G5 LOAD BALANCING

APPLICATION NOTE

17 AUG 2020

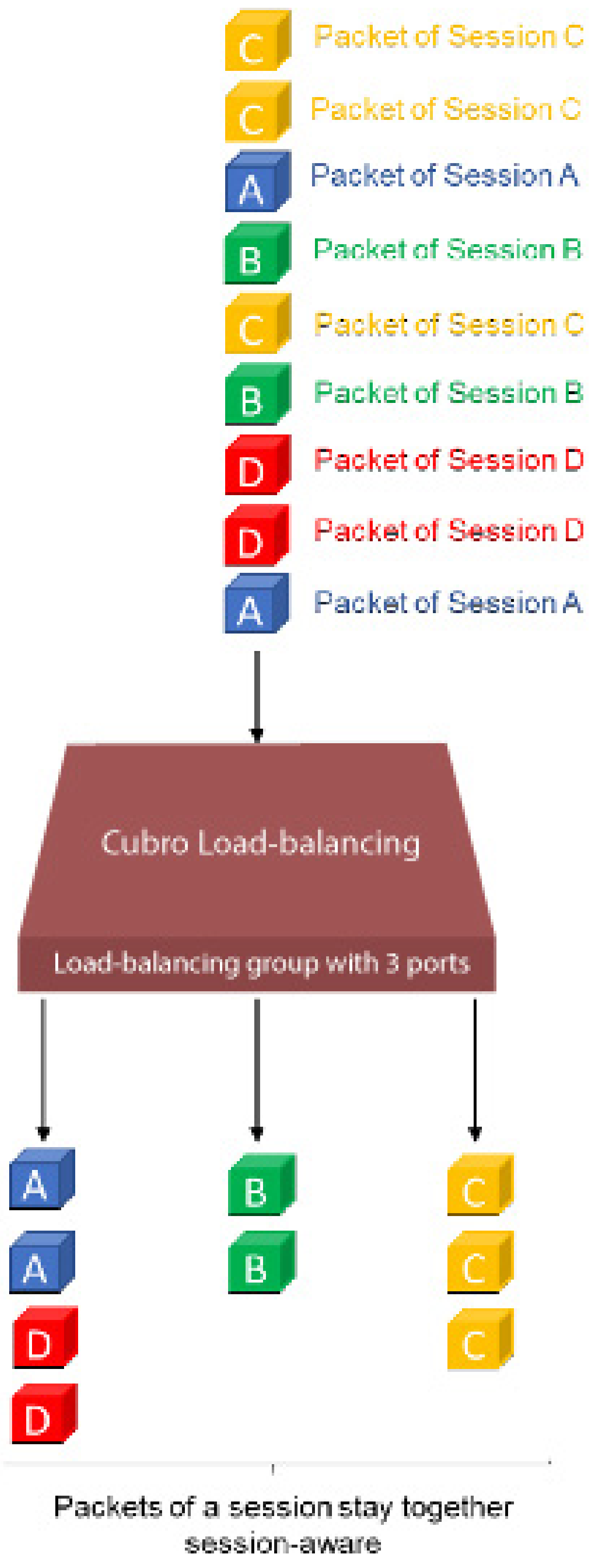
Author: Herbert Etlinger

Introduction

Load-balancing is a vital function to distribute traffic across different monitoring tools evenly and correctly. The Cubro EXA48600 & EXA32100 support Session-Aware Load balancing, meaning that every packet that belongs to the same conversation/flow is sent to the same physical output port within a load-balancing group. This ensures that connected packet sniffer or other monitoring tools will get every packet of a given conversation.

The EXA48600 & EXA32100 maintain the association of packets with each flow or conversation between any two network endpoints such that all traffic from a given flow will be output from a consistent monitor port within a load balanced group. Flow association is done by examining selected fields within each packet and performing a mathematical algorithm called hash key calculation. The result of the calculation is used to consistently separate and distribute traffic to specific ports within a load balanced group. Depending on the requirements, the EXA48600 & EXA32100 allow different hash-key calculations methods and thus allow to make sure that packets always arrive at the correct interface of the monitoring appliance.

