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 tolarated.

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 accross 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 propogate beyond that
- Need to continuesly obey the asssumed 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 ϑ:

$$t - t' \le H_v(t) - H_v(t') \le \vartheta(t - t')$$

Clock Synchronization: compute logical clocks
s.t. for every $v,w \in V_g$
$H_v(t) - H_v(t') \le L_v(t) - L_v(t') \le (1 + \mu)(H_v(t) - H_v(t'))$

(skew bound) $\max_{v,w \in Vg}\{L_v(t)-L_w(t)\} \le \mathcal{G}$

We define $H_v(t) - H_v(t') \le L_v(t) - L_v(t') \le \boldsymbol{\beta}(t - t')$

Pulse synchronization goals:

For each i ∈ $\mathcal{N}$, v ∈ V_g generate pulse i exactly once, ($p_{v,i}$ is the time when v generates pulse i), such that there exists S, P_{min}, P_{max}, satisfying:

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

Thus, **pulses** are **well aligned** and **well separated**

Basic Observation

Any pulse synchronization algorithm must satisfy:

1) $P_{max} - P_{min} \geq S$
2) $P_{max} \geq \vartheta P_{min}$

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 harware clock

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

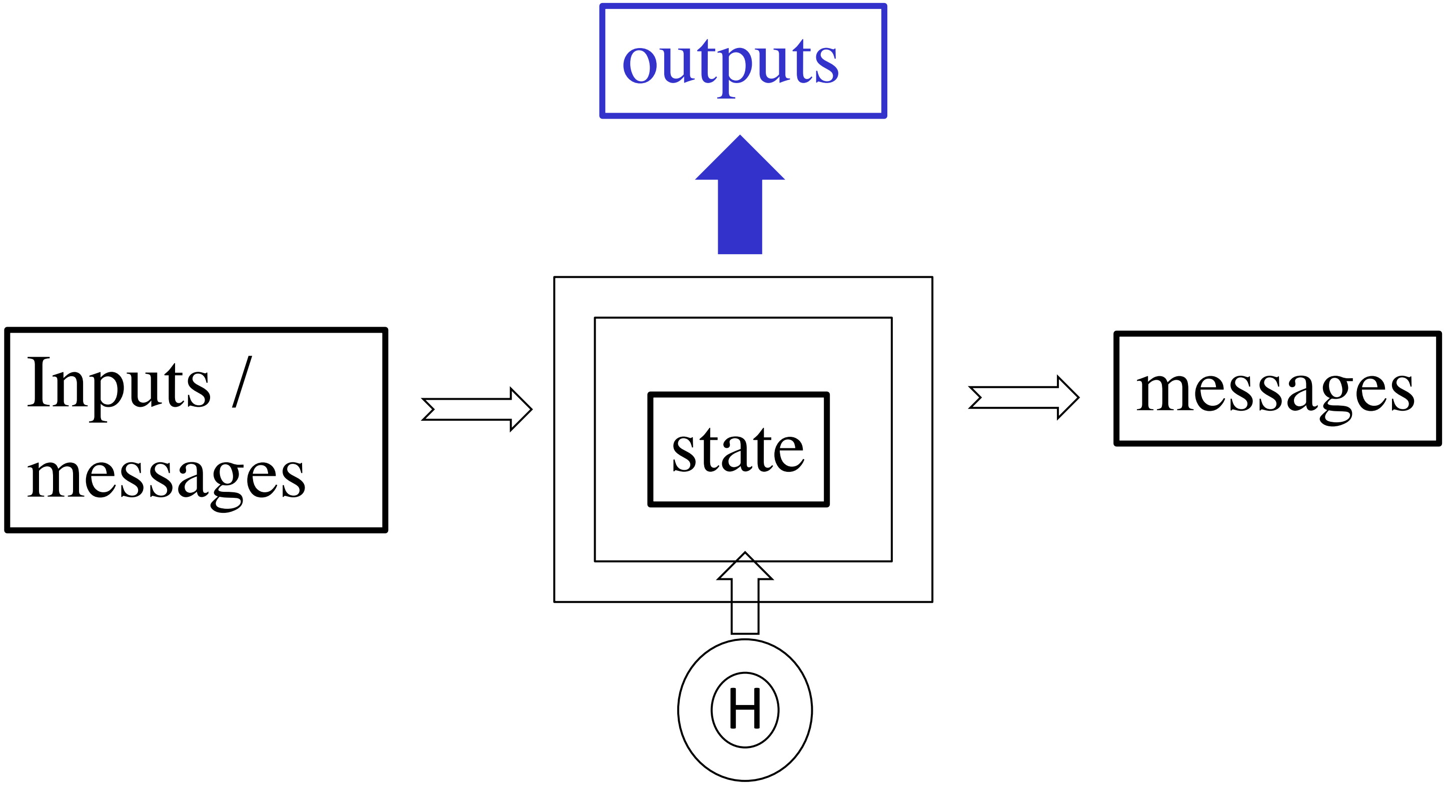
- **Unknow elements**: actual hardware clock drift,

