DNS A Record Filtering for the migration from dual stack networks to IPv6 only networks

draft-hazeyama-sunset4-dns-a-filter-00

H. Hazeyama, T. Ishihara, O. Nakamura
WIDE Project

Motivation

- In past wide camp experiments, several long timeout / fallback problems were observed when a dual stack host connects to an IPv6 only (wt. DNS64/NAT64) network
  - Those problems provide inconveniences to users
- We wanted to mitigate observed inconveniences

Motivation

- Through wide camp experiences, we developed a work around, so called "DNS A record filtering", to mitigate dual stack users inconveniences
  - draft-hazeyama-widecamp-ipv6-only-experience-02
  - draft-hiromi-sunset4-termination-ipv4-01

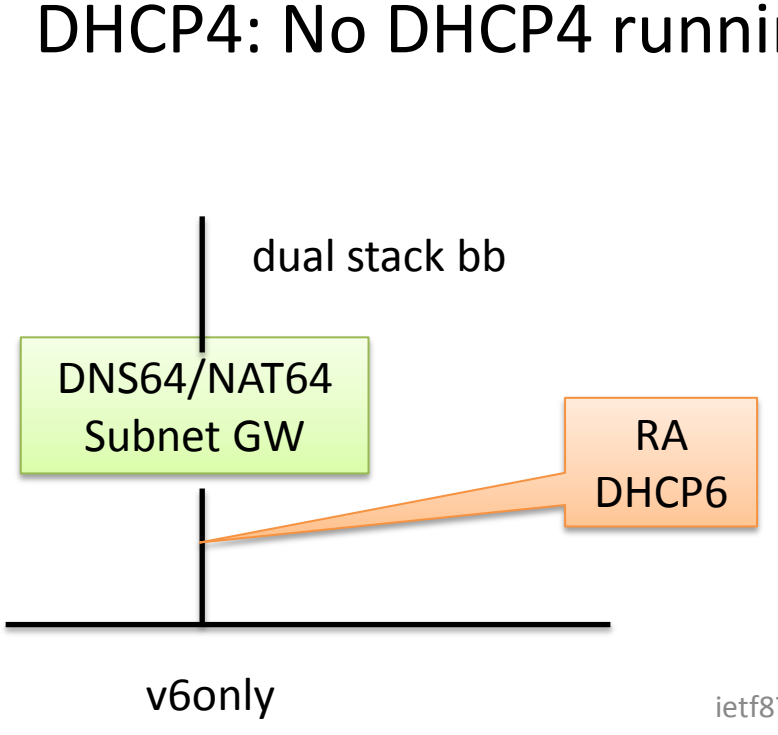
Sunset4 WG in IETF 86 Orland

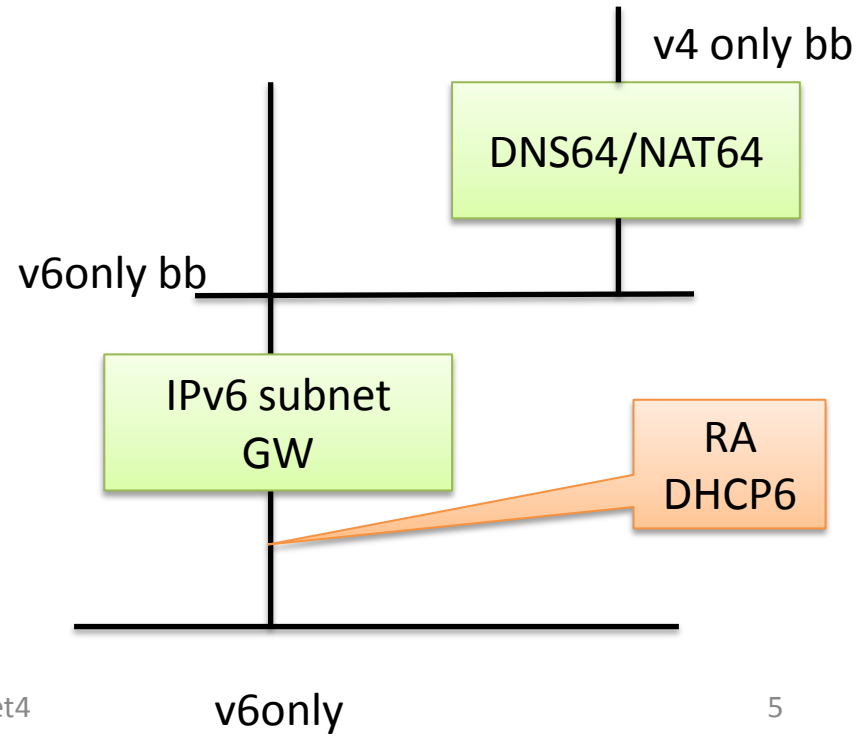
- We presented draft-hiromi-sunset4-termination-ipv4-01.txt

- We got several feedbacks
  - According to the advices, we separate technical part into this draft.

Typical IPv6 only with DNS64/NAT64

- DNS64/NAT64: Map all IPv4 on Internet into IPv6 RA: Enable Other Config (for DHCP6)
- DHCP6: Distribute the DNS64 address
- DHCP4: No DHCP4 running!





Observed Problems

1. Legacy Operating Systems cannot refer DNS properly (WinXP, MacOS Snow Leopard, Android)
   - DNS information should be got via DHCP6, but these OSs did not have DHCP6 function.
   - Android 2.x cannot be configured to use DNS over IPv6 even in manual configuration.

2. Network Settings is not completed in IPv6 only network (iOS5 in Some 3G carrier)
   - “Network Settings” will be completed only if IPv4 address, IPv4 router, and IPv4 DNS can be retrieved via DHCPv4 or manually configured all of these 3.

3. Users are waiting for IPv4 connection timeout (MacOS X)
   - MacOS X implements RFC3927 3.3 (Interaction with Hosts with Routable Addresses), which assumes all IPv4 address are on-link at Link-Local configuration.
   - MacOS X’s implementation of getaddrinfo() prefer IPv4 over IPv6, while Happy-Eyeball is implemented on Objective-C API. So Safari has no timeout problem, but Chrome, Firefox and most other application uses C API faced the problem.

Work around

1. Locating a DNS proxy on the subnet of the IPv6 only network
   - The DNS proxy has a private IPv4 addr and global IPv6 addr

