Randomized Algorithms for Network Security and Peer-to-Peer Systems

Micah Adler
University of Massachusetts, Amherst
Talk Outline

• Probabilistic Packet Marking for IP Traceback
  – Network Security
  – Appeared in STOC 2002

• Load balancing in Peer-to-peer networks
  – A Stochastic Process on the Hypercube
  – Appeared in STOC 2003

• More details: www.cs.umass.edu/~micah
The IP Traceback Problem

• **Denial of Service Attacks:**
  – Attacker sends MANY packets to victim.
  – Denies access to legitimate users.

• **Difficulties:**
  – Source of packets can be forged.
  – Tools for coordinating from multiple locations.

• **Enforcing accountability:** The IP Traceback problem.
  – Determine the source of a stream of packets.
Probabilistic Packet Marking

• Suggested in [BurchC2000].

• Protocol of [SavageWKA2000]
  – Reserve header bits for IP Traceback
  – Each router on path of packet:
    • With small probability:
      – Write IP address into header; reset hop count.
    • Otherwise: increment hop count.

  – Victim of attack receives many packets:
    • Can reconstruct entire path (with high probability.)
Existing Work

• Elegant protocol: produced flurry of research.
  – [DoeppnerKK2000]
  – [LeeS2001]
  – [DeanFS2001]
  – [ParkL2001]
  – [SongP2001]

• Objectives include:
  – Reducing header bits required.
    • Full protocol of Savage et al: 16 bits.
  – Robustness against multiple paths of attack.
New results: single path of attack

• New technique for probabilistic marking:
  – One header bit is sufficient.
  – Number of packets required:
    • $n$: number of bits to describe path.
    – Any protocol that uses one bit:
      \[ \Theta(2^{2n}) \]
      \[ \Omega(2^{2n}) \]
  – Packets required by optimal protocol:
    • Grows exponentially with $n$.
    • Decreases DOUBLY exponentially with $b$.

• Number of header bits used: $b$
  – Packets required by optimal protocol:
    \[ 2^{\Theta(n/2^b)} \]
New results: many paths of attack

• Number of paths attacker can use: \( k \)
• Lower bound:
  – For any valid protocol \( b = \log(2k-1) \).
• Protocol: \( b = \log(2k+1) \) sufficient.
  – Requires restrictions on attacker.
  – Introduces powerful new coding technique.
    • New use of Vandermonde matrices.
Model for protocols

• Path of length $n$: each node has one bit.
• Objective: inform victim of all $n$ bits.
  – Easy to adapt to IP Traceback over Internet.
• Attacker sends $b$-bit packets along path.
  – Chooses initial setting of packets.
• Requirement on intermediate nodes:
  – No state information.
The one bit scheme

- **Idea:** encode bits $b_1 \ldots b_n$ into
  
  $- \ p = \Pr[\text{bit received by victim} = 1]$ 

- **Packets provide estimate of $p$.**

\[ p = \sum_{i=1}^{n} \left( \frac{1}{2} \right)^i b_i \]
The one bit scheme

• Protocol for each node $i$:
  – $b_r$: bit received from predecessor.
  – $b_i$: bit known to $i$.
  – Probability node $i$ forwards 1:

<table>
<thead>
<tr>
<th>$b_i$</th>
<th>$b_r = 0$</th>
<th>$b_r = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>$\frac{1}{2}$</td>
</tr>
<tr>
<td>1</td>
<td>$\frac{1}{2}$</td>
<td>1</td>
</tr>
</tbody>
</table>
The one bit scheme

• **Claim:** if initial bit set to 0:  
  \[ p = \sum_{i=1}^{n} \left( \frac{1}{2} \right)^i b_i \]

• **Proof:**
  - \( b_s \): bit sent by node.
  - \( b_i = 0 \) then \( \Pr[b_s = 1] = \Pr[b_r = 1]/2 \)
  - \( b_i = 1 \) then \( \Pr[b_s = 1] = \Pr[b_r = 1]/2 + 1/2 \)

• **Problem:** attacker might set initial bit to 1.
  \[ p = \left( \frac{1}{2} \right)^n + \sum_{i=1}^{n} \left( \frac{1}{2} \right)^i b_i \]

Result:
The one bit scheme

- Solution:
  
  \[
  \begin{array}{c|c|c}
  b_i = 0 & b_r = 0 & 0 \\
  & b_r = 1 & \frac{1}{2} - \varepsilon \\
  b_i = 1 & \frac{1}{2} & 1 - \varepsilon
  \end{array}
  \]

- If victim knows \( p \) within \( \pm \frac{1}{\varepsilon} \left( \frac{1}{2} - \varepsilon \right)^{-n} \)
  
  - All bits in path can be decoded.

