Clock Synchronization with Bounded Global and Local Skew

Christoph Lenzen, ETH Zurich
Thomas Locher, ETH Zurich
Roger Wattenhofer, ETH Zurich
October 2008

49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)
Philadelphia, PA, USA

Distributed Computing Group

Motivation: No Global Clock

- Many tasks in distributed systems require a common notion of time
- Sometimes not all devices can be connected to a "global" clock
  ⇒ Equip each device with its own clock!

Problem 1: Different clocks have different clock rates
Even worse, these clock rates may vary over time!
Communication is required to synchronize the clocks!
Problem 2: What if the message delays vary?

⇒ Clock drifts!
Each message has a different delay...

How well can distributed clocks be synchronized?

Overview

I. Motivation
II. Model
III. Algorithms
IV. Conclusion

Model: Clocks

- Each device has a hardware clock $H \Rightarrow H(t) = \int_0^t h(\tau) \, d\tau$.
- The hardware clock rate $h$ is bounded $\Rightarrow \forall t: h(t) \in [1-\epsilon, 1+\epsilon]$
- Each device computes a logical clock value $L$ based on:
  Its hardware clock $H$ and its message history (the messages it received)
- Messages are required to correct clock skews!
- A clock synchronization algorithm specifies how the logical clock value $L$ is adapted!
- Minimize clock skew of logical clocks!
- And triggers synch messages!
Model: Graph & Communication

- Distributed system = Graph $G$ of diameter $D$
  - Node = Computational device
  - Edge = Bidirectional communication link
- Nodes communicate via reliable, but delayed messages
  - Each message may be delayed by any value $\in [0,1]$ **Simple normalization!**

$D=3$

Model: Optimization Criteria

- Good real time approximation: $\forall v \in V, \forall t: |L_v(t) - t| \leq \epsilon t$

$t = 0$

$\frac{(1+\epsilon)t}{t} \quad \frac{(1-\epsilon)t}{(1-\epsilon)t}$

$\forall v \in V, \forall t_2 > t_1: L_v(t_2) - L_v(t_1) \geq (1-\epsilon)(t_2-t_1)$

- Minimum progress:
- Minimize the skew among all nodes:

\[ \max_{v,w,t} |L_v(t) - L_w(t)| \]

Minimize the global skew!

Model: Optimization Criteria II

More importantly: We want a small clock skew between $v$ and $w$, if the distance between $v$ and $w$ is short!

Minimize the skew among neighboring nodes:

\[ \max_{v,w \in N(v),t} |L_v(t) - L_w(t)| \]

Minimize the local skew!

Model: Importance of Local Skew

For many applications, locally well synchronized clocks are more important!

- Monitoring applications
  (record $<event, timestamp>$)
- Tracking applications
  Use $<event, time>$ recordings to determine movement/speed etc.
- More fundamental:
  E.g., TDMA requires (locally) synchronized clocks!
Model: Old Results

A well-known result is that the skew between two nodes at distance $d$ is $\Omega(d)$ in the worst case! 
$\Rightarrow \Omega(D)$ lower bound on global skew!

Guaranteeing a global skew of $\Theta(D)$ is easy…

"Always set $L$ to largest clock value!"

Bounding the local skew is hard(er):

Many (reasonable) algorithms $\Rightarrow O(D)$

Best known bound $\Rightarrow O(\sqrt{D})$

Lower bound $\Rightarrow \Omega(\log D / \log \log D)$

Diameter determines the local skew!!!

True bound probably $\Omega(\log D)$...

Algorithm: Simple Strategies

Strategy I: "Always set $L$ to largest clock value!"

Problem:

\[
\begin{array}{cccccc}
50 & 80 & 90 & 90 & 50 & 5 \\
+10 & +20 & +30 & & & \\
\end{array}
\]

$O(D)$ local skew!

Strategy II: "Take the average clock value!"

Problem:

$O(D^2)$ global skew! ($\Rightarrow O(D)$ local skew...)

