Designing for and Living with NATs and Firewalls

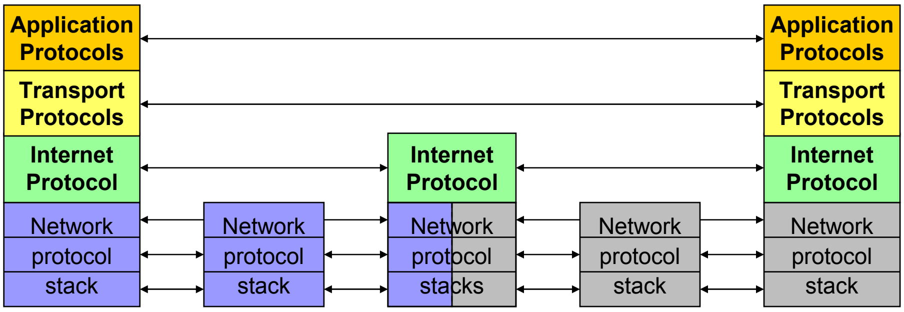
Protocol Design – S-38.3157

"The primary purpose of **firewalls** has always been to **shield buggy code** from **bad guys**."

Steve Bellovin, IETF Security AD

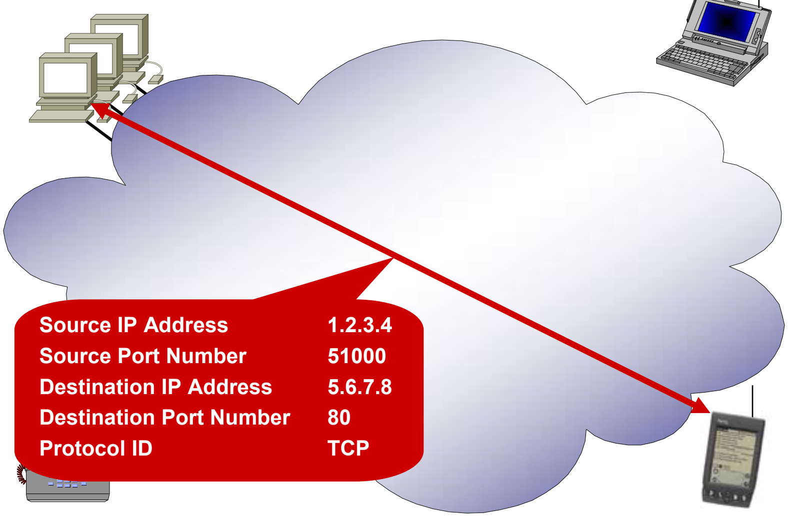
Reminder: Internet Architecture

- Transport and application protocols operate end-to-end
  - Port numbers: addressing of processes (applications)
  - Network-components and topology are invisible
  - All functions performed end-to-end



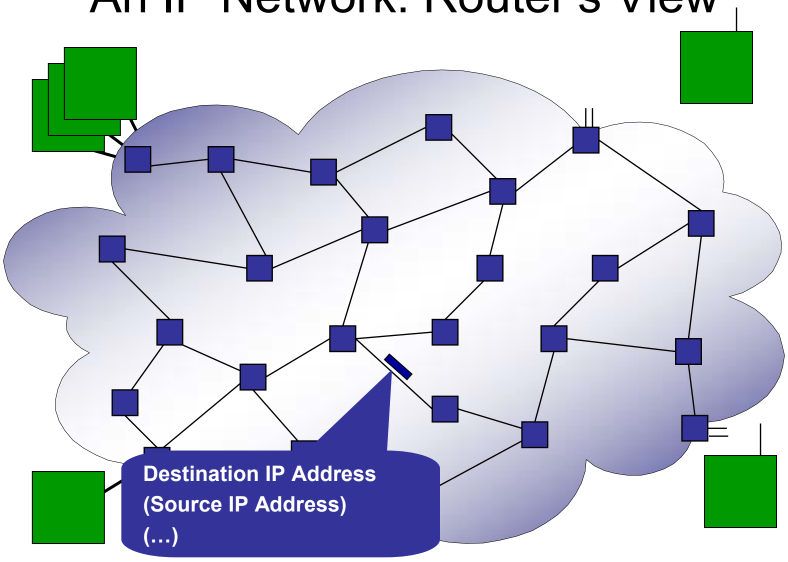
HELSINKI UNIVERSITY OF TECHNOLOGY
NETWORKING LABORATORY

An IP Network: Application’s View



HELSINKI UNIVERSITY OF TECHNOLOGY
NETWORKING LABORATORY

An IP Network: Router's View



HELSINKI UNIVERSITY OF TECHNOLOGY
NETWORKING LABORATORY

Key Concepts of the Internet Architecture

Hosts know nothing about the network.

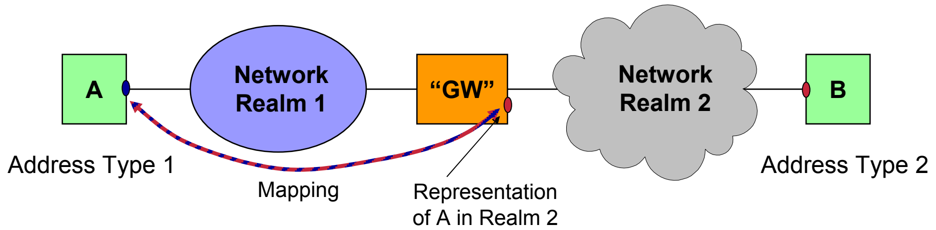
Routers know nothing about applications.

The Internet in the good ol' times…

…and today.

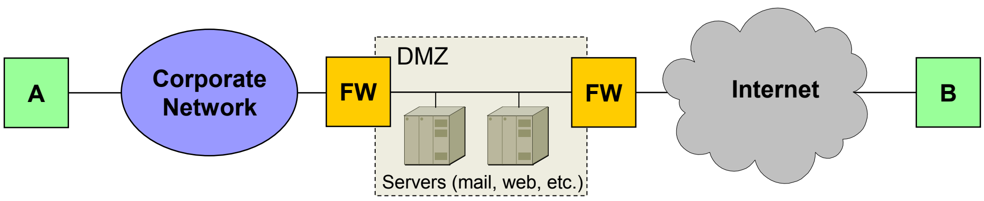
Fencing off (Sub)Networks in the Internet (1)

- Because they do not mix
- Issue 1: Technical incompatibility because of addressing
  - Historic motivation: lack of IPv4 addresses
  - Network Address (and Port) Translator (NAT, NAPT)
  - More general problem: translating between different addressing realms
  - Different example: parallel operation of IPv6 and IPv4

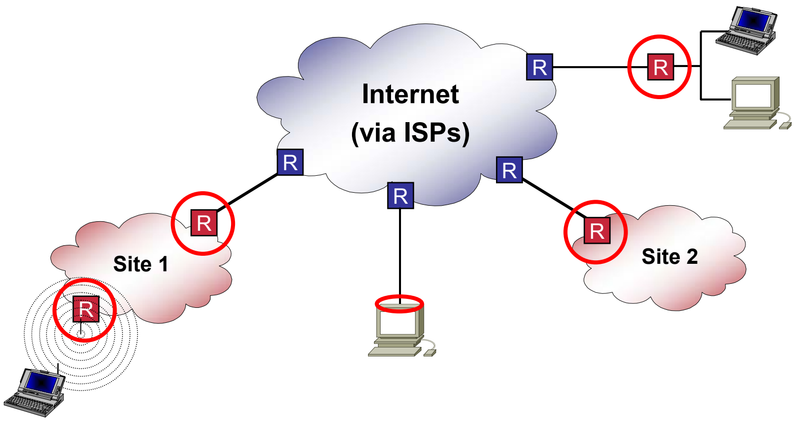


Fencing off (Sub)Networks in the Internet (2)

- Issue 2: Different levels of trustworthiness
  - Firewalls: "outside" vs. "inside" of corporate networks
  - Sometimes semi-trusted ("demilitarized") zone (DMZ)
  - Dedicated devices for an entire subnet
  - Complemented by host firewalls
    - Minimize the amount of code that needs to work properly for effective defense



A Sample Network Setup



HELSINKI UNIVERSITY OF TECHNOLOGY
NETWORKING LABORATORY

Recap: "Security Devices" for IP Networks

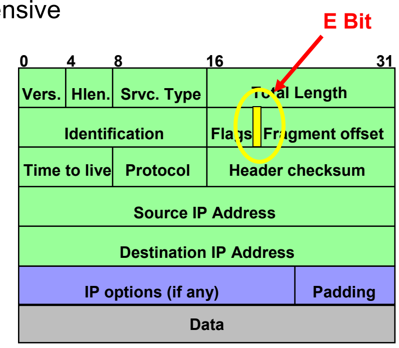
- Packet Filter
  - (dis)allow forwarding of packets to/from certain addresses
  - Protect networks from stray traffic
- Application Layer Gateway (ALG) / Proxy
  - control (and police) communications at application layer
- Firewall
  - Combination of the above
  - protect internal resources against access from the outside

---

- Network Address Translator (NAT)
  - minimize required fraction of "Internet" address space
  - hide internal IP addresses
  - perform packet filtering for unknown traffic

