IJETST- Vol.||03||Issue||06||Pages 4119-4131||June||ISSN 2348-9480 2016

Open access Journal International Journal of Emerging Trends in Science and Technology

Impact Factor: 2.838 DOI: http://dx.doi.org/10.18535/ijetst/v3i06.13

Passive IP Traceback: Tracing spoofer IP and Blocking IP spoofer from Data access using Greedy Algorithm

Authors

Mrs KarpagaPriya.P[1], Mrs Selvi.U[2]

[1,2] Department of Computer Science and Engineering, Professional Group of Institutions, K.N. Puram, Coimbatore - Trichy Road, Palladam, Tamil Nadu 641662, India

Email: [1]*karpagapriyaselvam.p@gmail.com*, [2]*slvunnikrishnan@gmail.com*

ABSTRACT

IP traceback can be used to find the origin of anonymous traffic; however, Internet-scale IP traceback systems have not been deployed due to a need for cooperation between Internet Service Providers (ISPs). This article presents an Internet-scale Passive IP Trackback (PIT) mechanism that does not require ISP deployment. PIT analyzes the ICMP messages that may scattered to a network telescope as spoofed packets travel from attacker to victim. An Internet route model is then used to help re-construct the attack path. Applying this mechanism to data collected by Cooperative Association for Internet Data Analysis (CAIDA), we found PIT can construct a trace tree from at least one intermediate router in 55.4% the fiercest packet spoofing attacks, and can construct a tree from at least 10 routers in 23.4% of attacks. This initial result shows PIT is a promising mechanism.

Categories and Subject Descriptors: C.2.0 [Computer Communication Networks]: General– Security and protection.

Keywords: *PIT, Network Telescope, DOS attack, ICMP messages, Greedy algorithm, CAIDA*

1. INTRODUCTION

1.1 BACKGROUND

The objective of IP traceback is to find the origin of spoofing traffic. If the origin of spoofing traffic is found, the attacker can be deterred from launching further attacks. Most IP traceback approaches trace the spoofed traffic to the edge of region where traceback is deployed. Unfortunately, the non-cooperation nature of Internet Service Providers (ISPs) means IP traceback approaches are only be deployed in a domain controlled by the single ISP, and can only trace to the edge of this domain. Most existing approaches to this problem have been tailored toward DoS attack detection. IP traceback approach uses

1 Probabilistic packet marking
2 Deterministic packet marking
3 Router-based approach
4 Out-of-band approaches
5 Trace-back of active attack flows

Network Telescope is an Internet system that allows one to observe traffic targeting the dark (unused) address-space of the network. Possible network attacks are random scanning worms, and DDoS backscatter. The resolution of the Internet telescope is dependent on the number of IP addresses it monitors. For example, a large Internet telescope that monitors traffic to 16,777,216 addresses (a /8 Internet telescope in IPv4), has a higher probability of observing a relatively small event than a smaller telescope that monitors 65,536 addresses (a /16 Internet telescope).The UCSD network telescope (aka a black hole, an Internet sink, dark space, or a darknet) is a globally routed /8 network (approximately 1/256th of all IPv4 Int ernet addresses) .

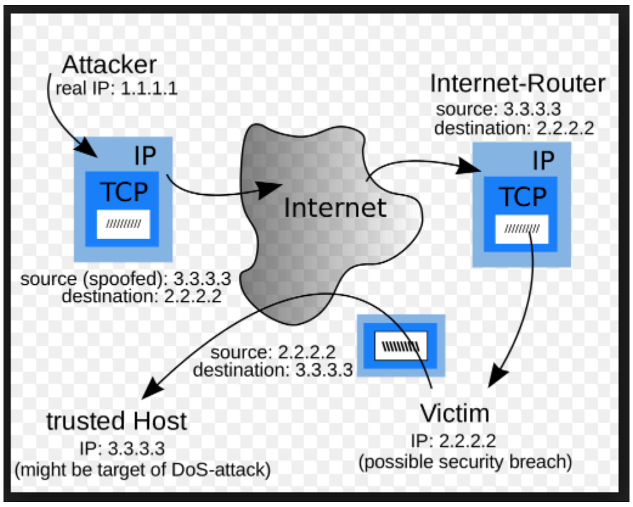


Fig 1: How IP spoofing works

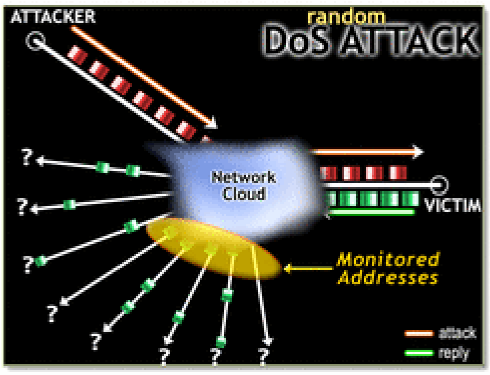


Fig 2: Because the network telescope composes 1/256th of the IPv4 address space, the telescope receives approximately 1/256th of the responses to spoofed packets generated by the denial-of-service attack victim.

CAIDA monitors traffic directed toward any one of a large block of IP (Internet protocol) addresses at the University of California at San Diego, a block so big that it makes up some 0.4% of the world's addresses.

SPF-Sender Policy Framework (SPF) is a simple email-validation system designed to detect email spoofing by providing a mechanism to allow receiving mail exchangers to check that incoming mail from a domain comes from a host authorized by that domain's administrators.[1] The list of authorized sending hosts for a domain is published in the Domain Name System (DNS) records for that domain in the form of a specially formatted TXT record.

