High-Speed Detection of Unsolicited Bulk Email

Sheng-Ya Lin, Cheng-Chung Tan, Jyh-Charn (Steve) Liu,
Computer Science Department, Texas A&M University

Michael Oehler
National Security Agency

Dec, 4, 2007

Outline

- Motivation
- Progressive Email Classifier (PEC) system architecture
- Experimental results

Email Spamming No Longer Just a Nuisance

- Some Facts:
  - Botnet farms can hit any target (> 10^6)
  - bandwidth waste (3:1 or higher)
  - Network resource exploit & information stealing (malware planting)
  - Highly effective hit and run strategy (BGP, DNS, domain name, credit card fraud)
- Existing anti-spamming ware
  - Large number of software copies and signatures to maintain
  - Comprehensive detection rules, but slow to respond
- Signatures management a major bottleneck
  - Acquisition and the deployment of signatures to numerous machines
  - A small variation in the known signatures can easily defeat a signature based filter
  - Spammers can test their designs with anti-spamming ware before starting the (hit and run) campaign

Spamming Behavior at a Glance

- Spammers do not have full freedom in launching spamming.
  - Follow the transport protocols to deliver messages
  - Messages must be perceivable and appealing to human users
  - Expensive to compose and personalize spamming messages:
    - interactive (click my URL links) or *passive*
- Low yield rate combined with greed lead to high spamming volumes
- Cheap to launch spamming: millions of zombie machines each send a few copies
  - Any “hit back, interactive” method could cause severe harm to the innocents
- Summary
  - Very difficult for spammers to achieve financial goals without leaving noticeable signatures, i.e. *feature instances*
  - A challenge is how to keep up with their **speed**, **volume**, and **diversity**

Our Approach

- Lossy detection:
  - focused mainly on the major offenders
  - Avoid false positive
- Timely acquisition of *instances* of selected *features*:
  - Position the detector at the Network Access Points (NAP)
    - Highest concentration of samples for an enterprise network
    - Detect them before the flood already enters the network
- Work on the algorithm & data structure level, rather than any hardware platform
  - Broad spectrum of computing resources/constraints
- Regular emails are expected to have random distributions of strings that happen to fall into the spamming feature space
  - Moderated delivery of bulk, legitimate email
- A spamming stream: *Invariant* and *variant parts*
  - An invariant that also appears in regular emails cannot be used for filtering
  - For the first cut effort: URL (over 95% spamming have them)

Competitive Aging-Scoring Scheme (CASS)

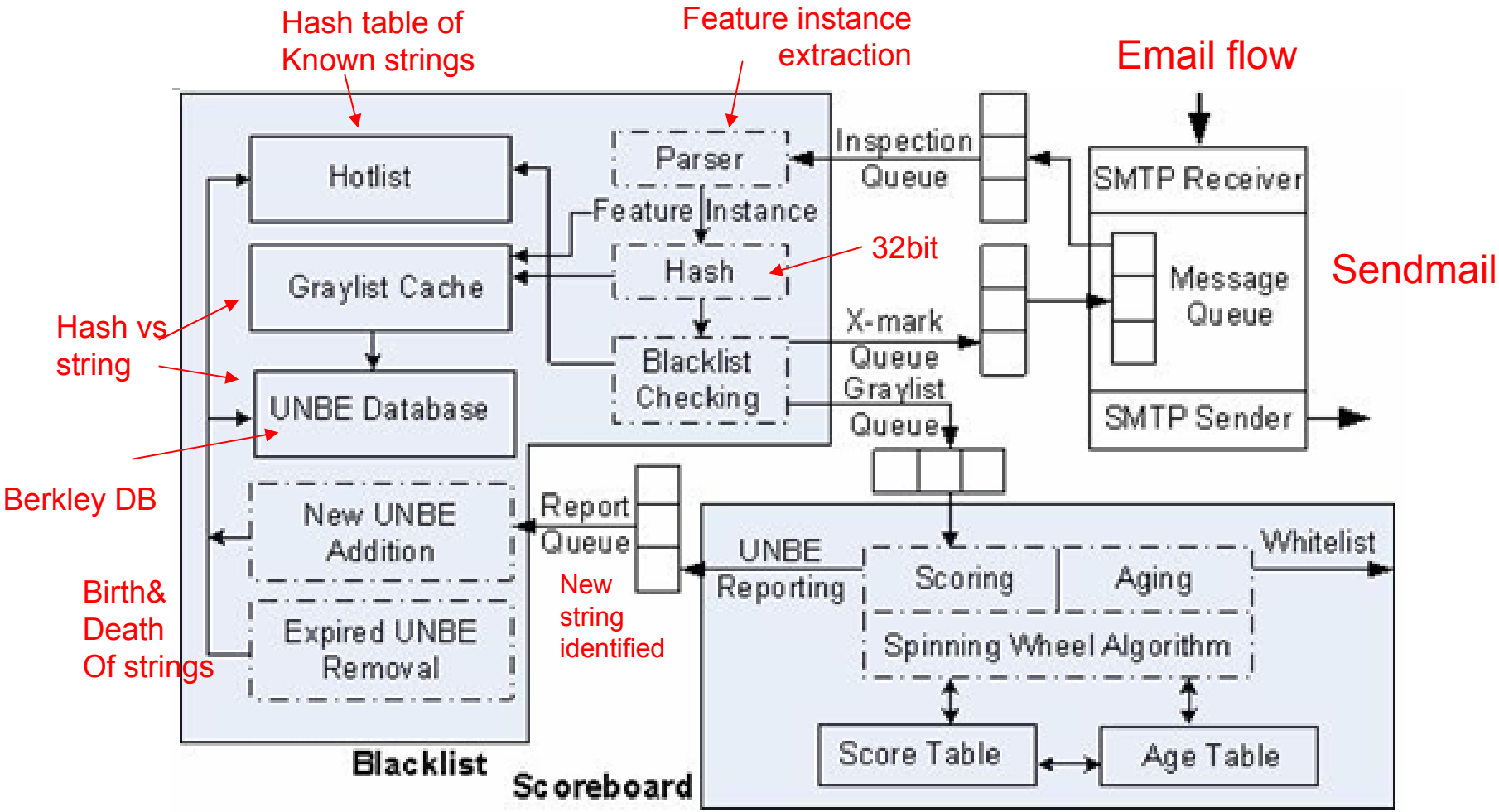
- A spamming invariant (string) is called its *feature instance (FI).* The essence of our technique:

