1 Topic

In this lecture, the professor discussed the paper: *Gradient Clock Synchronization* by Rui Fan and Nancy Lynch in PODC’04, July 25-28, 2004, St. John’s, Newfoundland, Canada.

2 Background

First, let us consider the target tracking problem in a sensor network. Suppose two sensor nodes want to measure the speed of an object. Each node records the time when the object crosses within its vicinity. Then the nodes exchange their time readings, and compute \( t \), the difference in their readings. The amount of error in \( t \) is related to the clock skew between the nodes. The object’s velocity is computed as \( v = \frac{d}{t} \), where \( d \) is the known Euclidean distance between the nodes. Suppose the nodes do not need to compute \( v \) exactly, but only to an accuracy of 1%. Since \( v = \frac{d}{t} \), then the larger the Euclidean distance is between the nodes, the more error is acceptable in \( t \), while still computing \( v \) to 1% accuracy. Thus, the acceptable clock skew of the nodes forms a gradient. Note that here we are assuming the Euclidean distance between two nodes corresponds to the uncertainty in their message delay. This is the case if, for example, there are multiple network hops between the nodes, with the number of hops proportional to the Euclidean distance between the nodes.

\( N \) is a network with \( n \) nodes \( P_1, P_2, \ldots, P_n \) and is complete so that all nodes in this network can communicate with each other. Let \( d_{ij} \) be the distance between two nodes \( P_i \) and \( P_j \) and \( D \) be the diameter of the network \( N \) (i.e. \( D = \max \{d_{ij}\} \)). The main result from this paper is that given any clock synchronization algorithm, there exists an execution in which two nodes that are distance \( d \) apart, for an arbitrary \( d \), have \( \Omega(d + \frac{\log D}{\log \log D}) \) clock skew. From the result, we can see that both the distance \( d \) and the diameter \( D \) determine the time skew of the two nodes.

Lundelius-Welch and Lynch [1984] proved that in a complete network of \( n \) nodes where the distance between each pair of nodes is \( d \), nodes cannot synchronize their clocks to closer than \( d(1 - \frac{1}{n}) \) and gave \( \Omega(D) \) as the lower bound of the clock skew. Srikant and Toueg [1987] gave an optimal clock synchronization algorithm, where optimal means that the maximum skew between any pair of nodes is as small as possible given the hardware clock drift. Their algorithm ensures that any pair of nodes have \( O(D) \) clock skew. However, it does not guarantee a gradient in the clock skew, because even nodes that are \( O(1) \) distance apart can have \( O(D) \) skew.
3 The Clock Synchronization Lower Bound

Now we prove the lower bound. We first prove a simpler lower bound in the case of 2 processors.

Definition 1 Given an execution $\alpha$ of a CS algorithm $A$ for two processors, the execution $\alpha' = shift(\alpha, < a_0, a_1 >)$ where every event of $P_0$ is shifted later by $a_0$ and every event of $P_1$ is shifted later by $a_1$. $L_i(t)$ is the local clock time on $P_i$ when the global time is $t$.

Theorem 1 Consider two processors, $P_0$ and $P_1$ at distance $d$ apart (so message delay is in $[0, d)$), the worst case clock skew of any algorithm is $\Omega(d)$.

Proof: Please refer to Figure 1. Let $A$ be a clock synchronization algorithm and $\alpha$ be an execution of $A$, in which the delay of all messages from $P_0$ to $P_1$ is 0 and $P_1$ to $P_0$ is $d$. Let $\alpha'$ be an execution of $A$ and defined as $shift(\alpha, < -d, 0 >)$, in which the delay of all messages from $P_0$ to $P_1$ is $d$ and $P_1$ to $P_0$ is 0. Suppose algorithm $A$ has clock skew $\epsilon$ (i.e. $|L_0(t) - L_1(t)| \leq \epsilon$).

Execution $\alpha'$ is similar to execution $\alpha$ except as follows. All events still happen at the same local times at both processors. $P_0$’s local clock moves ahead by $d$ with respect to global time. So, all events at $P_0$ are shifted earlier by real time $d$ in $\alpha'$. $P_1$’s local clock times and events
stay at the same global time as in $\alpha$. In order to disable $P_1$’s abilities to distinguish between these two executions, $P_0$ should send the message to $P_1$ with delay time of $d$ at global time $t_1$. Otherwise, $P_1$ will receive the message from $P_0$ at a local time that is $d$ time earlier than it does in execution $\alpha$. Similarly, $P_1$ should send a reply message to $P_0$ with delay time of $0$ such that $P_0$ can receive the message from $P_1$ at the local time of $T_0 + d$. As we can see in execution $\alpha'$, the delay time is in the range of $[0, d]$ so $\alpha'$ is legal. $P_0$ is in final state after the global time $t_3$ and $P_1$ is in final state after the global time $t_2$ so both processors are in final states at the global time $t$. At global time $t$ in execution $\alpha$, the algorithm $A$ has clock skew $\epsilon$ (i.e. $|L_0(t) - L_1(t)| \leq \epsilon$). At global time $t$ in execution $\alpha'$, the algorithm $A$ has clock skew $\epsilon$ (i.e. $|L_0(t) + d - L_1(t)| \leq \epsilon$). So we can get the inequalities as follows:

