IPv6 Only

What happens if a host only has IPv6 connectivity?

Brian Carpenter
The University of Auckland
January 2011

Outline

- Why?
- What?
- How?
- When and where?

Why?

- Layer 8
  - Some very big operators believe that OPEX will be lower if they run IPv6-only infrastructure ASAP.
    - Avoid OPEX of dual stack and/or tunnels.
    - Avoid massive scale deployment of double NAT for IPv4 *at the same time as* deploying IPv6.
    - Reduce footprint/power in mobile devices.
  - *Example:* "T-Mobile USA has launched an IPv6 beta with Nokia phones. The service is IPv6-only + NAT64 / DNS64. The beta service has been up for over 6 months with positive feedback." (Cameron Byrne, 9/2010)
- Layer 9
  - *Example:* CERNET2

What?

- Subscribers and subscriber networks
  - And in particular, seriously big ones, where IPv4 exhaustion is already a fact of life.
- *NOT*
  - Content or application service providers. For them, dual stack service is the only economically rational IPv6 model for the foreseeable future (IMNSHO)
  - Enterprise networks (ditto)
  - Transit and backbone networks (ditto)
- So I will only talk about the subscriber case.

Components

• IP stack in client host	Assumed IPv6
• Socket applications in client host	?
• Access network	} Assumed pure IPv6
• ISP core	
• ISP border	?
• Transit networks	Assumed dual stack
• Application servers	Dual stack or pure IPv4
• p2p peers	??

