라즈베리파이3 B+에 MPTCP 적용

환경

- version
  - Raspberry Pi kernel : 4.14.y
  - MPTCP : v0.94

- 준비
  - SD Card
  - SD Card 리더기
  - Linux OS가 탑재된 Desktop PC
  - Raspberry Pi3 B+

진행하기 앞서

✓ 본 내용의 목적은 **라즈베리파이3 B+ 위에 MPTCP 커널을 올려보는 것**이다.

✓ 본 내용은 독자가 라즈베리파이 커널 크로스 컴파일을 진행해보았다는 전제하에 작성되었습니다.
또한, 기본적인 리눅스 디스크 관련 명령어(mount, fdisk, mkfs 등) 및 git 관련 명령어도 포함입니다.

✓ https://www.multipath-tcp.org에 따라 mptcp-v0.94가 linux kernel 4.14.y를 기반으로 생성되었음으로
본 내용에서 rpi-4.14.y를 사용하여 mptcp-v0.94를 적용시켜보도록 한다.

목차

1. RPI kernel 4.14.y와 MPTCP kernel v0.94를 Merge 시키기

1.1. Linux PC에서 RPI kernel을 git clone 받기

```
$ git clone https://github.com/raspberrypi/linux.git
```

1.2. clone된 rpi kernel 디렉터리로 이동 후 MPTCP kernel을 remote add 후 fetch

```
$ cd linux
```

$ git remote add mptcp https://github.com/multipath-tcp/mptcp.git

$ git fetch mptcp

```
inlab@INLAB-SERVER:~/mptcp/archlinux-rpi-mptcp$ git clone https://github.com/raspberrypi/linux.git
fatal: destination path 'linux' already exists and is not an empty directory.
inlab@INLAB-SERVER:~/mptcp/archlinux-rpi-mptcp$ ^C
inlab@INLAB-SERVER:~/mptcp/archlinux-rpi-mptcp$ ls
linux
inlab@INLAB-SERVER:~/mptcp/archlinux-rpi-mptcp$ cd linux
inlab@INLAB-SERVER:~/mptcp/archlinux-rpi-mptcp/linux$ ls
COPYING        Kbuild    MAINTAINERS  arch   crypto   include  kernel  net      security  usr
CREDITS        Kconfig   Makefile     block  drivers  init     lib     samples  sound     virt
Documentation  LICENSES  README       certs  fs       ipc      mm      scripts  tools
inlab@INLAB-SERVER:~/mptcp/archlinux-rpi-mptcp/linux$ git remote add mptcp https://github.com/multipa
th-tcp/mptcp.git
inlab@INLAB-SERVER:~/mptcp/archlinux-rpi-mptcp/linux$ git fetch mptcp
remote: Enumerating objects: 158433, done.
remote: Counting objects: 100% (158433/158433), done.
remote: Compressing objects: 100% (87/87), done.
remote: Total 439354 (delta 158359), reused 158405 (delta 158346), pack-reused 280921
Receiving objects: 100% (439354/439354), 221.16 MiB | 10.17 MiB/s, done.
Resolving deltas: 100% (361096/361096), completed with 33837 local objects.
From https://github.com/multipath-tcp/mptcp
 * [new branch]               mptcp_trunk -> mptcp/mptcp_trunk
 * [new branch]               mptcp_v0.86 -> mptcp/mptcp_v0.86
 * [new branch]               mptcp_v0.87 -> mptcp/mptcp_v0.87
 * [new branch]               mptcp_v0.88 -> mptcp/mptcp_v0.88
 * [new branch]               mptcp_v0.89 -> mptcp/mptcp_v0.89
```

1.3. rpi-4.14.y 버전의 새로운 브랜치 생성

$ git checkout -b rpi_mptcp origin/rpi-4.14.y

1.4. rpi-4.14.y와 mptcp_v0.94의 코드를 merge (충돌코드는 mptcp 측으로 수정되도록 git 옵션 적용)

$ git merge -s recursive -X theirs mptcp/mptcp_v0.94

```
inlab@INLAB-SERVER:~/mptcp/archlinux-rpi-mptcp/linux$ git checkout -b rpi_mptcp origin/rpi-4.14.y
Checking out files: 100% (73328/73328), done.
Branch 'rpi_mptcp' set up to track remote branch 'rpi-4.14.y' from 'origin'.
Switched to a new branch 'rpi_mptcp'
inlab@INLAB-SERVER:~/mptcp/archlinux-rpi-mptcp/linux$ git merge -s recursive -X theirs mptcp/mptcp_v0
.94
Auto-merging sound/usb/quirks.c
Auto-merging sound/usb/format.c
Auto-merging sound/usb/card.h
Auto-merging sound/soc/codecs/pcm512x.c
Auto-merging sound/soc/codecs/cs4265.c
Auto-merging scripts/Kbuild.include
Auto-merging mm/page_alloc.c
Removing kernel/futex_compat.c
Auto-merging kernel/cgroup/cgroup.c
Auto-merging include/uapi/linux/videodev2.h
Auto-merging include/linux/phy.h
Auto-merging include/linux/mmc/card.h
Auto-merging fs/f2fs/inode.c
```

