NAT Tutorial

Dan Wing, dwing@cisco.com

IETF78, Maastricht

July 25, 2010

v3

Agenda

- NAT and NAPT
  - Types of NATs
- Application Impact
  - Application Layer Gateway (ALG)
  - STUN, ICE, TURN
- Large-Scale NATs (LSN, CGN)
- IPv6/IPv4 Translation (“NAT64”)
- NAT66

Agenda

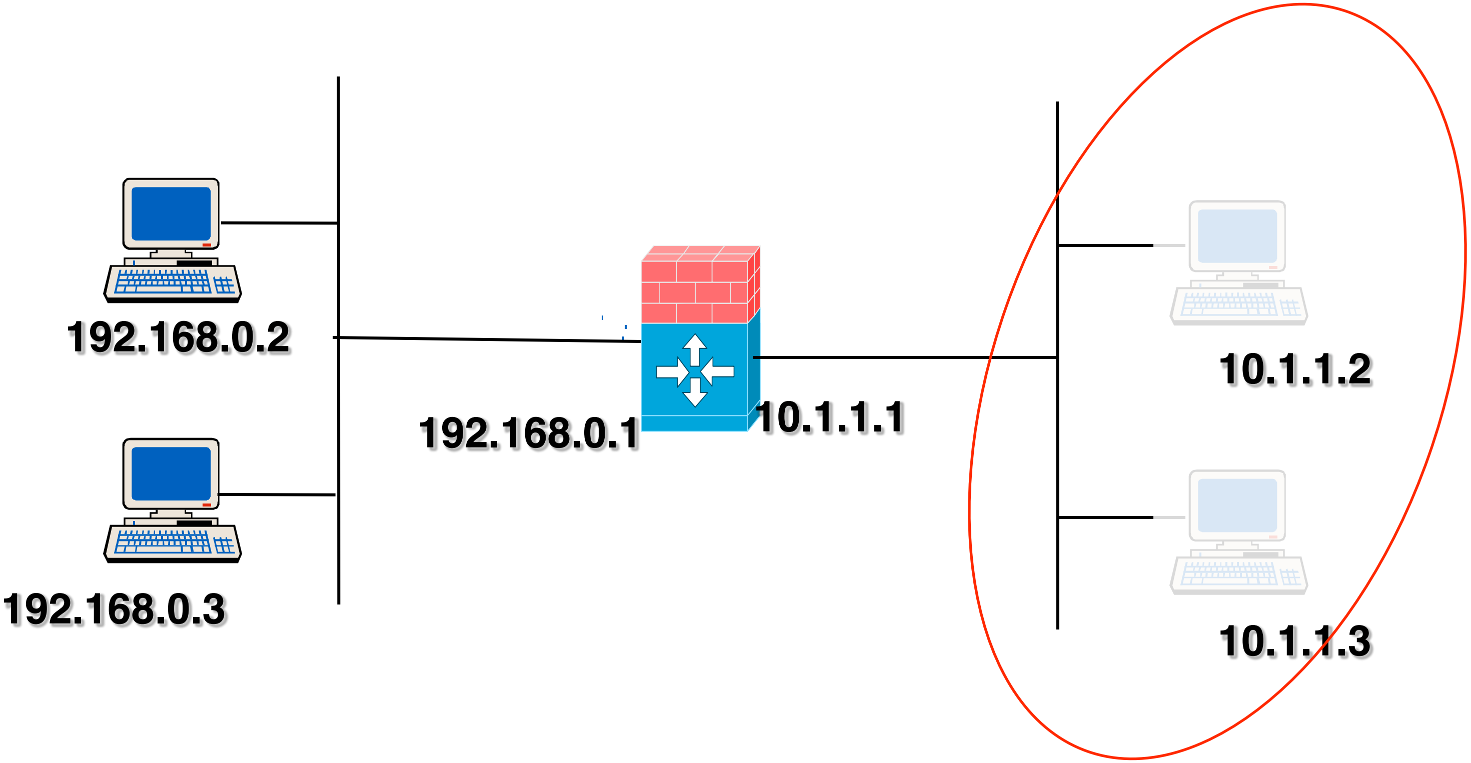
- NAT and NAPT
  - Types of NATs
- Application Impact
  - Application Layer Gateway (ALG)
  - STUN, ICE, TURN
- Large-Scale NATs (LSN, CGN, SP NAT)
- IPv6/IPv4 Translation (“NAT64”)
- NAT66

NAT

- First described in 1991
- 1:1 translation
  - Does not conserve IPv4 addresses
- Per-flow stateless
- Today’s primary use is inside of enterprise networks
  - Connect overlapping RFC1918 address space

NAT Diagram

- Hosts seem to have multiple IPv4 addresses – almost like "ghosts"

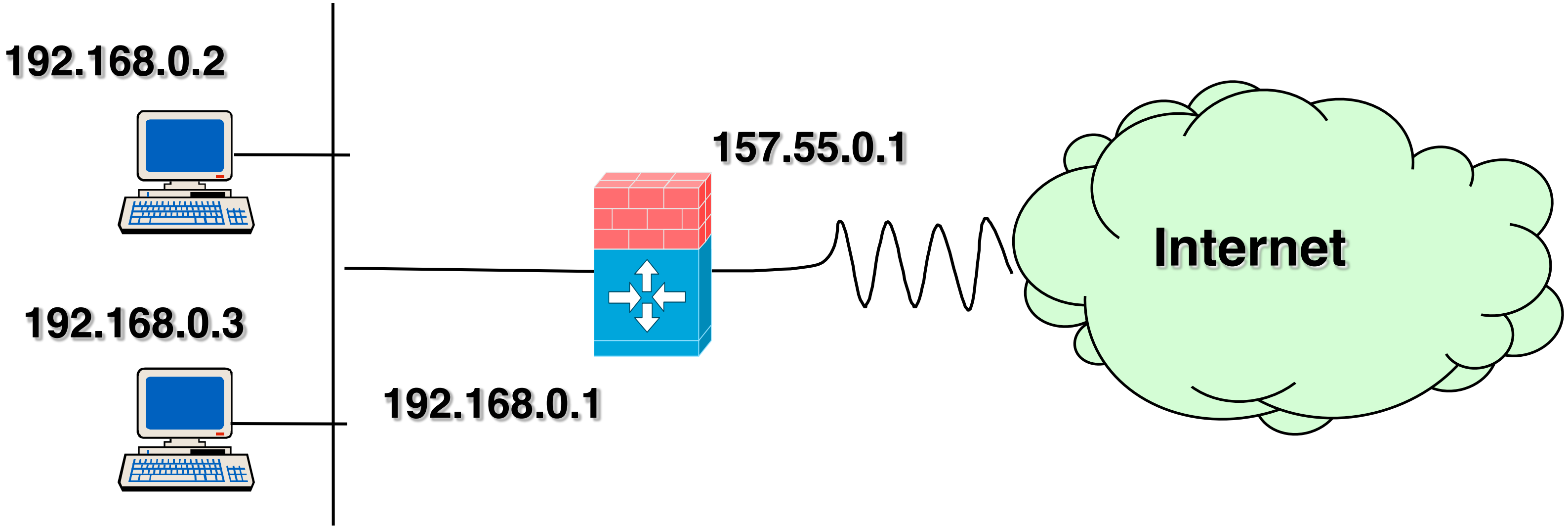


NAPT

- Described in 2001 (RFC3022)
- 1:N translation
  - Conserves IPv4 addresses
  - Allows multiple hosts to share one IPv4 address
  - Only TCP, UDP, and ICMP
  - Connection has to be initiated from ‘inside’
- Per-flow stateful
- Commonly used in home gateways and enterprise NAT

NAPT Diagram

- Hosts share an IPv4 address



NAPT complications

- NAPT requires connections initiated from ‘inside’
- Creates state in the network (in the NAPT)
  - This is bad
  - NAPT crashes -> connections break
- When to discard state?
  - TCP RST? Spoofed RSTs?
  - Timeout?

Terminology

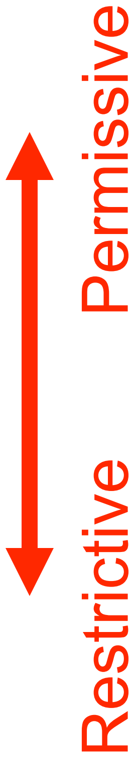
- “NAT” is spoken/written instead of “NAPT”
  - Even though NAPT is often more accurate
  - The more accurate “PAT” never caught on

- So, it’s “NAT”

- Now, often called “NAT44” to differentiate from NAT64 and NAT46

Types of NAT (old terms)

- Full Cone
- Restricted Cone
- Port Restricted Cone
- Symmetric



Permissive

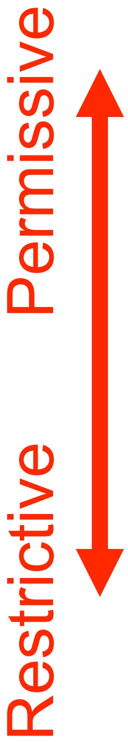
Restrictive

RFC3489 (obsoleted)

Types of NAT (new terms)

Mapping

- Endpoint-Independent
- Address-Dependent
- Address and Port-Dependent



Permissive

Restrictive

Filtering

- Endpoint-Independent
- Address-Dependent
- Address and Port-Dependent

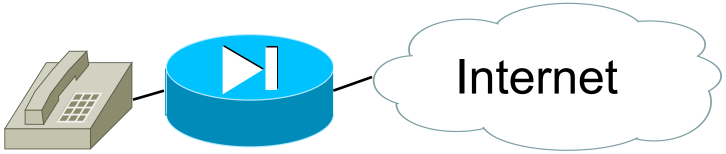
RFC4787

Agenda

- NAT and NAPT
  - Types of NATs
- Application Impact
  - Application Layer Gateway (ALG)
  - STUN, ICE, TURN
- Large-Scale NATs (LSN, CGN, SP NAT)
- IPv6/IPv4 Translation (“NAT64”)
- NAT66

NAT Philosophy

- “Be transparent”
- This means NATs are not proxies
  - Applications are generally unaware of a NAT
- Problem with IP addresses inside the application
  - Generally called a “referral”
  - Example: SIP



“my address is 10.1.1.1”

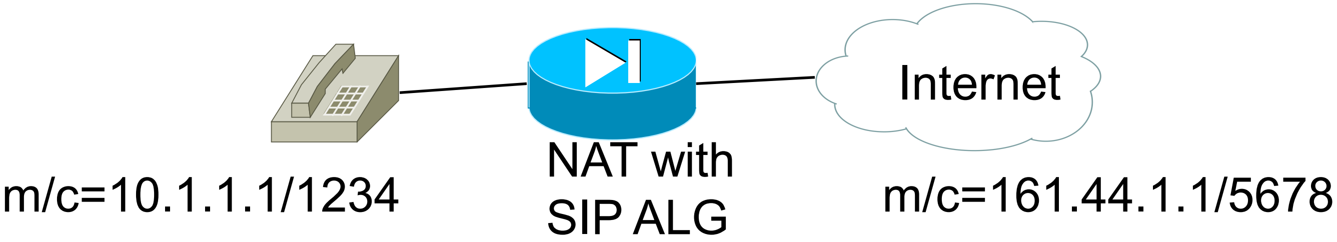
Internet sees 161.44.1.1

NAPT and servers

- NAPT: connection initiated from inside
- Incoming connections are difficult
- Significant problem for servers
  - Webcam, VoIP, RTSP receivers, etc.
- Port forwarding (“pinholing”, etc.)
  - web or CLI configuration
  - UPnP IGD, NAT-PMP
  - All have drawbacks

Application Layer Gateway (ALG)

- Application awareness inside the NAT
- ALG modifies IP addresses and ports in application payload, and creates NAT mapping
- Each application requires a separate ALG
  - FTP, SIP, RTSP, RealAudio, …



Problems with ALGs

- Requires ALG for each application
- Requires ALG that understands *this particular* application’s nuance
  - Proprietary extensions / deviations
  - New standard extensions
- ALG requires:
  - Un-encrypted signaling (!)
  - Seeing application’s signaling and media/data
    - easy with stub network; harder with mesh network

Application Solutions

- Applications cannot successfully rely on ALGs
- So, Applications have developed their own solutions
- FTP PASV
  - Data connection always to server. Has security side-effects.
- RTSP supports ‘interleaved data’ (RFC2326)
  - Streaming over RTSP’s TCP control channel
- RTSPv2 with ICE-like NAT traversal
- HTTP delivery
  - Flash (e.g., YouTube)
- ICE, STUN, TURN
  - Intelligence in endpoint
  - Useful for offer/answer protocols (SIP, XMPP, probably more)
  - Standardized in MMUSIC and BEHAVE
  - (more on next slides)

STUN, ICE, TURN

- Request/response protocol, used by:
  - STUN itself (to learn public IP address)
  - ICE (for connectivity checks)
  - TURN (to configure TURN server)
- The response contains IP address and port of request
  - Runs over UDP (typical) or TCP, port 3478
- Somewhat like http://whatismyip.com

