Post 7

Thursday, January 24, 2019 11:24 PM

CCNA: NAT/PAT

Not enough public IPv4's to assign unique addr to everyone: Private addr don't ID any single company/org:

Can't be routed over Internet: To be routed: Has to be translated to a public address

- **NAT:** Provides translation of private to public addr
- Allows device w/private IPv4 to access resources outside
- W/out NAT: Depletion of IPv4 would have happened before 2,000

RFC 1918: Private addresses

Class	Addr Range	CIDR
A	10.0.0.0-10.255.255.255	10.0.0/8
B	172.16.0.0-172.31.255.255	172.16.0.0/12
C	192.168.0.0-192.168.255.255	192.168.0.0/16

NAT: Primary use: Conserve public IPv4 addr:

- Allows networks to use private IPv4/translates to public ones
- Degree of privacy/sec b/c it hides internals

NAT pool: NAT-enabled rtrs config w/1/more valid public IPv4's

- Internal device sends traffic out network
- NAT-enabled rtr translates addr of device to a public addr from NAT pool
- To outside devices: All traffic entering/exiting appears to have public IPv4

NAT rtr: Typically ops at border of stub network

Stub network: Network w/single connection to neighbor network (1 way in/out)

When device inside stub wants to comm w/device outside:

- Packet fwded to border rtr: It performs NAT process
- Translates internal addr of device to pub routable outside addr

Inside network: Set of networks subject to translation

Outside network: All other networks

IPv4: Diff designations based on if priv/pub network (Internet): Whether traffic incoming/outgoing

NAT: 4 types of addr:

Inside local	Addr of source as seen from inside network
Inside global	Addr of source as seen from outside network • Translates inside local to inside global
Outside local	Addr of dest as seen from inside network: • Could be diff than globally rtable addr of dest (uncommon)
Outside global	Addr of dest as seen from outside network: • Globally routable IPv4 assigned to a host on Internet: Usually outside local/global addr are same

ALWAYS applied from perspective of device /translated addr:

Inside addr	Addr of device being translated by NAT
Outside addr	Addr of dest device
Local addr	Any addr that appears on inside of network
Global addr	Any addr that appears on outside of network

3 Types of NAT translation:

Static NAT	1-to-1 addr mapping bet local/global addr
Dynamic NAT	Many-to-many addr mapping bet local/global addr
PAT	**Port Address Translation:** AKA NAT overloading

	• Many-to-1 addr mapping bet local/global addr

Static NAT: 1-to-1: Mappings config by admin: Remain constant
When devices send traffic to Internet: Inside local addr translated to config inside global addr
To outside: Devices have public IPv4's
Useful for:

- Web servers/devices must have consistent addr accessible from Internet
- Devices must be accessible by auth personnel when offsite/not gen public
- Req: Enough public addr to satisfy total # of simultaneous usr sessions

Dynamic NAT: Uses pool of pub addr/assigns them on 1st-come/1st-serve basis

- When inside device req access to outside network: Dynamic assigns avail public IPv4 from pool
- Req: Enough public addr to satisfy total # of simultaneous usr sessions

PAT: Port Addr Translation (NAT overload): Maps multiple priv IPv4 to single/few public addr

- Multiple addr can be mapped to 1/more few addr
- B/c each private addr also tracked by port #
- When device initiates TCP/IP session: Generates TCP/UDP source port value to ID it
- When NAT rtr receives packet from client: Uses source port # to ID specific NAT translation

PAT ensures devices use diff TCP port # for each session w/server on Internet

- When response comes back from server: Source port #, becomes dest port # on return trip
- Determines to which device rtr fwds packets
- PAT process validates incoming packets requested: Adding degree of sec to session

Comparing NAT/PAT

- NAT translates IPv4 addr on 1:1 basis bet private/public IPv4's
- PAT mods both the addr/port #

NAT fwds incoming packets to their inside dest by referring to incoming source IPv4 given by host on public network
PAT generally only 1/few publicly exposed IPv4's

- Incoming packets from public are routed to dest on private by referring to a table in NAT rtr
- Connection tracking: Table tracks public/private port pairs

Packets w/out L4 Segment: What about IPv4 packets carrying data other than TCP/UDP segment?

