Chapter 9

Clock Synchronization

9.1 Slide 9/32

Theorem 9.1. *No matter what clock synchronization algorithm we run, the skew between two neighboring clocks may always be* $\Omega(\alpha \log_{\frac{\beta-\alpha}{\alpha\varepsilon}} D)$*, where* D *is the diameter of the network, hardware clocks have a rate between* $1-\varepsilon$ *and* $1+\varepsilon$ *(worst case), message delay is between* 0 *and* 1 *(worst case), and logical clocks must run at least at rate* α*, and at most at rate* β*. (On the slides we assumed that* $\alpha = 1$*.)*

Proof. (Sketch) The proof is on a chain of $D+1$ nodes $v_1, v_2, \ldots, v_{D+1}$; we set $l_0 := D$.[1] Assume that the nodes run their algorithm for time $T_0 := \frac{1+\varepsilon}{4\varepsilon} l_0 \leq \frac{l_0}{2\varepsilon}$, all nodes have a hardware clock rate of 1, and all messages are delayed for 1/2 time. This situation is *indistinguishable* for the nodes from a situation where the nodes $v_1, v_2, \ldots, v_{D+1}$ have hardware clock rates $1+\varepsilon, 1+\varepsilon-\varepsilon/l_0, 1+\varepsilon-2\varepsilon/l_0, \ldots, 1$ if we adapt message delays accordingly, i.e., "down" messages are slower than "up" messages. Since the difference between the hardware clock rates between neighbors is exactly $\frac{\varepsilon}{l_0}$ and $T_0 \leq \frac{l_0}{2\varepsilon}$, we need to modify the message delays by at most $\frac{\varepsilon}{l_0} \cdot \frac{l_0}{2\varepsilon} = 1/2$, i.e., all message delays are still in the valid range of $[0,1]$.[2]

Since the fastest node is running $1+\varepsilon$ times faster than in the original execution, and the executions are indistinguishable, it reaches the logical clock value that it reached at time T_0 already at time $T_0' := \frac{l_0}{4\varepsilon}$. Since the slowest node still runs at rate 1, it reaches the same logical clock value at time T_0' in both executions. As the fastest node increased its logical clock at least at rate α in the interval $T_0 - T_0' = \frac{l_0}{4}$, the clock skew between the fastest and the slowest node increased by at least $\frac{\alpha}{4} l_0$ until time T_0'.

Now, in a second phase, we give the nodes time to adapt again, starting at time T_0'. Assume that the nodes continue to run their algorithm for $T_1 := \frac{\alpha(1+\varepsilon)}{16(\beta-\alpha)} l_0 \leq \frac{\alpha}{8(\beta-\alpha)} l_0$ time, all nodes have a hardware clock rate of 1, and messages again take time 1/2. Since the lagging bottom node can run at most at rate β, and the top node must run at least at rate α, the clock skew between

[1] The proof also works on general graphs.

[2] In this short summary we will not prove this formally, but we encourage the reader to verify it with an example.

these nodes reduces by at most $(\beta-\alpha)\cdot\frac{\alpha}{8(\beta-\alpha)}l_0 = \frac{\alpha}{8}l_0$, i.e., the clock skew is still at least $\frac{\alpha}{8}l_0$. Because of the pigeonhole principle there is a sub-chain of length $l_1 := \frac{\alpha\varepsilon}{4(\beta-\alpha)}l_0$ with at clock skew of at least $\frac{\alpha}{8}l_1$ between the top and the bottom node of the sub-chain. Note that $T_1 = \frac{1+\varepsilon}{4\varepsilon}l_1 \le \frac{l_1}{2\varepsilon}$. We can again change the execution indistinguishably by setting the hardware clock rates along this subchain to $1+\varepsilon, 1+\varepsilon-\varepsilon/l_1, 1+\varepsilon-2\varepsilon/l_1, \ldots, 1$ and adapt the message delays (which again lie in the interval $[0,1]$). Again, the topmost node reaches the same logical clock value at time $T_1' := \frac{l_1}{4\varepsilon}$ that it reached before at time T_1. Due to the fact that it increased its logical clock value at least at rate α in the interval $T_1 - T_1' = \frac{l_1}{4}$, the clock skew between the fastest and the slowest node in this sub-chain increased by ast least $\frac{\alpha}{4}l_1$, i.e., the clock skew is now at least $\frac{\alpha}{4}l_1 + \frac{\alpha}{8}l_1 = \frac{3\alpha}{8}l_1$.

Now we repeat this process recursively for sub-chains of lengths l_2, l_3, etc. Since l_{i+1} is a factor of $\frac{\alpha\varepsilon}{4(\beta-\alpha)}$ smaller than l_i, we can only do this $\log_{4(\beta-\alpha)/(\alpha\varepsilon)} D$ often. However, in each of these $\log_{4(\beta-\alpha)/(\alpha\varepsilon)} D$ phases, the *average clock skew* between the top and the bottom node of a sub-chain will grow by $\frac{\alpha}{8}$. In other words, the skew between some neighboring nodes will be at least $\Omega(\alpha\log_{\frac{\beta-\alpha}{\alpha\varepsilon}} D)$. $\square$