ALSA device control service in user space

Takashi SAKAMOTO (o-takashi@sakamocchi.jp)

October 30, 2016

Introduction

- User space applications can:
  - communicate to devices, regardless of ALSA.
  - add control element sets to ALSA sound cards.
  - receive notifications of events to operate control elements.
- We can put control service in user space.
  - At least, for devices on USB and units on IEEE 1394 bus.

Advantages and disadvantages of this idea?

Communication from user space to devices on USB

- /dev/bus/usb/*/*
- I've never program to communicate to the devices.

Communication from user space to units on IEEE 1394 bus

- /dev/fw*
- Several ioctl(2) commands are supported by firewire-core to handle units.
- Asynchronous events can be read(2) from descriptor.

Design of ALSA control interface

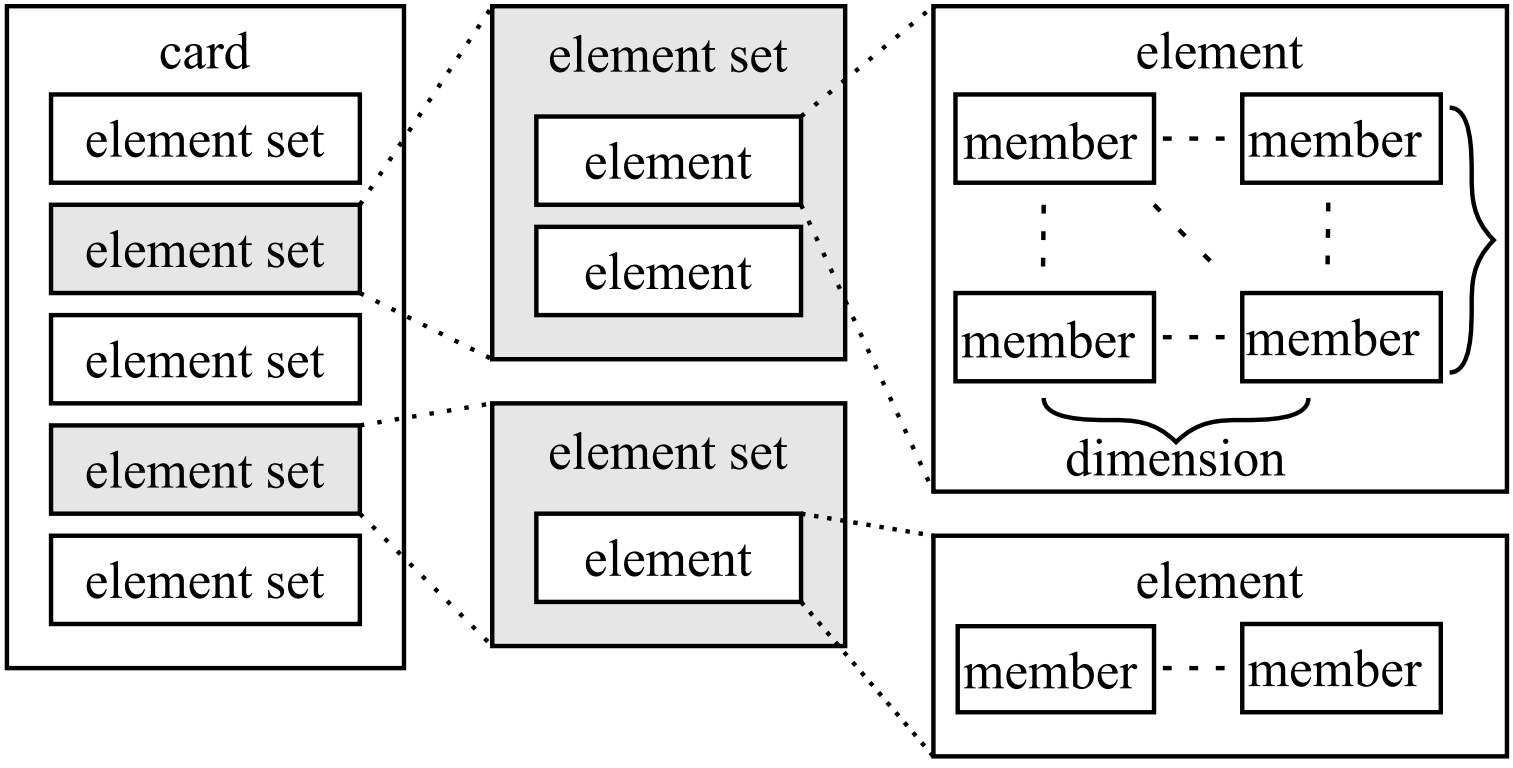

Figure 1: The relationship between card/element set/element/members

Element operations and event notification in user space

- ioctl(2) with several commands to operate elements
  - enumerate/information
  - read/write
  - lock/unlock
- read(2) to receive corresponding notifications
  - The same events to the same element are distilled to one event till being read.
  - Write operation generates `value` event.

Element set operations

- ioctl(2) with some commands to operate element sets
  - add
  - replace
  - remove
- Additional/removal of an element set generates `add`/`remove` events for included elements.
- This feature has been mainly used for PCM softvol plugin in alsa-lib
  - This feature appeared in alsa-driver-0.9.8 (2003/10/21)
  - It has been long-abandoned and fixed in Linux 4.1.

Type/Length/Value data (TLV) for an element set

- An element set can have arbitrary data in a shape of Type/Length/Value.
- This is mainly used to represent relationships between value and actual effect.
  - Sound pressure level (dB).
  - Physical channel mapping of data channel in a PCM substream.
- Abused in ALSA SoC part.
  - da7218/nau8810/nau8825/wm5102/soc-intel-haswell

