A   P. Thubert, Ed.
        uCisco Systems
        t
        h
        o
        r
        s
        :

Yet Another Double Address and Translation Technique

Abstract

This document provides a stepwise migration between IPv4 and IPv6
with baby steps from an IPv4-only stack/gateway/ISP to an IPv6-only
version, that allows portions of the nodes and of the networks to
remain IPv4, and reduces the need for dual stack and CG NATs between
participating nodes. A first mechanism named YADA to augment the
capacity of the current IPv4 Internet by interconnecting IPv4 realms
via a common footprint called the shaft. YADA extends RFC 1122 with
the support of an IP-in-IP format used to forward the packet between
parallel IPv4 realms. This document also provides a stateless
address and IP header translation between YADA and IPv6 called YATT
and extends RFC 4291 for the YATT format. The YADA and YATT formats
are interchangeable, and the stateless translation can take place as
a bump in the stack at either end, or within the network at any
router. This enables an IPv6-only stack to dialog with an IPv4-only
stack across a network that can be IPv6, IPv4, or mixed. YATT
requires that the IPv6 stack owns a prefix that derives from a YADA
address and that the IPv4 stack in a different realm is capable of
YADA, so it does not replace a generic 4 to 6 translation mechanism
for any v6 to any v4.

Status of This Memo

This Internet-Draft is submitted in full conformance with the
provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering
Task Force (IETF). Note that other groups may also distribute
working documents as Internet-Drafts. The list of current Internet-
Drafts is at https://datatracker.ietf.org/drafts/current/.

Table of Contents

1. **Introduction and Motivation**

   At the time of this writing, the transition to IPv6 started 20 years
   ago and large amounts of networks, hosts, and programs, are still
   IPv4-only. The IPv4 and IPv6 camps are quite entrenched, and there's
   no indication that things will change any time soon.

   During that endless transition, stacks must implements both
   protocols (aka dual stack) and a mechanism to use either based on
   the responsiveness (Happy Eyeballs). Service Providers must
   implement heavy weaponry called Carrier-Grade Network Address
   Translators (CG-NATs) to translate between protocols between legacy
   IPv4-only and IPv6-only stacks, and tunneling techniques such as DS-
   Lite and 464XLAT to traverse portions of the network that support

only one of the IP versions. This means both CAPEX to install dual stack infrastructures and NAT devices and OPEX to maintain them. The current situation is often qualified as the worst of both worlds and any indications is that it's here to stay, till each side suffered enough and is ready for a compromise.

This document prepares for that time where the players will effectively be ready for a compromise. An acceptable compromise must provide both sides with way to remain as long as desired, while eliminating the need for dual stack and CG-NATs between participating nodes. Certainly, an effort must be asked on each side to reduce the chasm, and that effort must come with enough benefits to effectively encourage a majority of interested parties to make the step.

Yet Another Double Address (YADA) refers to effort that is asked from the IPv4 side to support a new IP-in-IP model. YADA extends [INT-ARCHI] with the support of an IP-in-IP format used to forward the packet between parallel IPv4 realms. The proposed benefit is a thousandfold increase of the IPv4-addressable domain by building parallel realms each potentially the size of the current Internet. Only the stacks that need to talk to a parallel realm need to evolve. Routing and forwarding can remain IPv4-only with the same operations as today, though new routers with YADA capabilities must be deployed to route between realms.

Yet Another Translation Technique (YATT) refers to an effort to be made by the IPv6 side to support a new IPv6 Prefix with special properties, which impacts in particular source address selection (SAS). YATT extends [IPv6-ADDRESSING] for the YATT format. The proposed benefit is a prefix (say /32) per realm and a prefix (say /64) per host in the realm. This address space may for instance become handy for load balancing between physical servers / VMs / pods that operate a service associated with the virtual server that owns the host prefix.

The YADA and YATT formats are interchangeable, which means that the translation is stateless and can take place as a bump in the stack at either end or can be operated at line rate anywhere in the network by an upgraded hardware. The routers that connect the shaft also perform a stateless operation that can be achieved at line rate by upgraded hardware. This is how the chasm between IPv4 and IPv6 can be reduced, removing the need to deploy dual stack and CG-NATs between participating nodes.

This document provides a stepwise migration between IPv4 and IPv6 with baby steps from an IPv4-only stack/gateway/ISP to YADA to YATT to an IPv6-only version. The migration strategy allows portions of the nodes and of the networks to remain IPv4. This enables an IPv6-only stack to dialog with an IPv4-only stack across a network that can be IPv6, IPv4, or mixed.

YATT requires that the IPv6 stack owns a prefix that derives from a YADA address associated to a realm, even if there's absolutely no

IPv4 operation taking place in that realm. The resulting
connectivity without dual stack and CG-NAT is as follows:

   *A legacy IPv4-only node can only talk within its realm. It can
    talk to a IPv4 legacy node, and YADA IPv4-only node and a YATT
    IPv6-only node, e.g., leveraging a bump-in-the-stack in the YATT
    node if the network is IPv4-only.

   *In addition, a YADA IPv4-only node can talk across realms to a
    YADA IPv4-only node and to any YATT IPv6-only node.

   *In addition, a YATT IPv6-only node can talk to all the IPv6
    addressable space to any IPv6-only node.

Connectivity between an IPv4-only node and an IPv6-only node, or
between an IPv4-only node and a YADA node in different realm, still
requires a CG-NATs as of today, e.g., using the YATT format for the
IPv6 side in an unmodified CG-NAT.

