Linux Support for USB 3.0

Sarah Sharp

Linux Plumbers Conference

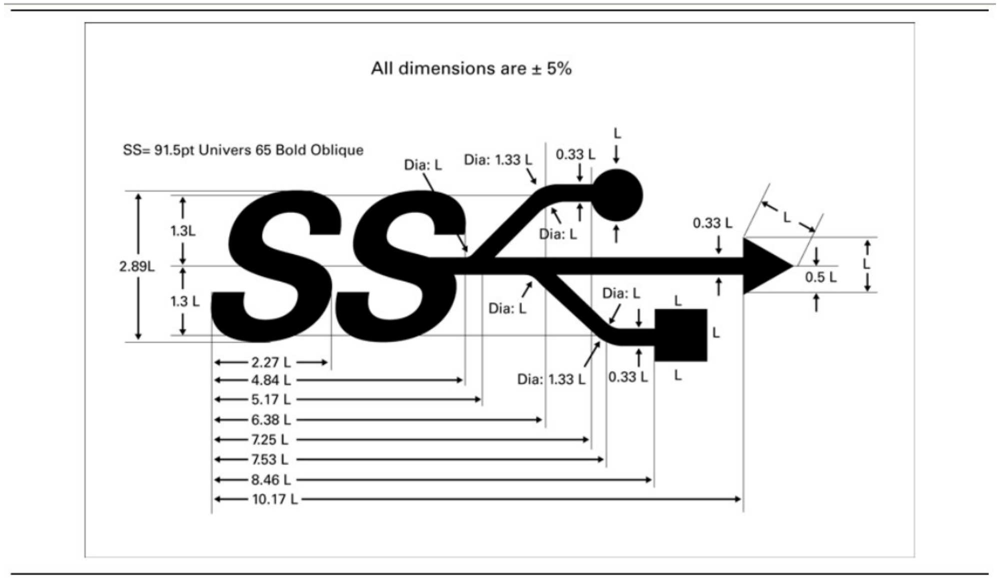


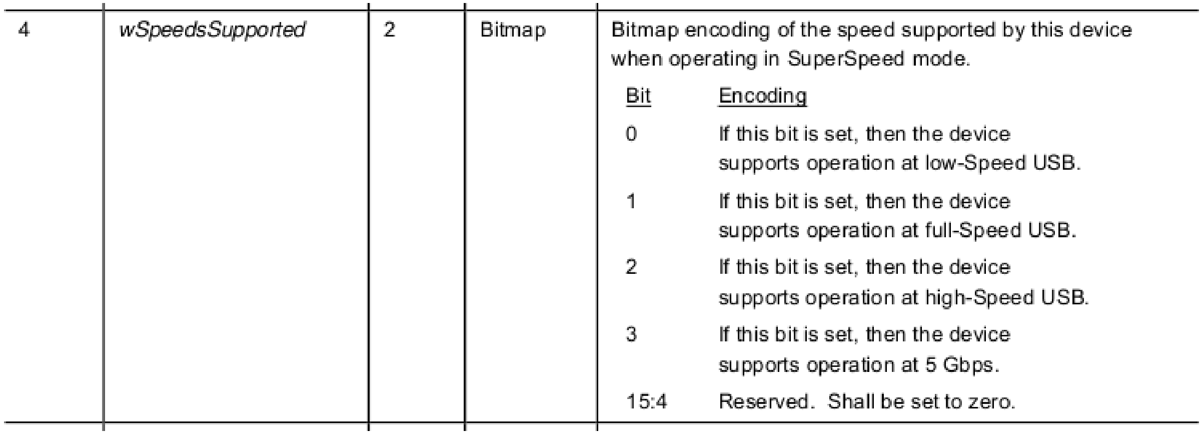
Figure 5-19. USB 3.0 Icon

Why USB 3.0?

- 480Mb/s is too slow
- USB 2.0 sucks power
  - Inefficient host controller design
  - Polling and broadcast messages
  - Many devices don't support auto-suspend
  - Start of frames (SOFs) sent with one active device

Why is USB 3.0 interesting?

- Backwards compatible
- Faster speed (5Gbps) with room to grow



4	*wSpeedsSupported*	2	Bitmap	Bitmap encoding of the speed supported by this device when operating in SuperSpeed mode. Bit Encoding 0 If this bit is set, then the device supports operation at low-Speed USB. 1 If this bit is set, then the device supports operation at full-Speed USB. 2 If this bit is set, then the device supports operation at high-Speed USB. 3 If this bit is set, then the device supports operation at 5 Gbps. 15:4 Reserved. Shall be set to zero.

- Bulk “streams” allow SCSI command queuing

Why is USB 3.0 interesting?

- Better power management
  - device notifications (no more polling)
  - unicast packets (not broadcast)
  - link power management
  - function power management
  - host controller schedule in HW, not system memory

USB 3.0 Implications

- 6 wires added for USB 3.0
- USB 2.0 devices use separate wires
  - Same PM/auto-suspend problems as before
- New host controller (xHCI), new host controller driver
  - scheduler in hardware, xHCI driver needs hooks for device changes

