Streaming Algorithms for Robust, Real-Time Detection of DDoS Attacks

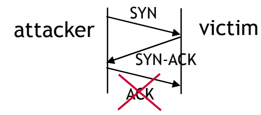
S. Ganguly, M. Garofalakis, R. Rastogi, K. Sabnani

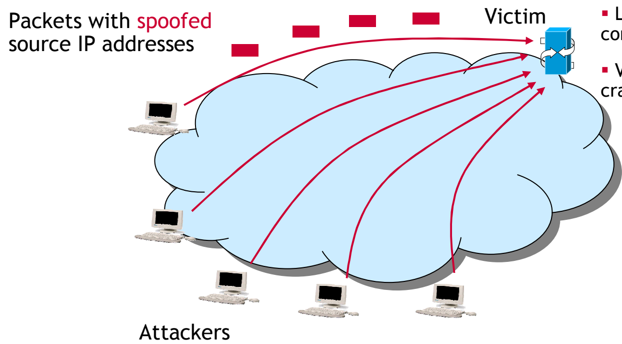
Krishan Sabnani
Bell Labs Research

Agenda

- TCP-SYN-flooding attack detection problem
- Distinct samples
- Distinct-Count sketches
- Experimental results
- Summary

TCP-SYN-flooding attack





- Large number of half-open connections
- Victim runs out of resources & crashes

TCP-SYN-flooding attack: Salient characteristics

TCP-SYN-flooding attacks are different from flash crowds

	TCP-SYN-floods	Flash crowds
Traffic volume	low	high
# of half-open connections	high	low
# of distinct connecting sources	high	high

→ Tracking top-k destinations with the highest traffic volume to detect attack victims won't work

- Attack traffic may not be high

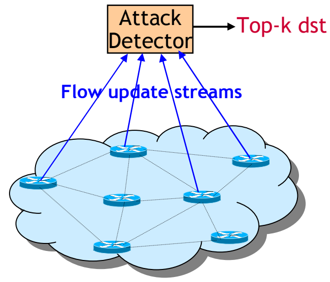
→ Right metric for robust attack detection:

Top-k destinations wrt number of distinct sources with half-open connections

System model

Continuous stream of (src IP, dst IP, ±1) flow updates

- +1 for SYN packet from src to dst (insert)
- -1 for ACK packet from src to dst (delete)



Assumptions

- 32-bit IP addresses; 64-bit (src, dst) pairs
- Number of distinct (src, dst) pairs: U

Constraints

- Single pass over update stream
- Small space (logarithmic in U)
  - → Solutions that store state for U (src, dst) pairs won't work
- Small processing time per update
- Continuous tracking of attack metric (top-k destinations)

Problem formulation

Distinct frequency f_v for dst v = number of distinct src's with unacknowledged SYN pkts (half-open connections) to v

$$f_v = |\{u : (\sum_{(u,v,\Delta)} \Delta) > 0\}|$$

Key observation: Attack victims will have high f_v values

→ To detect attack, track top-k f_v frequency values (f_{v1},…., f_{vk})

- Exact tracking of f_v values requires Θ(U) space, and is thus impractical

Approximate top-k dst tracking problem: Track top-k frequencies with a small (ε) relative error; if $\hat{f}_v$ is the estimate for top-k frequency value f_v then

$$|\hat{f}_v - f_v| \le \varepsilon f_v$$

Our contribution

Distinct-Count Sketch structure

- Enables tracking of top-k distinct frequencies with guaranteed accuracy
- Is resilient to deletes (necessary to ignore legitimate TCP connections)
- Low storage space overhead
- Low update processing time

Related work

- Estan and Varghese [SIGCOMM 02]
  - Use samples and hash-based filtering to identify large flows
  - A half-opened TCP flow is not large because no packets are exchanged
- Kompella et al. [IMC 04], Gao et al. [ICDCS 04]
  - Maintain multiple hash tables, dst that hashes into buckets with large counters in all hash tables is potential attack victim
  - No provable guarantees
- Gibbons [VLDB 01], Cormode and Muthukrishnan [PODS 05]
  - Distinct samples, cascaded summaries for (distinct) frequency estimation
  - Cannot handle deletions in update stream
- Venkataraman et al.[NDSS 05]
  - For threshold k, k-superspreaders identify src's that connect to >k dst
  - Determining threshold k may be difficult in practice

Revisiting the basics: Distinct samples [Gibbons, VLDB 01]

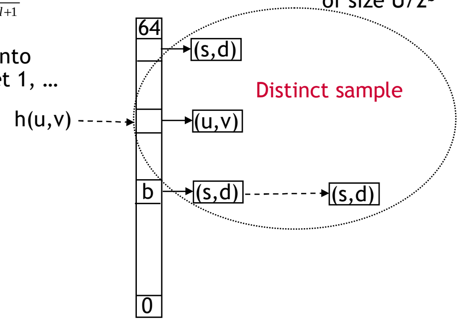
- Good for distinct frequency estimation, but cannot handle deletes
- Stream of (src, dst, +1) flow updates (inserts)

