Aalto University
School of Electrical Engineering

Customer Edge Switching & Realm Gateway
Tutorial Session – Day 2

Jesus Llorente Santos
jesus.llorente.santos@aalto.fi

www.re2ee.org

August 21st, 2015

Outline

- Recap from yesterday
  - Current Internet Model
  - Issues with the Current Model - NATs
- Benefits of Realm Gateway
- How does it work
  - Role of DNS
- Improving Efficiency and Scalability
- Application and Protocol Compatibility
  - Application Layer Gateways
- Additional Material
  - RGW64 – Transition to IPv6
  - Introduction to Testbed, System Architecture, OpenFlow…
  - ALGs and Future ALG Engine

Current Internet Model

- Internet goes mobile
  - Massive growth of connected users and devices
  - Expect an exponential growth with the arrival of IoT

- Dominant presence of Network Address Translator (NAT)
  - Driven by the IPv4 address exhaustion
  - Allow multiple hosts to connect to the Internet with the same public IP address
  - Separation of private and public networks
    - Reuse same private networks over and over (~18M IPs)
    - 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16
  - Requires binding state of IPs and ports when packets traverse the NAT: public-to-private and private-to-public
  - Acts as a first layer of security blocking inbound connections

Current Internet Model

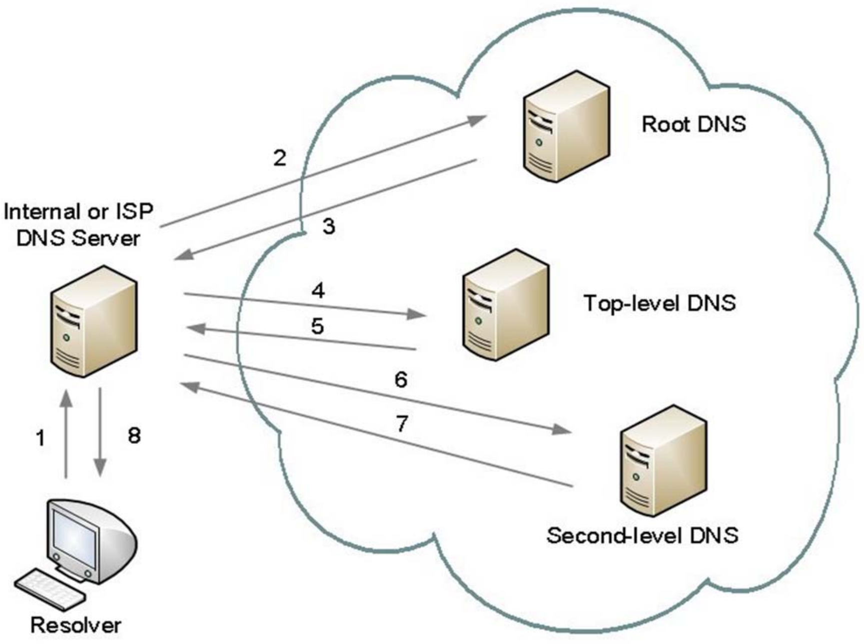
- Location of communicating nodes
  - Users typically located in private networks behind NATs
    - Reduce the amount of public IP addresses needed
    - Need to be able to initiate connections towards public servers
    - Example: computers, laptops, smartphones, etc.
  - Public servers and/or services must be publicly reachable
    - Directly reachable at IP layer via routing
    - Reachable via a proxy or frontend
    - Need to serve requests from connecting users
    - Example: Mail, SSH, HTTP(S), etc.

Current Internet Model

- Identification of hosts and services
  - By IP address
    - Valid on public networks may cause problems across private networks
    - Binds together host identity and routing locator
    - Not always easy to remember: *130.233.224.254*
  - By name
    - Typically following a hierarchical naming scheme, i.e. Fully Qualified Domain Name (FQDN) in DNS
    - Decouples host identity from routing locator
    - Easier to remember: *comnet.aalto.fi*

Current Internet Model

- Domain Name System – DNS
  - Resolves FQDN names to IP addresses (most typical function)
  - Transaction based Query/Response
  - Client-Server architecture