2. Setting DNS A Record filter on the DNS proxy
   - The DNS proxy removes A record on DNS responses

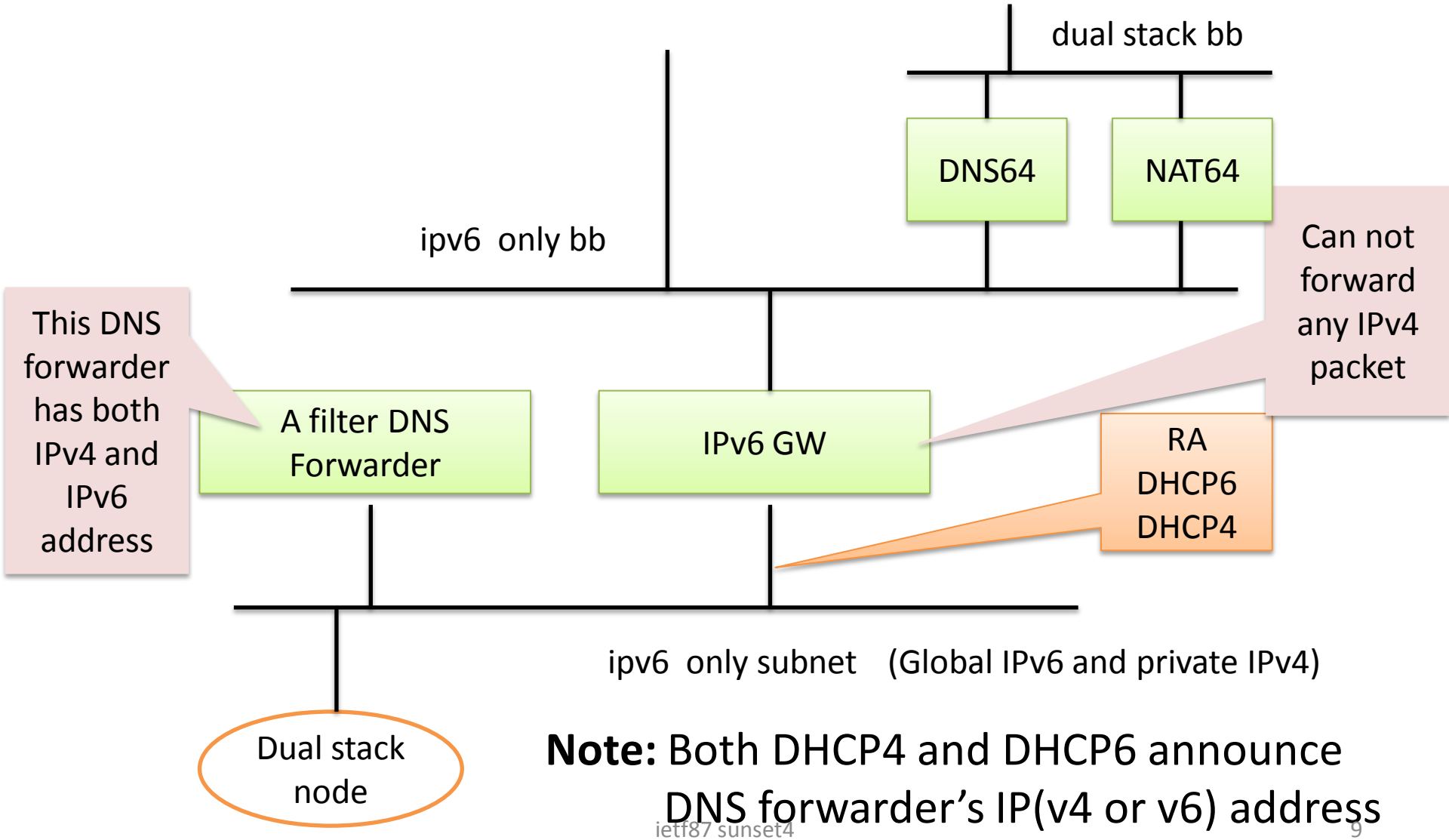
3. Placing a DHCP4 server and DHCP6 server on the subnet of an IPv6 only network
   - DHCP4 provides private IPv4 addr and the IPv4 address of the DNS proxy
   - DHCP6 provides the IPv6 addr of the DNS proxy at least

Effectiveness

1. DNS proxy
   - Provide a DNS lookup method for legacy Operating Systems that do not support DHCP6 or IPv6 DNS function
     - Windows XP, Android 2.x, …
2. DNS A Record filter
   - Mitigate TCP fallback of several Happy Eyeball

3. DHCP4 server
   - Notify the DNS proxy address for legacy Operating Systems
   - Mitigate the fallback by IPv4 link local assumption of Mac OS X

Example of a topology



Note: Both DHCP4 and DHCP6 announce DNS forwarder's IP(v4 or v6) address

Benefits of DNS forwarder with “A” filter

- Enable dual stack in name resolution, however, force single stack (v6 only) in IP forwarding

- Most Operating Systems can reach comfortable IPv6 only life without any changes on Operating System side

Comments from Ted Lemon

- How about the affinity of DNSSEC ?
  - This work around would work well when upper authoritative DNS servers / DNS64 servers set AD bit
    - As well as the answer by Ted Lemon
    - However, we have not tested on an actual implementation, yet

Comments from Ted Lemon

- This solution is seemed a complicated architecture.
  - It may be that I don't understand the motivation for the more complicated problem, but to me it looks like the DNS A record filter proxy is unnecessary? you just need to specify that the DNS64 resolver does A record filtering
- Several reasons
  - A DNS proxy is required to provide name resolution for legacy OSs
  - Of course, A record filtering on DNS64 resolver may work
    - Unfortunately, the current DNS A record filter patch for bind 9.9.2.p1 does not work well with DNS64 configuration
    - The patch is simple, just convert AAAA filter parts of bind to A filter
    - However, we met several unexpected segmentation fault errors
    - The patch works well on a simple DNS forwarder
    - If A record filtering does not affect other functions of DNS (cache, glue, etc.), A record filtering on DNS64 is reasonable

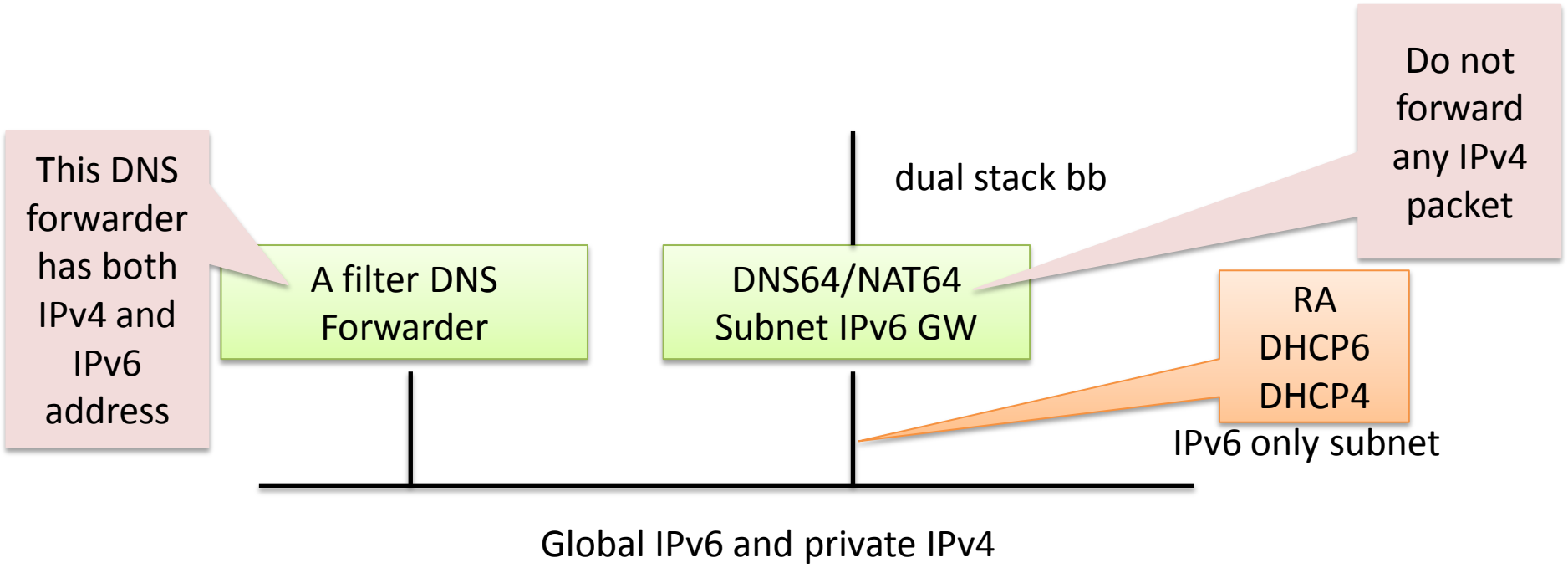
Comments from Ted Lemon

- Re-think the update of RFC 6147 to address the rogue A record problem there ?
  - The problem where your DNS resolver downstream of the DNS64 resolver gets A records as glue, and then returns them as answers to queries from stub resolvers
  - We want to know the definition of ‘glue’. Does it means ‘glue record’ for some NS or not ?

