An IPv6 implementation of the link-local metadata service for Tungsten Fabric

By Matvey Kraposhin (mkraposhin@gmail.com)

Objectives

To implement in Tungsten Fabric an access to the OpenStack Metadata service via IPv6 protocol

Until now TF allows to access this service only via IPv4 protocol (169.254.169.254)

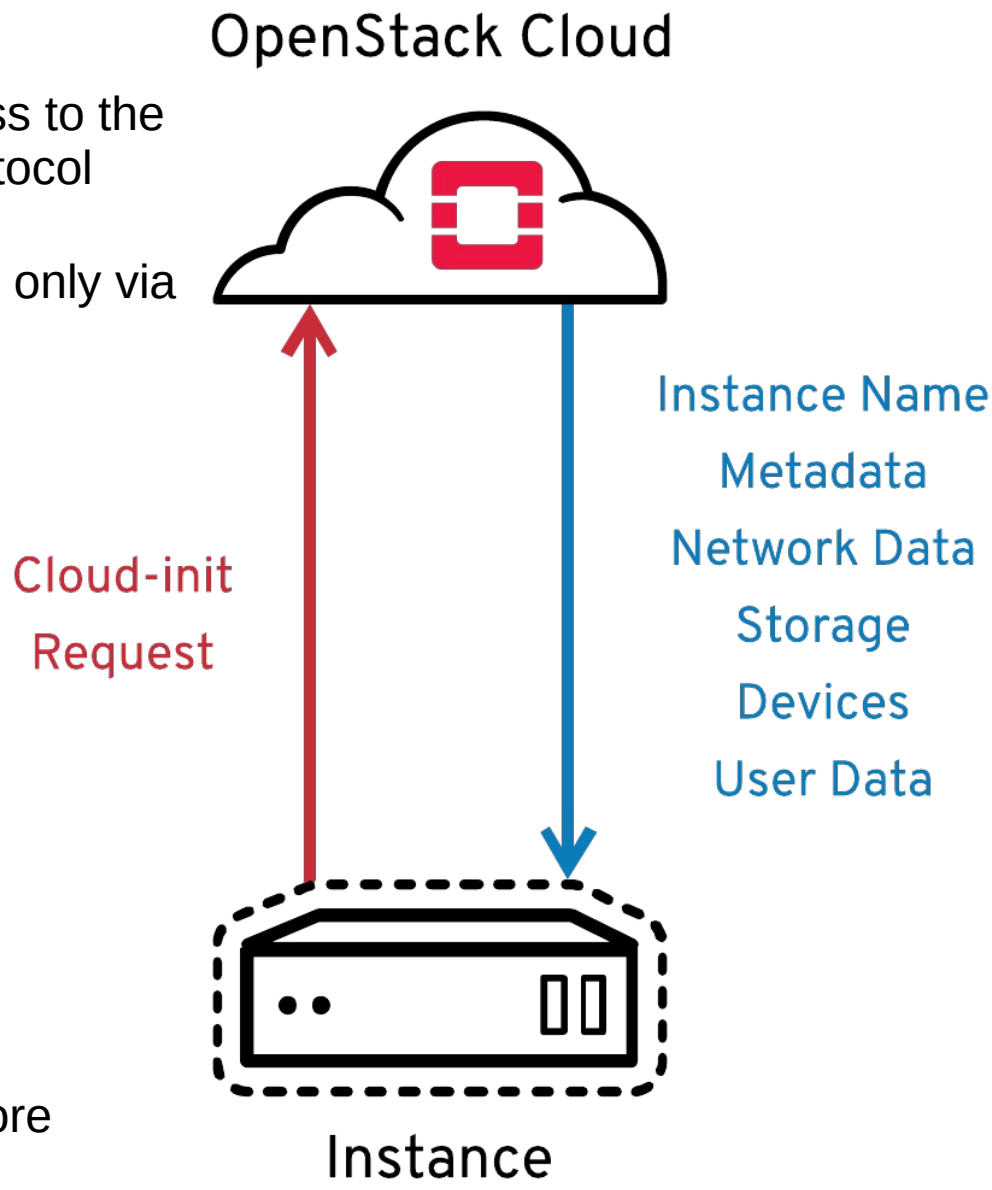
Motivation

1. OpenStack Metadata service

provides a way for instances to retrieve instance-specific data via a REST API. It can use both IPv4 and IPv6 stacks:
- 169.254.169.254
- fe80::a9fe:a9fe

2. Most of other TF' link local services can use both IPv4 and IPv6

3. IPv6 is not a technology of future anymore



TF Link Local Services

DNS
DNS6
DHCP
DHCP6
ICMP
ICMP6

General idea.
Intercept an incoming message, generate a reply to it and send the reply back.

