An Accurate and Scalable Clock Synchronization System for Commodity Networks

Balaji Prabhakar
VMware Founders Professor of Computer Science
Departments of Electrical Engineering and Computer Science
And, by courtesy, the Graduate School of Business
Stanford University

Joint work with: Yilong Geng, Shiyu Liu, Zi Yin, Ashish Naik, Mendel Rosenblum and Amin Vahdat

Background and Motivation

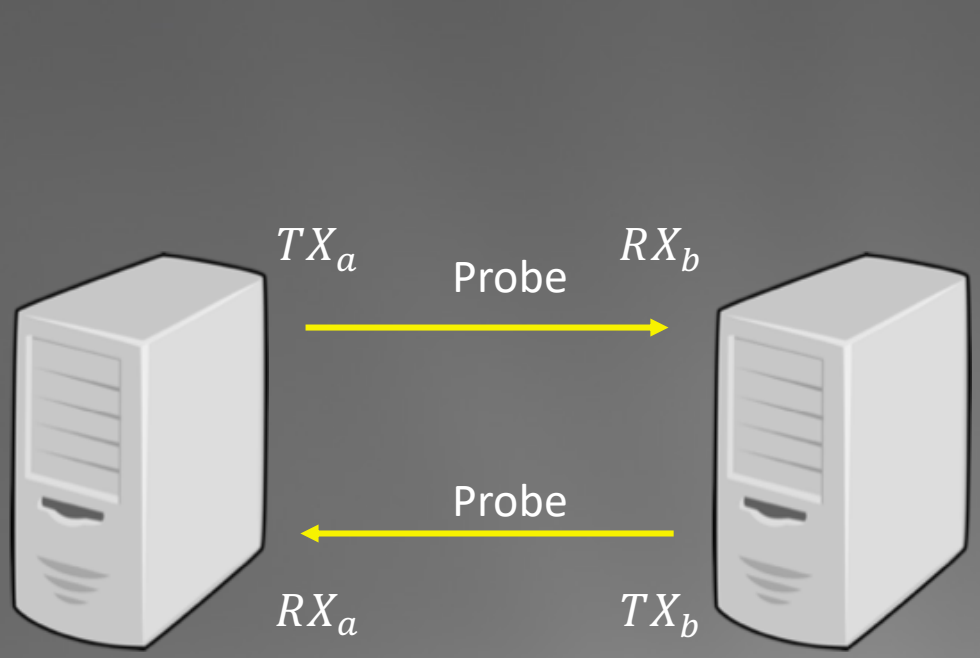
Synchronized clocks are fundamental to the operation of many distributed systems

- Distributed databases: improve performance and consistency
- Software-defined networking: enforce ordering of forwarding rule updates
- Financial systems: guarantee transaction execution order
- Network congestion control: TDMA-like flow scheduling (Fastpass)
- ……

Our goals:

- Synchronize clocks to 10s of nanoseconds in real-time and at scale
- Provide “accurate timestamping” as a fundamental primitive in data centers

