Chair for Network Architectures and Services – Prof. Carle
Department for Computer Science
TU München

Master Course
Computer Networks
IN2097

Prof. Dr.-Ing. Georg Carle
Christian Grothoff, Ph.D.

Chair for Network Architectures and Services
Institut für Informatik
Technische Universität München
http://www.net.in.tum.de

Technische Universität München

Chair for Network Architectures and Services – Prof. Carle
Department for Computer Science
TU München

Network Address Translation (NAT)

NAT: Network Address Translation

- Problem: shortage of IPv4 addresses
  - only 32bit address field
  - more and more devices

- Idea: local network uses just one IP address as far as outside world is concerned:
  - only one IP address for all devices needed from ISP
  - can change addresses of devices in local network without notifying outside world
  - can change ISP without changing local addresses
  - devices inside local net not explicitly addressable/visible by outside world (a security plus).

Address Space

- IP addresses are assigned by the Internet Assigned Numbers Authority (IANA)

- RFC 1918 directs IANA to reserve the following IPv4 address ranges for private networks
  - 10.0.0.0 – 10.255.255.255
  - 172.16.0.0 – 172.31.255.255
  - 192.168.0.0 – 192.168.255.255

- The addresses may be used and reused by everyone
  - Not routed in the public Internet
  - Therefore a mechanism for translating addresses is needed

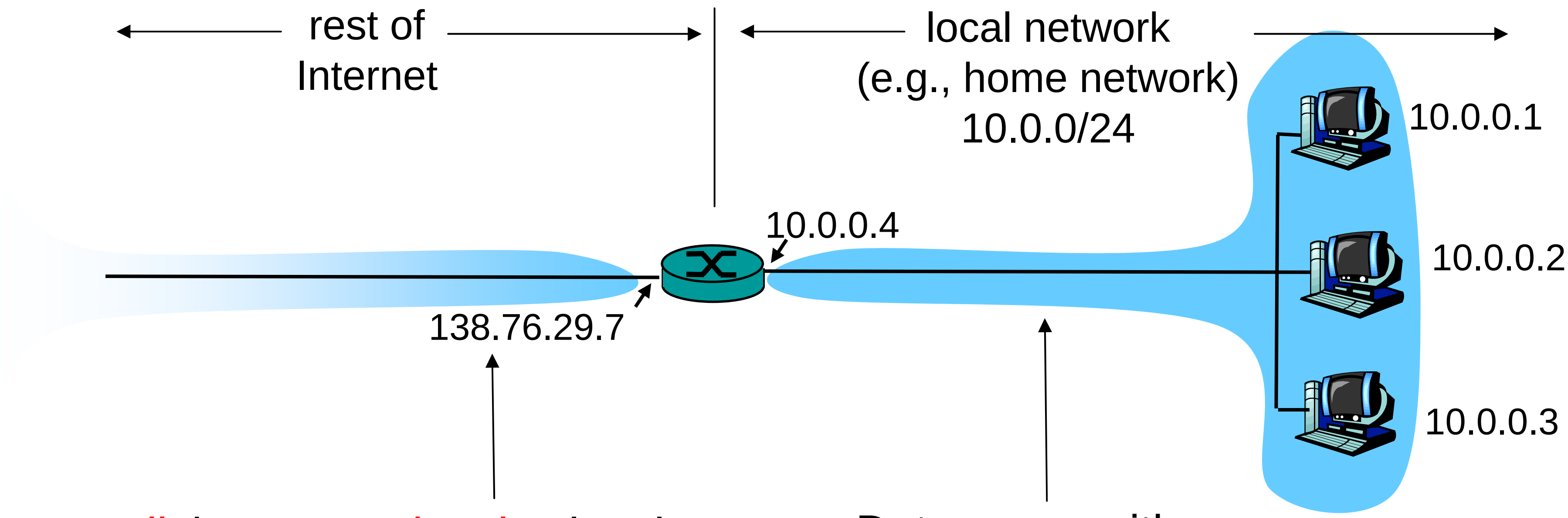
First approach – Network Address Translation

- Idea: only host communicating with the public Internet need a public address
  - Once a host connects to the Internet we need to allocate one
  - Communication inside the local network is not affected

- A small number of public addresses may be enough for a large number of private clients

- Only a subset of the private hosts can connect at the same time
  - not realistic anymore
  - We still need more than one public IP address

NAPT: Network Address and Port Translation



All datagrams *leaving* local network have same single source NAT IP address: 138.76.29.7, different source port numbers

Datagrams with source or destination in this network have 10.0.0/24 address for source, destination as usual

NAT: Network Address Translation

Implementation: NAT router must:

- *outgoing datagrams: replace* (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)

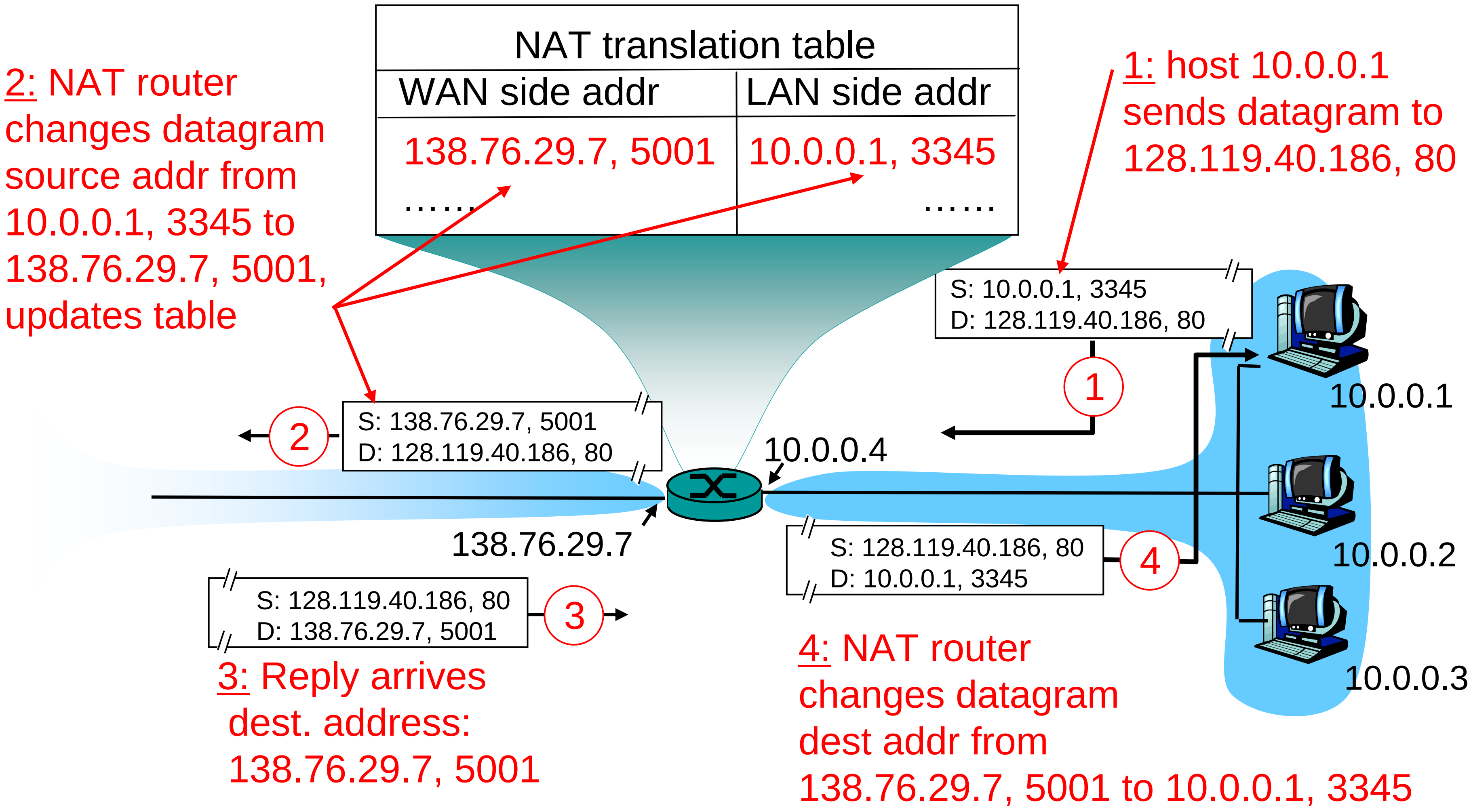
  . . . remote clients/servers will respond using (NAT IP address, new port #) as destination addr.

