Thoughts on a New Architecture for Network Timing

Darryl Veitch
dveitch@unimelb.edu.au
http://www.cubinlab.ee.unimelb.edu.au/∼darryl
Collaboration with **Julien Ridoux**

CUBIN, Department of Electrical & Electronic Engineering
The University of Melbourne

First GENI Measurement Workshop
Madison Wisconsin, June 26, 2009

The Importance of Network Timing

Canonical Applications

- Network measurement (delay; delay variation/jitter, RTT, IAT)
- Time critical applications (distributed gaming, automated trading)
- Latency a hard constraint for distributed systems
  – if you can't beat it, manage it

The NTP System

- One of the oldest IP protocols (circa 1983)
- Provides an absolute software clock in client computers
- Best effort service over NTP[UDP[IP]]]
- Clients connect to server hierarchy:
  Stratum1 <–> Stratum2 <–> Stratum3 <–> ...
- Low cost (software on commodity hardware), except for Stratum1 servers
- Client–Server mode message exchange (LAN broadcast rarely used)
- Feedback based ‘servo’ (synchronization algorithm)

LIMITATIONS

- Accuracy: nominal [1ms] doesn't push hardware limits [5μs]
- Robustness: feedback can be unstable (–> 100ms error)
- Modularity: couplings between algo, timestamping, system clock, ...
- Configurability: static and inflexible
  – servers taken from static list
  – algo parameters magic numbers (stability not guaranteed)
- Codebase: complex and undocumented code with many parameters
- Performance Monitoring: self-assessment unreliable

LIMITATIONS

- Accuracy: nominal [1ms] doesn't push hardware limits [5μs]
- Robustness: feedback can be unstable (–> 100ms error)
- Modularity: couplings between algo, timestamping, system clock, ...
- Configurability: static and inflexible
  – servers taken from static list
  – algo parameters magic numbers (stability not guaranteed)
- Codebase: complex and undocumented code with many parameters
- Performance Monitoring: self-assessment unreliable
- Neglects important synchronization issues
  - provision of difference clocks as well as absolute clocks
  - support for feedforward algorithms
  - path asymmetry management

The Need for Difference Clocks

Difference Clock: a clock $C_d(t)$ uncorrected for drift, used for measuring time differences below a critical timescale τ^*.

The Potential of a Difference Clock

- A stable rate (good to 10^{-7}) implies accurate $\Delta(t)$ measurement (oscillator drift, over enough small timescales, can be ignored)
- Example: error in RTT of 100ms just 10ns
- Average rate can be extremely robustly measured to this accuracy
- Accuracy of $\Delta_d(t) = C_d(t_1) - C_d(t_2)$ therefore of same order

The Folly of Absolute Differences

- Absolute clock $C_a(t)$ requires constant correction to negate drift
- Inherently difficult bias/variance tradeoff (network delay variability *vs* out of date timestamps)
- Estimated drift correction very noisy, pollutes $\Delta_a(t) = C_a(t_1) - C_a(t_2)$

Result: underlying stable rate wasted through adding a poor estimate of zero.

Path Asymmetry and its Immediate Impact

Fundamental Ambiguity

Asymmetry $A \equiv d^{\uparrow} - d^{\downarrow}$ and $2 \cdot \text{ClockError}(t)$ non-unique up to a constant.

Impact on Absolute Clock

- A unknown: generally forced to assume $A = 0$
- However, bounded by minimum RTT: $A \in (-r, r)$
- Creates constant errors from 5μs to 100's ms
- Causes jumps when server changed
- $\rightarrow$ Important to use a single, close, server.

