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 tolerated.
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 $\vartheta$:
  \[ t - t' \leq H_v(t) - H_v(t') \leq \vartheta(t - t') \]

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

(skew bound) \[ \max_{v, w \in V_g} \{L_v(t) - L_w(t)\} \leq G \]

\[ t - t' \leq H_v(t) - H_v(t') \leq L_v(t) - L_v(t') \leq \beta(t - t') \]
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 $S, P_{\text{min}}, P_{\text{max}}$, satisfying:

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

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 $3f < n$

Always consider the **fastest** correct, the **slowest** one and the **byzantine**
At the beginning of an iteration, all nodes transition to state `ready` within a bounded time span. This resets the flags.
Nodes wait in state \texttt{ready} until they are sure that all correct nodes reached it.

When a local timeout expires, they transition to \texttt{propose}.  

\begin{tabular}{|c|c|}
  \hline
  Guard & Condition \\
  \hline
  G1 & $H_v(t) = H_0$ \\
  G2 & $\langle T_1 \rangle \text{ expires or } > f \text{ PROPOSE flags set}$ \\
  G3 & $\geq n - f \text{ PROPOSE flags set}$ \\
  G4 & $\langle T_2 \rangle \text{ expires}$ \\
  G5 & $\langle T_3 \rangle \text{ expires or } > f \text{ PROPOSE flags set}$ \\
  \hline
\end{tabular}
The State Machine - Pulse Synch with $3f < n$

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

- Faulty nodes may also send `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

- **RESET**
  - **G1**: Propose
  - **G2**: START
  - **G3**: PROPOSE
  - **G4**: READY

<table>
<thead>
<tr>
<th>Guard</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>$H_v(t) = H_0$</td>
</tr>
<tr>
<td>G2</td>
<td>$T_1$ expires or $f$ PROPOSE flags set</td>
</tr>
<tr>
<td>G3</td>
<td>$\geq n - f$ PROPOSE flags set</td>
</tr>
<tr>
<td>G4</td>
<td>$T_2$ expires</td>
</tr>
<tr>
<td>G5</td>
<td>$T_3$ expires or $f$ PROPOSE flags set</td>
</tr>
</tbody>
</table>

- **clear**
- **send**
- **clear**
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 $2d$. 

<table>
<thead>
<tr>
<th>Guard</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>$H_v(t) = H_0$</td>
</tr>
<tr>
<td>G2</td>
<td>$\langle T_1 \rangle$ expires or $&gt; f$ PROPOSE flags set</td>
</tr>
<tr>
<td>G3</td>
<td>$\geq n-f$ PROPOSE flags set</td>
</tr>
<tr>
<td>G4</td>
<td>$\langle T_2 \rangle$ expires</td>
</tr>
<tr>
<td>G5</td>
<td>$\langle T_3 \rangle$ expires or $&gt; f$ PROPOSE flags set</td>
</tr>
</tbody>
</table>
at time $t$ the first correct moves to pulse (saw $n-f$)
by $t+d$, all correct will see $f+1$ propose
by $t+2d$ all correct see $n-f$ and move to pulse
The State Machine - Pulse Synch with 3f < n

<table>
<thead>
<tr>
<th>Guard</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>$H_v(t) = H_0$</td>
</tr>
<tr>
<td>G2</td>
<td>$&lt;T_1&gt; \text{ expires or } &gt; f \text{ PROPOSE flags set}$</td>
</tr>
<tr>
<td>G3</td>
<td>$\geq n - f \text{ PROPOSE flags set}$</td>
</tr>
<tr>
<td>G4</td>
<td>$&lt;T_2&gt; \text{ expires}$</td>
</tr>
<tr>
<td>G5</td>
<td>$&lt;T_3&gt; \text{ expires or } &gt; f \text{ PROPOSE flags set}$</td>
</tr>
</tbody>
</table>

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

The challenge: ensuring “cycle” separation despite any Byzantine behavior

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

$$H_0 > \max_{v \in V_g} \{H_v(0)\}$$  \hspace{2cm} (9.13)

$$\frac{T_1}{\theta} \geq H_0$$  \hspace{2cm} (9.14)

$$\frac{T_2}{\theta} \geq 3d$$  \hspace{2cm} (9.15)

$$\frac{T_3}{\theta} \geq \left(1 - \frac{1}{\theta}\right)T_2 + 2d$$  \hspace{2cm} (9.16)
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 $S = 2d$, $P_{\text{min}} = T$, and $P_{\text{max}} = \vartheta T + (5 + 2(\vartheta - 1))d$, 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 $v \in V$, these choices satisfy Equations (9.13) to (9.16).

\[
\begin{align*}
H_0 &> \max_{v \in V_g} \{H_v(0)\} & (9.13) \\
\frac{T_1}{\vartheta} &\geq H_0 & (9.14) \\
\frac{T_2}{\vartheta} &\geq 3d & (9.15) \\
\frac{T_3}{\vartheta} &\geq \left(1 - \frac{1}{\vartheta}\right)T_2 + 2d & (9.16)
\end{align*}
\]
Assume that when \( v \in V_g \) moves to \( \text{start} \) at time \( t_v \in [t-\Delta,t] \) no correct moves to \( \text{propose} \) 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 \( \text{pulse} \) 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 H₀, and T₁ > ϑH₀
  – the first correct moves due to timeout expiration (T₁)
• 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 $T \geq 3\theta d$. Then we can solve the pulse synchronization problem with $S = 2d$, $P_{\min} = T$, and $P_{\max} = \theta T + (5 + 2(\theta - 1))d$, where each node generates its first pulse by time $H_0 + (\theta - 1)T + (3 + 2(\theta - 1))d$.

*Proof.* Set $T_1 := \theta H_0$, $T_2 := T$, and $T_3 := (\theta - 1)T + 2\theta d$. 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.
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 $S$, $P_{\text{min}}$, $P_{\text{max}}$, satisfying:

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

Thus, pulses are well aligned and well separated
The Skew Proof

We already proved in the first lemma that all correct nodes join \textit{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.

\begin{center}
\begin{tabular}{|l|l|}
\hline
Guard & Condition \\
\hline
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 \\
\hline
\end{tabular}
\end{center}

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

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

This essentially proves the $P_{\text{min}}$ requirement.

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