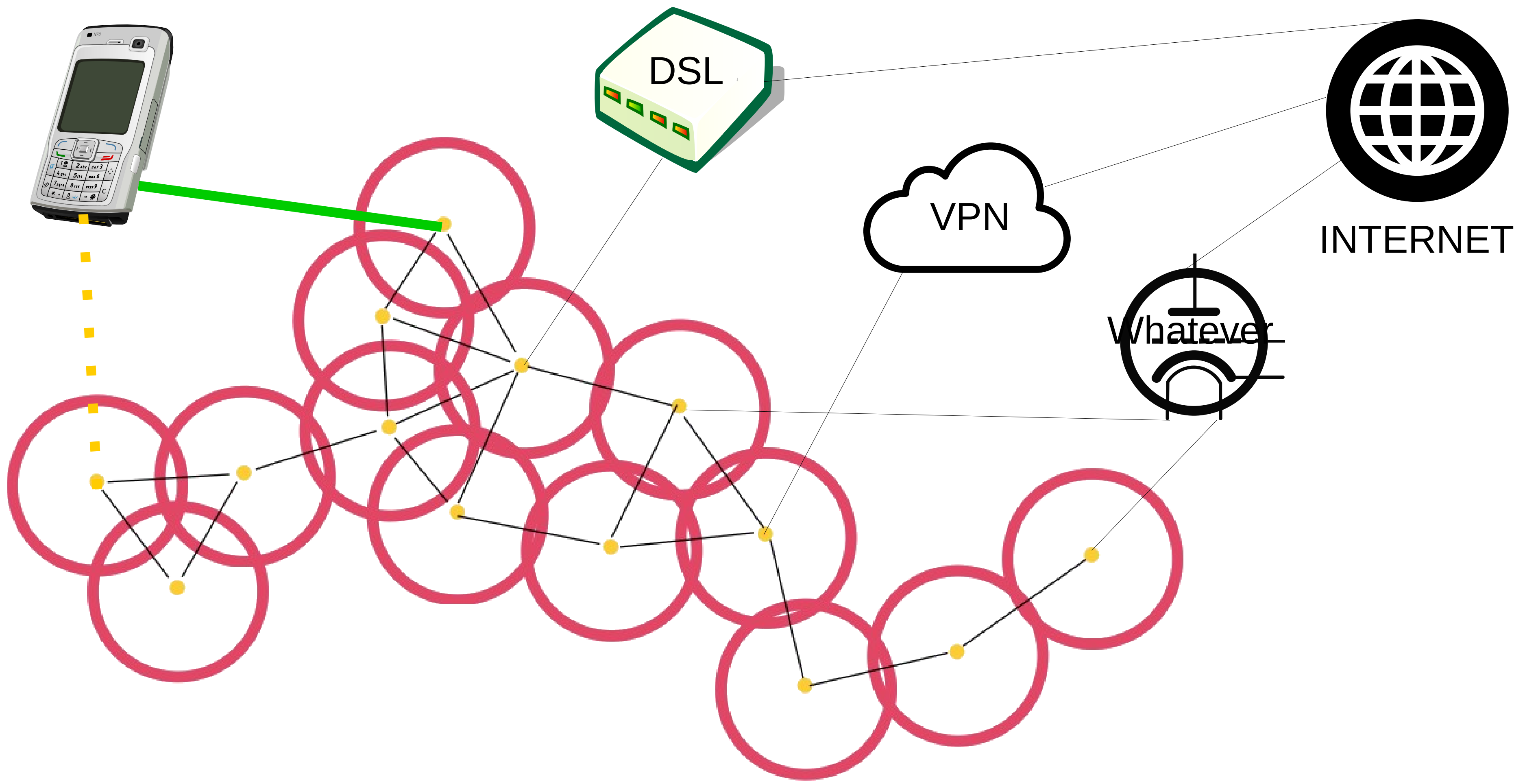
Roaming in Babel mesh networks



Roaming in Babel mesh networks

Prerequisites: What you should know about

- What mesh networks are ;-)
- How IPv6 works
- How Babel routing works
- (would be great) How L3roamd works in detail
  Please watch the talk “Short Status for L3roamd and Babel Integration in Gluon” for that

Status

This is a draft so some things might still change, but most of the tools used have already been implemented and IPv6 roaming is already working.

