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.
Why Byzantine Faults

Coverage
- no need to worry about a specific fault model

Testing
- saves the need to test whether the assumed fault model holds in practice

Scalability
- increasing system size and clock speed violates previously assumed fault models

Reusability
- moving from one system version to another does not require adapting it to fault variants that may pop up.
Fault Containment Regions

- Faults may crash systems
- A domino effect caused major power failures across USA
- Running a distributed system on a multi core computer does not increase reliability
- Need to identify independent elements, units, or regions, such that a single fault doesn’t propagate beyond that
- Need to continuously obey the assumed ratio of correct to faulty and handle dynamic changes to the system
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 $v, w \in V_g$

\[ H_v(t) - H_v(t') \leq L_v(t) - L_v(t') \leq (1 + \mu)(H_v(t) - H_v(t')) \]

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

We define \[ H_v(t) - H_v(t') \leq L_v(t) - L_v(t') \leq \beta(t - t') \]
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) $\inf_{i \in \mathcal{N}} \{\min_{v,\in V_g} \{p_{v,i+1}\} - \max_{v,\in V_g} \{p_{v,i}\}\} \geq P_{\text{min}}$
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
Basic Observation

Any pulse synchronization algorithm must satisfy:

1) \( P_{\text{max}} - P_{\text{min}} \geq S \)
2) \( P_{\text{max}} \geq \emptyset P_{\text{min}} \)

The first claim can be proved by a simple algebraic manipulation.

The proof of the second claim requires better understanding of the model and the uncertainties within it.
The Timed Message Passing model (TMP)

- Each network node has a local hardware clock

- Nodes actions are deterministic, i.e, actions are a function of the inputs, messages received and the local hardware clock

- There is a bound $d$ on end-to-end message transmission and processing time

- **Unknown elements**: actual hardware clock drift, actual message transmission time, which nodes are faulty and their behavior
A State Machine in TSM

The sequence of messages and outputs depends solely on
1. the initial state and initial input
2. the sequence of messages and inputs it receives
3. hardware clock readings
Due to drift and message transmission time uncertainties, nodes can’t know when a non-local event takes place.
The hardware clock time difference $H(t_3) - H(t_1)$ is bounded by $2d$ and clock drift, i.e., $2d \Theta$
On the left all delays are \( d \) and clock rates are \( 1 \)
On the right all delays are \( \frac{d}{\vartheta} \) and clock rates are \( \vartheta \)
\( H(t_1), H(t_3) \) at \( v \) are the same in both scenarios and \( H(t_2) \) at \( w \) as well
\( H(t_1), \ H(t_3) \) and \( H(t_2) \) are the same --

Therefore –identical messages are being exchanged.

By induction, assuming no faults, throughout the whole algorithm all nodes exchange the same messages in both scenarios.
To prove: $P_{\text{max}} \geq \vartheta P_{\text{min}}$

Observe that the values are external time values. Any pulse difference time in the left scenario is divided by $\vartheta$ on the other scenario.

Therefore, the value of $P_{\text{max}} / \vartheta$ needs also be $\geq P_{\text{min}}$
Clock Synchronization to Pulse Synchronization

- Assume we have a fault tolerant clock synch alg with parameters $\mathcal{G}$ and $\beta$

- We show how to construct pulse synchronization with parameters

\[
\begin{align*}
S &= \mathcal{G} \quad \text{(skew)} \\
P_{\text{min}} &= \frac{(T - \mathcal{G})}{\beta} \quad \text{(min period)} \\
P_{\text{max}} &= T + \mathcal{G} \quad \text{(max period)}
\end{align*}
\]

for any choice of $T$ satisfying $T > \mathcal{G}$
Pulse algorithm

Assume – \( L_v(0) \in [0, G] \) for all \( v \in V_g \)

1. \( i := 0 \) (performed only on wakeup)
2. While true do
3. wait until getL() = iT
4. generate the i-th pulse
5. \( i := i + 1 \)
6. end while

\( v \) generates its i-th pulse at a unique time \( p_{v,i} \) satisfying \( L_v(p_{v,i}) = iT \).

