Develop-NAT技术学习

姓名：宋延超
日期：2019-11-28

一、NAT44

- 从18.07版开始，NAT44代码已拆分为原始NAT44和NAT44的其他功能-**端点依赖模式(P2P)**。
- **NAT44端点依赖模式可为某些功能所需的所有会话启用终结点依赖过滤和映射**。现在，某些现有功能（例如服务负载平衡，两次nat，仅限于out2in的静态映射，未知协议动态翻译以及带有动态翻译的转发功能）仅在端点依赖模式下可用。**端点依赖模式使用6元组**（源IP地址，源端口，目标IP地址，目标端口，协议，FIB表索引）会话哈希表键，而**不是4元组**（源IP地址，源端口，协议，FIB表索引））。
- 要启用NAT插件端点相关模式，请在**statrup config**中添加以下内容：

```
    nat{端点相关}
```

二、API

- show NAT plugin startup config（显示NAT插件启动配置）：

```
define nat_show_config {
 u32 client_index;
 u32 context;
};
define nat_show_config_reply {
 u32 context;
 i32 retval;
 u8 static_mapping_only;
 u8 static_mapping_connection_tracking;
 u8 deterministic;
 u32 translation_buckets;
 u32 translation_memory_size;
 u32 user_buckets;
 u32 user_memory_size;
```

```
 u32 max_translations_per_user;
 u32 outside_vrf_id;
 u32 inside_vrf_id;
};
```

- set NAT plugin workers(设置NAT插件工作程序):

```
define nat_set_workers {
 u32 client_index;
 u32 context;
 u64 worker_mask;
};
```

- dump NAT plugin workers(dump NAT plugin workers:):

```
define nat_worker_dump {
 u32 client_index;
 u32 context;
};
define nat_worker_details {
 u32 context;
 u32 worker_index;
 u32 lcore_id;
 u8 name[64];
};
```

- enable/disable NAT IPFIX logging(启用/禁用NAT IPFIX日志记录):

```
define nat_ipfix_enable_disable {
 u32 client_index;
 u32 context;
 u32 domain_id;
 u16 src_port;
 u8 enable;
};
```

- add/delete NAT44 address range (twice_nat endpoint dependent mode only)(添加/删除NAT44地址范围（仅适用于twice_nat端点依赖模式）):

```
define nat44_add_del_address_range {
 u32 client_index;
```

```
 u32 context;
 u8 first_ip_address[4];
 u8 last_ip_address[4];
 u32 vrf_id;
 u8 twice_nat;
 u8 is_add;
};
```

- dump NAT44 addresses(转储NAT44地址):

```
define nat44_address_dump {
 u32 client_index;
 u32 context;
};
define nat44_address_details {
 u32 context;
 u8 ip_address[4];
 u8 twice_nat;
 u32 vrf_id;
};
```

- enable/disable NAT44 feature on the interface(在接口上启用/禁用NAT44功能):

```
define nat44_interface_add_del_feature {
 u32 client_index;
 u32 context;
 u8 is_add;
 u8 is_inside;
 u32 sw_if_index;
};
```

- dump interfaces with NAT44 feature(具有NAT44功能的转储接口):

```
define nat44_interface_dump {
 u32 client_index;
 u32 context;
};
define nat44_interface_details {
 u32 context;
 u8 is_inside;
 u32 sw_if_index;
```

```
};
```

- add/delete 1:1 NAT (twice_nat/out2in_only endpoint dependent mode only)(添加/删除1：1 NAT（仅twice_nat / out2in_only端点相关模式）):

```
define nat44_add_del_static_mapping {
 u32 client_index;
 u32 context;
 u8 is_add;
 u8 addr_only;
 u8 local_ip_address[4];
 u8 external_ip_address[4];
 u8 protocol;
 u16 local_port;
 u16 external_port;
 u32 external_sw_if_index;
 u32 vrf_id;
 u8 twice_nat;
 u8 out2in_only;
 u8 tag[64];
};
```

- dump 1:1 NAT(静态NAT):

```
define nat44_static_mapping_dump {
 u32 client_index;
 u32 context;
};
define nat44_static_mapping_details {
 u32 context;
 u8 addr_only;
 u8 local_ip_address[4];
 u8 external_ip_address[4];
 u8 protocol;
 u16 local_port;
 u16 external_port;
 u32 external_sw_if_index;
 u32 vrf_id;
 u8 twice_nat;
 u8 out2in_only;
 u8 tag[64];
};
```

