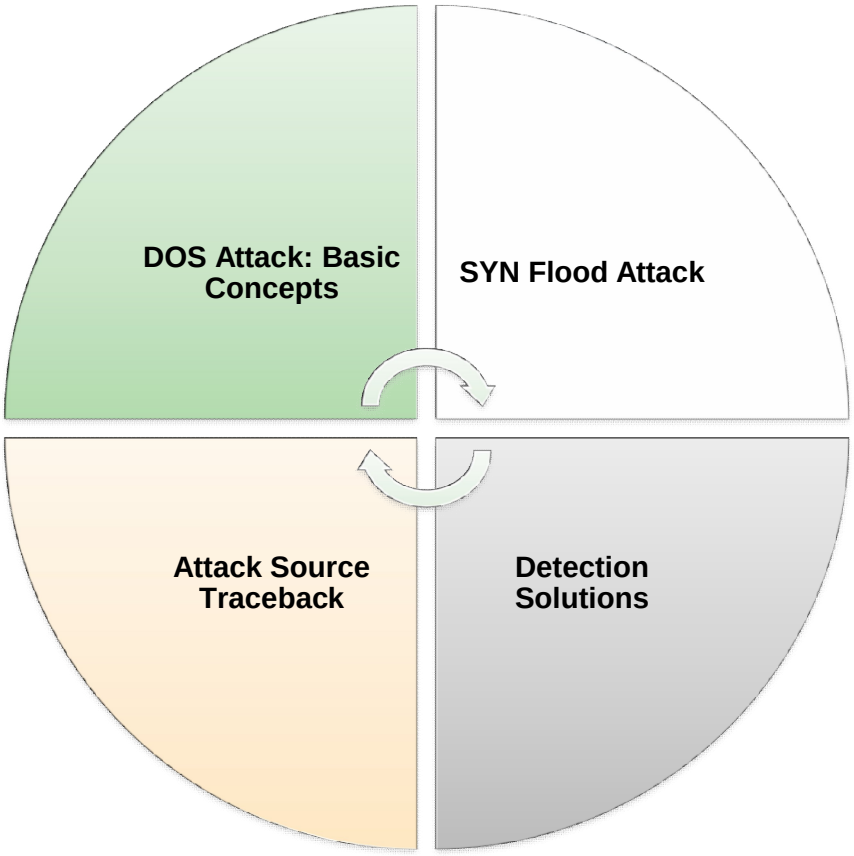
Denial of Service Attacks and Solutions

Van Nguyen – HUT

Hanoi – 2010

Agenda


DOS Attack: Basic Concepts
SYN Flood Attack
Attack Source Traceback
Detection Solutions

Denial-Of-Service

- Flooding-based
- Send packets to victims
  - Network resources
  - System resources
- Traditional DOS
  - One attacker
- Distributed DOS
  - Countless attackers

DDoS

- A typical DDoS attack consists of
  - Stealing partial control of a large number of hosts
  - Secretly manipulating them to send dummy packets to jam a victim or/and its Internet connection

- Can be done in following ways:
  - Exploiting system design weaknesses
    - e.g. ping to death
  - Imposing computationally intensive tasks on the victim
    - E.g. encryption and decryption
  - Flooding based DDoS Attack.

Attacks Reported

- *May/June, 1998*
  - First primitive DDoS tools developed in the underground:
    - Small networks, only mildly worse than coordinated point-to-point DoS attacks.
- *August 17, 1999*
  - Attack on the University of Minnesota reported to UW network operations and security teams.
- *February 2000*
  - Attack on Yahoo, eBay, Amazon.com and other popular websites.
- Once, more than 12,000 attacks during a three week period.

Reference: http://staff.washington.edu/dittrich/misc/ddos/timeline.html

DDoS Attacks

- Not necessarily rely on particular network protocols or system design weaknesses
- A major threat because of:
  - availability of a number of user-friendly attack tools
  - Lack of effective solutions to defend against them
- The attacks can be classified into
  - Direct Attacks
  - Reflector Attacks.

Direct Attacks

- A large number of packets sent directly towards a victim.
- Source addresses are usually spoofed
  - the response goes elsewhere (or no where)
- Examples:
  - *TCP-SYN Flooding:* The last message of TCP's 3 way handshake never arrives from source.
  - Congesting a victim using ICMP messages, RST packets or UDP packets.
  - Attack packet onserved: TCP packets (94%), UDP packets (2%) and ICMP packets(2%).

Direct Attack

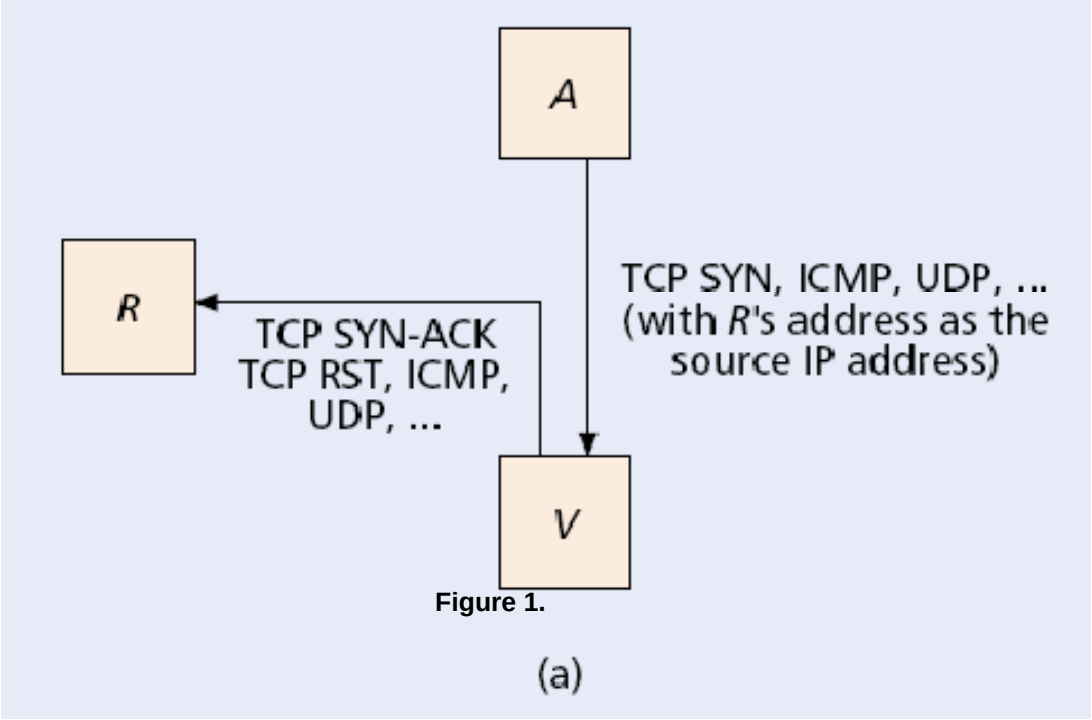


Figure 1.

Agent Programs: Trinoo, Tribe Flood Network 2000, and Stacheldraht

Reflector Attacks

- Uses innocent intermediary nodes (routers and servers) known as reflectors.
- An attacker sends packets that require responses to the reflectors with the packets' inscribed source address set to victim's address.
- Can be done using TCP, UDP, ICMP as well as RST packets.
- Examples:
  - Smurf Attacks: Attacker sends ICMP echo request to a subnet directed broadcast address with the victim's address as the source address.
  - SYN-ACK flooding: Reflectors respond with SYN-ACK packets to victim's address.

Reflector Attack

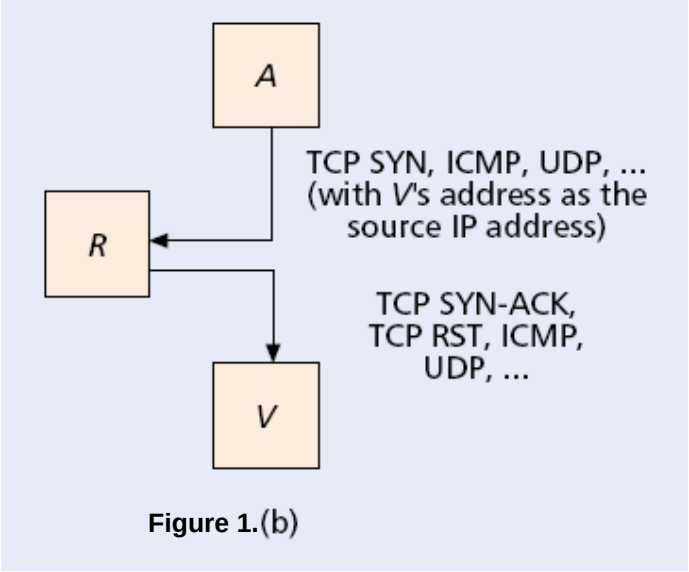


Figure 1.(b)

- Cannot be observed by backscatter analysis, because victims do not send back any packets.
- Packets cannot be filtered as they are legitimate packets.