Classifying Traffic: The E(vil)-Bit

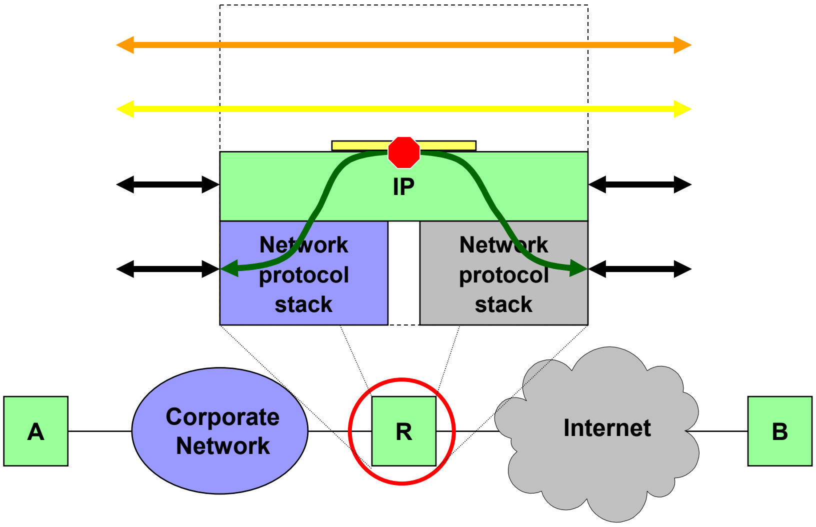
- Key question: how to identify malicious or other unwanted traffic
- Potentially intense processing required per packet
  - Source + destination IP addresses and port numbers, protocol type
  - Stateful packet inspection even more expensive
- Solution: RFC 3514 (1 April 2003)
  - "The Security Bit in the IPv4 Header"
  - Straightforward traffic identification
  - Fail-safe, easy to implement
  - E == 1: packet has evil content
  - E == 0: packet is ok
  - Firewalls simply discard evil packets
  - Extension for IPv6: "evil strength"



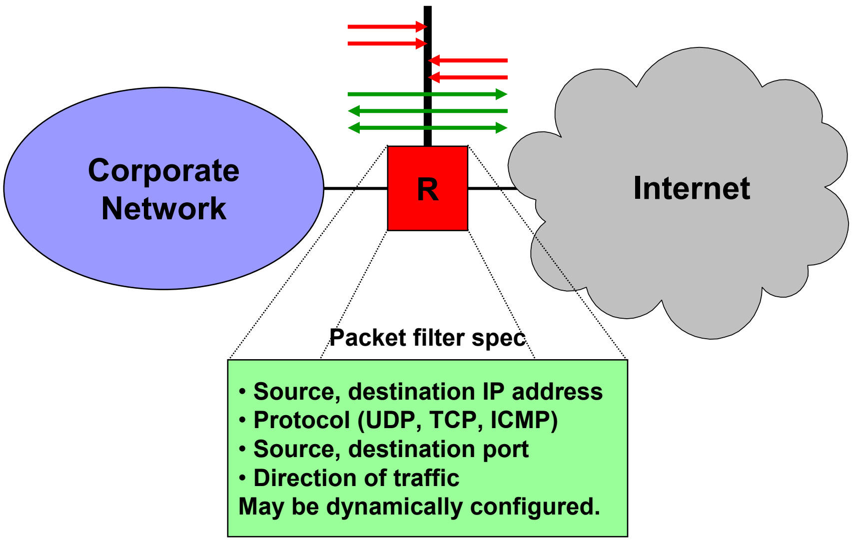
Classifying Traffic (2)

- Traditional approach: quintuple:
  (src IP address, dst IP address, protocol, src port, dst port)
  - Generally used for flow identification
- Hope to identify traffic as "legitimate"
- Issues
  - IP addresses often largely meaningless
  - Attackers also know what may be considered legitimate
    - E.g., src port 20 for ftp-data
  - Dynamic ports
  - IPsec protected traffic: ports become invisible
  - Application layer multiplexing
  - Future transport protocols?

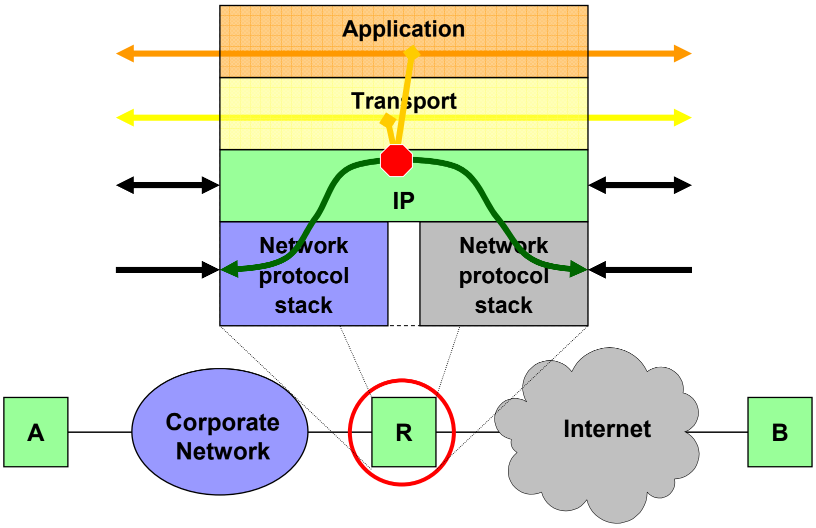
IP Layer: Packet Filter



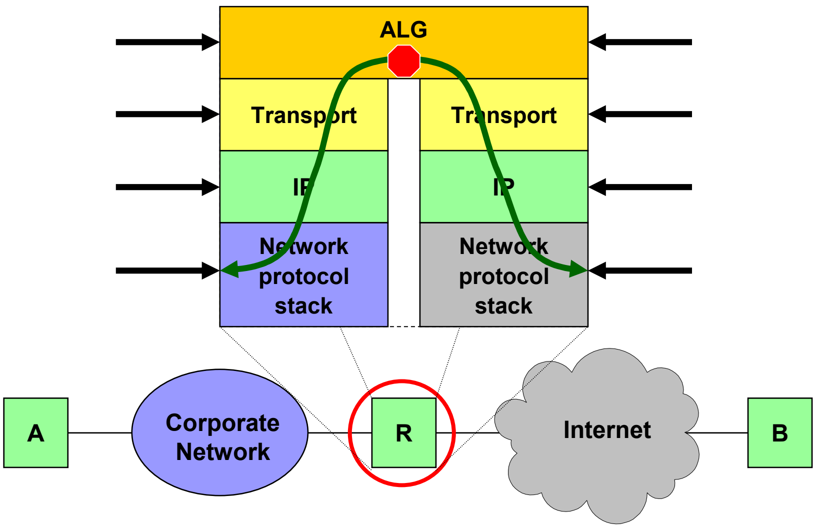
Packet Filter



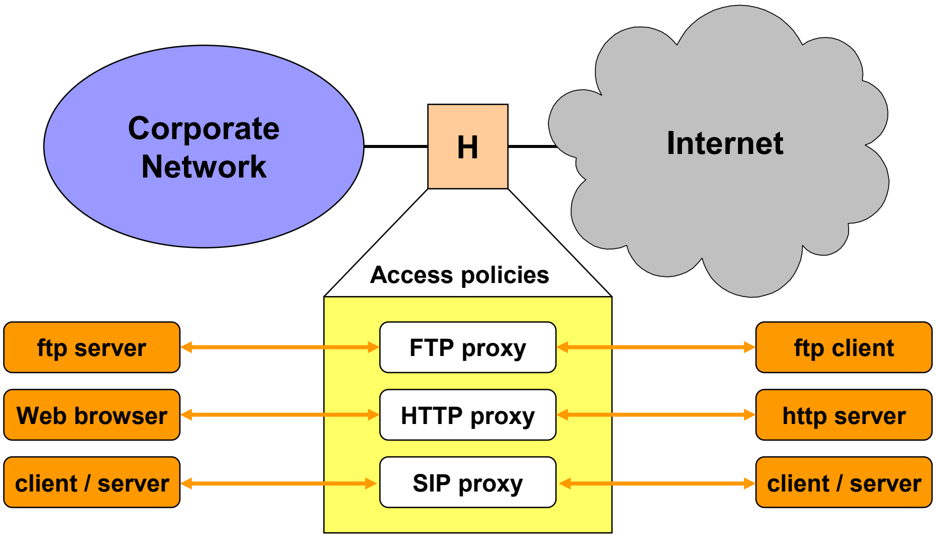
Stateful Packet Inspection



Application Layer Gateway



Application Layer Gateway (Proxy)



