Master Course Computer Networks IN2097

Prof. Dr.-Ing. Georg Carle
Christian Grothoff, Ph.D.
Stephan Günther
Andreas Müller (mueller@net.in.tum.de)

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

Recap

- NAT behavior on outgoing packets and incoming packets
  - Binding: Port and NAT
    - Endpoint/Connection independent vs. dependent
  - Filtering: independent vs. (port/address) restricted
  - Processing Model for describing individual steps

- NAT Traversal Problem
  - Realm specific IP addresses in the payload
  - P2P services
  - Bundled Session Applications
  - Unsupported protocol

- NAT Traversal techniques
  - Behavior based vs. active support by the NAT/ext. entities

Agenda

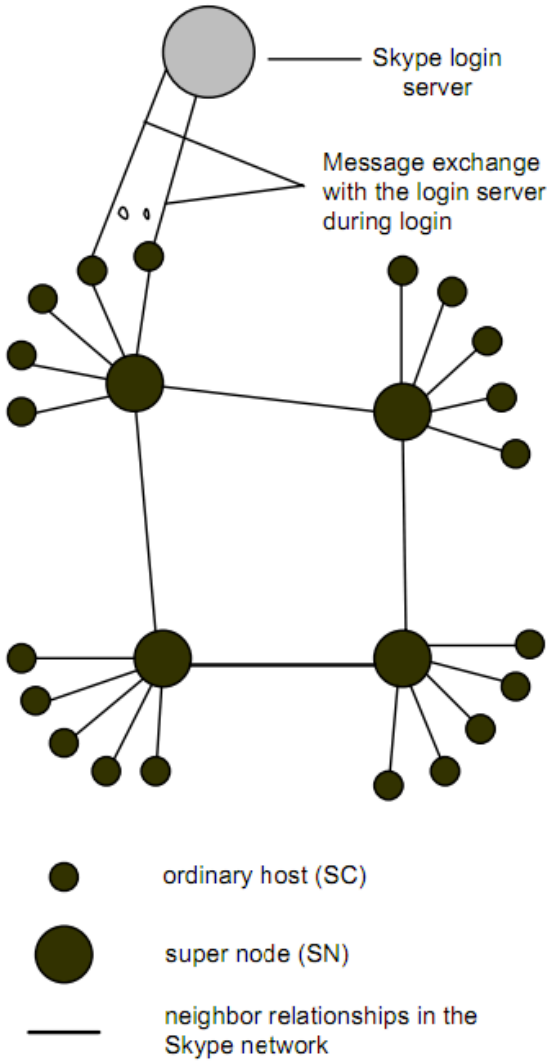
- Skype as one example for an application that is known to work in many environments
- Large Scale Network Address Translation
  - At provider side
  - Challenges and approaches
- Middleboxes
  - Today's Internet Architecture
  - Classification of and reasons for middleboxes
  - Behavior of middleboxes
  - Concerns

Skype

- Closed source P2P VoIP and IM Client
- Many techniques to make reverse engineering difficult
  - Code obfuscation
  - Payload obfuscation
- Known to work in most environments
- Extensive use of NAT Traversal techniques
  - STUN
  - Hole Punching
  - Relaying
  - UPnP
  - Port Prediction

Skype components

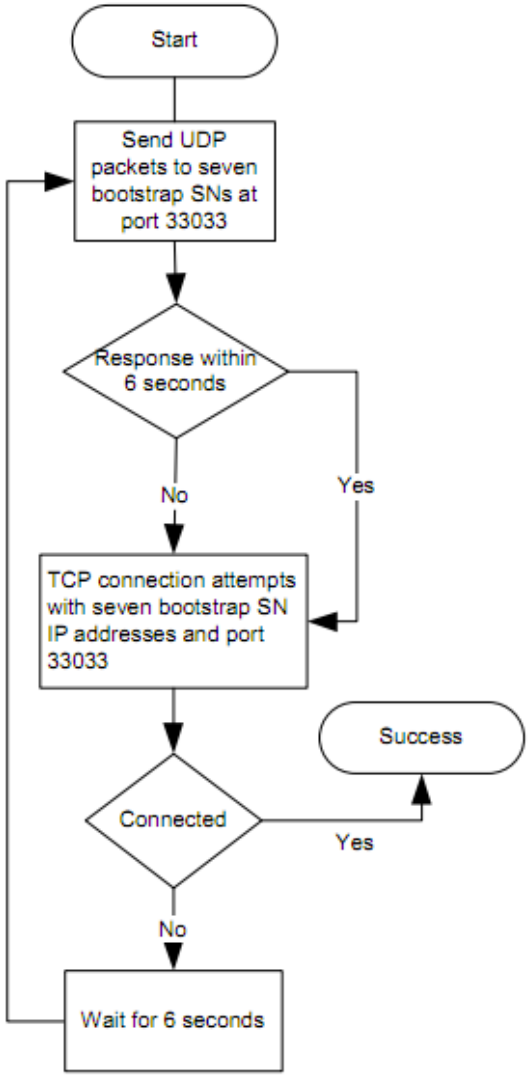
- Ordinary host (OH)
  - a Skype client (SC)
- Super nodes (SN)
  - a Skype client
  - has public IP address
  - sufficient bandwidth
  - CPU and memory
- Login server
  - stores Skype id's, passwords, and buddy lists
  - used at login for authentication



http://www.cs.columbia.edu/~salman/publications/skype1_4.pdf

„Join“ process

- Tasks performed
  - User authentication
  - Presence advertisement
  - Determine the type of NAT
  - Discover other Skype nodes
  - Check availability of latest software
- Needs to connect to at least one SN
  - SNs used for signaling
  - Host Cache holds ~200 SNs
  - 7 Skype bootstrap SN as last resort



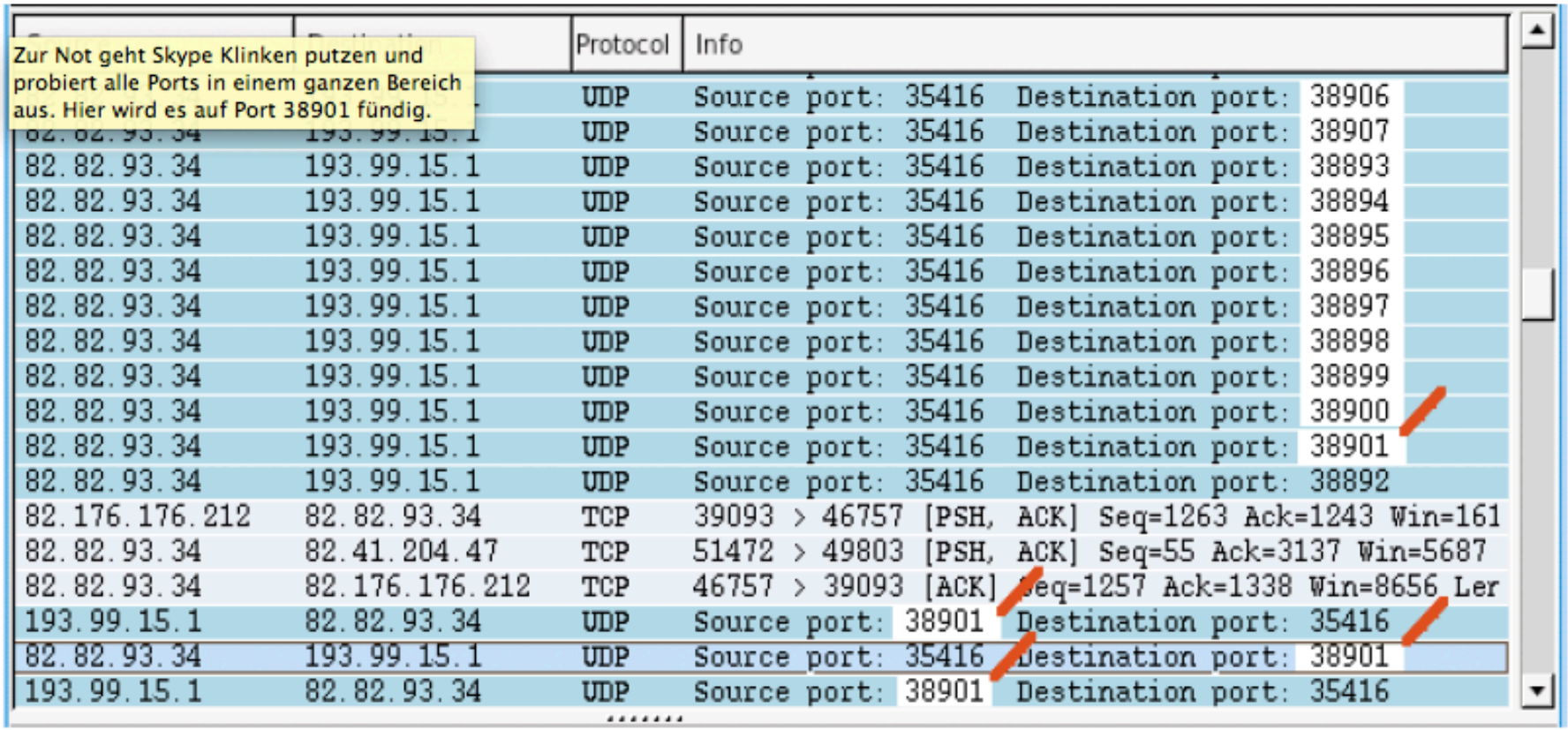
http://www.cs.columbia.edu/~salman/publications/skype1_4.pdf

NAT Traversal

- Ports
  - Randomly chosen (configurable) TCP and UDP port for the Skype client
  - Additionally: listen at port 80 and 443 if possible
    - If you become a SN
    - outgoing connections to 80/443 are usually possible for all SCs

- Skype SNs used as Rendezvous Points
  - SN acts as STUN-like server to determine external mappings
  - Signaling and exchange of public endpoints for HP
  - Used as relays if necessary
  - Otherwise, no centralized NAT helper

Hole Punching in Skype

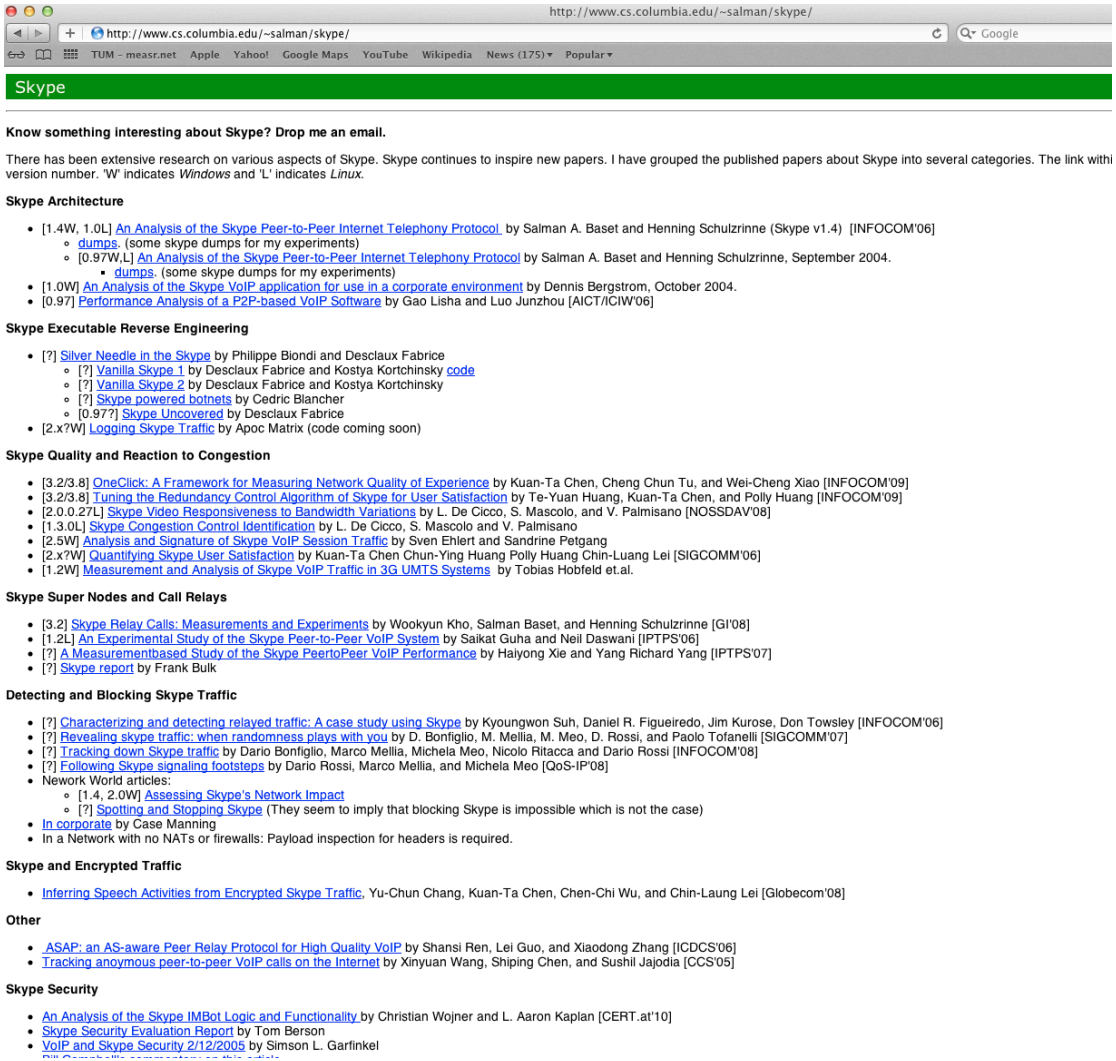


		Protocol	Info
		UDP	Source port: 35416 Destination port: 38906
		UDP	Source port: 35416 Destination port: 38907
82.82.93.34	193.99.15.1	UDP	Source port: 35416 Destination port: 38893
82.82.93.34	193.99.15.1	UDP	Source port: 35416 Destination port: 38894
82.82.93.34	193.99.15.1	UDP	Source port: 35416 Destination port: 38895
82.82.93.34	193.99.15.1	UDP	Source port: 35416 Destination port: 38896
82.82.93.34	193.99.15.1	UDP	Source port: 35416 Destination port: 38897
82.82.93.34	193.99.15.1	UDP	Source port: 35416 Destination port: 38898
82.82.93.34	193.99.15.1	UDP	Source port: 35416 Destination port: 38899
82.82.93.34	193.99.15.1	UDP	Source port: 35416 Destination port: 38900
82.82.93.34	193.99.15.1	UDP	Source port: 35416 Destination port: 38901
82.82.93.34	193.99.15.1	UDP	Source port: 35416 Destination port: 38892
82.176.176.212	82.82.93.34	TCP	39093 > 46757 [PSH, ACK] Seq=1263 Ack=1243 Win=161
82.82.93.34	82.41.204.47	TCP	51472 > 49803 [PSH, ACK] Seq=55 Ack=3137 Win=5687
82.82.93.34	82.176.176.212	TCP	46757 > 39093 [ACK] Seq=1257 Ack=1338 Win=8656 Ler
193.99.15.1	82.82.93.34	UDP	Source port: 38901 Destination port: 35416
82.82.93.34	193.99.15.1	UDP	Source port: 35416 Destination port: 38901
193.99.15.1	82.82.93.34	UDP	Source port: 38901 Destination port: 35416