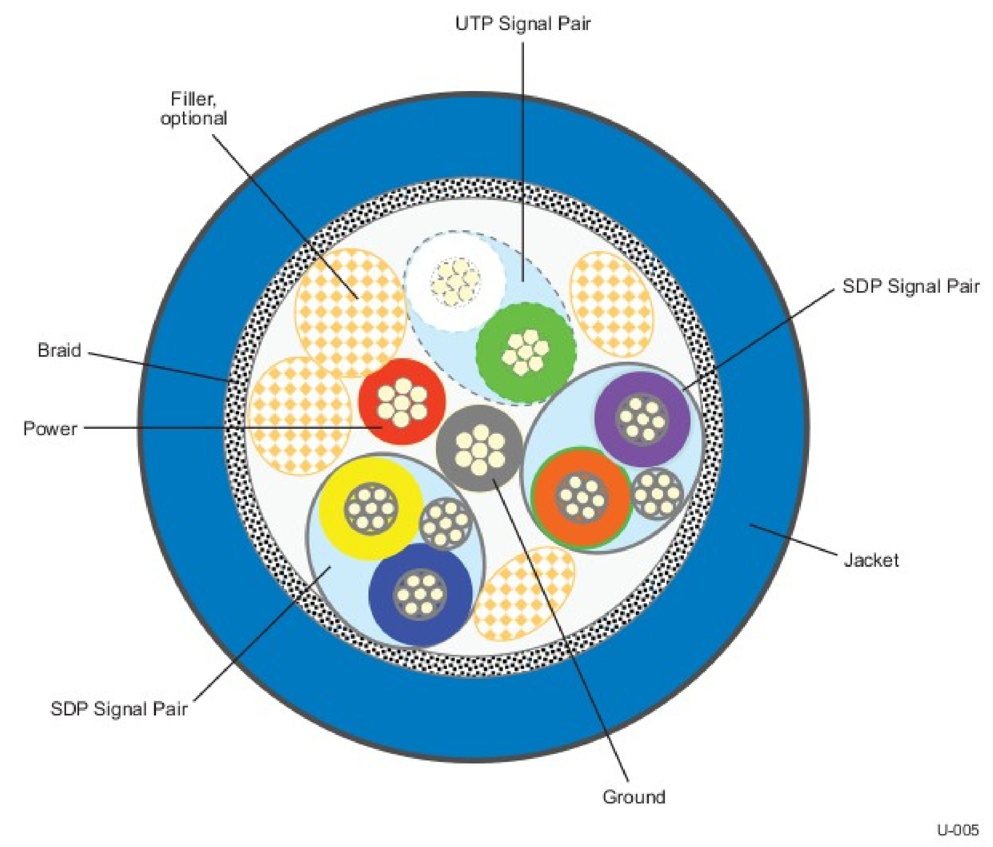


Figure 5-15. Illustration of a USB 3.0 Cable Cross-Section

USB 3.0 host-side cable (standard A)

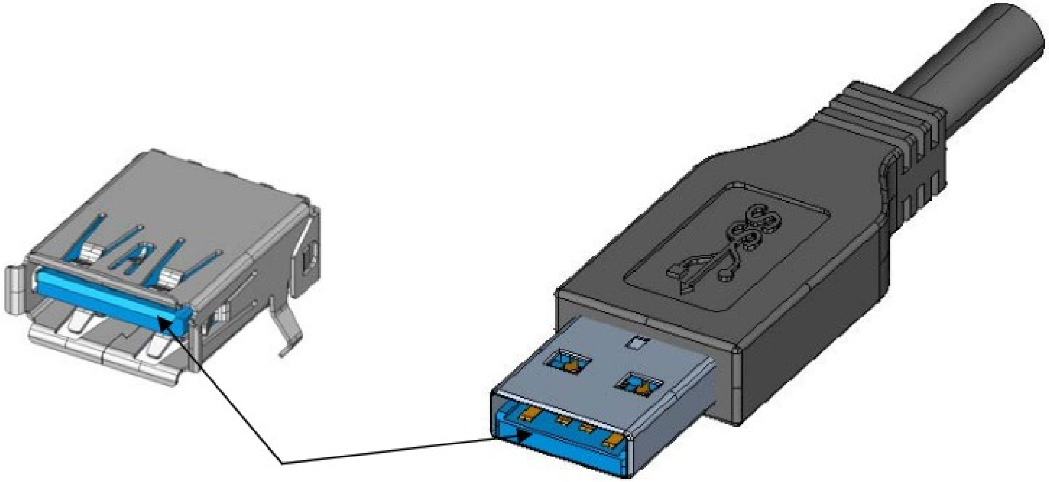
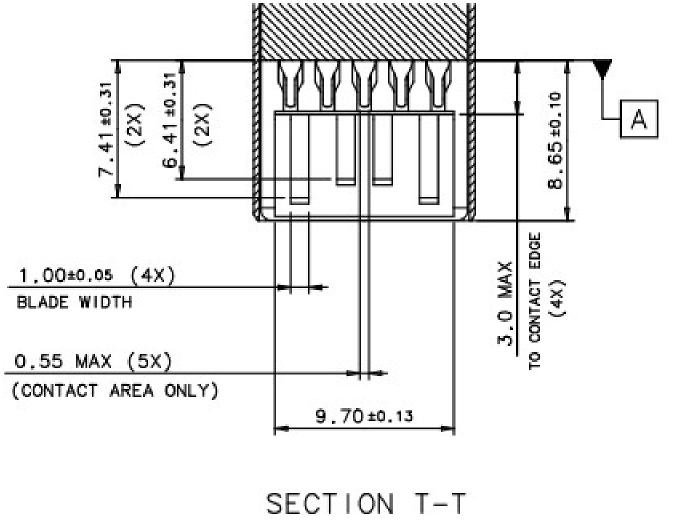
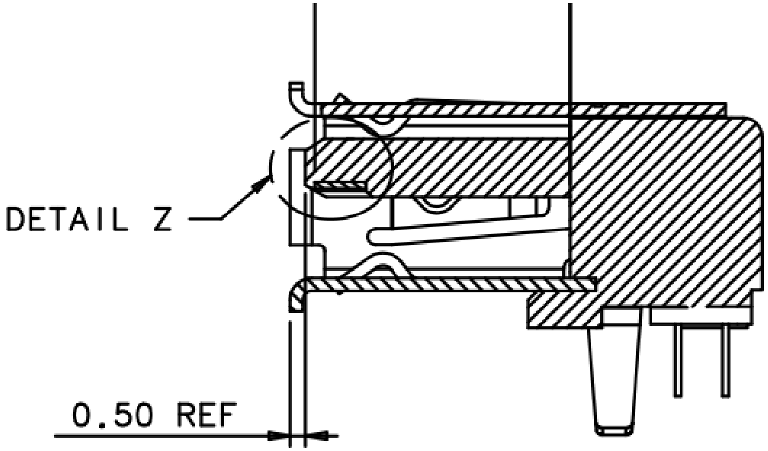
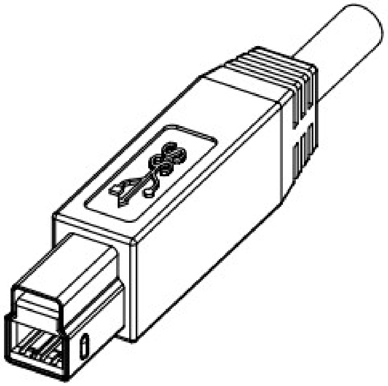


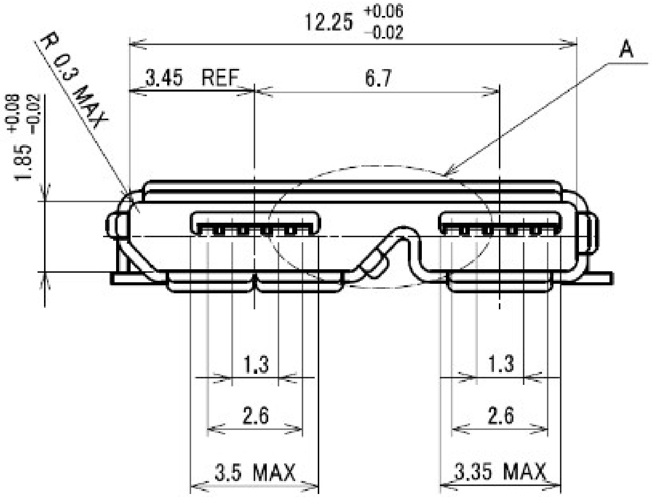
Figure 5-5. Illustration of Color Coding Recommendation for USB 3.0 Standa