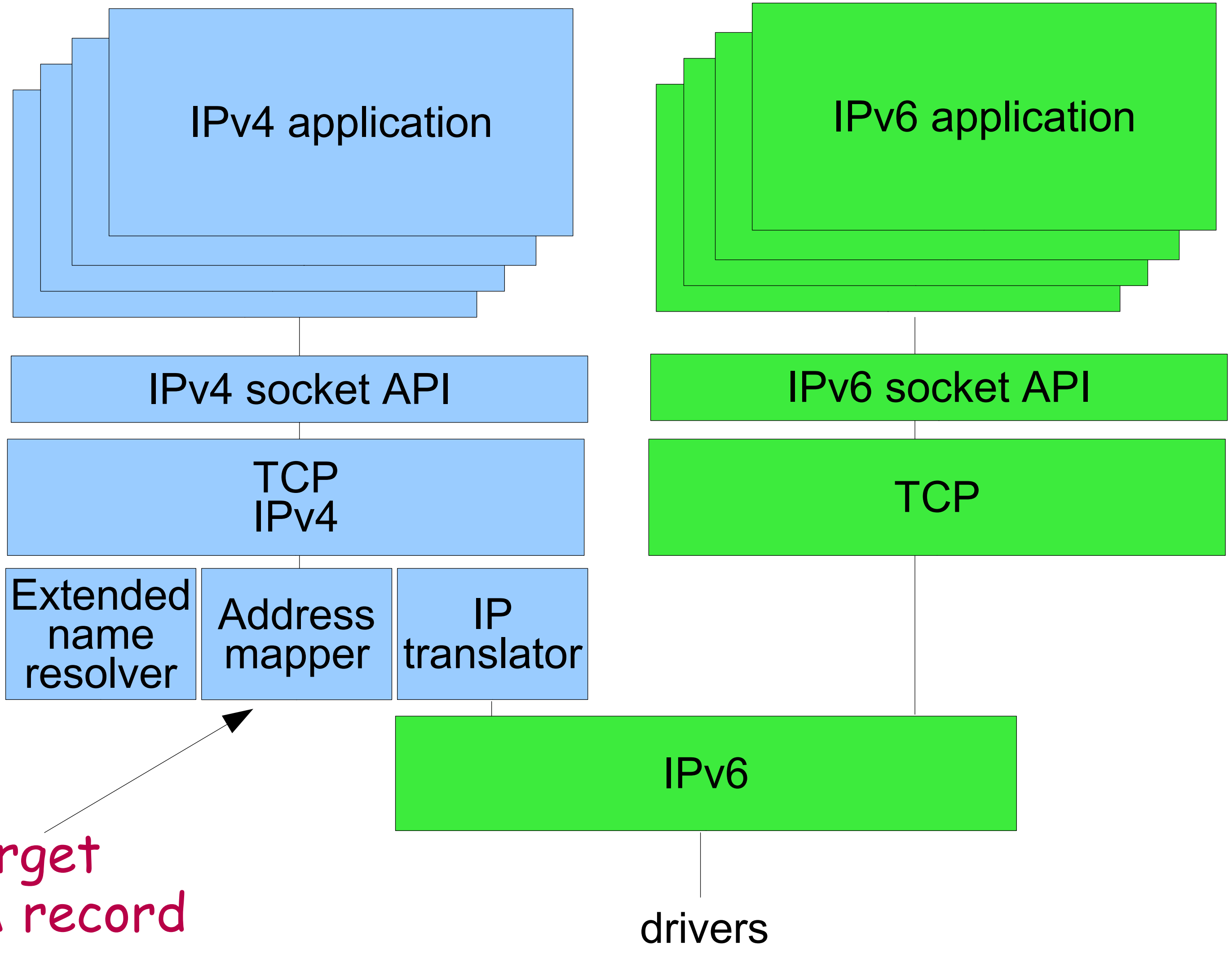
Question marks

- Is the host stack 100% pure IPv6, or does it contain some IPv4 pollution?
  - Opinions differ, but for mobiles the purists are in the majority
- Are the host applications all upgraded to the IPv6 socket API?
  - Ditto
- What do we need at the IPv6/dual stack border?
- Does heterogeneous p2p work somehow??

How (1): Bump in the Host (BIH)

- Iff the host must support legacy apps and the network is pure IPv6, you must pollute the host stack with IPv4.
- IPv6-purist operators want to avoid v4-in-v6 tunnels.
- Therefore they propose a “bump in the host” which is really a form of NAT46 built into the IP stack and API.
  - draft-ietf-behave-v4v6-bih
    (Hint: authors from China Mobile and Nokia)

Schematic (simplified)



