Ch 9 Goals

- Introduce Byzantine Faults
- Define pulse synchronization
- Show equivalence between solving clock synchronization and pulse synchronization
- Present a fault tolerant pulse synchronization algorithm
- Show basic lower bounds on the fraction of Byzantine faults that can be tolarated.

Byzantine Faults

A **Byzantine** faulty node is a node that may behave arbitrarily.

That is, such a node does not need to follow any algorithm prescribed by the system designer.

An algorithm is **resilient** to f Byzantine faults if its performance guarantees hold for any execution in which there are at most f Byzantine faulty nodes.

In the following, for a network G=(V,E) and a set F of faulty nodes, we denote by V_g the set of correct nodes.

Clock Synchronization – correct nodes

- arbitrary deterministic computations
- computations and message delivery satisfy (known) bounds
- hardware clock runs at rates between 1 and ϑ:

$$t - t' \leq H_v(t) - H_v(t') \leq \vartheta(t - t')$$

Clock Synchronization: compute logical clocks
s.t. for every $v,w \in V_g$, $t \leq t'$

(skew bound) $\max_{v,w \in Vg}\{L_v(t)-L_w(t)\} \leq \mathcal{G}$

$$t - t' \leq H_v(t) - H_v(t') \leq L_v(t) - L_v(t') \leq \boldsymbol{\beta}(t - t')$$

Pulse synchronization goals:

For each $i \in \mathcal{N}$, $v \in V_g$ generate pulse i exactly once,
($p_{v,i}$ is the time when v generates pulse i),
such that there exists S, P_{min}, P_{max}, satisfying:

1) $\sup_{i \in \mathcal{N},\, v,w \in Vg}\{|p_{v,i}-p_{w,i}|\} = S$ (skew)
2) $\inf_{i \in \mathcal{N}}\{\min_{v,\in Vg}\{p_{v,i+1}\}-\max_{v,\in Vg}\{ p_{v,i}\} \} \geq P_{min}$
3) $\sup_{i \in \mathcal{N}}\{\max_{v,\in Vg}\{p_{v,i+1}\}-\min_{v,\in Vg}\{ p_{v,i}\} \} \leq P_{max}$

Thus, **pulses** are **well aligned** and **well separated**

Breakout Room

Exchange ideas how to solve clock synch or pulse synch when facing Byzantine faults

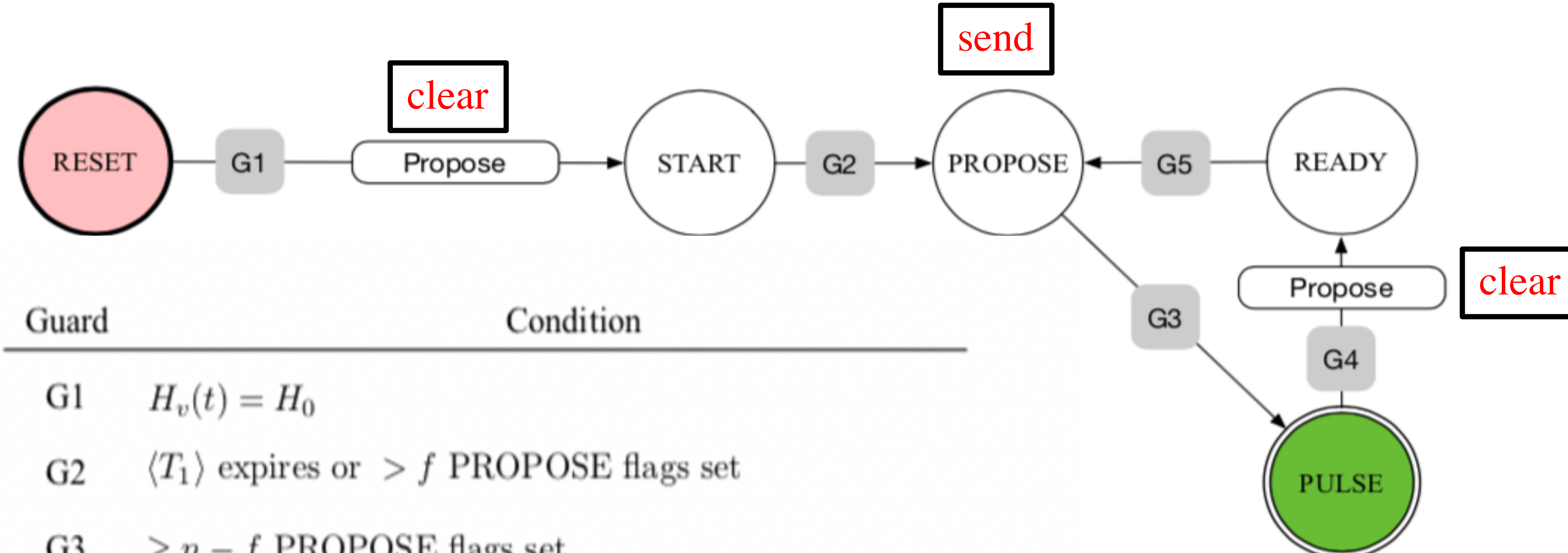
Pulse Synch with 3f < n

- Assume that correct nodes send the same message to all.

- Why 3f+1?
- If I get f+1 I am sure that at least one correct have sent one.
- If I get 2f+1 I am sure that every correct has seen f+1.
- Max number of messages I can wait for is n-f.

- Correct nodes send a simple message “**propose**” to all nodes
- Each node v has a memory flag for every node w, indicating whether v received such a message from w in the current iteration of the loop in the state machine.
- On some state transitions, v will reset all of its flags to 0, indicating that it starts a new iteration locally, in which it has not yet received any propose messages.