STUN, ICE, TURN

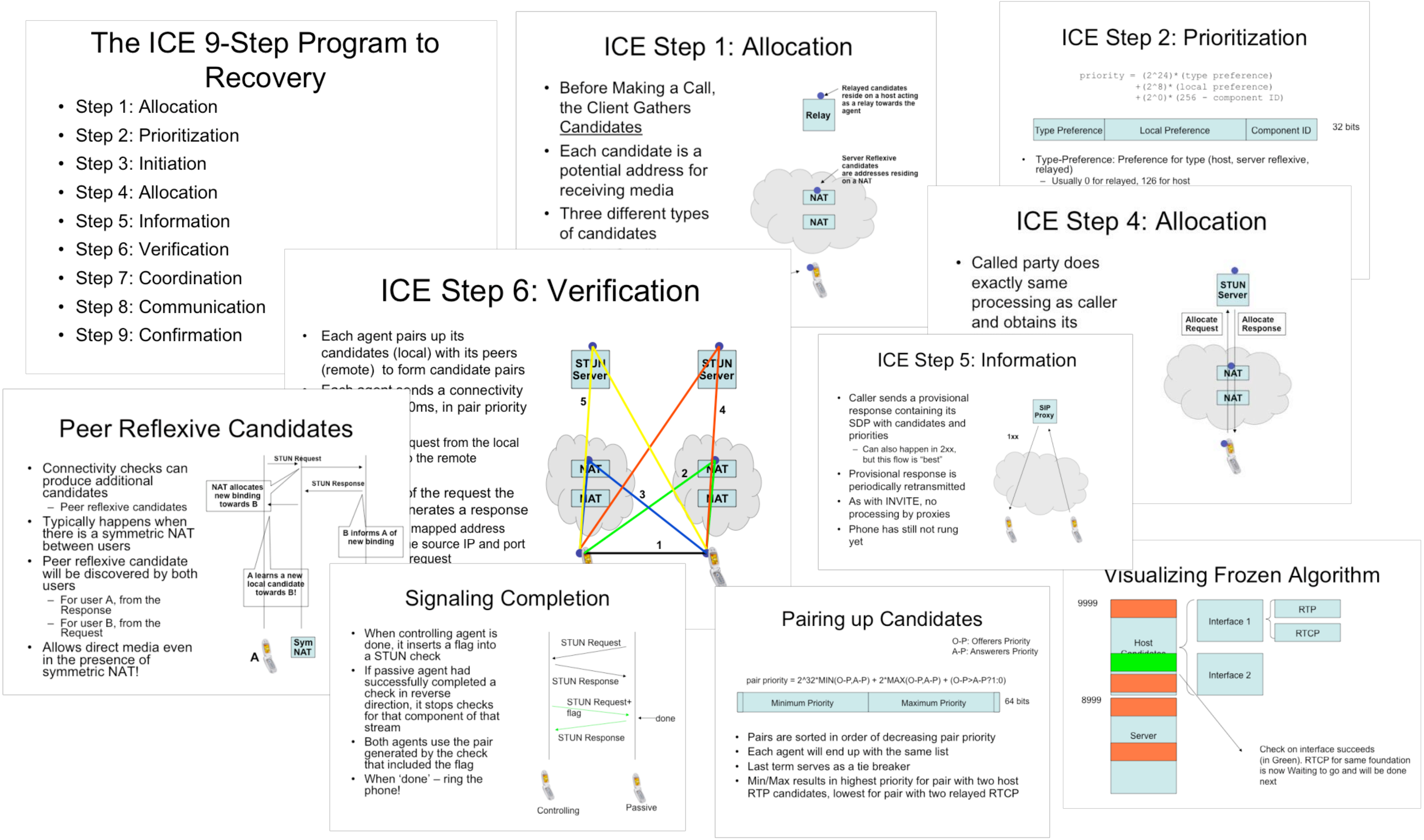
- Procedure for Optimizing Media Flows
- Defines SDP syntax to indicate ‘candidate addresses’
- Uses STUN messages for connectivity checks
  - Sent to RTP peer, using **same ports** as RTP
- First best path wins

- Think: gather all my IP addresses, send them to my peer, and do connectivity checks

STUN, ICE, TURN

- Media Relay Protocol and Media Relay Server
- Only used when:
  - **both** endpoints are behind ‘Address and Port-Dependent Filtering’ NATs (rare, about 25% of NATs), or
  - one endpoint doesn’t implement ICE, and is behind a ‘Address and Port-Dependent Filtering’ NAT

ICE: 119 Pages



The ICE 9-Step Program to Recovery

- 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 Step 1: Allocation

- Before Making a Call, the Client Gathers Candidates
- Each candidate is a potential address for receiving media
- Three different types of candidates

Relay
Relayed candidates reside on a host acting as a relay towards the agent
Server Reflexive candidates are addresses residing on a NAT
NAT
NAT

ICE Step 2: Prioritization

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

- Type-Preference: Preference for type (host, server reflexive, relayed)
  - Usually 0 for relayed, 126 for host

ICE Step 4: Allocation

- Called party does exactly same processing as caller and obtains its

ICE Step 6: Verification

- Each agent pairs up its candidates (local) with its peers (remote) to form candidate pairs
- Each agent sends a connectivity
0ms, in pair priority
quest from the local
the remote
of the request the
nerates a response
mapped address
e source IP and port
request

ICE Step 5: Information

- Caller sends a provisional response containing its SDP with candidates and priorities
  - Can also happen in 2xx, but this flow is "best"
- Provisional response is periodically retransmitted
- As with INVITE, no processing by proxies
- Phone has still not rung yet

Peer Reflexive Candidates

- Connectivity checks can produce additional candidates
  - Peer reflexive candidates
- Typically happens when there is a symmetric NAT between users
- Peer reflexive candidate will be discovered by both users
  - For user A, from the Response
  - For user B, from the Request
- Allows direct media even in the presence of symmetric NAT!

STUN Request
NAT allocates new binding towards B
STUN Response
B informs A of new binding
A learns a new local candidate towards B!
A
Sym NAT

Signaling Completion

- When controlling agent is done, it inserts a flag into a STUN check
- If passive agent had successfully completed a check in reverse direction, it stops checks for that component of that stream
- Both agents use the pair generated by the check that included the flag
- When 'done' – ring the phone!

Pairing up Candidates

O-P: Offerers Priority
A-P: Answerers Priority

pair priority = 2^32*MIN(O-P,A-P) + 2*MAX(O-P,A-P) + (O-P>A-P?1:0)

- Pairs are sorted in order of decreasing pair priority
- Each agent will end up with the same list
- Last term serves as a tie breaker
- Min/Max results in highest priority for pair with two host RTP candidates, lowest for pair with two relayed RTCP

Visualizing Frozen Algorithm

Check on interface succeeds (in Green). RTCP for same foundation is now Waiting to go and will be done next

ICE Deployments

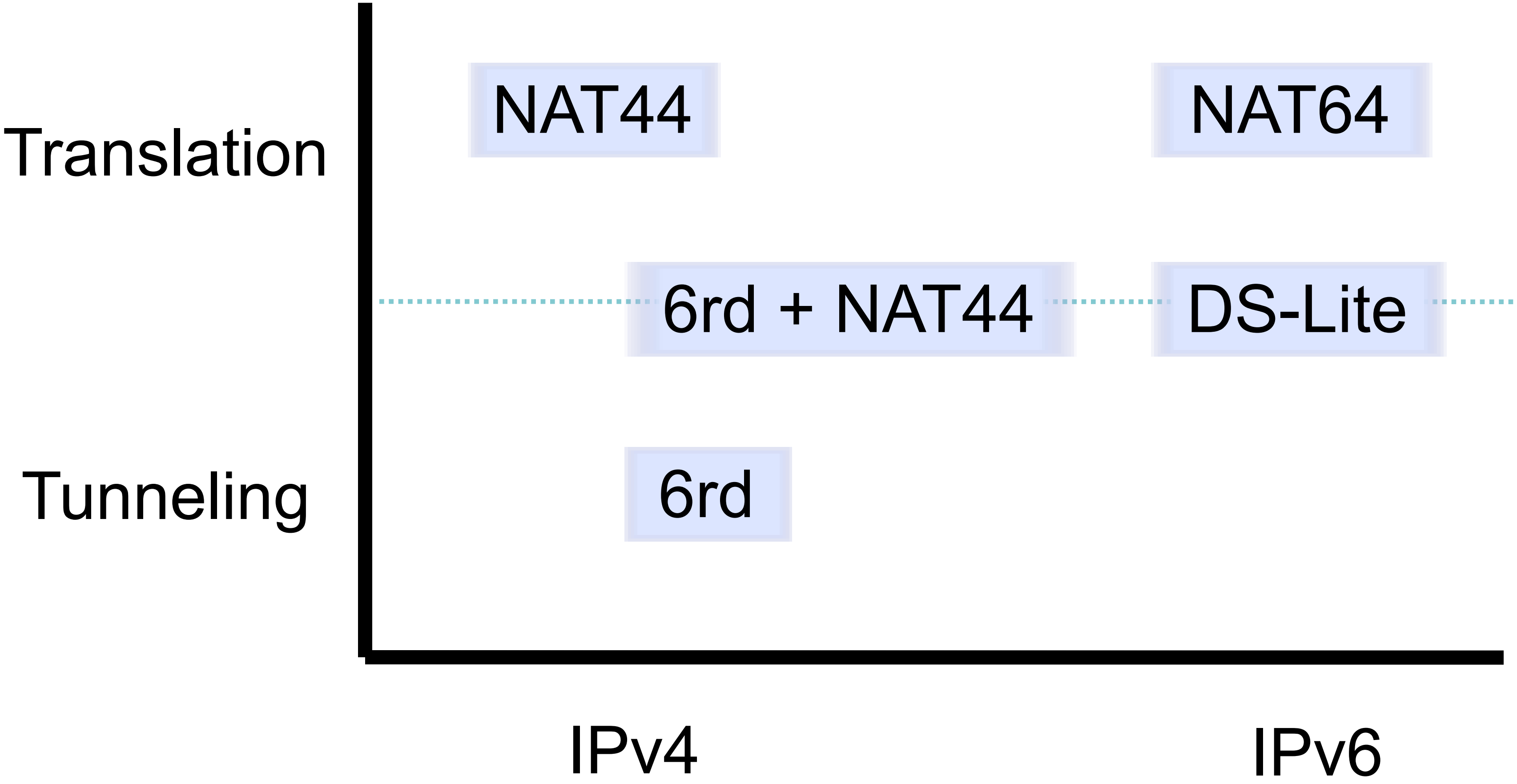
- Google chat (XMPP)
- Microsoft MSN chat
- Yahoo chat
- Counterpath softphone
- Apple Facetime

- Open source ICE libraries are available

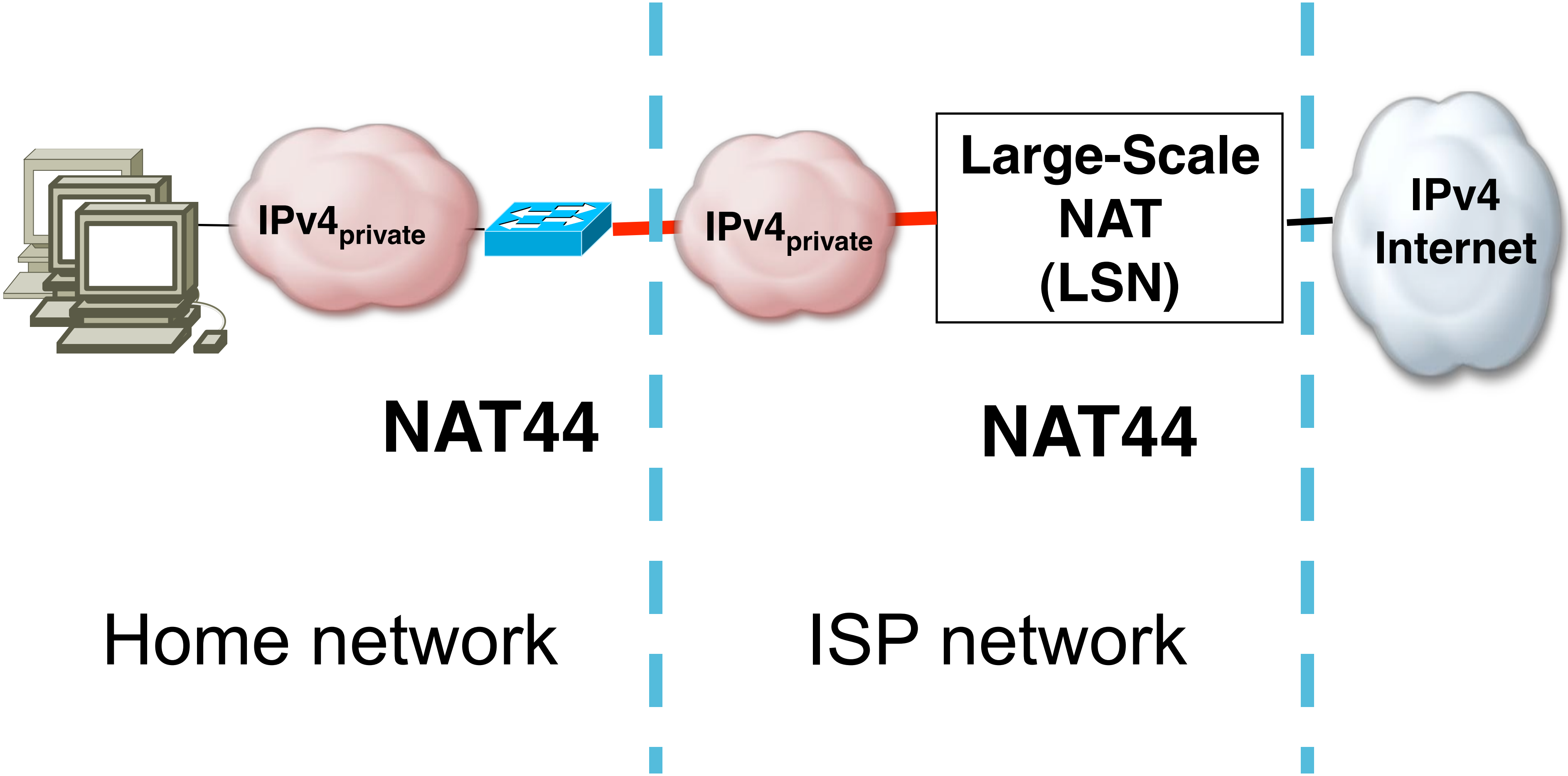
Agenda

- NAT and NAPT
  - Types of NATs
- Application Impact
  - Application Layer Gateway (ALG)
  - STUN, ICE, TURN
- Large-Scale NATs (LSN, CGN, SP NAT)
- IPv6/IPv4 Translation (“NAT64”)
- NAT66

How It Fits Together



NAT44 + NAT44 = “NAT444”



$\text{IPv4}_{\text{private}}$

$\text{IPv4}_{\text{private}}$

Large-Scale
NAT
(LSN)