Intercepts DNS A queries

Fails if target only has A record

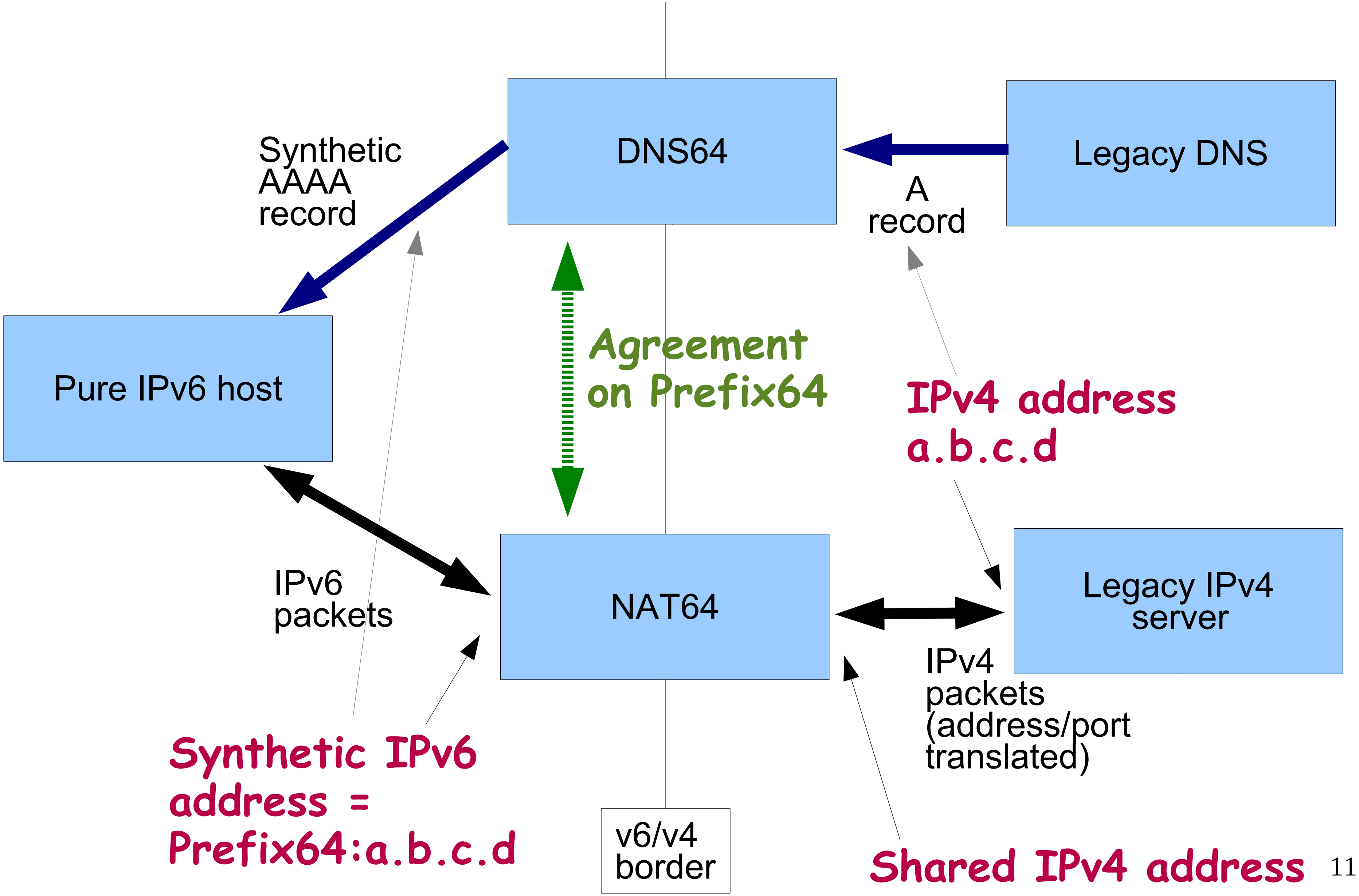
Personal comment

- So there’s the paradoxical failure mode for BIH with a pure IPv6 network: servers that only have A records can’t be reached by v4-only applications
- The suggested way round that is a 4-in-6 tunnel, not allowed in pure IPv6.
- Or, if you’re brave, combine BIH with NAT64, thereby creating NAT464 and DNS464.
- Not for me, thanks.

How (2): NAT64

- Keep the client 100% pure IPv6
- IPv6-only client (no v4 address, no v4 connectivity) needs to initiate communication with an IPv4-only server.
  - No dual stack, no tunnel
- NAT64 comes with a separate DNS64 magic box
  - (Deprecated NAT-PT came with a built-in DNS ALG)

Components


DNS64
Legacy DNS
Synthetic AAAA record
A record
Pure IPv6 host
Agreement on Prefix64
IPv4 address a.b.c.d
IPv6 packets
NAT64
Legacy IPv4 server
IPv4 packets (address/port translated)
Synthetic IPv6 address = Prefix64:a.b.c.d
v6/v4 border
Shared IPv4 address

Sequence of events

- The IPv6 host uses DNS64 as its regular DNS servier to look up names.
  - For native IPv6 hosts, DNS64 returns normal AAAA records.
  - For hosts with A records only, DNS64 concatenates the agreed Prefix64 and the IPv4 address from the A record, and synthesises an AAAA record.
- The IPv6 host just sends normal packets to the synthetic address, which is routed to the NAT64.
  - The NAT64 recognises a new session, extracts the server IPv4 address from the synthetic address, assigns a port on the IPv4 side and other NAT state, and otherwise does its standard NAT thing.
- From an application viewpoint, this looks pretty much like old fashioned NAT44.