  "Extract FIs of emails and keep track of their occurrences. If exceeding a threshold: an UNBE stream"

- In a naïve approach, it takes O(1) to update the score of an FI, but O(N) to age all other FIs
  - A major computing cost

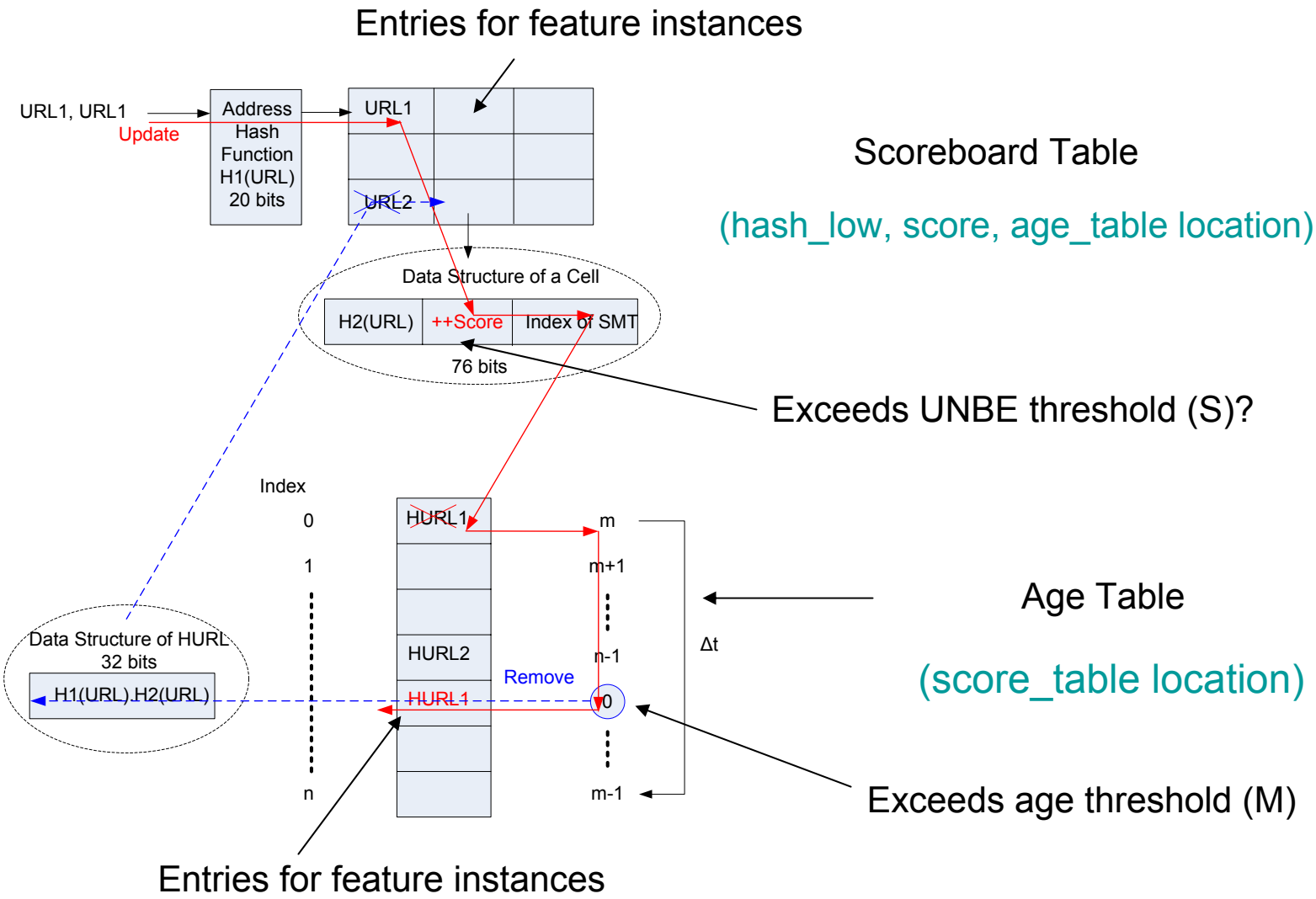
- CASS: a constant time algorithm
  - The time-to-live of an FI is reset each time when its score is increased by one (when a new copy arrives)
  - The time-to-live of all other FIs is reduced by one
  - New complexity: O(1) for both scoring and aging
  - Exceeding a threshold: **move it to the blacklist**
  - No further copies in a time-out period: **discard it**
    - It may not be a fixed physical time

