Algorithm 1 Follower Heartbeat and Timeout Adjustment Process

```
$N \leftarrow$ provided by administrator    ▷ Number of nodes within the cluster
$T_{\text{dlbc}}, T_{\text{bccc}}, T_{\text{dlcc}} \leftarrow$ null, null, null    ▷ Delays between respective nodes
$T_{\max}, a, b, \delta_{\max} \leftarrow$ administrator provided values    ▷ Leader election timeout configuration values
Nodes $\leftarrow \{x \in \mathbb{Z} \mid 1 \leq x \leq N-1\}$    ▷ Other nodes within the cluster
$L_{\text{latencies-from-node}} \leftarrow \{(node, \tau) \mid node \in \text{Nodes}, \tau \leftarrow \text{null}\}$    ▷ Node and delay-from-node pairs set
$L_{\text{latencies-to-node}} \leftarrow \{(node, \tau) \mid node \in \text{Nodes}, \tau \leftarrow \text{null}\}$    ▷ Node and delay-to-node pairs set
$L_{\text{leader-to-node}} \leftarrow \{(node, \tau) \mid node \in \text{Nodes}, \tau \leftarrow \text{null}\}$    ▷ Node and leader-to-node-delay pairs set
$maxL_M \leftarrow$ null    ▷ The slowest link delay within the nodes' most efficient majority
$\Theta \leftarrow \{(node, maxL_M) \mid node \in \text{Nodes}, maxL_M \leftarrow \text{null}\}$    ▷ Node and slowest-link-delay-within-majority pairs set
$T \leftarrow \text{RANDOM}(T_{\max})$    ▷ Leader election timeout
procedure RECEIVELEADERSHEARTBEAT($message$)
    $T_{\text{dlcc}} \leftarrow \text{GETCURRENTTIME} - message.currentTime$
end procedure
procedure SENDFOLLOWERHEARTBEAT($node$)
    $currentTime \leftarrow \text{GETCURRENTTIME}$
    $latencyFromLeader \leftarrow T_{\text{dlcc}}$
    for all $otherNode \in$ Nodes do
        SENDMESSAGE(otherNode, $\{currentTime, maxL_M, latencyFromLeader\}$)
    end for
end procedure
procedure PROCESSFOLLOWERHEARTBEATRESPONSE($node$, $message$)
    $L_{\text{latencies-to-node}}[node] \leftarrow message.delayFromSender$
end procedure
procedure RECEIVEFOLLOWERHEARTBEAT($node$, $message$)
    $receivedAt \leftarrow \text{GETCURRENTTIME}$
    $delayFromSender \leftarrow receivedAt - message.currentTime$
    $L_{\text{latencies-from-node}}[node] \leftarrow delayFromSender$
    $L_{\text{latencies-from-leader}}[node] \leftarrow message.latencyFromLeader$
    $\Theta[node] \leftarrow message.maxL_M$
    CALCULATETIMEOUT
    SENDMESSAGE(node, $\{delayFromSender\}$)
end procedure
function CALCULATETIMEOUT
    $maxL_M \leftarrow \max\left\{\tau \mid (node, \tau) \in \text{sort}\left(L_{\text{latencies-to-node}}\right)\left[: \left\lceil \frac{|L_{\text{latencies-to-node}}|}{2} \right\rceil\right]\right\}$
    $(node, maxL_M) \leftarrow \arg\max_{(node, maxL_M) \in \Theta} maxL_M$
    $max\Theta_M \leftarrow (node, maxL_M).maxL_M$
    if $max\Theta_M = maxL_M$ then
        $T \leftarrow \left(\frac{maxL_M + \text{RAND}(a,b)}{max\Theta_M}\right) \times T_{\max}$
    else
        $bestCandidate \leftarrow (node, maxL_M).\text{node}$
        $\text{T}_{\text{dlbc}} \leftarrow L_{\text{leader-to-node}}[bestCandidate]$
        $\text{T}_{\text{bccc}} \leftarrow L_{\text{leader-from-node}}[bestCandidate]$
        $T \leftarrow \left(\frac{maxL_M + \text{RAND}(a,b)}{max\Theta_M}\right) \times T_{\max} + \min(T_{\text{dlbc}} + T_{\text{bccc}} - T_{\text{dlcc}}, \delta_{\max})$
    end if
end function
```