DDoS Attack Architectures

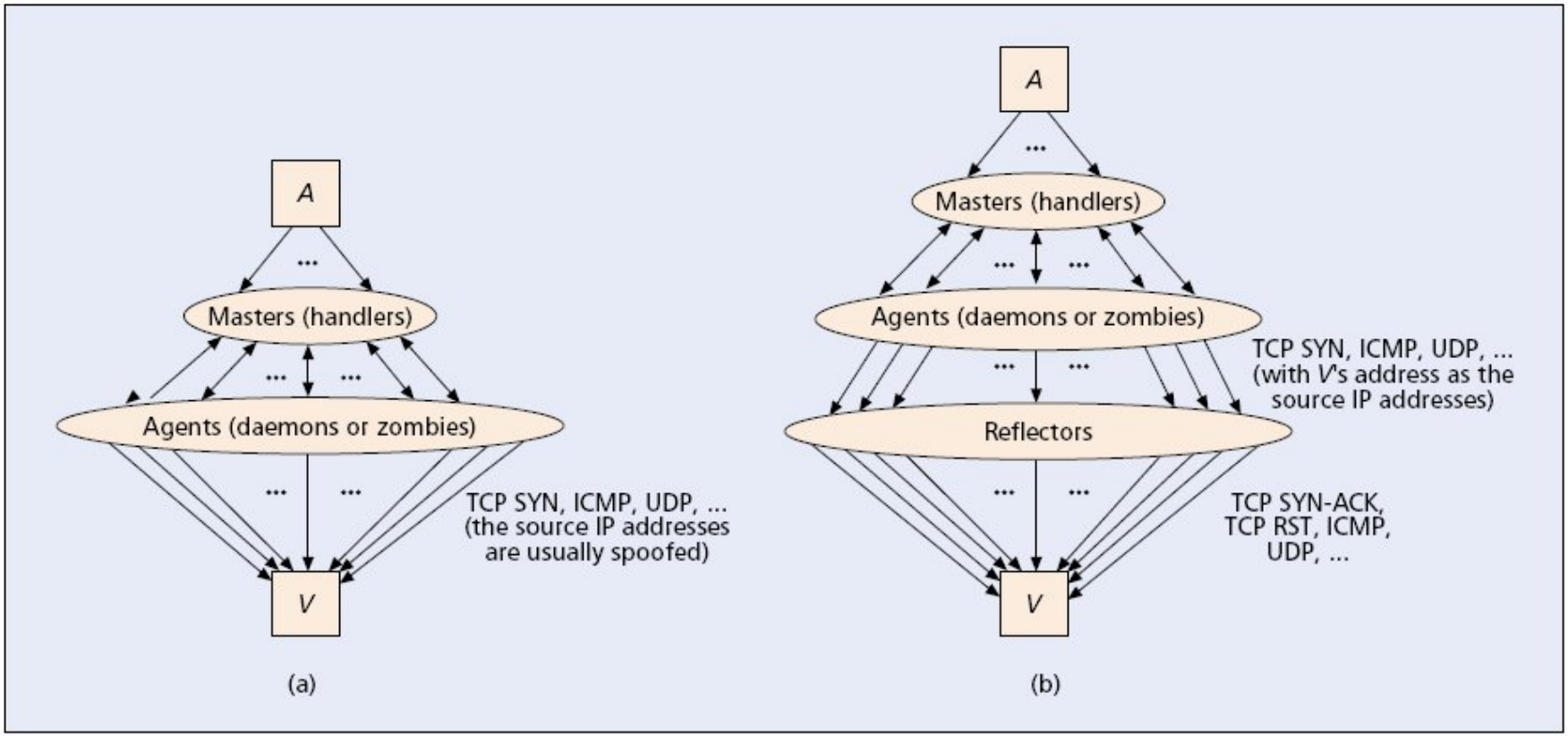


Figure 2. *DDoS attack architectures for a) direct and b) reflector attacks.*

Some Reflector Attack Methods

	Packets sent by an attacker to a reflector (with a victim's address as the source address)
Smurf	ICMP echo queries to a subnet-directed broadcast address
SYN flooding	TCP SYN packets to public TCP servers (e.g., Web servers)
RST flooding	TCP packets to nonlistening TCP ports
ICMP flooding	• ICMP queries (usually echo queries) • UDP packets to nonlistening UDP ports • IP packets with low TTL values
DNS reply flooding	DNS (recursive) queries to DNS servers
	Packets sent by the reflector to the victim in response
Smurf	ICMP echo replies
SYN flooding	TCP SYN-ACK packets
RST flooding	TCP RST packets
ICMP flooding	• ICMP replies (usually echo replies) • ICMP port unreachable messages • ICMP time exceeded messages
DNS reply flooding	DNS replies (usually much larger than DNS queries)

Solutions to the DDoS Problems

- There are three lines of defense against the attack:
  - Attack Prevention and Preemption *(before the attack)*
  - Attack Detection and Filtering *(during the attack)*
  - Attack Source Traceback and Identification *(during and after the attack)*

- A comprehensive solution should include all three lines of defense.

Attack Prevention and Preemption

- Protect hosts from master and agent implants:
  - Using signatures and scanning procedures to detect them.
  - Monitor network traffic for known attack messages sent between attackers and masters.
- Using cyber-spies to intercept attack plans (e.g., a group of cooperating agents).
- Inadequate, anyway.

Attack Source Traceback and Identification

- This is an act after-the-fact
- *IP Traceback* Identifying actual source of packet without relying on source information.
  - Deploy routers to help: can record information they have seen.
    - Routers can send additional information about seen packets to their destinations.
- Sometime Infeasible to use IP Traceback:
  - Cannot always trace packets' origins. (NATs and Firewalls!)
  - Ineffective in reflector attacks.
  - But helpful for post-attack law enforcement.

Attack Detection and Filtering

- Deployed in two phases:
  - *Attack Detection*: identifying DDoS attack packets.
  - *Packet Filtering*: classifying and dropping those packets .

- Effectiveness of Detection
  - FPR (False Positive Ratio):
    No. of *false positives*/Total number of confirmed normal packets
  - FNR (False Negative Ratio):
    No. of *false negatives*/Total number of confirmed attack packets

Both metrics should be low!

Attack Detection and Filtering

- Effectiveness of Filtering
  - Effective attack detection DOES NOT IMPLY Effective packet filtering
    - Detection phase uses victim identities (Address or Port No.), so even normal packets with same signatures can be dropped.
  - NPSR (Normal Packet Survival Ratio):

    Percentage of normal packets that can *survive* in the midst of an attack ➔ *NPSR should be high!*

Attack Detection and Filtering

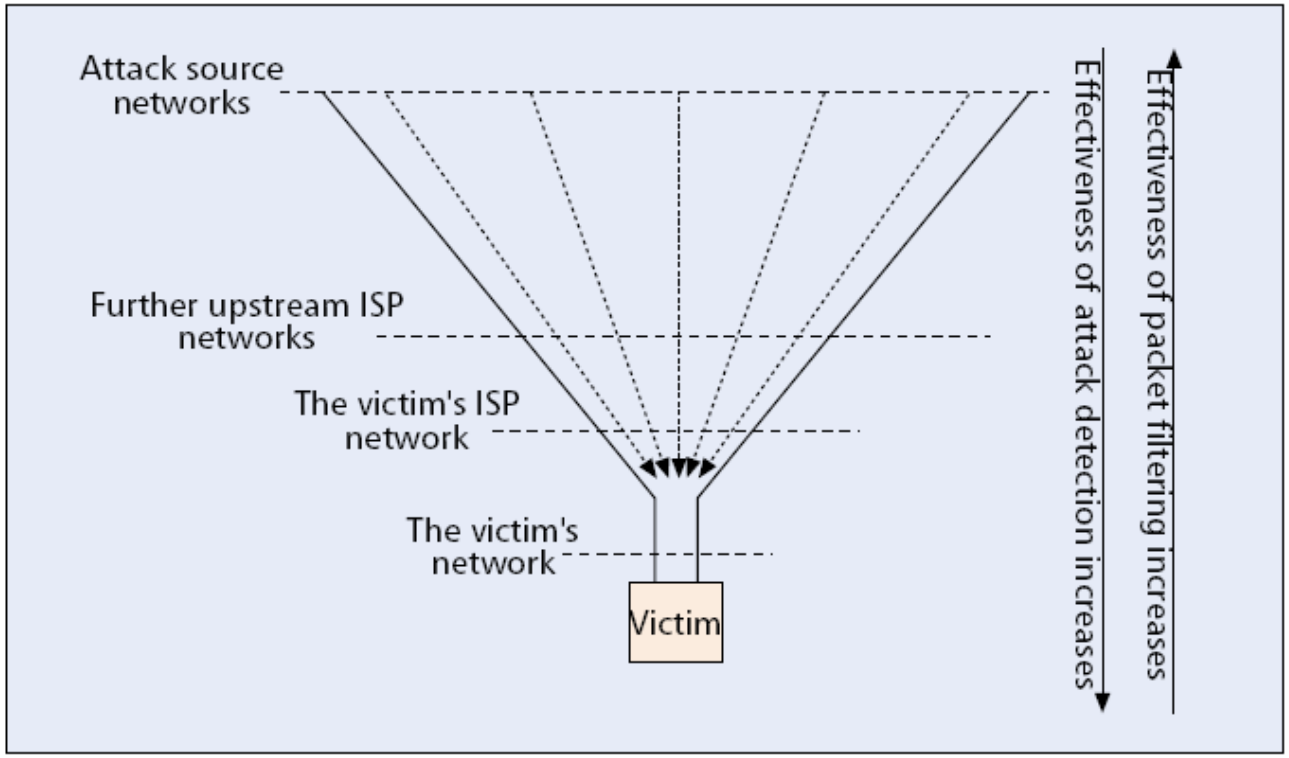


Figure 4. *Possible locations for performing DDoS attack detection and filtering.*

Attack Detection and Filtering

- At Source Networks:
  - One can filter packets based on address spoofing
  - Direct attacks can be traced easily, difficult reflector attacks
  - Ensure all ISPs have ingress packet filtering
    - Filter the spoofed address packets which's source IPs do not belong to the source network
    - Very difficult to deploy this for all ISP (Impossible?)
- At the Victim's Network:
  - Victim can detect attack based on volume of incoming traffic or degraded performance
    - Commercial solutions available.
  - Other mechanisms: *IP Hopping*
  - Last Straw: If incoming link is jammed, victim has to shut down and ask the upstream ISP to filter the packets.

Attack Detection and Filtering

- At a Victim's Upstream ISP Network:
  - Filter packets as asked by victim
  - Can be automated by carefully designed intrusion alert systems
  - Not a really good idea though
    - Normal packets can still be dropped
    - This upstream ISP network can still be jammed under large-scale attacks.
- At further Upstream ISP Networks:
  - The above approach can be further extended to other upstream networks.
  - Effective only if ISP networks are willing to co-operate and install packet filters.

An Internet Firewall

- A bipolar defense scheme cannot achieve both effective packet detection and packet filtering.

  ➔ deploy a global defense infrastructure.

  The plan is to detect attacks right at the Internet core!

- Two methods, which employ a set of distributed nodes in the Internet to perform attack detection and packet filtering.
  - Route-based Packet Filtering Approach (RPF)
  - Distributed Attack Detection Approach (DAD)

Route-Based Packet Filtering (RPF)

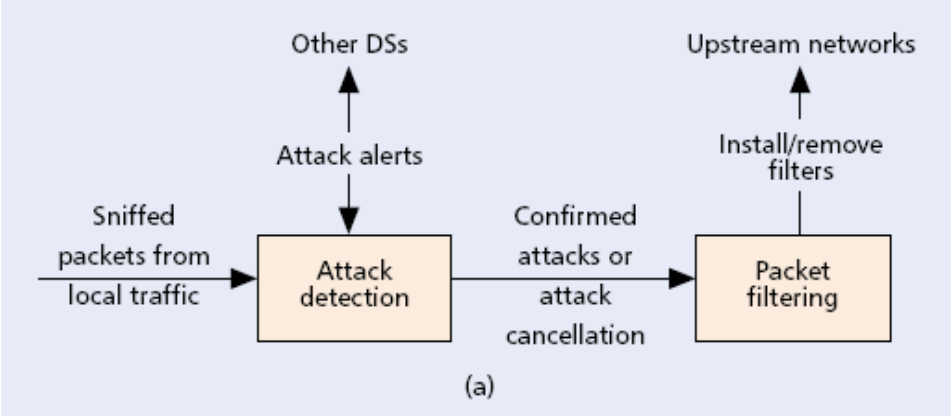
- Extends the idea of ingress packet filtering
  - Using distributed packet filters to examine the packets, based on addresses and BGP routing information.
    - A packet is considered an attack packet if it comes from an unexpected link.
- Major Drawbacks
  - BGP messages to carry the needed source addresses ➔ Overhead!
  - Deployment is extremely demanding
    - Once thought: Filters were to be placed in 1800 out of 10,000 ASs
    - # ASs is continuously increasing.
  - Can't work against reflected packets.