  actual message transmission time,

  which nodes are faulty and their behavior

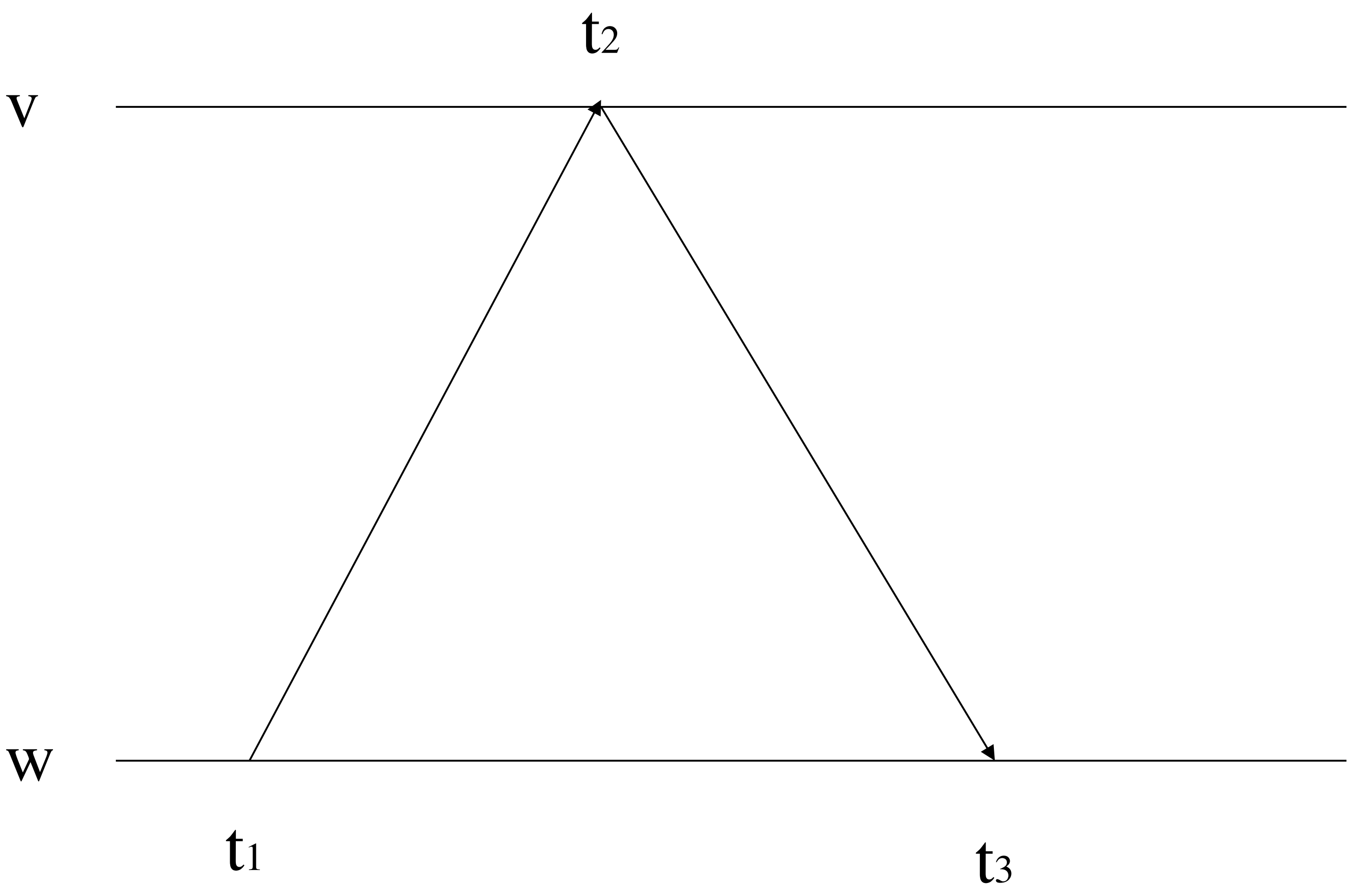
.

