SIP & NATs / Firewalls

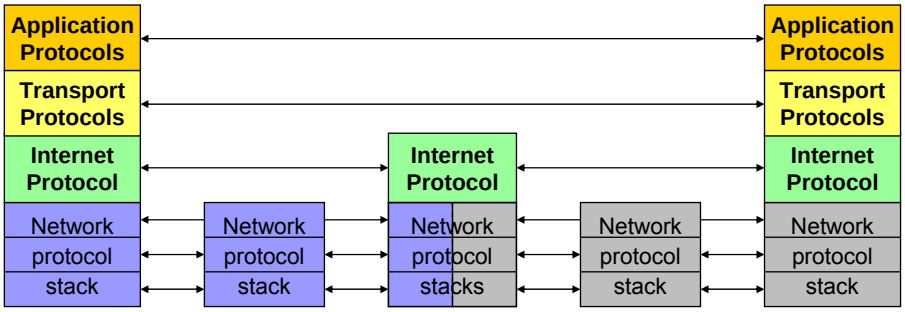
"The primary purpose of **firewalls** has always been to **shield buggy code** from **bad guys**."

Steve Bellovin, IETF Security AD

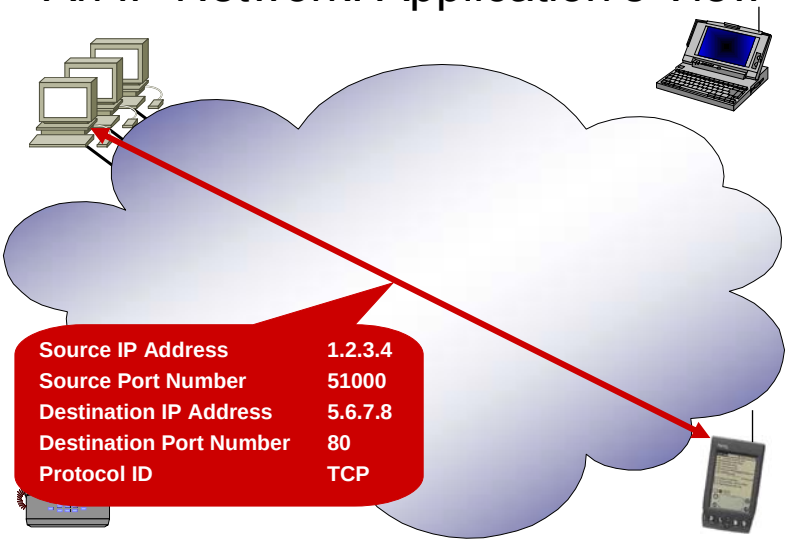
Slide contributions by Olaf Bergmann (Uni Bremen TZI)

Reminder: Internet Architecture

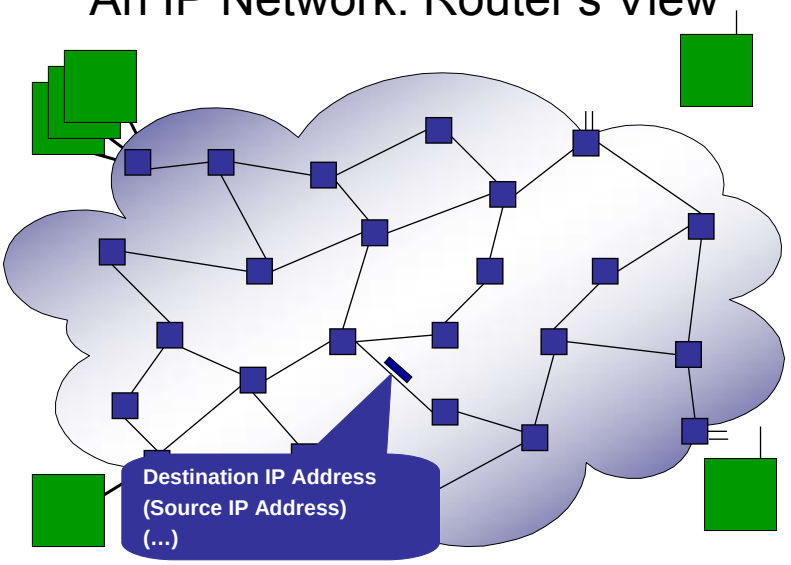
- Transport and application protocols operate end-to-end
  - Port numbers: addressing of processes (applications)
  - Network-components and topology are invisible
  - All functions performed end-to-end



An IP Network: Application's View



An IP Network: Router's View

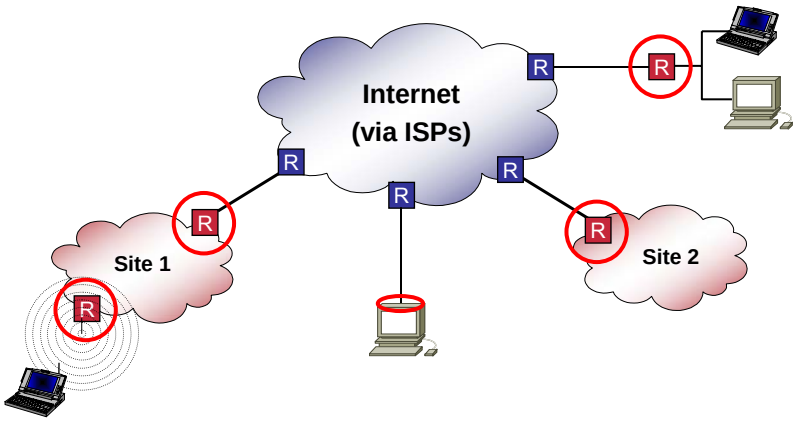


Key Concepts of the Internet Architecture

Hosts know nothing about the network.

Routers know nothing about applications.