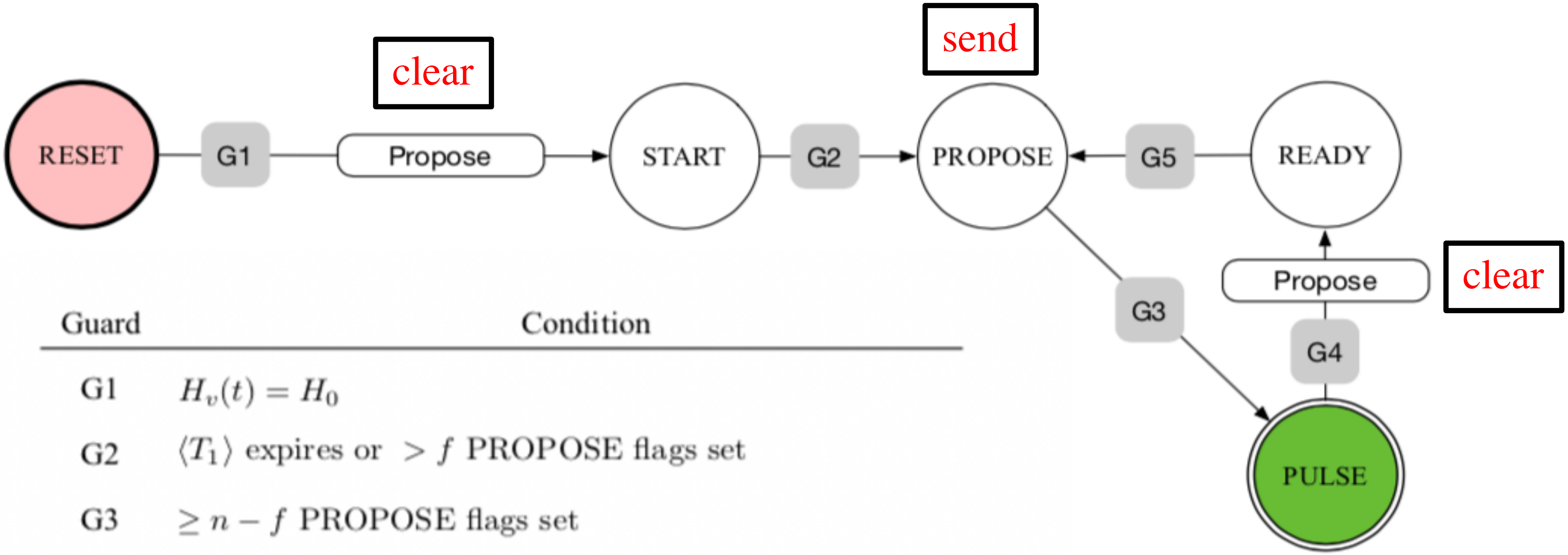
The State Machine - Pulse Synch with 3f < n



Guard	Condition
G1	$H_v(t) = H_0$
G2	$\langle T_1 \rangle$ expires or $> f$ PROPOSE flags set
G3	$\geq n - f$ PROPOSE flags set
G4	$\langle T_2 \rangle$ expires
G5	$\langle T_3 \rangle$ expires or $> f$ PROPOSE flags set

Always consider the **fastest** correct, the **slowest** one and the **byzantine**

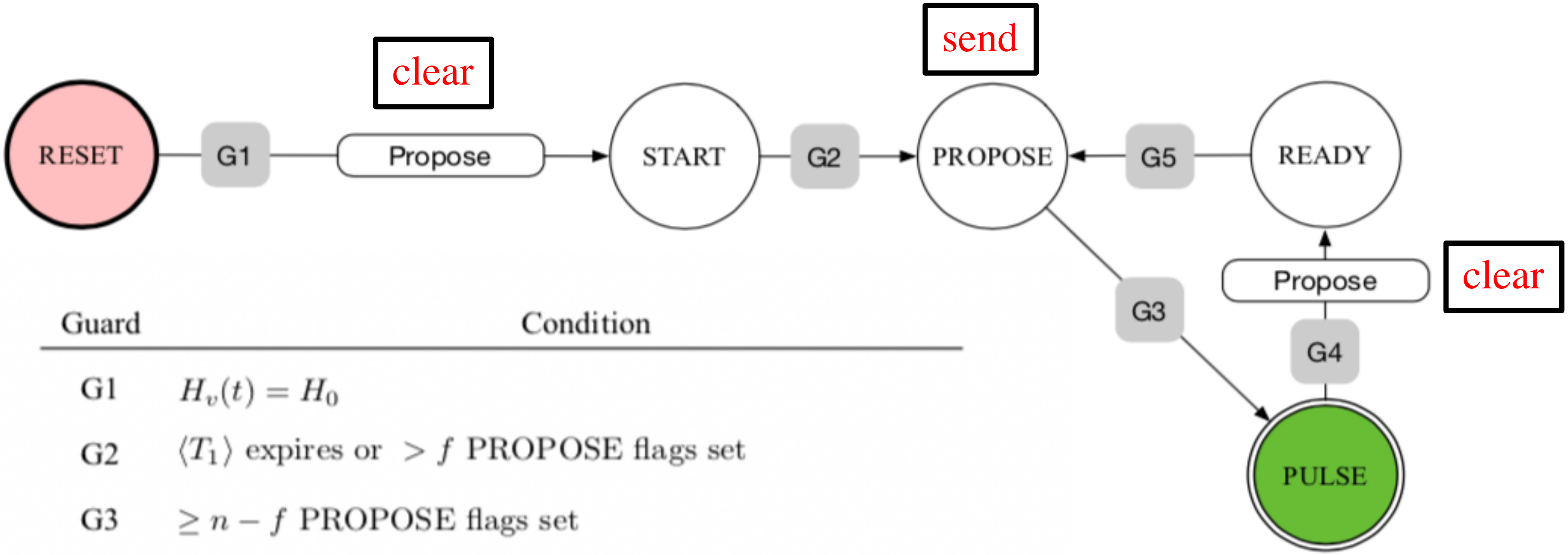
The State Machine - Pulse Synch with 3f < n



Guard	Condition
G1	$H_v(t) = H_0$
G2	$\langle T_1 \rangle$ expires or $> f$ PROPOSE flags set
G3	$\geq n - f$ PROPOSE flags set
G4	$\langle T_2 \rangle$ expires
G5	$\langle T_3 \rangle$ expires or $> f$ PROPOSE flags set

At the beginning of an iteration, all nodes transition to state <u>ready</u> within a bounded time span.
This resets the flags.