For complex overlay networks where traffic is encapsulated in tunnels the EXA48600 & EXA32100 support hashing based on inner tunnel information. This ensures that user-flows encapsulated in tunneling protocols, such as VXLAN or GRE, are output appropriately.



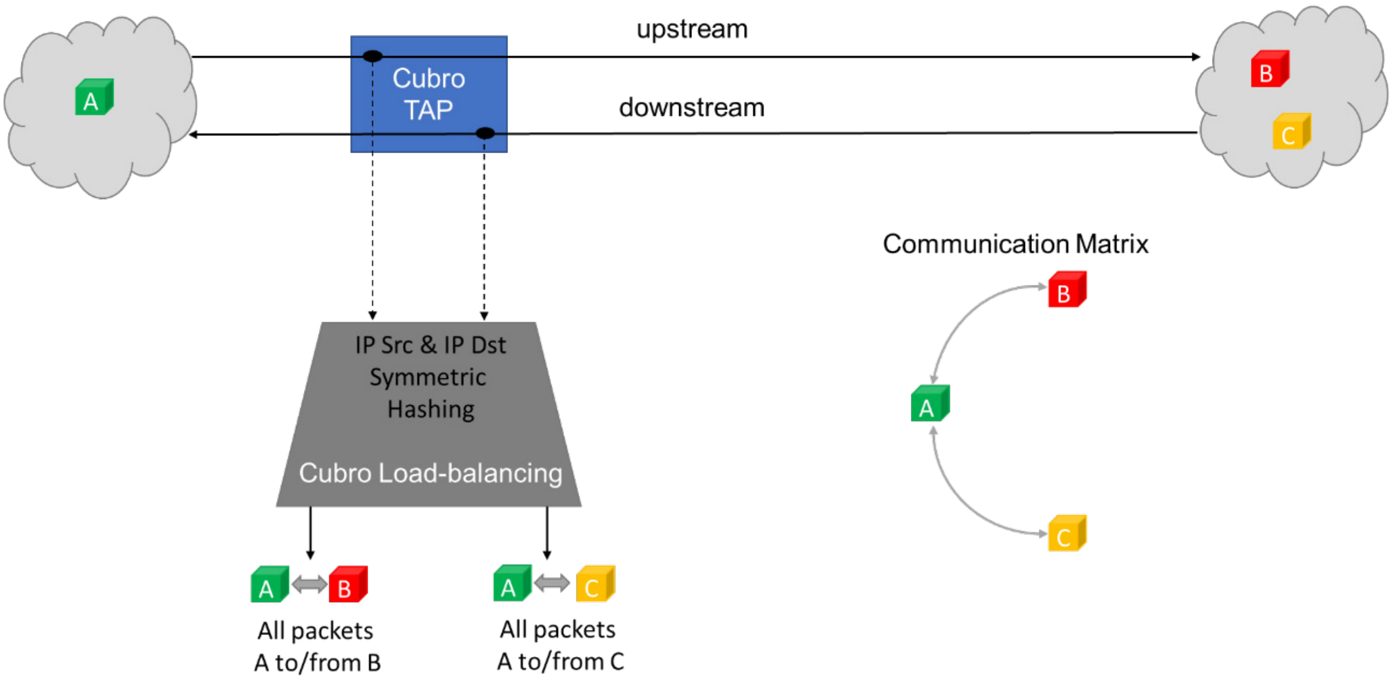
Hash-key calculation

To cope with a wide range of requirements, the EXA48600 & EXA32100 allow various methods to calculate the hash-key. Hash-keys are used to define the load-balancing behaviour among the various members in the load-balancing group. For example, if the hash-key is configured as "IP Source Address", the hashing would be performed based on the source IP address of the packet only. Therefore, all packets with the same source IP address will be available at the same physical output port. The EXA48600 and EXA32100 support following hash-key calculation methods:

Hash-key calculation method	Hash-key calculation based on	Remark
l3-src	full IP Src Addr	
l3-dst	full IP Dst Addr	
l3-src-dst	full IP Src & IP Dst Addr	Upstream & downstream direction give DIFFERENT hash results -> upstream & downstream is split apart -> not session aware E.g. 10.0.0.1 talks to 10.0.0.2: Hash result = x 10.0.0.2 talks back to 10.0.0.1: Hash result = y
l4-src-dst	full Layer 4 Src & Dst Port	
four-tuple	full IP Src & Dst Addr & Layer 4 Src & Dst Port	
four-tuple-m8	middle 8 Byte of IP Src & Dst Addr & full Layer 4 Src & Dst Port	
five-tuple	full IP Src & Dst Addr & Layer Protocol & Layer 4 Src & Dst Port	
l3-src-dst-m8	middle 8 Byte IP Src & IP Dst Addr	
five-tuple-m8	middle 8 Byte IP Src & Dst Addr & Layer Protocol & Layer 4 Src & Dst Port	
l3-src-dst-symmetric	full IP Src & IP Dst Addr	Upstream & downstream direction give SAME hash results -> upstream & downstream stay together-> session aware E.g. 10.0.0.1 talks to 10.0.0.2: Hash result = x 10.0.0.2 talks back to 10.0.0.1: Hash result = x
l4-src-dst-symmetric	full Layer 4 Src & Dst Port	
four-tuple-symmetric	full IP Src & Dst Addr & Layer 4 Src & Dst Port	
four-tuple-m8-symmetric	middle 8 Byte of IP Src & Dst Addr & full Layer 4 Src & Dst Port	
five-tuple-symmetric	full IP Src & Dst Addr & Layer Protocol & Layer 4 Src & Dst Port	
l3-src-dst-m8-symmetric	middle 8 Byte IP Src & IP Dst Addr	
five-tuple-m8-symmetric	middle 8 Byte IP Src & Dst Addr & Layer Protocol & Layer 4 Src & Dst Port	

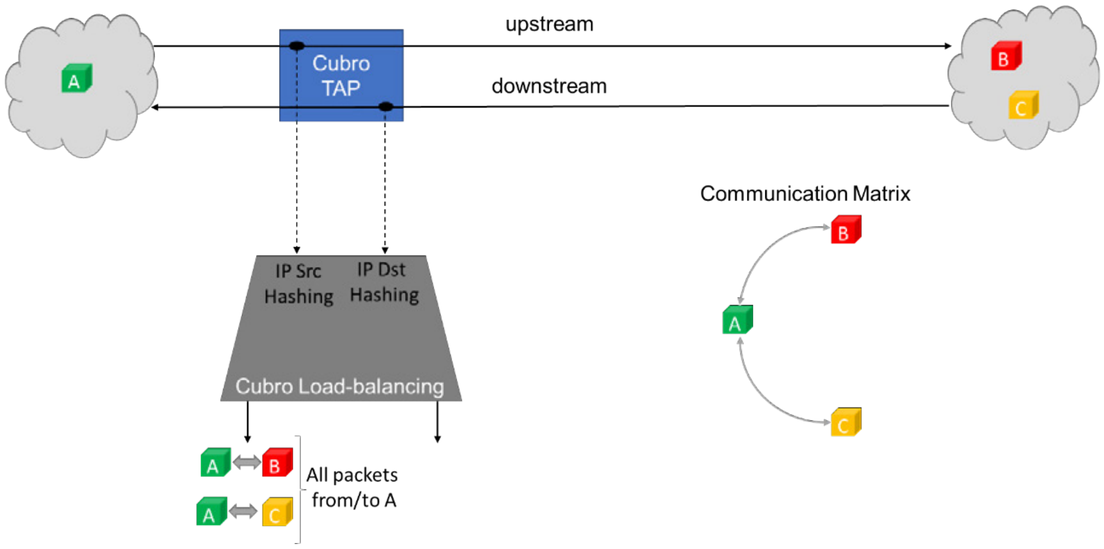
Moreover, the EXA48600 & EXA32100 support selecting the hash-key calculation method on a per-port basis. This is of particular interest when upstream and downstream communication of all sessions of a dedicated user (=dedicated IP Address) should be output together at one physical output port.