- *remember (in NAT translation table)* every (source IP address, port #) to (NAT IP address, new port #) translation pair

  -> we have to maintain a state in the NAT

- *incoming datagrams: replace* (NAT IP address, new port #) in dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

NAT: Network Address Translation



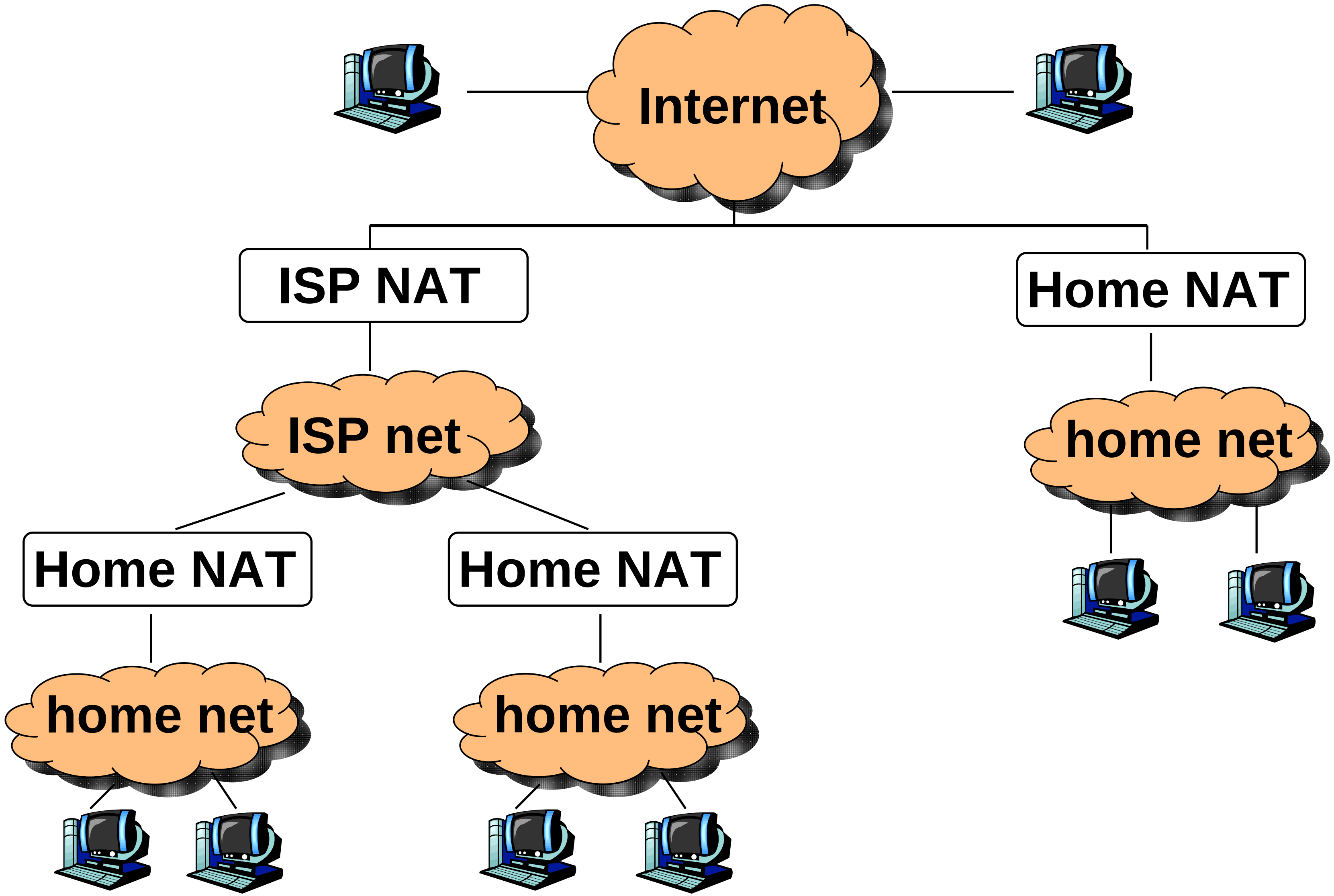
NAT: Network Address Translation

- 16-bit port-number field:
  - ~65000 simultaneous connections with a single LAN-side address!
  - helps against the IP shortage

- NAT is controversial:
  - routers should only process up to layer 3
  - violates end-to-end argument
    - NAT possibility must be taken into account by app designers, eg, P2P applications
  - address shortage should instead be solved by IPv6

Deployment of NAT

- Multiple levels of NAT possible



NAT Implementation

- Implementation not standardized
  - thought as a temporary solution
- implementation differs from model to model
  - if an application works with one NAT does not imply that is always works in a NATed environment
- NAT behavior
  - Binding
    - NAT binding
    - Port binding
  - Endpoint filtering

Binding

- When creating a new state, the NAT has to assign a new source port and IP address to the connection

- Binding covers context based packet translation

- **Port binding** describes the strategy a NAT uses for the assignment of a new external source port
  - source port can only be preserved if not already taken

- **NAT binding** describes the behavior of the NAT regarding the reuse of an existing binding
  - 2 consecutive connections from the same source
  - 2 different bindings?

Port binding

- Port-Preservation:
  - the local source port is preserved
- Port-Overloading:
  - port preservation is always used
  - existing state is dropped
- Port-Multiplexing:
  - ports are preserved and multiplexing is done using the destination transport address
  - more flexible
  - additional entry in the NAT table
- No Port-Preservation:
  - the NAT changes the source port for every mapping

NAT binding

- Reuse of existing bindings
  - two consecutive connections from the same transport address (combination of IP address and port)
  - NAT binding: assignment strategy for the connections
- Endpoint-Independent
  - the external port is only dependent on the source transport address
  - both connections have the same IP address and port
- Address (Port)-Dependent
  - dependent on the source and destination transport address
  - 2 different destinations result in two different bindings
  - 2 connections to the same destination: same binding
- Connection-Dependent
  - a new port is assigned for every connection
  - strategy could be random, but also something more predictable
  - Port prediction is hard

Endpoint filtering

- Filtering describes
  - how existing mappings can be used by external hosts
  - How a NAT handles incoming connections

- Independent-Filtering:
  - All inbound connections are allowed
  - Independent on source address
  - As long as a packet matches a state it is forwarded
  - No security

- Address Restricted Filtering:
  - packets coming from the same host (matching IP-Address) the initial packet was sent to are forwarded

- Address and Port Restricted Filtering:
  - IP address and port must match

NAT Types

- With Binding and Filtering 4 NAT types can be defined (RFC 3489)

- Full Cone NAT
  - Endpoint independent
  - Independent filtering

- Address Restricted NAT
  - Endpoint independent binding
  - Address restricted filtering

- Port Address Restricted NAT
  - Endpoint independent binding
  - Port address restricted filtering

- Symmetric NAT
  - Endpoint dependent binding
  - Port address restricted filtering

NAT Types

- With Binding and Filtering 4 NAT types can be defined (RFC 3489)

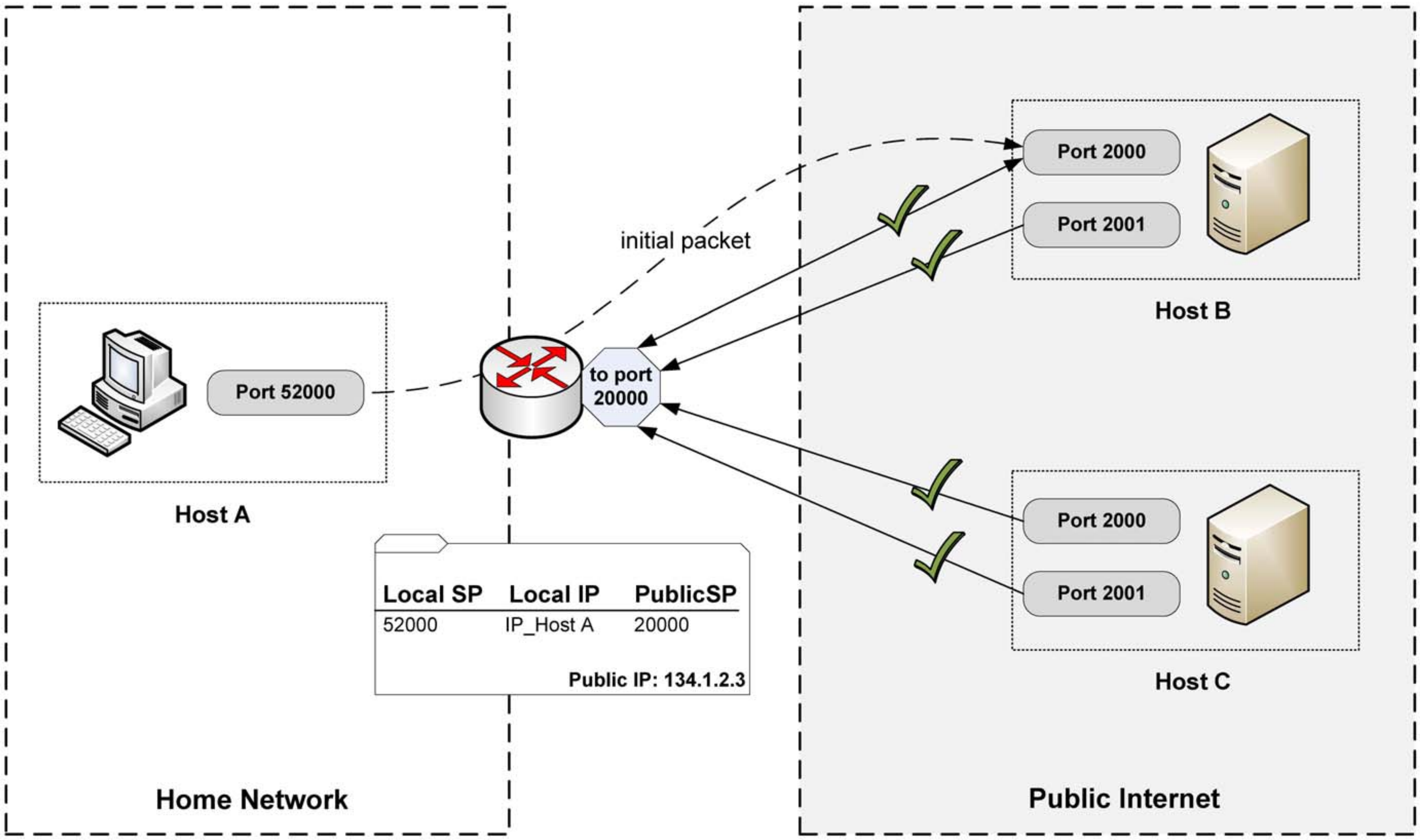
- **Full Cone NAT**
  - **Endpoint independent**
  - **Independent filtering**