- add/delete NAT44 pool address from specific interface (twice_nat endpoint dependent mode only):
  从特定接口添加/删除NAT44池地址（仅适用于twice_nat端点依赖模式）：

```
define nat44_add_del_interface_addr {
 u32 client_index;
 u32 context;
 u8 is_add;
 u8 twice_nat;
 u32 sw_if_index;
};
```

- dump NAT44 pool addresses interfaces(转储NAT44池地址接口):

```
define nat44_interface_addr_dump {
 u32 client_index;
 u32 context;
};
define nat44_interface_addr_details {
 u32 context;
 u32 sw_if_index;
 u8 twice_nat;
};
```

- dump NAT44 users(转储NAT44用户):

```
nat44_user_dump {
 u32 client_index;
 u32 context;
};
define nat44_user_details {
 u32 context;
 u32 vrf_id;
 u8 ip_address[4];
 u32 nsessions;
 u32 nstaticsessions;
};
```

- dump NAT44 user's sessions(转储NAT44用户的会话):

```
define nat44_user_session_dump {
 u32 client_index;
 u32 context;
 u8 ip_address[4];
 u32 vrf_id;
};
define nat44_user_session_details {
 u32 context;
 u8 outside_ip_address[4];
 u16 outside_port;
 u8 inside_ip_address[4];
 u16 inside_port;
 u16 protocol;
 u8 is_static;
 u64 last_heard;
 u64 total_bytes;
 u32 total_pkts;
 u8 is_twicenat;
 u8 ext_host_valid;
 u8 ext_host_address[4];
 u16 ext_host_port;
 u8 ext_host_nat_address[4];
 u16 ext_host_nat_port;
};
```

- enable/disbale NAT44 as an interface output feature (postrouting in2out translation):
  启用/禁用NAT44作为接口输出功能（路由后in2out转换）：

```
define nat44_interface_add_del_output_feature {
 u32 client_index;
 u32 context;
 u8 is_add;
 u8 is_inside;
 u32 sw_if_index;
};
```

- dump interfaces with NAT44 output feature(具有NAT44输出功能的转储接口):

```
define nat44_interface_output_feature_dump {
 u32 client_index;
 u32 context;
```

```
};
define nat44_interface_output_feature_details {
 u32 context;
 u8 is_inside;
 u32 sw_if_index;
};
```

- Add/delete NAT44 static mapping with load balancing (endpoint dependent mode only):
  使用负载平衡添加/删除NAT44静态映射（仅端点相关模式）：

```
typeonly define nat44_lb_addr_port {
 u8 addr[4];
 u16 port;
 u8 probability;
};
define nat44_add_del_lb_static_mapping {
 u32 client_index;
 u32 context;
 u8 is_add;
 u8 external_addr[4];
 u16 external_port;
 u8 protocol;
 u32 vrf_id;
 u8 twice_nat;
 u8 out2in_only;
 u8 tag[64];
 u8 local_num;
 vl_api_nat44_lb_addr_port_t locals[local_num];
};
```

- Dump NAT44 static mapping with load balancing(使用负载平衡转储NAT44静态映射):

```
define nat44_lb_static_mapping_dump {
 u32 client_index;
 u32 context;
};
define nat44_lb_static_mapping_details {
 u32 context;
 u8 external_addr[4];
 u16 external_port;
 u8 protocol;
 u32 vrf_id;
```

```
 u8 twice_nat;
 u8 out2in_only;
 u8 tag[64];
 u8 local_num;
 vl_api_nat44_lb_addr_port_t locals[local_num];
};
```

- Delete NAT44 session(删除NAT44会话):

```
define nat44_del_session {
 u32 client_index;
 u32 context;
 u8 is_in;
 u8 address[4];
 u8 protocol;
 u16 port;
 u32 vrf_id;
 u8 ext_host_valid;
 u8 ext_host_address[4];
 u16 ext_host_port;
};
```

- Add/delete NAT44 identity mapping(添加/删除NAT44身份映射):

```
define nat44_add_del_identity_mapping {
 u32 client_index;
 u32 context;
 u8 is_add;
 u8 addr_only;
 u8 ip_address[4];
 u8 protocol;
 u16 port;
 u32 sw_if_index;
 u32 vrf_id;
 u8 tag[64];
};
```