2. Terminology

2.1. Glossary

This document often uses the following acronyms:

YADA: Yet Another Double Address

YATT: Yet Another Translation Technique

NAT: Network address Translation

IID: Interface ID

CG-NAT: Carrier Grade NAT

2.2. New Terms

This document often uses the following new terms:

IPv4 realm: A full IPv4 network like the current Internet. YADA
   does not affect the traditional IPv4 operations within a realm.

The shaft: The shaft refers to a collection of IPv4 unicast and
   multicast prefixes that are assigned to Inter-realm communications
   and cannot be assigned to hosts or multicast groups within a
   realm.

Realm address: An IPv4 address that derives from a shaft prefix.

Uni-realm address: A realm address that is unicast or anycast. A
   realm may have more than one Uni-realm add ress.

Multi-realm address: A realm address that is multicast and denotes
   a collection of realms.

YADA realm prefix: A prefix assigned to the shaft and from which
   realm addresses can be derived.

YADA NAT prefix:
   A prefix assigned to the YADA bump-in-the-stack NAT operation.

Double-A or YADA address: A YADA address is a tuple (realm address,
   IPv4 address) where the IPv4 address is only significant within
   the realm denoted by the realm address.

YATT Space: An IPv6 range that is assigned for YATT operation.

YATT prefix: An IPv6 prefix that is derived from a YADA address by
   appending the YATT space prefix, the (truncated) realm address and
   the IPv4 address.

YATT-IID: A 64-bit assigned constant that is used in YATT to
   statelessly form an IPv6 address from a YATT prefix.

Multinternet: A collection of IPv4 realms interconnected using a
   common shaft.

3. Operation

This document provides a stepwise migration between IPv4 and IPv6
with baby steps from an IPv4-only stack/gateway/ISP to an IPv6-only
version. The baby steps reduce the gap between the only versions and
teh associated need for dual stack and CG-NATs.

The first step called YADA uses IPv4-only signaling. The second step
called Yet Another Translation Technique (YATT) offers an IPv6-only
signaling that is interchangeable with YADA, so any router or stack
may turn one into the other, allowing the stack or the link to be
one version only. A YADA-enabled IPv4 stack can thus talk to a YATT-
enabled IPv6 stack with neither CG-NATs nor dual stack network in
between, but a stack that is not aware of this specification will
still need a traditional NAT approach to communicate.

The effort in this specification is to provide enough value /
incentive for an IPv4-only stack/gateway/ISP to make the step
towards YADA, as a push towards IPv6, and for an IPv6-only stack to
support YATT on top to pull IPv4 space in IPv6, with a low barrier
for making the baby step. For IPv4, going YADA expands the size/
reach of the Internet, and allows multiple parties to build their
own IPv4 realm, with control of interconnection with other realms.
For an IPv6 node, supporting YATT provides connectivity to the YADA
world, and automatically assigns a prefix in the node.

This first mechanism called YADA allows to grow the Internet beyond
the current IPv4 [IPv4] realm that limits its capacity to form
public addresses. Depending on the assignments to be made, the model
allows to reuse all IP addresses and all Autonomous System Number
(ASN) currently available in the internet hundreds to millions of
times. This is achieved by interconnecting IPv4 realms via a common
footprint called the shaft.

In the analogy of a building, the ground floor would be the
Internet, and each additional floor would be another IPv4 realm. The
same surface of floor is available in each level, analog to the full
IPv4 addressing that is available in each realm. The same footprint
is dedicated across the building levels for the elevator shaft. The