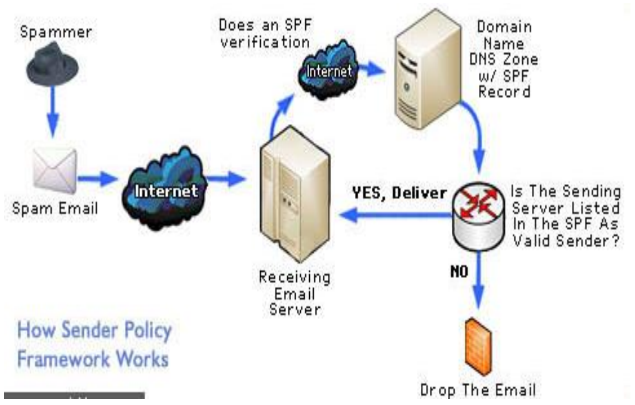

Spammer
Spam Email
Internet
Does an SPF verification
Internet
Domain Name DNS Zone w/ SPF Record
Receiving Email Server
YES, Deliver
Is The Sending Server Listed In The SPF As Valid Sender?
NO
Drop The Email
How Sender Policy Framework Works

Fig 3 : SPF working

1.2 MOTIVATION

The IP traceback approaches has various advantages and disadvantages. On observing the flaws of PIT we think about timing over networking. Those approaches are

1) Probabilistic packet marketing

The first approach is to XOR each node forming an edge in the path with each other. Node *a* inserts its IP address into the packet and sends it to *b*. Upon being detected at *b* (by detecting a 0 in the distance), *b* XORs its address with the address of *a*. This new data entity is called an edge id and reduces the required state for edge sampling by half. Their next approach is to further take this edge id and fragment it into *k* smaller fragments. Then, randomly select a fragment and encode it, along with the fragment offset so that the correct corresponding fragment is selected from a downstream router for processing. When enough packets are received, the victim can reconstruct all of the edges the series of packets traversed (even in the presence of multiple attackers).Due to the high number of combinations required to rebuild a fragmented edge id, the reconstruction of such an attack graph is computationally intensive according to research by Song and Perrig. Furthermore, the approach results in a large number of false positives. As an example, with only 25 attacking hosts in a DDoS attack the reconstruction process takes days to build and results in thousands of false positives.

2) Deterministic packet marketing

This describes a more realistic topology for the Internet – that is composed of LANs and ASs with a connective boundary – and attempt to put a single mark on inbound packets at the point of network ingress. Their idea is to put, with random probability of .5, the upper or lower half of the IP address of the ingress interface into the fragment id field of the packet, and then set a reserve bit indicating which portion of the address is contained in the fragment field. By using this approach they claim to be able to obtain 0 false positives with .99 probability after only 7 packets.

3) Router-based approach

The space needed at each router is limited and controllable (2n bits). A small *n* makes the probability of collision of packet hashes (and false identification) higher. When a packet is to be traced back, it is forwarded to originating routers where fingerprint matches are checked. As time passes, the fingerprint information is "clobbered" by hashes generated by other packets. Thus, the selectivity of this approach degrades with the time that has passed between the passage of the packet and the traceback interrogation.[7]

Another known take on the router-based schemes comes from Hazeyama et al. In their approach, they wish to integrate the SPIE approach as outlined by Snoeren,[7] with their approach of recording the layer 2 link-id along with the network ID (VLAN or true ID), the MAC address of the layer 2 switch that received the packet and the link id it came in on. This information is then put into two look-up tables – both containing the switch (layer 2 router) MAC id for look-up. They rely on the MAC:port tuple as a method of tracing a packet back (even if the MAC address has been spoofed)

To help mitigate the problem of storage limitations they use Snoeren's hashing approach and implementation (SPIE) – modifying it to accept their information for hashing. They admit their algorithm is slow (O(N2)) and with only 3.3 million packet hashes being stored the approximate time before the digest tables are invalid is 1 minute. This dictates that any attack response must be real-time – a possibility only on single-administrative LAN domains.

4) Out of bound approach

The ICMP traceback scheme proposes probabilistically sending an ICMP traceback packet forward to the destination host of an IP packet with some low probability. Thus, the need to maintain state in either the packet or the router is obviated. Furthermore, the low probability keeps the processing overhead as well as the bandwidth requirement low. The selection also be based on pseudo-random numbers to help block attempts to time attack bursts. The problem with this approach is that routers commonly block ICMP messages because of security issues associated with them.

5) Trace-back of active attack flows

An observer tracks an existing attack flow by examining incoming and outgoing ports on routers starting from the host under attack. Thus, such a solution requires having privileged access to routers along the attack path.

To bypass this restriction and automate this process, Stone proposes routing suspicious packets on an overlay network using ISP edge routers. By simplifying the topology, suspicious packets can easily be re-routed to a specialized network for further analysis. This is an interesting approach. By nature of DoS, any such attack will be sufficiently long lived for tracking in such a fashion to be possible. Layer-three topology changes, while hard to mask to a determined attacker, have the possibility of alleviating the DoS until the routing change is discovered and subsequently adapted to. Once the attacker has adapted, the re-routing scheme can once again adapt and re-route; causing an oscillation in the DoS attack; granting some ability to absorb the of such an attack.

1.3 OUR WORK

In this article we illustrate the generation, types, collection and the security issues of Greedy algorithm in section II. Then in section III we use greedy algorithm to improve the Computation speed of network routing, so that we can block IP spoofer from accessing data between login time and usage of a session. We use timing properties of networking like Accurate, Minimum over head, TTL exceeding, Available bandwidth estimation in high-speed wired networks to avoid IP spoofing. Because the routers can be close to the spoofers, the path bckscatter messages may potentially disclose the location of the spoofers. PIT exploits these path backscatter messages to find the location of the soofers. With the location of the spooferstom be known, the victim can seek help from the corresponding ISP to filter out the attacking packets, or take other counterattacks. PIT is especially useful for the victims in reflection based spoofing attacks. e.g., DNS amplification attacks. The victimscan find the locations of spoofers directly from the attacking traffic.