What is the Prefix64?

- Prefix64 is normally a /96 (leaving 32 for the IPv4 address)
- Could be a network-specific prefix (NSP) out of the operator's own prefix (one operator controls IPv6 host, NAT64 and DNS64)
- Could be 64:ff9b::/96, an IANA-assigned global WKP (well known prefix) [RFC6052]
  - But then what happens if a synthetic address "escapes" from the scope of the NAT64/DNS64 pair?

Example

- An ISP’s prefix is 2001:db8::/32
- The ISP chooses 2001:db8:122:344::/96 as the prefix for its NAT64.
  - The IGP routes that prefix to the NAT64 box
- IPv6 client queries www.example.com
  - Its A record contains 192.0.2.33
- DNS64 synthesises an AAAA record containing 2001:db8:122:344::192.0.2.33
- NAT64 will algorithmically map that to 192.0.2.33
  - And perform normal stateful NAT port mapping, since its own IPv4 address is shared.

DNSSEC

- Sadly, DNS64 interacts with DNSSEC
- 7 distinct cases are analysed in draft-ietf-behave-dns64. 5 cases work fine.
- The two tricky cases are when a DNSEC-aware DNS64 resolver receives a client query with the DO and CD bits set.
  - That means the client will perform DNSSEC validation itself. This is guaranteed to fail for a synthetic AAAA record.
  - In this case, the client needs to be fitted with its own DNS64 resolver, configured with the correct Prefix64.

Status

- Address formats: RFC 6052
- draft-ietf-behave-v6v4-framework
- draft-ietf-behave-v6v4-xlate
- draft-ietf-behave-v6v4-xlate-stateful
- draft-ietf-behave-dns64
- draft-ietf-behave-ftp64 WG Last Call

} RFC Editor queue

From the IETF viewpoint, NAT64 is a done deal.

When?

- As noted earlier, T-mobile (US) has a trial going
  - “beta service ... over 6 months with positive feedback”
- Viagenie has NAT64/DNS64 code for download
- Just install them, switch off IPv4 routing and DNS-over-IPv4, and see what happens.
- Ericsson: http://ripe61.ripe.net/presentations/140-ripe_rome_jari.pdf
  - “Failure rates through NAT64 are similar to those with dual stack (1% / 2% for IPv4/IPv6 destinations)”
  - “NAT64 introduces a small delay, comparable to router/NAT44 hop”
  - BUT - numerous messaging and gaming applications failed completely (i.e. no v6 support)
  - Details in draft-arkko-ipv6-only-experience