elevator shaft enables a third dimension that spans across the
levels and allows to traverse from any level to any other level. The
elevator shaft cannot be used for living or office space.

```
        /------------------------------------------------------
       /                                                     /
      /           |-----------|              realm 1  /
     /           /.           /.                            /
    /           / . shaft    / .  (current IPv4 Internet)  /
   /           |-----------|  .                           /
  /            .  .         .  .                         /
 --------------------------------------------------------/
              |  .          |  |
      /-----|-----------|--|-----------------------------
     /      |  .          |  |                          /
    /       |  |---------|--|          realm 2  /
   /        | /.          | /.                          /
  /         |/ . shaft    |/ .                         /
 /          |-----------|  .                          /
/           .  .          .  .                       /
--------------------------------------------------------/
            |  .          |  |
            |  .          |  |
            |             |  .
            |             |  .
            .             .  |
            .             .  |
            |  .          |  |
    /-----|-----------|--|-----------------------------
   /      |  .          |  |                          /
  /       |  |---------|--|          realm N  /
 /        | /          | /                            /
/         |/   shaft   |/                            /
/         |-----------|                             /
/                                                  /
--------------------------------------------------------/
```
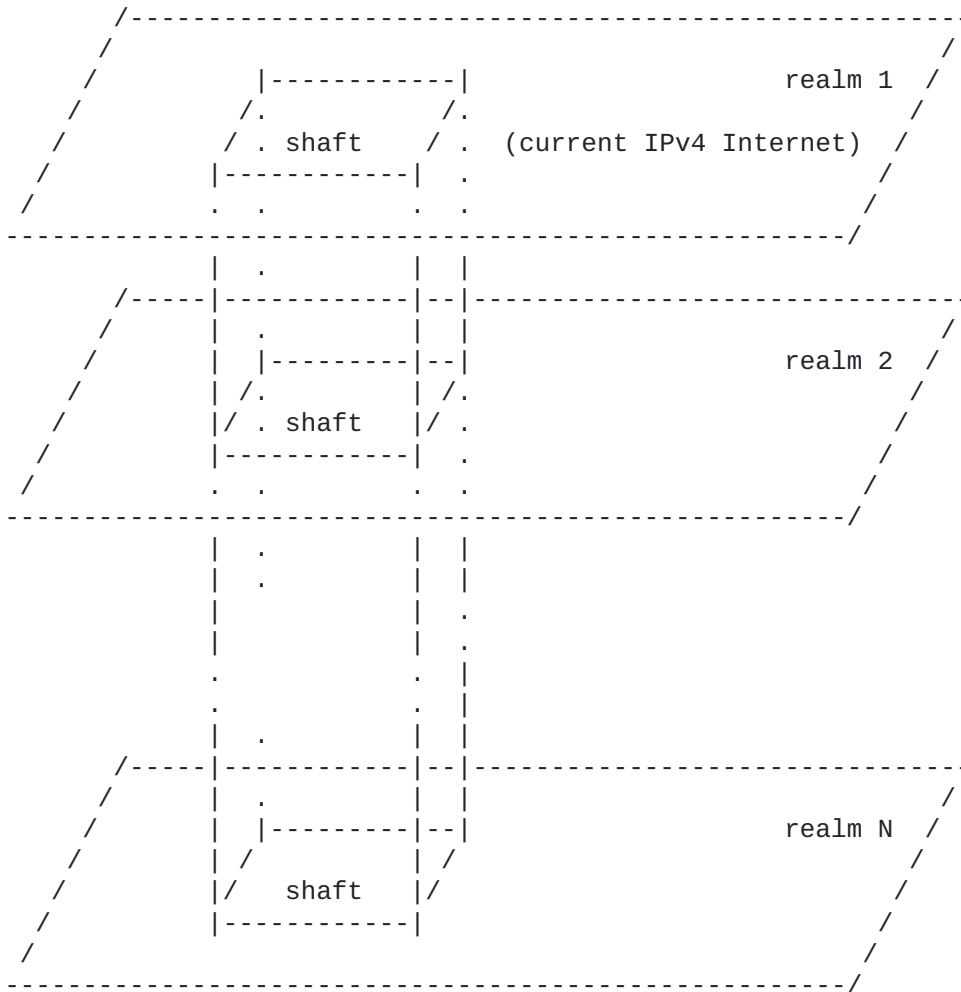
                     Figure 1: The shaft

By analogy, YADA assigns IPv4 prefixes to a multinternet shaft;
those prefixes are common across the realms that are interconnected
by the shaft. A single /24 IPv4 prefix assigned allows for > 250
times the capacity of the Internet as we know it at the time of this
writing. Multiple prefixes can be assigned to the shaft for unicast
and multicast communications, and each realm needs at least one
unicast address in the shaft called its realm address. A YADA
address is formed by the tuple (realm address, IPv4 address) and is
advertised in DNS as a new double-A record.

YADA leverages IP-in-IP encapsulation to tunnel packets across the
shaft while normal IPv4 operations happen within a realm. YADA
requires a change in the stack in the YADA endpoints that
communicate with other realms to support the IP-in-IP YADA

encapsulation. YADA also provides a bump in the stack method for
legacy applications. More in Section 6.

A second mechanism called YATT translates the YADA format into flat
IPv6 [IPv6]. While a YADA address pair can be seen as some foot
print in one level, the YATT prefix encompasses that same foot print
plus all the air above it. For unicast addresses, YATT forms an IPv6
prefix by collating an well-known assigned short prefix, the realm
address (in the shaft), and the host IPv4 address (locally
significant within the realm). The resulting IPv6 prefix is
automatically owned by the host that owns the IPv4 address in the
realm. YATT then forms an IPv6 address for that host by collating a
well-known Interface ID, so there's a one-to-one relationship
between the YADA and the IPv6 address derived from it. More in
Section 7.

A key concept for this specification is that YADA (the IPv4
formulation) and YATT (the IPv6 formulation) represent the same
thing. YADA uses IPv4 formats as plain IP-in-IP with no new
extension. YATT uses IPv6 format with the IPv4 addresses encoded on
the prefix. The formats are interchangeable, and a router can
convert one to another as the packet flows over a next-hop link that
can only carry the other address family.

4. Extending RFC 1122

YADA extends [INT-ARCHI] to add the capability for an IPv4 host to
recognize an special IP-in-IP format as an inter-realm IPv4 packet
and process it accordingly. It also adds a new DNS double-A record
format that denotes a YADA address.

5. Extending RFC 4291

YATT extends [IPv6-ADDRESSING] to add the capability for an IPv4
host to recognize an special IPv6 format as an YATT address
embedding a YADA address and process it accordingly. It also
automatically derives the ownership of the YATT prefix associated to
a owned YADA address.

6. YADA

YADA assigns IPv4 prefixes to a multinternet shaft; those prefixes
must be the same across all the realms that are interconnected by
the shaft. Multiple prefixes can be assigned to the shaft for
unicast and multicast communications, and each realm needs at least
one unicast address in the shaft called its realm address. A YADA
address is formed by the tuple (realm address, IPv4 address) and is
advertised in DNS as a new double-A record. Because the YADA
prefixes are assigned for YADA, a packet that has either source or
destination IPV4 address derived from a shaft prefix is a YADA
packet.

