DDoS Defense by Offense

Michael Walfish, Mythili Vutukuru, Hari Balakrishnan, David Karger, and Scott Shenker, SIGCOMM ‘06

Presented by Nikki Benecke, Nov. 7th, 2006, for CS577
Outline

• Introduction
• Design
• Implementation
• Evaluation
• Concerns
• Conclusions
Introduction
Basic Idea

• Defense against *application* level DDoS attacks
  – Way of dealing with attack as it occurs, not a prevention scheme
Application-level attacks

- Attacker sends proper looking requests to waste server’s resources
- Overwhelms server, not access links
- Cheaper than link-level attacks (for the attacker)
- Attack traffic is harder to identify
Application-level attacks

Current defenses focus on slowing down attackers/stopping the attack.

But good clients are totally drowned out in these defense systems – authors say it’s time for them to *speak-up*. 
3 Types of Defenses

- Overprovision computation resources massively
- Detect and block attackers
- Charge *all* clients a currency
Speak-up

- It’s a currency-based defense that uses bandwidth as the currency
  - Claim: attackers use most of their available bandwidth during attacks, victims don’t
  - Use encouragement to make victims send more traffic so they are better represented at the server
Two conditions to make it work…

- **Adequate Link Bandwidth**: there must be enough spare bandwidth to allow for speak-up inflated traffic
- **Adequate Client Bandwidth**: the aggregate bandwidth of all good clients must be on the same order as the attackers’
Three conditions where it wins…

• No predefined clientele: Makes filtering impossible, so use speak-up.
• Non-human clientele: Makes “human” tests (type in the word, etc) impossible, so use speak-up.
• Unequal requests or spoofing or smart bots: No method for dealing with the first, can’t use methods for the second two… use speak-up!!!
Design
Design Goal

Allocate resources to competing clients in proportion to bandwidth

If the good clients make $g$ requests per second in aggregate and have an aggregate bandwidth of $G$ requests per second to the server, and if the bad clients have an aggregate bandwidth of $B$ requests per second, then the server should process good requests at a rate of $\min(g, \frac{G}{G+B}c)$ requests per second.
3 Required Mechanisms

1. Way to limit the total requests to the server to its max, $c$
2. Mechanism to reveal available bandwidth/provide encouragement
3. Proportional allocation mechanism – let clients in proportional to delivered bandwidth

Hence, the *thinner* appears.
Explicit Payment Channel

Thinner wants to pad client traffic with dummy bytes, but how many should we pad with?

We don’t want to need to know that information!
Explicit Payment Channel

- When server is overloaded, thinner asks clients to open separate payment channels
- Client sends bytes on this channel, becomes a contender
- Thinner tracks how much each contender sends
- When server is free, thinner admits the highest bidder and closes the channel
Heterogeneous requests

Charging the same amount for unequal requests gives unfair advantage to attacker

So charge per “chunk” instead of per request
Heterogeneous requests

Instead of closing the bid channel after accepting a client, keep it open until request is served

Every unit of service time, reopen the auction
Heterogeneous requests

1. At time t, v is active connection, u is the highest contender

2. u > v, SUSPEND v, ADMIT (RESUME) u

3. v > u, let v continue sending, but reset its payment counter for time t+1

4. ABORT requests that have been suspended too long
Implementation
Implementation

- Clients send by Poisson process, limited windows (open requests)
- Deterministic service time (all reqs equal)
- Bad clients send faster, and have bigger windows
- Good client: $\lambda = 2, \ w = 1$
- Bad client: $\lambda = 40, \ w = 20$
Implementation

- Max. number of clients limited to 50 by testbed
- Small scale for representing DDoS
- However, they think it’ll still work on a larger scale
Evaluation
Evaluation

• Validating the thinner’s allocation
• Latency and byte cost
• Adversarial advantage
• Heterogeneous network conditions
• Good and bad clients sharing bottlenecks
• Impact of speak-up on other traffic
Validating the thinner’s allocation

Question 1: Do groups get service in proportion to bandwidth?

Setup: 50 clients over 100 Mb/s LAN
Each gets 2Mb/s
\( c = 100 \text{ req/s} \)
Vary \( f \), the fraction of good clients
Validating the thinner’s allocation
Validating the thinner’s allocation

- Speak-up defended clients are always a little behind ideal
  - Gaming
- Always fare better than undefended
Validating the thinner’s allocation

Question 2: What happens when we vary the capacity of the server?