- Address Restricted NAT
  - Endpoint independent binding
  - Address restricted filtering

- Port Address Restricted NAT
  - Endpoint independent binding
  - Port address restricted filtering

- Symmetric NAT
  - Endpoint dependent binding
  - Port address restricted filtering

Full Cone NAT



NAT Types

- With Binding and Filtering 4 NAT types can be defined (RFC 3489)

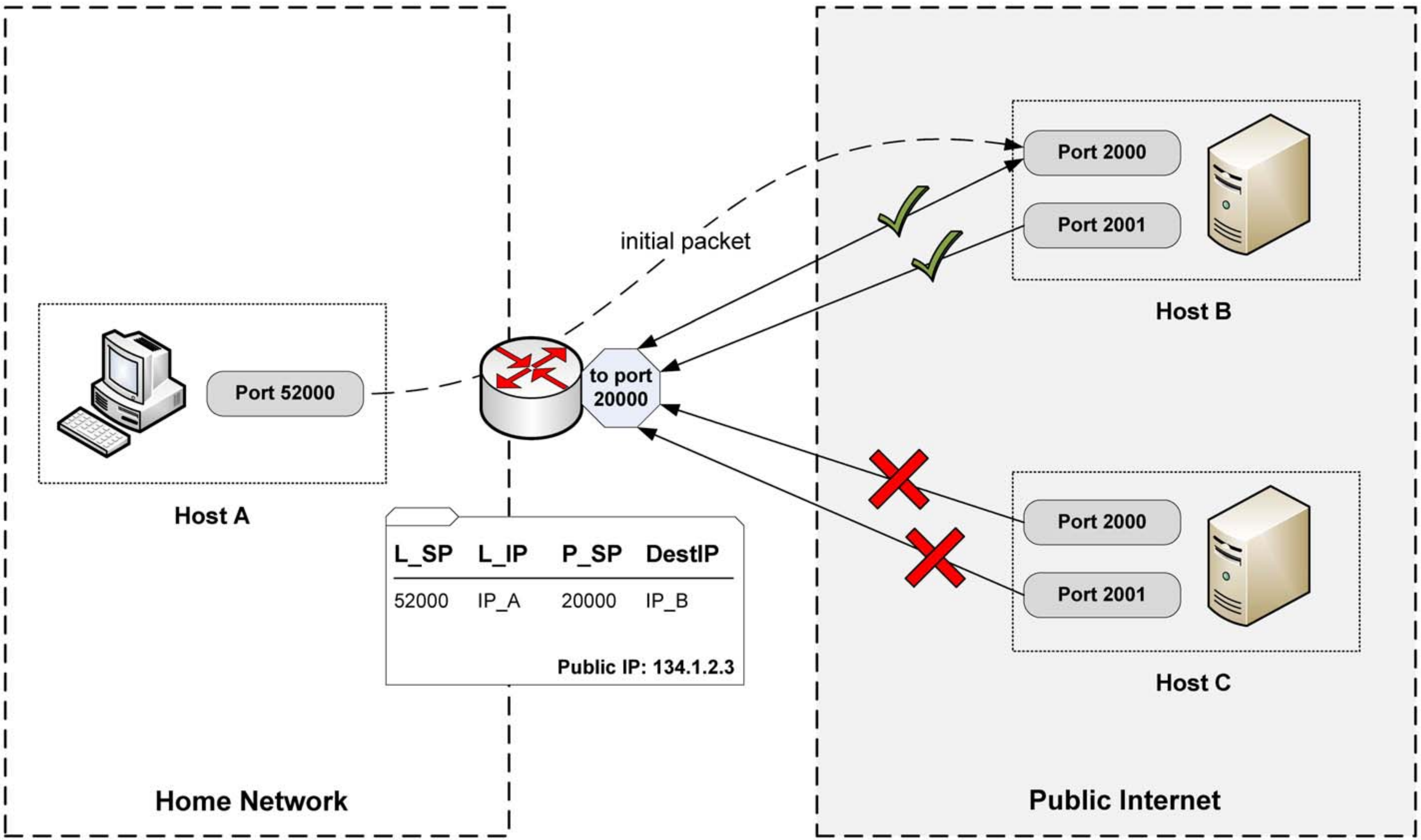
- Full Cone NAT
  - Endpoint independent
  - Independent filtering

- **Address Restricted NAT**
  - **Endpoint independent binding**
  - **Address restricted filtering**

- Port Address Restricted NAT
  - Endpoint independent binding
  - Port address restricted filtering

- Symmetric NAT
  - Endpoint dependent binding
  - Port address restricted filtering

Address Restricted Cone NAT



NAT Types

- With Binding and Filtering 4 NAT types can be defined (RFC 3489)

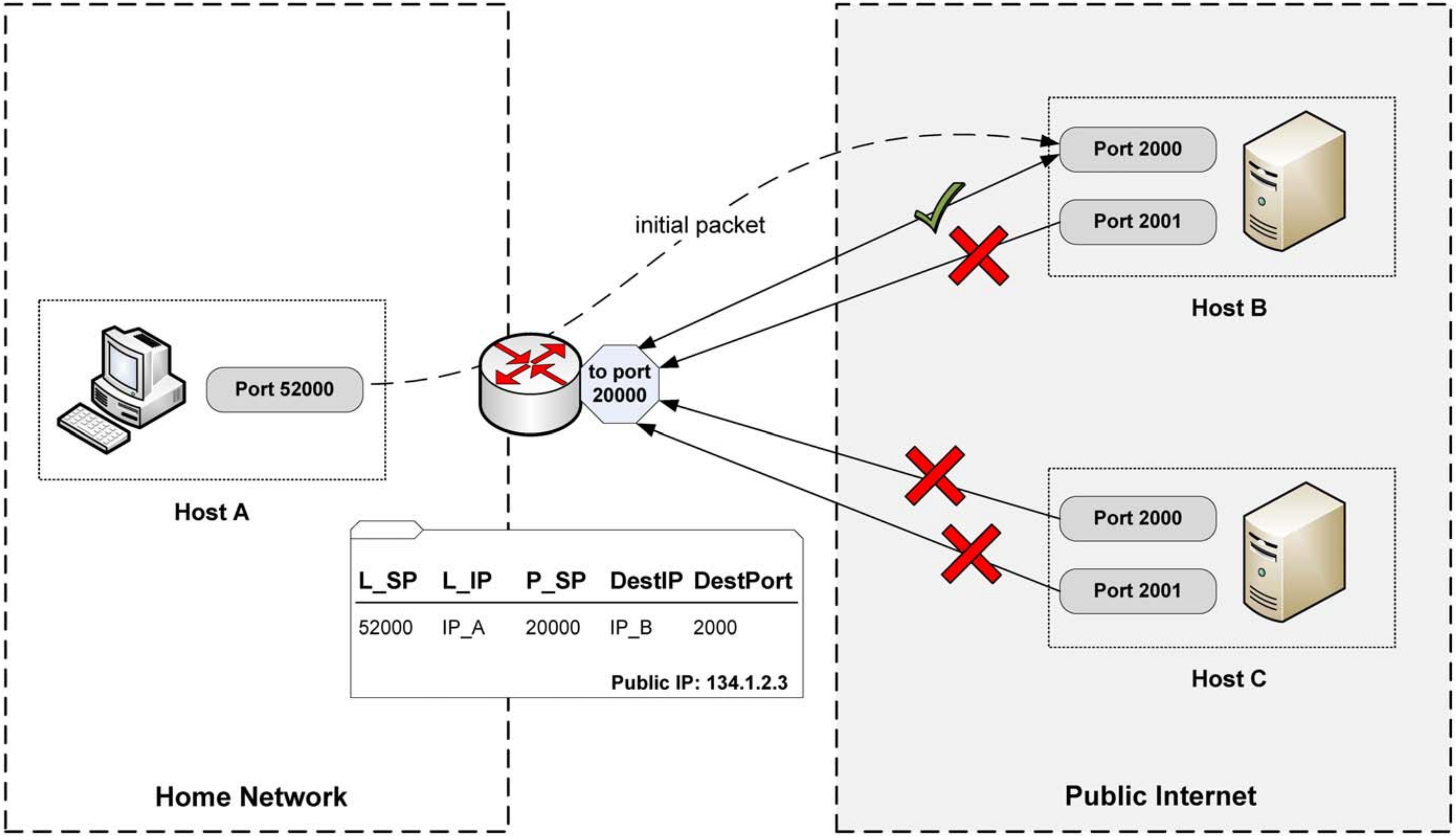
- Full Cone NAT
  - Endpoint independent
  - Independent filtering