Incoming message is intercepted at packet analysis stage when new flow is created by request from vrouter

Work at L3/L2 levels

Metadata

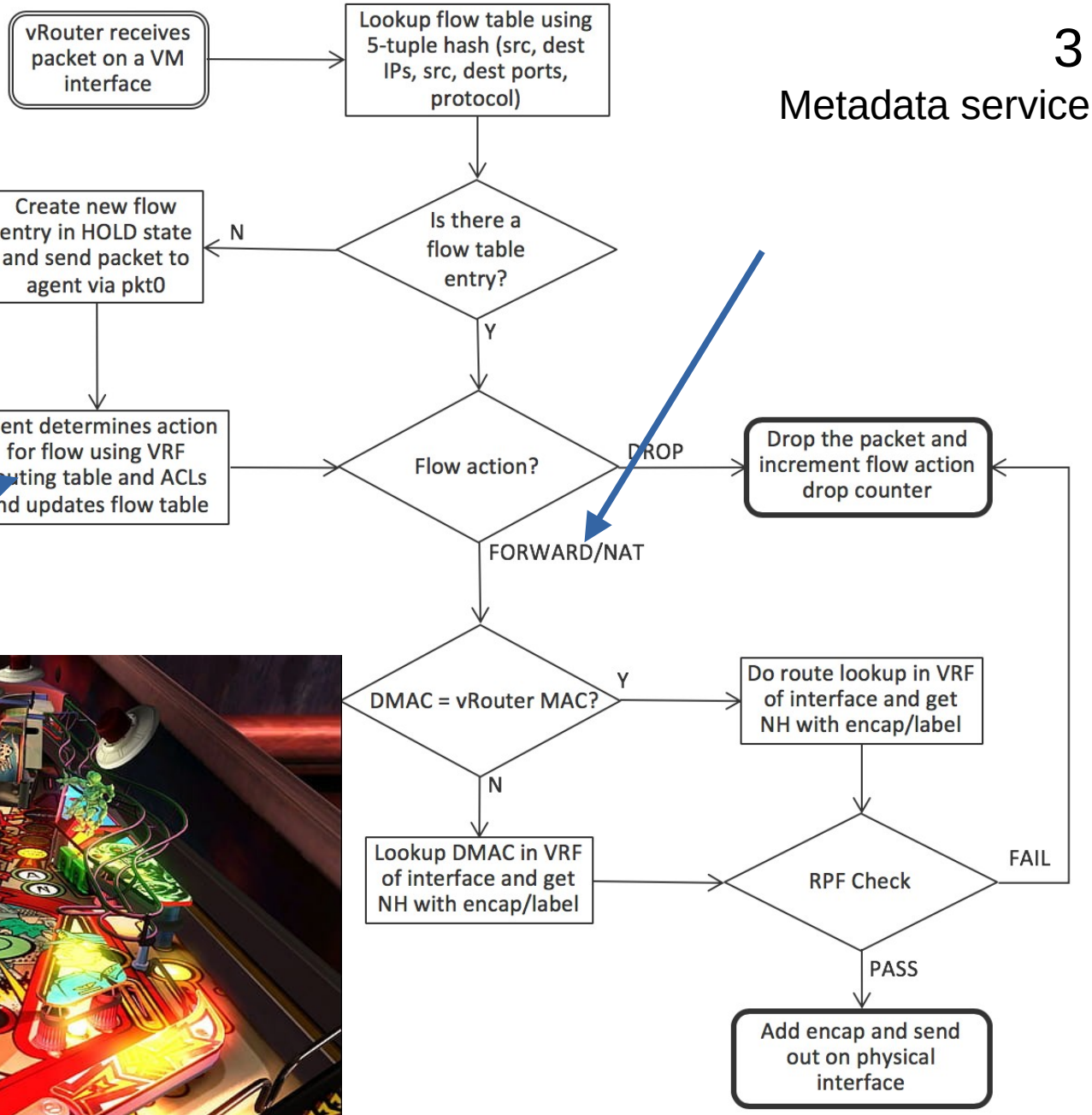
General idea.
Make a proxy: create 2 communicating TCPconnections (one for interaction with the actual client, second for interaction with Nova)

But: to connect over- and underlay networks port and network d/s address translation is used