2. Merge시 발생되는 코드 에러 잡기 (3번 수행 후 에러가 발생하면 진행)

2.1. merge 시킨 뒤 곧바로 cross compile을 진행하게 될 시에 아래와 같은 에러가 발생될 가능성이 존재

```
  CC [M]  drivers/mtd/spi-nor/spi-nor.o
drivers/mtd/nand/bcm2835_smi_nand.c: In function ‘bcm2835_smi_nand_probe’:
drivers/mtd/nand/bcm2835_smi_nand.c:225:15: error: passing argument 1 of ‘nand_release’ from incompat
ible pointer type [-Werror=incompatible-pointer-types]
  nand_release(mtd);
               ^~~
In file included from drivers/mtd/nand/bcm2835_smi_nand.c:42:0:
./include/linux/mtd/rawnand.h:41:6: note: expected ‘struct nand_chip *’ but argument is of type ‘stru
ct mtd_info *’
 void nand_release(struct nand_chip *chip);
      ^~~~~~~~~~~~
drivers/mtd/nand/bcm2835_smi_nand.c: In function ‘bcm2835_smi_nand_remove’:
drivers/mtd/nand/bcm2835_smi_nand.c:233:15: error: passing argument 1 of ‘nand_release’ from incompat
ible pointer type [-Werror=incompatible-pointer-types]
  nand_release(&host->mtd);
               ^
In file included from drivers/mtd/nand/bcm2835_smi_nand.c:42:0:
./include/linux/mtd/rawnand.h:41:6: note: expected ‘struct nand_chip *’ but argument is of type ‘stru
ct mtd_info *’
 void nand_release(struct nand_chip *chip);
      ^~~~~~~~~~~~
```

>> 충돌코드를 mptcp 측의 소스코드로 수정하여서 발생되는 에러

>> 간단한 코드 수정으로 에러를 잡을 수 있다. (에러 발생 일자 : 2021-03-18)

>> 또한, *cross compile을 수행해 보았을 시 해당 에러가 발생하지 않았다면 곧바로 4.번으로 넘어가도록 하자.*

2.2. 에러 발생 부분

- drivers/mtd/nand/bcm2835_smi_nand.c

```
220
221         ret = mtd_device_parse_register(mtd, NULL, &ppdata, NULL, 0);
222         if (!ret)
223                 return 0;
224
225         nand_release(mtd);   → 에러발생 부분
226         return -EINVAL;
227 }
228
229 static int bcm2835_smi_nand_remove(struct platform_device *pdev)
230 {
231         struct bcm2835_smi_nand_host *host = platform_get_drvdata(pdev);
232
233         nand_release(&host->mtd);   → 에러발생 부분
234
235         return 0;
236 }
```

2.3. nand_release()의 mptcp kernel과 rpi kernel 소스코드 비교분석

- rpi-4.14.y/drivers/mtd/nand/nand_base.c

```
5051  void nand_release(struct mtd_info *mtd)
5052  {
5053          mtd_device_unregister(mtd);
5054          nand_cleanup(mtd_to_nand(mtd));
5055  }
5056  EXPORT_SYMBOL_GPL(nand_release);
```

→ 기존 rpi-4.14.y의 소스코드

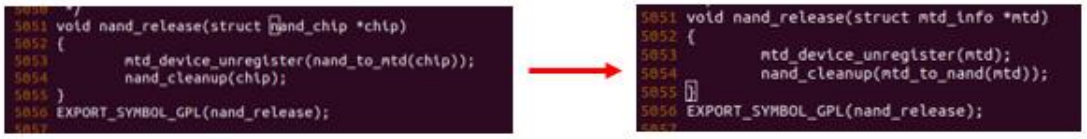
- mptcp/drivers/mtd/nand/nand_base.c

```
5051  void nand_release(struct nand_chip *chip)
5052  {
5053          mtd_device_unregister(nand_to_mtd(chip));
5054          nand_cleanup(chip);
5055  }
5056  EXPORT_SYMBOL_GPL(nand_release);
```

