



Tight Bounds for Clock Synchronization

Christoph Lenzen, Thomas Locher, Roger Wattenhofer
ETH Zurich, PoDC 2009

Presented by Noa Zilberman
Distributed Algorithms
April - 2010



Agenda

- Background
- The Problem
- Previous Work
- Suggested Model
- Algorithm
- Skew Bounds
- Complexity
- Network Dynamics

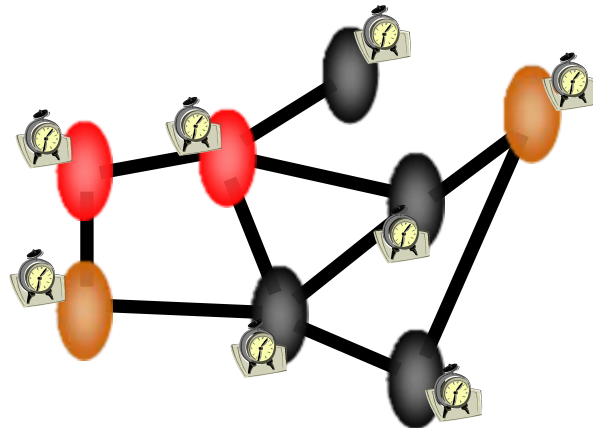
Background

- Some tasks in distributed systems require a notion of time.
 - TDMA-based MAC layer in wireless networks.
 - Assigning a timestamp to a globally sensed event (e.g. earthquake).
 - precise event localization (like multiplayer games).
- Each node has a hardware clock
 - This clock has a bounded variable drift.
- Each node computes a logical clock value
 - Use hardware clock and messages received from neighbors.



The Problem

- Given: A distributed communication system
- Variable clock rates are unknown
- Message delays are unknown
- No external clock can inform the nodes about the real time once in a while
- Goal: Synchronize clocks
 - Minimize the skew between logical clocks



The Problem

- Definition: **Global Skew** is the maximal worst case skew between two nodes in the graph.
- **Objective 1: Minimize the *Global Skew*.**

- Definition: **Local Skew** is the maximal worst case skew between two neighboring nodes.
- **Objective 2: Minimize the *Local Skew*.**
 - For some applications, only occurrence of local events is of importance.
 - Example: TDMA in wireless networks.

Previous Work

Global Skew:

- $\Omega(D/2)$ – Biaz & Welch, 2001
- $O(D)$ – Srikanth & Toueg, 1987
 - Leads to a worst case skew of $\Theta(D)$ between neighboring nodes

Local Skew:

- $\Omega(\log(D)/\log\log(D))$ – Fan & Lynch, 2004
 - Requires minimum clock progress rate
 - Assumes constant clock skew
- $O(\varepsilon\sqrt{D})$ – Locher & Wattenhofer, 2006
- $O(\log(D))$ in static networks – Lenzen, Locher & Wattenhofer, 2008
 - Clock values can jump by $\Theta(\log(D))$
 - Complicated, high message frequency and size
 - Neglect parameters such as maximum clock drift rate
 - Can not come close to lower bound

Algorithm Results

- Take into account many parameters:
 - Delay uncertainty, clock drift rate etc.
- Bound minimum and maximum progress of logical clocks
- For message delay T and Graph diameter D
- Global skew:
 - Lower bound: $\Omega(D \cdot T)$
 - Strict requirements for better than $(1+\varepsilon) \cdot D \cdot T$
- Local Skew: $\Theta(T \log D)$
 - Clock values do not change abruptly
- Good real time approximation
- Low messaging frequency and constant bit complexity

The Model – The Graph

- Graph $G=(V,E)$ with diameter D
- Each node can communicate with all neighboring nodes
- N_v is the set of v 's neighbors: $N_v := \{w \in V \mid \{v, w\} \in E\}$
- v can distinguish between messages from different neighbors
- All communication is reliable
 - Messages are never lost



The Model –Clocks

- Each node has its own hardware clock.
 - The clock starts running when node v was initiated.
 - The first node starts his clock at real time $t=0$.
 - The clock always has a positive progress.
 - The clock rate is bounded from above and below.

The Model –Clocks

- Each node has a logical clock.
 - This is the clock being synchronized.
 - Lower bound: the time that passed since it started running multiplied by hardware clock lower bound.
 - Upper bound: the time that passed since the first node started running multiplied by hardware clock upper bound.
 - Logical clock values may not change dramatically over a short time, and may not run backwards.
 - Thresholds are set by the algorithm.
- Clock skew refers to the difference between the values of logical clocks.

The Model – Delay & ticks

- A message delay has some uncertainty.
- Assumption: An upper bound on the delay is known.
- Assumption: a node $v \in V$ performs computations and handles messages only at *clock pulses*.
 - These events are referred to as ticks.
 - Discrete model.
- A message may be received ε after the tick
 - Will need to wait another clock pulse until it is handled
 - Meaning: the delay uncertainty is equal or larger than the bound on the hardware clock rate of v .

Algorithm

- Initialization: upon a first received synchronization message, execute an initialization algorithm.

