Internet-Scale Alias Resolution with MIDAR

Ken Keys **Young Hyun**

CAIDA

Matthew Luckie

WAND

ISMA 2010 AIMS Workshop
Feb 9, 2010

Introduction

- Goal: We want to produce a router-level map of the Internet using Ark topology data.
  - We need alias resolution.
- Let’s try RadarGun!
  - promising technique by Adam Bender, et al
    - Fixing Ally’s growing pains with velocity modeling (IMC’08)

Outline

- RadarGun
- MIDAR Design
- Measurements
- Toward MAARS

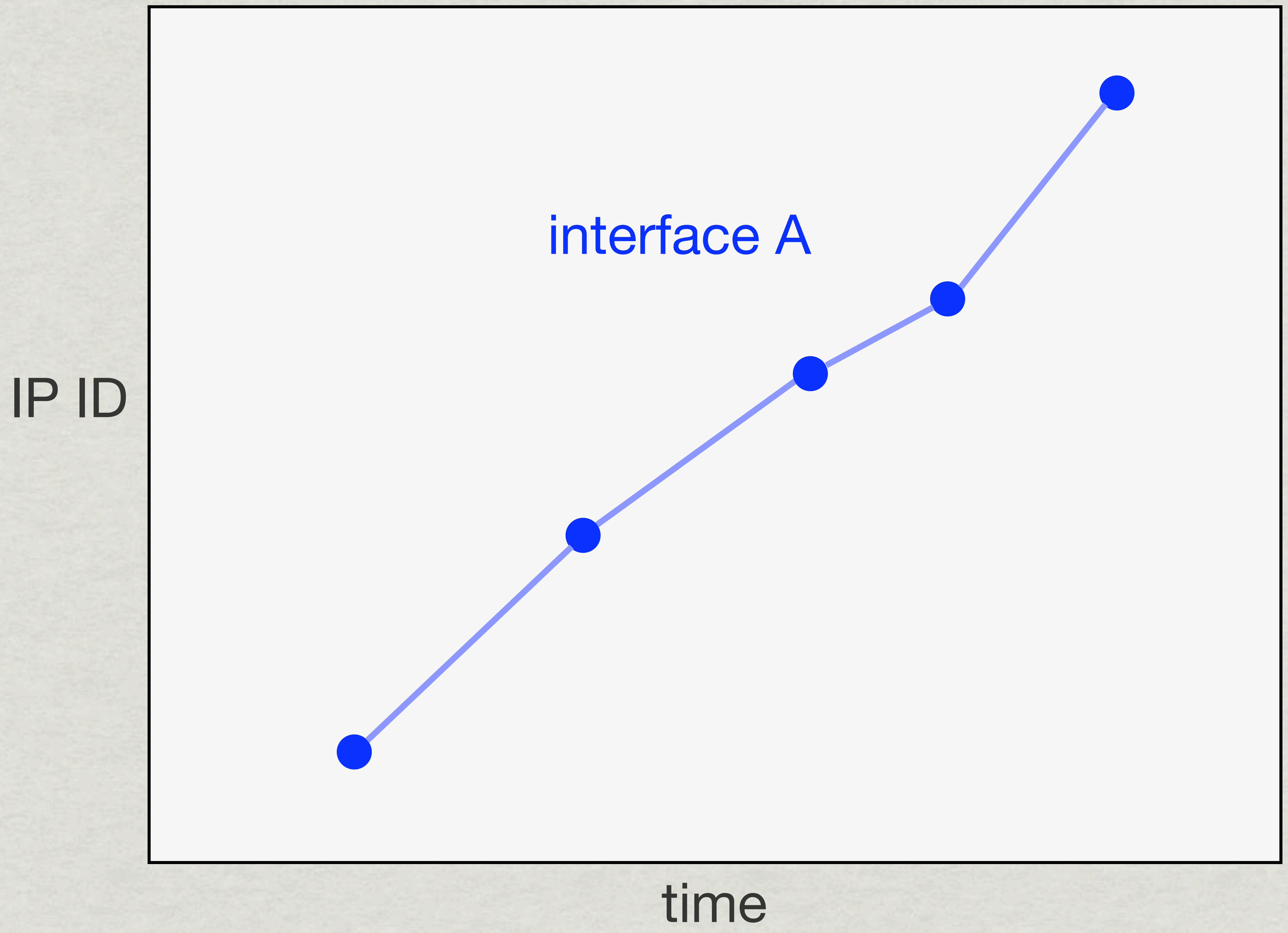
RadarGun

- based on a simple observation:
  - two interfaces belonging to the same router will respond to probes in a similar way
  - specifically, IP ID values in response packets can be used as *fingerprints* to find aliases
    - IP ID is a 16-bit value in the IP header normally used for packet fragmentation and reassembly

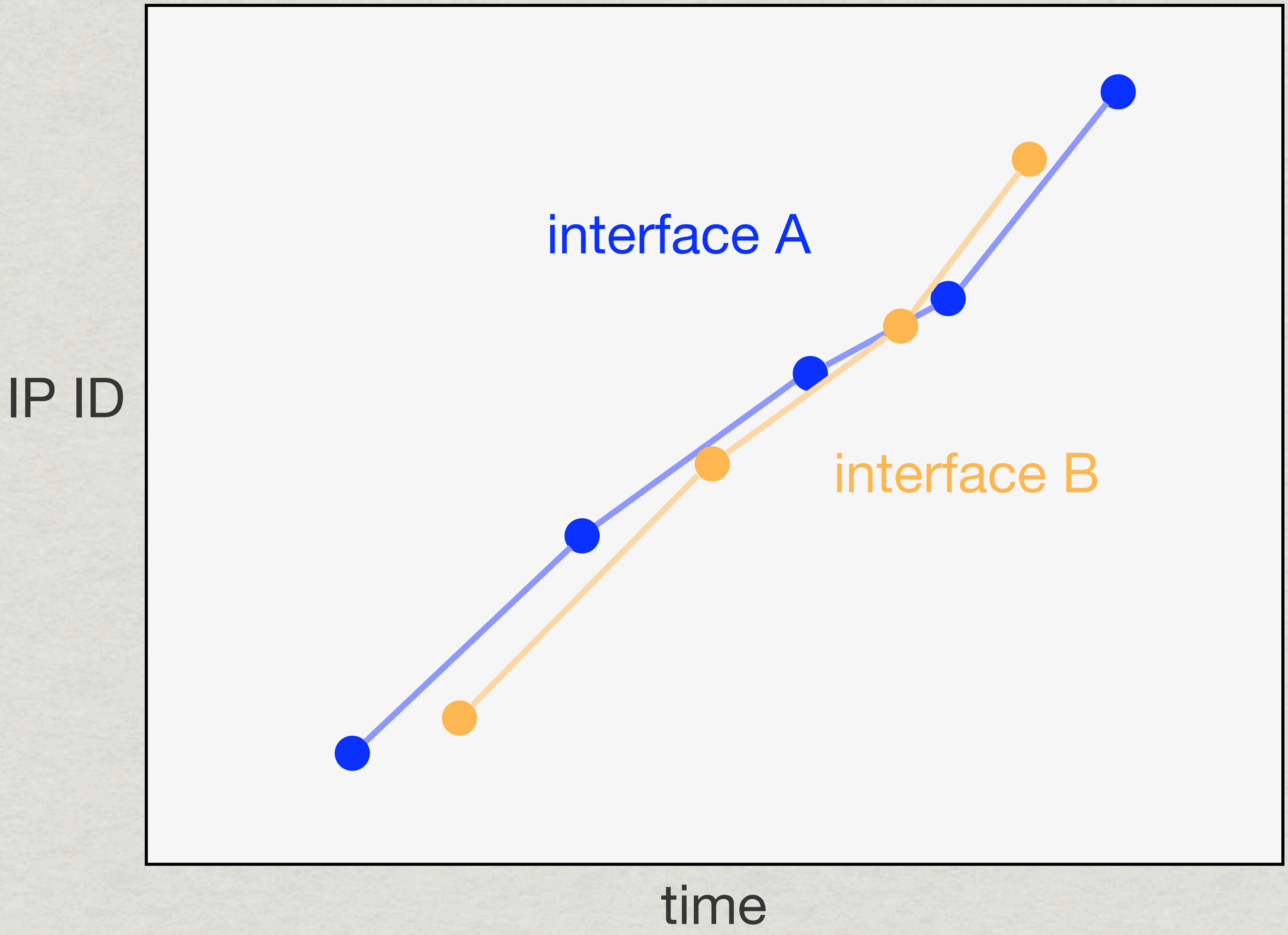
RadarGun

- assumption: a router uses an incrementing system-wide counter to generate IP ID values
  - that is, the router increments the counter whenever it sends out a packet
    - except when merely forwarding packets
- therefore:
  - two interfaces on the same router probed **closely in time** will return similar IP ID values
  - two interfaces on the same router probed repeatedly **over time** will return similar **time series** of IP ID values

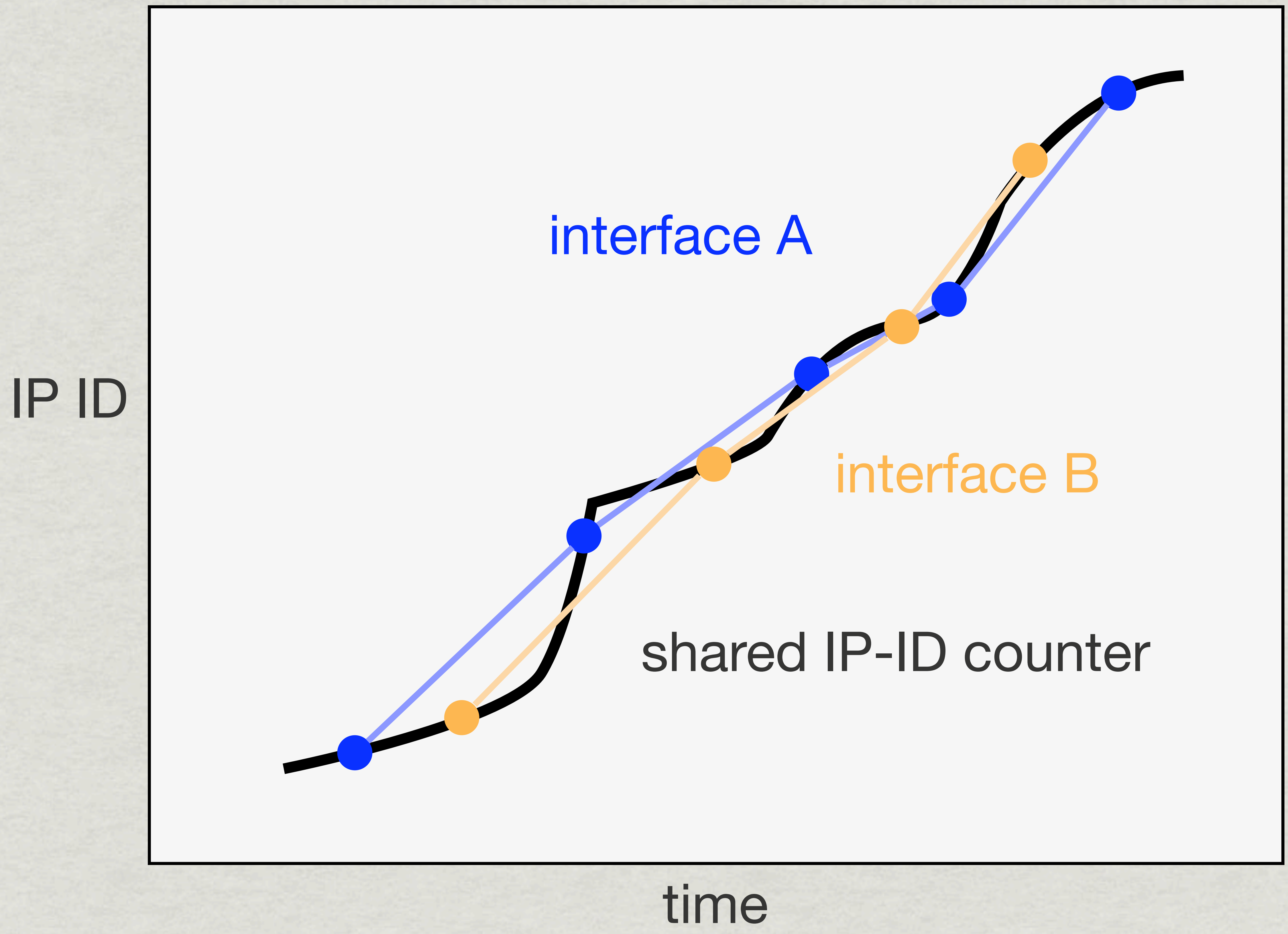
RadarGun



RadarGun



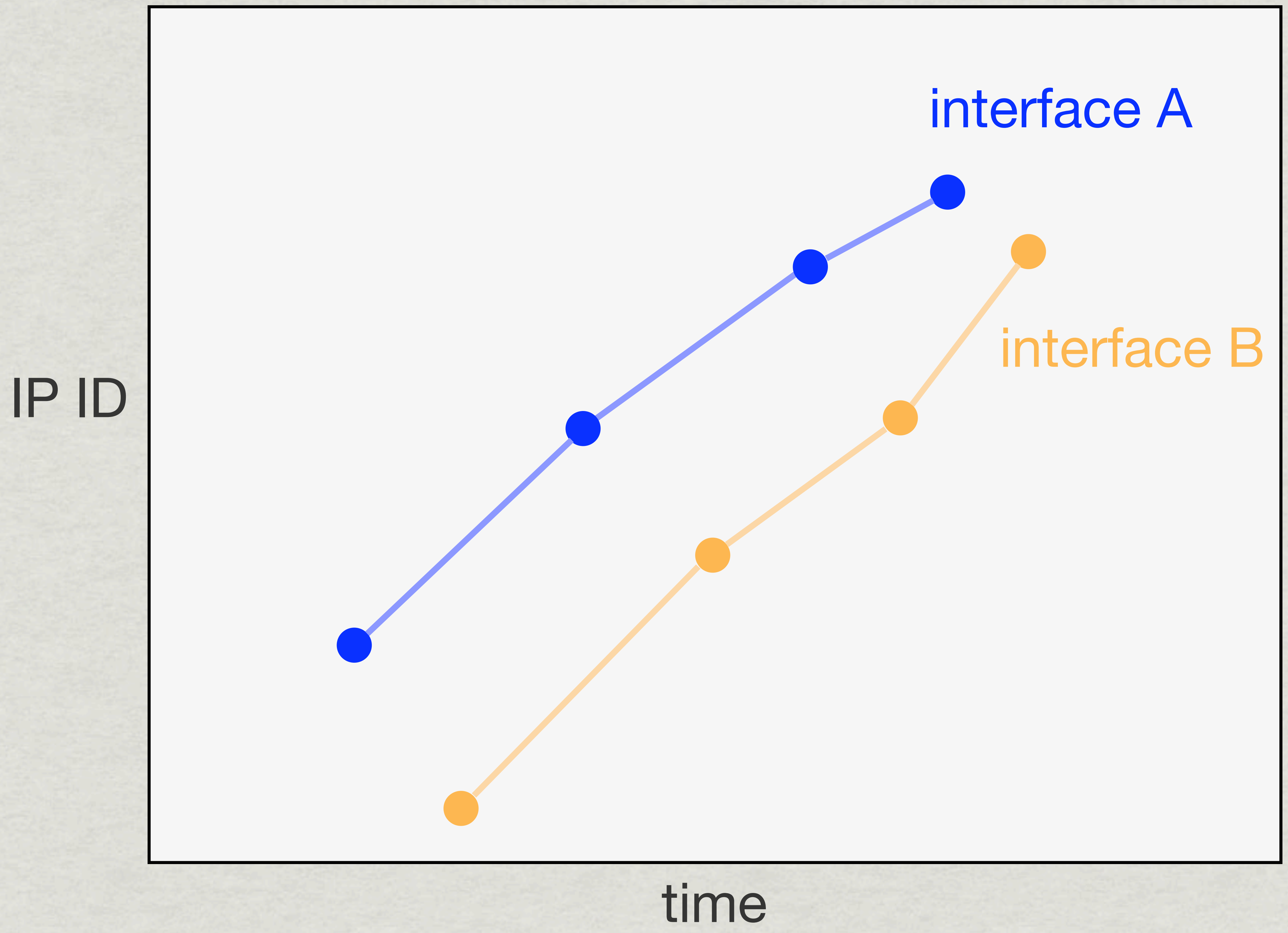
RadarGun



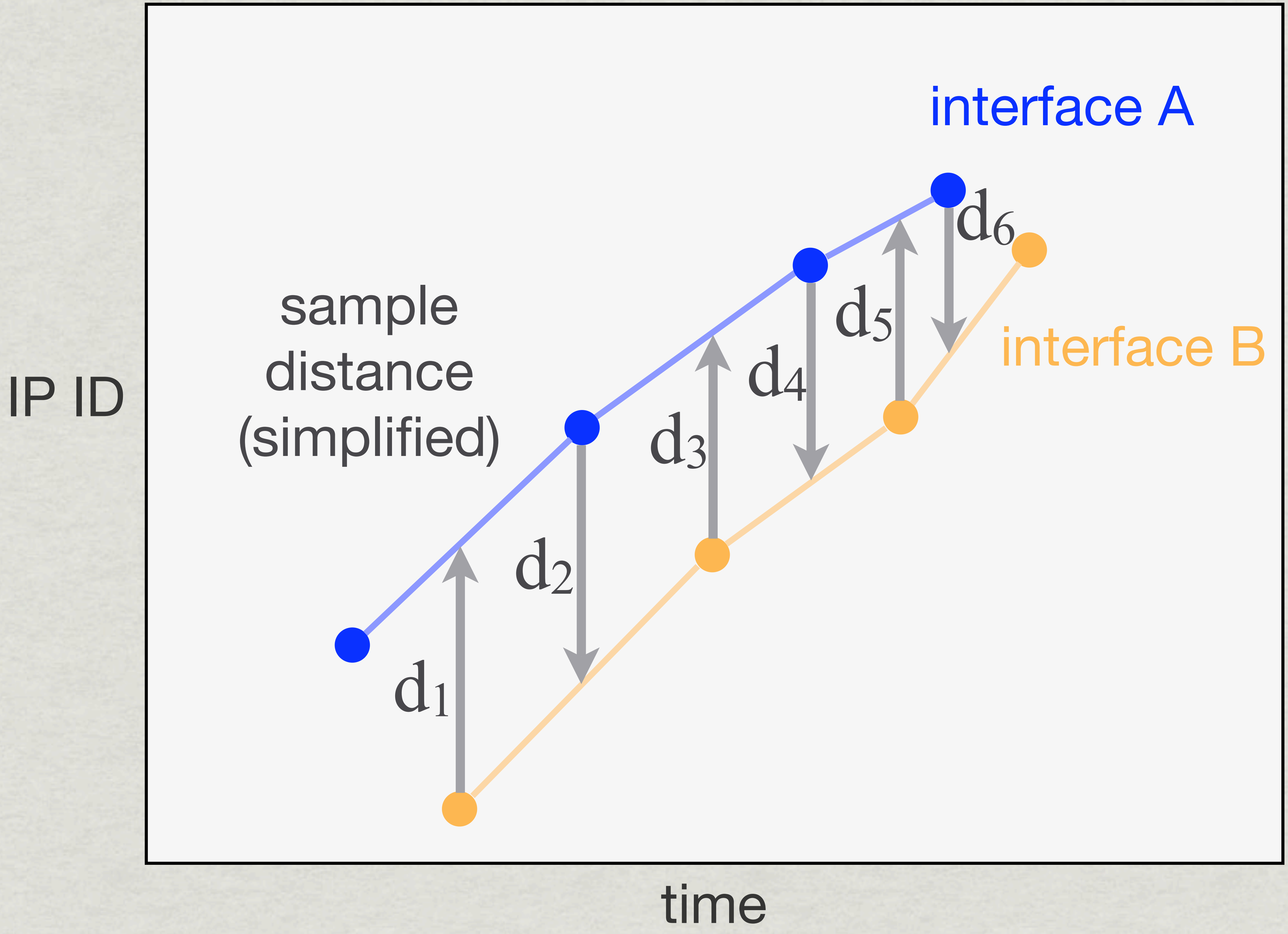
RadarGun

- RadarGun compares the IP-ID time series of two interfaces to determine whether they share a counter
  - share counter => belong to same router
- distance test:
  - compare the distance between two time series
  - if distance is “close”, then the interfaces share a counter