→ Merge시 바뀐 소스코드

2.4. 소스코드 수정

- drivers/mtd/nand/nand_base.c



- include/linux/mtd/rawnand.h

3. Arm 커널 크로스 컴파일

3.1. 의존패키지 설치

$ apt-get install build-essential bc make ncurses-dev wget unzip bison

3.2. Arm 크로스 컴파일러 설치

$ apt-get install gcc-arm-linux-gnueabihf

3.3. 설정파일 생성 (.config)

$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- bcm2709_defconfig

3.4. 설정파일의 내용 변경

$ vi .config

```
#
# General setup
#
CONFIG_INIT_ENV_ARG_LIMIT=32
CONFIG_CROSS_COMPILE=""
# CONFIG_COMPILE_TEST is not set
CONFIG_LOCALVERSION="-MPTCP"
# CONFIG_LOCALVERSION_AUTO is not set
CONFIG_HAVE_KERNEL_GZIP=y
```

>> CONFIG_LOCALVERSION="-v7" -> CONFIG_LOCALVERSION="-MPTCP"

>> 커널 버전 뒤의 이름을 MPTCP로 변경

3.5. 설정파일에 MPTCP 설정을 적용 (.config)

$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- menuconfig

4.4.1. "Networking support" 하위로 이동 (Enter)

4.4.2. "Networking options" 하위로 이동

4.4.3. 아래로 스크롤 하여 "The IPv6 protocol"을 check mark [*] (Space)

4.4.4. "IPv6 protocol"이 check mark가 되었다면, "MPTCP protocol (NEW)"가 생성됨 해당 옵션 check mark

4.4.5. "MPTCP protocol"이 check mark가 되었다면, "MPTCP: advanced path-manager control (NEW)"와 "MPTCP : advanced scheduler control (NEW)"가 생성됨 두 옵션 모두 check mark

4.4.6. 두 옵션을 모두 check mark 한 뒤, "MPTCP: advanced path-manager control" 하위로 이동

4.4.7. "MPTCP Full-Mesh Path-Manager", "MPTCP ndiff-ports", "MPTCP Binder" 모두 check mart

4.4.8. Default MPTCP Path-Manager를 "Full mesh"로 적용

4.4.9. 상위로 빠져나오기 (Esc 두 번)

4.4.10. "MPTCP: advanced scheduler control" 하위로 이동

4.4.11. "MPTCP Round-Robin", "MPTCP Redundant" 모두 check mark

4.4.12. Default MPTCP Scheduler는 Default로 적용

4.4.13. 상위로 빠져나오기

4.4.14. "TCP: advanced congestion control" 하위로 이동

4.4.15. "MPTCP Linked Increase", "MPTCP Opportunistic Linked Increase", "MPTCP WVEGAS CONGESTION CONTROL", "MPTCP BALIA CONGESTION CONTROL" 모두 check mark

4.4.16. Default TCP congestion control을 "Lia"로 적용

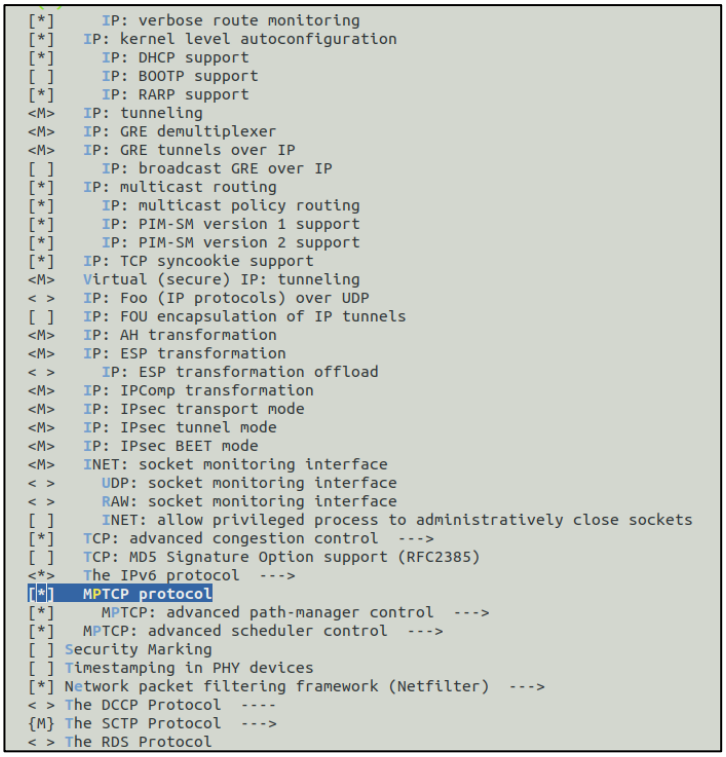
4.4.17. Yes와 No라는 문구가 나올 때까지 Esc를 연달아 누르기

