Overview

- Distributed Services for Distributed VPNs
- Objectives for Robust Time Synchronization
- Approach
  - Offset Estimation
  - Synchronization
- Evaluation
- Conclusion & Outlook
Distributed Services for Distributed VPNs

- Large VPNs, >100 end-points
- For scalable, robust operation distributed configuration
- But what about the centralized management?

- Secure time information available only in some places
- Must be distributed in the VPN
- NTP etc. would create exposed points
Objectives

• Operation in global environment (use no broadcast etc.)
• Synchronize internally & externally
• Integrity (against internal attackers)
• Robustness (jitter, asymmetric paths, perhaps DoS attacks)
• Scalability

Approach – Overview

• All nodes periodically
  – Exchange time information
  – Filter invalid data
  – Adapt towards measured differences

• Note: Also done in some WSN approaches, but do so more robust (as this works in this scenario)
Approach – Offset Estimation

- Measure RTT and time over encrypted tunnel
- Problem: $T_1$ and $T_2$ may be different due to:
  - Jitter
  - Queuing Delays
  - Asymmetric Paths
- Multiple measurements to filter out all invalid data we can

Approach – Siegel-Estimators

- Estimate RTTs and time offsets by linear functions
- Robust estimation by using repeated median
- Resistant against up to 50% outliers
- Slopes indicate “confidence”
Approach – Reducing the History

- Longer history → More resistant against short term changes
- But
  - Slower adaptation
  - More computations required
- History thinned out over time using Zipf distribution
- Newer values are more emphasized
- Old values still have an influence

Synchronization Step

- Offset estimates of different partners are aggregated
- Weighted median assures bad estimates and outliers have no influence
- Dampening assures over compensation

---

**Input:** estOffsets, confidences

1. medOffset ← wMedian(estOffsets, confidences);
2. if medOffset < minAdjOffset then
   3. return;
4. adjustTime(medOffset · dampFactor);
5. foreach neighbor n do
   6. offsets[n].adjustBy(medOffset · dampFactor);
Evaluation – Global operation

- Uses unicast only → might work globally
- But: What about asymmetric paths?
- Experiment:
  - 32 runs
  - Internet Delays
  - \( \gamma \)-distributed Jitter

→ No significant influence!

Evaluation – Synchronization

- Adaptation guaranteed, if and if graph of synchronization partners is
  - Strongly connected
  - No sub-graphs exist where all nodes are more connected to the sub-graph than to the outside
- Fortunately: This is the case for expander graphs and thus most peer-to-peer systems
  (Short proof in the paper)
Evaluation – Integrity

• Which influence have 10% of attackers (in a VPN with 100 nodes)?
• Attackers try to circumvent filter by gradually increasing reported offsets
• Measured $\sigma$ after 500 synchronization steps
  → Some nodes are pulled away, but as offset increases filter works

Evaluation – Scalability (I)

• How does VPN size affect synchronization precision?
• Measured $\sigma$ in steady state for growing VPN size
  → No significant impact!
Evaluation – Scalability (II)

• How does VPN size affect time to stabilization?
• Measured steps until error < 0.1s for growing VPN size
→ Sub-linear impact despite logarithmic scale!

Conclusion & Outlook

• Scalable & robust approach to synchronize clocks in distributed systems
• Can be applied always if network graph has expansion properties
• Optimizations still possible, e.g.:
  – Weighting of confidence factors
  – Blacklisting to avoid adaptive attackers
Thank you for your attention!

Michael Rossberg
michael.rossberg@tu-ilmenau.de
Ilmenau University of Technology
Germany