USB Status report 2012

Gerd Hoffmann
Red Hat
KVM Forum, Nov 7th

Outline

- Some USB Basics.
- What is new / updated / improved in QEMU USB support?
- Future plans / TODO list.
- Using the new bits.

USB Basics: Endpoints

- Communicate with the host using endpoints
  - Each endpoint is a data pipe.
  - One control endpoint.
  - Up to 15 IN (device -> host) endpoints
  - Up to 15 OUT (host ->device) endpoints.
- Four Endpoint types
  - Control
  - Bulk (bulky data: usb sticks)
  - Isochronous (streaming data: usb speakers)
  - Interrupt (events: mouse)

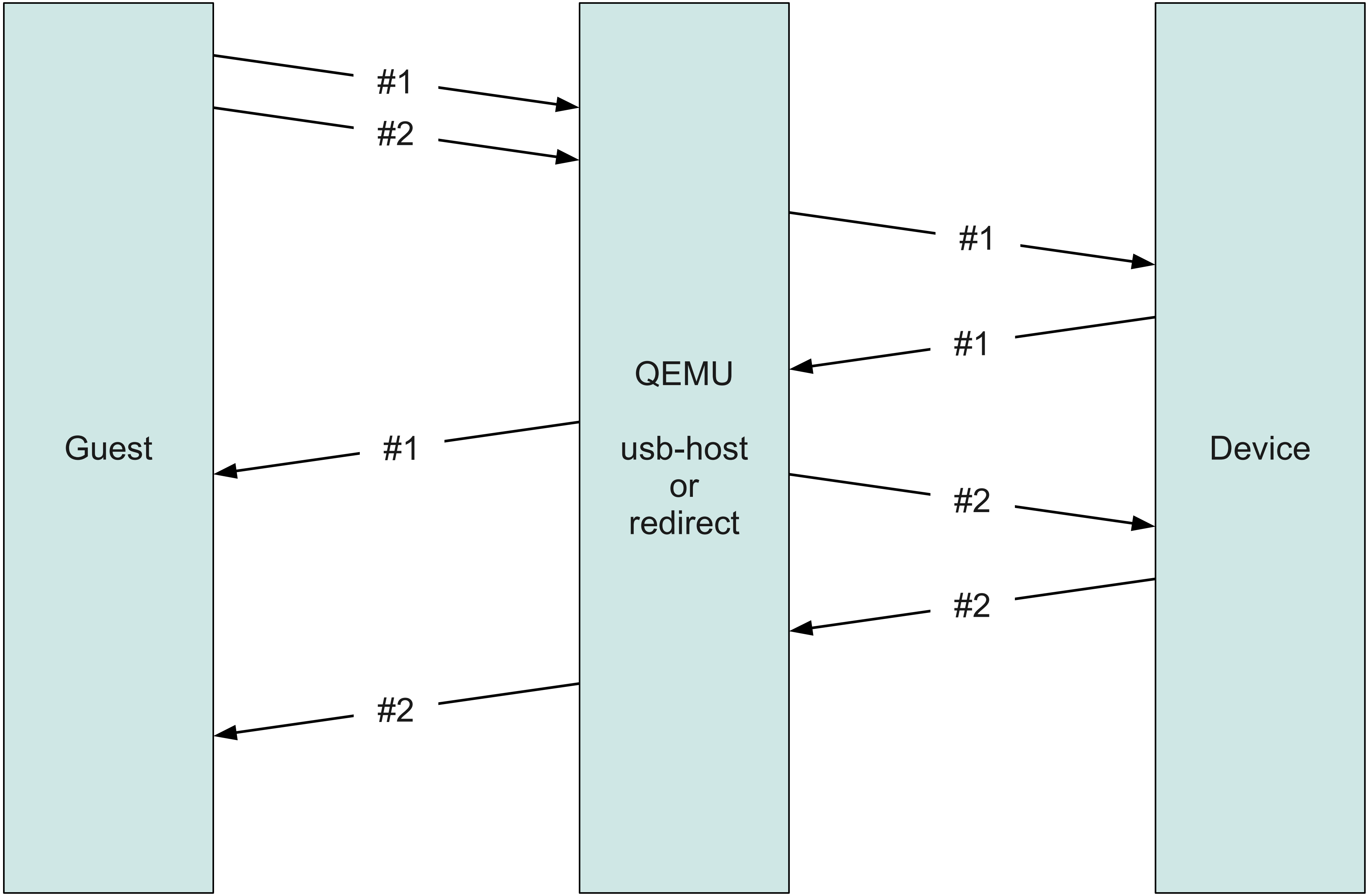
USB Basics: Functions

- Functional unit, OS typically has one driver per function.
- Each function has a set of endpoints.
- Multifunction examples:
  - Webcam with mic: one video, one audio.
  - Extra HID function for buttons.
  - Extra storage function with windows drivers.
- Most devices have a single function only.