Discussion

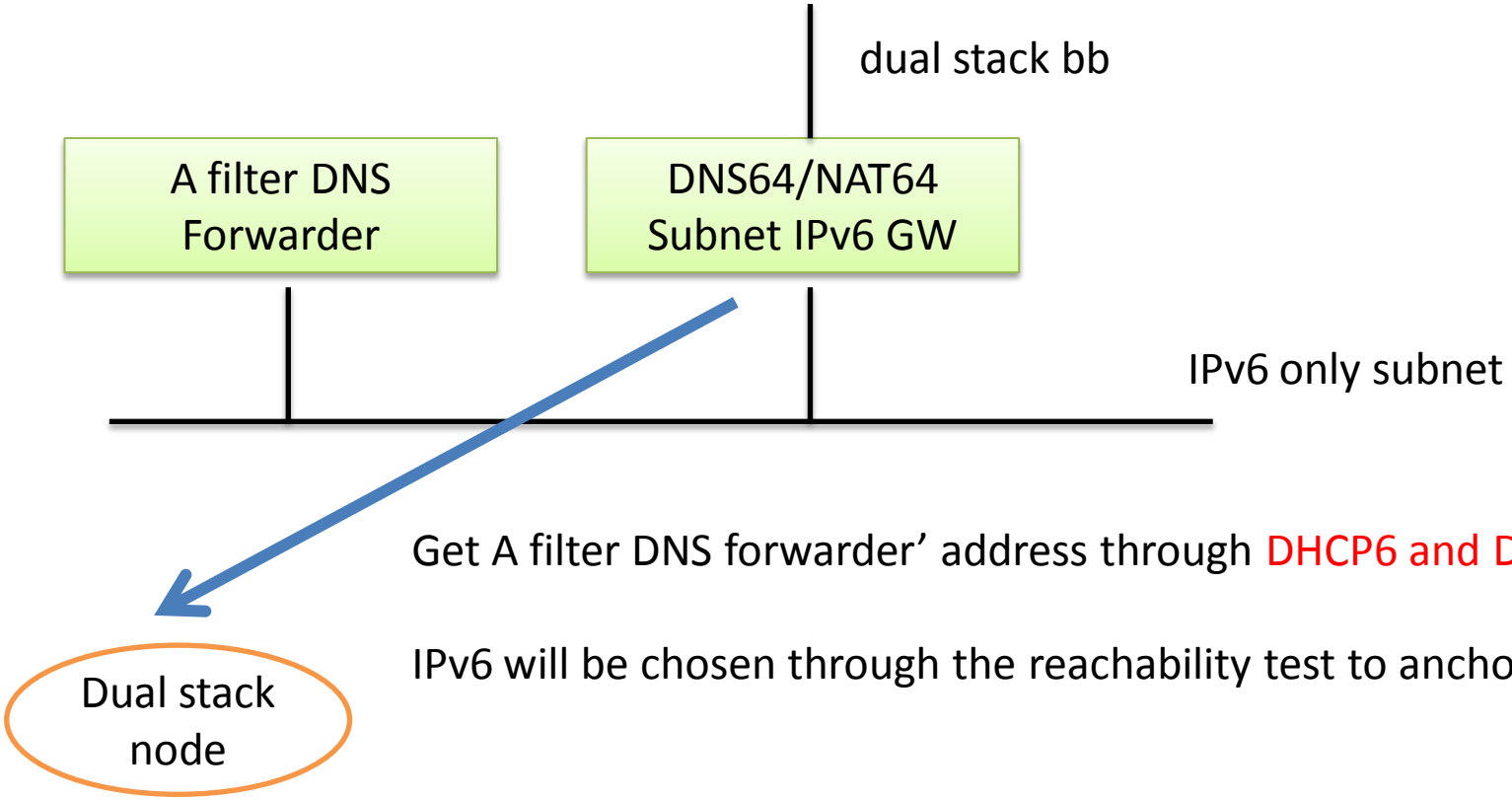
- We think this work around is useful – do you ?
- Adoption as WG item ?

Example of a query sequence



Note: Both DHCP4 and DHCP6 announce DNS forwarder's IP(v4 or v6) address

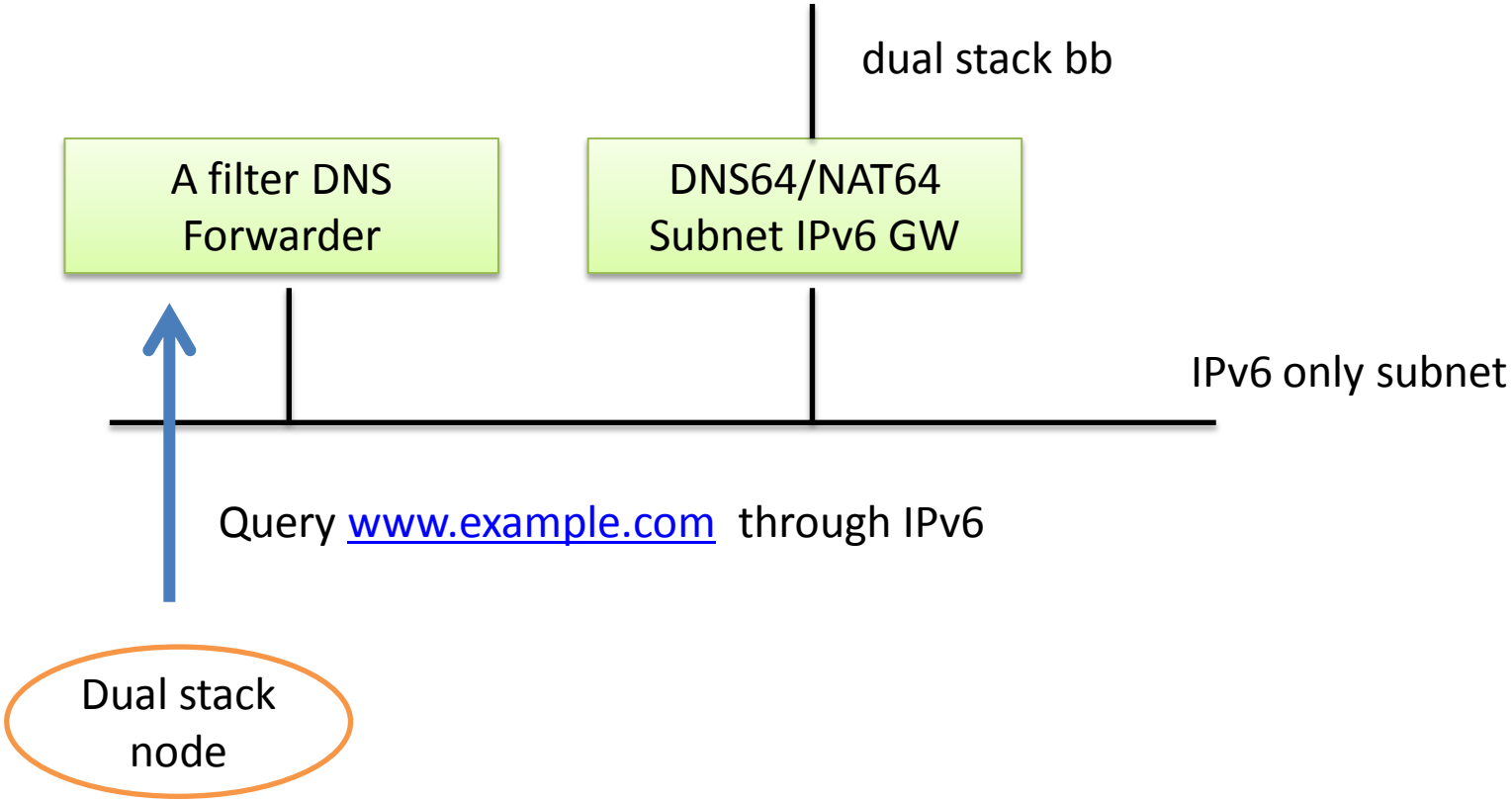
DHCP6 supported dual stack node
(Windows 7,8, Mac OS Lion and later, etc.)