An IP TTL is set initially by the system sending the packet. It can be set to any value between 1 and 255; different operating systems set different defaults. Each router that receives the packet subtracts at least 1 from the count; if the count remains greater than 0, the router forwards the packet, otherwise it discards it and sends an Internet Control Message Protocol (ICMP) message back to the originating host, which may trigger a resend. Trace route sends a stream of packets with successively higher TTLs so each will be discarded in turn by the next hop. The time between sending the packet and receiving back the ICMP message that it was discarded is used to calculate each successive hop travel time.

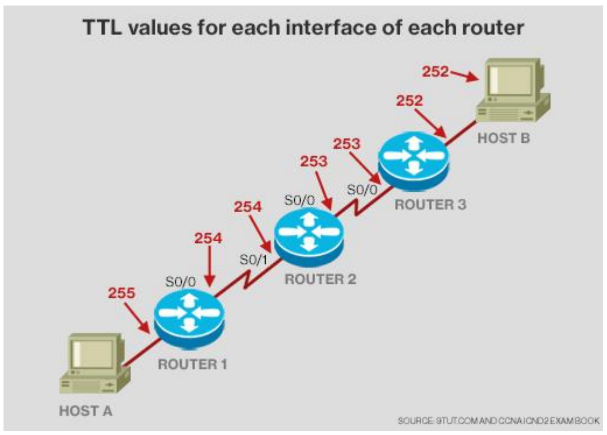

TTL values for each interface of each router
252
252
253
253
254
254
255
S0/0
S0/1
S0/0
S0/0
HOST B
ROUTER 3
ROUTER 2
ROUTER 1
HOST A
SOURCE: 9TUT.COM AND CCNA ICND2 EXAM BOOK

Fig 4: Time To Live ping detail

In IP multicast, the TTL controls the scope or range in which a packet may be forwarded.

Use of TTL field in IP header:

- 0 is restricted to the same host
- 1 is restricted to the same subnet
- 32 is restricted to the same site
- 64 is restricted to the same region
- 128 is restricted to the same continent
- 255 is unrestricted

2. OBSERVATION

Various attacks and measures on network are:

1) DOS attack

A **denial-of-service** (**DoS**) **attack** is an attempt to make a machine or network resource unavailable to its intended users, such as to temporarily or indefinitely interrupt or suspend services of a host connected to the Internet. A **distributed denial-of-service** (**DDoS**) is where the attack source is more than one, often thousands of, unique IP addresses. It is analogous to a group of people crowding the entry door or gate to a shop or business, and not letting legitimate parties enter into the shop or business, disrupting normal operations.

Criminal perpetrators of DoS attacks often target sites or services hosted on high-profile web servers such as banks, credit card payment gateways; but motives of revenge, blackmail or activism can be behind other attacks.

(1) HTTP POST DoS attack

First discovered in 2009, the HTTP POST attack sends a complete, legitimate HTTP POST header, which includes a 'Content-Length' field to specify the size of the message body to follow. However, the attacker then proceeds to send the actual message body at an extremely slow rate (e.g. 1 byte/110 seconds). Due to the entire message being correct and complete, the target server will attempt to obey the 'Content-Length' field in the header, and wait for the entire body of the message to be transmitted, which can take a very long time. The attacker establishes hundreds or even thousands of such connections, until all resources for incoming connections on the server (the victim) are used up, hence making any further (including legitimate) connections impossible until all data has been sent. It is notable that unlike many other (D)DoS attacks, which try to subdue the server by overloading its' network or CPU, a HTTP POST attack targets the *logical* resources of the victim, which means the victim would still have enough network bandwidth and processing power to operate. Further combined with the fact that Apache will, by default, accept requests up to 2GB in size, this attack can be particularly powerful. HTTP POST attacks are difficult to differentiate from legitimate connections, and are therefore able to bypass some protection systems. OWASP, an open source web application sec urity project, has released a testing tool to test the security of servers against this type of attacks.

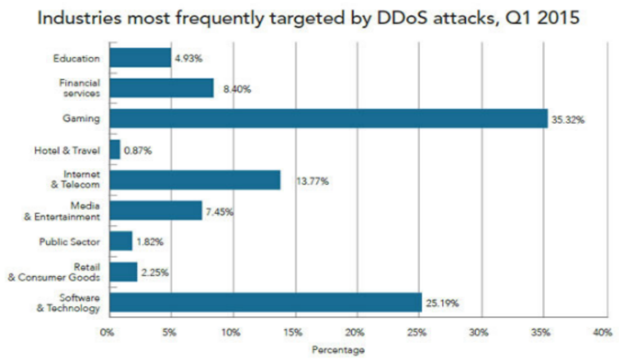


Fig 5: Q1 report on Dos attack

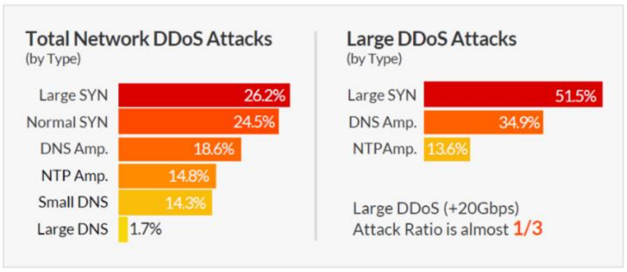


Fig 6: Over 20Gbps DDoS attacks now become common for Hackers

(2) DDS based defense

More focused on the problem than IPS, a DoS defense system (DDS) can block connection-based DoS attacks and those with legitimate content but bad intent. A DDS can also address both protocol attacks (such as teardrop and ping of death) and rate-based attacks (such as ICMP floods and SYN floods).