NAT64 performance comparisons at the University of Auckland

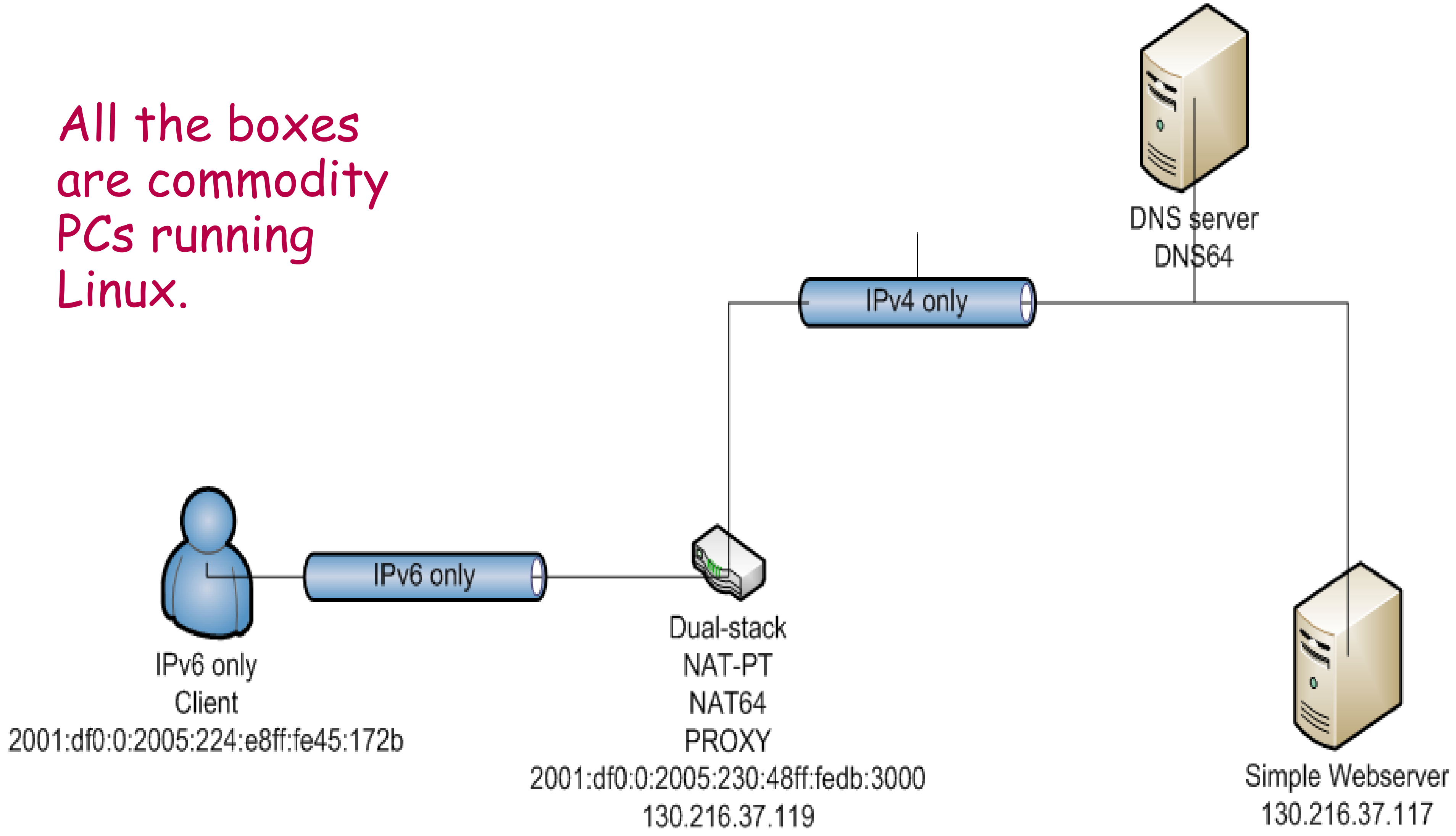
- Report on work done by Se-Young Yu
- Compared 5 scenarios for HTTP access:
  1. native IPv4
  2. native IPv6
  3. NAT64
  4. NAT-PT (deprecated solution)
  5. dual stack HTTP proxy

Experiment description

- A client sends 10000 packets for a connection
- A client establishes 1-100 simultaneous connections.
- A client sends simple or large packet size HTTP requests.
- A Linux router is able to run NAT-PT, NAT64 (Viagenie) or HTTP Proxy (apache web server), as well as forwarding native IPv4 and IPv6.
- Simple apache webserver is deployed as the target.

