The Linux kernel GTP tunneling module

Documentation by
Harald Welte <laforge@gnumonks.org> and Andreas Schultz <aschultz@tpip.net>

In 'drivers/net/gtp.c' you are finding a kernel-level implementation of a GTP tunnel endpoint.

What is GTP

GTP is the Generic Tunnel Protocol, which is a 3GPP protocol used for tunneling User-IP payload between a mobile station (phone, modem) and the interconnection between an external packet data network (such as the internet).

So when you start a 'data connection' from your mobile phone, the phone will use the control plane to signal for the establishment of such a tunnel between that external data network and the phone. The tunnel endpoints thus reside on the phone and in the gateway. All intermediate nodes just transport the encapsulated packet.

The phone itself does not implement GTP but uses some other technology-dependent protocol stack for transmitting the user IP payload, such as LLC/SNDCP/RLC/MAC.

At some network element inside the cellular operator infrastructure (SGSN in case of GPRS/EGPRS or classic UMTS, hNodeB in case of a 3G femtocell, eNodeB in case of 4G/LTE), the cellular protocol stacking is translated into GTP *without breaking the end-to-end tunnel*. So intermediate nodes just perform some specific relay function.

At some point the GTP packet ends up on the so-called GGSN (GSM/UMTS) or P-GW (LTE), which terminates the tunnel, decapsulates the packet and forwards it onto an external packet data network. This can be public internet, but can also be any private IP network (or even theoretically some non-IP network like X.25).

You can find the protocol specification in 3GPP TS 29.060, available publicly via the 3GPP website at http://www.3gpp.org/DynaReport/29060.htm

A direct PDF link to v13.6.0 is provided for convenience below:
http://www.etsi.org/deliver/etsi_ts/129000_129099/129060/13.06.00_60/ts_129060v130600p.pdf

The Linux GTP tunnelling module

The module implements the function of a tunnel endpoint, i.e. it is able to decapsulate tunneled IP packets in the uplink originated by the phone, and encapsulate raw IP packets received from the external packet network in downlink towards the phone.

It *only* implements the so-called 'user plane', carrying the User-IP payload, called GTP-U. It does not implement the 'control plane', which is a signaling protocol used for establishment and teardown of GTP tunnels (GTP-C).

So in order to have a working GGSN/P-GW setup, you will need a userspace program that implements the GTP-C protocol and which then uses the netlink interface provided by the GTP-U module in the kernel to configure the kernel module.

This split architecture follows the tunneling modules of other protocols, e.g. PPPoE or L2TP, where you also run a userspace daemon to handle the tunnel establishment, authentication etc. and only the data plane is accelerated inside the kernel.

Don't be confused by terminology: The GTP User Plane goes through kernel accelerated path, while the GTP Control Plane goes to Userspace :)

The official homepage of the module is at https://osmocom.org/projects/linux-kernel-gtp-u/wiki

Userspace Programs with Linux Kernel GTP-U support

At the time of this writing, there are at least two Free Software implementations that implement GTP-C and can use the netlink interface to make use of the Linux kernel GTP-U support:

- OpenGGSN (classic 2G/3G GGSN in C): https://osmocom.org/projects/openggsn/wiki/OpenGGSN
- ergw (GGSN + P-GW in Erlang): https://github.com/travelping/ergw

Userspace Library / Command Line Utilities

There is a userspace library called 'libgtpnl' which is based on libmnl and which implements a C-language API towards the netlink interface provided by the Kernel GTP module:

http://git.osmocom.org/libgtpnl/

Protocol Versions

There are two different versions of GTP-U: v0 [GSM TS 09.60] and v1 [3GPP TS 29.281]. Both are implemented in the Kernel GTP module. Version 0 is a legacy version, and deprecated from recent 3GPP specifications.

GTP-U uses UDP for transporting PDUs. The receiving UDP port is 2151 for GTPv1-U and 3386 for GTPv0-U.

There are three versions of GTP-C: v0, v1, and v2. As the kernel doesn't implement GTP-C, we don't have to worry about this. It's the responsibility of the control plane implementation in userspace to implement that.

IPv6

The 3GPP specifications indicate either IPv4 or IPv6 can be used both on the inner (user) IP layer, or on the outer (transport) layer.