Example using IP Src & IP Dst Symmetric Hash



In this case, the load-balancing is session-aware and all packets from communication A to/from B are sent out at a specific physical port while A to/from C is on a different port. If it is required that all communications to/from A regardless where to whom it is talking to should be sent out - see the following example.

Example using IP Src for uplink and IP Dst for downlink

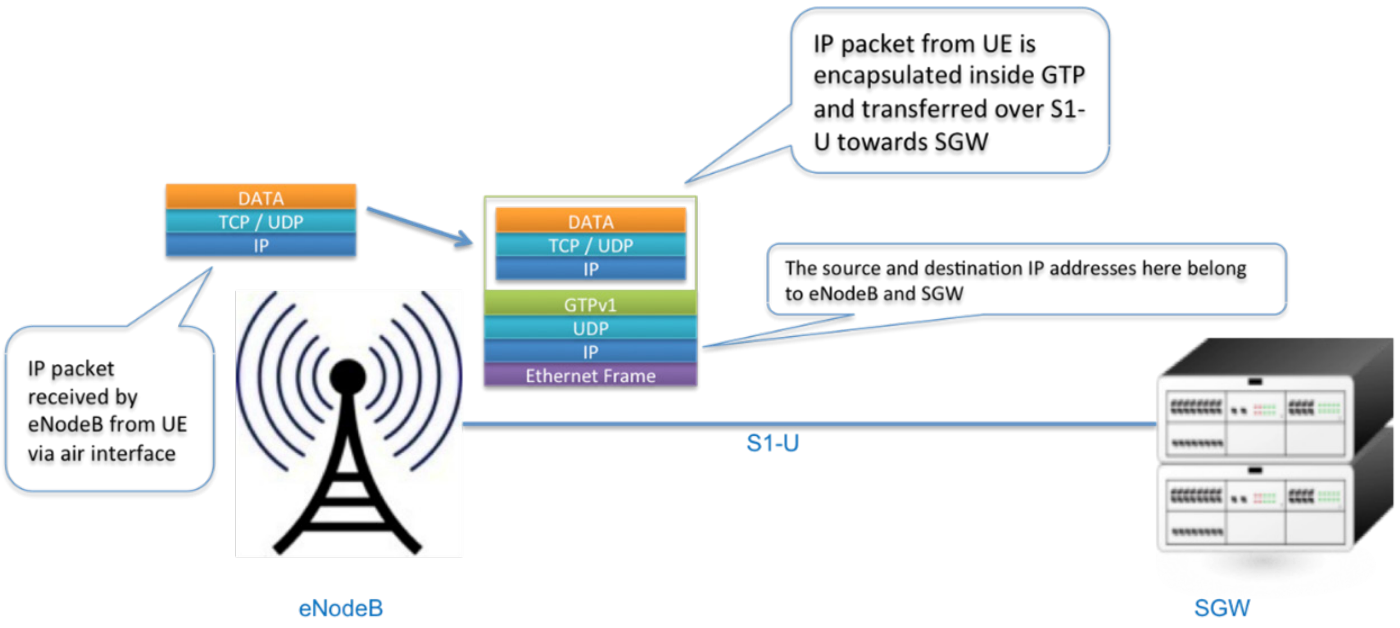


As the hashing method is different at each input port; all packets to and from A are sent out at the same physical port.