Setup

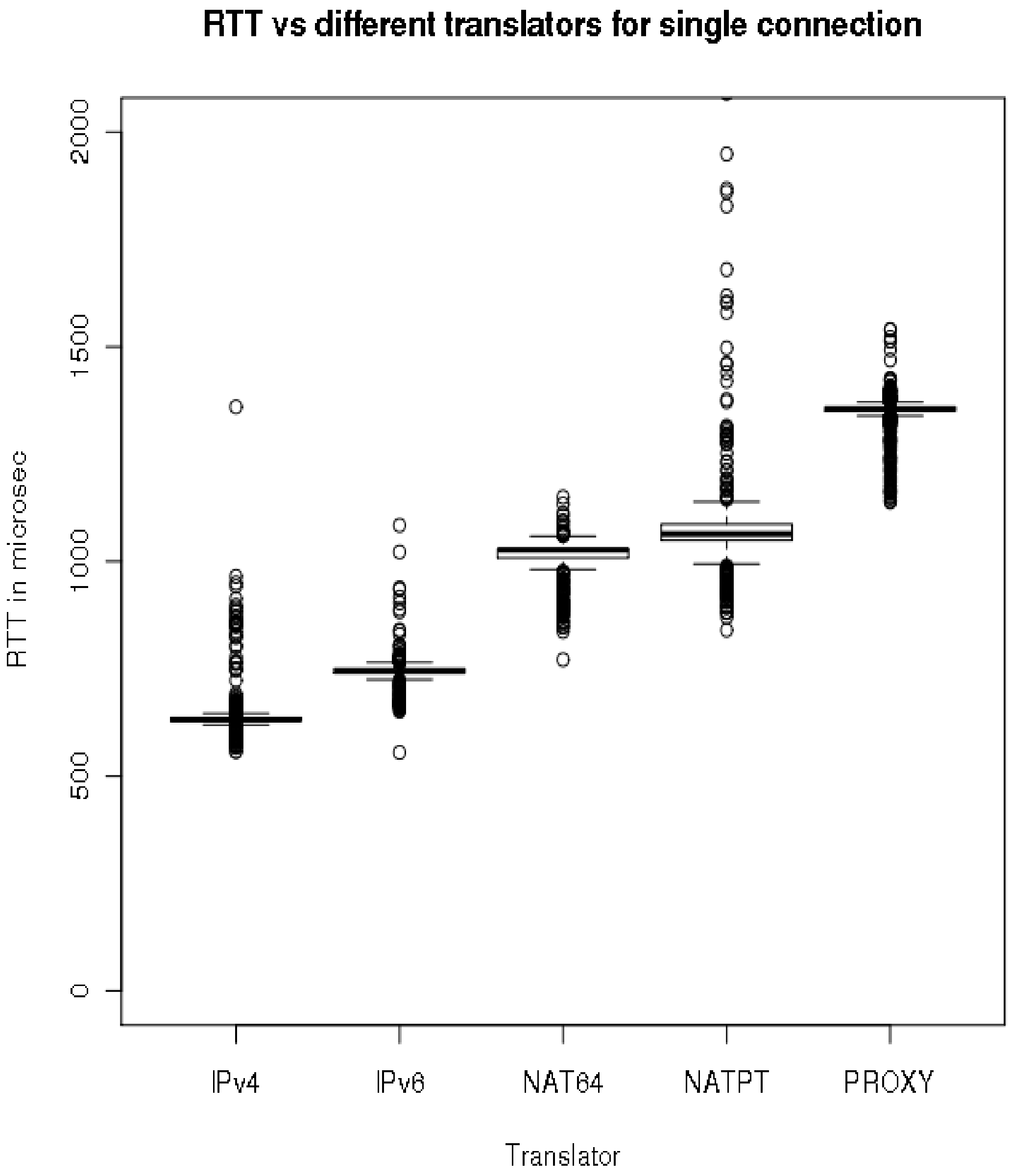
All the boxes are commodity PCs running Linux.