(3) Backscatter

In computer network security, backscatter is a side-effect of a spoofed denial-of-service attack. In this kind of attack, the attacker spoofs (or forges) the source address in IP packets sent to the victim. In general, the victim machine cannot distinguish between the spoofed packets and legitimate packets, so the victim responds to the spoofed packets as it normally would. These response packets are known as backscatter.

If the attacker is spoofing source addresses randomly, the backscatter response packets from the victim will be sent back to random destinations. This effect can be used by network telescopes as indirect evidence of such attacks. The term "backscatter analysis" refers to observing backscatter packets arriving at a statistically significant portion of the IP address space to determine characteristics of DoS attacks and victims.

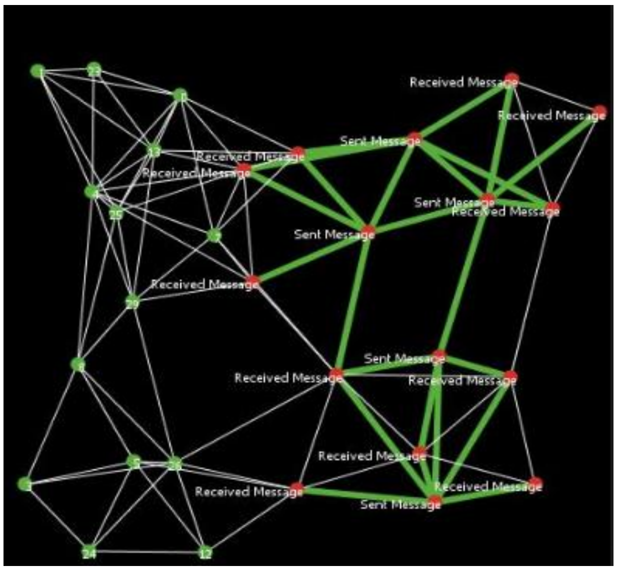


Fig 7: Flooding in Wireless Sensor Network

3. Evaluation

3.1 Path Backscatter

The "Virus Bounce Rule set" is a Spam Assassin rule set to catch "backscatter". Backscatter is mail we didn't ask to receive, generated by legitimate, non-spam-sending systems in response to spam.

- Misdirected virus/worm "OMG your mail was infected!" email notifications from virus scanners.

How do I block it?

There's a ruleset to block virus-blowback bounce messages which is included in Spam Assassin 3.2.0. It provides the following rules:

- __MY_SERVERS_FOUND: a whitelisted relay a la "whitelist_bounce_relays" was found
- BOUNCE_MESSAGE: an MTA-generated bounce from a non-whitelisted relay, "message was undeliverable" etc.
- CRBOUNCE_MESSAGE: Challenge-response bounce message from a non-whitelisted relay, eg. "please confirm your message was not spam"
- VBOUNCE_MESSAGE: a virus-scanner-generated bounce from a non-whitelisted relay, e.g. "You sent a virus"

* ANY_BOUNCE_MESSAGE: any of the *BOUNCE_MESSAGE types above will also trigger this

__MY_SERVERS_FOUND inhibits the other 4 rules from firing.

```
whitelist_bounce_relays myrelay.mydomain.net
spamassassin -Lt < sample-vbounce.txt
[...]
Content analysis details:   (2.6 points, 5.0 required)

 pts rule name              description
---- ---------------------- ----------------------------------
---------------
 0.0 NO_REAL_NAME           From: does not
include a real name
 0.0 FORGED_RCVD_HELO       Received:
contains a forged HELO
[...]
 0.1 BOUNCE_MESSAGE         MTA bounce
message
 0.1 ANY_BOUNCE_MESSAGE     Message is
some kind of bounce message
```

Ways to prevent DOS attack

TCP-AO (TCP authentication option) provides a MAC calculated over the TCP packet and some sections of the IP header (such as addresses). A MAC algorithm provides message authentication, meaning this that we can detect if these fields have been modified in transit.

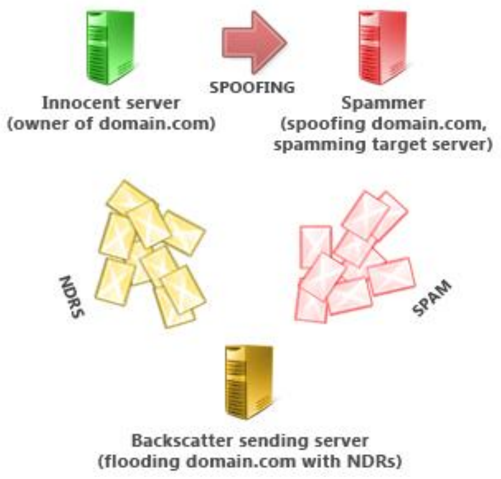


Fig 8: To stop Path backscatter messages

IPsec-AH calculates a MAC on the whole IP header and on the IP payload. This MAC allows us to detect whether the whole IP datagram has been modified in transit.

SYN cookies have nothing to do with IP Spoofing. SYN cookies are used to prevent DOS attacks. Port knocking is really focused on avoiding your machine ports to be scanned.