A State Machine in TSM

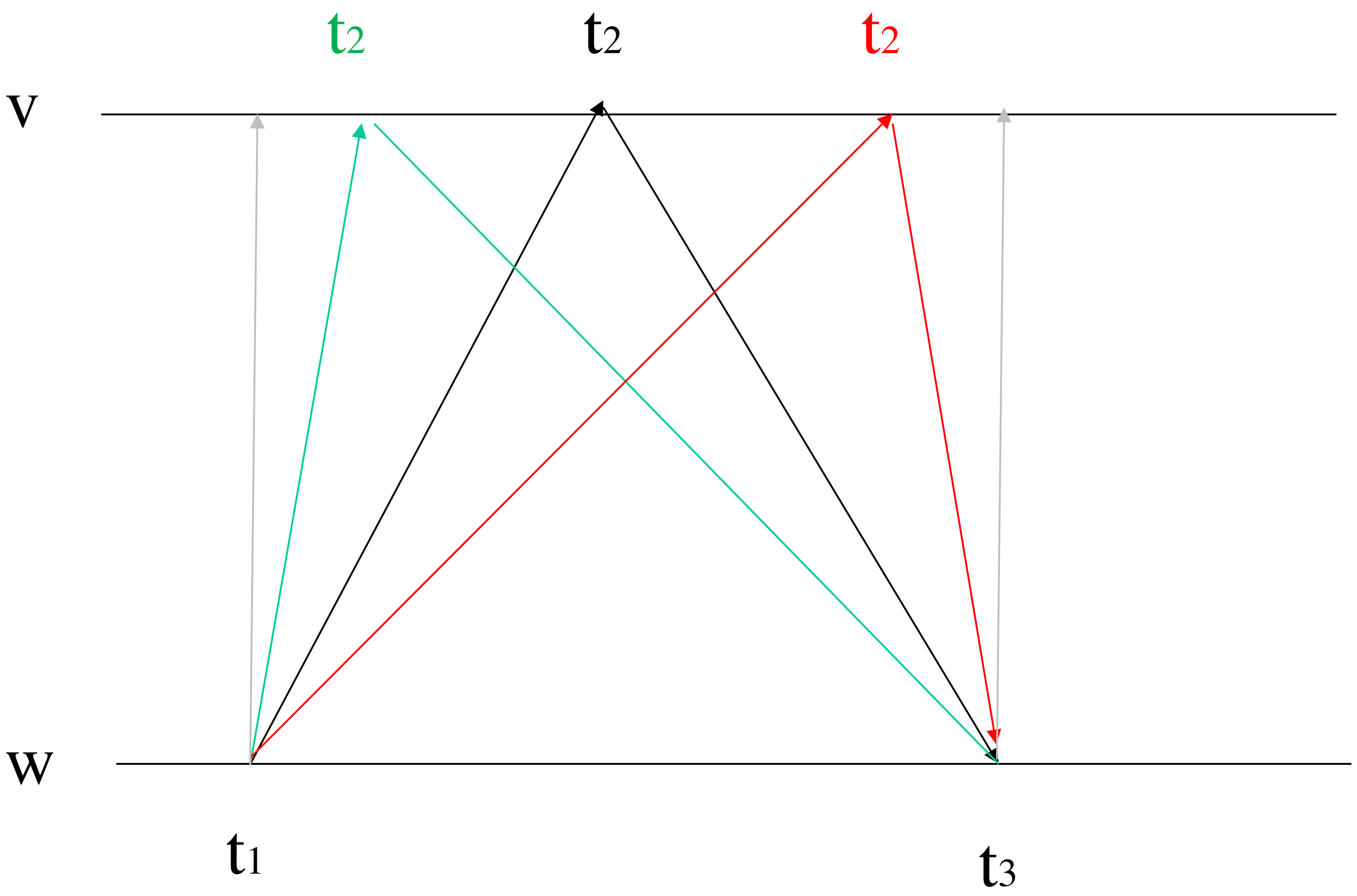


The sequence of messages and outputs depends **solely** on

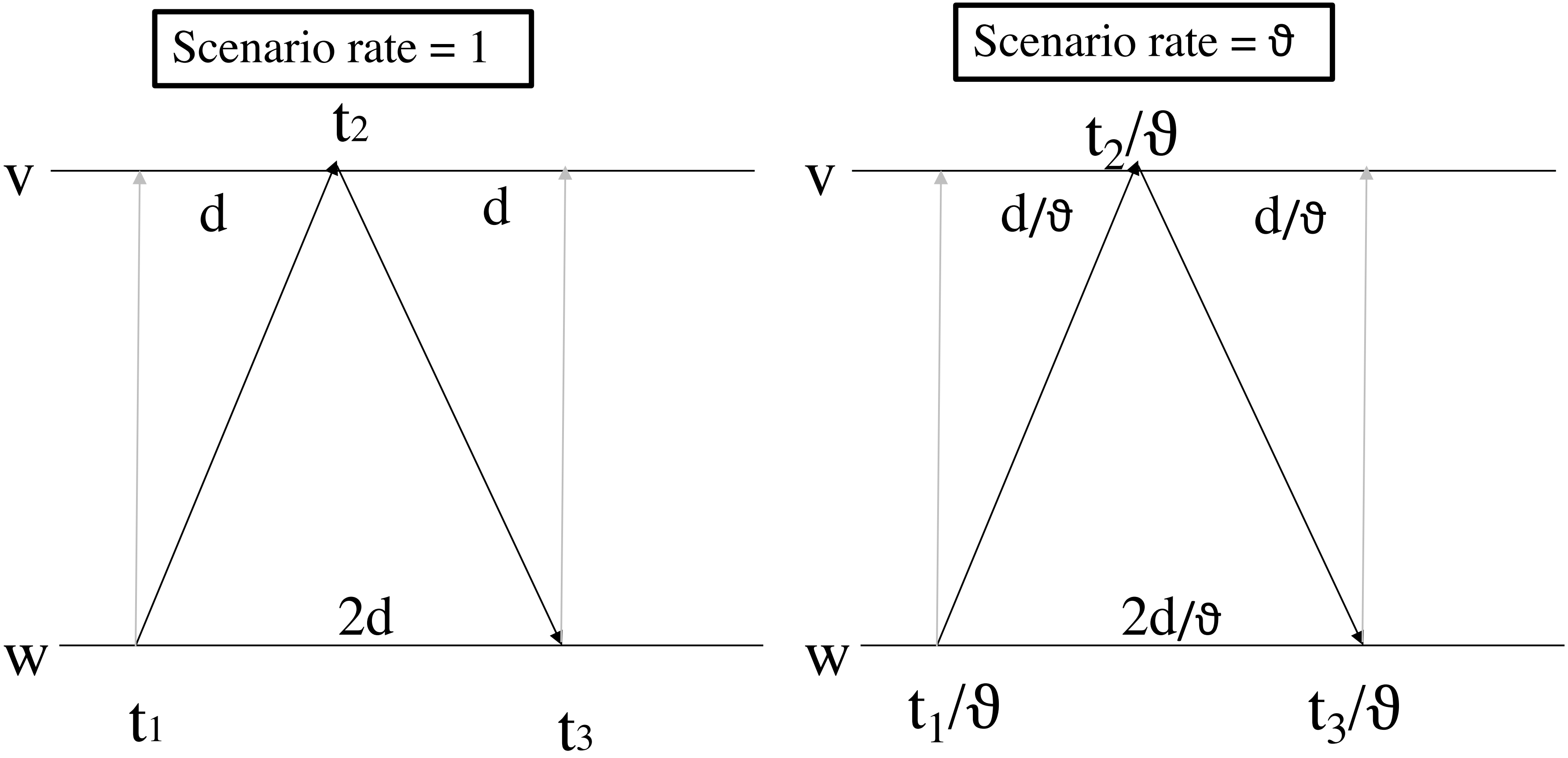
1. the initial state and initial input
2. the sequence of messages and inputs it receives
3. hardware clock readings