The State Machine - Pulse Synch with 3f < n

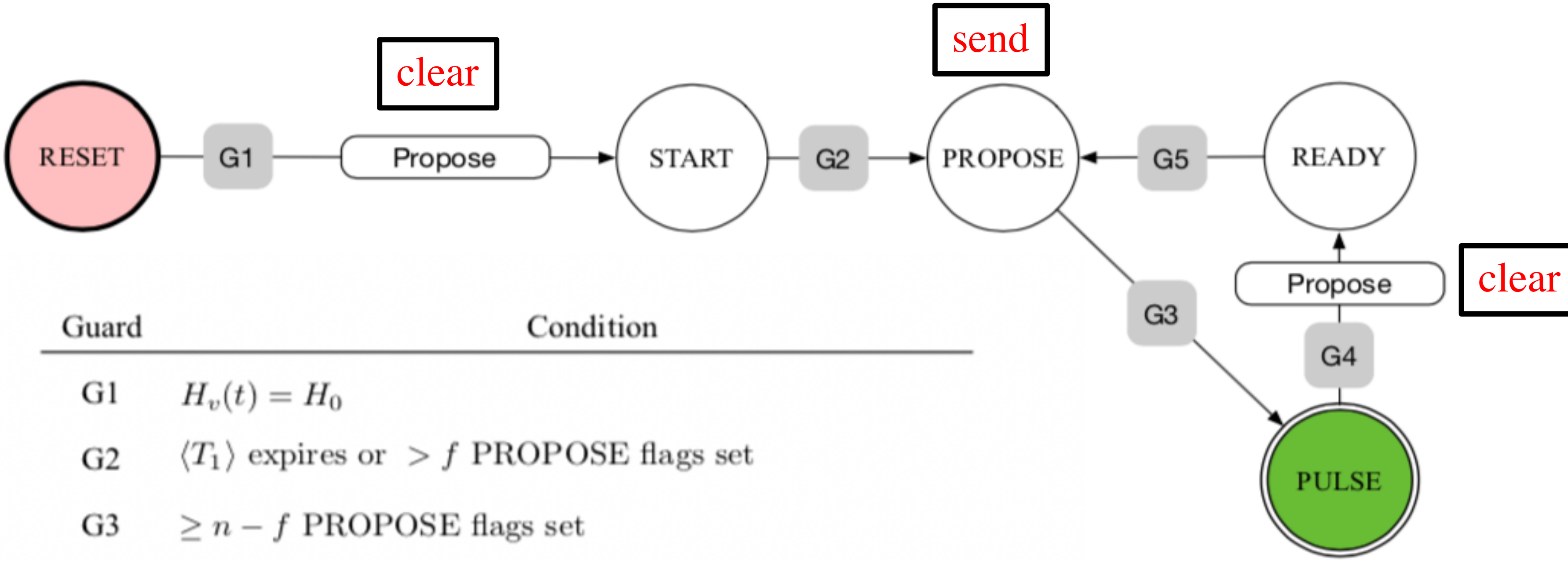


Guard	Condition
G1	$H_v(t) = H_0$
G2	$\langle T_1 \rangle$ expires or $> f$ PROPOSE flags set
G3	$\geq n - f$ PROPOSE flags set
G4	$\langle T_2 \rangle$ expires
G5	$\langle T_3 \rangle$ expires or $> f$ PROPOSE flags set

Nodes wait in state <u>ready</u> until they are sure that all correct nodes reached it.

When a local timeout expires, they transition to <u>propose</u>.

The State Machine - Pulse Synch with 3f < n



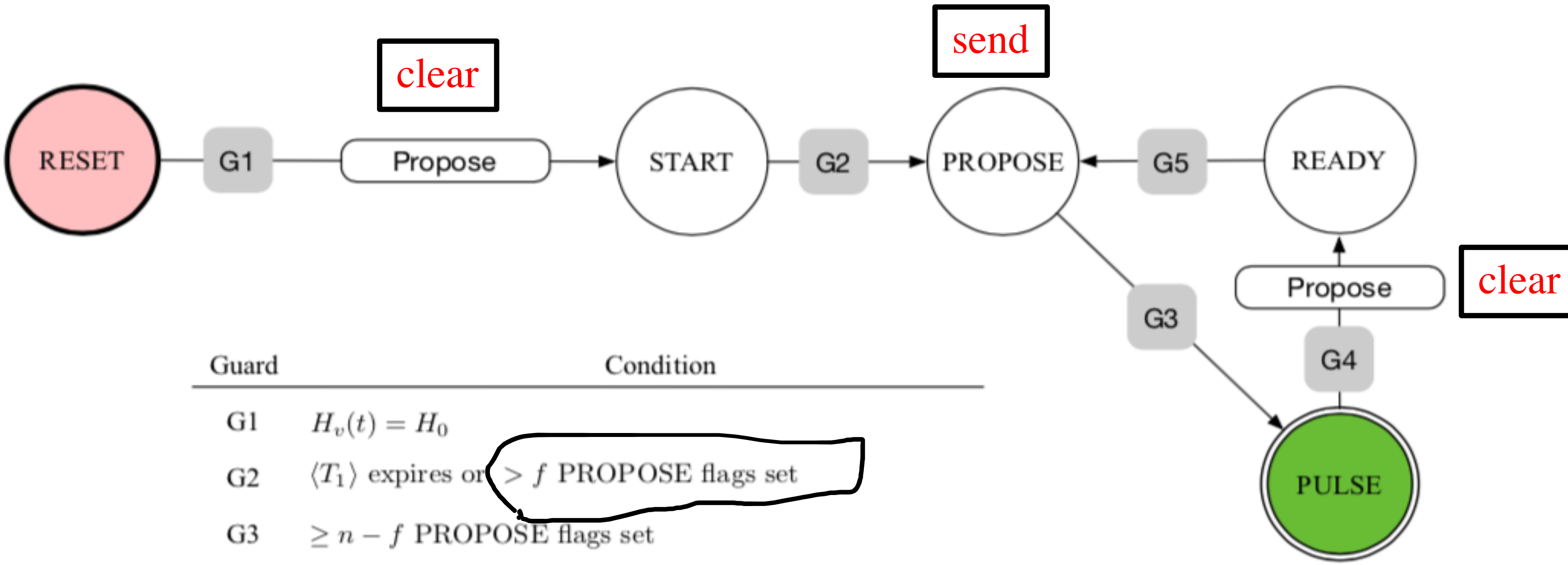
Guard	Condition
G1	$H_v(t) = H_0$
G2	$\langle T_1 \rangle$ expires or $> f$ PROPOSE flags set
G3	$\geq n - f$ PROPOSE flags set
G4	$\langle T_2 \rangle$ expires
G5	$\langle T_3 \rangle$ expires or $> f$ PROPOSE flags set

When it looks like all correct nodes have arrived to <u>propose</u>, they transition to <u>pulse</u>. As the faulty nodes might refuse to send any messages, this means to wait for n-f nodes having announced to be in <u>propose</u>.