- Dump NAT44 identity mappings(转储NAT44身份映射):

```
define nat44_identity_mapping_dump {
 u32 client_index;
```

```
 u32 context;
};
define nat44_identity_mapping_details {
 u32 context;
 u8 addr_only;
 u8 ip_address[4];
 u8 protocol;
 u16 port;
 u32 sw_if_index;
 u32 vrf_id;
 u8 tag[64];
};
```

三、命令行

```
set interface nat44 in <intfc> out <intfc> [output-feature] [del]
show nat44 interfaces
nat44 add address <ip4-range-start> [- <ip4-range-end>] [tenant-vrf <vrf-id>]
 [twice-nat] [del]
show nat44 addresses
nat44 add static mapping tcp|udp|icmp local <ip4-addr> [<port>] external (<ip
4-addr>|<intfc>) [<port>] [vrf <table-id>] [twice-nat] [out2in-only] [del]
nat44 add load-balancing static mapping protocol tcp|udp external <addr>:<por
t> local <addr>:<port> probability <n> [vrf <table-id>] [twice-nat] [out2in-o
nly] [del]
nat44 add identity mapping <interface>|<ip4-addr> [<protocol> <port>] [vrf <t
able-id>] [del]
show nat44 static mappings
set nat workers <workers-list>
show nat workers
nat ipfix logging [domain <domain-id>] [src-port <port>] [disable]
nat44 add interface address <interface> [twice-nat] [del]
show nat44 interface address
nat44 del session in|out <addr>:<port> tcp|udp|icmp [vrf <id>] [external-host
 <addr>:<port>]
show nat44 sessions [detail]
nat addr-port-assignment-alg default | map-e psid <n> psid-offset <n> psid-le
n <n> | port-range <start-port> - <end-port>
nat44 forwarding enable|disable
```

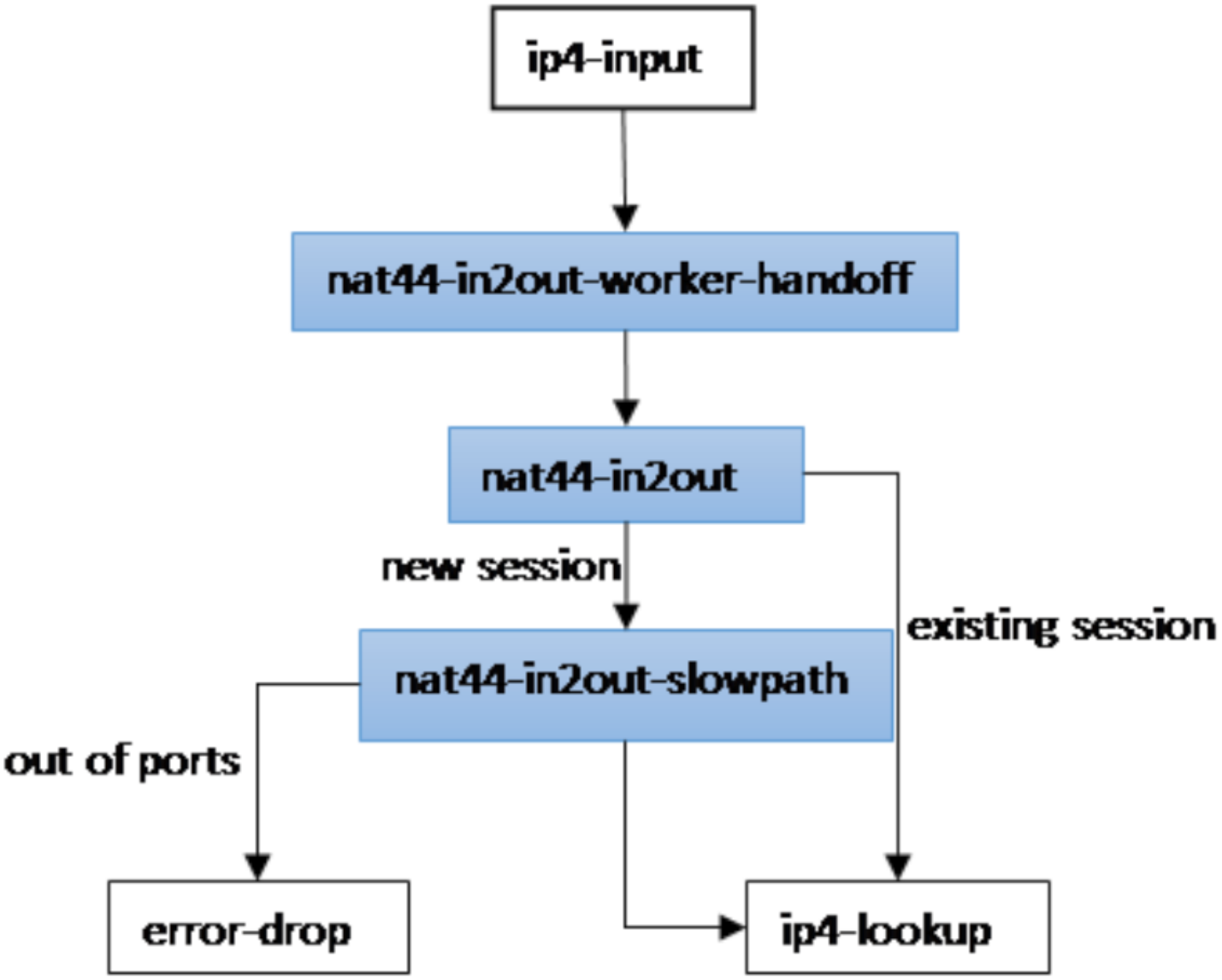
四、Startup config

```
translation hash buckets <n> - default 1024 (maximum sessions = 10 x <n>), nu
mber of buckets in session lookup hash tables
translation hash memory <n> - default 128<<20, memory size of session lookup
hash tables
user hash buckets <n> - default 128, number of buckets in NAT user lookup has
h table
user hash memory <n> - default 64<<20, memory size of NAT user lookup hash ta
ble
max translations per user <n> - default 100
outside VRF id <table-id> - default 0
inside VRF id <table-id> - default 0
static mapping only [connection tracking] - default dynamic traslations enabl
ed
deterministic - deterministic NAT/CGN
endpoint-dependent - endpoint dependent mode (6-tuple session key)
```

