An Internet Protocol Address Clustering Algorithm

Robert Beverly Karen Sollins

MIT Computer Science and Artificial Intelligence Laboratory
{rbeverly,sollins}@csail.mit.edu

December 11, 2008

USENIX SysML 2008

Scope of Talk

1. **Motivation:** Learning to operate in an increasingly complex and malicious Internet
2. **Challenges:** Many at Internet-scale, in dynamic environment
3. **Needed:** Building-blocks for network and systems designers
4. **Approach (And why we didn't do *X*):** An IP Clustering Algorithm as one building-block with many practical applications
5. **Results:** Predictive performance, including ability to detect changed network portions
6. **Future:** What's next, work building upon this research

CSAIL

Outline

Evolution of Internet Architecture

The Internet is a phenomenal success, but original assumptions underlying its design have changed, e.g.:

- Security ... historically a second concern
- Trust ... in a world of botnets, phishers, etc
- Scale ... traffic, routes, multi-homing, etc
- Complexity ... policy constraints, network demands, economics

And it's continuing to evolve, grow more complex. E.g.:

- Scale along new dimension: bad hosts/users
- Support increasingly critical services
- Trend to content-based networking
- Adding devices with intermittent connectivity (sensor nets, DTNs)

CSAIL

The Research Challenge

- Apply statistical learning to *embrace* Internet's natural complexity
- Find predictive models: generalize to unseen data, new situations

Networking problems are a challenging learning environment:

- Non-stationary
- On-line
- Distributed
- Tradeoff between effort vs. improvement obtained vs. errors

Needed:

Building blocks to realize ML promise while mitigating challenges

CSAIL

Outline

IP Clustering as a Building Block

- Internet Protocol (IP) v4 addresses are unsigned 32-bit integers
  - e.g. **`18.26.0.230`**
- Hosts given addresses based on the network on which they reside

An IP Address Clustering Algorithm:

- Supervised learning (describe change detection later)
- Given (informally):
  - Training samples from a portion of the IP address space
  - Labeled with a real or discrete property (e.g. latency, security reputation, etc)
- Find a “good” partitioning of the space

Why?

IPs as Identifiers:

For better or worse, IP addresses are overloaded. IPs serve as identifiers for:

- End hosts
- Location in the network topology
- Location in the physical topology

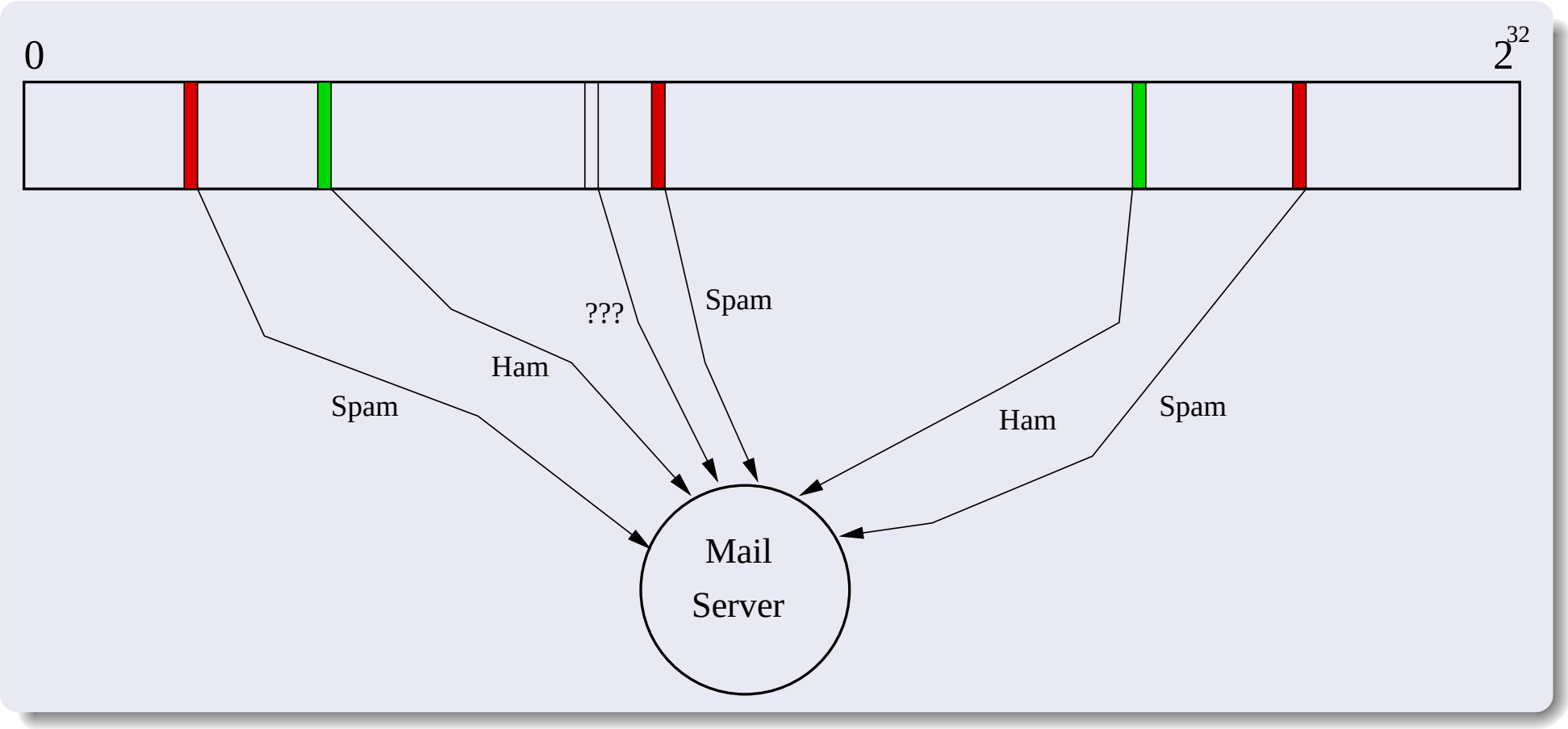
Implications of this conflation:

- Security policy (firewalls, etc)
- Reputation (spam sources, etc)
- Service selection, load balancing, performance optimization (P2P, CDNs, etc)
- User-directed routing, grid computing, more...

For example...

CSAIL

Practical Example: Internet Mail Server



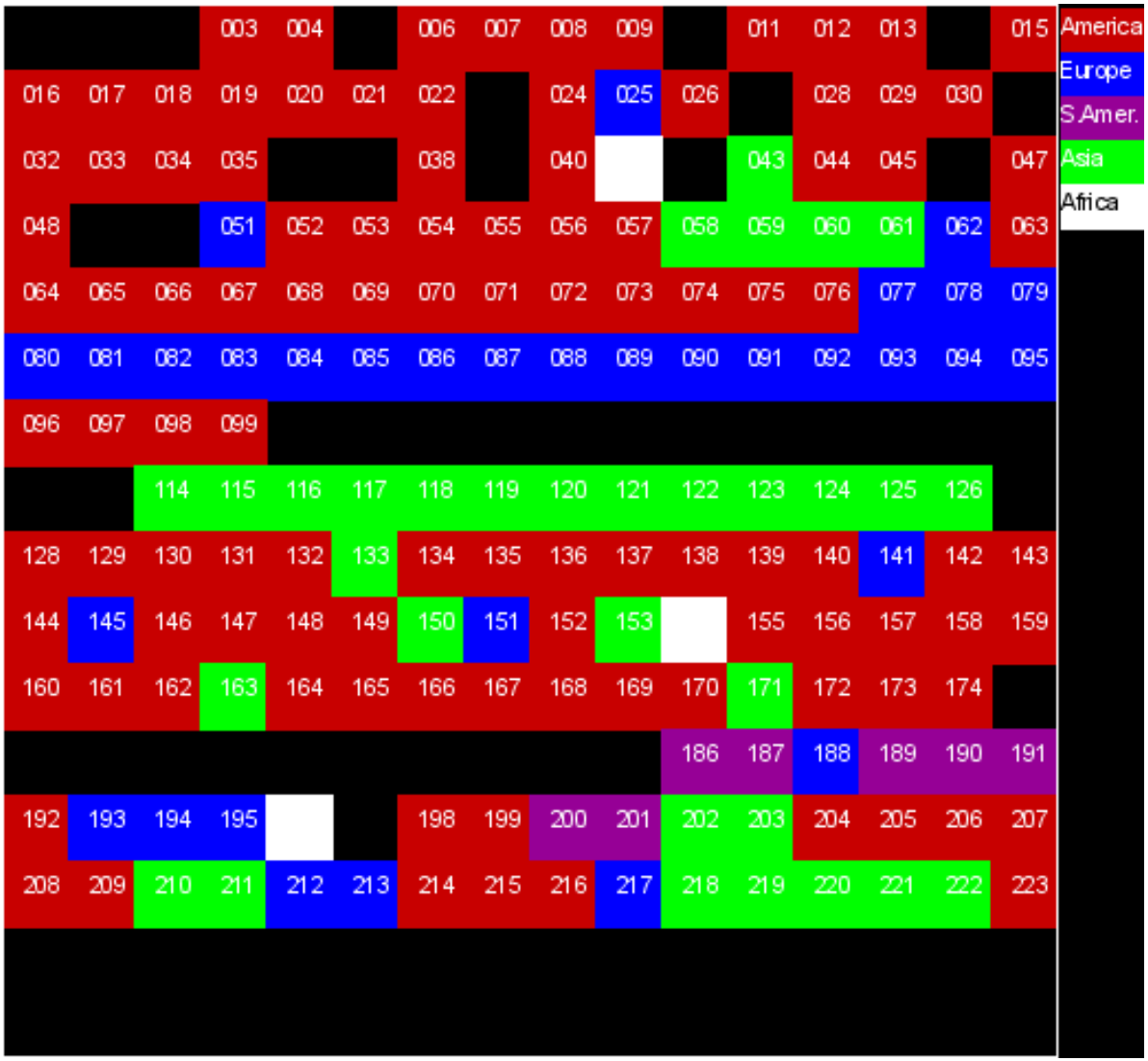
- Assuming spam originates from “grouped” hosts/networks
- Can a mail server build a *predictive* model of likely spam sources/networks?

Emulating Ideal World

- Ideally, a “knowledge plane” would provide oracle information on every node in the network
- Unfortunately, the size ($\sim 3B$ addresses, $\sim 300K$ networks) and dynamics of the Internet generally precludes complete knowledge
- Instead, leverage Internet’s inherent *structure* due to physical, logical and administrative boundaries

How much structure exists?

IANA /8 Allocations by Continent

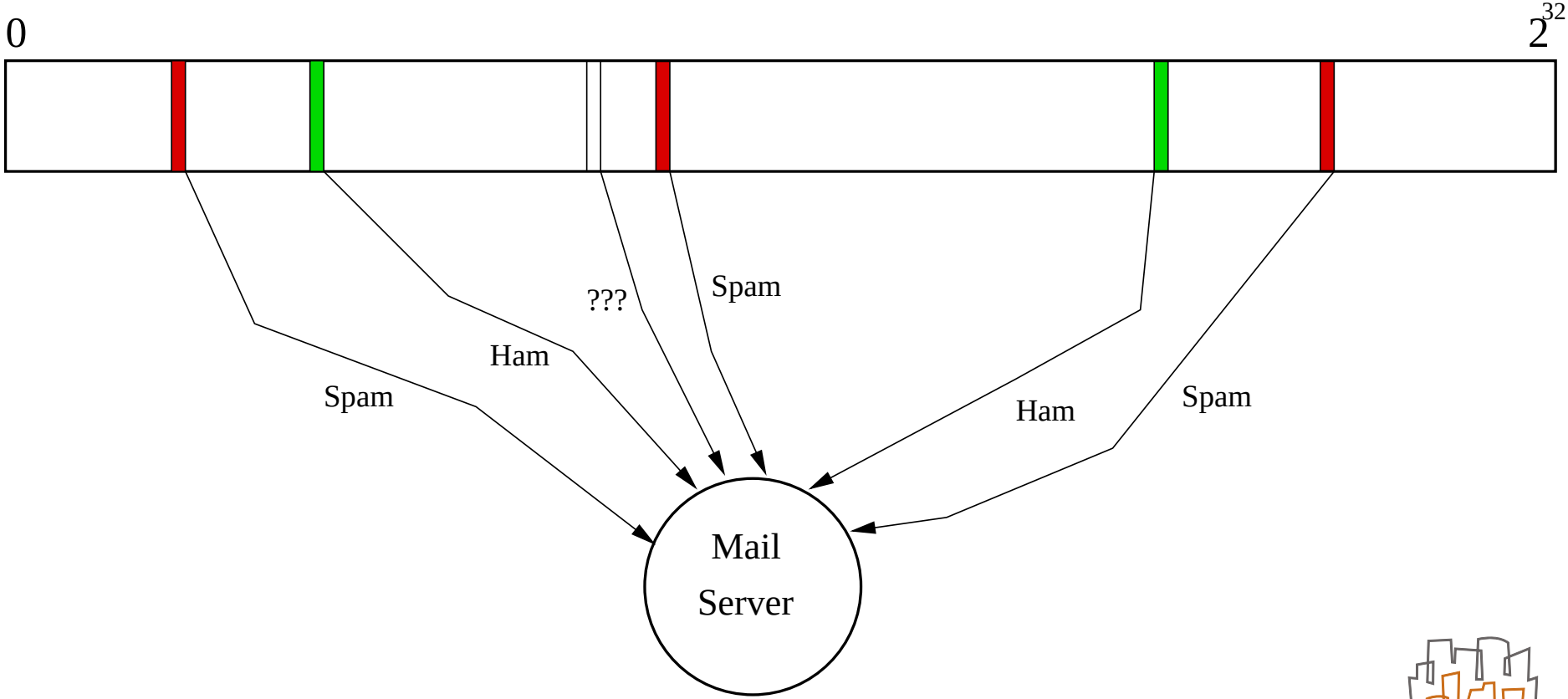


- IP addressing is hierarchical
- Discontinuous, fragmented
- Correct granularity?
- Hosts within same sub network likely have consistent policy, latencies, routes, etc.