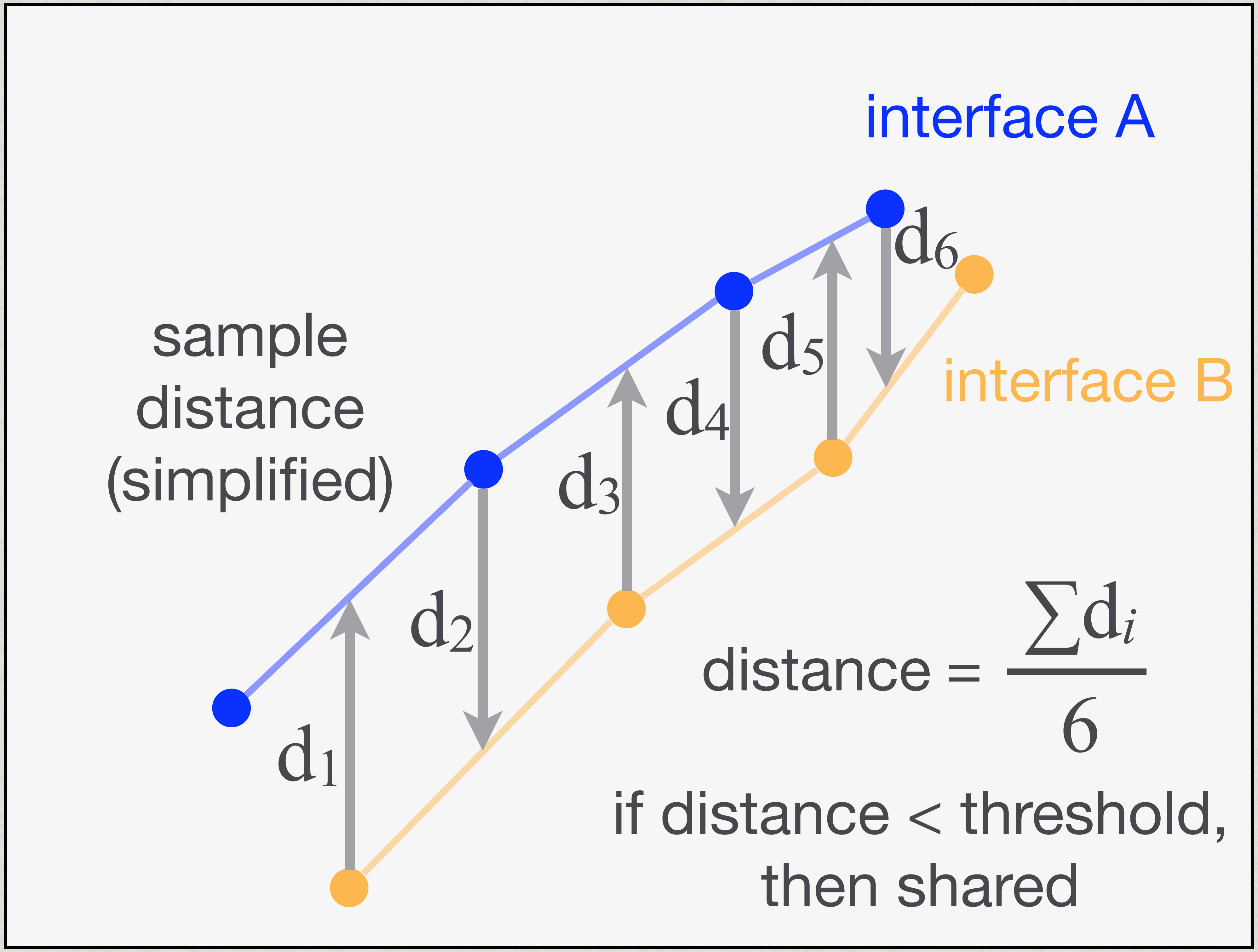
Distance Test



Distance Test



Distance Test

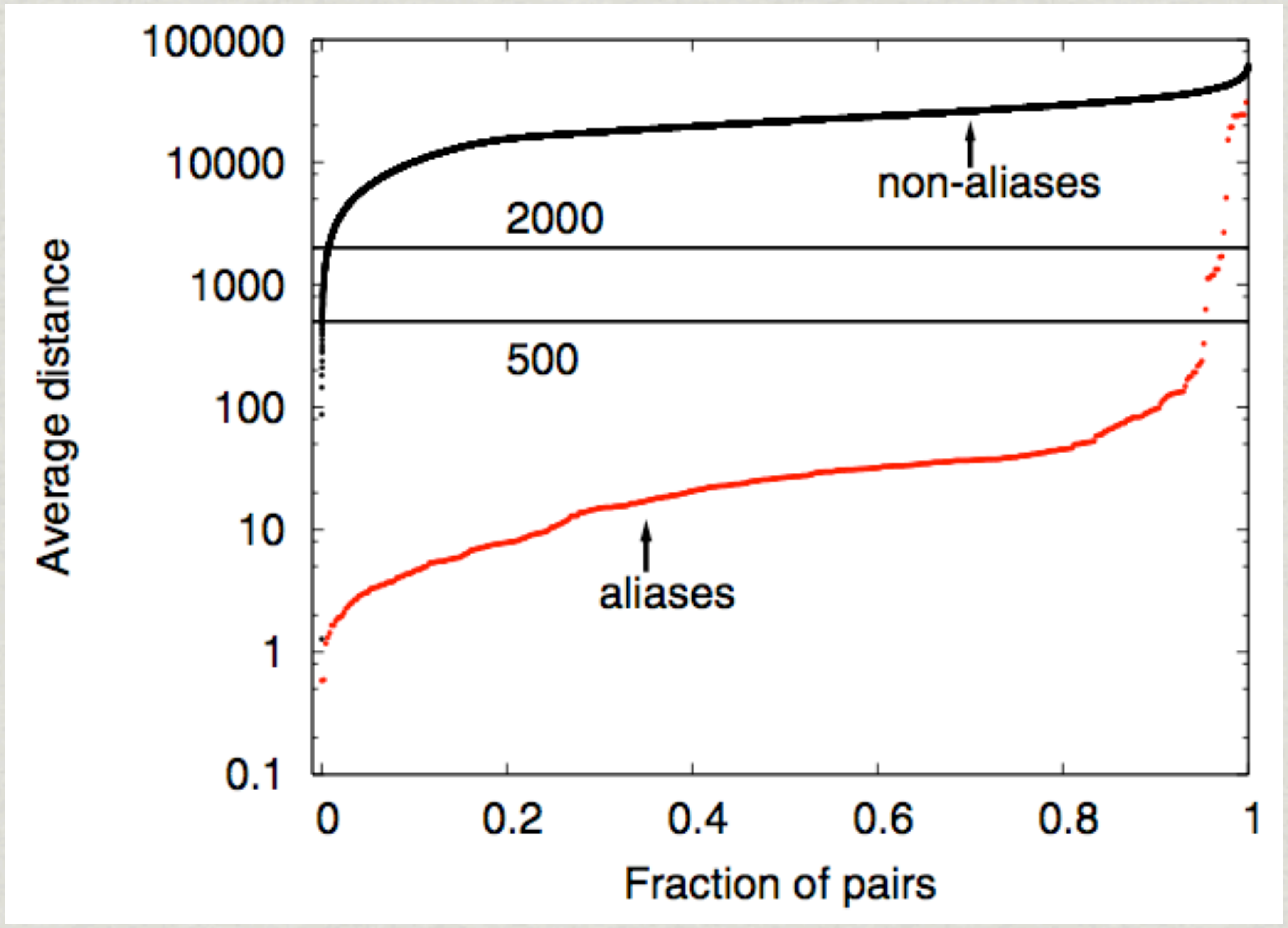


$$\text{distance} = \frac{\sum d_i}{6}$$

if distance < threshold, then shared

Distance Test

- *if distance < **500 IP ID units**, then shared counter*
  - RadarGun authors chose a threshold of 500 based on the distance distribution of alias pairs
    - 932 aliases confirmed with Mercator technique

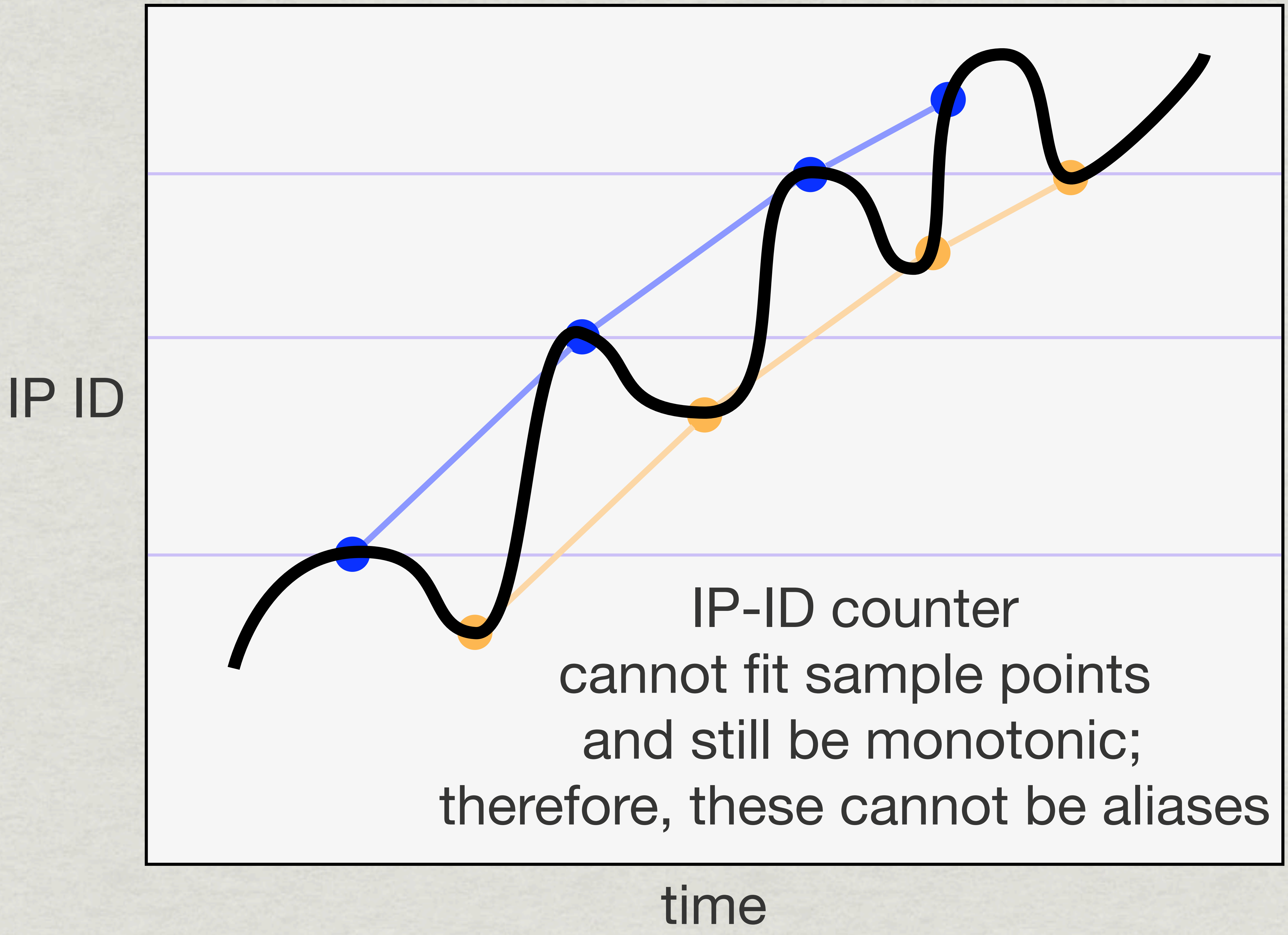


Bender, et al. IMC'08

RadarGun Issues

- RadarGun is groundbreaking work but has both theoretical and practical issues
  - **the distance test for aliases is insufficient**
    - *threshold dependent on underlying dataset*
      - Bender et al used traceroutes between PlanetLab nodes
      - Ark traceroutes are taken to the entire routed space
        - distance distribution noticeably different
    - *threshold doesn't account for velocity*
      - RadarGun *velocity* is the slope of the IP-ID time series
      - setting the threshold high enough to allow high-velocity aliases allows false positives in low-velocity cases
    - *false positives can exist for* ***any*** *chosen threshold*
      - even for a very low threshold

RadarGun false positive for **any** chosen threshold



IP-ID counter
cannot fit sample points
and still be monotonic;
therefore, these cannot be aliases

RadarGun Issues

- **applying RadarGun to 1 million addresses is problematic because RadarGun needs overlapping IP-ID time series for all targets in a short period of time**
    - looks like DDoS attack
    - triggers rate limiting
    - requires high probing rate or large number of machines

RadarGun Issues

$$\frac{\text{interface set size}}{\text{probing rate}} = \text{round duration}$$

or

$$\frac{\text{interface set size}}{\text{round duration}} = \text{probing rate}$$

- probing rate must increase if ...
  - interface set size increases
  - round duration decreases

RadarGun Issues

$$\frac{9{,}056 \text{ interfaces}}{35 \text{ sec}} = 260 \text{ pps}$$

$$\frac{1\text{M interfaces}}{35 \text{ sec}} = 28{,}500 \text{ pps}$$

$$\frac{1\text{M interfaces}}{5 \text{ sec}} = 200{,}000 \text{ pps}$$

- RadarGun’s 35-sec round duration is arbitrary
  - 5 seconds is more appropriate based on the highest actual velocity in our dataset
  - RadarGun needs 769 monitors to probe 1M interfaces with 5-sec duration at 260pps
- 1 week of Ark traces has 1 million interfaces
  - expect possibly 2 million in 1 month of traces
    - possibly 3 million with more monitors

MIDAR

- ***M****onotonic* ***ID****-Based* ***A****lias* ***R****esolution* (MIDAR) is our extension of the RadarGun approach
  - *monotonic bounds test* for accurate testing of pairs
  - *sliding window* for scaling up probing
  - 4 probing methods
  - multiple monitors

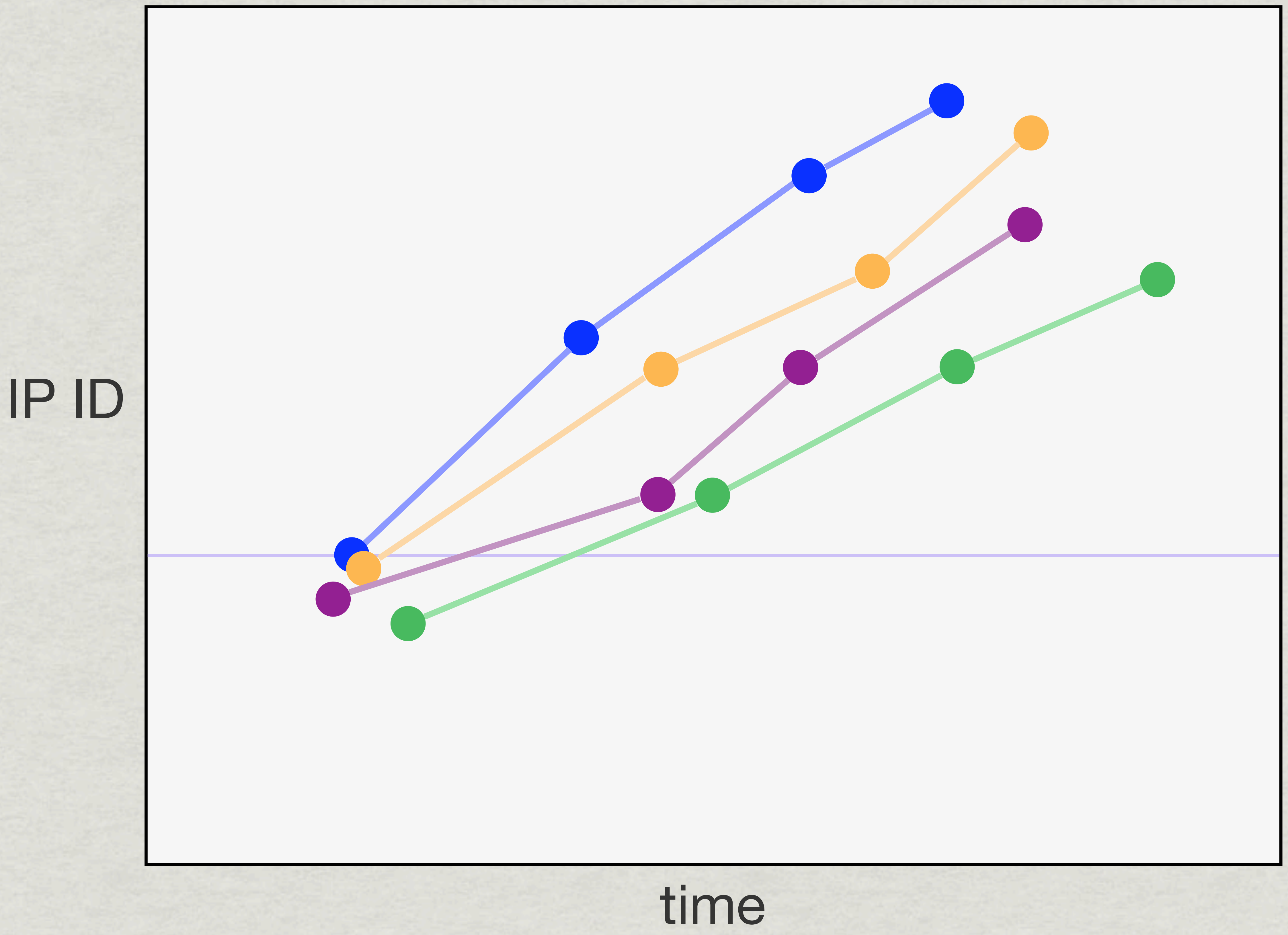
False Positives

- potential for false positives is high when using IP-ID time series for alias resolution
  - IP-ID space has only 2^{16} = 65,536 possibilities
  - birthday paradox (when number of targets < 2^{16}):
    - even with just 9,053 targets, the probability of two targets having the same IP ID value is nearly 1.0
    - only takes 302 targets to have 50% chance for same IP ID
  - pigeonhole principle (when number of targets > 2^{16}):
    - 1 million targets cannot each have unique IP ID values
    - 1 million / 2^{16} = 15 targets per IP ID value on average
  - even worse: nearby IP ID values can cause false positives

