Advanced USB 3.0 for Linux

@sarahsharp

Libre Software Meeting 2010

Figure 5-19. USB 3.0 Icon
Agenda

- Background
- Changes in USB 3.0 communication
- Host scheduling
- USB Attached SCSI (UAS)
Background

- EHCI handles USB 2.0 devices
- EHCI lets a companion controller (OHCI or UHCI) handle USB 1.1 devices
Background

- xHCI handles all device speeds (USB 3.0, USB 2.0, and USB 1.1)
Changes in USB communication

USB 1.1

USB 2.0

USB 3.0
USB 2.0 communication

- Four basic endpoint types
  - asynchronous: bulk & control
  - periodic: interrupt & isochronous
- max packet size
- Single direction
  - Only host or device can talk
USB 3.0 communication

- Still four basic endpoint types
- Bi-directional wires added
  - Both host and device can talk
- Max packet size is the same
  - Devices.getHost can "burst" up to 16 packets
- Faster communication
Implications for device driver writers

- Submitting many small transfers doesn't take advantage of bursts
  - host controller can't group packets into bursts
- Larger transfers are better
- Submitting lists of buffers (scatter gather lists) is best
  - less processing, less interrupts
USB scheduling
USB scheduling

- Scheduling when packets go out over USB wires
USB 1.1 & 2.0 scheduling

- For older host controllers, software (OS) does the packet scheduling
  - EHCI, UHCI, OHCI
USB 3.0 scheduling

- For newer (xHCI) host controllers, the hardware does the packet scheduling
USB scheduling

- Interrupt and isochronous endpoints are polled
- Device specifies minimum polling interval
  - e.g. 1ms or 64ms
  - may specify an alternative interval in an "alt setting"

<table>
<thead>
<tr>
<th>Frame</th>
<th>1ms interval</th>
<th>2ms interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>199</td>
<td>packet</td>
<td>packet</td>
</tr>
<tr>
<td>200</td>
<td>packet</td>
<td>packet</td>
</tr>
<tr>
<td>201</td>
<td>packet</td>
<td>packet</td>
</tr>
<tr>
<td>202</td>
<td></td>
<td>packet</td>
</tr>
</tbody>
</table>
USB scheduling example

You have the schedule below, and want to add a new USB device. Choices:

- alt setting 0: polled at 2ms
- alt setting 1: polled at 1ms
USB scheduling example

- You have the schedule below, and want to add a new USB device. Choices:
  - alt setting 0: polled at 2ms
  - alt setting 1: polled at 1ms
- Result: alt setting 1 won't fit, choose alt setting 0
Current way of scheduling

- Drivers pick an alternate setting
- Drivers submit a transfer with a polling interval
  - if that fails, they pick a different alt setting
  - works for EHCI, UHCI, OHCI where schedule is in software
New way of scheduling

- But schedule in xHCI hardware is set up sooner
  - picking an alternate setting may fail
  - drivers have to be able to handle this
More changes in USB scheduling

- USB time is broken up into frames
  - 1ms interval
- Drivers specify the frame to start the transfer in
  - urb->start_frame
- or they can send "ASAP"
More changes in USB scheduling

- Drivers can specify the start_frame for every transfer
- EHCI, OHCI, and UHCI will honor that

<table>
<thead>
<tr>
<th>Frame 199</th>
<th>Frame 200</th>
<th>Frame 201</th>
<th>Frame 202</th>
<th>Frame 203</th>
<th>Frame 204</th>
</tr>
</thead>
<tbody>
<tr>
<td>packet</td>
<td>packet</td>
<td>packet</td>
<td>packet</td>
<td>packet</td>
<td>packet</td>
</tr>
</tbody>
</table>

1ms interval, ASAP

2ms interval, start frame 200 then start frame 203
More changes in USB scheduling

- Drivers can specify the start_frame for every transfer
- EHCI, OHCI, and UHCI will honor that
- xHCI cannot.
Next Gen
USB 3.0 Mass Storage Devices
Old Mass storage devices

- Bulk only Transport (BoT)
  - only one outstanding SCSI command
  - slow for spinning disks

Diagram:

- Host PC
  - Setup SCSI command 1
  - Data for command 1
  - Setup SCSI command 2
- BoT device
New Mass storage devices

- USB Attached SCSI (UAS)
  - multiple outstanding SCSI commands
  - device can reorder commands, or transfer part of commands

Diagram:

- Host PC
  - Setup SCSI command 1
  - Setup SCSI command 2
  - some data for command 2
  - data for command 1
- BoT device
USB Attached SCSI (UAS)

- Problem: How do you tell which SCSI command is being transferred?
- Solution: Tag the command with a "stream ID"
Implications for USB drivers

- Other USB 3.0 devices may need stream IDs
- stream IDs can only be used with USB 3.0 devices
USB Attached SCSI (UAS)

- Work needed to support UAS:
  - Need new UAS driver
  - USB core and xHCI driver have to support stream IDs (added in 2.6.35)
Questions?