Observe

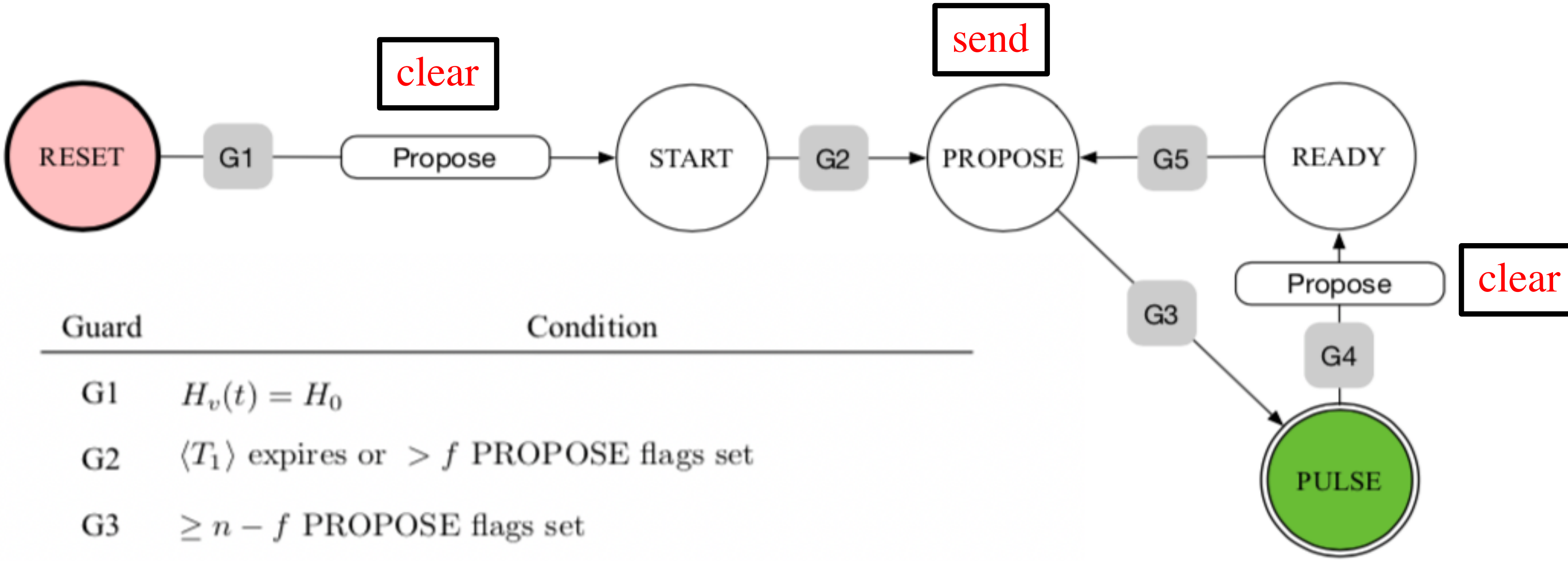
- Faulty nodes may also sent **propose** messages, meaning that the threshold might be reached despite some nodes still waiting in ready for their timeouts to expire. To ``pull'' such stragglers along, nodes will also transition to propose if more than f of their memory flags are set. This is a proof that at least one correct node transitioned to propose due to its timeout expiring, so no ``early'' transitions are caused by this rule.

The State Machine - Pulse Synch with 3f < n



Guard	Condition
G1	$H_v(t) = H_0$
G2	$\langle T_1 \rangle$ expires or $> f$ PROPOSE flags set
G3	$\geq n - f$ PROPOSE flags set
G4	$\langle T_2 \rangle$ expires
G5	$\langle T_3 \rangle$ expires or $> f$ PROPOSE flags set

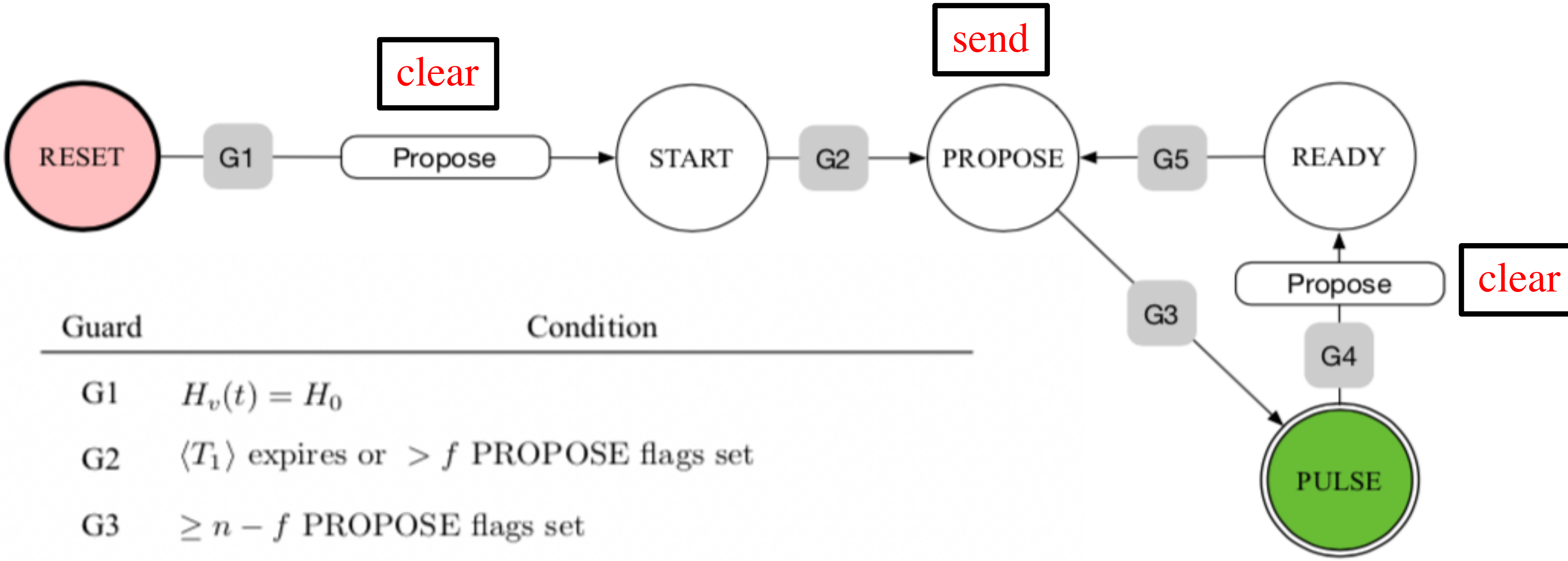
The State Machine - Pulse Synch with 3f < n