YADA leverages IP-in-IP encapsulation to tunnel packets across the
shaft for inter-realm communications, while the IPv4 operations
within a realm are unaffected. The YADA address is found by using
both inner and outer header and combining that information. The pair
of IP headers is seen by a YADA stack as a single larger header

though a non-YADA forwarder only needs the outer header and plain
IPv4 operations on the outer IPv4 header to forward.

YADA requires a change in the stack in the YADA endpoints that
communicate with other realms to support the YADA encapsulation.
YADA also provides a bump in the stack method for legacy
applications. A stack that resolve a DNS name with a double-A record
indicating a different realm generates an IP-in-IP packet to signal
both the source and destination realms and the source and
destination IPv4 addresses within the respective realms.

Inside the source realm, the outer IPv4 header indicates the node's
IPv4 address as source, to remain topologically correct, and the
local realm address as source in the inner header, as shown in
Figure 2

```
<---------------------------- 20 bytes ---------------------------->
+------------ ... ------------+----------------+------------------+
|    IPv4 header fields       | Source node    | destination realm |
|       (outer)              | IPv4 Address   |   IPv4 Address    |
+------------ ... ------------+----------------+------------------+
|    IPv4 header fields       | Source realm   | destination node  |
|       (inner)              | IPv4 Address   |   IPv4 Address     |
+------------ ... ------------+----------------+------------------+
.                             Options                             .
+------------ ... ------------------------------------------------+
|                                                                 |
.                              Data                               .
|                                                                 |
+-----------------------------------------------------------------+
```
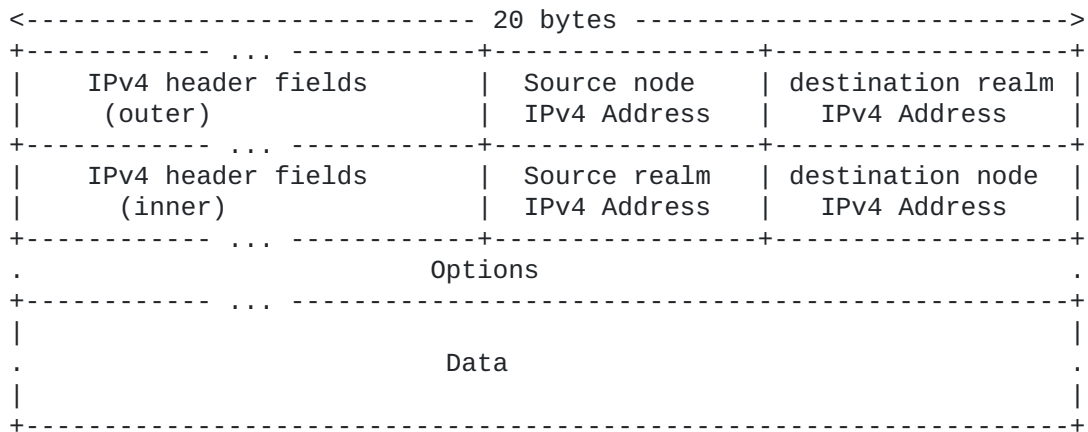
               Figure 2: YADA format in the source realm

 YADA also requires a change for the routers that serve the shaft.
 Those routers play a special role for packets that are delivered
 from the shaft to the destination realm, and for ICMP errors across
 realms. All other IPv4 nodes in the realm continue to operate
 routing and forwarding as before.

 Routers serving the shaft advertise the shaft prefix(es) in their
 respective realms, and their realm addresses within the shaft, as
 host routes for unicast and anycast addresses.

 Inside the source realm, the IPv4 destination in the outer header is
 an address is the shaft and it is attracted by a router that serves
 the shaft in the source realm. The packet source in the outer header
 is the address of the source node in the local realm, so the packet
 does not defeat BCP 38 rules in the ISP network, as shown in Figure
 3.

```
                |        |
    /-------|-----------|--------------------------------
   /        |           |                                /
  /     |   |     |   |                                 /
 /      |   |-------|---|          Source Node         /
/       | /         |   /                             /
/       | /.      <--|----  outer(src=src-addr       /
 /      |/ .         |/ .          dst=dst-realm)    /
/       |-----------|  .   inner(src=src-realm     /
 /        . .        . .          dst=dst-addr)   /
 /        . .        . .                         /
 /        . .        . .                        /
/         . .        . .                       /
  ---------------------------------------------/
          |            | |
          |            |
          |            |
          |            |
```
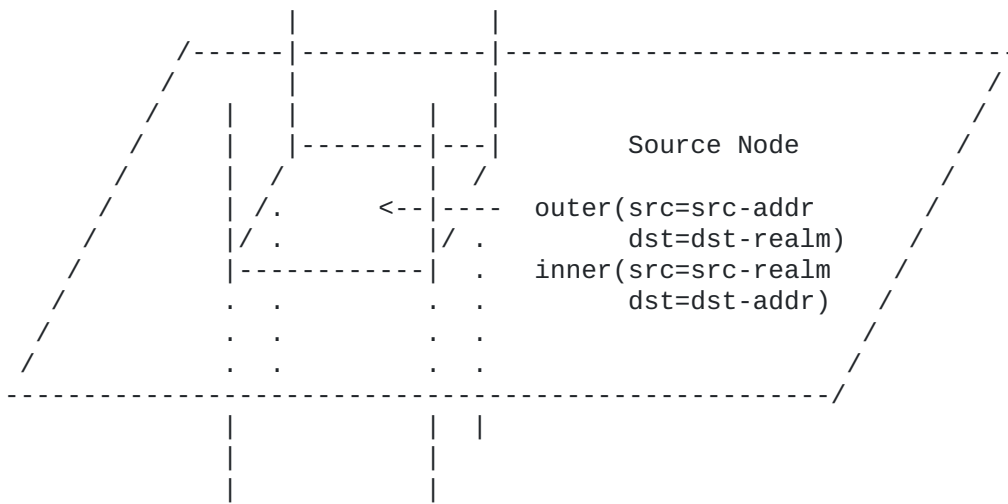
            Figure 3: Packets Entering the shaft

