 12 Lynch-Welch pulse synchronization algorithm with recovery mechanism, code for node $v \in V_g$.

1: wait until getH() $\geq \mathcal{S}$ // $H_w(0) \in [0, \mathcal{S})$ for correct nodes w
2: **for all** rounds $r \in \mathbb{N}$ **do**
3: generate r-th pulse
4: $h \leftarrow$ getH()
5: wait until getH() $\geq h + 2\vartheta\mathcal{S}$ or getH() $< h$ // all nodes are in round r
6: broadcast empty message to all nodes (including self)
7: wait until getH() $\geq h + 2(\vartheta^2 + \vartheta)\mathcal{S} + \vartheta d$ or getH() $< h + 2\vartheta\mathcal{S}$ // denote this time by $\tau_{v,r}$
// correct nodes' messages should have arrived
8: **if** received messages from $n - f$ distinct nodes during current loop iteration **then**
9: $h' \leftarrow$ median of $\{h_x \mid h_x$ local reception time of latest message from some $x \in V\}$ (as multiset, i.e., values may repeat)
10: **for** each node $w \in V$ **do**
11: **if** received message from w during current loop iteration **then**
12: let h_w be local time at latest received message from w
13: $\Delta(w) \leftarrow h_w - h - d + u - 2\mathcal{S}$
14: **else**
15: $\Delta(w) \leftarrow h' - h - d + u - 2\mathcal{S}$ // replace missing ones by median
16: **end if**
17: **end for**
18: $U \leftarrow \{\Delta(w) \mid w \in V\}$ (as multiset, i.e., values may repeat)
19: $\Delta \leftarrow \left(U^{(f+1)} + U^{(n-f)}\right)/2$
20: wait until getH() $\geq h + \Delta + T$ or getH() $< h + \Delta - 3\mathcal{S}$
21: **else**
22: wait for $n - f$ messages from distinct nodes within $\vartheta^2\mathcal{S} + \vartheta u$ local time
23: let h' be the local reception time of the $(f+1)$-th such message
24: wait until getH() $\geq h' - d + u - 2\mathcal{S} + T$ or getH() $< h' - (\vartheta^2\mathcal{S} + \vartheta u)$
25: **end if**
26: **end for**

Adding Recovery to Lynch-Welch

Main changes:

1. timing checks

   => no “getting stuck”

   due to buggy state

Algorithm 12 Lynch-Welch pulse synchronization algorithm with recovery mechanism, code for node $v \in V_g$.

1: wait until getH() $\geq \mathcal{S}$ // $H_w(0) \in [0, \mathcal{S})$ for correct nodes w
2: **for all** rounds $r \in \mathbb{N}$ **do**
3: generate r-th pulse
4: $h \leftarrow$ getH()
5: wait until getH() $\geq h + 2\vartheta\mathcal{S}$ or getH() $< h$ // all nodes are in round r ←
6: broadcast empty message to all nodes (including self)
7: wait until getH() $\geq h + 2(\vartheta^2 + \vartheta)\mathcal{S} + \vartheta d$ or getH() $< h + 2\vartheta\mathcal{S}$ // ← denote this time by $\tau_{v,r}$
// correct nodes' messages should have arrived
8: **if** received messages from $n - f$ distinct nodes during current loop iteration **then**
9: $h' \leftarrow$ median of $\{h_x \mid h_x$ local reception time of latest message from some $x \in V\}$ (as multiset, i.e., values may repeat)
10: **for** each node $w \in V$ **do**
11: **if** received message from w during current loop iteration **then**
12: let h_w be local time at latest received message from w
13: $\Delta(w) \leftarrow h_w - h - d + u - 2\mathcal{S}$
14: **else**
15: $\Delta(w) \leftarrow h' - h - d + u - 2\mathcal{S}$ // replace missing ones by median
16: **end if**
17: **end for**
18: $U \leftarrow \{\Delta(w) \mid w \in V\}$ (as multiset, i.e., values may repeat)
19: $\Delta \leftarrow \left(U^{(f+1)} + U^{(n-f)}\right)/2$
20: wait until getH() $\geq h + \Delta + T$ or getH() $< h + \Delta - 3\mathcal{S}$ ←
21: **else**
22: wait for $n - f$ messages from distinct nodes within $\vartheta^2\mathcal{S} + \vartheta u$ local time
23: let h' be the local reception time of the $(f+1)$-th such message
24: wait until getH() $\geq h' - d + u - 2\mathcal{S} + T$ or getH() $< h' - (\vartheta^2\mathcal{S} + \vartheta u)$ ←
25: **end if**
26: **end for**