Two ways to defend against IP spoofing are: *Ingress Filtering* and *Egress filtering*

3.2 Sender Policy Framework: Spf Record Syntax

Domains define zero or more mechanisms. Mechanisms can be used to describe the set of hosts which are designated outbound mailers for the domain.

all | ip4 | ip6 | a | mx | ptr | exists | include

Mechanisms can be prefixed with one of four qualifiers:

"+"	Pass
"-"	Fail
"~"	SoftFail
"?"	Neutral

If a mechanism results in a hit, its qualifier value is used. The default qualifier is "+", i.e. "Pass". For example:

"v=spf1 -all"
"v=spf1 a -all"
"v=spf1 a mx -all"
"v=spf1 +a +mx -all"

3.3 ICMP Redirects

An ICMP redirect is an error message sent by a router to the sender of an IP packet. Redirects are used when a router believes a packet is being routed sub optimally and it would like to inform the sending host that it should forward subsequent packets to that same destination through a different gateway. In theory a host with multiple gateways could have one default route and learn more optimal specific routes over time by way of ICMP redirects. Consider the following diagram.

In Figure, host A wishes to send a packet to host B. The routing table on host A consists of a default route through router R1. When R1 receives a packet destined for host B, it looks at its routing table and notices that it has a route (not default) to R2 for packets destined to host

B. Therefore, it forwards the datagram to R2, who in turn passes it along to host B. Router 1 also notices that the outgoing interface and network for the packet was the same as the interface it arrived on. Since it is configured to send ICMP redirects, it sends an error message to host A informing it that all packets destined for host B should be forwarded to R2.

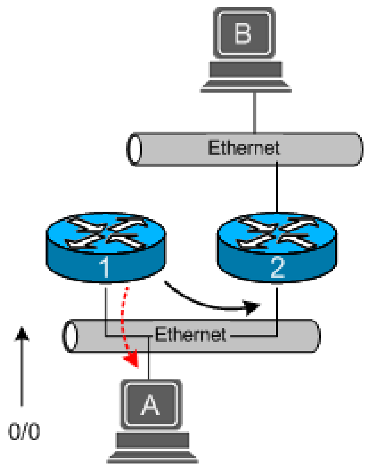


Fig 9: ICMP Redirect from Router 1.

3.4 SMTP

SMTP (Simple Mail Transfer Protocol) is a connection-oriented, text-based protocol in which a mail sender communicates with a mail receiver by issuing command strings and supplying necessary data over a reliable ordered data stream channel, typically a Transmission Control Protocol (TCP) connection. An *SMTP session* consists of commands originated by an SMTP client (the initiating agent, sender, or transmitter) and corresponding responses from the SMTP server (the listening agent, or receiver) so that the session is opened, and session parameters are exchanged. A session may include zero or more SMTP transactions. An *SMTP transaction* consists of three command/reply sequences (see example below.) They are:

1. **MAIL** command, to establish the return address, a.k.a. Return-Path,[15] reverse-path,[16] bounce address, mfrom, or envelope sender.
2. **RCPT** command, to establish a recipient of this message. This command can be issued multiple times, one for each recipient. These addresses are also part of the envelope.
3. **DATA** to signal the beginning of the *message text*; the content of the message, as opposed to its envelope. It consists of a *message header* and a *message body* separated by an empty line. DATA is actually a group of commands, and the server replies twice: once to the *DATA command* proper, to acknowledge that it is ready to receive the text, and the second time after the end-of-data sequence, to either accept or reject the entire message.

Ports: Communication between mail servers generally always uses the standard TCP port 25 designated for SMTP. Mail *clients* however generally don't use this, instead using specific "submission" ports. Mail services generally accept email submission from clients on one of:

- 587 (Submission), as formalized in RFC 6409 (previously RFC 2476)
- 465 This port has been deprecated since RFC 2487, after being briefly assigned for *secure SMTP* in the 1990s. Despite this, it is commonly used by mail providers
- Port 2525 and others may be used by some individual providers, but have never been officially supported.

Most Internet service providers now block all outgoing port 25 traffic from their customers as an anti-spam measure. For the same reason, businesses will typically configure their firewall to only allow outgoing port 25 traffic from their designated mail servers.

4. IMPLEMENTATION

4.1 Bandwidth problems in high speed networks:

The first problem is topology design and bandwidth allocation, and it is concerned with the ability to dynamically reconfigure a network in order to efficiently benefit from network resources. The second problem is concerned with flow control and congestion avoidance. Bandwidth management (BWM) protocols are used to prevent congestion, essentially by accepting or refusing a new-arrival cell. The third problem, which is the most critical one, is bandwidth allocation, which is concerned with successful integrat ion of link capacities through the different types of services. Given that a virtual path is a logical direct link, composed of a number of virtual circuits, between any two nodes, the last problem is concerned with how to assign bandwidth to each virtual path in the network, in order to optimize performance for all users. This paper may be a good guide to researchers concerned with high-speed networks in general.

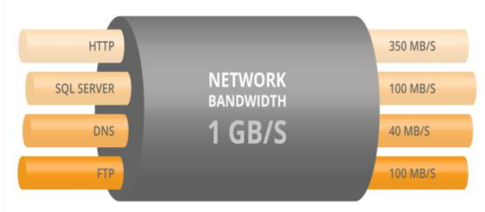


Fig 10: Bandwidth usage

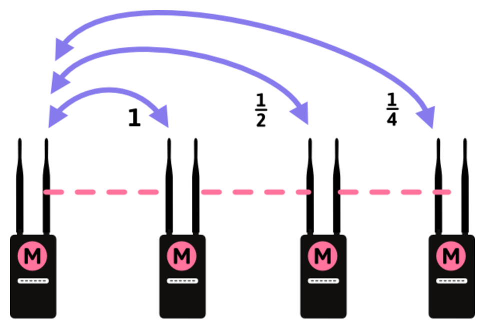


Fig 11: Mesh hop: Bandwidth-over-hops-problem

4.2 Greedy Algorithm

A **greedy algorithm** is an algorithm that follows the problem solving heuristic of making the locally optimal choice at each stage with the hope of finding a global optimum. In many problems, a greedy strategy does not in general produce an optimal solution, but nonetheless a greedy heuristic may yield locally optimal solutions that approximate a global optimal solution in a reasonable time.