- A good rule of thumb is that "user hash buckets" is set as expected_number_of_users/4 and "translation hash buckets" as (expected_number_of_users * max_translation_per_user)/4. The amount of memory selected should easily contain all of the records, with a generous allowance for hash collisions. Hash memory is allocated separately from the main heap, and won't cost anything except kernel PTE's until touched, so it's OK to be reasonably generous.
- 根据经验，“**用户哈希桶”设置为expected_number_of_users/4**，“**翻译哈希桶”设置为(expected_number_of_users * max_translation_per_user)/4**。所选的内存量应该很容易包含所有的记录，并允许大量的散列冲突。散列内存是与主堆分开分配的，在触及内核PTE之前，除了内核PTE之外，不会消耗任何东西，所以适当地使用散列内存是可以的。

五、NAT44数据包路径

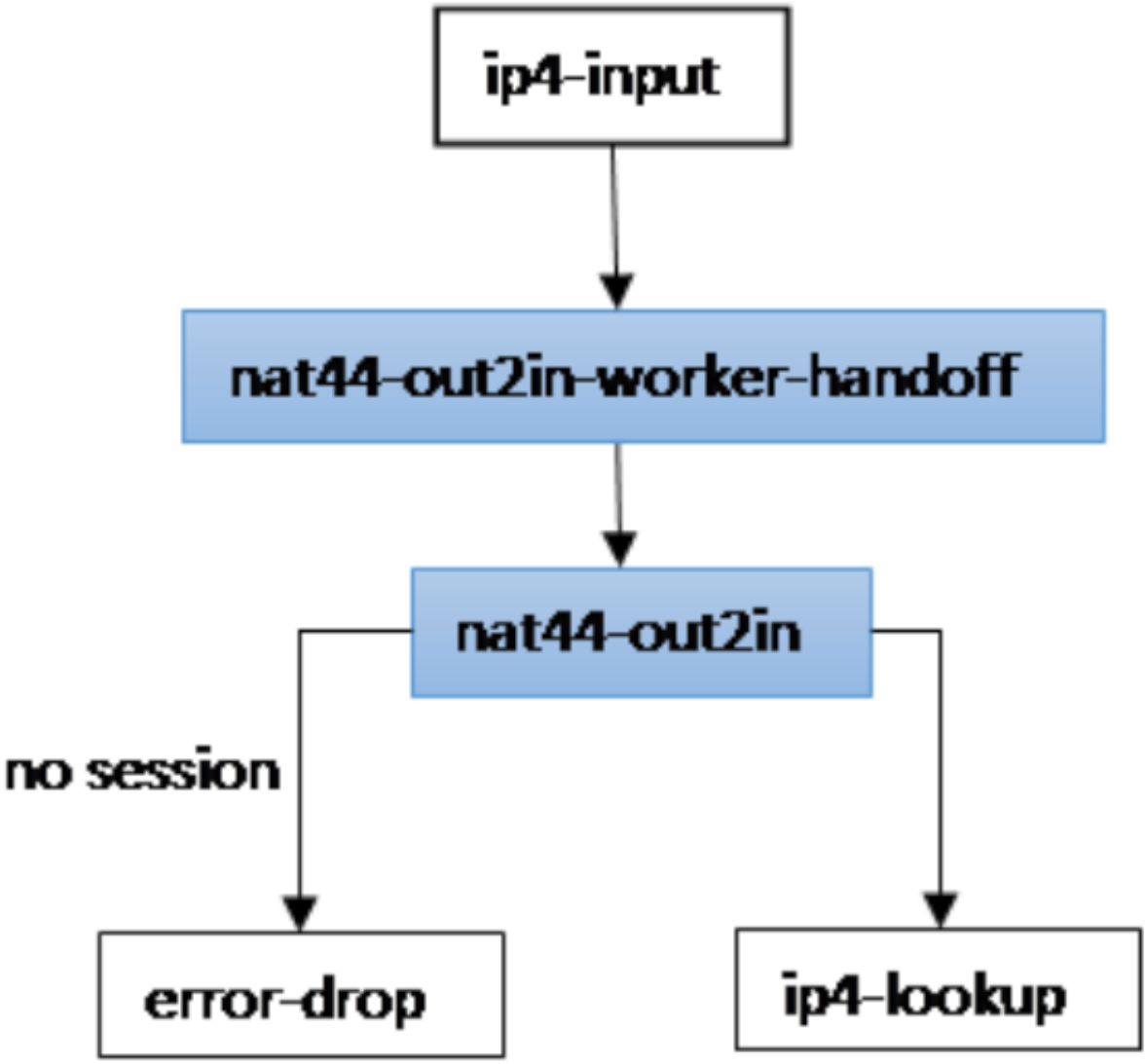
1.in2out（接收非翻译包）



- **节点nat44-in2out-worker-handoff**
  - 确定对应的worker索引
    - 通过hash src addr
  - 分发报文到对应的核
    - 如果工作者索引与当前工作者索引相同，则转到nat44-in2out节点
      否则做交换操作
- **节点Node nat44-in2out**
  - 确定对应的session
    - Key：src addr、src port、L4 protocol、Rxfib index
    - Hash table：in2out
    - 如果session不存在则下一个节点是nat44-in2out-slowpath
  - 执行原地址、端口翻译

        - 下一个节点：ip4-lookup
        - 更新session计数
        - 更新user的lru链