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 ϑ**

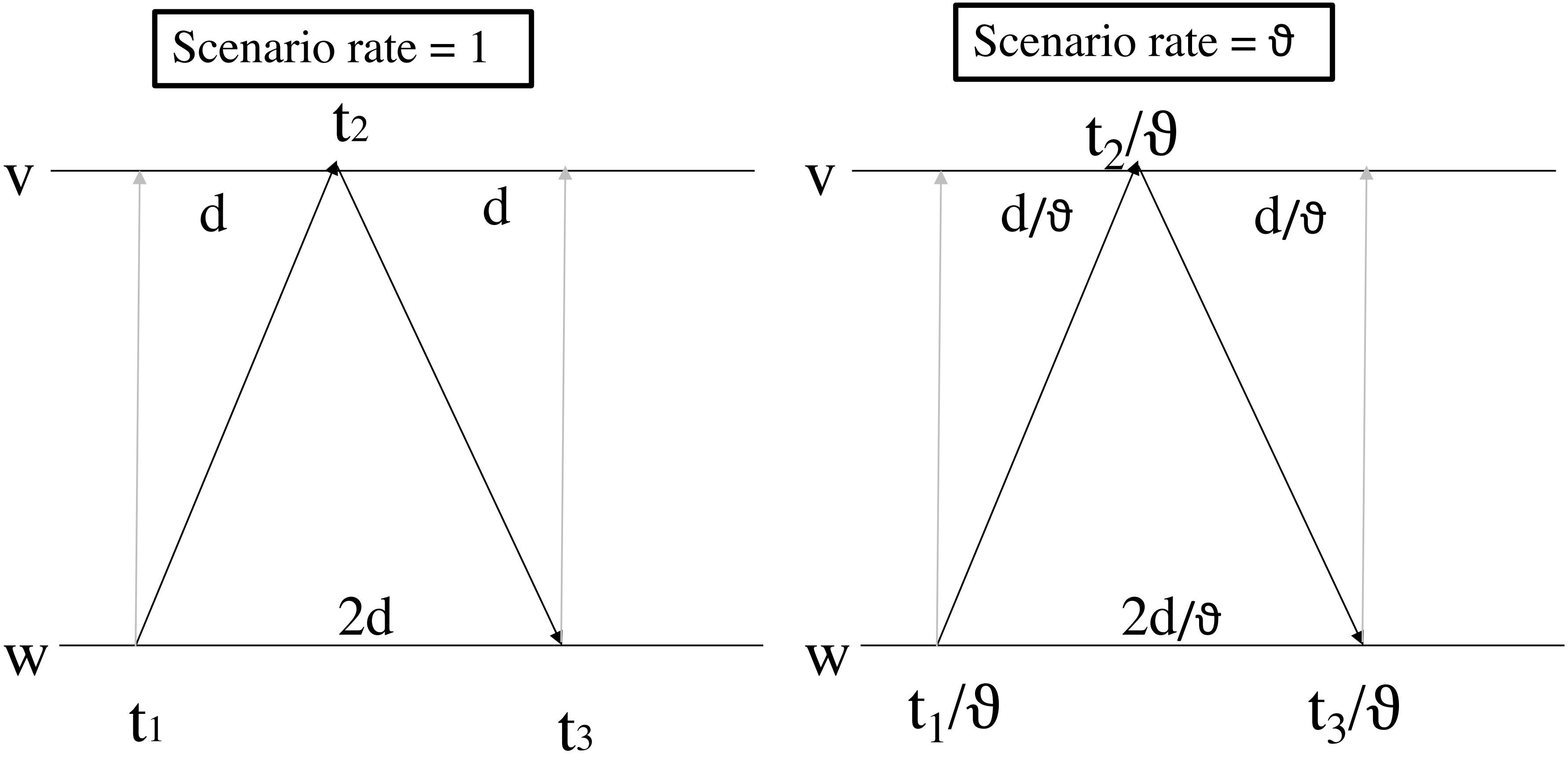


On the left all delays are d and clock rates are 1

On the right all delays are d/ϑ and clock rates are ϑ

$H(t_1)$, $H(t_3)$ at v are the same in both scenarios

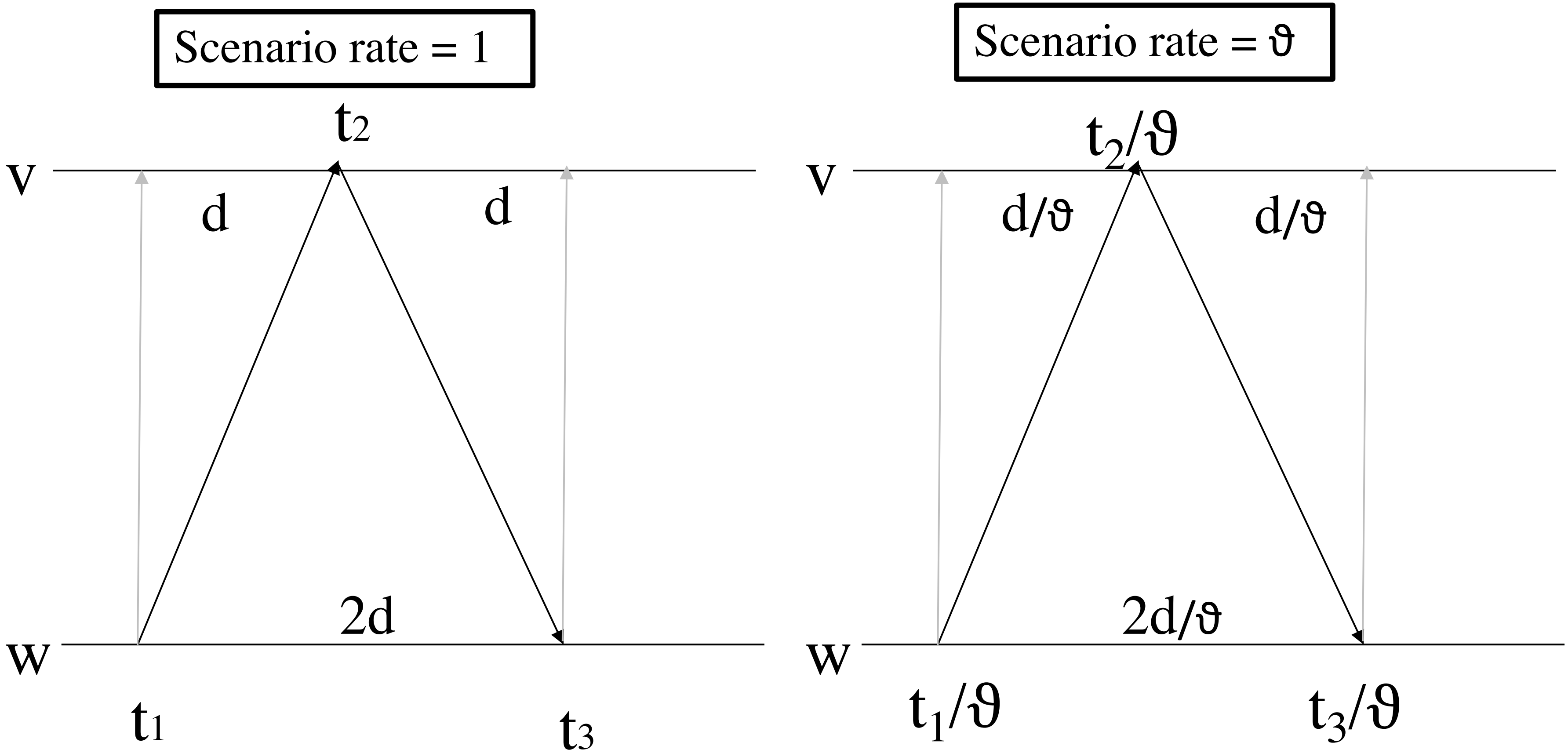
and $H(t_2)$ at w as well



$H(t_1)$, $H(t_3)$ 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: $P_{max} \geq \vartheta\, P_{min}$

Observe that the values are external time values.

Any pulse difference time in the left scenario is divided by ϑ on the other scenario.

Therefore, the value of P_{max} / ϑ needs also be $\geq P_{min}$

Clock Synchronization to Pulse Synchronization

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

- We show how to construct pulse syncronization with parameters

- $S = \mathcal{G}$ (skew)
- $P_{min} = (T- \mathcal{G}) / \boldsymbol{\beta}$ (min period)
- $P_{max} = T + \mathcal{G}$ (max period)