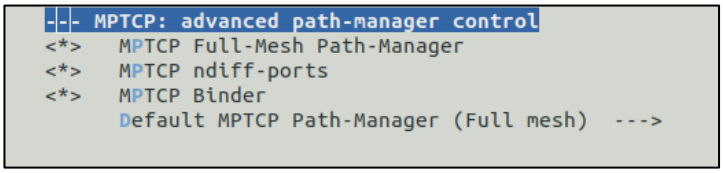
4.4.18. Yes를 눌러서 설정파일(.config)에 MPTCP 설정 적용시키기

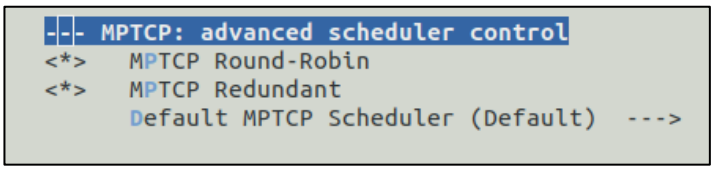
```
    -*- Patch physical to virtual translations at runtime
        General setup  --->
    [*] Enable loadable module support  --->
    [*] Enable the block layer  --->
        System Type  --->
        Bus support  --->
        Kernel Features  --->
        Boot options  --->
        CPU Power Management  --->
        Floating point emulation  --->
        Userspace binary formats  --->
        Power management options  --->
    [*] Networking support  --->
        Device Drivers  --->
        Firmware Drivers  --->
        File systems  --->
        Kernel hacking  --->
        Security options  --->
    -*- Cryptographic API  --->
        Library routines  --->
    [ ] Virtualization  ----
```

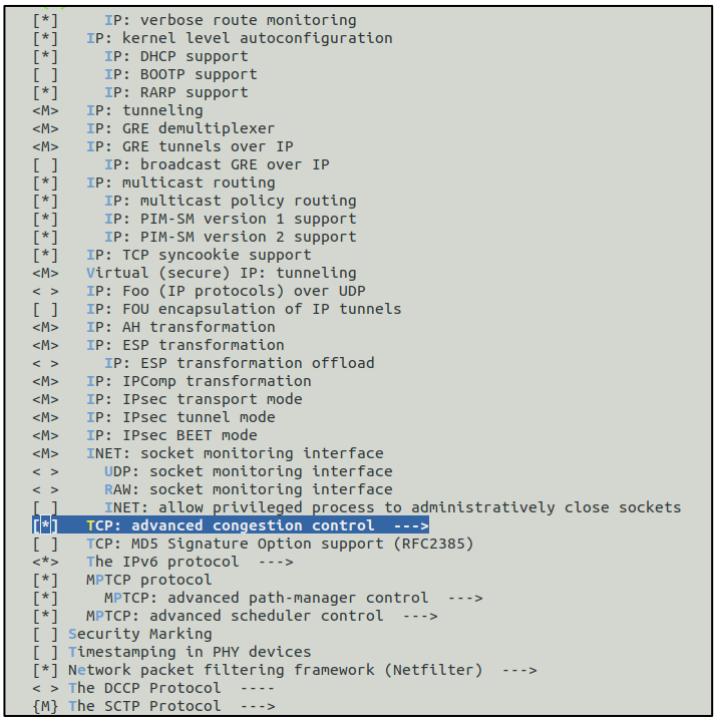
```
--- Networking support
        Networking options  --->
[*]   Amateur Radio support  --->
<M>   CAN bus subsystem support  --->
<M>   Bluetooth subsystem support  --->
< >   RxRPC session sockets
< >   KCM sockets
-*-   Wireless  --->
<M>   WiMAX Wireless Broadband support  --->
<M>   RF switch subsystem support  --->
<M>   Plan 9 Resource Sharing Support (9P2000)  --->
< >   CAIF support  ----
< >   Ceph core library
<M>   NFC subsystem support  --->
< >   Packet-sampling netlink channel  ----
< >   Inter-FE based on IETF ForCES InterFE LFB  ----
[ ]   Network light weight tunnels
< >   Network physical/parent device Netlink interface
```