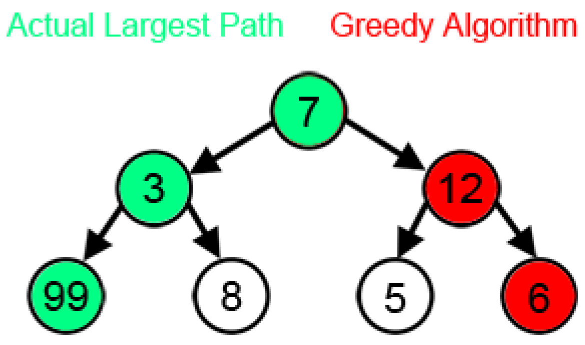


Fig 12: Greedy search path

With a goal of reaching the largest-sum, at each step, the greedy algorithm will choose what appears to be the optimal immediate choice, so it will choose 12 instead of 3 at the second step, and will not reach the best solution, which contains 99.

1 Specifics

In general, greedy algorithms have five components:

1. A candidate set, from which a solution is created
2. A selection function, which chooses the best candidate to be added to the solution
3. A feasibility function, that is used to determine if a candidate can be used to contribute to a solution
4. An objective function, which assigns a value to a solution, or a partial solution, and
5. A solution function, which will indicate when we have discovered a complete solution

Most problems for which algorithm work will have two properties:

2 Greedy choice property:

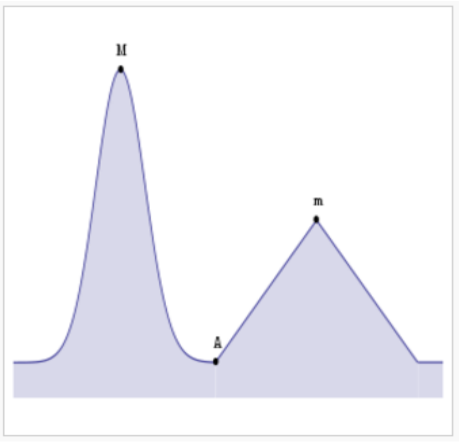
We can make whatever choice seems best at the moment and then solve the subproblems that arise later. The choice made by a greedy algorithm may depend on choices made so far, but not on future choices or all the solutions to the subproblem. It iteratively makes one greedy choice after another, reducing each given problem into a smaller one. In other words, a greedy algorithm never reconsiders its choices. This is the main difference from dynamic programming, which is exhaustive and is guaranteed to find the solution. After every stage, dynamic programming makes decisions based on all the decisions made in the previous stage, and may reconsider the previous stage's algorithmic path to solution.

3 Optimal substructure

"A problem exhibits optimal substructure if an optimal solution to the problem contains optimal solutions to the sub-problems."

4 Cases of failure

Examples on how a greedy algorithm may fail to achieve the optimal solution.



Starting at A, a greedy algorithm will find the local maximum at "m", oblivious of the global maximum at "M".

With a goal of reaching the largest-sum, at each step, the greedy algorithm will choose what appears to be the optimal immediate choice, so it will choose 12 instead of 3 at the second step, and will not reach the best solution, which contains 99.For many other problems, greedy algorithms fail to produce the optimal solution, and may even produce the *unique worst possible* solution.

4.3 Activity selection problem

The activity selection problem is a combinatorial optimization problem concerning the selection of non-conflicting activities to perform within a given time frame, given a set of activities each marked by a start time (s_i) and finish time (f_i). The problem is to select the maximum number of activities that can be performed by a single person or machine, assuming that a person can only work on a single activity at a time. A classic application of this problem is in scheduling a room for multiple competing events, each having its own time requirements (start and end time), and many more arise within the framework of operations research.

(a) Formal definition

Assume there exist *n* activities with each of them being represented by a start time s_i and finish time f_i. Two activities *i* and *j* are said to be non-conflicting if $s_i \geq f_j$ or $s_j \geq f_i$. The activity selection problem consists in finding the maximal solution set (S) of non-conflicting activities, or more precisely there must exist no solution set S' such that |S'| > |S| in the case that multiple maximal solutions have equal sizes.

(b) Optimal Solution

The activity selection problem is notable in that using a greedy algorithm to find a solution will always result in an optimal solution.

(c) Algorithm

```
Greedy-Iterative-Activity- Selector(A, s, f):

 Sort A by finish times stored in f'

 S = {A[1]}
 k = 1

 n = A.length

for i = 2 to n:
   if s[i] ≥ f[k]:
        S = S U {A[i]}
        k = i
```

```
return S
```

Explanation

Line 1: This algorithm is called *Greedy-Iterative-Activity-Selector*, because it is first of all a greedy algorithm, and then it is iterative. There's also a recursive version of this greedy algorithm.

- is an array containing the *activities*.
- is an array containing the *start times* of the activities in .
- is an array containing the *finish times* of the activities in .

Note that these arrays are indexed starting from 1 up to the length of the corresponding array.

Line 3: Sorts in *increasing order of finish times* the array of activities by using the finish times stored in the array . This operation can be done in time, using for example merge sort, heap sort, or quick sort algorithms.

Line 5: Creates a set to store the *selected activities*, and initialises it with the first activity . Note that, since the has already been sorted according to the finish times in , is the activity with the smallest finish time.

Line 6: Creates a variable that keeps track of the index of the last selected activity.

Line 10: Starts iterating from the second element of that array up to its last element.

Line 11: If the *start time* of the activity () is greater or equal to the *finish time* of the *last selected activity* (), then is compatible to the selected activities in the set , and thus it can be added to ; this is what is done in line 12.

Line 13: The index of the last selected activity is updated to the just added activity .