USB 3.0 device side cable (standard-B and mini-B)





State of xHCI/USB 3.0 in Linux

- Supported in 2.6.31:
  - device enumeration
  - bulk and control TX
  - all device speeds (LS/FS/HS/SS)
  - stalls
  - cancellation
- Ready for 2.6.32:
  - interrupt TX
  - devices under 2.0 hubs
  - babbles

YES

SOON

MAYBE

SOON

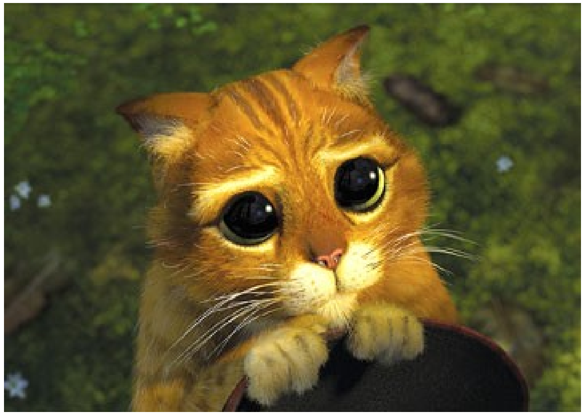
NOT YET

xHCI driver future changes

- setting alternate interfaces
- isochronous TX
- non-standard polling rates
- resetting devices
- little endian support
- USB 3.0 bulk streams
- USB autosuspend
- xHCI PCI device suspend
- virtualization

Kernel Changes separate from xHCI

- New USB device class drivers
- USB 3.0 hub support
- USB 3.0 Function PM
- USB 3.0 Link PM
- Can you help with these?

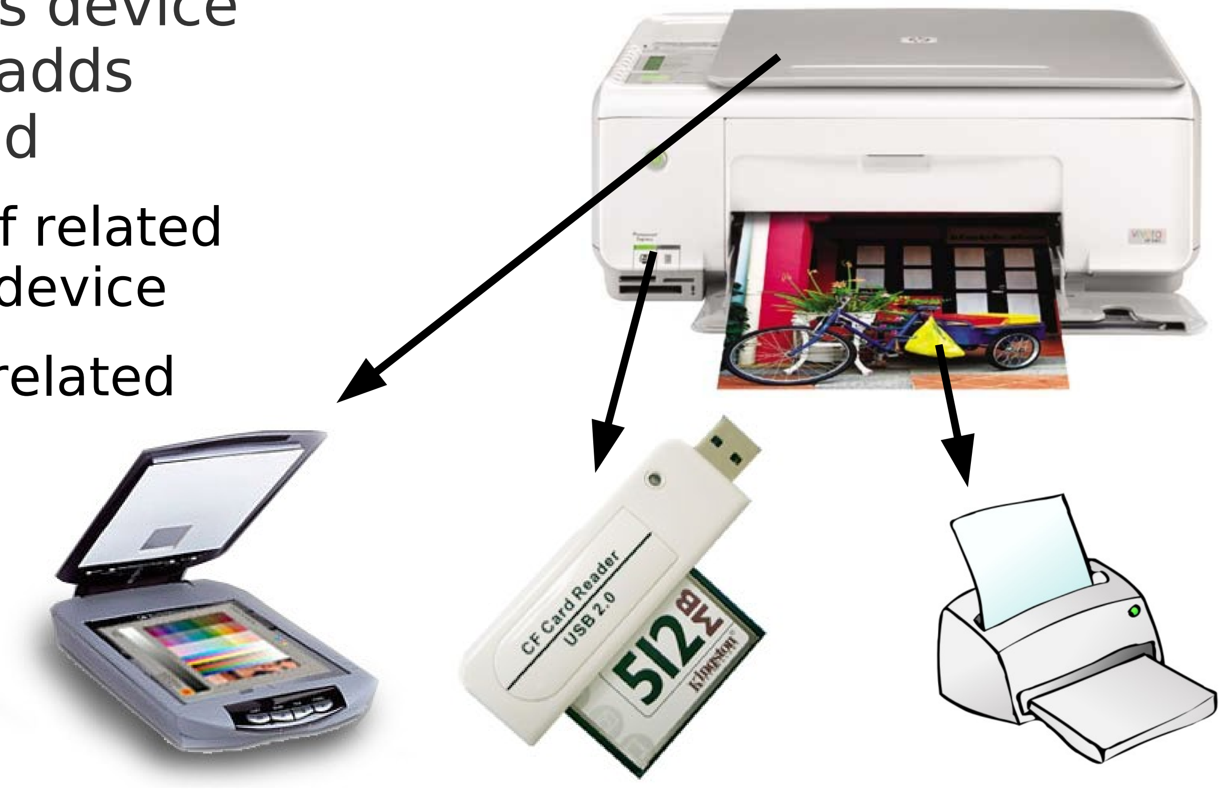


Current USB power management

- Automatically suspend the whole device
- Userspace must enable auto-suspend
- Drivers must support auto-suspend
- USB core keeps track of idleness
- Devices have to not break!

USB 3.0 function PM

- USB 2.0 has device suspend
  - suspend whole USB device
- USB 3.0 also has device suspend, but it adds function suspend
  - suspend a set of related interfaces on a device
  - use IAD to find related interfaces

