SixOne Router Implementation

An implementation story

Javier Ubillos Swedish Institute of Computer Sience
Box 1263
SE-164 29 Kista
jav@sics.se

ABSTRACT
ABSTRACT!

1. DESCRIPTION
SixOne Router introduces a method to solve the problem of rapidly increasing amount of routing prefixes in the Default Free Zone (DFZ). Routers in the DFZ are required to have a global knowledge of available routes. Theese routes are communicated using Border Gateway Protocol (BGP). However, BGP is scailing to an unsustainable point, the size of the forwarding tables are growing at an quadratic (or possibly exponential) rate. The wish of operators to multi-home, to be mobile and to be able to apply traffic engineering policies put added strain on the scalability. Alternatives to SixOne also present solutions within the same domain, by improving on the aggregation possibility and separating the entities performing routing- and forwarding-decisions. SixOne Router, however, does so without breaking backwards compatibility by making it a trivial case of transport rather than a special case as is the case for Map-n-Encap schemes. SixOne Router also divides the problem domain into two orthogonal problems, a transport problem and a resolution problem respectively. The division makes SixOne Router a pure transport solution, independent of the chosen resolution architecture allowing SixOne Router to function with an arbitrary resolution mechanism.

2. MOTIVATION
As the Internet core grows, so does the amount of possible path choices. The routing information in the DFZ is growing rapidly [Hus05]. Edge operators demand Provider Independent (PI) name spaces for multi-homing, mobility and intra domain traffic engineering. This causes them to increasingly de-aggregate their name space, hence exacerbating the growth of the global routing tables, the global forwarding tables as well as the rate of update and withdrawal messages [PS06, Ful07]. A core router today must maintain all information for routing decision to be able to forward the packets correctly. Doing this using BGP, requires the router to maintain a full Routing Information Base (RIB), of which it needs to compute and compile the necessary Forwarding Information Base (FIB). The routers are burdened with both maintaining and computing the RIB/FIB as well as forwarding individual packets. One node is hence carrying two different functions. We believe that these functions are orthogonal and can be better performed by two separate entities.

3. RELATED WORK
3.1 Transport mechanism
3.1.1 LISP
3.1.2 INLP
3.1.3 IVIP
3.2 Resolution mechanisms
3.2.1 APT
3.2.2 NERD
3.2.3 LISP-DHT

4. FEATURES
4.1 Name space division
SixOne separates the available addresses into two name-spaces, one transit network name space with Provider Aggregatable (PA) addresses and one edge network name space with PI addresses (see figure 1). The transit name space is dedicated to the core routers while the edge name space is used by edge operators and end hosts. Each edge router maps one transit address range to an edge address range address range. A global mapping service maintains the routers mappings between the transit domain and the edge domains. A set of routers for a multi homed edge network may simply register the same edge network with multiple transit addresses. With this split the DFZs aggregatable addressing structure

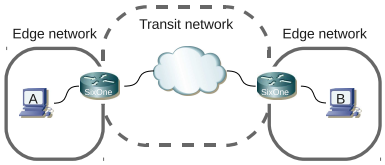
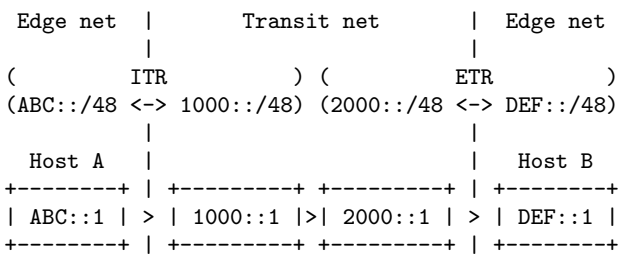


Figure 1: Division of name spaces.

is maintain yet providing edge operators with the PI address space required for multi-homing and mobility. This limits the size of the DFZ, and hence the weight BGP places on the core routers.

4.2 Translation

The mapping is implemented through address translation. In the case of communication between hosts in upgraded networks, the end-to-end semantics are maintained as the original source and destination addresses are restored at the receiving Egress Translation (or Tunnel) Router (ETR).