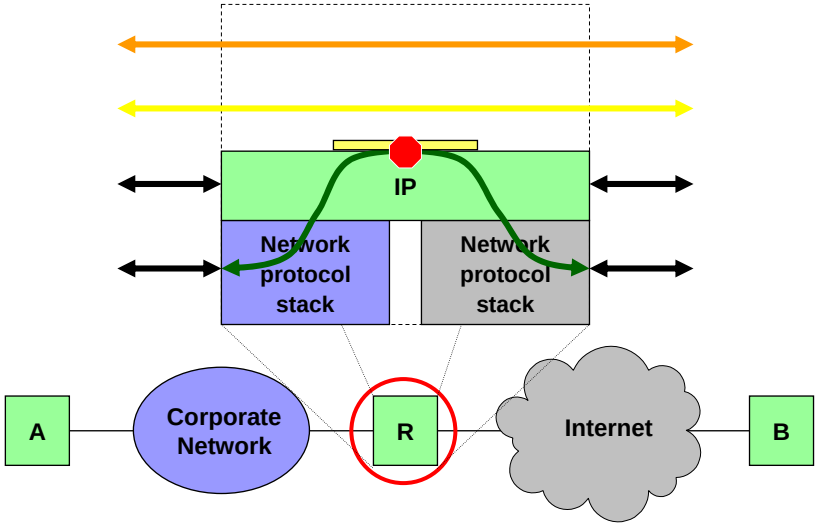
A Sample Network Setup



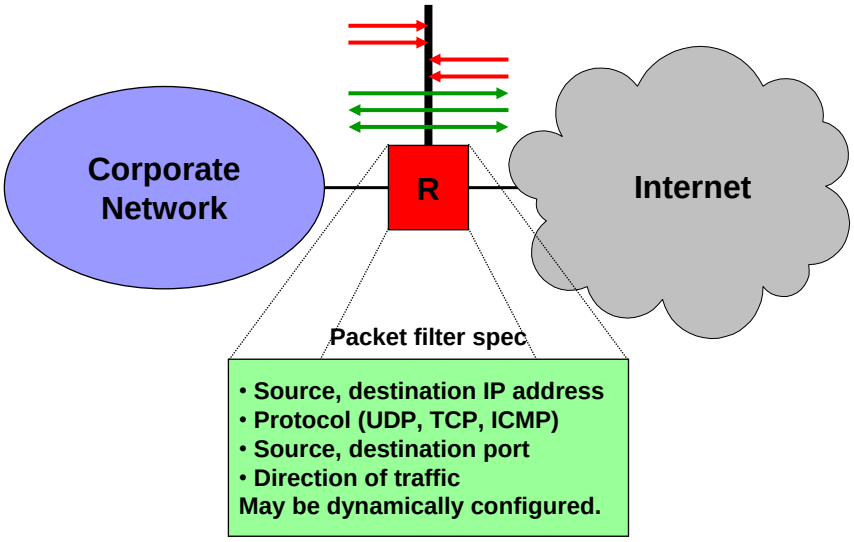
Recap: “Security Devices” for IP Networks

- 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

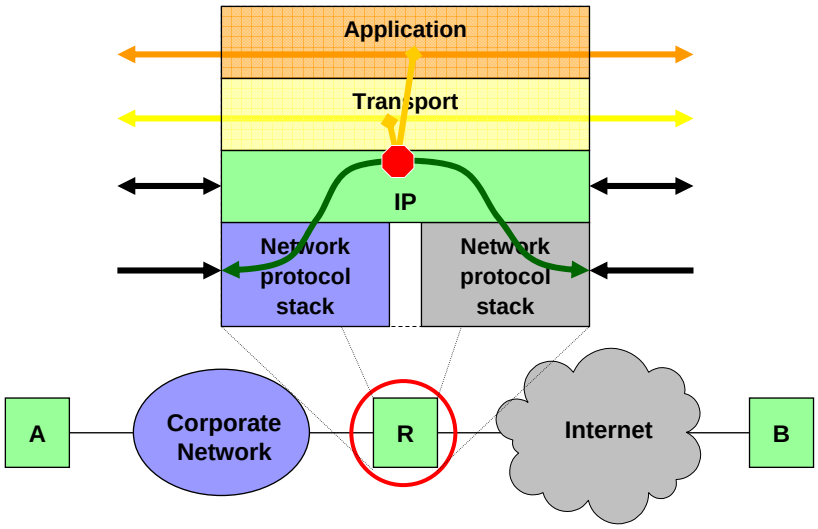
IP Layer: Packet Filter



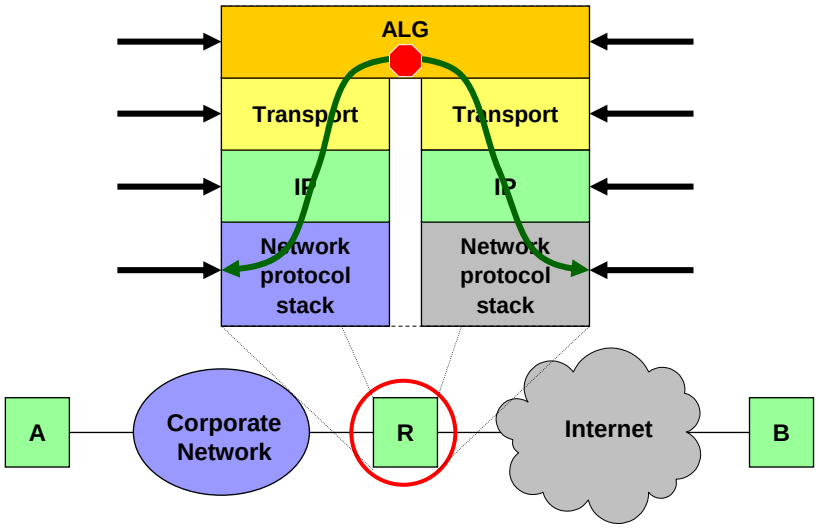
Packet Filter



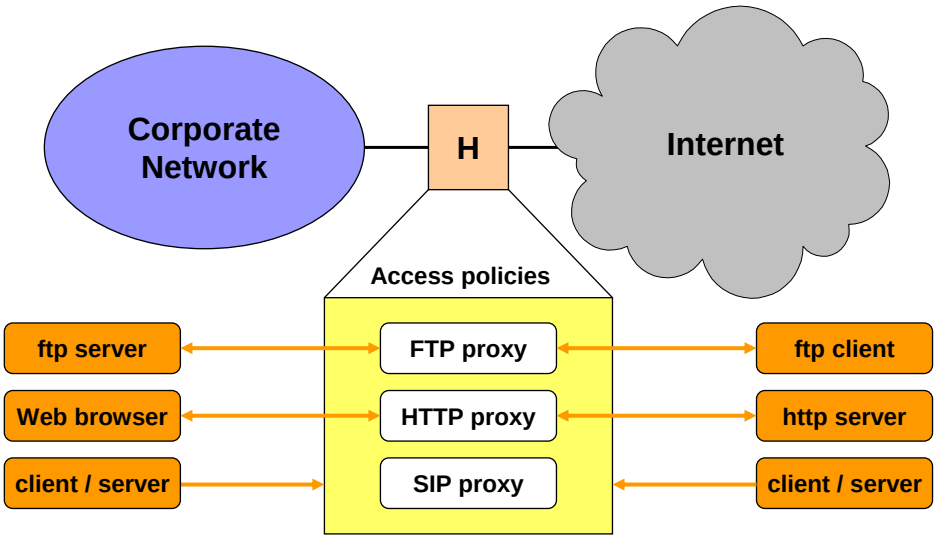
Stateful Packet Inspection



Application Layer Gateway



Application Layer Gateway (Proxy)