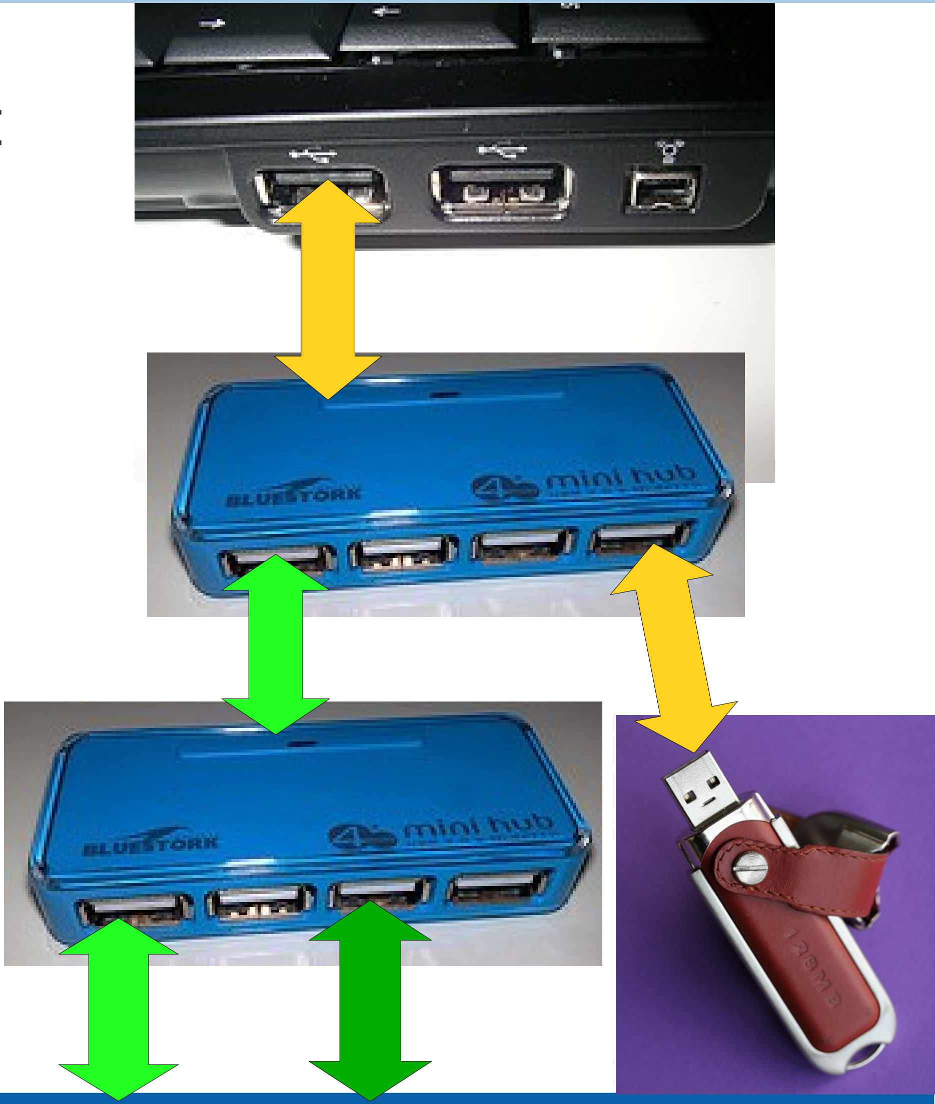


OS changes for USB 3.0 function PM

- USB core needs to handle function PM
- Track when an interface is claimed or busy
- Use Interface Association Descriptor (IAD)
- Send function suspend when interfaces are idle
- Handle Function Wake Device Notifications
- Putting all functions into suspend does not put the **device** into suspend; still need to send device suspend request

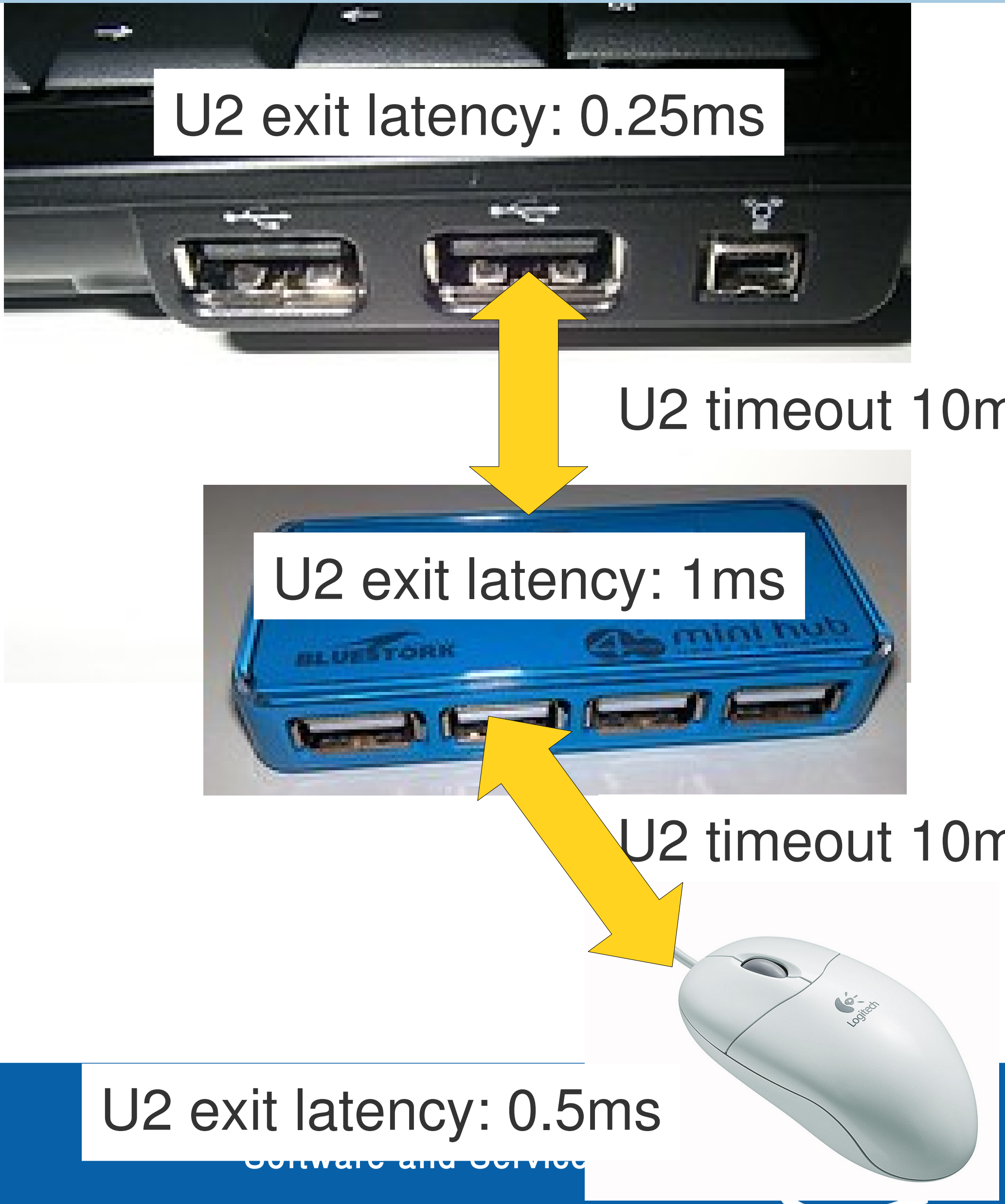
USB 3.0 Link PM

- USB 3.0 traffic is unicast
- Each idle link can be put into lower-power states (U0, U1, U2)
- Each link state has an exit latency
- Sort of like CPU C-states
- Each link partner can ask to go into a lower link state
- Highest link state is propagated up



OS changes for USB 3.0 Link PM

- Hardware does most of the work
- Software needs to set backup policy
- Need to set U1/U2 timeouts for each hub port
- Need some “wiggle room” in timeouts - maybe 5 to 10 times max exit latency?