- Packets don't contain L4 port #
- PAT translates most common protocols that don't use TCP/UDP as a transport

Example: ICMPv4

- Each of these types of protocols handled diff by PAT
- ICMPv4 query msgs/echo req/replies include a Query ID\
- ICMPv4 uses Query ID to ID echo req w/corresponding reply
- Query ID is incremented w/each echo req sent
- PAT uses Query ID instead of L4 port #

Benefits/Disadvantages: NAT

Benefits	• **Conserves legally registered addr scheme** ▪ Through app port-lvl multiplexing ▪ Internal hosts can share single public IPv4 for all external comm • **Increases flexibility of connections to public network** ▪ Multiple/backup/load-balancing pools can be used to ensure reliable public connections • **Consistency for internal addr schemes** ▪ Network NOT using private IPv4/NAT: □ Changing public IPv4 scheme req readdr of all hosts on existing network: Pricey ▪ NAT allows existing private IPv4 scheme to remain while allowing easy change to new public scheme □ Org could change ISPs: No need to change inside clients • **Net sec: Private networks don't advertise addr/internal topology**
Disadvantages	• **Performance degraded** ▪ Affects real time protocols like VoIP ▪ NAT increases switching delays b/c translations of each IPv4 w/in packet headers takes time ▪ The 1st packet always process-switched going through a slower path ▪ Rtr must look at every packet to decide whether it needs translation ▪ Rtr must alter IPv4 header: Possibly alter TCP/UDP headers

	▪ IPv4/TCP/UDP header checksums must be recalc each time translation made ▪ Remaining packets go through fast-switched path if cache entry exists; otherwise delayed • **End-to-end functionality is degraded** ▪ Many protocols/apps depend on end-to-end addressing from source to dest ▪ Some apps don't work w/NAT ▪ Example: Sec apps (digital sigs) fail b/c source IPv4 changes before reaching dest ▪ If apps use phys addr (instead of qualified domain name) don't reach dest translated across NAT rtr ▪ Can be avoided at times by implementing static NAT mappings • **End-to-end IP traceability is lost** ▪ More diff to trace packets that undergo numerous packet addr changes over multiple NAT hops • **Tunneling becomes more complicated** ▪ Example: IPSec: NAT mods values in headers that interfere w/integrity checks • **Initiating TCP connections can be disrupted** ○ Services req initiation of TCP connections from outside ○ Stateless protocols (those using UDP) can be disrupted ○ Unless NAT rtr has been config to support these protocols: ▪ Incoming packets can't reach dest ▪ Some protocols can use 1 instance of NAT bet participating hosts (passive mode FTP): ▪ Fail when both sys separated from Internet by NAT

Config Static NAT: 1-to-1 mapping bet inside/outside addr

- Allows external devices to initiate connections to internal devices using statically assigned public addr

2 Basic Tasks when config static NAT translations:

1. Create a mapping bet inside local address/inside global addresses
2. After mapping: Ints participating in translation are config as inside/outside relative to NAT

Establish static translation bet inside local/inside global
R1(config)# ip nat inside source static local-ip global-ip
R1(config)# no ip nat inside source static [*global*] to remove dynamic source translation

Specify inside int
R1(config)# int type number

Mark int as connected to inside
R1(config-if)# ip nat inside

Specify outside int
R1(config)# int type number

Mark int as connected to the outside
R1(config-if)# ip nat outside

Example Syntax:
R1(config)# ip nat inside source static 192.168.10.254 209.165.201.5
R1(config)# int s0/0/0
R1(config-if)# ip address 10.1.1.2 255.255.255.252
R1(config-if)# ip nat inside
R1(config-if)# exit
R1(config)# int s0/1/0
R1(config-if)# ip address 209.165.200.225 255.255.255.224
R1(config-if)# ip nat outside

Verify Static NAT
R1# show ip nat translations Shows active NAT translations: Static translations (unlike dynamic) always in NAT table

- If cmd issued during active session: Output also indicates addr of outside device

R1# show ip nat statistics Display total # of active translations/NAT config params/# of addr in pool/# of addr allocated
R1# clear ip nat statistics Verify translation working: Clear stats from past translations
Dynamic NAT Op: Dynamic NAT allows auto mapping of inside local to inside global

- Inside global addr typically public IPv4 || Uses group/pool of public IPv4
- Req: Config of inside/outside ints participating in NAT
- Static NAT: Permanent mapping to single addr
- Dynamic NAT: Pool of addr

Pool of public IPv4 (inside global addr pool) avail to any device inside network on 1st-come 1st-serve basis

- Dynamic: Single inside addr translated to single outside addr
    - Must be enough addr in pool to accommodate all inside devices needing access to outside at same time
    - If all addresses in pool have been used: Device must wait for avail addr before it can access outside network

Config Dynamic NAT

- **Define pool of addr that will be used for translation using ip nat pool**
    - This pool of addr typically group of public addr
    - Addr defined by indicating starting/ending IP addr of pool
    - Netmask/prefix-length keyword indicates:
        - Which addr bits belong to network
        - Which belong to host for range of addr
- **Config standard ACL to ID (permit) addr to be translated:**
    - An ACL too permissive can lead to unpredictable results
    - Remember an implicit deny all statement at end of each ACL
- **Bind ACL to pool**
    - ip nat inside source list access-list-number pool pool name
    - Config used by rtr to ID which devices (list) receive which addresses (pool)
- **ID which ints inside (any int that connects to inside)**
- ID which ints outside (any int that connects to outside)