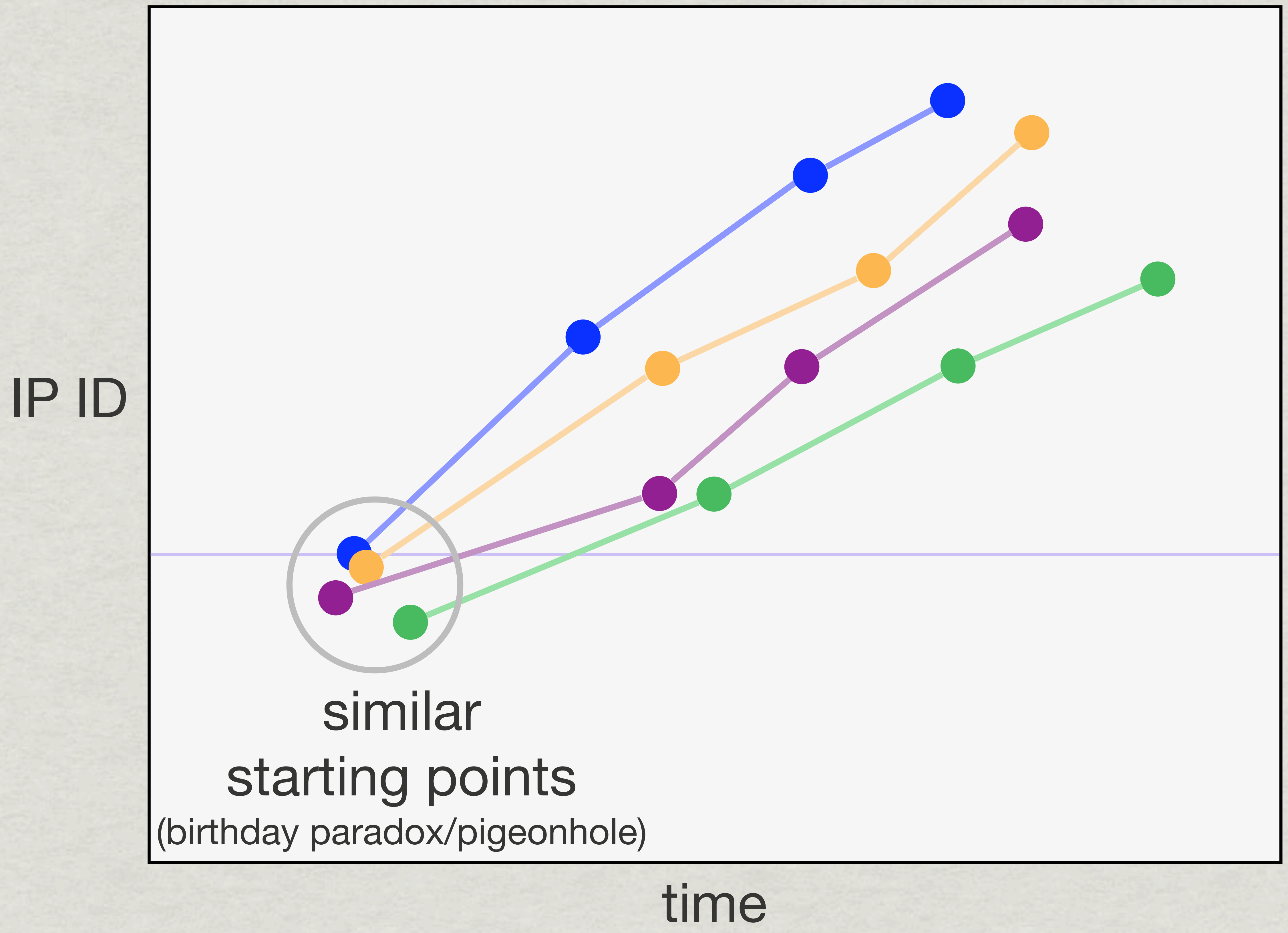
False Positives

- we compare time series, not just individual sample points, for alias resolution
- however, the potential for false positives is still high because the velocity distribution of targets is heavily skewed
  - not much variability (or entropy) in practice
  - ~80% of targets have velocity of 10 IDs/sec or less
  - ~50% have velocity of 1 IDs/sec or less

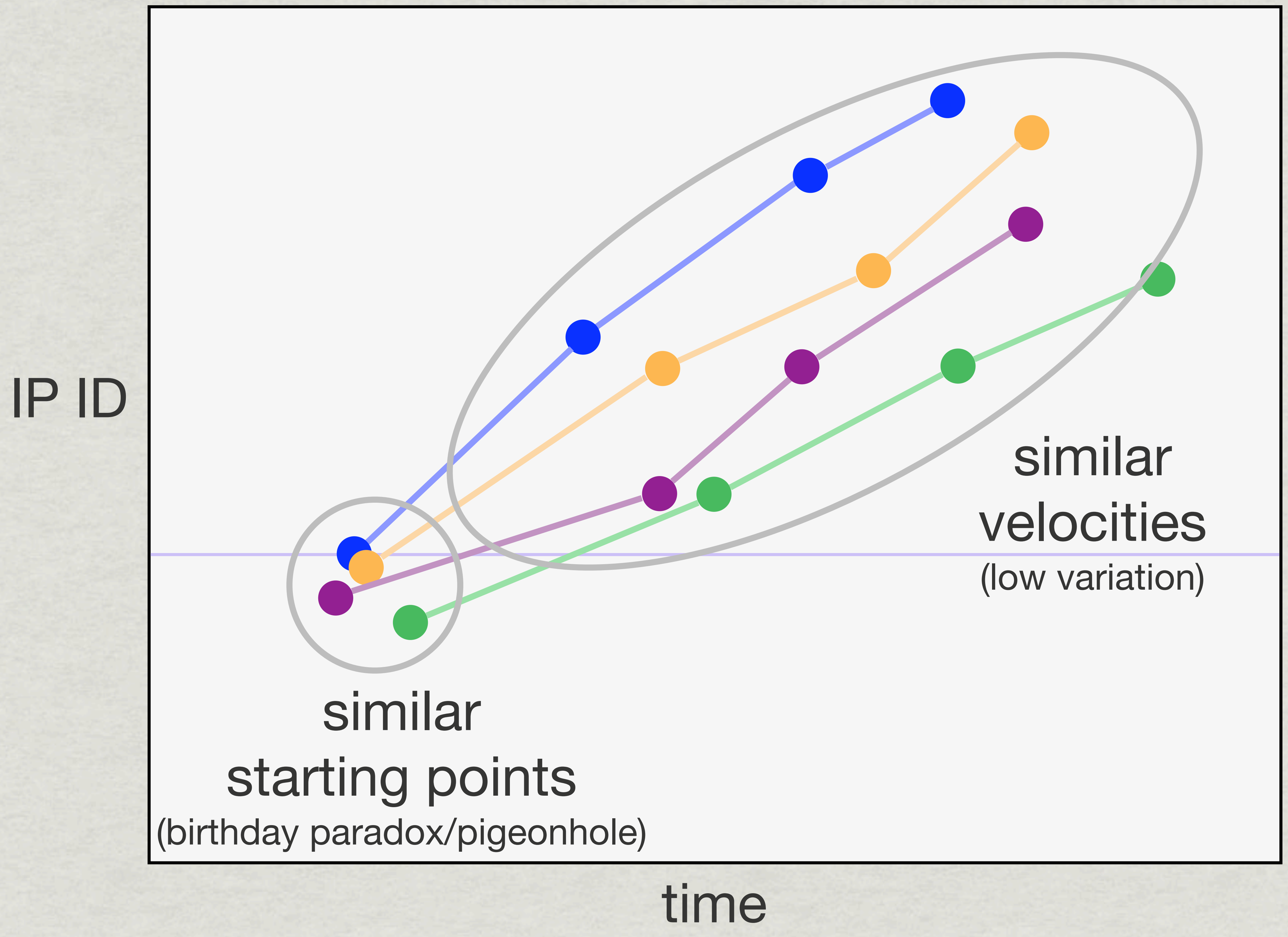
False Positives



False Positives



False Positives



False Positives

- an accurate shared counter test is critical
  - the number of true alias pairs is low
    - with N addresses:
      - number of true aliases is $O(N)$
      - number of false positives is a fraction of total pairs, or $O(N^2)$

addrs	all pairs	true alias pairs[1]	
10k	50M	245k	0.490%
100k	5G	2.5M	0.049%
1M	500G	25M	0.005%

[1] alias pairs extrapolated from tier 1 ISP

  - false positives are amplified when combining alias pairs into routers with transitive closure

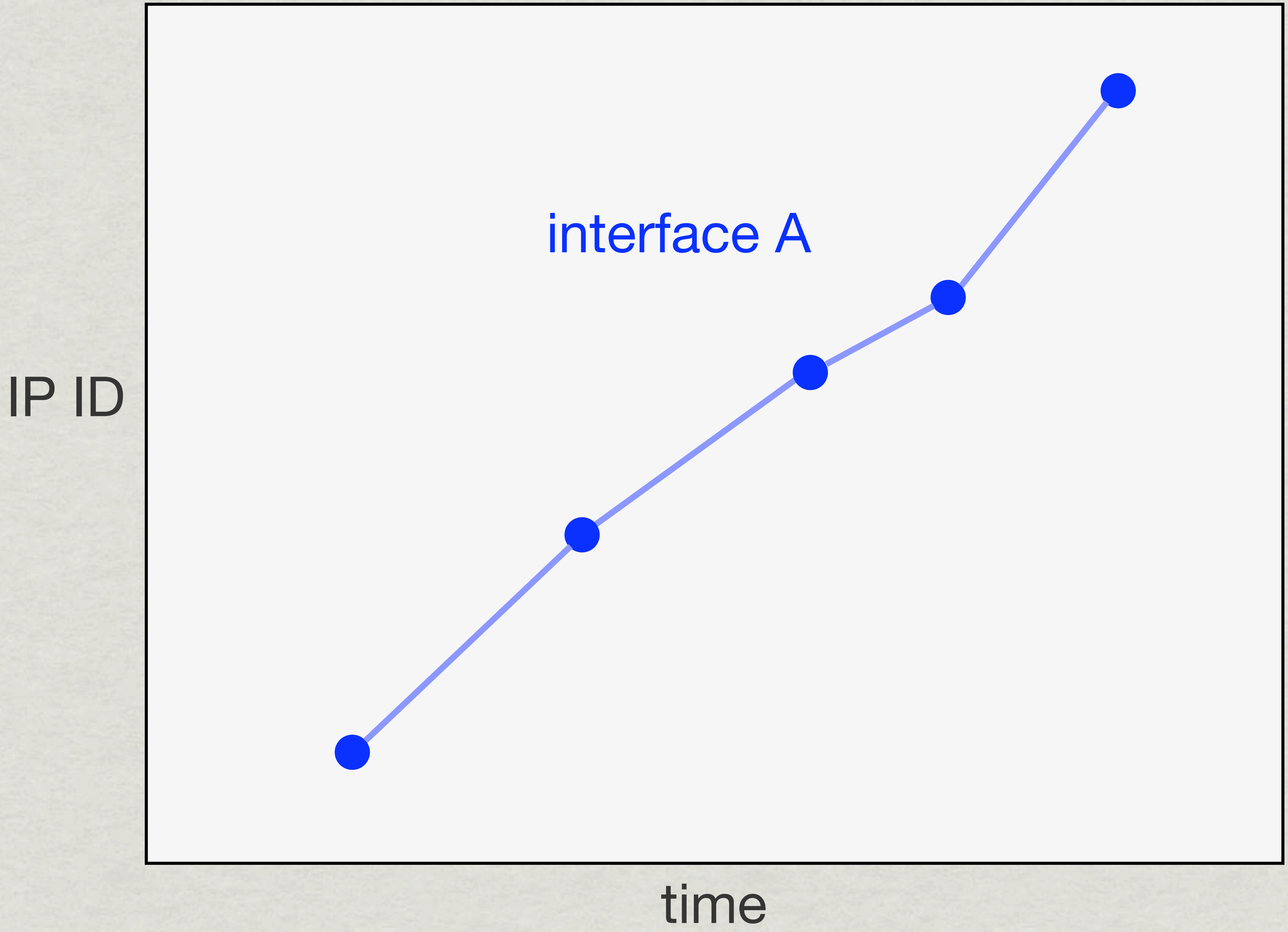
MIDAR

- MIDAR uses the *monotonic bounds test*:
  - based on a **necessary** condition, not an arbitrary threshold
  - failing the test means definitively “not a shared counter”
    - that is, provides *negative* information
    - “not shared” is not as strong as “not alias” but still useful
  - extremely low false positives when repeated

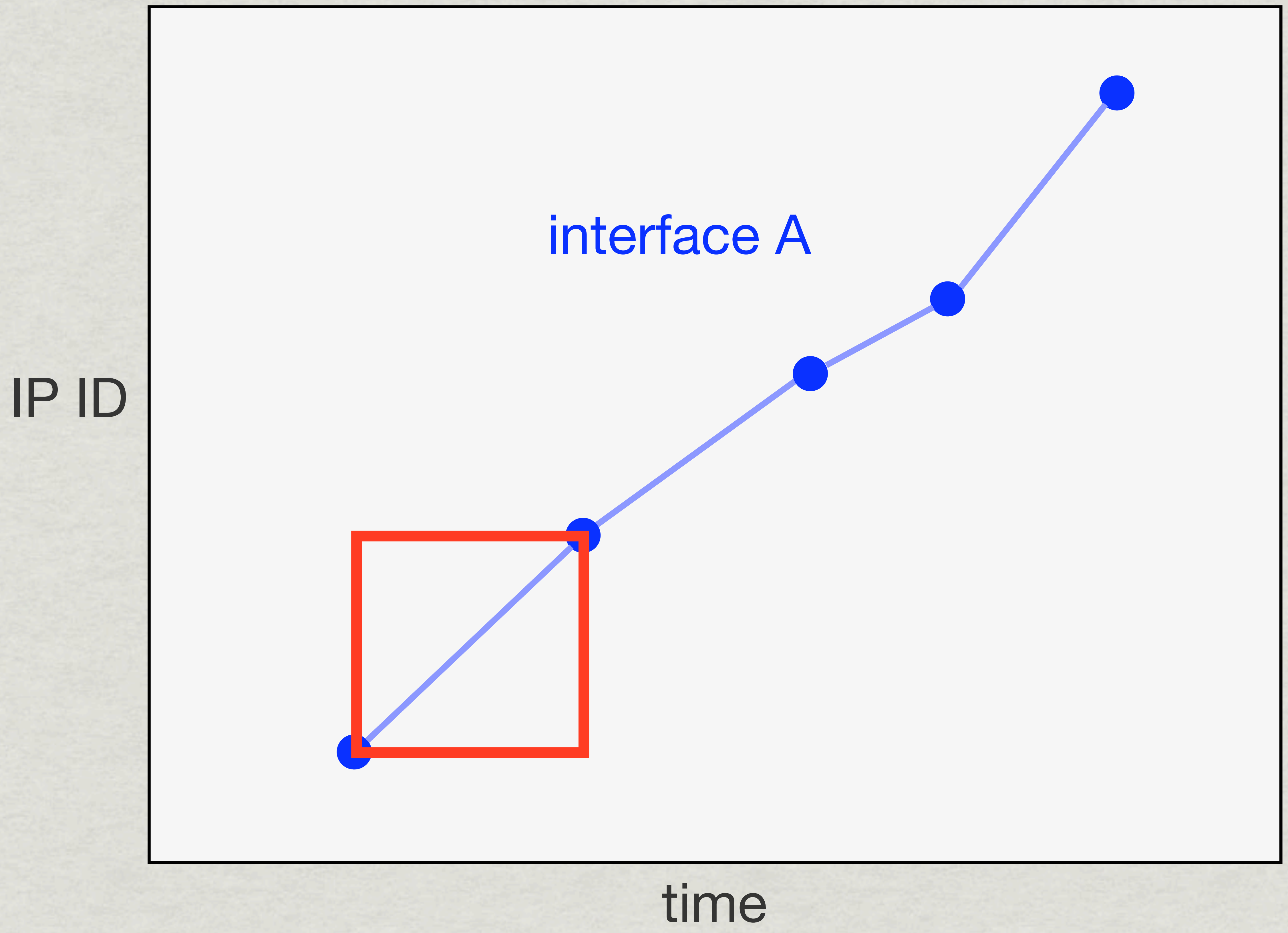
Monotonic Bounds Test

- the *monotonic bounds test* rationale:
  - if two interfaces use a shared counter for their IP-ID values, then they are aliases
    - the same observation underlying RadarGun
    - careful: converse not true; aliases need not share a counter
      - so time-series analysis can only detect aliases that share a counter (which applies to RadarGun & MIDAR equally)
  - if two interfaces share a counter, then their IP-ID time series must form a strictly increasing sequence (“must be monotonic”) when merged together
    - therefore, having a monotonic combined time series is a **necessary** condition for being a shared counter and thus a necessary condition for being a **detectable** alias