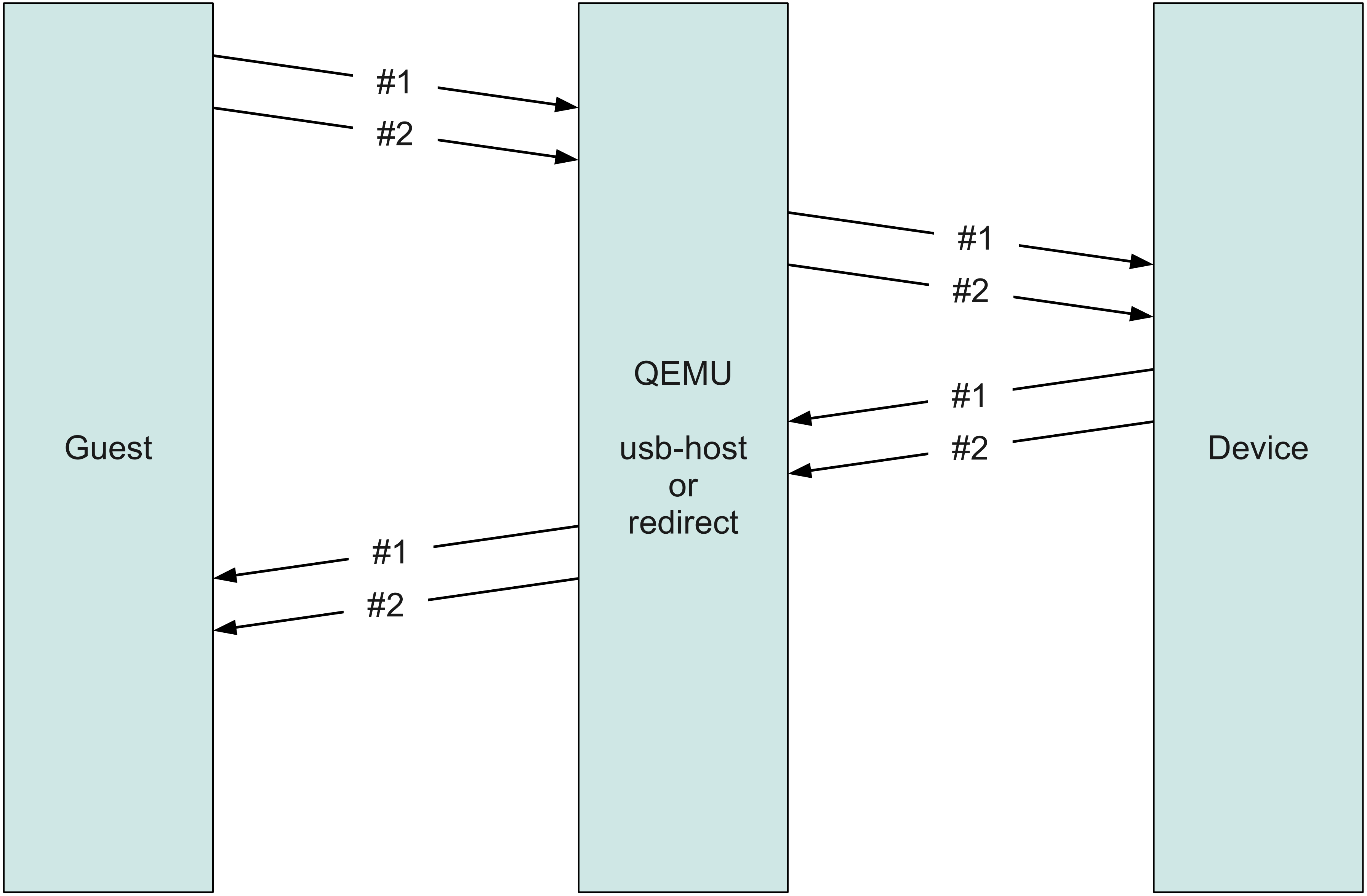
USB Core changes

- Model endpoints & packet queues.
  - move from packet-by-packet to datapipe processing (next slides).
- USB3 descriptor support
  - generate endpoint companion descriptors.
  - generate binary object store descriptors.
- Usual share of cleanups.