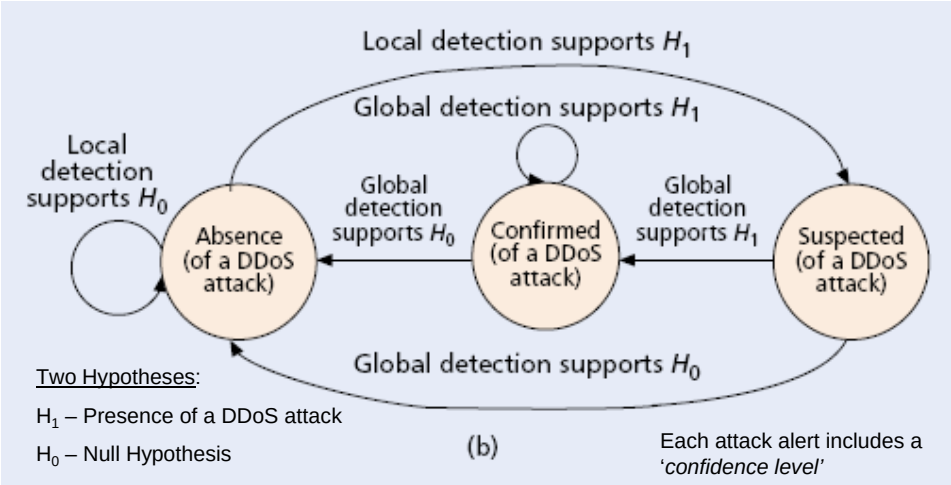
Distributed Attack Detection (DAD)

- The idea is to deploys multiple Distributed Detection Systems (DSs) to observe network anomalies and misuses.
  - *Anomaly detection*: Identifying traffic patterns that significantly deviate from normal
    - e.g., unusual traffic intensity for specific packet types.
  - *Misuse detection*: Identifying traffic that matches a known attack signature.
- DSs to exchange attack information from local observations
  - Statefull in respect to the DDoS attacks.
- Still a challenging to ask for an effective and deployable architecture

Distributed Attack Detection

DS Design Considerations





Other considerations:

- Filters should be installed only on attack interfaces on 'CONFIRMED' state
- All DSs should be connected '*always*'
- Works in Progress:
  Intrusion Detection Exchange Protocol
  Intrusion Detection Message Exchange Format

Figure 5. *a) High-level DS architecture and b) a state diagram of two-level attack detection in the distributed detection approach.*

SYN FLOOD DEFENSE SOLUTIONS

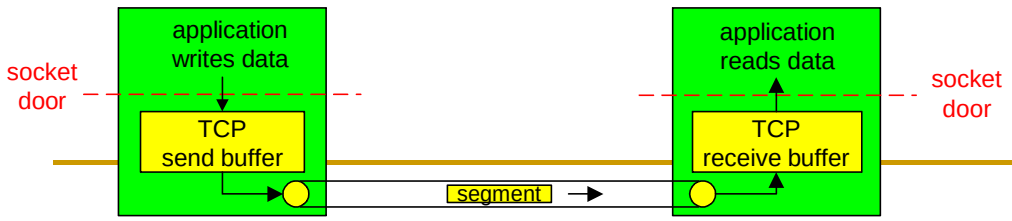
TCP SYN-Flooding Attack

- TCP services are often susceptible to various types of DoS attacks
  - SYN flood: external hosts attempt to overwhelm the server machine by sending a constant stream of TCP connection requests
    - Streaming spoofed TCP SYNs
    - Forcing the server to allocate resources for each new connection until all resources are exhausted
  - 90% of DoS attacks use TCP SYN floods
  - Takes advantage of three way handshake
    - Server start "half-open" connections
    - These build up… until queue is full and all additional requests are blocked

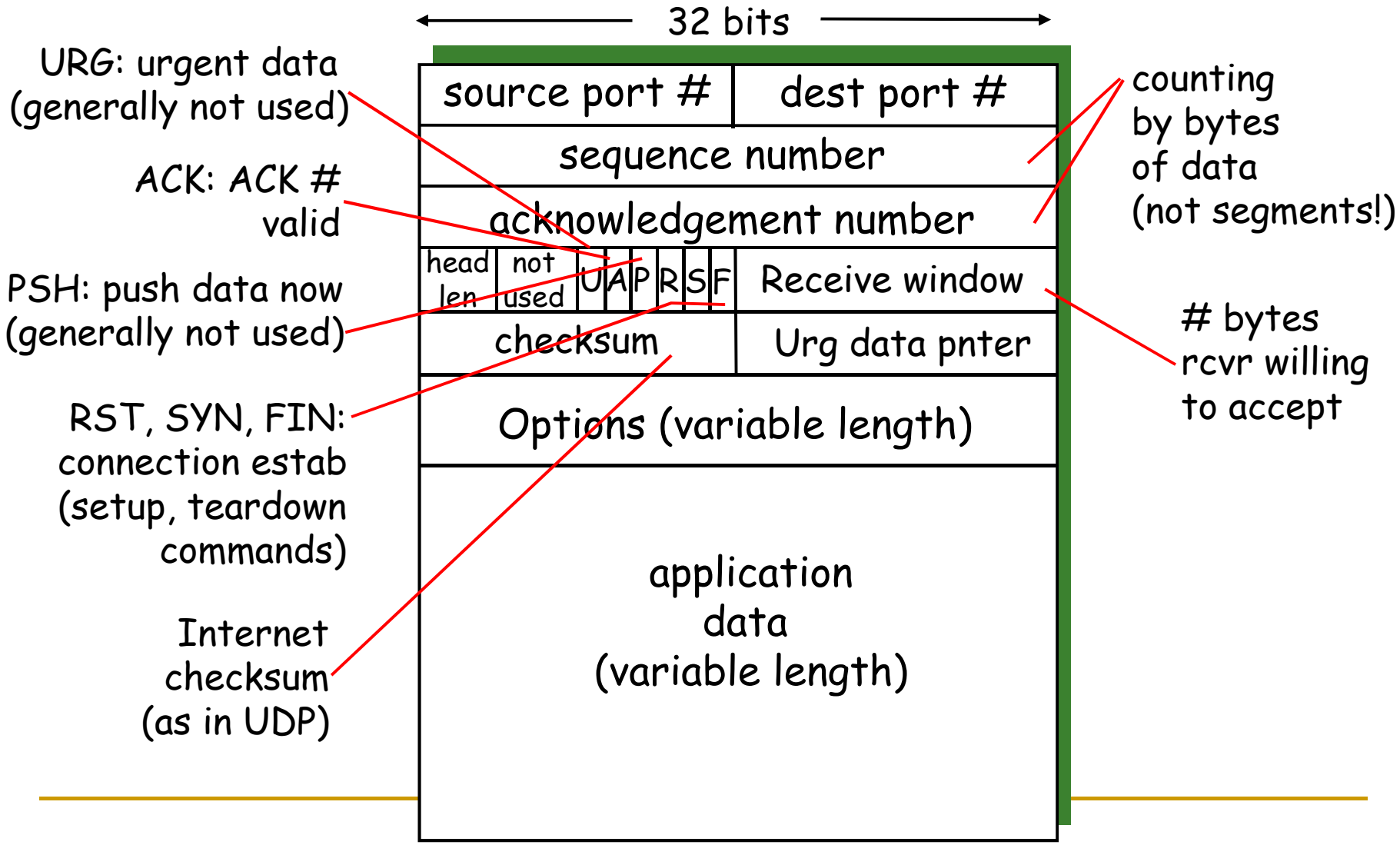
TCP: Overview RFCs: 793, 1122, 1323, 2018, 2581

- point-to-point:
  - one sender, one receiver
- reliable, in-order *byte steam:*
  - no "message boundaries"
- pipelined:
  - TCP congestion and flow control set window size
- *send & receive buffers*

- full duplex data:
  - bi-directional data flow in same connection
  - MSS: maximum segment size
- connection-oriented:
  - handshaking (exchange of control msgs) init's sender, receiver state before data exchange
- flow controlled:
  - sender will not overwhelm receiver

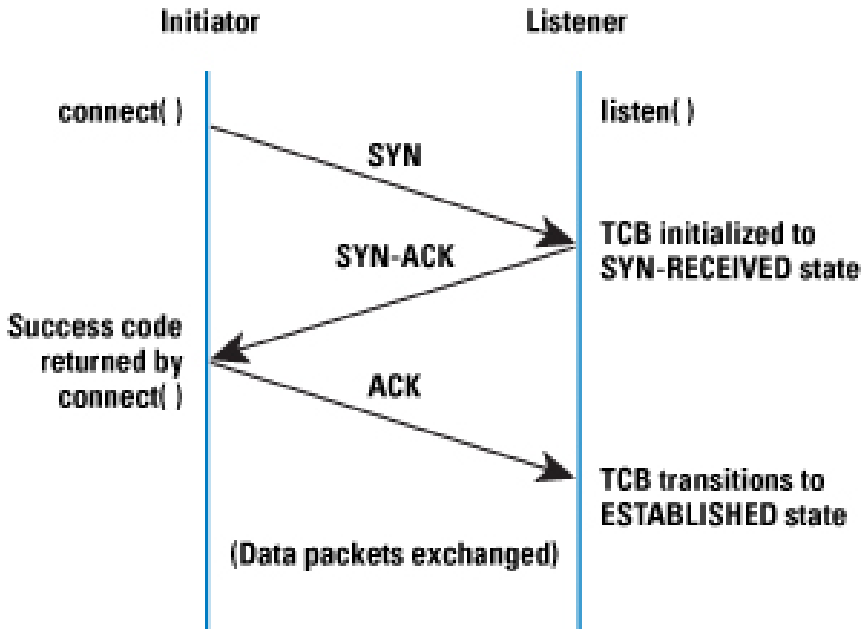


TCP segment structure



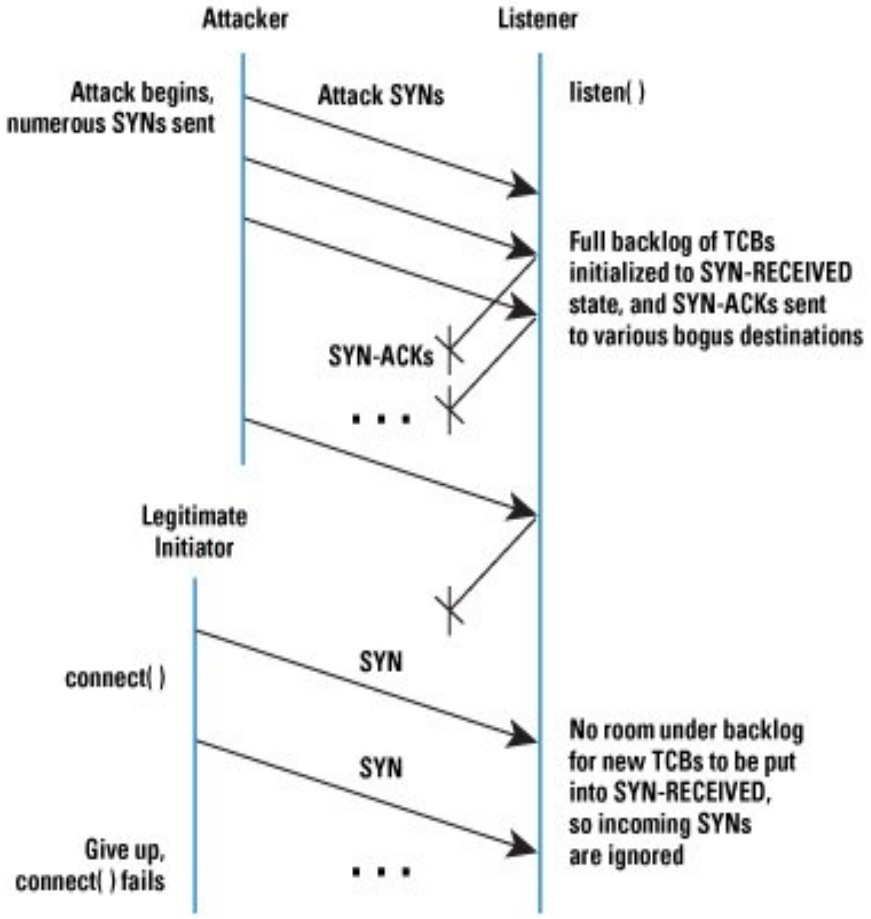
Attack Mechanism

- Transmission Control Block (TCB) is reserved
- TCP SYN-RECEIVED state: connection is half-opened
  - Up on receiving SYN, segment TCB
  - Transited to ESTABLISHED until last ACK



Attack Mechanism

- attacker sends a flood of SYNs ➔ too manyTCB ➔ host is exhauted in memory.
- To avoid this, OS only allows a fixed maximum number of TCBs in SYN-RECEIVED
- If this threshold is reached, new coming SYN will be rejected



TCP Connection Management

Recall: TCP sender, receiver establish "connection" before exchanging data segments

- initialize TCP variables:
  - seq. #s
  - buffers, flow control info (e.g. **`RcvWindow`**)
- *client:* connection initiator
- *server:* contacted by client