U2 exit latency: 0.25ms

U2 timeout 10ms

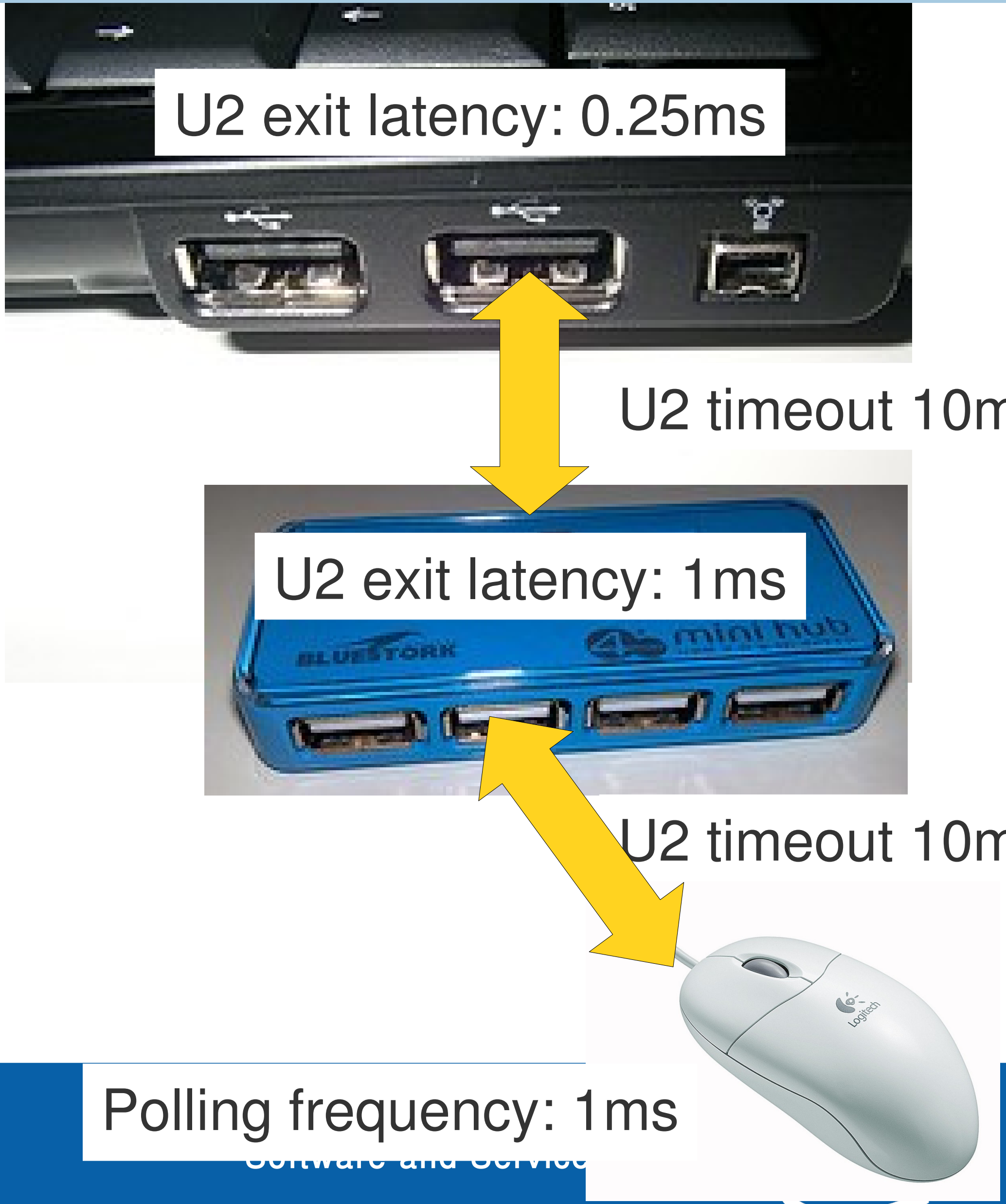
U2 exit latency: 1ms

U2 timeout 10ms

U2 exit latency: 0.5ms

OS changes for USB 3.0 Link PM

- Decide if it's worth it to enable U1/U2 for a device
- Is a periodic device too deep in the device tree?
- Are the hubs too slow?



U2 exit latency: 0.25ms

U2 timeout 10ms

U2 exit latency: 1ms

U2 timeout 10ms

Polling frequency: 1ms

OS changes for USB 3.0 Link PM

- Most of the work in USB core
- xHCI will trap roothub timeouts
- xHCI needs to set the maximum propagation delay for each device

USB 3.0 hubs

- Changes need to be made to khubd
  - new device descriptor
  - new class-specific requests
  - different port status bits
  - no transaction translators
  - hot reset vs. warm reset

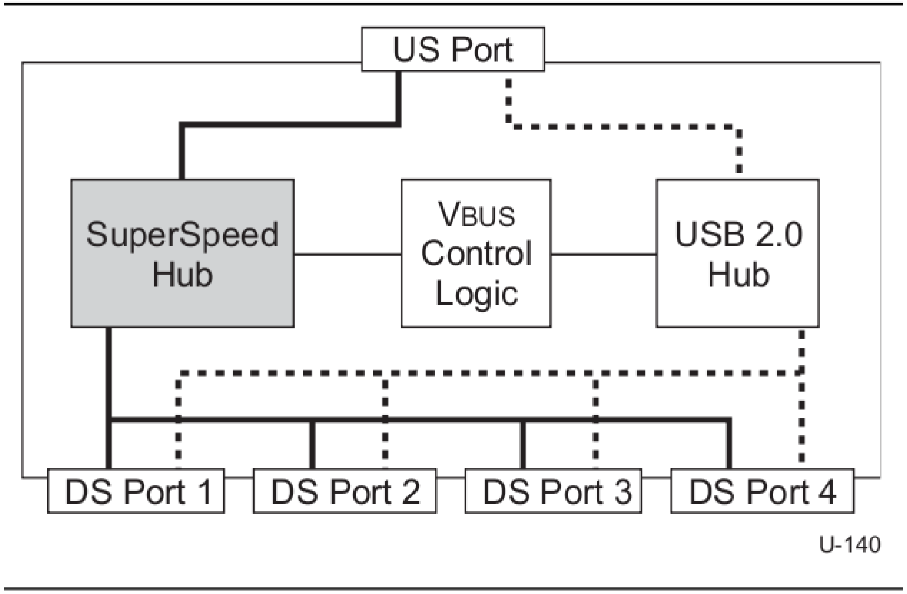


Figure 10-1. Hub Architecture

USB 3.0 Bulk “streams”