- Address Restricted NAT
  - Endpoint independent binding
  - Address restricted filtering

- **Port Address Restricted NAT**
  - **Endpoint independent binding**
  - **Port address restricted filtering**

- Symmetric NAT
  - Endpoint dependent binding
  - Port address restricted filtering

Port Address Restricted Cone NAT



L_SP	L_IP	P_SP	DestIP	DestPort
52000	IP_A	20000	IP_B	2000

Public IP: 134.1.2.3

NAT Types

- With Binding and Filtering 4 NAT types can be defined (RFC 3489)

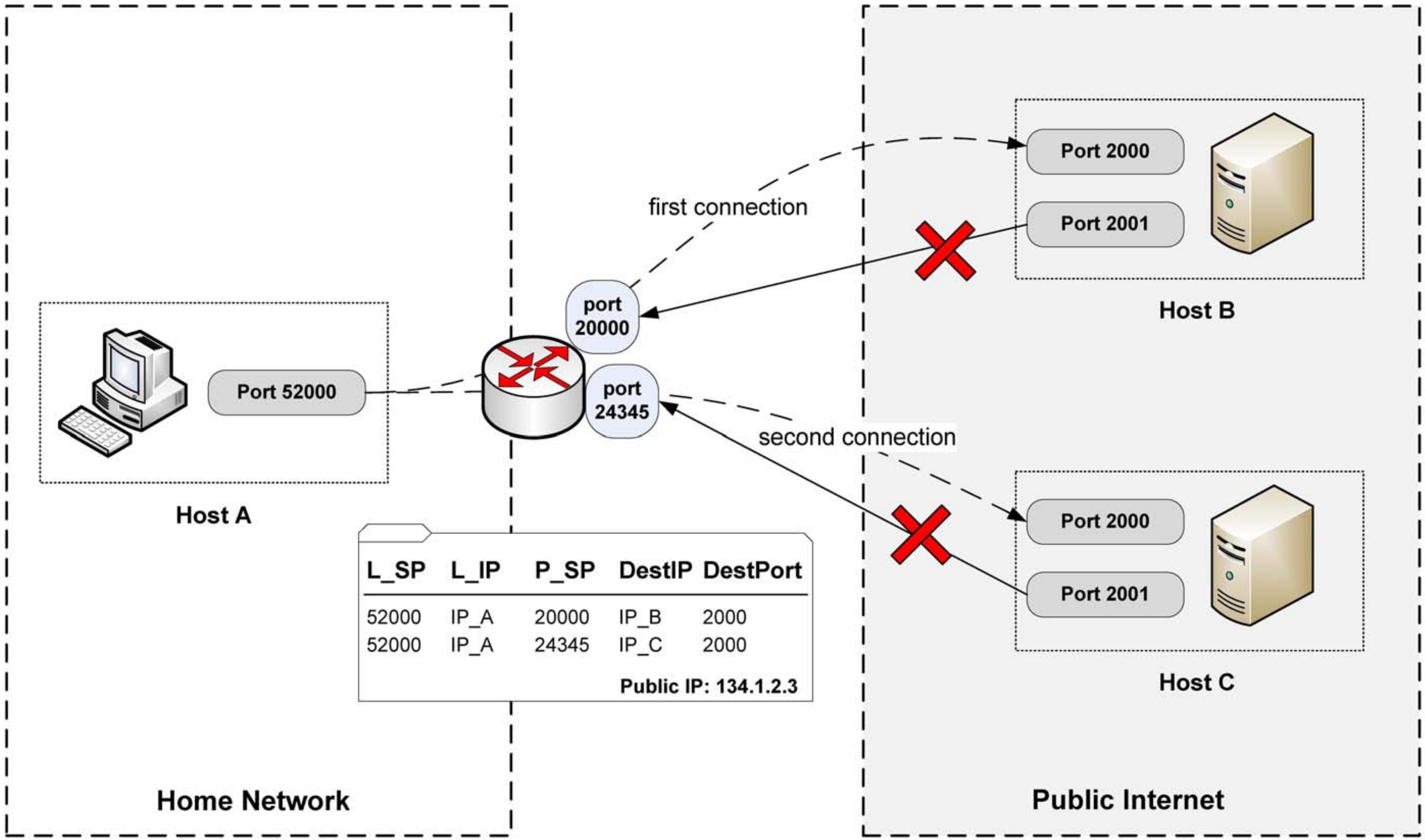
- Full Cone NAT
  - Endpoint independent
  - Independent filtering

- Address Restricted NAT
  - Endpoint independent binding
  - Address restricted filtering

- Port Address Restricted NAT
  - Endpoint independent binding
  - Port address restricted filtering

- **Symmetric NAT**
  - **Endpoint dependent binding**
  - **Port address restricted filtering**

Symmetric NAT



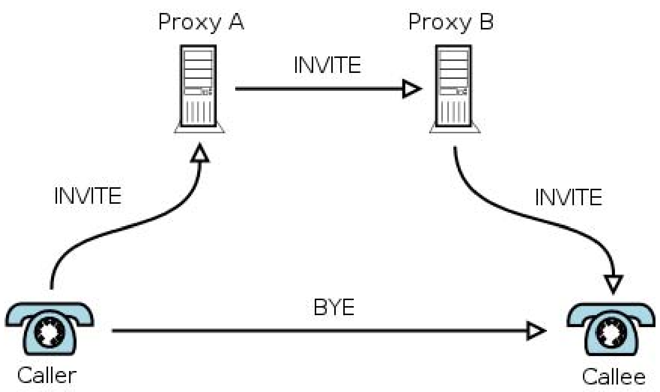
L_SP	L_IP	P_SP	DestIP	DestPort
52000	IP_A	20000	IP_B	2000
52000	IP_A	24345	IP_C	2000

Public IP: 134.1.2.3

NAT-Traversal Problem

- Divided into four categories: (derived from IETF-RFC 3027)
  - **Realm-Specific IP-Addresses in the Payload**
    - *SIP*
  - **Peer-to-Peer Applications**
    - *Any service behind a NAT*
  - **Bundled Session Applications (Inband Signaling)**
    - *FTP*
    - *RTSP*
    - *SIP together with SDP*
  - **Unsupported Protocols**
    - *SCTP*
    - *IPSec*

Example: Session Initiation Protocol (SIP)



Request/Respone Line	INVITE sip:Callee@200.3.4.5 SIP/2.0
Message-Header	Via: SIP/2.0/UDP **192.168.1.5:5060** From: < sip:Caller@**192.168.1.5** > To: <sip:Callee@200.3.4.5> CSeq: 1 INVITE Contact: <sip:Caller@192.168.1.5:5060> Content-Type: application/sdp
Message-Body (optional)	v=0 o=Alice 214365879 214365879 IN IP4 **192.168.1. 5** c=IN IP4 **192.168.1.5** t= 0 0 m=audio 5200 RTP/AVP 0 9 7 3 a=rtpmap:8 PCMU/8000 a=rtpmap:3 GSM/8000

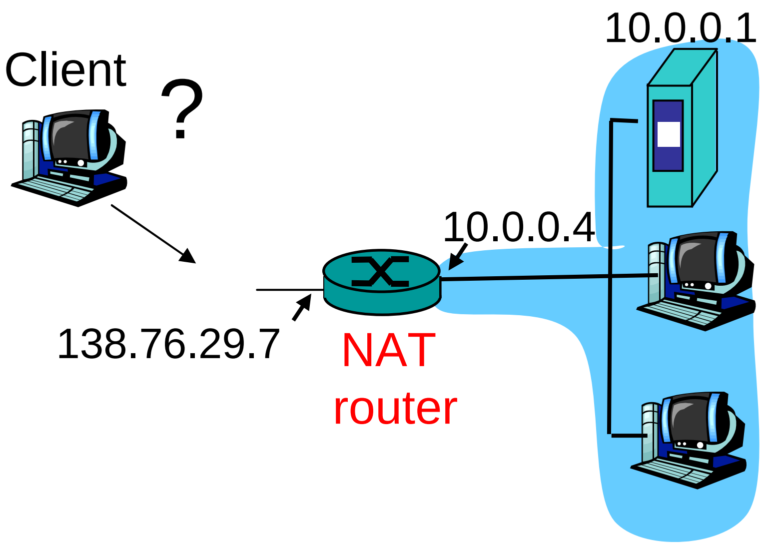
RTP-Session Specification (for 2nd channel) — v=0 … t= 0 0

Media description for 2nd channel — m=audio … a=rtpmap:3 GSM/8000

SDP — entire message body

example: p2p applications

- client wants to connect to server with address 10.0.0.1
  - server address 10.0.0.1 local to LAN (client can't use it as destination addr)
  - only one externally visible NATted address: 138.76.29.7
- solution 1: statically configure NAT to forward incoming connection requests at given port to server
  - e.g., (123.76.29.7, port 2500) always forwarded to 10.0.0.1 port 25000



Existing Solutions to the NAT-Traversal Problem

- Individual solutions
  - Explicit support by the NAT
    - static port forwarding, UPnP, NAT-PMP
  - NAT-behavior based approaches
    - dependent on knowledge about the NAT
    - hole punching using STUN (IETF - RFC 3489)
  - External Data-Relay
    - TURN (IETF - Draft)