\[
\begin{align*}
L_1(t) &\leq L_0(t) + \epsilon, \quad \text{for execution of } \alpha \\
L_0(t) + d &\leq L_1(t) + \epsilon, \quad \text{for execution of } \alpha' \\
\Rightarrow \quad L_0(t) + d &\leq L_0(t) + 2\epsilon \\
\Rightarrow \quad d &\leq 2\epsilon \\
\Rightarrow \quad \epsilon &\geq \frac{d}{2}
\end{align*}
\]

So the lower bound of the clock skew is $\Omega(d)$.

Now, we show the general lower bound for $n$ processors.

**Theorem 2** Given a set of $n$ processors that are at distance $d$ from each other, the worst case clock skew is $\Omega(d(1 - \frac{1}{n}))$.

**Proof:** Let $A$ be a CS algorithm. Let $\alpha$ be an execution of $A$ in which the delay of all messages from $p_i$ to $p_j$ is $0$, if $i < j$ and $d$, if $i > j$. Let algorithm $A$ have skew $\epsilon$ and $L_i(t)$ is the local clock time on $P_i$ when the global time is $t$.

\[
\begin{align*}
L_{n-1}(t) &\leq L_0(t) + \epsilon \\
&\leq L_1(t) - d + 2\epsilon \quad \text{by Lemma 3} \\
&\leq L_2(t) - 2d + 3\epsilon \quad \text{by Lemma 3} \\
&\cdots \\
&\leq L_{n-1}(t) - (n - 1)d + n\epsilon \\
(n - 1)d &\leq n\epsilon \\
\epsilon &\geq (1 - \frac{1}{n})d
\end{align*}
\]

**[Homework 4]** This proof is completed using the following lemma. Prove the lemma and then complete the proof.

**Lemma 3** For any $k$, $1 \leq k \leq n - 1$, $L_{k-1}(t) \leq L_k(t) - d + \epsilon$ where $L_i(t)$ is the local clock time on $P_i$ when the global time is $t$. 
Figure 2: Algorithm 1 in the case where $\delta = d$, the maximum delay.

4 Simple Clock Synchronization Algorithms

Let us consider the algorithms from Srikant and Toueg [1987]. Algorithm 1 solves the clock synchronization problem for two processors when the maximum delay $d$ is known. Algorithm 2 solves the clock synchronization problem for two processors when the maximum delay $d$ is unknown. In both cases, the actual delay of any message is between 0 and $d$.

**Algorithm 1** A CS algorithm in which the distance $d$ is known.

1: $P_0$ sends a message with time stamp $T_0$ to $P_1$ at global time $t_1$

2: when $P_1$ receives the message from $P_0$ at global time $t_2$, $P_1$ sets its local time as $T_0 + \frac{d}{2}$

4.1 Analysis of Algorithm 1

Suppose the message from $P_0$ to $P_1$ takes time $\delta$. Then, at global time $t_2 = t_1 + \delta$, $L_0(t_2) = T_0 + \delta$, while $L_1(t_2) = T_0 + d/2$. Since $0 \leq \delta \leq d$, $L_0(t_2) = T_0 + \delta$ and $L_1(t_2) = T_0 + d/2$ are at most $d/2$ apart. So, the clock skew between $P_0$ and $P_1$ is $d/2$ in the worst case. Figure 2 illustrates the case where the message delay is exactly $d$ and the clock skew is $d/2$. Also, if message delay were 0, the clock skew would still be $d/2$. On the other hand, if message delay were $d/2$, the clock skew would be 0.
Algorithm 2 A CS algorithm in which the distance \( d \) is unknown.

1: \( P_0 \) sends a "hello" message to \( P_1 \)
2: when \( P_1 \) receives "hello" message at local time \( T_1 \), \( P_1 \) sends a reply message with time stamp \( T_1 \) to \( P_0 \)
3: when \( P_0 \) receives the message from \( P_1 \), \( P_0 \) sets \( \sigma \) to be the difference in \( P_0 \)'s local time from step 1 to step 3 and sets its local time as \( T_1 + \frac{\sigma}{2} \)

4.2 Analysis of Algorithm 2

In order to compute the clock skew between \( P_0 \) and \( P_1 \), we need to consider the four extreme cases that have been illustrated in Figure 3.