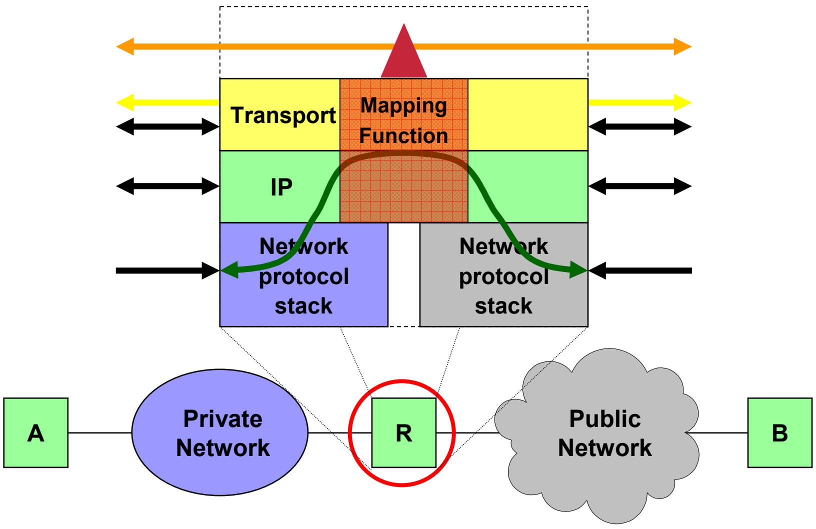
Summary: Firewalls

- Packet filters, enforcing packet altering/forwarding policies
  - Filter specification: Usually statically configured
  - Most configurations disallow packets for “non-standard ports”
- Stateful packet inspection
  - Detect transport or application context of packets
  - Dynamically adapt filter specification
- Application layer gateways
  - Terminate connections: act as transparent or explicitly visible proxies
  - Monitor connection: parse contents of application protocols
    - Functioning precludes end-to-end security!
  - Dynamically adapt filter specification
- Policies may be applied at all layers

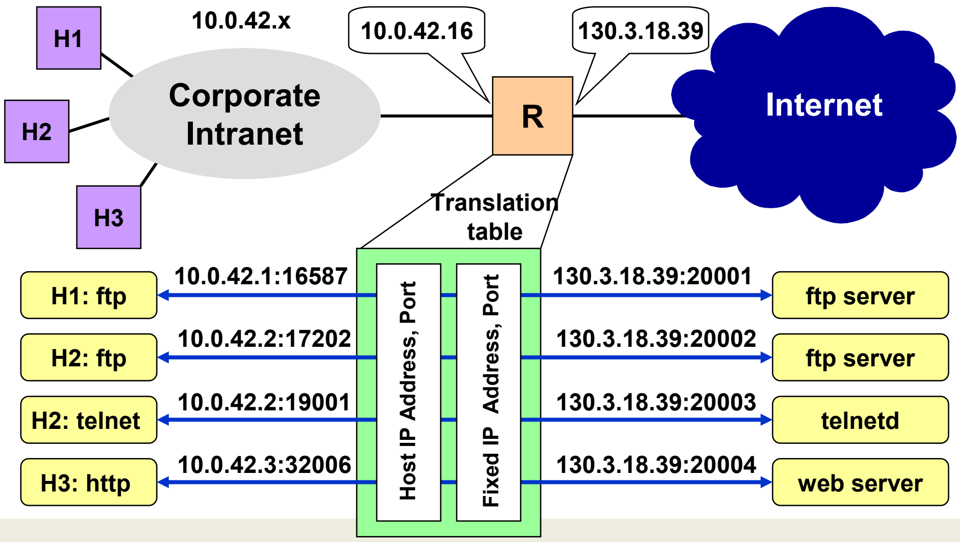
Network Address Translators

- Intermediate systems that can translate addresses (and port numbers) in IP packets
  - Often used to map global addresses to address/port number combination of hosts in a corporate network
- Different motivations
  - Efficient usage of address space
    - Share one globally unique address
    - Use a private address space in the enterprise (10.x.x.x, 192.168.x.x, …)
  - Security
    - Make internal host inaccessible from the public Internet
    - Hide addresses / address structure
- Include dynamically configured packet filters, stateful packet inspection

Network (+Port) Address Translators (NAT)



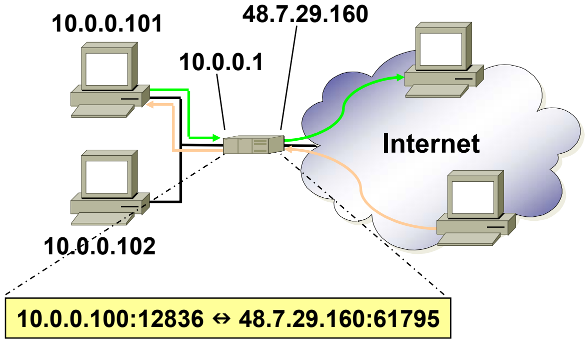
Network Address Translators



Operation of NA(P)Ts

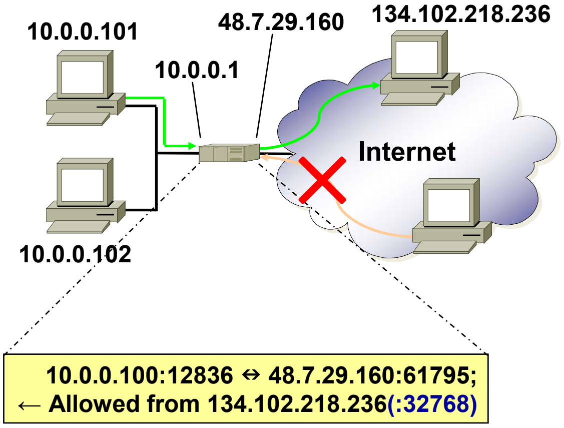
- NATs usually only one-way permeable for initiating connections
  - From private to public network
  - Other direction limited to statically pre-configured addresses
- NATs create address/port number mappings
  - Mappings are usually created dynamically, e.g. on connection setup
  - Static configurations also possible
  - Works best with connection-oriented communication
  - Most common case: TCP connection from client-server sessions
    - Client in private address space, server in public Internet
  - NATs have to keep state for mappings that are tied to “connections”
    - To allow for traffic in the opposite direction to pass
- Which traffic is allowed back in depends on NAT type
  - Important for UDP traffic (i.e. media streams)!

“Full Cone” NAT



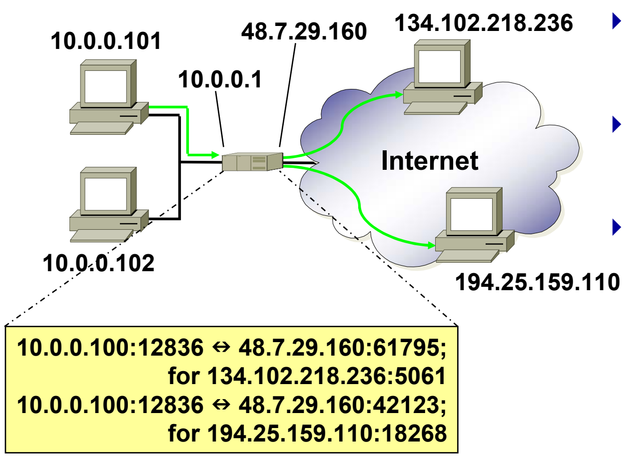
- Outbound packets establish temporary address/port binding
- Any host may respond to mapped address/port
- Incoming packets are dropped when no binding exists

“(Port) Restricted Cone” NAT



- Outbound packets establish temporary address/port binding
- Incoming packets are dropped when no binding exists
- Binding valid only for destination IP address (and optionally port)
- Packets from other hosts (and ports) are dropped

“Symmetric” NAT



- Port Restricted Cone NAT behavior
- Different bindings for different destinations
- Drop incoming packets when no appropriate binding present

Some Assumptions for NATs and Firewalls

- Applications follow client-server paradigm
  - Communications are usually invoked from the inside
- Traffic is self-describing
  - Example: applications use well-defined ports
  - Example: TCP ACK bit indicates established connection
- Connection-oriented protocols (e.g. TCP) dominate
  - Beginning and end of communication session can be identified
- Communications from the outside limited to a few servers
  - Often placed in a DMZ

Some Issues with Firewalls and NATs (1)

- Fragmentation
  - Outbound: Fragmentation ID collision (unique per source IP address)
  - Inbound: Fragments cannot (easily) be forwarded (port numbers are missing)
- Packet forwarding
  - IPsec end-to-end does not work
  - ICMP state needed
  - Integrated services?
- Configuring NATs / firewalls
  - Inbound vs. outbound connections – what is inbound, what is outbound?
  - Per-endpoint restriction (sender, receiver) may be desirable
  - How to identify and authenticate users and their flows in a middlebox

Some Issues with Firewalls and NATs (2)

- Running servers (on well-defined transport addresses)
  - Firewalls: Allow specific transport addresses to be reachable (“www.tkk.fi:80”)
  - NATs: Specify port forwarding for specific nodes
    - Port 80 of a public IP address is mapped to one particular private IP address
    - Issue: Only one entity per port number
- Running peer (and peer-to-peer) protocols
  - Firewalls: issue with dynamically assigned IP addresses
  - NATs: Port forwarding impossible: only one entity per port number

Some Issues with Firewalls and NATs (3)

- Major issue: Non-predictable addresses
  - Dynamically negotiated addresses during communications
  - Symmetric communication relationships with different client addresses
  - (Invocation of) communications from/to unknown peers
- Trivial example: FTP
  - Data transfer uses newly opened TCP connection (from server to client)
  - Client supplies parameters dynamically (valid only for limited period of time)
  - Firewall: who is prepared to receive incoming connections when?
  - NAT: address translation renders specified address unusable
    - Private address “leaks” to a public node
  - FTP remedy: passive mode → reverse connection setup direction
  - Implicit assumption: server resides in public address space and is not protected by a firewall

Some Issues with Firewalls and NATs (4)

- Non-trivial example: SIP-based telephony
  - Both peers may or many not be behind NATs/firewalls
  - Many peers may be behind the same NAT/firewall
  - Signaling (reachability) solved moderately well within SIP
  - One issue (out of many): Uses UDP-based media streams
  - No connection setup, no client-server relationship
  - Firewalls will drop packets: Phones allow specifying fixed port ranges
  - NATs will invalidate addresses
- Side issue: 10.0.0.5 ≠ 10.0.0.5 ?
  - Private address spaces are often the same (meant to be!)
  - Is a received address local (and thus valid) or remote (and hence not valid)?