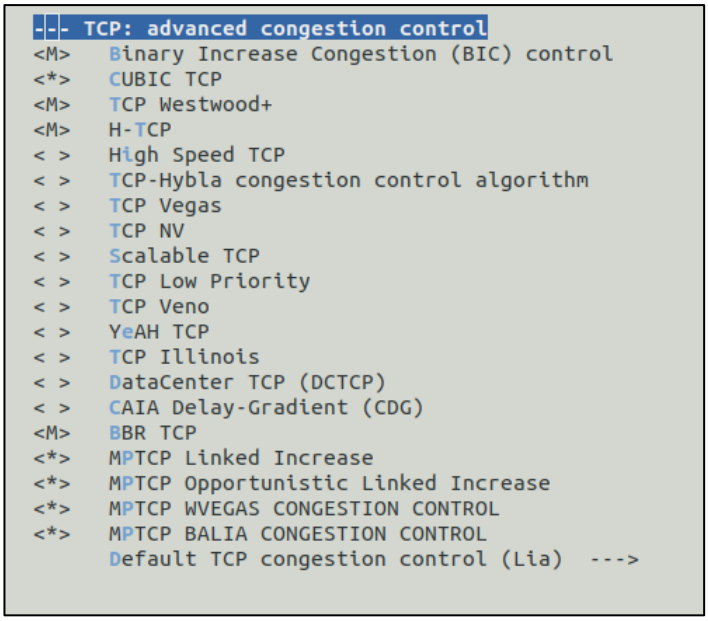
```
[*]     IP: verbose route monitoring
[*]   IP: kernel level autoconfiguration
[*]     IP: DHCP support
[ ]     IP: BOOTP support
[*]     IP: RARP support
<M>   IP: tunneling
<M>   IP: GRE demultiplexer
<M>   IP: GRE tunnels over IP
[ ]     IP: broadcast GRE over IP
[*]   IP: multicast routing
[*]     IP: multicast policy routing
[*]     IP: PIM-SM version 1 support
[*]     IP: PIM-SM version 2 support
[*]   IP: TCP syncookie support
<M>   Virtual (secure) IP: tunneling
< >   IP: Foo (IP protocols) over UDP
[ ]   IP: FOU encapsulation of IP tunnels
<M>   IP: AH transformation
<M>   IP: ESP transformation
< >     IP: ESP transformation offload
<M>   IP: IPComp transformation
<M>   IP: IPsec transport mode
<M>   IP: IPsec tunnel mode
<M>   IP: IPsec BEET mode
<M>   INET: socket monitoring interface
< >     UDP: socket monitoring interface
< >     RAW: socket monitoring interface
[ ]     INET: allow privileged process to administratively close sockets
[*]   TCP: advanced congestion control  --->
[ ]   TCP: MD5 Signature Option support (RFC2385)
<*>   The IPv6 protocol  --->
[ ]   MPTCP protocol (NEW)
[ ] Security Marking
[ ] Timestamping in PHY devices
[*] Network packet filtering framework (Netfilter)  --->
< > The DCCP Protocol  ----
{M} The SCTP Protocol  --->
< > The RDS Protocol
< > The TIPC Protocol  ----
<M> Asynchronous Transfer Mode (ATM)
```


[*] IP: verbose route monitoring
[*] IP: kernel level autoconfiguration
[*] IP: DHCP support
[ ] IP: BOOTP support
[*] IP: RARP support
<M> IP: tunneling
<M> IP: GRE demultiplexer
<M> IP: GRE tunnels over IP
[ ] IP: broadcast GRE over IP
[*] IP: multicast routing
[*] IP: multicast policy routing
[*] IP: PIM-SM version 1 support
[*] IP: PIM-SM version 2 support
[*] IP: TCP syncookie support
<M> Virtual (secure) IP: tunneling
< > IP: Foo (IP protocols) over UDP
[ ] IP: FOU encapsulation of IP tunnels
<M> IP: AH transformation
<M> IP: ESP transformation
< > IP: ESP transformation offload
<M> IP: IPComp transformation
<M> IP: IPsec transport mode
<M> IP: IPsec tunnel mode
<M> IP: IPsec BEET mode
<M> INET: socket monitoring interface
< > UDP: socket monitoring interface
< > RAW: socket monitoring interface
[ ] INET: allow privileged process to administratively close sockets
[*] TCP: advanced congestion control --->
[ ] TCP: MD5 Signature Option support (RFC2385)
<*> The IPv6 protocol --->
[*] MPTCP protocol
[*] MPTCP: advanced path-manager control --->
[*] MPTCP: advanced scheduler control --->
[ ] Security Marking
[ ] Timestamping in PHY devices
[*] Network packet filtering framework (Netfilter) --->
< > The DCCP Protocol ----
{M} The SCTP Protocol --->
< > The RDS Protocol