- **节点Node nat44-in2out-slow-path**
    - 创建新会话确定相应的用户
        - User Key：src addr、L4 protocol、Rxfib index
        - Hash ： user_hash
        - 如果不存在则创建一个新用户
    - 检查相应用户的会话数
        - 如果超过配额，则回收最近使用的（每个用户会话列表）
    - 创建session
        - 使用静态映射（1：1 NAT）或从NAT池中选择地址和端口
        - 每个会话的线程池
        - 创建每个用户的翻译列表元素（dlist）
        - 创建会话查找哈希条目（in2out，out2in）
    - 执行源地址和端口转换
        - 下一个节点= ip4-lookup
        - 更新会话计数器
        - 每个用户会话列表更新LRU

2.out2in（接收翻译的数据包）



- **Node nat44-out2in-worker-hadoff**
  - 确定相应的worker核索引
    - Key：目的地址，目的端口
  - 将数据包分发给对应的worker
    - 如果worker索引与当前worker索引相同，请转到节点中的nat44-out2in
    - 否则做worker交接
- **Node nat44-out2in**
  - 确定相应的会话
    - Key：源地址，源端口，L4协议，Rx Fib索引
    - 哈希表：（bihash_8_8）out2in
    - 如果会话不存在，请尝试匹配静态映射（1：1 NAT），否则下一个节点= error-drop
  - 执行源地址和端口转换
    - 下一个节点= ip4-lookup

        - 更新会话计数器
        - 每个用户会话列表更新LRU
          ###3.数据结构

- **snat_main_t**
    - /* Static mappings (1:1 NAT) lookup hash tables */
        - clib_bihash_8_8_t static_mapping_by_local;
        - clib_bihash_8_8_t static_mapping_by_external;
    - /* Per thread data */
        - snat_main_per_thread_data_t * per_thread_data;