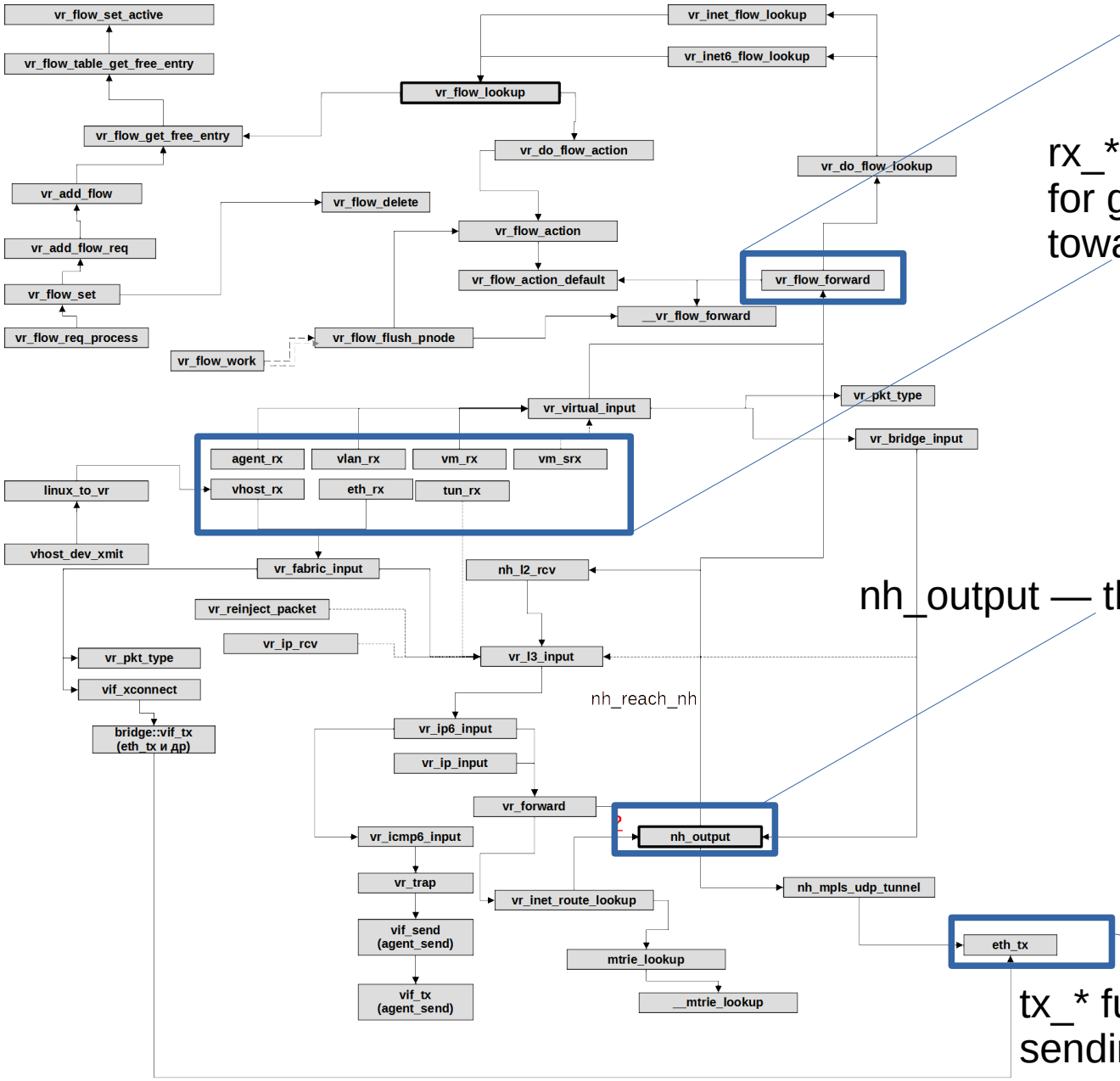
Work at L4/L3 levels

Data flow in TF

DNS, DHCP, ICMP, and other similar services



Data flow in vrouter: a detailed picture