Load-balancing using inner tunnel information

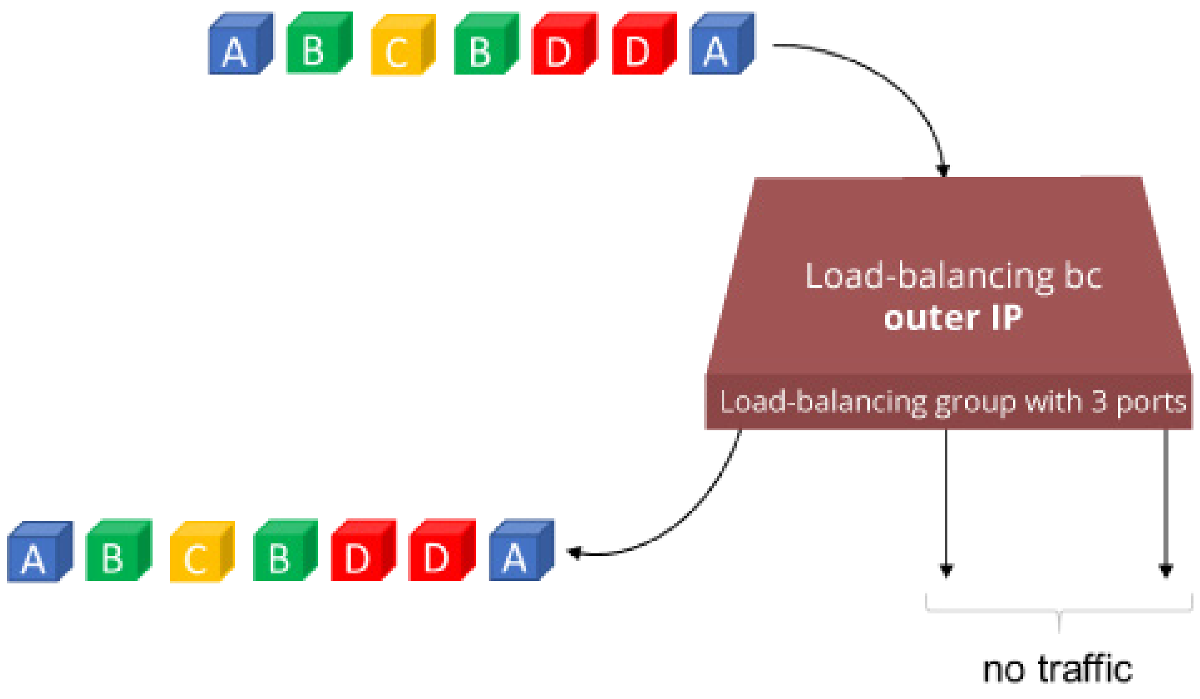
In modern overlay communication networks, packets are usually encapsulated in tunnels. Typical encapsulation protocols in use are VXLAN, GRE or MPLS over UDP. In mobile networks we also see GTP encapsulated traffic. The ability of the EXA32100 & EXA48600 to use the inner tunnel information like the inner IP Address for the hash-key calculation makes these products the ideal choice for load-balancing encapsulated packets. Consider this example of how to efficiently load-balance mobile GTP traffic.

GTP is used in mobile networks to transport packets from the NodeB to the internet via an IP tunnel.