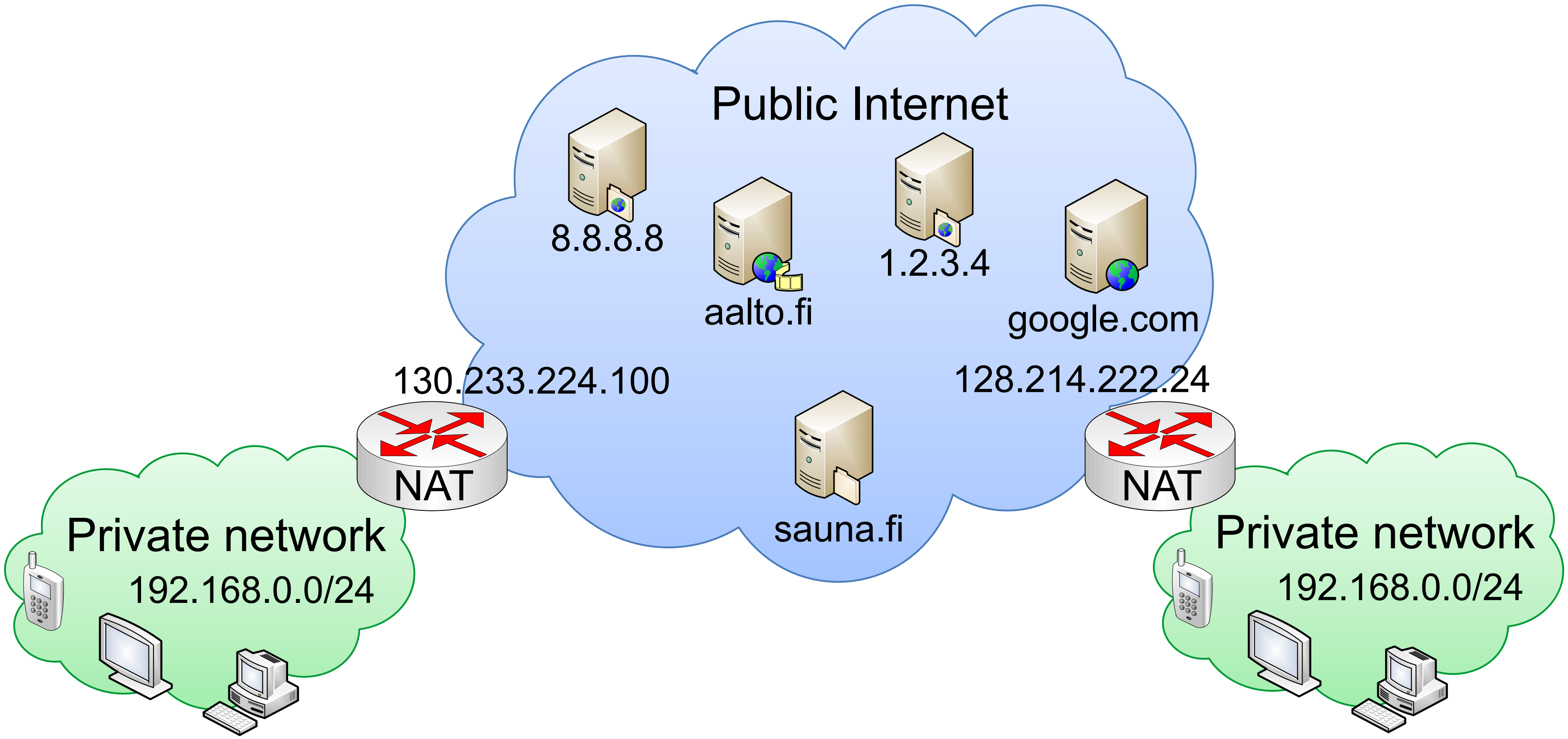


Current Internet Model

- Some example of DNS records
  - A: Resolution of IPv4 addresses
    - a.foo. IN A 192.0.2.100
  - AAAA: Resolution of IPv6 addresses
    - a.foo. IN AAAA 2001:DB8::192.0.2.100
  - CNAME: Canonical names pointing to other domains
    - a.foo. IN CNAME another-host.foo.
  - NAPTR: Name authority pointer
    - IN NAPTR 100 10 "U" "E2U+sip" "!^.*$!sip:a@demo.foo!" .
  - SRV: Service location including ports and protocols
    - _ssh._tcp.a.foo. ttl IN SRV p w 22 a.foo.

Current Internet Model

- Internet Architecture

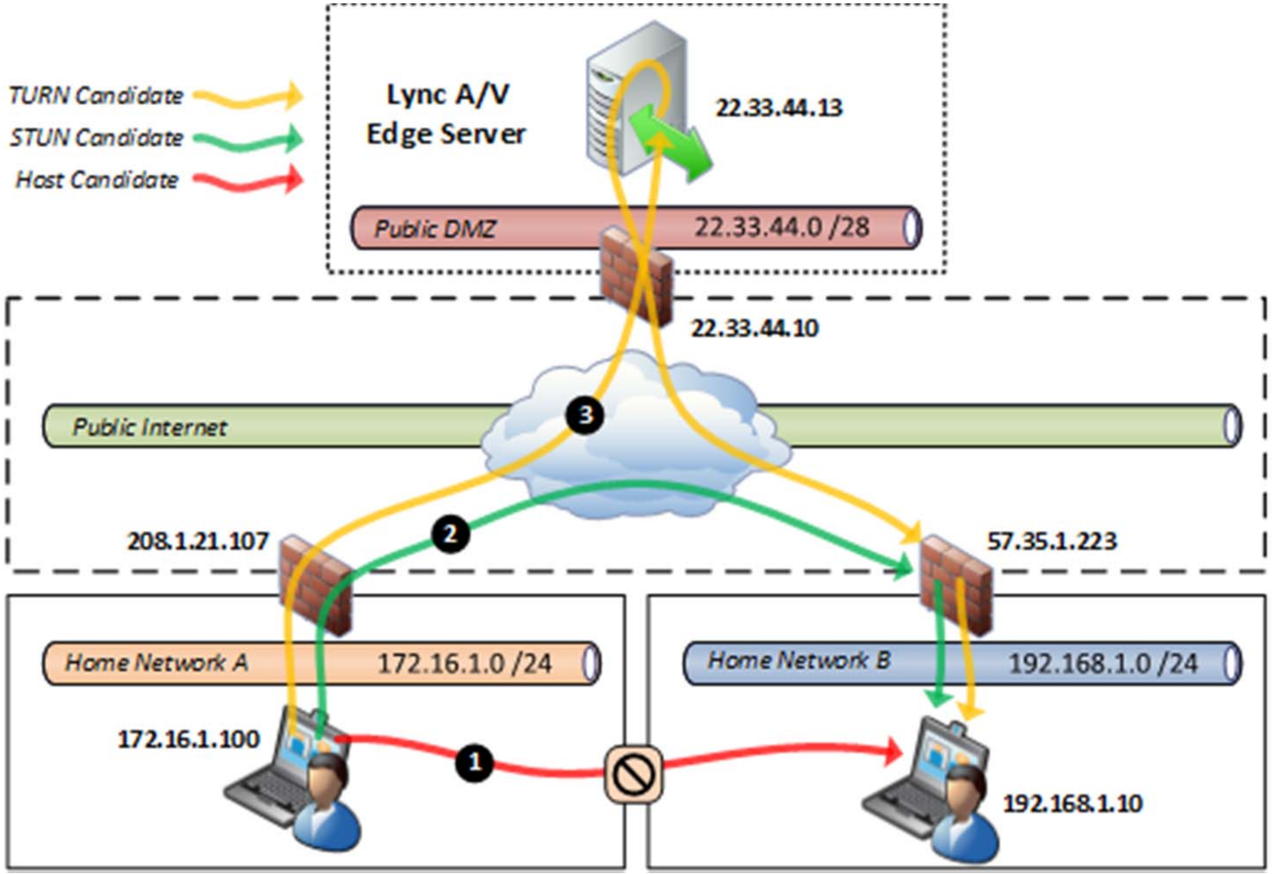


Issues with the current Internet Model

- NAT introduces reachability problem
  - Block inbound connections from reaching the private network
  - Connection state keeps track of private, public and remote IP addresses, ports numbers and protocols in use.
  - NAT-unfriendly protocols are negatively affected by NATs
    - Use of IP address literals or separate control/data connections
    - Require specific Application Layer Gateways e.g. SIP, FTP
  - Traversal of the NAT requires additional protocols
    - STUN/TURN/ICE
    - Results in increased delays during connection setup
    - Requires specific application code and increases network traffic

Issues with the current Internet Model

- More on STUN / TURN / ICE



Issues with the current Internet Model

- Unwanted traffic: Any source can send a packet to any destination address

- Possibility of source address spoofing makes it difficult to attribute evidence of misbehavior to the legitimate source

Benefits of Realm Gateway (RGW)

- Compatible with current NATs
  - Implemented as Address and Port-Dependent as per RFC-4787
  - Do not require changes to either hosts or protocols
  - Outbound connections are handled exactly the same

- Provide better-than-NAT service
  - Inbound connections are handled by the Circular Pool
  - Overcome the reachability problem towards private networks with temporary public address allocation and reuse

- Supports ALGs for NAT-unfriendly protocols
  - Compatible with current NAT-Traversal protocols

Benefits of Realm Gateway (RGW)