Unfortunately, the Kernel module currently supports IPv6 neither for the User IP payload, nor for the outer IP layer. Patches or other Contributions to fix this are most welcome!

Mailing List

If you have questions regarding how to use the Kernel GTP module from your own software, or want to contribute to the code, please use the osmocom-net-grps mailing list for related discussion. The list can be reached at osmocom-net-gprs@lists.osmocom.org and the mailman interface for managing your subscription is at https://lists.osmocom.org/mailman/listinfo/osmocom-net-gprs

Issue Tracker

The Osmocom project maintains an issue tracker for the Kernel GTP-U module at https://osmocom.org/projects/linux-kernel-gtp-u/issues

History / Acknowledgements

The Module was originally created in 2012 by Harald Welte, but never completed. Pablo came in to finish the mess Harald left behind. But doe to a lack of user interest, it never got merged.

In 2015, Andreas Schultz came to the rescue and fixed lots more bugs, extended it with new features and finally pushed all of us to get it mainline, where it was merged in 4.7.0.

Architectural Details

Local GTP-U entity and tunnel identification

GTP-U uses UDP for transporting PDU's. The receiving UDP port is 2152 for GTPv1-U and 3386 for GTPv0-U.

There is only one GTP-U entity (and therefor SGSN/GGSN/S-GW/PDN-GW instance) per IP address. Tunnel Endpoint Identifier (TEID) are unique per GTP-U entity.

A specific tunnel is only defined by the destination entity. Since the destination port is constant, only the destination IP and TEID define a tunnel. The source IP and Port have no meaning for the tunnel.

Therefore:

- when sending, the remote entity is defined by the remote IP and the tunnel endpoint id. The source IP and port have no meaning and can be changed at any time.
- when receiving the local entity is defined by the local destination IP and the tunnel endpoint id. The source IP and port have no meaning and can change at any time.

[3GPP TS 29.281] Section 4.3.0 defines this so:

```
The TEID in the GTP-U header is used to de-multiplex traffic
incoming from remote tunnel endpoints so that it is delivered to the
User plane entities in a way that allows multiplexing of different
users, different packet protocols and different QoS levels.
Therefore no two remote GTP-U endpoints shall send traffic to a
GTP-U protocol entity using the same TEID value except
for data forwarding as part of mobility procedures.
```

The definition above only defines that two remote GTP-U endpoints *should not* send to the same TEID, it *does not* forbid or exclude such a scenario. In fact, the mentioned mobility procedures make it necessary that the GTP-U entity accepts traffic for TEIDs from multiple or unknown peers.

Therefore, the receiving side identifies tunnels exclusively based on TEIDs, not based on the source IP!

APN vs. Network Device

The GTP-U driver creates a Linux network device for each Gi/SGi interface.

[3GPP TS 29.281] calls the Gi/SGi reference point an interface. This may lead to the impression that the GGSN/P-GW can have only one such interface.

Correct is that the Gi/SGi reference point defines the interworking between +the 3GPP packet domain (PDN) based on GTP-U tunnel and IP based networks.

There is no provision in any of the 3GPP documents that limits the number of Gi/SGi interfaces implemented by a GGSN/P-GW.

[3GPP TS 29.061] Section 11.3 makes it clear that the selection of a specific Gi/SGi interfaces is made through the Access Point Name (APN):

```
   2. each private network manages its own addressing. In general this
      will result in different private networks having overlapping
      address ranges. A logically separate connection (e.g. an IP in IP
      tunnel or layer 2 virtual circuit) is used between the GGSN/P-GW
      and each private network.

      In this case the IP address alone is not necessarily unique.  The
      pair of values, Access Point Name (APN) and IPv4 address and/or
      IPv6 prefixes, is unique.
```

In order to support the overlapping address range use case, each APN is mapped to a separate Gi/SGi interface (network device).

Note

The Access Point Name is purely a control plane (GTP-C) concept. At the GTP-U level, only Tunnel Endpoint Identifiers are present in GTP-U packets and network devices are known

Therefore for a given UE the mapping in IP to PDN network is:

- network device + MS IP -> Peer IP + Peer TEID,

and from PDN to IP network:

- local GTP-U IP + TEID -> network device

Furthermore, before a received T-PDU is injected into the network device the MS IP is checked against the IP recorded in PDP context.