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

Pulse algorithm

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

```
i := 0        (performed only on wakeup)
While true do
    wait until getL() = $iT$
    generate the i-th pulse
    i := i +1
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

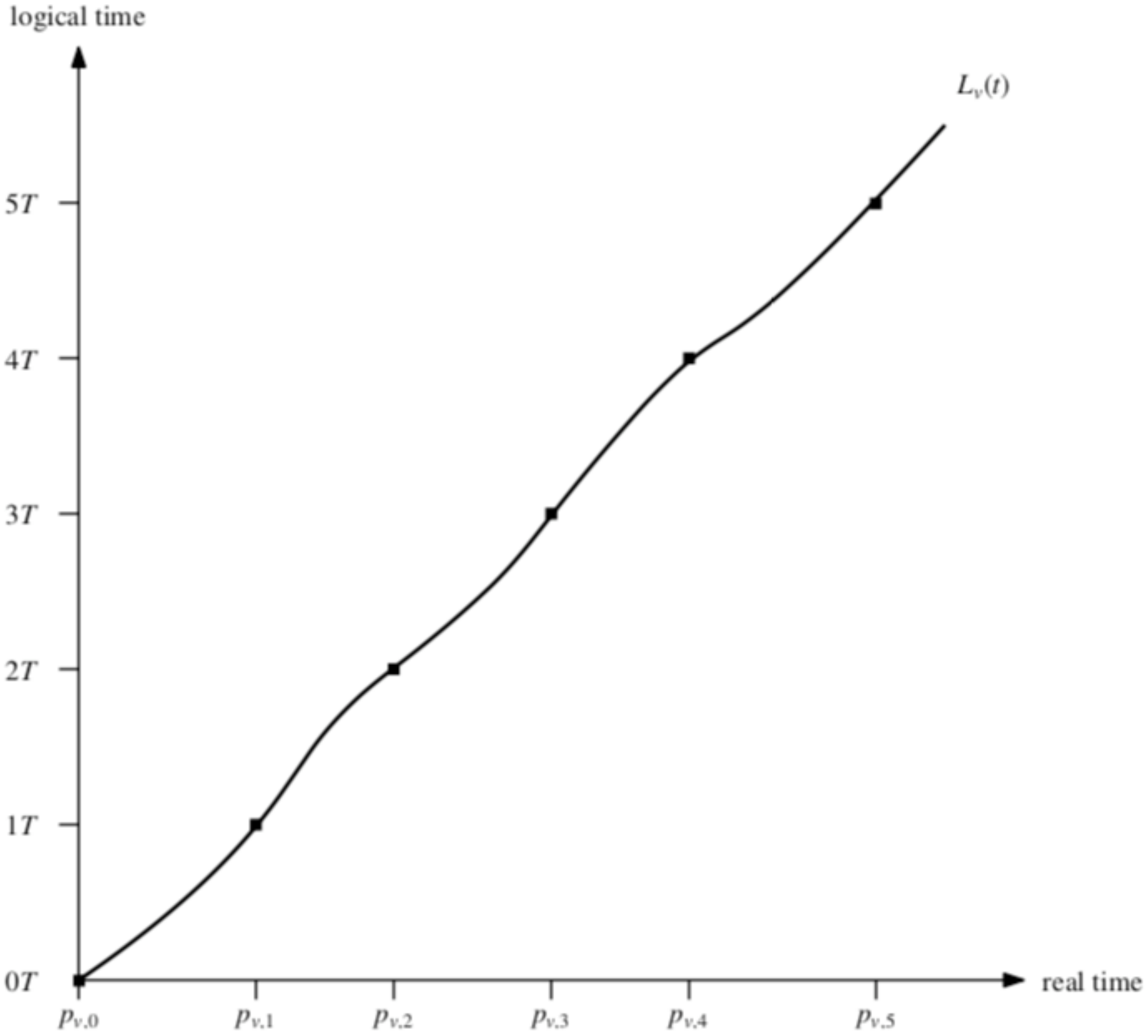


Figure 9.2

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 β, where typically $\beta - 1 \ll 1$. Hence the real time between pulses fluctuates slightly.

The Skew

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

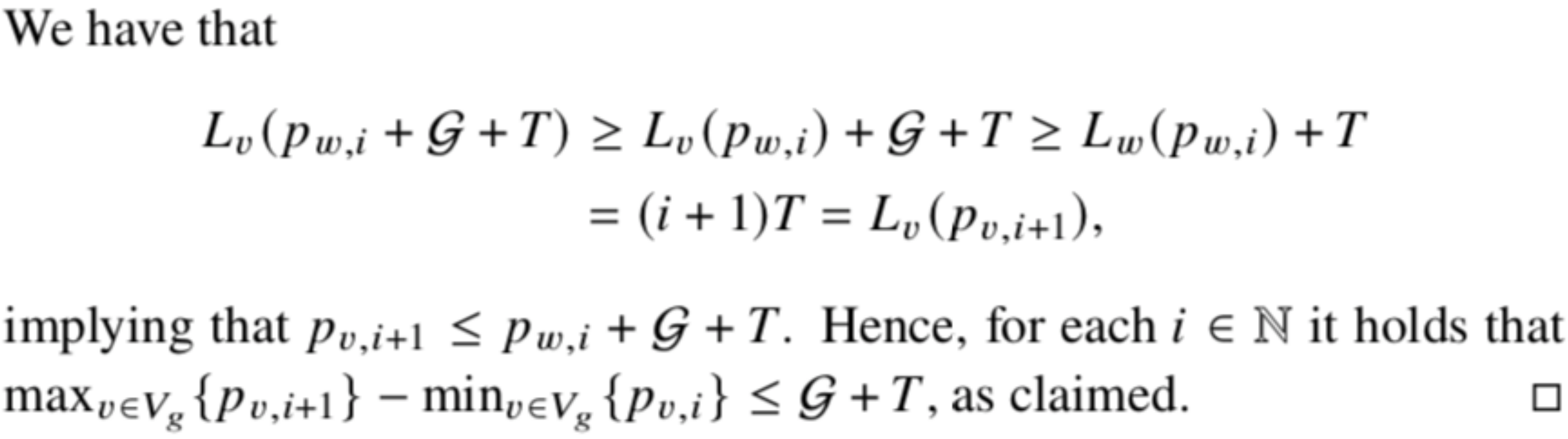