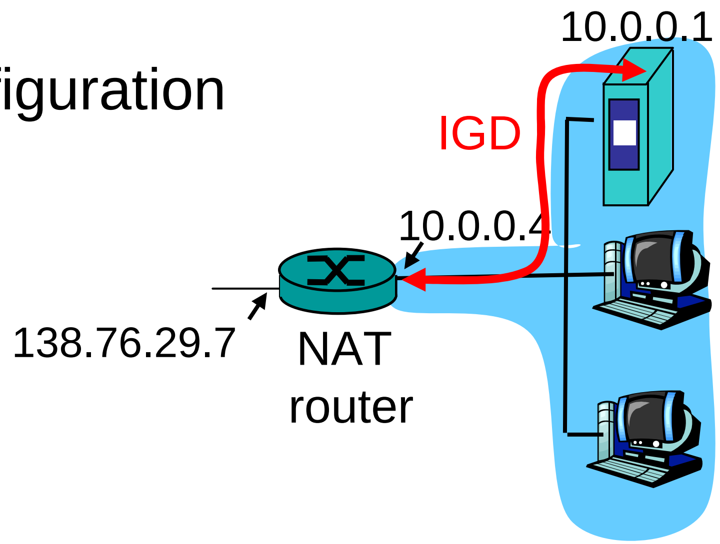
- Frameworks integrating several techniques
  - framework selects a working technique
  - ICE as the most promising for VoIP (IETF - Draft)

Explicit support by the NAT (1)

- Application Layer Gateway (ALG)
  - implemented on the NAT device and operates on layer 7
  - supports Layer 7 protocols that carry realm specific addresses in their payload
    - SIP, FTP
- Advantages
  - transparent for the application
  - no configuration necessary
- Drawbacks
  - protocol dependent (e.g. ALG for SIP, ALG for FTP...)
  - may or may not be available on the NAT device

Explicit support by the NAT (2)

- Universal Plug and Play (UPnP)
  - Automatic discovery of services (via Multicast)
  - Internet Gateway Device (IGD) for NAT-Traversal
- IGD allows NATed host to
  - automate static NAT port map configuration
  - learn public IP address (138.76.29.7)
  - add/remove port mappings (with lease times)



- Drawbacks
  - no security, evil applications can establish port forwarding entries
  - doesn‘t work with cascaded NATs

Behavior based (1): STUN

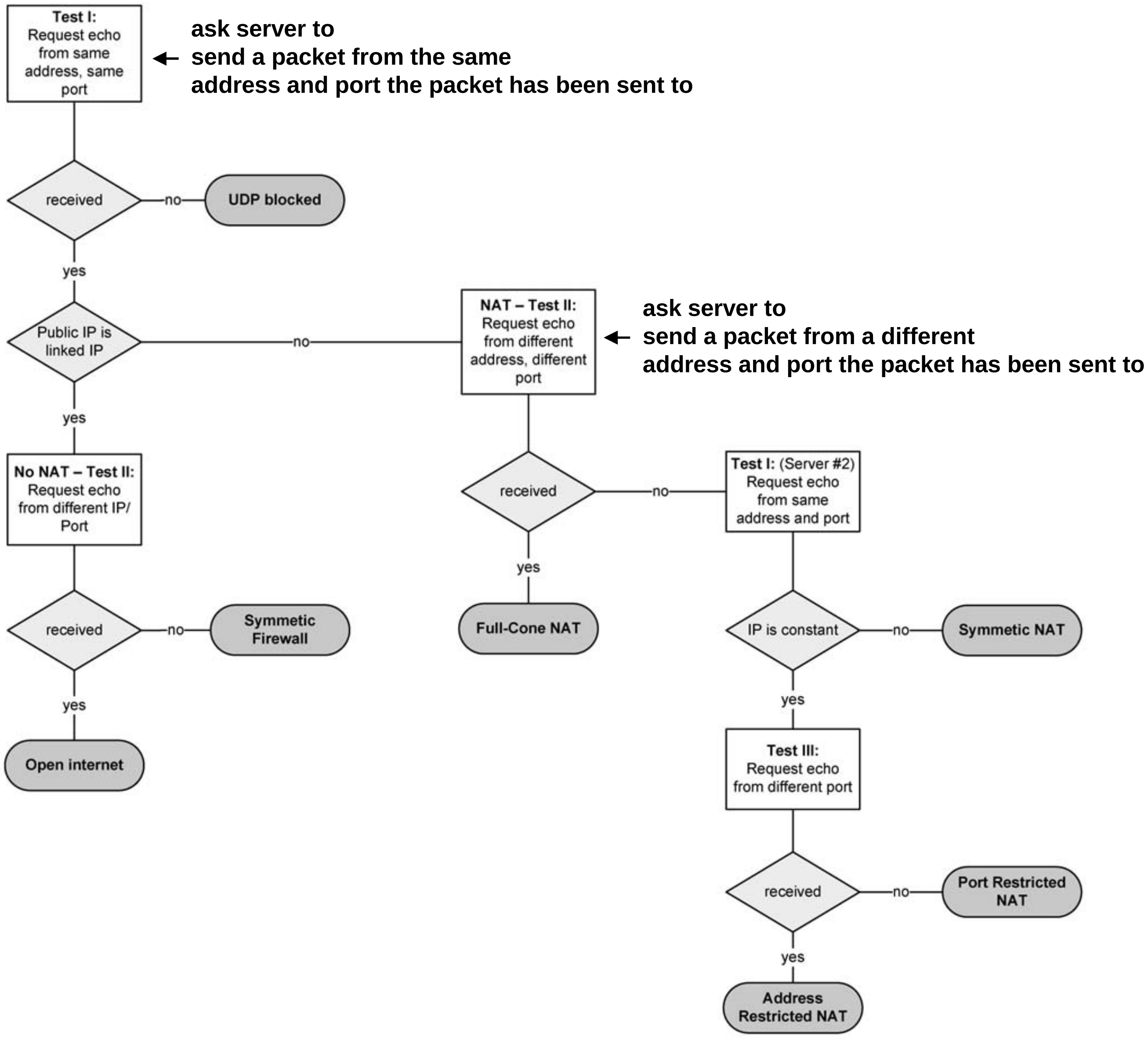
- Simple traversal of UDP through NAT (old) (RFC 3489)
  - Session Traversal Utilities for NAT (new) (RFC 5389)

- Lightweight client-server protocol
  - queries and responses via UDP (optional TCP or TCP/TLS)

- Helps to determine the external transport address (IP address and port) of a client.
  - e.g. query from 192.168.1.1:5060 results in 131.1.2.3:20000

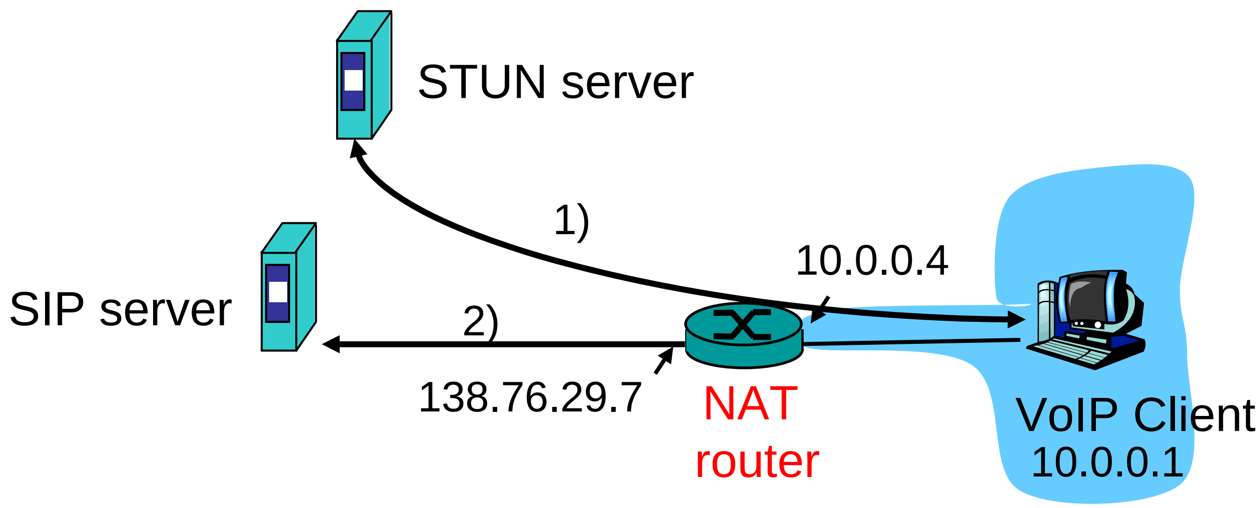
- Algorithm to discover NAT type
  - server needs 2 public IP addresses

STUN Algorithm



Example: STUN and SIP

- VoIP client queries STUN server
  - learns its public transport address
  - can be used in SIP packets



Request/Respone Line	INVITE sip:Callee@200.3.4.5 SIP/2.0
Message-Header	Via: SIP/2.0/UDP **138.76.29.7:5060** From: < sip:Caller@**138.76.29.7** > To: <sip:Callee@200.3.4.5> CSeq: 1 INVITE Contact: <sip:Caller@**138.76.29.7:5060**> Content-Type: application/sdp

Limitations of STUN

- STUN only works if
  - the NAT assigns the external port (and IP address) only based on the source transport address
  - Endpoint independent NAT binding
    - Full Cone NAT
    - Address Restricted Cone NAT
    - Port Address restricted cone NAT
  - Not with symmetric NAT!