Synchronizing clocks with probes



t

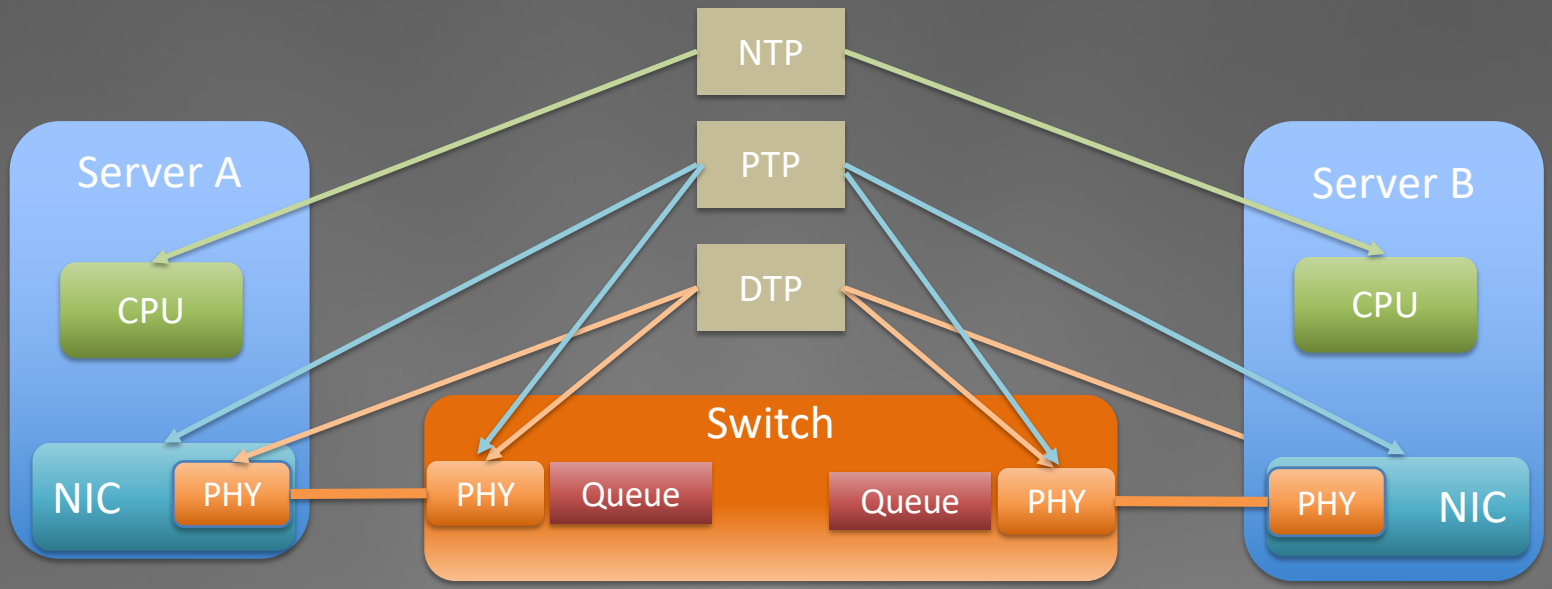
$t_a = t + \Delta t_a$

$t_b = t + \Delta t_b$

All methods of clock sync use these 4 timestamps; main differences:

1. Where to take timestamps?
2. How to process the timestamps?