   When the packet reaches the shaft, the router that serves the shaft
   swaps the inner and outer source IPv4 address, so the packet remains
   topologically correct inside the shaft, as shown in Figure 4.


```
<----------------------------- 20 bytes ----------------------------->
+------------ ... -----------+----------------+------------------+
|    IPv4 header fields      |  Source realm  | destination realm |
|       (outer)             |  IPv4 Address  |   IPv4 Address    |
+------------ ... -----------+----------------+------------------+
|    IPv4 header fields      |  Source node   | destination node  |
|       (inner)             |  IPv4 Address  |   IPv4 Address    |
+------------ ... -----------+----------------+------------------+
.                           Options                              .
+------------ ... -----------------------------------------------+
|                                                                |
.                            Data                                .
|                                                                |
+----------------------------------------------------------------+
```

           Figure 4: YADA format inside the shaft

   Based on longest match, the router forwards the packet inside the
   shaft following the host route to a router that serves the
   destination realm, as shown in Figure 5.

```
                     |          |
          /------|-----------|-------------------------------
         /       |           |                              /
        /    |   |       |   |                             /
       /     |   |-------|---|       Source Node          /
      /      |  /        |  /                            /
     /       | /.    +   | /  outer(src=src-realm      /
    /        |/ .    |   |/ .         dst=dst-realm)   /
   /         |-----------|  .   inner(src=src-addr    /
  /          . .     |   . .          dst=dst-addr)  /
 /           . .     |   . .                        /
 /           . .     |   . .                       /
 ---------------------------------------------------/
            |        |   |  |
            |        |   |  |        forwarded unchanged
            |        |   |  |           down the shaft
                     v
```
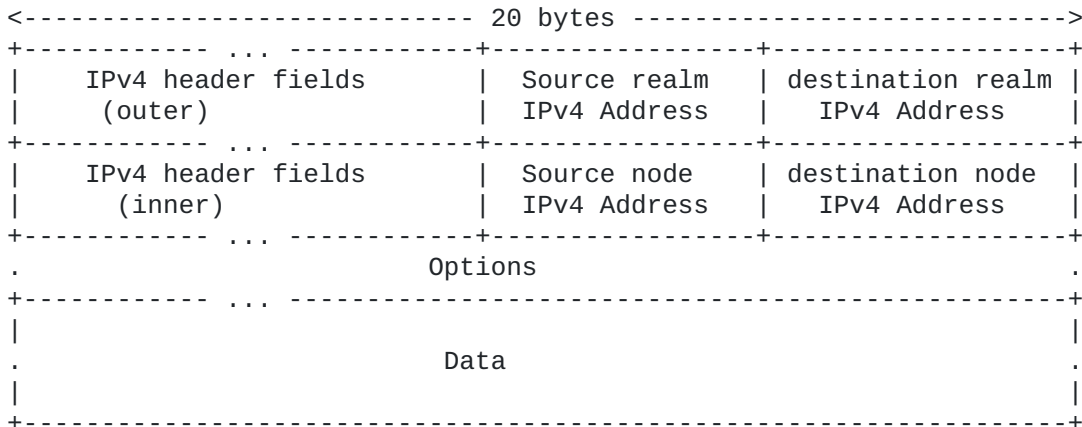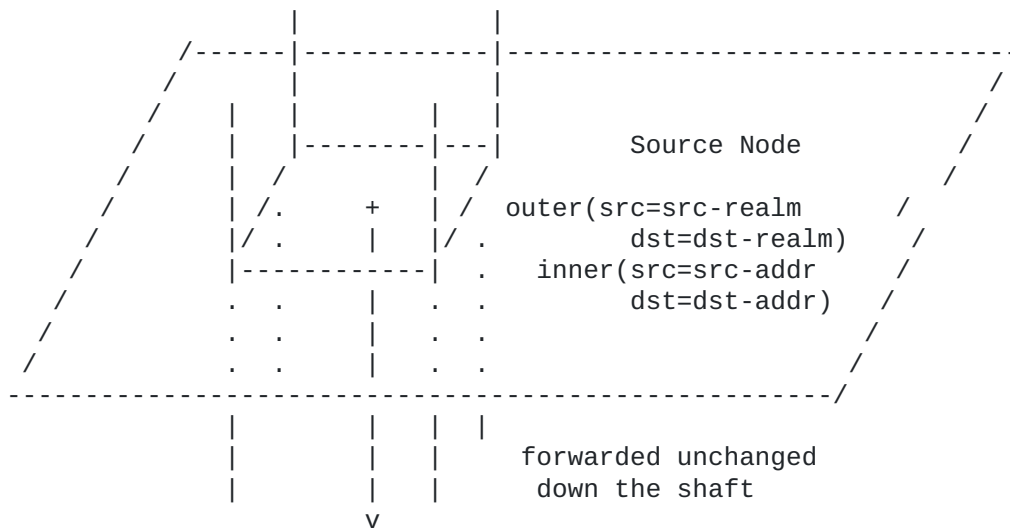
                 Figure 5: Packets Entering the shaft