Overview of this service: card registration

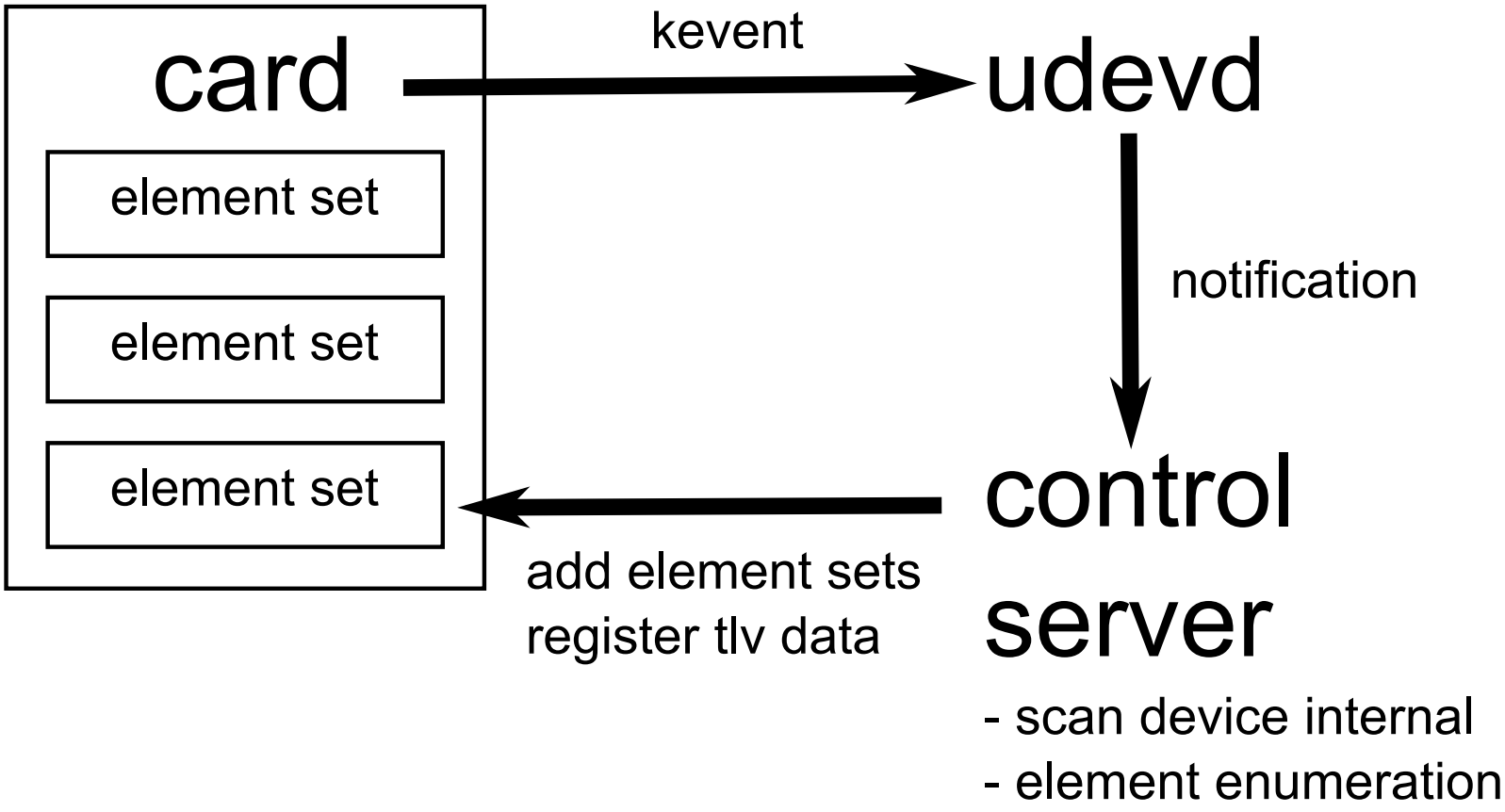

Figure 2: Processing of card registration

Overview of this service: element operation

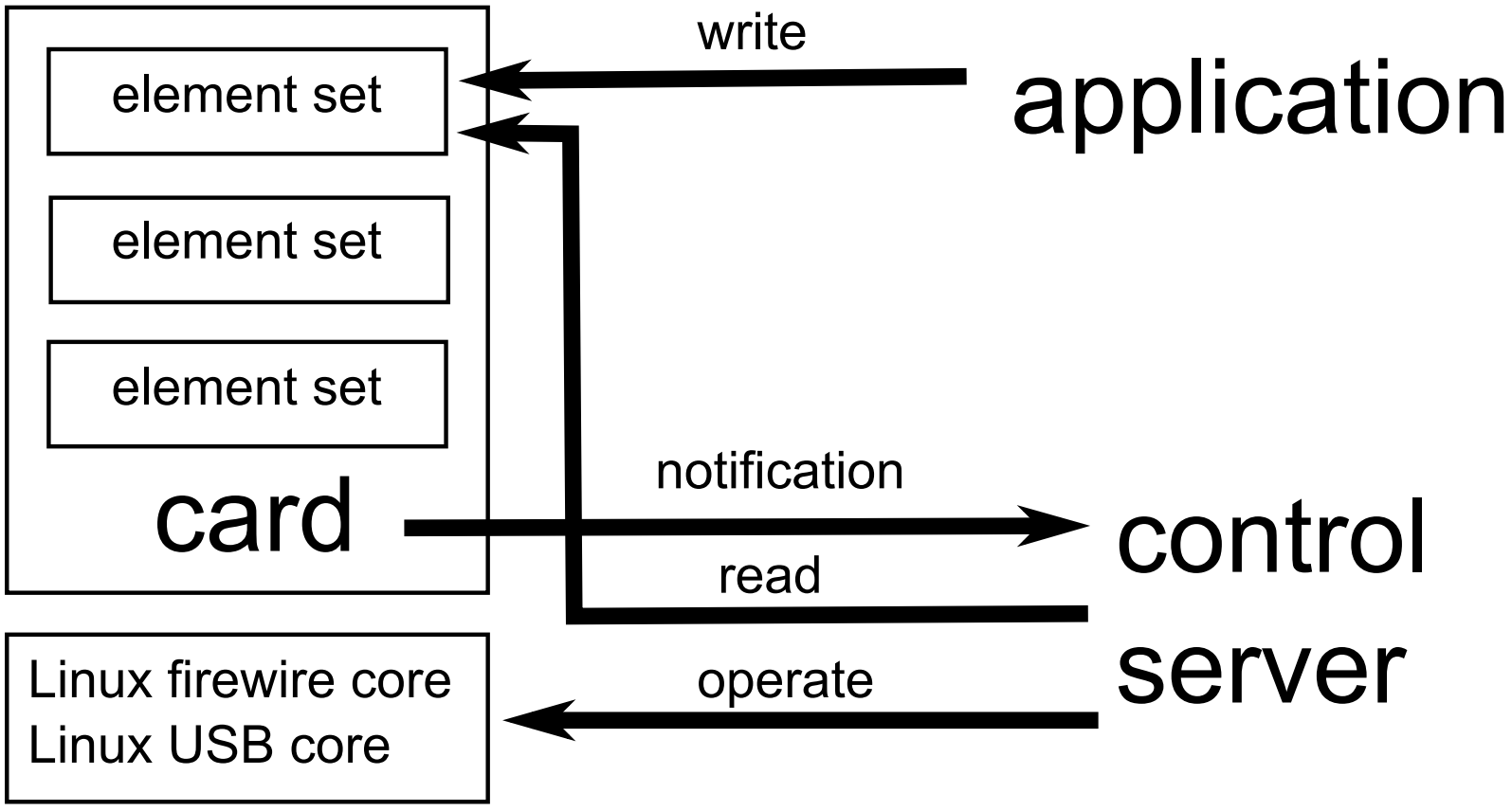

Figure 3: Processing of control element operation

Overview of this service: card disconnect

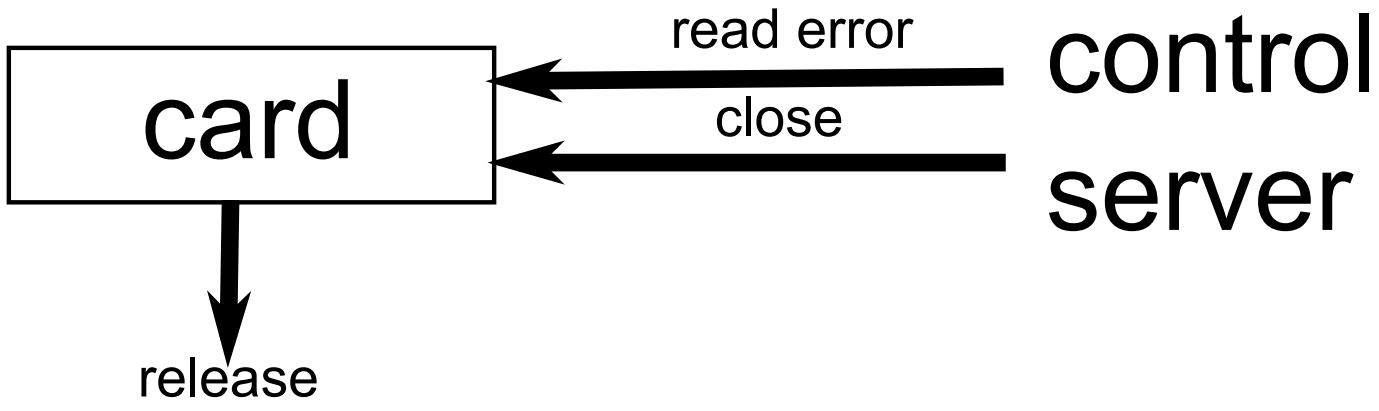

Figure 4: Processing of card disconnect

Advantages of this idea

1. Complexity reduction of in-kernel drivers
   - Vendor-dependent codes can be put into user space.
2. Independent development from kernel land
   - There might be more developers familiar with user space.

Disadvantages of this idea

1. No way to synchronize application processes to the server process.
   - The server cannot propagate communication failure to the application processes.
   - Operations of elements returns to application before actual effects.
2. Increase our cost to maintain system service for multiple devices.
   - It may not be an easy work.

Issues of this idea

1. How to represent device internal?
   - In most devices, we need to hard-code them.
   - As a common infrastructure, I expected topology framework, but it's just for in-kernel drivers
2. How to clean up alsa-lib for Mixer API?
   - Mixer API of alsa-lib includes much bugs and design defects.
   - I have no motivation because no documentation for its design.
3. How many devices can be supported by this idea?
   - If just for Audio and Music units on IEEE 1394 bus, I'm unwilling to work for it.