PEC Architecture

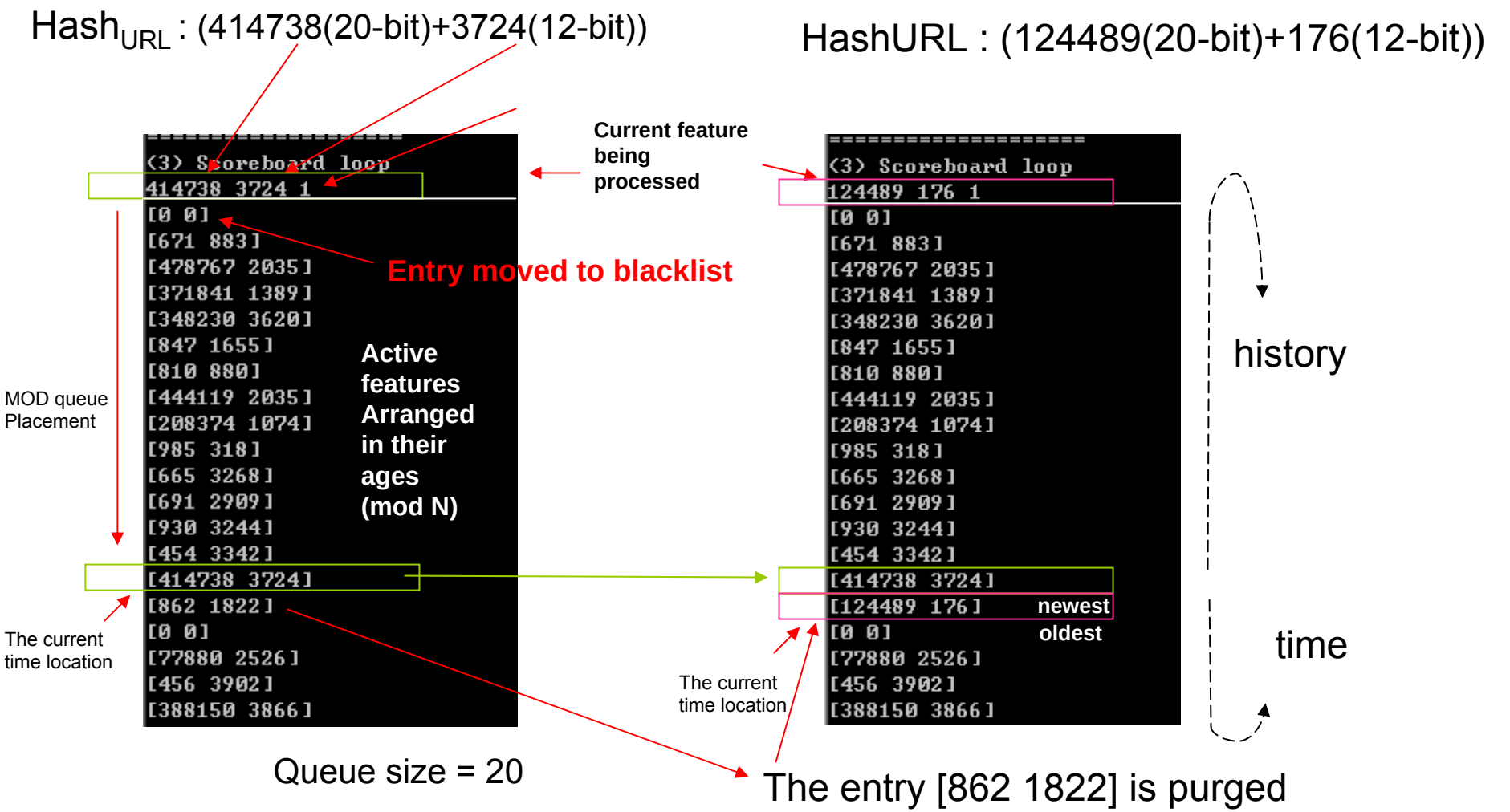


Aging and scoring of unknown strings

Data Structure of Scoreboard



A Snapshot



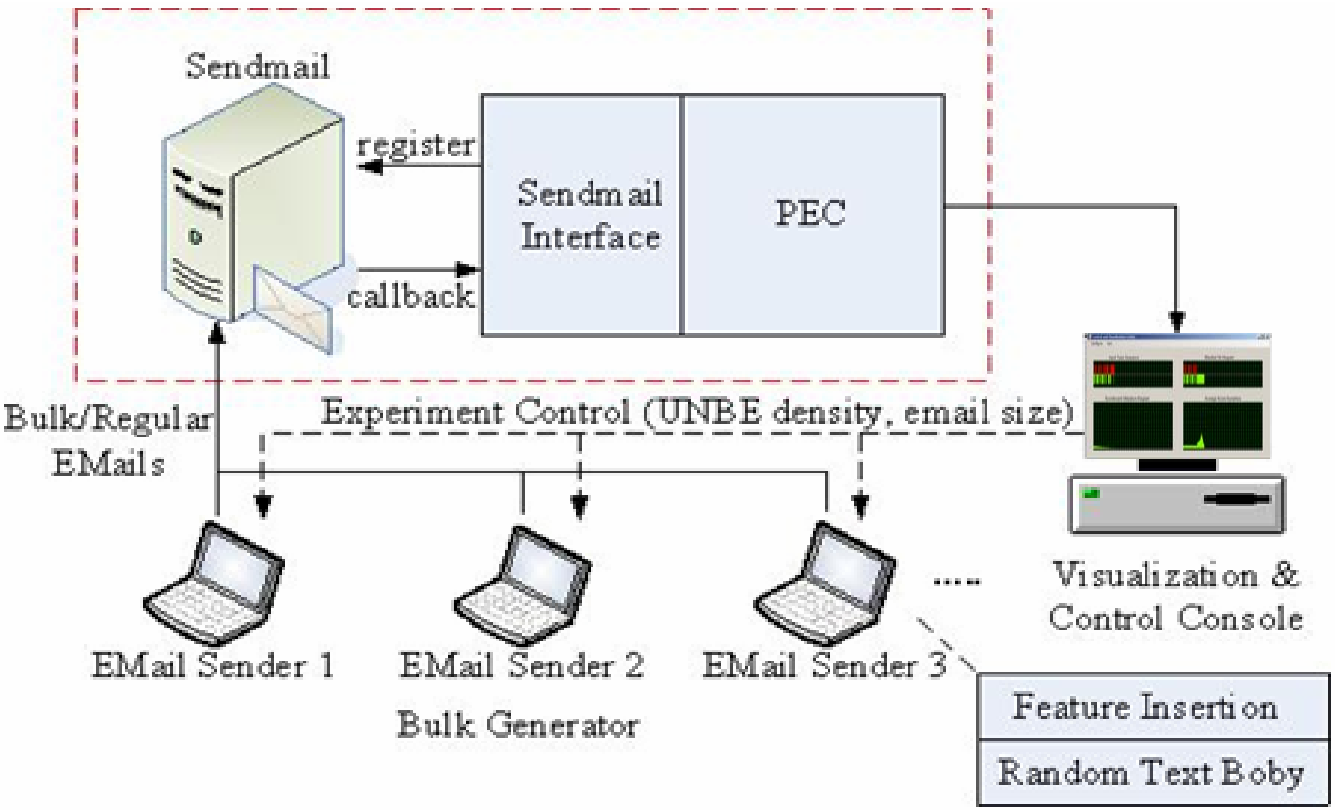
S =10, M =20

Next feature instance

Testbed Environment

Three Modules included:

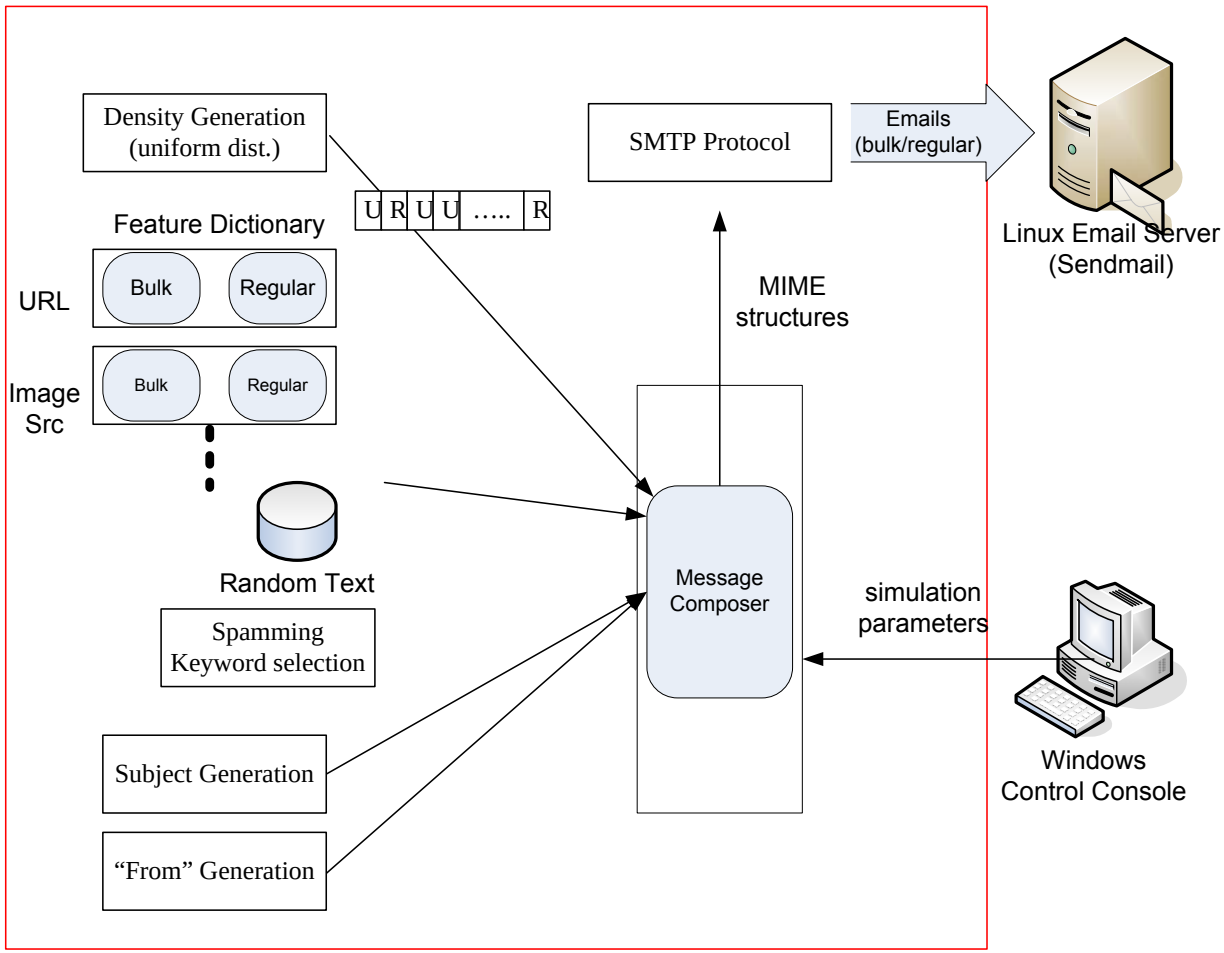
1. Email generation
2. PEC (Blacklist and scoreboard):
3. Control and visualization console



Experimental Configuration

- Email generator: Intel P4-3.0 Windows XP
- Email Server: Xeon 3.0GHz, two single core CPUs, Linux, Sendmail 8.14.1
- Within a batch, the sender sends 2000 copies of emails (uniformly mixed UNBEs and regulars).
  - S: 50
  - M: 2048
  - The average mail size: 1.5K bytes
  - One mail per 0.088 seconds on average.