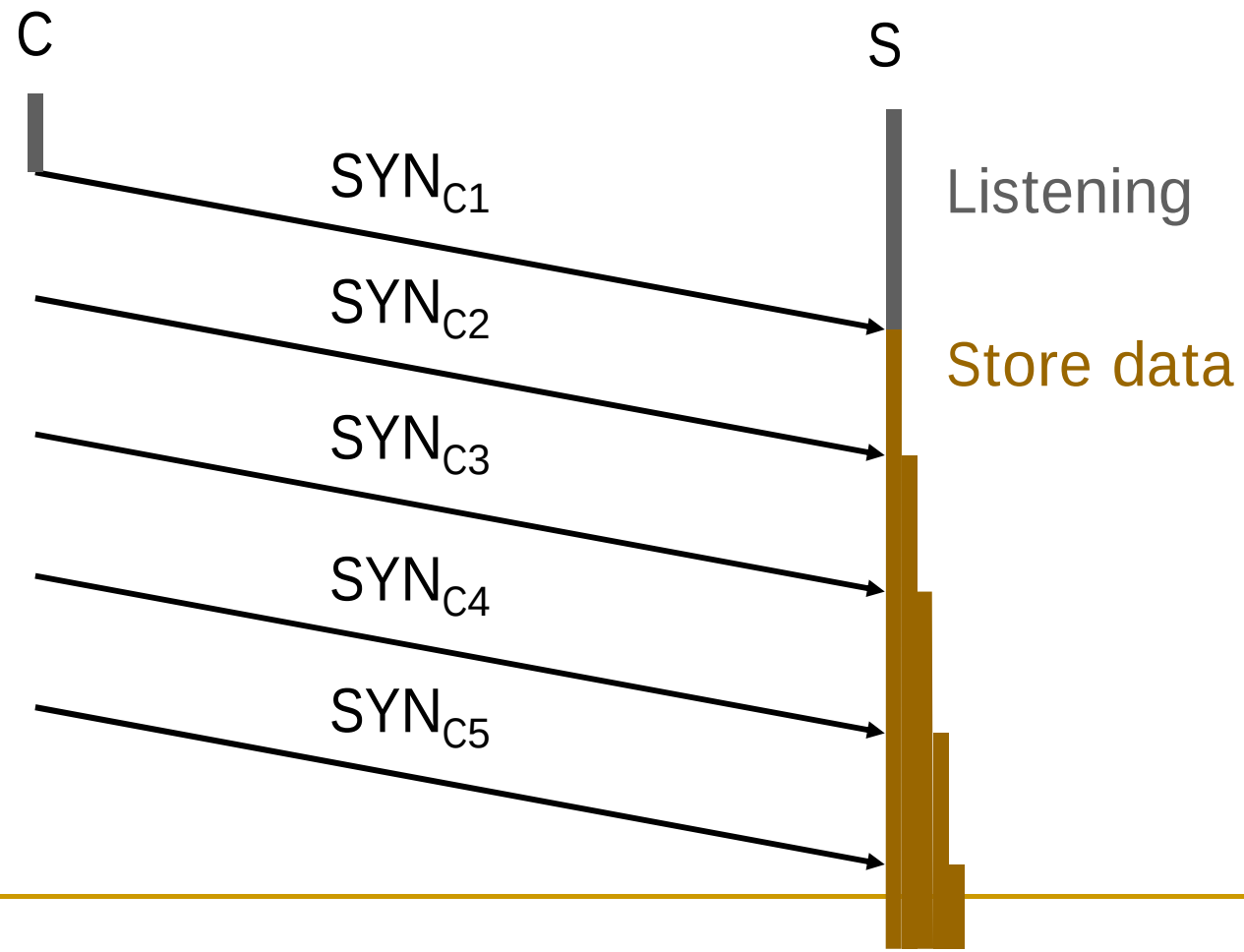
Three way handshake:

Step 1: client host sends TCP SYN segment to server
- specifies initial seq #
- no data

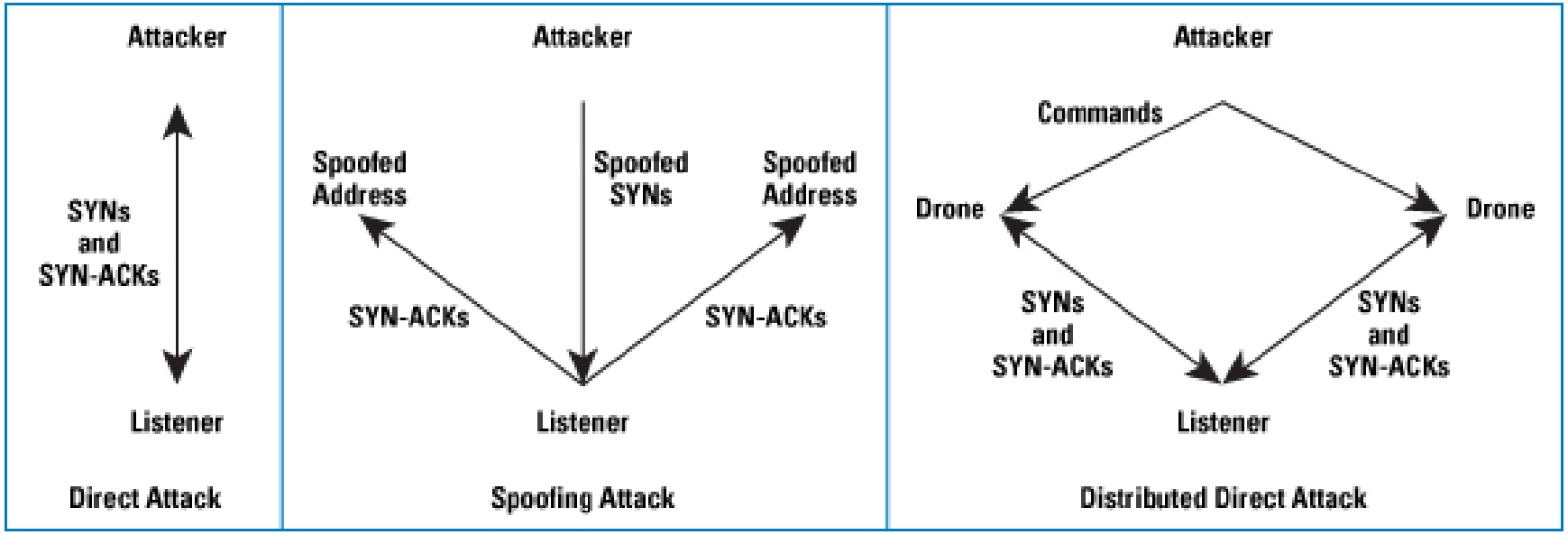
Step 2: server host receives SYN, replies with SYNACK segment
- server allocates buffers
- specifies server initial seq. #

Step 3: client receives SYNACK, replies with ACK segment, which may contain data

SYN Flooding



Implementation Method



How to create a successful flood

- Making drops of incomplete connection (IC)
  - Standard TCP: a connection times out only after some retranmisstion ➔ 511 sec
  - Assuming 1024 ICs are allowed per socket➔ 2 connection attempts per second to exhaust all allocated resources.
  - Note that existing ICs are dropped when a new SYN request is received.
- If an ACK arrives at the server but does not find a corresponding IC state ➔ the server fail to establish such required connection
  - Round trip time (RTT): time required for the server to have the client reply
  - Forcing the server to drop IC state at a rate larger than the RTT, ➔ no connections are able to complete ➔ success in attack!
- The goal of attack is to recycle every connection before the average RTT
  - For a listen queue size of 1024, and a 100 millisecond RTT ➔ need 10,000 packets per second.
  - A minimal size TCP packet is 64 bytes, so the total bandwidth used is only 4Mb/second ➔ practical!

Firewall based Defense

- Examples: SYN Defender, SYN proxying
- Filters packets and requests before router
- Maintains state for each connection

- Drawbacks: can be overloaded, extra delay for processing each packet

Server Based Defense

- Examples: SYN Cache, SYN cookies
- SYN cache:
  - hash table, to partially store states,
  - If the SYN-ACK is "acked" then the connection is established with the server
- SYN cookies
  - do not store states in server but in the network
    - Using a cryptographic function to encode all information into a value that is sent to the client with the SYN,ACK
    - Upon receiving ACK, this value can be extracted then used as a authentic proof of the source machine

SYN kill

- Monitors the network
  - If detects SYNs that are not being acked ➔ automatically generates RST packets to free resources,
    - also it classifies addresses as likely to be spoofed or legitimate…

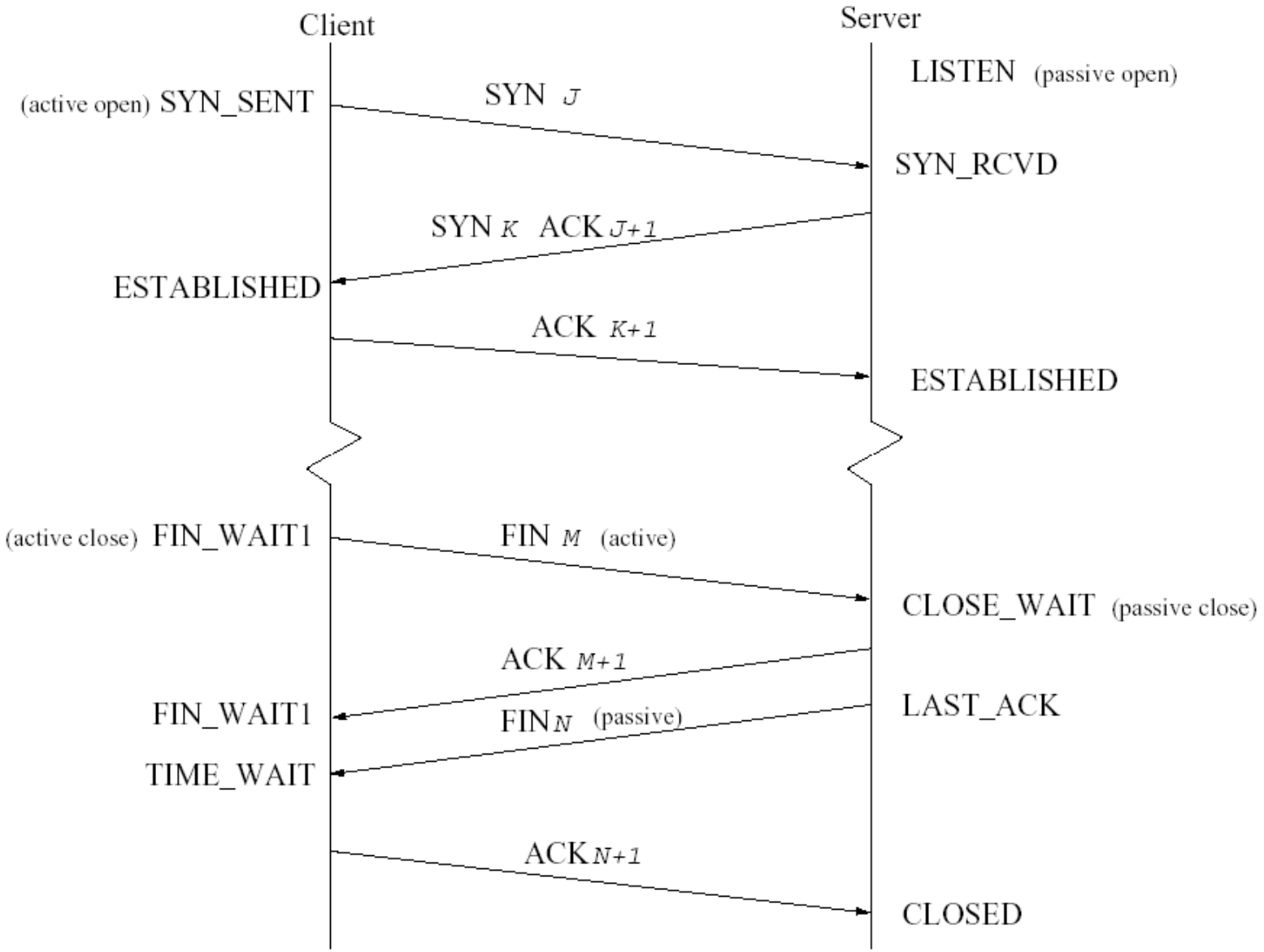
Problems with previous solutions

- Solutions such as SYN cookies, SYN cache, SYN Defender/SYN proxying, SYN kill
  - Knew nothing of source
  - Relied on "costly IP traceback"
  - These solutions are "statefull" so they can be overwhelmed by SYN attacks
    - 14,000SYN/sec can overload