- Why?
  - Since we first query the STUN server (different IP and port) and then the actual server

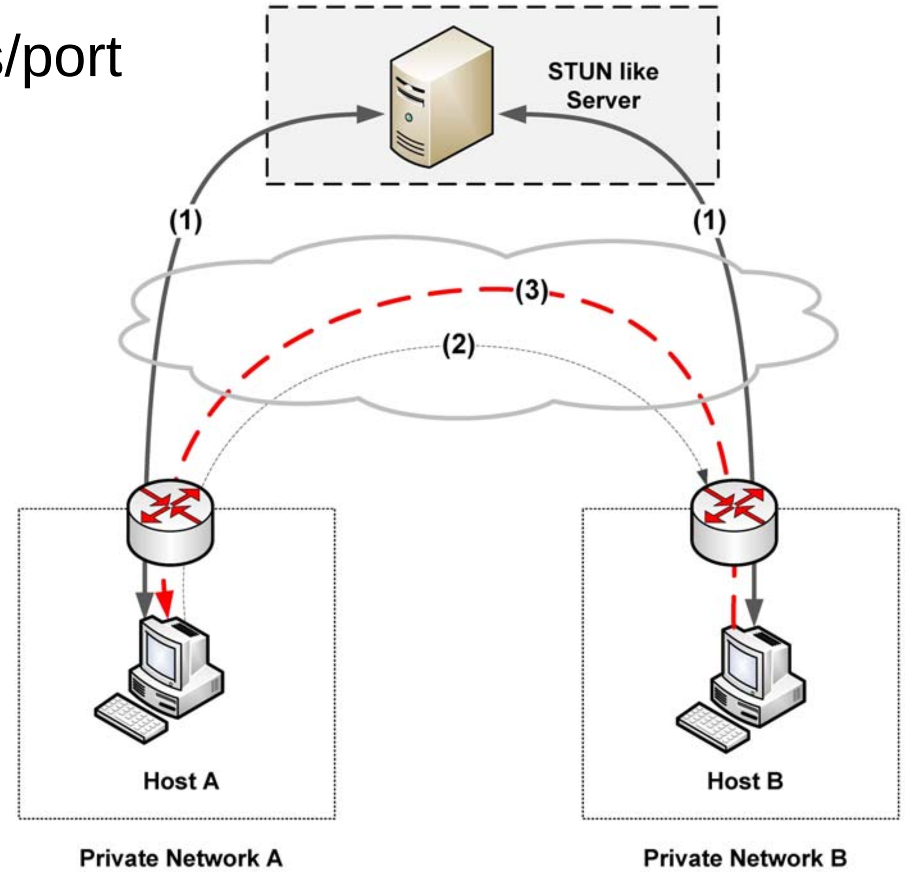
STUN and Hole Punching

- STUN not only helps if we need IP addresses in the payload
  - for establishing a direct connection between two peers

1) determine external IP address/port and exchange it through Rendezvous Point

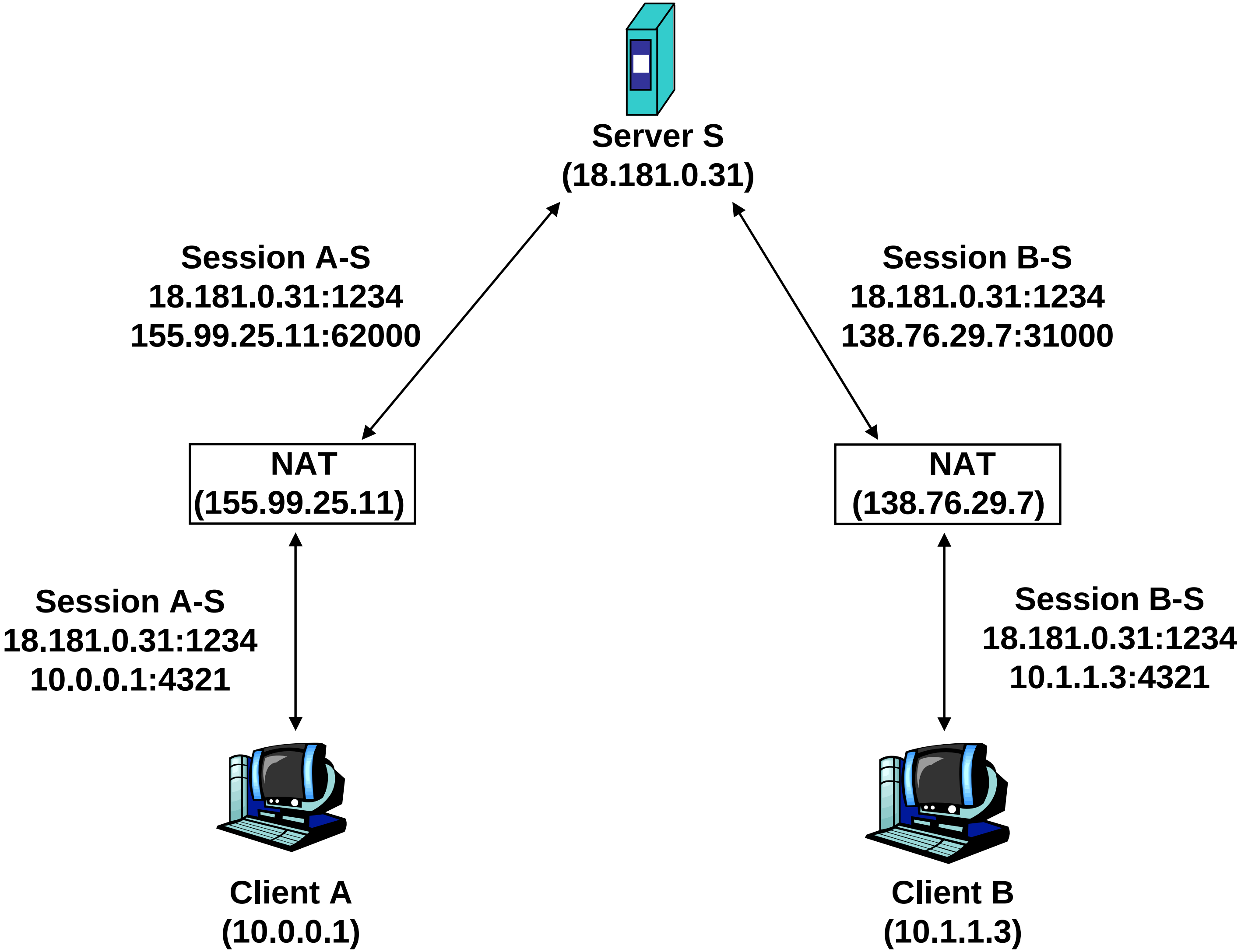
2) both hosts send packets towards the other host outgoing packet creates hole