Workflow of Email Generation



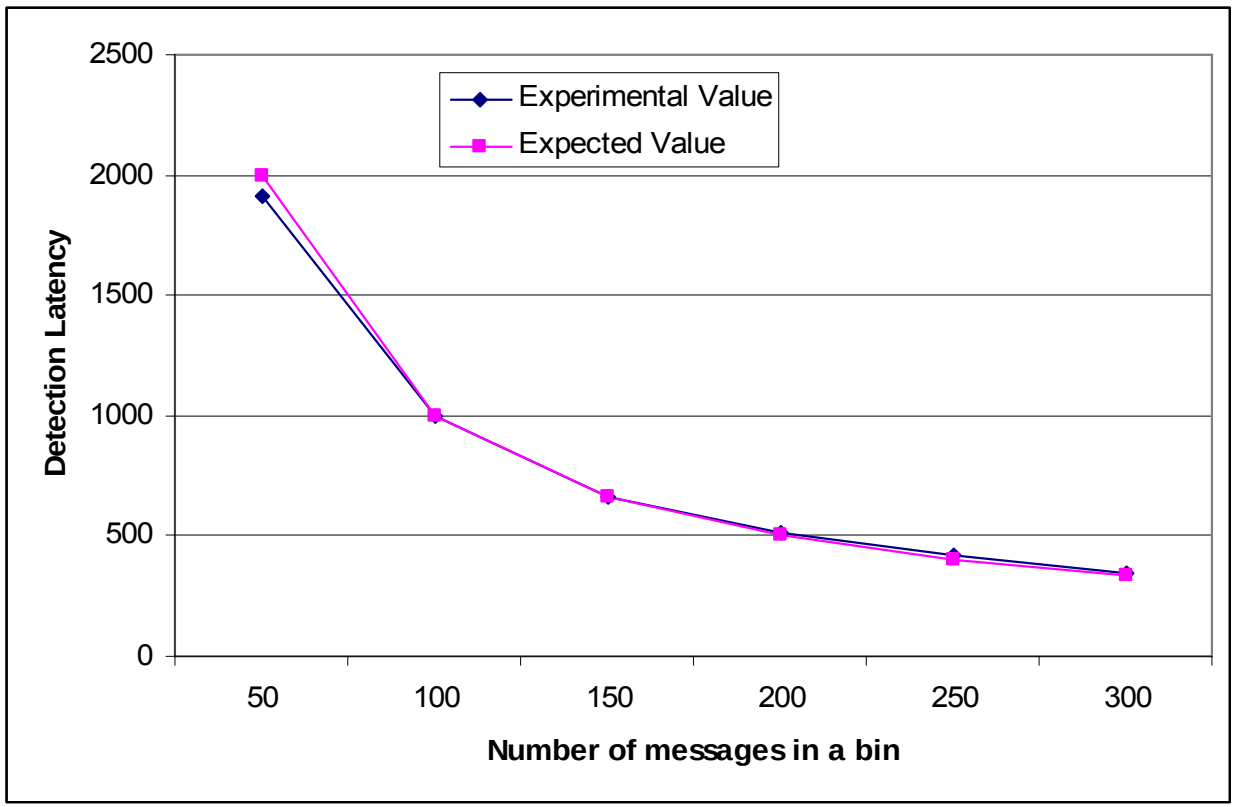
UNBE Generation

- Both UNBE and regular copies are injected with URL links or remote image sources
  - Can adjust density, locations of variants and invariants in the body of each copy to generate MIME messages.
  - UNBE features extracted from 2005 TREC Public Spam Corpus, http://plg.uwaterloo.ca/~gvcormac/treccorpus/about.html
  - Variants: random text taken from web sites
  - Keywords: User defined (not tested in this report)
- The message composer calls an SMTP library to send the generated emails to Sendmail

Detection Latency of Single UNBE source

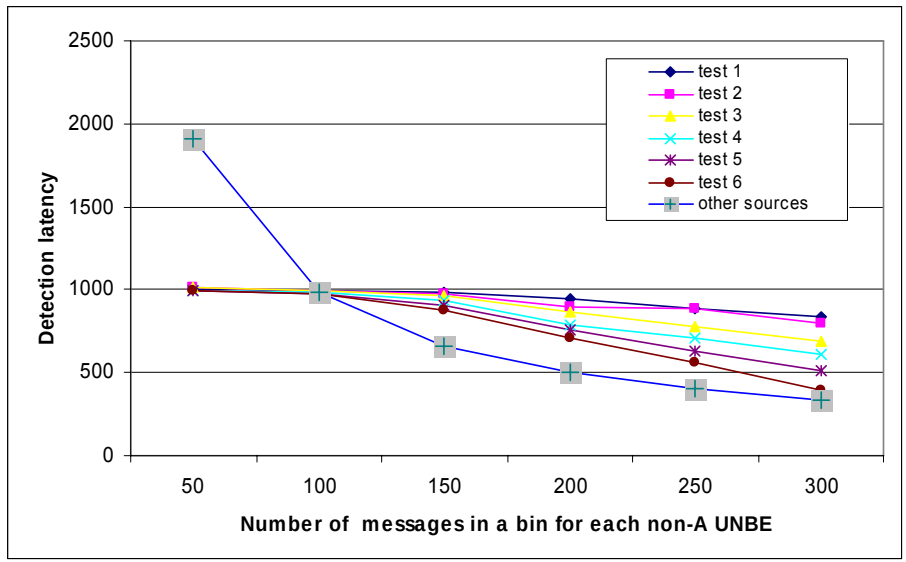
- Fix threshold and age table length under different densities.
- Test six different UNBE densities (50, 100, 150, 200 …, 300 UNBE messages/bin)