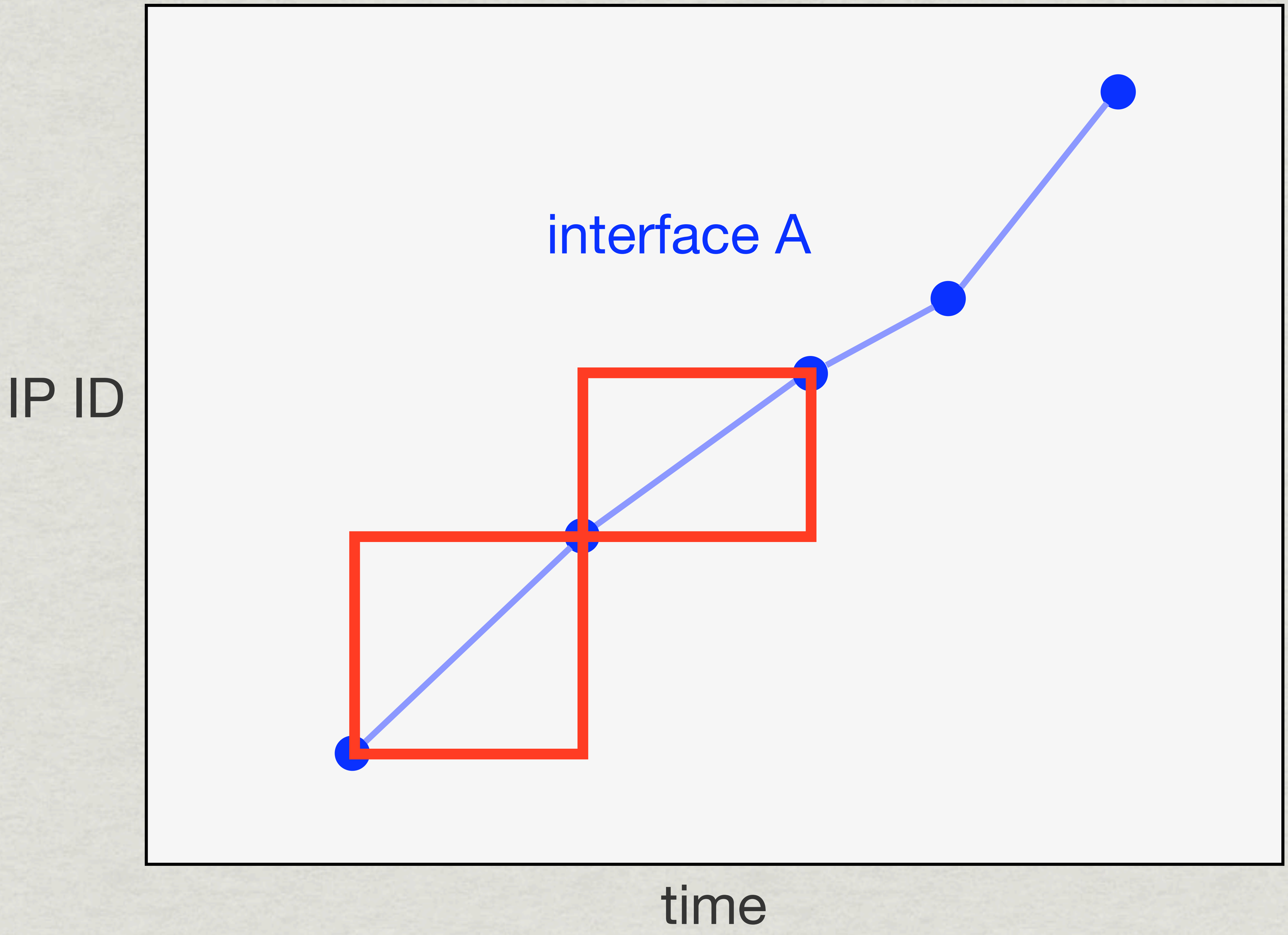
Monotonic Bounds Test



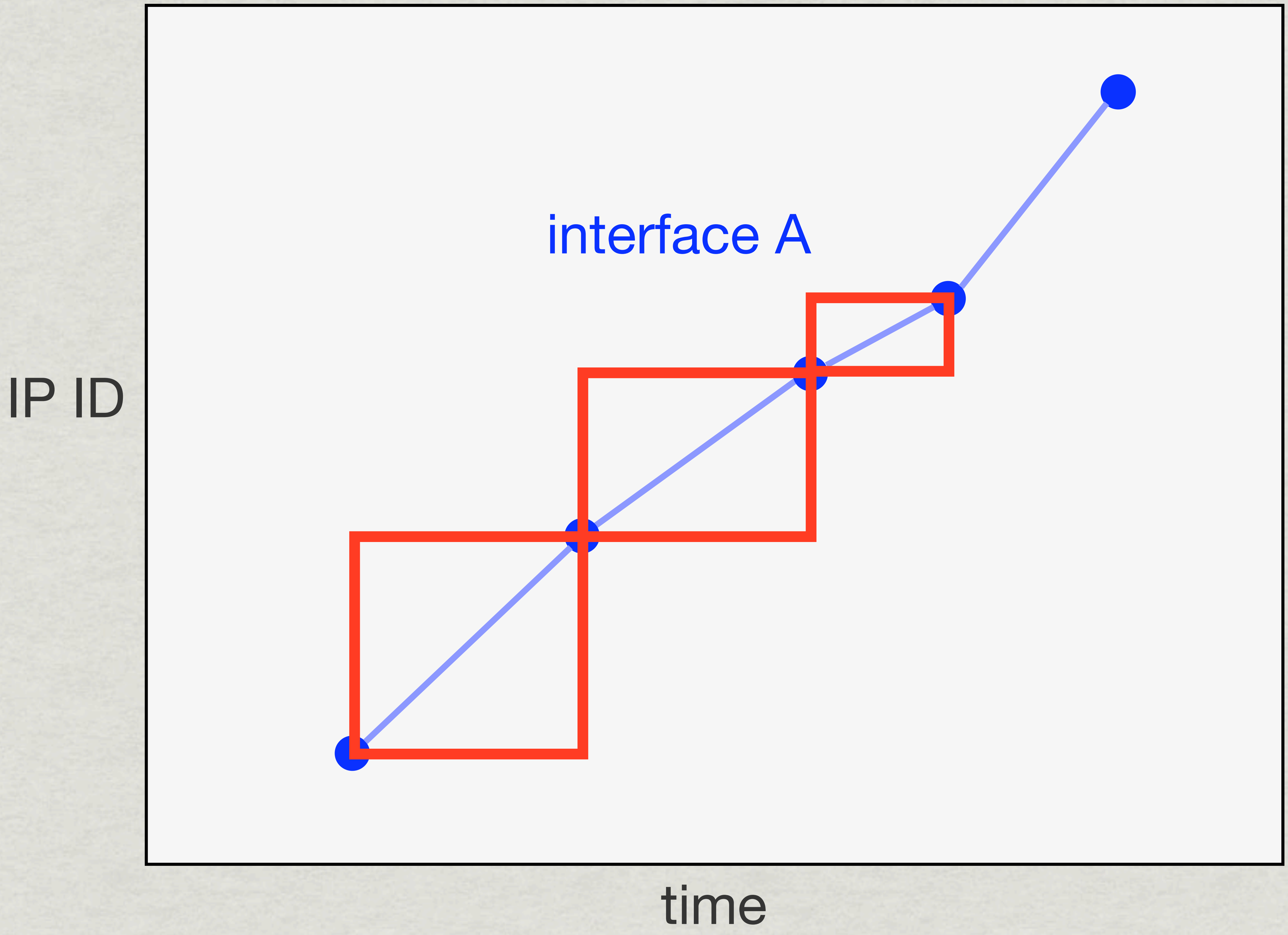
Monotonic Bounds Test



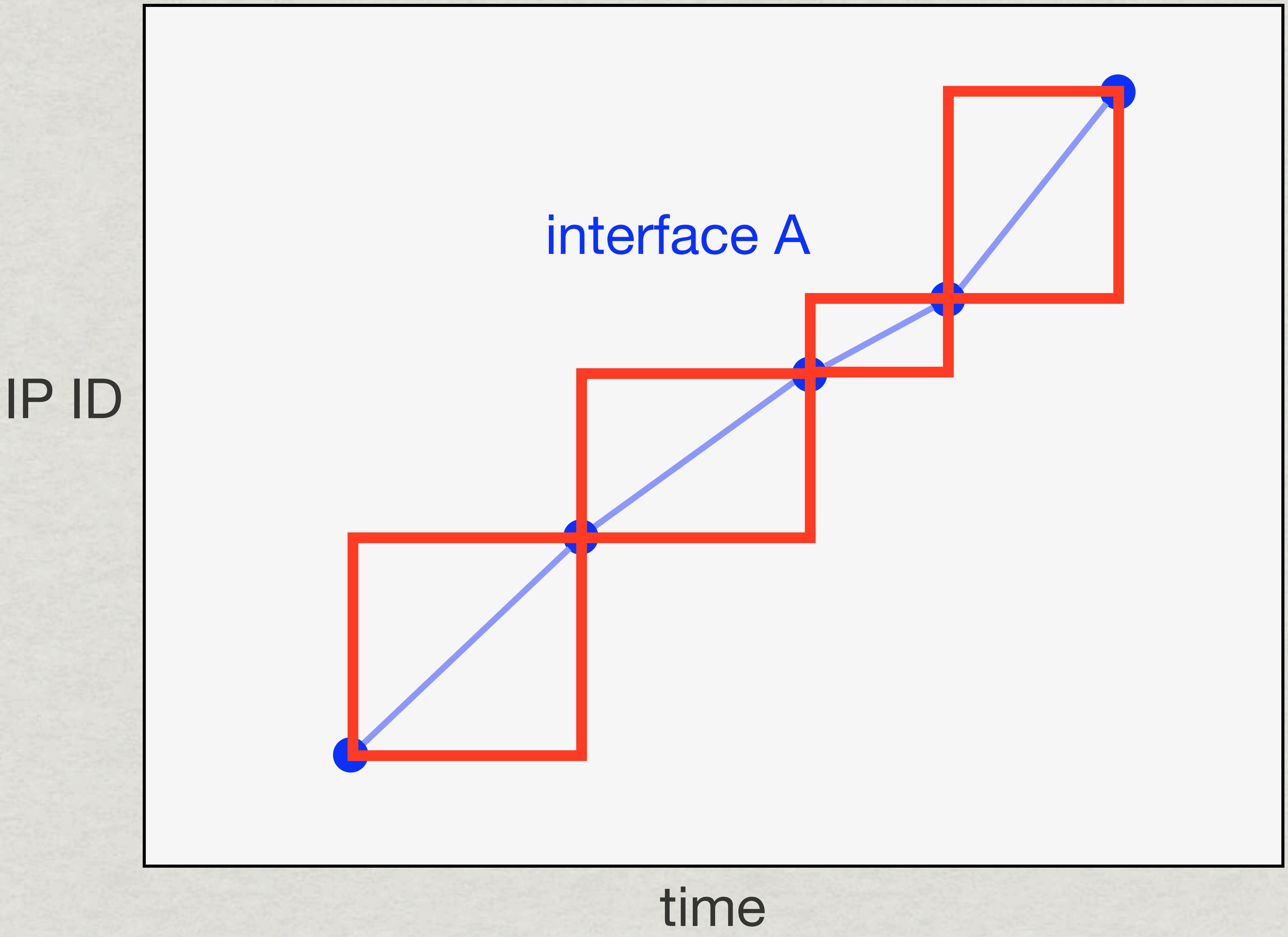
Monotonic Bounds Test



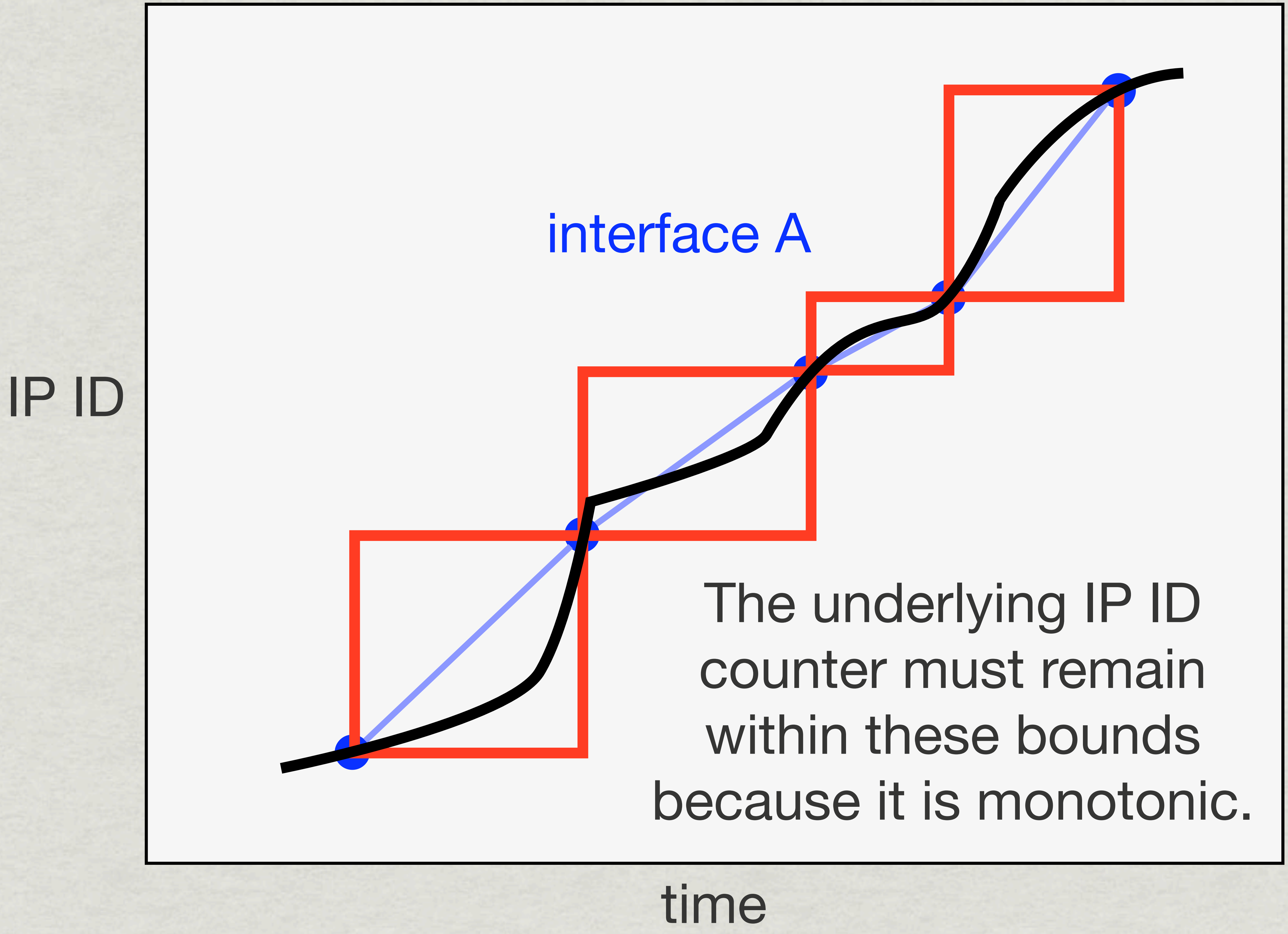
Monotonic Bounds Test



Monotonic Bounds Test

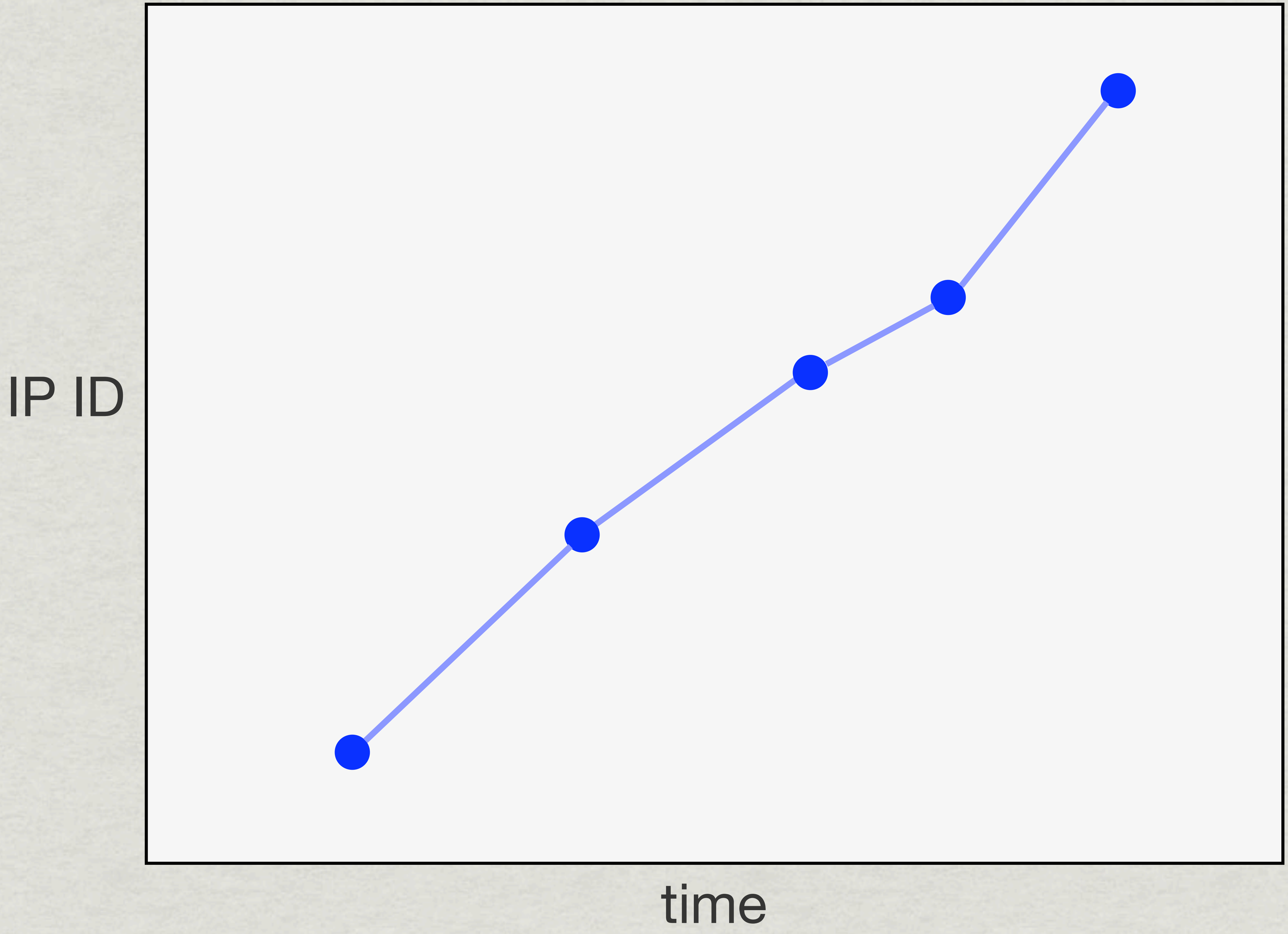


Monotonic Bounds Test

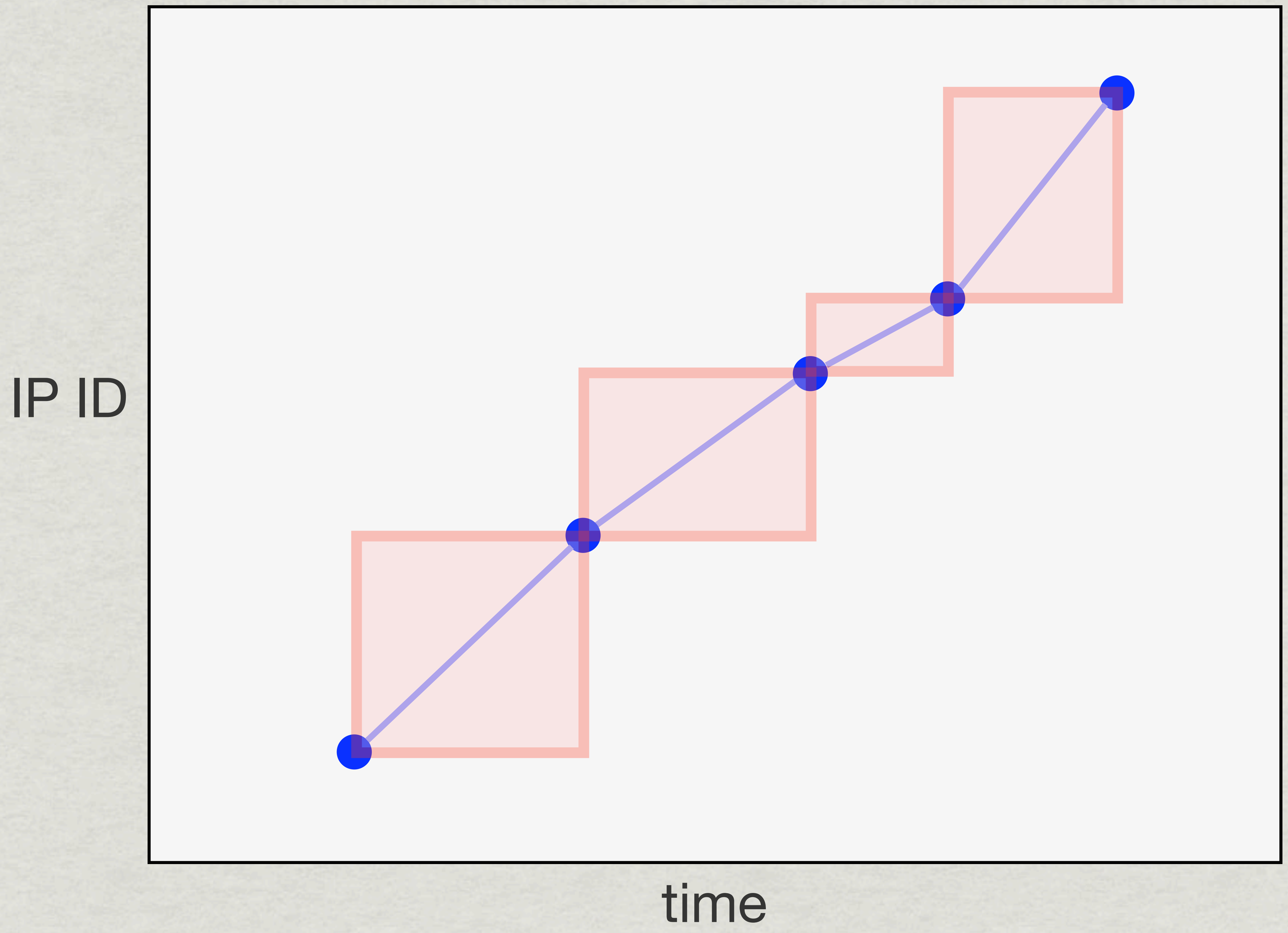


The underlying IP ID counter must remain within these bounds because it is monotonic.