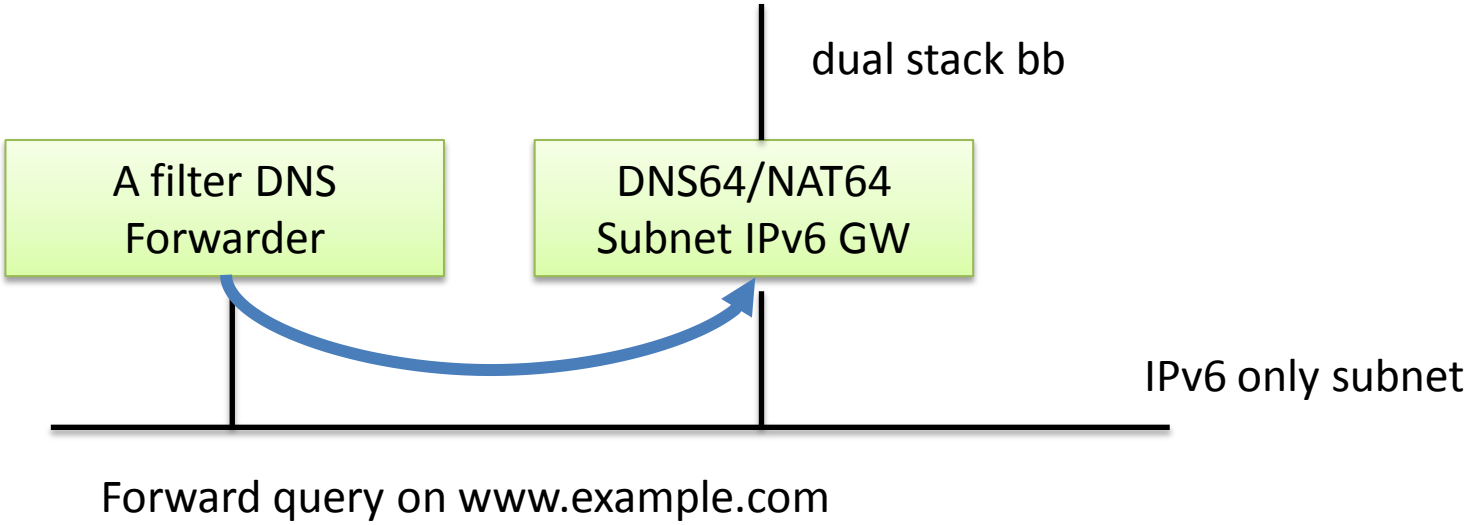
Get A filter DNS forwarder' address through DHCP6 and DHCP4