Adding Recovery to Lynch-Welch

Main changes:

1. timing checks
   => no "getting stuck"
   due to buggy state

2. extra slack for message reception
   => works for skew 2S

Algorithm 12 Lynch-Welch pulse synchronization algorithm with recovery mechanism, code for node $v \in V_g$.

1: wait until getH() $\geq \mathcal{S}$ // $H_w(0) \in [0, \mathcal{S})$ for correct nodes w
2: **for all** rounds $r \in \mathbb{N}$ **do**
3: generate r-th pulse
4: $h \leftarrow$ getH()
5: wait until getH() $\geq h + 2\vartheta\mathcal{S}$ or getH() $< h$ // all nodes are in round r ←
6: broadcast empty message to all nodes (including self)
7: wait until getH() $\geq h + 2(\vartheta^2 + \vartheta)\mathcal{S} + \vartheta d$ or getH() $< h + 2\vartheta\mathcal{S}$ // ←
denote this time by $\tau_{v,r}$
// correct nodes' messages should have arrived
8: **if** received messages from $n - f$ distinct nodes during current loop iteration **then**
9: $h' \leftarrow$ median of $\{h_x \mid h_x$ local reception time of latest message from some $x \in V\}$ (as multiset, i.e., values may repeat)
10: **for** each node $w \in V$ **do**
11: **if** received message from w during current loop iteration **then**
12: let h_w be local time at latest received message from w
13: $\Delta(w) \leftarrow h_w - h - d + u - 2\mathcal{S}$
14: **else**
15: $\Delta(w) \leftarrow h' - h - d + u - 2\mathcal{S}$ // replace missing ones by median
16: **end if**
17: **end for**
18: $U \leftarrow \{\Delta(w) \mid w \in V\}$ (as multiset, i.e., values may repeat)
19: $\Delta \leftarrow \left(U^{(f+1)} + U^{(n-f)}\right)/2$
20: wait until getH() $\geq h + \Delta + T$ or getH() $< h + \Delta - 3\mathcal{S}$
21: **else**
22: wait for $n - f$ messages from distinct nodes within $\vartheta^2\mathcal{S} + \vartheta u$ local time
23: let h' be the local reception time of the $(f+1)$-th such message
24: wait until getH() $\geq h' - d + u - 2\mathcal{S} + T$ or getH() $< h' - (\vartheta^2\mathcal{S} + \vartheta u)$
25: **end if**
26: **end for**

Adding Recovery to Lynch-Welch

Main changes:

1. timing checks
   => no "getting stuck" due to buggy state

2. extra slack for message reception
   => works for skew 2S

3. recovery mode if < n-f messages received
   => resync to sync'ed majority (skew <2S)

Algorithm 12 Lynch-Welch pulse synchronization algorithm with recovery mechanism, code for node $v \in V_g$.