- Some USB 3.0 bulk endpoints support multiple “streams”
- Packets are tagged with a stream ID
- Device is notified when a stream has new data
- Device can start and stop any stream it wants to

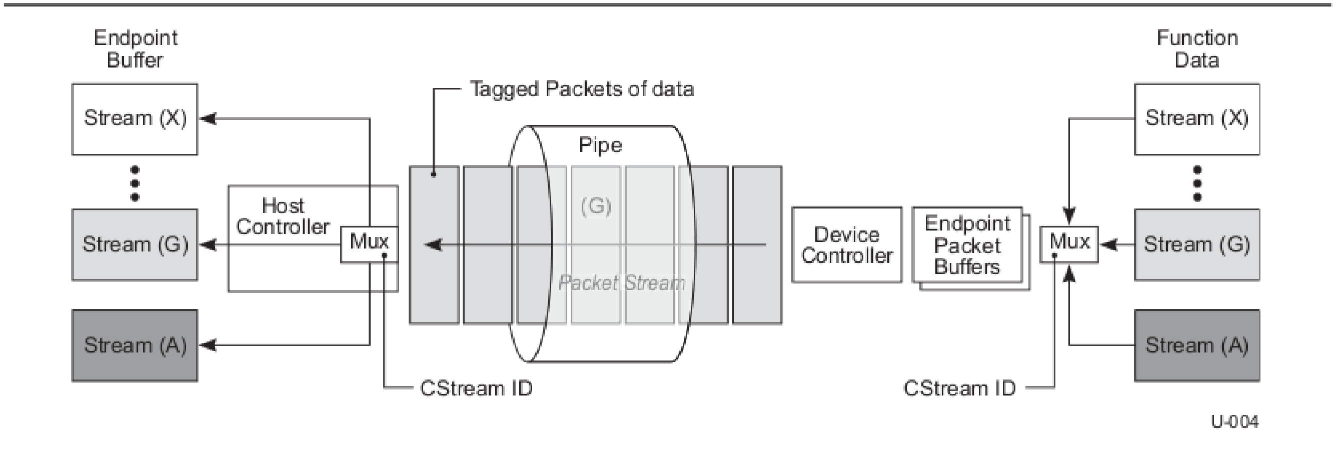


Figure 4-3. USB SuperSpeed IN Stream Example

USB 3.0 Bulk “streams”

- Allows each SCSI command to be tagged with a stream ID
- MSC device decides which command to start
- Spinning disks can sort commands
- Flash & SSDs can start prefetching sooner

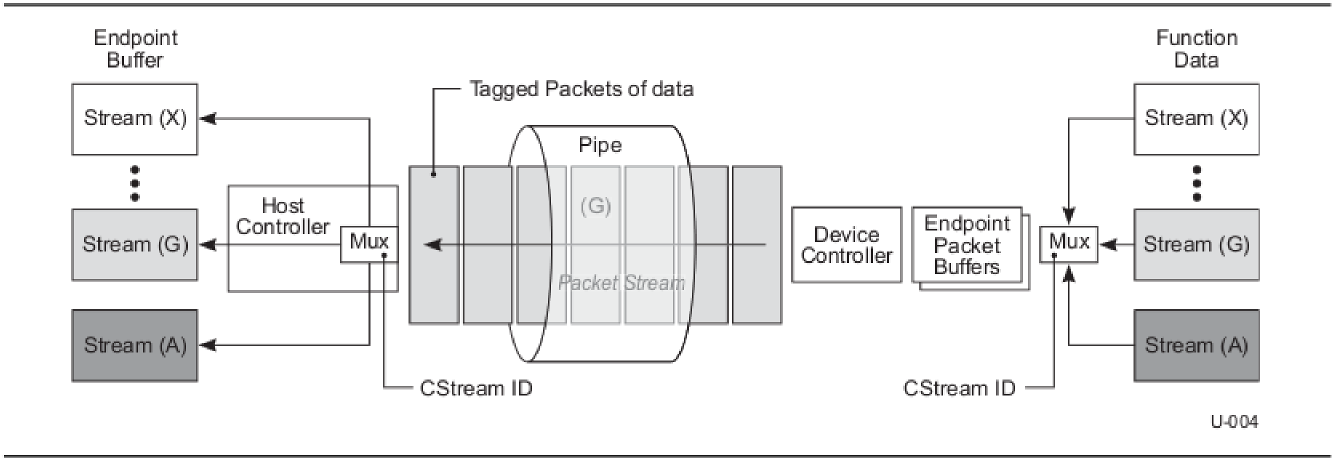


Figure 4-3. USB SuperSpeed IN Stream Example

USB 3.0 storage devices

- Some will be legacy (BOT)
- USB Attached SCSI Protocol (UASP)
- Can be a USB 2.0 or USB 3.0 device
- Uses USB 3.0 bulk streams to queue multiple SCSI commands to device
- New USB class driver
- xHCI needs to support bulk streams

USB 3.0 webcams

- Point Grey webcam announced at IDF
- uncompressed 1080p video
- Will V4L layer handle this?
- Some USB video drivers have assumptions based on speed
  - e.g. driver picks a different polling interval based on FS or HS

Kernel/Userspace Interface changes for USB 3.0

- usbfs and libusb need to become aware of USB 3.0 stream IDs.
- Is it fast enough? Do we need a scatter-gather interface?
- USBMon needs to understand scatter gather lists and stream IDs.

Userspace changes for USB 3.0

- New UASP class with SCSI command queuing should have little impact on userspace
- How will applications like cheese handle faster USB webcams?
- Is HAL ready for USB 3.0?

How can I help?

- Areas you can help in:
  - New USB device class drivers
  - Readying old class drivers for USB 3.0 devices
  - USB 3.0 hub support
  - USB 3.0 Link PM
  - USB 3.0 Function PM
- Patches and discussion on the Linux USB mailing list:
  - linux-usb@vger.kernel.org
  - http://www.linux-usb.org/mailing.html
- xHCI git tree on kernel.org

Questions?