Roaming in Babel mesh networks

Services we created to make it work

- L3roamd
  detects clients associating with an AP and sets routes accordingly
- prefixd
  allows IPv6 segmentation of the network using Babel source routing
- mmfd
  allows multicast for management traffic via L3 networks by translating packets to unicast and distributing them to neighbour nodes
- NAT426
  Specialized kernel module used to translate IPv4 packets to IPv6
- DDHCPD
  Distributed DHCP server

Roaming in Babel mesh networks

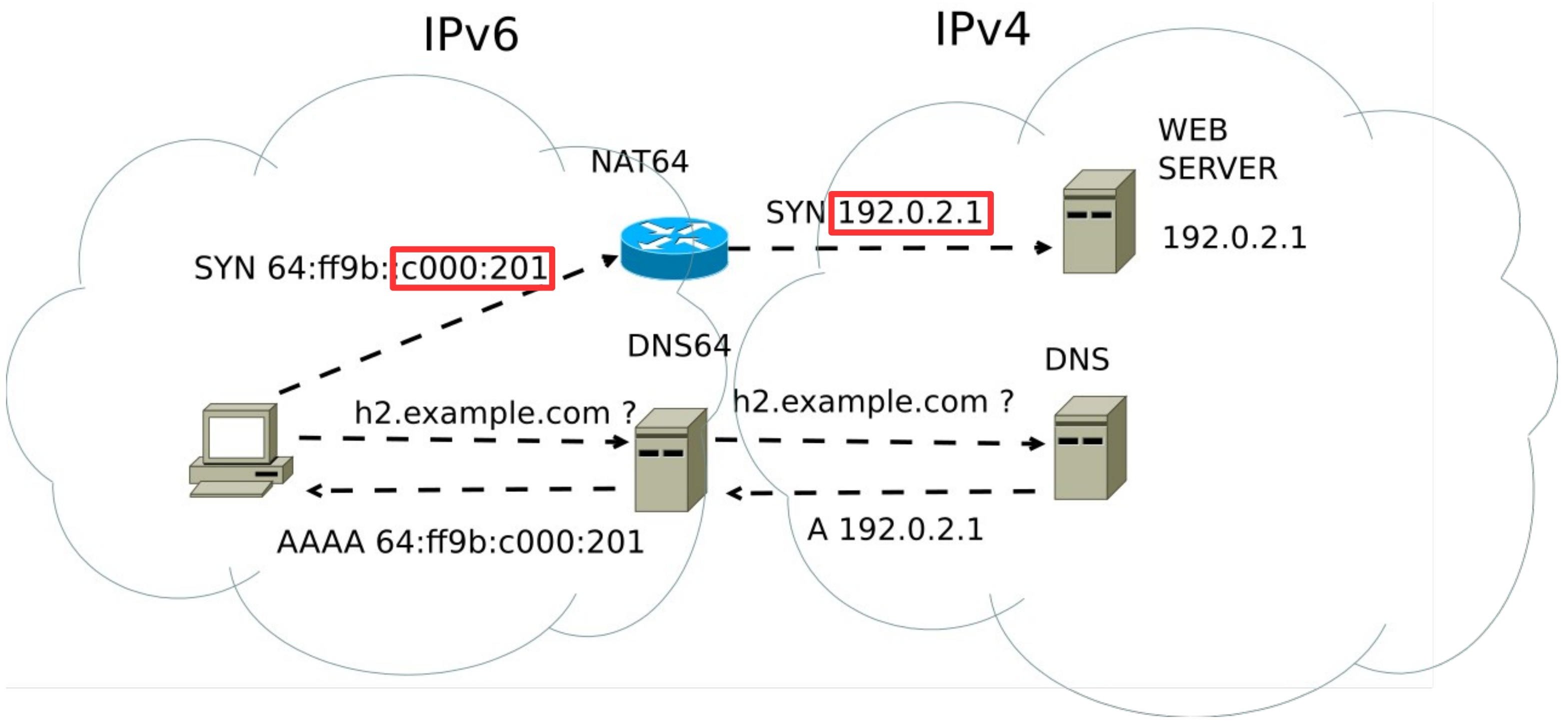
L3roamd

1) When a client associates with a node a special IPv6 ULA address called “Client-Node address” is being calculated from the clients MAC address and the node tries to send a CLAIM packet to this address