Increasingly relevant for modern protocols beyond plain client-server!

Summary: Firewall and NAT Applicability

- Firewalls and NATs help against unwanted traffic from the outside
  - Denial-of-Service attacks, port scans, break-in attacks, worms
  - ALGs against viruses
- But: Firewalls and NATs may also prevent legitimate traffic
  - Evil effect on IP communications: Break end-to-end model
  - Have many implicit assumptions about protocols
  - Do not work well with a number of protocols
    - Including their security features
- Just one piece in a security portfolio, to be applied wisely
- Applications and protocols still need security
- Users and their behavior still pose a significant risk

But they are real and they will stay around!

Dealing with Firewalls and NATs

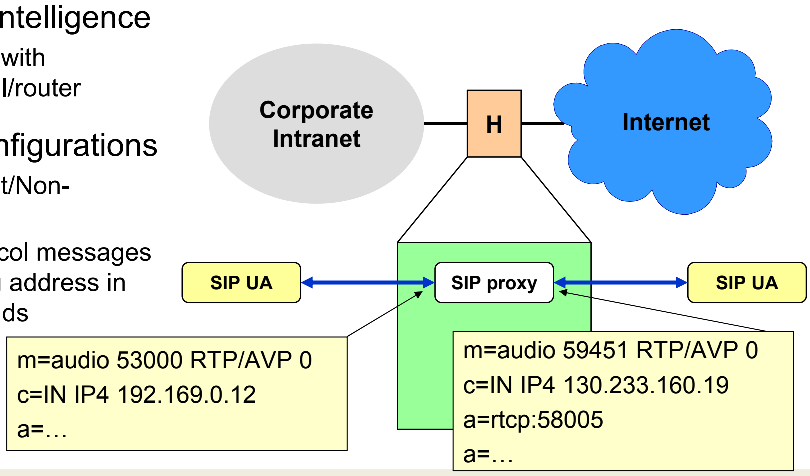
- [Write only client-server protocols and place the server in the open Internet — or something similar…]
- Application Layer Gateways
- Middlebox Communications (MIDCOM)
- Simple Traversal of UDP through NATs (STUN*)
- Travel Using Relay NAT (TURN*)
- Interactive Connectivity Establishment (ICE*)

*) Unilateral Self-address fixing (UNSAF) considerations (RFC 3424)

Application Layer Gateways (ALGs)

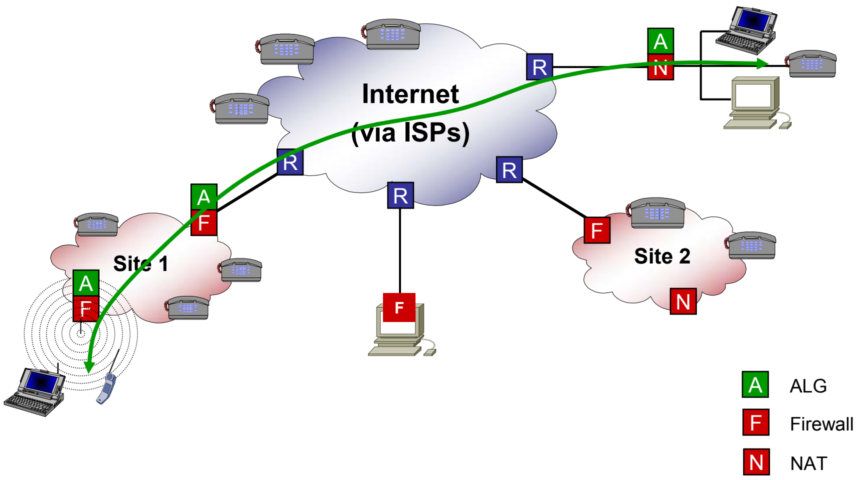
Application Layer Gateway (1)

- Gateway system with application intelligence
  - Co-located with NAT/firewall/router
- Different configurations
  - Transparent/Non-transparent
  - Fixes protocol messages by adapting address in different fields



HELSINKI UNIVERSITY OF TECHNOLOGY
NETWORKING LABORATORY

Application Layer Gateways



HELSINKI UNIVERSITY OF TECHNOLOGY
NETWORKING LABORATORY

SIP Application Layer Gateway (2)

- Many issues
  - Conflicts with security (e.g., signed or encrypted message contents)
    - TLS: client-side certificate check will not succeed
    - Snooping-only ALG may not even see the relevant information
    - Essence: ALG must become part of (trusted?) application infrastructure
  - ALG solution requires application-specific support for each application
    - Have to be upgraded for new applications
    - Application protocols may be complex (ALG builders may not get them right)
    - Feature race between application protocol designers (and implementers) and ALG vendors
  - Scalability
    - Functionality concentrated on single NAT/ALG box
    - Must be available on all entities along the path
  - Robustness
    - Intermediary boxes become single points of failure (unless state sharing protocol implemented) even if the application protocol itself supported failover
  - Reliability
    - Rewriting of protocol messages not robust with respect to extensions, future protocol versions etc.

Explicit Middlebox Signaling

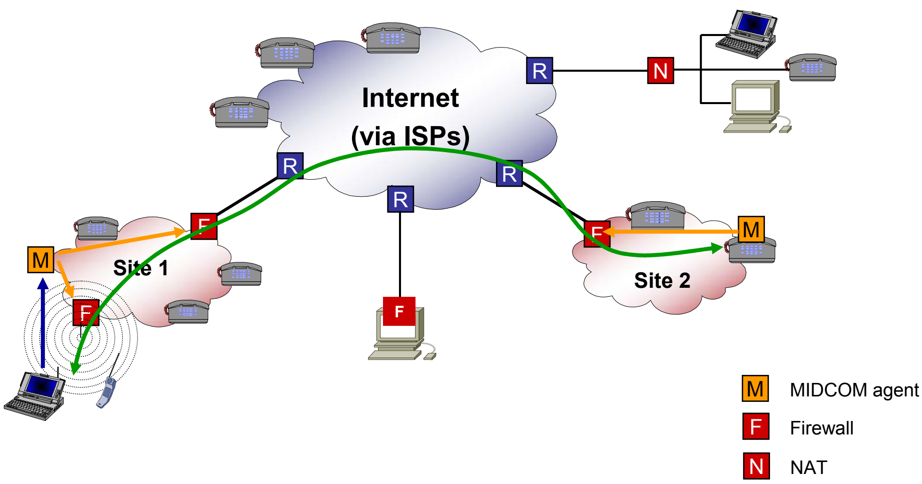
MIDCOM

- Idea: Application-independent Control Protocol
  - SIP UA (or proxy) controls on-path intermediaries
    - Open pinholes, obtain NAT bindings etc.
  - Example: UPnP control of DSL routers
- Requirements specification: RFC 3304
- Abstract protocol semantics: RFC 3989
- Evaluation of Candidate Protocols: RFC 4097
  - Simple Network Management Protocol (SNMP)
  - Realm-specific IP (RSIP)
  - Media Gateway Control (MEGACO)
  - Diameter
  - Common Open Policy Service (COPS)

Trivial Example: SOCKS (RFC 1928)

- SOCKS allows a client to communicate via a middlebox
  - Protocol between client "behind" middlebox and middlebox
- Operations
  - Bind to an externally visible address (and obtain this address) at the middlebox
  - Connect via a middlebox to a TCP peer
  - Create an association for a UDP flow via the middlebox
    - UDP-in-UDP tunneling of datagrams
- Authentication with the middlebox needed
- Usable for
  - IPv4-IPv6 translation
  - NAT and firewall traversal

MIDCOM



MIDCOM Issues

- Needs to be standardized in the first place
- Must be supported by vendors (may lose their competitive edge)
  - If so, products need to become available and to be deployed
- Location problem: How to discover intermediaries?
- Organizational problems: Security Policy
  - Cannot control NAT box of public ISP
    - E.g., in a WLAN hot-spot
    - Motivation for the hot-spot operator?
  - Authentication of users and authorization of operations
- Must be really secure (authentication, authorization)
  - Hard to achieve
  - Example: UPnP is rather insecure today
  - And: third parties may misuse pinholes once created

Short Excursion:
End-to-Middle Communications

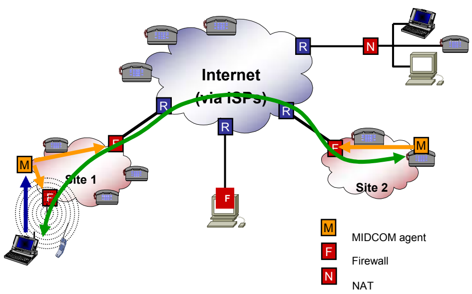
Some thoughts inspired by Xiaoming Fu (Uni Göttingen)