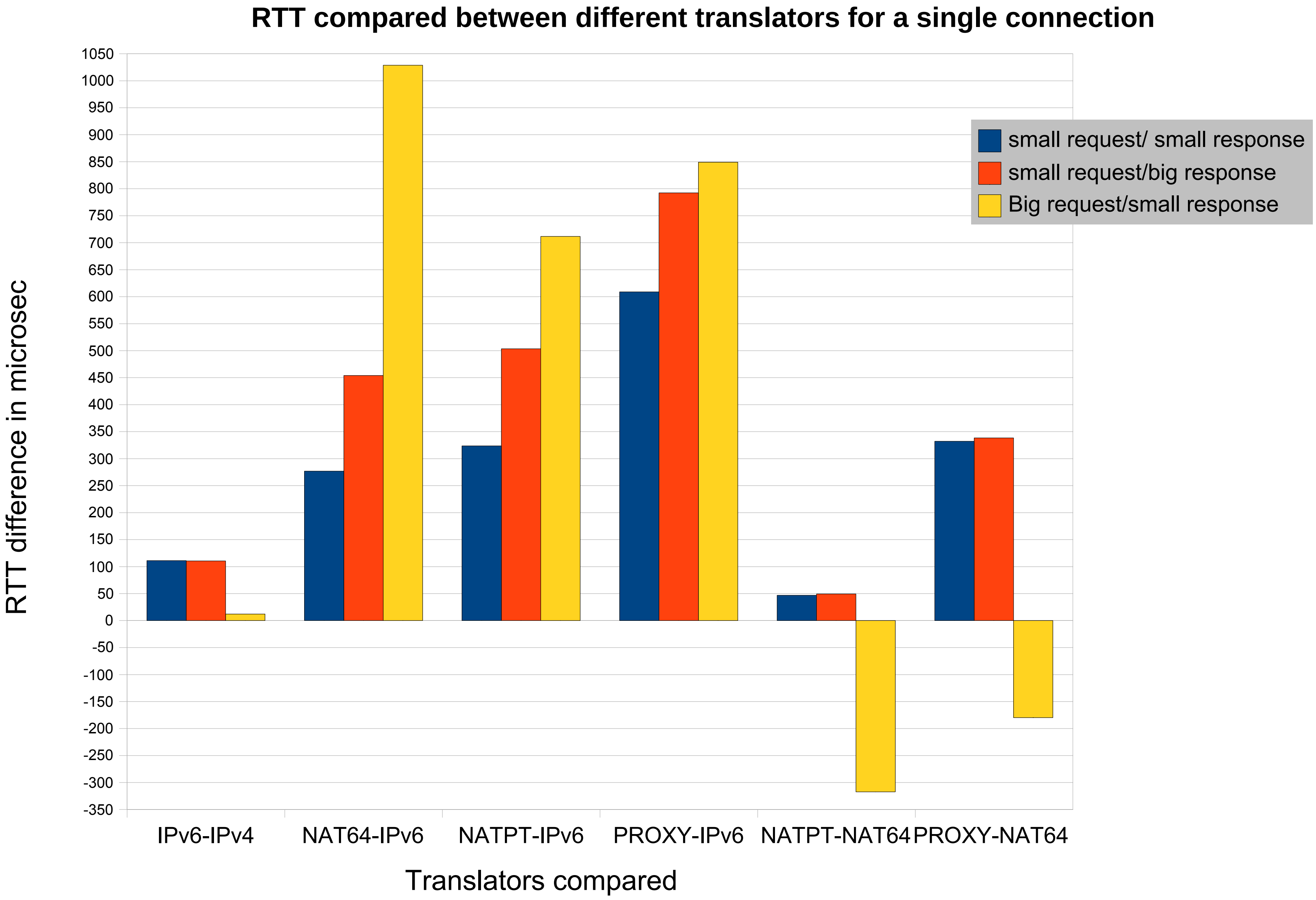
Experiment Results: Simplest case

Median RTT

- Native IPv4 :
  - 631 μsec
- Native IPv6:
  - 745 μsec
- NAT64:
  - 1027 μsec
- NAT-PT:
  - 1064 μsec
- HTTP Proxy:
  - 1355 μsec



RTT differences by packet size



Conclusions

- Considering the Ericsson and UoA results, NAT64 works and has sub-millisecond impact on RTT
  - except for one anomaly which is presumably an artefact of implementation
  - and an HTTP proxy across the v4/v6 boundary is not so bad either
- These are real technologies approaching operational deployability
- Non-HTTP app developers (p2p, messaging, gaming) need to get their IPv6 act together.