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

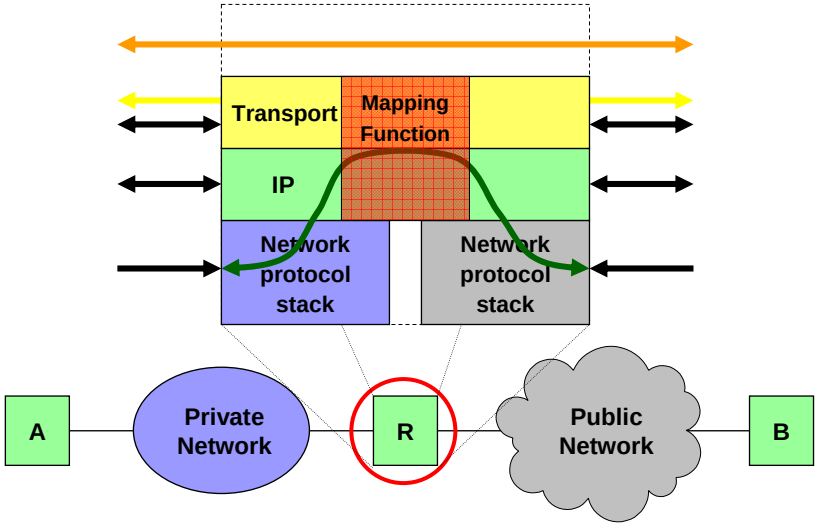
Firewalls and SIP

- Default configuration of many firewalls:
  - Allow outbound TCP and outbound UDP for DNS
  - Allow inbound TCP for well-known TCP ports only (e.g. HTTP)
- SIP communication disallowed in most cases
  - Port 5060 (UDP, TCP)
  - Simple fix: change filter firewall policy to allow port 5060
    - Still a problem with SIP communication on other ports…
- The real problem: Media Sessions
  - Transport parameters are signalled by SIP messages
    - Negotiable, not known in advance
    - Different transport mechanisms for different applications
- Solution approaches
  - Have firewalls be controlled by ALGs → SIP proxy server
  - Adapt SIP to make it more firewall-friendly → SIP firewall traversal
  - Define firewall-control protocols to open firewalls dynamically
    - Issues: Authentication and authorization → MIDCOM

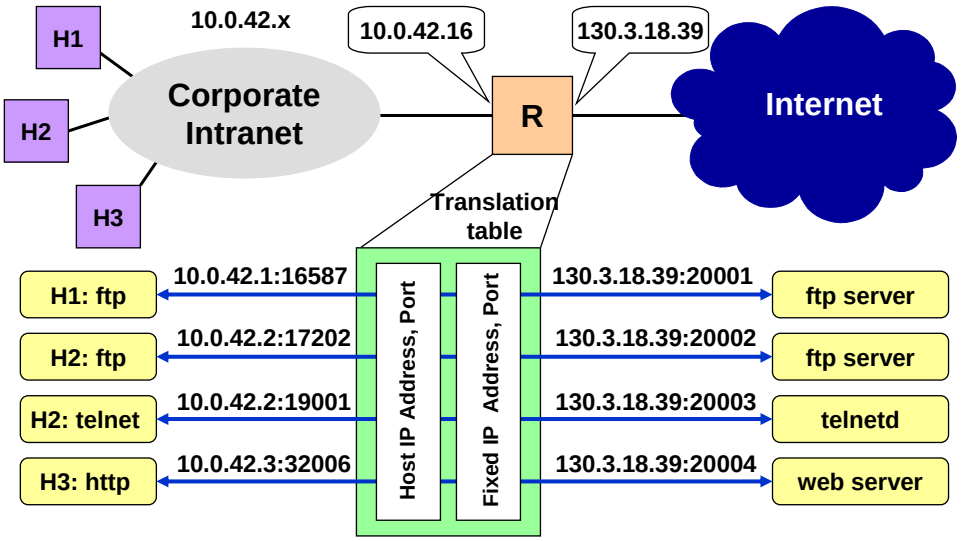
Network Address Translators

- 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)



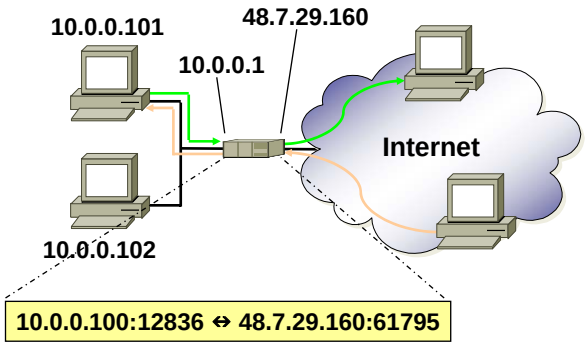
Network Address Translators



Operation of NA(P)Ts

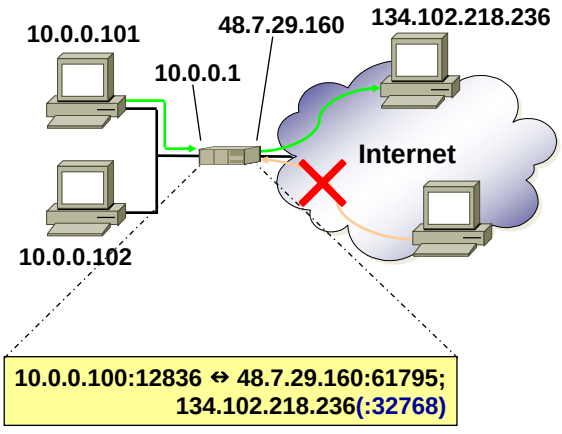
- NATs usually only one-way permeable for initiating connections
  - From private to public network
  - Other direction limited to statically pre-configured addresses
- NATs create address/port number mappings
  - Mappings are usually created dynamically, e.g. on connection setup
  - Static configurations 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



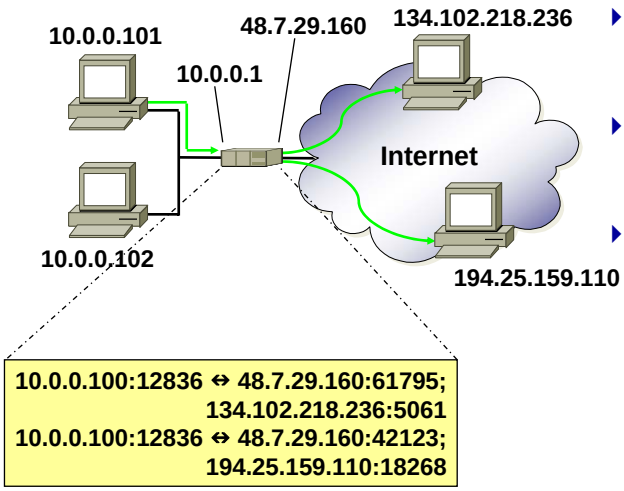
- Outbound packets establish temporary address/port binding
- Any host may respond to mapped address/port
- Incoming packets are dropped when no binding exists

(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
- Applications use well-defined ports
- Communications are usually invoked from the inside
- Communications from the outside limited to a few servers
  - Often placed in a DMZ
- Connection-oriented protocols (e.g. TCP) dominate
  - Beginning and end of communication session can be identified

Some Issues with Firewalls and NATs

- 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 in a middlebox
- Dynamically negotiated addresses
- Symmetric communication relationships
- (Invocation of) communications from unknown peers
- …

Pretty much all of this applies to SIP!