- Scalable solution with efficient address reuse
  - On the public side only requires a single IP address
  - Additional public IP addresses exponentially increase scalability

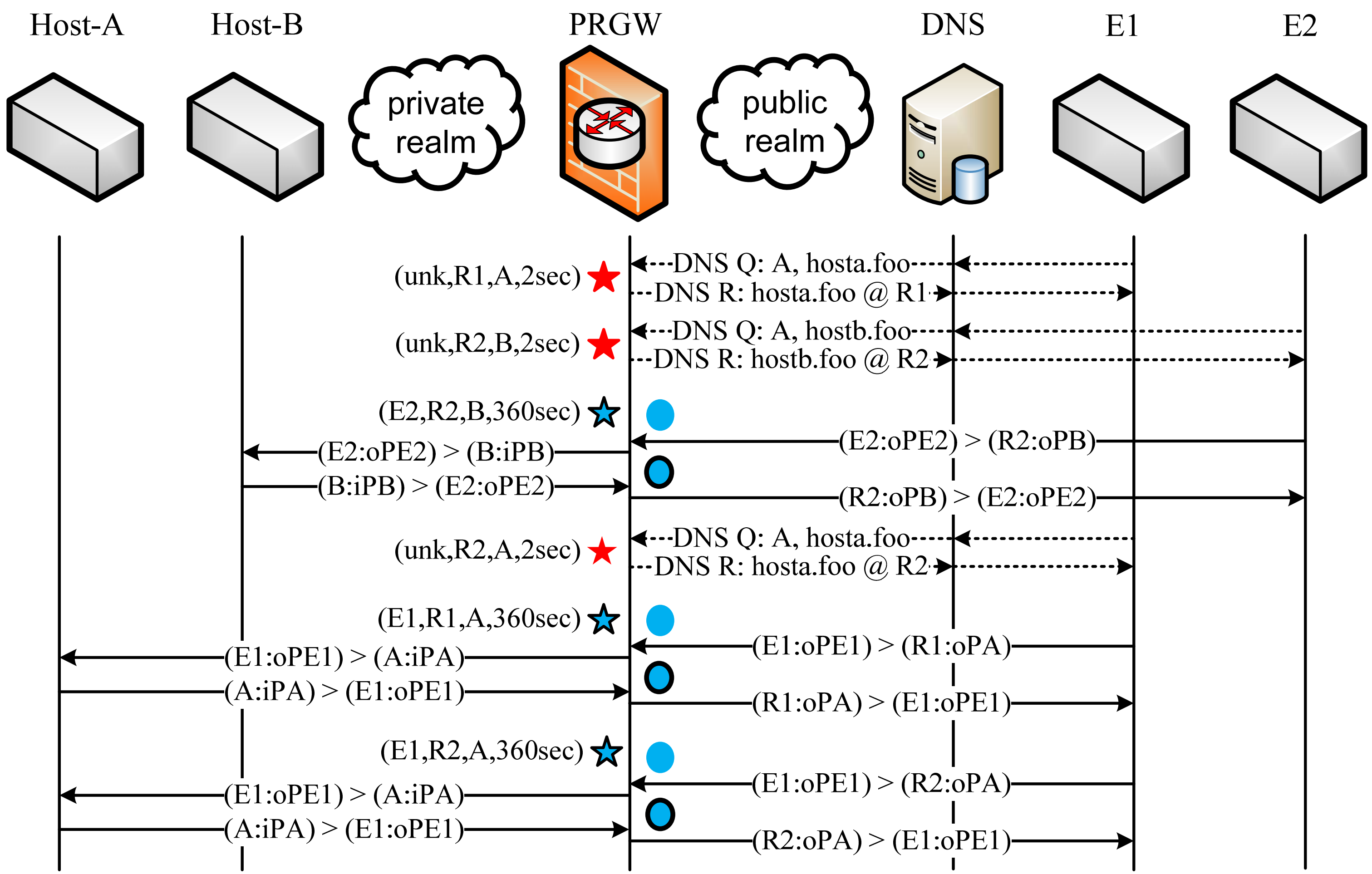
- Enables deployment one network at a time
  - Require redirection of DNS zone of authority
  - No other changes required in the network nodes
  - Reuses well-known and widely-used protocols, i.e. DNS

- Additional features
  - Implements compatibility with IPv6 networks featuring a stateful implementation of NAT64 and DNS64

How does RGW work?

- RGW acts as a DNS leaf node, hosting the DNS records of the hosts connected to the private network
- The Circular Pool algorithm contains a set of public IP addresses {R1,R2,R3}
- Incoming DNS requests (A) arrive at the RGW requesting an IP address to communicate with a private host
  - An address is allocated from the Circular Pool and offered in the DNS response. TTL is set to zero to avoid caching in intermediate nodes
  - The address is reserved for a time Tmax ≈ 2 sec before is automatically released
- Following incoming data packets not belonging to an ongoing connection, can claim the state
  - The address is released upon creating the new connection
  - A public host succeeds at communicating with a private host

How does RGW work?



