README.i2400m.txt

Driver for the Intel Wireless Wimax Connection 2400m

This provides a driver for the Intel Wireless WiMAX Connection 2400m
and a basic Linux kernel WiMAX stack.

1. Requirements

* Linux installation with Linux kernel 2.6.22 or newer (if building from a separate tree)
* Intel i2400m Echo Peak or Baxter Peak; this includes the Intel Wireless WiMAX/WiFi Link 5x50 series.
* build tools:
    + Linux kernel development package for the target kernel; to build against your currently running kernel, you need to have the kernel development package corresponding to the running image installed (usually if your kernel is named linux-VERSION, the development package is called linux-dev-VERSION or linux-headers-VERSION).
    + GNU C Compiler, make

2. Compilation and installation

2.1. Compilation of the drivers included in the kernel

Configure the kernel; to enable the WiMAX drivers select Drivers >
Networking Drivers > WiMAX device support. Enable all of them as
modules (easier).

If USB or SDIO are not enabled in the kernel configuration, the options
to build the i2400m USB or SDIO drivers will not show. Enable said
subsystems and go back to the WiMAX menu to enable the drivers.

Compile and install your kernel as usual.

2.2. Compilation of the drivers distributed as an standalone module

To compile

```
$ cd source/directory
$ make
```

Once built you can load and unload using the provided load.sh script;
load.sh will load the modules, load.sh u will unload them.

To install in the default kernel directories (and enable auto loading
when the device is plugged):

```
$ make install
$ depmod -a
```

If your kernel development files are located in a non standard
directory or if you want to build for a kernel that is not the
currently running one, set KDIR to the right location:

```
$ make KDIR=/path/to/kernel/dev/tree
```

For more information, please contact linux-wimax@intel.com.

3. Installing the firmware

The firmware can be obtained from http://linuxwimax.org or might have been supplied with your hardware.

It has to be installed in the target system:

```
      *
$ cp FIRMWAREFILE.sbcf /lib/firmware/i2400m-fw-BUSTYPE-1.3.sbcf
```

* NOTE: if your firmware came in an .rpm or .deb file, just install it as normal, with the rpm (rpm -i FIRMWARE.rpm) or dpkg (dpkg -i FIRMWARE.deb) commands. No further action is needed.
* BUSTYPE will be usb or sdio, depending on the hardware you have. Each hardware type comes with its own firmware and will not work with other types.

4. Design

This package contains two major parts: a WiMAX kernel stack and a driver for the Intel i2400m.

The WiMAX stack is designed to provide for common WiMAX control services to current and future WiMAX devices from any vendor; please see README.wimax for details.

The i2400m kernel driver is broken up in two main parts: the bus generic driver and the bus-specific drivers. The bus generic driver forms the drivercore and contain no knowledge of the actual method we use to connect to the device. The bus specific drivers are just the glue to connect the bus-generic driver and the device. Currently only USB and SDIO are supported. See drivers/net/wimax/i2400m/i2400m.h for more information.

The bus generic driver is logically broken up in two parts: OS-glue and hardware-glue. The OS-glue interfaces with Linux. The hardware-glue interfaces with the device on using an interface provided by the bus-specific driver. The reason for this breakup is to be able to easily reuse the hardware-glue to write drivers for other OSes; note the hardware glue part is written as a native Linux driver; no abstraction layers are used, so to port to another OS, the Linux kernel API calls should be replaced with the target OS's.

5. Usage

To load the driver, follow the instructions in the install section; once the driver is loaded, plug in the device (unless it is permanently plugged in). The driver will enumerate the device, upload the firmware and output messages in the kernel log (dmesg, /var/log/messages or /var/log/kern.log) such as:

```
...
```

```
i2400m_usb 5-4:1.0: firmware interface version 8.0.0
i2400m_usb 5-4:1.0: WiMAX interface wmx0 (00:1d:e1:01:94:2c) ready
```

At this point the device is ready to work.

Current versions require the Intel WiMAX Network Service in userspace to make things work. See the network service's README for instructions on how to scan, connect and disconnect.

5.1. Module parameters

Module parameters can be set at kernel or module load time or by echoing values:

```
$ echo VALUE > /sys/module/MODULENAME/parameters/PARAMETERNAME
```

To make changes permanent, for example, for the i2400m module, you can also create a file named /etc/modprobe.d/i2400m containing:

```
options i2400m idle_mode_disabled=1
```

To find which parameters are supported by a module, run:

```
$ modinfo path/to/module.ko
```

During kernel bootup (if the driver is linked in the kernel), specify the following to the kernel command line:

```
i2400m.PARAMETER=VALUE
```

5.1.1. i2400m: idle_mode_disabled

The i2400m module supports a parameter to disable idle mode. This parameter, once set, will take effect only when the device is reinitialized by the driver (eg: following a reset or a reconnect).