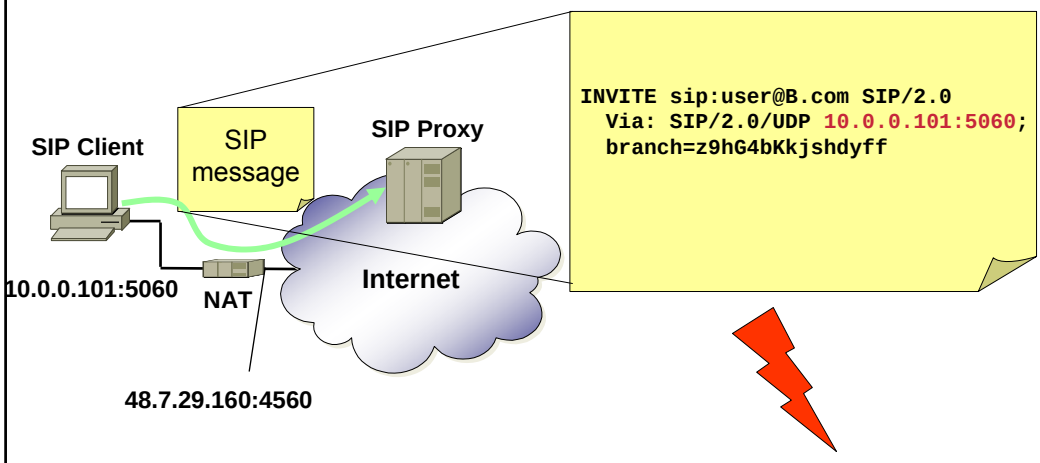
NAT & SIP: Problems

- Signaling-specific
  - SIP/UDP through NATs
  - SIP message routing
  - SIP registration
- Media session-specific
  - SDP description (endpoint parameters)

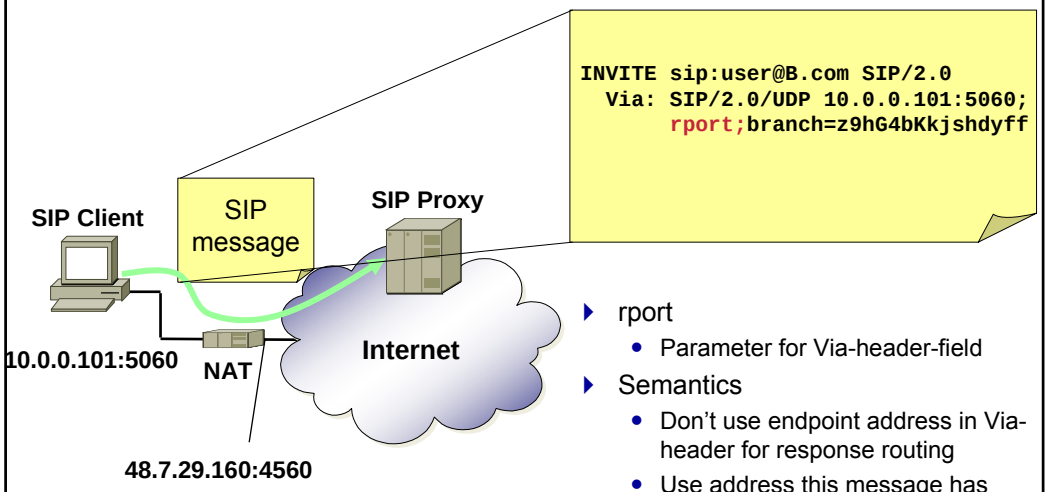
NAT & SIP: Signaling Problems

- Static ports for signaling traffic (typically 5060, 5061)
  - Static configuration for port forwarding (if NAT box supports this)
  - Need multiplexing in subnets with multiple SIP clients (alternative: local SIP proxy)
- Response routing
  - RFC3261: Reuse connection for stream-oriented protocols (TCP, SCTP)
  - UDP requires overriding of entries in Via headers to honor NAT binding
  - Need frequent keep-alive messages for INVITE transactions
- Registration
  - No good solution yet
  - Clients must construct “working” Contact URI
  - Works if registrar and proxy are co-located