Improving Address Efficiency

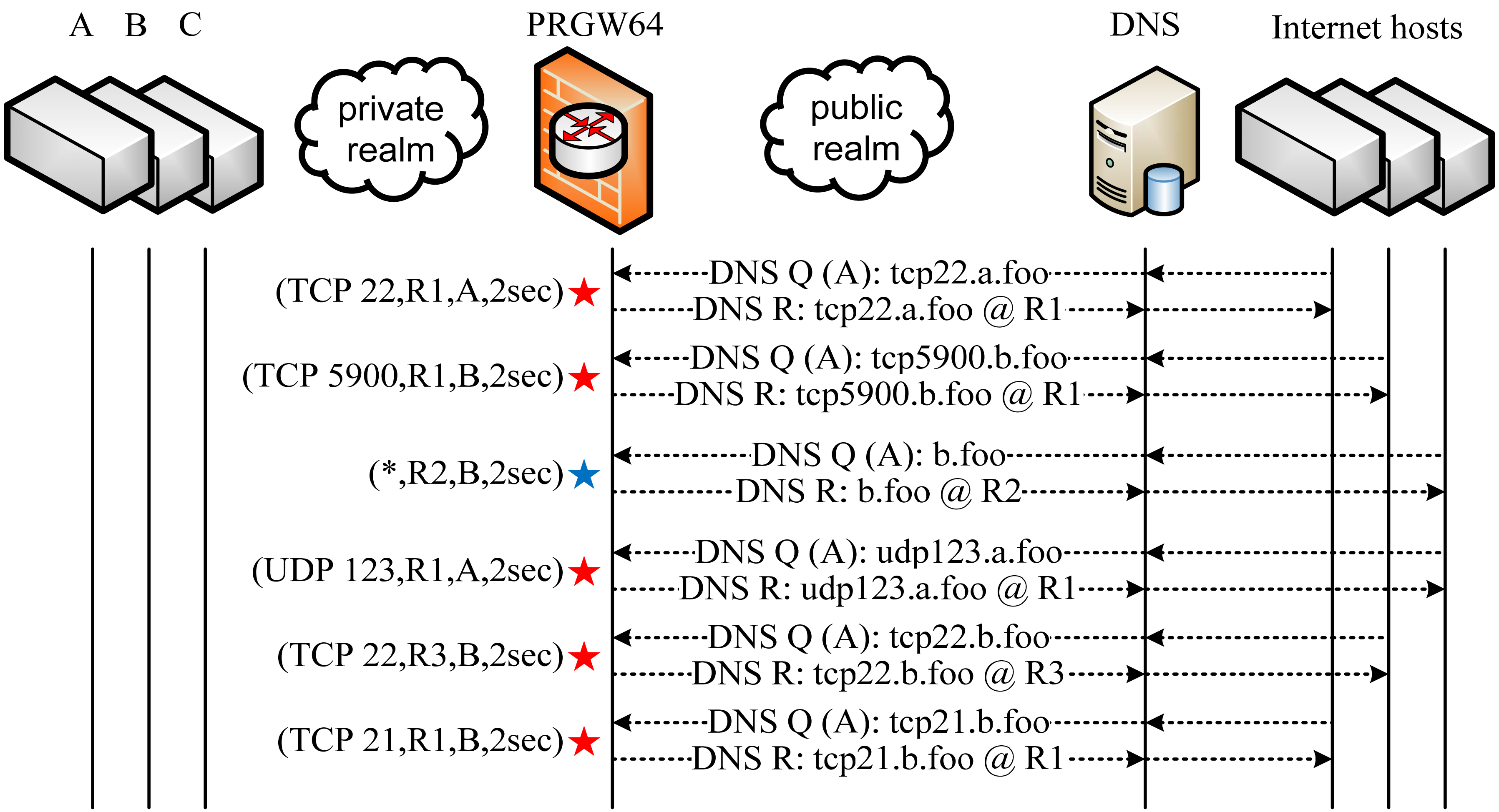
- Could we use DNS to specify the destination service and create a specific binding in RGW?
  - Yes! DNS SRV records do exactly that
    - Example: _stun._tcp.example.net
    - Not used by many applications

- Can we use DNS A records to mimic SRV behaviour?
  - Yes! We call them Service FQDN
  - They include the port and protocol used for the data connection
  - Example: tcp22.hosta.foo
  - SFQN are simple domain names that encode meta information to create a new naming scheme

Improving Address Efficiency

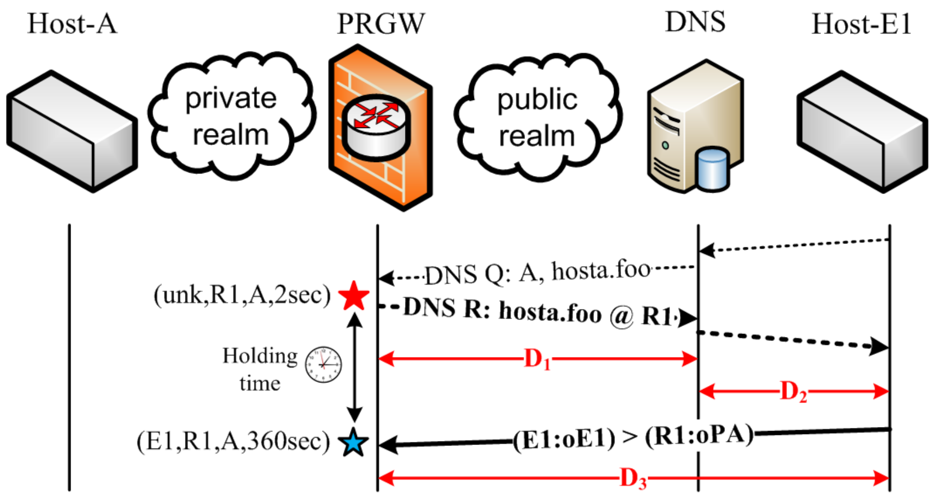
- The Circular Pool algorithm understands these SFQDN
- Rather than reserving a whole IP address per inbound connection we can create specific bindings
  - New binding: (Public IP, port, protocol)
- It is possible to overload the same IP address with multiple SFQDN connections
  - A new IP address is only required if there is already a waiting state for the same tuple (port, protocol)
- SFQDN maximizes address reuse and boosts efficiency of the Circular Pool but requires the sending host to adhere to the new naming scheme

Improving Address Efficiency



Realm Gateway Scalability

- Number of new connections that can be established per unit of time with an acceptable level of success at the first DNS request
- Define *service time* $T_{service}$, as the time duration that a public IP address R_X is reserved for establishing an inbound connection
  - $T_{service}$ spans the time from the creation of the temporary binding state until the first data packet of the connection is received, when R_X is released and returned to the pool.



Realm Gateway Scalability

- For standard FQDN queries the upper bound comes determined by the expression

$$\boldsymbol{\eta} = \frac{\boldsymbol{Pool\ size}}{\boldsymbol{Tservice}}$$

- For Service FQDN queries with built-in connecting service the upper bound comes determined by the expression

$$\boldsymbol{\eta}' = \frac{\boldsymbol{Nport\ x\ Nproto\ x\ Nip}}{\boldsymbol{Service\ time}}$$