IPv4
Internet

NAT44

NAT44

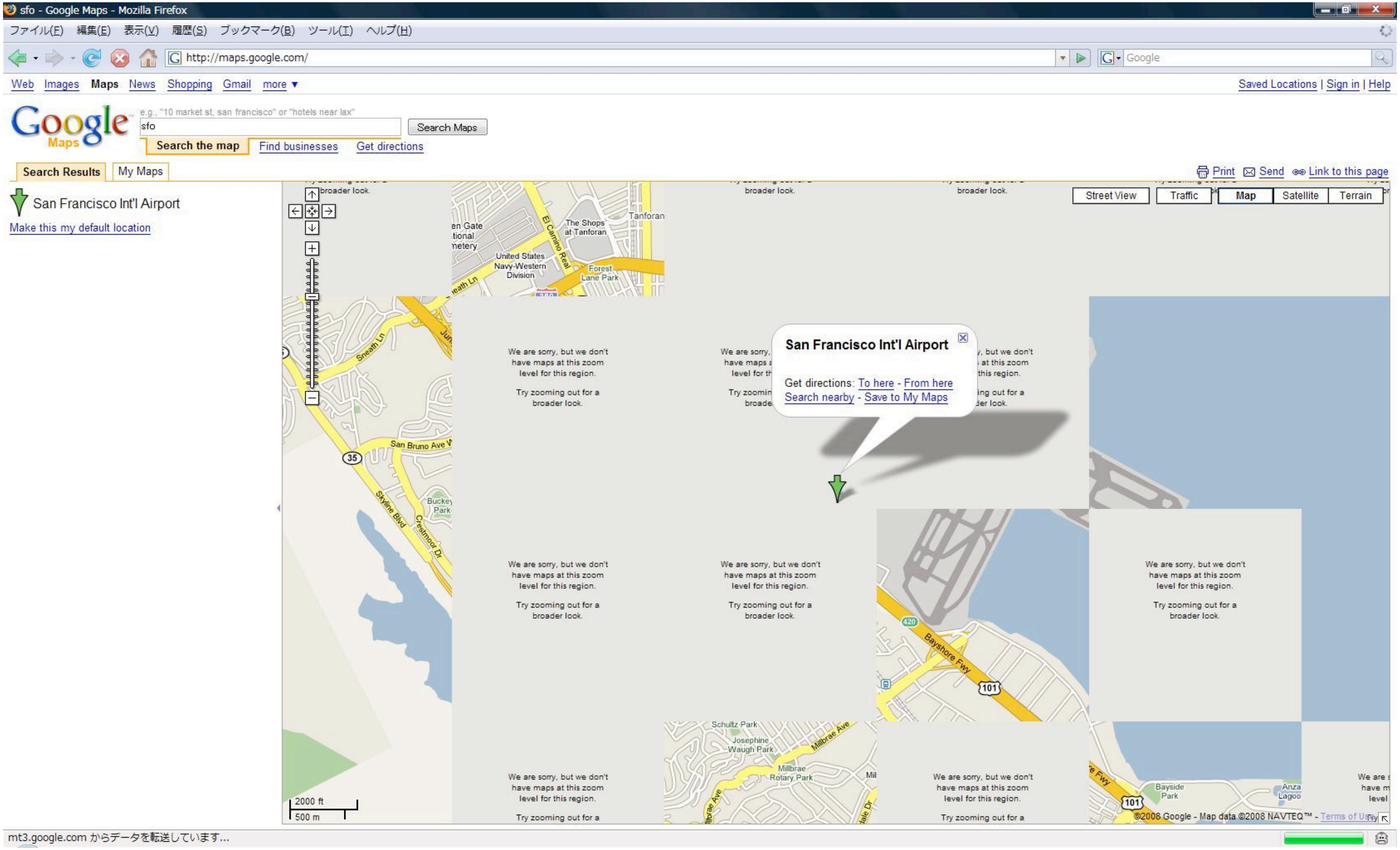
Home network

ISP network

Large Scale NAT (LSN)

- Essentially, just a big NAPT44
- Needs per-subscriber TCP/UDP port limits
  - Prevent DoS
  - If too low, can interfere with applications
    - Classic example: Google maps
- How to number network between subscriber and LSN?
  - RFC1918 conflicts with user’s space, breaks some NATs
  - Using routable IPv4 addresses is … wasteful

Insufficient Port Example



sfo - Google Maps - Mozilla Firefox

http://maps.google.com/

e.g., "10 market st, san francisco" or "hotels near lax"

sfo

San Francisco Int'l Airport

Make this my default location

We are sorry, but we don't have maps at this zoom level for this region.

Try zooming out for a broader look.

San Francisco Int'l Airport

Get directions: To here - From here
Search nearby - Save to My Maps

©2008 Google - Map data ©2008 NAVTEQ™ - Terms of Use

mt3.google.com からデータを転送しています...

Source: Shin Miyakawa, NTT Communications

LSN and ALG

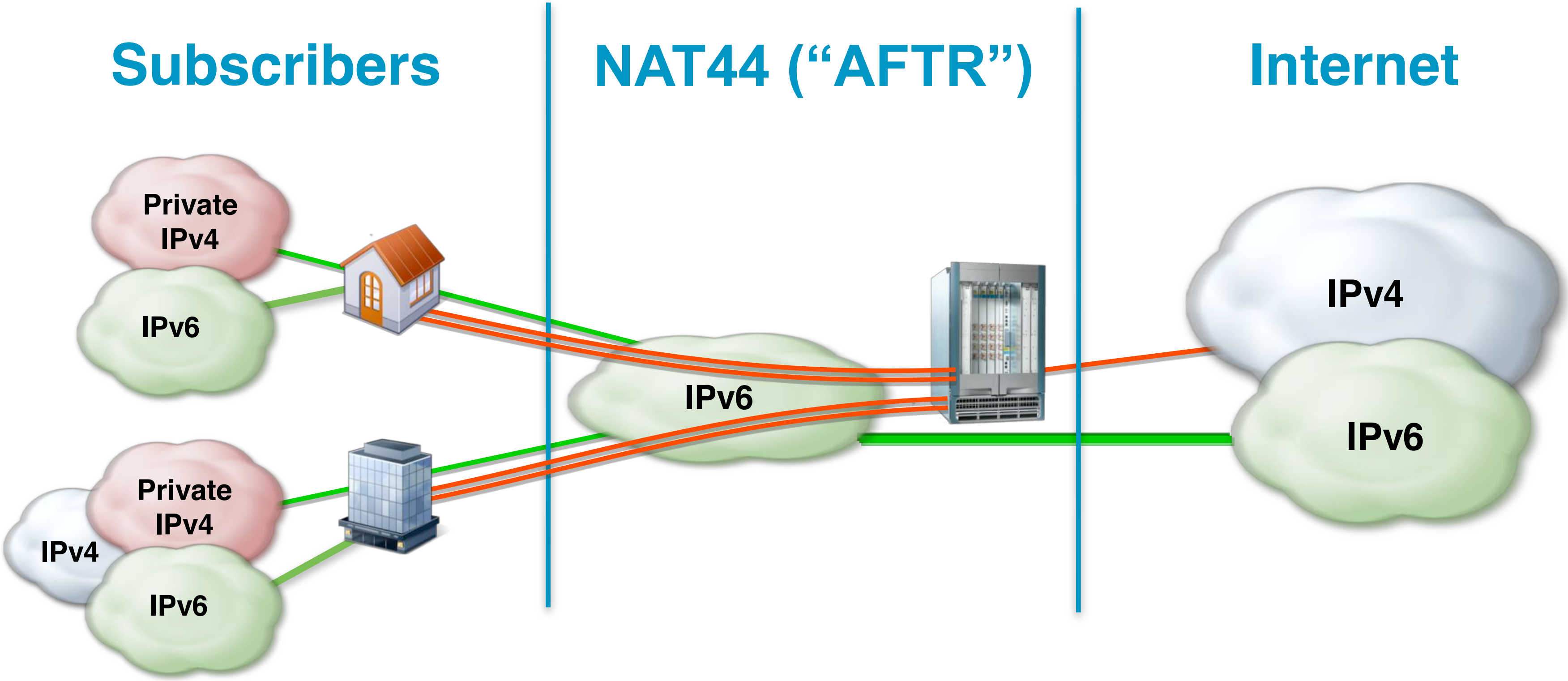
- Operationally complex in a LSN
  - Application X works but Application Y breaks. Upgrade ALG??
  - How long is vendor turn-around for patches?
- Interfering with competitor's over-the-top application (e.g., SIP, streaming video)

IPv4 Address Sharing

- Problem most noticed with LSN
- Reputation and abuse reporting are based on IPv4 address
  - Shared IP address = shared suffering
  - Law Enforcement
  - “Which subscriber posted on www.example.com at 8:23pm?”
  - Requires LSN log source port numbers
  - Requires web servers log source port numbers
- Everybody can’t get port 80
- Breaks geographic location (services and ads)

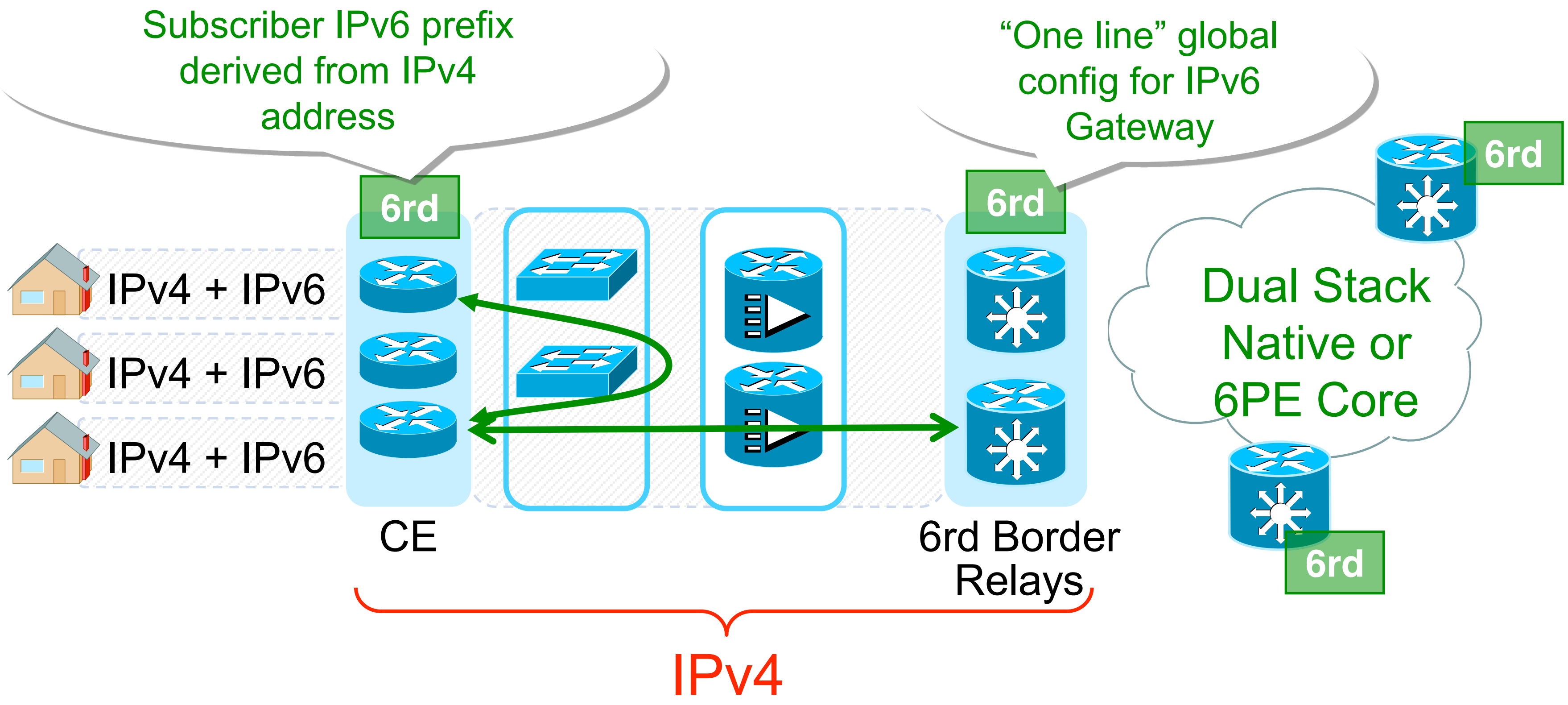
draft-ietf-intarea-shared-addressing-issues

Dual-Stack Lite: IPv4 over IPv6 Access



draft-ietf-softwire-dual-stack-lite

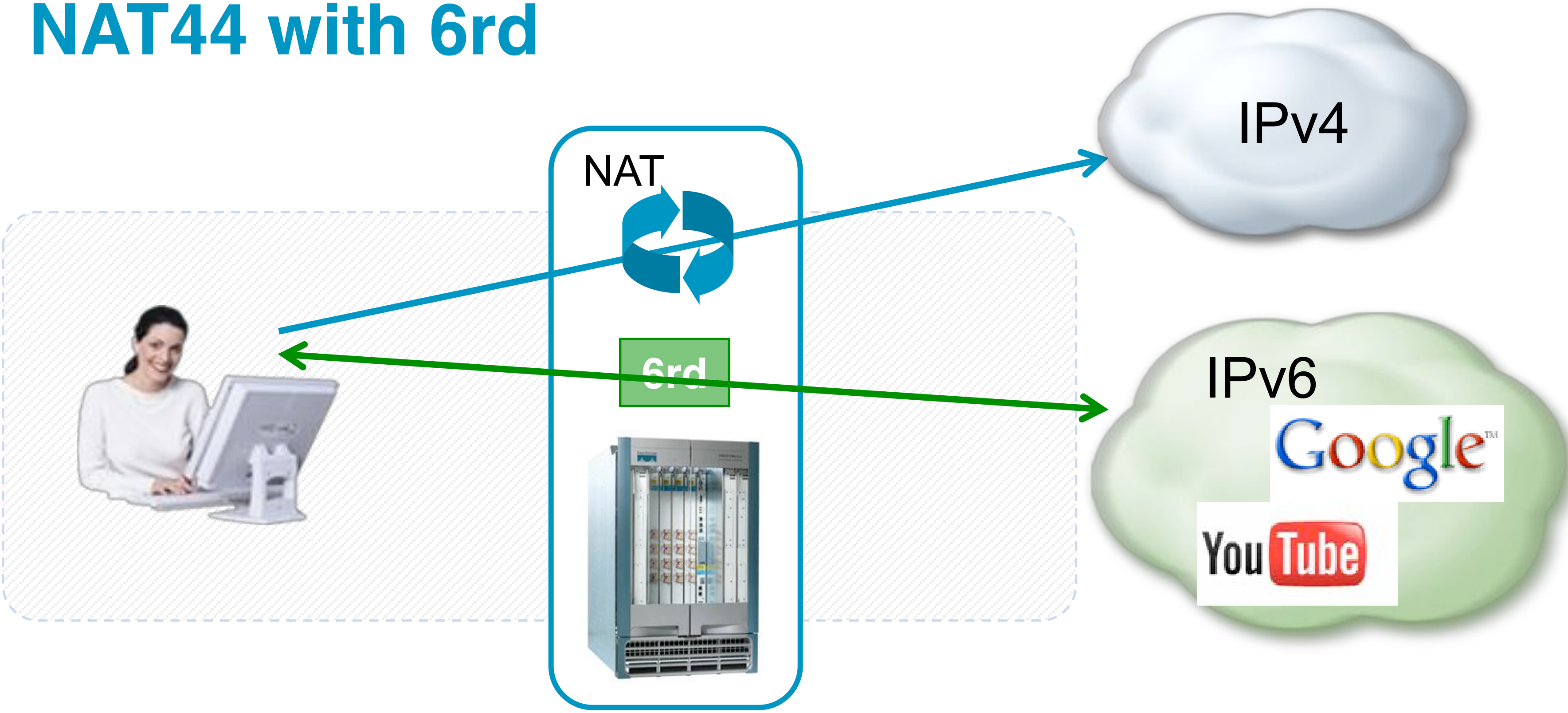
6rd in One Slide


Subscriber IPv6 prefix derived from IPv4 address
"One line" global config for IPv6 Gateway
6rd
6rd
6rd
6rd
IPv4 + IPv6
IPv4 + IPv6
IPv4 + IPv6
Dual Stack Native or 6PE Core
CE
6rd Border Relays
IPv4