Guard	Condition
G1	$H_v(t) = H_0$
G2	$\langle T_1 \rangle$ expires or $> f$ PROPOSE flags set
G3	$\geq n - f$ PROPOSE flags set
G4	$\langle T_2 \rangle$ expires
G5	$\langle T_3 \rangle$ expires or $> f$ PROPOSE flags set

Thus, if any node hits the n-f threshold, no more than d time later each node will hit the f+1 threshold. Another d time later all nodes hit the n-f threshold, i.e., the algorithm has skew 2d.

The State Machine - Pulse Synch with 3f < n



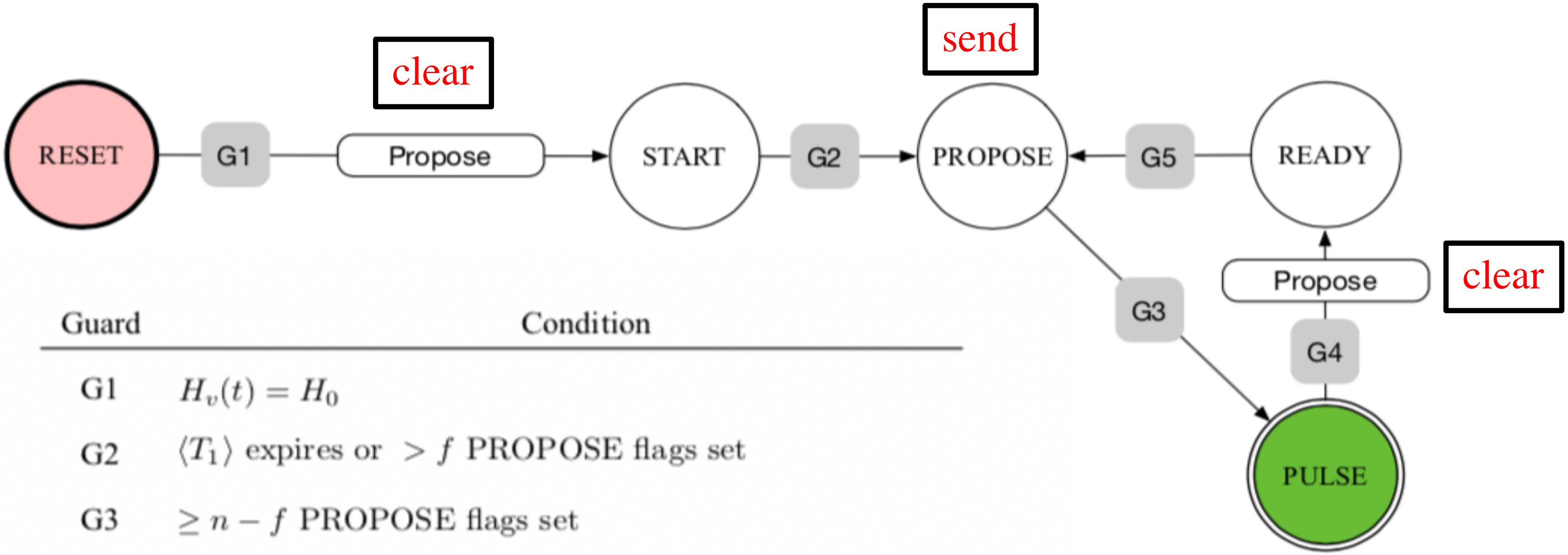
Guard	Condition
G1	$H_v(t) = H_0$
G2	$\langle T_1 \rangle$ expires or $> f$ PROPOSE flags set
G3	$\geq n - f$ PROPOSE flags set
G4	$\langle T_2 \rangle$ expires
G5	$\langle T_3 \rangle$ expires or $> f$ PROPOSE flags set

at time t the first correct moves to <u>pulse</u> (saw n-f)