- **snat_main_per_thread_data_t**
    - /* Session lookup hash tables */
        - clib_bihash_8_8_t out2in;
        - clib_bihash_8_8_t in2out;
    - /* User lookup hash table */
        - clib_bihash_8_8_t user_hash;
    - /* User pool */
        - snat_user_t * users;
    - /* Session pool */
        - snat_session_t * sessions;
    - /* Pool of double-linked list elements (per user session list) */
        - dlist_elt_t * list_pool;

- **snat_user_t**
    - ip4_address_t addr;
    - u32 fib_index;
    - u32 sessions_per_user_list_head_index;
    - u32 nsessions;
    - u32 nstaticsessions;

- **snat_session_t**
    - snat_session_key_t out2in;
    - snat_session_key_t in2out;
    - u32 flags;

    - u32 per_user_index;
    - u32 per_user_list_head_index;
    - f64 last_heard;
    - u64 total_bytes;
    - u32 total_pkts;
    - u32 outside_address_index;
    - ip4_address_t ext_host_addr;
    - u16 ext_host_port;

- **snat_session_key_t**

    - ip4_address_t addr;
    - u16 port;
    - u16 protocol:3, fib_index:13;

- **snat_user_key_t**

    - ip4_address_t addr;
    - u32 fib_index;

六、NAT IPFIX（流信息测量的标准协议） logging

- **Supported NAT events（支持的NAT事件）：**

Event Name	Value
NAT Addresses exhausted（NAT 地址耗尽）	3
NAT44 Session create(session 创建)	4
NAT44 Session delete(session 删除)	5
Quota Exceeded Events(超出配额的事件)	13

- **Supported NAT quota exceeded events（NAT配额超过事件）：**

Event Name	Value
Maximum Session Entries Exceeded(超过最大会话条目数)	1

- **Address Exhausted event template（地址耗尽事件模板）：**

Field Name	Size (bits)
observationTimeMilliseconds（观察时间毫秒)	64
natEvent（nat事件)	8
natPoolId（nat pool ID)	32

- **NAT44 Session delete/create template（NAT44会话删除/创建模板）：**

Field Name	Size (bits)
observationTimeMilliseconds(观察时间毫秒)	64
natEvent(nat事件)	8
sourceIPv4Address(原地址)	32
postNATSourceIPv4Address(发布NAT源IPv4地址)	32
protocolIdentifier(协议标识符)	8
sourceTransportPort(源端口)	16
postNAPTSourceTransportPort(发布NAT源端口)	16
ingressVRFID(入口VRFID)	32

- **Maximum Session Entries Exceeded template（超过最大会话条目数模板）：**

Field Name	Size (bits)
observationTimeMilliseconds(观察时间毫秒)	64
natEvent(nat事件)	8
natQuotaExceededEvent(超过配额的事件)	32
maxSessionEntries (会话数上限)	32

七、CGN - deterministic（确定性） NAT

- 内部用户静态映射到一组外部端口，目的是**启用确定性NAT，以减少日志记录并在CGN部署中实现大规模/高性能**。支持与端点有关的映射，以处理外部端口的过载。**为每个内部用户预先分配1000个会话插槽**。使用顺序端口范围分配算法（第一个块到达地址1，第二个块到达地址2，依此类推）
- 支持的协议：
  - TCP
  - UDP
  - ICMP
- NAT会话刷新：
  - NAT出站刷新行为
  - 默认UDP空闲超时5分钟
  - 默认的TCP建立的连接空闲超时2小时4分钟
  - 默认的TCP临时连接空闲超时4分钟
  - 默认的ICMP空闲超时60秒
  - TCP会话关闭检测
  - 可配置的空闲超时
- 支持的IPFIX NAT事件：
  - 超出配额-超出每个用户的最大条目数（NAT事件类型= 13，NAT超出配额事件类型= 3

- IPFIX templates

  - 每个用户的最大条目数超出：

Field Name	Size (bits)
observationTimeMilliseconds(观察时间毫秒)	64
natEvent(nat事件)	8
natQuotaExceededEvent(超过配额的事件)	32
sourceIPv4Address (源地址)	32

