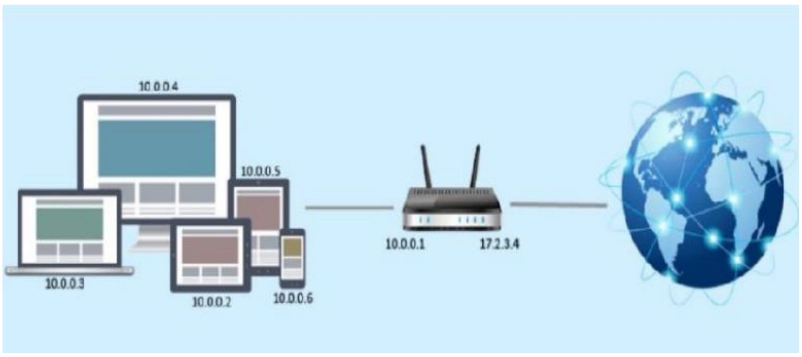
NAT

A classic scenario: a computer opens an outgoing connection to a public IP address. The router itself acts as a NAT, replacing the internal IP with the router's public IP address: then when data comes back, the router translates back to the internal address.

NAT is transparent for outgoing connections. *This means that other connections are a bit of a problem.*



Network of IP local addresses (10.0.0.). The interface 10.0.0.1 is used as an interface from the router to the local network. The public IP address of the router is 17.2.3.4.*

***All** packets have as receiver 10.0.0/24*

All** packets are sent as 17.2.3.4 **(different ports!)

Why this? Because of a lack of Ipv4 addresses, because it simplifies the network administration *(modifying local addresses without notifying it to all the network and viceversa - changing the NAT address without modifying the local addresses)*, plus security reasons: *can't directly address from outside a local host and there's an additional place for filtering policies.*

So the router remembers for all requests who sends to what, and then with the use of ports reverts the mechanism. All of this is pure trickery, because the router works at the IP level using the information on transport level (TCP port).

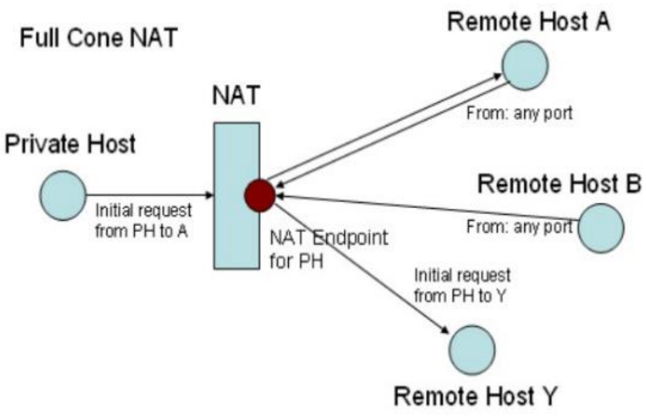
The NAT has been defined during the Client-Server era, dynamic hosts connecting to public (stable IP) servers. But this doesn't help at all in P2P *(full Ipv6 will solve this, but we're not there yet)*, due to asymmetry: *peers want to connect to other peers and even accept connections from unknown ones!*

So we need some ways to traverse these transparent NATs. The **LibP2P** we introduced earlier is defined to run everywhere *(not only data centers/stable public address machines)* and implements the **NAT transversal techniques** so that others may connect even through a NAT.

Different types of NAT

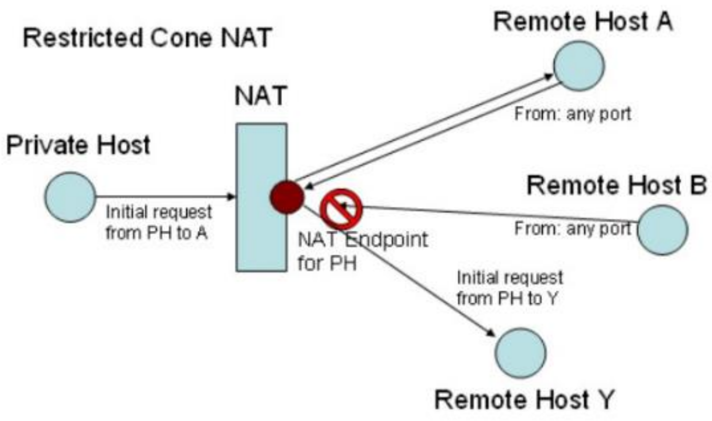
- **Full Cone:** *(easier)*

  All requests from the same intern IP are mapped to the same *IP_addr + Port* without checking the remote host address. *"You know him, you can target him"*



