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

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!
  - Minimize clock skew of logical clocks!
  - A clock synchronization algorithm specifies how the logical clock value $L$ is adapted!
  - 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 $\in [0,1]$.

Model: Optimization Criteria

- Good real time approximation: $\forall v \in V, \forall t: |L_v(t)-t| \leq \epsilon t$
- Minimum progress:
  - $\forall v \in V, \forall t_2 > t_1: L_v(t_2)-L_v(t_1) \geq (1-\epsilon)(t_2-t_1)$
- Minimize the skew among all nodes:
  - $\max_{v,w,t} |L_v(t)-L_w(t)|$

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!
  - Allow more skew with increasing distance!
- Minimize the skew among neighboring nodes:
  - $\max_{v,w \in N(v), t} |L_v(t)-L_w(t)|$

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! → $\Omega(D)$ 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 $\Omega(\log D)$...

Overview

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

Algorithm: Simple Strategies

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

$O(D)$ 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:

How can we fix this?!?

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

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

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

**Strategy III+:** "Tolerate B skew, but if v experiences a skew of i·B -> Tolerate i·B skew!"

1. Build up 2B skew!
2. Tolerance increases!
3. Skew "moves away"!

Thomas Locher, ETH Zurich @ FOCS 2008

Algorithm: Intuition

If the adversary wants to build up 3B skew -> A chain with 2B skew between neighbors is needed:
- The longer the better!
- Only $O(D/B)$ time to build chain!

If $l$ is the length of the chain -> $\Omega(B \cdot l/e)$ time is needed
- $\Omega(B \cdot l/e) \in O(D/B) \rightarrow l \in O(e \cdot D/B^2) \in O(D/B^2)$

Inductively:
A skew of $(i+1)\cdot B$ requires a chain with $i\cdot B$ skew between nodes
- $l_i \in O(D/B^i)$

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

Thomas Locher, ETH Zurich @ FOCS 2008

Algorithm: Why It Fails

That's it? Unfortunately, no.
The message delays cause problems:
- Progress = x
- v thinks w is B behind!
- Skew < B-x!

Thomas Locher, ETH Zurich @ FOCS 2008

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

We get the following picture:
- $i\cdot B-x$ $i-1\cdot B-x$ $i-2\cdot B-x$ $\ldots$ $B-x$

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

How can we fix this?!?
- React earlier! If a neighbor w is $i\cdot B-x$ behind, ask w to increase its clock value!!!

That's it?
Fortunately, yes.

Thomas Locher, ETH Zurich @ FOCS 2008
Overview

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

Conclusion: Results

- Local skew $\rightarrow O(\log D)$
  $|L_v - L_w| \in O(d(v,w) \cdot \log(D/d(v,w)))$

- Global skew $\rightarrow O(D)$
  $|L_v - L_w| \leq (1 + O(\epsilon))D$

- Bit complexity $\rightarrow O(\Delta \log^2 D)$
- Space complexity $\rightarrow O(\Delta \log \log D + \log^2 D)$

Probably asymptotically optimal!
In fact, only a factor $\approx 2$ larger than the lower bound!

Conclusion: Outlook

Open problems?

- Bound the logical clock rate!
  Ideally: $l(t) \in [1-O(\epsilon), 1+O(\epsilon)]$

- 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@tk.ee.ethz.ch
http://dcg.ethz.ch/members/thomasl.html