1: wait until getH() $\geq \mathcal{S}$ // $H_w(0) \in [0, \mathcal{S})$ for correct nodes w
2: **for all** rounds $r \in \mathbb{N}$ **do**
3: generate r-th pulse
4: $h \leftarrow$ getH()
5: wait until getH() $\geq h + 2\vartheta\mathcal{S}$ or getH() $< h$ // all nodes are in round r
6: broadcast empty message to all nodes (including self)
7: wait until getH() $\geq h + 2(\vartheta^2 + \vartheta)\mathcal{S} + \vartheta d$ or getH() $< h + 2\vartheta\mathcal{S}$ // denote this time by $\tau_{v,r}$
// correct nodes' messages should have arrived
8: **if** received messages from $n - f$ distinct nodes during current loop iteration **then**
9: $h' \leftarrow$ median of $\{h_x \mid h_x$ local reception time of latest message from some $x \in V\}$ (as multiset, i.e., values may repeat)
10: **for** each node $w \in V$ **do**
11: **if** received message from w during current loop iteration **then**
12: let h_w be local time at latest received message from w
13: $\Delta(w) \leftarrow h_w - h - d + u - 2\mathcal{S}$
14: **else**
15: $\Delta(w) \leftarrow h' - h - d + u - 2\mathcal{S}$ // replace missing ones by median
16: **end if**
17: **end for**
18: $U \leftarrow \{\Delta(w) \mid w \in V\}$ (as multiset, i.e., values may repeat)
19: $\Delta \leftarrow \left(U^{(f+1)} + U^{(n-f)}\right)/2$
20: wait until getH() $\geq h + \Delta + T$ or getH() $< h + \Delta - 3\mathcal{S}$
21: **else**
22: wait for $n - f$ messages from distinct nodes within $\vartheta^2\mathcal{S} + \vartheta u$ local time
23: let h' be the local reception time of the $(f+1)$-th such message
24: wait until getH() $\geq h' - d + u - 2\mathcal{S} + T$ or getH() $< h' - (\vartheta^2\mathcal{S} + \vartheta u)$
25: **end if**
26: **end for**

Adding Recovery to Lynch-Welch

Why it works (1):

- can‘t get stuck on these waiting statements due to checks

Algorithm 12 Lynch-Welch pulse synchronization algorithm with recovery mechanism, code for node $v \in V_g$.

1: wait until getH() $\geq \mathcal{S}$ // $H_w(0) \in [0, \mathcal{S})$ for correct nodes w
2: **for all** rounds $r \in \mathbb{N}$ **do**
3: generate r-th pulse
4: $h \leftarrow$ getH()
5: wait until getH() $\geq h + 2\vartheta\mathcal{S}$ or getH() $< h$ // all nodes are in round r ←
6: broadcast empty message to all nodes (including self)
7: wait until getH() $\geq h + 2(\vartheta^2 + \vartheta)\mathcal{S} + \vartheta d$ or getH() $< h + 2\vartheta\mathcal{S}$ // ← denote this time by $\tau_{v,r}$
// correct nodes' messages should have arrived
8: **if** received messages from $n - f$ distinct nodes during current loop iteration **then**
9: $h' \leftarrow$ median of $\{h_x \mid h_x$ local reception time of latest message from some $x \in V\}$ (as multiset, i.e., values may repeat)
10: **for** each node $w \in V$ **do**
11: **if** received message from w during current loop iteration **then**
12: let h_w be local time at latest received message from w
13: $\Delta(w) \leftarrow h_w - h - d + u - 2\mathcal{S}$
14: **else**
15: $\Delta(w) \leftarrow h' - h - d + u - 2\mathcal{S}$ // replace missing ones by median
16: **end if**
17: **end for**
18: $U \leftarrow \{\Delta(w) \mid w \in V\}$ (as multiset, i.e., values may repeat)
19: $\Delta \leftarrow \left(U^{(f+1)} + U^{(n-f)}\right)/2$
20: wait until getH() $\geq h + \Delta + T$ or getH() $< h + \Delta - 3\mathcal{S}$ ←
21: **else**
22: wait for $n - f$ messages from distinct nodes within $\vartheta^2\mathcal{S} + \vartheta u$ local time
23: let h' be the local reception time of the $(f+1)$-th such message
24: wait until getH() $\geq h' - d + u - 2\mathcal{S} + T$ or getH() $< h' - (\vartheta^2\mathcal{S} + \vartheta u)$ ←
25: **end if**
26: **end for**

Adding Recovery to Lynch-Welch

Why it works (1):

- can‘t get stuck on these waiting statements due to checks
- can‘t get stuck on this waiting statement due to sync‘ed pulses

Algorithm 12 Lynch-Welch pulse synchronization algorithm with recovery mechanism, code for node $v \in V_g$.