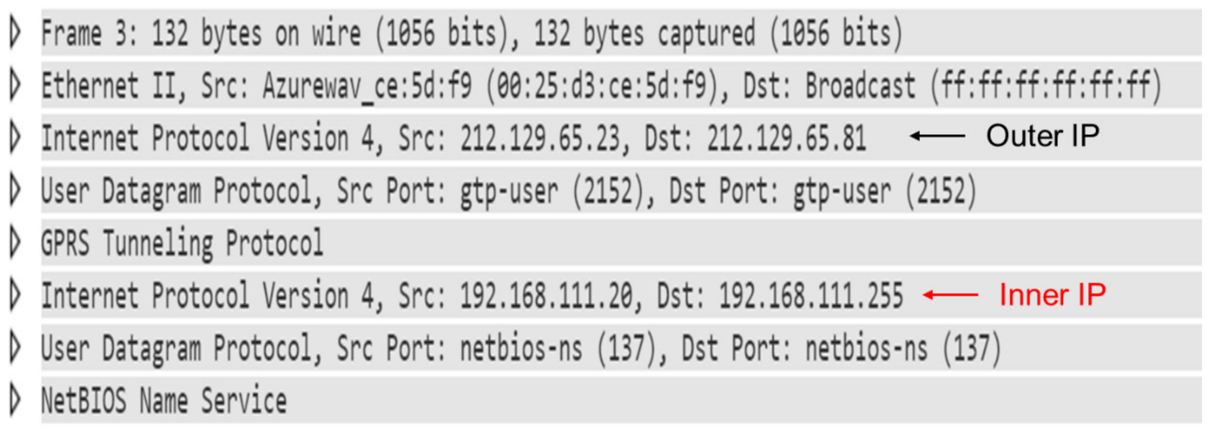

IP packet from UE is encapsulated inside GTP and transferred over S1-U towards SGW
DATA
TCP / UDP
IP
DATA
TCP / UDP
IP
GTPv1
UDP
IP
Ethernet Frame
The source and destination IP addresses here belong to eNodeB and SGW
IP packet received by eNodeB from UE via air interface
S1-U
eNodeB
SGW

Load-balancing could be based on outer IP Addresses which are the IP Addresses of the eNodeBs and SGWs. The problems with using the outer tunnel for the hash-key calculation for load-balancing are:

Load-balancing works best when more IP combinations are available. The more variations are present, the better the load-balancing mechanism will work. In a case where there are only a small number of the IP Addresses, when the outer IP is used, the LB could be very asymmetric resulting in uneven distribution of the traffic.

