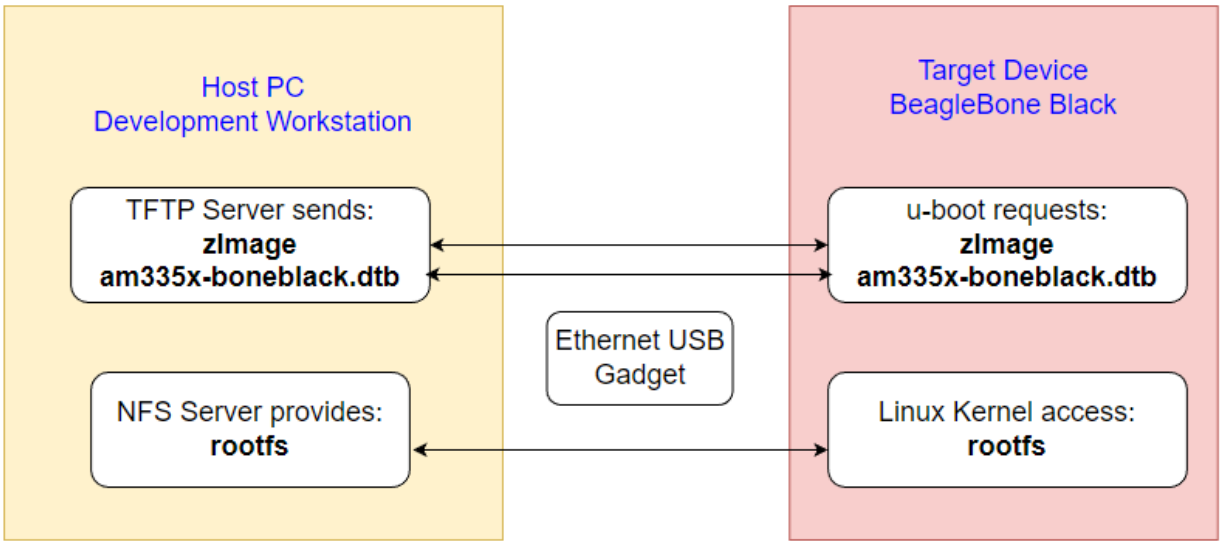
Ethernet over USB Gadget

Objetivo

El presente documento pretende mostrar una serie de pasos para habilitar el soporte de red del bootloader y del kernel utilizando el soporte de “USB Ethernet” Gadget con el que cuenta la tarjeta de desarrollo BeagleBone Black. Estos pasos se ejecutan tanto del lado del “Host” (Ubuntu VM) como en el “Target/Device” (BeagleBone Black, de ahora en adelante **BBB**)

Digarama

En el siguiente diagrama se ejemplifica las 2 partes involucradas en la red local y el funcionamiento o “rol” de cada uno de los nodos en la red local.



U-boot “de fábrica” y configurado para ejecutarse en una **BBB** tiene el soporte de “USB Ethernet” Gadget, el cual no es más que dotar a la tarjeta de desarrollo de las capacidades suficientes para que el puerto micro-USB y la tarjeta funcionen como un adaptador Ethernet, y el propio cable micro-USB conectado entre la PC y la tarjeta funcione como el medio físico sobre el cual la conexión Ethernet se va a establecer. El Linux Kernel, a pesar de contar con un soporte de red para este y muchos más casos de uso, las configuraciones necesarias para activar el soporte de “USB Ethernet” Gadget no están habilitadas por defecto. Uno mismo tiene que habilitar estas configuraciones como primer paso, construir el Linux Kernel y probarlo en el sistema. Pero para efectos prácticos, se proporcionará un Kernel, un DTB y un Rootfs ya precompilados listos para usarse.

Preparativos generales

Antes que nada, hay que verificar que la **BBB** es capaz de arrancar o “bootear” desde una tarjeta uSD como se ha visto en sesiones anteriores. La tarjeta uSD debe de estar correctamente particionada y dentro de la partición primaria y “booteable” se deben de encontrar tanto el SPL como el u-boot.

Tanto el repositorio oficial de u-boot en mainline (https://github.com/u-boot/u-boot.git) como la configuración por defecto que hemos estado usando hasta ahora (am335x_evm_defconfig) contiene habilitado todo el soporte necesario para ofrecer la funcionalidad de "USB Ethernet" Gadget. Si usted desea probar con otra configuración distinta compatible con la tarjeta, o desea usar el repositorio oficial de BeagleBoard.org para u-boot (https://github.com/beagleboard/u-boot.git) puede revisar si el soporte y la experiencia es la misma que usando las fuentes descritas en este ejercicio.