Learning Structure

Idea 1: Statically divide input space

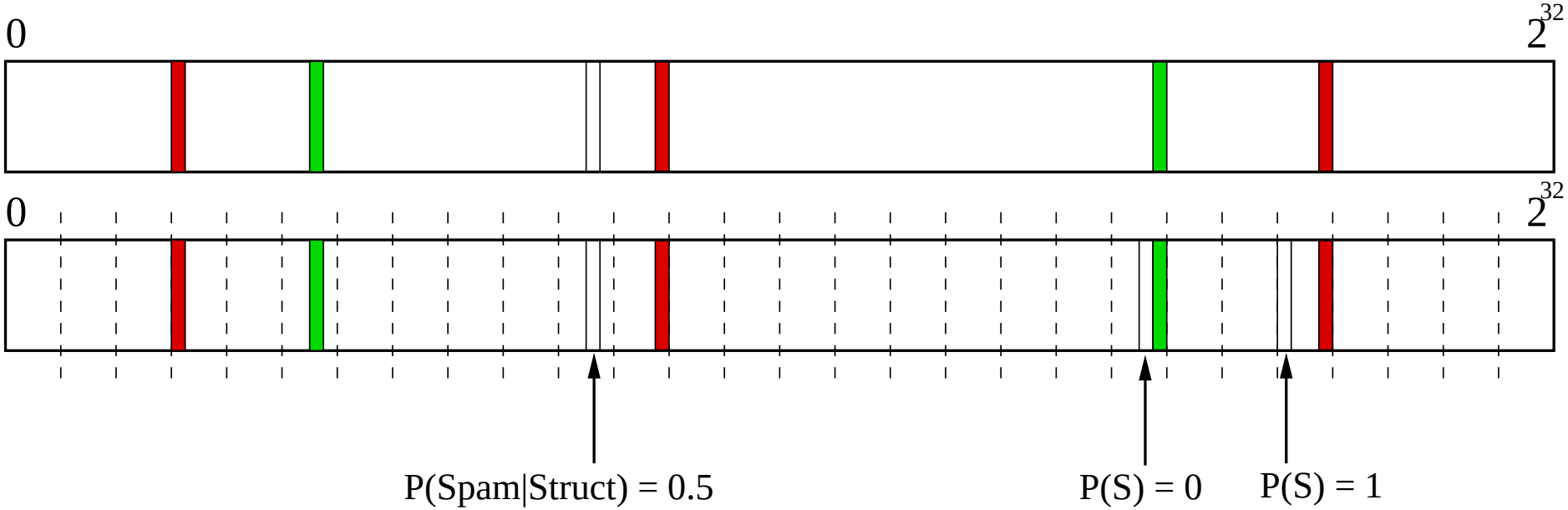
Email server example:



Learning Structure

Idea 1: Statically divide input space

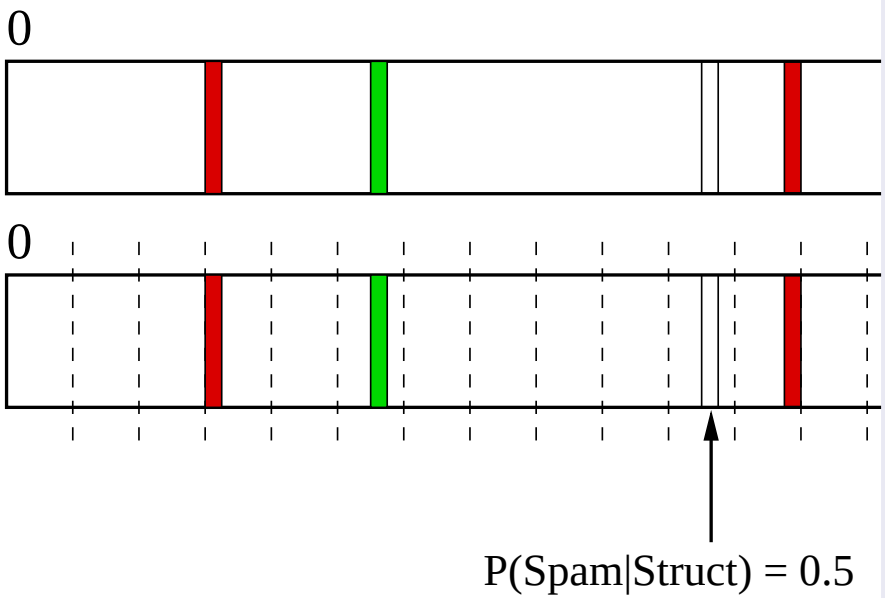
Email server example:



Learning Structure

Idea 1: Statically divide input space

Email server example:



Issues:

- Pre-supposes a structure; we may *want* to infer this
- Requires large amount of memory to perform decently
- Static alignment with data leads to inferior performance compared to other approaches

Idea 2: Leverage network routing

IP Hierarchy and Aggregation:

- Blocks (varying size) of contiguous addresses assigned to networks (e.g. AT&T, UCSD, Level3, etc)
  - Aggregated unit: **`prefix/mask`** (defined precisely in paper)
  - E.g. **`18.0.0.0/8`** is a large prefix with 2^{24} addresses
- Smaller blocks are further sub-delegated (“smaller” prefixen)
- Routers exchange aggregated prefixes, perform per-packet longest-match forwarding to get packet closer to destination

Implication:

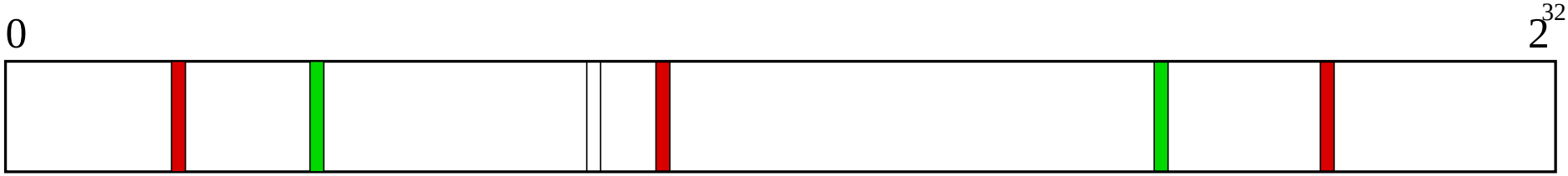
- There’s an existing source of rich data
- e.g. [Balachandar & Wang]

For example...

Learning Structure

Idea 2: Leverage network routing

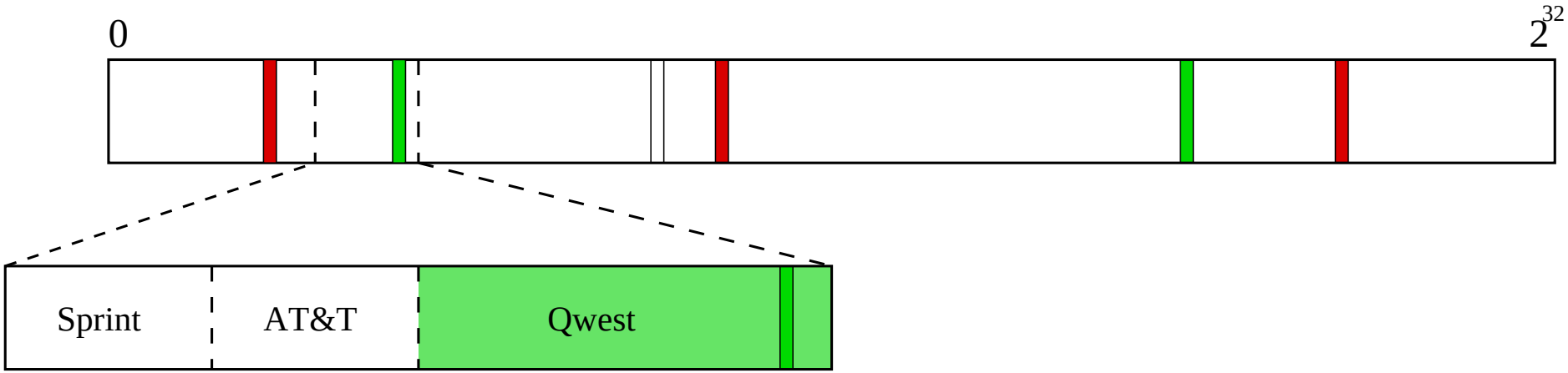
Email server example:



Learning Structure

Idea 2: Leverage network routing

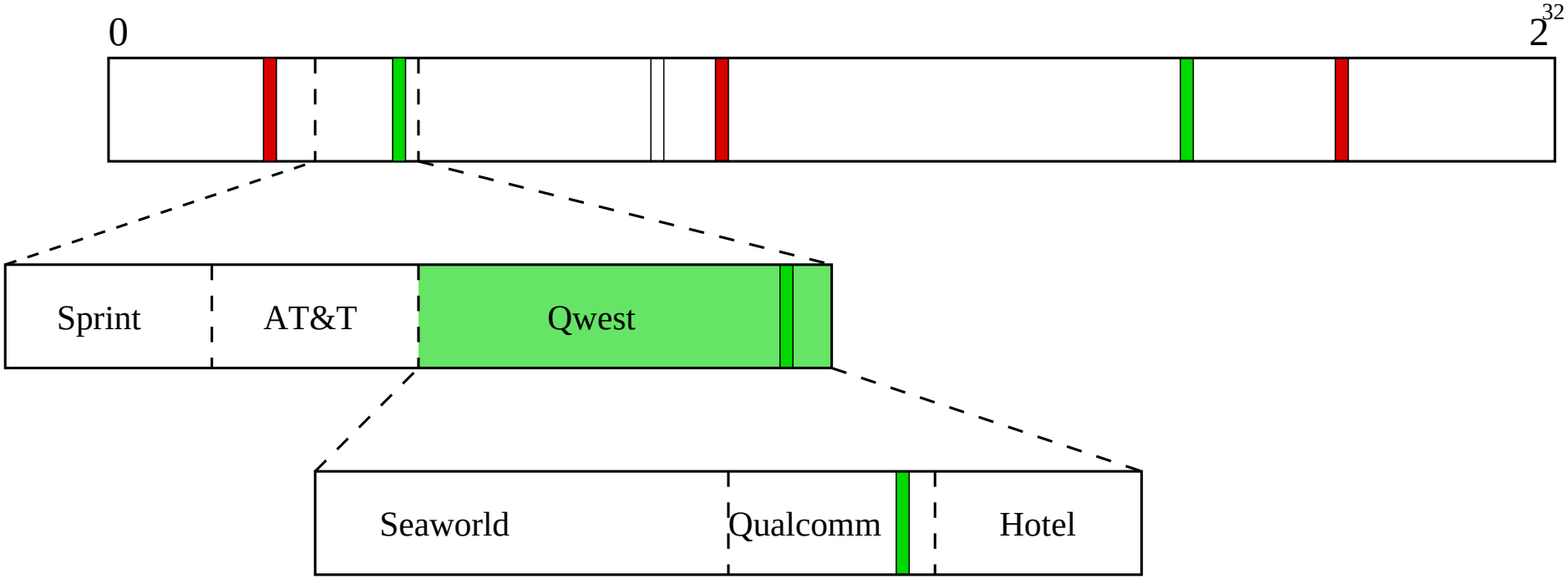
Email server example:



Learning Structure

Idea 2: Leverage network routing

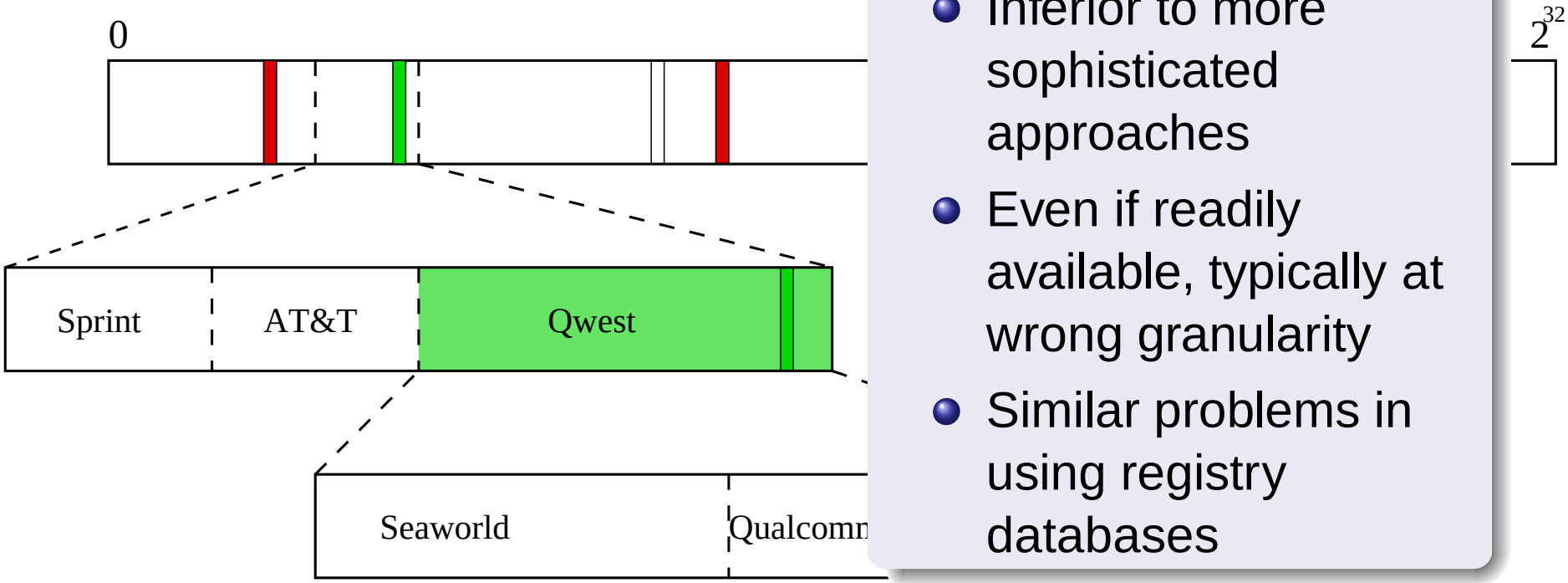
Email server example:



Learning Structure

Idea 2: Leverage network routing

Email server example:



Issues:

- Inferior to more sophisticated approaches
- Even if readily available, typically at wrong granularity
- Similar problems in using registry databases

CSAIL

How to Best Learn/Exploit Structure?

- Temptation to formulate network task into a learning problem (i.e. use out-of-the-box “black-box” algorithms)
  - Often suboptimal
  - e.g. how to set thresholds, regularization parameter, kernel, etc?
- How about Internet-specific learning algorithms?
  - Leverage domain-specific knowledge
- Learn in a way amenable to non-stationary environment, on-line directed learning

As Important:

- Must be fast (ideally suitable for Internet core / high-speed routers)
- Memory efficient (think FIBs not RIBs)

CSAIL

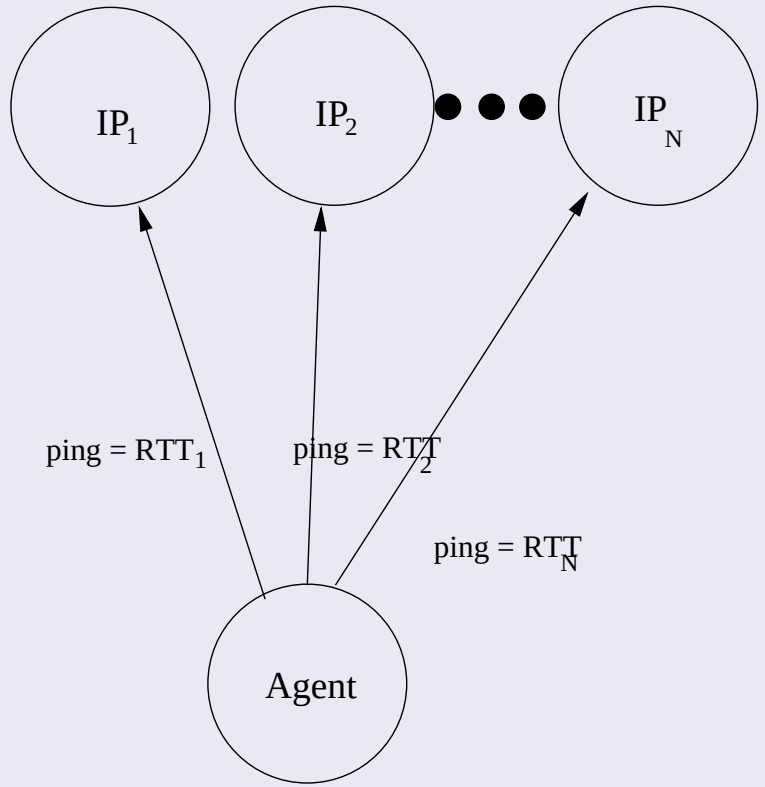
Outline

Data Set

Latency Data Set

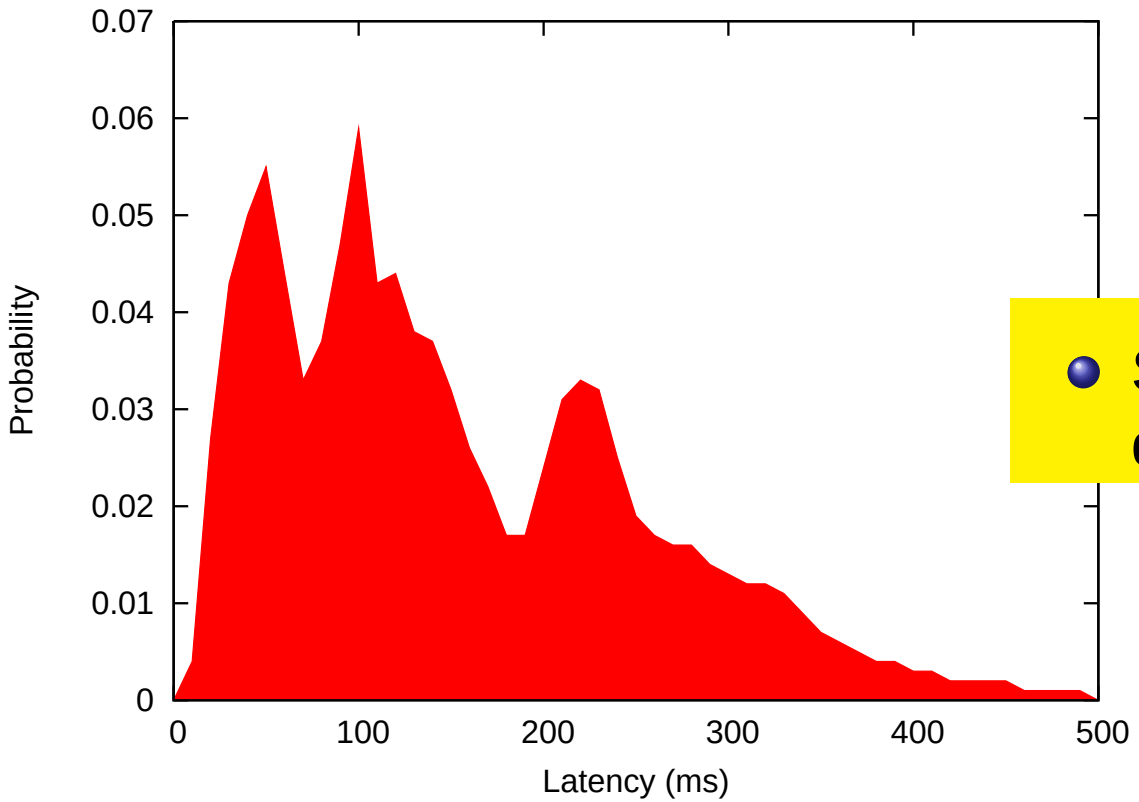
- Reference data set drawn from live Internet measurements
- Use round-trip latency as per-IP property (label)
- Note algorithm isn't specific to latency prediction
- Latency is evocative of many structural properties (e.g. latencies of sub-networks are often a function of the network to which they belong)

Live RTT Measurements:



Data Set

- Find: 30,000 random Internet hosts responding to ping
- Gather: Average latency to each over 5 pings

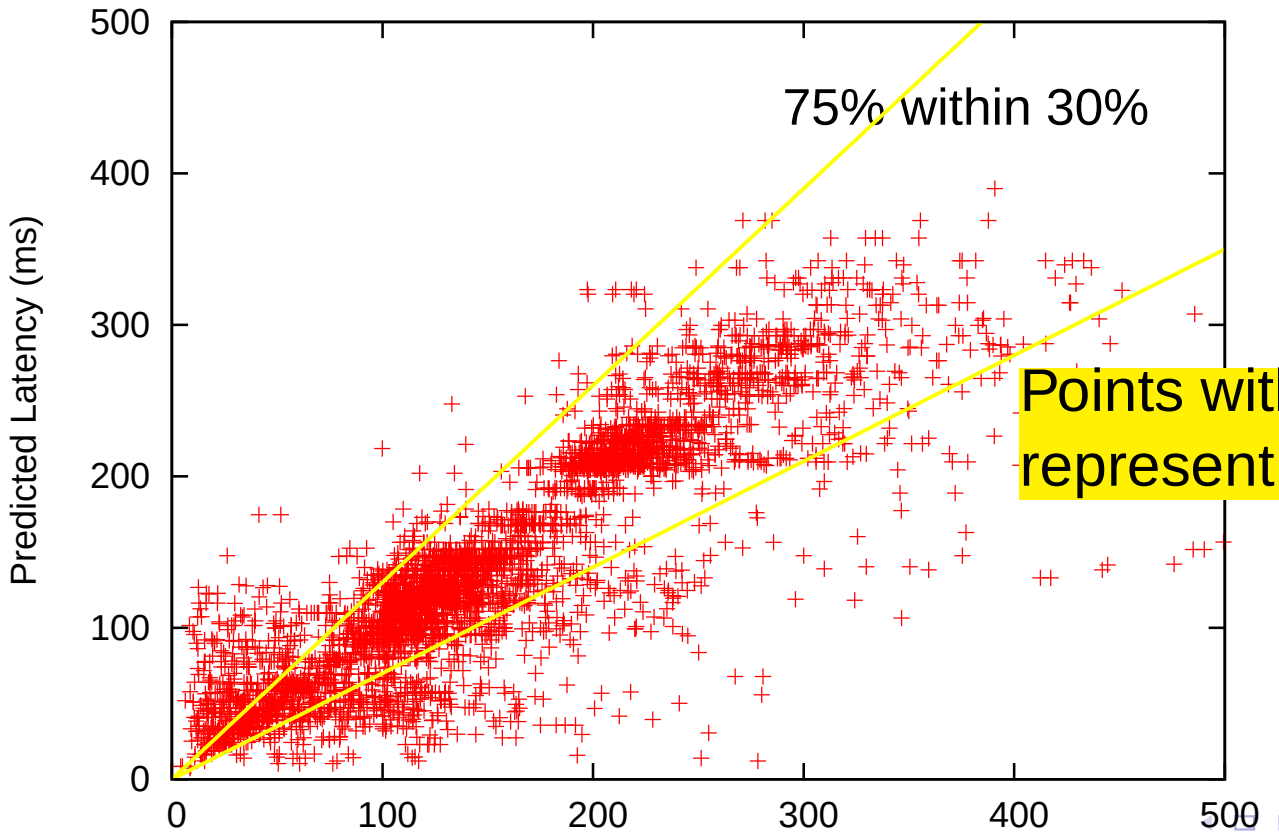


- Several modes, non-trivial distribution

Black-block Performance

Let's try out-of-the-box SVM regression:

- Predict latency to unknown destinations
- With lots of tuning, performs reasonably well; several insights from feature selection



Points within yellow lines represent good predictions

What about the network?