Where to take timestamps



Huygens

- Uses NIC or CPU timestamps and, respectively, gets ns- and us-level sync accuracy
- *Does not need* timestamps from switches
- is a software-based approach

Huygens: Basic equations

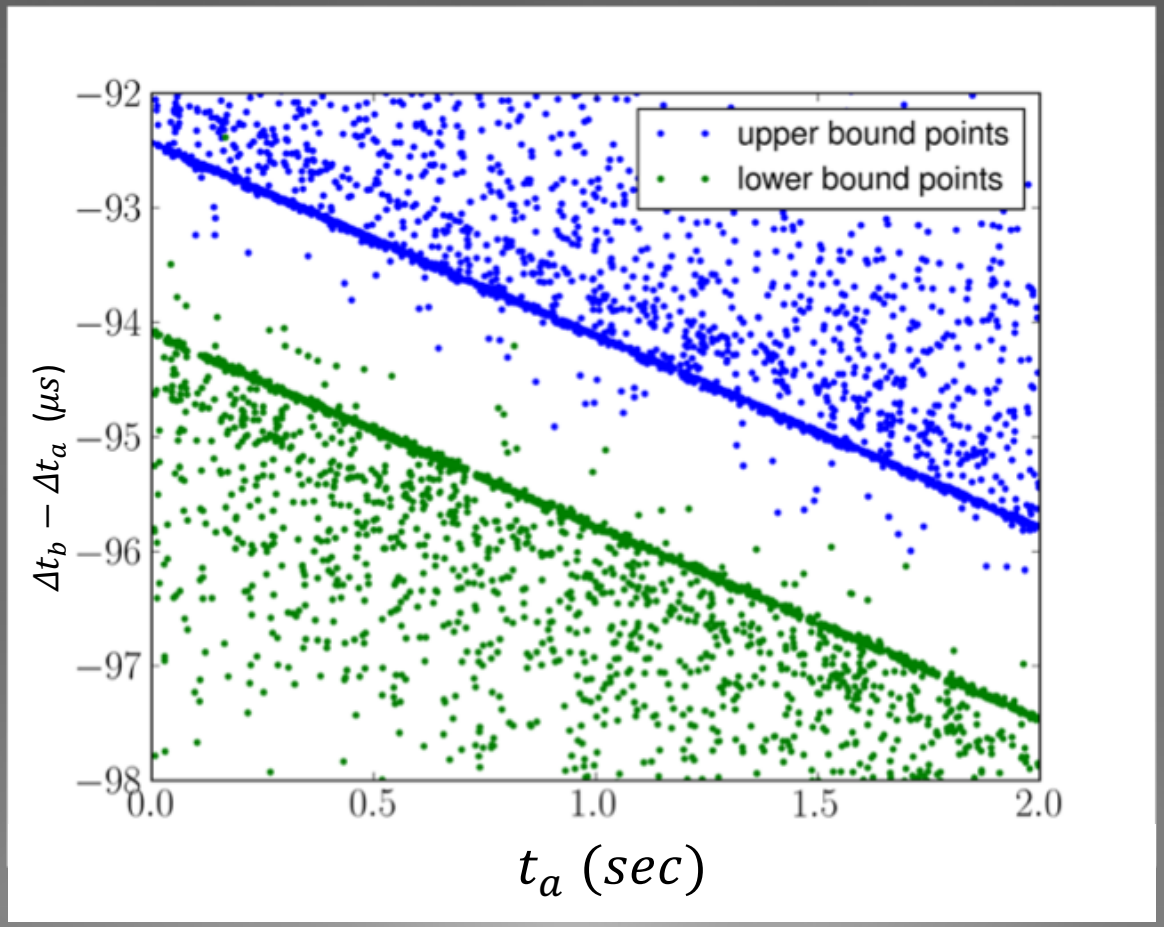
Probe from A to B:

- Receive time = transmit time + delay
- $RX_b - \Delta t_b = TX_a - \Delta t_a + Propogation\ and\ queueing\ delay$
- $\Delta t_b - \Delta t_a = RX_b - TX_a - Propogation\ and\ queueing\ delay$
- $\Delta t_b - \Delta t_a < RX_b - TX_a$

Probe from B to A:

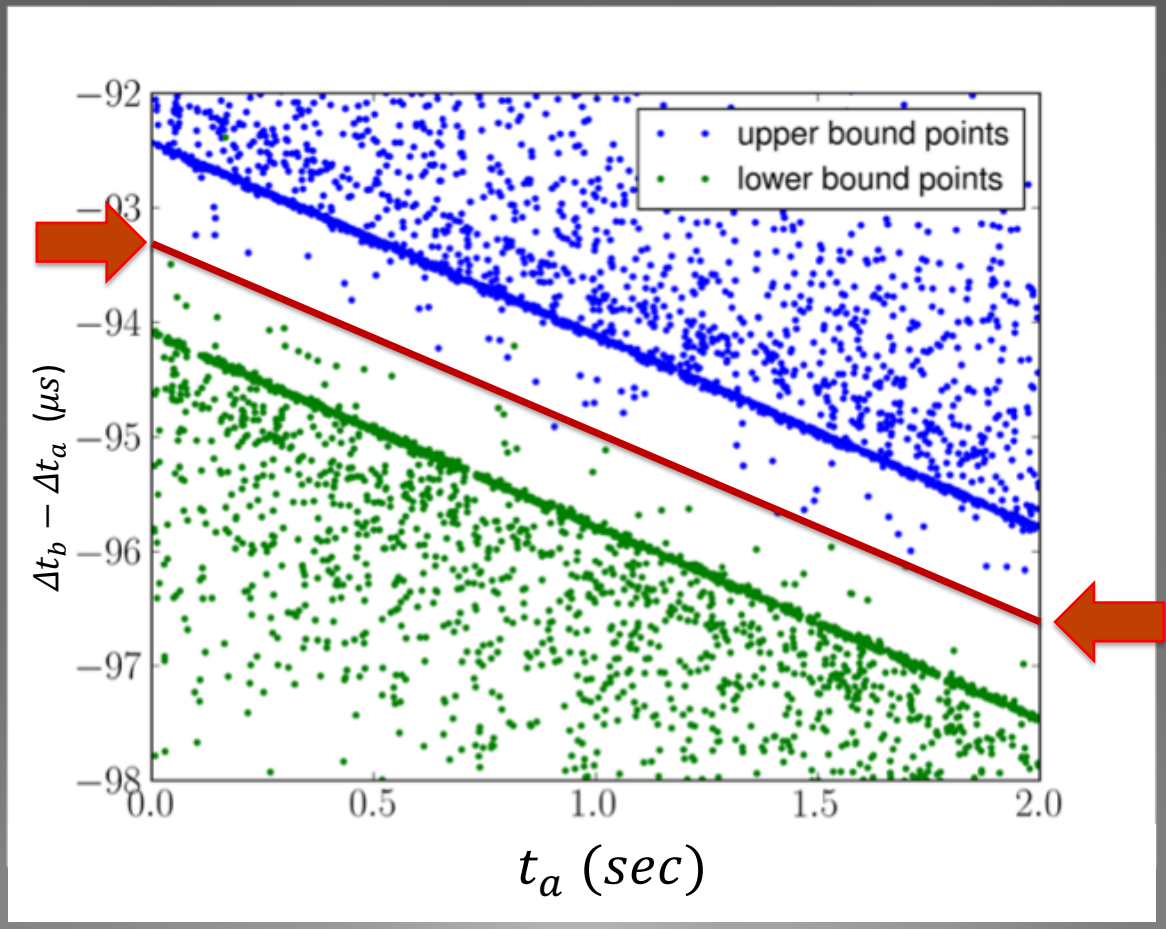
- $\Delta t_b - \Delta t_a > TX_b - RX_a$