- Startup config

  - 要启用NAT插件确定性模式，请在statrup config中添加以下内容：

```
nat { deterministic }
```

  - 要验证NAT插件模式，请使用：
    vpp# show nat44
    NAT plugin mode: deterministic mapping

- CLI
  - 支持的CLI命令：

```
set interface nat44 in <intfc> out <intfc> [del]
nat44 deterministic add in <addr>/<plen> out <addr>/<plen> [del]
show nat44 deterministic mappings
nat44 deterministic forward <addr>
nat44 deterministic reverse <addr>:<port>
set nat44 deterministic timeout [udp <sec> | tcp-established <sec> | t
cp-transitory <sec> | icmp <sec> | reset]
show nat44 deterministic timeouts
nat44 deterministic close session out|in <addr>:<port> <ext_endp_addr>
:<ext_endp_port>
show nat44 deterministic sessions
```

- Example configuration(事例暂未总结)
- API
  - 添加/删除确定性NAT映射：

```
define nat_det_add_del_map {
 u32 client_index;
 u32 context;
 u8 is_add;
 u8 is_nat44;
 u8 addr_only;
 u8 in_addr[16];
 u8 in_plen;
 u8 out_addr[4];
 u8 out_plen;
};
```

- 从内部地址获取外部地址和端口范围：

```
define nat_det_forward {
 u32 client_index;
 u32 context;
 u8 is_nat44;
 u8 in_addr[16];
};
define nat_det_forward_reply {
 u32 context;
 i32 retval;
 u16 out_port_lo;
 u16 out_port_hi;
 u8 out_addr[4];
};
```

- 从外部地址和端口获取内部地址：：

```
define nat_det_reverse {
 u32 client_index;
 u32 context;
 u16 out_port;
 u8 out_addr[4];
};
define nat_det_reverse_reply {
 u32 context;
 i32 retval;
 u8 is_nat44;
 u8 in_addr[16];
};
```

- 转储确定性(deterministic)NAT映射：

```
define nat_det_map_dump {
 u32 client_index;
 u32 context;
};
define nat_det_map_details {
 u32 context;
 u8 is_nat44;
 u8 in_addr[16];
```

```
 u8 in_plen;
 u8 out_addr[4];
 u8 out_plen;
 u32 sharing_ratio;
 u16 ports_per_host;
 u32 ses_num;
};
```

- 设置确定性NAT的超时值（以秒为单位，0 =默认值）：

```
define nat_det_set_timeouts {
 u32 client_index;
 u32 context;
 u32 udp;
 u32 tcp_established;
 u32 tcp_transitory;
 u32 icmp;
};
```

- 获取确定性NAT的超时值（以秒为单位）：

```
define nat_det_get_timeouts {
 u32 client_index;
 u32 context;
};
define nat_det_get_timeouts_reply {
 u32 context;
 i32 retval;
 u32 udp;
 u32 tcp_established;
 u32 tcp_transitory;
 u32 icmp;
};
```

- 使用外部地址和端口关闭确定性NAT：

```
define nat_det_close_session_out {
 u32 client_index;
 u32 context;
 u8 out_addr[4];
```

```
 u16 out_port;
 u8 ext_addr[4];
 u16 ext_port;
};
```

- 使用内部地址和端口关闭确定性NAT：

```
define nat_det_close_session_in {
 u32 client_index;
 u32 context;
 u8 is_nat44;
 u8 in_addr[16];
 u16 in_port;
 u8 ext_addr[16];
 u16 ext_port;
};
```

- 转储确定性NAT会话：

```
define nat_det_session_dump {
 u32 client_index;
 u32 context;
 u8 is_nat44;
 u8 user_addr[16];
};
define nat_det_session_details {
 u32 client_index;
 u32 context;
 u16 in_port;
 u8 ext_addr[4];
 u16 ext_port;
 u16 out_port;
 u8 state;
 u32 expire;
};
```

- 内存需求
  - 具有确定性的NAT预分配向量，该向量具有来自网络范围内的每个主机的1000个会话插槽（一个会话15B）。堆大小是通过heapsize参数配置的。对于大型内部网络，您需要

使用64位向量（-DCLIB_VEC64）编译VPP。

八、DS-Lite

- Dual-Stack Lite通过结合两项众所周知的技术：IPv4-in-IPv6和NAT，使宽带服务提供商可以在客户之间共享IPv4地址。
- **CLI**
  - 支持的CLI命令：

```
dslite add pool address <ip4-range-start> [- <ip4-range-end>] [del]
show dslite pool
dslite set aftr-tunnel-endpoint-address <ip6>
show dslite aftr-tunnel-endpoint-address
dslite set b4-tunnel-endpoint-address <ip6>
show dslite b4-tunnel-endpoint-address
show dslite sessions
```