- Native dual-stack IP service to the Subscriber
- Simple, stateless, automatic IPv6-in-IPv4 encapsulation and decapsulation
- IPv6 traffic automatically follows IPv4 Routing
- 6rd Border Relay placed at IPv6 edge

draft-ietf-softwire-ipv6-6rd

NAT44 with 6rd

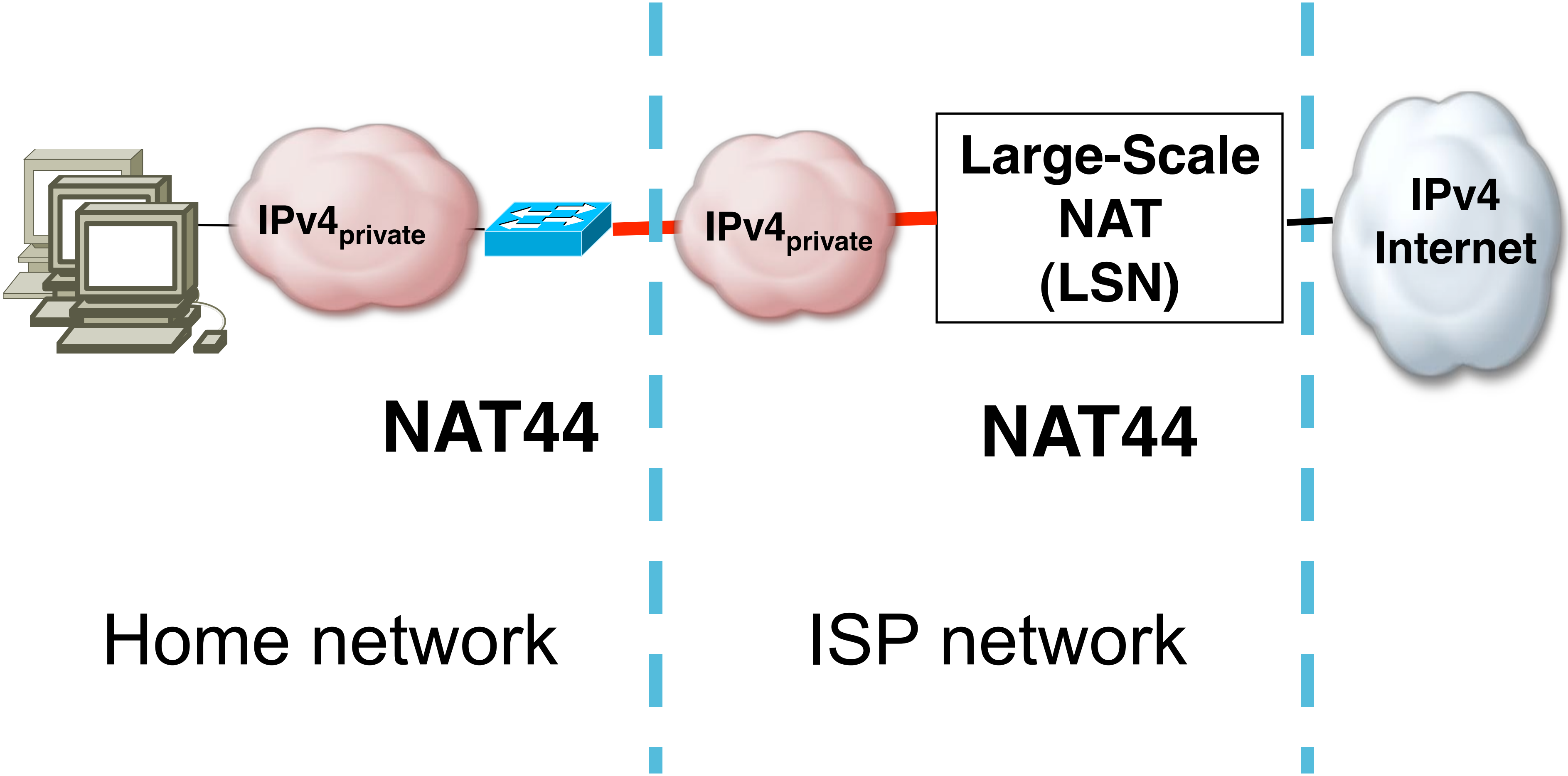


- **NAT44 works with 6rd**

    IPv6 content flows directly

    IPv6 content does **not** go through the NAT function

NAT44 + NAT44 = “NAT444”



IPv4$_{private}$

IPv4$_{private}$

Large-Scale NAT (LSN)

IPv4 Internet

NAT44

NAT44

Home network

ISP network

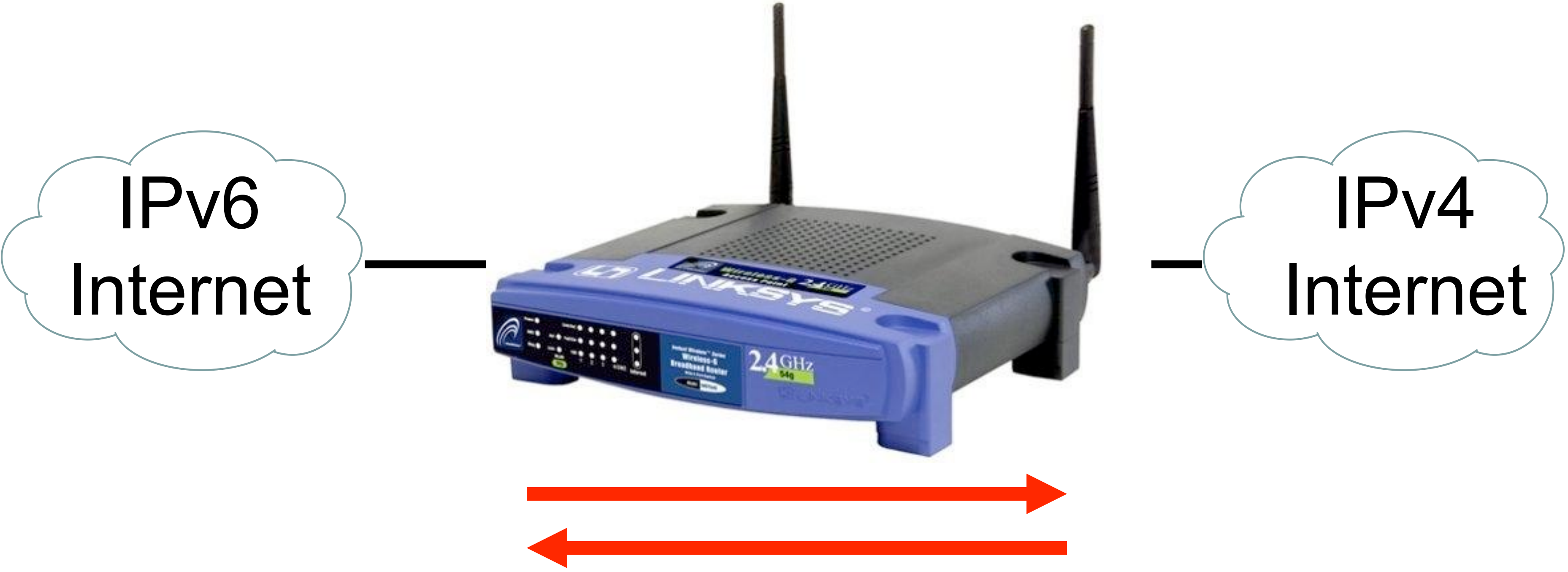
Agenda

- NAT and NAPT
  - Types of NATs
- Application Impact
  - Application Layer Gateway (ALG)
  - STUN, ICE, TURN
- Large-Scale NATs (LSN, CGN, SP NAT)
- IPv6/IPv4 Translation (“NAT64”)
- NAT66

Purpose of NAT64

- IPv6-only host to IPv4-only host
- Usually not needed
- *Try to use dual-stack*
  - *with NAPT44 to share IPv4 addresses*

The Ideal IPv6/IPv4 Translation



IPv6 Internet

IPv4 Internet

Translation versus Tunneling

- If you have a choice, tunnel
  - 6rd (IPv6 over IPv4)
  - Dual-Stack Lite (IPv4 over IPv6)
- Translate only when crossing between address families
  - IPv4-only host to IPv6-only host
  - IPv6-only host to IPv4-only host

Then, Why Translate?

- Will exhaust IPv4 addresses in 2011-2012
- IPv6-only clients need to access IPv4-only content
- Long tail of IPv4-only content
  - Children’s soccer practice schedule

- Longer term: need to access IPv6-only servers from IPv4-only clients

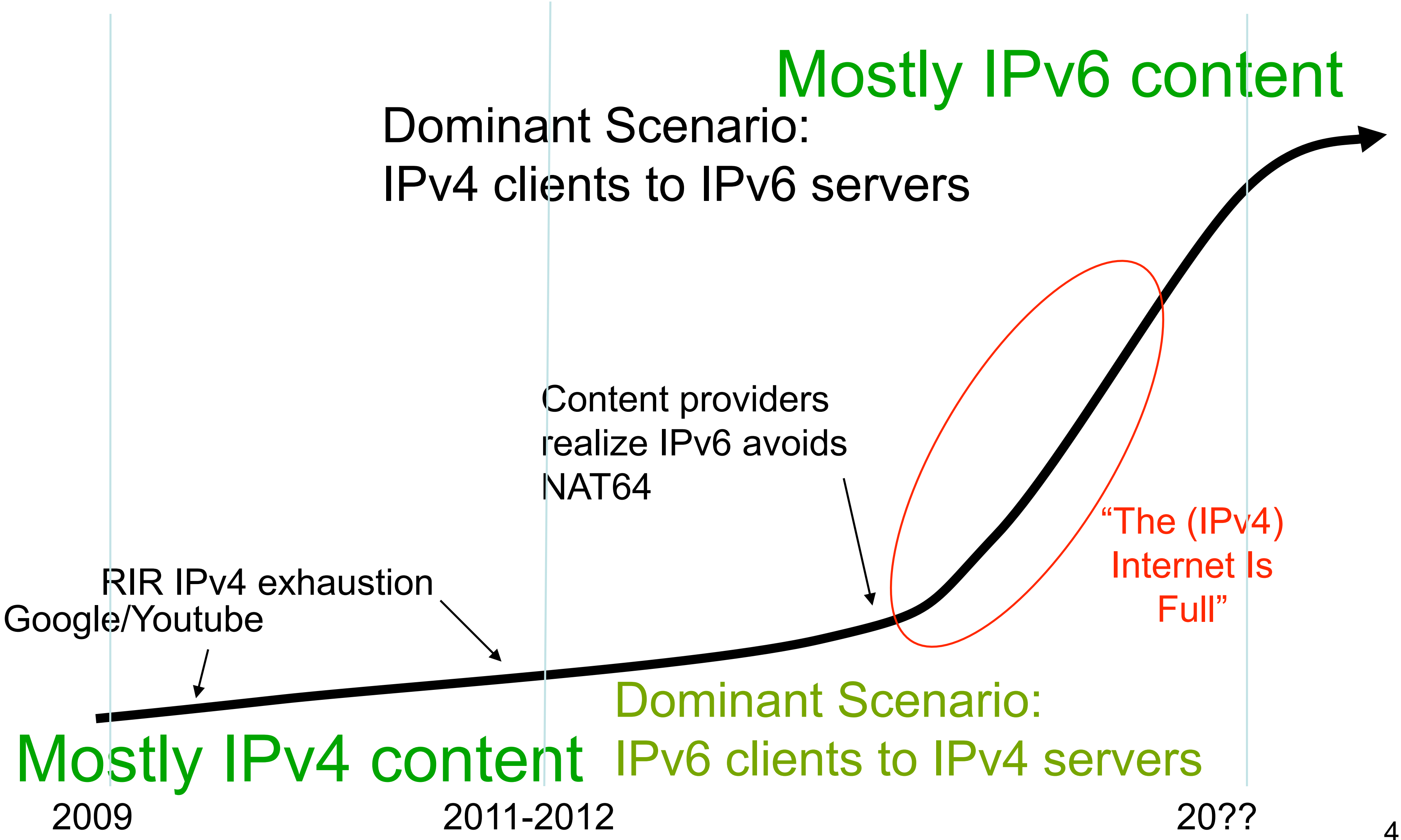
NAT-PT

- NAT-PT combined all scenarios
  - IPv4 to IPv6 is problematic; IPv6 space is bigger
  - Broke DNSSEC
- RFC4966 said IPv6/IPv4 translation causes other side effects
  - (But some are not solvable!)

