Model & Synchronizing w/o Faults
network $G = (V,E)$
node = state machine with hardware clock
edge = communication link (message passing)
Model: What Nodes Can Do

- arbitrary deterministic computations
- computation times satisfy (known) bounds
- hardware clock runs at rates between 1 and $\vartheta$:
  \[ t - t' \leq H_v(t) - H_v(t') \leq \vartheta(t - t') \]
goal: compute logical clocks such that
\[ H_v(t) - H_v(t') \leq L_v(t) - L_v(t') \leq (1 + \mu)(H_v(t) - H_v(t')) \]
- communication by message passing
- messages sent as result of computations
- transmission times satisfy (known) bounds
- (end-to-end) delay, i.e., message transmission + computation time, is between $d-u$ and $d$
- delay $d$, uncertainty $u$, and drift $\vartheta$ are known and can be used in computations
- fix network \( G = (V,E) \) and algorithm
- fix \( H_v \) (and a wake-up time) for each node
- (inductively) fix delay of each sent message
- this specifies an execution

Model: Executions
Model: Executions

- fix network $G = (V,E)$ and algorithm
- fix $H_v$ (and a wake-up time) for each node
- (inductively) fix delay of each sent message
- this specifies an execution*

*event-driven; events are:
- waking up (initialization)
- receiving a message
- reaching specified value of $H_v$
IMPORTANT NOTICE:

Delays include computations, so the time a message is “received” equals the time when any immediately triggered messages are sent!
Example: Max Algorithm

Algorithm 4 Basic Max Algorithm. Parameter $T \in \mathbb{R}^+$ controls how frequently messages are sent. The code lists the actions of node $v$ at time $t$ and provides $\text{getL}()$.

1: if $t = 0$ (i.e., $v$ just woke up) then
2: \hspace{1em} $h \leftarrow \text{getH}()$
3: \hspace{1em} $\ell \leftarrow h$ \hspace{1em} $\triangleright$ initialize $L_v(0)$ to $H_v(0)$
4: end if
5: if received $\langle \ell' \rangle$ at time $t$ and $\ell' > \text{getL}()$ then
6: \hspace{1em} $h \leftarrow \text{getH}()$
7: \hspace{1em} $\ell \leftarrow \ell'$ \hspace{1em} $\triangleright$ increase logical clock to received value
8: end if
9: if $\text{getL}() = kT$ for some $k \in \mathbb{N}$ then
10: \hspace{1em} send $\langle kT \rangle$ to all neighbors
11: end if
12: \textbf{procedure} $\text{getL}()$ \hspace{1em} $\triangleright$ returns $L_v(t)$
13: \hspace{1em} \textbf{return} $\ell + \text{getH}() - h$ \hspace{1em} $\triangleright$ logical clock increases at rate $\frac{dH_v}{dt}$
14: \textbf{end procedure}

- $\text{getH}()$ returns $H_v(t)$
- all nodes are assumed to wake up at time 0
Example: Max Algorithm

Theorem
The Max Algorithm guarantees
\[ \max_{v,w \in V} \{L_v(t) - L_w(t)\} \leq \varnothing dD + (\varnothing - 1)T \]
at time \( t \geq dD + T \),
where \( D \) is the network diameter of \( G \).

\[ D = 3 \]
Example: Max Algorithm

Theorem
The Max Algorithm guarantees
\[
\max_{v,w \in V} \{L_v(t) - L_w(t)\} \leq \vartheta dD + (\vartheta - 1)T
\]
at time \( t \geq dD + T \),
where \( D \) is the network diameter of \( G \).

Proof sketch:
- every \( T \) (logical) time \( v \) broadcast \( L_v \)
- receiving nodes adjust their clock (if needed) and broadcast, too (if they still need to)
- in the \( dD \) time for a value to spread, \( v \)'s clock advances by at most \( \vartheta dD \)
- \( (\vartheta - 1)T \) is added to account for the broadcast interval \( T \)