Dynamic NAT Config Steps
Define pool of global addr used for translation
R1(config)# ip nat pool name start-ip end-ip netmask [netmask] | prefix-length [prefix-length]

Config standard access list permitting addr should be translated
R1(config)# access-list ACL# permit source [source-wildcard]

Establish dynamic source translation: Specify access list/pool defined in prior steps
R1(config)# ip nat inside source list ACL# pool name

ID inside int
R1(config)# int type number
R1(config)# ip nat inside

ID outside int
R1(config)# int type number
R1(config#) ip nat outside

Example Syntax:
R1(config)# ip nat pool NAT-POOL1 209.165.200.226 209.165.200.240 netmask 255.255.255.224
R1(config)# access-list 1 permit 192.168.0.0 0.0.255.255
R1(config)# ip nat inside source list 1 pool NAT-POOL1
R1(config)# int s0/0/0
R1(config-if)# ip nat inside

R1(config)# int s0/1/0
R1(config-if)# ip nat outside

Verify Dynamic NAT
show ip nat translations Displays details of 2 previous NAT assignments
- Displays all static translations/dynamic translations created by traffic
- Adding verbose keyword displays addl info about each translation [how long ago entry created/used]

Default: Translation entries time out after 24 hrs
- Unless timers have been reconfig w/**ip nat translation timeout timeout-seconds** [global]

clear ip nat translation [global] Clear dynamic entries before timeout expired
- Useful to clear dynamic entries when testing NAT config

clear ip nat translation * [global] Clear all translations from table
- Only dynamic translations cleared from table: Static translations can't be cleared from translation table

show ip nat statistics Displays info about:
- Total # of active translations/NAT config params/# of addr in pool/how many addr have been allocated

sh run Look NAT/ACL/int/pool
clear ip nat translation inside global-ip local-ip [outside local-ip global-ip]
- Clear a dynamic translation entry containing inside translation/both inside/outside translation

clear ip nat translation protocol inside global-ip global-port local-ip local-port [outside local-ip local port global-ip global-port] Clears an extended dynamic translation entry
Configuring PAT: Address Pool
- PAT conserves addr in inside global addr pool by allowing rtr to use 1 inside global addr for many inside local addr
- A single public IPv4 can be used for 100's/1000's of internal private IPv4's
- Rtr maintains info from high-lvl protocols/TCP/UDP port #'s to translate inside global addr to correct inside local
- When multiple inside local addr map to 1 inside global address:
  - TCP/UDP port #'s of each inside host distinguish bet local addr

2 ways to config PAT: Depends on how ISP allocates public IPv4's
1. ISP allocates more than 1 public IPv4 to org
2. ISP allocates single public IPv4 req for org to connect to ISP

Config PAT for Pool of Public IP's
- If site has been issued more than 1 public IPv4:
  - These addr can be part of pool used by PAT
  - Similar to dynamic NAT, but not enough public addr for 1-to-1 mapping of inside to outside addr
  - Small pool of addr shared among larger # of devices

Steps:

- **Define pool of global addr to be used for overload translation**

ip nat pool name start-ip end-ip [netmask netmask | prefix-length prefix-length]

- **Define a standard access list perm addr that should be translated**

access-list access-list-number permit source [source-wildcard]

- **Establish overload translation: Specify access list/pool defined in prior steps**

ip nat inside source list access-list-number pool name overload

- **ID inside int int type number > ip nat inside**
- ID outside int int type number > ip nat outside

Syntax:
Define a pool of public IPv4's under pool name NAT-POOL2
R1(config)# ip nat pool NAT-POOL2 209.165.200.225 209.165.200.240 netmask 255.255.255.224

Define which addr are eligible to be translated
R1(config)# access-list 1 permit 192.168.0.0 0.0.255.255

Bind NAT-POOL2 w/ACL 1
R1(config)# ip nat inside source list 1 pool NAT-POOL2 overload

ID int s0/0/0 as inside NAT int
R1(config)# int s0/0/0
R1(config-if) ip nat inside

ID int s0/1/0 as outside NAT int
R1(config)# int s0/1/0
R1(config-if)# ip nat outside

Config PAT: Single Address

1. Define ACL to permit traffic to be translated
2. Config source translation using **int/overload** keywords

int keyword ID's which int IP addr to use when translating inside addr
overload keyword directs rtr to track port #'s w/each NAT entry

1. ID which ints inside in relation to NAT: Any int that connects to inside
2. ID which int outside in relation to NAT: Same int ID'd in source translation statement in 2

Syntax:

- **Define standard access list permitting the addr that should be translated**

access-list access-list-# permit source [source-wildcard]