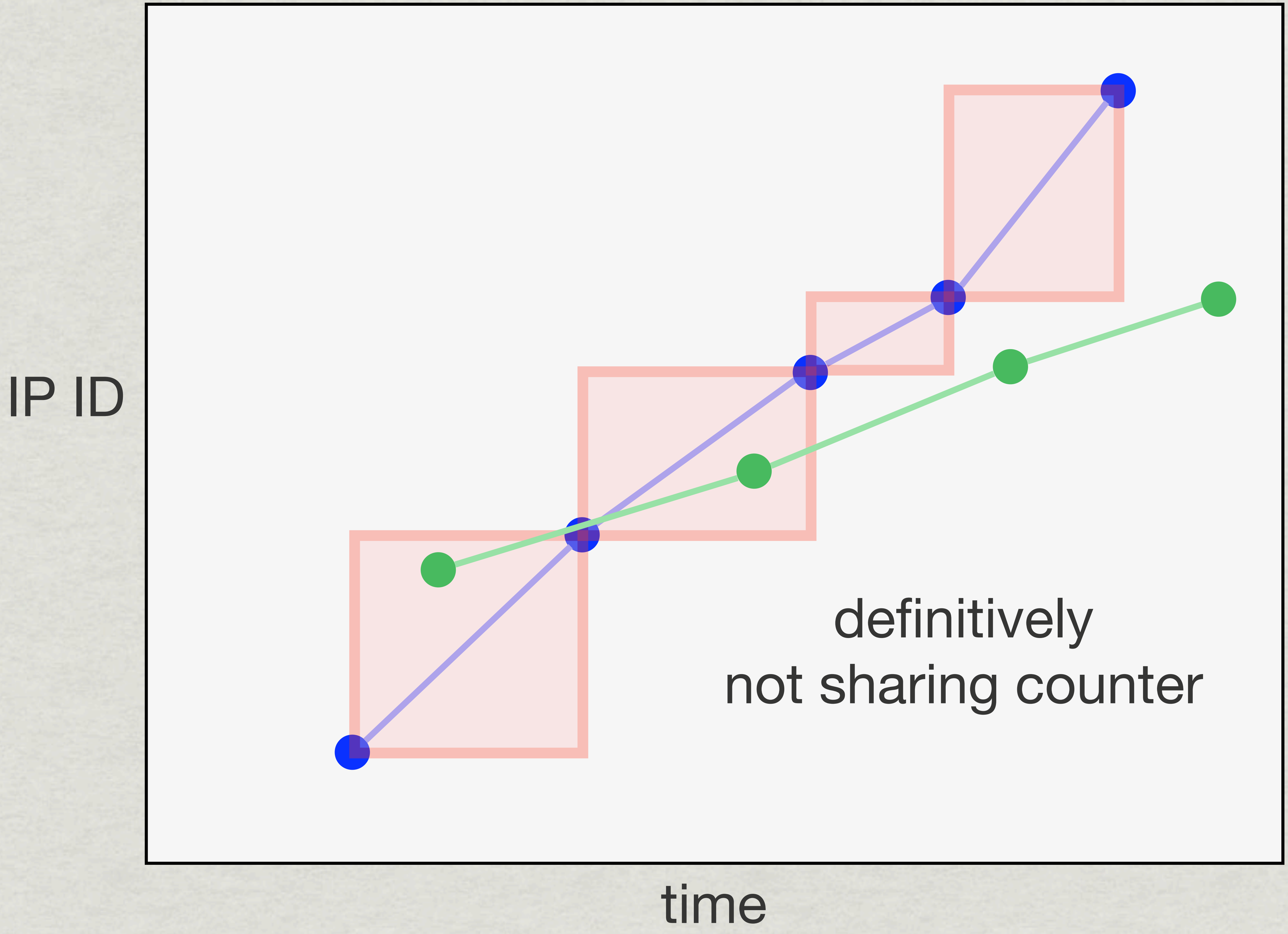
Monotonic Bounds Test



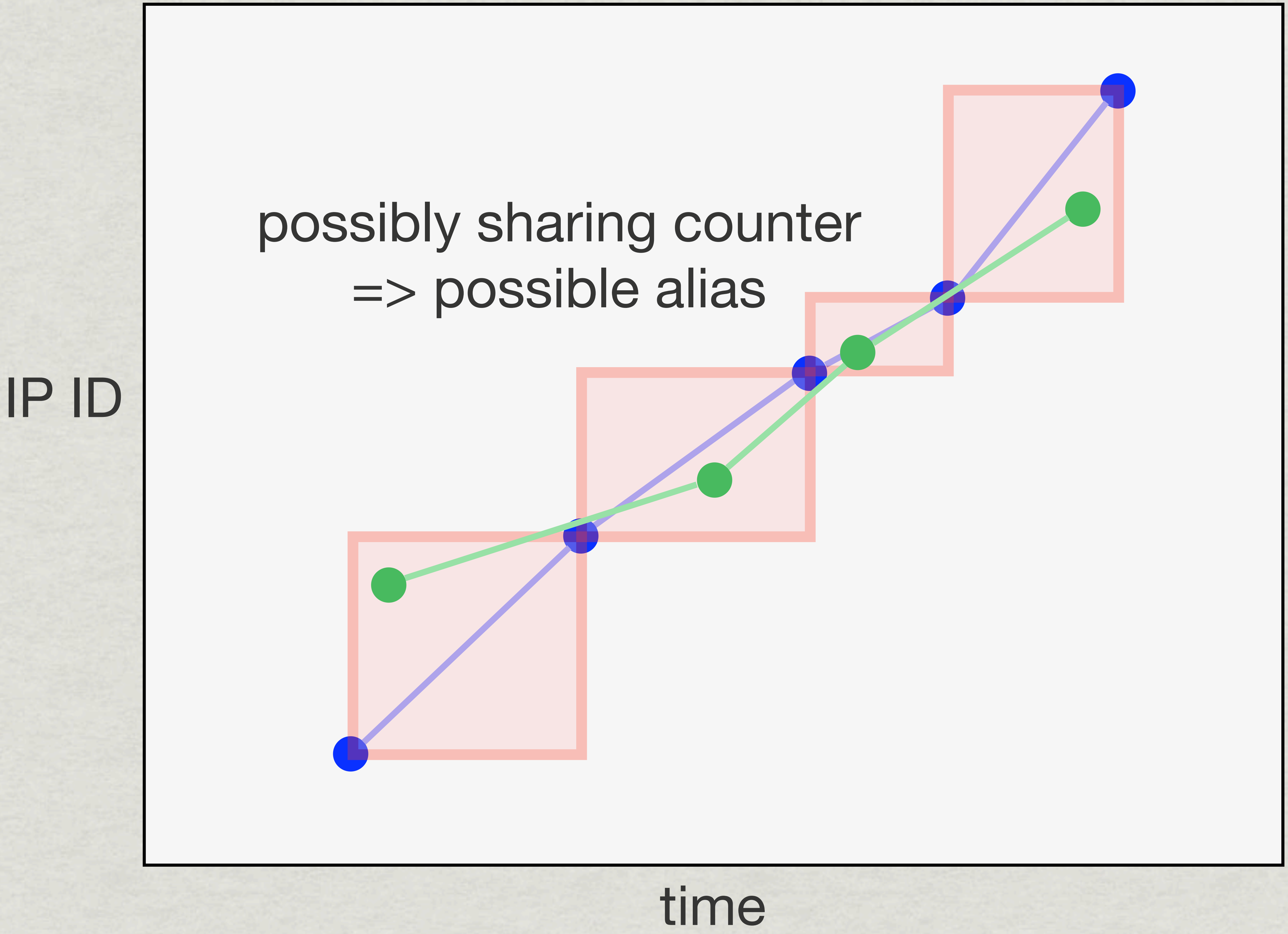
Monotonic Bounds Test



Monotonic Bounds Test



Monotonic Bounds Test



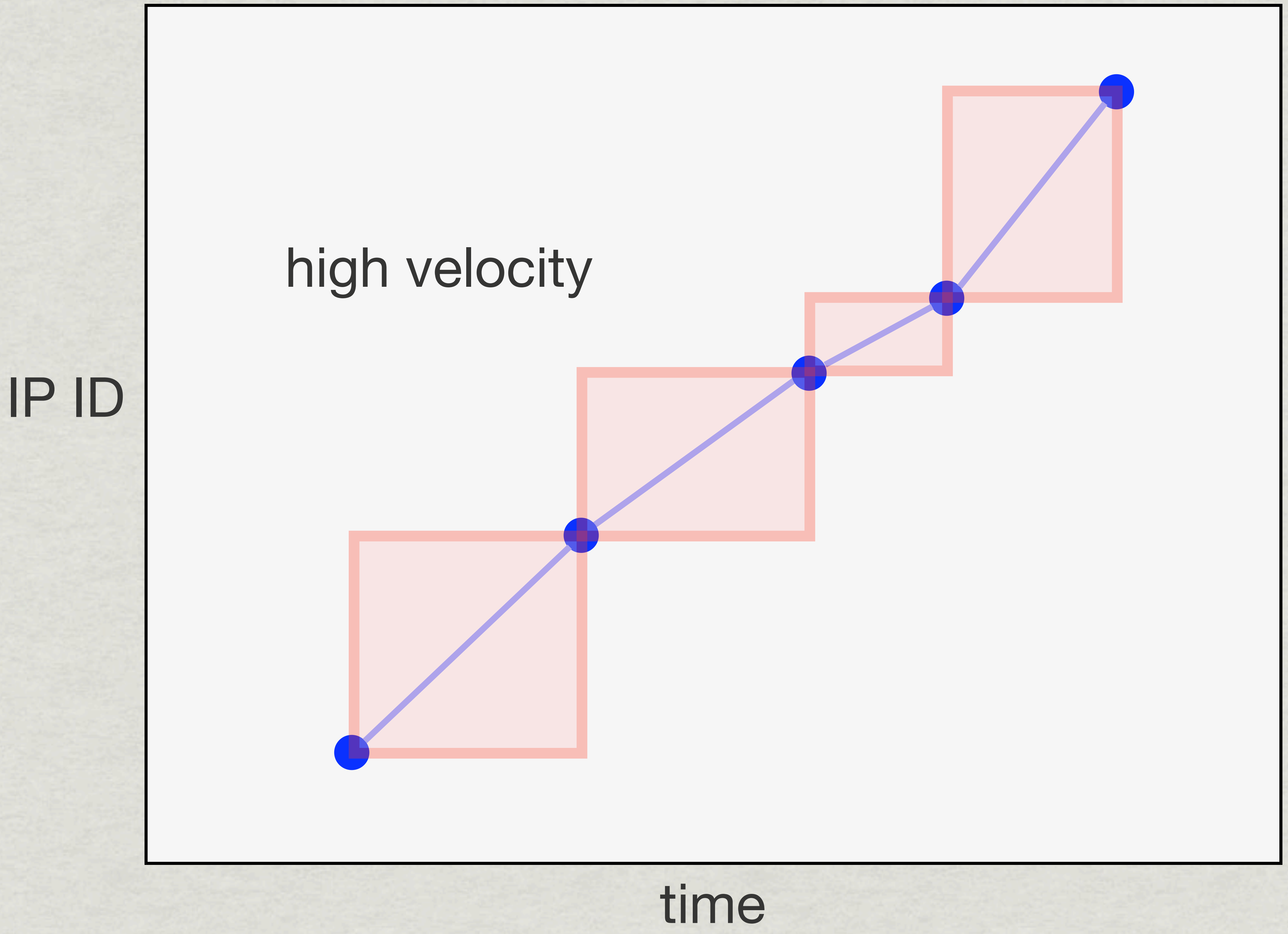
Monotonic Bounds Test

- passing the monotonic bounds test is not a **sufficient** condition for sharing a counter
  - false positives from chance alignment, just as with the distance test
  - but **crucial** point:
    - the monotonic bounds test guarantees the **necessary** condition for sharing a counter
    - we can exploit this guarantee to ensure sufficiency

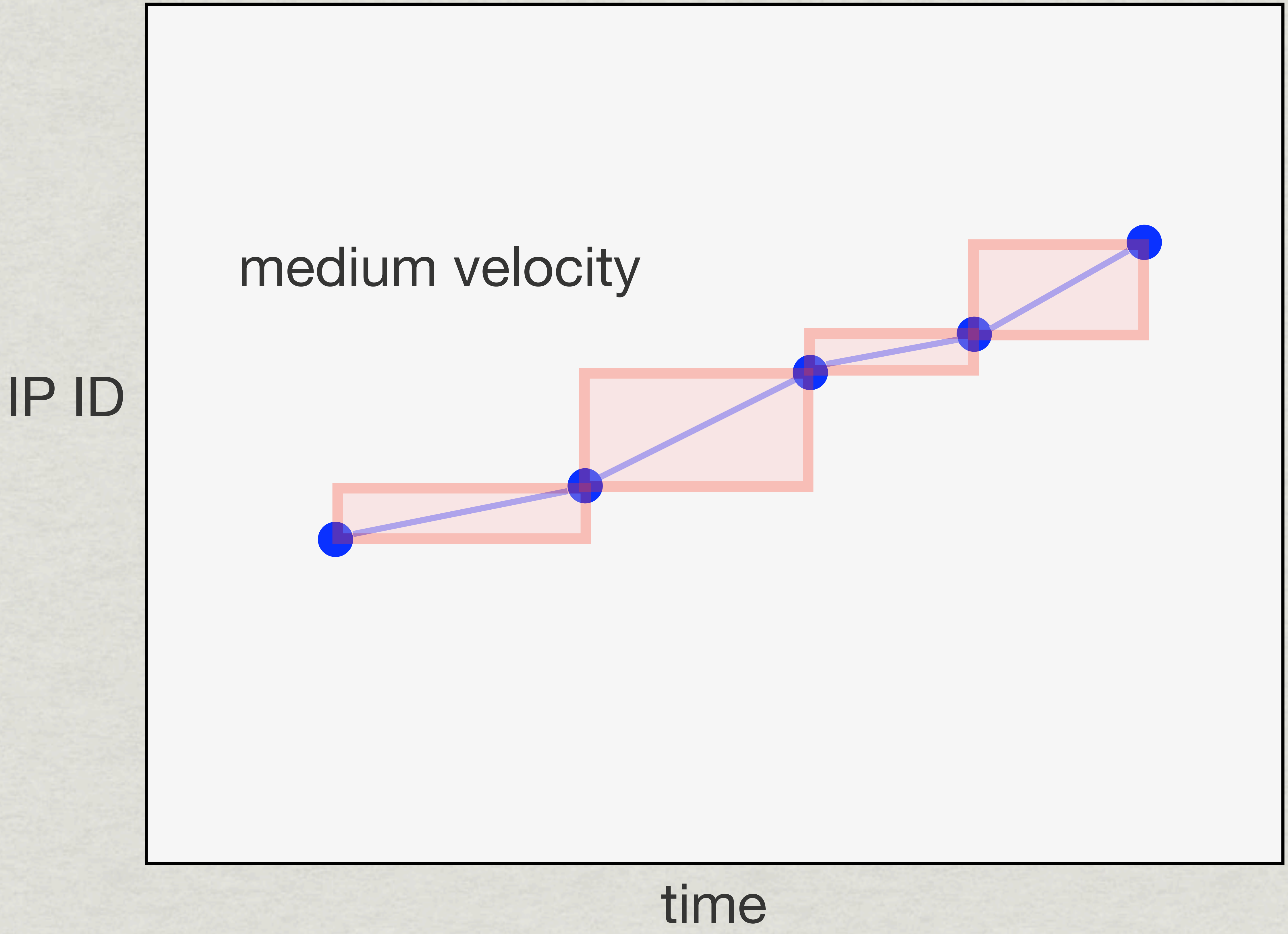
Monotonic Bounds Test

- we can improve confidence by repeating the test at a later time after non-shared counters have had a chance to diverge
  - each application of the monotonic bounds test only removes false positives
    - never rejects real aliases (that is, does not create false negatives)
    - so repetition is helpful and never harmful
  - the test converges quickly and with high confidence to the set of true positives with repetition
    - because the test takes advantage of varying velocities and probe spacing