Unit: Virtual clock

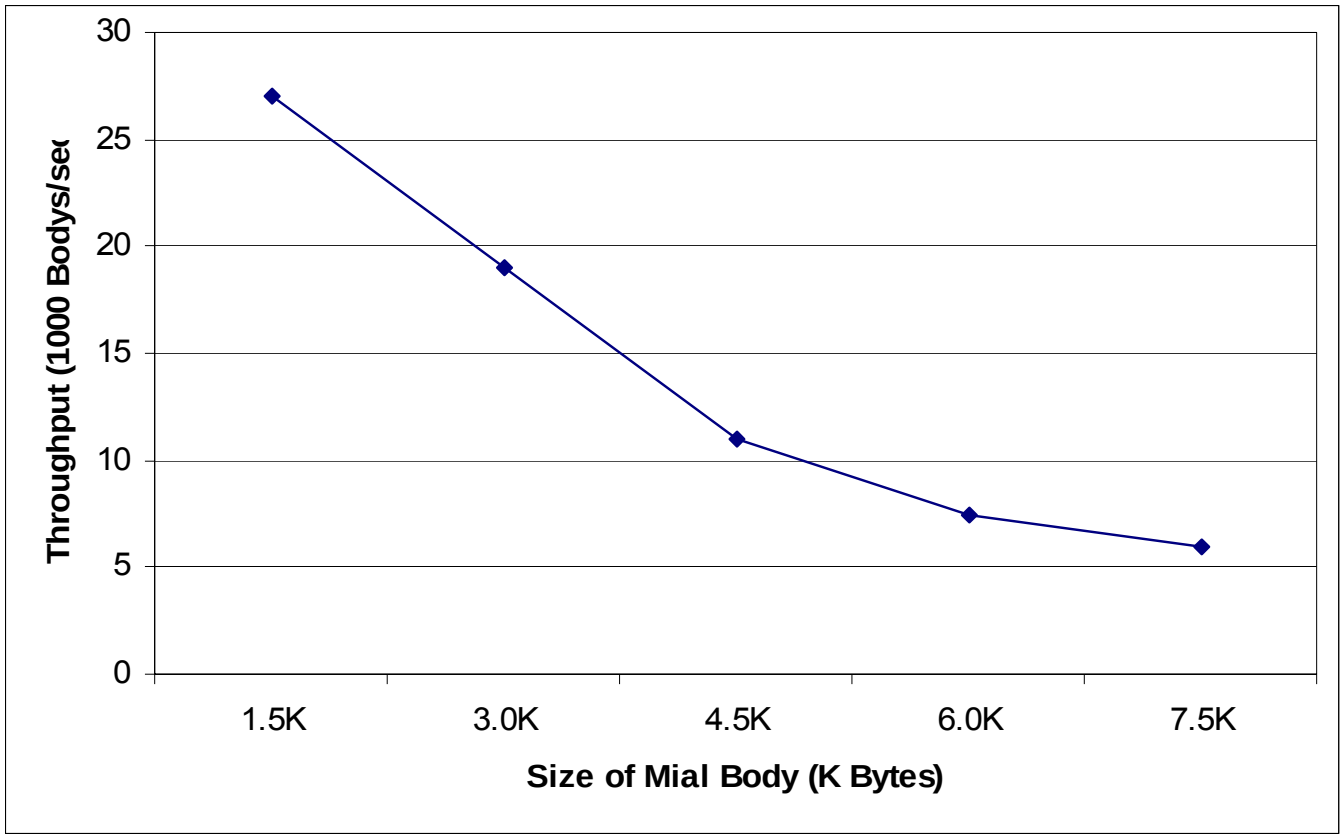


Effects of Multiple UNBE Sources

- Given an UNBE source *A*, six tests were made where one addition UNBE source is added to the experiment at a time.
  - The six lines marked as test[1-6]
- The density (instances/batch)
  - A: is fixed at 100
  - Other UNBE sources: increased from 50 to 300
- Result:
  - The detection latency of an UNBE decreases with the number of UNBE sources
    - When a source is captured, it is blocked form the scorebaord. The density measure in VC for others increases



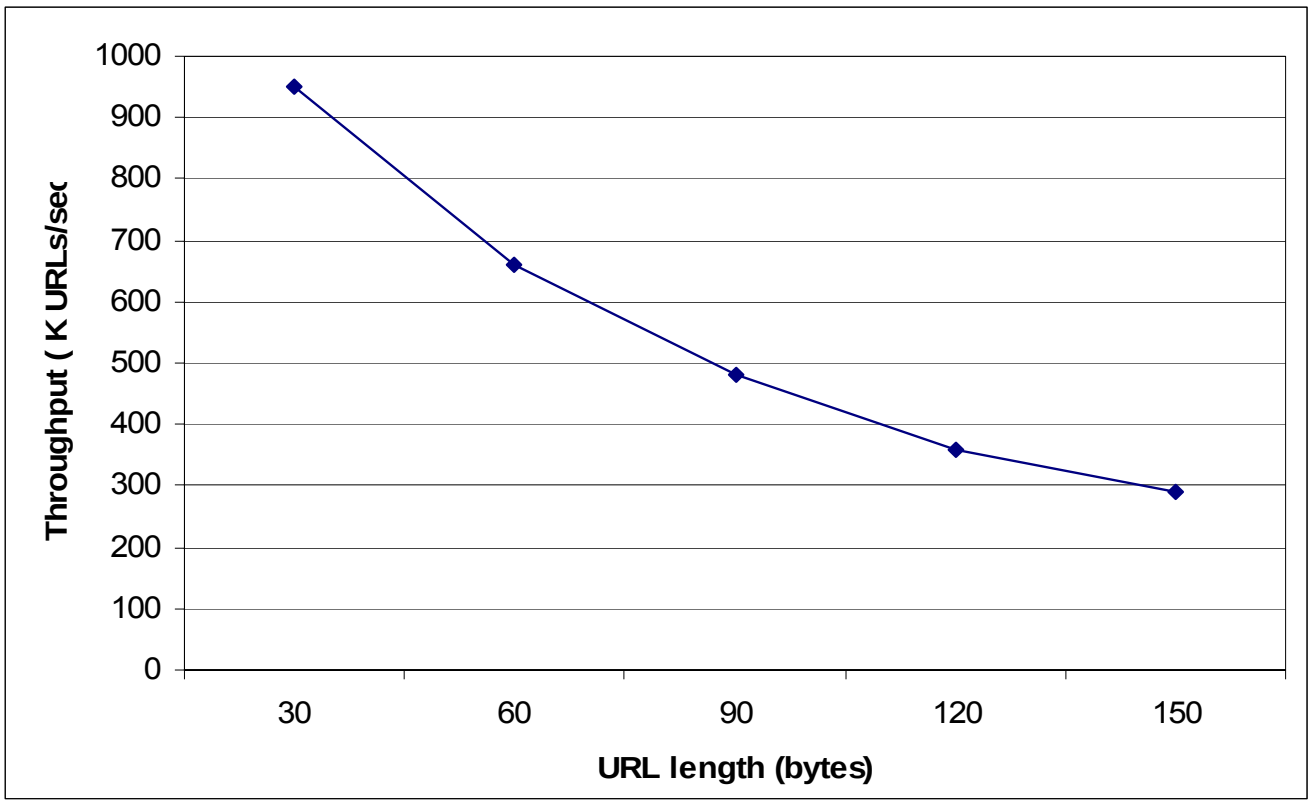
Throughput of URL Parser



The average Email size is from 1.5 KB to 7.5 KB, and each email has 2 URLs.

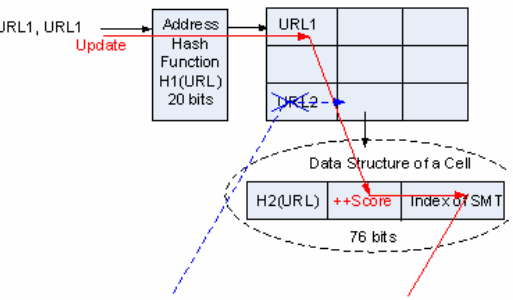
Throughput of Scoreboard and Blacklist

- Scoreboard: 1.2M transactions
- Blacklist: 0.9M (avg. 30 B) URLs, without including database access



Pointer Table: reduce memory need (at a small cost of delay)

- In the detection window, a limited number of hashed values need to be tracked
- Full table for 32-bit hash system takes too much space
- Higher order bits used as the index, and the rest, and the rest bits maintained by a linked list (for each entry)
  - If pointer table uses 20 bits for indexing, that means it has 1M entries, and age table length is 20K~70K, the maximum depth of linked list pointed by pointer table is 2.