Motivations

- Previous slides: enabling/disabling traffic from/to certain nodes for certain applications using certain protocols
- Historically more general problem: Quality of Service
  - Going beyond best effort traffic treatment of a media flow
  - “What if we can change the network?”
- Signaling from application (hosts) to routers about flow handling
  - Per-flow QoS provisioning
  - Flow blocking (“extreme QoS”) or passing
  - Flow routing/forwarding (path selection, label distribution)
  - Flow processing (“Active Networks”)
- Flow identification
  - Quintuples, flow labels, ToS fields, … (extreme: contents)
- Remember in all cases: routers have to remain efficient and scalable

Functional Requirements

- Endpoints need to agree
  - Two or more than two?
- Endpoints need to locate (the relevant) routers on the path
  - In both directions (remember: asymmetric paths are possible)
- Endpoints want to install state
  - Routers need to authorize actions
  - Need to consider policies
    - Intra- and inter-domain
- Need to deal with route changes
  - Follow the routes or fix the routes
- Need to remove state
  - Invoked by the endpoints or cleanup



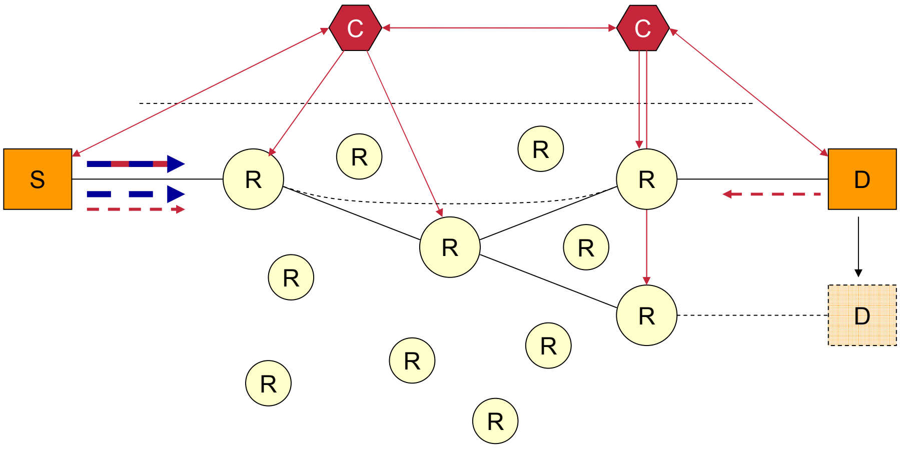
How to Signal?

- Initiator
  - Source vs. destination(s) vs. all
- Router location
  - Configured (known) routers: use some "end-to-end" protocol
  - Implicit location: control packets pass through nodes and cause actions
  - Explicit location: running a separate location protocol
- In-band vs. out-of-band
  - In-band: data and control share the same communication channel (packets)
  - Out-of-band: uses separate signaling channel ("control plane", separate packets)
- Path-coupled vs. decoupled
  - Path-coupled: data and control take the same path (all the way)
  - Path-decoupled: uses an independent path for control (parts/all of the way)
- Relationship to IP routing
  - Integrated: signaling state is established that guides data packet forwarding
  - Influencing: extends the basis for forwarding decisions (beyond the destination IP address)
  - Independent: data and control packets (and state) follow the IP routing/forwarding

How to signal? (2)

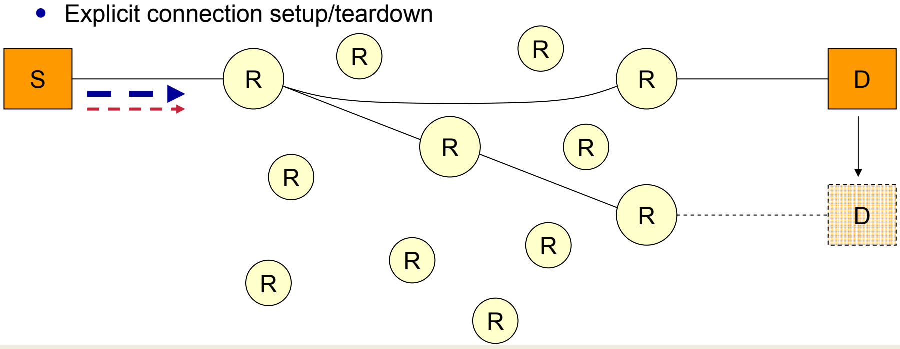
- Soft-state vs. hard-state
  - Reliability: hop-by-hop vs. end-to-hop
  - Responsibility for state: next hop vs. hosts
- Mobility
  - How to minimize the (end-to-end) overhead when hosts change points of attachment to a network?
  - How to ensure seamless QoS?
  - How to maintain QoS in the first place (and how to deal with failures)?
- Security
  - Authentication, policies, authorization
- Multicasting...
  - Point-to-point a special case of multicasting?
  - Treat point-to-point separately?
  - Multicasting adds too much complexity for too little value (painful experience)

How to signal? (3)



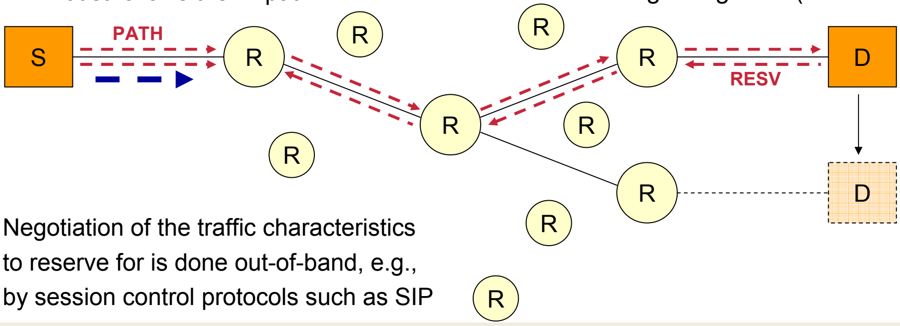
Example 1: ST-II (RFC 1190)

- Source-initiated
- Implicit router location
- Out-of-band signaling (but own data packets)
- Integrates routing
  - Explicit connection setup/teardown
- Hard state
- No mobility support
  - Requires explicit reconfiguration
- Security: essentially none
- Multicasting: integrated (limited scale)



Example 2: RSVP

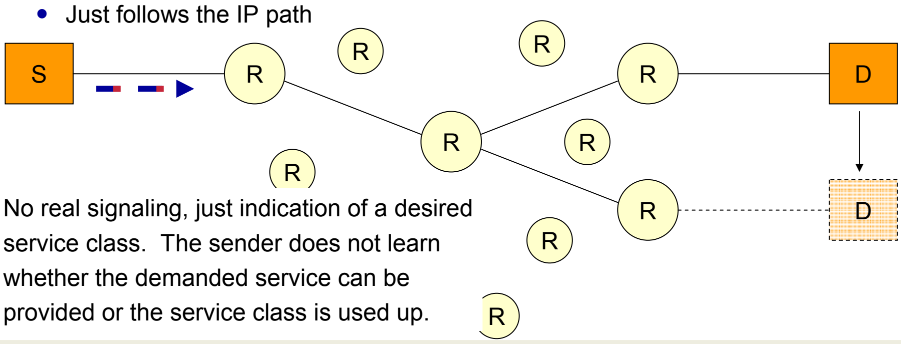
- Source-initiated PATH message and destination initiated RESV message
- Implicit router location
- Out-of-band signaling
- Independent of IP routing
  - Just follows the IP path
- Soft state (end-to-end)
- Mobility support through soft state
  - But optimizations required for efficiency
- Security: security and policy objects
- Multicasting: integrated (limited scale)



Negotiation of the traffic characteristics to reserve for is done out-of-band, e.g., by session control protocols such as SIP

Example 3: Differentiated Services

- “Source-initiated”
- Implicit router location
- In-band signaling (ToS field)
- Path-coupled
- Independent of IP routing
  - Just follows the IP path
- No state (just handling indication)
- Implied mobility support
- Security: none
  - Validation through border routers
- Multicasting: integrated



No real signaling, just indication of a desired service class. The sender does not learn whether the demanded service can be provided or the service class is used up.