1: wait until getH() $\geq \mathcal{S}$ // $H_w(0) \in [0, \mathcal{S})$ for correct nodes w
2: **for all** rounds $r \in \mathbb{N}$ **do**
3: generate r-th pulse
4: $h \leftarrow$ getH()
5: wait until getH() $\geq h + 2\vartheta\mathcal{S}$ or getH() $< h$ // all nodes are in round r
6: broadcast empty message to all nodes (including self)
7: wait until getH() $\geq h + 2(\vartheta^2 + \vartheta)\mathcal{S} + \vartheta d$ or getH() $< h + 2\vartheta\mathcal{S}$ // denote this time by $\tau_{v,r}$
// correct nodes' messages should have arrived
8: **if** received messages from $n - f$ distinct nodes during current loop iteration **then**
9: $h' \leftarrow$ median of $\{h_x \mid h_x$ local reception time of latest message from some $x \in V\}$ (as multiset, i.e., values may repeat)
10: **for** each node $w \in V$ **do**
11: **if** received message from w during current loop iteration **then**
12: let h_w be local time at latest received message from w
13: $\Delta(w) \leftarrow h_w - h - d + u - 2\mathcal{S}$
14: **else**
15: $\Delta(w) \leftarrow h' - h - d + u - 2\mathcal{S}$ // replace missing ones by median
16: **end if**
17: **end for**
18: $U \leftarrow \{\Delta(w) \mid w \in V\}$ (as multiset, i.e., values may repeat)
19: $\Delta \leftarrow \left(U^{(f+1)} + U^{(n-f)}\right)/2$
20: wait until getH() $\geq h + \Delta + T$ or getH() $< h + \Delta - 3\mathcal{S}$
21: **else**
22: wait for $n - f$ messages from distinct nodes within $\vartheta^2\mathcal{S} + \vartheta u$ local time
23: let h' be the local reception time of the $(f+1)$-th such message
24: wait until getH() $\geq h' - d + u - 2\mathcal{S} + T$ or getH() $< h' - (\vartheta^2\mathcal{S} + \vartheta u)$
25: **end if**
26: **end for**

Adding Recovery to Lynch-Welch

Why it works (1):

- can‘t get stuck on these waiting statements due to checks
- can‘t get stuck on this waiting statement due to sync‘ed pulses

=> new loop iteration within O(T) time (clears all state but timing of new iteration)

Algorithm 12 Lynch-Welch pulse synchronization algorithm with recovery mechanism, code for node $v \in V_g$.

1: wait until getH() $\geq \mathcal{S}$ // $H_w(0) \in [0, \mathcal{S})$ for correct nodes w
2: **for all** rounds $r \in \mathbb{N}$ **do**
3: generate r-th pulse
4: $h \leftarrow$ getH()
5: wait until getH() $\geq h + 2\vartheta\mathcal{S}$ or getH() $< h$ // all nodes are in round r
6: broadcast empty message to all nodes (including self)
7: wait until getH() $\geq h + 2(\vartheta^2 + \vartheta)\mathcal{S} + \vartheta d$ or getH() $< h + 2\vartheta\mathcal{S}$ // denote this time by $\tau_{v,r}$
// correct nodes' messages should have arrived
8: **if** received messages from $n - f$ distinct nodes during current loop iteration **then**
9: $h' \leftarrow$ median of $\{h_x \mid h_x$ local reception time of latest message from some $x \in V\}$ (as multiset, i.e., values may repeat)
10: **for** each node $w \in V$ **do**
11: **if** received message from w during current loop iteration **then**
12: let h_w be local time at latest received message from w
13: $\Delta(w) \leftarrow h_w - h - d + u - 2\mathcal{S}$
14: **else**
15: $\Delta(w) \leftarrow h' - h - d + u - 2\mathcal{S}$ // replace missing ones by median
16: **end if**
17: **end for**
18: $U \leftarrow \{\Delta(w) \mid w \in V\}$ (as multiset, i.e., values may repeat)
19: $\Delta \leftarrow \left(U^{(f+1)} + U^{(n-f)}\right)/2$
20: wait until getH() $\geq h + \Delta + T$ or getH() $< h + \Delta - 3\mathcal{S}$
21: **else**
22: wait for $n - f$ messages from distinct nodes within $\vartheta^2\mathcal{S} + \vartheta u$ local time
23: let h' be the local reception time of the $(f+1)$-th such message
24: wait until getH() $\geq h' - d + u - 2\mathcal{S} + T$ or getH() $< h' - (\vartheta^2\mathcal{S} + \vartheta u)$
25: **end if**
26: **end for**

Adding Recovery to Lynch-Welch

Why it works (2):

≥ n-f messages received

=> n-2f > f from sync'ed

=> (f+1)-th within range spanned by sync'ed

Algorithm 12 Lynch-Welch pulse synchronization algorithm with recovery mechanism, code for node $v \in V_g$.