Flood Detection System (FDS)

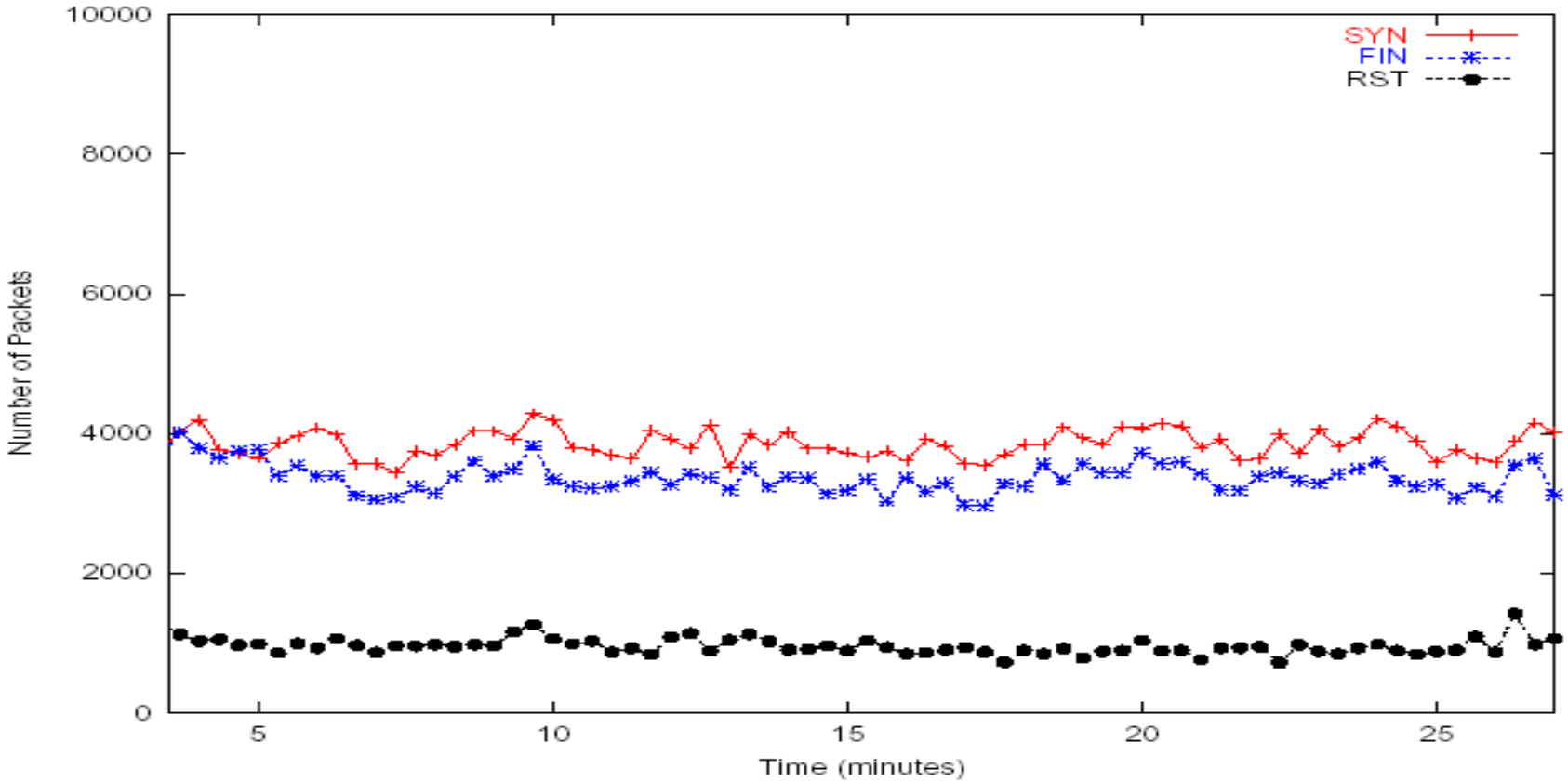
- Stateless, simple, edge (leaf) routers
- Utilize SYN-FIN pair behavior
- Include (SYNACK - FIN) so client or server
- However, RST violates SYN-FIN behavior
- Placement: First/last mile leaf routers
  - First mile – detect large DoS attacker
  - Last mile – detect DDoS attacks that first mile would miss

SYN – FIN Behavior



SYN – FIN Behavior

- Generally every SYN has a FIN



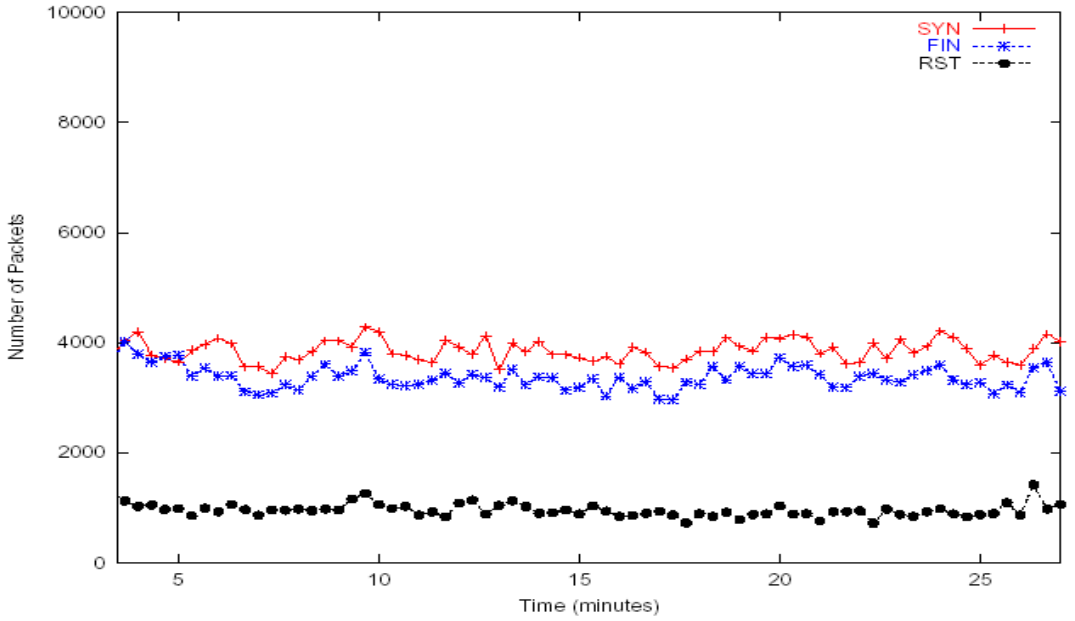
What to do about "RST"

- We can't tell if RST is active or passive
- Consider 75% active
- Active represents a SYN passive does not
- Should balance out to be background noise

RED – SYN

BLUE- FIN

BLACK – RST



Statistical Attack Detection

- There are very many SYN's necessary to accomplish a DoS attack
- At least 500 SYN/sec
- 1400 SYN/sec can overwhelm firewall
- 300,000 SYNs necessary to shut down server for 10 minutes
- So SYN-FIN ratio should be very skewed during an attack

False Positive Possibilities

- Many new online users with long sessions
  - More SYNs coming in than FINs
- A major server is down which would result in 3 SYNs to a FIN
  - Because clients would retransmit the SYN

CUSUM Algorithm

- Finding average number of FINs over a time period, and looking for a time period and testing for statistical homogeneity . If there are significant changes to this, then find when they changed.

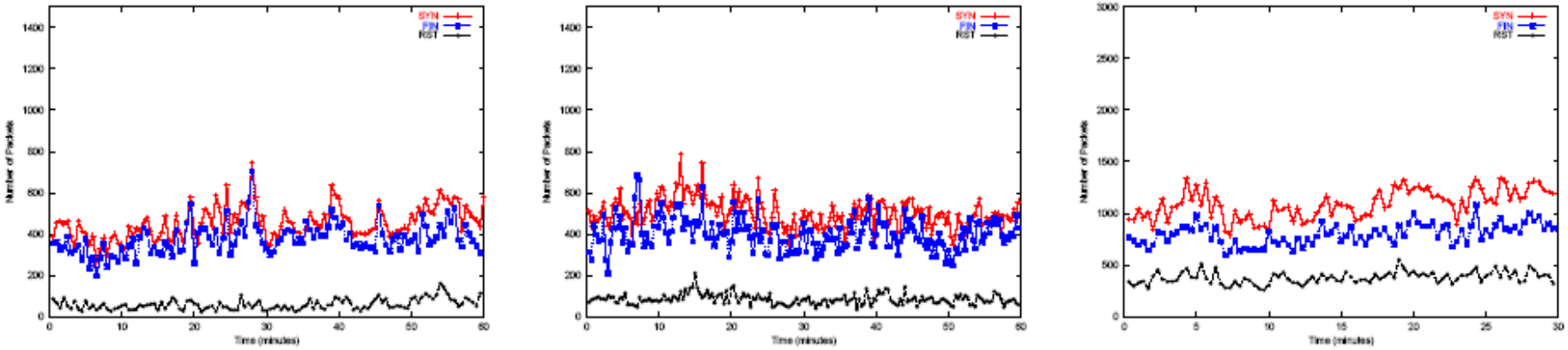
Detection

- The algorithm will result in zero for all normal activity and cumulatively track (i.e. CUSUM = "cumulative sum")

Detection

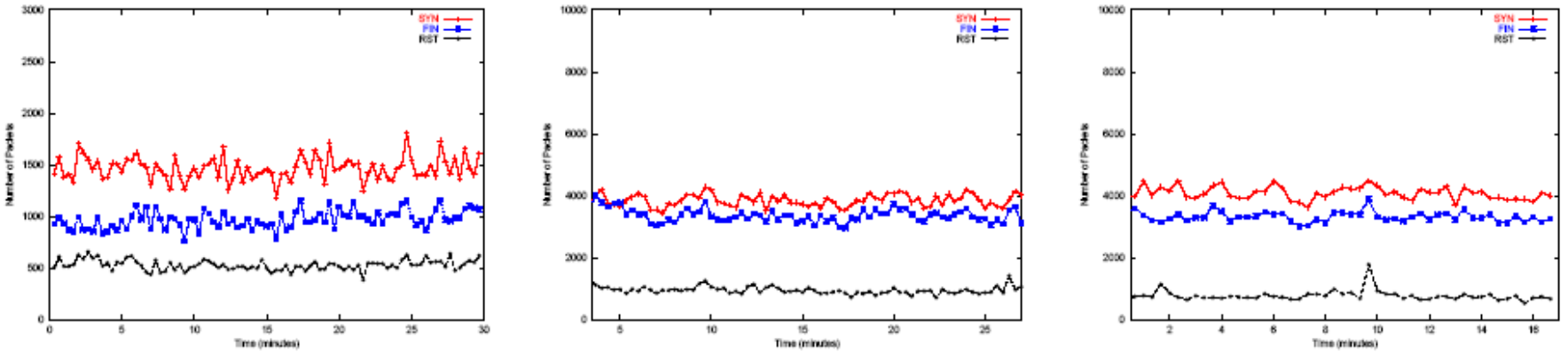
- The internet can be quite dynamic and too complicated for a parametric estimation, so we use sequential testing which requires much less computation
- Two aspects of detection:
  - 1) False alarm time: the time without attacks between unique false alarms
  - 2) Detection time: the detection delay after the attack starts.
- The goal is to minimize the second and maximize the first. However, the conflict and require trade offs