End of Excursion

- Controlling nodes (“middleboxes”, routers) in the network is tricky
  - Need to get many things right
  - May easily increase brittleness
  - May raise interoperability issues
  - Surely has deployment problems
- Careful design required
  - To maintain the robustness properties of the Internet
  - Not to create unforeseen feature interactions
  - Beware of security issues and new angles for DoS and other attacks
- At the end of the day, the applications cannot rely on a completely controlled path in the open Internet
  - Need to (be prepared to) work around these issues
  - Need to be adaptive to the networking conditions

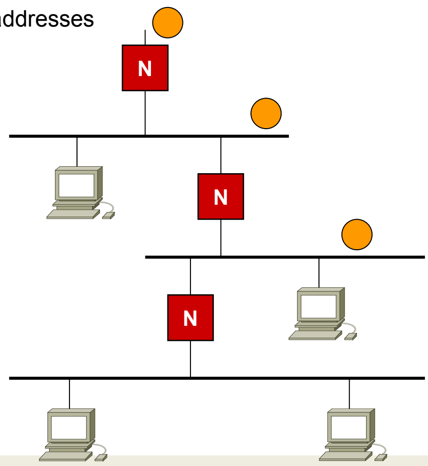
NATs: Determining usable “outside” addresses in the endpoints

(Unilateral Self-Address Fixing, UNSAF)

- Maintain end-to-end idea as much as possible
- Examples: STUN, TURN, ICE

Reminder

- NATs translate “internal” transport addresses (IP address, port) to “external” ones
  - Using one or more “external” IP addresses
  - External address may or may not be public IP addresses
- NATs may be cascaded
- Address space re-use in different realms
  - 10.0.0.5 ≠ 10.0.0.5 ?
- Different NATs use different rules
  - How to choose the “next” external address
  - When to choose a new external address
    - Source IP address and port number mandatory
    - Optional: Destination IP address and port number
  - Which filter rules to install for inbound traffic
    - Traffic directed at an allocated port mapping
    - Traffic originating from a transport address a packet was previously sent to
  - Cleaning up NAT bindings
    - TCP: typically tied to connection state
    - UDP: typically handled via timeouts
    - Other protocols: may or may not be supported



UNSAF Considerations for NATs

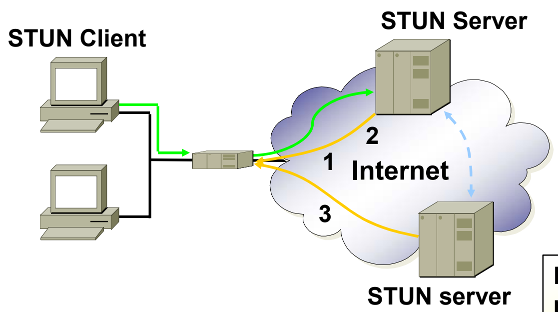
- There is no uniquely determinable “outside” to NATs
- Addresses can only be determined relative to a specific point in the network
  - It may not be known “where” this point is
  - An UNSAF service may have a different viewpoint with respect to an entity and thus see a different “relative” address compared to the peer of the entity
- Enabling incoming traffic may circumvent other security measures
- Basing future operation on past observations is risky
- UNSAF services and middleboxes may increase brittleness

MIDCOM Issues

- Needs to be standardized in the first place
- Must be supported by vendors (may loose their competitive edge)
  - If so, products need to become available and to be deployed
- Needs to be really secure (authentication, authorization) – hard to achieve
  - Example: UPnP is rather insecure today
- Location problem: How to discover intermediaries?
- Organizational problems: Cannot control NAT box of public ISP
  - e.g., in a WLAN hot-spot
  - Authentication of users and authorization of operations
  - Motivation for the hot-spot operator?

RFC 3489: Simple of UDP Through NATs (STUN)

- Detect NAT type and public IP address
  - External server echos observed source address and port
  - Optionally request IP address and/or port change for response
- Still not available for requests from any host outside...

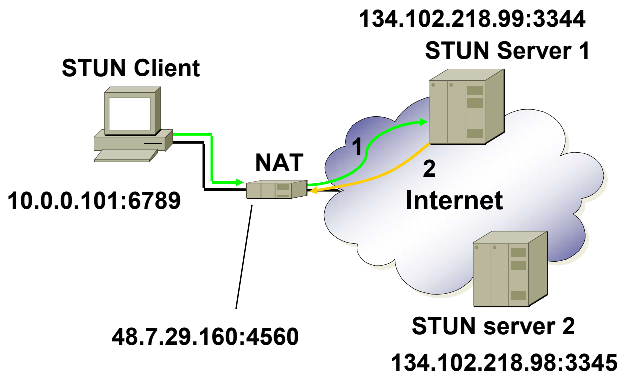


1. Echo source address, send from recv port
2. Client requested port change
3. Client requested address change

Received/dropped responses determine type of NAT

RFC 3489: Simple Traversal of UDP Through NATs (STUN)

1. **Binding request to STUN server 1**

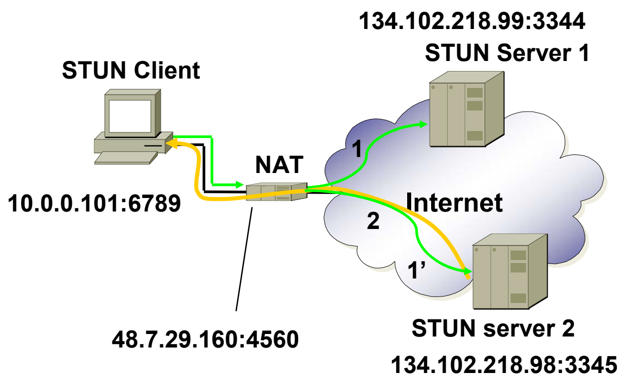


(may be the same physical machine
and just use a different IP address/port)

- No Response?
  - No UDP connectivity, give up
- Response
  - Server returns MAPPED-ADDRESS (48.7.29.160:4560)
  - "Reflexive address" learned by the client
  - ➔Client is behind NAT
- But what type of NAT?

RFC 3489: Simple Traversal of UDP Through NATs (STUN)

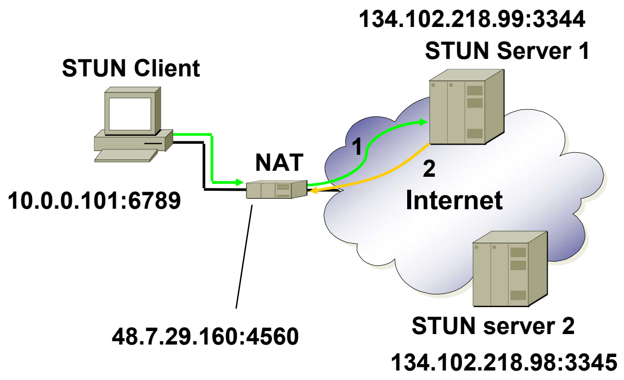
2. **Binding request to STUN server 1 with change IP and change port flags**



- Response?
  - ➔Client is behind full cone NAT
- No Response?
  - Repeat Test 1 (1')
    - Send Binding Request to server 2
  - Server 2 returns MAPPED-ADDRESS (48.7.29.160:4560)
    ➔Client is behind restricted/ port restricted NAT
  - Server 2 returns other MAPPED-ADDRESS
    ➔ Client is behind a symmetric NAT

RFC 3489: Simple Traversal of UDP Through NATs (STUN)

3. Binding request to STUN server 1 with change port flag



- Response?
  - ➔Client is behind restricted NAT
- No Response?
  - ➔Client is behind port restricted NAT
- Repeat transmissions because of potential packet loss

STUN Security

- Anybody could send UDP messages with faked IP addresses
  - Gives rise to numerous attacks
- Establish a shared secret between client and server
  - Performed via TLS (i.e., reliable and secured transport)
  - Server authenticated by means of certificate
  - Server issues temporary “username” and “password”
  - Used in subsequent UDP-based STUN binding requests for authentication
- Alternative: STUN client and server share a signaling relationship
  - E.g. a SIP dialog when the STUN server runs on the peer system
    - STUN server dynamically instantiated on each RTP or RTCP port
  - Leverage the trust previously established – no need for TLS connection

RFC 3489bis

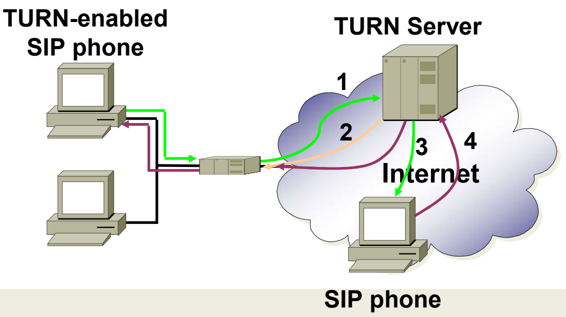
- Simple Traversal Underneath Network Address Translators (STUN)
  - draft-ietf-behave-rfc3489bis-05.txt
- Removes attempt to understand and identify NAT types
  - Full cone, (port) restricted cone, and symmetric are only a rough classification
  - Symmetric is the most important
    - Existence determined differently → see TURN and ICE
- Adds XORed reflected transport addresses
  - Plus some other fields
  - More thought on demultiplexing
- Generalizes operations: base protocol + usages
  - Request-response pairs + server-initiated indications
  - Short-term password usage: TLS-based sharing of a secret
  - Binding usage: simple address discovery
  - Keepalive usage: maintain the NAT bindings alive
  - External: TURN usage: support packet reflection by a server

STUN Summary

- STUN provides a means for an application to traverse NATs
  - Detect existence of NATs
  - [Detect type of NATs]
  - Maintain address bindings alive in NAT
  - Learn address bindings and usable public address
  - Intended for enabling peer-to-peer communication in NAT scenarios
- Not a complete solution
  - Symmetric NATs still a problem
  - Does not help if both peers are behind NATs
- Approach to deal with symmetric NATs
  - Run STUN server with each media endpoint
  - (on each RTP/RTCP port)
  - Does not help if both endpoints are behind different NATs

Traversal Using Relay NAT (TURN)