IPv6 will be chosen through the reachability test to anchor hosts

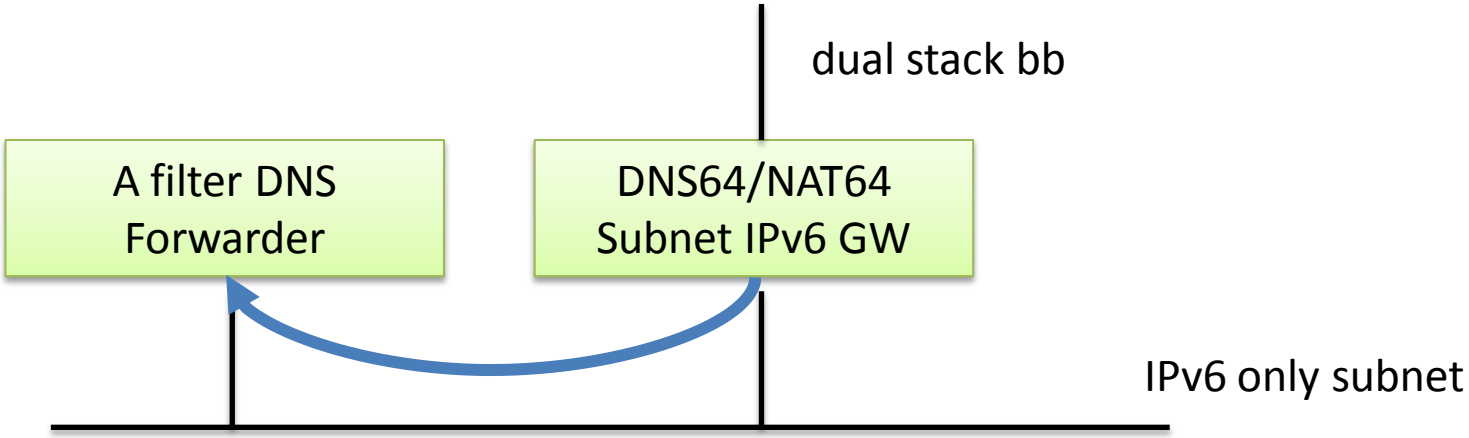
DHCP6 supported dual stack node



DHCP6 supported dual stack node



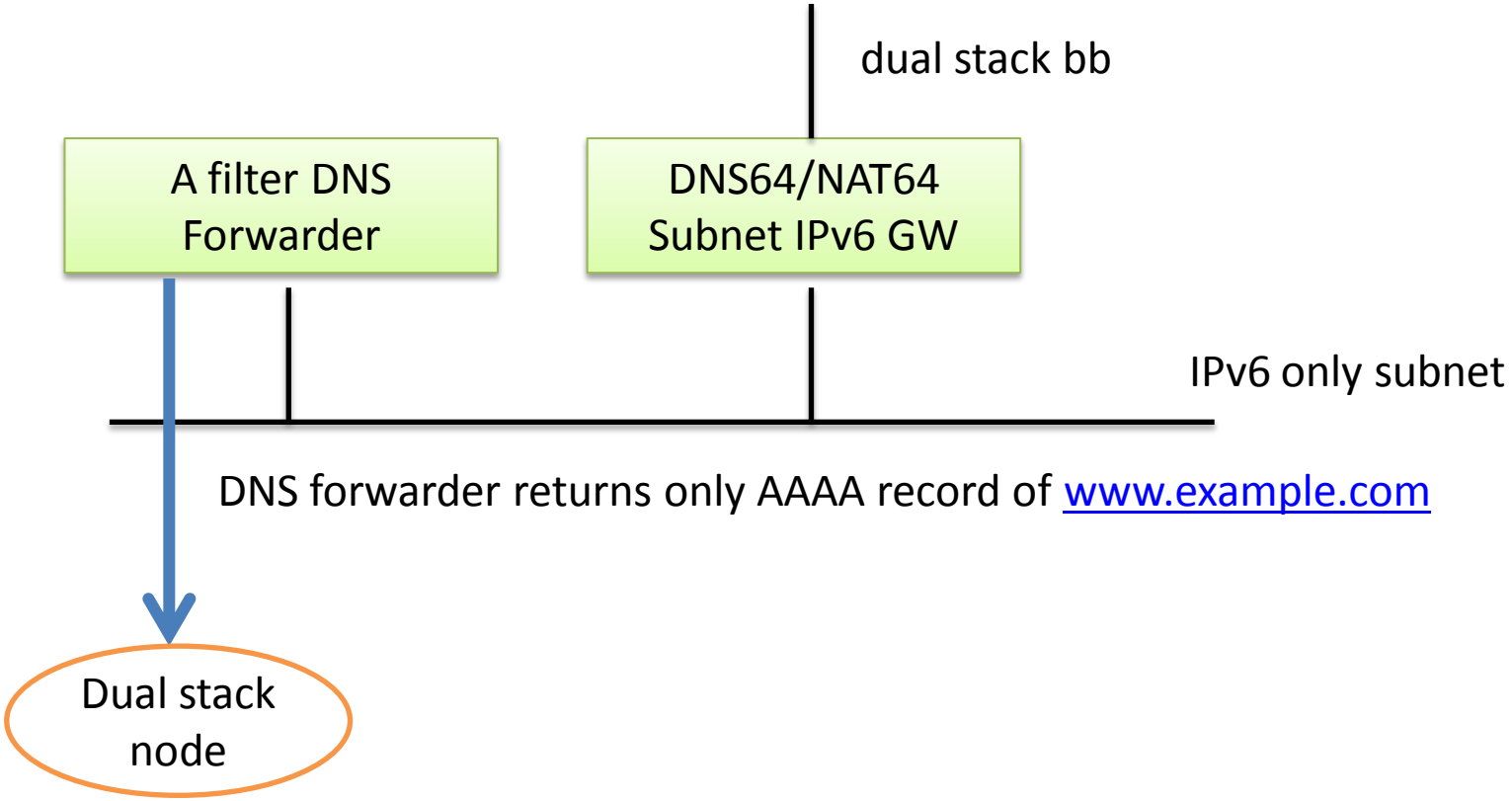
DHCP6 supported dual stack node



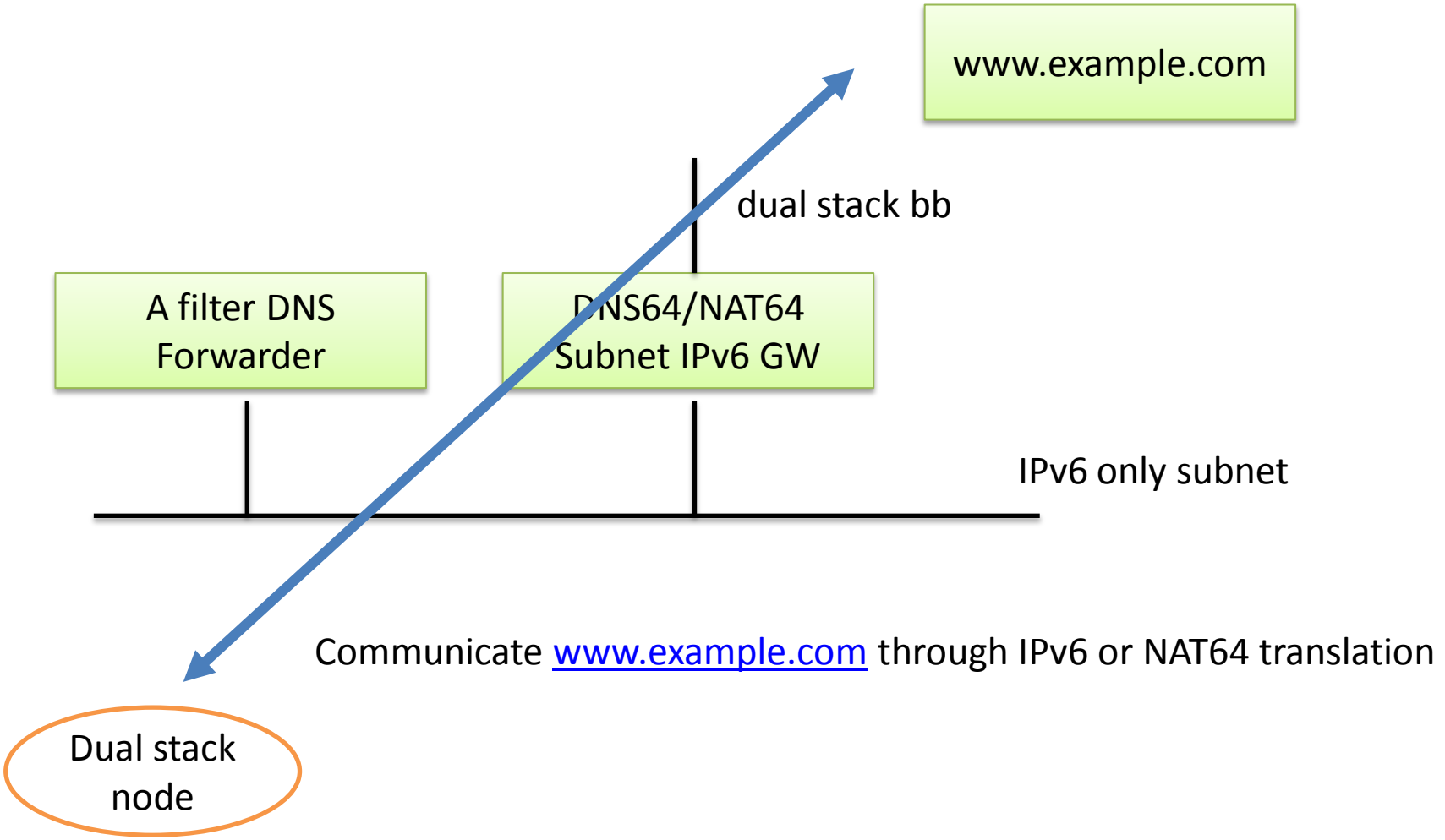
DNS64 may return A and AAAA records on www.example.com
The AAAA record may be NAT64 prefix-mapped IPv4 address