Realm Gateway Scalability

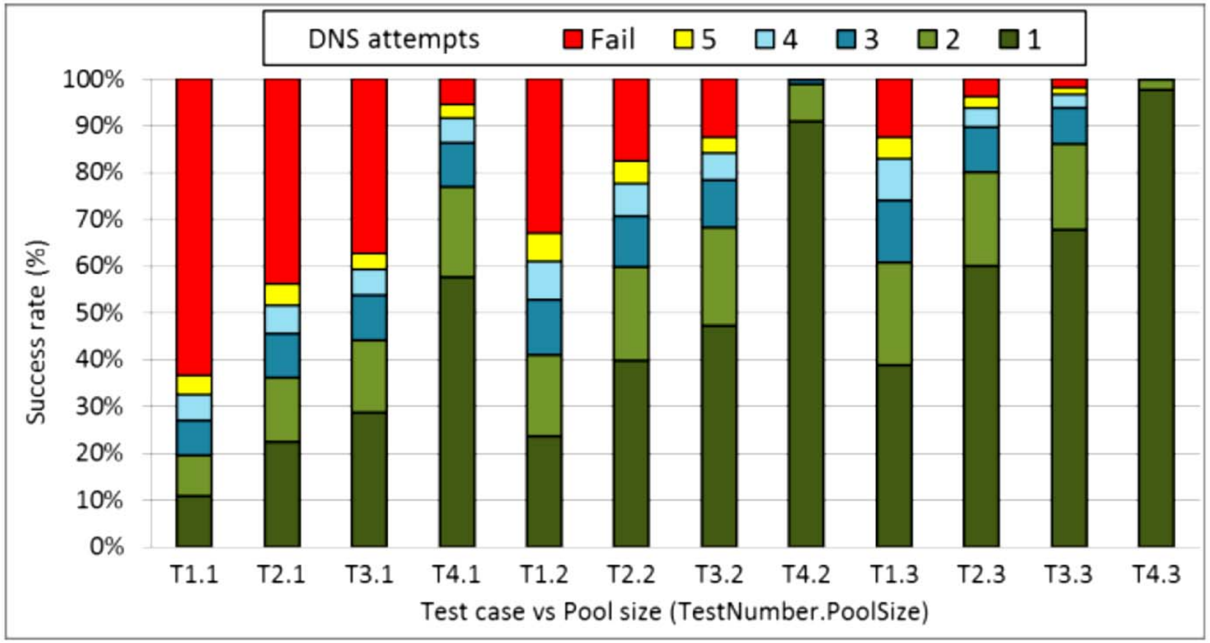


Fig. 3. Traffic distribution impact on performance

- Test 1: 100% FQDN
- Test 2: 50% FQDN + 50% SFQDN 5 geometric
- Test 3: 50% FQDN + 50% SFQDN 2x5 geometric
- Test 4: 100% SFQDN 2x5 geometric

50 ms delay
60 conn/sec

Realm Gateway Scalability

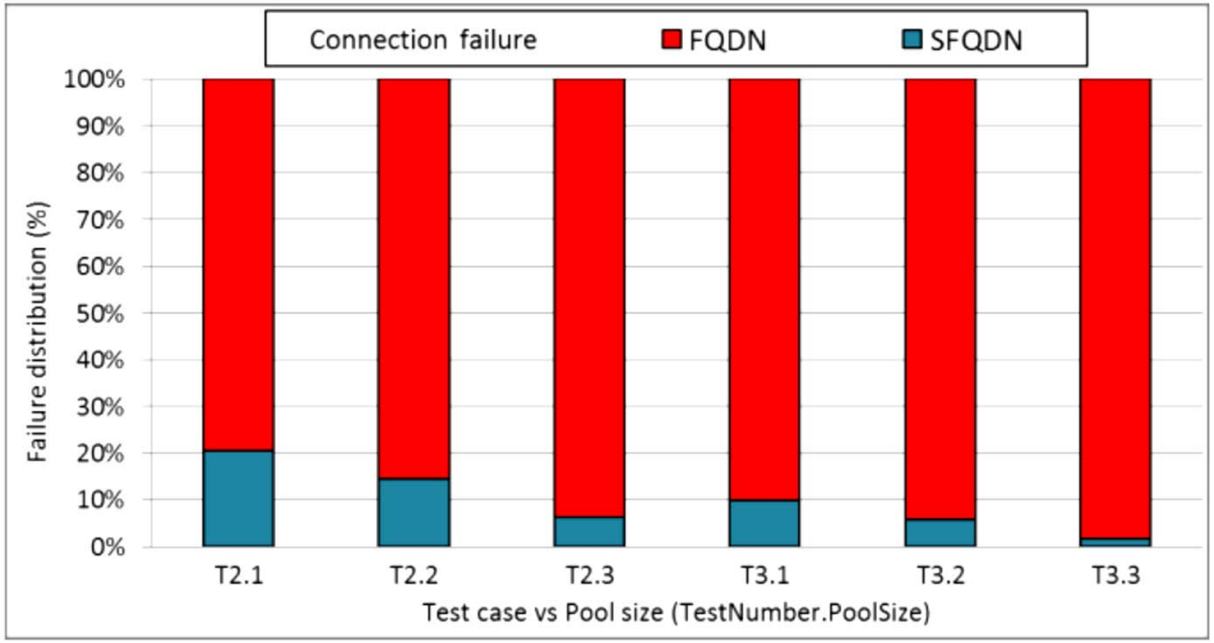


Fig. 4. Connection failure distribution FQDN vs SFQDN

- Test 2: 50% FQDN + 50% SFQDN 5 geometric
- Test 3: 50% FQDN + 50% SFQDN 2x5 geometric