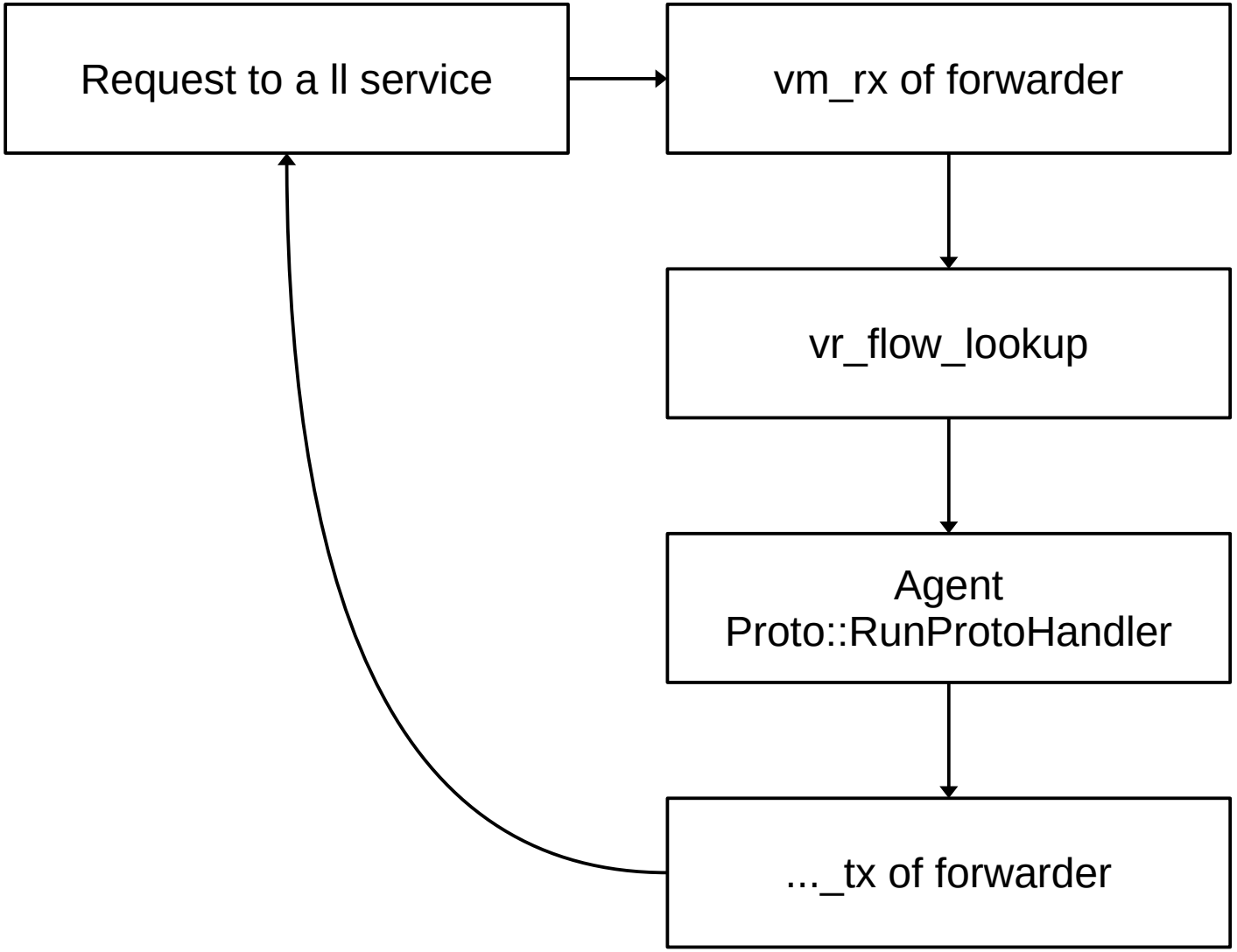
flows are managed here

rx_* functions are responsible for guiding packets from input toward the distribution point