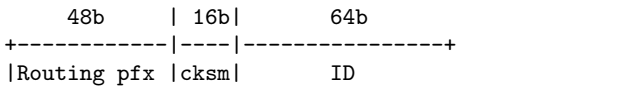
Translation has the advantages of

- Lower packet overhead [Vog08]
- Innate backwards compatibility

A single IPv6 packet is normally 40 bytes long equaling to the same size as two IPv4 packets (20 bytes long each). This would lead us to believe that tunneling (IPv4 over IPv4) has the same overhead as translation (using IPv6), however, IPv6 performs MTU discovery automatically (without new added complexity) while IPv4 might fragment packets along its path, adding overhead as translation does not change the total packet size, while tunneling increases it without the originating hosts knowledge.

4.3 Address structure

The IP headers are syntactically identical, however semantically the address fields are used slightly different. In the case of communicating with a legacy host, the whole address is computed as an traditional IP address. The legacy host does not need to be aware of any new or overloaded semantics.



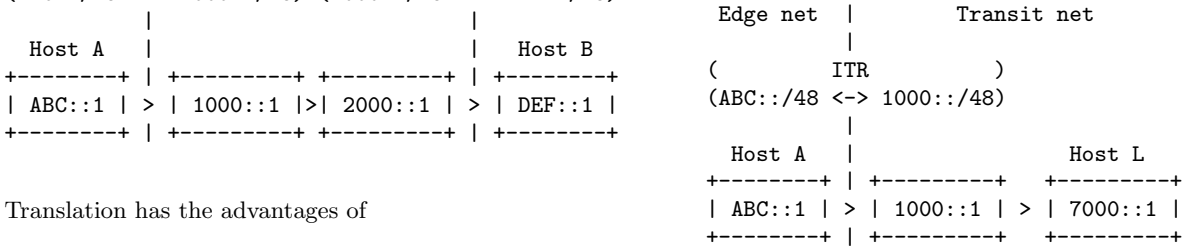
The first 48 bits are the routing prefix. This section specifies which network this packet is addressed to. In the case for edge networks, this denotes which edge network the client is located in. The transit address case is divided into two possibilities. Either it is the routing prefix denoting a particular Six/One router, in other words, an entry point to an edge network or in the legacy host case, it is computed as part of a traditional IP address.

The middle part, bits 49 to 64 is a reserved 16 bit field used to maintain checksum neutrality, described in section 4.5.

The last 64 bits are the host identifier. This section does not change through a packets path. It uniquely, for an edge net, identifies a host.

4.4 Backwards compatibility

When using translation rather than tunneling, backwards compatibility becomes a trivial case of translation. All hosts in legacy networks are simply located in the transit domain. The Six/One router simply behaves as a Network Address Translation (NAT) node. Each host in the upgraded network does receive an unique address in the mapping. Please note that the example below is simplified, the real world case would also modify the external address (1000::1) as specified by the checksum neutral IP (see section 4.5).



4.5 Checksum neutral IPs

During translation, to adhere to the checksum schemes of protocols such as e.g TCP, UDP or ICMP, the SixOne router implements a checksum neutral IP rewriting scheme. Each edge address is mapped to it's checksum neutral transit correspondent.

Through out this paper, for simplicity reasons, we have insisted that edge address ABC::1 is mapped to transit address 1000::1. This translation would require re-computation of the checksums. When using checksum neutral IPs the mapping is shifted in a manner which maintains checksum neutrality between the two mapped addresses.

$$\text{Checksum}(\texttt{ABC::1}) \equiv \text{Checksum}(\texttt{1000:0:0:F9BB::1})$$

The checksum neutrality assumes the use of 16-bit one's complement, as is the case for ICMP, TCP and UDP. Employing this scheme allows the implementation of Six/One router to be protocol agnostic. It means that in the communication with legacy hosts, it does not need to implement any functionality for special treatment for certain protocols due to the change of the IP addresses. In the upgraded network to upgraded network this becomes an non-issue.

4.6 Mapping resolution system

The stance of SixOne is that the transport mechanism and that the mapping resolution mechanism are two orthogonal problems, which should be solved separately. This approach allows us to join the best transport system with the best mapping resolution system. The SixOne Router leaves this part of the implementation open. The interface consists of a single function `retrieve_mappings`. The input consists of the edge IP we need a transit address for. The output consists of a list with transit addresses which are available to reach the edge network.