50 ms delay
60 conn/sec

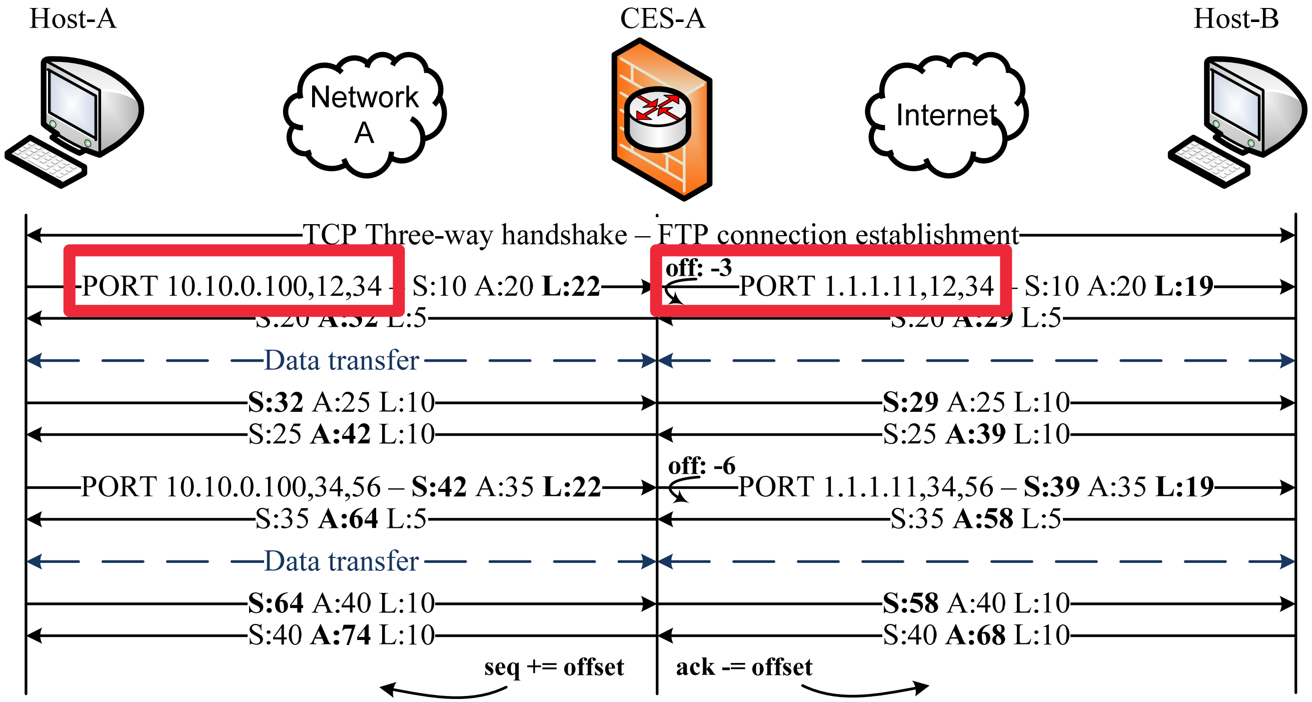
RGW Application Layer Gateway

Application Layer Gateways (ALG) developed for the following protocols

- ICMP and ICMP error packets
  - Address transformation when traversing the node
- UDP based SIP – Session Initiation Protocol
  - Replacement of IP address literals by FQDN
  - Create inbound mapping for media streams
- TCP based FTP – File Transfer Protocol
  - Replacement of IP address literals by FQDN
  - Create inbound mapping for data streams
  - Introduces an offset in subsequent TCP segments (SEQ, ACK)
- TCP based RTSP - Real Time Streaming Protocol
  - Replacement of IP address literals by FQDN
  - Create inbound mapping for media streams
  - Introduces an offset in subsequent TCP segments (SEQ, ACK)

RGW Application Layer Gateway

FTP Case – Stateful ALG with TCP header rewrite



$$\text{Offset} = \text{Length}_{new} - \text{Length}_{original} + \Delta\text{Offset}$$
$$\text{ACK}_{new} = \text{ACK}_{current} - \text{Offset}$$
$$\text{SEQ}_{new} = \text{SEQ}_{current} + \text{Offset}$$

RGW Application Layer Gateway

- Web servers in private hosts are supported by RGW via an HTTP/HTTPS reverse proxy
- RGW redirects all incoming queries containing the prefix www to a single IP address where the proxy is listening
  - hosta.cesa.isp => Uses Circular Pool
  - tcp80.hosta.cesa.isp => Uses Circular Pool on TCP port 80
  - www.hosta.cesa.isp => Uses reverse proxy for hosta.cesa.isp
- Currently, the prototype makes use of Nginx at the private hosts and as the reverse proxy of RGW

Realm Gateway Compatibility

Protocol Compatibility RGW

Application in realm		Protocol	Direction	Result
Private	Public			
Netcat client/server	Netcat client/server	TCP & UDP	Both	Success
Ping request	Ping response	ICMP	Outgoing	Success
Ping response	Ping request	ICMP	Incoming	Success
-	Ping req / Dig	DNS & ICMP	Incoming	Success
NTP client	NTP server	UDP	Both	Success
SSH client/server	SSH client/server	TCP & UDP	Both	Success
Skype	Skype	TCP & UDP	Both	Success
Traceroute	Traceroute	ICMP error	Both	ALG
HTTP client	HTTP server	TCP	Outgoing	Success
HTTP server	HTTP client	TCP	Incoming	Proxy
FTP client	FTP server	*Active mode*	Outgoing	ALG
FTP client	FTP server	*Passive mode*	Outgoing	Success
FTP server	FTP client	*Active mode*	Incoming	Success
FTP server	FTP client	*Passive mode*	Incoming	ALG
SIP client/server	SIP client/server	UDP	Both	ALG

Extra 1: RGW64 Transition to IPv6

Transitions mechanisms defined by IETF – RFC.4213

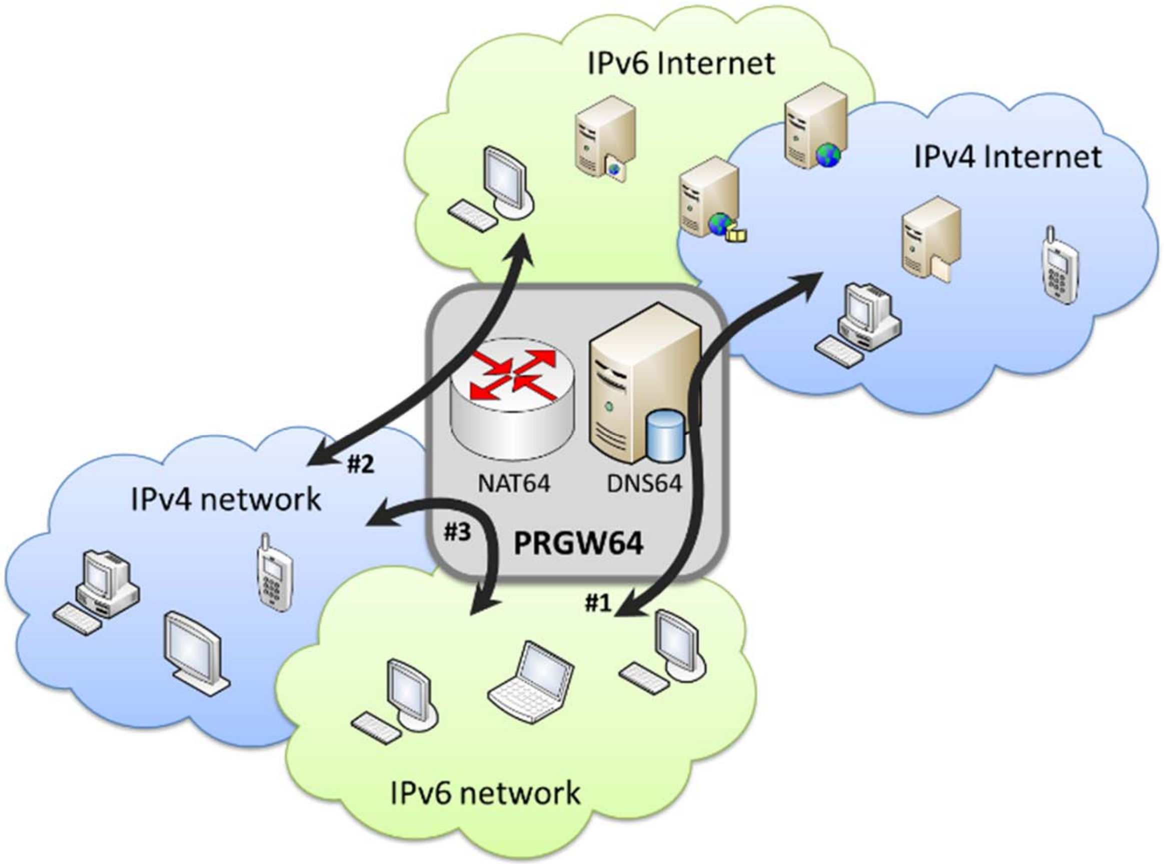
- Dual-Stack
  - Requires both IPv4 and IPv6 hosts and networks
  - Gradual migration towards IPv6-only

- Tunnelling
  - Using IPv6 links over currently deployed IPv4 networks
  - Examples: 6to4, 6rd, ISATAP, Teredo, etc.