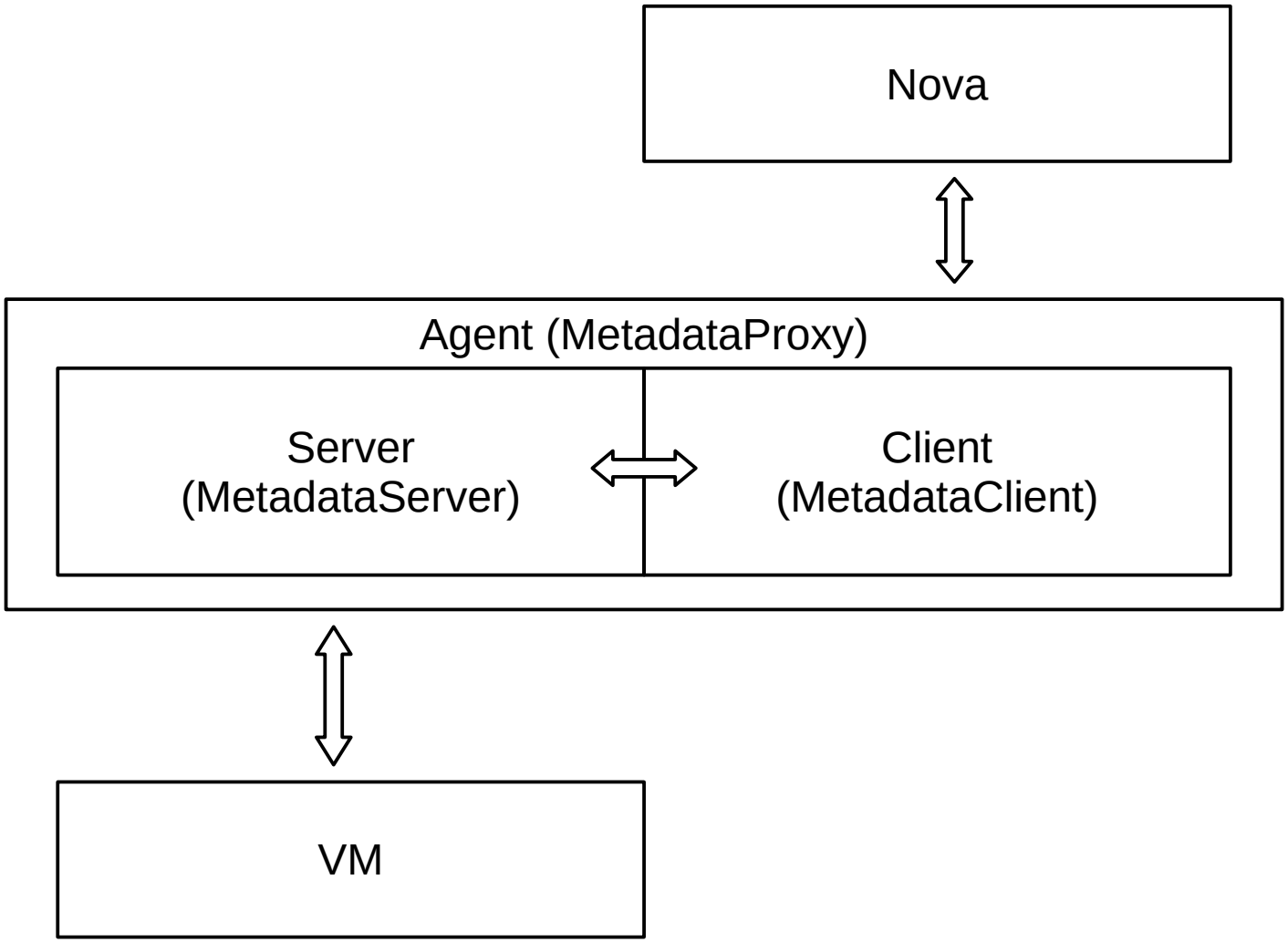
nh_output — the distribution point

tx_* functions are responsible for sending data outwards of forwarder

Data flux within majority of TF ll services



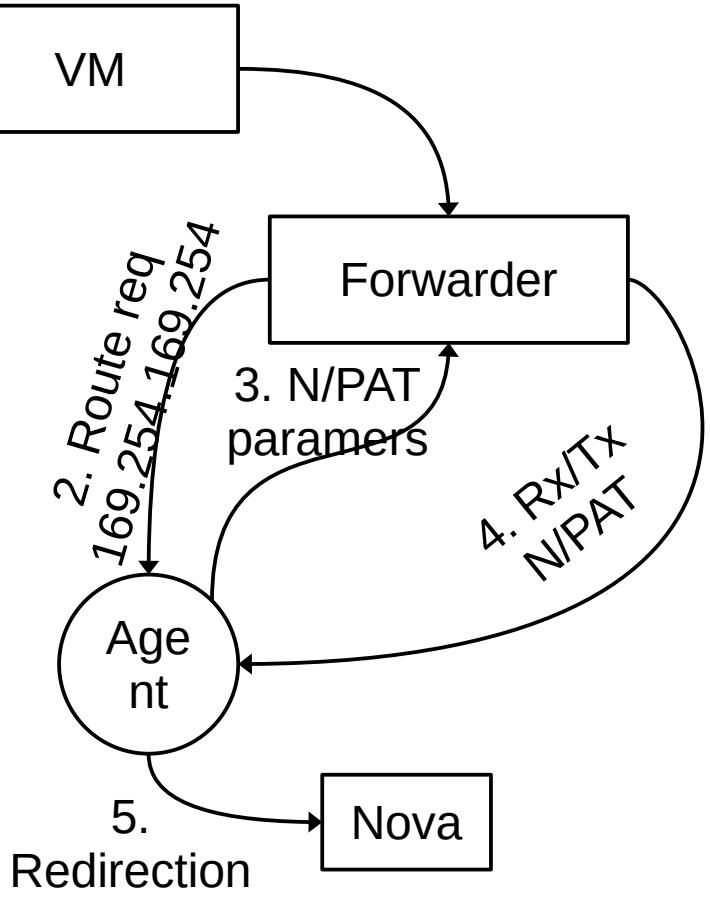
Data flux within the Metadata TF ll service