Dual stack node

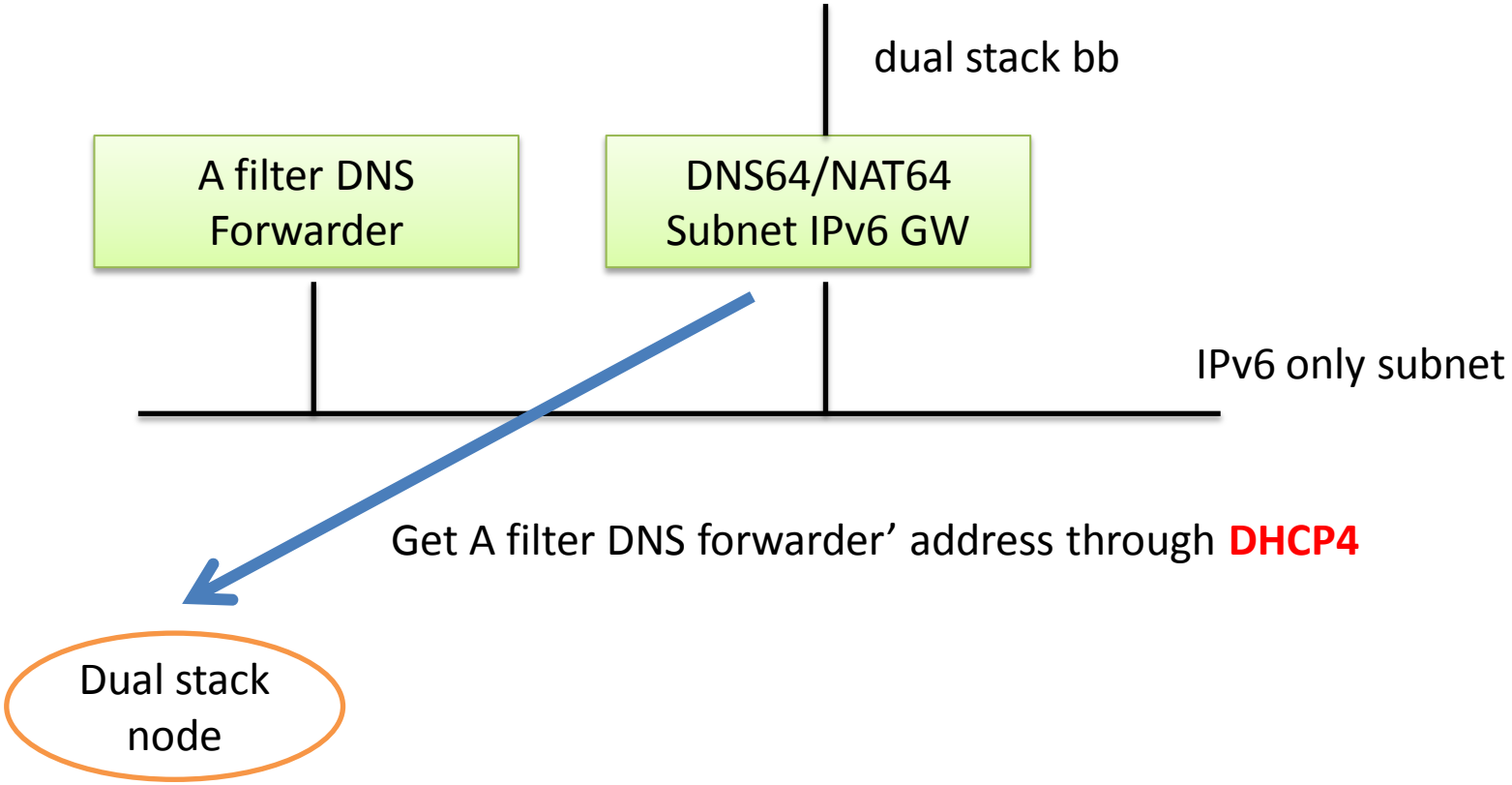
DHCP6 supported dual stack node



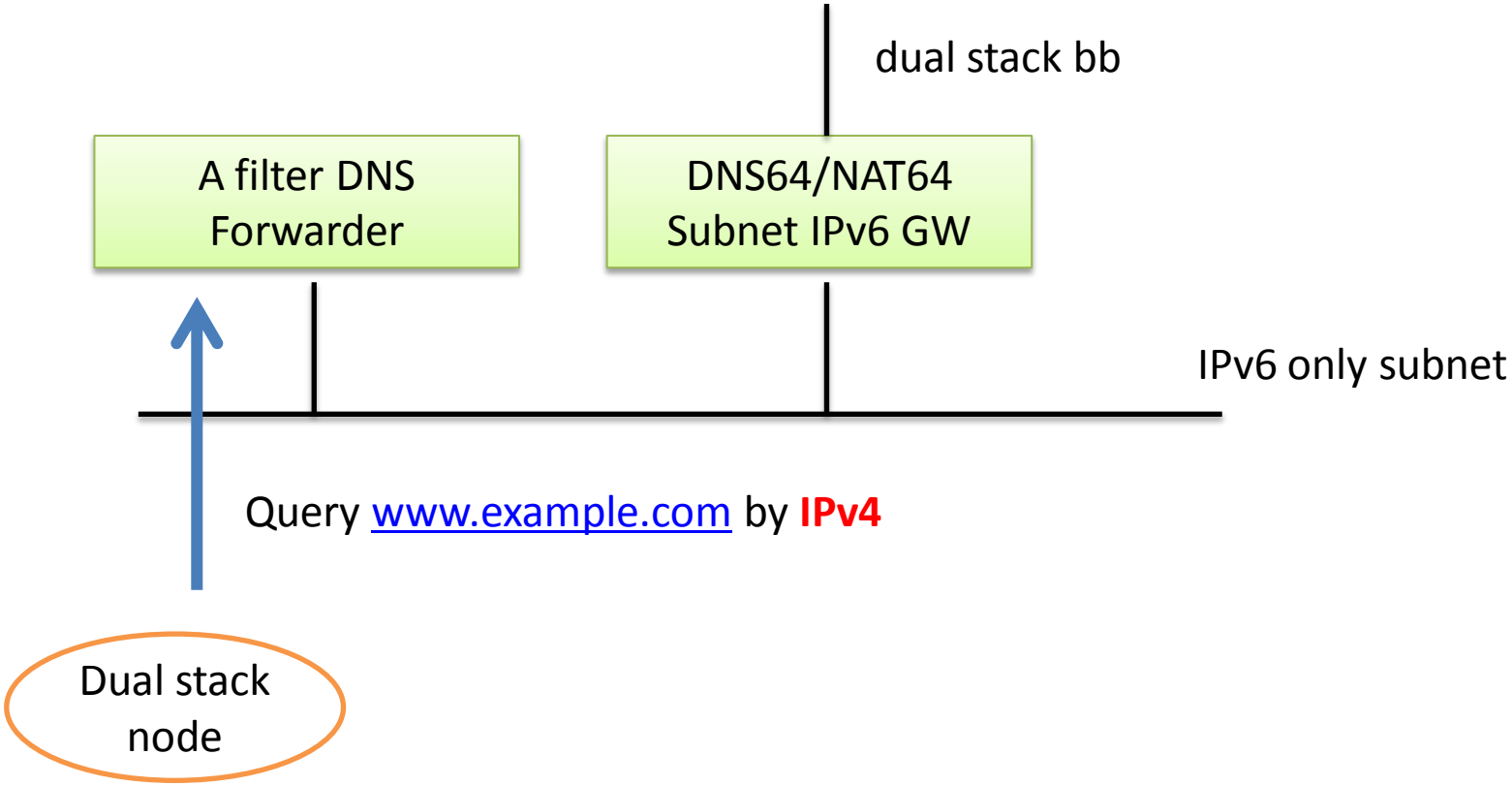
DHCP6 supported dual stack node



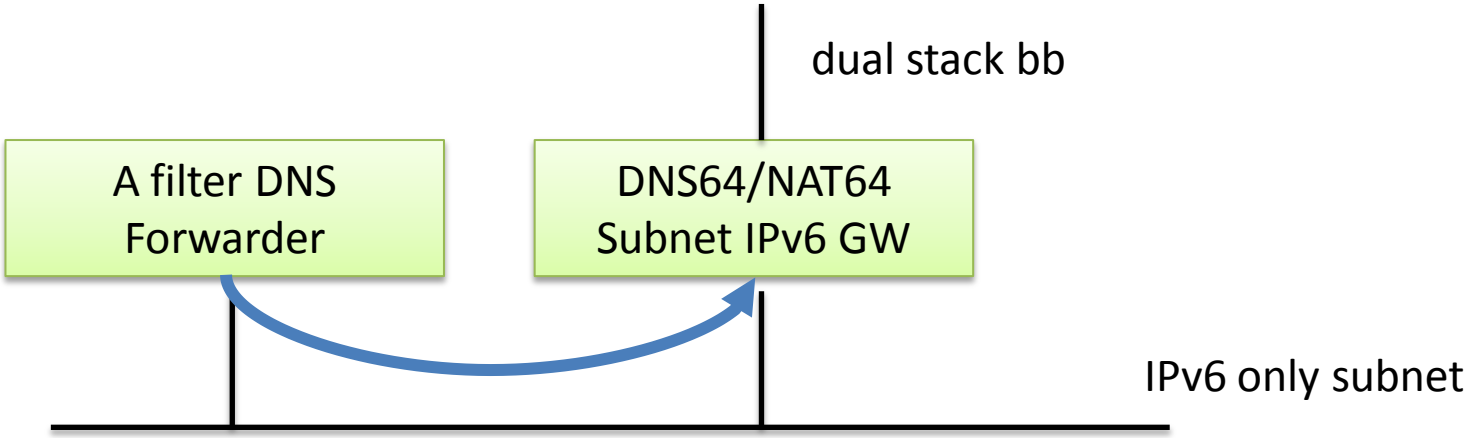
IPv4 depending dual stack node
(Windows XP, Mac OS X snow leopard etc.)