Cool, but...

- Highly (unnatural) parametric models?

What about the network?

Cool, but...

- Highly (unnatural) parametric models?

$$\sum_{t=1}^{n} \alpha_t - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \mathbf{K}(\phi(x_i), \phi(x_j)) \text{ s.t. } C \geq \alpha_t \geq 0, \sum_{t=1}^{n} \alpha_t y_t = 0$$

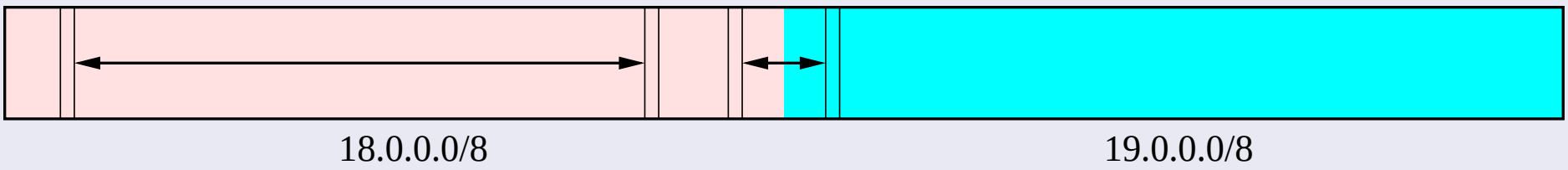
What about the network?

Cool, but...

- Highly (unnatural) parametric models?

$$\sum_{t=1}^{n} \alpha_t - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \mathbf{K}(\phi(x_i), \phi(x_j)) \text{ s.t. } C \geq \alpha_t \geq 0, \sum_{t=1}^{n} \alpha_t y_t = 0$$

- Artificial geometries? (How "close" are `18.255.255.255` and `19.0.0.1`?)



What about the network (con't)?

And...

- Structural, temporal network dynamics?
- When, how often to retrain?
- On-line learning?

For Example, Latency Prediction:

- Structural changes $\rightarrow$ new link, routing change
- Temporal effects $\rightarrow$ congestion, time-of-day

Change Point Detection

Change point detection:

- Assume errors are normally distributed
- Change from known initial world $\theta_0 = N(\mu_0, \sigma)$
- To θ_1, unknown μ_1 change

Generalized Likelihood Ratio:

- Perform double maximization, derivative:

$$g_k = \frac{1}{2\sigma^2} \max_{1 \le j \le k} \frac{1}{k - j + 1} \left[\sum_{i=j}^{k} (x_i - \mu_0) \right]^2$$

Change Point Detection

Change point detection:

- Assume errors are normally distributed
- Change from known initial world $\theta_0 = N(\mu_0, \sigma)$
- To θ_1, unknown μ_1 change

Generalized Likelihood Ratio:

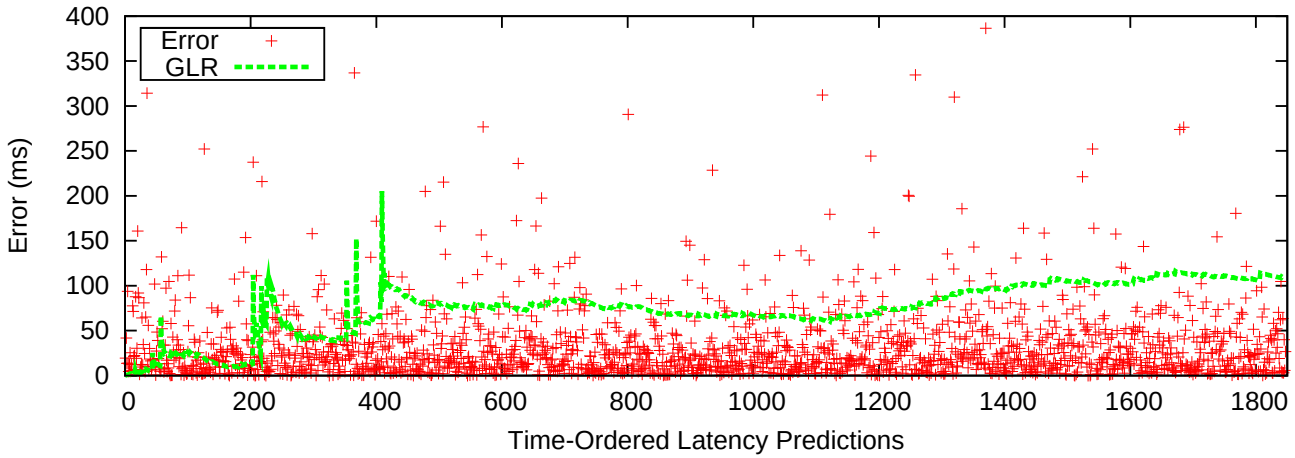
- Perform double maximization, derivative:

$$g_k = \frac{1}{2\sigma^2} \max_{1 \le j \le k} \frac{1}{k - j + 1} \left[\sum_{i=j}^{k} (x_i - \mu_0) \right]^2$$

GLR

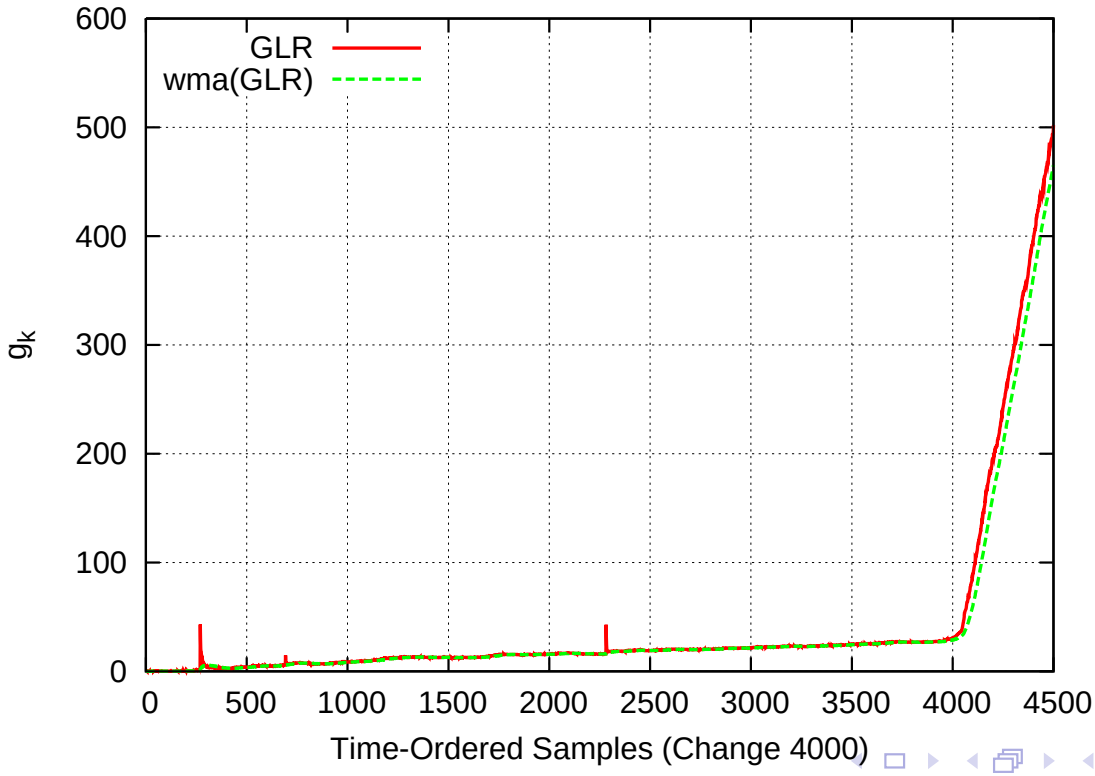
GLR as a test statistic on our data

- Learn model, predict, receive ground truth, update
- Real data, synthetic errors begin at 1000th sample



GLR

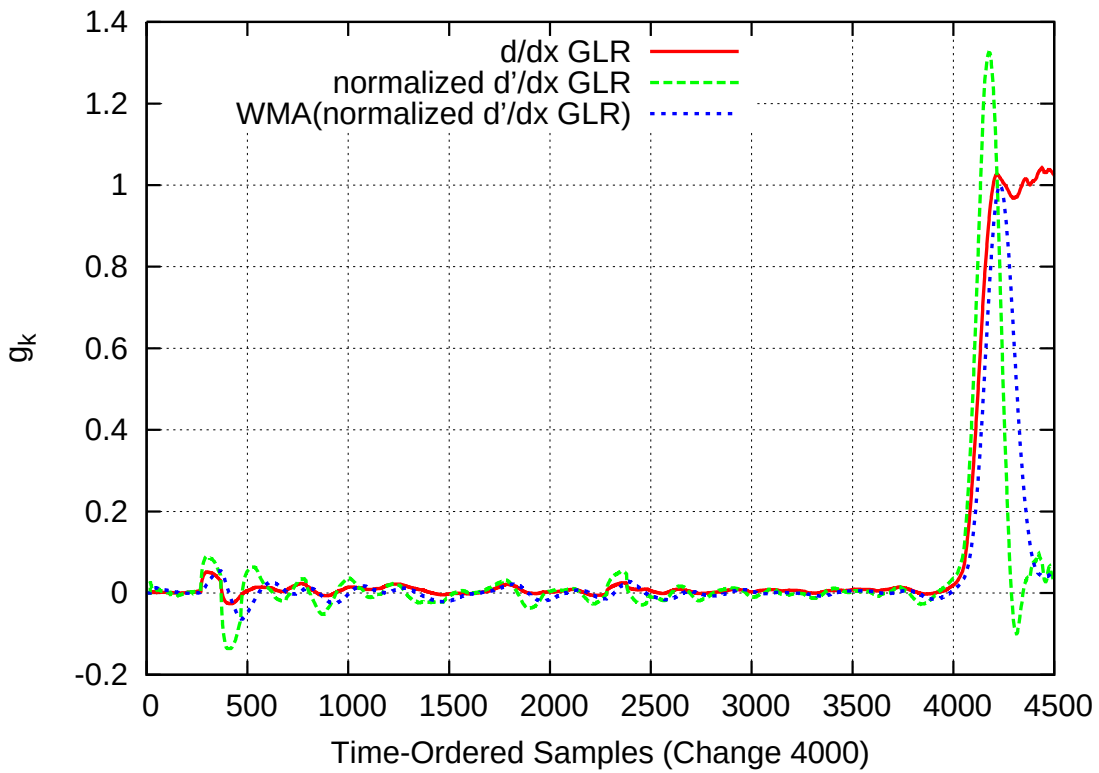
- GLR traditionally used in operations management, etc.
- In our environment, test error $\neq$ train error
- $\rightarrow g_k$ drifts positive
- g_k drifts with slope of β until change to β'



GLR

Overcoming the drift effect

- Take first derivative to get step function
- Take second derivative to get impulse response