http://www.heise.de/security/artikel/Klinken-putzen-271494.html

More on Skype

- http://www.cs.columbia.edu/~salman/skype/



http://www.cs.columbia.edu/~salman/skype/

http://www.cs.columbia.edu/~salman/skype/

TUM - measr.net Apple Yahoo! Google Maps YouTube Wikipedia News (175) Popular

Skype

Know something interesting about Skype? Drop me an email.

There has been extensive research on various aspects of Skype. Skype continues to inspire new papers. I have grouped the published papers about Skype into several categories. The link withi version number. 'W' indicates *Windows* and 'L' indicates *Linux*.

Skype Architecture

- [1.4W, 1.0L] An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol by Salman A. Baset and Henning Schulzrinne (Skype v1.4) [INFOCOM'06]
  - dumps. (some skype dumps for my experiments)
  - [0.97W,L] An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol by Salman A. Baset and Henning Schulzrinne, September 2004.
    - dumps. (some skype dumps for my experiments)
- [1.0W] An Analysis of the Skype VoIP application for use in a corporate environment by Dennis Bergstrom, October 2004.
- [0.97] Performance Analysis of a P2P-based VoIP Software by Gao Lisha and Luo Junzhou [AICT/ICIW'06]

Skype Executable Reverse Engineering

- [?] Silver Needle in the Skype by Philippe Biondi and Desclaux Fabrice
  - [?] Vanilla Skype 1 by Desclaux Fabrice and Kostya Kortchinsky code
  - [?] Vanilla Skype 2 by Desclaux Fabrice and Kostya Kortchinsky
  - [?] Skype powered botnets by Cedric Blancher
  - [0.97?] Skype Uncovered by Desclaux Fabrice
- [2.x?W] Logging Skype Traffic by Apoc Matrix (code coming soon)

Skype Quality and Reaction to Congestion

- [3.2/3.8] OneClick: A Framework for Measuring Network Quality of Experience by Kuan-Ta Chen, Cheng Chun Tu, and Wei-Cheng Xiao [INFOCOM'09]
- [3.2/3.8] Tuning the Redundancy Control Algorithm of Skype for User Satisfaction by Te-Yuan Huang, Kuan-Ta Chen, and Polly Huang [INFOCOM'09]
- [2.0.0.27L] Skype Video Responsiveness to Bandwidth Variations by L. De Cicco, S. Mascolo, and V. Palmisano [NOSSDAV'08]
- [1.3.0L] Skype Congestion Control Identification by L. De Cicco, S. Mascolo and V. Palmisano
- [2.5W] Analysis and Signature of Skype VoIP Session Traffic by Sven Ehlert and Sandrine Petgang
- [2.x?W] Quantifying Skype User Satisfaction by Kuan-Ta Chen Chun-Ying Huang Polly Huang Chin-Luang Lei [SIGCOMM'06]
- [1.2W] Measurement and Analysis of Skype VoIP Traffic in 3G UMTS Systems by Tobias Hobfeld et.al.

Skype Super Nodes and Call Relays

- [3.2] Skype Relay Calls: Measurements and Experiments by Wookyun Kho, Salman Baset, and Henning Schulzrinne [GI'08]
- [1.2L] An Experimental Study of the Skype Peer-to-Peer VoIP System by Saikat Guha and Neil Daswani [IPTPS'06]
- [?] A Measurementbased Study of the Skype PeertoPeer VoIP Performance by Haiyong Xie and Yang Richard Yang [IPTPS'07]
- [?] Skype report by Frank Bulk

Detecting and Blocking Skype Traffic

- [?] Characterizing and detecting relayed traffic: A case study using Skype by Kyoungwon Suh, Daniel R. Figueiredo, Jim Kurose, Don Towsley [INFOCOM'06]
- [?] Revealing skype traffic: when randomness plays with you by D. Bonfiglio, M. Mellia, M. Meo, D. Rossi, and Paolo Tofanelli [SIGCOMM'07]
- [?] Tracking down Skype traffic by Dario Bonfiglio, Marco Mellia, Michela Meo, Nicolo Ritacca and Dario Rossi [INFOCOM'08]
- [?] Following Skype signaling footsteps by Dario Rossi, Marco Mellia, and Michela Meo [QoS-IP'08]
- Nework World articles:
  - [1.4, 2.0W] Assessing Skype's Network Impact
  - [?] Spotting and Stopping Skype (They seem to imply that blocking Skype is impossible which is not the case)
- In corporate by Case Manning
- In a Network with no NATs or firewalls: Payload inspection for headers is required.

Skype and Encrypted Traffic

- Inferring Speech Activities from Encrypted Skype Traffic, Yu-Chun Chang, Kuan-Ta Chen, Chen-Chi Wu, and Chin-Laung Lei [Globecom'08]

Other

- ASAP: an AS-aware Peer Relay Protocol for High Quality VoIP by Shansi Ren, Lei Guo, and Xiaodong Zhang [ICDCS'06]
- Tracking anoymous peer-to-peer VoIP calls on the Internet by Xinyuan Wang, Shiping Chen, and Sushil Jajodia [CCS'05]

Skype Security

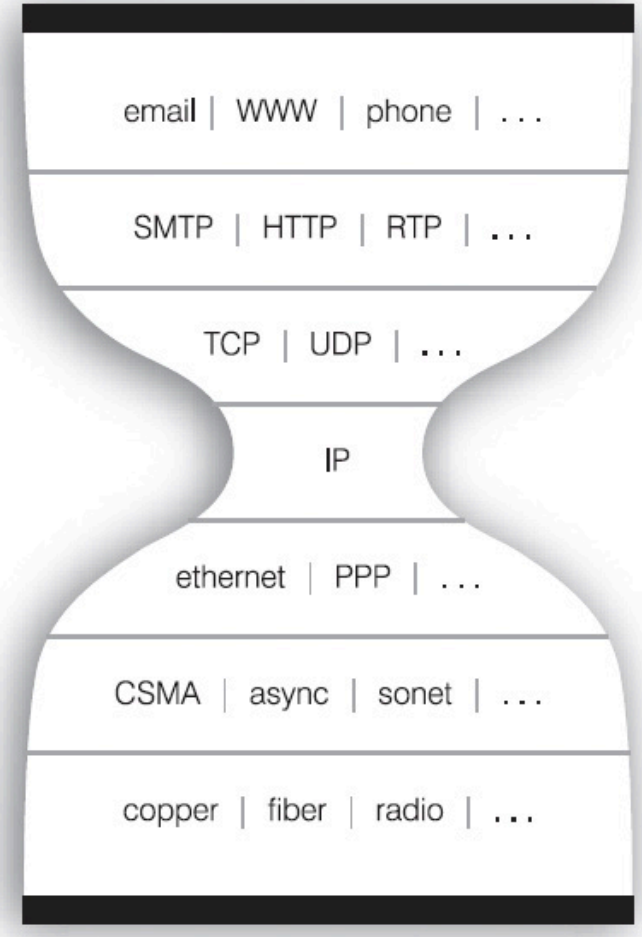
- An Analysis of the Skype IMBot Logic and Functionality by Christian Wojner and L. Aaron Kaplan [CERT.at'10]
- Skype Security Evaluation Report by Tom Berson
- VoIP and Skype Security 2/12/2005 by Simson L. Garfinkel

Recap: Problem

- More and more devices connect to the Internet
  - PCs
  - Cell phones
  - Internet radios
  - TVs
  - Home appliances
  - Future: sensors, cars...

- IP addresses need to be globally unique
  - IPv4 provides a 32bit field
  - Many addresses not usable because of classful allocation

→ We are running out of IP addresses

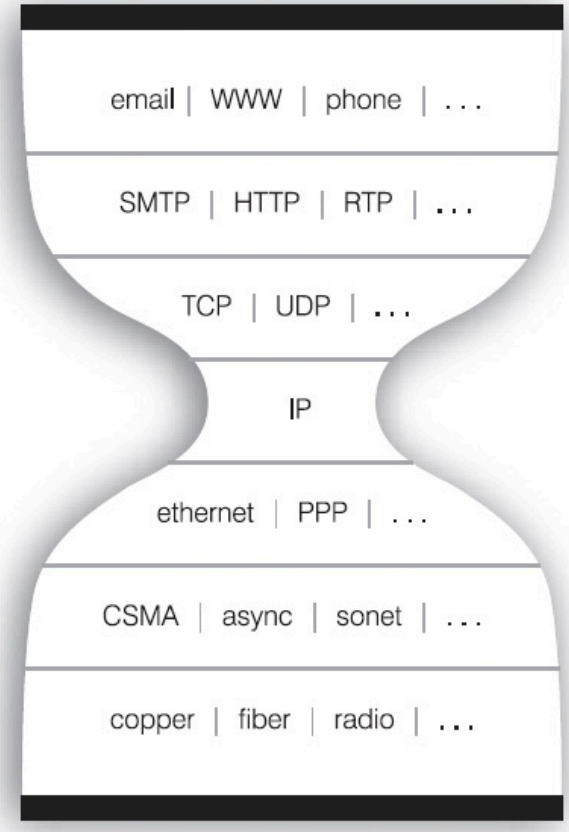


The problem is becoming even worse

- **More and more devices connect to the Internet**
  - PCs
  - Cell phones
  - Internet radios
  - TVs
  - Home appliances
  - Future: sensors, cars...

- With NAT, **every NAT** router needs an IPv4 address

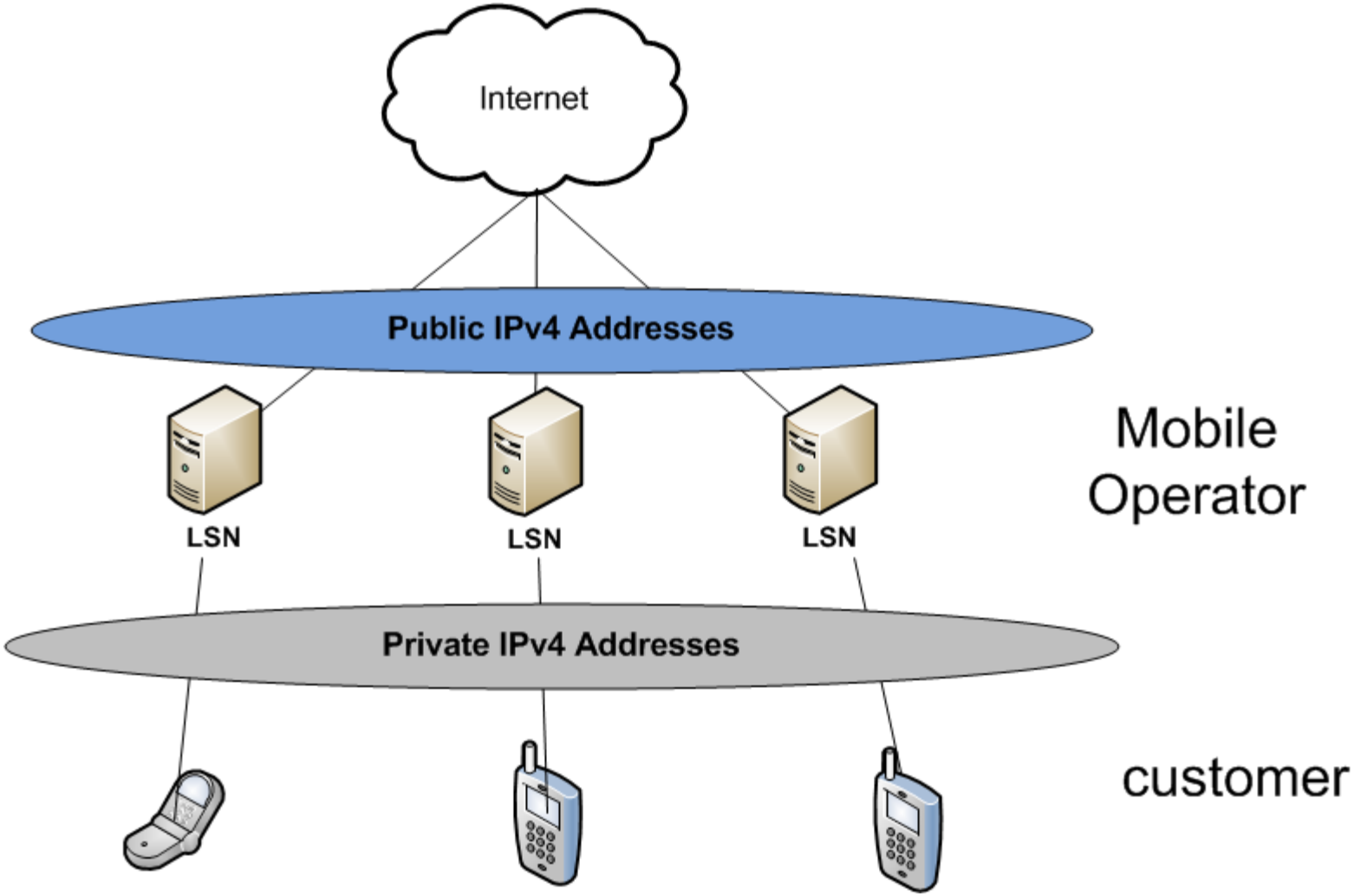
- → **ISPs** run out of global IPv4 addresses