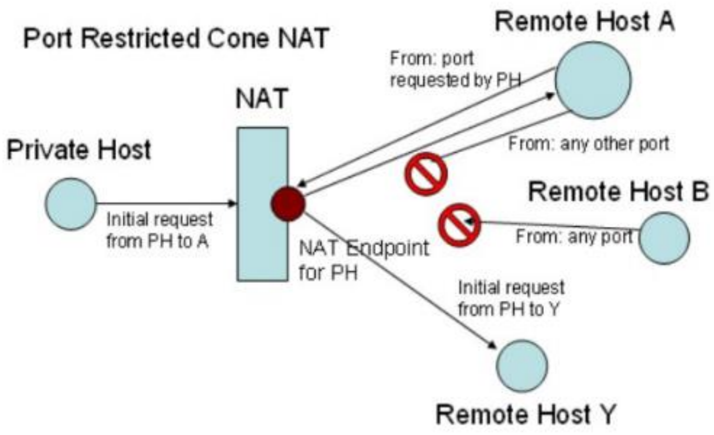
- **Address Restricted:**

  Endpoint isn't enough, target someone if the internal host has sent a packet towards you



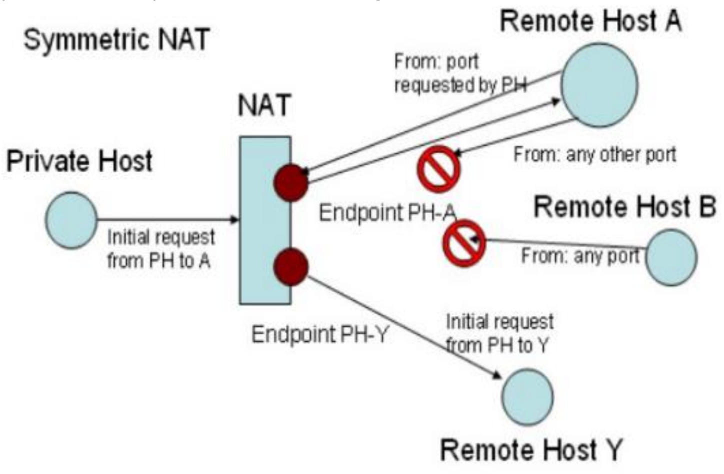
- **Address and Port Restricted:**

  Same as before, but port too



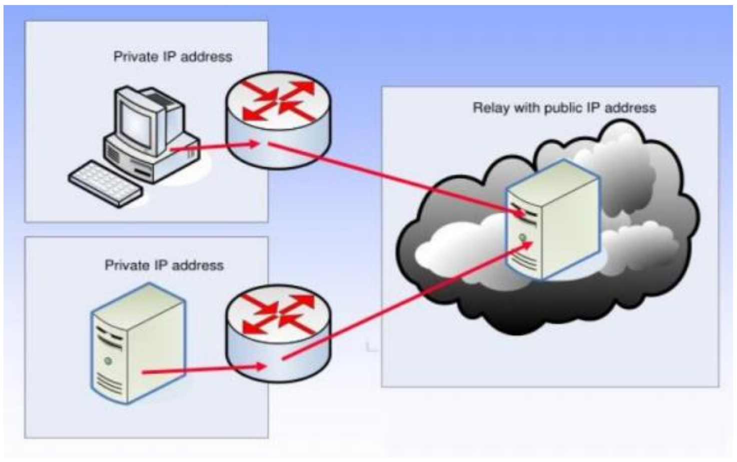
- **Symmetric:** *(harder)*

  If I open a bind towards an external host, it's for that external host only. *Only that and through that only, a.k.a. everyone has its binding.*