3) establish connection. hole is created by first packet



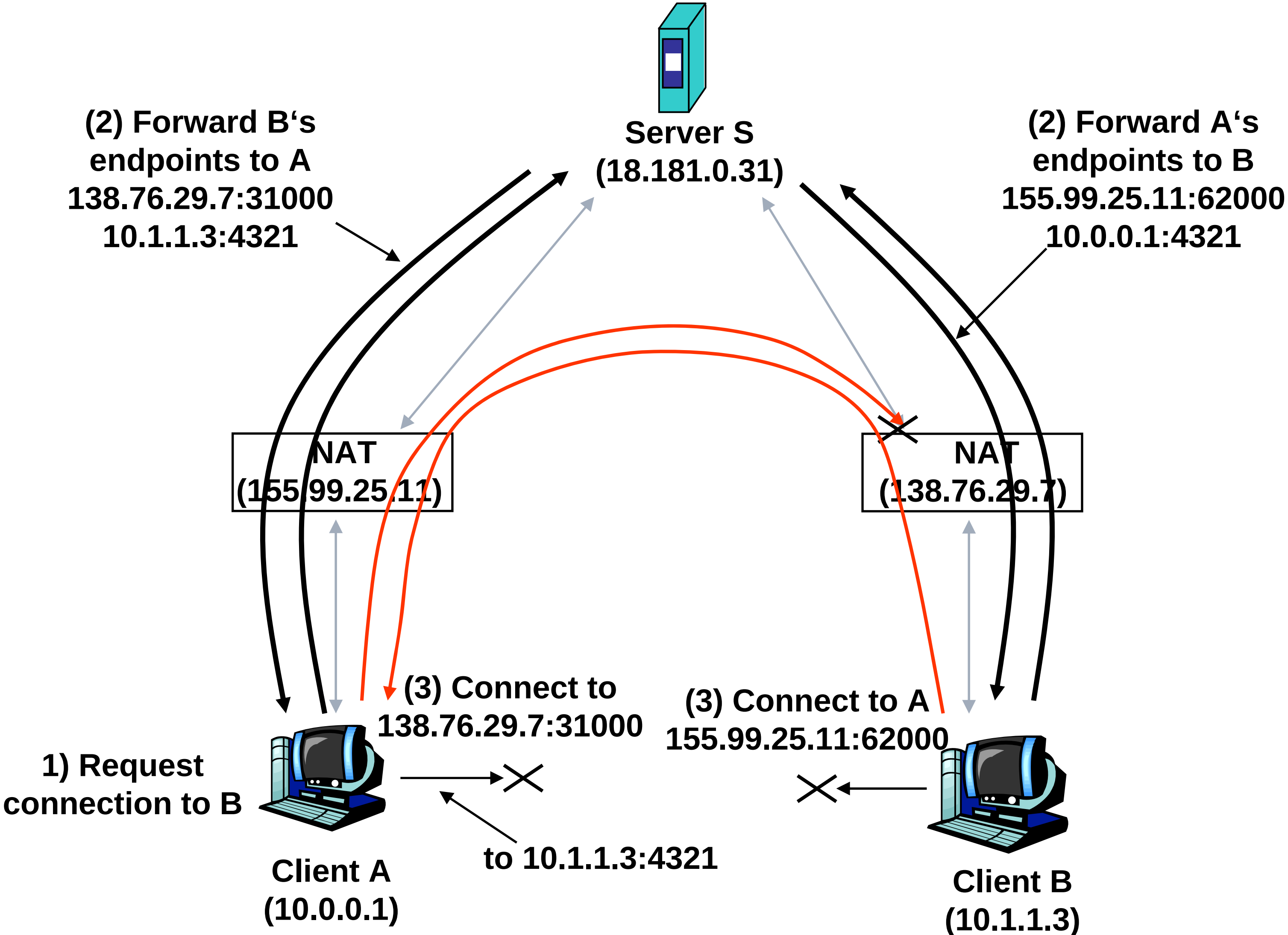
Hole Punching in detail

- Before hole punching



Hole Punching in detail

- Hole punching



DIY Hole Punching: practical example

- You need 2 hosts
  - One in the public internet (client)
  - One behind a NAT (server)

- Firstly start a UDP listener on UDP port 20000 on the “server” console behind the NAT/firewall
  - server/1# nc -u -l -p 20000

- An external computer “client" then attempts to contact it
  - client# echo "hello" | nc -p 5000 -u serverIP 20000
  - Note: 5000 is the source port of the connection

- as expected nothing is received because the NAT has no state

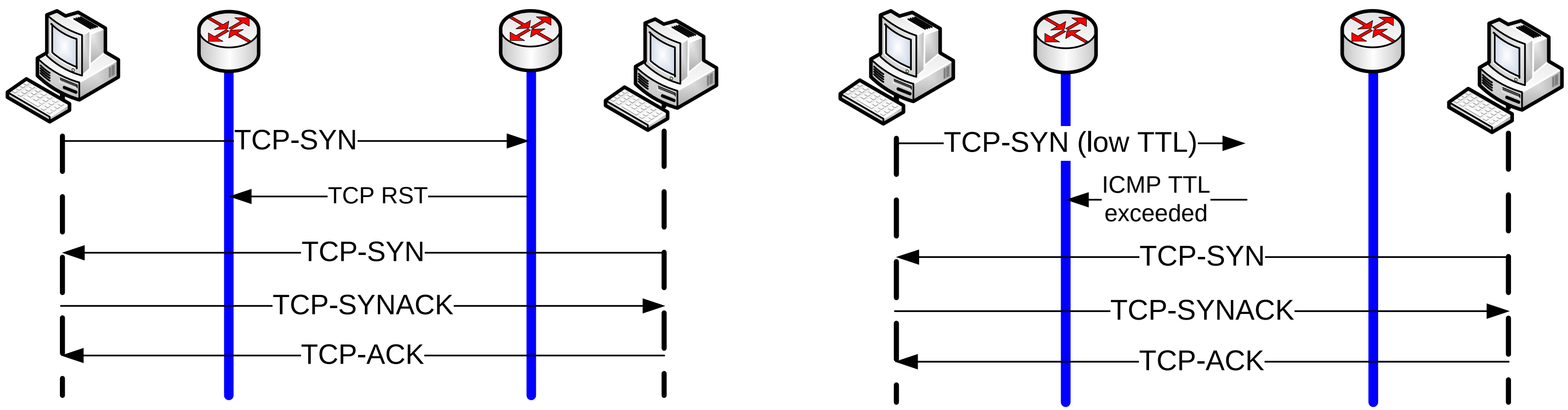
- Now on a second console, server/2, we punch a hole
  - Server/2# hping2 -c 1 -2 -s 20000 -p 5000 clientIP

- On the second attempt we connect to the created hole
  - client# echo "hello" | nc -p 5000 -u serverIP 20000

TCP Hole Punching

- Hole Punching not straight forward due to stateful design of TCP
  - 3-way handshake
  - Sequence numbers
  - ICMP packets may trigger RST packets

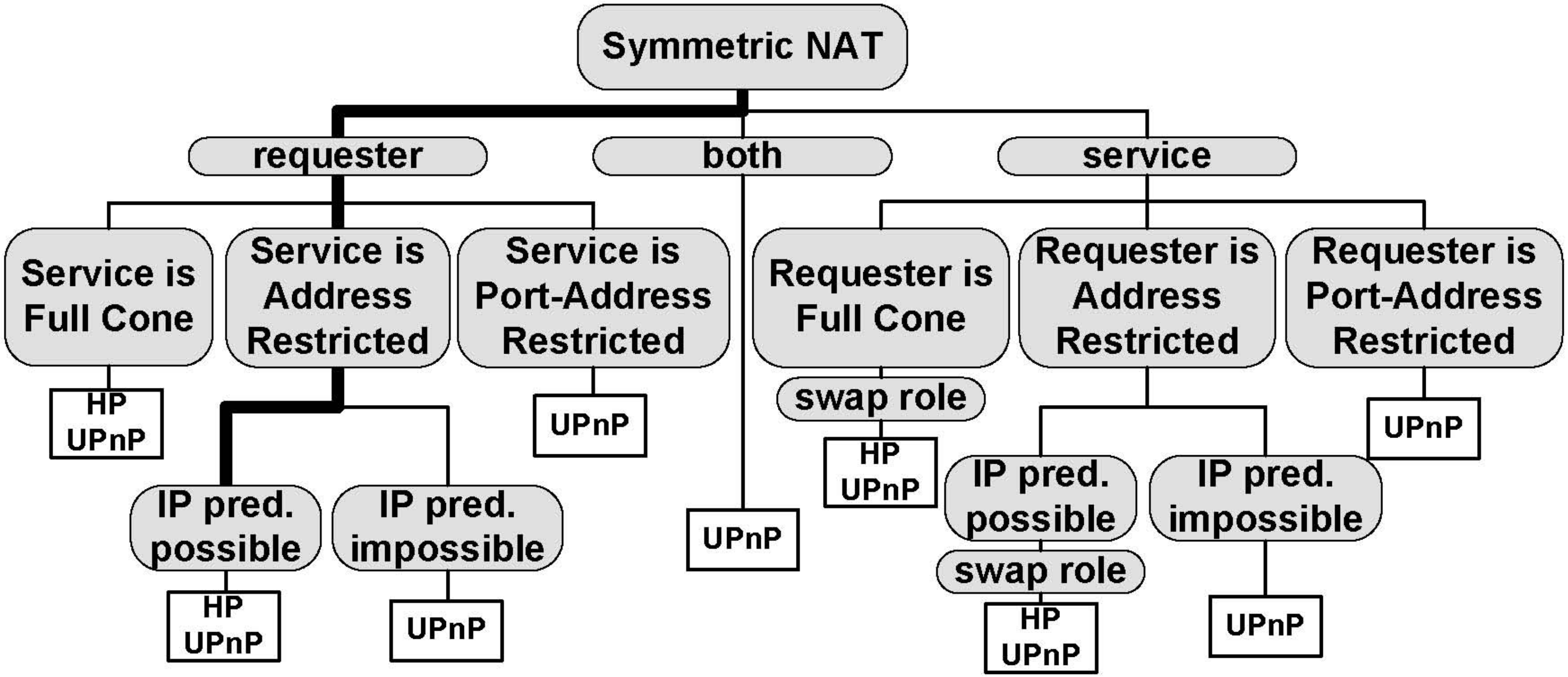
- Low/high TTL(Layer 3) of Hole-Punching packet
  - As implemented in STUNT (Cornell University)



