PPP over serial link

PPP is a data link layer protocol mainly intended to establish point to point connection over the serial link like the console port we have on our Switchfin targets.
Let's see how we can demonstrate this.

In the latest Switchfin we have PPP option in the menuconfig. Please enable it and build the image.

Connect the serial port of a PC with PPP support with the PR1 Appliance console port.
Establish the ssh sessian with the PR1 Appliance.

On the PR1 Appliance set the PPP configuration file
#/etc/ppp/options
lock
nocrtscts
10.0.0.209:10.0.0.2
linkname ppp0
local
nodefaultroute
/dev/ttyBF0
115200
-detach

Those lines say:

- PR1 Appliance will use ***/dev/ttyBF0*** device for the PPP
- PR1 Appliance will get address ***10.0.0.209*** and the other end will be ***10.0.0.2***
- The PR1 Appliance console has no flow control signals wired (RTS,CTS, DTR, DSR, etc.) so we say ***nocrtscts***
- we will give name ***ppp0*** to the link

On the host we may have configuration like this:
#/etc/ppp/options
lock
nocrtscts
10.0.0.2:10.0.0.209
linkname ppp0
local
nodefaultroute
/dev/ttyS0
115200
-detach

Finally let's load the ppp over serial link module
root@pr1:~> modprobe ppp_async

and we will get the link up

root@pr1:~> ifconfig
.....
ppp0 Link encap:Point-to-Point Protocol
inet addr:10.0.0.2 P-t-P:10.0.0.209 Mask:255.255.255.255
UP POINTOPOINT RUNNING NOARP MULTICAST MTU:1500 Metric:1
RX packets:6 errors:0 dropped:0 overruns:0 frame:0
TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:3

and we may ping PR1 Appliance from the PC (or vice versa) like this:

[root@Switchfin ~]# ping 10.0.0.209
PING 10.0.0.209 (10.0.0.209) 56(84) bytes of data.
64 bytes from 10.0.0.209: icmp_seq=1 ttl=64 time=26.7 ms
64 bytes from 10.0.0.209: icmp_seq=2 ttl=64 time=25.9 ms

64 bytes from 10.0.0.209: icmp_seq=3 ttl=64 time=25.9 ms

Similarly you may ssh the PR1 Appliance thru the new ppp0 interface.

Point-to-Point *Tunneling* Protocol and VPN

PPP can work not only over serial link. Let's see how it can be used to create VPN over the Ethernet.
In addition we will test the MPPE (Microsoft Point-to-Point Encryption/Compression) support built in as kernel module.

Let's use one of the Linux PC in the network as PPPoE server.
PPPoE is a simple protocol and basically it is pretty much like the DHCP negotiation.
Description how to do PPPoE server can be found at **PPPoE Server Under Ubuntu/Debian**
The necessary configuration at the PC are

#/etc/ppp/pppoe-server-options
 # PPP options for the PPPoE server
 # LIC: GPL
lcp-echo-interval 10
lcp-echo-failure 2
debug
auth
 # MPPE requires mschap-v2
require-mschap-v2
require-mppe
refuse-pap
refuse-chap
refuse-mschap

#/etc/ppp/chap-secrets
 # Secrets for authentication using CHAP
 # client□□□□□□□□ server□ secret□□□□□□□□□□□□□□□□□□□□ IP addresses
 # allow any client, any server, any IP addr
*** * "" ***

CHAP is an authentication schema.
To prevent exchanging the username/password (as it is in the other popular authentication PAP) this method sends random number (challenge) which is processed by both PPP ends using hash function.
The results of the hash is compared.

Let's load the MPPE module and start the server

root@switchfin:~# modprobe ppp-mppe
root@switchfin:~#/usr/sbin/pppoe-server -T 60 -I eth0 -N 250 -R 192.168.0.1

Now let's go to the PR1 Appliance side. We use the same image with the built PPP option.

The required configuration files are:
 # /etc/ppp/peers/mylink
plugin rp-pppoe.so eth0
mtu 1492
mru 1492
require-mppe
debug

PPPoE we will start using pppd plugin rp-pppoe.so.
We will use eth0 as carrier interface.
We will use Microsoft Point-to-Point Encryption MPPE

 # /etc/ppp/chap-secret
 # Secrets for authentication using CHAP
 # client server secret IP addresses
 # allow any client, any server, any IP addr
*** * "" ***

Some of the PPP stuff is compiled as modules. We don't need ***ppp_async*** now but we need few others:

root@pr1:~>modprobe sha1_generic
root@pr1:~>modprobe ppp_mppe
root@pr1:~>modprobe pppoe

Now we have the following modules loaded in PR1 Appliance:

root@pr1:~> lsmod
pppoe 6973 2 - Live 0x043e4000
pppox 1123 1 pppoe, Live 0x04607800
ecb 1141 2 - Live 0x04606000
ppp_mppe 4733 2 - Live 0x043e2000
ppp_generic 13111 7 pppoe,pppox,ppp_mppe, Live 0x05b9c000
slhc 3685 1 ppp_generic, Live 0x04411000
sha1_generic 1309 4 - Live 0x0460c000
wpr1 16852 62 - Live 0x04730000
dahdi 165436 65 wpr1, Live 0x04780000
crc_ccitt 953 1 dahdi, Live 0x042f8000
mmc_block 6180 2 - Live 0x0408e000
mmc_spi 6493 0 - Live 0x0408c000
crc7 711 1 mmc_spi, Live 0x042f5000
crc_itu_t 980 1 mmc_spi, Live 0x042f4c00
mmc_core 36905 2 mmc_block,mmc_spi, Live 0x043f0000

and we can start the PPP link

root@pr1:~> pppd call mylink &
[1] 478 pppd call mylink
root@pr1:~> Plugin rp-pppoe.so loaded.
RP-PPPoE plugin version 3.3 compiled against pppd 2.4.4

On the PR1 Appliance we see that the ppp is established

root@pr1:~> ifconfig
...
ppp0 Link encap:Point-to-Point Protocol
inet addr:192.168.0.14 P-t-P:10.0.0.1 Mask:255.255.255.255
UP POINTOPOINT RUNNING NOARP MULTICAST MTU:1488 Metric:1
RX packets:10 errors:0 dropped:0 overruns:0 frame:0
TX packets:10 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:3

on the host we have

root@switchfin:~# ifconfig
...
ppp0 Link encap:Point-to-Point Protocol
inet addr:10.0.0.1 P-t-P:192.168.0.14 Mask:255.255.255.255
UP POINTOPOINT RUNNING NOARP MULTICAST MTU:1488 Metric:1
RX packets:10 errors:0 dropped:0 overruns:0 frame:0
TX packets:10 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:3

RX bytes:322 (322.0 B) TX bytes:322 (322.0 B)

and we can ssh the PR1 Appliance over ppp

root@yni:~# ssh 192.168.0.14