Performance Trends SYN-FIN



(a) DEC-1 (b) DEC-2 (c) Harvard-1

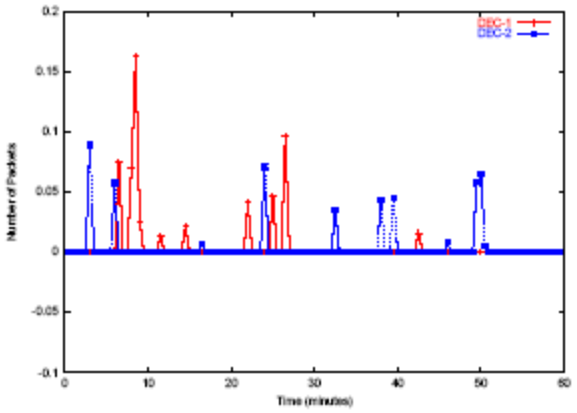
Fig. 4. The dynamics of SYN and FIN (RST) packets (part I)



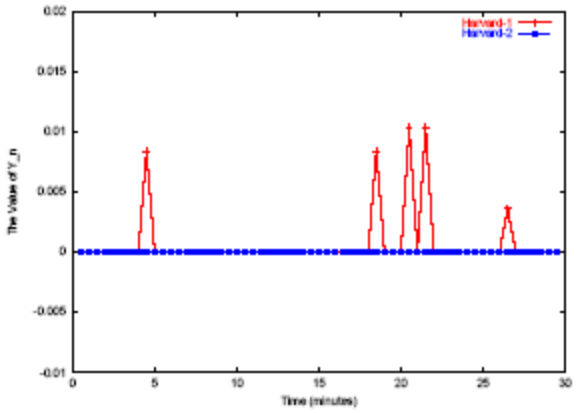
(a) Harvard-2 (b) UNC-in (c) UNC-out

Fig. 5. The dynamics of SYN and FIN (RST) packets (part II)

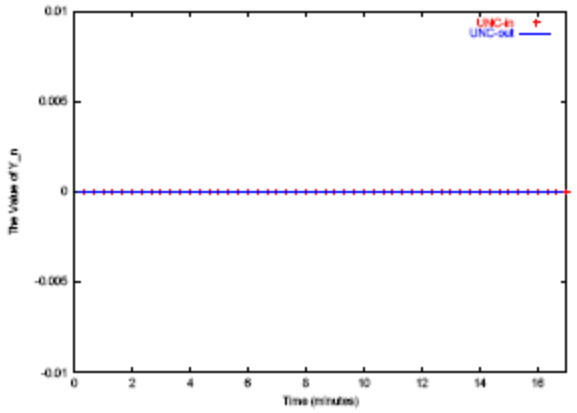
SYN attack vs Normal operation



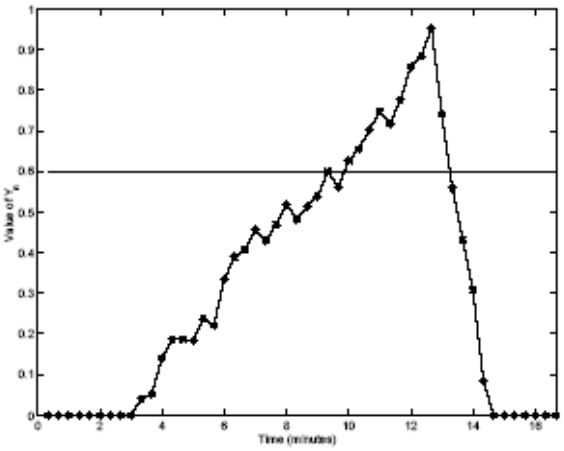
(a) DEC



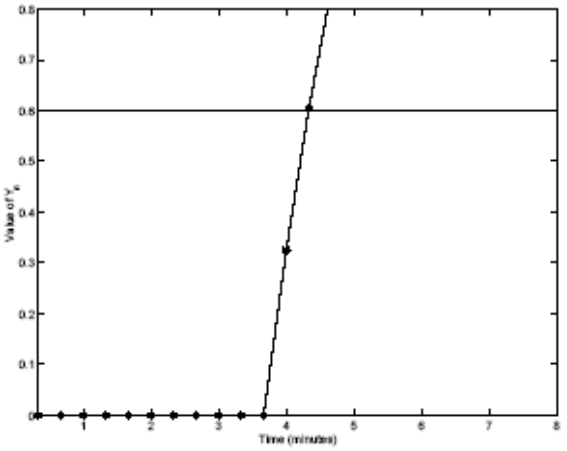
(b) Harvard



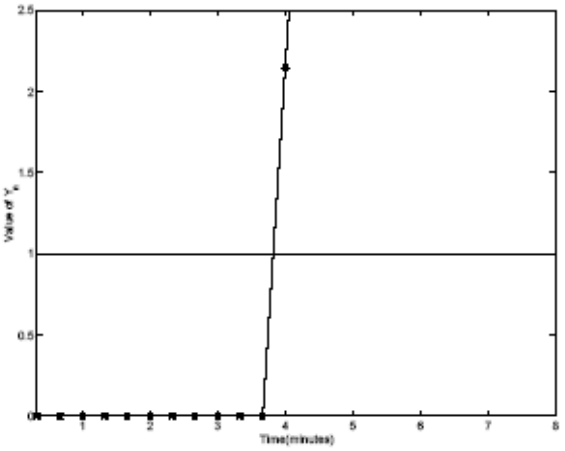
(c) UNC



(a) 35 SYNs per second



(b) 80 SYNs per second



(c) 500 SYNs per second

Fig. 7. SYN flooding detection sensitivity (a) and (b) at first-mile FDS; (c) at last-mile FDS

Sensitivity of Detection

- 500 SYN/sec are required to shut down a server
- It takes a last mile FDS 20 seconds to detect 500 SYN/sec DDoS attack

TABLE II
DETECTION PERFORMANCE OF THE FIRST-MILE FDS

f_i (SYNs/s)	Detection Prob.	Detection Time
33	70%	24.36
35	100%	17.25
40	100%	9.2
50	100%	4.75
60	100%	3.0
70	100%	2.4
80	100%	1.8
90	100%	1.2
100	100%	1.0

Detection is able to

- Distinguish between attacks and background noise
- Detect DDoS with last mile FDS
- Not effected by changes in overall traffic
- Detect attacks within seconds and implement protection.
- Do you think this would always work?
- Can you think of any exceptions??

Đề kiểm tra giữa kỳ

1. Trình bày giao thức Bit-commimment sử dụng hàm băm và cho biết ý nghĩa cụ thể của các giá trị ngẫu nhiên sử dụng trong giao thức. Có thể cải tiến giao thức này để thực hiện "phép tung đồng xu" công bằng giữa 2 người kết nối trực tuyến như thế nào?
2. Thuật toán trao chuyển khóa Diffie-Hellman:

- Trình bày ý tưởng giải pháp
- Trình bày thuật toán với một ví dụ minh họa với q=23.
- Phân tích tấn công Man-in-the-middle và nêu giải pháp phòng tránh

Detecting SYN-Flood using Bloom Filters

SFD-BF

Idea

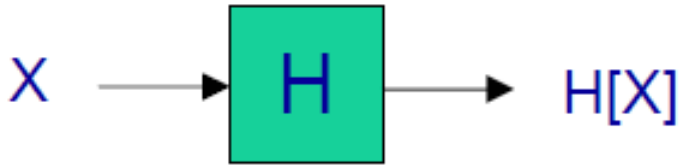
- Improving SFD: using Bloom Filter match FINs against SYN with low error probability
  - For each TCP packets we are interested in this 4-tuple: (source and destination IP, source and destination Port) ➔ Matching FIN against SYNs on this 4-tuple

Hash Function

Input : x

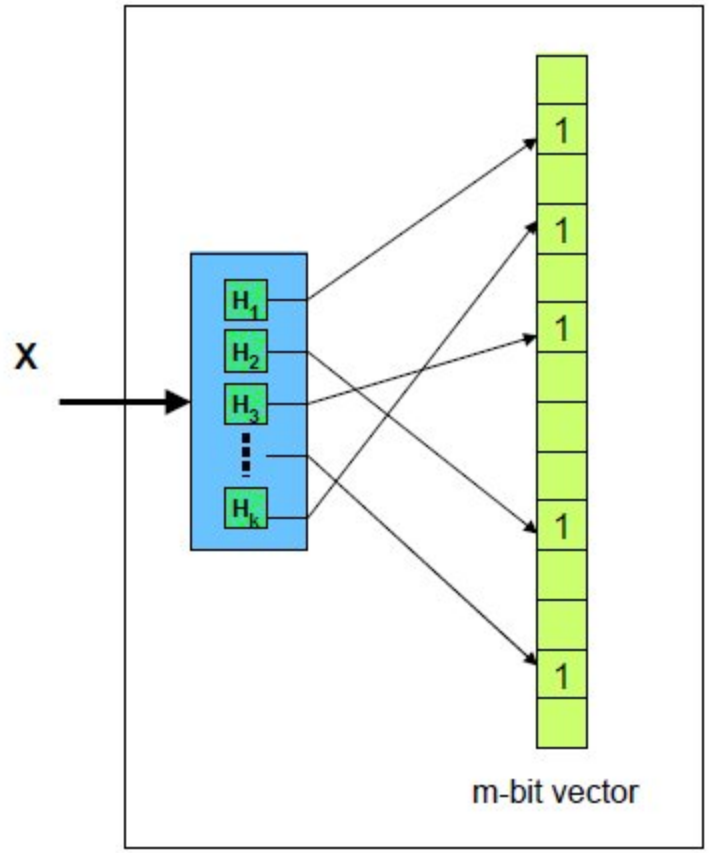
- Output : H[x]
- Properties
  - Each value of x maps to a value of H[x]
  - Typically: Size of (x) >> Size of (H[x])
- Implementation
  - Hash Function

XOR of bits, Shifting, rotates ..