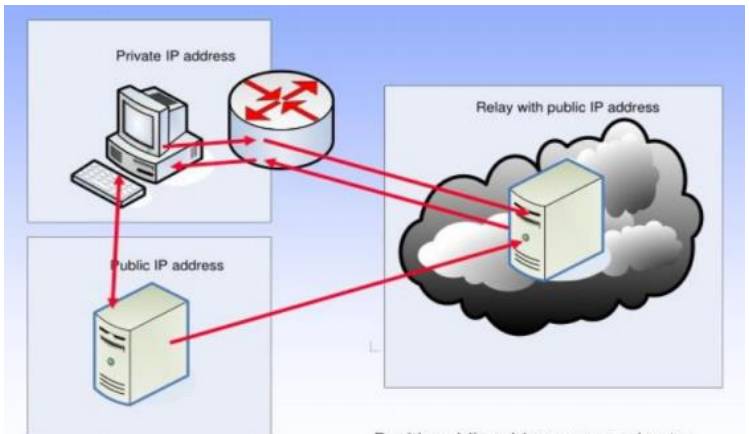
NAT transversal

- **Relying:**



Uses an intermediate relay node (rendez-vous node): *all the traffic between the peers goes through the relay*

- **Connection Reversal:**

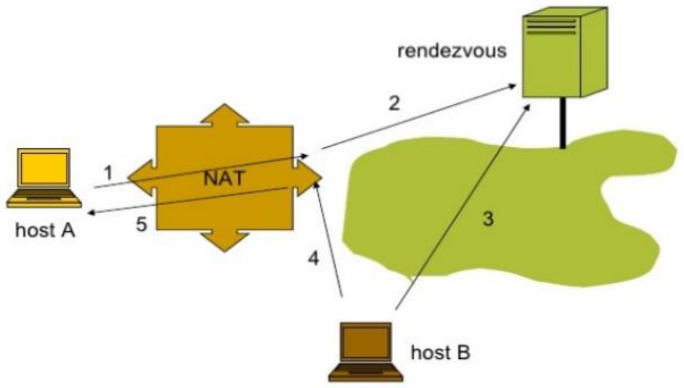


B with public address uses relay to send the address to NATed node

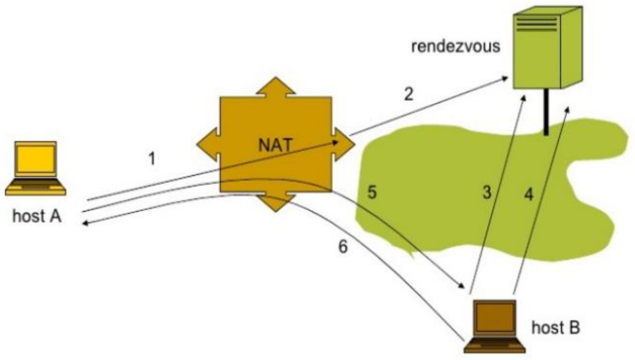
(One public and one private IP) Both peers open a connection with the Relay. *The peer with public IP asks the relay to tell the NATted peer to open an outgoing connection towards itself. So, it will send its address to the relay*

- **(UDP) Hole Punching:**

  Allows a direct communication between two peers by exploiting a rely server *(used only to put the peers in contact, packets are then sent P2P)*. It can also be exploited with TCP.



In the Full Cone



In the Restricted Cone

  Note: impossible in Symmetric NAT! Because it assigns new mapping for different destinations *(random port numbers)*. This means that the only way to traverse this NAT is by Connection Reversal or Relaying, rendez-vous is not useful here.

Depending on the configuration's complexity, there are different solutions.

If the peers are behind the same NAT, then each peer opens a connection with the rendez-vous and sends a packet including also its private local address. Then the rendez-vous notifies to each peer both the private and the public address of the other one. Finally, peers try to send packets both to the private address and to the public address of the other one.

Otherwise, both peers must "punch a hole" in the other peer's NAT. *Multiple messages may be required!*

Implementation note: Hole Punching is made through the *STUN (Session Traversal Utilities for NAT)* protocol. *Idea: peers help each other. When someone gets to know a not mapped (natted?) external address, I share it.*