- \( O \left( \left( \frac{1}{2} - \varepsilon \right)^{-2n} \right) \) packets sufficient (w.h.p.)
Extension to $b$ bits.

- Computing $p$ w/precision $\sqrt[2^n]{p}$: requires $\Theta(2^{2n})$ packets.

- Idea: use added bits to reduce precision needed.

- Protocol for each node:
  - Increment $(b-1)$-bit counter.
  - If counter overflows, perform $1$ bit protocol.

- Effective path length reduced by $\frac{1}{2^{b-1}}$
Extension to $b$ bits.

- **Problem:** How to guarantee victim sees all bits?
  - If attacker always sets initial bits the same, Victim only sees one type of counter.
  - Only provides $\frac{n}{2^{b-1}}$ bits on path.

- **Solution:**
  - Each node resets counter w/ small probability.
Extension to $b$ bits.

- Decoding:
  - More involved than single bit case.
  - Practical algorithm for decoding in software. $O(bn^2 2^b 2^{2n/2^{b-1}})$
  - Sufficient: packets.

- Proof of correctness fairly involved.

- Lower bound for any protocol:
Lower Bound.

- **Theorem:** for any protocol using less than $2^{b}2^{n/2^{b}}$ packets,
  \[ \Omega\left(2^{b}2^{n/2^{b}}\right) \quad \Pr[\text{wrong}] \geq \frac{1}{2} \]

- **Model:**
  - Network sends $n$-bit string to victim. 
  - Communication: $b$-bit packets. 
  - Requirement: network has no memory.
Wrapup of Probabilistic Packet Marking

• **Summary:**
  – Significantly more efficient new encoding technique.
  – Tradeoff header bits for packets.
  – Simple enough to be practical.
  – Multiple paths (many open problems . . .).

• **Other related work:**
  – Simulation experiments: tradeoffs seen in practice.
    • Joint work with Q. Dong and K. Hirata
  – Applications of PPM to congestion control.
    • Joint work with J. Cai, J. Shapiro, and D.
Talk Outline

• Probabilistic Packet Marking for IP Traceback
  – Network Security
  – Appeared in STOC 2002

• Load balancing in Peer-to-peer networks
  – A Stochastic Process on the Hypercube
  – Appeared in STOC 2003

• More details: www.cs.umass.edu/~micah
Coupon Collector’s Problem

• Objective: collect each of $n$ coupons.
  – Each step: receive one random coupon.
  – Well known: $n \log n \pm o(n \log n)$ steps required to obtain every coupon (whp).

• Natural variant:
  – Each step: check $\log n$ random coupons.
  – Receive one coupon if any are missing.
Structured Coupon Collector’s Problem

• Underlying graph $G=(V,E)$.

• Initially: all vertices uncovered.

• Each step: choose random vertex $v$.
  – If $v$ uncovered, cover it.
  – Else if any neighbors of $v$ uncovered, cover random neighbor.

• How many steps until all vertices covered?
Outline of rest of talk

• Application: distributed hash tables (DHTs).
  – Fundamental tool for Peer-to-Peer Networks.

• Load balancing in DHTs:
  – Analyze w/vertex covering process on hypercube.

• Theorem:
  \( O(n) \) steps enough for \( \log n \)-degree hypercube (whp)

• Implication: asymptotically optimal load
Distributed hash tables

Data Item Names → Addresses

hash

Storage partitioned over available nodes

Objectives:

• Find data items quickly.
• Balance load fairly.
Partitioning the address space

Strategy: maintain binary tree w/nodes at leaves

Handles addresses with prefix 011

Based on DHT of [RFHKS 2001] called CAN
Finding region of address space

• Nodes maintain pointers to each other:

  0 0 0 0 0 1 1 1 1 1
  0 0 1 1 0 0 1 1
  0 1 0 1 0 1 0 1

• Complete binary tree: pointers are hypercube
  – Nodes adjacent iff hamming distance = 1.