by t+d, all correct will see f+1 **propose**

by t+2d all correct see n-f and move to <u>pulse</u>

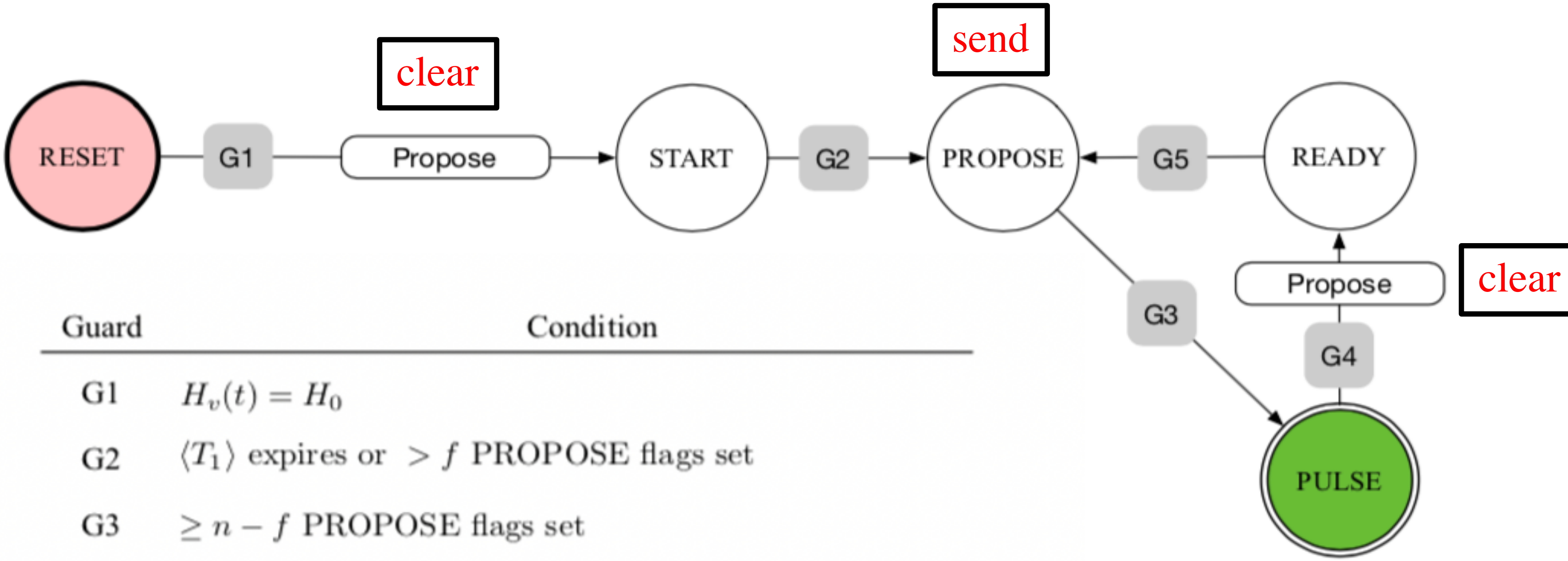
The State Machine - Pulse Synch with 3f < n



Guard	Condition
G1	$H_v(t) = H_0$
G2	$\langle T_1 \rangle$ expires or $> f$ PROPOSE flags set
G3	$\geq n - f$ PROPOSE flags set
G4	$\langle T_2 \rangle$ expires
G5	$\langle T_3 \rangle$ expires or $> f$ PROPOSE flags set

The correct nodes wait in pulse sufficiently long to ensure that no **propose** messages are in transit any more before transitioning to ready and starting the next iteration.

The State Machine - Pulse Synch with 3f < n



Guard	Condition
G1	$H_v(t) = H_0$
G2	$\langle T_1 \rangle$ expires or $> f$ PROPOSE flags set
G3	$\geq n - f$ PROPOSE flags set
G4	$\langle T_2 \rangle$ expires
G5	$\langle T_3 \rangle$ expires or $> f$ PROPOSE flags set

$$H_0 > \max_{v \in V_g}\{H_v(0)\} \quad (9.13)$$

$$\frac{T_1}{\vartheta} \geq H_0 \quad (9.14)$$

$$\frac{T_2}{\vartheta} \geq 3d \quad (9.15)$$

$$\frac{T_3}{\vartheta} \geq \left(1 - \frac{1}{\vartheta}\right) T_2 + 2d \quad (9.16)$$

The challenge: ensuring "cycle" separation despite any Byzantine behavior

The Main Theorem

Theorem 9.17. *Suppose $3f < n$, $H_v(0) \in [0, H_0)$ for all $v \in V$ and some known $H_0 \in \mathbb{R}^+$, and choose any $T \geq 3\vartheta d$. Then we can solve the pulse synchronization problem with $\mathcal{S} = 2d$, $P_{\min} = T$, and $P_{\max} = \vartheta T + (5 + 2(\vartheta - 1))d$, where each node generates its first pulse by time $H_0 + (\vartheta - 1)T + (3 + 2(\vartheta - 1))d$.*

Proof. Set $T_1 := \vartheta H_0$, $T_2 := T$, and $T_3 := (\vartheta - 1)T + 2\vartheta d$. By the assumption that $H_0 > H_v(0)$ for all $v \in V_g$, these choices satisfy Equations (9.13) to (9.16).

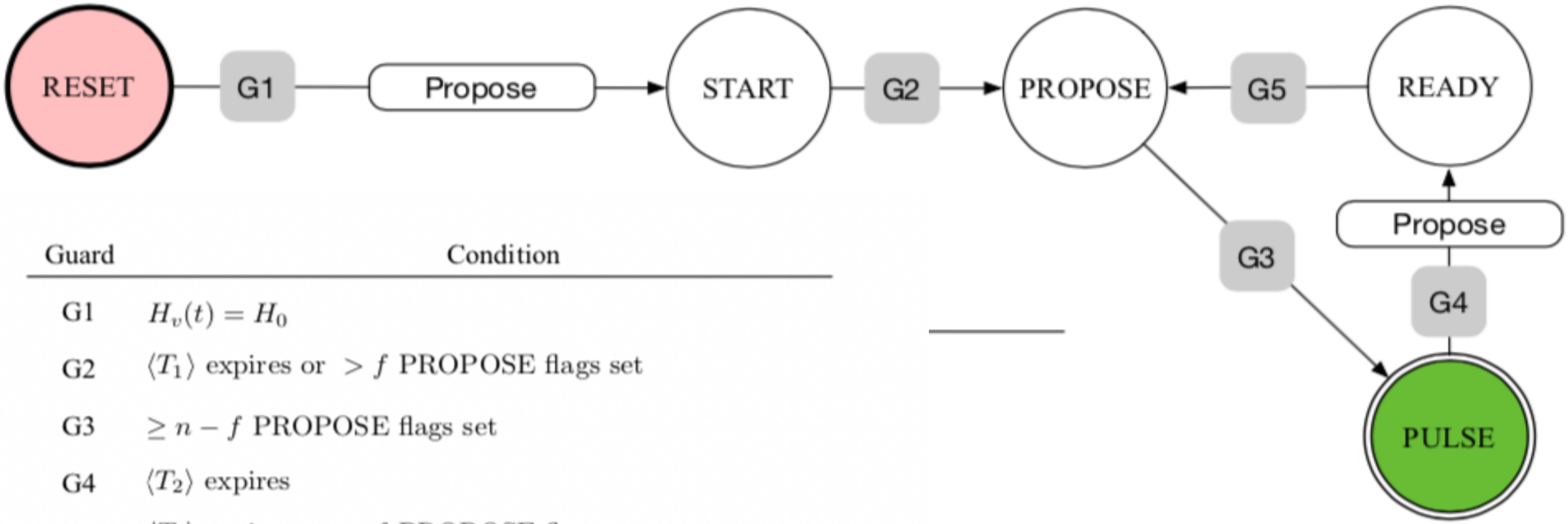
$$H_0 > \max_{v \in V_g}\{H_v(0)\} \tag{9.13}$$

$$\frac{T_1}{\vartheta} \geq H_0 \tag{9.14}$$

$$\frac{T_2}{\vartheta} \geq 3d \tag{9.15}$$

$$\frac{T_3}{\vartheta} \geq \left(1 - \frac{1}{\vartheta}\right) T_2 + 2d \tag{9.16}$$