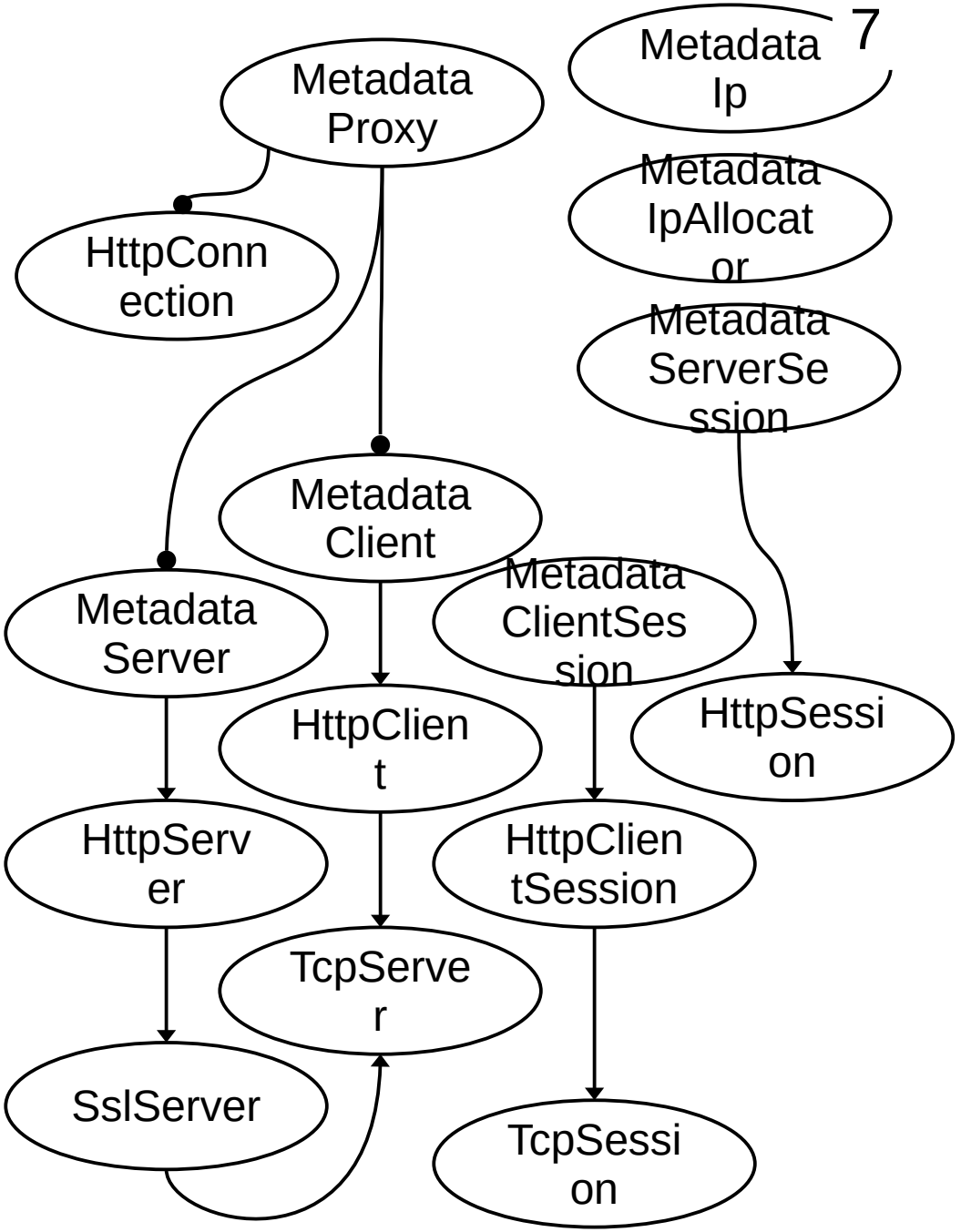


Challenges

1. Modification at both L4 and L3 levels, including NAT(6). TF doesn't still have NAT6

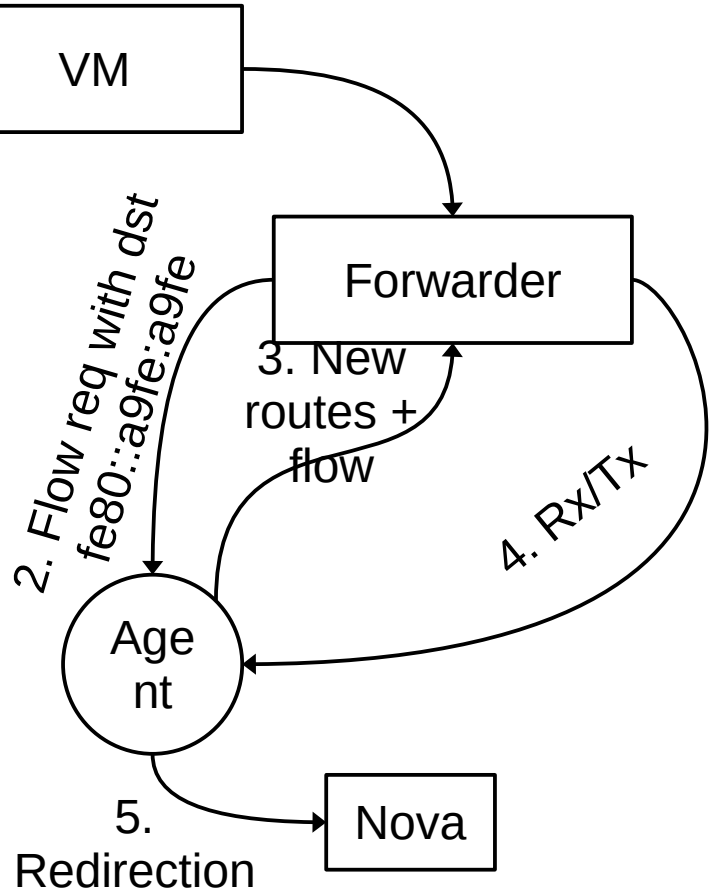
2. Complex relations between classes used to redirect requests to NOVA and to retrieve reply back





Demanded code changes

Key idea: use package analysis to detect metadata requests (like in DNS, DHPC, etc) and TCP proxying to exchange data between user and NOVA



Changes encompass next parts of Vrouter Agent:

- MetaDataProxy (Http/Tcp servers for IPv4 and IPv6)
- InterfaceTable (find creds usign ll ipv6 address)
- VmInterface (Routes announcements)
- PktFlowInfo (packet interception, routes announcement)
- TcpServer (support for IPv6)
- address_util.cc: support for IPv6 in ResolveCanonicalNameIPv6(…)
-GlobalVrouter (announcement of routes to vhost0 for __default__ vrf)