IPv4 depending dual stack node



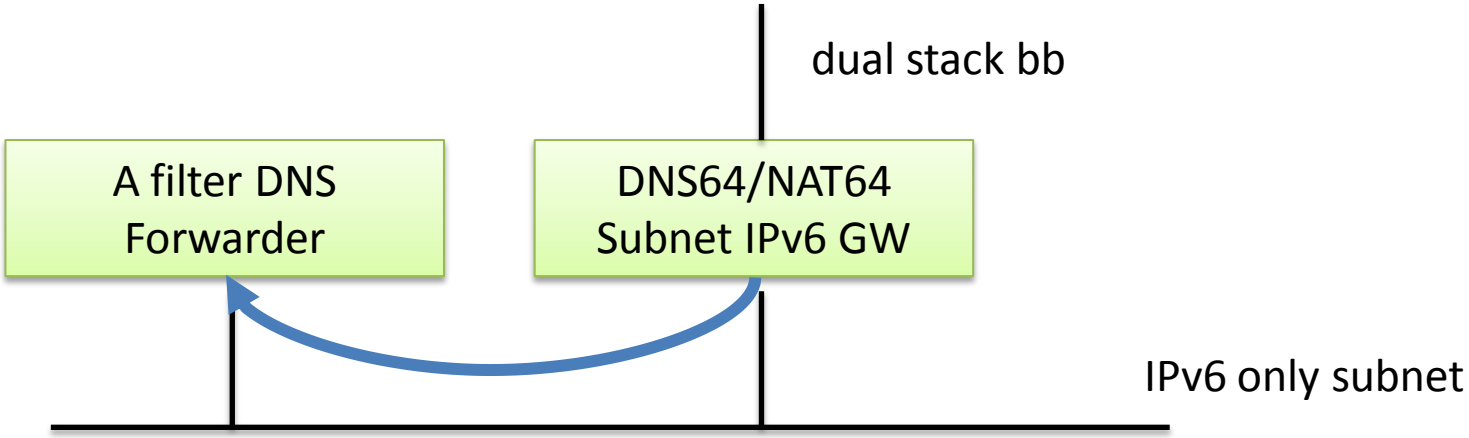
IPv4 depending dual stack node



Forward query on www.example.com to DNS64 or other cache servers

Dual stack node

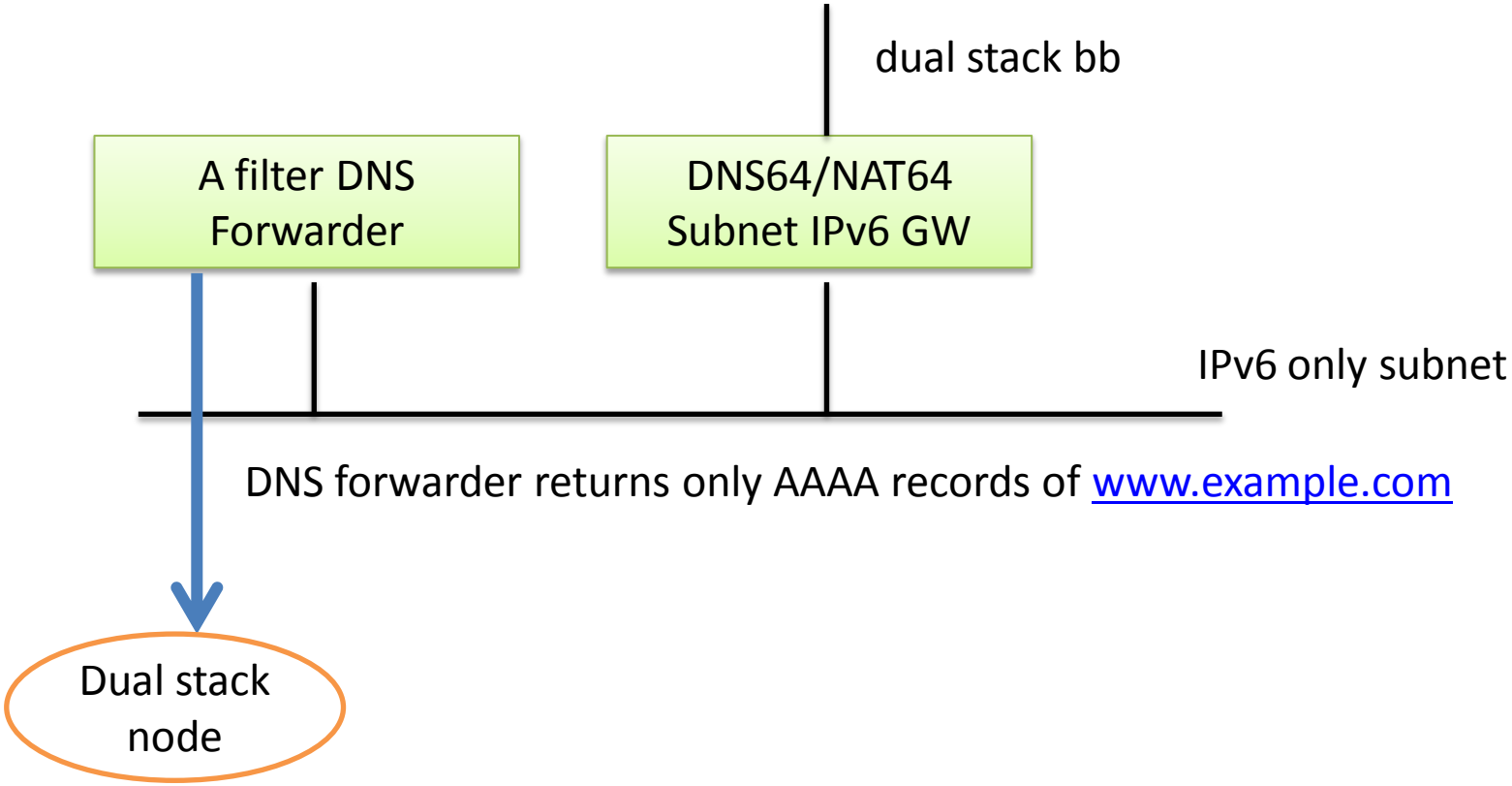
IPv4 depending dual stack node



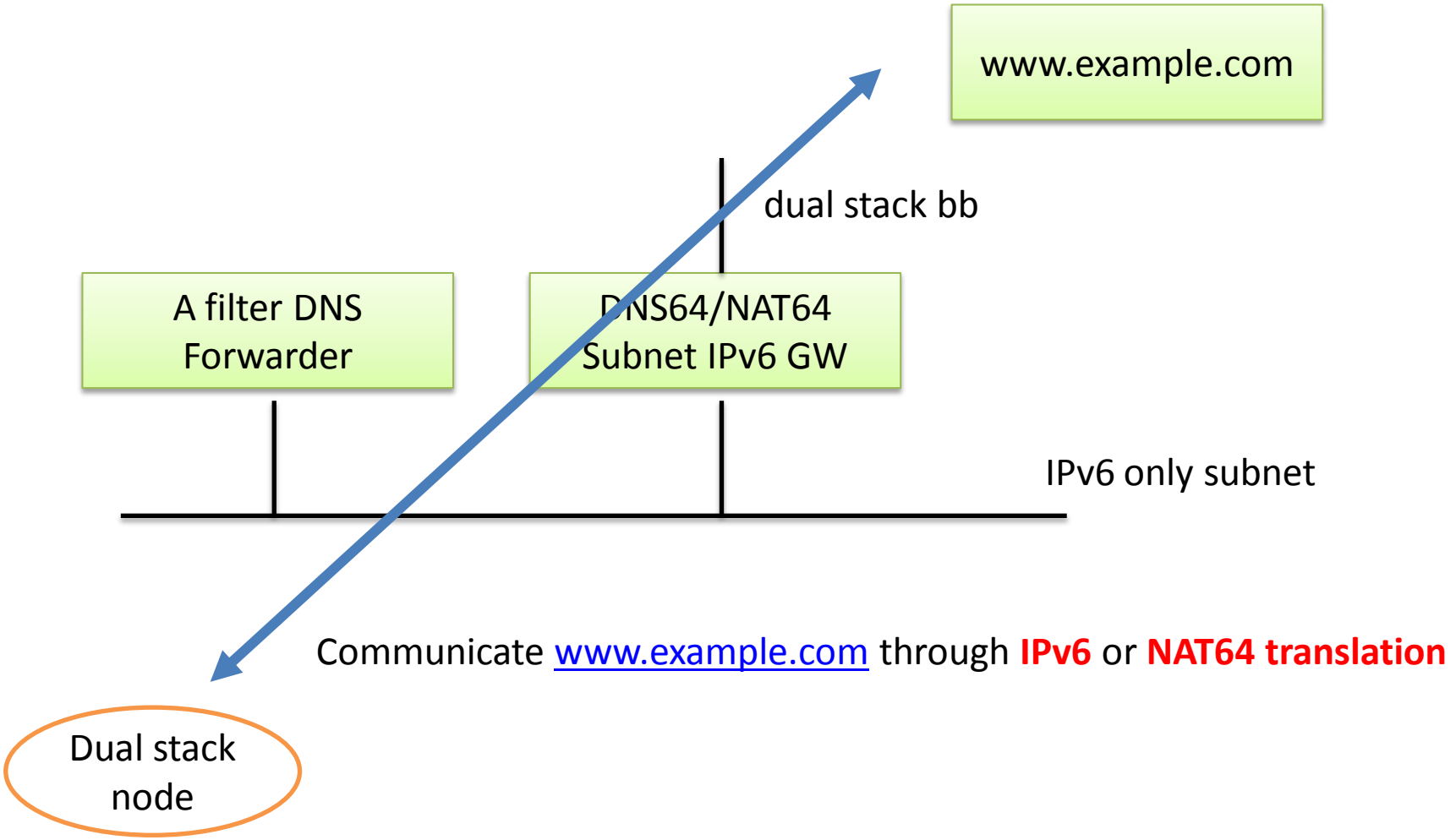
DNS64 may return A and AAAA records on www.example.com
The AAAA record may be NAT64 prefix-mapped IPv4 address

Dual stack node

IPv4 depending dual stack node



IPv4 depending dual stack node



Appendix

The summary of test and experiment in past wide camp

Tested OSs in past wide camp

- Windows Vista, 7, 8, XP
- MacOS X (Leopard, Snow Leopard, Lion, Mountain Lion)
- Apple iOS 4, 5
- Android 2.3.4, 4.1

- All of them can get IPv6 address through RA
- See the details in section 4.2 of draft-hazeyama-widecamp- ipv6-only-experience-02

Test Results

- Windows Vista, 7, 8
  - No problem except for waiting time out of DHCP4
- Windows XP, MacOS X (~Snow Leopard)
  - Need manual settings of DNS
  - Most people cannot set up by themselves
- MacOS X
  - Waiting for IPv4 connection timeout for Chrome, Firefox. (Safari has no problem).
  - Many Mac OS users met nightmare !
- iOS5
  - In some 3G carrier, network Settings was not completed ,and continuously failed due to retry
  - In other 3G carrier, there are no problem
- Android 2.3.4, 4.1
  - DNS was not usable (DNS queries employed IPv4 function).
  - No manual configuration was available.

Issues in detail

- DNS is not available (WinXP, MacOS Snow Leopard, Android)