For example:
In Figure 3(A), the value of \( L_0(t_2) \) is equal to \( T_1 + \frac{\sigma}{2} \) (i.e. \( T_1 + d \)) and the value of \( L_1(t_2) \) is equal to \( T_1 + d \), so the clock skew between \( P_0 \) and \( P_1 \) at global time \( t_2 \) is 0.

We can extend the two processors algorithm to get the following result:

Theorem 4 Given \( n \) processors and messages with delay in range \([0, d]\), there is a CS algorithm with clock skew \( d(1 - \frac{1}{n}) \).

Each node \( i \) is equipped with a hardware clock. We define the value of the hardware clock in terms of its rate of change. Specifically, we denote \( i \)'s hardware clock rate at real time \( t \) of an execution \( \alpha \) by \( h^\alpha_i(t) \). We define \( i \)'s hardware clock value at time \( t \) in \( \alpha \) to be \( H^\alpha_i(t) = \int_0^t h^\alpha_i(t)dr \). We assume all the hardware clocks have bounded drift. That is, we assume that there exists a constant \( \rho \), where \( 0 \leq \rho < 1 \), such that for any execution the following holds.

\[ \forall \forall t : 1 - \rho \leq h^\alpha_i(t) \leq 1 + \rho \]

Let us describe a validity condition which we require any CSA to satisfy. Let \( A \) be a CSA, and consider any execution \( \alpha \) of \( A \). Then we require the following.

\[ \forall \forall t \forall r > 0 : \frac{r}{2} \leq L^\alpha_i(t + r) - L^\alpha_i(t) \]

where \( L^\alpha_i(t) \) is denoted as the logical clock value of node \( i \) at time \( t \) in an execution \( \alpha \). This requirement says that the rate of increase of each node’s logical clock must be at least \( \frac{1}{2} \), at all times.

\[ \frac{dL^\alpha_i(t)}{dt} \geq \frac{1}{2} \]
(1) \( L_0(t_2) = T_1 + \sigma/2 = T_1 + d \)
(2) \( L_1(t_2) = T_1 + d \)
So clock skew is 0

(A)

(1) \( L_0(t') = T_1 + \sigma/2 = T_1 + d/2 \)
(2) \( L_1(t') = T_1 + d \)
So clock skew is \( d/2 \)

(B)

(1) \( L_0(t') = T_1 + \sigma/2 = T_1 + d/2 \)
(2) \( L_1(t') = T_1 \)
So clock skew is \( d/2 \)

(C)

(1) \( L_0(t') = T_1 \)
(2) \( L_1(t') = T_1 \)
So clock skew is 0

(D)
**Definition 2** (Gradient Property). Let \( n \) be any positive integer, and let \( N \) be any network with nodes \( 1, \ldots, n \), and distances between the nodes \( \{d_{i,j} | 1 \leq i, j \leq n \} \). In every execution \( \alpha \) of \( A \) in \( N \), we have

\[
\forall \forall i, j : |L_i^\alpha(t) - L_j^\alpha(t)| \leq f(d_{i,j}).
\]

**Theorem 5** For any \( f - GCG \) algorithm, \( f(d) = \Omega(d + \frac{\log D}{\log \log D}) \).

**Proof:** Let \( A \) be an arbitrary \( f \)-GCS algorithm. To prove the lower bound, we show the following:

1. For every real number \( d \geq 1 \), there exists a network containing two nodes at distance \( d \) from each other, such that the two nodes have \( \Omega(d) \) clock skew in some execution of \( A \) in the network. This implies \( f(d) = \Omega(d) \). The reason why we only consider \( d \geq 1 \) is because we defined 1 as the minimum distance between any two nodes, for each network.

2. For every integer \( D \geq 1 \), there exists a network of diameter \( D \) and an execution of \( A \) in the network, such that two nodes at distance 1 from each other in the network have \( \Omega(\frac{\log D}{\log \log D}) \) clock skew in the execution. This implies \( f(1) = \Omega(\frac{\log D}{\log \log D}) \).

\]

[**Homework 5**] In a network which is a chain of \( n \) processors \( p_1, p_2, \ldots, p_n \), Fan and Lynch show that the worst case clock skew between neighbors depends on the size of \( n \), even though the distance between them is 1!. Consider the following algorithm:

At real time \( t \), neighboring nodes \( p_i \) and \( p_{i+1} \) send their local times \( L_i(t) \) and \( L_{i+1}(t) \) to each other. When they receive the message, each node sets its clock to the larger of its own clock value and the received timestamp.

Prove that at real time \( t+1 \), the local clocks of \( p_i \) and \( p_{i+1} \) differ by at most 1. Argue why this does not contradict the lower bound by Fan and Lynch.