Large Scale NAT (LSN)

- Facts
  - ISPs run out of global IPv4 addresses
  - Many hosts (in customer‘s network) are IPv4 only
  - Not all content in the web is (and will be) accessible via IPv6
    - infact: < 5% of the Top 1M Websites (12/2012)
- Challenges for ISPs
  - access provisioning for new customers
  - allow customers to use their IPv4 only devices
  - provide access to IPv4 content
- Approach: move public IPv4 addresses from customer to provider
  - Large Scale NAT (LSN) / Carrier Grade NAT (CGN) at provider for translating addresses

Large Scale NAT already common today

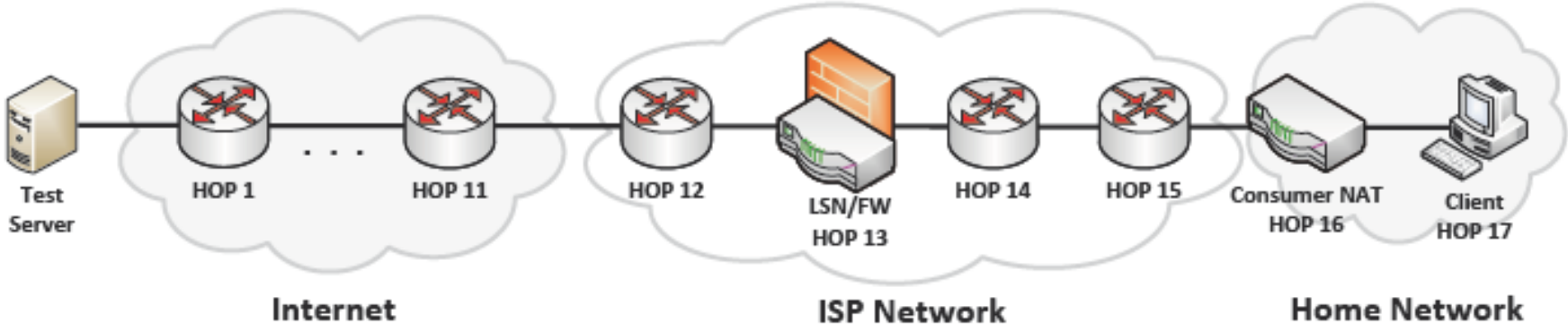


Example – Vodafone LTE network

- Algorithm to detect network topology
  - Master thesis Florian Wohlfart 2012

- LTE network for remote areas are usually double NATed

- Test Server @TUM, Client in Vodafone LTE network (17 hops)



Topology detection algorithm

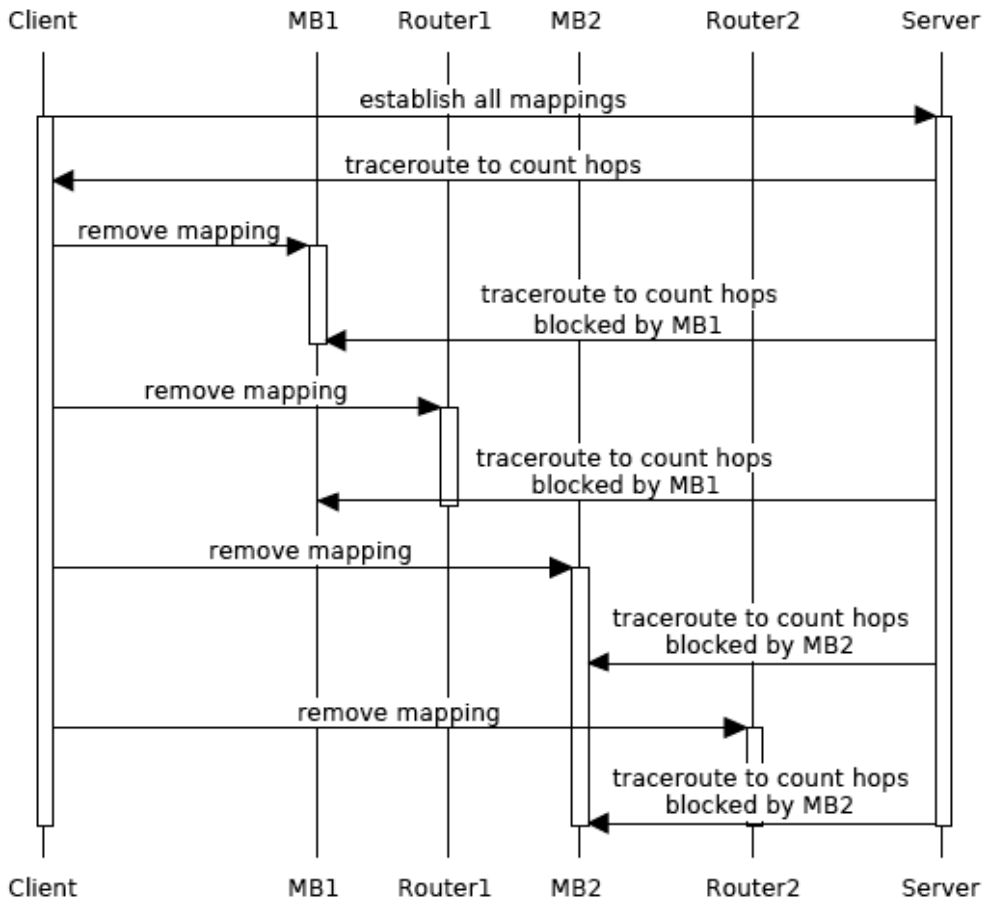
1) Establish state in all hops by sending outgoing packets

2) Count number of hops towards server

3) Remove mappings hop by hop

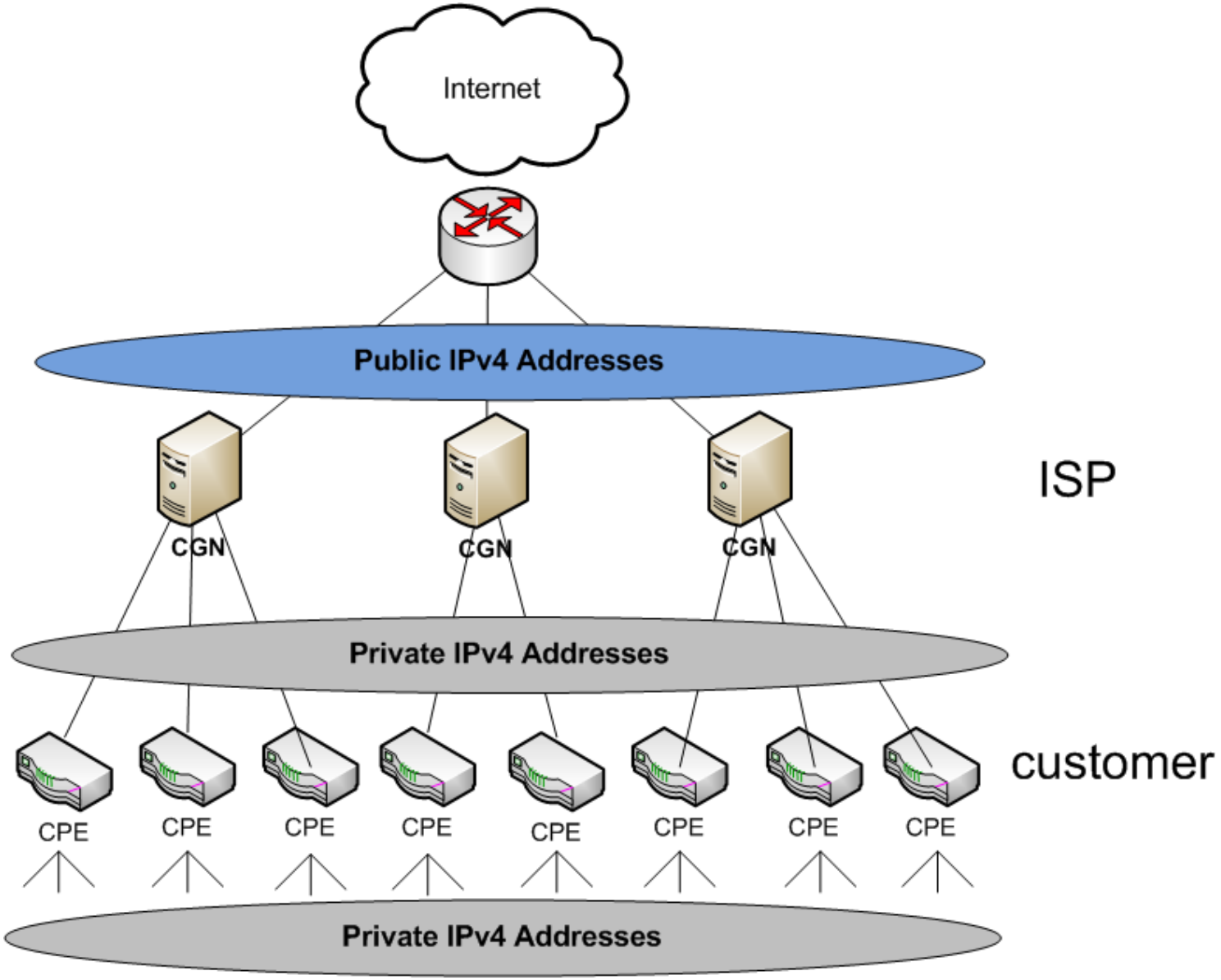
3) count number of hops towards server

if stateful hop (e.g. NAT) less hops detected



1 **foreach** *hop from 1 to n do* **do**
2 | establish mapping: send UDP packet from client to server;
3 | server waits for UDP Timeout and sends keep-alive packets with $TTL = \#Hops - n$;
4 | traceroute from server to client;
5 **end foreach**