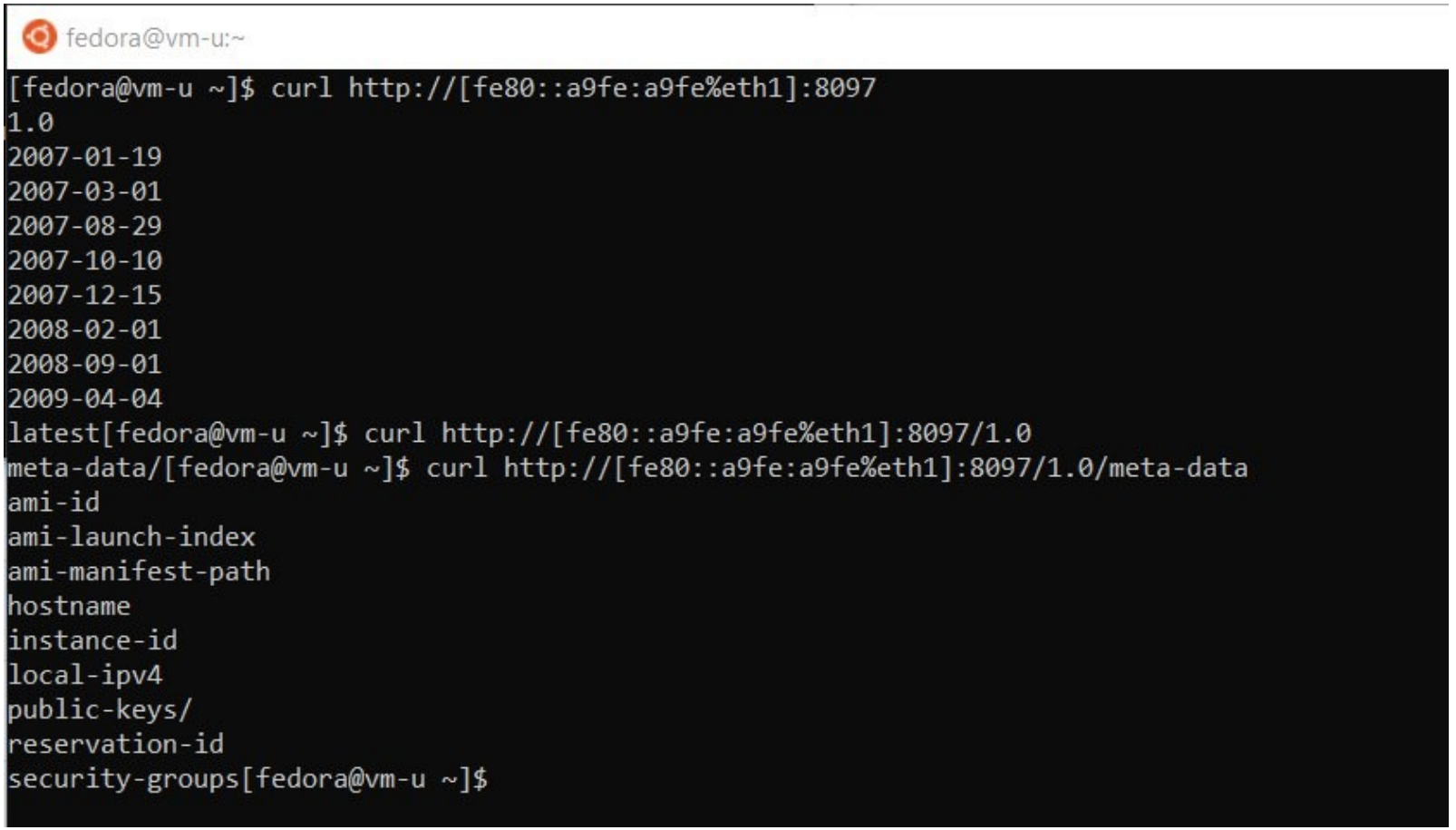
Source code to intercept incoming metadata request


```
void PktFlowInfo::IngressProcess(const PktInfo *pkt, PktControlInfo *in,
                                 PktControlInfo *out) {
```

```
if (pkt->ip_saddr.is_v6())
    std::cout << "IngressProcess" << std::endl;
MetadataProxy *metadata_proxy = agent->services()->metadataproxy();
if(metadata_proxy && pkt->ip_saddr.is_v6()
   && pkt->ip_daddr.to_string() == metadata_proxy->Ipv6ServiceAddress().to_string()) {
   Ip6Address ll_ip = pkt->ip_saddr.to_v6();
   std::cout<< "A request to fe80::a9fe:a9fe"
            << " from " << ll_ip.to_string()
            << " with vrf " << (in->vrf_ ? in->vrf_->GetName() : "NONE") << std::endl;
    //Step 1. Check port
    uint16_t nova_port, linklocal_port;
    Ip4Address nova_server, linklocal_server;
    std::string nova_hostname;
    if (agent->oper_db()->global_vrouter()->FindLinkLocalService(
        GlobalVrouter::kMetadataService, &linklocal_server, &linklocal_port,
        &nova_hostname, &nova_server, &nova_port))
    {
        std::cout << "Reseting port to: " << linklocal_port << std::endl;
        metadata_proxy->ResetIp6Server(linklocal_port);
    }
    //Step 2.
    metadata_proxy->AnnounceMetaDataLinkLocalRoutes(vm_port,
        ll_ip, in->vrf_);
}
```

Current state

1) Preliminary version has been implemented
2) It allows to access Nova metadata service via TF IPv6 stack



Outcomes & Future work

Outcomes:
- introduction of a new feature into TF
- development of other http/tcp services on top of link local networks
- TF programming tutorial

Future work:
- code cleaning
- employment of netlink messages to add ll neighbours
- constants removal (vrf names, plen’s, etc)
- feature testing
- TF programming tutorial (possibly)
- FreeBSD testing?