- **Establish dynamic source translation, specify ACL, exit int/overload options**

ip nat inside source list acl-# int type number overload

- **ID inside int > int type # > ip nat inside**
- ID outside int > int type # > ip nat outside

Port Forwarding (AKA tunneling)

- Act of fwding traffic addr to specific network port from 1 network node to another
- Allows external usr to reach a port on priv IPv4 from outside through NAT-enabled rtr

P2P file sharing programs/ops [web serving/outgoing FTP] require rtr ports be fwded/opened to allow app to work

- B/c NAT hides internal addr: P2P only works from inside out: NAT can map outgoing reqs against incoming replies
- NAT doesn't allow reqs initiated from the outside
- Resolved w/manual intervention: Port fwding can be config to ID specific ports that can be fwded to inside hosts

Example: http://www.example.com:8080
Port fwding: Allows usrs on Internet to access internal servers by using WAN port addr of rtr/matched external port #

- Internal servers typically config w/RFC 1918 private IPv4's
- When req sent to IPv4 of WAN port via Internet: Rtr fwds req to appropriate server on LAN
- For sec reasons: Broadband rtrs by default don't permit any external network req to be fwded to inside host

Port fwding can be enabled for apps by specifying inside local addr that reqs should be fwded to

- Port other than default can be specified
- External usr would have to know specific port # to use

Config Port Fwding w/IOS
R1(config)# ip nat inside source { static [tcp | udp] local-ip local-port global-ip global-port } [extendable]

tcp/udp	Indicates if TCP/UDP port #
local-ip	IPv4 addr assigned to host on inside

local-port	Local TCP/UDP port range 1-65,535: Port # server listening on
global-ip	IPv4 of inside host: IP outside clients use
global-port	Global TCP/UDP port range 1-65,535: Port # outside client will use to reach server
global-ip	IPv4 of inside host: IP addr outside clients will use to reach internal server
global-port	Port # outside client will use to reach internal server
extendable	Applied automatically: • Keyword allows usr to config 7 ambiguous static translations • Translations w/same local/global addr • Allows rtr to extend translation to more than 1 port if necessary

Establishes static translation bet inside local addr/local port and inside global addr/port
R1(config)# ip nat inside source static tcp 192.168.10.254 80 209.165.200.225 8080

ID int s0/0/0 as inside NAT int
R1(config)# int s0/0/0
R1(config-if)# ip nat inside

ID int s0/1/0 as outside NAT int
R1(config)# int s0/1/0
R1(config-if)# ip nat outside

NAT for IPv6? Since early 1990s: Concern about depletion of IPv4 space has been priority of IETF
- Combo of RFC 1918 and NAT has been instrumental in slowing depletion
- RFC 5902: IAB: Internet Arch Board (IAB)

IPv6 Unique Local Addresses
ULA: IPv6 unique local addresses similar to RFC 1918 private addresses in IPv4, but significant diff
- Intent: Provide IPv6 space for comm w/in local site: Not meant to provide addl space, nor meant to provide lvl of sec
- Unique local addresses defined in RFC 4193

Also known as: Local IPv6 addr (not to be confused w/link-local addr)
Characteristics include:
- Allows sites to be combined/privately interconnected, w/out creating any addr conflicts
  - or req renumbering of ints that use these prefixes
- Independent of any ISP: Can be used for comm w/in site w/out having connectivity
- Not routable across Internet: If accidentally leaked by routing/DNS: No conflict w/other addr
- Not as straight-fwd as RFC 1918 addr
  - Hasn't been the intention of IETF to use form of NAT to translate bet unique local addr/IPv6 global unicasts

NAT for IPv6: Used in much diff context than IPv4
- Used to transparently provide access bet IPv6-only/IPv4-only networks
- NOT used as form of private IPv6 to global IPv6 translation

Dual-stack: When devices are running protocols associated w/both IPv4/6
Tunneling for IPV6: Process of encapsulating an IPv6 packet inside an IPv4 packet
- Allows IPv6 packet to be transmitted over IPv4-only network

NAT IPv6 should not be used as a long term strategy
- Over the years, there have been 7 types of NAT for IPv6

Troubleshooting NAT: debug
debug ip nat Verify op of NAT by displaying info about every packet translated by rtr
- Generates description of each packet considered for translation
- Provides info about certain errors/exceptions (like failure to allocate a global addr)

debug ip nat detailed More overhead than debug ip nat
- Can provide detail that may be needed to troubleshoot NAT problem
- Turn off debugging when finished

When decoding debug output:

*	Next to NAT: Indicates translation is occurring in fast-switched path • 1st packet in convo always process-switched: Slower • Remaining packets go through fast-switched path if cache entry exists
s=	Source IP

a.b.c.d—>w.x.y.z	Source addr a.b.c.d translated to w.x.y.z
d=	Destination IP
[xxxx]	IP ID #: Enables correlation w/other packet traces from protocol analyzer