```
i := 0        (performed only on wakeup)
While true do
        wait until getL() = iT
        general i-th pulse
        i := i +1
end while
```

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

$S = \mathcal{G}$, because $\mathcal{G}$ bounds the logical clocks difference

The Min – Max Periods

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



We have that

$$L_v(p_{w,i}+\mathcal{G}+T) \geq L_v(p_{w,i})+\mathcal{G}+T \geq L_w(p_{w,i})+T$$
$$=(i+1)T = L_v(p_{v,i+1}),$$

implying that $p_{v,i+1} \leq p_{w,i}+\mathcal{G}+T$. Hence, for each $i \in \mathbb{N}$ it holds that $\max_{v\in V_g}\{p_{v,i+1}\} - \min_{v\in V_g}\{p_{v,i}\} \leq \mathcal{G}+T$, as claimed. □

Pulse Synchronization to Clock Synchronization

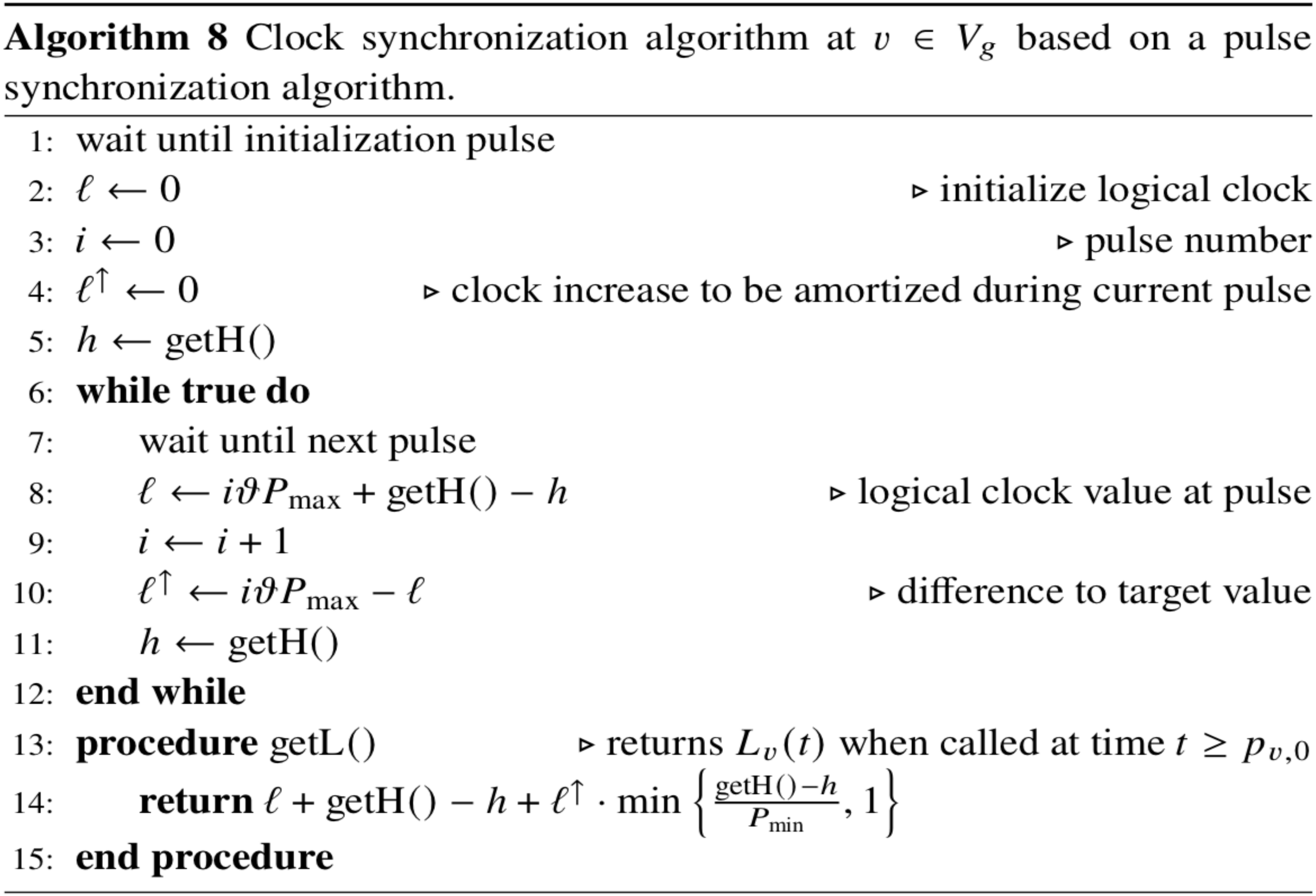
- We now assume we have a fault tolerant pulse synch alg with parameters S, P_{min} and P_{max}

- We show how to construct clock synchronization with parameters

- $\boldsymbol{\beta} = \vartheta^2 P_{max} / P_{min}$

- $\boldsymbol{\mathcal{G}} = (\vartheta - 1) P_{max} + \boldsymbol{\beta} S$

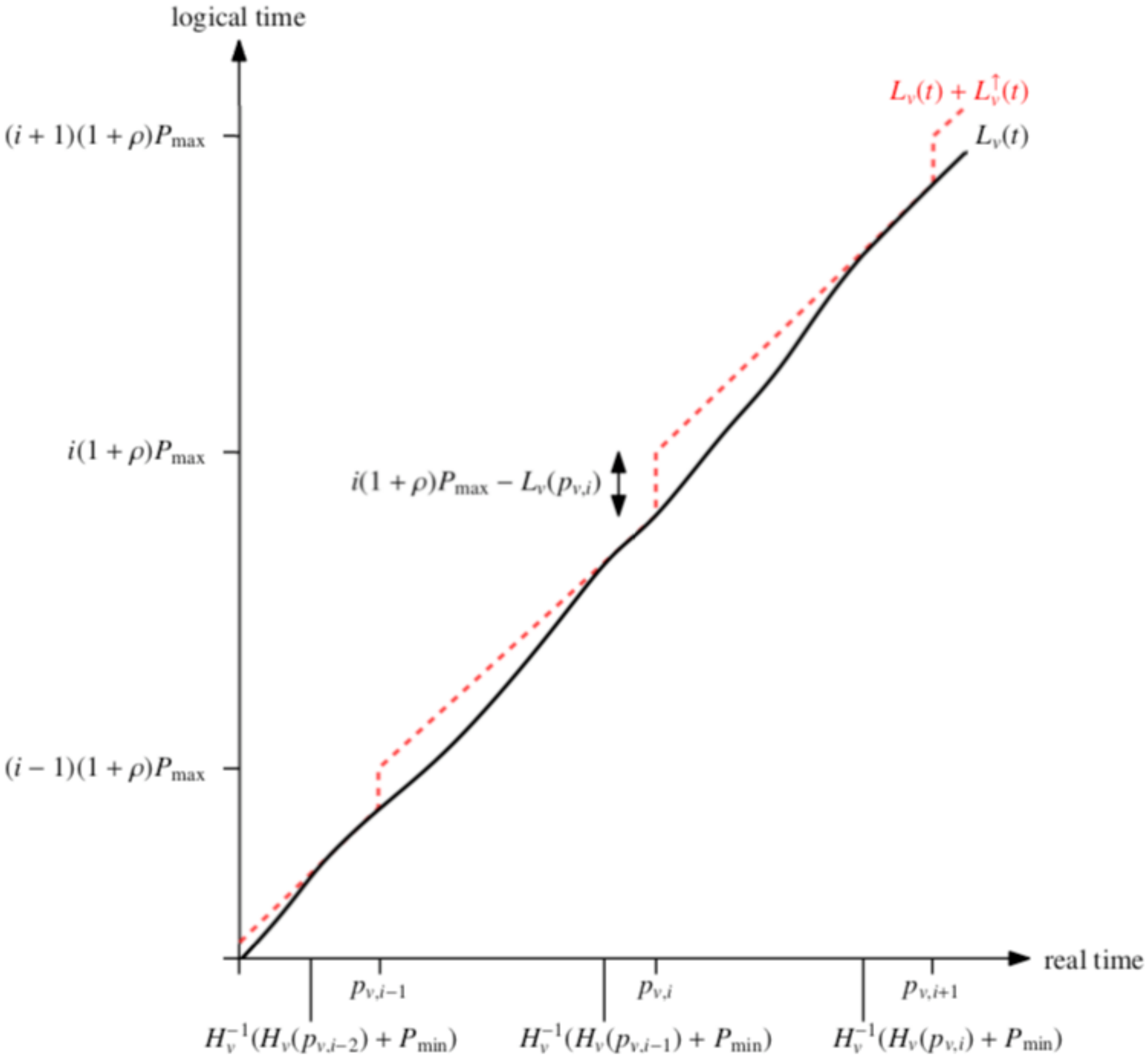
.



Algorithm 8 Clock synchronization algorithm at $v \in V_g$ based on a pulse synchronization algorithm.