5.2. Debug operations: debugfs entries

The driver will register debugfs entries that allow the user to tweak debug settings. There are three main container directories where entries are placed, which correspond to the three blocks a i2400m WiMAX driver has:

* /sys/kernel/debug/wimax:DEVNAME/ for the generic WiMAX stack controls
* /sys/kernel/debug/wimax:DEVNAME/i2400m for the i2400m generic driver controls
* /sys/kernel/debug/wimax:DEVNAME/i2400m-usb (or -sdio) for the bus-specific i2400m-usb or i2400m-sdio controls).

Of course, if debugfs is mounted in a directory other than /sys/kernel/debug, those paths will change.

5.2.1. Increasing debug output

The files named *dl_* indicate knobs for controlling the debug output of different submodules:

```
     *
# find /sys/kernel/debug/wimax\:wmx0 -name \*dl_\*
/sys/kernel/debug/wimax:wmx0/i2400m-usb/dl_tx
/sys/kernel/debug/wimax:wmx0/i2400m-usb/dl_rx
/sys/kernel/debug/wimax:wmx0/i2400m-usb/dl_notif
/sys/kernel/debug/wimax:wmx0/i2400m-usb/dl_fw
/sys/kernel/debug/wimax:wmx0/i2400m-usb/dl_usb
/sys/kernel/debug/wimax:wmx0/i2400m/dl_tx
/sys/kernel/debug/wimax:wmx0/i2400m/dl_rx
/sys/kernel/debug/wimax:wmx0/i2400m/dl_rfkill
/sys/kernel/debug/wimax:wmx0/i2400m/dl_netdev
/sys/kernel/debug/wimax:wmx0/i2400m/dl_fw
/sys/kernel/debug/wimax:wmx0/i2400m/dl_debugfs
/sys/kernel/debug/wimax:wmx0/i2400m/dl_driver
/sys/kernel/debug/wimax:wmx0/i2400m/dl_control
/sys/kernel/debug/wimax:wmx0/wimax_dl_stack
/sys/kernel/debug/wimax:wmx0/wimax_dl_op_rfkill
/sys/kernel/debug/wimax:wmx0/wimax_dl_op_reset
/sys/kernel/debug/wimax:wmx0/wimax_dl_op_msg
/sys/kernel/debug/wimax:wmx0/wimax_dl_id_table
/sys/kernel/debug/wimax:wmx0/wimax_dl_debugfs
```

By reading the file you can obtain the current value of said debug level; by writing to it, you can set it.

To increase the debug level of, for example, the i2400m's generic TX engine, just write:

```
$ echo 3 > /sys/kernel/debug/wimax:wmx0/i2400m/dl_tx
```

Increasing numbers yield increasing debug information; for details of what is printed and the available levels, check the source. The code uses 0 for disabled and increasing values until 8.

5.2.2. RX and TX statistics

The i2400m/rx_stats and i2400m/tx_stats provide statistics about the data reception/delivery from the device:

```
$ cat /sys/kernel/debug/wimax:wmx0/i2400m/rx_stats
45 1 3 34 3104 48 480
```

The numbers reported are
- packets/RX-buffer: total, min, max
- RX-buffers: total RX buffers received, accumulated RX buffer size in bytes, min size received, max size received

Thus, to find the average buffer size received, divide accumulated RX-buffer / total RX-buffers.

To clear the statistics back to 0, write anything to the rx_stats file:

```
$ echo 1 > /sys/kernel/debug/wimax:wmx0/i2400m_rx_stats
```

Likewise for TX.

Note the packets this debug file refers to are not network packet, but packets in the sense of the device-specific protocol for communication to the host. See drivers/net/wimax/i2400m/tx.c.

5.2.3. Tracing messages received from user space

To echo messages received from user space into the trace pipe that the i2400m driver creates, set the debug file i2400m/trace_msg_from_user to 1:

*

```
$ echo 1 > /sys/kernel/debug/wimax:wmx0/i2400m/trace_msg_from_user
```

5.2.4. Performing a device reset

By writing a 0, a 1 or a 2 to the file /sys/kernel/debug/wimax:wmx0/reset, the driver performs a warm (without disconnecting from the bus), cold (disconnecting from the bus) or bus (bus specific) reset on the device.

5.2.5. Asking the device to enter power saving mode

By writing any value to the /sys/kernel/debug/wimax:wmx0 file, the device will attempt to enter power saving mode.

6. Troubleshooting

6.1. Driver complains about 'i2400m-fw-usb-1.2.sbcf: request failed'

If upon connecting the device, the following is output in the kernel log:

```
i2400m_usb 5-4:1.0: fw i2400m-fw-usb-1.3.sbcf: request failed: -2
```

This means that the driver cannot locate the firmware file named /lib/firmware/i2400m-fw-usb-1.2.sbcf. Check that the file is present in the right location.