Timing and Bandwidth Issues in Active Measurement

or

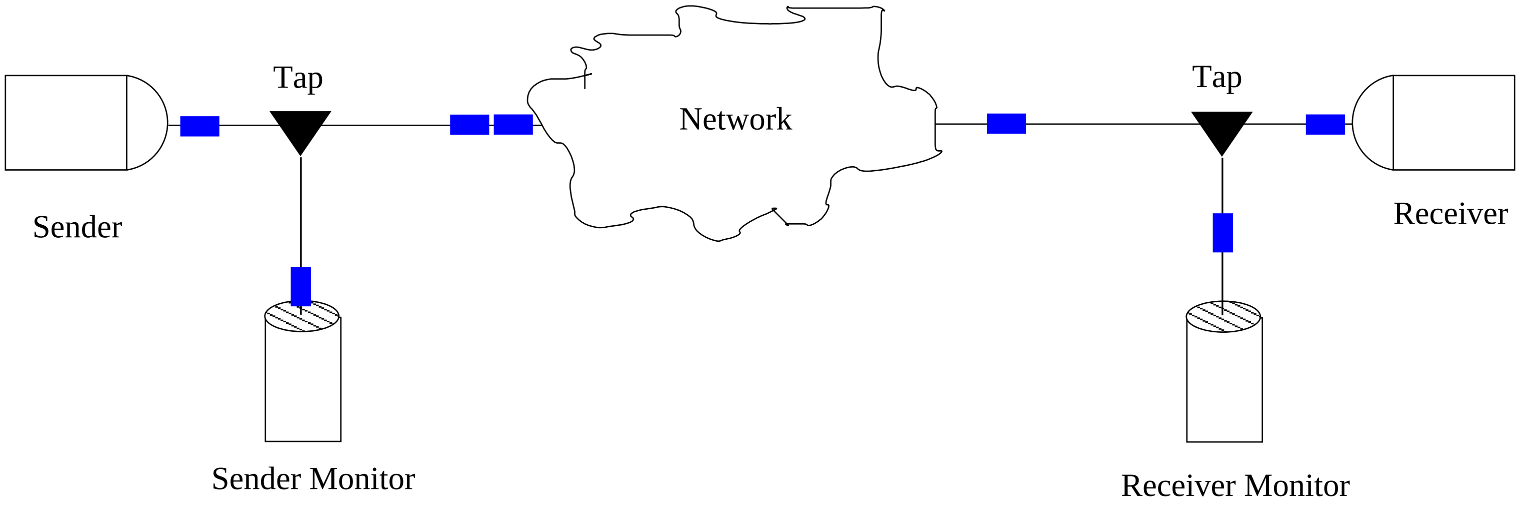
Physical Constraints on Active Probing

MICRO-TUTORIAL

ISMA 2003 Bandwidth Estimation Workshop (BEst)

Darryl Veitch

Essentials of Active Measurement



- Probe packets are sent from sender to receiver.
- Arrival and Departure times, and losses, are monitored.
- Measurements used to infer network characteristics and conditions.

As loss is rare, Timestamps are central.

Physical layer constraints and software limitations both affect precision.

Factors Affecting Raw Timestamps

The Probing Software [sender, sender monitor, receiver monitor]:

- The software clock and its synchronisation
- Location of software timestamping
- Interfaces to operating system
- Degree of kernel integration
- System scheduling behaviour
- System and event definitions

The Probing Hardware [PC, clock reference, hardware monitor]:

- PC clock stability
- Reference clock reliability and availability
- Interrupt latencies
- Location of monitor tap (if in hardware)
- Kernel - NIC communication

The Network [links, NIC, hubs, routers, switches]:

- Architecture of switching elements (FIFO, store & forward, slow/fast path)
- Hardware clock rate in switching elements
- Link layer multiple paths

A Hierarchy of Probing Accuracy

- **Low end:** $ Ethernet card, PC.
  Unix, Software clock, NTP, `tcpdump`, User sender/receiver.
- **‘Common’ GPS solution:** $ $ $ Ethernet card, PC, GPS.
  Unix, GPS synchronised clock, `tcpdump`, User sender/receiver.
- **Linux–TSC solution:** $ Ethernet card, PC.
  Unix, TSC clock, driver timestamper, User sender/receiver.
- **RT–Linux–TSC solution:** $ Ethernet card, PC.
  Unix, TSC clock, driver timestamper, RT sender/receiver.
- **A Reference solution:** $ $ $ $ DAG3.2e cards, GPS, Ethernet card, PC.
  GPS sync’d DAG monitors, Unix, TSC clock, driver timestamper, RT sender.
- **High end:** $ $ $ $ $ All hardware solution.

Obstacles to Inexpensive Accuracy

‘Features’ of the Low End: the SW–NTP–tcpdump solution

- The Standard Software Clock (SW):
  - Based on two underlying oscillators with large skews.
  - `getimeofday()` has 1μs resolution and takes 1μs to call.
- SW Synchronisation under NTP:
  - Offset: only bounded to $\approx$1ms under optimal conditions.
  - Rate: altered to control offset! up to 500PPM!!
- System Noise under Unix (Linux, BSD):
  - Uncontrolled scheduling delays in setting, synchronising, reading, sending..
  - Hardware interrupt latencies.
- Timestamping and Sending
  - `tcpdump` timestamps with `getimeofday()` after driver.
  - User sender tries to schedule using `getimeofday()` and hopes for the best.

Accuracy Comparison

TSC: CPU cycle register.

Infrastructure	Timing Accuracy Metric		
	Offset	Skew	System Noise
Low End	1ms –	5 – 500 PPM	$10\mu s$ – 10ms
Common GPS	$10\mu s$	5 – 50 PPM	$10\mu s$ – 10ms
Linux-TSC	0.1 – 2ms	0.1 PPM	$1\mu s$ – 1ms
RT-Linux-TSC	0.1 – 2ms	0.1 PPM	$1\mu s$ – $10\mu s$
Reference	100ns	0.01 PPM	$<$ 100ns
All Hardware	$<$100ns	$<$0.01 PPM	$<$ 100ns

System Noise: use TSC timestamper (in driver), and RT–Linux.

Skew: use TSC with accurate remote calibration.

Offset: use TSC and nearby NTP primary server timestamps.

Limitations at High Bandwidths

- Timestamps too demanding:
  p/μ: $[40, 1500]$ bytes over 1Bbps $= [0.32, 12]\mu$s
- Interrupt latencies too high, clock synchronisation insufficient
- Hardware time grid: coarse in low speed switches – wipes fine details