Hash function h maps (src, dst) pairs to buckets with exponentially decreasing probabilities

$$\Pr[h(u,v) = l] = \frac{1}{2^{l+1}}$$

U/2 (src, dst) pairs hash into bucket 0, U/4 into bucket 1, …

(src, dst) pairs that hash into buckets $\geq$ b yield distinct sample of size $U/2^b$



Alcatel·Lucent

Top-k frequency estimation procedure

Let $v_1, \ldots, v_k$ be dst with highest frequencies (say $f^s_{v1}, \ldots, f^s_{vk}$) in distinct sample from buckets $\geq$ b
Return $(v_1, \hat{f}_{v1} = 2^b f^s_{v1}), \ldots, (v_k, \hat{f}_{vk} = 2^b f^s_{vk})$

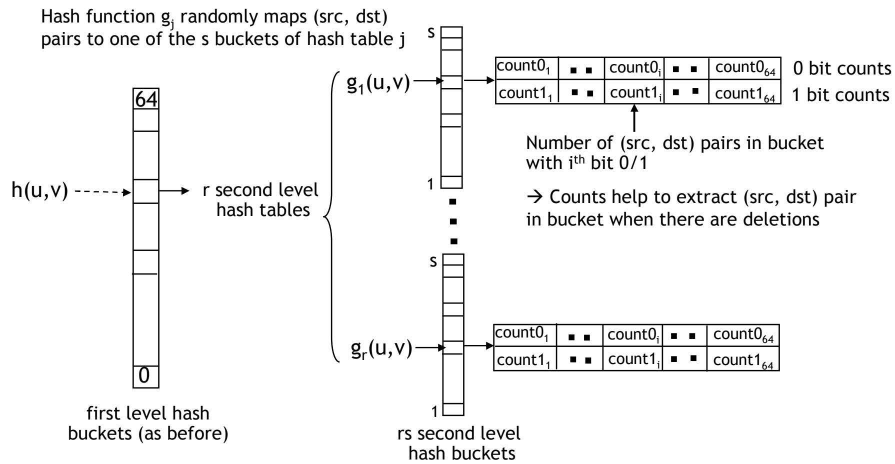
Key result: If distinct sample size $> \Theta(\frac{U \log U}{f_{v_k} \varepsilon^2})$, then for each top-k distinct frequency f_v whp

$$|\hat{f}_v - f_v| \leq \varepsilon f_v$$

Alcatel·Lucent

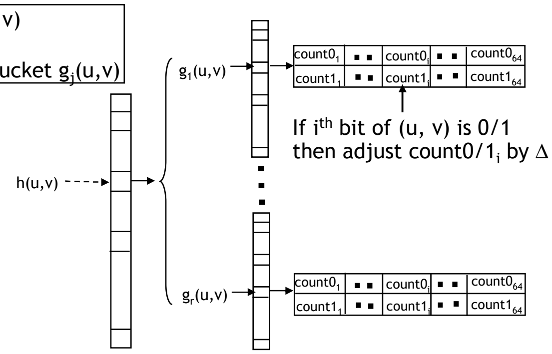
Distinct-Count Sketch

Distinct-Count Sketches produce distinct samples in the presence of deletions



Maintenance procedure for incoming stream update (u, v, Δ)

Consider first-level hash bucket h(u,v)
For j = 1 to r
    Update counts in second-level bucket g_j(u,v)



Example (updating counts):

2	0	0	2	0 bit
0	2	2	0	1 bit

(u,v) = 0110, Δ=+1 ↓

3	0	0	3	No collision
0	3	3	0	

2	0	0	2	0 bit
0	2	2	0	1 bit

(u,v) = 1010, Δ=+1 ↓

2	1	0	3	Collision
1	2	3	0	

Extracting (src, dst) pair from second-level bucket

If for all i=1 to 64, exactly one of $count0_i$ or $count1_i$ is non-zero /* no collision */
Then
  (src, dst) = sequence of bit values with non-zero counts
  Return (src, dst)
Else /* collision */
  Return "empty (src, dst)"

Example (extracting (src, dst) pair):

3	0	0	3	0 bit
0	3	3	0	1 bit

↓ no collision

(src, dst) = 0110

2	1	0	3	0 bit
1	2	3	0	1 bit

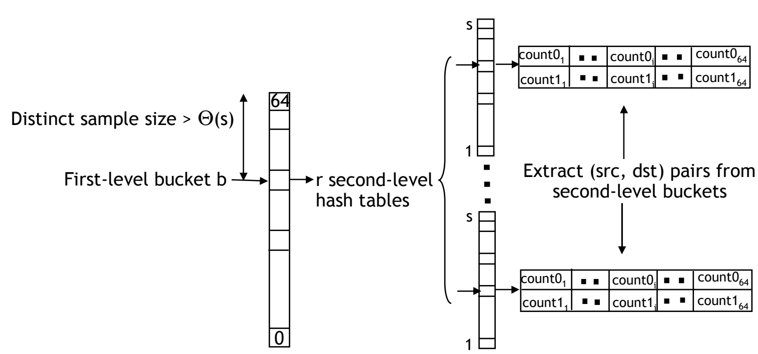
↓ collision

empty (src, dst)