Case 1 - The client was associated with another node before
2) The old node is listening on this address and replies with an INFO packet which contains the IPv6 addresses the client uses and drops the routes of the client and the Client-Node-address
3) When the new node receives the INFO packet it sets the routes of the client IP addresses and sets the Client-Node-address pointing to the node itself
Case 2 - The client is new to the network
2) The client gets IP addresses by router advertisement or DHCP and starts sending traffic to e.g. one of the gateways
3) E.g. the gateway does not know a route to the client and thus propagates a SEEK packet trough the network
4) If one of the nodes discovers this client address using NDP it sets the route and the neighbour table accordingly
In all cases Babel distributes the routes automagically

NAT64

- is an IPv6 transition mechanism
- allows access to IPv4 services over IPv6-only networks



Roaming in Babel mesh networks

NAT64

Limitations for IPv6-only clients

Protocols that embed IPv4 literal addresses don't work
(e.g. SIP, SDP, FTP, WebSocket, Skype, MSN and poorly written HTML)

How to overcome this limitation?

The client's software needs to think it is talking directly to the IPv4 server

What is the common solution for this problem?

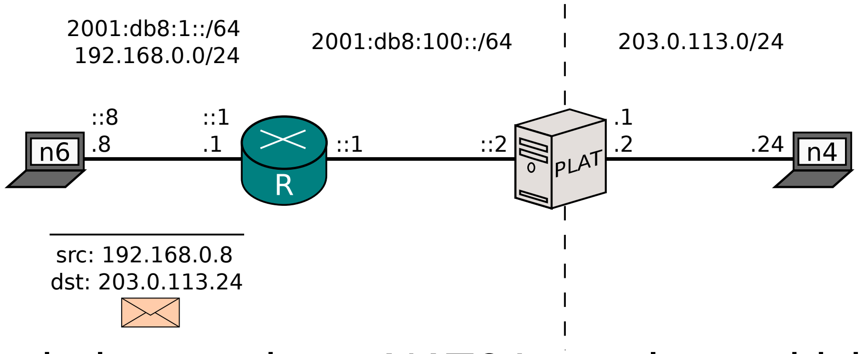
It is called **464XLAT** and was described in **RFC6877**

Roaming in Babel mesh networks

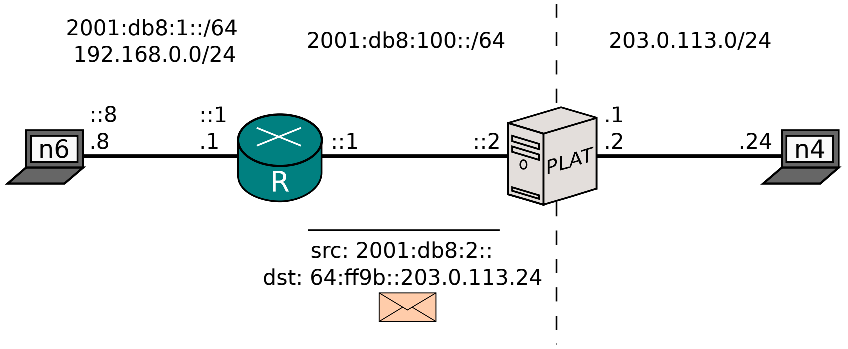
464XLAT (we don't use it but you should understand it to understand NAT426)

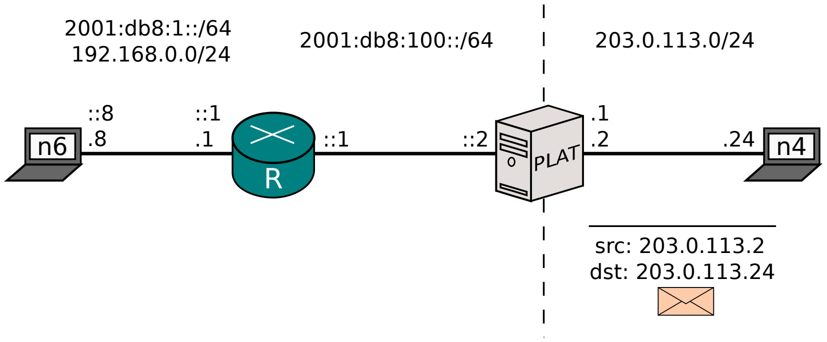
Packet flow (client request)

1) The client's router uses SIIT to translate IPv4 packets into IPv6
   (btw SIIT is also called stateless NAT46)



2) Packets are being send to a NAT64 translator which translates them back into IPv4