  - AFTR元素示例配置(暂时不添加到文档)
  - Example configuration B4 element (CE)(暂时不添加到文档)
  - API
    - 向DS-Lite池添加/删除地址范围：

```
define dslite_add_del_pool_addr_range {
 u32 client_index;
 u32 context;
 u8 start_addr[4];
 u8 end_addr[4];
 u8 is_add;
};
```

    - Set AFTR address:

```
define dslite_set_aftr_addr {
```

```
 u32 client_index;
 u32 context;
 u8 ip4_addr[4];
 u8 ip6_addr[16];
};
```

- Get AFTR address:

```
define dslite_get_aftr_addr {
 u32 client_index;
 u32 context;
};
define dslite_get_aftr_addr_reply {
 u32 context;
 i32 retval;
 u8 ip4_addr[4];
 u8 ip6_addr[16];
};
```

- Set B4 address:

```
define dslite_set_b4_addr {
 u32 client_index;
 u32 context;
 u8 ip4_addr[4];
 u8 ip6_addr[16];
};
```

- Get B4 address:

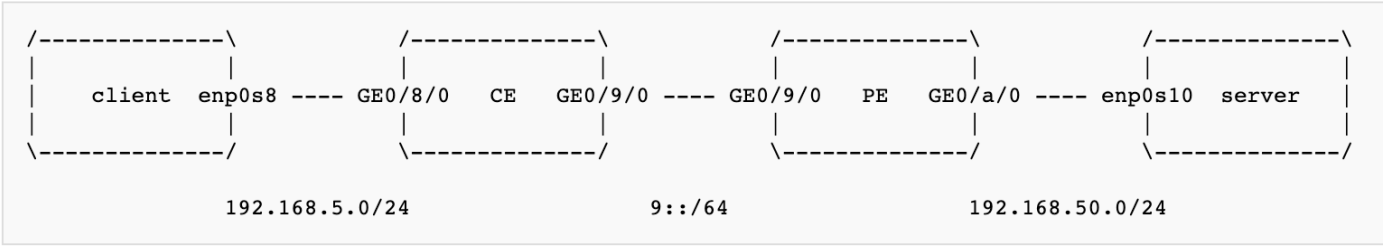
```
define dslite_get_b4_addr {
 u32 client_index;
 u32 context;
};
define dslite_get_b4_addr_reply {
 u32 context;
 i32 retval;
 u8 ip4_addr[4];
 u8 ip6_addr[16];
};
```

九、Stateful NAT64

- VPP有状态NAT64实现支持TCP，UDP和ICMP转换。
- **暂时不将NAT 4-6相关的过渡技术填写到此文档**

十、464XLAT

- **暂时不将NAT 4-6相关的过渡技术填写到此文档**



十一、NAT插件虚拟碎片重组

- 非初始片段没有第4层标头，因为它通常与初始片段一起传播。因此，NAT无法从数据包中收集端口信息。 VPP NAT插件支持接收有序和无序片段。
- **默认最大阈值：**
  - 每次重新组装的最大碎片数量：5
  - 并发重组的最大数量：1024
  - 重组超时：2秒
- CLI
  - 支持的CLI命令：

```
nat virtual-reassembly ip4|ip6 [max-reassemblies <n>] [max-fragments <
n>] [timeout <sec>] [enable|disable]
show nat virtual-reassembly
```

- API

- 设置NAT虚拟碎片重组：

```
define nat_set_reass {
 u32 client_index;
 u32 context;
 u32 timeout;
 u16 max_reass;
 u8  max_frag;
 u8  drop_frag;
 u8  is_ip6;
};
```

- 获取NAT虚拟碎片重组配置：

```
define nat_get_reass {
 u32 client_index;
 u32 context;
};
define nat_get_reass_reply {
 u32 context;
 i32 retval;
 u32 ip4_timeout;
 u16 ip4_max_reass;
 u8  ip4_max_frag;
 u8  ip4_drop_frag;
 u32 ip6_timeout;
 u16 ip6_max_reass;
 u8  ip6_max_frag;
 u8  ip6_drop_frag;
};
```

- 转储NAT虚拟碎片重组：

```
define nat_reass_dump {
 u32 client_index;
 u32 context;
};
define nat_reass_details {
 u32 context;
 u8 is_ip4;
 u8 src_addr[16];
```

```
 u8 dst_addr[16];
 u32 frag_id;
 u8 proto;
 u8 frag_n;
};
```