- But:
- IPv4 addresses running out
- Effectively no IPv6 Internet access and no IPv6 content anywhere in the world
- We can’t tunnel everywhere

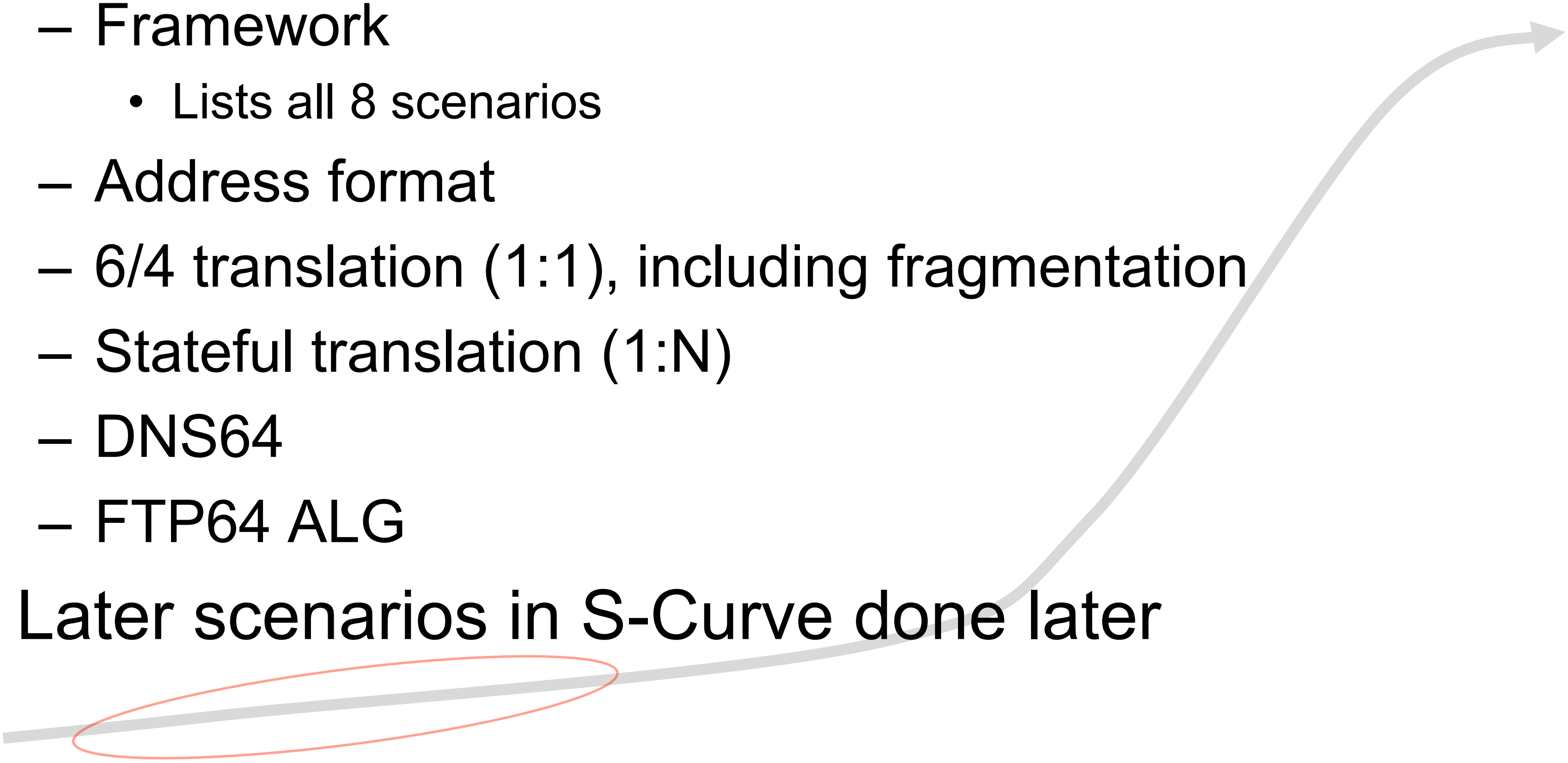
RFC2766

Translation Evolution S-Curve



BEHAVE’s Approach

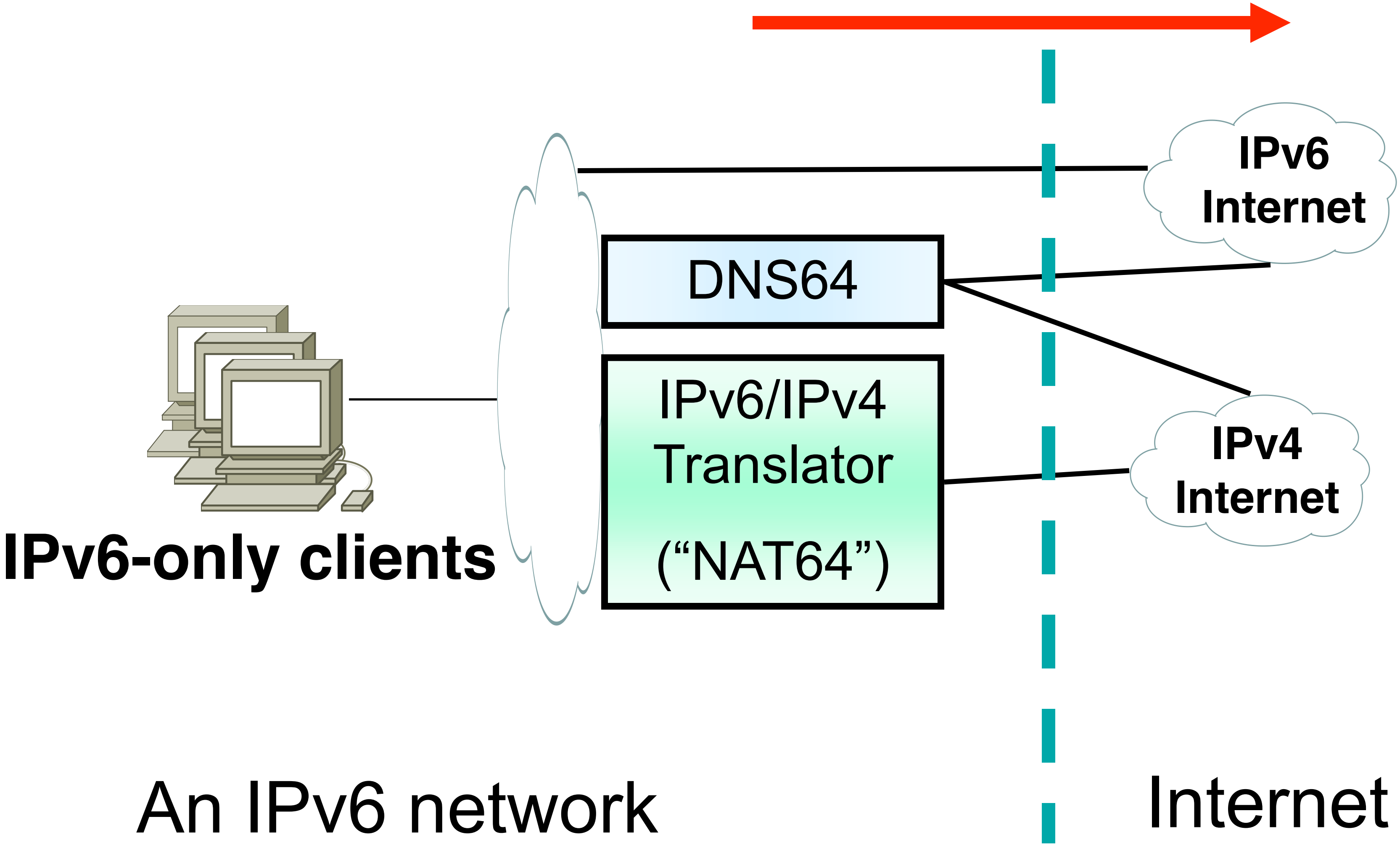
- Do first part of S-Curve first
- Split problem into separate documents
  - Framework
    - Lists all 8 scenarios
  - Address format
  - 6/4 translation (1:1), including fragmentation
  - Stateful translation (1:N)
  - DNS64
  - FTP64 ALG
- Later scenarios in S-Curve done later



IPv6/IPv4 Translation: some detail

- Connecting an IPv6 network to the IPv4 Internet
  - You built an IPv6-only network, and want to access servers on the IPv4 Internet
- Connecting the IPv6 Internet to an IPv4 network
  - You have IPv4 servers, and want them available to the IPv6 Internet
- Connecting the IPv4 Internet to an IPv6 network
  - You built an IPv6-only network, and want its servers available to the IPv4 Internet

Connecting an IPv6 network to the IPv4 Internet



IPv6 Internet

DNS64

IPv6/IPv4 Translator

(“NAT64”)

IPv4 Internet

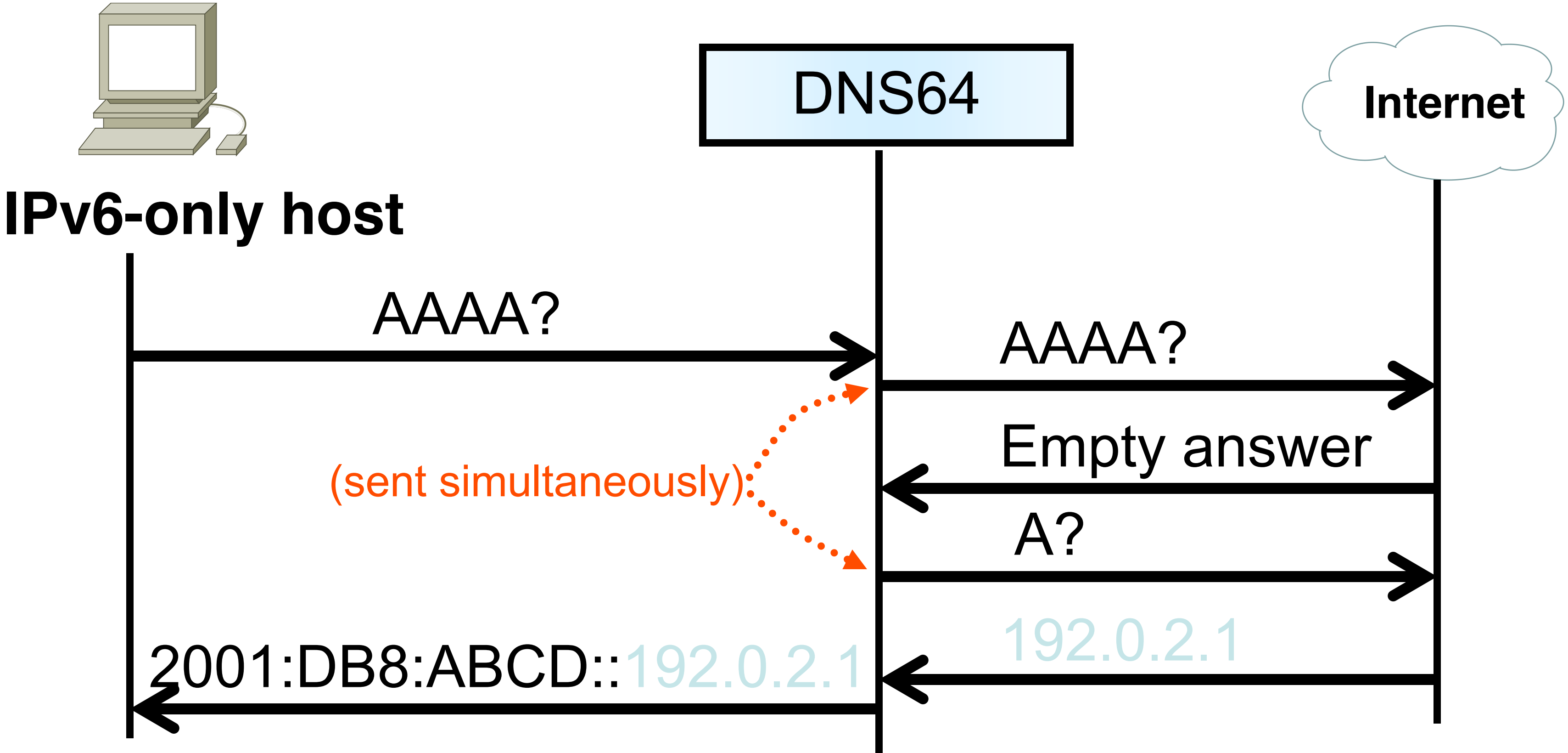
IPv6-only clients

An IPv6 network

Internet

DNS64

- Synthesizes AAAA records when not present
  - With IPv6 prefix of NAT64 translator



IPv6/IPv4 Translation

Stateless

- 1:1 translation
- "NAT"
- Any protocol
- No IPv4 address savings
  - Just like dual-stack