   That router swaps the destination address in the inner and outer
   headers and forwards within its realm to the final destination, as
   shown in Figure 6.

```
<---------------------------- 20 bytes ---------------------------->
+------------ ... -----------+----------------+------------------+
|    IPv4 header fields      | Source realm   | destination node |
|       (outer)             | IPv4 Address   |   IPv4 Address    |
+------------ ... -----------+----------------+------------------+
|    IPv4 header fields      | Source node    | destination realm |
|       (inner)             | IPv4 Address   |   IPv4 Address    |
+------------ ... -----------+----------------+------------------+
.                        Options                                  .
+------------ ... -----------------------------------------------+
|                                                                 |
.                          Data                                   .
|                                                                 |
+-----------------------------------------------------------------+
```
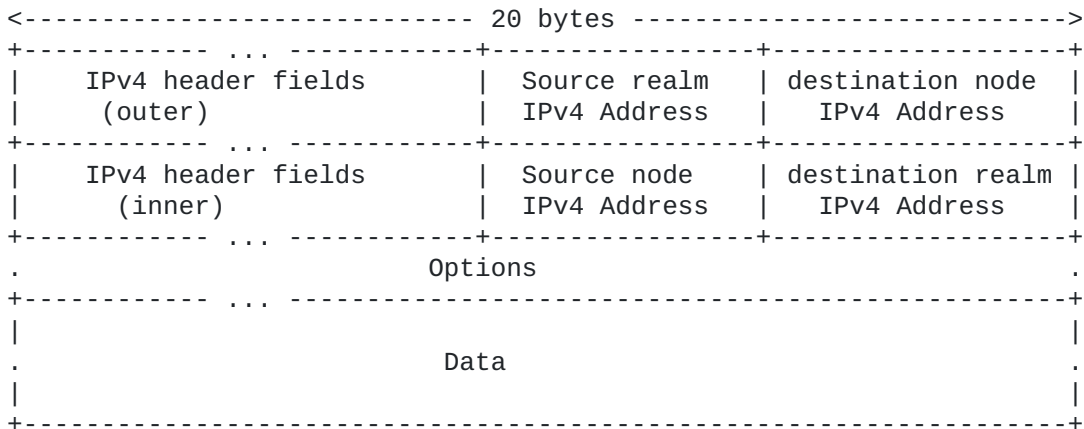
              Figure 6: YADA format in the destination realm

   In normal conditions, the stack of the destination node recognizes
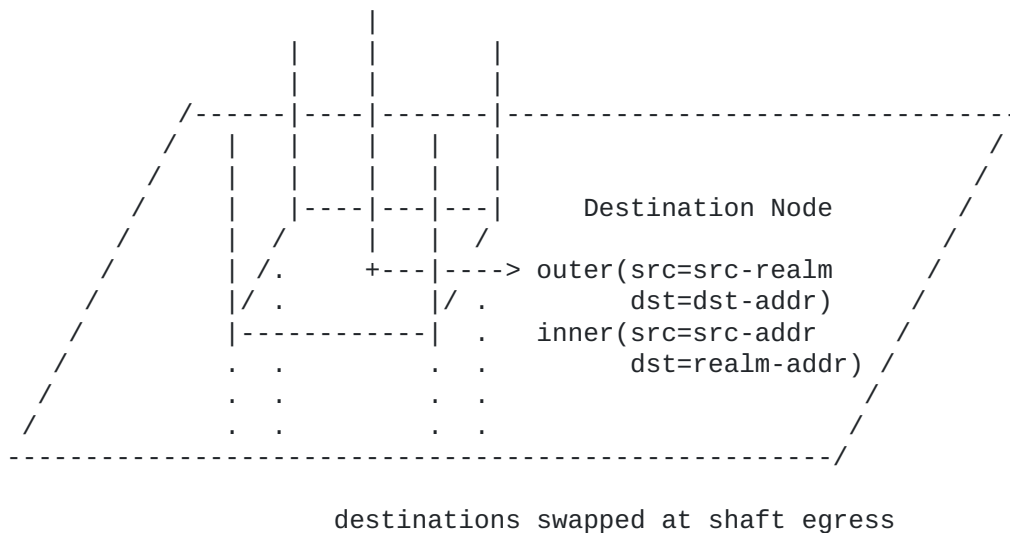   the YADA format and replies accordingly.

```
                  |
              |   |    |
              |   |    |
     /------|----|-------|-------------------------------
    /   |   |   |   |   |                                /
   /    |   |   |   |   |                               /
  /     |   |----|---|---|      Destination Node       /
 /      |  /     |   |  /                              /
/       | /.     +---|----> outer(src=src-realm       /
  /     |/ .         |/ .        dst=dst-addr)        /
 /      |-----------|   .   inner(src=src-addr       /
/       .  .        .  .         dst=realm-addr)    /
 /      .  .        .  .                            /
/       .  .        .  .                           /
 ------------------------------------------------/
```

                destinations swapped at shaft egress

                Figure 7: Packets Outgoing the shaft

    In case of an error down the shaft or in the destination realm, if
    an ICMP message is generated by a node that is not YADA-aware, the
    message reaches the router that serves the shaft in the source
    realm. If the inner header is present in the ICMP payload, then the
    Router extracts it and forwards to the packet source. If the
    destination stack does not support YADA and decapsulates, the
    message reaches the router that serves the destination realm which
    logs and drops. based on the log, the node may be updated, or the
    DNS records may be fixed to avoid pointing on a node that does not
    support YADA.

