Random Flow Network Modeling and Simulations for DDoS Attack Mitigation

Jiejun Kong, Mansoor Mirza, James Shu, Christian Yoedhana, Mario Gerla, Songwu Lu
Computer Science Department
University of California, Los Angeles

Outline
- Problem
- Current countermeasures
- Our model
- Simulation & conclusions
Problem Statement

- Distributed clients “maliciously” send data packets to a service site, and regular service requests are starved due to congestion
  - Connection bandwidth is defined as the bottleneck link bandwidth en route. DDoS is effective if this metric approaches zero
  - “Maliciousness”: junk traffic, source address spoofing

Countermeasure: Source Traceback

- An indirect counterattack against IP spoofing, a common trick used by DDoS attackers
- Probabilistic Packet Marking (PPM, Sigcomm 2000)
  - X: # of marked packets needed
d: # of hops
p: marking probability
  - Expected # of packets needed \[ E(X) < \frac{\ln(d)}{p(1 - p)^{d-1}} \]
- Hash-based traceable logging (SPIE, Sigcomm 2001)
  - Use collision-resistant hash functions to generate and record checksum of data packets
  - Path verification upon victim’s call
Countermeasure: Filtering “Malicious” Traffic

- Effective means to decrease # of those sources using “wrong” source address
- Ingress filtering (RFC2267, RFC2867)
  - Effective in access networks only
- Distributed Packet Filter (DPF, Sigcomm 2001)
  - Effective in core networks and when topological update is negligible
- Source Address Validity Enforcement (SAVE, Infocom 2002)
  - Extends ingress filtering to core networks

Countermeasure: Rate Control

- Reduce DDoS to a congestion control problem
- Pushback (NDSS 2002)
  - Aggregate-based Congestion Control (ACC)
  - Aggregate: a subset of traffic with an identifiable property
  - Identify aggregates responsible for congestion, and preferentially drop them at routers
- Router throttle (IWQoS 2002)
  - Achieving max-min fairness $[L_s, U_s]$ by (de)activating the throttles at upstream routers
Our Observations

- Lack of general modeling so far
  - Smart punches and maneuvers exchanged between attackers and network designers
- No clear boundary between DDoS and service availability
  - Large amount of clients $\rightarrow$ service availability
  - Large amount of “malicious” clients $\rightarrow$ DDoS
  - But many aspects of “maliciousness” are subjective notions without clear definition
- Many countermeasures violate the end-to-end argument, thus cannot be realized in the near future

Modeling: Random Flow Network

Supersource $\rightarrow$ Supersink

sinknet$\rightarrow$sourcenet
Modeling: Problem Statement

- Max-flow min-cut
  - Maximum flow from $S$ to $T$ determines a min-cut $(X,\overline{X})$
- Updates in SourceNet and SinkNet incur changes in the min-cut
  - $(X,\overline{X}) \rightarrow (X',\overline{X}')$
- The DDCP (DDoS Countermeasure Problem) is to minimize the sum cost of 3 positive functions
  - Source cost function $f(-\Delta S)$
  - Sink cost function $g(+\Delta T)$
  - Partition penalty function $h(S,T)$
    in particular, $h(X,\overline{X}) - h(X',\overline{X}')$

DDCP is a hard integer programming problem

- DDCP is a very complex integer programming with large number of variates
  - Theoretic answer hard to obtain
  - In practice, all the cost modeled can be obtained “post-mortem” after an attack (but assuming ubiquitous traffic logging)
- For effective countermeasures, we require the sum $C$ to be at least negative

$$C = \sum_{i \in \Delta S} f(i) + \sum_{i \in \Delta T} g(i) + \left( \sum_{j \in X', k \in X} h(j, k) - \sum_{j \in X, k \in \overline{X}} h(j, k) \right)$$
$$C < 0$$
The Necessity of Simulation

- Function $f$ and $g$ depend on out-of-band issues not addressable in this work
- Theoretic answers not available
- At least we can use simulations to evaluate the partition penalty cost (i.e., the portion with $h$ function)

$$
\left( \sum_{j \in X, k \in X} h(j, k) - \sum_{j \in X, k \in X} h(j, k) \right)
$$

Simulation Illustration: Internet-like Random Flow Networks

- A random network generated by Michigan’s INET generator
  - Server in red
  - Clients in cyan
- Conforming to Internet power-law constraints
Simulation Setup

- NS2
- Clients/servers using HTTP
- Regular clients follow a Pareto model in transmission
  - Pareto model ($\alpha=1.4$) is empirically observed as the Internet application transmission model
- Malicious attackers pump traffic without intermittence
- # of malicious attackers increases from 0 to 40% of all nodes

Simulation Results

- X-axis: sink size
  - i.e. # of servers
- Y-axis: source size
  - # of attackers
- Z-axis: normalized goodput
  - 100% when # of server is 1 and # of attackers is zero
Simulation Results 2
(Trivia)

- When source size increases from 0 to 40% of all senders, goodput decreases to ≈70%
- When sink size increases from 0 to ≈10, goodput comes back to ≈100% for all sink sizes simulated
  - This means the partition penalty cost is 0 ➜ DDoS is neutralized by service availability

Simulation Results 3
(Implications)

- Both source size decrement and sink size increment are effective countermeasures
- In a network topology, deploying a limited number of middlewares or proxies is effective against DDoS
  - Middlewares/proxies are particularly necessary to address single point of failure (with respect to DDoS)
  - When sink size is greater than a threshold, no clear boundary between DDoS and service availability
Sinknet Example: Pushback Router
(Ioannidis & Bellovin, NDSS’02)

- The victim being attacked is the supersink
- A successful pushback to a pushback router means adding the router into the sinknet
- Expected to be an effective countermeasure in a power-law network like the Internet

Conclusions

- Besides SourceNet decrement, SinkNet increment is also critical to resist DDoS attacks
  - It verifies the effectiveness of pro-sink countermeasures (e.g., pushback router, content delivery network, network middlewares)
  - May not need to differentiate DDoS attack and service availability: e.g., service middlewares solve both problems