  - DNS information should be got via DHCP6, but these OSes did not have DHCP6 function.
  - Android 2.x cannot be configured to use DNS over IPv6 even in manual configuration.
- Network Settings is not completed in IPv6 only network (iOS5 in Some 3G carrier)
  - “Network Settings” will be completed only if IPv4 address, IPv4 router, and IPv4 DNS can be retrieved via DHCPv4 or manually configured all of these 3.
- Waiting for IPv4 connection timeout (MacOS X)
  - MacOS X implements RFC3927 3.3 (Interaction with Hosts with Routable Addresses), which assumes all IPv4 address are on-link at Link-Local configuration.
  - MacOS X’s implementation of getaddrinfo() prefer IPv4 over IPv6, while Happy-Eyeball is implemented on Objective-C API. So Safari has no timeout problem, but Chrome, Firefox and most other application uses C API faced the problem.

Experiment of the work around

1. Placing a DHCP4 server on the subnet of an IPv6 Only network
   - Long timeout problems due to the IPv4 link local assumption of Mac OS X are mitigated
   - Long timeout by DHCP4 request is also resolved
   - Windows 7, 8 was working well
   - Other OSs still had problems
2. Locating a DNS proxy with * deny- answer-addresses { 0.0.0.0/0; }; *
   - Android 2.x was working well, but Windows XP did not work well
   - TCP timeout problems still occur on Mac OSX or several browsers

3. Setting DNS A Record filter on the DNS proxy instead of * deny- answer-addresses { 0.0.0.0/0; }; *
   - Windows XP, MacOS X variants, iOS, Android were working well
   - TCP timeout problems are mitigated