Claim: from Quite Stage to Coordinated Move



Guard	Condition
G1	$H_v(t) = H_0$
G2	$\langle T_1 \rangle$ expires or $> f$ PROPOSE flags set
G3	$\geq n - f$ PROPOSE flags set
G4	$\langle T_2 \rangle$ expires
G5	$\langle T_3 \rangle$ expires or $> f$ PROPOSE flags set

Assume that when $v \in V_g$ moves to <u>start</u> at time $t_v \in [t-\Delta, t]$ no correct moves to <u>propose</u> during $(t-\Delta-d, t_v)$, and $T_1 \geq \vartheta\Delta$.

Then there exists time $t' \in \left(t - \Delta + \frac{T_1}{\vartheta}, t + T_1 - d\right)$

such that every correct node transition to <u>pulse</u> in

$[t', t' + 2d]$

Proof of the First Claim

- Before the first correct moves from start to propose, all correct are in start
  - all correct are awake before H_0, and $T_1 > \vartheta H_0$
  - the first correct moves due to timeout expiration (T_1)
- d after the first f+1 correct moves to propose, all correct are in propose (or already moved further to pulse)
  - no **propose** message is erased, so all correct get these messages
- Let t' be the time that the first correct moves from propose to pulse.
  - There is such a time.
  - it moves because of n-f propose messages
  - within d every correct receives f+1 and will be in propose, and within another d all correct will see n-f and move to pulse.
  - One can verify that $t' \in \left(t - \Delta + \frac{T_1}{\vartheta}, t + T_1 - d\right)$
- **Similar claim holds for the move from ready to propose.**
- **Thus, essentially we can see that the skew S=2d.**

The Main Theorem (cont.)

Theorem 9.17. *Suppose $3f < n$, $H_v(0) \in [0, H_0)$ for all $v \in V$ and some known $H_0 \in \mathbb{R}^+$, and choose any $T \geq 3\vartheta d$. Then we can solve the pulse synchronization problem with $\mathcal{S} = 2d$, $P_{\min} = T$, and $P_{\max} = \vartheta T + (5 + 2(\vartheta - 1))d$, where each node generates its first pulse by time $H_0 + (\vartheta - 1)T + (3 + 2(\vartheta - 1))d$.*

Proof. Set $T_1 := \vartheta H_0$, $T_2 := T$, and $T_3 := (\vartheta - 1)T + 2\vartheta d$. By the assumption that $H_0 > H_v(0)$ for all $v \in V_g$, these choices satisfy Equations (9.13) to (9.16).

The choice of parameters implies:

$S = 2d$; $T_{min} = T_2$; $T_{max} = T_2 + T_3 + 3d$

We will now argue that the pulse synchronization requirements hold.

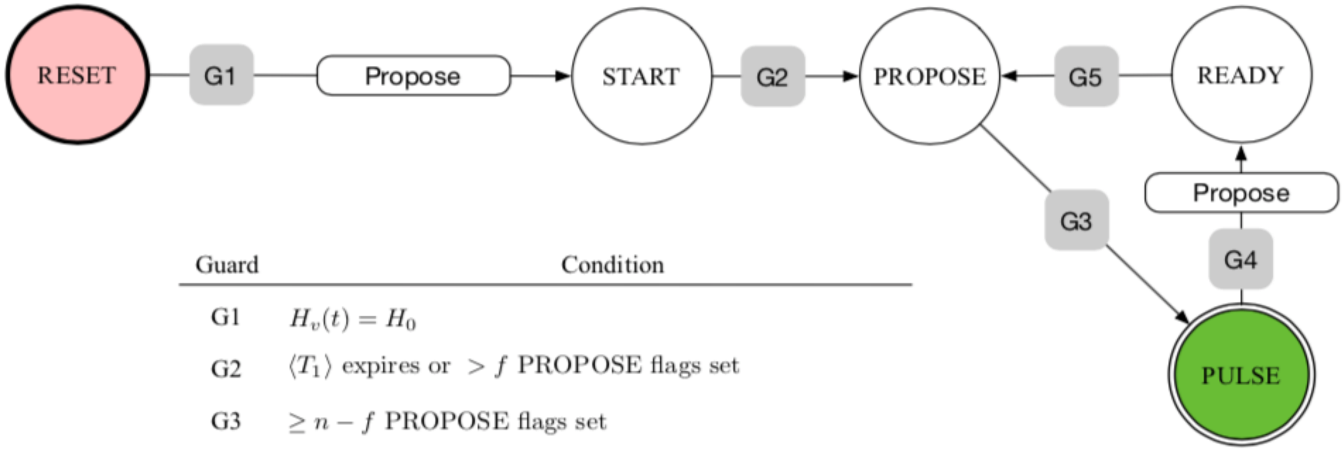
RECALL: Pulse synchronization

For each $i \in \mathcal{N}$, $v \in V_g$ generate pulse i exactly once,
($p_{v,i}$ is the time when v generates pulse i),
such that there exists S, P_{min}, P_{max}, satisfying:

1) $\sup_{i \in \mathcal{N},\, v,w \in Vg}\{|p_{v,i}-p_{w,i}|\} = S$ (skew)
2) $\inf_{i \in \mathcal{N}}\{\min_{v,\in Vg}\{p_{v,i+1}\}-\max_{v,\in Vg}\{ p_{v,i}\} \} \geq P_{min}$
3) $\sup_{i \in \mathcal{N}}\{\max_{v,\in Vg}\{p_{v,i+1}\}-\min_{v,\in Vg}\{ p_{v,i}\} \} \leq P_{max}$