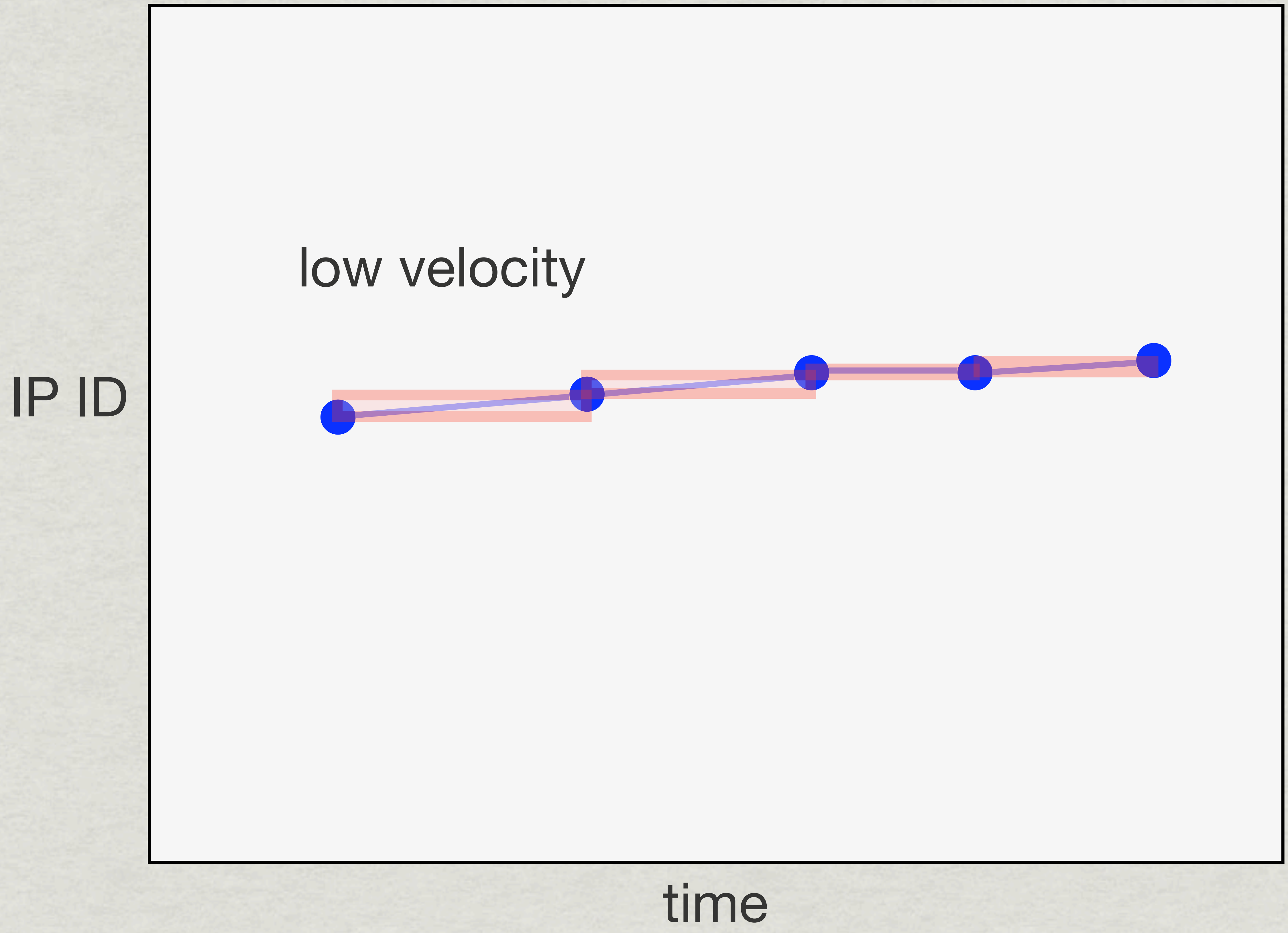
Monotonic Bounds Test



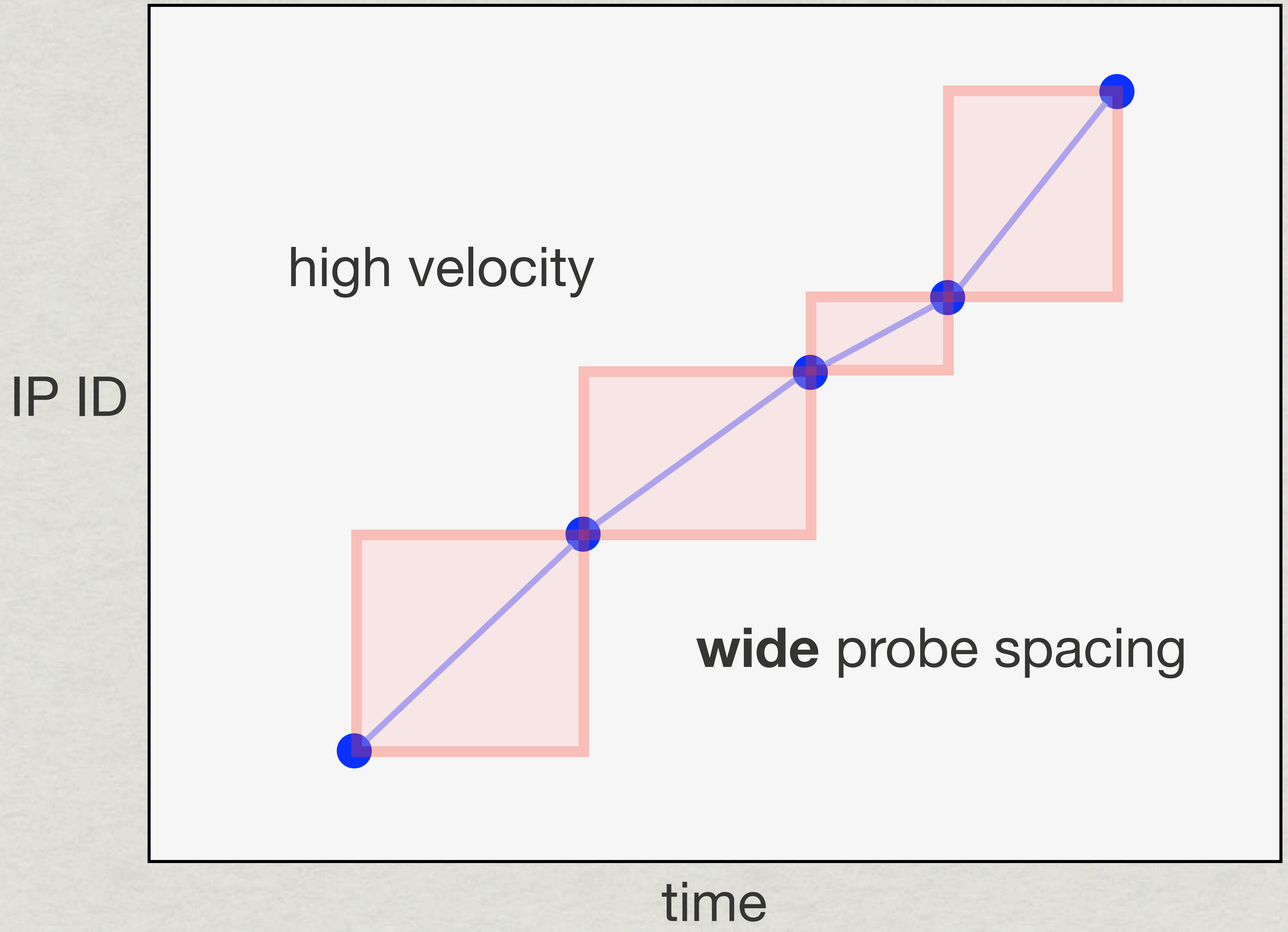
Monotonic Bounds Test



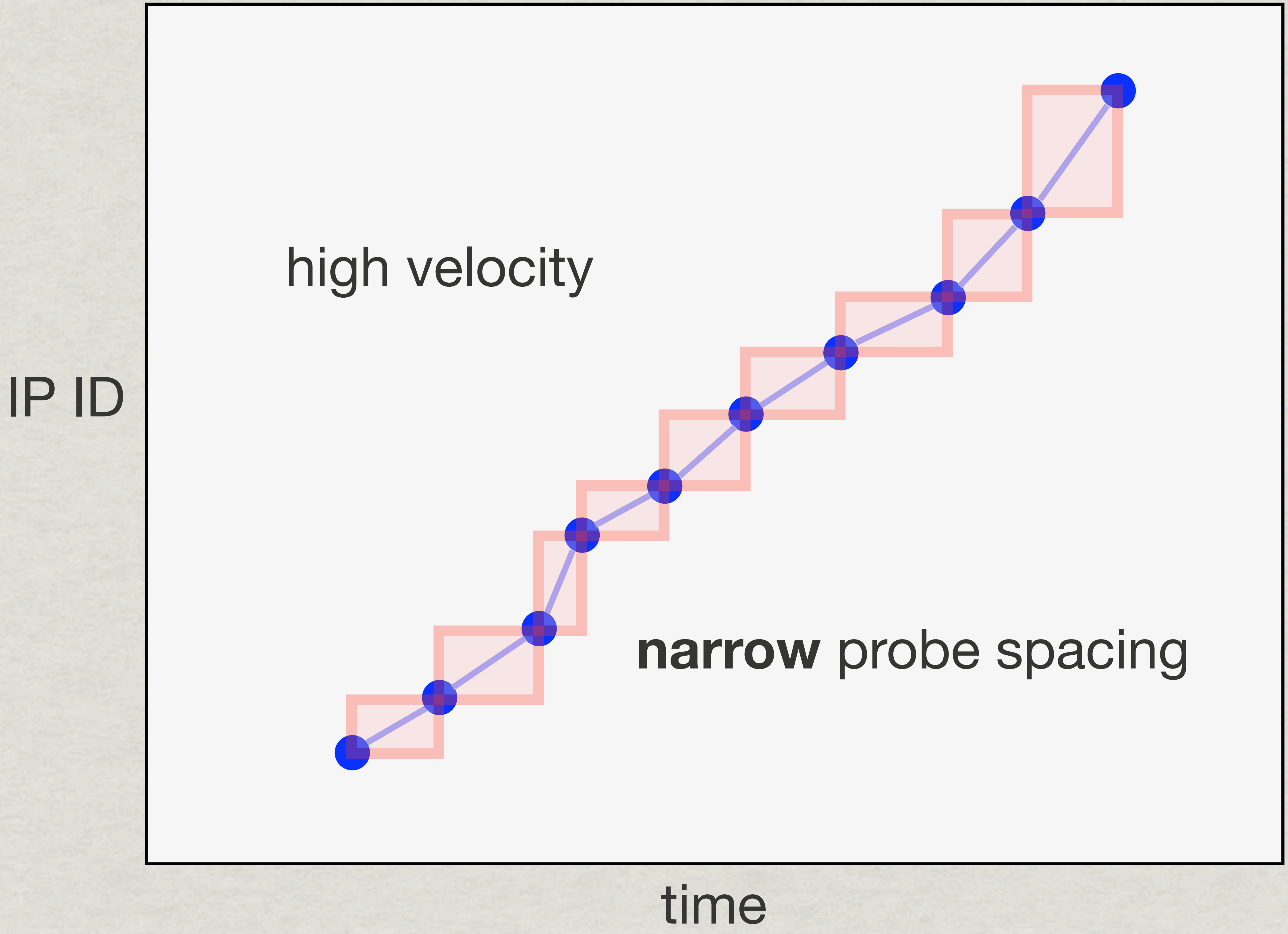
Monotonic Bounds Test



Monotonic Bounds Test

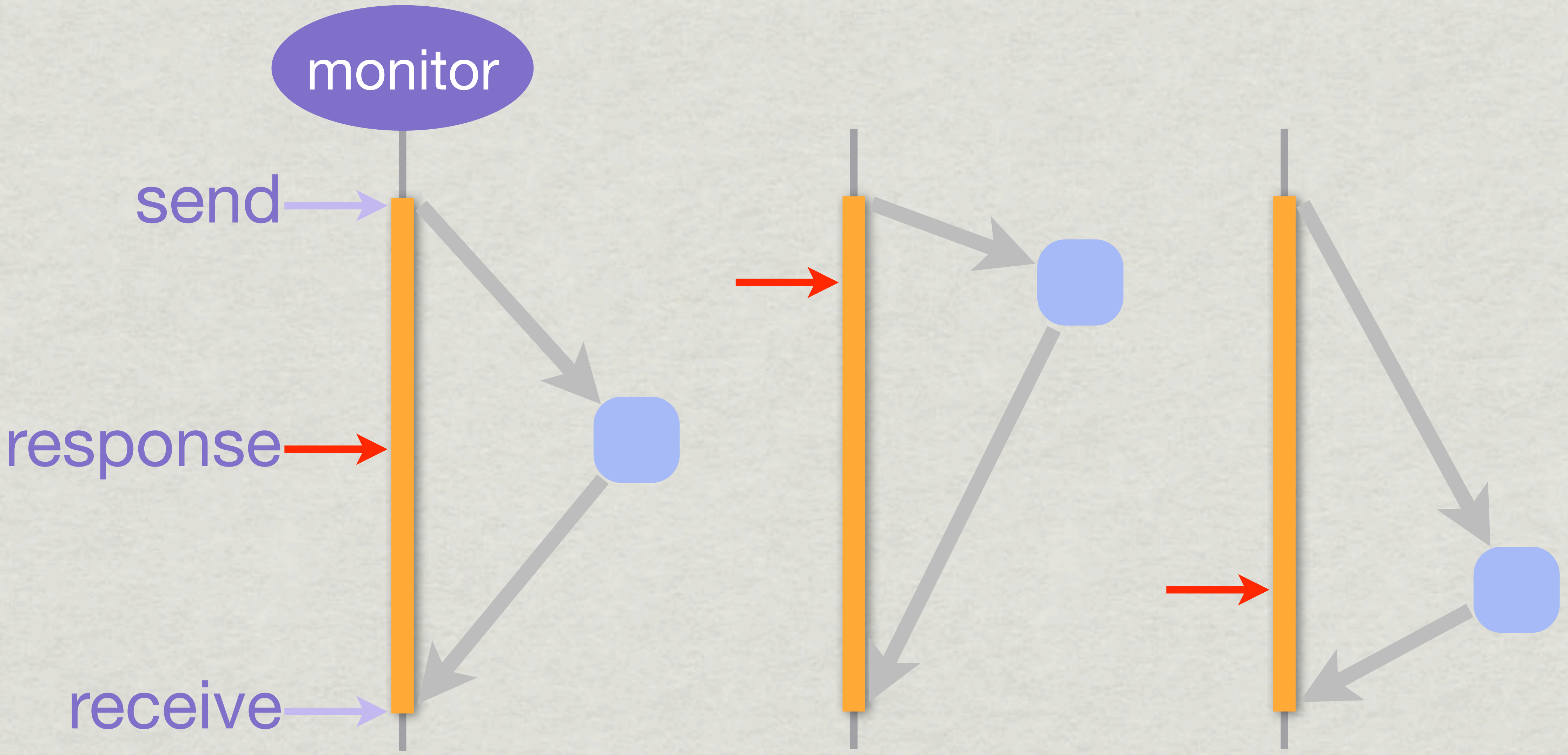


Monotonic Bounds Test



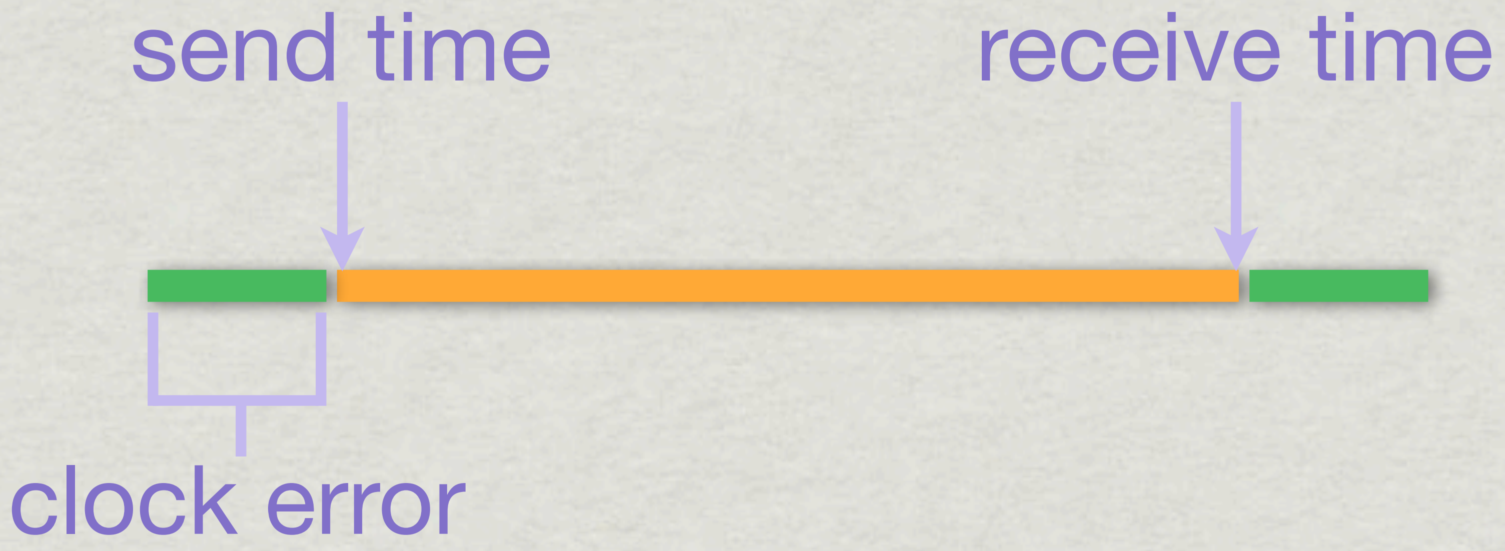
Monotonic Bounds Test

- the monotonic bounds test is slightly more complicated in practice
  - first, exact time of response unknown:



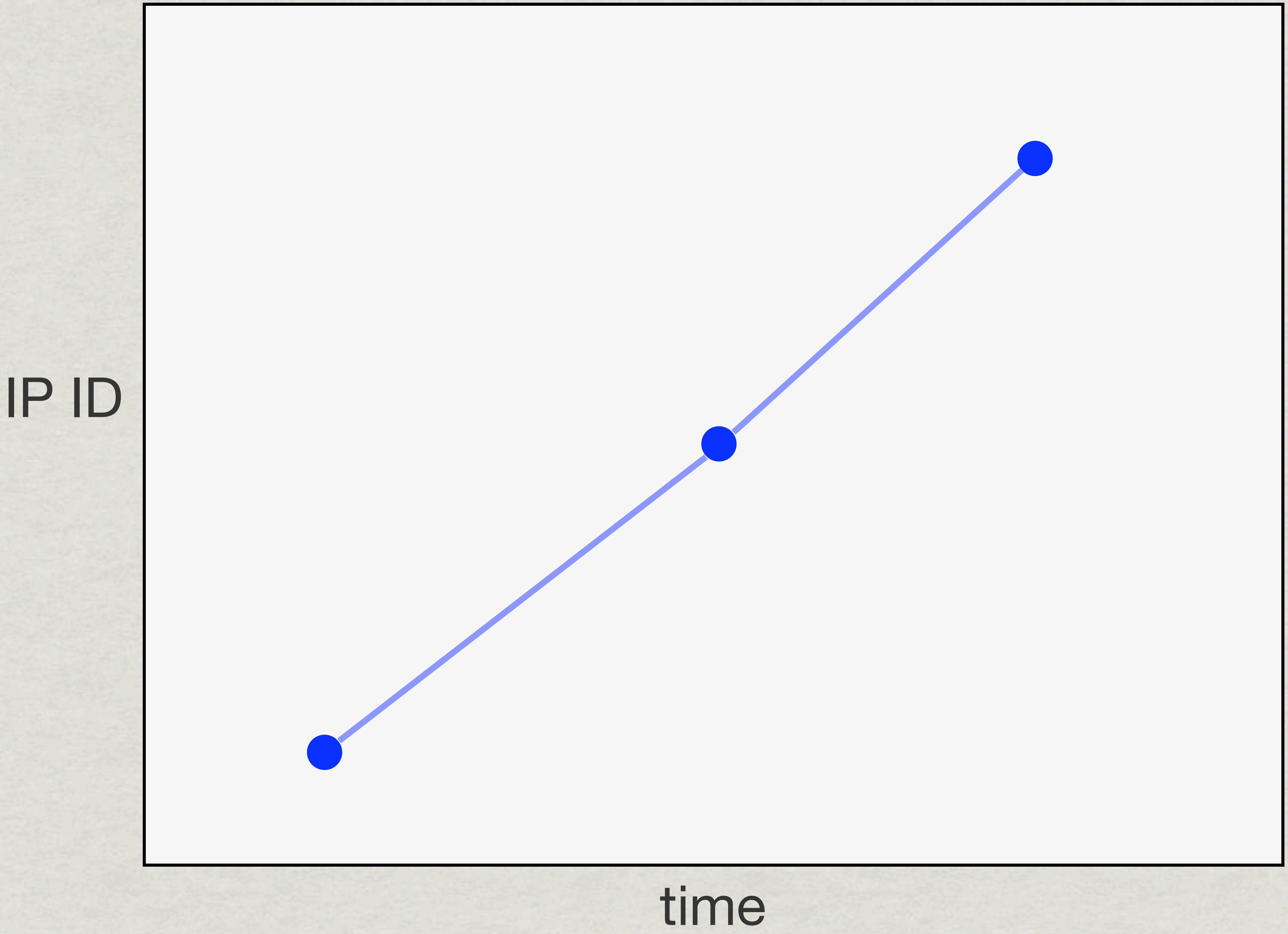
Monotonic Bounds Test

- second, clocks are not perfectly synchronized across monitors
  - only matters when comparing data from multiple monitors

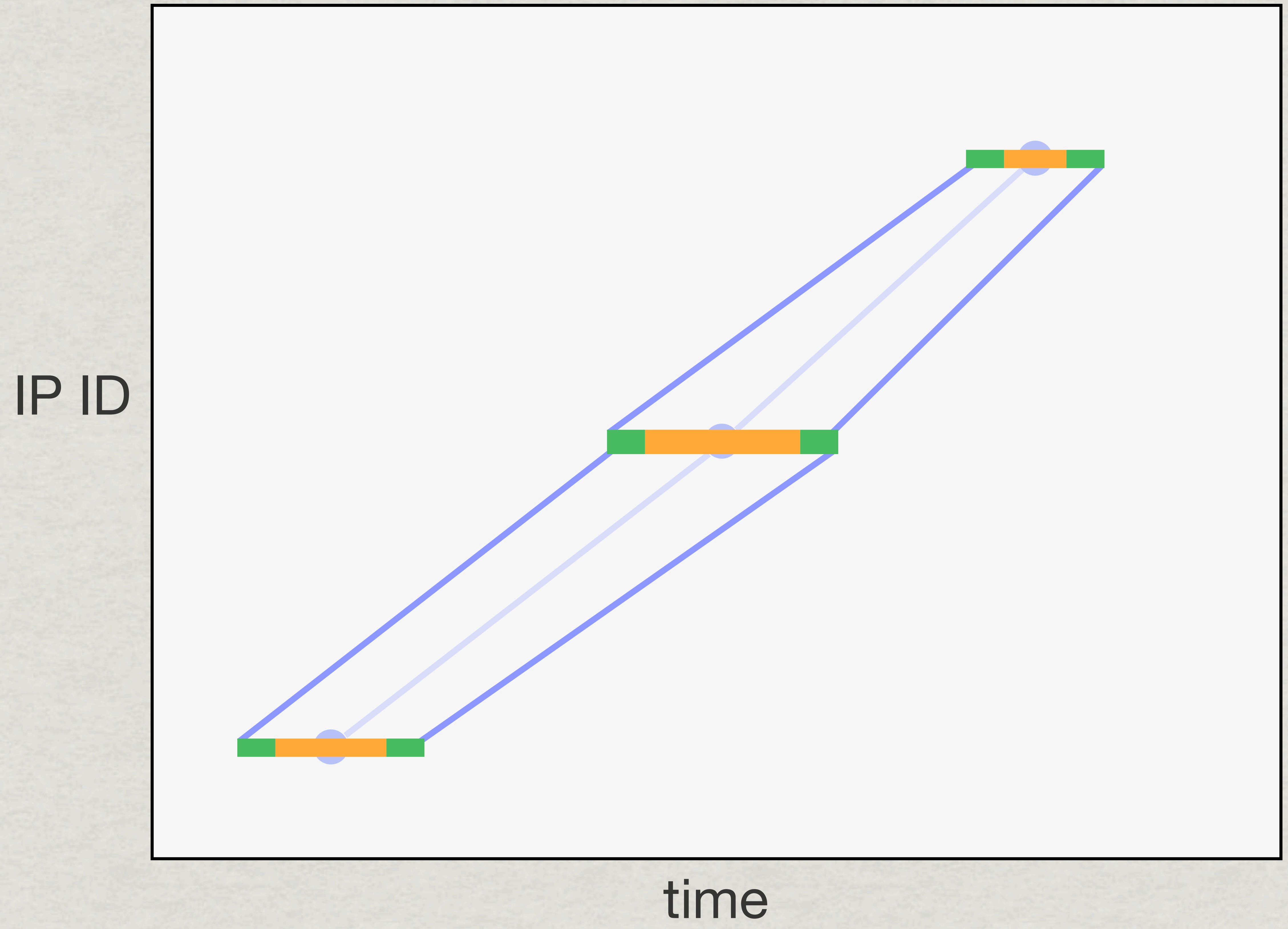


- we can accommodate uncertainties in both the response time and clock offset without compromising the rigor of the monotonic bounds test