At each tick event:

- Update variables – adapt local variables according to information received since last tick event.
- SetClock – increase logical clock and adapt local variables.
- SendMessage – send a message if necessary.

Algorithm

Initialization algorithm:

- Set logical clock L_v to zero.
- Set estimated maximal clock skew, to the largest logical clock received.
- Set estimated largest hardware clock to the lower round of the maximal logical clock received.
- For every neighbor w of v :
 - Save the maximal logical clock value received from neighbor
 - Set the local clock skew between w and v by subtracting saved neighbor logical clock from local logical clock.
- Send to all neighbors your logical clock L_v and estimated largest logical clock $L_v + \Lambda_v^{\max}$

Algorithm

Update Variables

- Save maximum logical clock received.
- Update global largest hardware clock (estimated)
 - Round down largest logical clock received
- Update global largest skew (estimated)
 - Maximum logical clock minus local logical clock, minus 1 (for tick time).
- Save neighbors logical clock
 - May receive more than a single message per tick
 - Save the highest value per neighbor
- Update estimated skew to each neighbor
 - local logical clock (plus 1 for tick time) minus neighbor's logical clock.
- Save/Indicate maximum and minimum (possibly negative) local skew.

Algorithm

SetClock:

- Goal: determine by how much to increase logical clock.
- Idea:
 - Tolerate a clock skew of κ
 - Ensures that even the node with smallest clock value can raise its value
 - even if it received delayed messages with clock values smaller than its own.
 - A node can not increase its clock value:
 - by more than μ
 - Higher than the estimated highest logical clock value.
- As a result, slow nodes can catch up with fast nodes
 - Fast nodes may increase their values in a low rate only

Algorithm

SetClock:

- R_v marks added value
- Set the added value R_v to be at least 0 and the minimum from:
 - Maximal global skew
 - Bound on maximal logical clock progress rate
 - The maximum between:
 - The integer value that is average of maximum and minimum local skew (must be at least zero)
 - The gap between tolerated clock skew κ and minimum local skew
 - So that the slowest neighbor clock can increase itself by κ to match with local clock.

Algorithm

SetClock – continued:

- Add to the logical clock 1 (for tick) plus R_v .
- Reduce from the maximal estimated skew R_v .
 - This is the compensation we did in this tick to the global skew
- Add to every neighbor's estimated skew R_v .
 - This is the compensation we did in this tick to the local skew

Algorithm

Send Message:

- Condition on sending messages:
 - The largest logical clock estimation was updated by a message
OR
 - Messaging delay passed, and estimated largest logical clock is higher than announced before.
 - If true, also increase largest hardware clock estimation by a predefined step.
- The message contains the local logical clock L_v and the estimated highest logical clock $L_v + \Lambda_v^{\max}$
- Send to all neighbors.

Global Skew Bounds

Lower bound:

- Without knowledge of lower bound on hardware clock rate ε , a global skew of $D \cdot T$ can not be avoided.
- Without knowledge of bounds on message delay T stronger than:

$$T \in \left[\frac{(1 - \varepsilon)}{(1 + \varepsilon)} \hat{T}, \hat{T} \right] \leftarrow \text{Upper bound on message delay}$$

a bound on global skew better than $(1 + \varepsilon) \cdot D \cdot T$ can not be achieved.

Local Skew Bounds

- If $\kappa \in \Omega(\mathcal{T})$, the algorithm achieves an asymptotically optimal skew of:

$$\Theta\left(T \log_{\mu/\varepsilon} D\right)$$

- For $D \rightarrow \infty$ and $\varepsilon \rightarrow 0$, the approximation ratio tends to $2^{\hat{T}/T}$

- For nodes in distance d , where difference in hardware clock rates, $\beta - \alpha \in O(1)$, the best worst case guarantee on clock skew achieved is:

$$\Theta\left(\alpha T d \left(1 + \log_{(\beta-\alpha)/\alpha\varepsilon} (D / d)\right)\right)$$

Complexity

■ Amortized message frequency:

□ $freq = \Theta\left(\frac{\hat{\varepsilon}}{\hat{T}}\right)$

Upper bound on clock skew
Upper bound on message delay

□ Over a short period of time, worst case:

- Either send an update for every message received
- -OR- send every period of time and increase global skew

■ Bit Complexity:

□ $O(1)$

■ Memory complexity:

$$\Theta\left(\log T + \log(\mu D) + \Delta\left(\log 1/\mu + \log(\varepsilon \mu D) + \log \log_{\mu/\varepsilon} D\right)\right)$$

↑
Local
counter

↑
Global
skew

↑
Node's
Degree

↑
Elapsed
time

↑
Past
update

↑
Skew

Network Dynamics

- How to handle new nodes or nodes that crash?
- New node:
 - Requires an adaptive integration scheme, since if clock skew is small, waiting may be very long
 - the skew is in the order of global skew.
- Crash:
 - If a message from a neighbor w is “overdue”, remove w from neighbors list
 - Can be a crash or if delay is longer than estimated...
 - If a message from w is received again
 - Add w back to N_v , Update κ and \hat{T}
 - Flood the estimated new delay through the network
 - Does not hold for temporary link failures



Thank You!