1: wait until getH() $\geq \mathcal{S}$ // $H_w(0) \in [0, \mathcal{S})$ for correct nodes w
2: **for all** rounds $r \in \mathbb{N}$ **do**
3: generate r-th pulse
4: $h \leftarrow$ getH()
5: wait until getH() $\geq h + 2\vartheta\mathcal{S}$ or getH() $< h$ // all nodes are in round r
6: broadcast empty message to all nodes (including self)
7: wait until getH() $\geq h + 2(\vartheta^2 + \vartheta)\mathcal{S} + \vartheta d$ or getH() $< h + 2\vartheta\mathcal{S}$ // denote this time by $\tau_{v,r}$
// correct nodes' messages should have arrived
8: **if** received messages from $n - f$ distinct nodes during current loop iteration **then**
9: $h' \leftarrow$ median of $\{h_x \mid h_x$ local reception time of latest message from some $x \in V\}$ (as multiset, i.e., values may repeat)
10: **for** each node $w \in V$ **do**
11: **if** received message from w during current loop iteration **then**
12: let h_w be local time at latest received message from w
13: $\Delta(w) \leftarrow h_w - h - d + u - 2\mathcal{S}$
14: **else**
15: $\Delta(w) \leftarrow h' - h - d + u - 2\mathcal{S}$ // replace missing ones by median
16: **end if**
17: **end for**
18: $U \leftarrow \{\Delta(w) \mid w \in V\}$ (as multiset, i.e., values may repeat)
19: $\Delta \leftarrow \left(U^{(f+1)} + U^{(n-f)}\right)/2$
20: wait until getH() $\geq h + \Delta + T$ or getH() $< h + \Delta - 3\mathcal{S}$
21: **else**
22: wait for $n - f$ messages from distinct nodes within $\vartheta^2\mathcal{S} + \vartheta u$ local time
23: let h' be the local reception time of the $(f+1)$-th such message
24: wait until getH() $\geq h' - d + u - 2\mathcal{S} + T$ or getH() $< h' - (\vartheta^2\mathcal{S} + \vartheta u)$
25: **end if**
26: **end for**

Adding Recovery to Lynch-Welch

Why it works (2):

≥ n-f messages received

=> n-2f > f from sync'ed

=> (f+1)-th within range spanned by sync'ed

=> skew S (from sync'ed) plus δ (meas. error) plus ≈(θ-1)T (drift) < 2S

Algorithm 12 Lynch-Welch pulse synchronization algorithm with recovery mechanism, code for node $v \in V_g$.

1: wait until getH() $\geq \mathcal{S}$ // $H_w(0) \in [0, \mathcal{S})$ for correct nodes w
2: **for all** rounds $r \in \mathbb{N}$ **do**
3: generate r-th pulse
4: $h \leftarrow$ getH()
5: wait until getH() $\geq h + 2\vartheta\mathcal{S}$ or getH() $< h$ // all nodes are in round r
6: broadcast empty message to all nodes (including self)
7: wait until getH() $\geq h + 2(\vartheta^2 + \vartheta)\mathcal{S} + \vartheta d$ or getH() $< h + 2\vartheta\mathcal{S}$ // denote this time by $\tau_{v,r}$
// correct nodes' messages should have arrived
8: **if** received messages from $n - f$ distinct nodes during current loop iteration **then**
9: $h' \leftarrow$ median of $\{h_x \mid h_x$ local reception time of latest message from some $x \in V\}$ (as multiset, i.e., values may repeat)
10: **for** each node $w \in V$ **do**
11: **if** received message from w during current loop iteration **then**
12: let h_w be local time at latest received message from w
13: $\Delta(w) \leftarrow h_w - h - d + u - 2\mathcal{S}$
14: **else**
15: $\Delta(w) \leftarrow h' - h - d + u - 2\mathcal{S}$ // replace missing ones by median
16: **end if**
17: **end for**
18: $U \leftarrow \{\Delta(w) \mid w \in V\}$ (as multiset, i.e., values may repeat)
19: $\Delta \leftarrow \left(U^{(f+1)} + U^{(n-f)}\right)/2$
20: wait until getH() $\geq h + \Delta + T$ or getH() $< h + \Delta - 3\mathcal{S}$
21: **else**
22: wait for $n - f$ messages from distinct nodes within $\vartheta^2\mathcal{S} + \vartheta u$ local time
23: let h' be the local reception time of the $(f+1)$-th such message
24: wait until getH() $\geq h' - d + u - 2\mathcal{S} + T$ or getH() $< h' - (\vartheta^2\mathcal{S} + \vartheta u)$
25: **end if**
26: **end for**