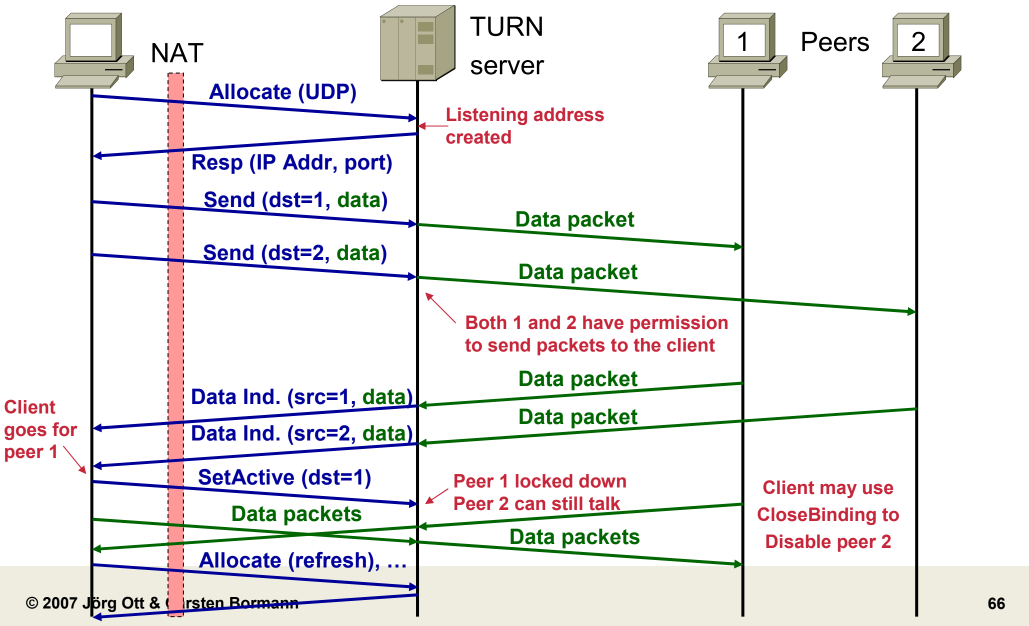
- Idea: Provide forwarding service in the public internet
  - Client behind NAT has single connection to TURN server
  - Server forwards incoming packets destined for TURN client → *Relay NAT*
  - Protocol-agnostic – no ALG needed
  - Authentication to prevent DoS-attacks (similar to STUN)



1. **Allocate TURN port**
2. **TURN-response, including address/port**
3. **SIP registration with TURN contact**
4. **SIP request from outside is routed to TURN Server**

(similar for media)

STUN Relay Details



STUN Relay Details (2)

- Uses STUN framework for message exchanges
  - Defines new STUN usage
  - Uses the same authentication mechanisms
  - STUN and TURN servers likely to be identical
- Relaying of both UDP and TCP
  - Mapping between different transport protocols possible
    UDP → UDP, TCP → TCP, TCP → UDP, TLS → TCP, TLS → UDP
  - Identification of a transport relationship by means of a 5-tuple
    - Source, Destination IP address and port, protocol id
    - Internal 5-tuple: NAT–STUN/TURN server
    - External 5-tuple: STUN/TURN server – remote peer
- Introduces additional 4-byte framing
  - Distinguish STUN requests from application data
  - Distinguish framed from unframed STUN messages

Interactive Connectivity Establishment (ICE)

- Networks with segmented connectivity, different address realms
  - Try to find optimal connection between endpoints
  - Use relays only if necessary
  - Support for STUN and TURN
- draft-ietf-mmusic-ice-15.txt
- An end-to-end solution avoiding assumptions about middle-boxes
  - May be obsoleted by middlebox control some fine day…
- Applies to media path, not signaling
  - But signaling must be aware of ICE (specific SDP attributes)
  - Poor default behavior for non-ICE clients
- Abstract signaling model
  - Fits SIP, H.323, RTSP and similar protocols

Operation

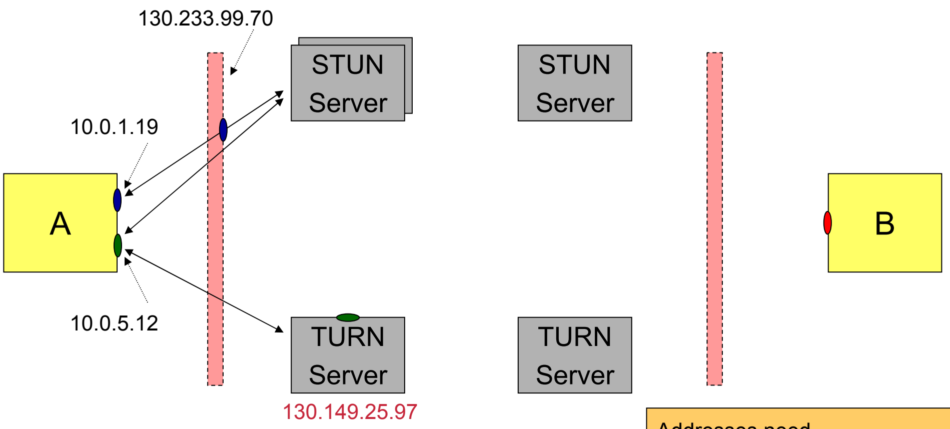
- Idea: peers exchange lists of transport addresses, mutual connectivity tests
- Clients must detect own transport addresses
  - The more, the better
  - Local interfaces (including private addresses, e.g. in 10/8 net)
  - Detection using “external” reflectors (e.g. STUN, TURN)
  - Assigned tunnel addresses (e.g. PPTP)
- Clients run STUN servers on every published transport address
  - Explicit keep-alives for NAT binding
  - Shared with media streams

Operation Details: 9 Steps

[some of the following slides inspired by Jonathan Rosenberg’s ICE tutorial given on 7 November 2006 at the 67th IETF]

- Step 1: Allocation
- Step 2: Prioritization
- Step 3: Initiation
- Step 4: Allocation
- Step 5: Information
- Step 6: Verification
- Step 7: Coordination
- Step 8: Communication
- Step 9: Confirmation

ICE: Address Gathering



Host candidates: 10.0.1.19:51000, 10.0.5.12:51002
Server-reflexive candidates: 130.233.99.70:58000, 130:233.99.70:58090
Relayed candidates: 130.149.25.97:63673

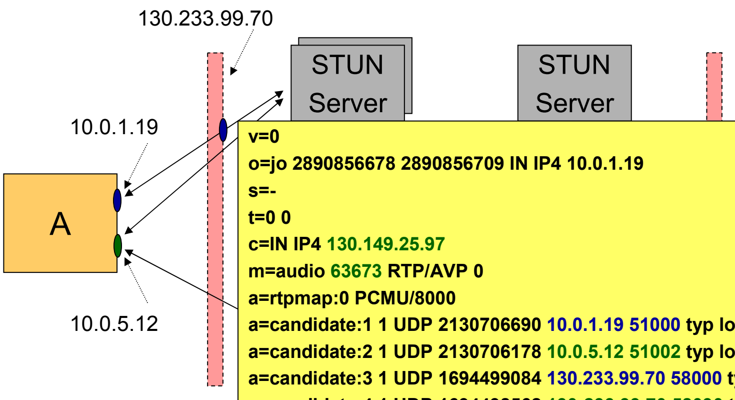
Addresses need
- per media stream (audio, video)
- per component (RTP, RTCP)

Address Gathering and Prioritization

- Address gathering can cause significant traffic
  - Multiple interfaces, IP address versions, STUN servers
  - Multiple media streams and components per stream
  - May cause network or NAT overload
- Pace transmission (20ms intervals)
- Prioritization across candidates:
  - Reflect the quality (e.g., in terms of minimal overhead)
  - Host addresses are better than reflexive ones are better than relayed
  - RTP over RTCP

```
priority = (2^24)*(type preference)
         +(2^8)*(local preference)
         +(2^0)*(256 - component ID)
```