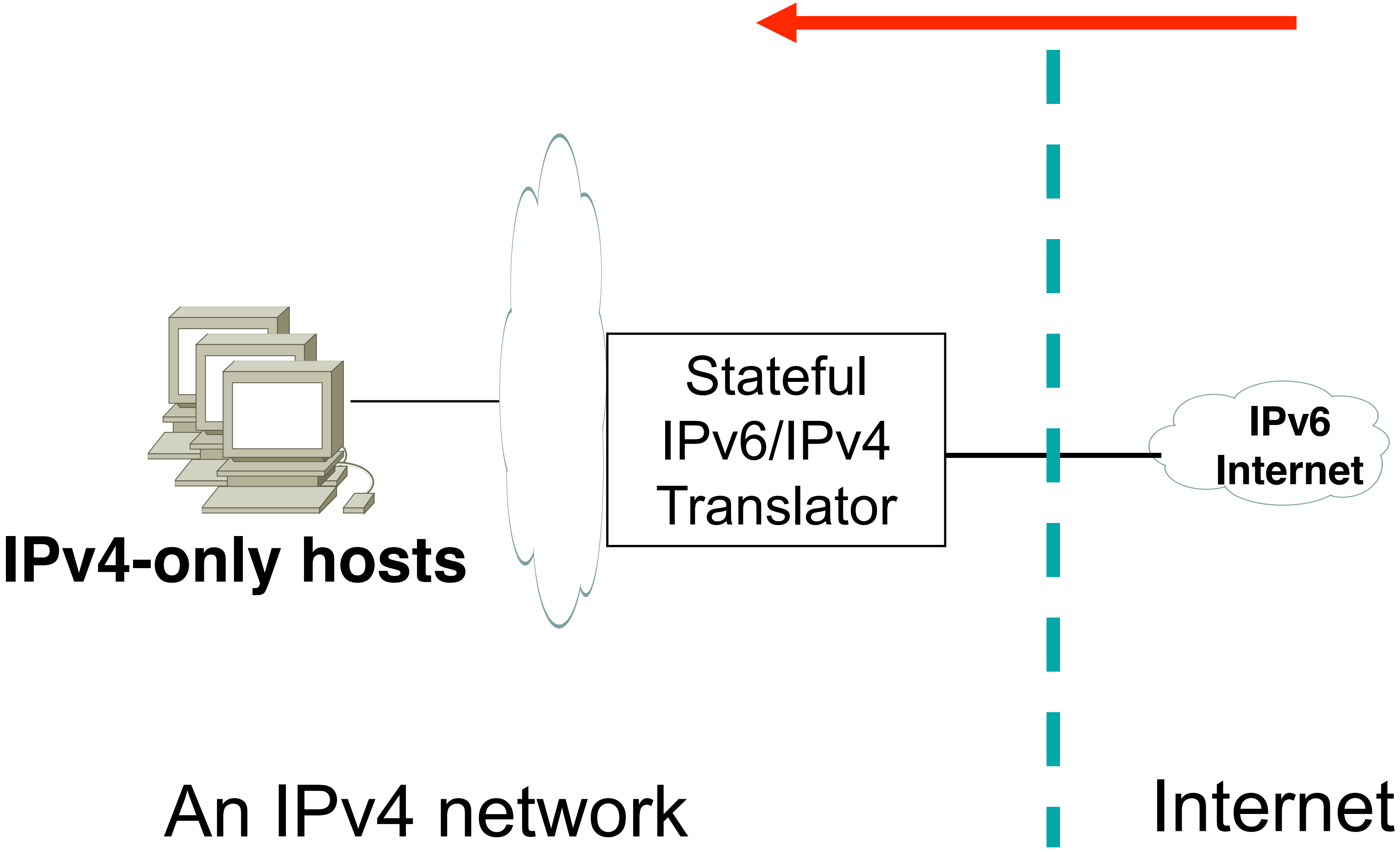
Stateful

- 1:N translation
- "NAPT"
- TCP, UDP, ICMP
- Saves IPv4 addresses

IPv6/IPv4 translation issues

- IPv4 address literals
  - http://1.2.3.4
  - SIP, RTSP, SAP
- IP Family sensitive protocols
  - FTP (EPSV, PASV)
- How to resolve?
  - Application proxies, make application smarter, ALG (FTP64)

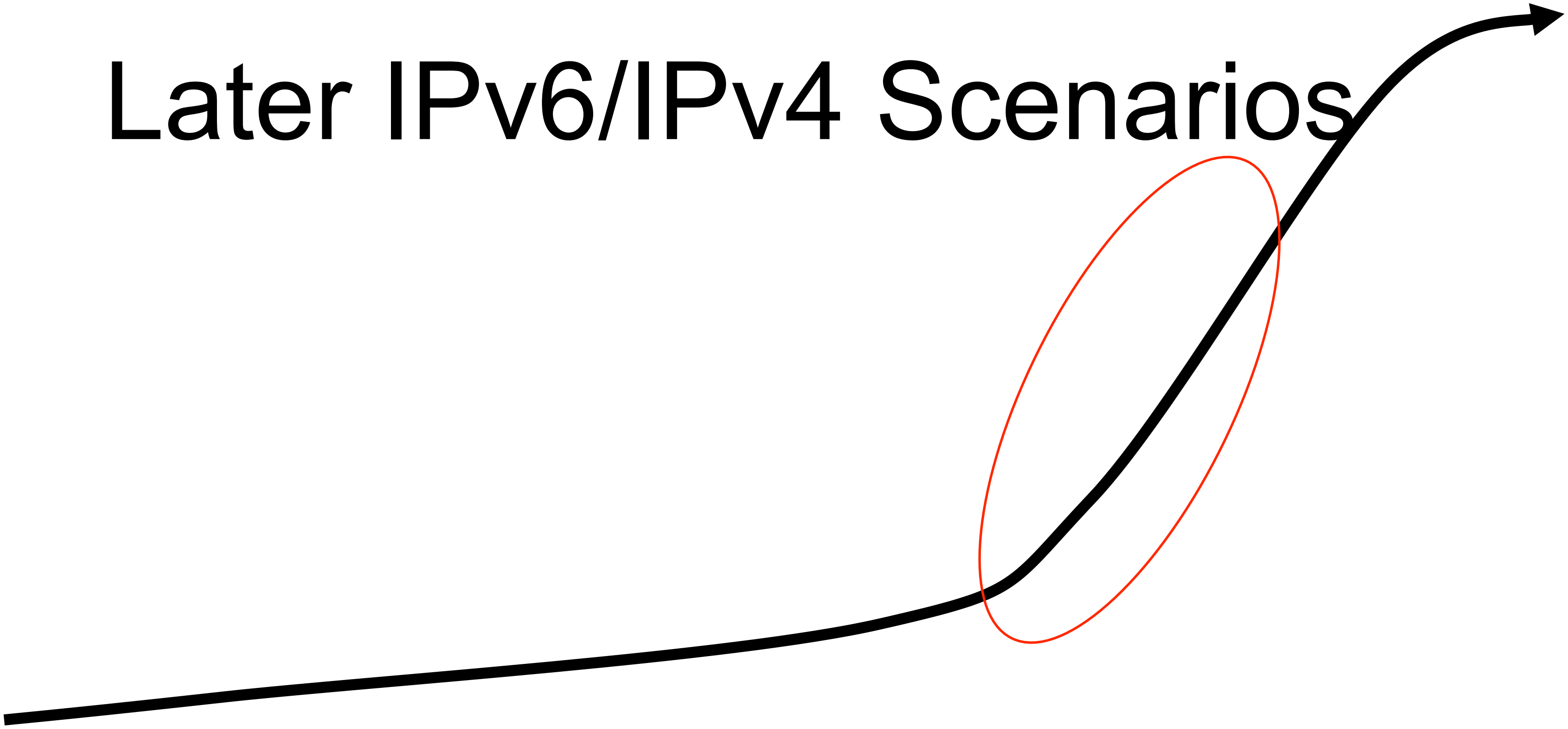
Connecting the IPv6 Internet to an IPv4 network



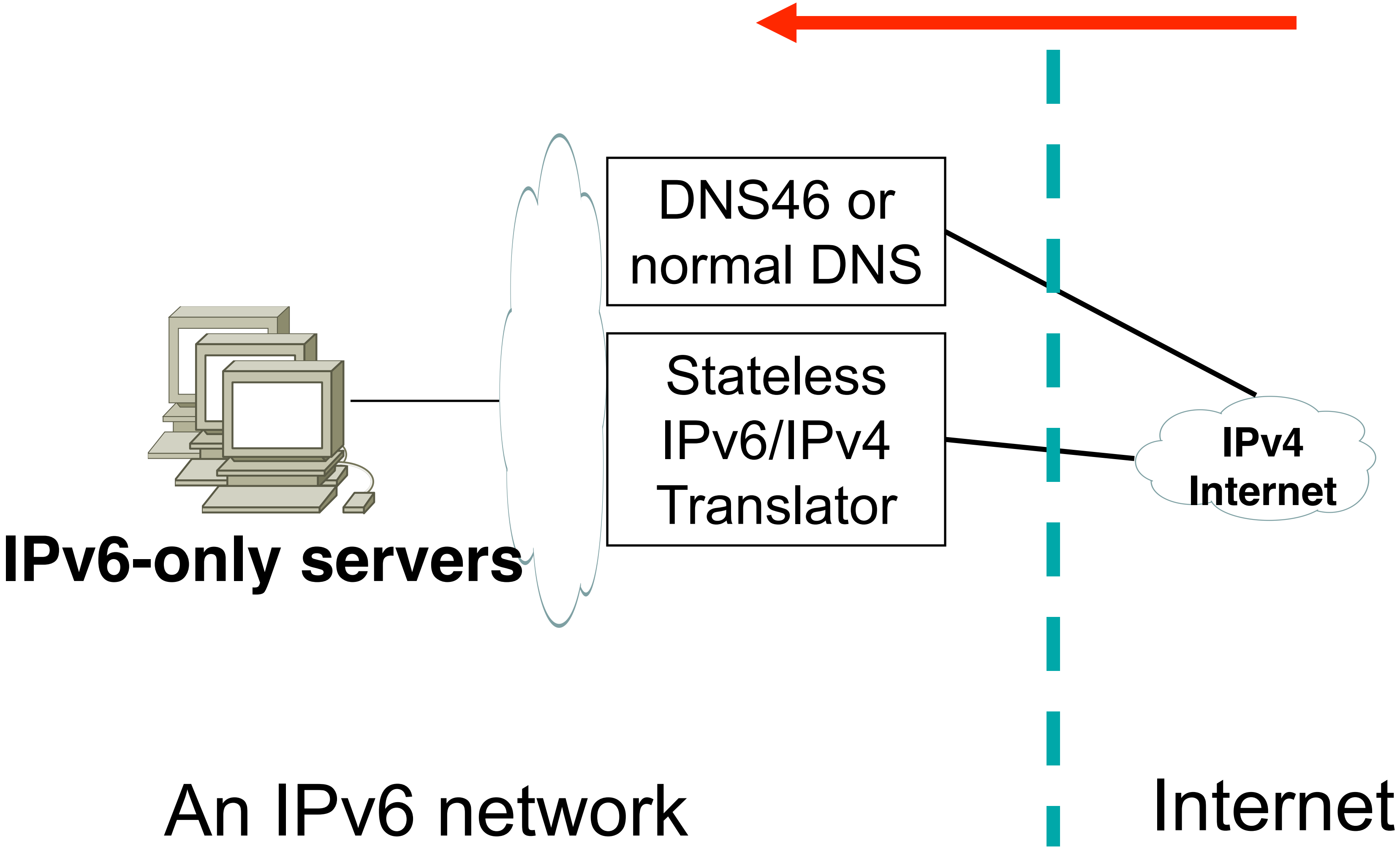
Connecting the IPv6 Internet to an IPv4 network

- Makes IPv4-only servers accessible on the IPv6 Internet
- Requires stateful translation
  - Because IPv6 Internet is bigger than IPv4
  - (can’t represent every address in IPv4)
- All connections come from translator’s IPv4 address
  - Problem for abuse logging
  - Lack of X-Forwarded-For: header
- Maybe application proxy is superior?
  - E.g., lighthttpd
  - But has poor TLS interaction

Later IPv6/IPv4 Scenarios



Connecting the IPv4 Internet to an IPv6 network



DNS46 or normal DNS

Stateless IPv6/IPv4 Translator

Connecting the IPv4 Internet to an IPv6 network

- Stateless works well, one IPv4 address for each IPv6 server
  - Same IPv4 consumption as dual-stack
- Just like with NAT64 case, don’t use IPv6 address literals
  - IPv4-only client can’t understand them!

Agenda

- NAT and NAPT
  - Types of NATs
- Application Impact
  - Application Layer Gateway (ALG)
  - STUN, ICE, TURN
- Large-Scale NATs (LSN, CGN, SP NAT)
- IPv6/IPv4 Translation (“NAT64”)
- NAT66

NAT66 Is **Not**

- Sharing IP addresses
- Modifying TCP or modifying UDP ports
- Stateful

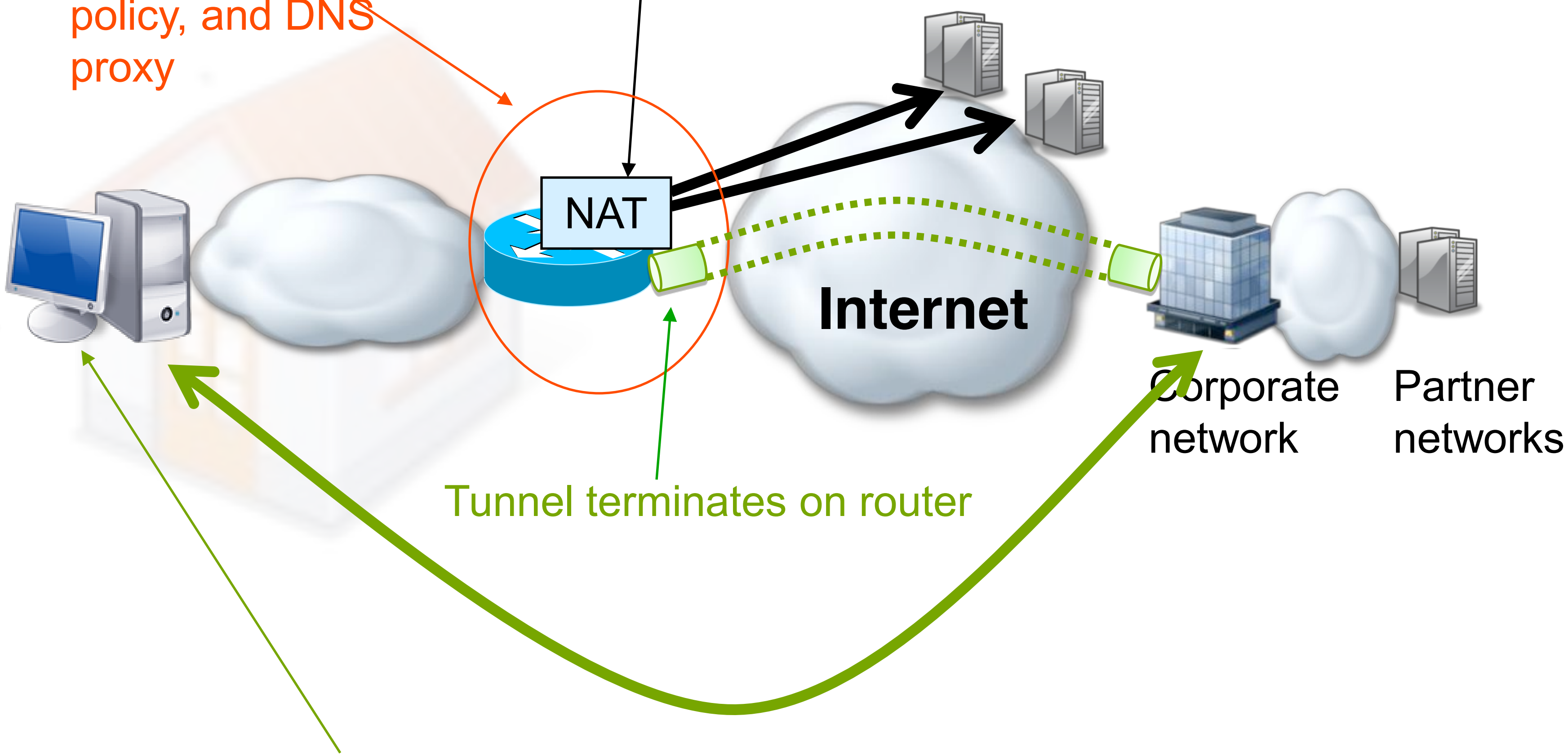
NAT66 Is

- Rewriting IPv6 prefixes

Goal

- Give host multiple IPv6 prefixes
  - Belonging to different networks
- Host does "The Right Thing"
- Not yet achievable