Clock bounds over time: Google 40G testbed



Clock bounds over time: Google 40G testbed

Offset: -93.3 us

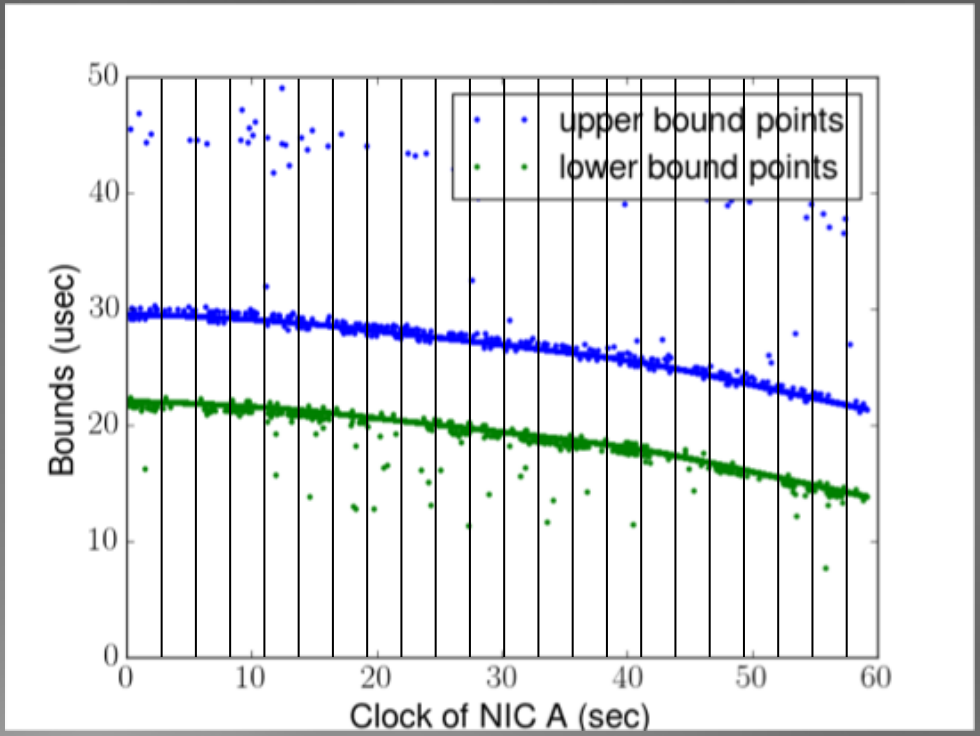


Offset: -96.6 us

Drift: -1.65 us/sec

Nonlinear clock drifts

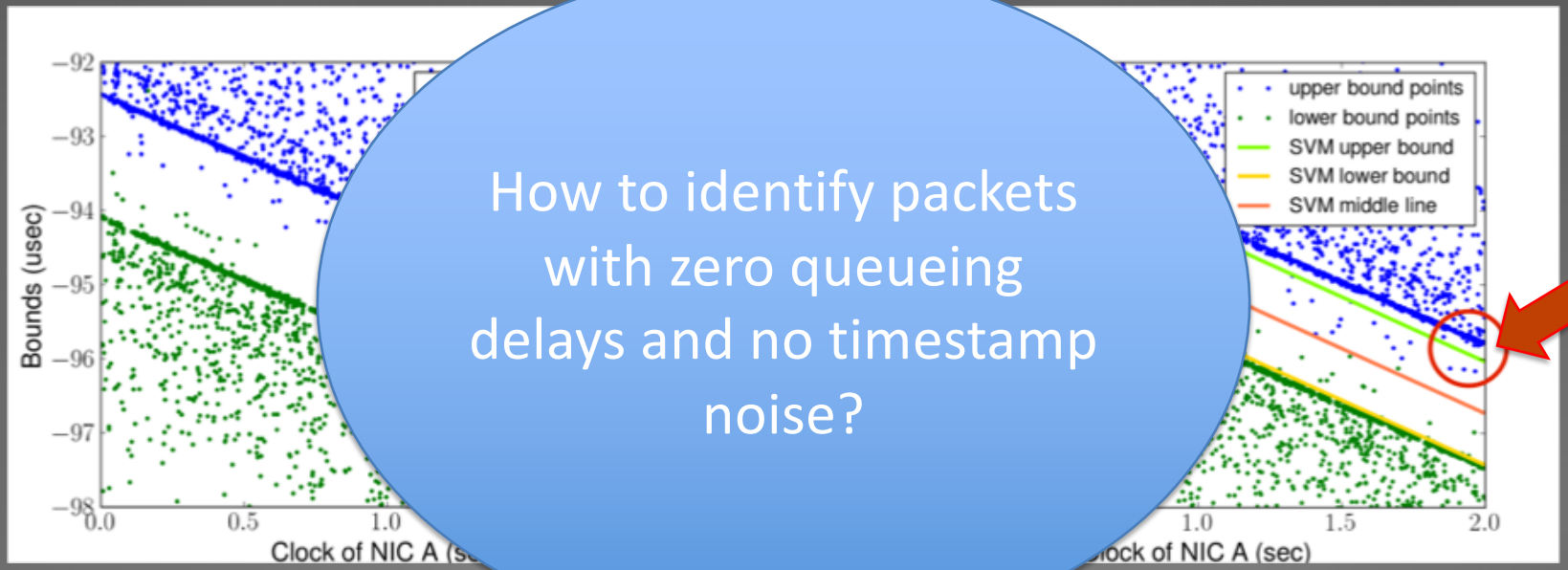
- Clock frequency varies over longer periods of time due to temperature variations
- Solution: Estimate clock offsets every 2-4 seconds since the frequency difference is nearly constant over these intervals