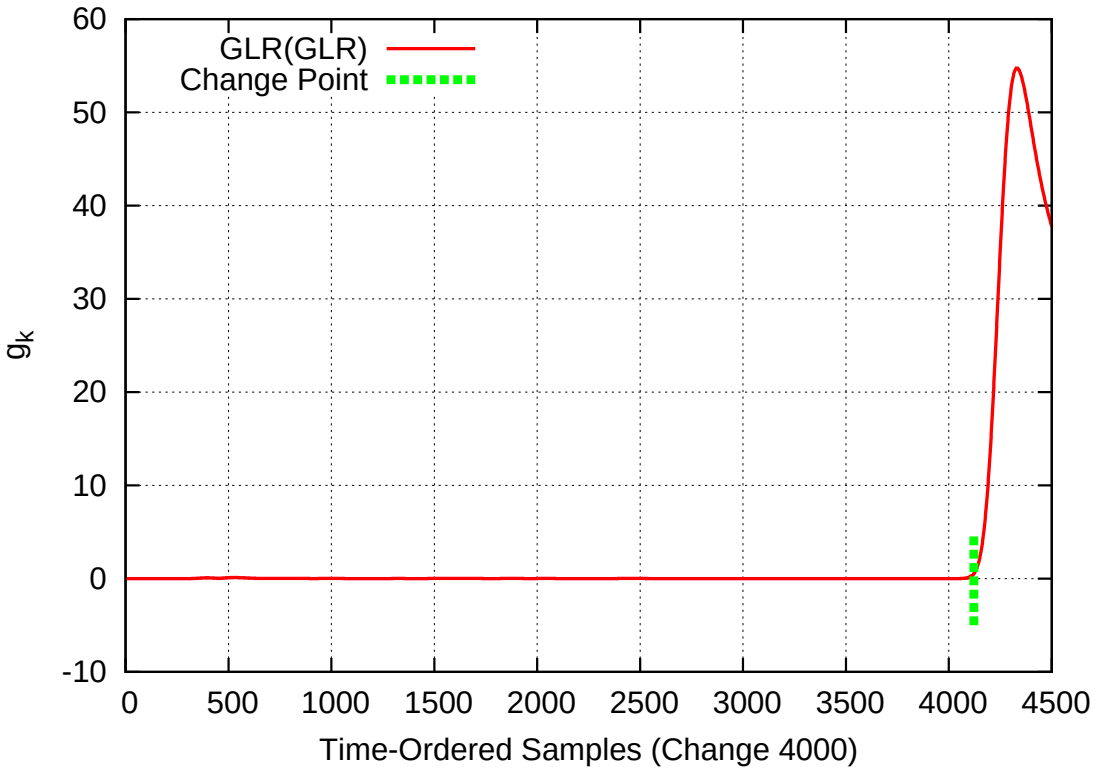
Intuition:

- g_k drifts with slope of β until change to β'
- 1st derivative step function still requires thresholding
- 2nd derivative: $\frac{d}{dx}$ of a constant is zero
- Can now edge trigger on impulse response

GLR

Result:

- Decision function for predicting a change in a supervised learning problem



Take away: a principled method to detect structural network change

An IP Clustering Algorithm

Dynamics:

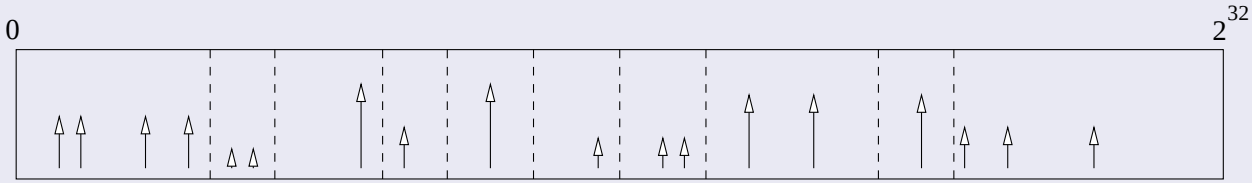
- Incorporate dynamics into model
- GLR provides a means to detect change
- But what *portion* of the network?

Domain knowledge

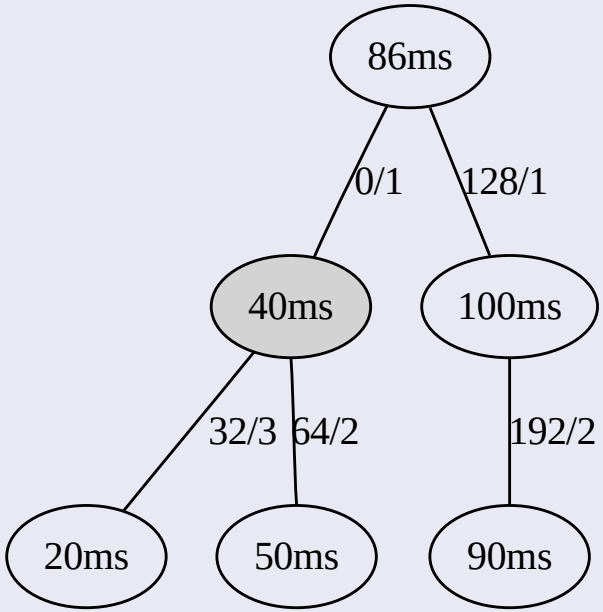
- IP address blocks are assigned on 2^x boundaries
- Can we incorporate this domain-specific knowledge?

An IP Clustering Algorithm

Induces a partitioning over an IP address space:



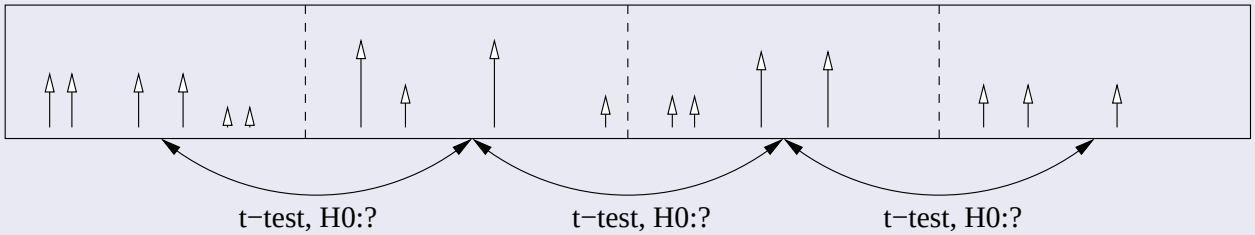
Maintain partitioning in a binary radix trie:



An IP Clustering Algorithm

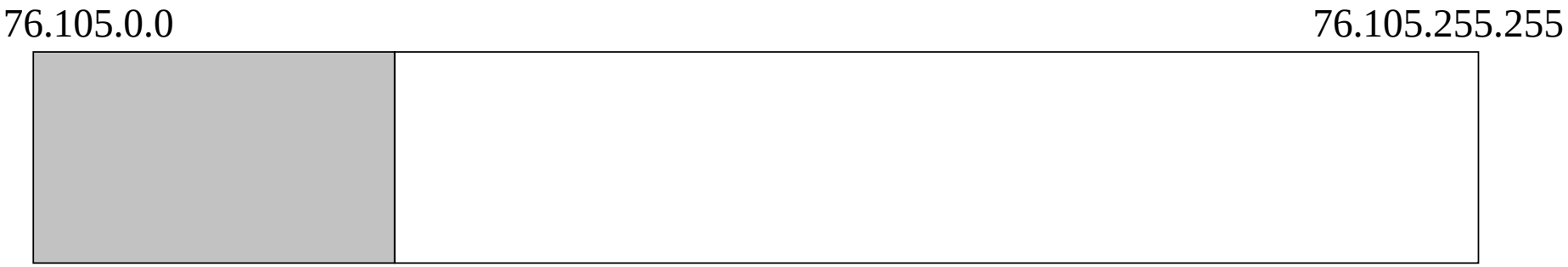
Divisive formulation

- Perform a t-test on permutations of 2^i input partitionings



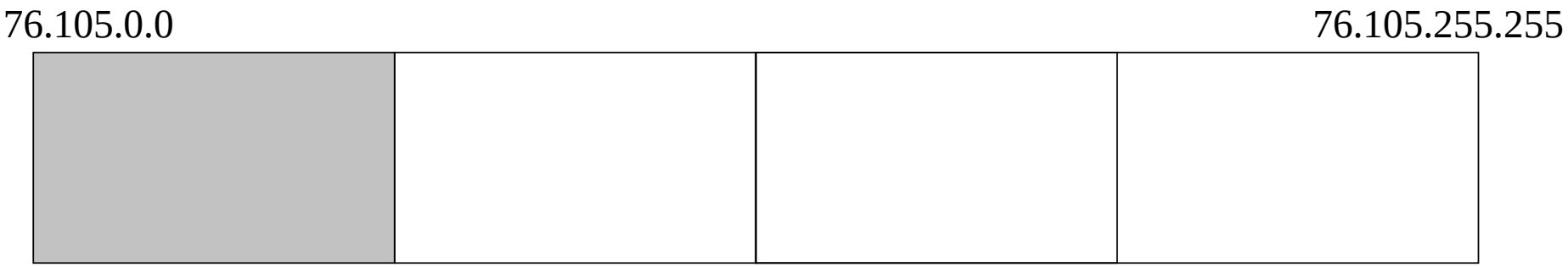
- Gives a strong statistical notion of whether points come from same distribution, i.e. common latencies
- Use t-test to drive partitioning; each partition inserted into radix trie $\rightarrow$ longest prefix matching
- Also an agglomerative version (build up)

Maximal Partitioning



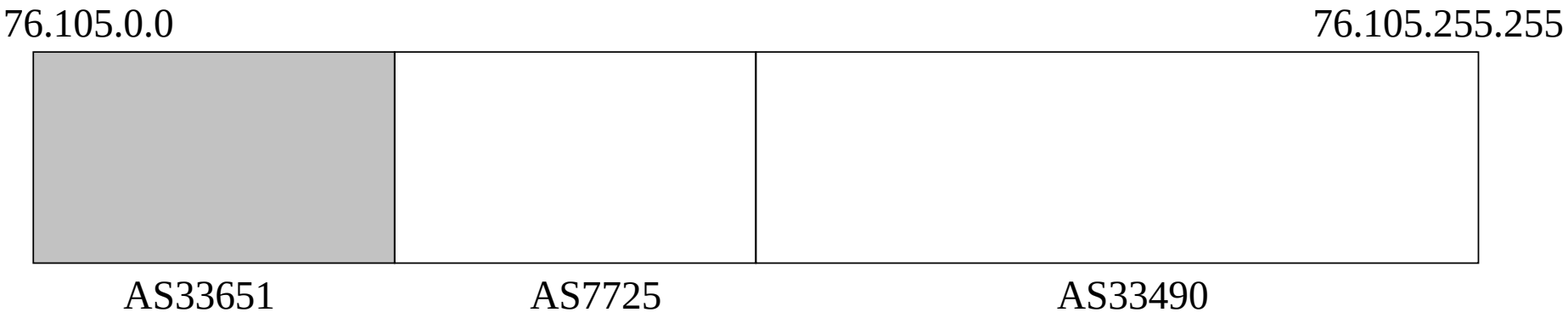
- Partition from `76.105.64.0` to `76.105.255.255` is not valid

Maximal Partitioning



- Partition from `76.105.64.0` to `76.105.255.255` is not valid
- Divide into 4 equally sized 2^{14} prefixes?

Maximal Partitioning



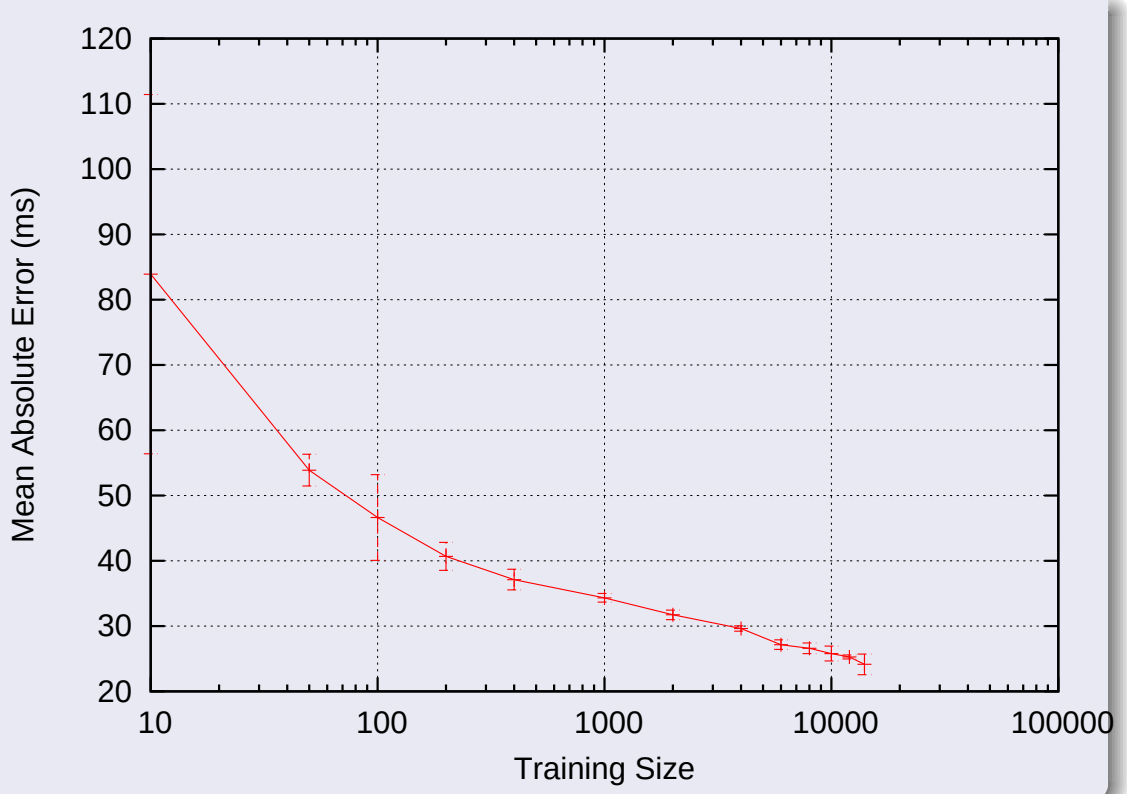
- Partition from `76.105.64.0` to `76.105.255.255` is not valid
- Divide into 4 equally sized 2^{14} prefixes?
- No, example shows three different ASes:
  - `76.105.0.0/18`: Sacramento, CA
  - `76.105.64.0/18`: Atlanta, GA
  - `76.105.128.0/17`: Oregon