Setup: 25 good clients, 25 bad clients

\[ cid = 100 \]
\[ c = 50, 100, 200 \]
Validating the thinner’s allocation

![Graph showing server allocation to good and bad clients and fraction of good requests served for different server capacities (50, OFF; 50, ON; 100, OFF; 100, ON; 200, OFF; 200, ON).]
Validating the thinner’s allocation
Validating the thinner’s allocation

• Good clients do best when the server has ability to process all requests (i.e., large $c$)
• Service proportional to bandwidth even when server can’t process all requests
Latency cost

• Same setup as last experiment
• Measures the length of time clients spend uploading dummy bytes
Latency cost
Latency cost

• With a large $c$, cost isn’t very high
• Even with a small $c$, worst added delay is 1s.
  – Pretty bad, but not as bad as getting no service during an attack, right?
Byte Cost

- Still the same setup
- Measure the average number of bytes uploaded for served requests
Byte Cost

DDoS: Defense by Offense
Byte Cost

• Bad clients do end up paying more than good clients
  – But do they pay significantly more?
Heterogeneous Network Conditions

- First, look at varied bandwidth
- 5 client categories, 10 clients in each category
- Bandwidth for category $i = 0.5i$ Mbps ($1 \leq i \leq 5$)
- All clients are *good clients*
- $c = 10$
Heterogeneous Network Conditions

![Graph showing fraction of server allocated vs. bandwidth (Mbits/sec)]
Heterogeneous Network Conditions

• Roughly proportional to bandwidth of clients
• Close to ideal
Heterogeneous Network Conditions

• Now look at effect of varied RTT
• 5 client categories, with 10 clients in each
• RTT for category $i = 100i$ ms ($1 \leq i \leq 5$)
• Bandwidth per client = 2 Mbps
• $c = 10$
• Run with all good or all bad clients
Heterogeneous Network Conditions

![Graph showing fraction of server allocated vs. RTT (ms). The graph compares all-good experiment, all-bad experiment, and ideal for both experiments.](image_url)
Heterogeneous Network Conditions

• Good clients with long RTTs do worse than any bad clients
• “Effect is limited”
  – No one gets > 2*ideal
  – No one gets < 1/2*ideal
Good and Bad Sharing a Bottleneck

- 30 clients, each with 2 Mbps, connect to thinner through link $l$
- $l$‘s bandwidth = 40 Mbps
- 10 good, 10 bad clients, each with 2Mbps, connect directly to thinner
- $C = 50$ req/s
- Vary number of good/bad behind $l$
Good and Bad Sharing a Bottleneck

- Actual fraction of ‘bottleneck service’ to good
- Actual fraction of ‘bottleneck service’ to bad
- Ideal fraction of ‘bottleneck service’ to good
- Ideal fraction of ‘bottleneck service’ to bad
- Ideal fraction served: bottlenecked good
- Actual fraction served: bottlenecked good

Diagram showing the fraction of clients served under different conditions:
- 5 good, 25 bad
- 15 good, 15 bad
- 25 good, 5 bad

Number of clients behind shared bottleneck

DDoS: Defense by Offense
Good and Bad Sharing a Bottleneck

- Clients behind \( I \) capture half of the server’s capacity
- Good behind \( I \) suffer some, especially with greater number of bad clients
- Effect on good clients greater when bottleneck is smaller
Impact of speak-up on other traffic

- Bottleneck, $m$, shared between speak-up clients and TCP endpoint, $H$
- Run with H as a sender, $m$ is shared fairly
- Run with H as a receiver, H’s ACKs will get lost, H’s requests will be delayed
Impact of speak-up on other traffic

- Experimented specifically with H as a receiver
- 10 good speak-up clients, 1 HTTP client downloading with wget
- m = 1Mbps, rest = 2Mbps
- c = 2
Impact of speak-up on other traffic

![Graph showing the impact of speak-up on other traffic. The x-axis represents the size of HTTP transfer in KBytes, and the y-axis represents end-to-end latency in seconds. The graph compares the latency with and without speak-up, showing a significant reduction with speak-up.]
Impact of speak-up on other traffic

- Huge impact on H
- “Pessimistic” result according to authors
- Only happens during attack, might be worth it to help “defend the Internet”
Concerns
Concerns/Cautions/Objections

• Does speak-up hurt small sites?
  – Yes, for current sized botnets
  – But this might get better with smaller botnets

• Does speak-up hurt the whole Internet?
  – Not really
  – Only for servers under attack
  – Core overprovisioning dampens the effect
  – Congestion control will keep speak-up under control
Concerns/Cautions/Objections

• Bandwidth envy
  – Only “more better off” during attacks
  – ISPs could offer high bw proxies to low bw clients

• Variable bandwidth costs
  – Again, offer a high bw proxy
  – Or let customers decide whether or not to bid
Concerns/Cautions/Objections

• Incentives for ISPs
  – The basic goodness of society will protect us!!!
• Solving the wrong problem
  – Cleaning up botnets is good, but we need to do something in the meantime
• Flash crowds
  – Reasonable to treat them as attacks
  – Wouldn’t effect low bw sites in the first place
Conclusions
Conclusions

• Not sure who wants/needs speak-up
  – Survey to find out
• Speak-up does what it proposes to do
Conclusions

• Main advantages
  – Network elements don’t need to change
  – Only need to modify servers and add thinners

• Main disadvantages
  – Everyone floods, so harder to detect bad clients
  – Hurts edge networks
  – Rendered useless if access links to thinner are saturated
My Questions

• Are *speak-up’s* assumptions reasonable?
Questions?