(bit length) / (depth)	2	3	4	5	6	7
20	20k	70k	167k	297k	582k	657k
22	45k	165k	337k			
24	145k	515k				
26	375k					
28	520k					
30	650k					

Threshold Setting

- **Q1**: “What is the minimum value of *M* to detect an UNBE attack (of known density) with a success probability of higher than α?”
  - (smaller M retire sooner)
- **Q2**: “For a given *M,* what is the maximum value of *S* to guarantee that the probability of the detection latency < ς is greater than α*?”*
  - (large S less likely false positive, but more enter network before detection)

Compute M and S

$$\lambda = \mu_f /(\mu_b + \mu_f)$$

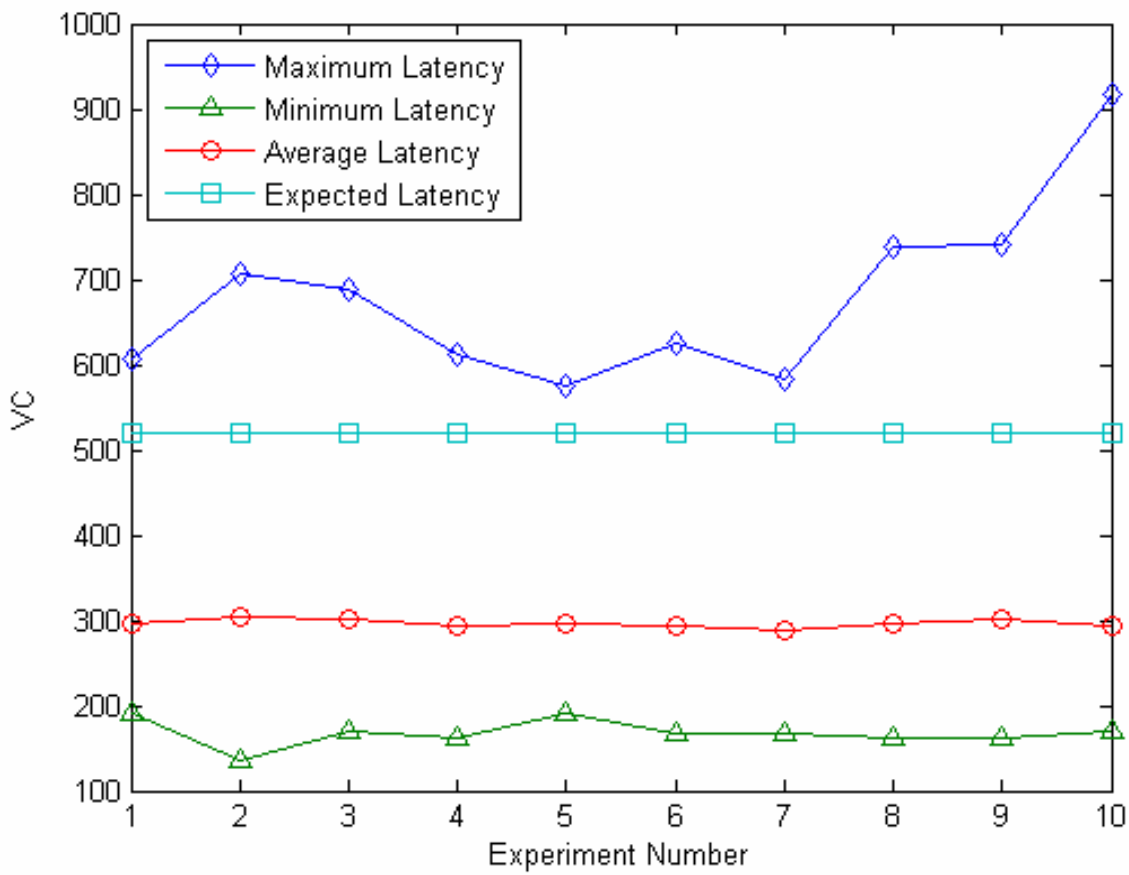
$$\Gamma(2, \lambda M) = 1 - \alpha \quad \longleftarrow \text{Get M}$$

$$\varsigma = E[H_{S+1}]/\lambda$$

$$E(H_{S+1}) = 1/\alpha + 1/\alpha^2 + \ldots + 1/\alpha^{(S-1)} + 2/\alpha^S$$

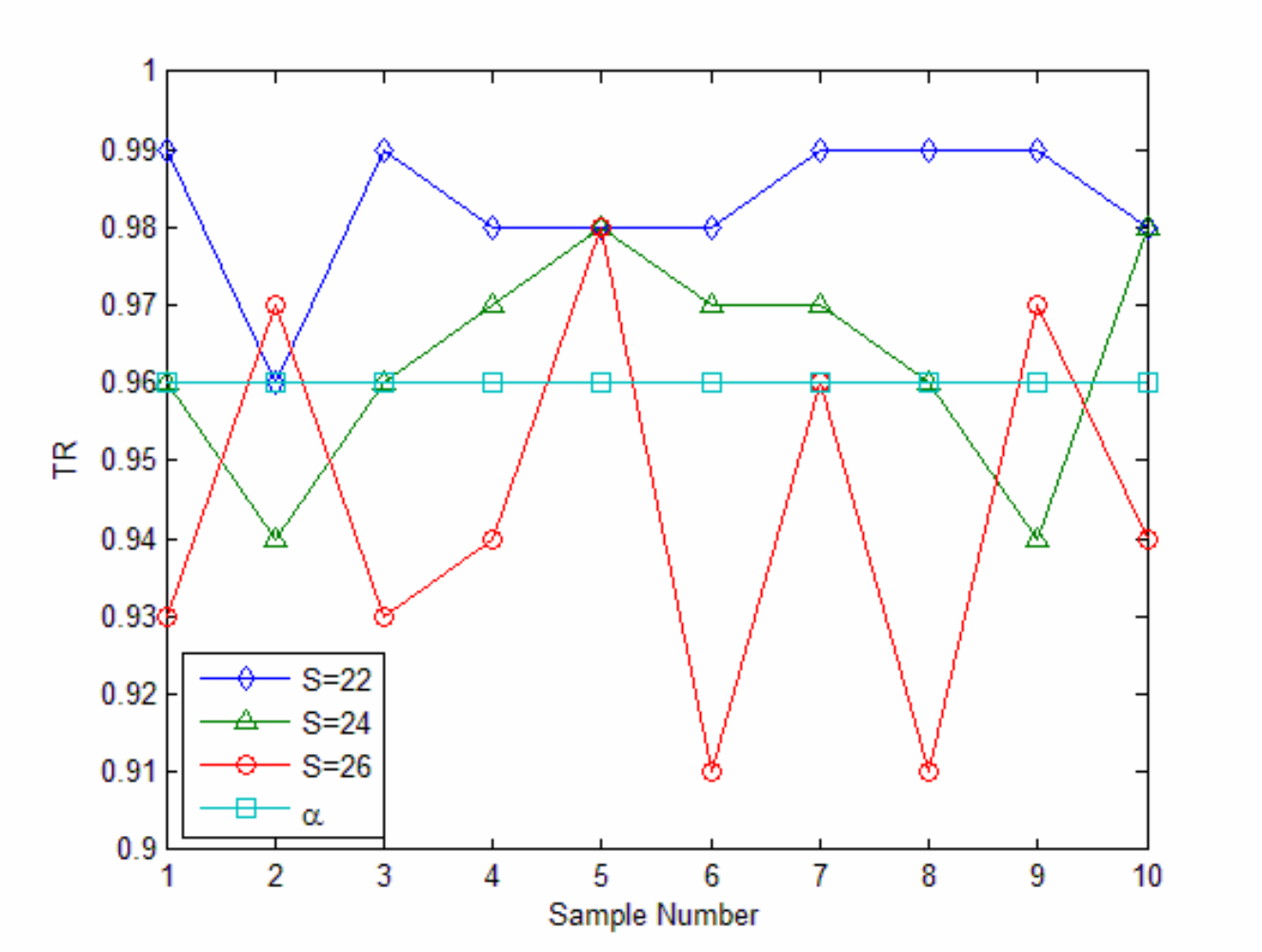
$$= (1 - 2\alpha^{-S} + \alpha^{-S+1})/(\alpha - 1) \quad \longleftarrow \text{Get S}$$