--- MPTCP: advanced path-manager control
<*> MPTCP Full-Mesh Path-Manager
<*> MPTCP ndiff-ports
<*> MPTCP Binder
Default MPTCP Path-Manager (Full mesh) --->


--- MPTCP: advanced scheduler control
<*> MPTCP Round-Robin
<*> MPTCP Redundant
Default MPTCP Scheduler (Default) --->


[*] IP: verbose route monitoring
[*] IP: kernel level autoconfiguration
[*] IP: DHCP support
[ ] IP: BOOTP support
[*] IP: RARP support
<M> IP: tunneling
<M> IP: GRE demultiplexer
<M> IP: GRE tunnels over IP
[ ] IP: broadcast GRE over IP
[*] IP: multicast routing
[*] IP: multicast policy routing
[*] IP: PIM-SM version 1 support
[*] IP: PIM-SM version 2 support
[*] IP: TCP syncookie support
<M> Virtual (secure) IP: tunneling
< > IP: Foo (IP protocols) over UDP
[ ] IP: FOU encapsulation of IP tunnels
<M> IP: AH transformation
<M> IP: ESP transformation
< > IP: ESP transformation offload
<M> IP: IPComp transformation
<M> IP: IPsec transport mode
<M> IP: IPsec tunnel mode
<M> IP: IPsec BEET mode
<M> INET: socket monitoring interface
< > UDP: socket monitoring interface
< > RAW: socket monitoring interface
[ ] INET: allow privileged process to administratively close sockets
[*] TCP: advanced congestion control --->
[ ] TCP: MD5 Signature Option support (RFC2385)
<*> The IPv6 protocol --->
[*] MPTCP protocol
[*] MPTCP: advanced path-manager control --->
[*] MPTCP: advanced scheduler control --->
[ ] Security Marking
[ ] Timestamping in PHY devices
[*] Network packet filtering framework (Netfilter) --->
< > The DCCP Protocol ----
{M} The SCTP Protocol --->



3.6. 커널 이미지 및 모듈 cross compile 진행

$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- -j$(nproc) zImage modules dtbs

$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- INSTALL_MOD_PATH=../modules -j$(nproc) modules_ install

```
inlab@INLAB-SERVER:~/mptcp/archlinux-rpi-mptcp/linux$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf
- -j$(nproc) zImage modules dtbs
scripts/kconfig/conf  --silentoldconfig Kconfig
  CHK     include/config/kernel.release
  SYSHDR  arch/arm/include/generated/uapi/asm/unistd-common.h
  SYSHDR  arch/arm/include/generated/uapi/asm/unistd-oabi.h
  WRAP    arch/arm/include/generated/uapi/asm/bitsperlong.h
  WRAP    arch/arm/include/generated/uapi/asm/errno.h
  WRAP    arch/arm/include/generated/uapi/asm/ioctl.h
  WRAP    arch/arm/include/generated/uapi/asm/ipcbuf.h
  WRAP    arch/arm/include/generated/uapi/asm/msgbuf.h
  WRAP    arch/arm/include/generated/uapi/asm/param.h
  WRAP    arch/arm/include/generated/uapi/asm/poll.h
  WRAP    arch/arm/include/generated/uapi/asm/resource.h
  WRAP    arch/arm/include/generated/uapi/asm/sembuf.h
  WRAP    arch/arm/include/generated/uapi/asm/shmbuf.h
  WRAP    arch/arm/include/generated/uapi/asm/siginfo.h
  WRAP    arch/arm/include/generated/uapi/asm/socket.h
  UPD     include/config/kernel.release
  WRAP    arch/arm/include/generated/uapi/asm/sockios.h
  WRAP    arch/arm/include/generated/uapi/asm/termbits.h
                              :
                              :
                              :
inlab@INLAB-SERVER:~/mptcp/archlinux-rpi-mptcp/linux$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf
- INSTALL_MOD_PATH=../modules -j$(nproc) modules_install
```

3.7. 생성된 커널 이미지와 커널 모듈을 SD카드로 이동

$ mv arch/arm/boot/zImage boot/kernel7.img

$ mv arch/arm/boot/dts/*.dtb boot/.

$ mv arch/arm/boot/dts/overlays/*.dtb* boot/overlays/.

$ mv ../modules/lib/modules/4.14.???-MPTCP/ root/lib/modules/.