Notice that faults do not affect the algorithm.
Figure 9.2
Relation between pulse times and logical clock at node $v \in V_g$ for Algorithm 10. Note that the logical clock rate varies between 1 and $\beta$, where typically $\beta - 1 \ll 1$. Hence the real time between pulses fluctuates slightly.
The Skew

Assume – \( L_v(0) \in [0, G] \) for all \( v \in V_g \)

1. \( i := 0 \) \hspace{1cm} \text{(performed only on wakeup)}
2. While true do
3. \hspace{1cm} \text{wait until getL()} = iT
4. \hspace{1cm} \text{general i-th pulse}
5. \hspace{1cm} i := i +1
6. end while

i-th pulse: \( v \) at \( L_v(p_{v,i}) = iT \) and \( w \) at \( L_w(p_{w,i}) = iT \)

\( S = G \), because \( G \) bounds the logical clocks difference
The Min – Max Periods

We prove just one of them, since both are a simple derivation

We have that

\[ L_v(p_{w,i} + G + T) \geq L_v(p_{w,i}) + G + T \geq L_w(p_{w,i}) + T \]

\[ = (i + 1)T = L_v(p_{v,i+1}), \]

implying that \( p_{v,i+1} \leq p_{w,i} + G + T \). Hence, for each \( i \in \mathbb{N} \) it holds that

\[ \max_{v \in V_g} \{ p_{v,i+1} \} - \min_{v \in V_g} \{ p_{v,i} \} \leq G + T, \]

as claimed. \( \square \)
Pulse Synchronization to Clock Synchronization

- We now assume we have a fault tolerant pulse synch alg with parameters $S$, $P_{\text{min}}$ and $P_{\text{max}}$

- We show how to construct clock synchronization with parameters

- $\beta = \varnothing^2 P_{\text{max}} / P_{\text{min}}$

- $G = (\varnothing - 1) P_{\text{max}} + \beta S$
Algorithm 8 Clock synchronization algorithm at \( v \in V_g \) based on a pulse synchronization algorithm.

1: wait until initialization pulse
2: \( \ell \leftarrow 0 \) \hspace{1cm} \triangleright \) initialize logical clock
3: \( i \leftarrow 0 \) \hspace{1cm} \triangleright \) pulse number
4: \( \ell^\uparrow \leftarrow 0 \) \hspace{1cm} \triangleright \) clock increase to be amortized during current pulse
5: \( h \leftarrow \text{getH()} \)
6: \textbf{while true do}
7: \hspace{1cm} wait until next pulse
8: \hspace{1cm} \ell \leftarrow i \partial P_{\text{max}} + \text{getH()} - h \hspace{1cm} \triangleright \) logical clock value at pulse
9: \hspace{1cm} i \leftarrow i + 1
10: \hspace{1cm} \ell^\uparrow \leftarrow i \partial P_{\text{max}} - \ell \hspace{1cm} \triangleright \) difference to target value
11: \hspace{1cm} h \leftarrow \text{getH()}
12: \textbf{end while}
13: \textbf{procedure} \text{getL()} \hspace{1cm} \triangleright \) returns \( L_v(t) \) when called at time \( t \geq p_{v,0} \)
14: \hspace{1cm} \text{return} \( \ell + \text{getH()} - h + \ell^\uparrow \cdot \min \left\{ \frac{\text{getH()} - h}{P_{\text{min}}}, 1 \right\} \)
15: \textbf{end procedure}

\[
\ell^\uparrow_v(p_{v,i}) = \partial P_{\text{max}} - (H_v(p_{v,i}) - H_v(p_{v,i-1})).
\]
The Drift Bound

- We have a fault tolerant pulse synch alg with parameters $S, P_{\text{min}}$ and $P_{\text{max}}$

- The rate of the logical clock we produce is bounded from above by the drift of the hardware clocks ($\vartheta$) multiplied by the drift caused by the amortization of the extra $P_{\text{max}}$ we add at each pulse.

- We amortize it over $P_{\text{min}}$ time, which implies a factor of $\vartheta \frac{P_{\text{max}}}{P_{\text{min}}}$.

- So the resulting bound is $\beta = \vartheta^2 \frac{P_{\text{max}}}{P_{\text{min}}}$
The Skew Bound

- We have a fault tolerant pulse synch alg with parameters $S$, $P_{\text{min}}$ and $P_{\text{max}}$

- The drift rate is $\beta = \varpi^2 \frac{P_{\text{max}}}{P_{\text{min}}}$

- If we compare the logical clocks of two nodes the difference is a result of the pulse skew and of having them, for some time, in separate pulse index. Reading the proof one can see that we obtain:

- $G = (\varpi - 1) P_{\text{max}} + \beta S$