Sarah Sharp
sarah.a.sharp@linux.intel.com
twitter: @sarahsharp

Creative Commons Attributions

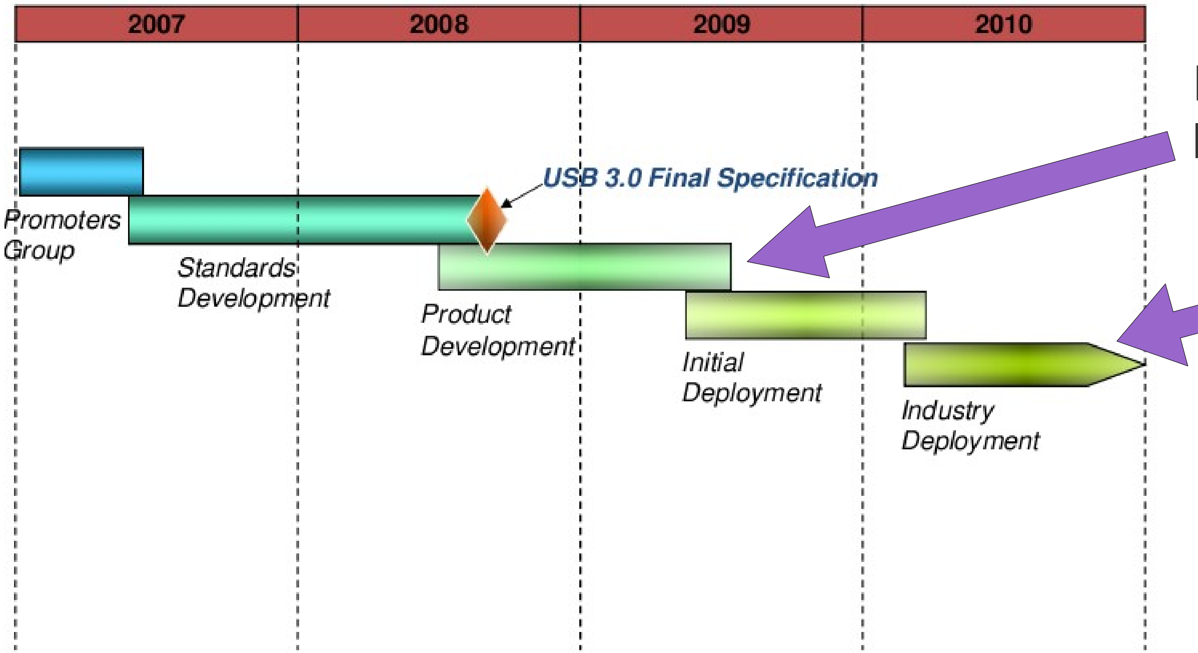
- light bulb http://www.flickr.com/photos/mr_beaver/3486761520/
- “W” thumb drive http://www.flickr.com/photos/ivanwalsh/3657331200/
- keyboard & mice http://www.flickr.com/photos/m0php/3862857014/
- webcam http://www.flickr.com/photos/mrtea/772346725/
- blue hub http://www.flickr.com/photos/jeanbaptistem/3486039048/
- work in progress http://www.flickr.com/photos/hellochris/2801931497/
- host ports http://www.flickr.com/photos/kikus/3732845777/
- purple thumb drive http://www.flickr.com/photos/caroslines/2046327031/
- USB to SATA http://www.flickr.com/photos/cavemonkey50/427366996/
- webcam under linux http://www.flickr.com/photos/phylevn/2948896990/
- plumbers nightmare http://www.flickr.com/photos/ejbsf/3413576188/
- printer http://www.flickr.com/photos/davesag/192584714/
- are we there yet? http://www.flickr.com/photos/caseya/372922053/
- logitech mouse http://www.flickr.com/photos/blogitech/2883630458/

Other photos

- USB 3.0 devices at IDF from engadget and reghardware

When will USB 3.0 devices appear?

- Jeff Ravencraft's (USB-IF Pres.) estimated timeline:



PCI add-in cards late 2009

Integrated solutions 2010

- NEC announced certified discrete host controller

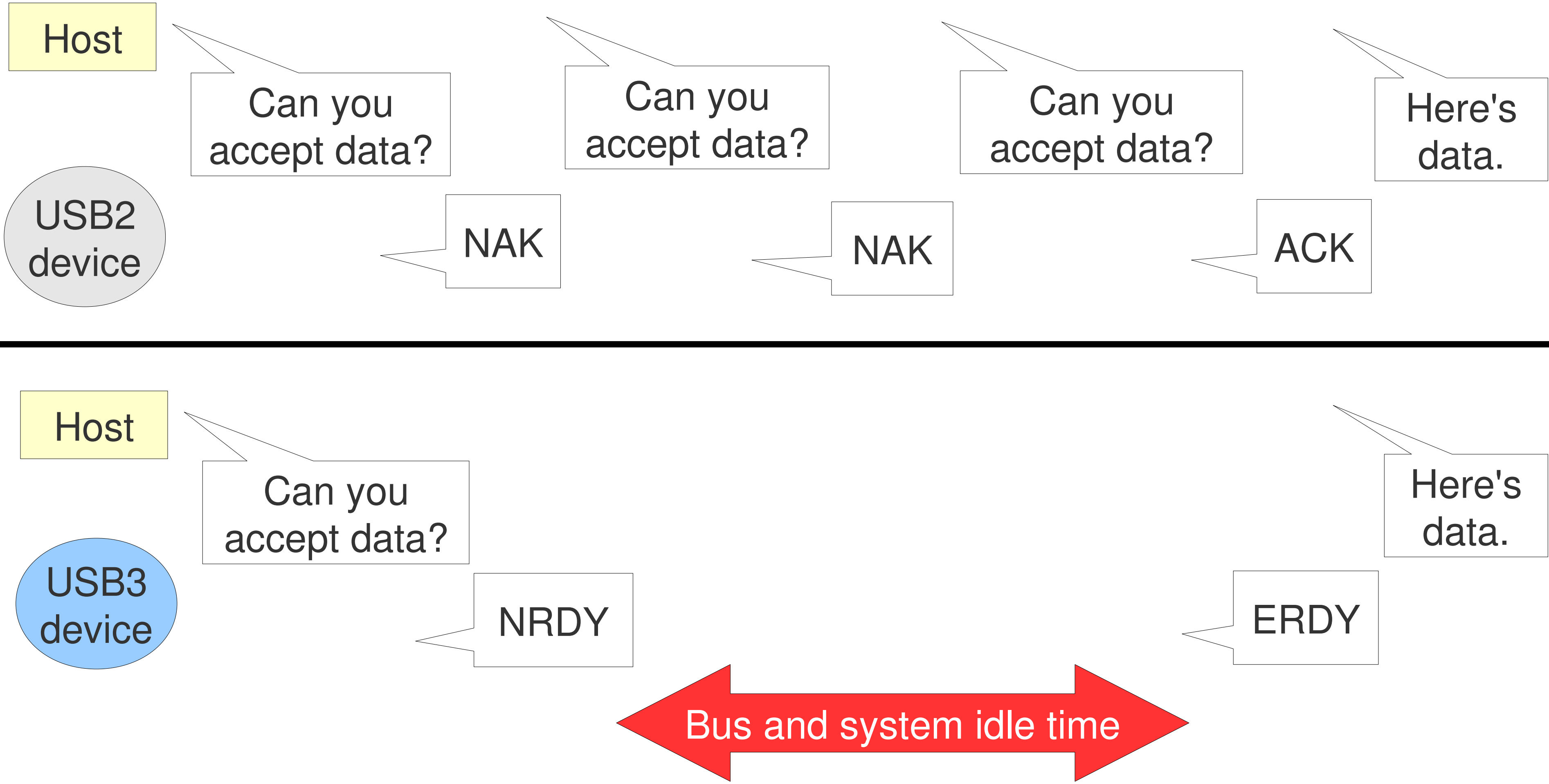
Upsides of USB 3.0: No more polling

- High, full, and low speed devices can NAK an OUT transfer if they aren't ready to process the data.
- Leads to a lot of bus activity.
- USB 3.0 devices can say they aren't ready for data yet (NRDY)
- When they are ready, they asynchronously notify the host (ERDY)