Tunnel to Enterprise, IPv4



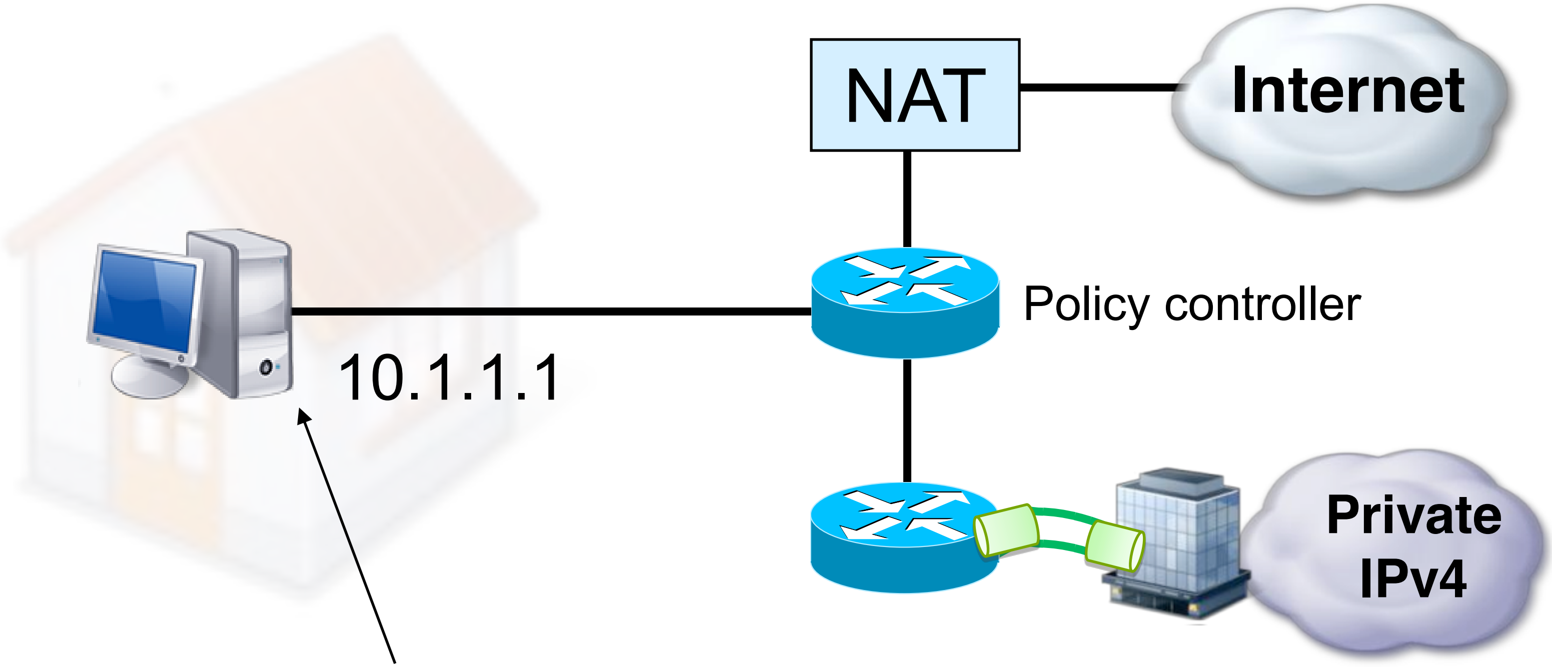
Contains routing and source address policy, and DNS proxy

NAT traffic to Internet

Tunnel terminates on router

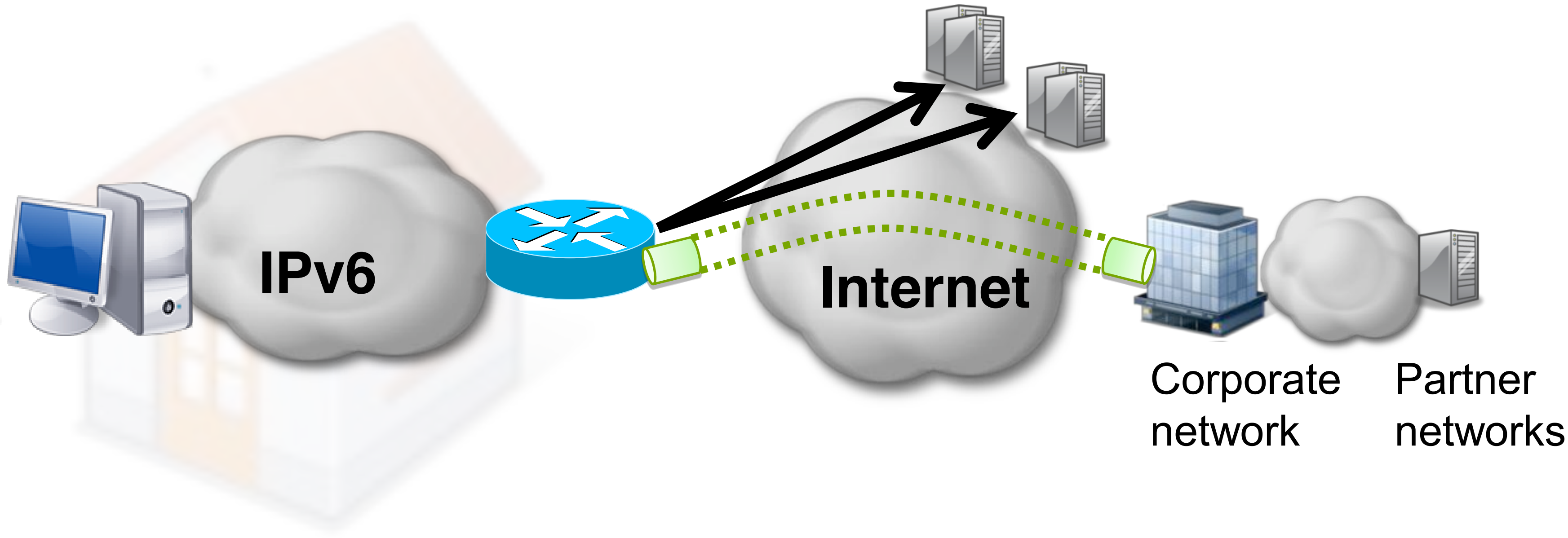
one IP address: corporate IP address

Simplified Tunnel Diagram, IPv4



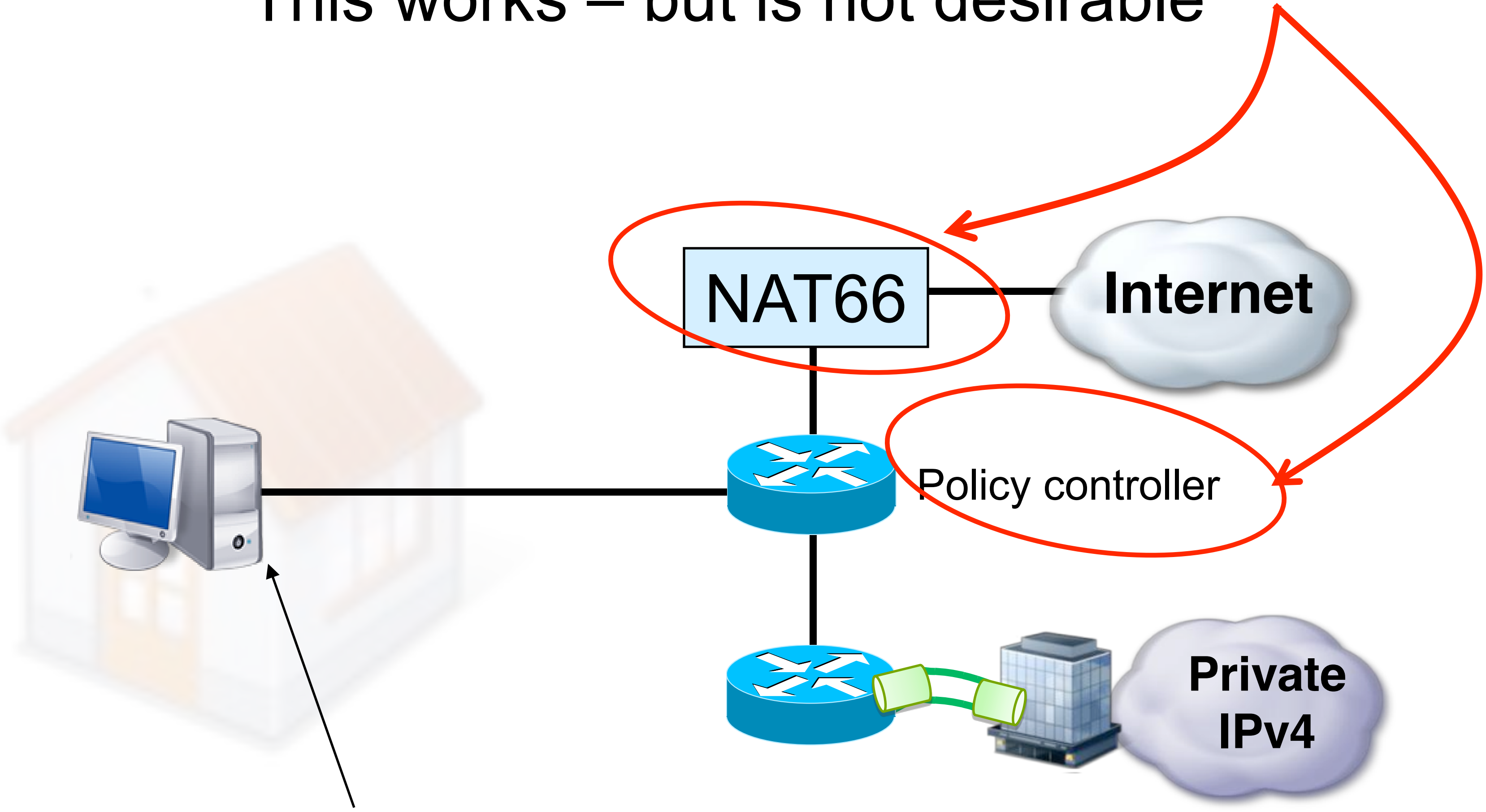
Single IP address

Same Scenario, IPv6



Simplified Tunnel Diagram, IPv6

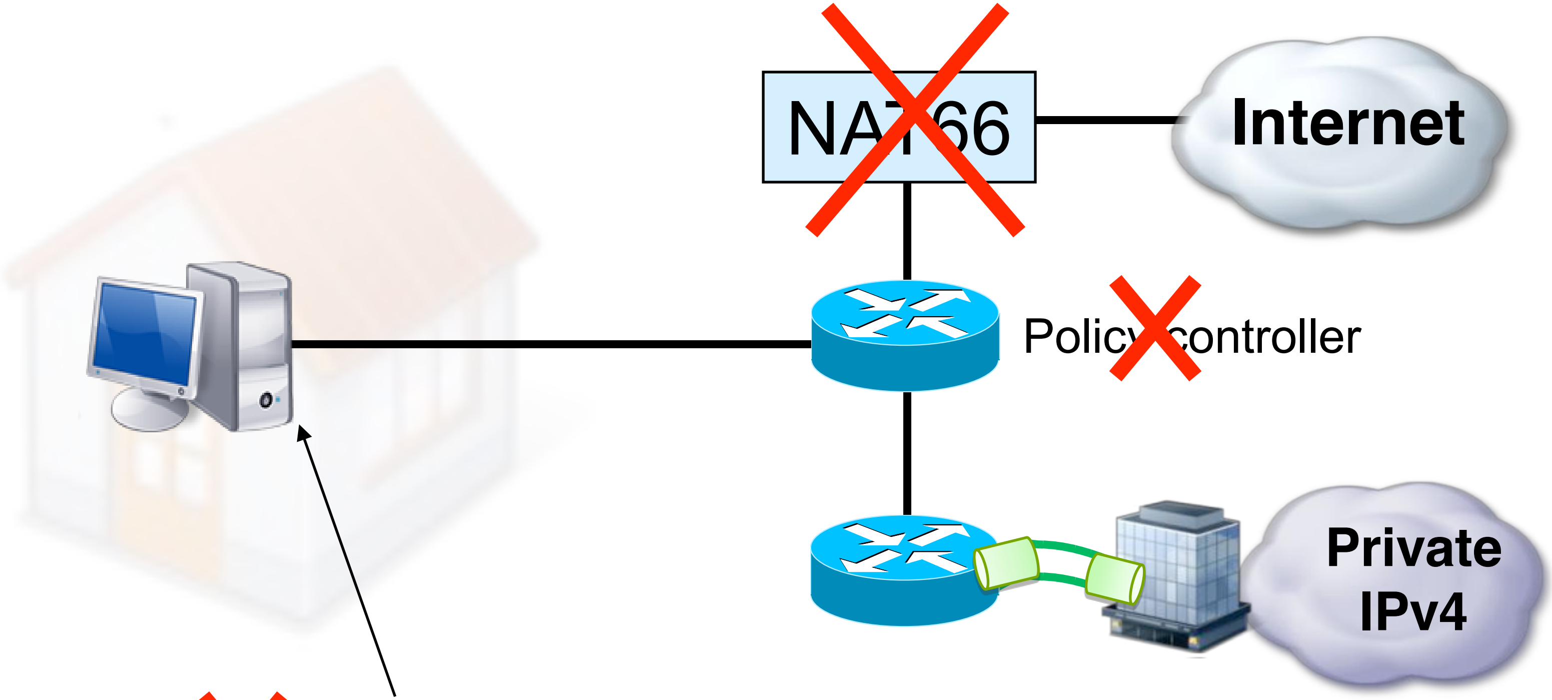
This works – but is not desirable



Single IPv6 address

Simplified Tunnel Diagram, IPv6

Desired



~~Single~~ Multiple IPv6 addresses

Why Consider NAT66

- Host and standards deficiencies:

1. Source Address Selection
2. Next-Hop Route Selection
3. Split-zone DNS
4. (Identifying Supporting Hosts)

Multihome with Provider-Dependent Address

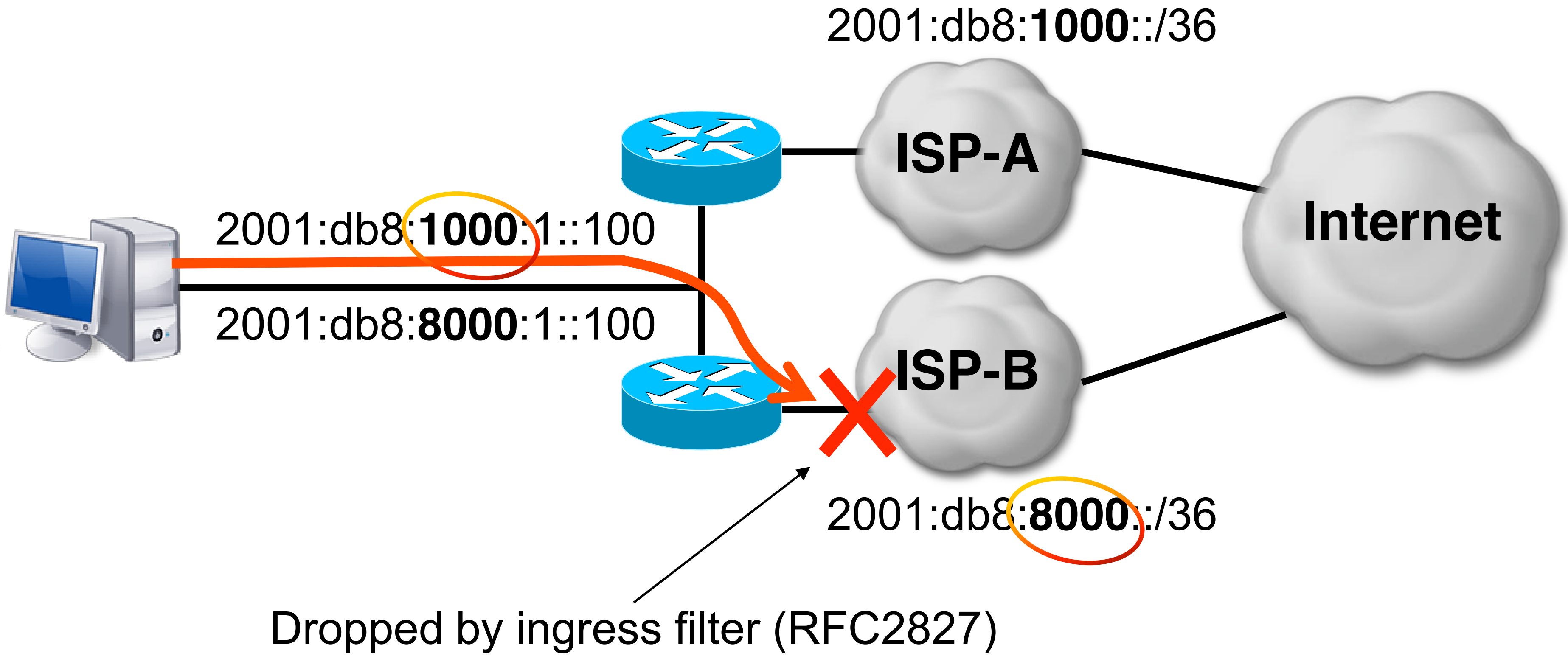
Privacy (RFC4941)

~~IP address sharing~~

Avoid renumbering

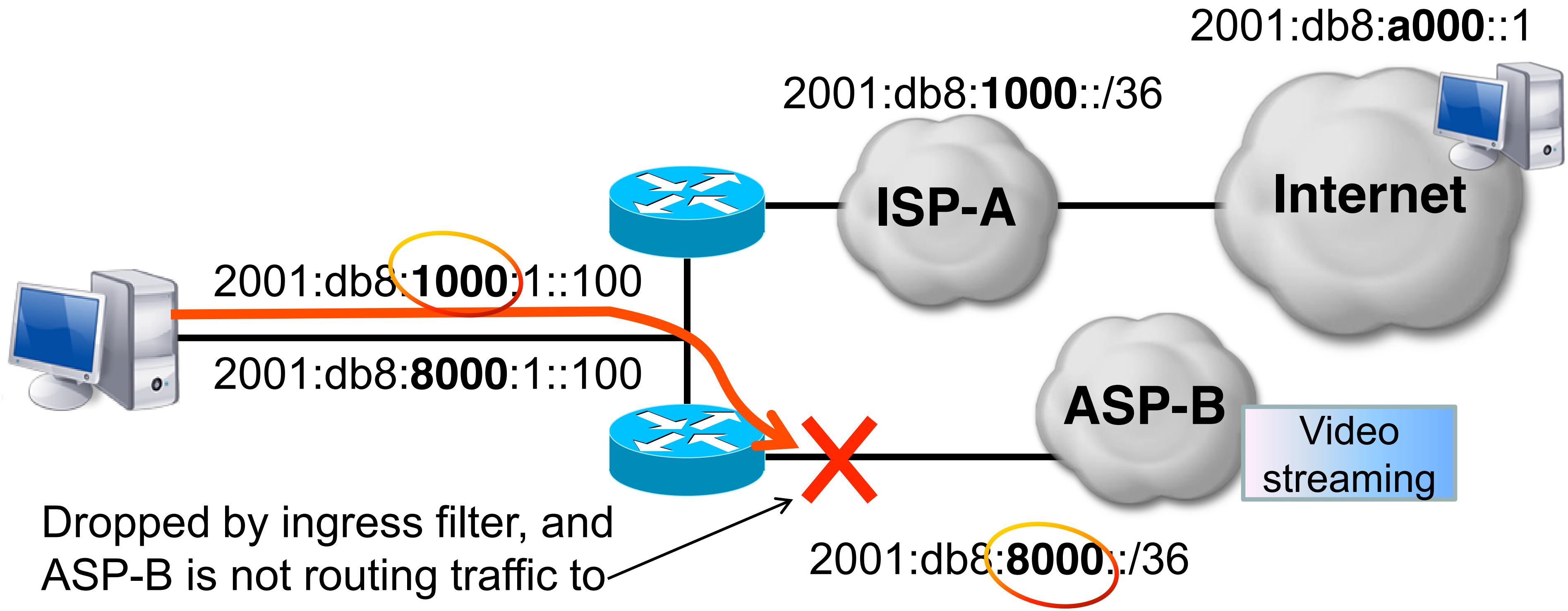
Problem: Source Address Selection

- Multiple prefixes on one physical interface
- Wrong ISP



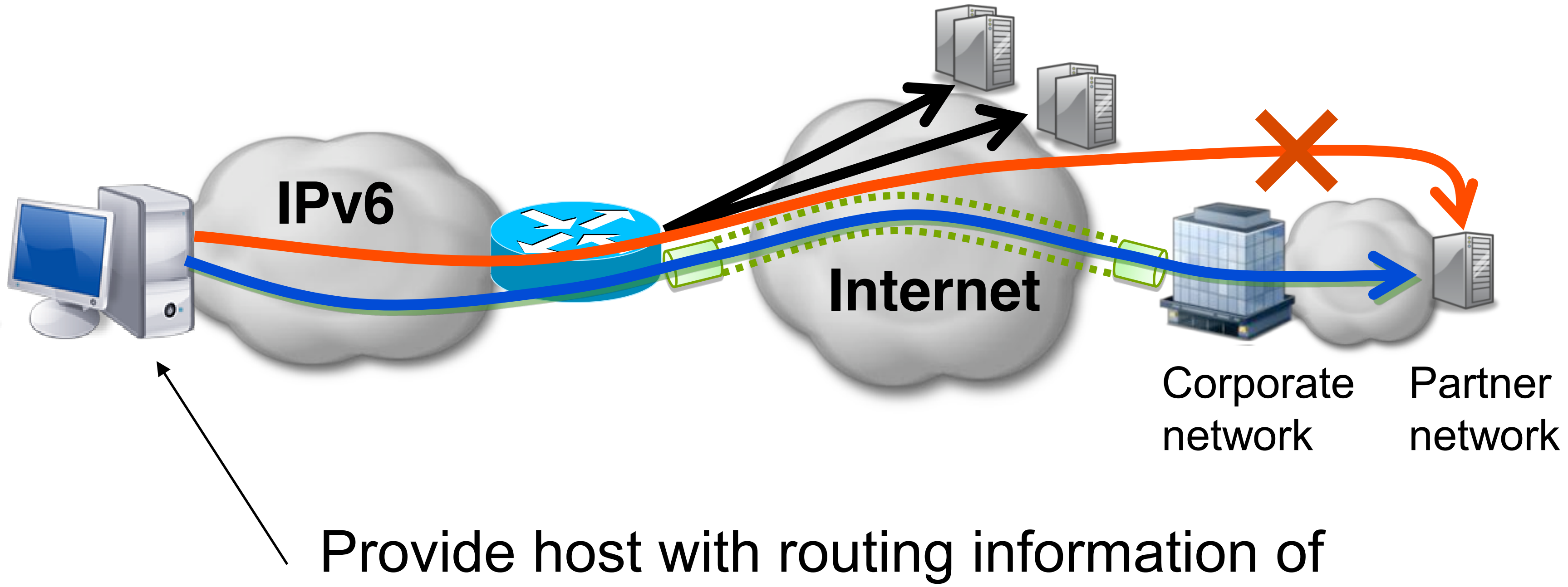
Problem: Source Address Selection

- Multiple prefixes on one physical interface
- Disconnected network



Dropped by ingress filter, and ASP-B is not routing traffic to Internet

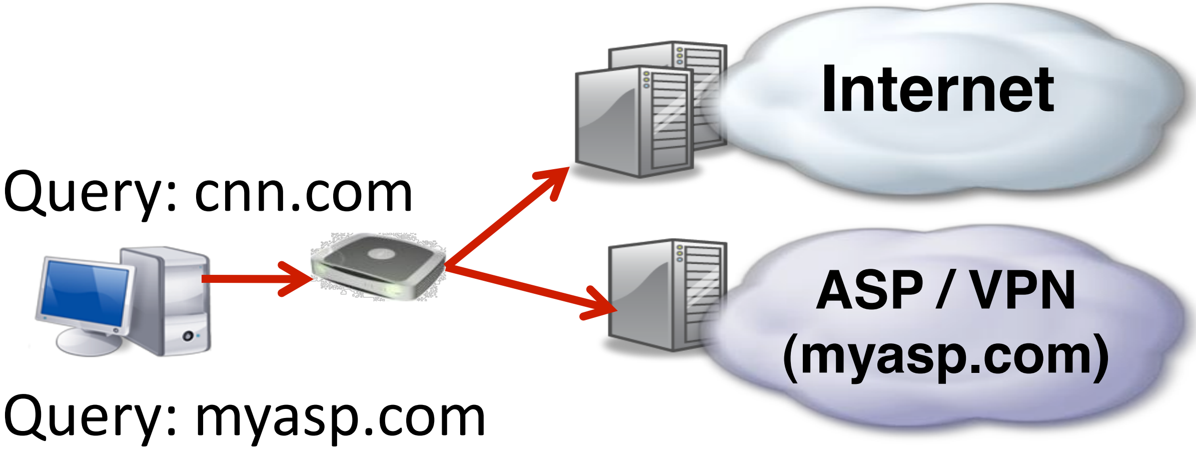
Problem: Next-Hop Route Selection



Provide host with routing information of Partner network – so that Address Selection (RFC3484) can choose correct source address. **RFC4191 does that** (but there is a problem..)

Problem: DNS Server Selection

- Split DNS
  - – Public DNS returns empty answer
  - – Private DNS returns IP address
- Solution: host queries proper DNS server
- long-existing industry practice



Problem: Identifying Supporting Hosts

- Supporting Host:
  - Chooses proper source address
  - Accepts next-hop route information
  - Supports split-zone DNS
- Network would like to determine:
  - If ‘supporting host’, give it two prefixes
  - If ‘non-supporting host’, give it one prefix and NAT66 its traffic

Scope of New Work

	Multiple physical interfaces	**Multiple prefixes**
Source Address Selection	√ RFC3484	Revise standard
Next-Hop Route	√ (RFC4191)	√ (RFC4191)
Split-Zone DNS	new standard	new standard
Identify supporting hosts	new standard	new standard

Actions

- Accelerate standards and implementations to avoid NAT66
  - Source address selection ← IETF: 6MAN
  - Route selection
  - Split-zone DNS

  IETF: MIF
- Mechanism to identify supporting hosts

draft-fujisaki-dhc-addr-select-opt
draft-dec-dhcpv6-route-option
draft-savolainen-mif-dns-server-selection

BEHAVE Status

Major Finished Work

- RFC
  - NAT44 behaviors: TCP, UDP, ICMP
  - STUN, TURN, ICE (MMUSIC)

BEHAVE Nearly Finished Work

- IPv6/IPv4 Translation Scenarios
  - √ 1: an IPv6 network to the IPv4 Internet
  - – 2: the IPv4 Internet to an IPv6 network
  - √ 3: the IPv6 Internet to an IPv4 network
  - – 4: an IPv4 network to the IPv6 Internet
  - √ 5: an IPv6 network to an IPv4 network
  - – 6: an IPv4 network to an IPv6 network

BEHAVE Finished 6/4 Translation Documents

- draft-ietf-behave-address-format
- draft-ietf-behave-dns64
- draft-ietf-behave-v6v4-framework
- draft-ietf-behave-v6v4-xlate-stateful
- draft-ietf-behave-v6v4-xlate

BEHAVE Outstanding NAT Work

- draft-ietf-behave-ftp64
- draft-ietf-behave-sctpnat

Summary

- NAT and NAPT
  - – Types of NATs
- Application Impact
  - – Application Layer Gateway (ALG)
  - – STUN, ICE, TURN
- Large-Scale NATs (LSN, CGN, SP NAT)
- IPv6/IPv4 Translation ("NAT64")
- NAT66

Questions

Dan Wing, dwing@cisco.com