• New arrival:
  – Choose leaf node; split into two new leaves.
  – Node adjacency rule: truncate longer string.
Resulting distributed hash table:
Performance of DHT with $n$ nodes:

• Depends on rule for choosing node to split.
• Pointers per node: $O(\log n)$
• Queries to locate content: $O(\log n)$
• Load balance:

$$V(x) = \frac{\max_{x \in \text{nodes}} V(x)}{\min_{x \in \text{nodes}} V(x)}$$

- $V(x)$: fraction of address space stored at $x$.
  - $V(x) = 2^{-\text{depth}(x)}$
Rules for choosing node to split

• Simple rule:
  – Choose hash address uniformly at random.
  – Split node storing that address.
  – Resulting load balance: $\mathcal{T}(\log n)$ w.h.p.

• Our main contribution: analyze a better rule.
  – Choose node as in simple rule.
  – Split shallowest neighbor of that node.
  – Resulting load balance: $O(1)$ w.h.p.
  • First $O(1)$ with $O(\log n)$ pointers,
Previous Work

- **CAN [RFHKS 2001]:** \(k\)-Dim. Torus
  - Our hypercubic DHT is CAN with \(k = 8\)
  - Suggested both splitting rules.
    - No analysis of resulting load balance.

- **Pastry [RD 2001], Tapestry [ZKJ 2001]**
  - Based on [PRR 1997]
  - Pointers, queries, load balance, all \(T(\log n)\)
More Previous Work

• **Chord [SMKKB 2001]**:
  – Pointers, queries, load balance, all $T(\log n)$
  – Additional techniques:
    • load balance $O(1)$ but pointers $T(\log^2 n)$

• **Viceroy [MNR 2002]**:
  – Pointers $O(1)$, queries $T(\log n)$.
  – Does not address load balance.
  – Combine with technique from [SMKKB 2001]:
    • Results similar to ours.
Reduction to hypercube covering process

To show: w.h.p.,
• \( d - \log n \) not too large.
• \( \log n - s \) not too large (hypercube process).
No node “falls behind”

• Consider progress of nodes at level $s$:
  – Each arrival is step of covering process.
  – Node is covered when it is split.

• Theorem:
  – Vertex covering process on $n$-node hypercube: $O(n)$ steps sufficient w.h.p.

• Corollary:
  – $\log n - s$ is always $O(1)$ w.h.p.
Easier result: $O(n \log \log n)$ steps.

- $\log \log n$ phases of $O(n)$ steps each.
- w.h.p.: at end of phase $i$:
  - Each node has $< \log n / 2^i$ uncovered neighbors.

What is $\Pr[\text{hit } L_1 \text{ during step of phase } i]$?

- Assume $\log n / 2^{i-1} = |L_1| = \log n / 2^i$
Easier result: $O(n \log \log n)$ steps.

$L_2 = \text{the covered neighbors of } L_1$

- $\Pr[L_1 \text{ hit in one step}] = \sum_{u \in L_2} \frac{1}{n} \frac{2^{i-1}}{\log n} = |L_2| \frac{2^{i-1}}{n \log n}$

- $|L_2| = \frac{1}{4} |L_1| \log n = \frac{\log^2 n}{2^{i+2}}$

- Thus: $\Pr[L_1 \text{ hit in one step}] = \frac{\log n}{8n}$

- Chernoff bounds: $\Pr[\text{Any } L_1 \text{ not halved in phase}] = l / \text{poly}(n)$. 
Why $O(n)$ seems possible.

Phase $i$: expected steps until $L_1$ halved:

- $L_1$ has size $\log n / 2^i$.
- $\Pr[L_1 \text{ hit in one step}] = \frac{\log n}{8n}$
- Expected steps: $O\left(\frac{n}{2^i}\right)$
- $O(n)$ steps guarantees $O(\log n)$ expected hits.
  - $\Pr[\text{not halving}] = 1/n^c$
Intuition for a bound of $O(n)$.

- **Idea:**
  - Phase $i$: 
    - $O\left(\frac{ni}{2^i}\right)$ steps to shrink $L_3$
    - $L_3$ larger, so more likely to be close to expectation
  - $\Pr[L_1\text{ hit in a step}] = \frac{2^i \log n}{n}$
  - $O\left(\frac{n}{2^i}\right)$ steps sufficient to halve all $L_1$s whp.
Extensions:

• Sufficient (whp) for any $d$-regular graph:
  \[ O(n\left(1 + \frac{\log n \cdot \log d}{d}\right)) \]

• Sufficient whp for random $d$-regular graphs:
  \[ O(n\left(1 + \frac{\log n}{d}\right)) \]

• All results hold if never cover chosen node.
Open problems for stochastic process

• Adding deletions
• Improving the constants
• $O(n)$ for all $\log n$-regular graphs?