- *Translation*
  - Framework for IPv4/IPv6 Translation – RFC.6144
  - Defines protocol translations for IP and DNS – NAT64/DNS64
  - Examples: Microsoft, Cisco, Juniper, Ecdysis, TAYGA, ISC Bind
  - Experiments: J. Arkko and A. Keranen, Experiences from an IPv6-Only Network RFC.6586

Extra 1: RGW64 Transition to IPv6

Framework for IPv4/IPv6 Translation – Use Cases



Extra 1: RGW64 Transition to IPv6

Design choices: Stateless vs Stateful

NAT 64 Operating Modes

Stateless NAT64	Stateful NAT64
1:1 IPv6-to-IPv4 translations	N:1 IPv6-to-IPv4 translations
No conservation of IPv4 addresses	Conservation of IPv4 addresses
End-to-end address transparency	Address overloading lacks end-to-end transparency
No state or binding per translation	Requires state and binding per translation
Mandatory IPv4-translatable-IPv6 address assignment	Arbitrary IPv6 address assignment
Requires manual or dynamic host configuration for IPv6 addressing	It can use either manual, dynamic or stateless configuration for IPv6

Extra 1: RGW64 Transition to IPv6

Packet forwarding – NATxx

- Built-in support for all communicating scenarios
  - NAT44: IPv4 private network and IPv4 Internet – Traditional NAT
  - NAT46: IPv4 private network and IPv6 Internet
  - NAT64: IPv6 *private* network and IPv4 Internet
  - NAT66: IPv6 *private* network and IPv6 Internet

Address synthesis of DNS records – DNS64

- Borrow the concept of IP proxy address from CES
  - Proxy IPv6 addresses as Unique Local Address (ULA)
    - Locally generated so no globally unique (does not need to be!)
    - Prefix fc00::/8
  - Not compatible with Well-Known Prefix (64:ff9b::/96)
    - Cannot be used to represent non-global IPv4 addresses

Extra 1: RGW64 Transition to IPv6

Example: IPv6 host to IPv4 Internet via CES/RGW

1. Hosts sends DNS AAAA query to aalto.fi
2. CES issues NAPTR query to aalto.fi
   – NAPTR resolution fails => there is no CES service available
3. RGW issues AAAA query to aalto.fi
   – AAAA resolution fails => there is no IPv6 service available
4. RGW issues A query to aalto.fi
   – A resolution succeeds=> 130.233.224.254
5. RGW answers DNS query with a proxy IPv6 ULA
   – Additional connection state is created for NAT64
6. *Private* IPv6 host can connect to IPv4 Internet

Extra 1: RGW64 Transition to IPv6

Example: IPv4 host to IPv6-only Internet via CES/RGW

1. Hosts sends DNS A query to ipv6.cybernode.com
2. CES issues NAPTR query to ipv6.cybernode.com
   – NAPTR resolution fails => there is no CES service available
3. RGW issues A query to ipv6.cybernode.com
   – A resolution fails => there is no IPv4 service available
4. RGW issues AAAA query to ipv6.cybernode.com
   – AAAA resolution succeeds=> 2001:470:1:1b9::31
5. RGW answers DNS query with a proxy IPv4
   – Additional connection state is created for NAT46
6. *Private* IPv4 host can connect to IPv6-only Internet

Extra 1: RGW64 Transition to IPv6

Protocol Compatibility RGW64

Application in realm		Protocol	Direction	Result
Private	Public			
Netcat 4/6 client/server	Netcat 4/6 client/server	TCP & UDP	In & Out	Success
Ping/Echo 4/6	Ping/Echo 4/6	ICMP	In & Out	Success ALG
Traceroute 4/6	Traceroute 4/6	ICMP Error	In & Out	Success ALG
NTP client 4/6	NTP server 4/6	UDP	Out	Success
SSH 4/6 client/server	SSH 4/6 client/server	TCP	Both	Success
HTTP client 4/6	HTTP server 4/6	TCP	Outgoing	Success
HTTP server 4/6	HTTP client	TCP	Incoming	Success Proxy
FTP 4/6 client/server	FTP 4/6 client/server	TCP	Both	Untested ALG
SIP client/server	SIP client/server	UDP	Both	Untested ALG
Skype	Skype	TCP & UDP	Both	Fail