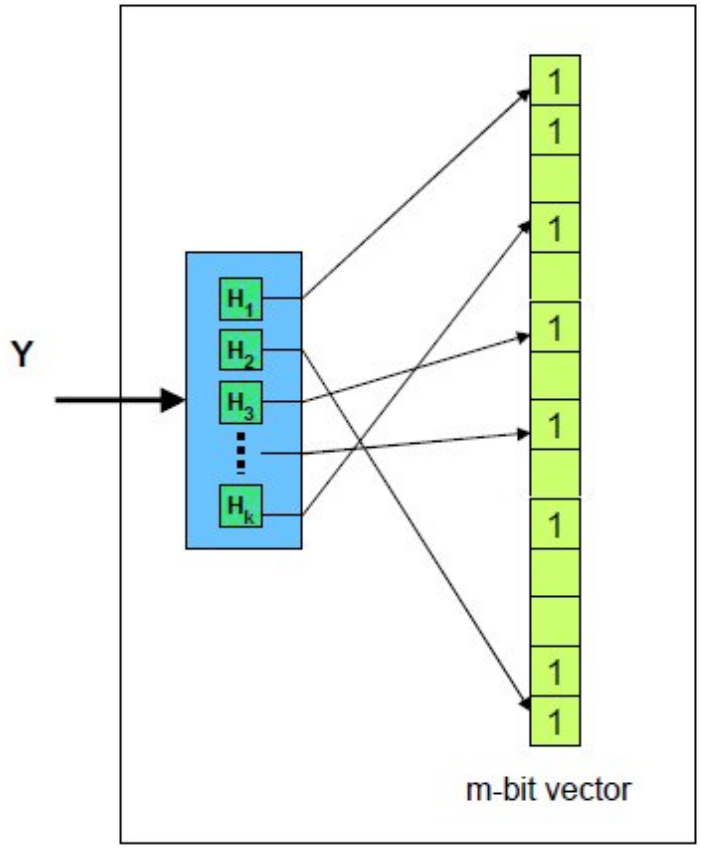


Bloom-Filter (BF)

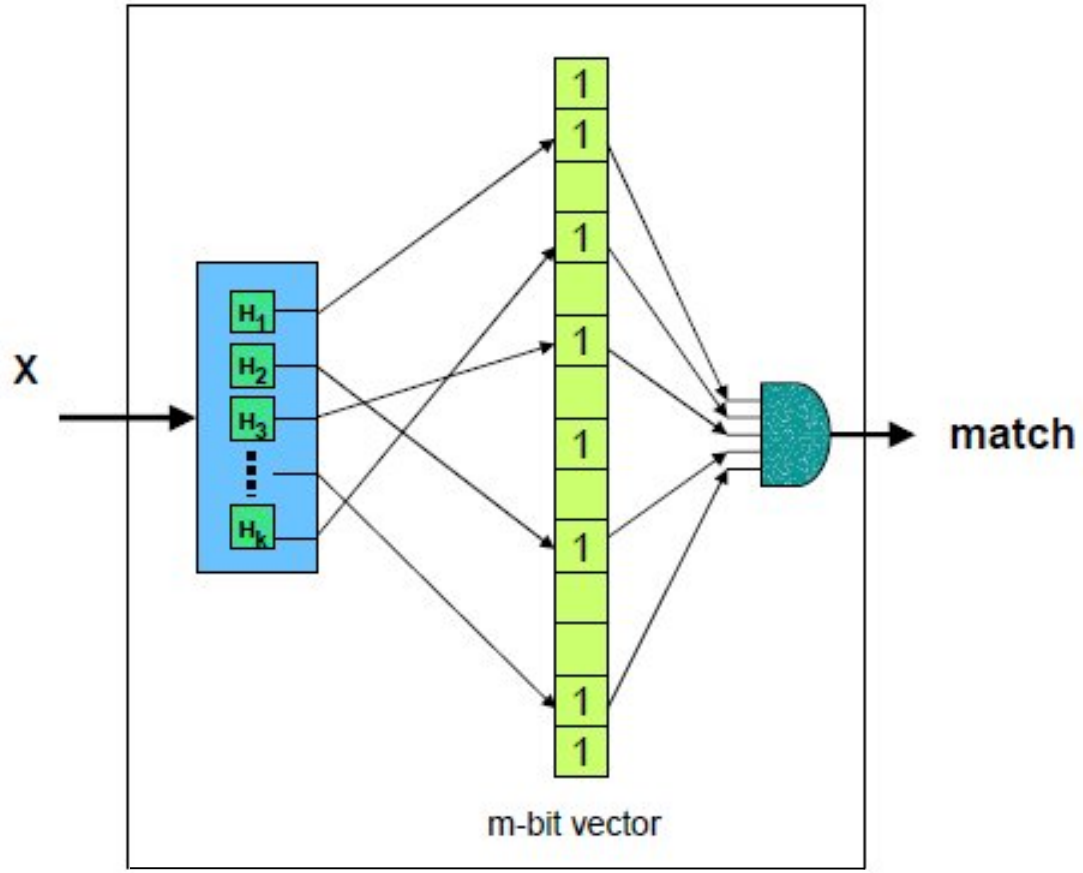
- Bloom Filter uses k hash functions



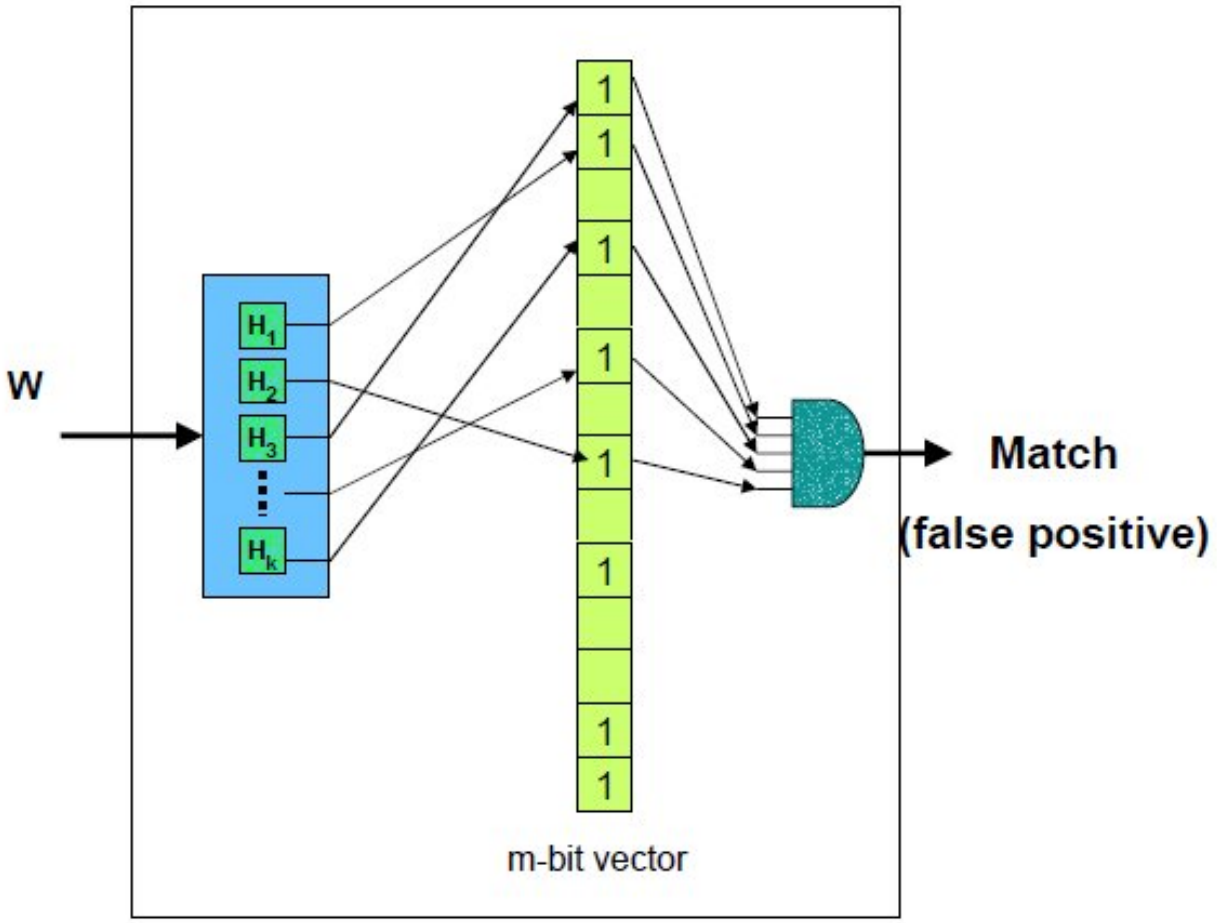
Bloom-Filter (BF)



Querying a Bloom Filter



Querying a Bloom Filter

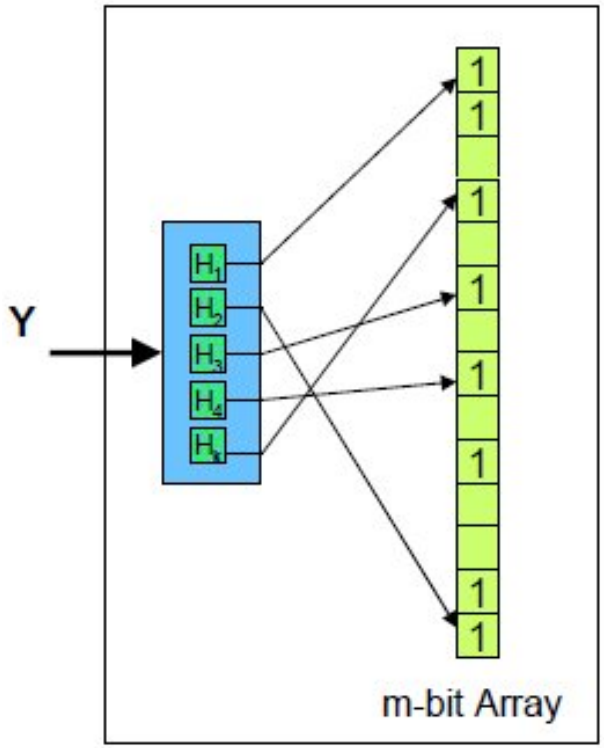


Optimal Parameters of a Bloom filter

- *n : number of Item to be stored*
- *k : number of hash functions*
- *m : the size of the bit-array (memory)*
- The false positive probability

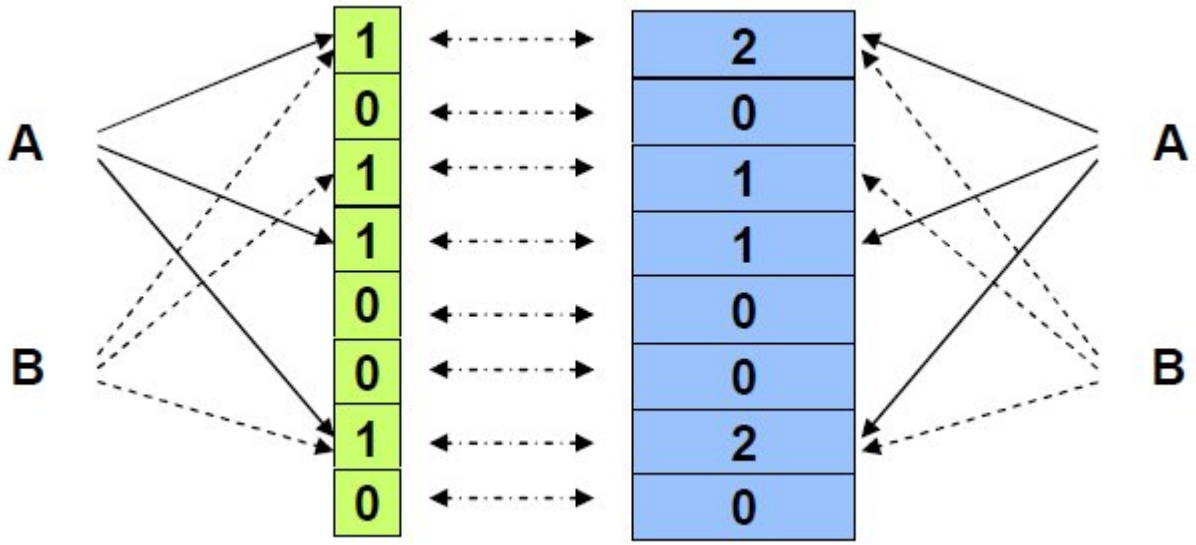
  $f = (\frac{1}{2})^k$
- The optimal value of hash functions, k,is:

  $k = \ln 2 \times m/n = 0.693 \times m/n$



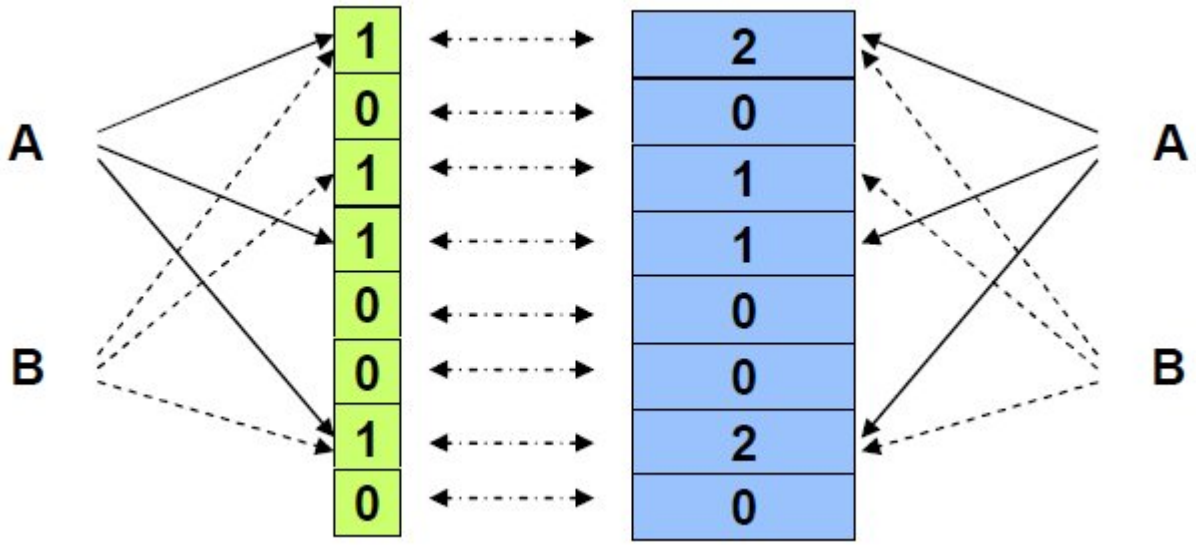
Counting Bloom Filters

- Extension: the array of m bits becomes the array of m small counters.
  - When insert or delete a value x from BF, we simply increase or decrease the corresponding counter



Counting Bloom Filters

- For small false positive probability, usually set m/n=16➔ FSP is at most 0,000459
- Also give counters of 8 bits.
- When the number of counters are being used >= m/2 ➔ reset BF to improve for smaller FSP



SFD - Method

1- Classification of packets

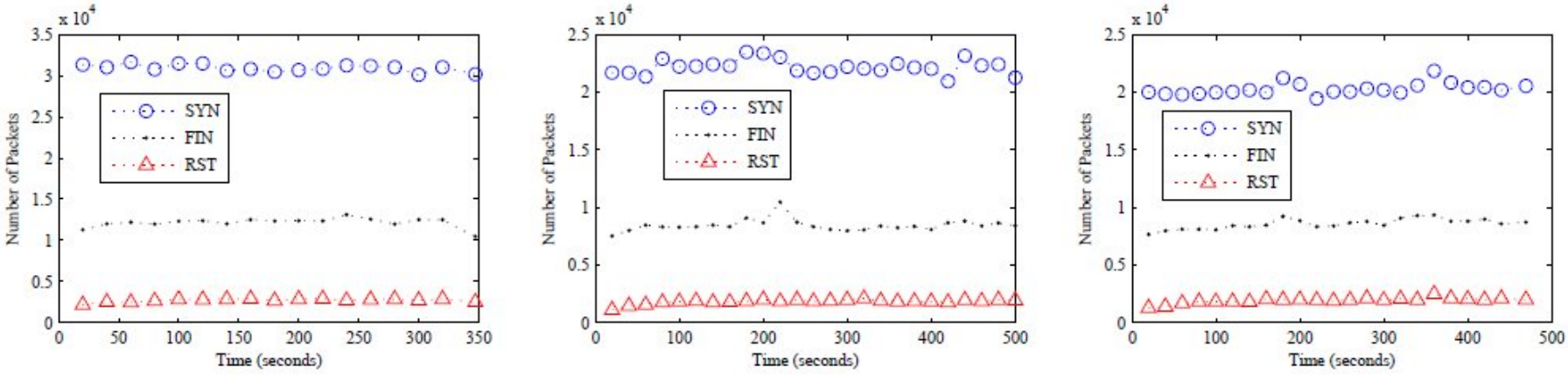
2-Computing the # of SYN and FIN packets going through

3-Using algorithm CUSUM to analyze the (SYN-FIN) pair behaviour

SFD-BF Method

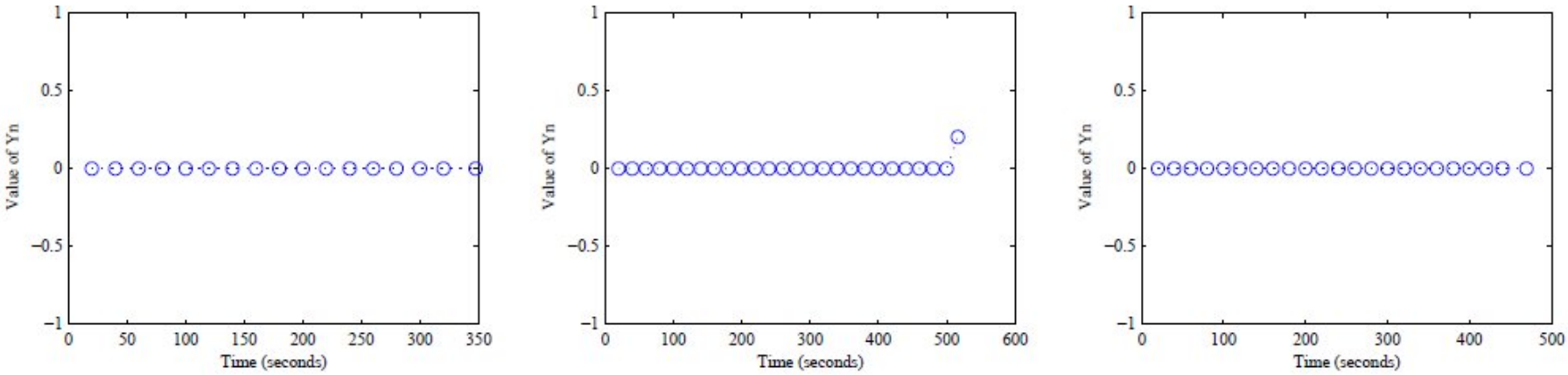
- Improvement on previous SFD:
  - Compute the difference between #SYN and #FIN when the packets are matched on the 4-tuple:
    - When a SYN packet comes, determine the corresponding 4-tuple and insert this into BF. Increase the counter specified by this 4-tuple.
    - When a FIN/RST packet comes: determines the 4-tuples and find it's hash in BF to decrease the corresponding counter

Result



(a) THU-1 (b) THU-2 (c) THU-3

Fig. 2. The dynamics of *valid SYN and FIN packets*.



(a) THU-1 (b) THU-2 (c) THU-3

Result

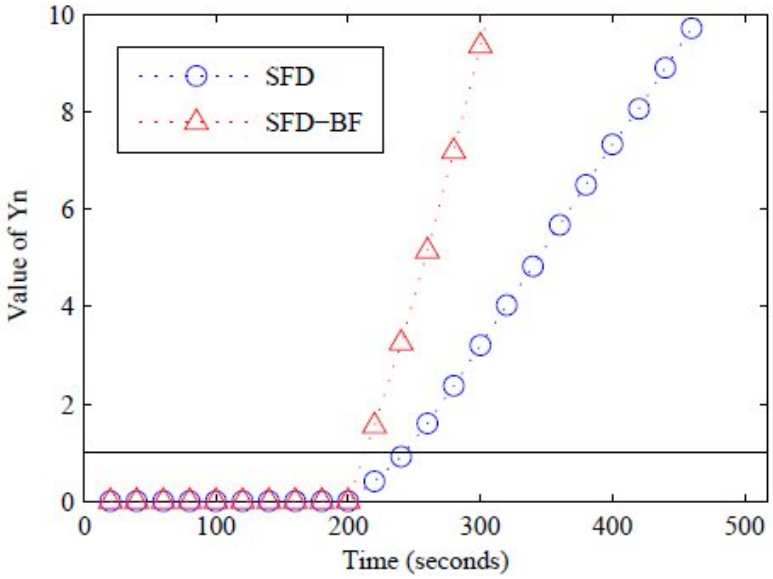


Fig. 4. SYN flooding detection under simple attacks.

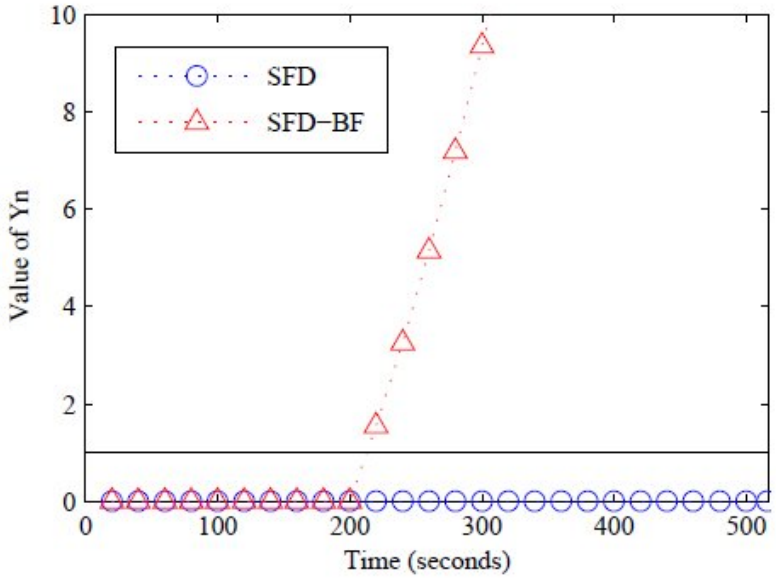


Fig. 5. SYN flooding detection under complex attacks.

Intentional Dropping Scheme for SYN Flooding Mitigation

Idea

- Normally, if it does not receive a SYN-ACK after sending a SYN for a certain time a client machine then would resend another SYN until it gets connected to the wanted server.
- The idea of this method is to drop all the first SYN from all the source machine, which would help to reduce SYN flood which is usually first SYNs with spoofed addresses

Method

- The solution is to propose using 3 different BFs:
  - BF1: stores the 4-tuple address of the first SYN coming from a given source
  - BF2: stores the 4-tupple of all SYNs, with which the 3-way handshake is already completed
  - BF-3: Store the 4-tupple of other SYNs.

Method

Once a SYN arrives, its 4-tuple address is checked against the 3 BFs, where occurs 1 of the 3 following cases:

- 1. Not in any BF➔ This is the first SYN then will be dropped, also insert the 4-tuple into BF1
- 2. If found in BF-1➔ this is a second SYN we just move the 4-tuple from BF1 to BF3
- 3. If in BF-2 ➔ Let it go through.
- 4. If in BF-3 ➔ let it goes through with probability p=1/n, where n is the value of corresponding counter in BF-3

Method

When an ACK comes, its 4-tupple address is checked against the BFs, which may results in 1 of 3 following cases"

1. Not in any BF ➔ drop the packet
2. If it matches one in BF-2 ➔ let it through
3. If in BF-3 ➔ the connection is completed ➔ move the 4-tuple address from BF3 to BF-2

Result

- First SYN from any source will be dropped
- The second SYN from the same source will go through
- If this same source continue sending SYNs, the probability that the SYN numbered n is allowed to go through is 1/n

➔ Thus, the SYN flood caused by an attacking source will be mitigated.