Detection latency when *S*=24, *M*=55

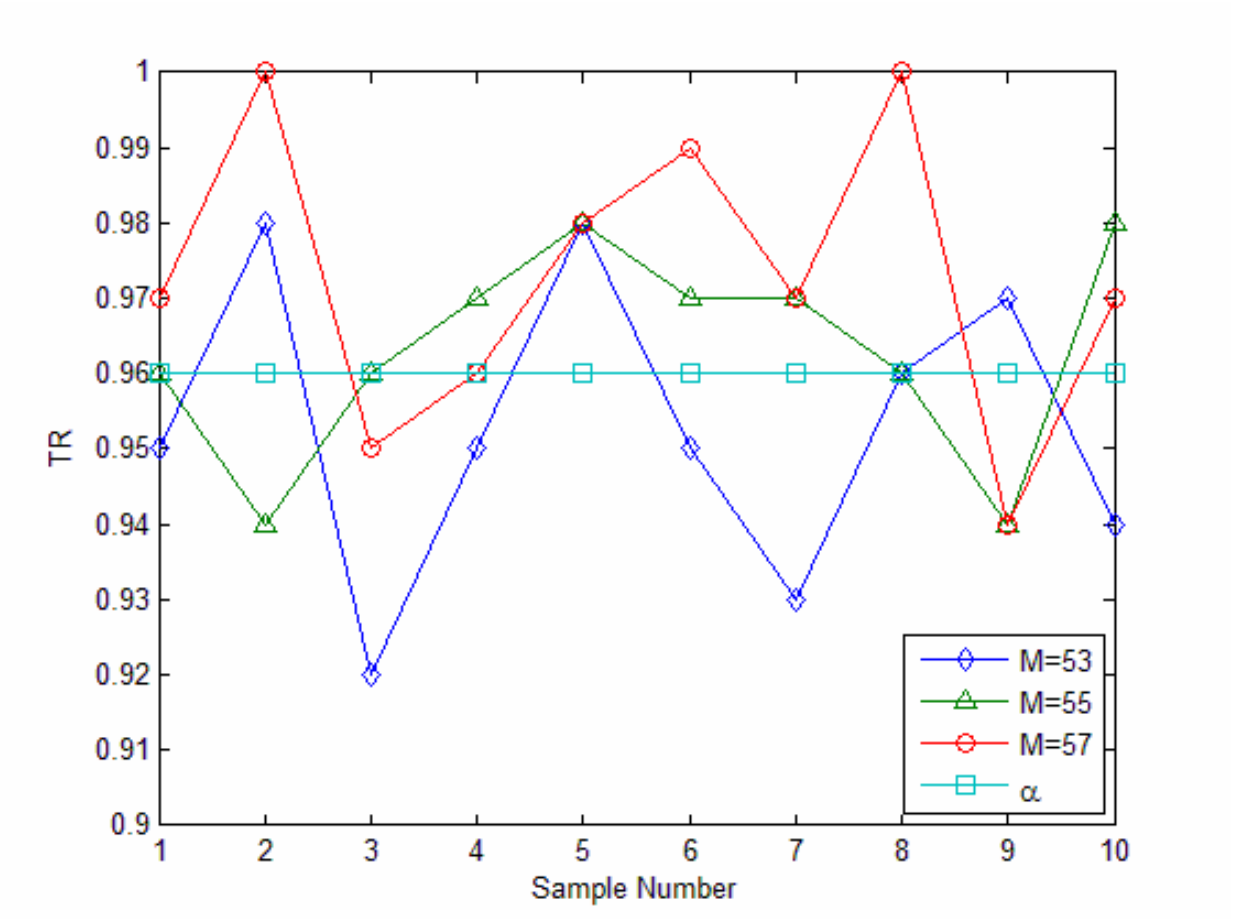


The prediction model is conservative

Sensitivity of TR vs. *S*



Sensitivity of TR vs. *M* (*S*=24)



Summary

- PEC demonstrates the feasibility of high speed UNBE filtering at the network vantage points
- The method is not meant to replace existing solutions, but to defeat major offenders
  - 80-20 rule
- Expansion of the techniques to handle multiple features (bad words, dirty subnets, black lists, etc)
  - Integration/interface with existing tools

Thank You!