- Take away: incorporate domain specific knowledge

CSAIL

Outline

Performance



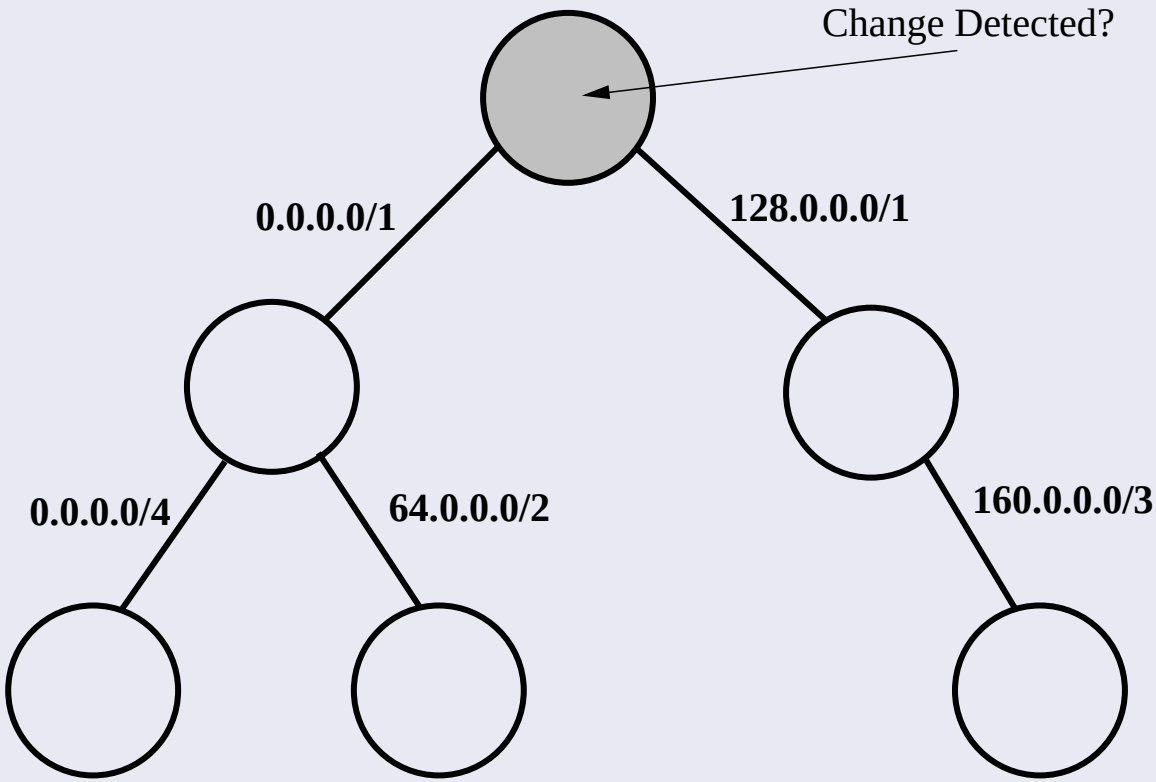
- Less than 40ms error with only 1000 training points
- $\sim$ 24ms error, tight bounds with 10,000 training points
- $\sim$ 130kB memory to maintain binary trie
- Lookups in $O(b)$ time

An IP Clustering Algorithm

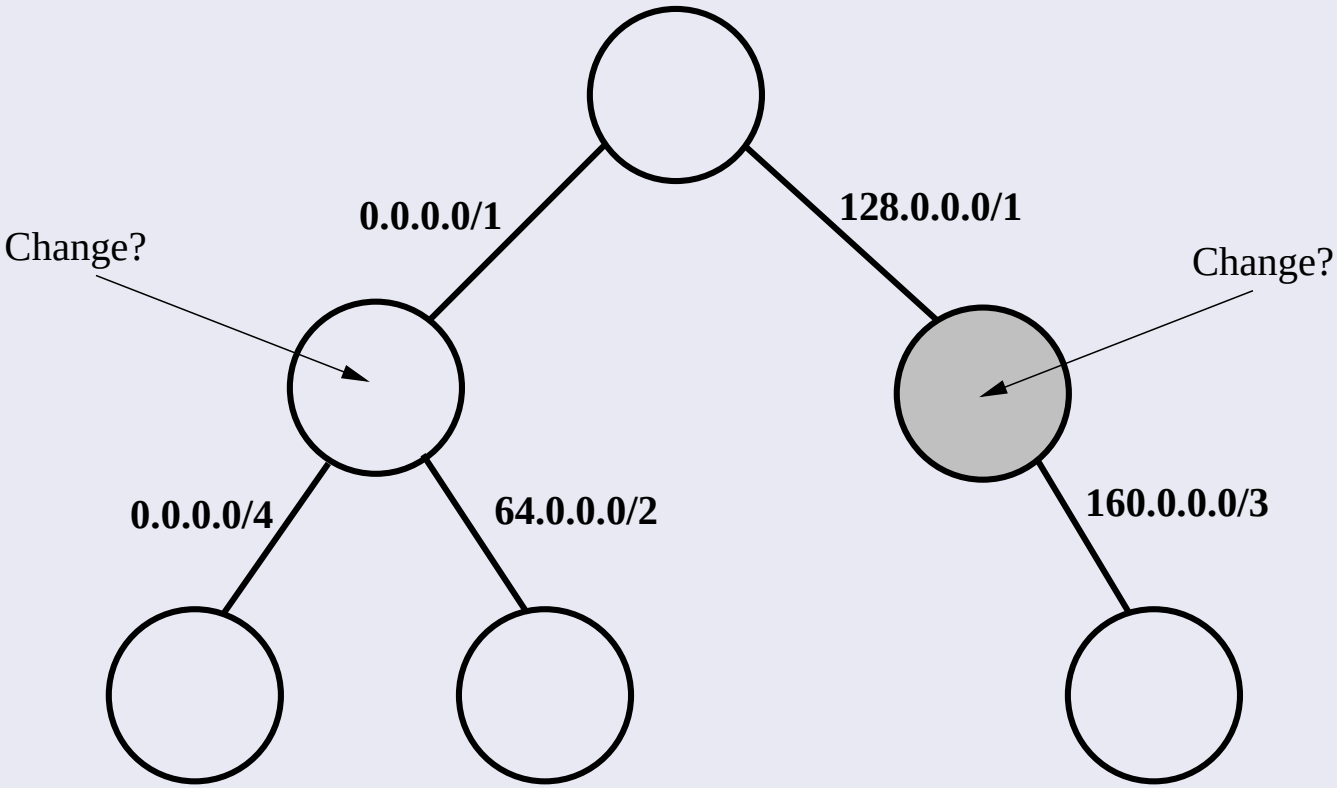
Advantages

- A natural means to penalize model ***complexity***
- A natural means to bound ***memory***
- Accommodate ***change detection***
- Allows for ***active learning***