Monotonic Bounds Test

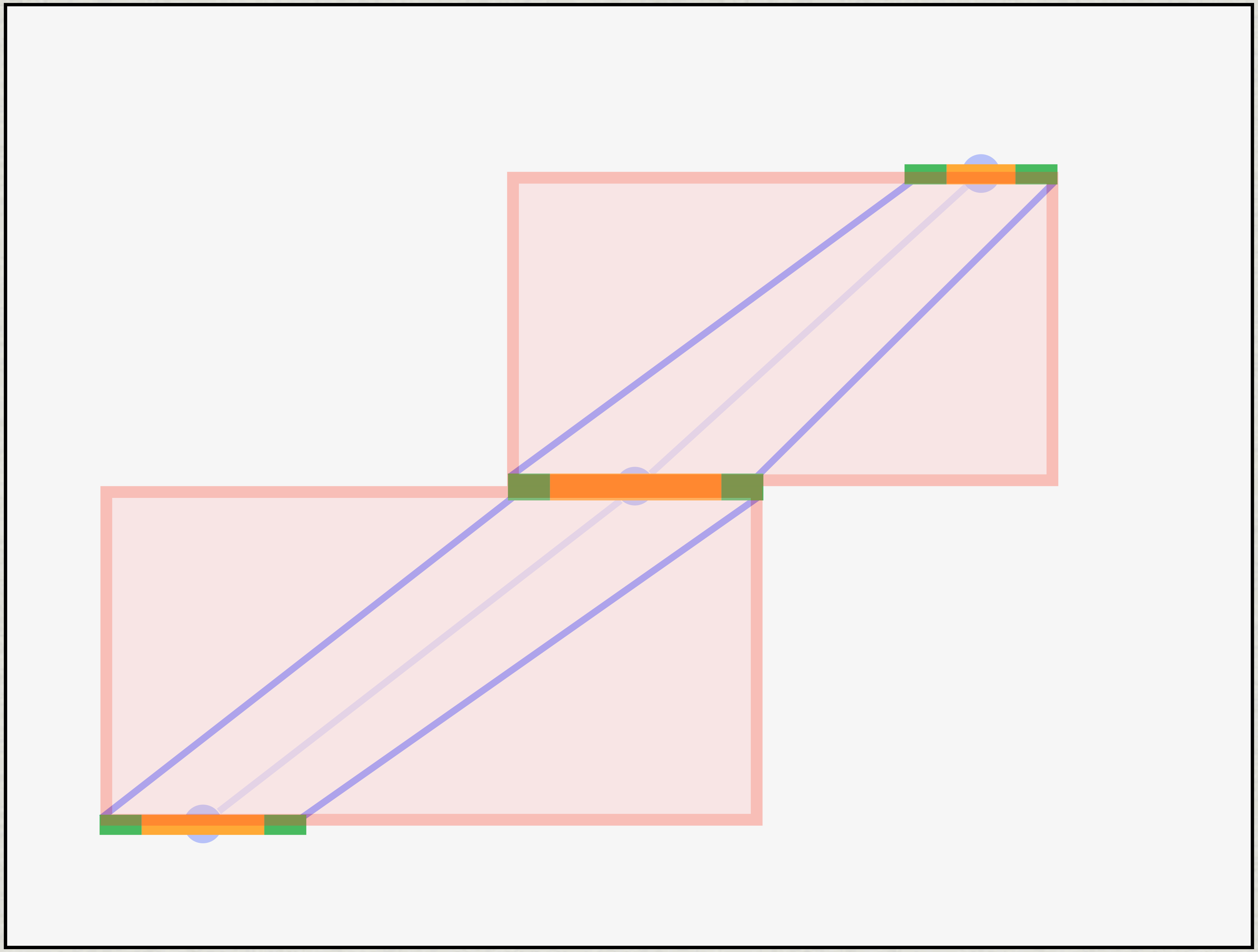


Monotonic Bounds Test



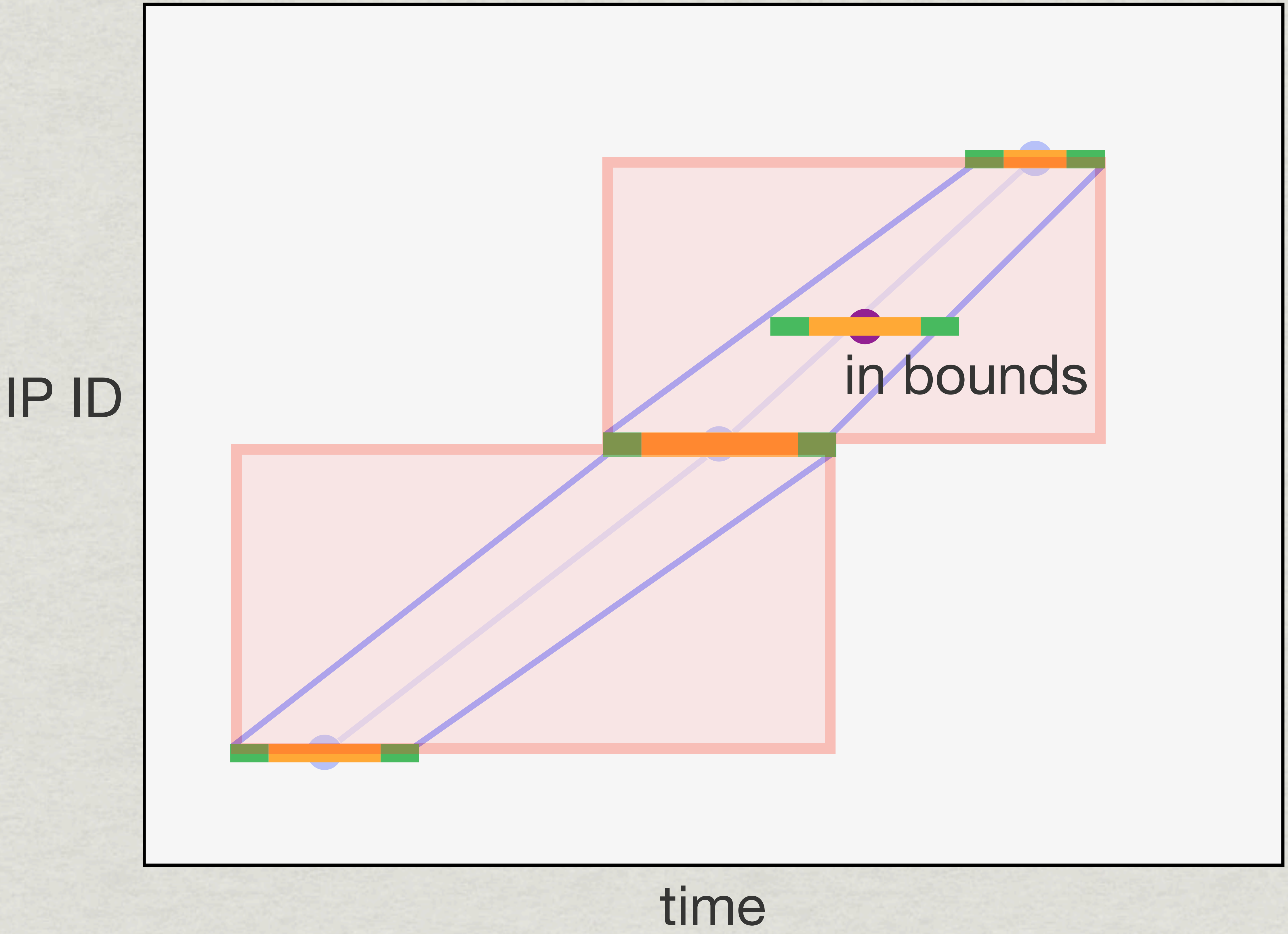
Monotonic Bounds Test

IP ID

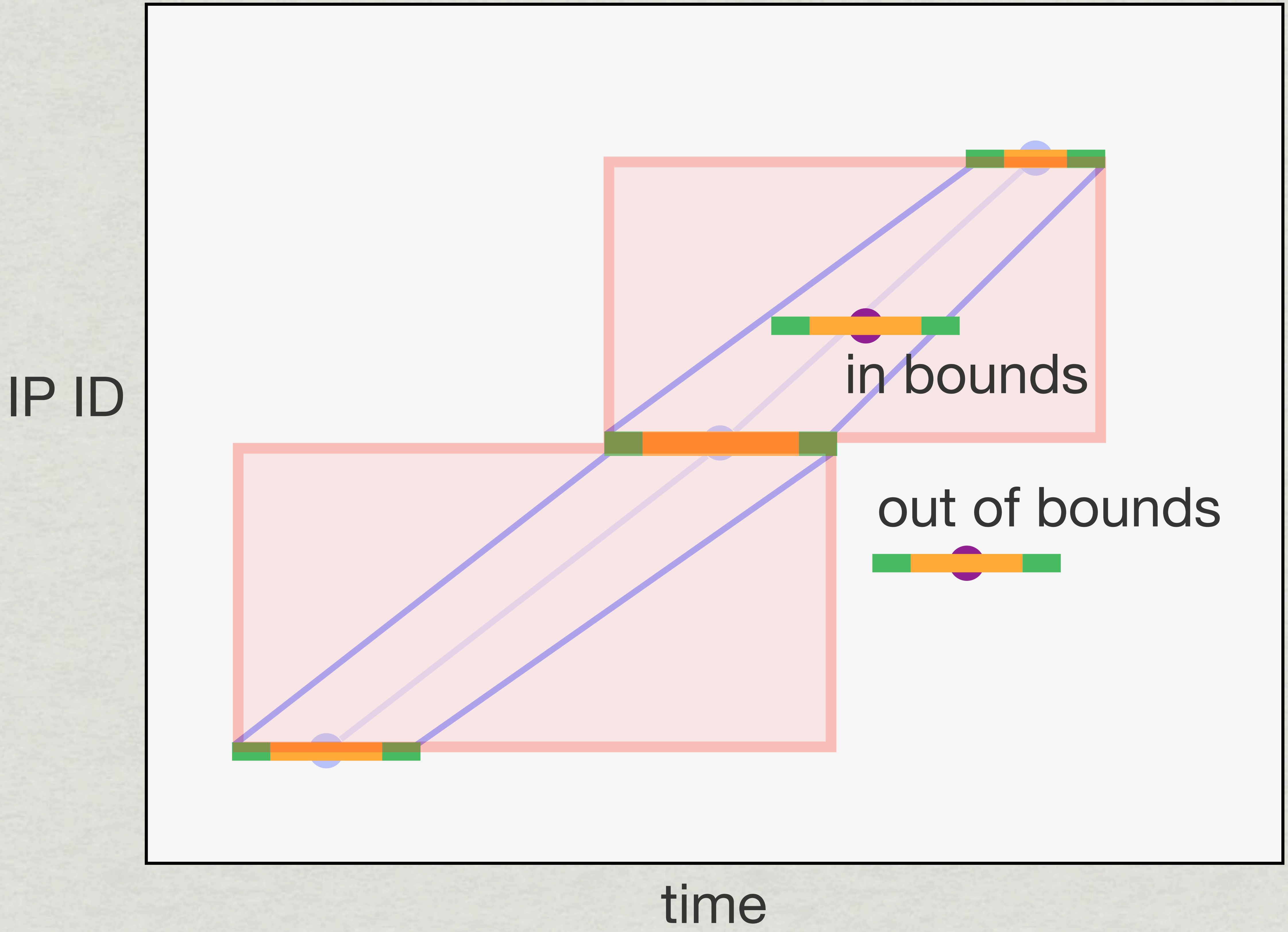


time

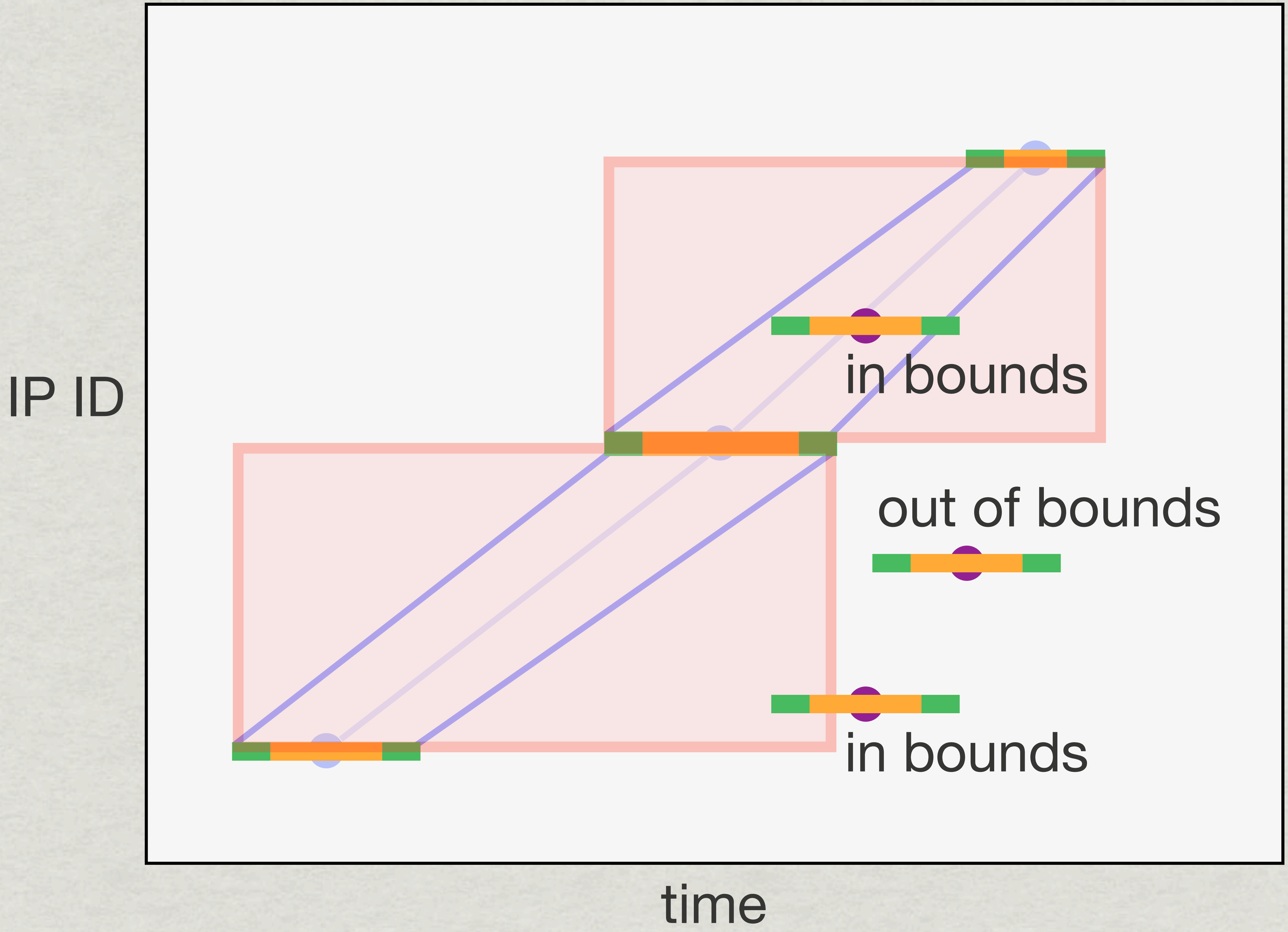
Monotonic Bounds Test



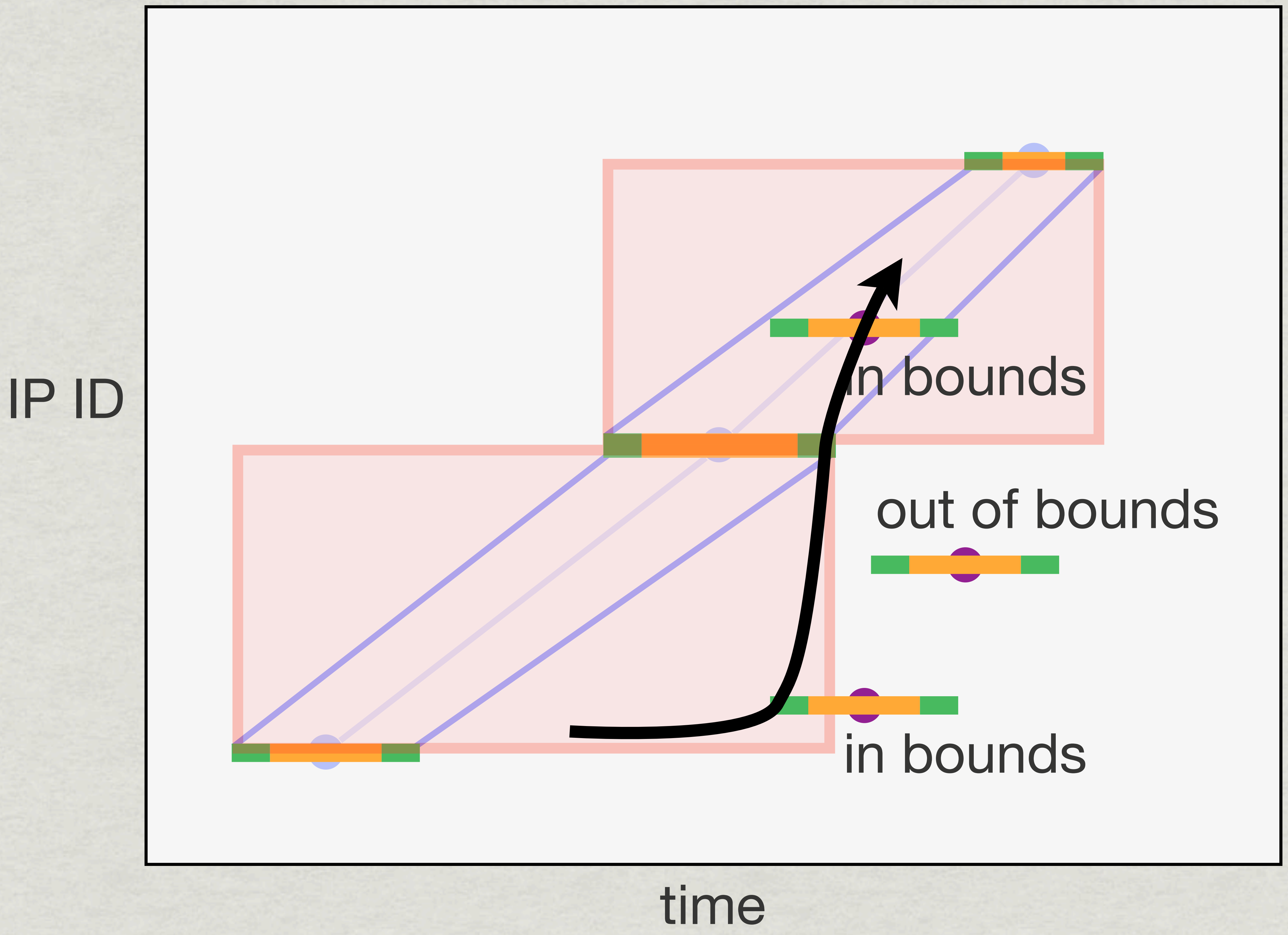
Monotonic Bounds Test



Monotonic Bounds Test



Monotonic Bounds Test



Monotonic Bounds Test

- summary: the monotonic bounds test provides a high-confidence test of a shared counter, and ultimately of aliases
  - based on a necessary condition that ensures convergence to the true positives
    - difference of kind, not just of degree, with RadarGun’s distance test
  - very low false positive rate minimizes further errors caused by taking the transitive closure of alias pairs

MIDAR

- MIDAR probes with a *sliding window* for scalability
  - scales up gracefully
    - can accommodate varying numbers of monitors
    - use “what you have”, not “what you **must** have”
  - reduces chances of rate limiting
- we ran MIDAR on 1 million interfaces with just 27 monitors at 100pps/monitor