Impact on Difference Clock

- None
- Difference clock can be used to measure r

The Need for Path Asymmetry Management

Problems arise in Multiple Server/Client Scenarios

- Perceived causality failure: measure OWD could be negative!
  – break software
  – confuse delay reporting
  – cause accurate clocks to be reset
- Change of server results in jump in measure OWD's
  – perception that OWD has changed when it hasn't
  – previously believable delays may turn 'unbelievable'

The Need to Avoid 'Asymmetry Jitter'

- Any change of server or route, even improvements, can cause problems
- Initialize estimated A values very carefully
- Select server with servers of other clients in mind

An Outline of a New System

Objectives

- Highly robust synchronization algorithms achieving limits set by underlying hardware and network
- Difference clock support at all levels
- Asymmetry awareness at all levels
- More advanced stratum1 layer
- Modular design
- Quality monitoring

An Outline of a New System

Objectives

- Highly robust synchronization algorithms achieving limits set by underlying hardware and network
- Difference clock support at all levels
  - Client algorithms
  - Timestamp and packet formats
  - System support (eg raw counter timestamping in kernel)
  - Universal uni-directional mode (supports most measurement needs)
    – wired and wireless LAN
    – WAN (multicasting for highly scalable rate synchronization)
- Asymmetry awareness at all levels
- More advanced stratum1 layer
- Modular design
- Quality monitoring

An Outline of a New System

Objectives

- Highly robust synchronization algorithms achieving limits set by underlying hardware and network
- Difference clock support at all levels
- Asymmetry awareness at all levels
  - Server selection (initial, backup, and for delay among client groups)
  - For asymmetry jitter management and error bounds
- More advanced stratum1 layer
- Modular design
- Quality monitoring

An Outline of a New System

Objectives

- Highly robust synchronization algorithms achieving limits set by underlying hardware and network
- Difference clock support at all levels
- Asymmetry awareness at all levels
- More advanced stratum1 layer
  - Recognise status as core infrastructure -> harden against abuse
  - Certification for synchronisation quality
  - Interconnected of layer to support quality and disperse asymmetry data
- Modular design
- Quality monitoring

An Outline of a New System

Objectives

- Highly robust synchronization algorithms achieving limits set by underlying hardware and network
- Difference clock support at all levels
- Asymmetry awareness at all levels
- More advanced stratum1 layer
- Modular design
  - Separation of timestamping, synchronisation, asymmetry management, network layer, difference and absolute
  - Unification of IEEE1588, network based, wirelessLAN synchronization
- Quality monitoring

An Outline of a New System

Objectives

- Highly robust synchronization algorithms achieving limits set by underlying hardware and network
- Difference clock support at all levels
- Asymmetry awareness at all levels
- More advanced stratum1 layer
- Modular design
- Quality monitoring
  - Accurate error bound estimates
  - Validation methodologies techniques for algorithms
  - Server and client monitoring tools and services (at multiple resolutions) – 'tracesync'
  - Accreditation and certification

An Outline of a New System

Technical Elements

- Retention from existing approach
- For replacement
- To be introduced

An Outline of a New System

Technical Elements

- Retention from existing approach
  - Low cost, using software and existing hardware
  - Server hierarchy
  - In-band solution over 'IP'
  - One class of service
  - No follow up pkt
- For replacement
- To be introduced

An Outline of a New System

Technical Elements

- Retention from existing approach
- For replacement
  - Synchronisation algorithms
  - Naive ‘multi-homing’ of clients for ‘robust’ synchronisation
- To be introduced

An Outline of a New System

Technical Elements

- Retention from existing approach
- For replacement
- To be introduced
  - Feedforward not feedback (except possibly for stratum-1)
    – enables difference clocks
    – inherently more robust
  - Decoupled synchronization and path asymmetry management algorithms
    – synchronization (avoid multihoming but vet candidates)
    – asymmetry (talk to relevant servers and clients)

THE RADCLOCK PROJECT
(ROBUST ABSOLUTE AND DIFFERENCE CLOCK)

- Aims to provide the next generation timing system for packet networks
- Client software released under GNU GPL for Linux & BSD
- Based on server-client, feedforward, asynchronous paradigm
- Provides difference and absolute clocks
  – algorithms
  – timestamping support (user and kernel)
  – API
- Exploits common counters (TSC, HPET, ACPI..) ["formally" TSCclock]
- Highly robust, accuracy pushes system noise
- Validated with state of the art testbed and methodology
- Now working on server and asymmetry issues
- Code and Papers at *http://www.cubinlab.ee.unimelb.edu.au/radclock/*

Sponsors: CUBIN, ARC, Cicso, Google