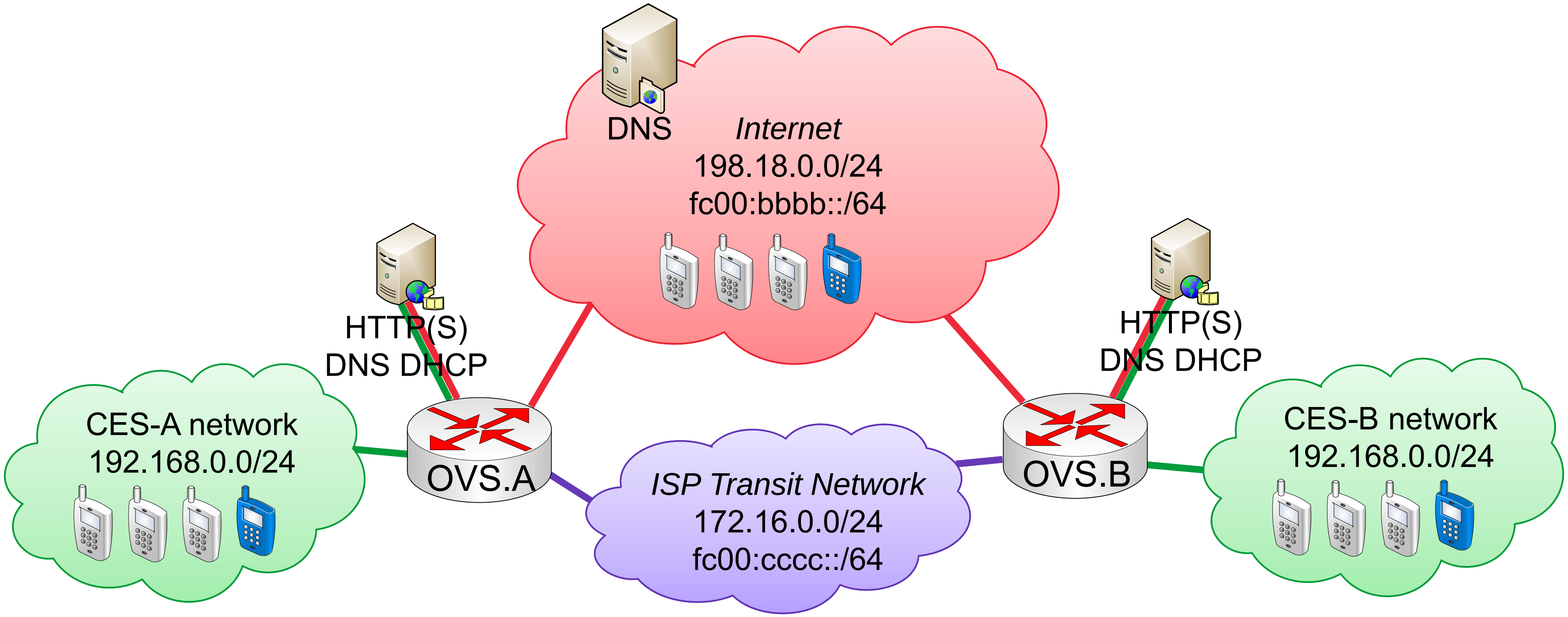
Extra 2: Development Architecture

Current testbed relies on Proxmox VE 3.4

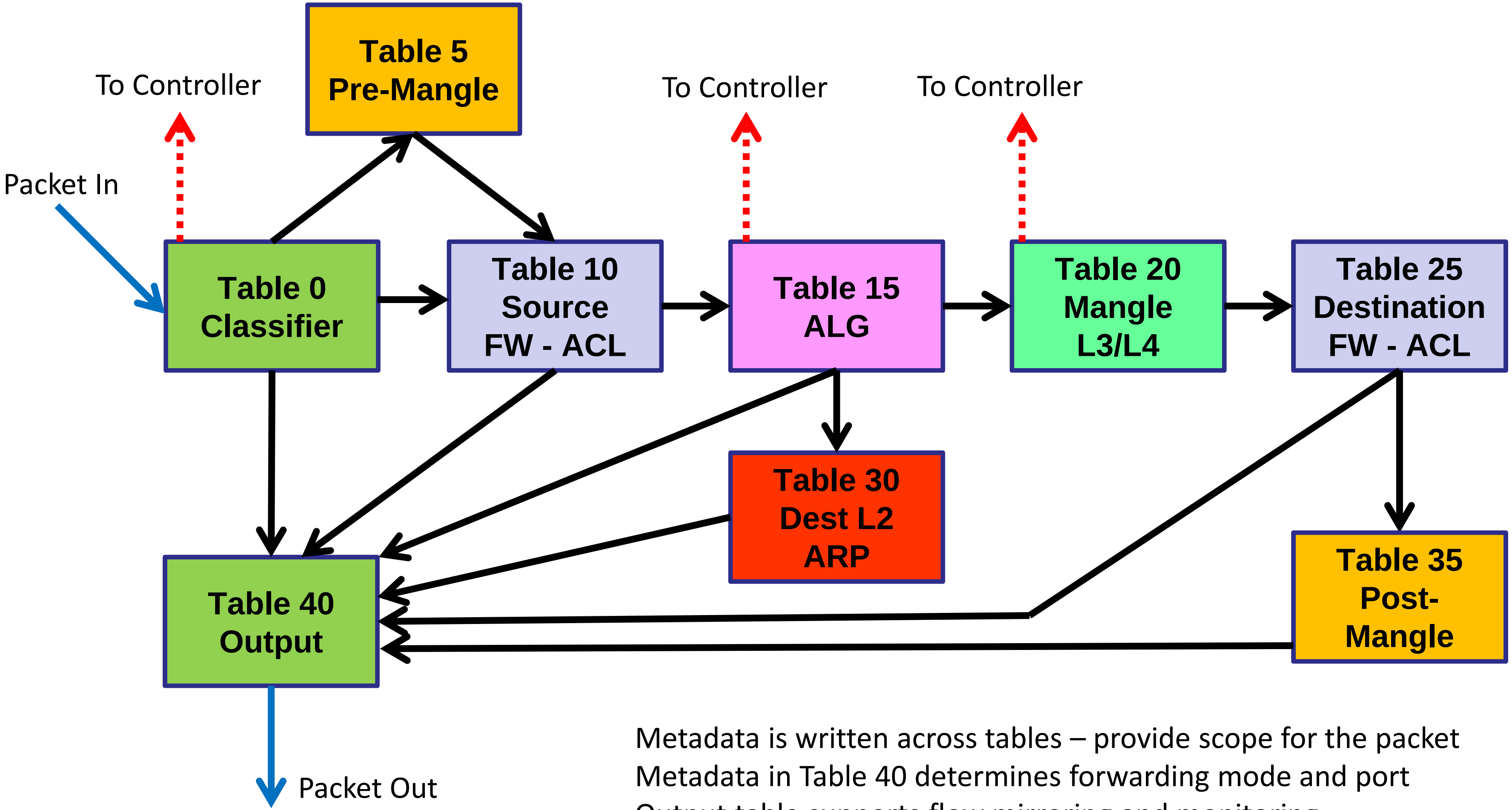
- Supports both KVM and containers with OpenVZ
- Containers are more lightweight compared to full-blown VM
- Available at http://proxmox.com/en/proxmox-ve

- Our whole testbed sits on a single VM running Proxmox
  - All hosts and nodes are virtualized with containers
  - Includes kernel support for OpenvSwitch
  - Networking scenario is made of:
    - Linux bridges
    - OpenvSwitch bridges
    - Virtual Ethernet pairs

Extra 2: Development Architecture



Extra 3: OpenFlow Tables



Metadata is written across tables – provide scope for the packet
Metadata in Table 40 determines forwarding mode and port
Output table supports flow mirroring and monitoring

Extra 4: DNS Relay

- Behaves as a *regular* DNS forwarder
- Supports IPv4 & IPv6 and UDP & TCP
- Encodes meta information in "*Additional Records*"
  - Sender's IP address, port and protocol
- Defines zones for message forwarding
- Alleviates congestion in signalling channel of the OpenFlow switch (data path)
  - Reduces the PacketIn events received by SDN Controller
  - Reduces the parsing required by SDN Controller
  - SDN application can receive DNS messages directly via socket

Extra 5: Future ALG System

- ALG design principles:
  - Addressing: Traversing realms requires address translation – Use FQDN if possible! e.g. SIP
  - Connection State: TCP-based protocols that require payload modifications require state to track the introduced offset and modify the TCP header of following segments
- Interworking with OpenFlow switch and OpenFlow Table
- Signalling channel with SDN CES/RGW application
  - Request connection or host information
  - Establish new bindings in the NATxx
- Stores own connection table for specific ALG flows
  - Any stateful ALG is required to maintain it's own state
  - FTP & RTSP modify user payload introducing offsets in data sent