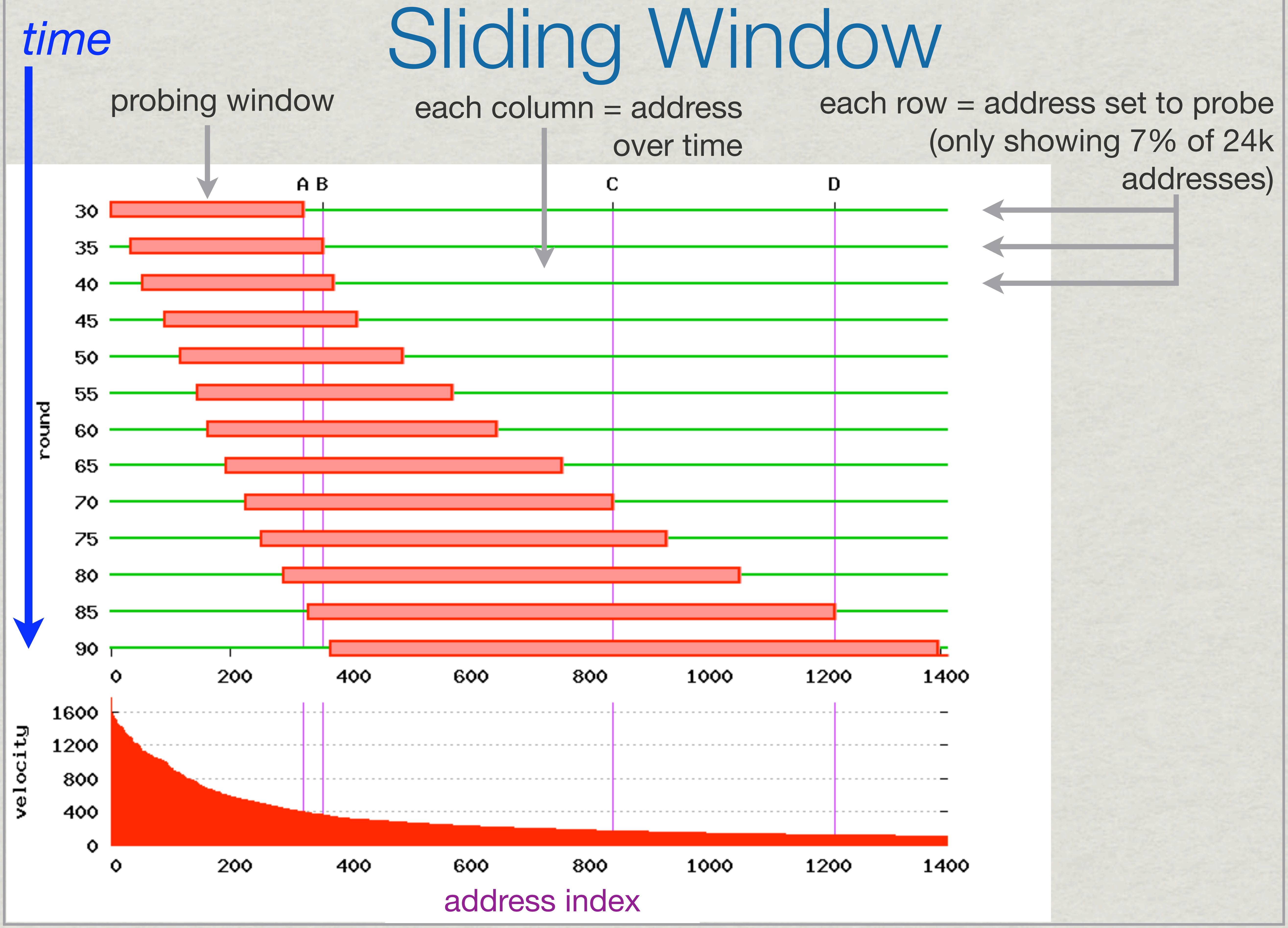
Sliding Window

- the *sliding window* rationale:
  - two interfaces that share a counter will have similar time series and thus similar velocities
    - that is, interfaces with very different velocities cannot be shared, and so we do not need to probe such interfaces closely in time
  - high velocity targets should be probed with tighter probe spacing than low velocity targets
    - need to reduce the bounds in the monotonic bounds test
    - need to be able to detect random IP ID’s
  - low velocity targets can be probed with wide probe spacing because their IP-ID counter changes slowly

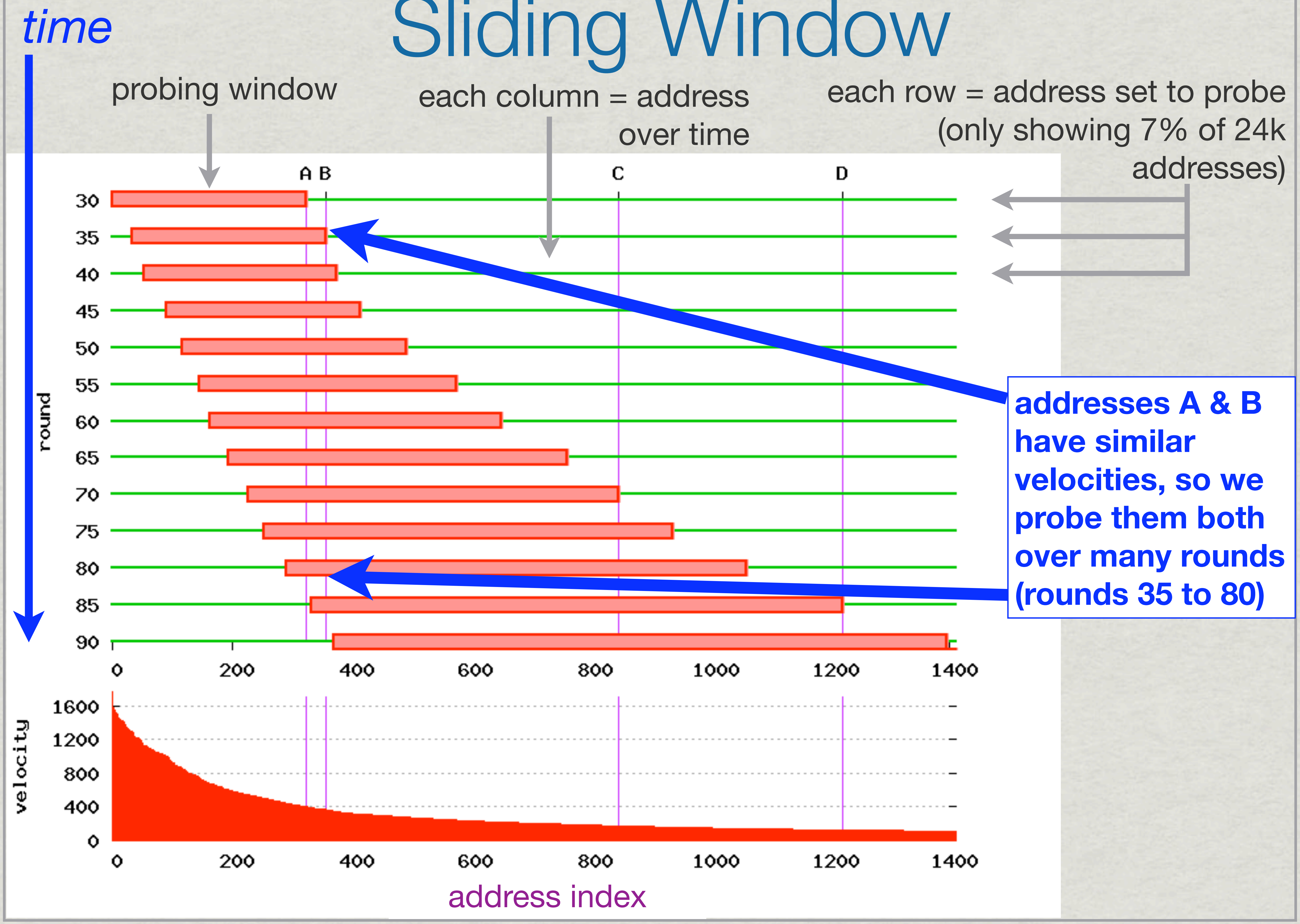
Sliding Window

- implementation:
  - sort targets by descending velocity
  - set up a *window* over an initial segment of the target list
  - loop:
    - probe each address in the window
    - slide window forward (to lower velocity targets) by a small fraction of its size, and increase window size by a small fraction

Sliding Window



Sliding Window



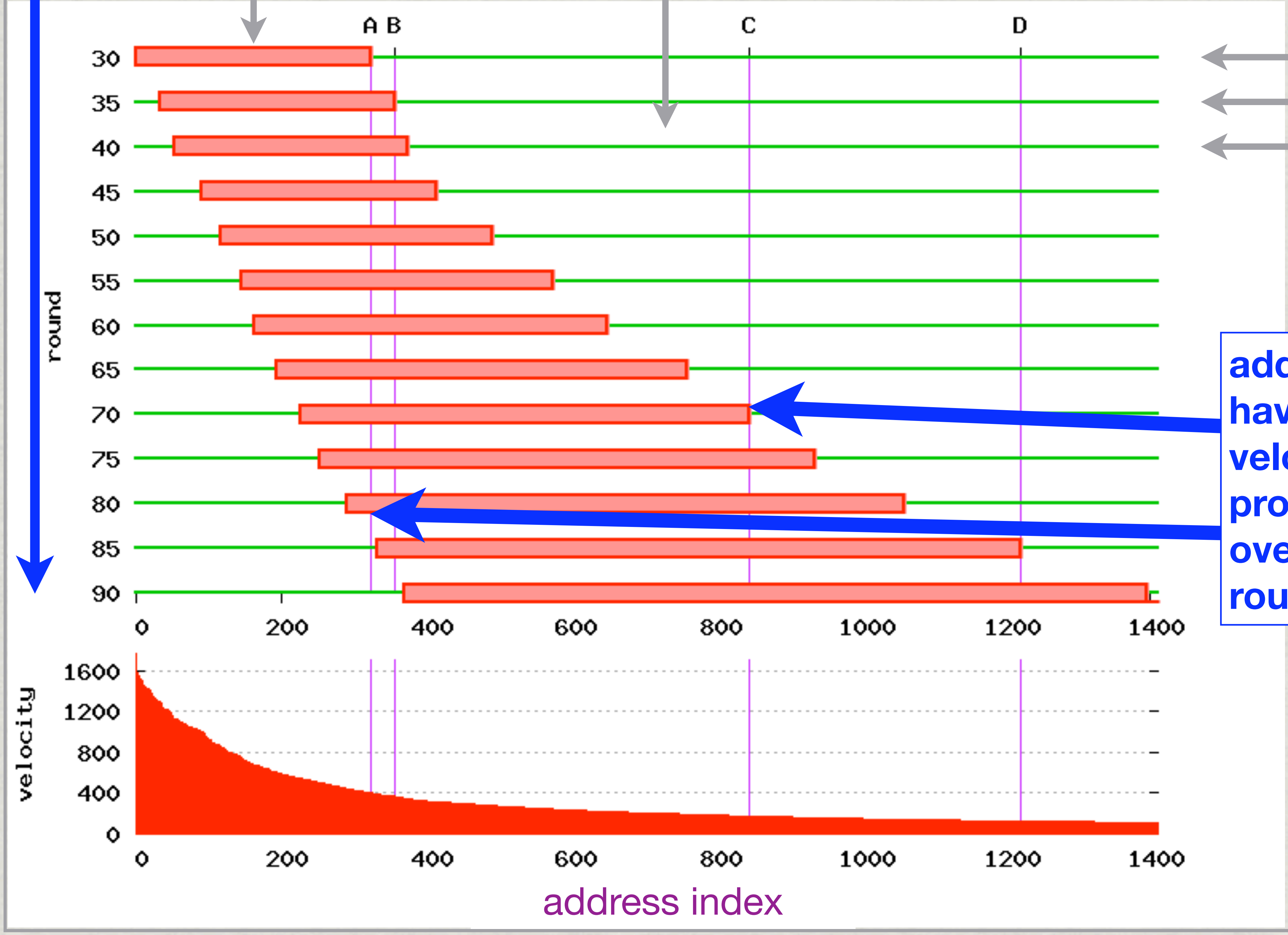
Sliding Window

time

probing window

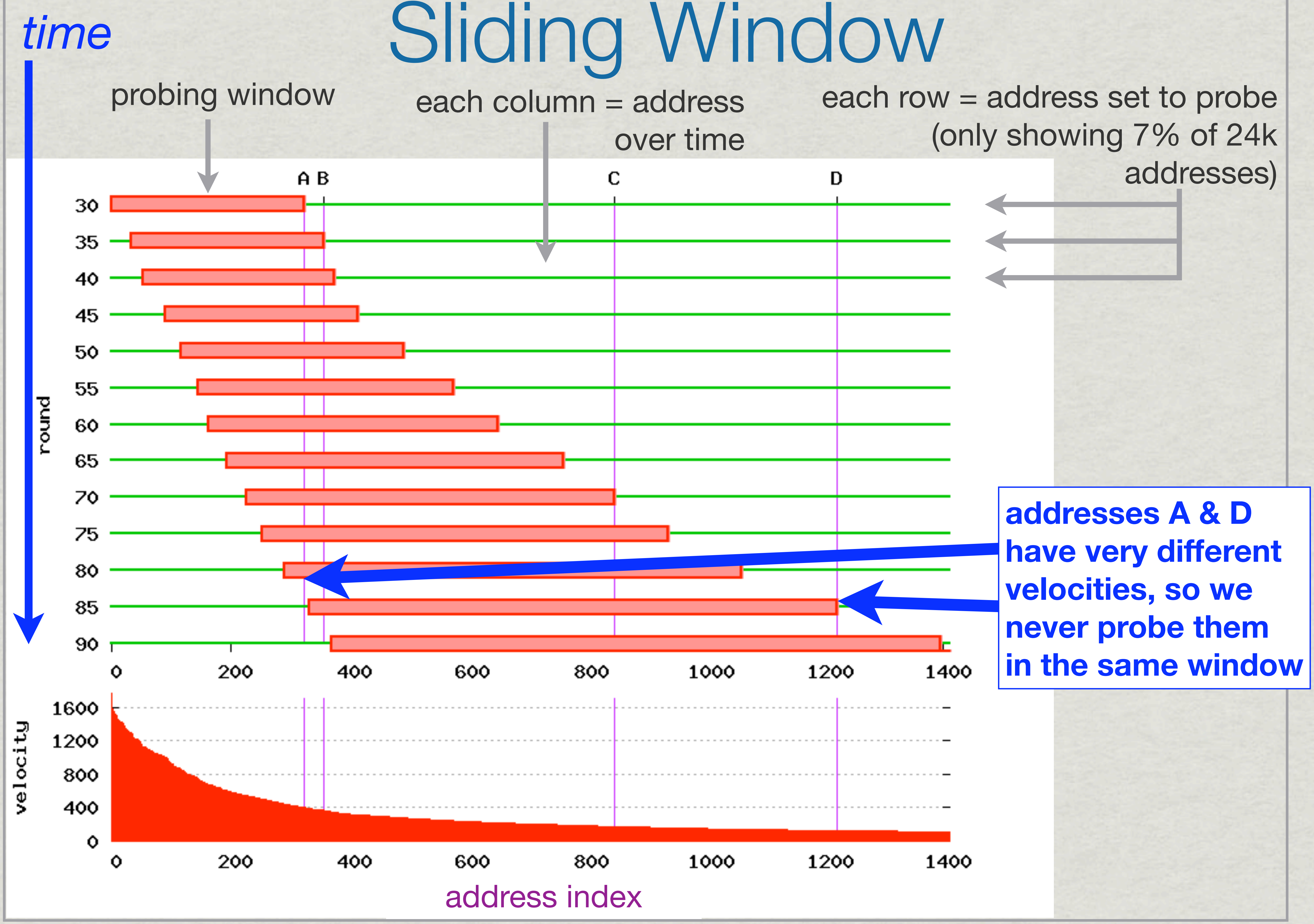
each column = address over time

each row = address set to probe (only showing 7% of 24k addresses)



addresses A & C have different velocities, so we probe them both over only a few rounds (70 to 80)

Sliding Window



addresses A & D have very different velocities, so we never probe them in the same window

More Probe Methods

- MIDAR uses 4 probing methods:
  - TCP ACK (same as RadarGun), UDP, ICMP, and *indir*
    - *indir* reproduces the conditions of the original traceroute used to obtain an interface address
- using additional methods improves response rate

methods				responsive		monotonic	
tcp				747,408	66.57%	481,999	42.93%
	udp			664,742	59.21%	645,103	57.46%
		icmp		953,562	84.94%	390,827	34.81%
			indir	973,199	86.69%	838,826	74.72%
tcp	**udp**	**icmp**	**indir**	1,088,572	96.96%	1,014,999	90.41%

responsive = target responded to at least 75% of probes
monotonic = target's IP-ID time series is monotonic

Counter Sharing

- addresses that respond to multiple methods frequently share counters across methods:

	udp	icmp	indir
tcp	94.97%	87.05%	90.57%
udp		96.15%	95.91%
icmp			95.84%

- cross-method comparison of different addresses may be useful
    - but negative results should not be treated as conclusive
        - caused by per-method or per-interface counter

MIDAR Execution

- **discovery stage**: find candidate alias pairs
- **corroboration stage**: confirm candidates

Discovery Stage

- estimation run
  - find velocities needed for sliding window
  - identify each target's best probe method
    - prefer in descending order TCP, UDP, ICMP, indir
  - can probe each target independently of others
- sliding window run
  - discover candidate alias pairs, including many false positives

Corroboration Stage

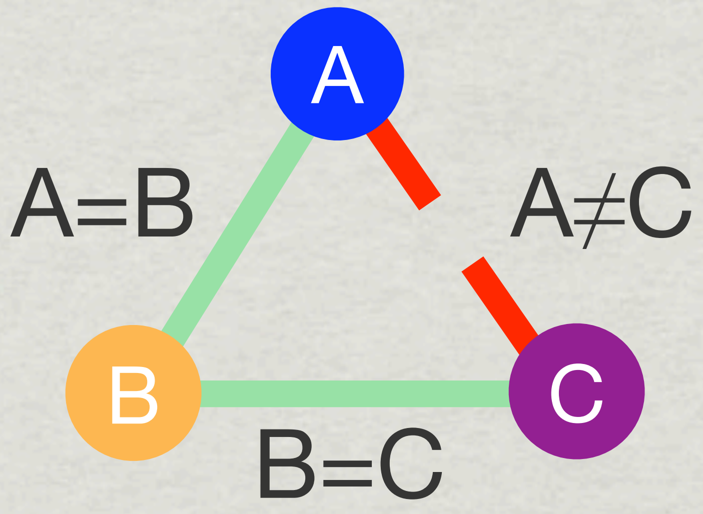
- goal: eliminate all false positives
- probing several hours after discovery stage gives non-shared counters time to diverge
- naive implementation: repeat sliding window run
  - unnecessarily tests pairs that we have already rejected
- optimized implementation:
  - only compare pairs in the transitive closure of the potential alias pairs found in the discovery stage
  - probe alias set members one at a time, with smallest possible spacing that doesn't trigger rate limiting (>500ms)
    - tight spacing reduces false positives

MIDAR Results

- discovery stage (sliding window):
  - probed 1.0 million addresses
  - 486 **billion** pairs compared
  - shared pairs found: 1.6 million (0.00093%)
  - 55k alias sets containing 497k addrs
- corroboration stage:
  - shared pairs found: 428k (26% of discovery stage)
    - not actually 1.2 million false positives; inflated by human error
  - 69k alias sets containing 186k addrs
    - stable across multiple corroboration runs

MIDAR Results

- consistency check: out of 69k sets,187k addrs, 428k pairs after corroboration ...
  - every pair inferred by transitive closure was tested with the monotonic bounds test at least once and passed every time
  - all but 80 pairs were tested at least twice and passed every time
  - only 12 sets (49 addrs) contained transitive *closure conflicts:*



We suspect real network change caused these conflicts and not false positives.

MIDAR Validation

- we compared MIDAR results to ground truth for a tier 1 ISP
  - for comparison, we only consider routers that appear with multiple interfaces in Ark traces
    - observed multi-interface routers (OMIRs)
- **0 false positives**

	full ISP topology	**OMIRs**	**MIDAR**
routers	1,986	983	434
addresses	24,429	4,008	1,284
pairs	611,407	16,900	2,133

Future Work

- MIDAR improvements
  - adapt corroboration spacing to responsiveness
- MAARS: Multi-Approach Alias Resolution System
  - combine MIDAR, kapar, iffinder (and others?)
  - How to use MIDAR negatives to reduce false positives in kapar?