4.6.1 Default resolution system

The current implementation of the SixOne Router implements a simplified resolution function. It is not intended as a suggestion to how to solve the mapping function. If the

`sixone_resolv` function pointer is *null*, the default resolution system is called. The default resolution is a plain text file `mappings.txt`=. The format of the file is illustrated by the file `mappings.txt.sample`.

```
abc::/64 1000::
def::/64 1100::
```

The format is `<edge net>/<prefix length><whitespace> <transit net>`. Resolution can be performed in both directions, from edge network address to transit network address and vice versa.

4.7 Traffic Control (Resolution Policies)

Traffic control policies can be implemented when selecting which address to resolve to. Any given packet may be delivered through any of the available Ingress Translation (or Tunnel) Router (ITR)s and/or ETRs (see Figure 2

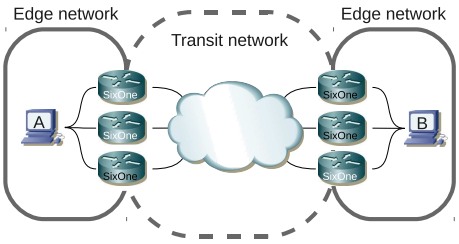


Figure 2: Multihomed sites may implement traffic control policies.

Traffic control policies are implemented through the policy interface. The policy interface is divided into two functions to be implemented:

```
policy_pic_dst
policy_pick_src
```

Both are function pointers stored as members in the struct.

```
global_settings->policy
```

Both have the same input (ip_list) and output (sixone_ip). If either is set to `null` (which they are by default), their respective defaults are called.

4.7.1 Default Resolution Policies

If either

```
verb=global_settings->policy->sixone_policy_src
global_settings->policy->sixone_policy_dst
```

is set to `null`, their respective default implementation is executed. Either default function is implemented in the same trivial manner, select (return) the first element in the provided list of paths.

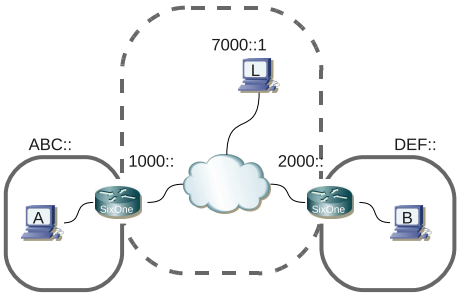


Figure 3: Scenario setup.

5. SCENARIO

The test scenario consists of three hosts. Two residing in separate upgraded edge networks and one legacy host located in the transit domain. The two upgraded networks are each separated from the transit domain by a SixOne router.

5.1 Setup

The three hosts are labeled A,B and L. A and B are the hosts in the respective upgraded networks. The IP address mappings are:

(a) Edge: ABC::/48 ↔ Transit 1000::/48

(b) Edge: DEF::/48 ↔ Transit 2000::/48

(c) Legacy host 7000::1

Both host A and host B are located in the upgraded networks.

- A in the ABC::/48 network
- B in the DEF::/48 network

Their respective edge-domain numbering are 1000::/48 and 2000::/48.

5.2 VMware

5.2.1 Clients

5.2.2 Routers

5.2.3 Interconnection

6. ROUTER IMPLEMENTATION

The sixone router is implemented as an user space program in FreeBSD 7.0.

6.0.4 Mapping resolution

The mapping resolution interface consists of the following three functions:

```
ip_list retrieve_mappings(sixone_ip ip,
                          u_int only_sixone);
sixone_ip policy_pick_src(ip_list list);
sixone_\ip policy_pick_dst(ip_list list);
```