Algorithm: Better Strategies

Strategy III: "Always increase the clock value $L$ UNLESS a neighbor is $B$ behind."

Problem:

\[
\begin{array}{cccccc}
V & W & & & & \\
L=2 & \Rightarrow L=8 \\
B & & & \Rightarrow & B & \Rightarrow B & \Rightarrow \\
\end{array}
\]

Length of this chain $\Rightarrow O(D/B)$

$v$ can built up skew to $w$ at rate $O(\epsilon)$ for $O(D/B)$ time $\Rightarrow O(\epsilon \cdot D/B) = O(D)$ skew!!!

How can we fix this?!

$\Rightarrow$ Choose $B \in O(\sqrt{D}) \Rightarrow O(\sqrt{D})$ local skew!!!

Ok, but can we do better?
**Algorithm: Increase Tolerance**

Strategy III+: “Tolerate B skew, but if v experiences a skew of $iB \rightarrow$ Tolerate $iB$ skew!”

For any $i \in \{2, 3, \ldots\}$

- Build up $2B$ skew!
- Tolerance increases!
- Skew “moves away”!

**Algorithm: Intuition**

If the adversary wants to build up $3B$ skew $\rightarrow$ A chain with $2B$ skew between neighbors is needed!

$\rightarrow$ The longer the better!

$\rightarrow$ Only $O(D/B)$ time to build chain!

If $l$ is the length of the chain $\rightarrow \Omega(B/l)$ time is needed

$\rightarrow \Omega(B/l)$ $\in O(D/B) \Rightarrow l \in O(cD/B^2) \in O(D/B^2)$

Inductively:

A skew of $(i+1)B$ requires a chain with $iB$ skew between nodes $\Rightarrow l_i \in O(D/B^i)$

Lose a factor of $B$!

Local Skew $\in O(B\cdot \log B D)$!

**Algorithm: Why It Fails**

That's it? Unfortunately, no. The message delays cause problems:

- Progress = $x$
- Skew < $B-x$!

$\Rightarrow$ v thinks w is $B$ behind!

Build up skew!

Increase tolerance!

Sees < 2B skew!!! $\Rightarrow$ No increase!

**Algorithm: How bad is it? How can we fix it?**

We get the following picture:

- $iB-x$ $\rightarrow$ $(i-1)B-x$ $\rightarrow$ $(i-2)B-x$ $\rightarrow \ldots$ $B-x$

Local skew $\rightarrow O(\sqrt{D})$ $\Rightarrow$ Since global skew $\in O(D)$

How can we fix this?!?

$\rightarrow$ React earlier! If a neighbor w is > $iB-x$ behind, ask w to increase its clock value!!!

That's it?

Fortunately, yes.

If $iB-x+r$ behind, increase by $r$!
Overview

I. Motivation
II. Model
III. Algorithms
IV. Conclusion

Conclusion: Results

- **Local skew** \( \rightarrow O(\log D) \)
  \[ |L_v - L_w| \in O(d(v,w) \cdot \log(D/d(v,w))) \]
  - Probably asymptotically optimal!
- **Global skew** \( \rightarrow O(D) \)
  \[ |L_v - L_w| \leq (1 + O(\varepsilon)) D \]
  - In fact, only a factor \( \approx 2 \) larger than the lower bound!
- **Bit complexity** \( \rightarrow O(\Delta \log^2 D) \)
- **Space complexity** \( \rightarrow O(\Delta \log \log D + \log^2 D) \)

Conclusion: Outlook

Open problems?

- **Bound the logical clock rate!**
  - Ideally: \( l(t) \in [1-O(\varepsilon), 1+O(\varepsilon)] \)
  - Clock skew is built up at a low rate!

- **Reduce the bit complexity!**
  - Send less bits per message
  - Reduce the message frequency
  - Enable piggybacking!

- **Prove tight bounds for global/local skew!**

Questions and Comments?

Thank you for your attention!

Thomas Locher
Distributed Computing Group
ETH Zurich, Switzerland
lochert@ti.com.ethz.ch
http://dcg.ethz.ch/members/thomasl.html