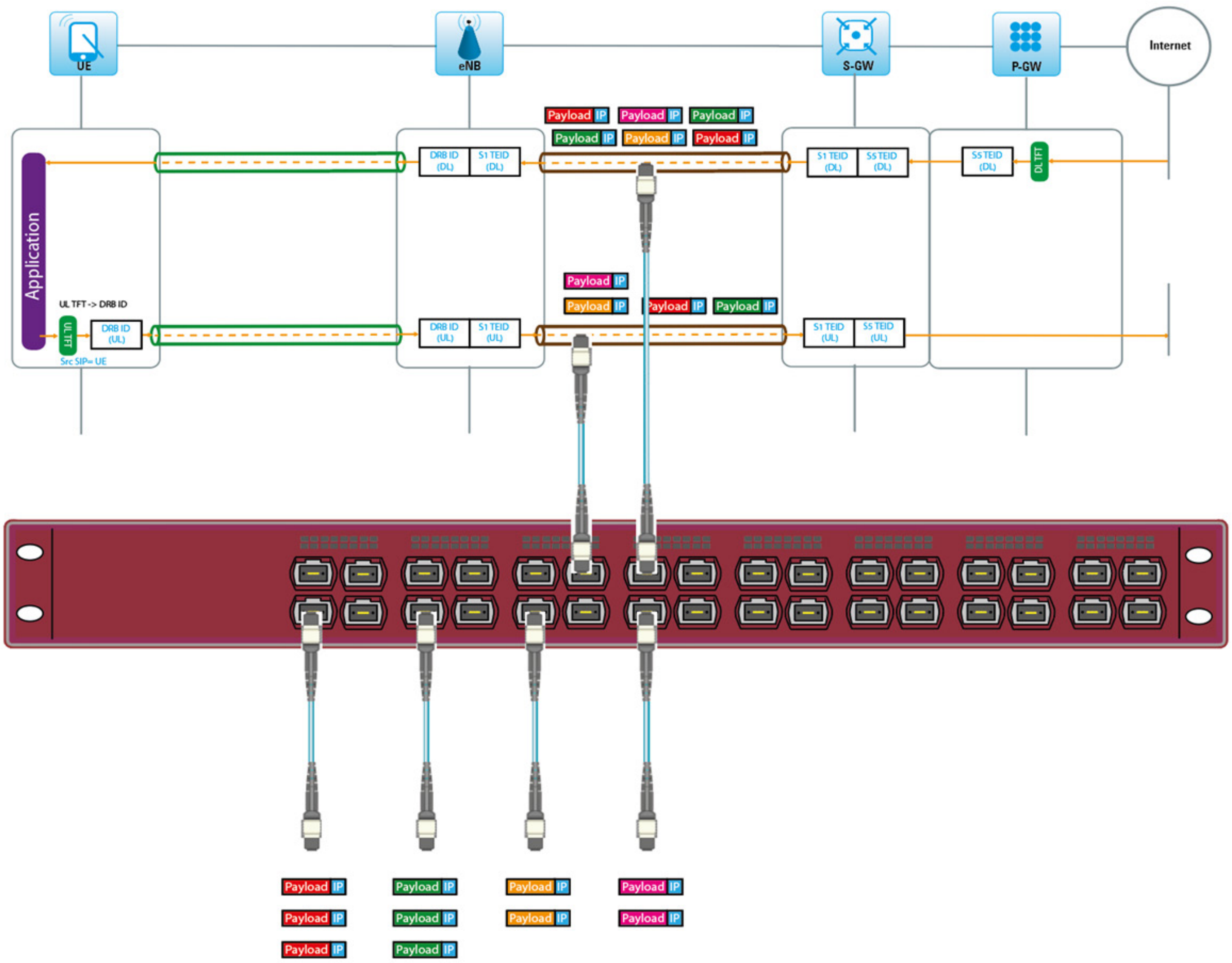


The hash-key calculation result will change when, for instance, the user changes location and thus changes which tower they connect to. When the IP Address of the outer tunnel changes (e.g. eNodeB change) the session will move to another output port and thus will be available at a different port of the monitoring appliance. Therefore, load-balancing will not be session-aware from a user perspective, resulting in more processing power being required to do correlation/call analysis. The EXA48600 & EXA32100 allow the user to perform load-balancing on the inner IP of the tunneled traffic. This will solve both aforementioned problems.



```
▷ Frame 3: 132 bytes on wire (1056 bits), 132 bytes captured (1056 bits)
▷ Ethernet II, Src: Azurewav_ce:5d:f9 (00:25:d3:ce:5d:f9), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
▷ Internet Protocol Version 4, Src: 212.129.65.23, Dst: 212.129.65.81   ←— Outer IP
▷ User Datagram Protocol, Src Port: gtp-user (2152), Dst Port: gtp-user (2152)
▷ GPRS Tunneling Protocol
▷ Internet Protocol Version 4, Src: 192.168.111.20, Dst: 192.168.111.255   ←— Inner IP
▷ User Datagram Protocol, Src Port: netbios-ns (137), Dst Port: netbios-ns (137)
▷ NetBIOS Name Service
```

The inner IP is also referred to as GTP user IP as it is the IP address of the endpoint (e.g. mobile phone). The range of inner IP addresses are much higher than outer IPs and thus the load-balancing mechanism will work much better. Depending on the size of the mobile operator, the range of inner IPs will reach millions easily. Secondly, the user IP will not change when the user is moving and thus, the user session will always be available at the same physical output port.



Summary

Besides functions like traffic aggregation, de-encapsulation, and powerful filtering the Cubro EXA48600 & EXA32100 support highly sophisticated session-aware load-balancing. With its ability to keep the packets of a session together even when traffic is encapsulated in tunnels like VXLAN, GTP or GRE, these advanced network packet brokers are a perfect choice for modern overlays and mobile networks.

For further queries, contact support@cubro.com.