Secure and Self-Stabilizing Clock Synchronization in Sensor Networks

Jaap-Henk Hoepman, Andreas Larsson
Elad M. Schiller, Philippas Tsigas

13 November 2007
Attacks against clocks
Outline

- Motivation
- Implementation
- Attacks
- Correctness
- Earlier work
- Conclusion
The need for clock synchronization

- Pinpointing events geographically
- Time division message scheduling
- Radio shutoff periods
- Certain mathematical functions
- ...
Need for precision

Required result

Result of traditional protocols
Adversary

- Much more powerful than the nodes
  - Intercepting
  - Replaying
  - Delaying
- Capturing nodes and impersonating
Self-stabilization, Security & Fault tolerance

- Dealing with transient faults
- Security needs self-stabilization
  - Security under certain assumptions
  - Attacks eventually violate assumptions

  **Arbitrary starting configuration**

- Fault tolerance – message loss
  - Noise
  - Collisions
Outline

- Motivation
- **Implementation**
- Attacks
- Correctness
- Earlier work
- Conclusion
The clock model

- Offset is arbitrary
- Rate, $\rho$, is varying
  - Manufacturing variations
  - Environmental variations
- Clock rate stays within a certain interval

\[ \rho_{\text{min}} < \rho < \rho_{\text{max}} \]
Roundtrip synchronization

Offset
Delay
Reference Broadcast

Offset with higher precision
The protocol layers

<table>
<thead>
<tr>
<th>Policy for accuracy and energy budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock adjustments</td>
</tr>
<tr>
<td>Filtering out delays</td>
</tr>
<tr>
<td><strong>Beacon scheduling</strong></td>
</tr>
<tr>
<td><strong>No self-stabilizing implementation exists</strong></td>
</tr>
<tr>
<td>Secure communication primitives</td>
</tr>
</tbody>
</table>
Combining the two approaches

Beacon sent by node i:

\[ R_0 \quad \cdots \quad R_{i-1} \quad A_i \quad R_{i+1} \quad \cdots \quad R_n \]
Dealing with message loss

<table>
<thead>
<tr>
<th></th>
<th>$R_0$</th>
<th>$\cdots$</th>
<th>$R_{i-1}$</th>
<th>$A_i$</th>
<th>$R_{i+1}$</th>
<th>$\cdots$</th>
<th>$R_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Delivering to upper layer

- Data held by a node
  - Its beacon send times
  - Its receive times of beacons
  - The corresponding data received from others

- Delivery to upper layer is delayed
  - Collect as much as possible before reporting
Randomized beacon scheduling

Partition time
Divide partitions into slots \((n \log^2 n)\)
Randomly send one beacon per partition
**Time complexity**

\[ n \text{ nodes send a message of size } O(n) \text{ each} \]

<table>
<thead>
<tr>
<th>Optimal</th>
<th>Our randomized strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ n^2 ]</td>
<td>[ n^2 \log^2 n ]</td>
</tr>
</tbody>
</table>

\( n = \text{bound on degree of nodes} \)
Outline

- Motivation
- Implementation
- **Attacks**
- Correctness
- Earlier work
- Conclusion
The attacker model

- Interception of messages
  - Stop receipt
  - Replay later

- Capturing nodes
  - Get data including keys
  - Stop nodes
  - Impersonate nodes
Delay attacks

- Cryptography does not help
- Nonce does not help
Dealing with delay attacks

- Locally calculate delay
- Filter out over-delayed beacons
  - Byzantine agreement [Ganeriwal et al. 05]
  - Outlier filtering [Song et al. 06]
Dealing with captured nodes

- Impersonated nodes send misleading data
  - Send at one time, claim another
- Filter out misleading beacons
  - Byzantine agreement [Ganeriwal et al. 05]
  - Outlier filtering [Song et al. 06]
Outline

- Motivation
- Implementation
- Attacks
- **Correctness**
- Earlier work
- Conclusion
Correctness proof

- Beacon scheduler
  - Partially synchronous system
  - Message collision and omission
- Probabilistic delivery guarantees
  - Every node sends a beacon that every node receives
  - Every node receives a response to its beacon from every node
- Beacon aggregation (appears in TR)
Outline

- Motivation
- Implementation
- Attacks
- Correctness
- Earlier work
- Conclusion
Self-stabilizing but not Secure

- [Herman and Zhang 06]
  - a model for clock synchronization in sensor network
  - show that the converge-to-max approach is stabilizing

- A single captured node attack
  - At any time introduce the maximal clock value

- Adversary sends the clock “far into the future”
  - Preventing a continuous time approximation function
Secure but not Self-stabilizing

- No existing secure and self-stabilizing implementations
  - Many implementations require initial clock synchronization prior to the first pulse-delay attack
- The adversary can risk detection and intercept all beacons for a long period
  - As a result: arbitrary clock offsets
  - The system has to use global restart
  - No global restart after deployment!
Secure but not Self-stabilizing

- [Sun et al. 05] cluster-wise synchronization
  - Based on synchronous rounds
  - Byzantine agreement
  - Synchronized clock at the starting configuration

- We make no assumptions on synchronous rounds or start
Secure but not Self-stabilizing

[Manzo et al. 05]
- Consider attacks on unsecured clock synchronization
- Suggest counter measures
- Use a randomly selected “core” of nodes to minimize the effect of captured nodes
- Do not consider the cases in which the adversary captures nodes after the core selection

We make no assumption regarding the distribution of the captured nodes
Secure but not Self-stabilizing

- [Farrugia and Simon 06]
  - A cross-network spanning tree in which the clock values propagate for global clock synchronization
  - No pulse-delay attacks are considered

- [Sun et al. 06]
  - Use external source nodes to increase the resilience against an attack that compromises source nodes

- We use no source nodes
Outline

- Motivation
- Implementation
- Attacks
- Correctness
- Earlier work
- Conclusion
Conclusion

- System settings of traditional networks
  - cannot be assumed
- Designer assumptions
  - cannot hold forever
- Self-stabilization can provide self-defense capabilities
Thank you for your attention

Thank you for your attention

Andreas Larsson
larandr@cs.chalmers.se
Chalmers University of Technology