1. Conectar el cable USB Serial TTL-232R-3V3 a la interfaz de debug de la BBB y a la computadora
2. Iniciar sesión con una herramienta de emulación de terminal como picocom, minicom, etc.
3. Preparar la tarjeta uSD con nuestro SPL y u-boot, insertar la tarjeta uSD en el puerto de la BBB y arrancar el sistema desde ahí.
4. Interrumpir el proceso de arranque presionando alguna tecla en la consola
5. Desde una consola en nuestra máquina virtual de Linux, ejecutar el comando "**ip address**" y revisar cuántas conexiones de red están configuradas actualmente
6. Probar que el comando ping funciona: **ping 192.168.0.1**
7. En este momento el comando debería de funcionar, pero no debería de regresar una respuesta válida ya que no hemos configurado los parámetros de la red y la dirección IP no puede ser localizable aún.
8. Mientras el comando **ping** en u-boot está en ejecución, ir nuevamente a la consola de Linux y ejecutar el comando "**ip address**" y revisar cuántas conexiones de red están configuradas actualmente. En este momento deberíamos de ver que hay una diferencia entre la salida de la primera ejecución del comando "ip address" y la segunda. Tomar nota del nombre de la nueva conexión de red que se muestra.

Preparación del ambiente en la VM (Ubuntu)

El objetivo de preparar el ambiente en una Workstation de desarrollo con Linux es que el proceso de probar cambios en la tarjeta sea lo más eficiente y cómodo posible. Para ello tenemos que instalar dos paquetes y configurarlos: los servidores de los servicios de **TFTP y NFS**. Con esto convertiremos nuestra máquina virtual de Linux en un servidor que atiende las peticiones de uno o más clientes, en nuestro caso el cliente será la BBB.

1. Ahora es momento de configurar la red, al menos en la máquina virtual. Aunque la configuración se puede utilizando la interfaz gráfica de usuario, siempre es preferible hacer la configuración desde consola: **nmcli con add type ethernet ifname enxf8dc7a000001 ip4 192.168.0.1/24**
2. El parámetro después de **ifname** no es mas que el nombre que apareció en nuestra segunda ejecución del comando **ip address**
3. Instalar el paquete "**nfs-kernel-server**"
4. Instalar el paquete "**tftpd-hpa"** y revisar el contenido del archivo **"/etc/default/tftpd-hpa"**. Es en este directorio donde se configura cuál va a ser el directorio donde se van a "servir" los archivos que los clientes deseen descargar por medio del protocolo TFTP. Verificar que el directorio especificado en el archivo de configuración exista, o de lo contrario crear el directorio.
5. Si se hace un cambio a la configuración, hay que reiniciar el servicio: **sudo /etc/init.d/tftpd-hpa restart**

6. Hagamos una prueba: intenta creando un archivo llamado hello.txt con el contenido “Hello, u-boot!” y guárdalo en el directorio donde se sirven los archivos del servidor TFTP.
7. Con esto acabamos los preparativos en la máquina virtual Linux.

Preparación del ambiente en la tarjeta de desarrollo (BBB)

La tarjeta de desarrollo, o en nuestro caso la BBB, será quien se encargue de solicitar la descarga de los archivos por red necesarios al servidor, que en nuestro caso es una máquina virtual Llinux.

1. Con la consola de u-boot lista para usarse, es momento de configurar la red del lado de la tarjeta en u-boot. Hay 5 variables de ambiente que te van a permitir configurar todos los parámetros necesarios para conectarnos con la máquina virtual Linux:
    a. **ipaddr**: Dirección IP de la tarjeta
    b. **serverip**: Dirección IP de la máquina virtual Linux
    c. **ethprime**: Especifica cuál interfaz de red va a ser usada primeramente. u-boot puede contar con varias interfaces de red disponibles al mismo tiempo y, para las operaciones que hacen uso de la red, u-boot trata de usar todas las interfaces de red una a la vez hasta que una de ellas resulta en una conexión exitosa. Es por eso se necesita especificar cuál va a ser la primera para evitar usar una interfaz no deseada.
    d. **usbnet_devaddr**: la dirección MAC del lado del dispositivo
    e. **usbnet_hostaddr**: la dirección MAC del lado de la máquina virtual Linux
2. Usando el comando **setenv**, configura las 5 variables de entorno con sus respectivos valores:
    a. setenv ipaddr 192.168.0.100
    b. setenv serverip 192.168.0.1
    c. setenv ethprime usb_ether
    d. setenv usbnet_devaddr f8:dc:7a:00:00:02
    e. setenv usbnet_hostaddr f8:dc:7a:00:00:01
3. Guarda el estado del ambiente con el comando **saveenv**
4. Ejecuta el comando tftp para descargar el archivo que recién creamos en la máquina virtual Linux: **tftp 0x81000000 textfile.txt**
5. El contenido del archivo **textfile.txt** se debería de haber guardado en la memoria RAM de la tarjeta con dirección **0x81000000**. Prueba leyendo el contenido de esa dirección: **md 0x81000000**
6. El comando md significa “memory display”. Es parte de una familia de comandos cuya función principal es realizar operaciones de lectura y escritura de la memoria RAM. En la consola debería de mostrarse el contenido “a partir” de esa dirección. Como salida, en la consola de u-boot debería de mostrarse la leyenda “Hello, u-boot!” en hexadecimal de un lado y su representación imprimible en ASCII del otro.
7. Con esto acabamos los preparativos en la tarjeta y u-boot.