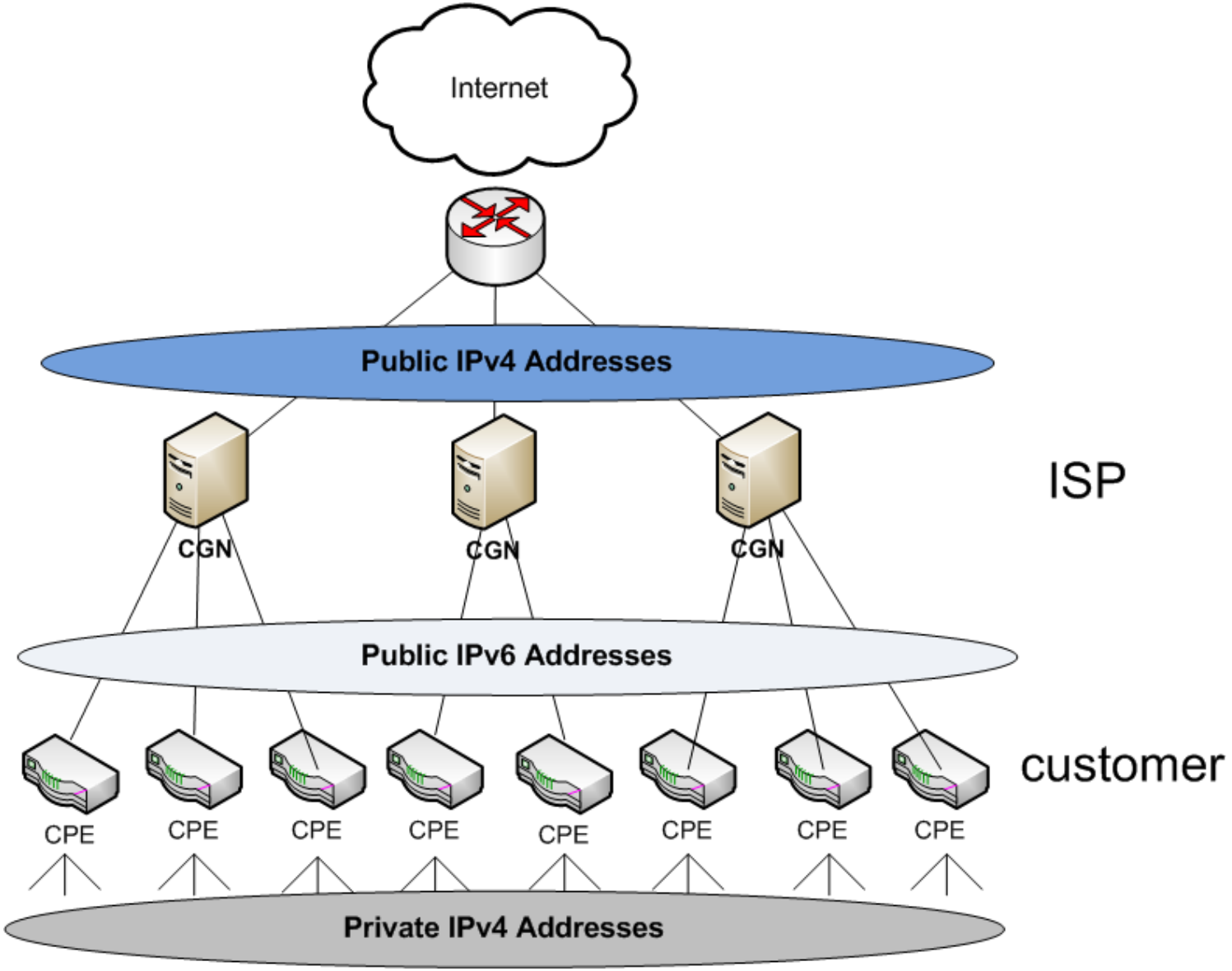
NAT 444


Internet
Public IPv4 Addresses
CGN
CGN
CGN
ISP
Private IPv4 Addresses
CPE
CPE
CPE
CPE
CPE
CPE
CPE
CPE
customer
Private IPv4 Addresses

NAT 444

- Easiest way to support new customers
  - immediately available
  - no changes at CPEs (Customer Premises Equipment)
- Problems:
  - Address overlap → same private IP address on both sides
  - Firewalls on CPE may block incoming packets with a private source address
- Solutions
  - declare a range of public IP addresses as „ISP shared“ and reuse it as addresses between CGN and CPE
  - NAT 464: IPv6 between CPE and CGN
    - Problem: CPEs must implement NAT64

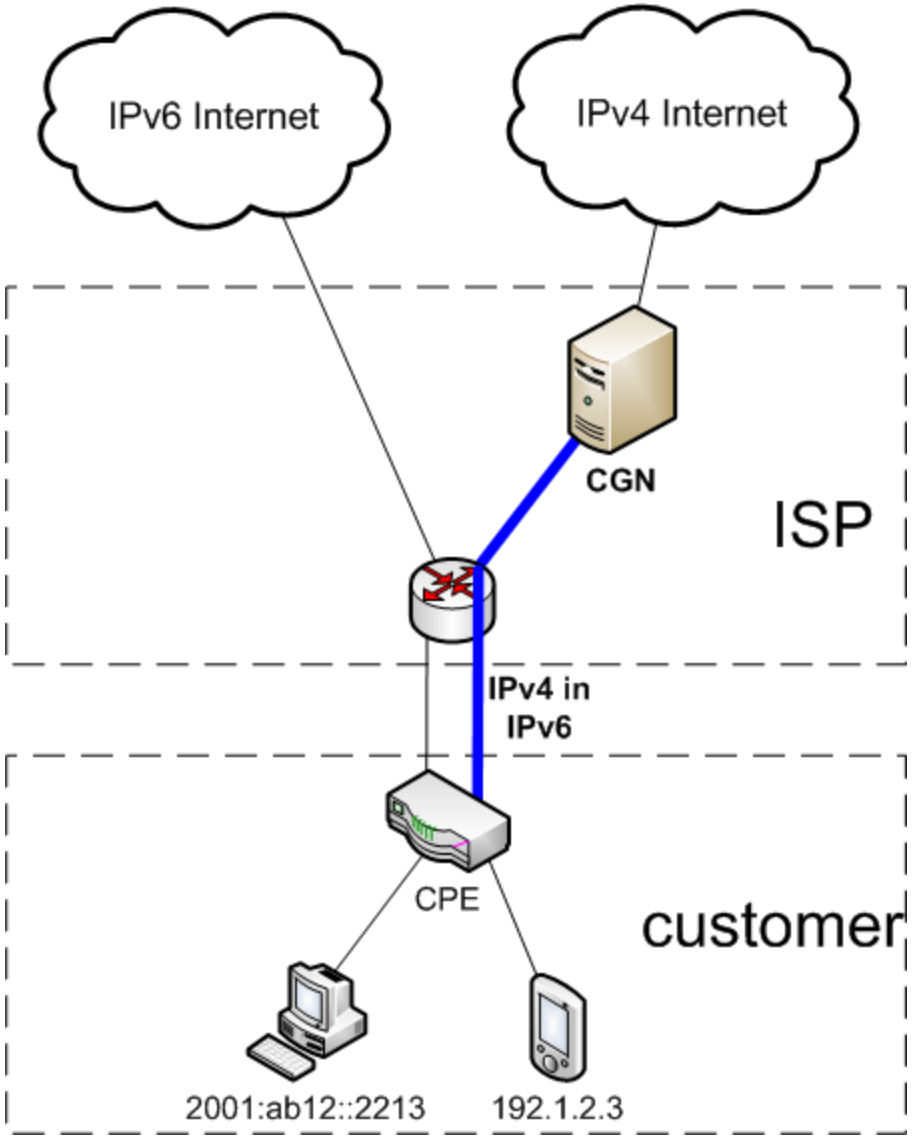
NAT 464


Internet
Public IPv4 Addresses
CGN
CGN
CGN
ISP
Public IPv6 Addresses
CPE
CPE
CPE
CPE
CPE
CPE
CPE
CPE
customer
Private IPv4 Addresses

Dual Stack Lite

- Mixture of NAT 444 and NAT 464
- IPv4 in IPv6 tunnel between CPE and ISP
  - No need for protocol translation
  - No cascaded NATs
- Allows to deploy IPv6 in the ISP network while still supporting IPv4 content and IPv4 customers
  - As IPv6 devices become available they can be directly connected without the need for a tunnel
- Pushed by Juniper, Cisco, Comcast and Apple
  - IETF RFC 6333

Dual Stack Lite



LSN/CGN Challenges

- As currently discussed in the IETF BEHAVE working group
- Mainly: how to manage resources
  - Ports (number of ports, allocation limit (time))
  - Addresses
  - Bandwidth
  - legal issues (logging)
- NAT behavior
  - desired: first packet reserves a bin for the customer -> less logging effort
  - IP address pooling: random vs. paired (same external IP for internal host)
- Impacts of double NAT for users
  - Blacklisting as done today (based on IP addressed) will be a problem
  - No control of ISP NATs
- Possible Approaches
  - Small static pool of ports in control of customer
  - Needs configuration/reservation/security protocols

Network Address Translation today

- Thought as a temporary solution

- Home Users
  - to share one public IP address
  - to hide the network topology and to provide some sort of security

- ISPs
  - for connecting more and more customers
  - for the planned transition to IPv6

- Mobile operators
  - to provide connectivity to a large number of customers
  - „security“

- Enterprises
  - to hide their topology
  - to be address independent

NAT Conclusion

- NAT helps against the shortage of IPv4 addresses
- NAT works as long as the server part is in the public internet
- P2P communication across NAT is difficult
- 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
- New challenges with the transition to IPv6 and LSN/CGN
  - Topology becomes important

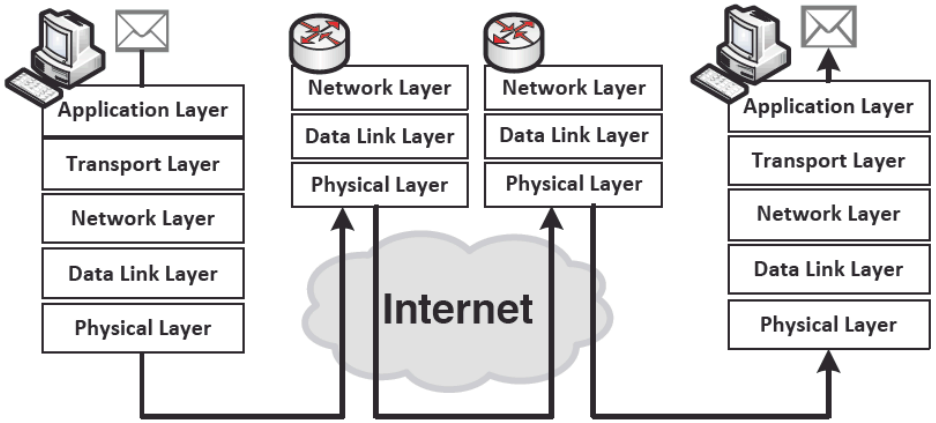
Middleboxes

Internet Architecture

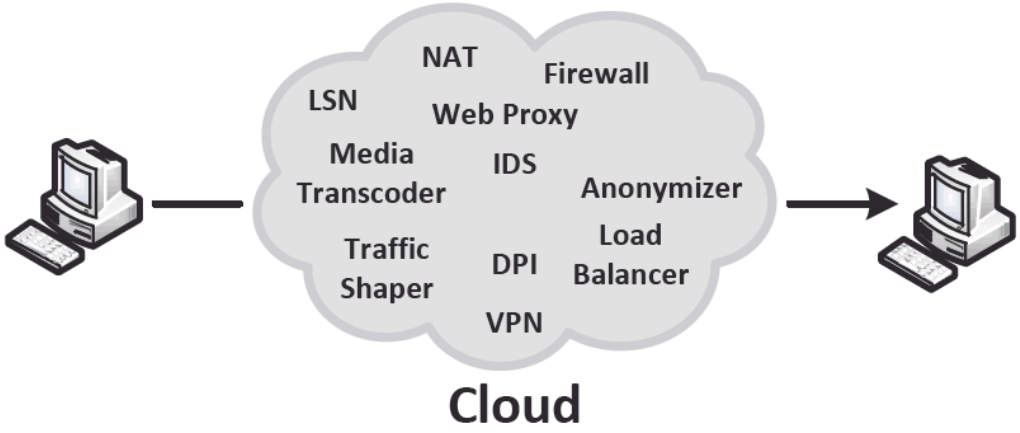
- Initial end-to-end principle
  - Saltzer, Reed and Clark (1984)

"certain functionality can only be implemented correctly with the knowledge and help of the application standing at the end points of the communication system."

  - Hosts in the Internet only for routing and forwarding
  - No state otherwise



- Today
  - Intermediate hosts offer additional functionality



Some Reasons for today's architecture

- Independent Autonomous Systems
  - ISPs ran as businesses
  - Interconnection driven by contracts rather than performance
  - Discussion about network neutrality

- Lack of Security
  - Not part of the initial (end-to-end) architecture
  - Security should be implemented on end-hosts only
  - Often solved by introducing additional functionality to the network (Firewalls, Intrusion Detection Systems…)

- Protection of Innovation
  - Initially: changes only in the end-hosts
  - Today: changes in the end-hosts hard due to heterogeneity of devices/operating systems

    → additional functionality as a black box in the network

Recent Study by UCL (2011)

- Marc Handley et al. (University College London)
  - "Flow processing and the rise of the middle"
  - "Is it still possible to extend TCP?" (SIGCOMM 2011)

- Ran tests to measure what happens to TCP in the Internet
  - e.g. Are new TCP options permitted (options field)?
  - Are sequence numbers modified?