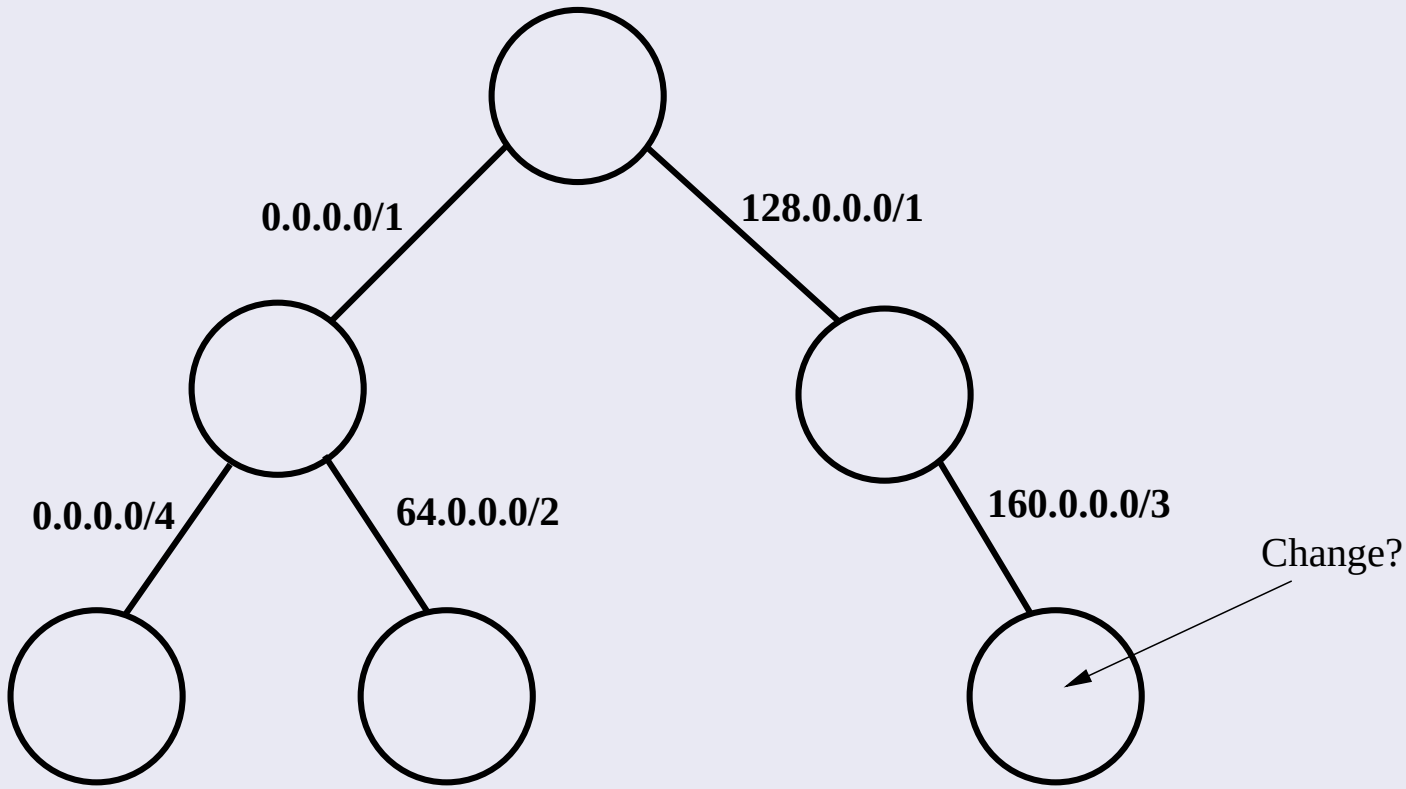
Change Detection



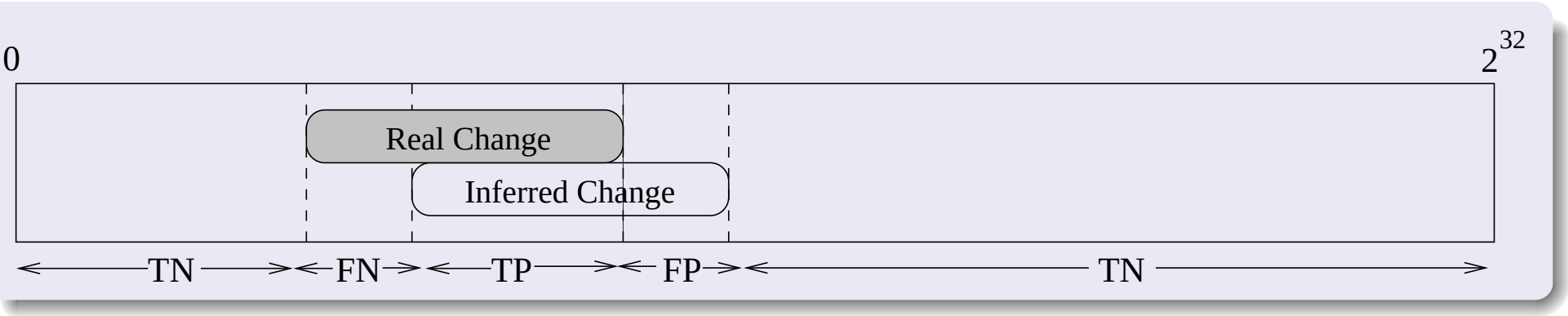
Change Detection

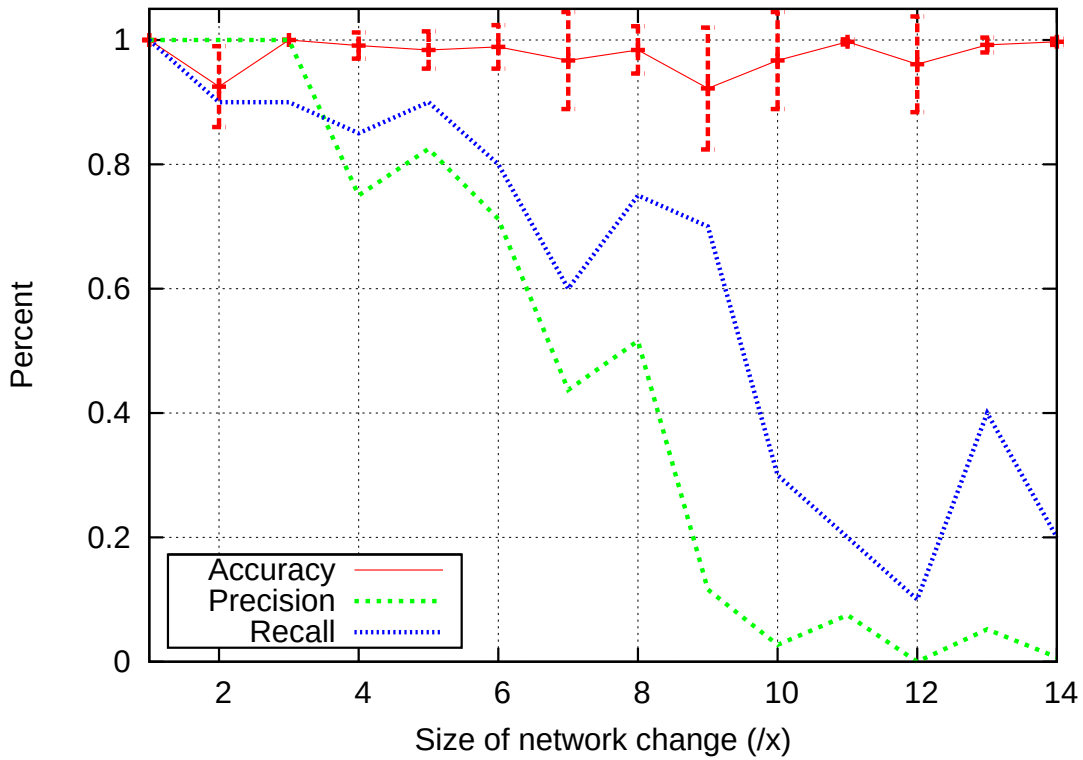


Change Detection



Change Detection Accuracy





- Induced change point game
- Accuracy high across size of change
- Large changes detected very well
- More samples required to perform well for smaller changes

Further Research

- Improving algorithm:
  - Agglomerative version has appealing properties
  - Address stability of optimal split in sequential t-test with a random forest algorithm
- Variability change point detection
- Better understanding tradeoff between pruning stale data and the cost of retraining
- Perform active learning on poorly performing or sparse portions of tree
- Coping with adversarial agents that disrupt learning?

Summary

- Learning useful for many problems in a complex Internet
- But, must be cognizant of difficult issues when employing learning in an Internet-context
- IP Address Clustering is one building block with wide applicability
  - Learns underlying structure
  - Leverages domain-specific knowledge
  - Detects environment dynamics
  - Provides a means to penalize model complexity and memory in a network-natural way

Thanks! Questions?

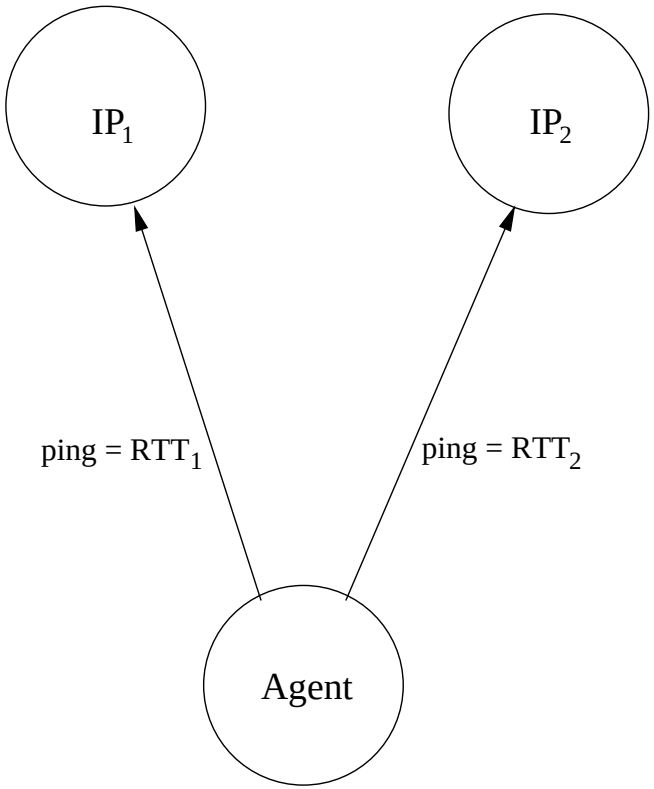
Backup Slides

IP Prefixes:

- **`prefix/mask`**
- $p/m := [p, p + 2^{b-m} - 1]$
- $b = 32$ for IPv4
- p/m has 2^{b-m} addresses

Examining the hypothesis of structure

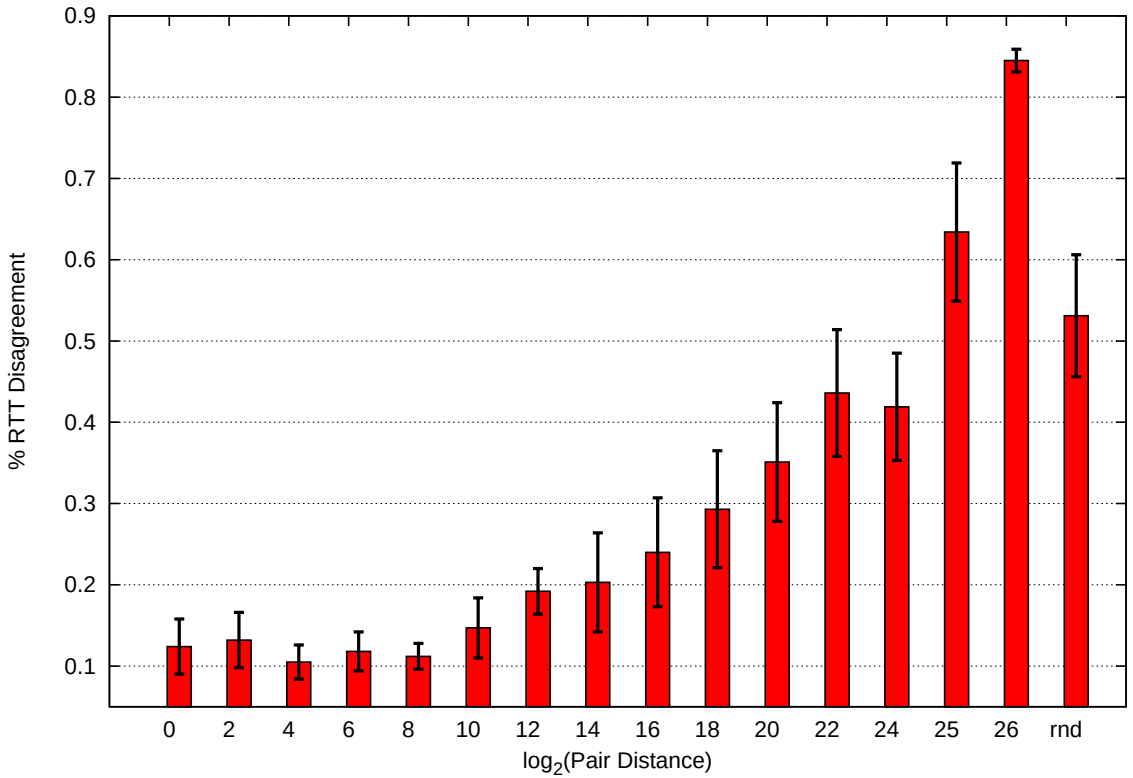
- Before trying to learn, let's sanity check ⌣̈
- $d = |IP_1 - IP_2|$, numerical "distance"
- For a random pair of d-distant IPs, how well do their RTTs agree given d?



i.e. $Pr\left(RTT_1 = \epsilon RTT_2 | d\right)$?

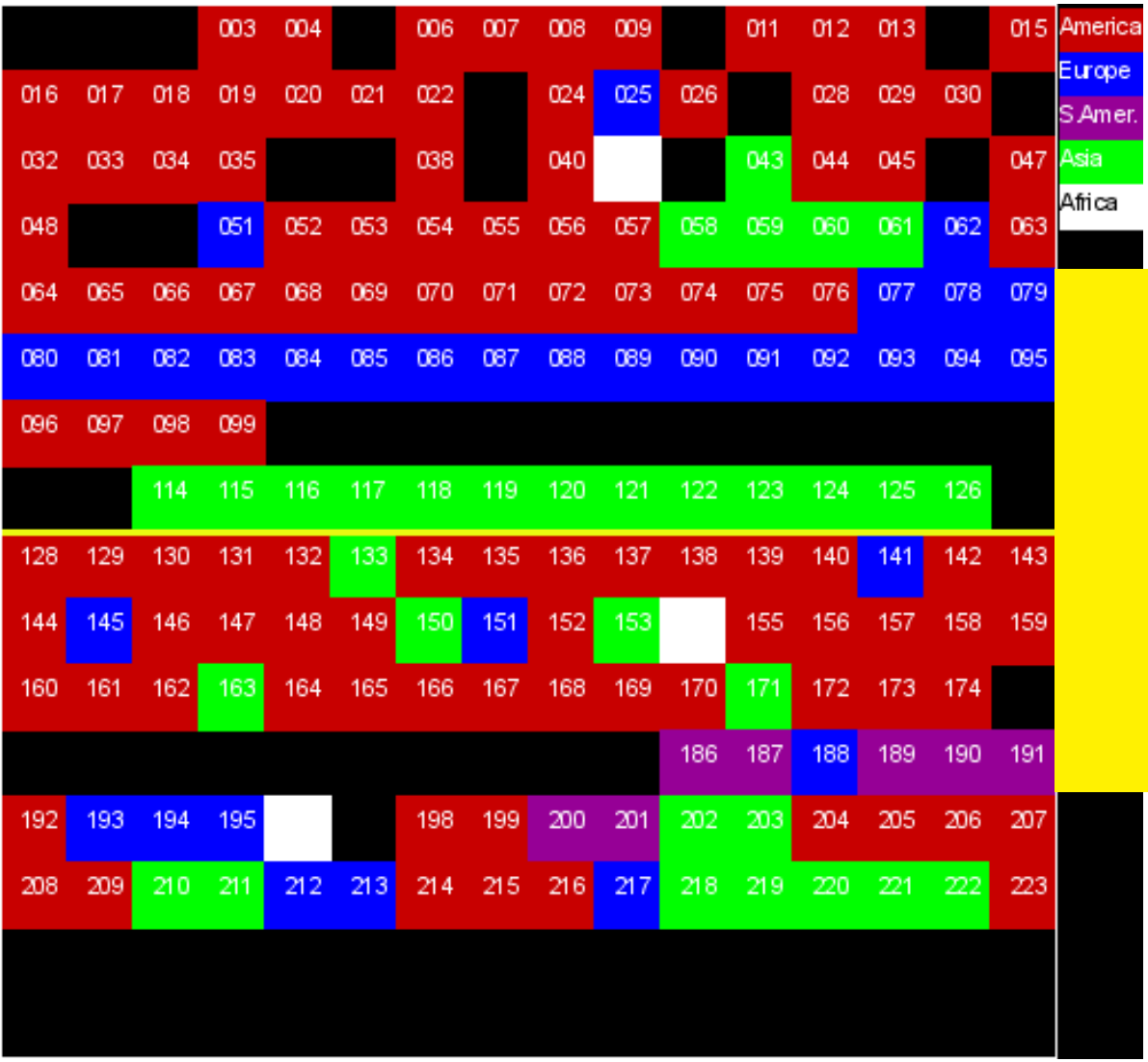
Examining the hypothesis of structure

- $d = |IP_1 - IP_2|$, numerical "distance"
- Probability that the RTT of a pair of $log_2(d)$-distant IP address disagrees?