Response Routing and NATs

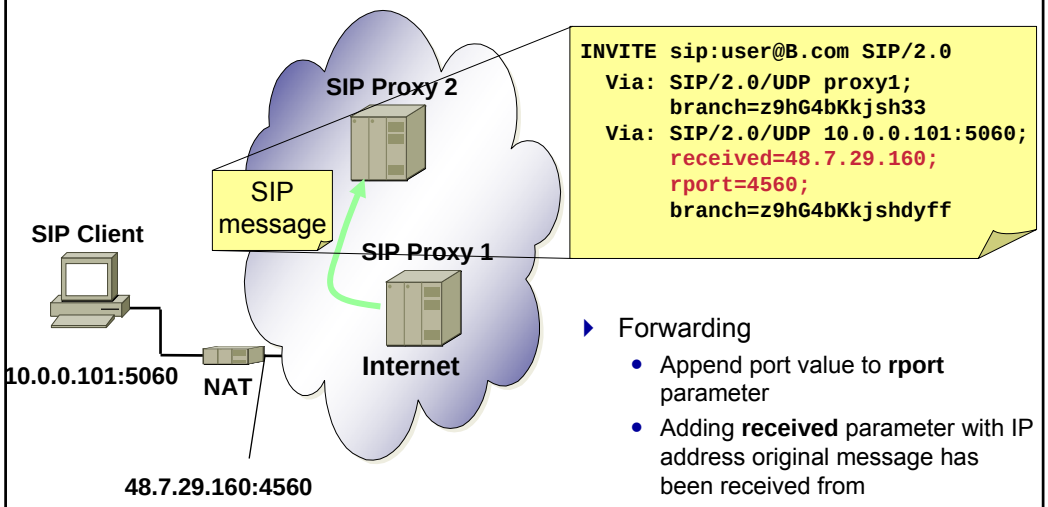


SIP Extension for Symmetric Response Routing



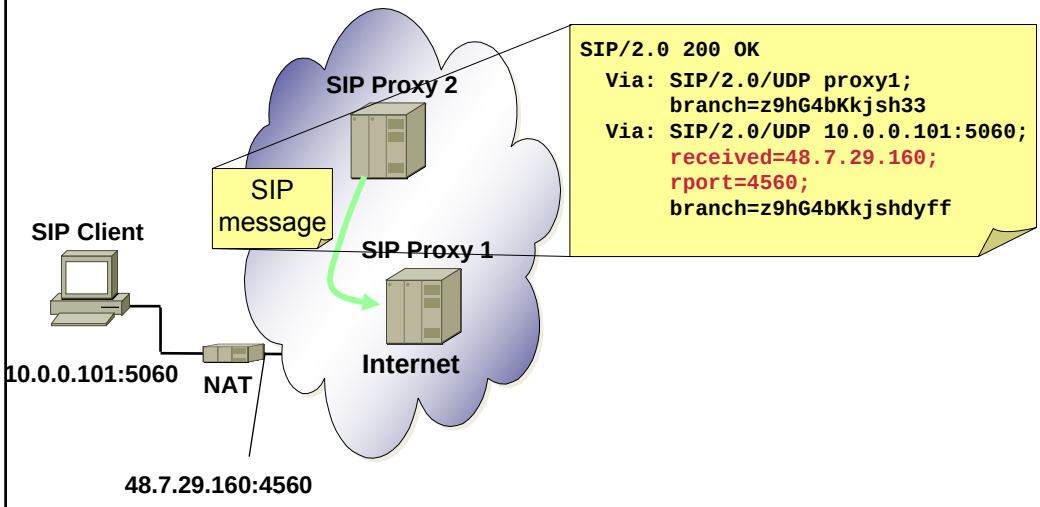
- rport
  - Parameter for Via-header-field
- Semantics
  - Don't use endpoint address in Via-header for response routing
  - Use address this message has been received from

Symmetric Response Routing: Server Behavior

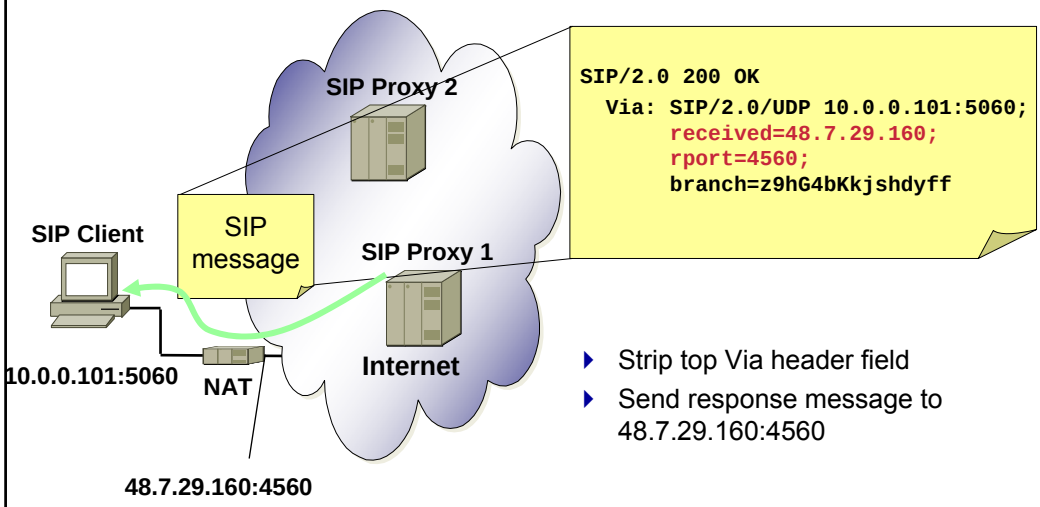


- Forwarding
  - Append port value to **rport** parameter
  - Adding **received** parameter with IP address original message has been received from

Processing Response Messages (1)



Processing Response Messages (2)



- Strip top Via header field
- Send response message to 48.7.29.160:4560

SIP Registration and NATs

- SIP user agent registers contact address (current reachable endpoint address) with registrar
- Problem in the presence of NATs
  - How to determine valid contact address?
- No standard solution today, but multiple approaches
  - Send message to server and evaluate `received` and `rport` parameters
  - Use non-SIP servers for self-address fixing (STUN)
- Issue: Can only receive messages from server the REGISTER request has been sent to (depending on NAT type)
  - All messages to UA must traverse this proxy
  - Solution: `Path:` extension header
    - Discover and record sequence of proxies that must be used for contacting UA
    - Use of pre-loaded Route mechanism

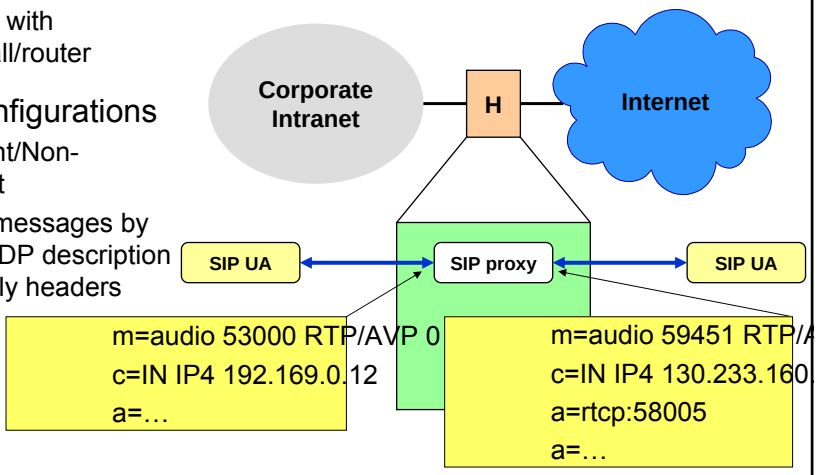
Solutions (for Media Session Startup)

- [Application Layer Gateways]
- Middlebox Communications (MIDCOM)
- Simple Traversal of UDP through NATs (STUN*)
- Travel Using Relay NAT (TURN*)
- Interactive Connectivity Establishment (ICE*)

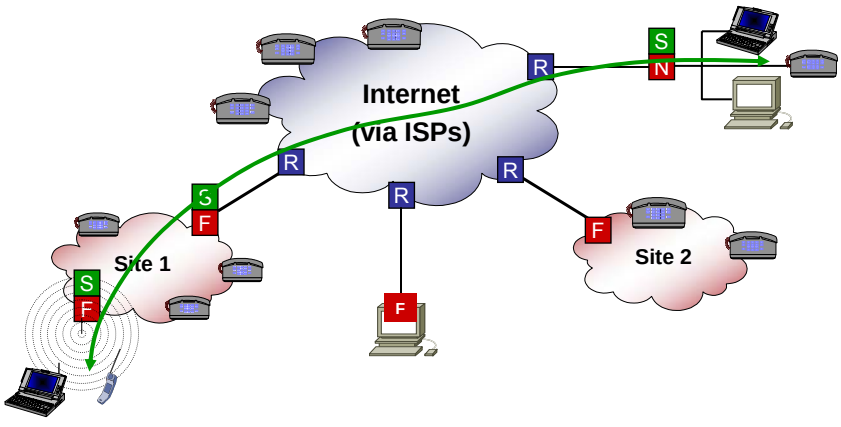
*) Unilateral Self-address fixing (UNSAF) considerations (RFC 3424)

