Linux kernel WiMAX stack

This provides a basic Linux kernel WiMAX stack to provide a common control API for WiMAX devices, usable from kernel and user space.

1. Design

The WiMAX stack is designed to provide for common WiMAX control services to current and future WiMAX devices from any vendor.

Because currently there is only one and we don't know what would be the common services, the APIs it currently provides are very minimal. However, it is done in such a way that it is easily extensible to accommodate future requirements.

The stack works by embedding a struct wimax_dev in your device's control structures. This provides a set of callbacks that the WiMAX stack will call in order to implement control operations requested by the user. As well, the stack provides API functions that the driver calls to notify about changes of state in the device.

The stack exports the API calls needed to control the device to user space using generic netlink as a marshalling mechanism. You can access them using your own code or use the wrappers provided for your convenience in libwimax (in the wimax-tools package).

For detailed information on the stack, please see include/linux/wimax.h.

2. Usage

For usage in a driver (registration, API, etc) please refer to the instructions in the header file include/linux/wimax.h.

When a device is registered with the WiMAX stack, a set of debugfs files will appear in /sys/kernel/debug/wimax:wmxX can tweak for control.

2.1. Obtaining debug information: debugfs entries

The WiMAX stack is compiled, by default, with debug messages that can be used to diagnose issues. By default, said messages are disabled.

The drivers will register debugfs entries that allow the user to tweak debug settings.

Each driver, when registering with the stack, will cause a debugfs directory named wimax:DEVICENAME to be created; optionally, it might create more subentries below it.

2.1.1. Increasing debug output

The files named *dl_* indicate knobs for controlling the debug output

of different submodules of the WiMAX stack:

*

```
# find /sys/kernel/debug/wimax\:wmx0 -name \*dl_\*
/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
/sys/kernel/debug/wimax:wmx0/.... # other driver specific files
```

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

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 id-table submodule, just write:

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

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.