Thus, **pulses** are **well aligned** and **well separated**

The Skew Proof



Guard	Condition
G1	$H_v(t) = H_0$
G2	$\langle T_1 \rangle$ expires or $> f$ PROPOSE flags set
G3	$\geq n - f$ PROPOSE flags set
G4	$\langle T_2 \rangle$ expires
G5	$\langle T_3 \rangle$ expires or $> f$ PROPOSE flags set

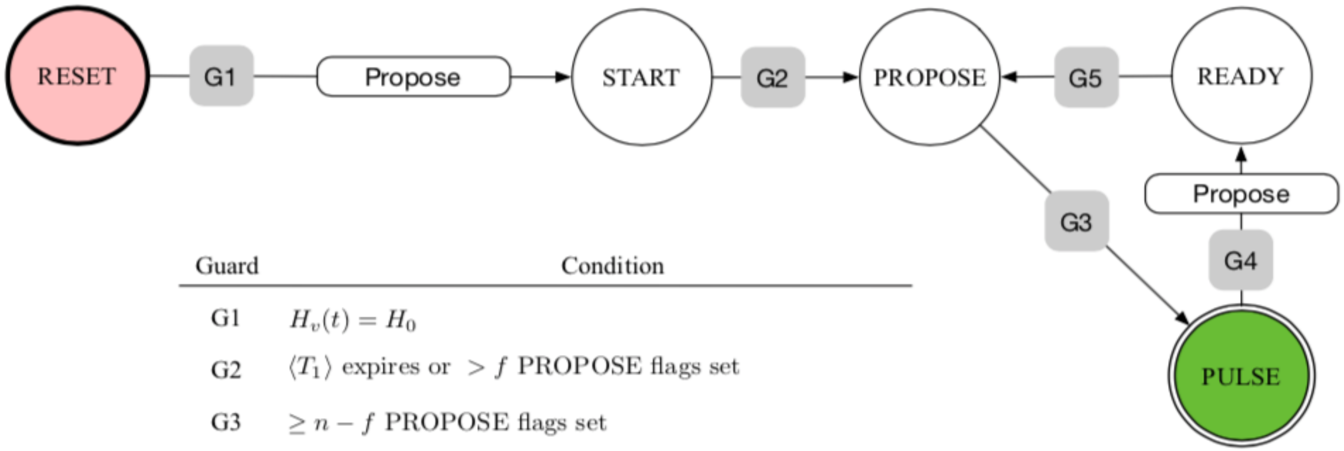
$$S = 2d; P_{min} = T_2 ; P_{max} = T_2 + T_3 + 3d$$

We already proved in the first lemma that all correct nodes join pulse within 2d, given a quite stage. (we just need to choose $H_0 = \Delta$).

Thus, S holds for the first pulse.

Moreover, we can show that the choice of parameters ensures a quite stage before every pulse, therefore, S holds for every iteration.

The P_{min} Proof



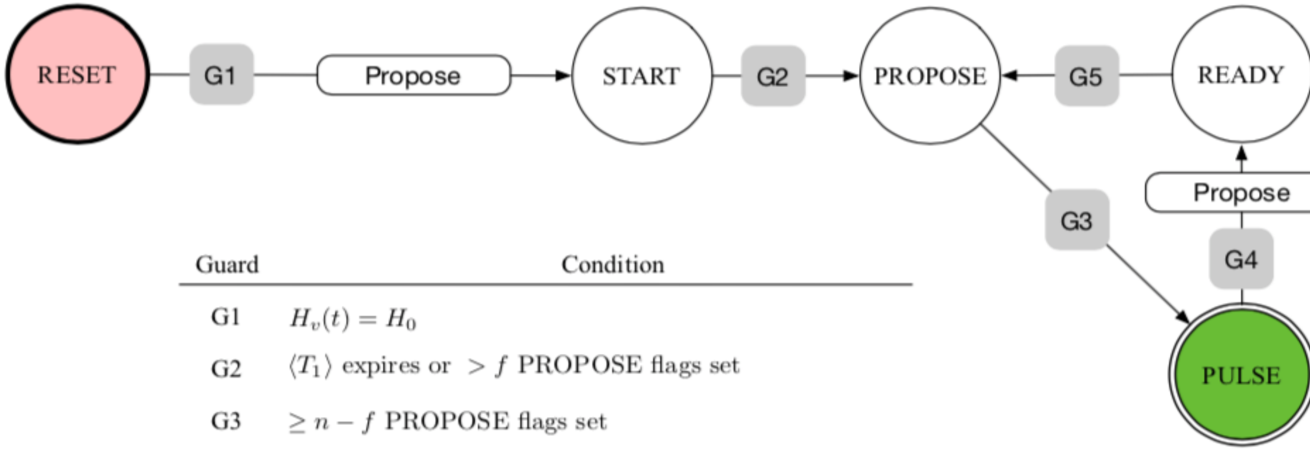
Guard	Condition
G1	$H_v(t) = H_0$
G2	$\langle T_1 \rangle$ expires or $> f$ PROPOSE flags set
G3	$\geq n - f$ PROPOSE flags set
G4	$\langle T_2 \rangle$ expires
G5	$\langle T_3 \rangle$ expires or $> f$ PROPOSE flags set

$$S = 2d;\ P_{min} = T_2\ ;\ P_{max} = T_2 + T_3 + 3d$$

Look at any node leaving pulse. It needs to wait T_2 before moving to ready. So it takes it at lease T_2 before it fires the next pulse.

This essentially proves the P_{min} requirement.

The P_{max} Proof



Guard	Condition
G1	$H_v(t) = H_0$
G2	$\langle T_1 \rangle$ expires or $> f$ PROPOSE flags set
G3	$\geq n - f$ PROPOSE flags set
G4	$\langle T_2 \rangle$ expires
G5	$\langle T_3 \rangle$ expires or $> f$ PROPOSE flags set

$$S = 2d;\ P_{min} = T_2\ ;\ P_{max} = T_2 + T_3 + 3d$$

Let v be first node leaving pulse.

It waits for T_2 to enter ready and not more than T_3 to reach propose.

We know that all nodes entered pulse within 2d. So within 2d more or less after the v reached propose all the correct nodes have send their **propose** message. So within another d, v will see n-f propose and move to pulse.

Thus, it can take it up to $T_2 + T_3 + 3d$ to send the next pulse.

This completes the proof of the theorem.