- 142 access networks in 24 countries
  - 25% of paths interfered with TCP in some way beyond basic firewalling.
  - 20% remove new TCP options on port 80 (only 4% on port 34343)
  - 18% rewrite sequence numbers (initial sequence numbers differ)

  → Many black boxes in the network, especially for HTTP

Netalyzr (ICSI, Berkeley)

- Network Measurement and Debugging Service
  - http://netalyzr.icsi.berkeley.edu/
- Web-based (JAVA Applet) testing
  - Port filtering
  - HTTP caches and proxies
  - DNS manipulation
  - Network Buffers
  - Fragmentation and Buffers
- Selected Results (130k measurements, IMC Paper 2010)
  - 90% of all sessions behind NAT, 80%: 192/168 range
  - SMTP blocked for 25%, FTP for 20%
  - 8.4% implement HTTP Proxy

Middleboxes

- The phrase "middlebox" was coined by
  Lixia Zhang (UCLA)

- RFC 3234 defines middleboxes as:
  “intermediary devices performing functions other than the normal, standard functions of an IP router on the datagram path between a source host and destination host”

- Middleboxes are never the end-system of an application session and may
  - drop
  - insert
  - transform
  - and modify packets

Middleboxes often address practical challenges

- "Plenty of box vendors will sell you a solution… Whatever you think your problem is." (Mark Handley, UCL)
- IP address depletion
  - Allowing multiple hosts to share a single address
- Host mobility
  - Relaying traffic to a host in motion
- Security concerns
  - Discarding suspicious or unwanted packets
  - Detecting suspicious traffic
- Performance concerns
  - Controlling how link bandwidth is allocated
  - Storing popular content near the clients

RFC 3234 - Middleboxes: Taxonomy and Issues

- Gives an overview about current middleboxes

1) Layer of Operation (ISO/OSI)

2) Transparency
   - Part of the protocol (not transparent) or transparent

3) Purpose
   - Functional (part of the application) vs. Optimizing (addition)

4) Operation
   - Routing (plain forwarding) or Processing of packets

5) Stateful or stateless

Middlebox Functionality according to RFC 3234

	Layer	Transparency	Purpose	Operation	State
Reason for Introduction: Address depletion					
NAT	3+4	Transparent	Functional	Processing	Stateful
Other examples: NAT44 with ALG					
Reason for Introduction: Security					
Firewall	3+4	Transparent	Functional	Routing	Stateful
Other examples: Deep Packet Inspection, Anonymizer, Tunnel Endpoint					
Reason for Introduction: Performance					
Web-Cache	7	both	both	Processing	Stateless
Other examples: Proxy					

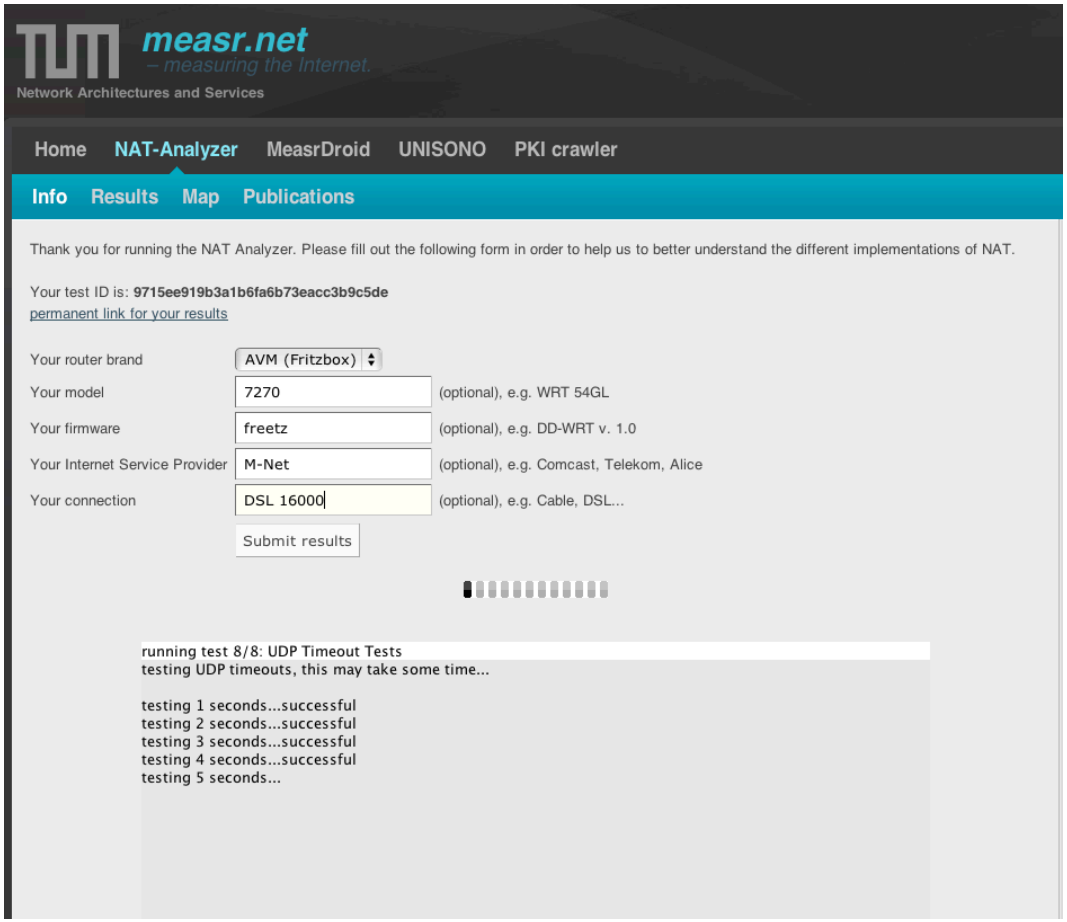
Middlebox behavior not standardized

- Just like NAT (as one middlebox example)
- Many possibilities to implement practically the same functionality
  - e.g. address translation
  - But many ways of allocating new mappings
- Idea: If exact behavior is understood and if can be expressed, coping with middleboxes becomes easier
- Model for formalizing and describing middlebox behavior
  - Measure behavior and create model
  - Model holds properties of the middlebox
  - Traversal solutions based on model

NAT Analyzer – Measuring NAT and MB Behavior

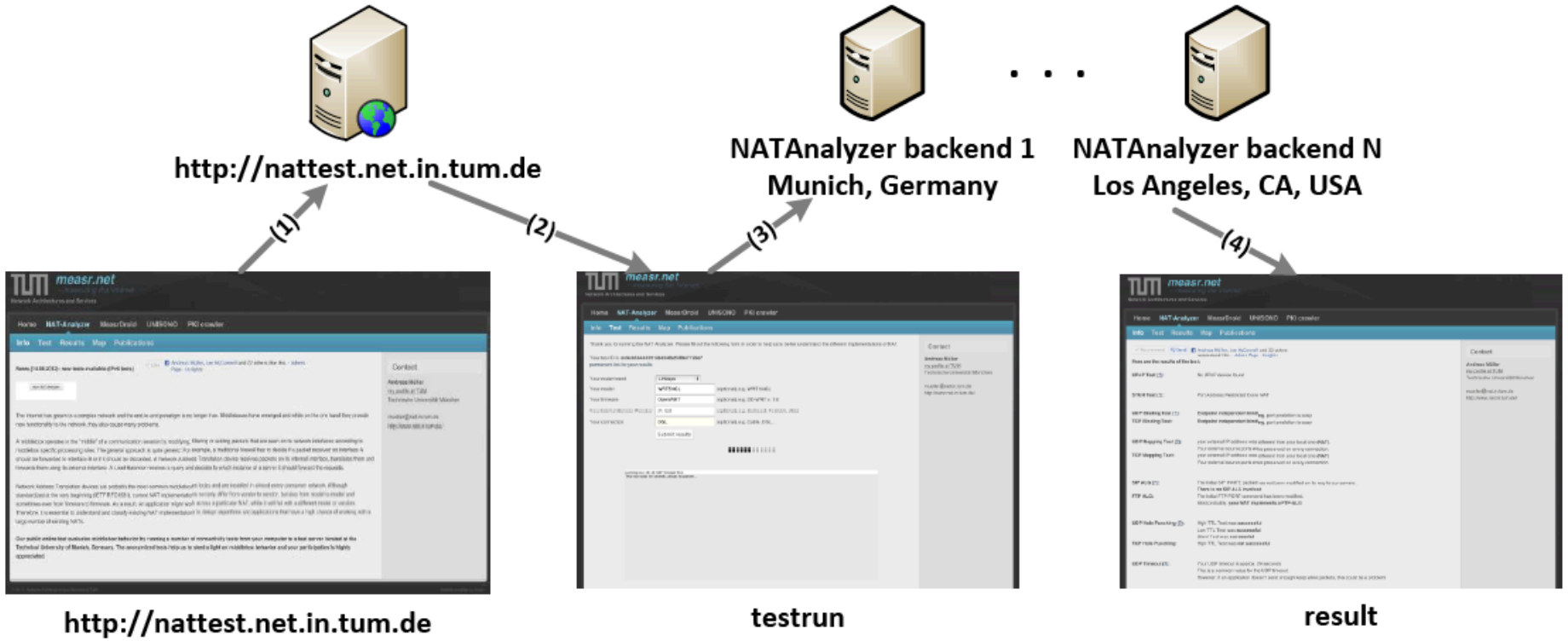
- Public field test with more than 4000 NATs
  - understand existing traversal techniques and NAT behavior

    (http://nattest.net.in.tum.de)