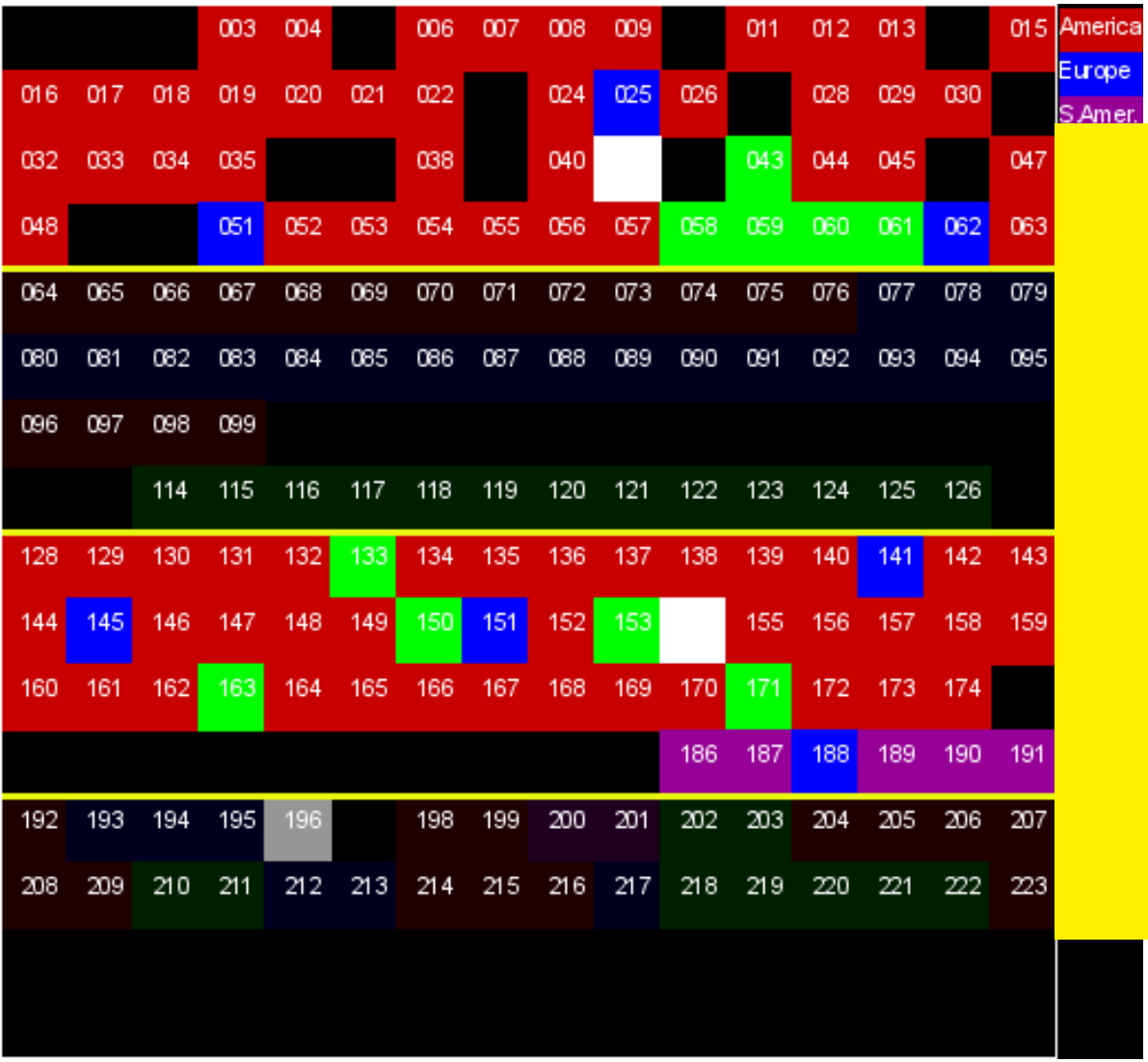
Take away: structure present to learn upon

Feature Selection



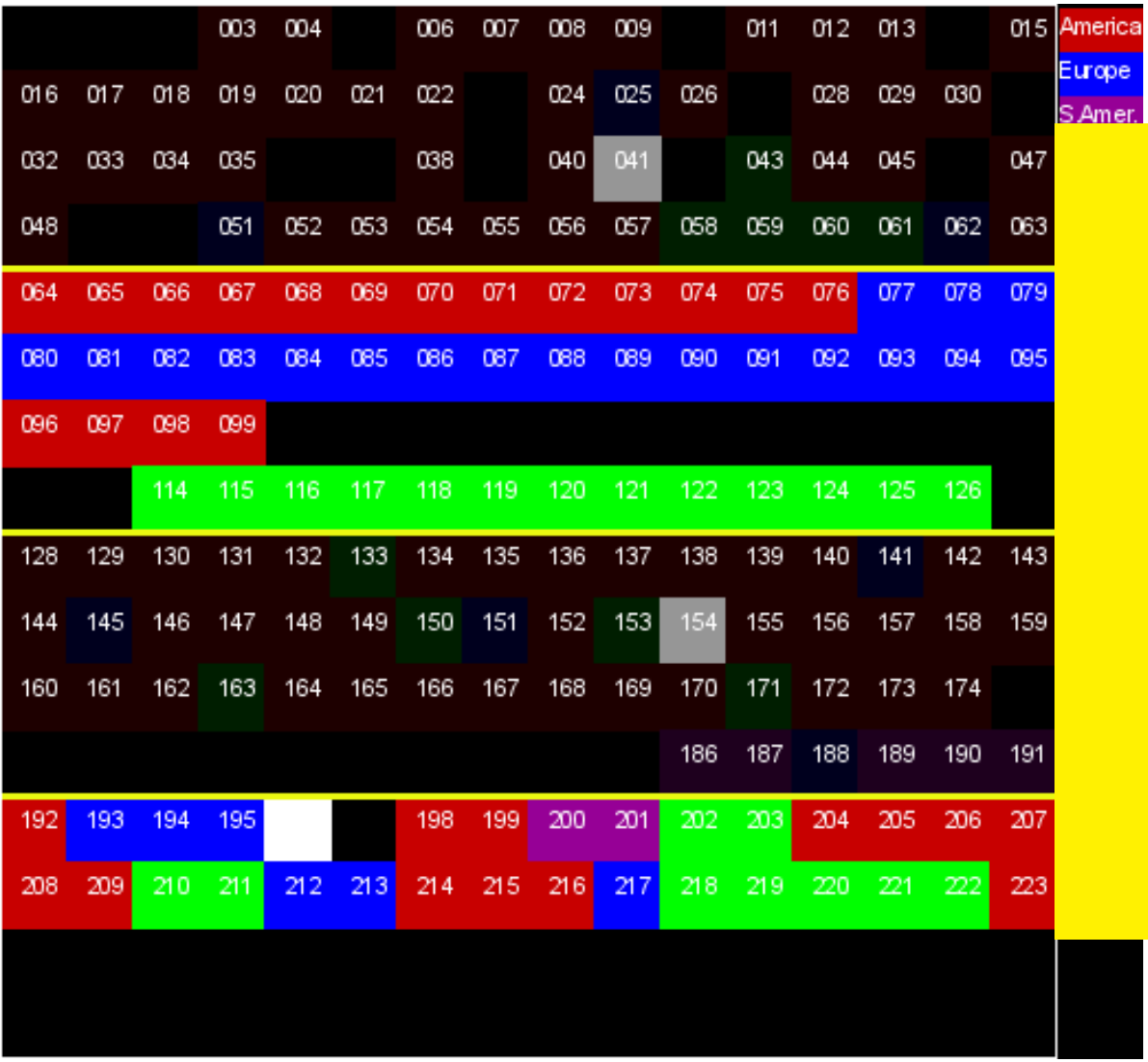
b	Range	US	Euro
1	0-127	56	21
	128-255	51	10

Feature Selection



b	Range	US	Euro
1	0-127	56	21
	128-255	51	10
2	0-63 128-191	77	7
	64-127 192-255	30	24

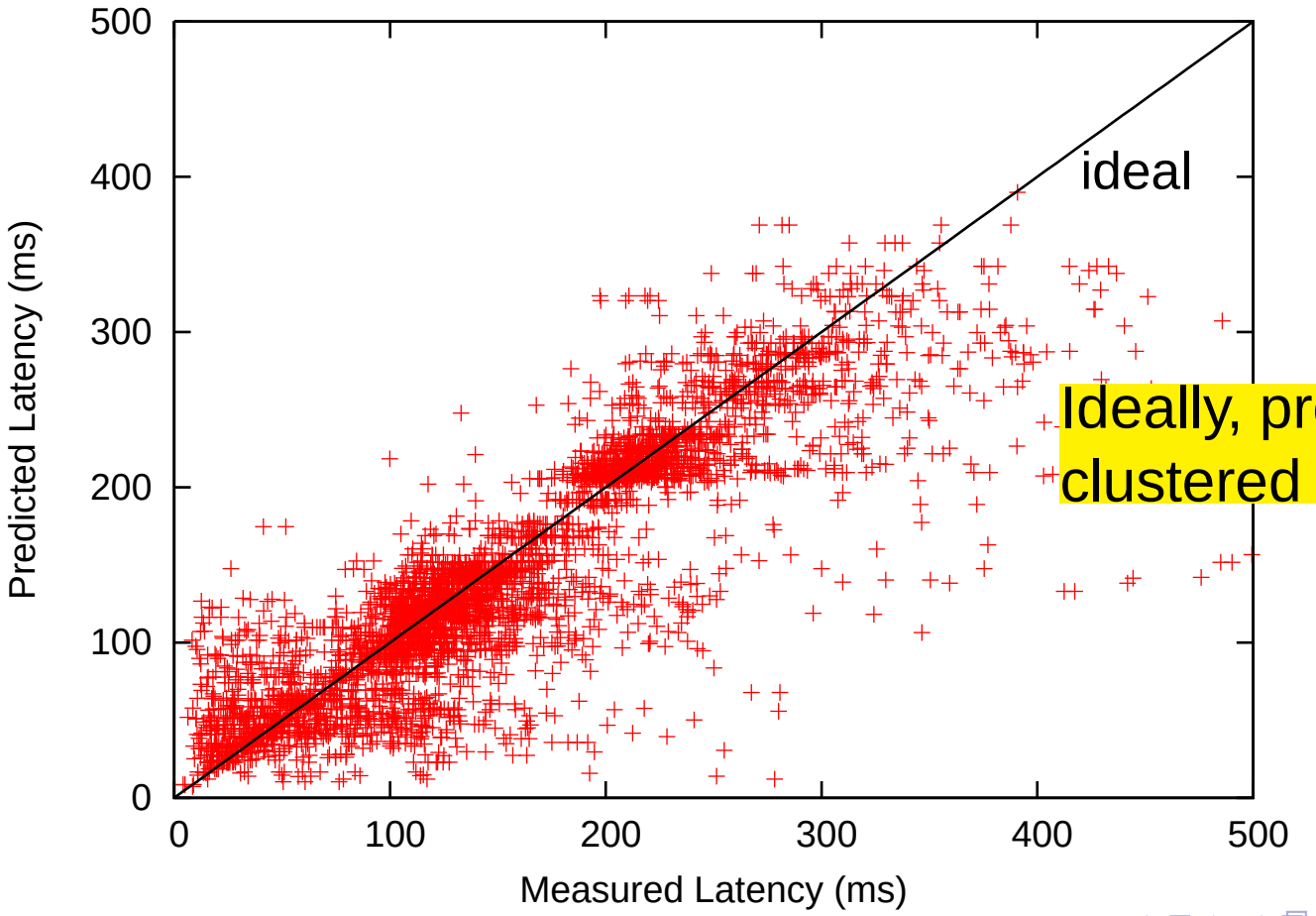
Feature Selection



b	Range	US	Euro
1	0-127	56	21
	128-255	51	10
2	0-63 128-191	77	7
	64-127 192-255	30	24

Performance

- Using Support Vector Regression
- Predict latency to unknown destinations



Ideally, predictions clustered on 45^o line

CSAIL

Coping with Network Dynamics

Comparative Results:

- Outperforms SVR approach
- Does not require SVR parametric “tuning”
- Linear lookup time in number IP address bits; fast in practice