3 key ideas to find the red middle line

- Support vector machine
- Coded probes
- Network effect

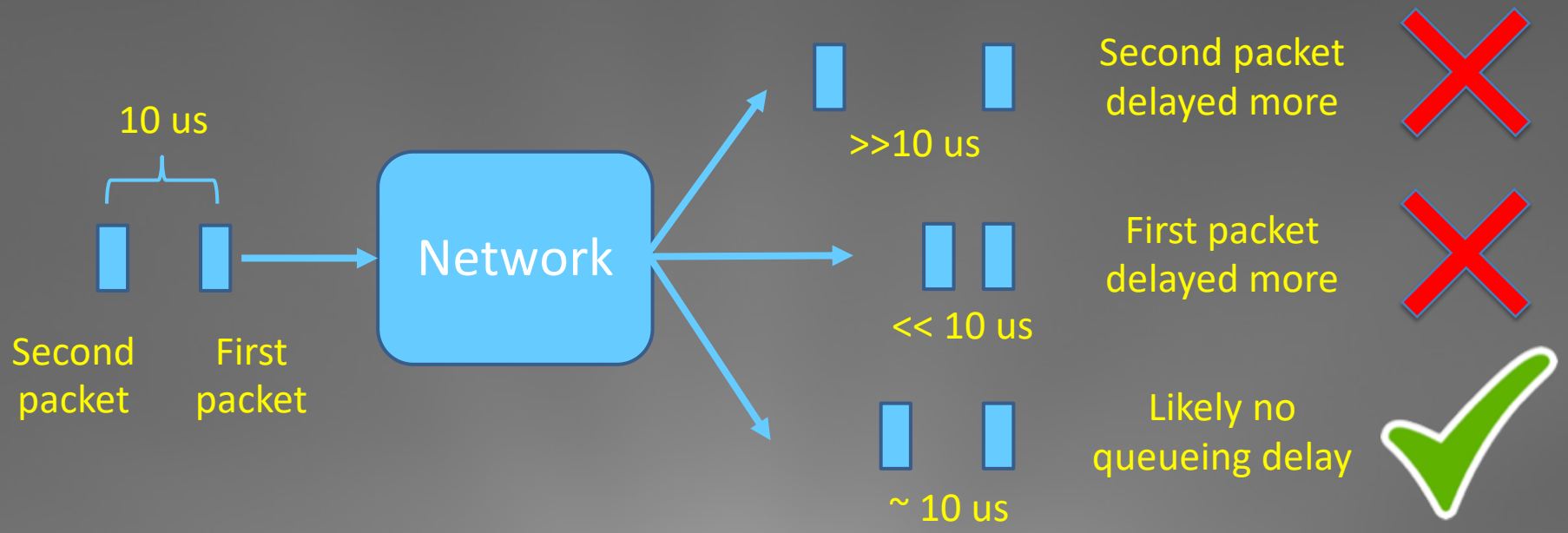
Syncing clocks with SVMs



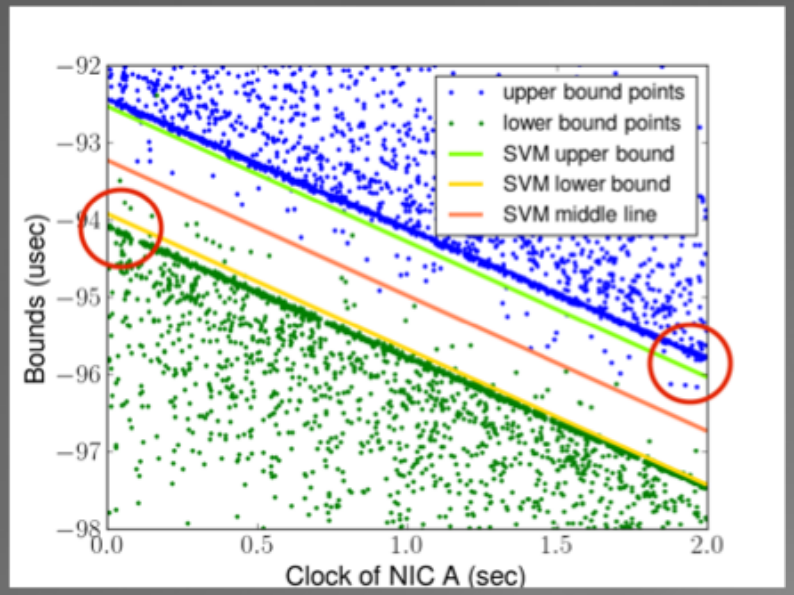
How to identify packets with zero queueing delays and no timestamp noise?

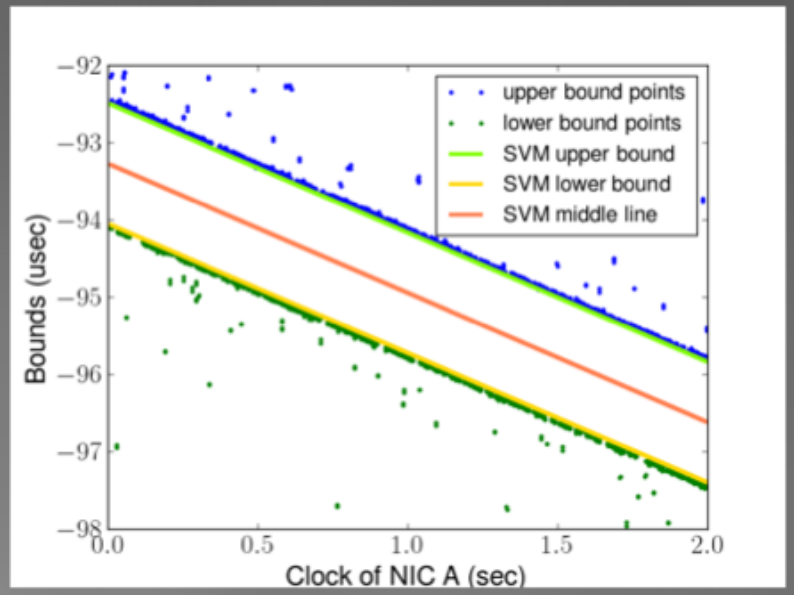
SVM achieves sync accuracy of 300~400 ns. Noisy timestamps cause synchronization errors!