(d) Proof of optimality

Let be the set of activities ordered by finish time. Thus activity 1 has the earliest finish time.

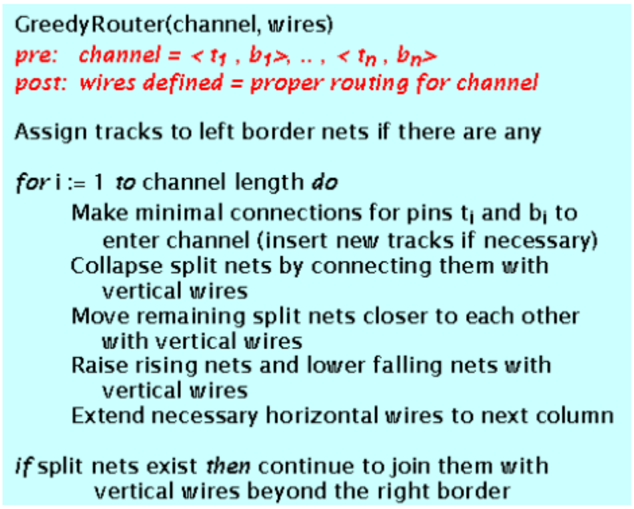
Suppose *A* is a subset of *S* is an optimal solution and let activities in *A* be ordered by finish time. Suppose that the first activity in *A* is $k \neq 1$, that is, this optimal solution *does not*start with the "greedy choice." We want to show that there is another solution B that begins with the greedy choice, activity 1. Because , the activities in *B* are disjoint and since *B* has same number of activities as *A*, i.e., |*A*| = |*B*|, *B* is also optimal.Once the greedy choice is made, the problem reduces to finding an optimal solution for the subproblem. If *A* is an optimal solution to the original problem *S*, then is an optimal solution to the activity-selection problem .Why? If we could find a solution *B*′ to *S*′ with more activities then *A*′, adding 1 to *B*′ would yield a solution *B* to *S* with more activities than *A*, contradicting the optimality.

In our project greedy algorithm is used in database to block spoofer,

```
CREATE TABLE [dbo].[Block_list](
        [cus_name] [varchar](50) NULL,
        [cus_ac] [varchar](50) NULL,
        [Hack_date] [datetime] NULL,
        [cus_ip] [varchar](50) NULL,
        [IpSpoofer_IP]     [varchar](50)
NULL
) ON [PRIMARY]
```

5. APPLICATIONS

For a network channel router greedy applications



```
GreedyRouter(channel, wires)
pre:  channel = < t_1 , b_1>, .. , < t_n , b_n>
post: wires defined = proper routing for channel

Assign tracks to left border nets if there are any

for i := 1 to channel length do
    Make minimal connections for pins t_i and b_i to
        enter channel (insert new tracks if necessary)
    Collapse split nets by connecting them with
        vertical wires
    Move remaining split nets closer to each other
        with vertical wires
    Raise rising nets and lower falling nets with
        vertical wires
    Extend necessary horizontal wires to next column

if split nets exist then continue to join them with
        vertical wires beyond the right border
```

Fig 13: Greedy channel router

To come up with a Greedy Algorithm:

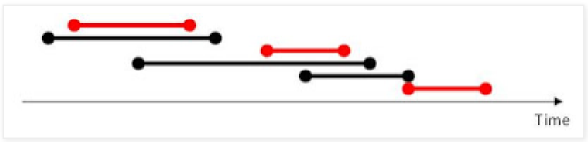
Let A = {5, 3, 4, 2, 1} and T = 6

After sorting, A = {1, 2, 3, 4, 5}

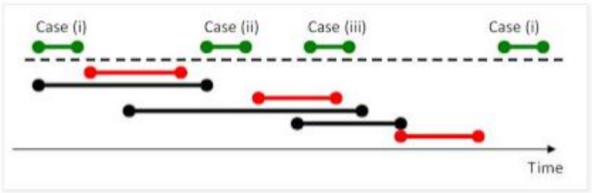
After 1st iteration, current Time = 1 and number Of Things = 1
After 2nd iteration, current Time = 1 + 2 = 3 and number Of Things = 2
After 3rd iteration, current Time = 3 + 3 = 6 and number Of Things = 3
After 4th iteration, current Time = 6 + 4 = 10 which is greater than T, answer will be 3.
Completion times is the total time needed to complete the work,
C(j) = T[1] + T[2] + ... + T[j] where 1 <= j <= N

Where to use Greedy Algorithms?
A problem must exhibit these two ingredients in order for a greedy algorithm to work:

1 It has optimal substructures. Optimal solution to the problem contains optimal solutions to the sub-problems.
2 It has a greedy property (hard to prove its correctness!). If we make a choice that seems best at the moment and solve the remaining subproblems later, we still reach optimal solution. We never have to reconsider our previous choices.



In the diagram, all the activities are sorted in ascending order of ending time and the red intervals are the chosen activities by the above method. Suppose the contrary that the above method does not yield an optimal solution, i.e. there exists an activity X (green in colour) within the above diagram and it is not chosen. 3 cases are considered:



Case (i): Activity X ends earlier than start of 1st chosen activity OR starts later than the end of last chosen activity.

This is impossible because if such activity X exists, the "greedy" method mentioned is able to choose it, and this contradicts to the real situation.