    YADA requires the assignment of a second IPv4 prefix, this time for
    a internal NATing operation. A bump-in-the-stack intercepts the DNS
    lookups, and when the response yields a double-A record with a
    foreign realm, the record is augmented with an IPv4 address taken
    from a local NAT pool. When the stack sends a packet to that
    particular address, the bump-in-the-stack translates to the YADA
    format, using the information in the double-A record for the
    destination, and the local realm as source realm. The other way
    around, if a packet arrives with a YADA format but the stack does
    not support it, the bump-in-the-stack allocates an address from the
    pool, and NATs to IPv4 using that address as source.

    YADA was initially published as USPTO 7,356,031, filed in February
    2002.

7. **YATT**

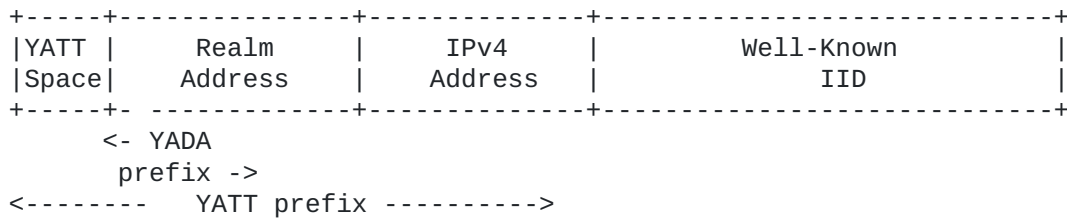   A second mechanism called YATT translates the YADA format into flat
    IPv6.

```
+-----+---------------+-------------+-----------------------------+
|YATT |    Realm      |    IPv4     |         Well-Known          |
|Space|    Address    |   Address   |            IID              |
+-----+- ------------+-------------+-----------------------------+
      <- YADA
       prefix ->
<--------    YATT prefix ---------->
```

                      Figure 8: YATT format

 For unicast addresses, YATT forms an IPv6 prefix by collating an
 well-known assigned short prefix called the YATT space, the realm
 address, and the host IPv4 address (locally significant within the
 realm). The resulting IPv6 prefix is automatically owned by the host
 that owns the IPv4 address in the realm.

 Depending on assignment, the leftmost piece realm prefix may be
 truncated if it is well-known, to allow the YATT space and the realm
 address to fit in a 32-bit DWORD. This way, the YATT prefix can be a
 full /64 prefix that is entirely owned by the host that owns the
 associated YADA address.

 YATT then forms an IPv6 address for that host by collating a well-
 known Interface ID, so there's a one-to-one relationship.

 The formats can not be strictly provided till the YATT space and
 YADA prefix are assigned. But say that the YATT Space is F000::/6
 and the YADA prefix is 240.0.0.0/6. In that case the values
 perfectly overlap and the YATT format becomes as follows:

```
+-----+----------+----------------+-------------------------------+
| Realm Address  |   IPv4 Host    |         Well-Known            |
| in 240.0.0.0/6 | Public Address |            IID               |
+-----+- --------+----+----------+-------------------------------+
<--- 32 bits ---><--- 32 bits ---><------------ 64 bits ------------>
<------    YATT IPv6 prefix ------->
```
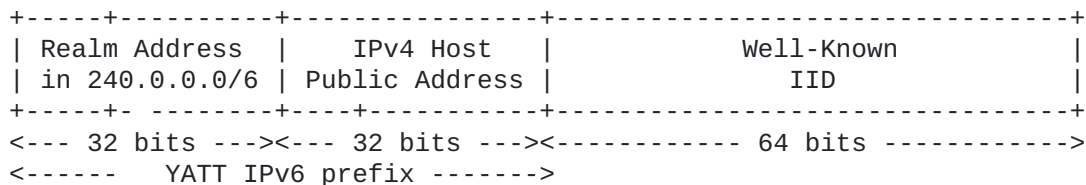
                Figure 9: YATT format using 240.0.0.0/6

 In that case, the NAT operation is a plain insertion. Depending on
 the assignment, it might be that the Realm address must be placed in
 full after YATT space. In that case, the length of the YATT prefix
 will be more than 64 bits.

 Also, since 240.0.0.0/6 is currently unassigned, using it for the
 shaft would allow literally to reuse every ASN and every IPv4
 address currently available in the Internet in each and every other
 realm and reallocate them in any fashion desirable in that realm.

 If the network supports IPv6 to the shaft, it makes sense for the
 YADA host or the bump-in-the-stack to generate the packets in the
 YATT form natively. The shaft router must then attract the shaft
 YADA realm prefix in both IPv4 and YATT forms.

 If the network is IPv4 only, the packets are still generated using
 IP-in-IP, and the YATT NAT operation may happen at the router that

delivers the packet in the destination realm, if it is v6-only, or
in the destination host, if its stack is v6-only.

YATT was initially published as USPTO 7,764,686, filed in December
2002.

8. **The structure of the shaft**

A 10 miles view of the shaft could be as follows: it is implemented
in one IXP, spans all realms, and each realm has one address in the
shaft, with one router serving that realm. The address of the realm
is encoded in a loopback in the router, and advertised through an
IGP inside the shaft, while BGP is used inside the realms but not
inside the shaft. The shaft has a single large prefix that is
advertised in each realm by the router that serves the shaft, and
that is disaggregated into host routes inside the shaft.