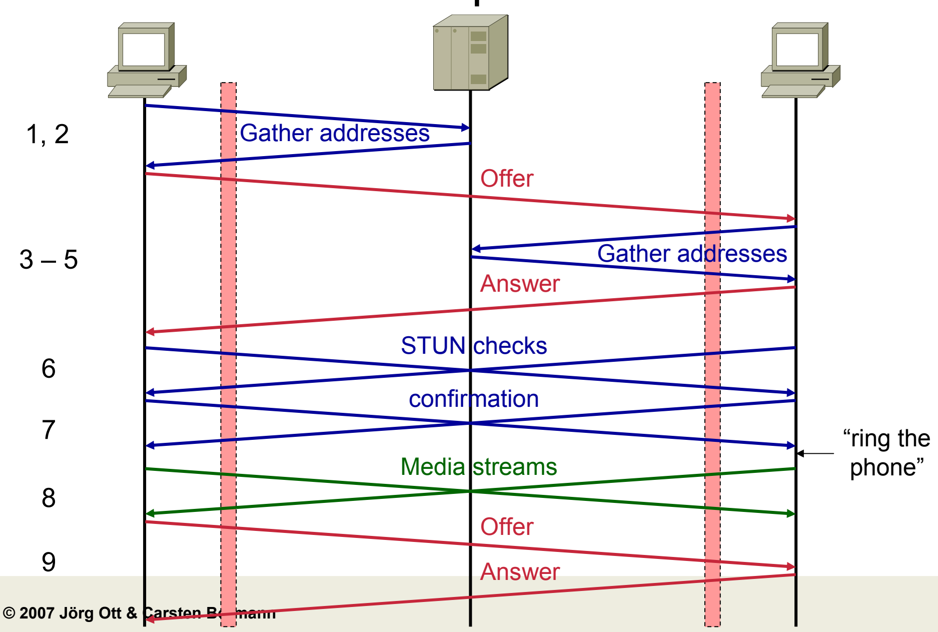

ICE: Address Gathering and Encoding



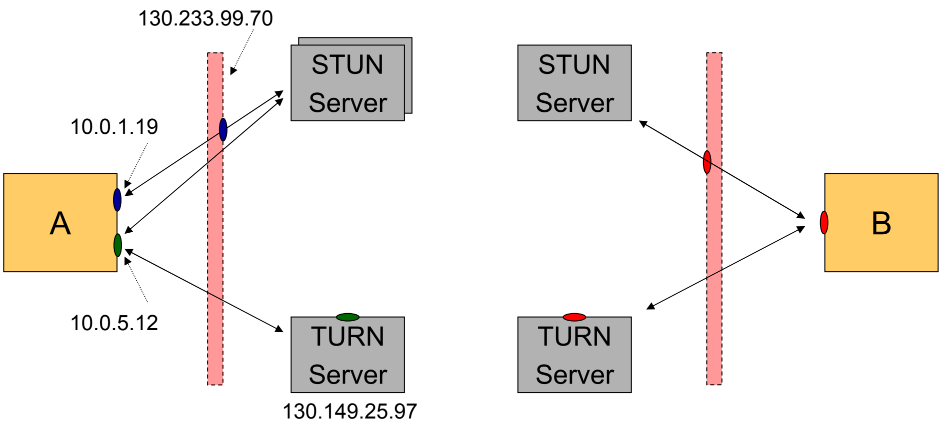

```
v=0
o=jo 2890856678 2890856709 IN IP4 10.0.1.19
s=-
t=0 0
c=IN IP4 130.149.25.97
m=audio 63673 RTP/AVP 0
a=rtpmap:0 PCMU/8000
a=candidate:1 1 UDP 2130706690 10.0.1.19 51000 typ local
a=candidate:2 1 UDP 2130706178 10.0.5.12 51002 typ local
a=candidate:3 1 UDP 1694499084 130.233.99.70 58000 typ srflx raddr … rport …
a=candidate:4 1 UDP 1694498562 130:233.99.70 58090 typ srflx raddr … rport …
a=candidate:5 1 UDP 767 130.149.25.97 58090 typ relay raddr 130.233.99.70 rport …
```

Host candidates: 10.0.1.19:51000, 10.0.5.12:51002
Server-reflexive candidates: 130.233.99.70:58000, 130:233.99.70:58090
Relayed candidates: 130.149.25.97:63673

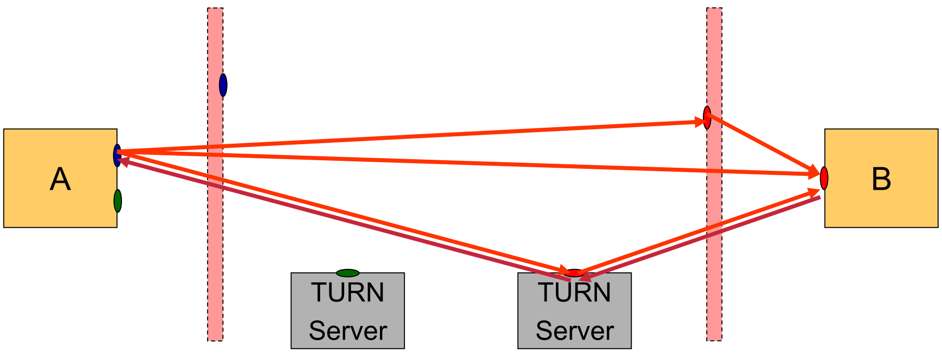
ICE Operation



6. ICE Address Verification



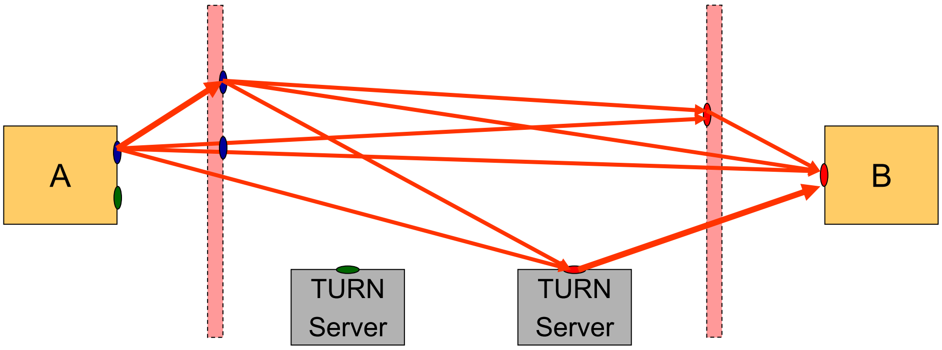
6. ICE Address Verification



- Send test messages and await replies: from all candidates to all candidates
- Successively go (again paced) through all address pairs
- Use priority ordering (pairing) to determine the trial order
  (shall ensure that better ones are tested first)

HELSINKI UNIVERSITY OF TECHNOLOGY
NETWORKING LABORATORY

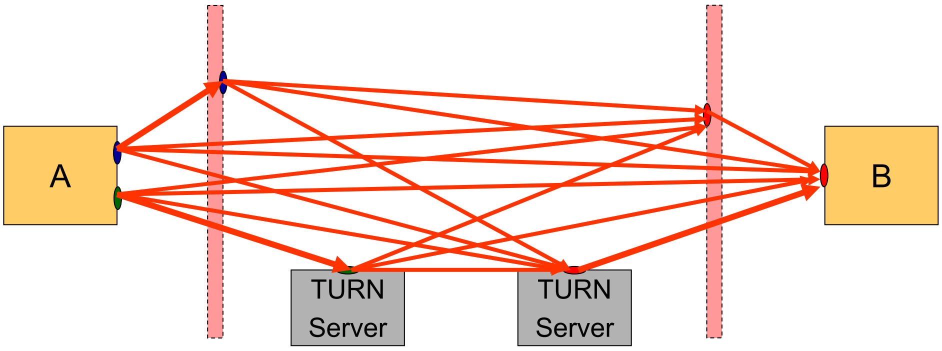
6. ICE Address Verification



- Nodes may learn further candidates from the address checks: peer-reflexive candidates (e.g., B's view of an interface of A)

HELSINKI UNIVERSITY OF TECHNOLOGY
NETWORKING LABORATORY

6. ICE Address Verification



- Successively go through multiple components and multiple media streams
- Use previously successful candidate pairs earlier (adapt prioritization)

Summary: Top 10 ICE Facts

1. ICE makes use of Simple Traversal Underneath NAT (STUN) and Traversal Using Relay NAT (TURN)
2. ICE is a form of p2p NAT traversal
3. ICE only requires a network to provide STUN and TURN servers
4. ICE allows for media to flow even in very challenging network conditions
5. ICE can make sure the phone doesn't ring unless media connectivity exists
6. ICE dynamically discovers the shortest path for media to travel between endpoints
7. ICE has a side effect of eliminating a key DoS attack on SIP (Voice Hammer)
8. ICE works through nearly any type of NAT and firewall
9. ICE does not require the endpoint to discover the NATs, their type, or their presence
10. ICE only uses relays in the worst case – when BOTH sides are behind symmetric NAT

Design Aspects for Application Protocols (1)

- Operation without specific support from middleboxes
  - Guidelines for application protocol design for NATs: RFC 3235
    - Fairly general statements of limited usefulness (nothing really new in 2002)
    - Don't send addresses in the payload
    - Avoid session bundles
    - Session bundles originate from the same end (typically the client)
    - Prefer connection-oriented transport
  - STUN, TURN, ICE: one solution set preserving end-to-end model
- Frequent "fallback" position: tunneling through HTTP (port 80)
  - This SHOULD NOT be the default option — may subvert security
  - Endless race between firewall vendors and application designers
  - "Smart" firewalls analyzing port 80 contents may have undesired side effects
  - The same applies to other well-known ports

Design Options for Application Protocols (2)

- If you want to work with ALGs
  - Design your protocol "in the open" (publish it!)
    - Need to motivate middlebox vendors to support it — or forget about it
  - Self-describing (ideally per packet!) traffic; easy to parse
  - Separate communicated transport addresses from other protocol parameters
  - If needed, avoid securing these (only) in the signaling protocol
    - Move validating towards the dynamically established transport instead
  - Perform in-band protocol validation and negotiation (within a session)
    - Minimize cross-session dependencies
- Communication architecture
  - Make use of representative nodes ("servers", "proxies", "super-nodes", etc.) if possible and useful for the application
  - But beware of introducing additional points of failure, scaling issues, etc.
    - And the need for operations and management

Design Options for Application Protocols (3)

- Protocol design itself
  - Don't fragment
  - Introduce additional (application layer) demultiplexing
    - To reduce the need for transport bundles
  - Avoid communicating addresses in the payload if possible
  - Otherwise: make use of UNSAF and/or middlebox traversal mechanisms as applicable
    - Using STUN, TURN, ICE requires demultiplexing e.g. STUN and application protocol messages on the same transport address ("socket")
    - Negotiation protocol needed (currently ICE only specified for SDP and offer/answer)
  - Minimize brittleness
    - Use minimal number of addresses
    - Observe and deal with communication failures
  - Be careful with assumptions
    - (non-)existence of middleboxes; operation of a middlebox
    - Which side of the middlebox you are on