Coded probes



Coded probes

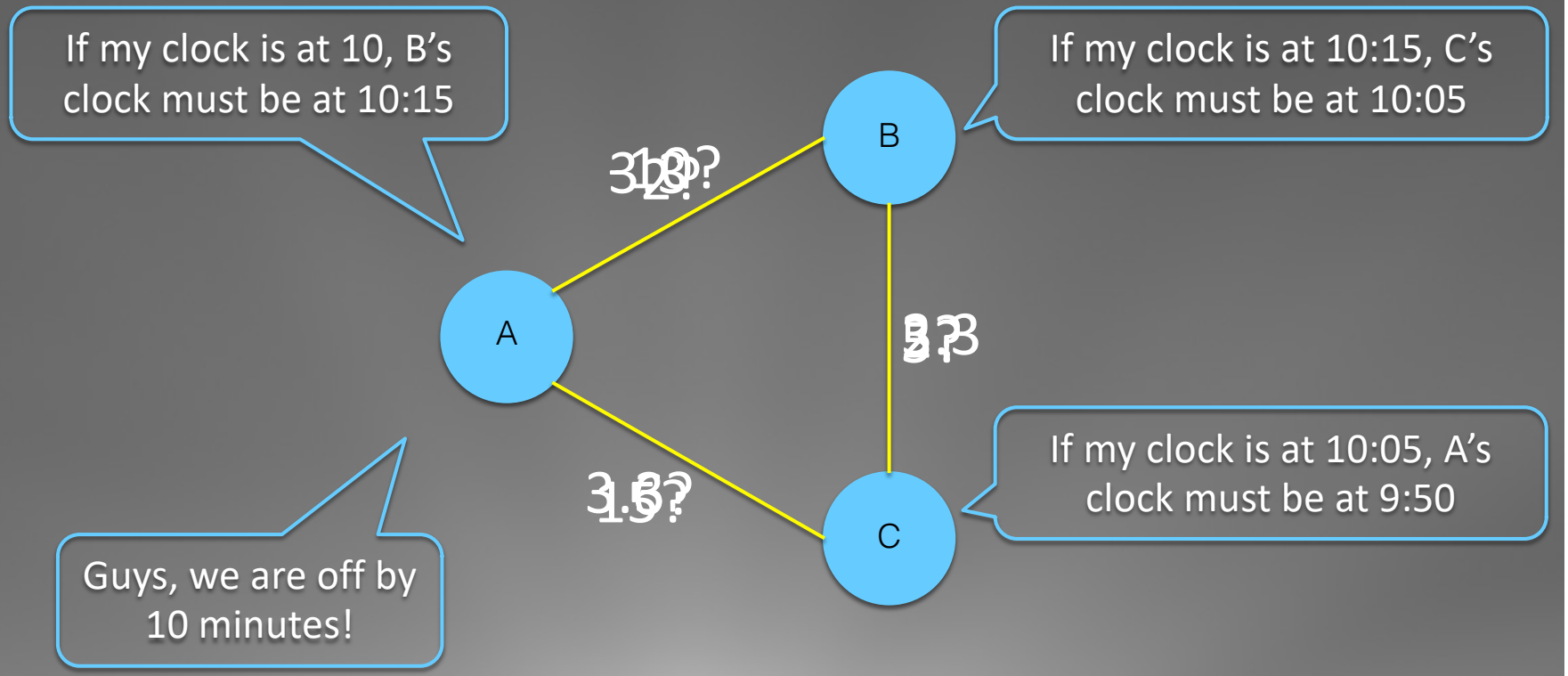




Empirically, coded probes filter out 90% of bad data and reduce the clock sync error by a factor of 4.