Case (ii): Activity X lies between end of a chosen activity and start of next chosen activity.
Similar to case (i), if activity X exists like this, the "greedy" method is able to handle such case. Thus case (ii) also leads to contradiction.
Case (iii): Activity X has time clash with at least one of the chosen activites (red one).
This can be a possible situation. But the problem now is to find out a better solution than the "optimal" one found by the above "greedy" method. Case (iii) obviously does not yield a better solution, since if activity X is chosen instead, at least one of the activities in "red" must be discarded.
As a result, such "greedy" method is correct.
If we claim that a shorter total time can be reached when we produce toy a prior to toy b, then

$$\max(\max(D-ma, 0) + pa-mb, 0) + pb < \max(\max(D-mb, 0) + pb-ma, 0) + pa$$
$$\max(\max(D-ma, 0), mb-pa) + pb + pa - mb < \max(\max(D-mb, 0), ma-pb) + pa + pb - ma$$
$$\max(\max(D-ma, 0), mb-pa) - mb < \max(\max(D-mb, 0), ma-pb) - ma$$
$$\max(D-ma, 0, mb-pa) - mb < \max(D-mb, 0, ma-pb) - ma$$
$$\max(D, ma, ma+mb-pa) - mb - ma < \max(D, mb, ma+mb-pb) - ma-mb$$
$$\max(D, ma, ma+mb-pa) < \max(D, mb, ma+mb-pb)$$
$$\max(ma, ma+mb-pa) < \max(mb, ma+mb-pb)$$
$$\max(-mb, -pa) + ma + mb < \max(-ma, -pb) + mb + ma$$
$$\max(-mb, -pa) < \max(-ma, -pb)$$
$$-\min(mb, pa) < -\min(ma, pb)$$
$$\underline{\min(ma, pb) < \min(mb, pa)}$$

This is exactly what we have proposed.
Advantage: simple and efficient. Disadvantage: may miss the best path.

CONCLUSION

On evaluating the major problems of bandwidth allocation, DOS attack, PIT and path backscatter we come to a conclusion that timing is a major part playing in networking. This TTL can be used to block spoofer by reducing the TTL of a session and take immediate measure on hacking. When path backscatter occurs the traffic is loaded to sender asking to resend the missing packet, immediately ICMP error messages are overloaded to spoofer so automatically Bandwidth Management Protocol (BWM) applies and reroutes through new gateway avoiding spoofer node. At this stage greedy algorithm helps rerouting less traffic path. These study can help future IP spoofing countermeasures.

ACKNOWLEDGEMENT

The authors would like to thank the anonymous reviewers and the editor for their valuable comments and suggestions to improve the quality of this paper. They are also grateful to authors of the publishers.

REFERENCES

1. M. Ramalho, "Intra- and Inter-Domain Multicast Routing Protocols: A Survey and Taxonomy," IEEE Communications Surveys & Tutorials, 1st Quarter 2000.
2. Sally Floyd, Van Jacobson, Ching-Gung Liu, Steven McCanne, Lixia Zhang, "A reliable multicast framework for light-weight sessions and application level framing," IEEE/ACM Transactions on Networking, December 1997.
3. Dan Pei, Lan Wang, Daniel Massey, S. Felix Wu, Lixia Zhang, "A Study of Packet Delivery Performance during Routing Convergence," IEEE DSN 2003.
4. Sarang Dharmapurikar, Praveen Krishnamurthy and David E. Taylor, "Longest Prefix Matching using Bloom Filters," SIGCOMM 2003.
5. Pierre Reinbold, and Olivier Bonaventure, "IP Micro-mobility Protocols," IEEE Communications Surveys & Tutorials, Third Quarter 2003.
6. Kunwadee Sripanidkulchai, Bruce Maggs, Hui Zhang, "Efficient Content Location Using Interest-Based Locality in Peer-to-Peer Systems," INFOCOM 2003.
7. L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "RSVP: A New Resource Reservation Protocol," IEEE Network, Sep 1993.
8. Mir, N.F. (2006) *Computer and Communication Networks*, Prentice Hall.
9. “Cryptography and Network Security: Principles and Practice” by William Stallings
10. “Activity Selection Problem” http://en.wikipedia.org/wiki/Activity_selection_problem
11. “Saving Endeavour” http://poj.org/problem?id=2751
12. Applications of greedy approaches: http://hkuccst9003.blogspot.in/2011/09/applications-of-greedy-approaches.html
13. Tom Anderson, Scott Shenker, Ion Stoica, and David Wetherall, "Design Guidelines for Robust Internet Protocols," HotNets-I, Oct 2002.
14. http://link.springer.com/article/10.1007%2FBF01584082
15. Ion Stoica, Daniel Adkins, Shelley Zhuang, Scott Shenker, Sonesh Surana, "Internet Indirection Infrastructure," SIGCOMM 2002.
16. Larry Peterson, Tom Anderson, David Culler, and Timothy Roscoe, "A Blueprint for Introducing Disruptive Technology into the Internet," HotNets-I, Oct 2002.
17. M. R. Pearlman and Z. J. Haas, "Determining the Optimal Configuration for the Zone Routing Protocol, " IEEE Journal on Selected Areas in Communications: Wireless Ad-Hoc Networks, vol. 17, no. 8, p. 1395-1414, August 1999.
18. Hari Balakrishnan, M. Frans Kaashoek, David Karger, Robert Morris, Ion Stoica,

"Looking up data in P2P systems," Communications of the ACM, Feb 2003.

19. P. Mehra, A. Zakhor, and C. D. Vleeschouwer, "Receiver-Driven Bandwidth Sharing for TCP," INFOCOM 2003
20. Rongmei Zhang, Y. Charlie Hu and Peter Druschel, "Optimizing Routing in Structured Peer-to-peer Overlay Networks Using Routing Table Redundancy," International Workshop on Future Trends of Distributed Computing Systems (FTDCS 2003), May 2003.
21. Firewall/IDS Evasion and Spoofing https://nmap.org/book/man-bypass-firewalls-ids.html
22. Jack Edmonds, Matroids and the greedy algorithm, Springer-Verlag, 0025-5610, Volume 1, Issue 1 , pp 127-136