Cargar el Linux Kernel por medio del protocolo de red TFTP

Con el ambiente de red probado, ahora podemos pasar a cargar algo más interesante que un archivo de texto: el Kernel de Linux.

1. Copia los dos archivos adjuntos a esta práctica al directorio donde se sirven los archivos con el servidor TFTP:
    a. **zImage**
    b. **am335x-boneblack.dtb**
2. Descarga por medio de TFTP el Linux Kernel en la dirección **0x81000000**
3. Descarga por medio de TFTP el DTB en la dirección **0x82000000**
4. Continua el arranque del sistema utilizando estos dos archivos y el comando bootz. El comando bootz recibe dos parámetros separados por un espacio, un guión medio otro espacio " - ". El primer parámetro es la dirección en memoria RAM donde se encuentra el Linux Kernel, y el segundo es para el DTB
5. Verifica que el Linux Kernel haya cargado correctamente
6. Si el Linux Kernel no cargó correctamente es porque faltó especificar algo muy importante: el Root File System o rootfs. El kernel está esperando cargar el rootfs (la última etapa en el arranque de un sistema Linux) en alguna parte, pero como el Kernel está precompilado y fue pensado para ser cargado por red (este kernel no viene con un rootfs integrado dentro del kernel, o un "initramfs"), lo más lógico seria que el rootfs sea cargado también por red.

Cargar el rootfs por medio del protocolo de red NFS

El Linux Kernel no tiene "hard-codeado" (al menos no este) el rootfs que va a utilizar para terminar de cargar un sistema. Muchas veces puede ser especificado por medio de unas variables o parámetros que el bootloader le pasa al kernel en donde le "avisa" donde se encuentra el rootfs. Estos parámetros se conocen como los kernel parameters o "Kernel's command-line parameters". Es uno de estos parámetros en donde se le especifica en donde está el rootfs y, si dicho rootfs se encuentra como un recurso en la red, también se puede configurar la dirección IP del servidor NFS que está "sirviendo" actualmente el rootfs que el Linux Kernel necesita para terminar de cargar el sistema.

1. Descomprimir el archivo nfsroot.tar.xz en algún lugar dentro del directorio home. Por ejemplo: /home/uidr7643/diplomado/modulo_3/
2. El archivo .tar.xz ya cuenta con un directorio llamado nfsroot dentro de su estructura. Hay que tomar esto en consideración al descomprimir el archivo en el sitio que se desee.
3. En u-boot, imprimir el contenido de la variable **bootargs.** Esta variable de entorno es la que especifica cuáles van a ser los parámetros que se le van a pasar al Linux Kernel cuando este sea cargado. Esta variable guarda los Kernel Params.
4. Al parecer la variable no existe o no tiene ningún valor. El ambiente no ha sido configurado para cargar un kernel + parámetros aún. Configuremos los bootargs con el comando setenv
    a. setenv bootargs root=/dev/nfs rw ip=192.168.0.100:::::usb0 console=ttyS0,115200n8 g_ether.dev_addr=f8:dc:7a:00:00:02 g_ether.host_addr=f8:dc:7a:00:00:01 nfsroot=192.168.0.1:/home/uidr7643/diplomado/modulo_3/nfsroot,nfsvers=3,tcp
5. La ruta especificada en el parámetro **nfsroot** necesita ser adecuada al caso específico de cada usuario.

6. Edita el archivo **/etc/exports** (como root) y agrega la siguiente línea, asumiendo que la dirección IP de la tarjeta será la **192.168.0.100**
   /home/uidr7643/diplomado/modulo_3/nfsroot 192.168.0.100(rw,no_root_squash,no_subtree_check)
7. Guarda y cierra el archivo, y reinicia el servidor NFS:
   a. sudo /etc/init.d/nfs-kernel-server restart
8. La mayoría de los errores que aparecen al reiniciar un servidor después de haber editado el archivo de configuración son debido a una mala sintaxis en el cambio que recientemente se introdujo. Por favor revisa que todo el cambio se acabas de añadir se encuentra en una sola línea. Revisa que SÍ exista un espacio entre la ruta con el rootfs y la dirección IP. Revisa también que NO exista un espacio entre la dirección IP y el paréntesis de apertura con los parámetros.
9. Si todo sale bien, después de unos momentos el sistema terminará de arrancar y se la consola preguntará por un nombre de usuario y contraseña. Normalmente funciona root:root.
10. Una vez pasando la autenticación, prueba explorando el rootfs. Este debería tener el mismo contenido que la carpeta en la máquina virtual. Trata creando directorios y archivos en ambos lados (VM y BBB) y observa como los cambios son visibles al momento en ambos lados de la red local.