Alcatel·Lucent

Top-k frequency estimation procedure

Let DS_l be (src, dst) pairs from (second-level buckets of) first-level buckets $\geq l$
Let b be the largest first-level bucket such that size of distinct sample $DS_b > \Theta(s)$
Let $v_1, \ldots, v_k$ be dst with highest frequencies (say $f^s_{v1}, \ldots, f^s_{vk}$) in distinct sample DS_b
Return $(v_1, \hat{f}_{v1} = 2^b f^s_{v1}), \ldots, (v_k, \hat{f}_{vk} = 2^b f^s_{vk})$



Alcatel·Lucent

Distinct-Count Sketch: Key result

Key result: If $r > \Theta(\log \text{streamsize})$ and $s > \Theta(\frac{U \log U}{f_{v_k} \varepsilon^2})$, then for each top-k distinct frequency f_v whp

$$|\hat{f}_v - f_v| \leq \varepsilon f_v$$

Intuition: Consider first-level bucket with Θ(s) (src, dst) pairs

- With r = Θ(log streamsize) second-level hash tables with s buckets each, every pair occurs without collisions in some second-level bucket whp
- Thus, possible to obtain a distinct sample of size $\Theta(s) = \Theta(\frac{U \log U}{f_{v_k} \varepsilon^2})$

Experimental results

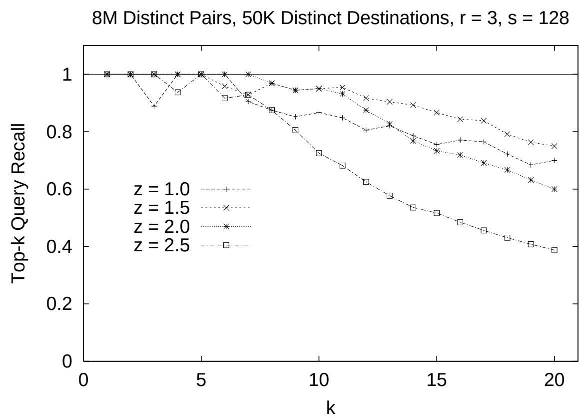
Synthetic data generator used to produce stream of (src, dst) pair updates

- Number of distinct (src, dst) pairs (U): 8 million
- Number of distinct dst: 50K
- Dst IP addresses follow Zipf distribution
  - Zipf parameter varied between 1 and 2.5 to control skew

Distinct-Count Sketch

- r=3 second-level hash tables with s=5 buckets each
- Size = 4.5 MB
  - → over an order of magnitude space savings
  - Space to maintain counts for 8 million (src, dst) pairs = 96 MB
- Processing time per stream update = 40-60 microsec on 1GHz Pentium-III

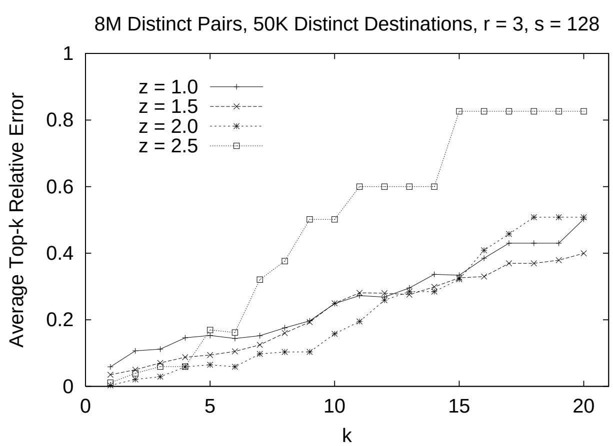
Top-k recall



- Recall for top-5 dst is almost always 100% (for all skew values z)
- For z < 2, recall is >86% and >73% for top-10 and top-15 dst, respectively

Alcatel·Lucent

Estimate relative errors



- Relative error is < 17% for top-5 dst (for all skew values z)
- For z<2, relative error is < 25% and < 35% for top-10 and top-15 dst, respectively

Alcatel·Lucent

Summary

Robust, real-time TCP-SYN-flooding attack detection requires the ability to track top-k destinations wrt

- Number of distinct connecting sources (as opposed to traffic volumes)
- Number of half-open connections (to distinguish from flash crowds)

Our proposed Distinct-Count Sketch

- Enables tracking of top-k distinct frequencies with guaranteed accuracy
- Is resilient to deletes (necessary to ignore legitimate TCP connections)

Experimental results indicate that Distinct-Count Sketches can accurately track top-k frequent destinations

- Low storage space overhead
- Low update processing time
- Low errors