```
wait until initialization pulse
$\ell \leftarrow 0$                                   ▷ initialize logical clock
$i \leftarrow 0$                                      ▷ pulse number
$\ell^{\uparrow} \leftarrow 0$        ▷ clock increase to be amortized during current pulse
$h \leftarrow \mathrm{getH}()$
while true do
    wait until next pulse
    $\ell \leftarrow i\vartheta P_{\max} + \mathrm{getH}() - h$        ▷ logical clock value at pulse
    $i \leftarrow i + 1$
    $\ell^{\uparrow} \leftarrow i\vartheta P_{\max} - \ell$            ▷ difference to target value
    $h \leftarrow \mathrm{getH}()$
end while
procedure getL()                  ▷ returns $L_v(t)$ when called at time $t \geq p_{v,0}$
    return $\ell + \mathrm{getH}() - h + \ell^{\uparrow} \cdot \min\left\{\frac{\mathrm{getH}()-h}{P_{\min}}, 1\right\}$
end procedure
```

$$\ell_v^{\uparrow}(p_{v,i}) := \vartheta P_{\max} - (H_v(p_{v,i}) - H_v(p_{v,i-1})).$$


logical time
$L_v(t) + L_v^{\uparrow}(t)$
$L_v(t)$
$(i+1)(1+\rho)P_{\max}$
$i(1+\rho)P_{\max}$
$i(1+\rho)P_{\max} - L_v(p_{v,i})$
$(i-1)(1+\rho)P_{\max}$
real time
$p_{v,i-1}$
$p_{v,i}$
$p_{v,i+1}$
$H_v^{-1}(H_v(p_{v,i-2}) + P_{\min})$
$H_v^{-1}(H_v(p_{v,i-1}) + P_{\min})$
$H_v^{-1}(H_v(p_{v,i}) + P_{\min})$

The Drift Bound

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

- The rate of the logical clock we produce is bounded from above by the drift of the hardware clocks (ϑ) multiplied by the drift caused by the amortization of the extra P_{max} we add at each pulse.
- We amortize it over P_{min} time, which implies a factor of $\vartheta\, P_{max} / P_{min.}$
- So the resulting bound is $\boldsymbol{\beta} = \vartheta^2\, P_{max} / P_{min}$

.

The Skew Bound

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

- The drift rate is $\boldsymbol{\beta} = \vartheta^2 P_{max} / P_{min}$

- 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:

- $\mathcal{G} = (\vartheta - 1) P_{max} + \boldsymbol{\beta} S$

.