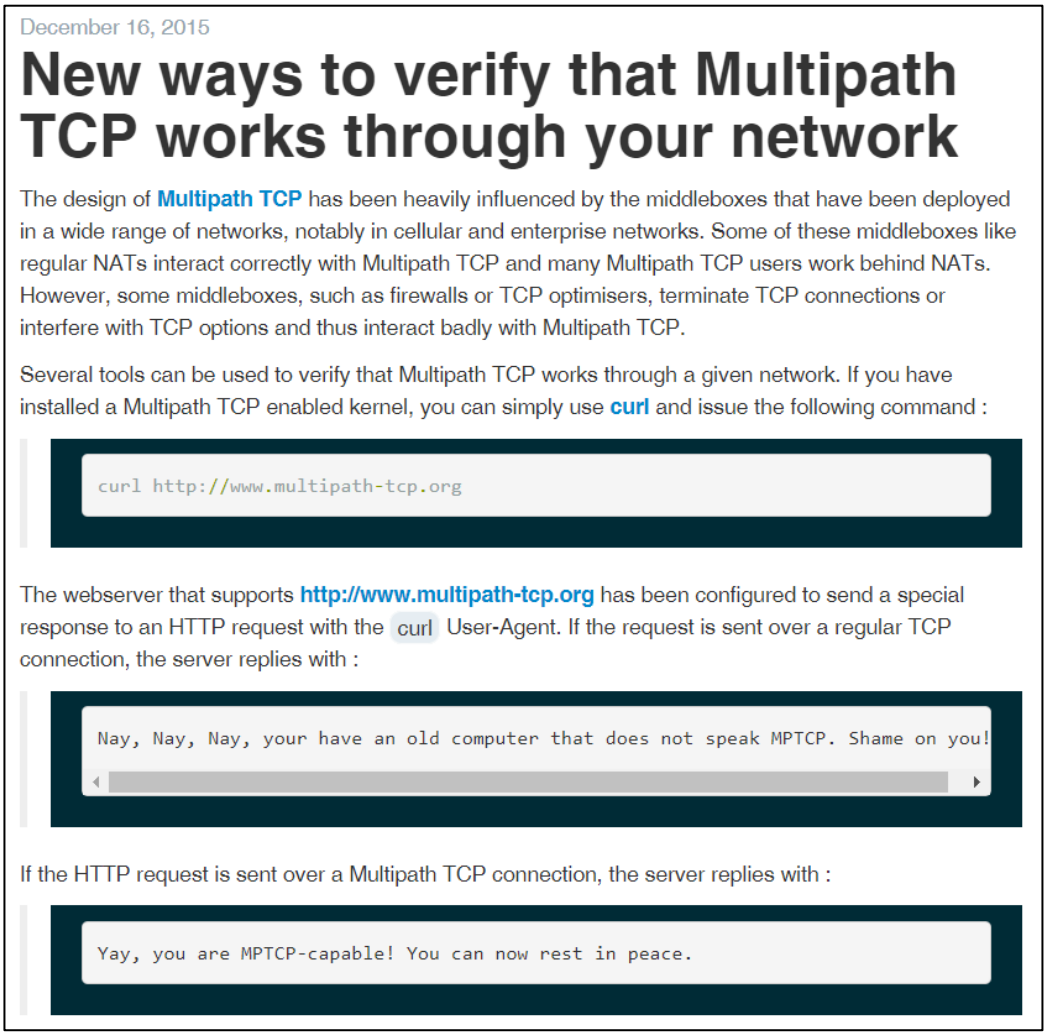
>> 필자는 독자의 boot와 root 디렉터리의 위치를 정확하게 알지 못함.

>> 독자가 cross compile에 익숙한 사람이라면 아마도 이해가 가능할 것 같아 이렇게만 작성하였다.

3.8. 이제 MPTCP가 적용된 SD 카드를 RPI3 B+에 삽입시켜 정상적으로 부팅되는지 확인하도록 하자.
만약, 정상적으로 부팅이 되지 않는다면 앞서 진행한 내용 중 놓친 부분이 없는지 다시 확인하고 진행하도록 하자. 계속해도 부팅되지 않는다면 https://technofaq.org/posts/2018/09/how-to-compile-mptcp-linux-kernel-on-raspberry-pi-2-and-raspberry-pi-3-on-archlinux-arm/ 해당 사이트를 참조하도록 하자.

4. MPTCP 테스트

4.1. 이제 MPTCP가 RPI Kernel에서 정상적으로 작동되는지 확인해보도록 하자.
아래 그림에서 해당 도메인으로 HTTP Request를 보내게 되면 Response를 통해 MPTCP를 지원하는 kernel인지 아닌지 확인시켜 준다고 한다. 이 도메인을 활용하여 테스트해보도록 하자.