- Bottom line: NAT is not standardized

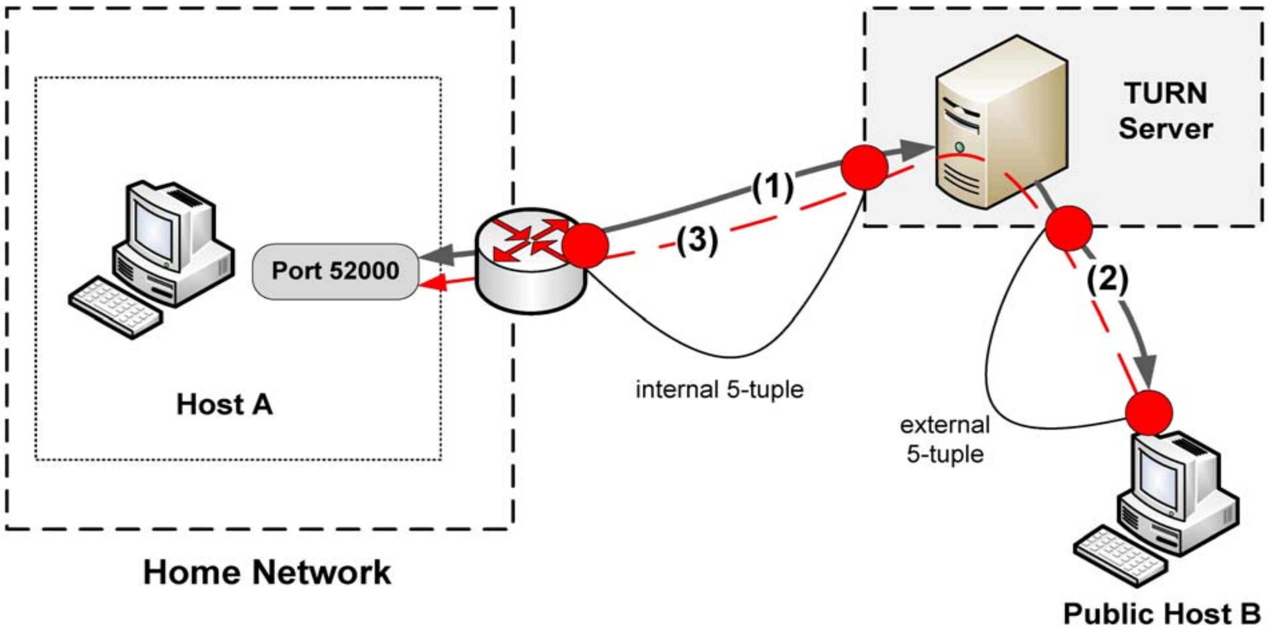
Symmetric NATs

- How can we traverse symmetric NATs
  - Endpoint dependent binding
    - hole punching in general only if port prediction is possible
  - Address and port restricted filtering



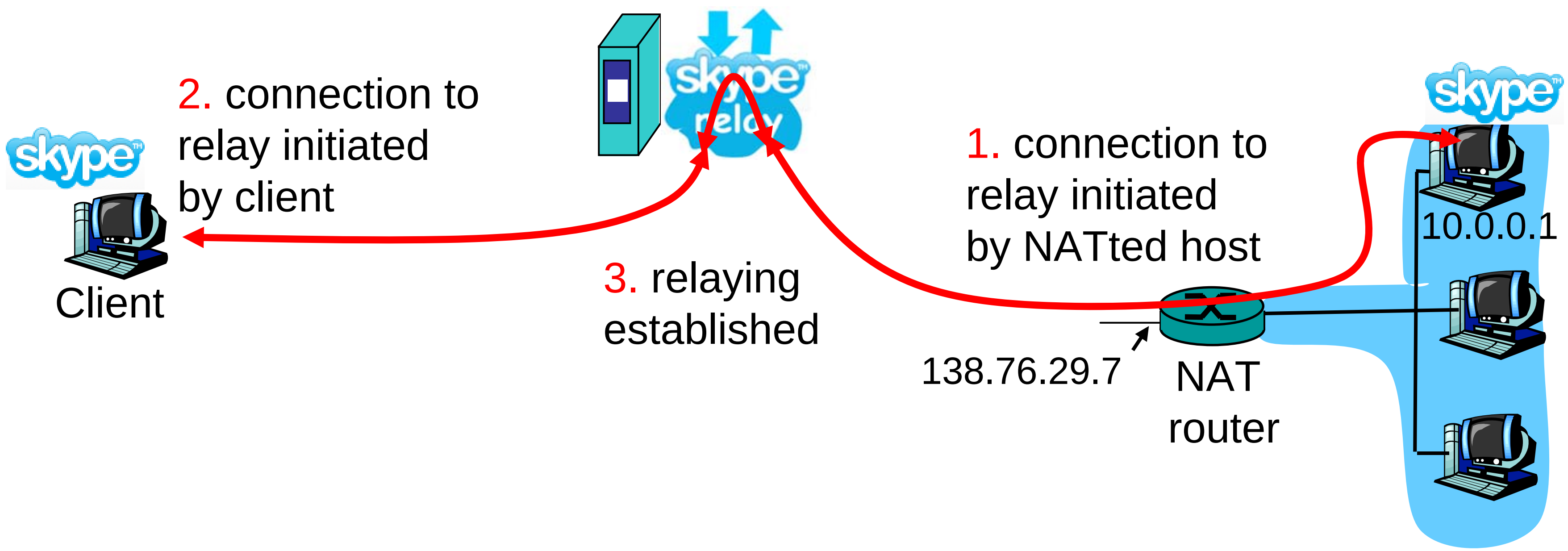
Data Relay (1)

- Idea: Outbound connections are always possible
- 3rd party (relay server) in the public internet
- Both hosts actively establish a connection to relay server
- Relay server forwards packets between these hosts
- TURN as IETF draft



Data Relay

- relaying (used in Skype)
  - NATed client establishes connection to relay
  - External client connects to relay
  - relay bridges packets between to connections
  - IETF draft: TURN



Frameworks

- Interactive Connectivity Establishment (ICE)
  - IETF draft
  - mainly developed for VoIP
  - signaling messages embedded in SIP/SDP

- All possible endpoints are collected and exchanged during call setup
  - local addresses
  - STUN determined
  - TURN determined

- All endpoints are „paired“ and tested (via STUN)
  - best one is determined and used for VoIP session

- Advantages
  - high sucess rate
  - integrated in application

- Drawbacks
  - overhead
  - latency dependent on number of endpoints (pairing)

Success Rates for existing solutions

- http://nattest.net.in.tum.de

- UPnP 31 %

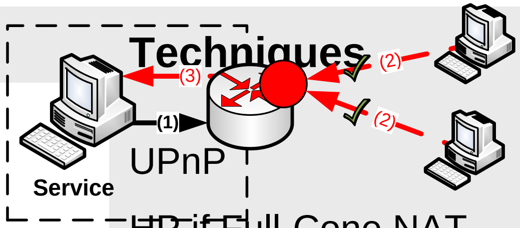
- Hole Punching
  - UDP 73%
  - TCP low TTL 42%
  - TCP high TTL 35%

- Relay 100%

- Propabilities for a direct connection
  - UDP Traversal: 85 %
  - TCP Traversal: 82 %
  - TCP inclusive tunneling: 95 %

Service Categories for NAT-Traversal (TUM)

- Global Service-Provisioning (GSP)
  - Globally accessible public endpoint
  - Only the service host needs software support



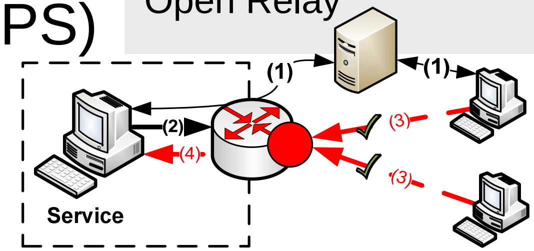
Techniques

UPnP

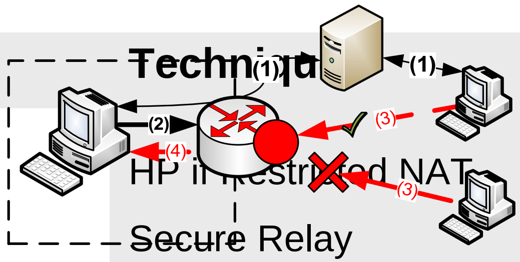
HP if Full-Cone NAT

Open Relay

- Service-Provisioning using Pre-Signaling (SPPS)
  - Pre-Signaling through Rendezvous-Point
  - No assumptions about NAT-Traversal techniques
  - Both hosts need software support



- Secure Service-Provisioning (SSP)
  - Extension for SPPS
  - Only authorized users can allocate mappings
  - Created mapping can only be accessed by the creator



Techniques

HP if Restricted NAT

Secure Relay

- ALG Service-Provisioning (ALG-SP)
  - Explicit support for Layer 7 protocols (SIP-VoIP)

New approach (TUM)

- Advanced NAT-Traversal Service (ANTS)
  - considers different service categories
    - who runs framework
    - which external entities are available?
  - pre-signaling and security
  - knowledge based
    - NAT-Traversal decision is made upon knowledge
  - performance
    - Less latency through knowledge based approach
  - success rates
    - 95% for a direct connection for TCP
  - available for new (API) and legacy applications (TUN)

- for more information
  - http://nattest.net.in.tum.de/?mod=publications

NAT Conclusion

- NAT helps against the shortage of IPv4 addresses
  - only the border gateway needs a public IP address
  - NAT maintains mapping table and translates addresses
- NAT works as long as the server part is in the public internet
- P2P communication across NAT is difficult
  - NAT breaks the end-to-end connectivity model
- NAT behavior is not standardized
  - keep that in mind when designing a protocol
- many solutions for the NAT-Traversal problem
  - none of them works with all NATs
  - framework can select the most appropriate technique

NAT and IPv6

- IPv6 provides a 128bit address field
  - do we still need NAT?
- Firewall traversal
  - bundled session applications
- Topology hiding
  - „security“
- Business models of ISPs
  - how many IP addresses do we really get (for free)?
- NAT for IPv6 (NAT66) standardization already started (IETF)
  - goal: „well behaved NAT“