Adding Recovery to Lynch-Welch

Why it works (2):

≥ n-f messages received

=> n-2f > f from sync'ed

=> (f+1)-th within range spanned by sync'ed

=> skew S (from sync'ed) plus δ (meas. error) plus ≈(θ-1)T (drift) < 2S

=> receive all sync'ed mess. on next cycle

=> sync to n-f sync'ed

Algorithm 12 Lynch-Welch pulse synchronization algorithm with recovery mechanism, code for node $v \in V_g$.

1: wait until getH() $\geq \mathcal{S}$ // $H_w(0) \in [0, \mathcal{S})$ for correct nodes w
2: **for all** rounds $r \in \mathbb{N}$ **do**
3: generate r-th pulse
4: $h \leftarrow$ getH()
5: wait until getH() $\geq h + 2\vartheta\mathcal{S}$ or getH() $< h$ // all nodes are in round r
6: broadcast empty message to all nodes (including self)
7: wait until getH() $\geq h + 2(\vartheta^2 + \vartheta)\mathcal{S} + \vartheta d$ or getH() $< h + 2\vartheta\mathcal{S}$ // denote this time by $\tau_{v,r}$
// correct nodes' messages should have arrived
8: **if** received messages from $n - f$ distinct nodes during current loop iteration **then**
9: $h' \leftarrow$ median of $\{h_x \mid h_x$ local reception time of latest message from some $x \in V\}$ (as multiset, i.e., values may repeat)
10: **for** each node $w \in V$ **do**
11: **if** received message from w during current loop iteration **then**
12: let h_w be local time at latest received message from w
13: $\Delta(w) \leftarrow h_w - h - d + u - 2\mathcal{S}$
14: **else**
15: $\Delta(w) \leftarrow h' - h - d + u - 2\mathcal{S}$ // replace missing ones by median
16: **end if**
17: **end for**
18: $U \leftarrow \{\Delta(w) \mid w \in V\}$ (as multiset, i.e., values may repeat)
19: $\Delta \leftarrow \left(U^{(f+1)} + U^{(n-f)}\right)/2$
20: wait until getH() $\geq h + \Delta + T$ or getH() $< h + \Delta - 3\mathcal{S}$
21: **else**
22: wait for $n - f$ messages from distinct nodes within $\vartheta^2\mathcal{S} + \vartheta u$ local time
23: let h' be the local reception time of the $(f+1)$-th such message
24: wait until getH() $\geq h' - d + u - 2\mathcal{S} + T$ or getH() $< h' - (\vartheta^2\mathcal{S} + \vartheta u)$
25: **end if**
26: **end for**

Adding Recovery to Lynch-Welch

Why it works:

< n-f messages received

=> switch to recovery

=> (f+1)-th within range spanned by sync'ed (can't have missed first sync'ed, as then all sync'ed would have been out of window before branch)

Algorithm 12 Lynch-Welch pulse synchronization algorithm with recovery mechanism, code for node $v \in V_g$.