None of the above is expected to remain true for long. As YADA and
YATT get deployed, the shaft will be implemented in different sites
over the world. A realm may be multihomed to be reached from a
different physical instance of the shaft, meaning that the shaft is
composed of either more prefixes or the shaft prefix is
disaggregated. Multiple routers will serve the same realm with high
availability and load balancing taking place inside the shaft to
maintain connectivity. Some shafts may be deployed to interconnect
only a subset of the realms, in which case those shafts would share
a specific prefix that would not be advertised outside the concerned
realms.

9. **Applicability**

YADA And YATT enable communication between YADA-enabled IPv4 nodes
across realms, and with IPv6 nodes that own a YADA address from
which a YATT address can be derived. Communication from a legacy
IPv4 application/stack that is not YADA-enabled, or to an IPv6
address that is not a YATT address, is not provided.

Since the YATT translation is stateless, the header translation can
happen anywhere in the network, e.g., as a bump in the stack at
either end, or within the network, e.g., at the routers that serve
the realms on the shaft. The shaft itself is expected to be dual
stack to forward packets in their native form, either v4 or v6.

For a legacy IPv4 node to communicate with YADA-enabled IPv4 node in
another realm, a NAT operation similar to NAT46 [NAT-DEPLOY], but
between IPv4 and YADA addresses, is required. The same would be
required to allow an IPv4-only YADA node to communicate with an IPv6
node a a non-YATT address.

In summary:

  *this specification does not allow any IPv4 legacy node to talk to
   any pure IPv6 node, and recognizes that this Graal may actually
   be a non-goal.

  *With YADA the current IPv4 Internet operations are not affected

*YADA extends the IPv4-reachable world by creating (millions of)
      parallel realms and changing (only) the stack on the hosts that
      require inter-realm communication and specific routers at the
      ingress of the realms

     *A YADA node can talk (using IPv4) to a YATT node (using IPv6)
      with a stateless translation. The translation can happen anywhere
      in the network or in the stack.

     *a YATT node being an IPv6 can talk to any other IPv6 nodes.

10. Backwards Compatibility

   YADA operation does not affect the intra-realm communication. The
   only affected stacks are the endpoints that communicate between
   realms leveraging YADA.

11. Security Considerations

   YADA introduces an IP-in-IP format that might be used to obfuscate
   an IP address impersonation performed in the inner header. A proper
   implemetation of BCP 38 should thus include the capability to
   recognize a YADA format and look in the source IP field that
   expresses the source realm in the inner header.

   Upgrading the rules in his Broadband Network Gateways (BNGs)
   represents additional work for an ISP, which should be done before
   the shaft addresses are routable within the ISP network, and whether
   the ISP intends to provide improved NAT functions in the home
   gateways and CPEs.

12. IANA Considerations

   This document requires the creation of a registry for IPv4 YADA
   realm prefixes, and the assignment of at least one YADA realm
   prefix.

   This document requires the creation of a registry for IPv4 YADA NAT
   prefixes, and the assignment of at least one YADA NAT prefix.

   This document requires the creation of a new record in the Resource
   Record (RR) TYPEs subregistry of the Domain Name System (DNS)
   Parameters. The new record would be of type AA meaning a YADA
   address.

13. Acknowledgments

   The author wishes to recognize the pioneer work done by Brian
   carpenter in the space of IPv4 augmentation with [I-D.carpenter-
   aeiou]

   The author wishes to thank Greg Skinner as the first reviewer/
   contributor to this work. Also Dave Bell, to remind that even if
   routing is not touched much inside an IPv4 realm vs. the current
   art, there is still work for the ISP, e.g., update the BCP 38 rules
   in the BNGs.

14. References

14.1. Normative References

[IPv4]   Postel, J., "Internet Protocol", STD 5, RFC 791, DOI
            10.17487/RFC0791, September 1981, <https://www.rfc-
            editor.org/info/rfc791>.

[INT-ARCHI] Braden, R., Ed., "Requirements for Internet Hosts -
            Communication Layers", STD 3, RFC 1122, DOI 10.17487/
            RFC1122, October 1989, <https://www.rfc-editor.org/info/
            rfc1122>.

[IPv6-ADDRESSING] Hinden, R. and S. Deering, "IP Version 6
            Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291,
            February 2006, <https://www.rfc-editor.org/info/rfc4291>.

[IPv6]   Deering, S. and R. Hinden, "Internet Protocol, Version 6
            (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/
            RFC8200, July 2017, <https://www.rfc-editor.org/info/
            rfc8200>.

14.2. Informative References

[NAT-DEPLOY] Palet Martinez, J., "Additional Deployment Guidelines
            for NAT64/464XLAT in Operator and Enterprise Networks",
            RFC 8683, DOI 10.17487/RFC8683, November 2019, <https://
            www.rfc-editor.org/info/rfc8683>.

[I-D.carpenter-aeiou]
            Carpenter, B. E., "Address Extension by IP Option Usage
            (AEIOU)", Work in Progress, Internet-Draft, draft-
            carpenter-aeiou-00, 21 March 1994, <https://
            datatracker.ietf.org/doc/html/draft-carpenter-aeiou-00>.

Author's Address

Pascal Thubert (editor)
Cisco Systems, Inc
Building D
45 Allee des Ormes - BP1200
06254 Mougins - Sophia Antipolis
France

Phone: +33 497 23 26 34
Email: pthubert@cisco.com