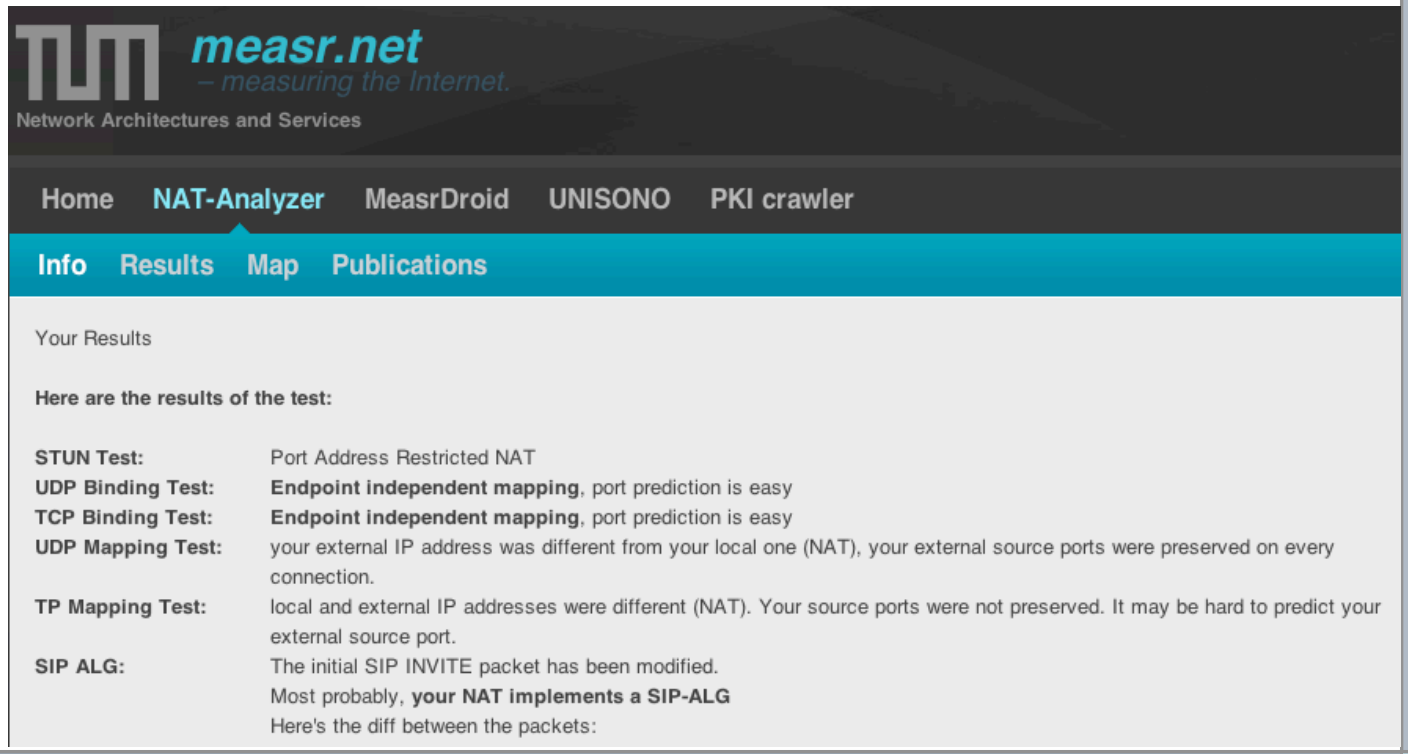
Field Test

- Idea: ask volunteers to run the algorithms in their network



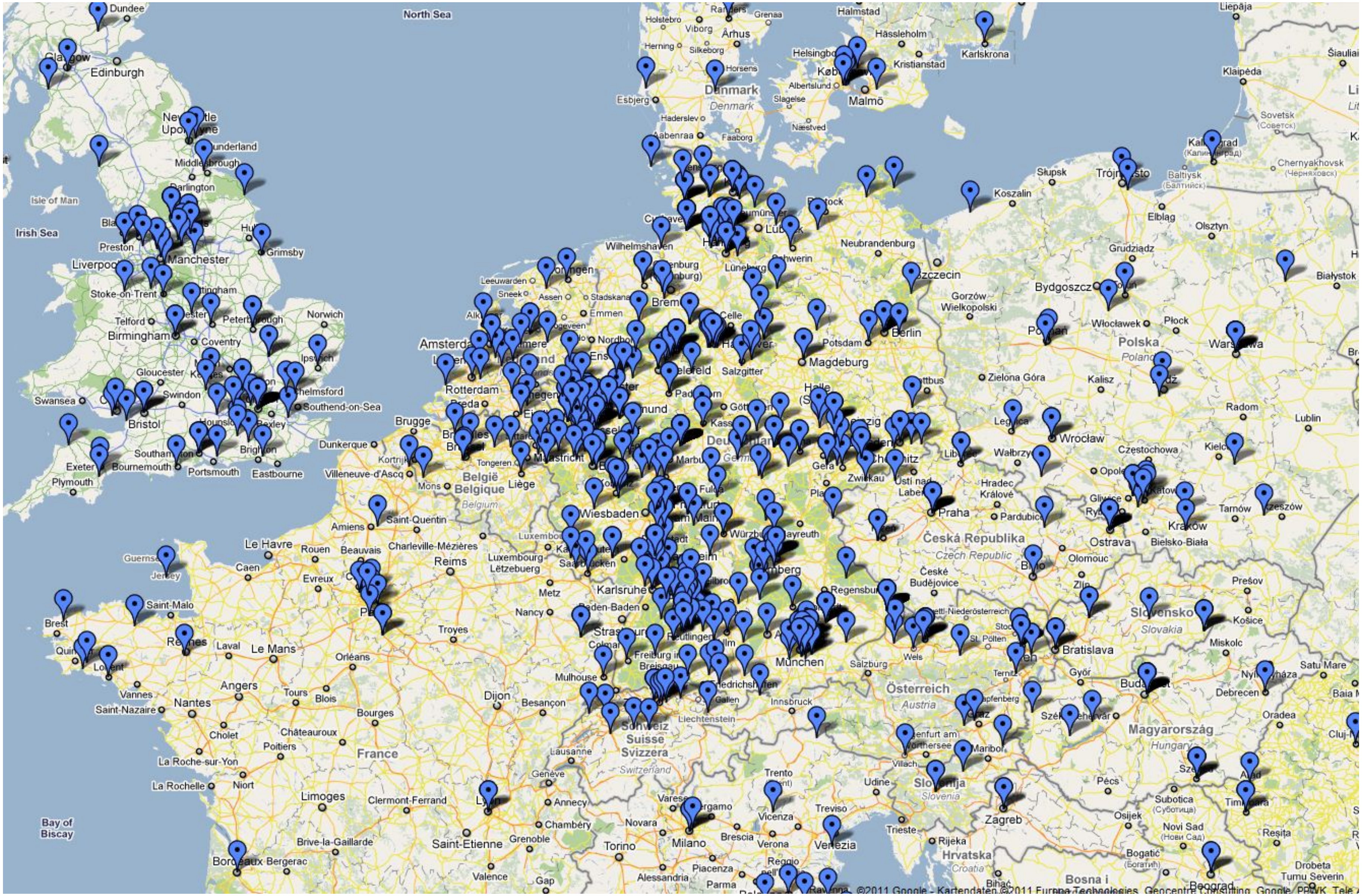
NAT Analyzer

- Connectivity tests with a server at TUM reveals
  - NAT Type
  - Mapping strategy
  - Binding Strategy
  - Hole Punching behavior using different techniques
  - Timeouts
  - ALGs
- Example Result



NAT Analyzer– Results (World)

NAT Analyzer– Results (Central Europe)



Success Rates for existing traversal solutions

- UPnP 31 %
- Hole Punching
  - UDP 80%
  - TCP low TTL 62%
  - TCP high TTL 52%
  - TCP combined 67%
- Relay 100%
- Probabilities for a direct connection
  - UDP Traversal: 85 %
  - TCP Traversal: 82 %
  - TCP inclusive tunneling: 95 %

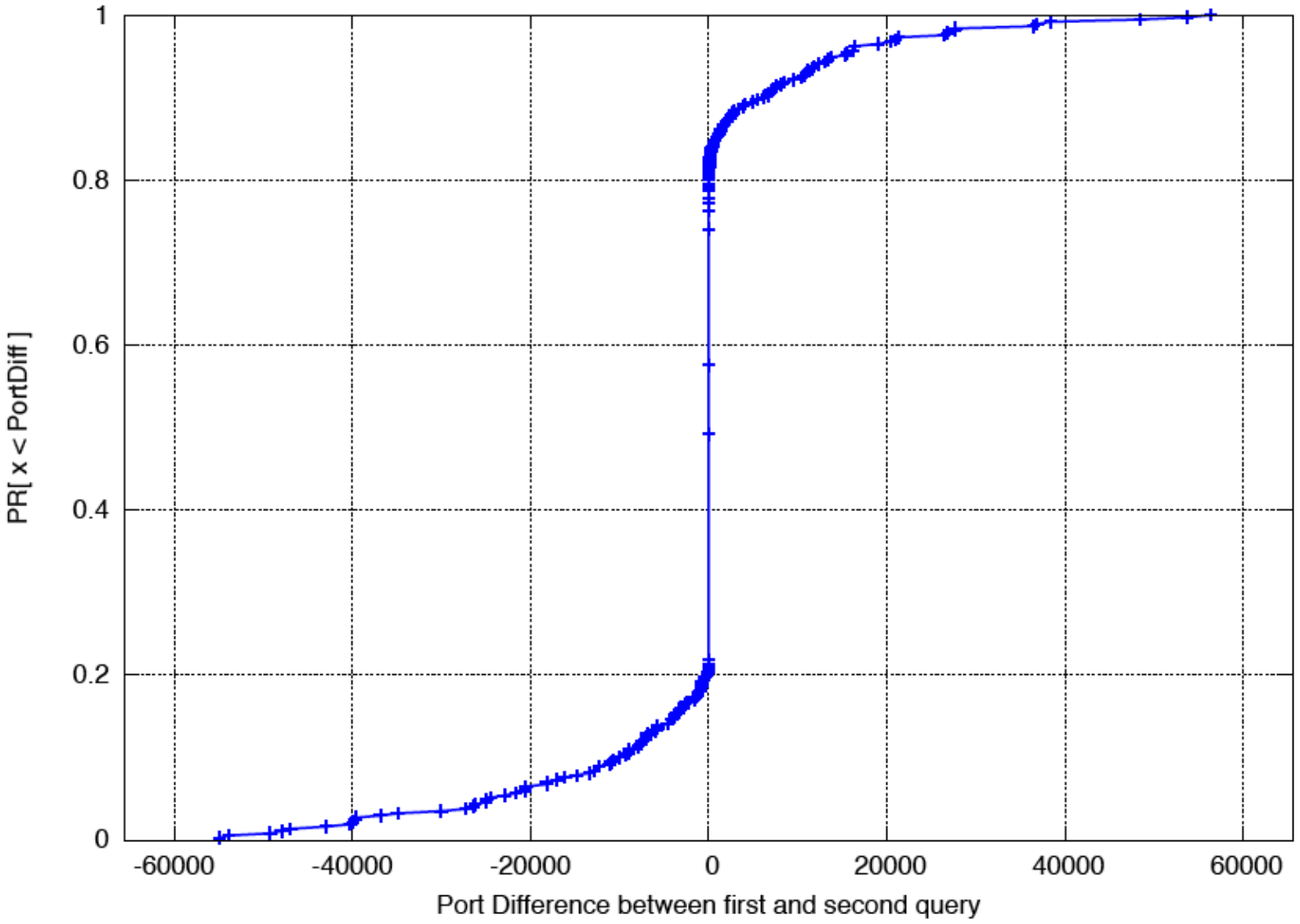
NAT Binding Results - Recap

- **NAT Binding**:
  - two consecutive connections from the same source to different destinations create two external mappings X and Y
    - Endpoint independent: X == Y
    - Connection dependent: X != Y

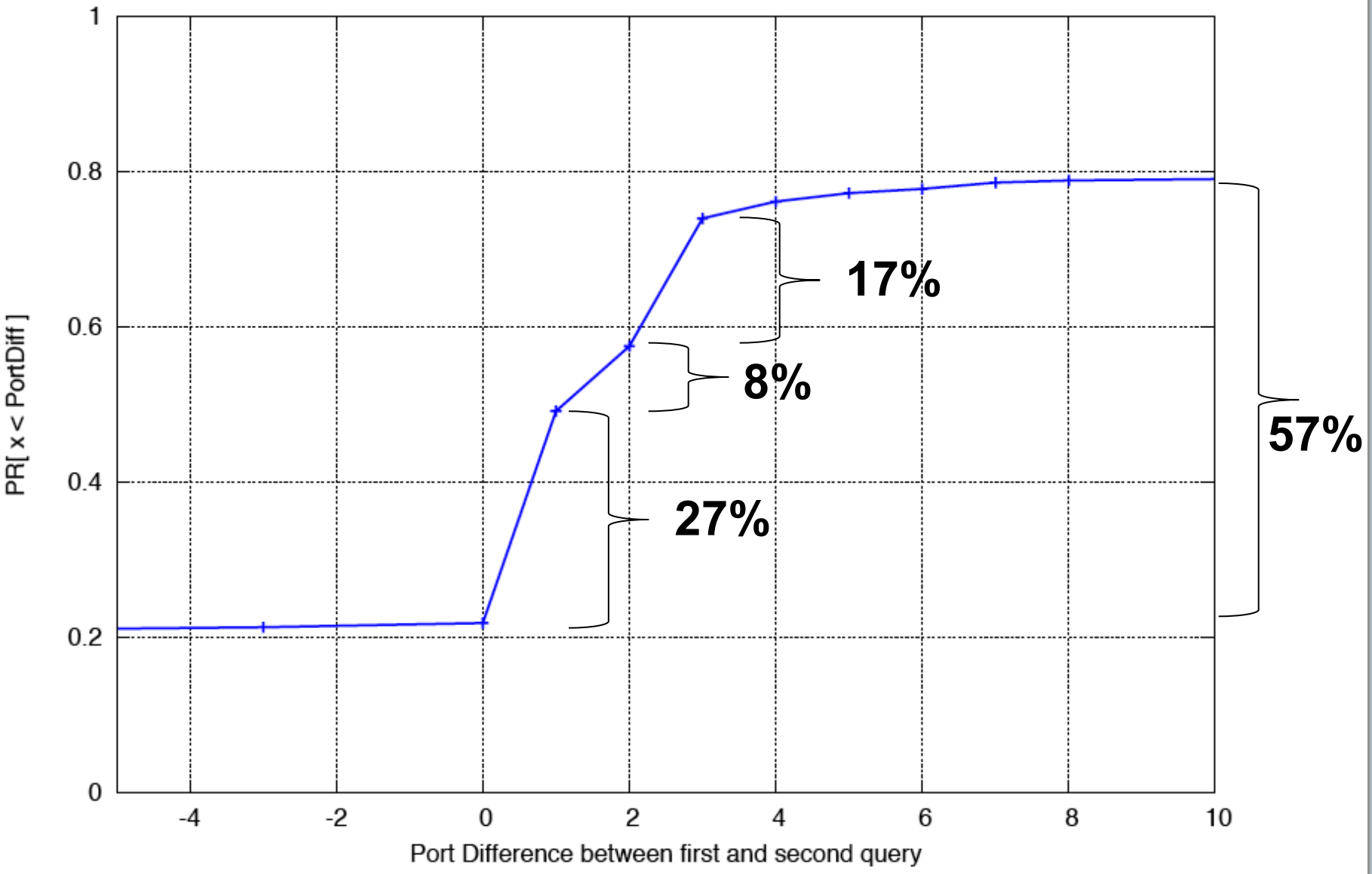
- Problem:
  - Connection dependent binding hard to predict
    - STUN not possible

- Goal: Improve Port Prediction for connection dependent binding
  - Send two packets from same source port to STUN-like server, look at the two different external ports and calculate difference
    - e.g. extPort 1 = 20000, extPort2=20001 → difference = 1

Connection dependent binding



Connection dependent binding (2)

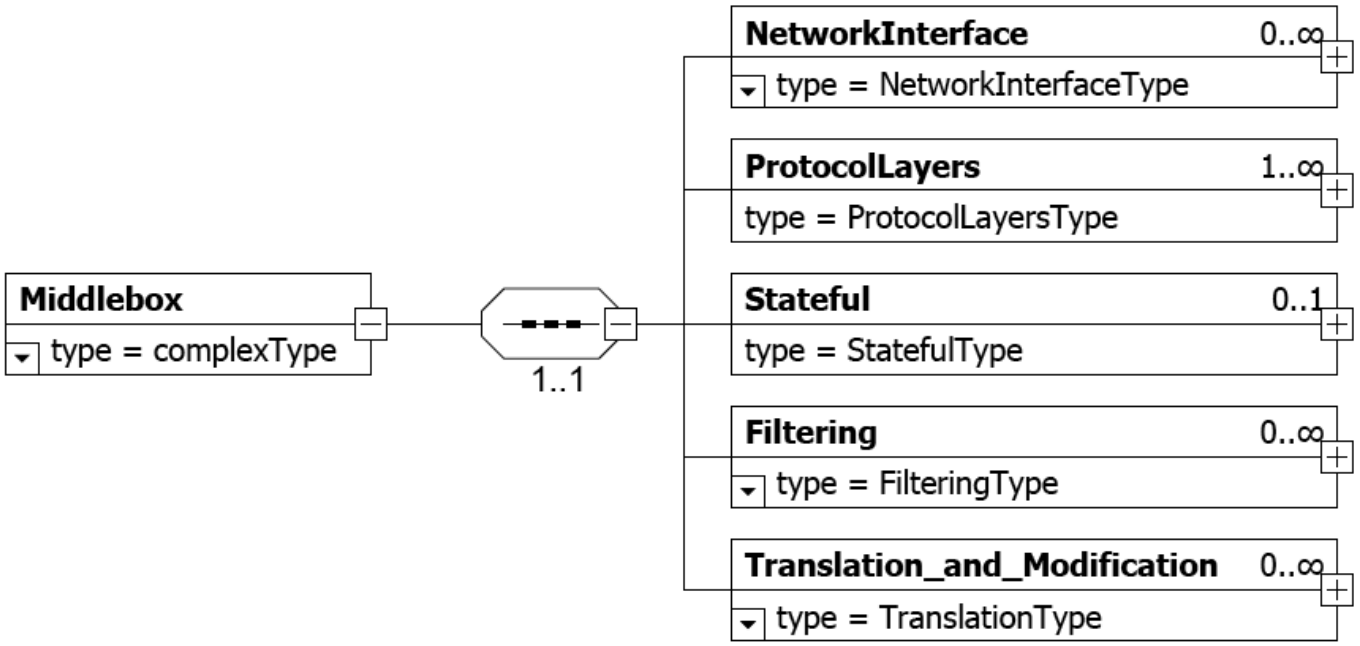


Binding Analysis Results

- State of the Art
  - Connection dependent binding (Symmetric NAT) is hard to traverse
  - Cannot query external port using STUN and reuse it for an actual connection
- Field Test results
  - 22% for UDP and 25% for TCP implement connection dependent binding
  - In 57% for UDP (44% for TCP) port prediction is possible by analyzing binding patterns
- Question: How to express this information?