1: wait until getH() $\geq \mathcal{S}$ // $H_w(0) \in [0, \mathcal{S})$ for correct nodes w
2: **for all** rounds $r \in \mathbb{N}$ **do**
3: generate r-th pulse
4: $h \leftarrow$ getH()
5: wait until getH() $\geq h + 2\vartheta\mathcal{S}$ or getH() $< h$ // all nodes are in round r
6: broadcast empty message to all nodes (including self)
7: wait until getH() $\geq h + 2(\vartheta^2 + \vartheta)\mathcal{S} + \vartheta d$ or getH() $< h + 2\vartheta\mathcal{S}$ // denote this time by $\tau_{v,r}$
// correct nodes' messages should have arrived
8: **if** received messages from $n - f$ distinct nodes during current loop iteration **then**
9: $h' \leftarrow$ median of $\{h_x \mid h_x$ local reception time of latest message from some $x \in V\}$ (as multiset, i.e., values may repeat)
10: **for** each node $w \in V$ **do**
11: **if** received message from w during current loop iteration **then**
12: let h_w be local time at latest received message from w
13: $\Delta(w) \leftarrow h_w - h - d + u - 2\mathcal{S}$
14: **else**
15: $\Delta(w) \leftarrow h' - h - d + u - 2\mathcal{S}$ // replace missing ones by median
16: **end if**
17: **end for**
18: $U \leftarrow \{\Delta(w) \mid w \in V\}$ (as multiset, i.e., values may repeat)
19: $\Delta \leftarrow \left(U^{(f+1)} + U^{(n-f)}\right)/2$
20: wait until getH() $\geq h + \Delta + T$ or getH() $< h + \Delta - 3\mathcal{S}$
21: **else**
22: wait for $n - f$ messages from distinct nodes within $\vartheta^2\mathcal{S} + \vartheta u$ local time
23: let h' be the local reception time of the $(f+1)$-th such message
24: wait until getH() $\geq h' - d + u - 2\mathcal{S} + T$ or getH() $< h' - (\vartheta^2\mathcal{S} + \vartheta u)$
25: **end if**
26: **end for**

Adding Recovery to Lynch-Welch

Why it works:

< n-f messages received

=> switch to recovery

=> (f+1)-th within range spanned by sync'ed (can't have missed first sync'ed, as then all sync'ed would have been out of window before branch)

=> also restart loop with skew 2S & recover

Algorithm 12 Lynch-Welch pulse synchronization algorithm with recovery mechanism, code for node $v \in V_g$.

1: wait until getH() $\geq \mathcal{S}$ // $H_w(0) \in [0, \mathcal{S})$ for correct nodes w
2: **for all** rounds $r \in \mathbb{N}$ **do**
3: generate r-th pulse
4: $h \leftarrow$ getH()
5: wait until getH() $\geq h + 2\vartheta\mathcal{S}$ or getH() $< h$ // all nodes are in round r
6: broadcast empty message to all nodes (including self)
7: wait until getH() $\geq h + 2(\vartheta^2 + \vartheta)\mathcal{S} + \vartheta d$ or getH() $< h + 2\vartheta\mathcal{S}$ // denote this time by $\tau_{v,r}$
// correct nodes' messages should have arrived
8: **if** received messages from $n - f$ distinct nodes during current loop iteration **then**
9: $h' \leftarrow$ median of $\{h_x \mid h_x$ local reception time of latest message from some $x \in V\}$ (as multiset, i.e., values may repeat)
10: **for** each node $w \in V$ **do**
11: **if** received message from w during current loop iteration **then**
12: let h_w be local time at latest received message from w
13: $\Delta(w) \leftarrow h_w - h - d + u - 2\mathcal{S}$
14: **else**
15: $\Delta(w) \leftarrow h' - h - d + u - 2\mathcal{S}$ // replace missing ones by median
16: **end if**
17: **end for**
18: $U \leftarrow \{\Delta(w) \mid w \in V\}$ (as multiset, i.e., values may repeat)
19: $\Delta \leftarrow \left(U^{(f+1)} + U^{(n-f)}\right)/2$
20: wait until getH() $\geq h + \Delta + T$ or getH() $< h + \Delta - 3\mathcal{S}$
21: **else**
22: wait for $n - f$ messages from distinct nodes within $\vartheta^2\mathcal{S} + \vartheta u$ local time
23: let h' be the local reception time of the $(f+1)$-th such message
24: wait until getH() $\geq h' - d + u - 2\mathcal{S} + T$ or getH() $< h' - (\vartheta^2\mathcal{S} + \vartheta u)$
25: **end if**
26: **end for**