SIP Application Layer Gateway (1)

- Gateway system with application intelligence
  - Co-located with NAT/firewall/router
- Different configurations
  - Transparent/Non-transparent
  - Fixes SIP messages by adapting SDP description and possibly headers



SIP and NATs / Firewalls



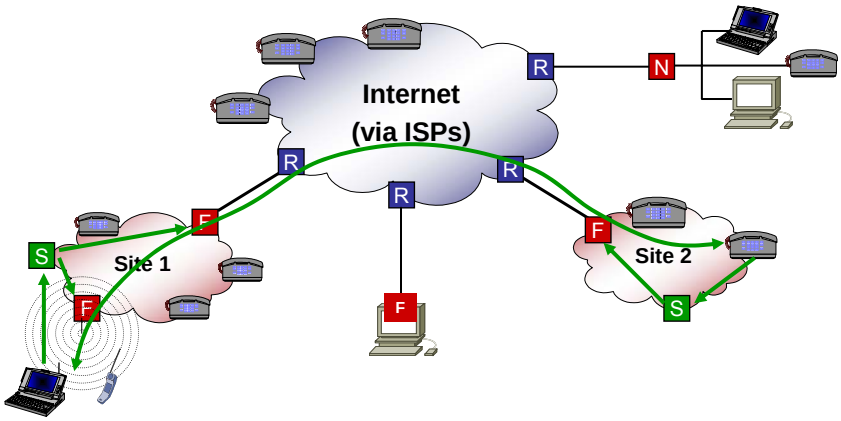
SIP Application Layer Gateway (2)

- Issues
  - SIP proxies not allowed to change session description
  - Conflicts with security, i.e., SIP and S/MIME
  - 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)
  - Scalability
    - Functionality concentrated on single NAT/ALG box
  - Reliability
    - Rewriting of SIP messages not robust with respect to extensions, future protocol versions etc.

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)

MIDCOM

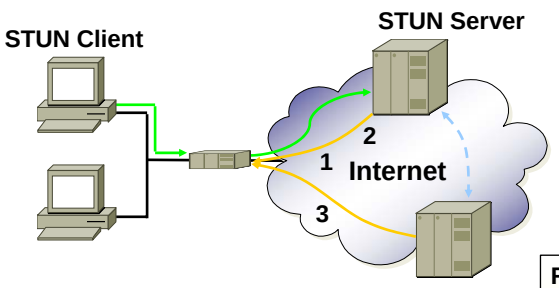


MIDCOM Issues

- 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...

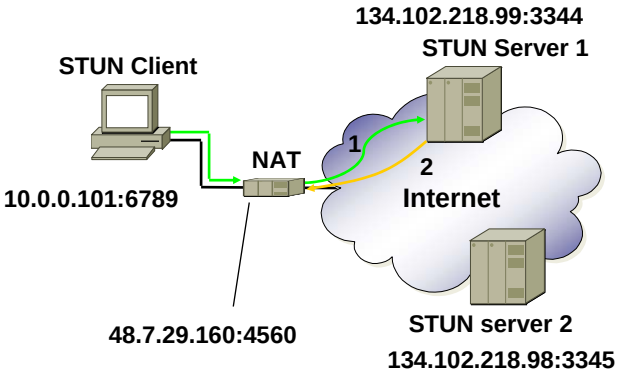


1. **Echo source address, send from recv port**
2. **Client requested port change**
3. **Client requested address change**

Received/dropped responses determine type of NAT

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

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

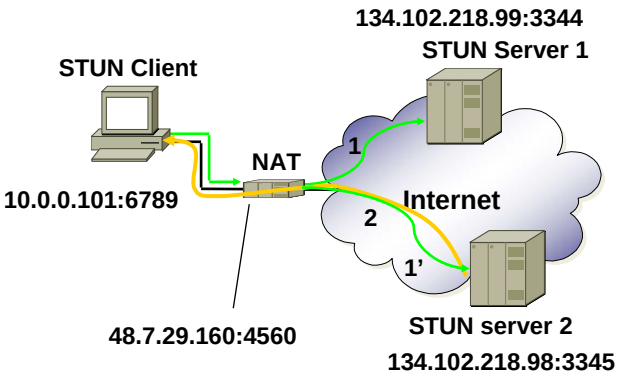


(may be the same physical machine and just use a different IP address/port)

- 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)

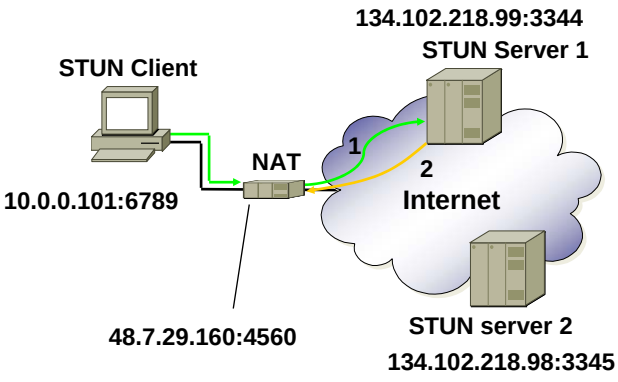
2. Binding request to STUN server 1 with change IP and change port flags



- 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

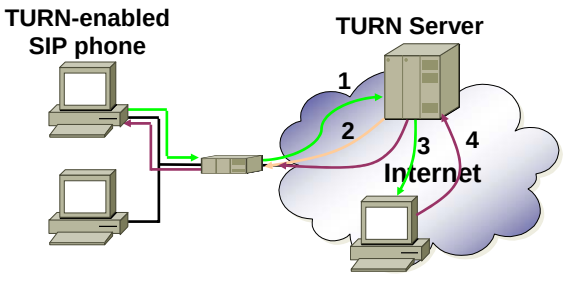
- Simple Traversal Underneath Network Address Translators (STUN)
  - draft-ietf-behave-rfc3489bis-05.txt
- Removes attempt to understand and identify NAT types
  - Full cone, (port) restricted cone, and symmetric are only a rough classification
  - Symmetric is the most important
    - Existence determined 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)