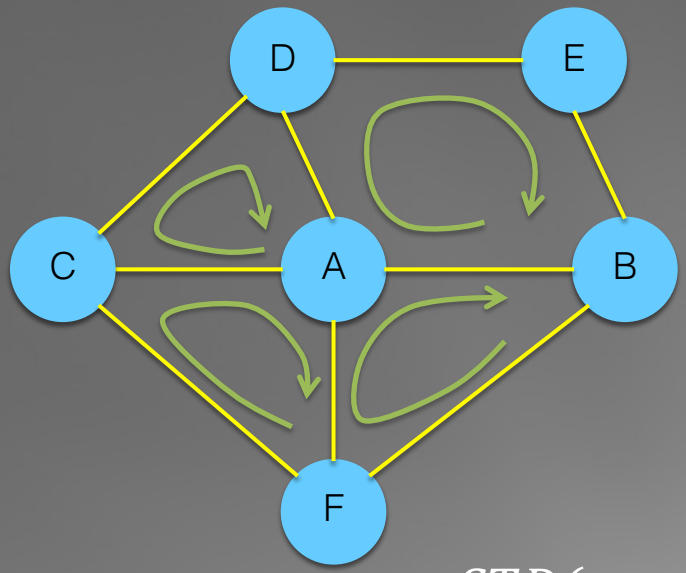
The network effect

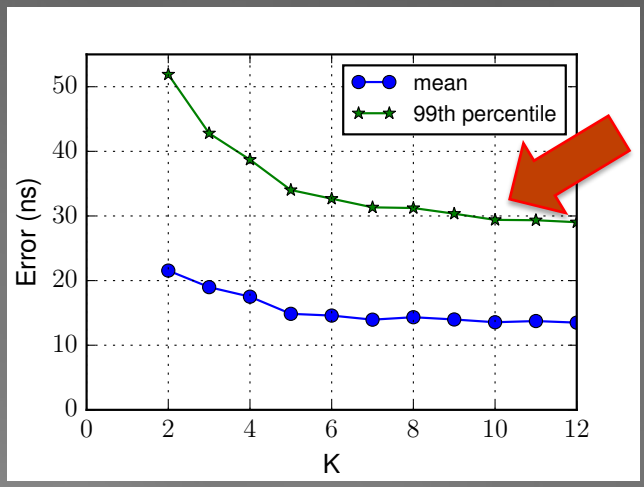
-- identifying the error



The network effect

-- fixing the error





$$STD(err\ after\ N.E.) \approx \frac{1}{\sqrt{K}} STD(err\ before\ N.E.)$$

Pilots and deployments

Google – Jupiter testbed

- 3-stage 40Gb/s Clos network
- 20 racks, 237 servers

A 10G-40G production network

- 5-stage Clos network

Stanford testbed

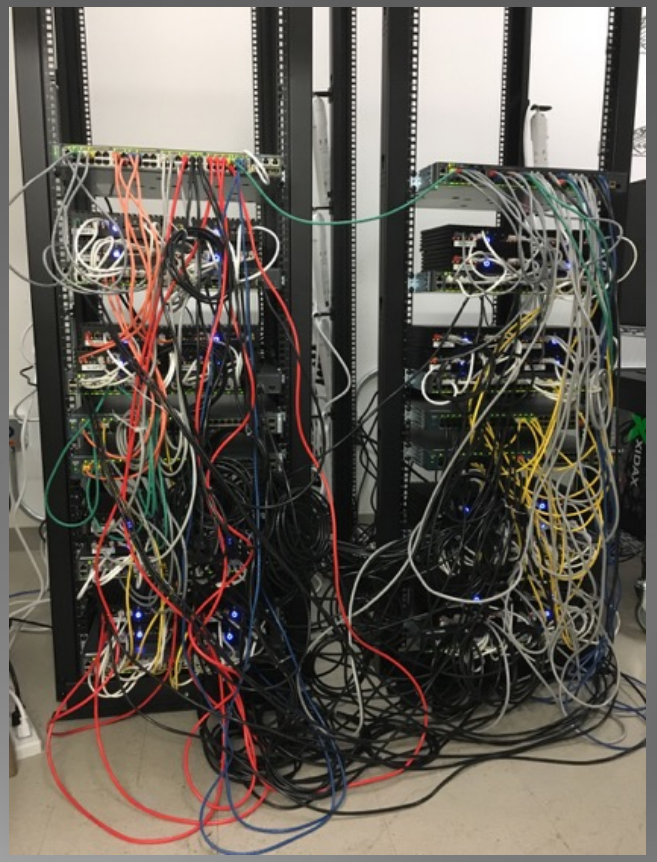
- 2-layer 1G network
- 8 racks, 128 servers
- Cisco 2960 and Cisco 3560 switches

Many financial firms

Stanford Testbed

Stanford

- 2-stage 1Gb/s Clos network
- 8 racks, 128 servers



NetFPGA verification

- Single NetFPGA acts as 4 independent NICs sharing the same clock
- Different NetFPGAs synced with I/O pins

Comparison with NTP

	NTP (with NIC timestamps)		Huygens	
	Mean abs. error	99th percentile abs. error	Mean abs. error	99th percentile abs. error
0% load	177.7 ns	558.8 ns	10.2 ns	18.5 ns
40% load	77,975 ns	347,638 ns	11.2 ns	22.0 ns
80% load	211,011 ns	778,070 us	14.3 ns	32.7 ns

Conclusion

- Huygens synchronizes
  - NIC clocks to within 10s of nanoseconds
  - CPU/VM/container clocks to within a few microseconds
- It is a software-based end-to-end system, and is lightweight and scalable (bandwidth, CPU overhead)
- Tick Tock Networks commercializing the tech
  - Contact hello@ticktocknetworks.com