December 16, 2015

New ways to verify that Multipath TCP works through your network

The design of **Multipath TCP** has been heavily influenced by the middleboxes that have been deployed in a wide range of networks, notably in cellular and enterprise networks. Some of these middleboxes like regular NATs interact correctly with Multipath TCP and many Multipath TCP users work behind NATs. However, some middleboxes, such as firewalls or TCP optimisers, terminate TCP connections or interfere with TCP options and thus interact badly with Multipath TCP.

Several tools can be used to verify that Multipath TCP works through a given network. If you have installed a Multipath TCP enabled kernel, you can simply use **curl** and issue the following command :

```
curl http://www.multipath-tcp.org
```

The webserver that supports **http://www.multipath-tcp.org** has been configured to send a special response to an HTTP request with the `curl` User-Agent. If the request is sent over a regular TCP connection, the server replies with :

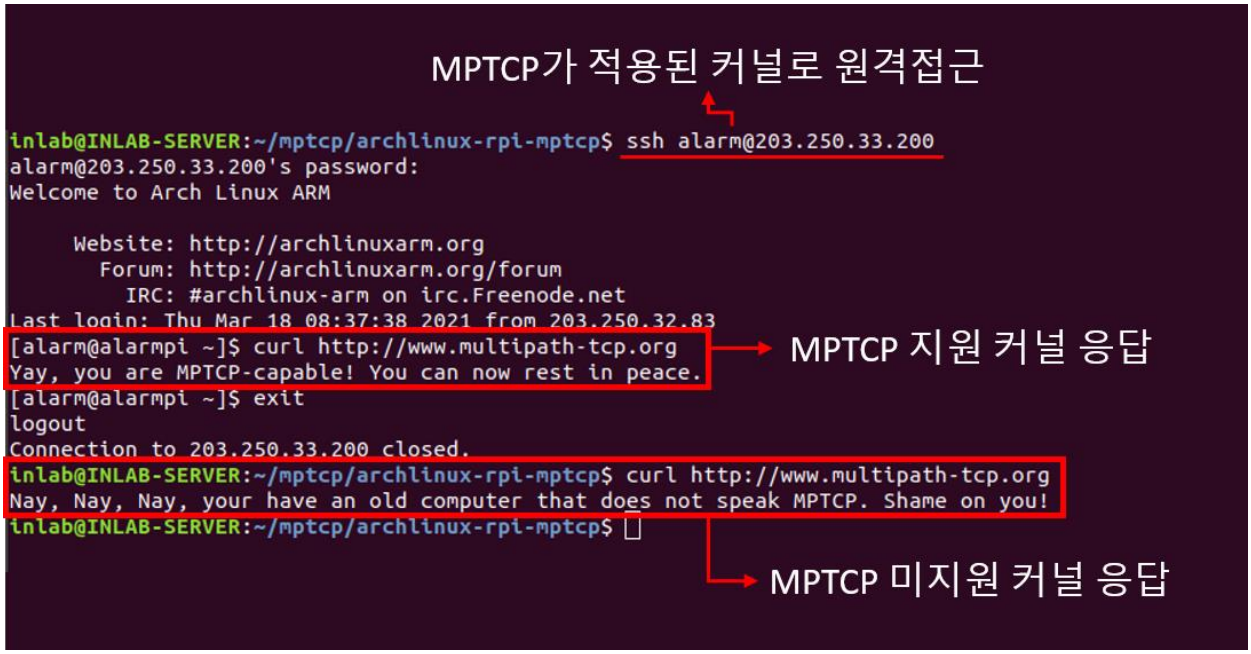
```
Nay, Nay, Nay, your have an old computer that does not speak MPTCP. Shame on you!
```

If the HTTP request is sent over a Multipath TCP connection, the server replies with :

```
Yay, you are MPTCP-capable! You can now rest in peace.
```

>> http://blog.multipath-tcp.org/blog/html/2015/12/16/mptcp_tools.html

4.2. 테스트 진행



>> 필자는 ssh를 활용해 rpi로 접근하였고 앞 5.1.의 그림을 참조하여 curl 명령어를 통해 테스트를 진행하였다.

>> 정상적으로 rpi는 MPTCP를 지원한다고 응답 받았다.

>> 그리고 필자의 Linux Desktop PC에는 MPTCP를 올리지 않았으니 당연히 MPTCP를 지원하지 않는다고 응답 받았다.

참조링크

- https://technofaq.org/posts/2018/09/how-to-compile-mptcp-linux-kernel-on-raspberry-pi-2-and-raspberry-pi-3-on-archlinux-arm/
- http://blog.multipath-tcp.org/blog/html/2015/12/16/mptcp_tools.html