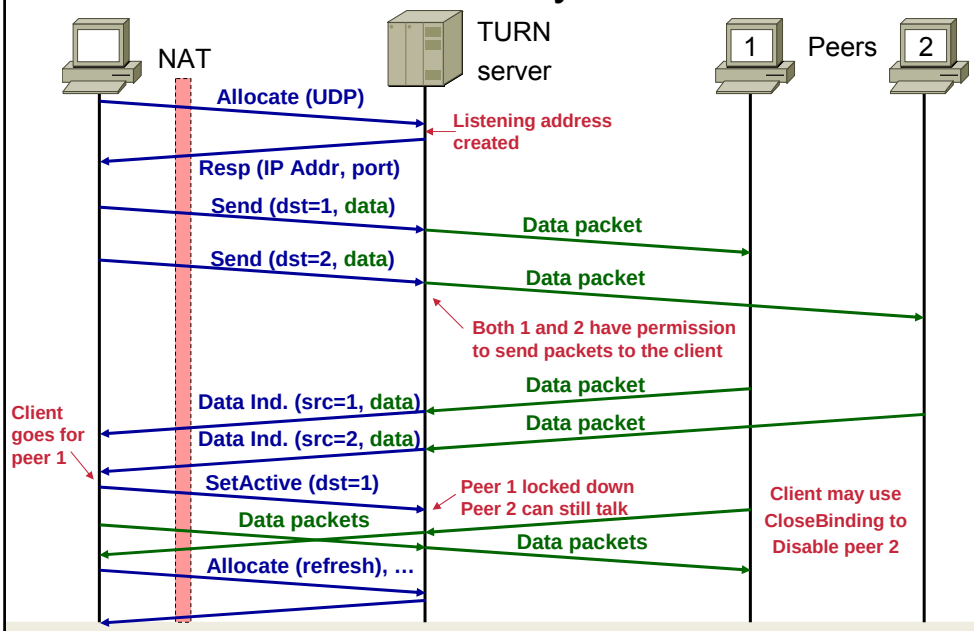
- Idea: Provide forwarding service in the public internet
  - Client behind NAT has single connection to TURN server
  - Server forwards incoming packets destined for TURN client
    → *Relay NAT*
  - Protocol-agnostic – no ALG needed
  - Authentication to prevent DoS-attacks (similar to STUN)



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-19.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

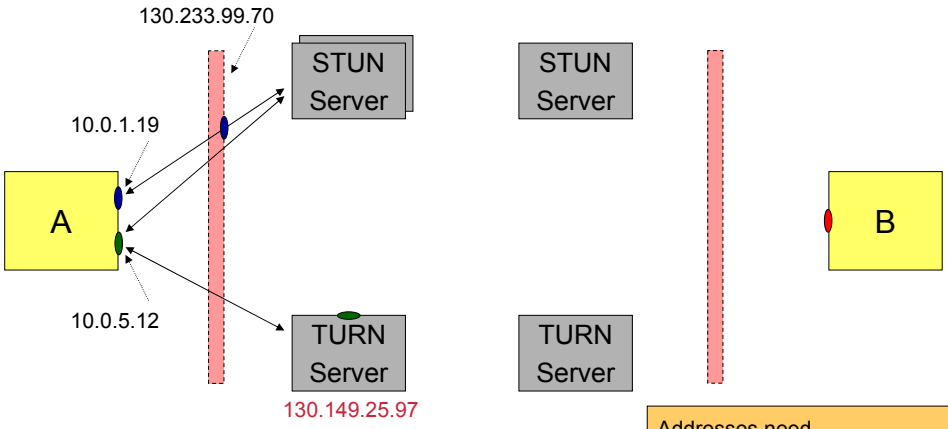
- Idea: peers exchange lists of transport addresses, mutual connectivity tests
- Clients must detect own transport addresses
  - The more, the better
  - Local interfaces (including private addresses, e.g. in 10/8 net)
  - Detection using “external” reflectors (e.g. STUN, TURN)
  - Assigned tunnel addresses (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

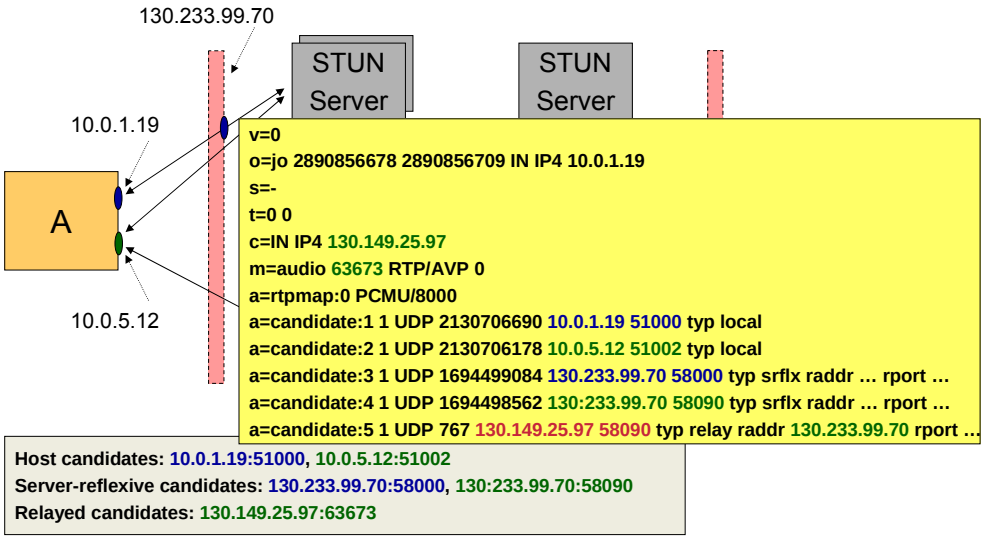
Addresses need
- per media stream (audio, video)
- per component (RTP, RTCP)