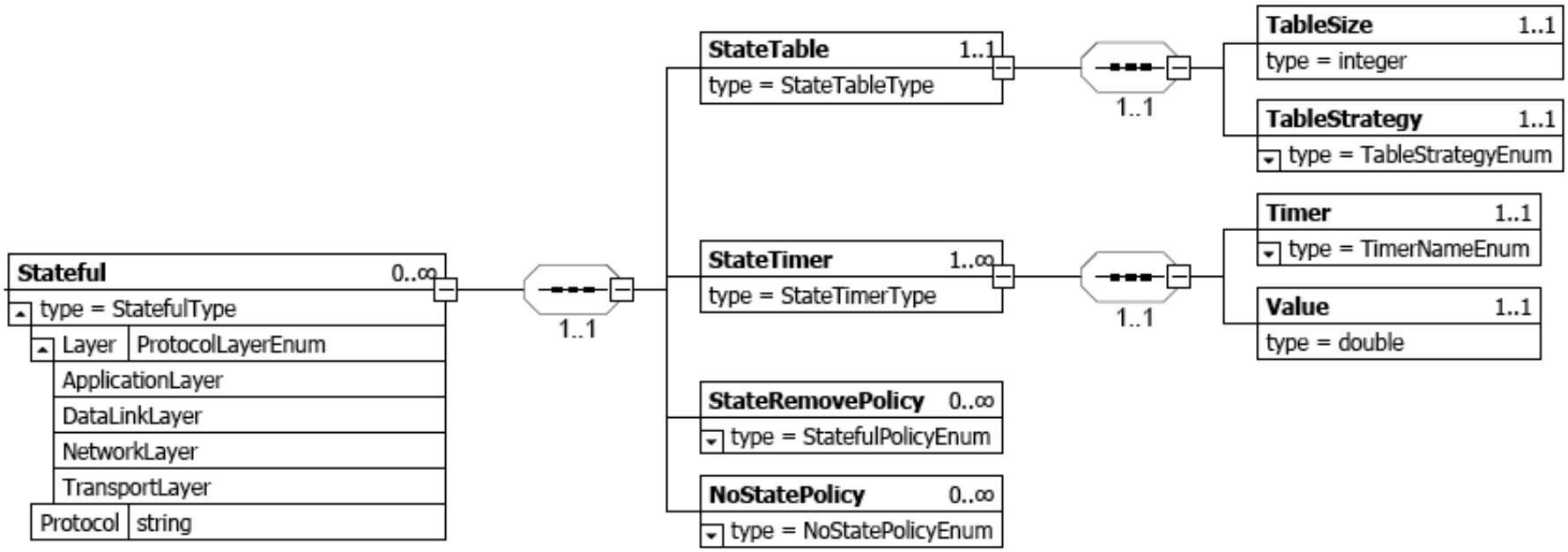
Approach: Information Model

- Describes characteristics and behavior properties
  - measured in our field test and from the state of the art
- XML Schema
- Model instance describes middlebox behavior



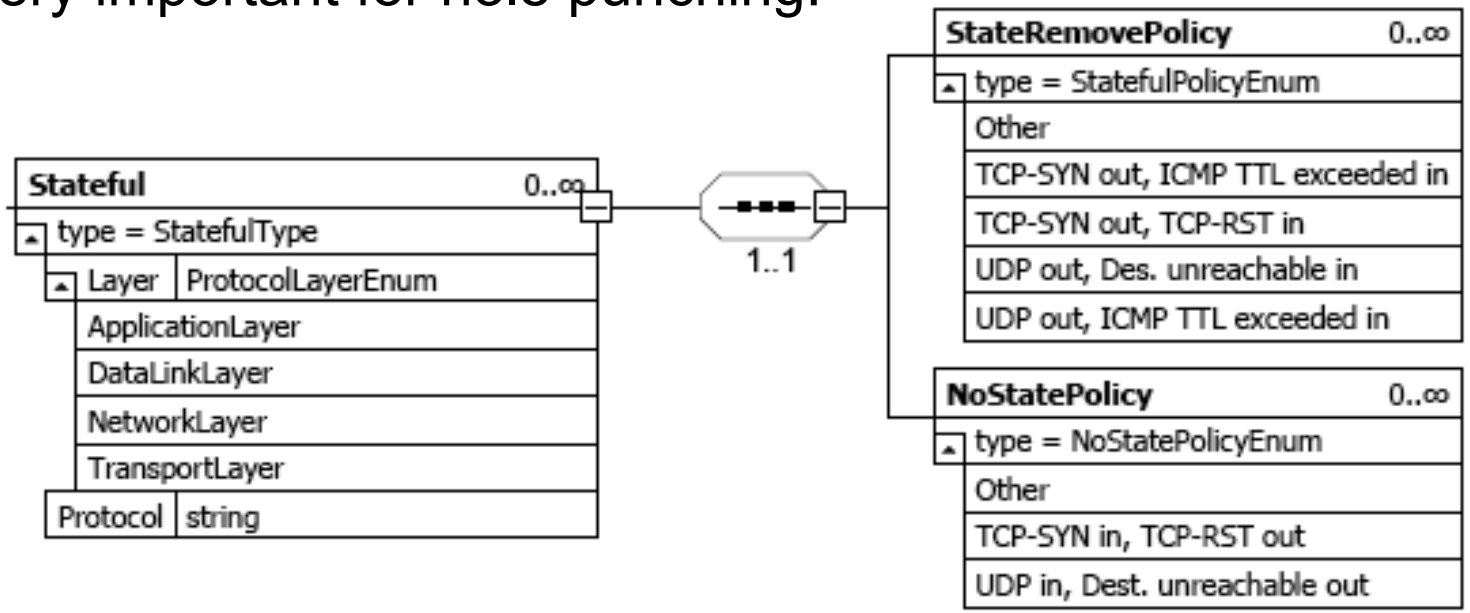
Stateful Element

- Stateful middleboxes maintain state tables
  - e.g. NAT mapping table
- Size and strategy (what happens if table is full) depends on resources and implementation *(Stateful:StateTable)*
- State entries expire after a certain amount of time (*StateTimer*)



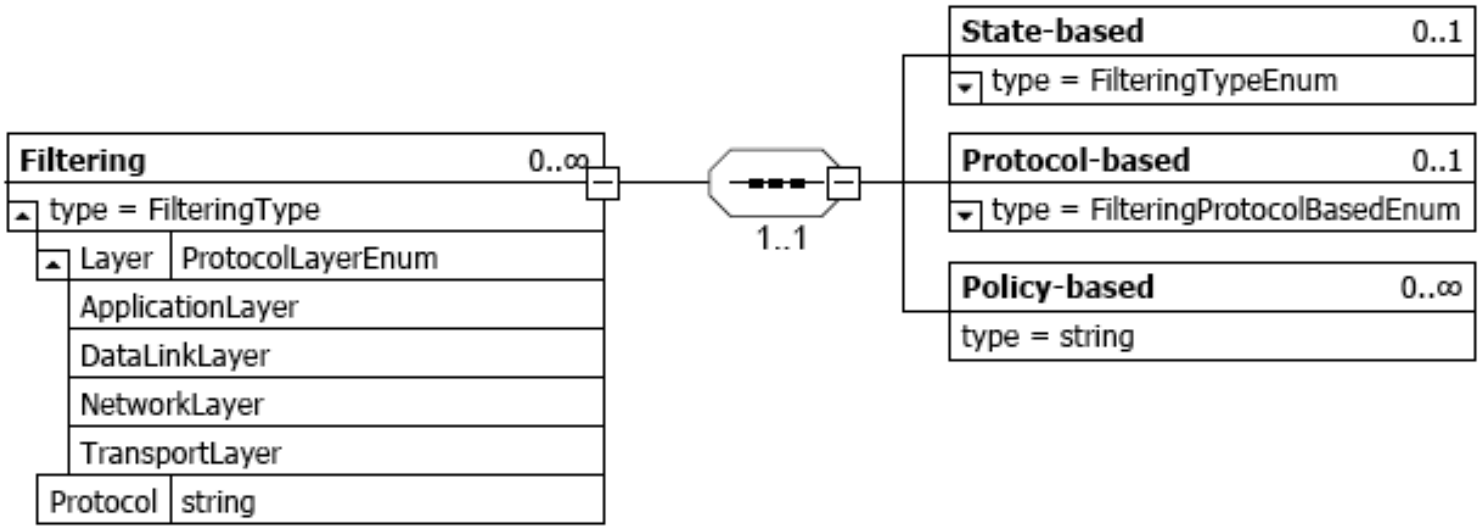
Stateful Element (2)

- State is removed on certain events / packet sequences (*StateRemovePolicy*)
- MB may send packets as a response to packets sent to a non existing mapping – e.g. TCP RST (*NoStatePolicy*)
- Very important for hole punching!



Filtering Element

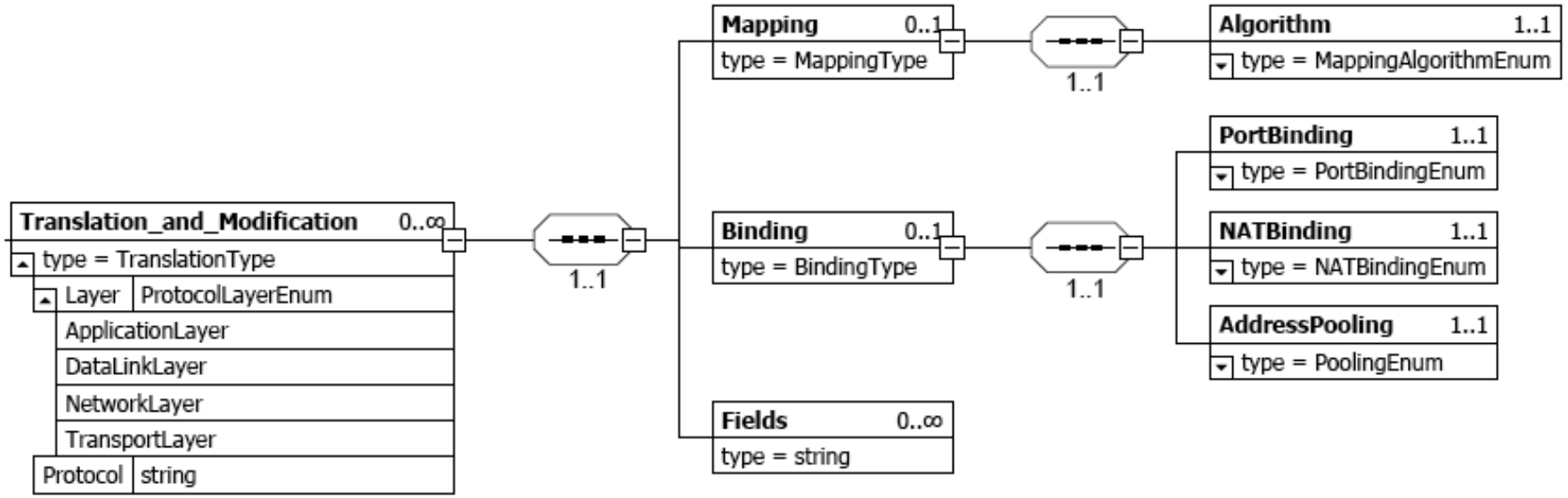
- Based on state (State-based)
  - Independent, Address Restricted, Port Restricted
- Unusual Sequences, e.g. SYN out, SYN in (Protocol-based)
- Based on a (user-defined) policy *(Policy-based)*



Translation and Modification

- For NAT: how to allocate external mappings
  - See last week's slides for binding etc.
  - Address Pooling: see Large Scale NAT (multiple ext. IP addr)

- Mapping: correlation of internal and external mappings
  - Mainly: is port prediction possible



Middlebox Instance

- XML Schema

  →Description results in XML file

```
<?xml version = "1.0" encoding = "utf-8"?>
<n:Middlebox xmlns:n="http://example.org/MiddleboxSchema" xmlns:xsi="http://www.w3.org
    /2001/XMLSchema-instance" xsi:schemaLocation="http://example.org/MiddleboxSchema"
    MiddleboxType="referenceNAT">

  <NetworkInterface NetworkInterfaceName="lan">
    <ipv4>
      <IPAddress>192.168.1.1</IPAddress>
      ...
    </ipv4>
  </NetworkInterface>
  <NetworkInterface NetworkInterfaceName="wan">
  ...
  </NetworkInterface>
  ...
  <Stateful Layer="TransportLayer" Protocol="UDP">
    <StateTable>
      <TableSize>8000</TableSize>
      <TableStrategy>Block</TableStrategy>
    </StateTable>
  ...
    <StateTimer>
```