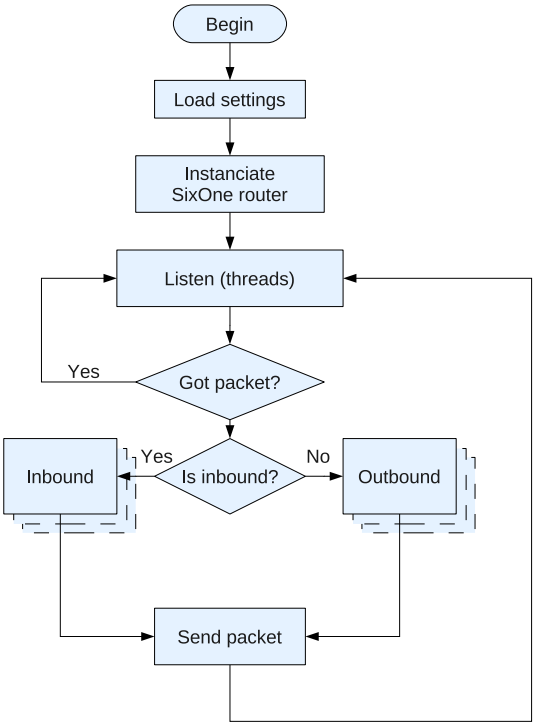


Figure 4: Main flow chart.

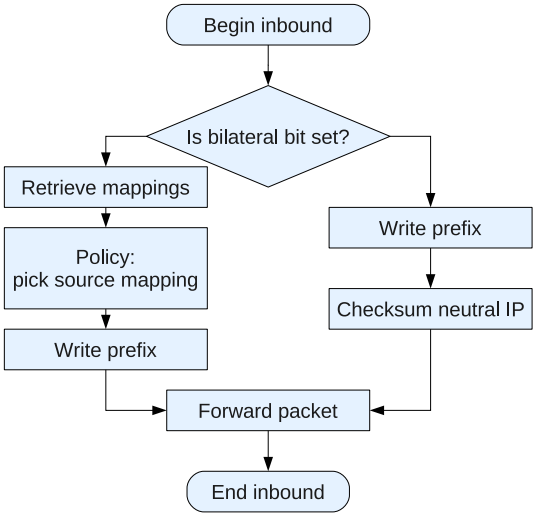


Figure 5: Inbound flow chart.

All three functions check whether or not their respective function pointer is set to NULL or not. If it is not, the pointed to function is executed and it's return value is returned.

The three pointers and respective formats are:

```
global_settings->resolv->sixone_resolv
ip_list (*sixone_resolv)(sixone_ip ip,
                         u_int only_sixone);

global_settings->policy->sixone_policy_dst
sixone_ip (*sixone_policy_dst)(ip_list list);

global_settings->policy->sixone_policy_src
sixone_ip (*sixone_policy_src)(ip_list list);
```

6.1 State diagram

6.1.1 Inbound

6.1.2 Outbound

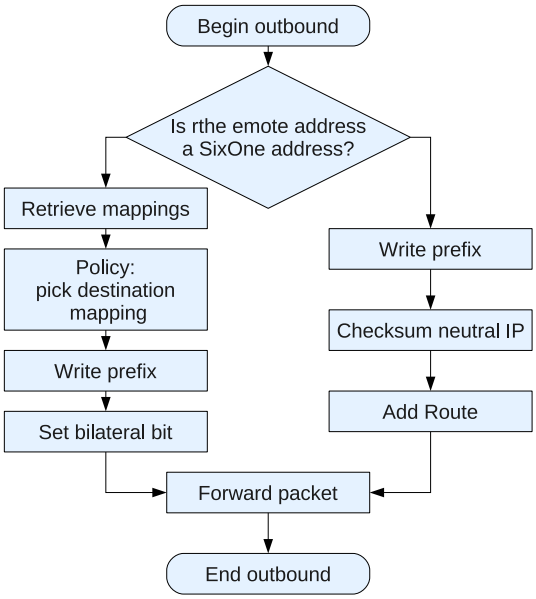


Figure 6: Outbound flow chart.

7. REFERENCES

[Ful07] Vince Fuller, 02 2007, http://www.vaf.net/ vaf/apricotÂ■plenary.pdf.

[Hus05] Geoff Huston, *A bgp year in review*, 03 2005.

[PS06] Mike Hughes Philip Smith, Rob Evans, *Ripe routing working group recommendations on route aggregation*, 12 2006, `http://www.ripe.net/ripe/docs/ripe-399.html`.

[Vog08] Christian Vogt, *Six/One Router - Design and Motivation*, July 2008.