Address Gathering and Prioritization

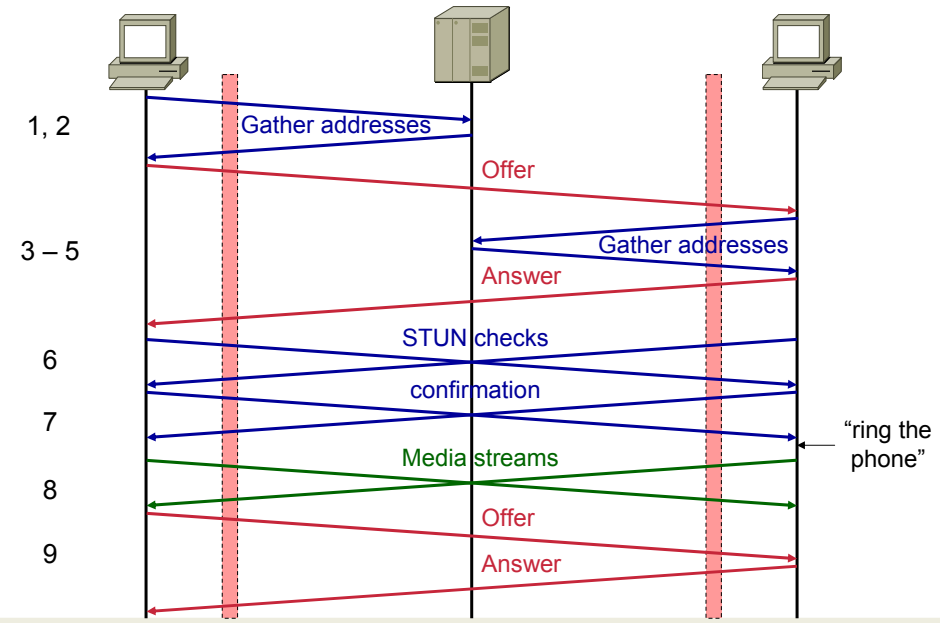
- Address gathering can cause significant traffic
  - Multiple interfaces, IP address versions, STUN servers
  - Multiple media streams and components per stream
  - May cause network or NAT overload
- Pace transmission (20ms intervals)
- Prioritization across candidates:
  - Reflect the quality (e.g., 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)
```

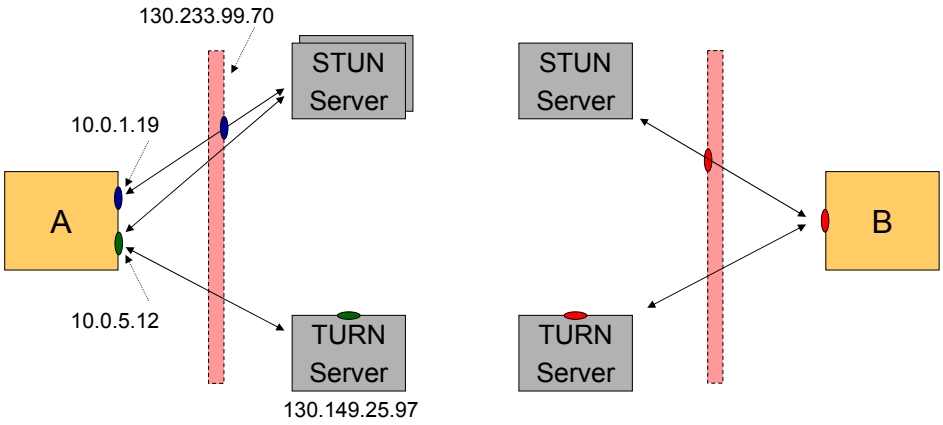

ICE: Address Gathering and Encoding



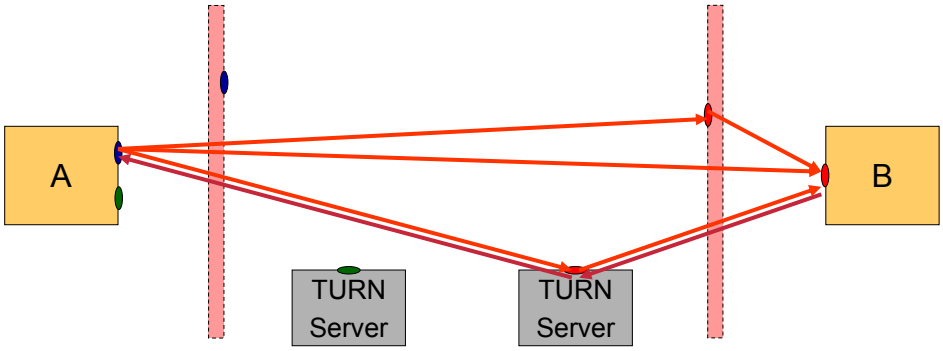
ICE Operation



6. ICE Address Verification

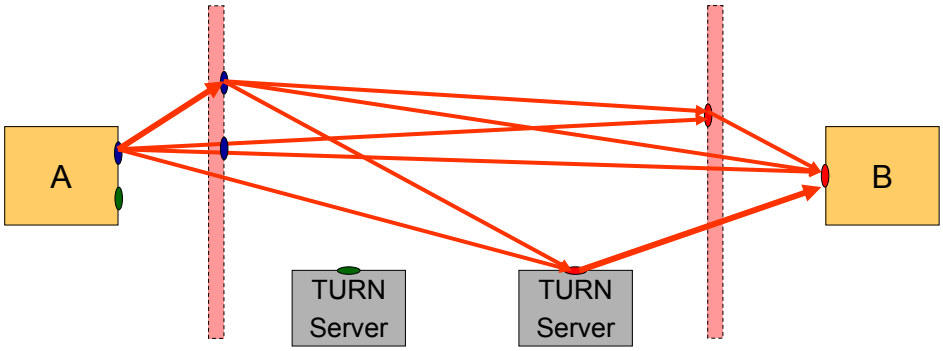


6. ICE Address Verification



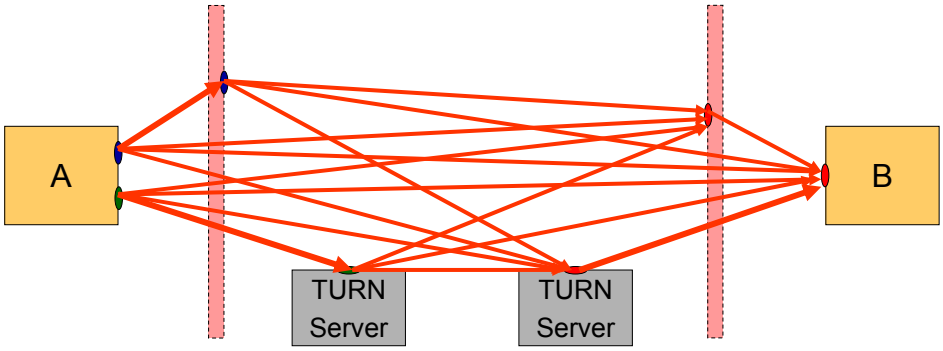
- Send test messages and await replies: from all candidates to all candidates
- Successively go (again paced) through all address pairs
- Use priority ordering (pairing) to determine the trial order
  (shall ensure that better ones are tested first)

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)

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

Summary: SIP and Firewalls / NATs

- Do not go together well…
  - SIP servers for enterprises may be reachable for SIP signaling
  - NAT-based address translation invalidates SIP message contents
  - Firewalls do not let voice packets pass
- Problem not restricted to SIP
  - RTSP to signal media streams, multicast media streams, etc.
  - New application protocols (yet) unknown to NATs
  - Guidelines for application protocol design for NATs: RFC 3235
- Frequent “fallback” position: tunneling through HTTP (port 80)
- ALGs for controlled environments
- ICE: one solution set preserving end-to-end model