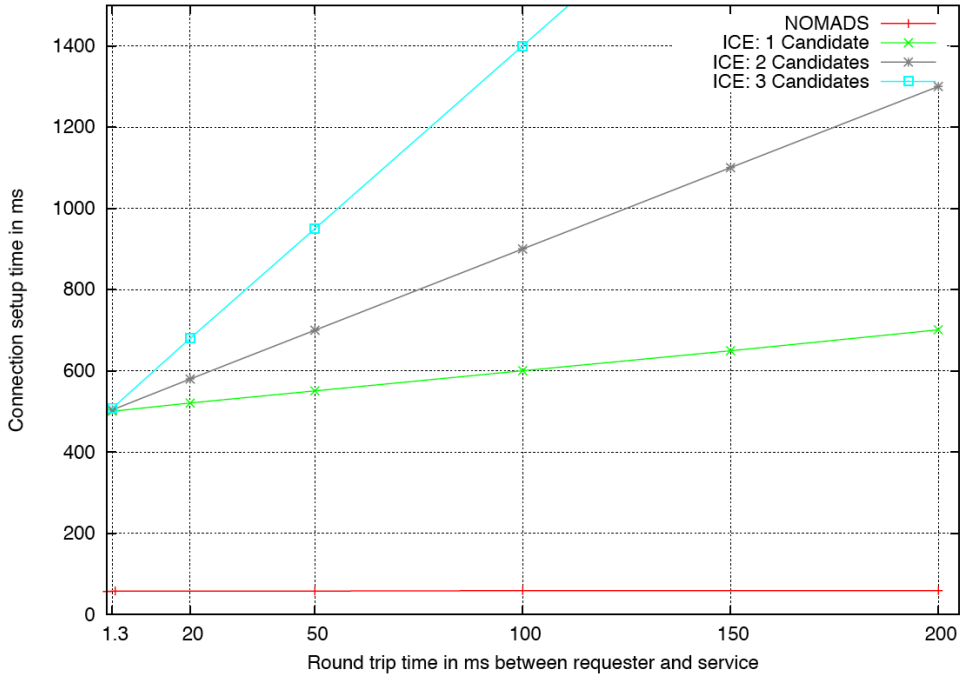
Middlebox Instance (cont)

```
<?xml version = "1.0" encoding = "utf-8"?>
<n:Middlebox xmlns:n="http://example.org/MiddleboxSchema" xmlns:xsi="http://www.w3.org
    /2001/XMLSchema-instance" xsi:schemaLocation="http://example.org/MiddleboxSchema"
    MiddleboxType="referenceNAT">

  <NetworkInterface NetworkInterfaceName="lan">
    <ipv4>
      <IPAddress>192.168.1.1</IPAddress>
      ...
    </ipv4>
  </NetworkInterface>
  <NetworkInterface NetworkInterfaceName="wan">
  ...
  </NetworkInterface>
  ...
  <Stateful Layer="TransportLayer" Protocol="UDP">
    <StateTable>
      <TableSize>8000</TableSize>
      <TableStrategy>Block</TableStrategy>
    </StateTable>
  ...
    <StateTimer>


      <Timer>UDP</Timer>
      <value>30</value>
    </StateTimer>
        ...
  </Stateful>
  <Filtering Layer="TransportLayer" Protocol="UDP">
    <State_based>Address and Port Restricted</State_based>
  </Filtering>
  <Translation_and_Modification Layer="TransportLayer" Protocol="UDP">
    <Mapping>
      <Algorithm>Increment</Algorithm>
    </Mapping>
    <Binding>
      <PortBinding>PortPreservation</PortBinding>
      <NATBinding>EndpointIndependent</NATBinding>
    </Binding>
  </Translation_and_Modification>
</n:Middlebox>
```

Implication of Middlebox Behavior for Traversal

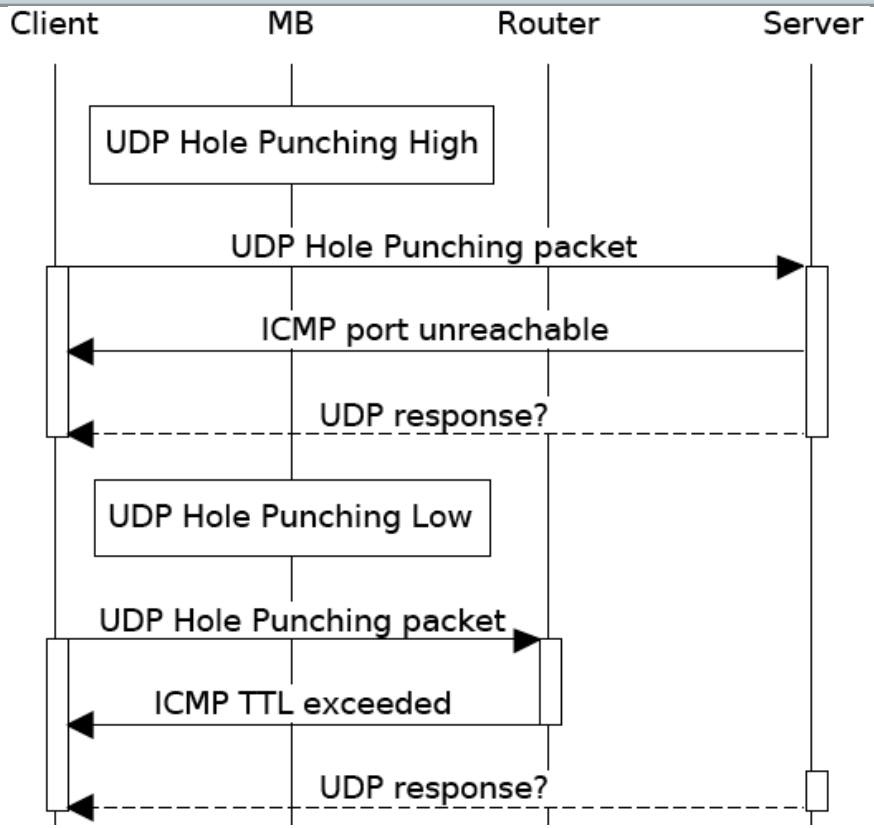
- State of the Art (e.g. ICE, Skype)
  - Trial and error
  - “brute force” pairing of endpoints

- Goal: Traversal based on requester’s and service’s model
  - Pairing based on knowledge



UDP Hole Punching Example

- initial packet creates state and gets dropped at remote host
- remote host may send ICMP dest. unreachable as a reply
- requester's NAT may drop state if such a packet is seen
- Alternative: set low TTL and provoke ICMP TTL exceeded



1) assess predictability of external mappings

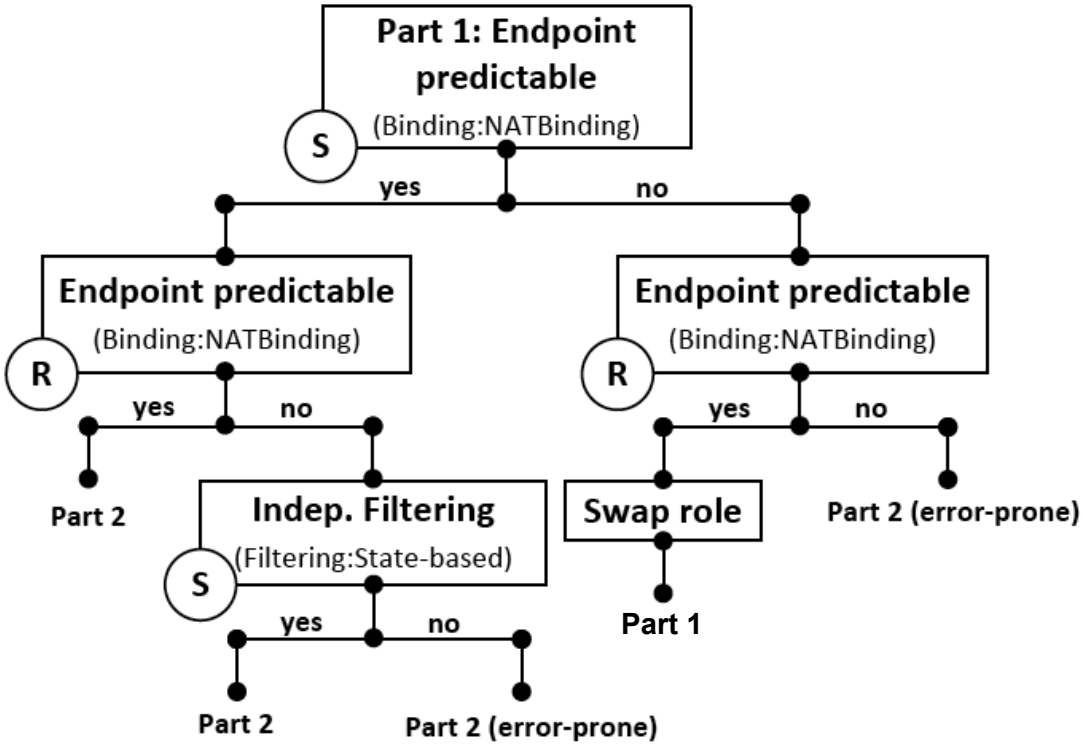
Restricted filtering at service needs more accurate prediction (requesters IP addr. + port) than independent filtering

2) prevent state from being closed by accident

Does answer to hole punching packet close mapping at service?

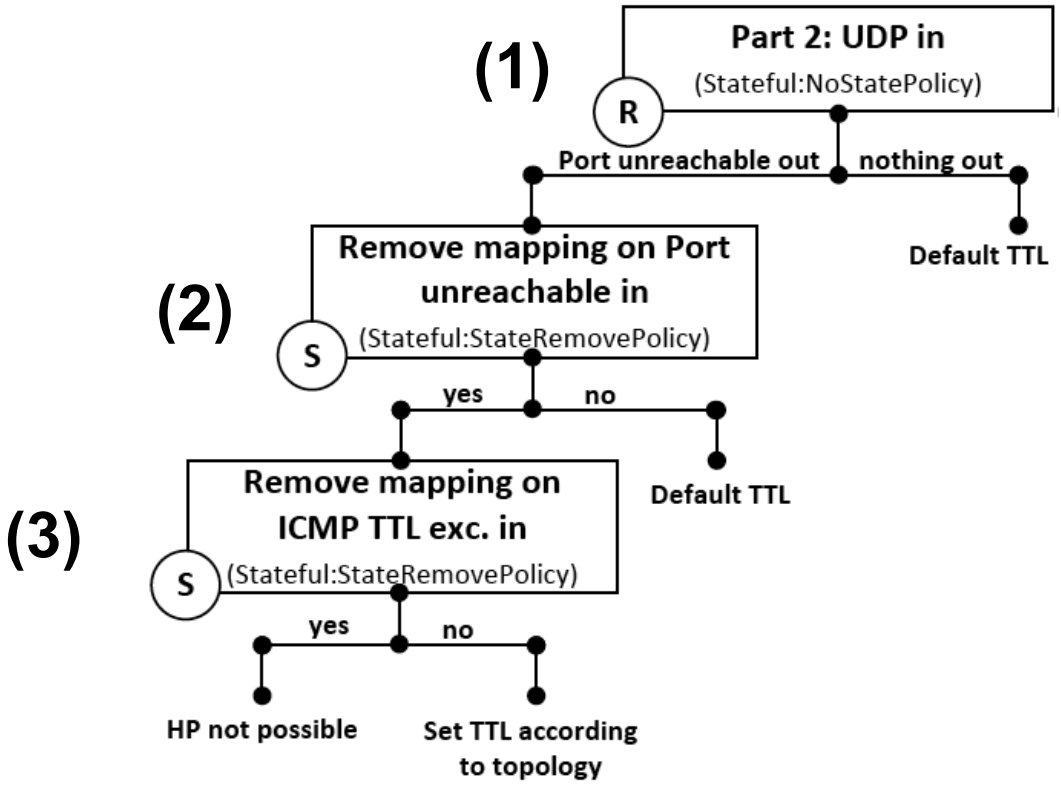
Step 1) Endpoint predictability

- Either via STUN or via Port Prediction Algorithm
- Endpoint predictable on service and requester → no problem
- If one is not able to predict endpoint
  → also consider filtering and swap role if necessary



Step 2) Prevent State from being closed

- If requester sends ICMP unreachable as a response to a received UDP packet for a non-existing state (1)
  and Service removes mapping on ICMP unreachable in (2)
  try to set a lower TTL (3)



General Concerns with Middleboxes

- New middleboxes challenge **old protocols**
  - Protocols designed without consideration of middleboxes may fail, predictably or unpredictably

- Middleboxes introduce **new failure modes**;
  rerouting of IP packets around crashed routers is no longer the only case to consider. The fate of sessions involving *crashed middleboxes* must also be considered.

- **Configuration** is no longer limited to the two ends of a session; middleboxes may also require configuration and management.

- **Diagnosis** of failures and misconfigurations is more complex.

Assessment

- Future application protocols should be designed in recognition of the likely presence of middleboxes (e.g. network address translation, packet diversion, and packet level firewalls)

- Approaches for failure handling needed
  - soft state mechanisms
  - rapid failover or restart mechanisms

- Common features available to many applications needed
  - Middlebox discovery and monitoring
  - Middlebox configuration and control
  - Routing preferences
  - Failover and restart handling
  - Security

Conclusion

- Middleboxes violate the initial end-to-end argument
- In many cases: valid reason for introduction
  - Address depletion
  - Security
  - Performance
- Cause problems with many existing protocols
- If behavior is known, algorithms can be adapted accordingly
  - Information Model (also processing model)
  - Pairing based on knowledge