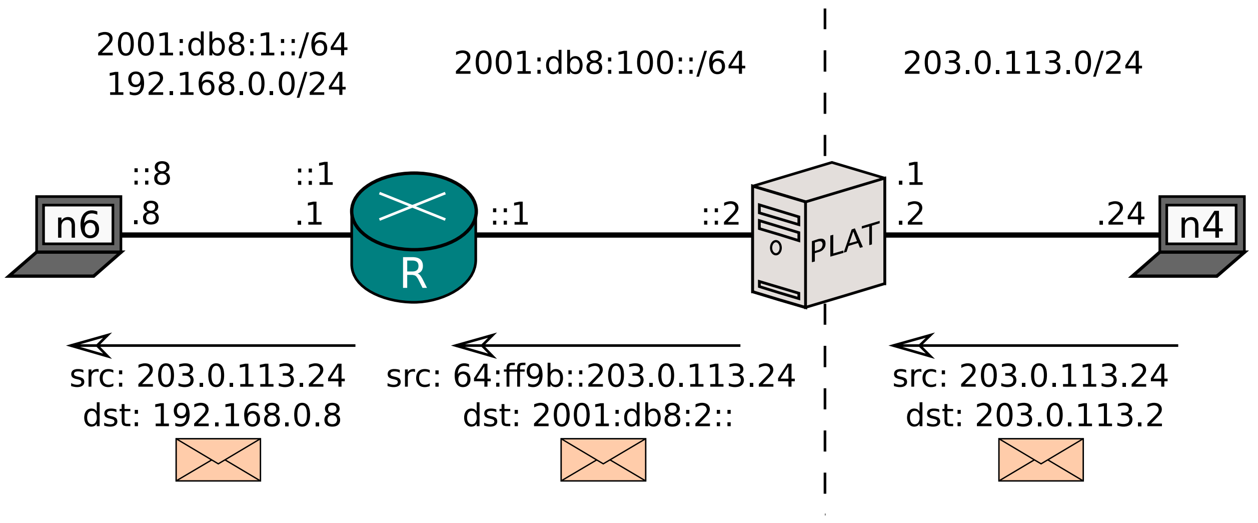


Roaming in Babel mesh networks

464XLAT

Packet flow (server reply)

3/4) The response is being translated to IPv6 by the NAT64 translator
The SIIT translator translates the response to IPv4



Limitations

Only UDP, TCP, and ICMP work (but we can live with these limitations)

NAT426

Is very similar to 464XLAT, but

- it uses the client's MAC to calculate the source IPv6 address in the same format as the Client-Node-address. Thus no additional route needs to be propagated.
  Using source routing the node can recognize if a packet is destined for the client or L3roamd.
- it can dynamically reply to ARP requests to “fake” an IPv4 network
- it can optionally answer DHCP requests on its own (but that is a different story)

Possible approaches for IPv4 support

- Not supporting legacy IPv4, but only using NAT64
  Contra: Breaks services like FTP

- Using only NAT426 and its DHCP functionality
  Contra: Clients can't communicate with each other via IPv4 as they share one address
  Pro: No additional route for client IPv4 addresses needs to be propagated

- Not using NAT426, but assigning IPv4 addresses to clients and gateways using DDHCP
  Contra: To allow IPv4 routing every node needs to have a dedicated IPv4 address, we end up with many routes and need a big IPv4 range

- Using DDHCP for clients and gateways, but translating packets to IPv6 using NAT426
  Contra: none?!
  <u>Thus we decided to use this approach for our initial implementation</u>

Our use of NAT64

If a node wants to share e.g. its DSL line or a VPN, it is running a NAT64 instance.

Every exit in the network needs a dedicated NAT64 prefix.
Thus we introduced something called a NAT64 metaprefix which is shared by all exit nodes in an L3 segment.
It requests an IPv4 address using DDHCP in a dedicated range for exits.

The format of the NAT64 prefix in the setup described here is:

Metaprefix:IPv4ofExit::/96

NAT426 can reply to ARP requests for the IPv4 range of the exit nodes.
It replies with a MAC address which OUI is dedicated to exit nodes with the IPv4 address of the exit node appended.

If NAT426 receives a packet from the client destined for a MAC with this special OUI it extracts the IPv4 address and can thus translate it to the correct NAT64 prefix.

Roaming in Babel mesh networks

What did we accomplish?

- Local exit of traffic

- Seamless roaming of clients
  - supporting legacy IPv4 clients
  - without special setups on the client side
  - (optional) allowing client to client communication via IPv4
    (the technique is explained in detail by the talk “Distributed DHCP Daemon” afterwards)
  - without TCP connections breaking (in most cases)

- Possibility to segment the network to conquer scaling issues
  (using prefixd, but that is another story)

Any questions?