Can you print a page yet?

Are we there yet?

USB 2.0 polling vs. USB 3.0 NRDY/ERDY



Host

USB2 device

Can you accept data?

NAK

Can you accept data?

NAK

Can you accept data?

ACK

Here's data.

Host

USB3 device

Can you accept data?

NRDY

ERDY

Here's data.

Bus and system idle time

Implications of USB 3.0 NRDY/ERDY

- EHCI sets NAK count to 4
  - host controller gives up after 4 NAKs
  - max wait time of 4ms for FS/LS device response
- xHCI has no timeout on NRDY'ed transfers
  - Could be on the order of seconds?
- Implication: Userspace shouldn't block on USB transactions
  - X polling /dev/eventN for mouse movement - should be fine since it uses fnotify (and no one will make a USB3 mouse)
  - What about HAL polling?

USB 3.0 Link Power Management

- Routed packets means some bus links will be idle
- Two new link power management states
- Deeper power savings and higher exit latencies

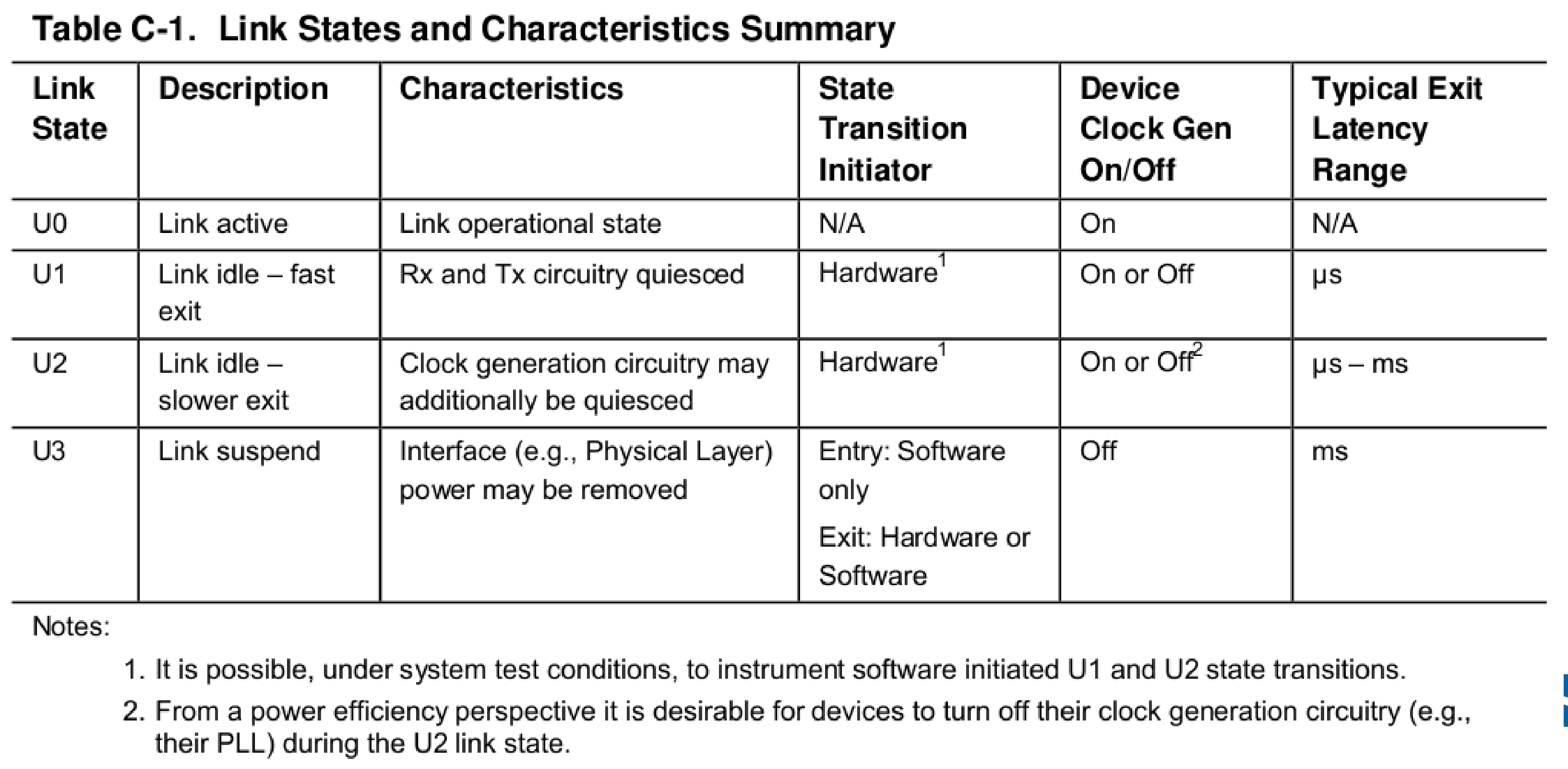


Table C-1. Link States and Characteristics Summary

Link State	Description	Characteristics	State Transition Initiator	Device Clock Gen On/Off	Typical Exit Latency Range
U0	Link active	Link operational state	N/A	On	N/A
U1	Link idle – fast exit	Rx and Tx circuitry quiesced	Hardware[1]	On or Off	µs
U2	Link idle – slower exit	Clock generation circuitry may additionally be quiesced	Hardware[1]	On or Off[2]	µs – ms
U3	Link suspend	Interface (e.g., Physical Layer) power may be removed	Entry: Software only Exit: Hardware or Software	Off	ms

Notes:

1. It is possible, under system test conditions, to instrument software initiated U1 and U2 state transitions.
2. From a power efficiency perspective it is desirable for devices to turn off their clock generation circuitry (e.g., their PLL) during the U2 link state.