Packet queues: without pipelining



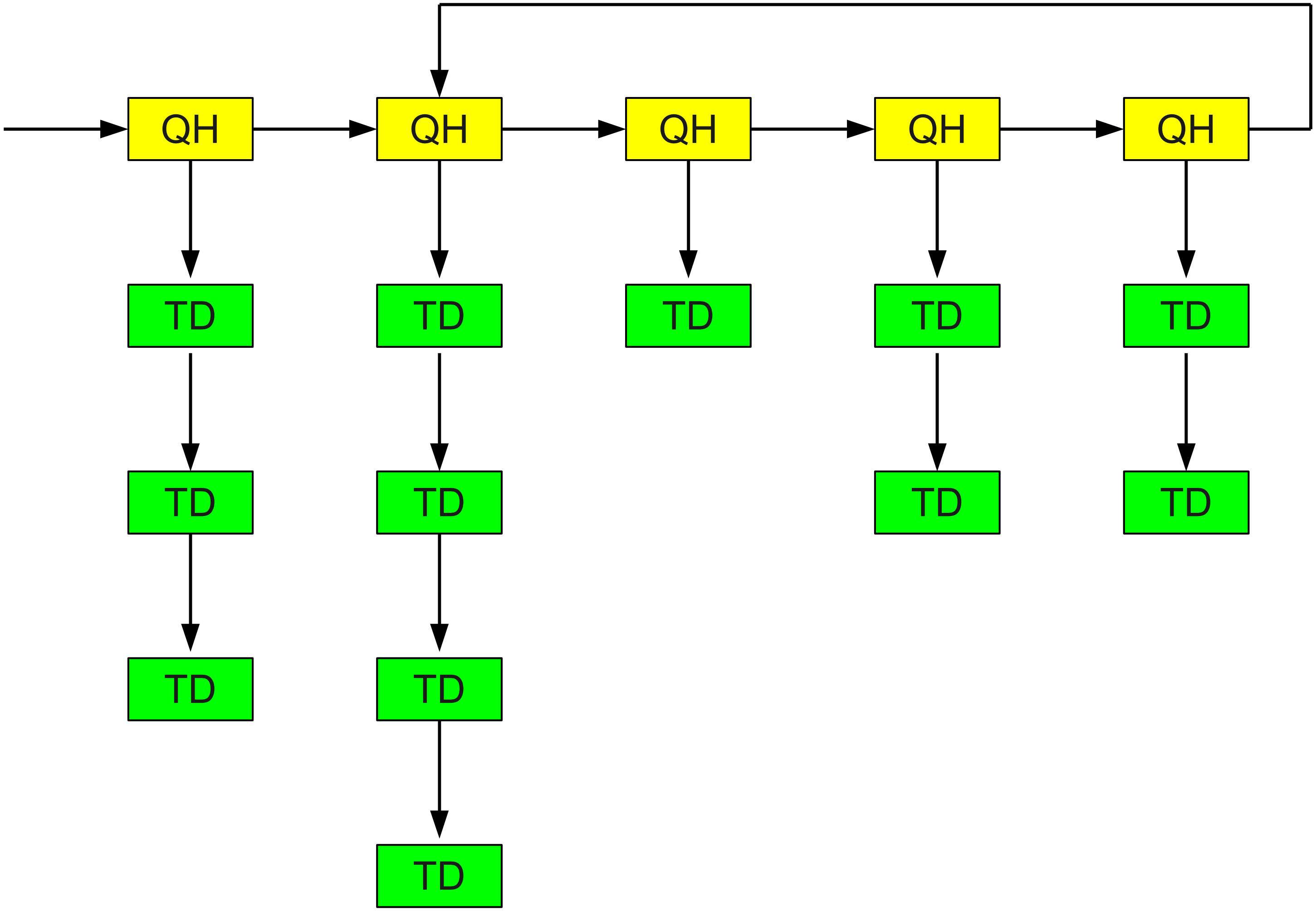
Packet queues: with pipelining



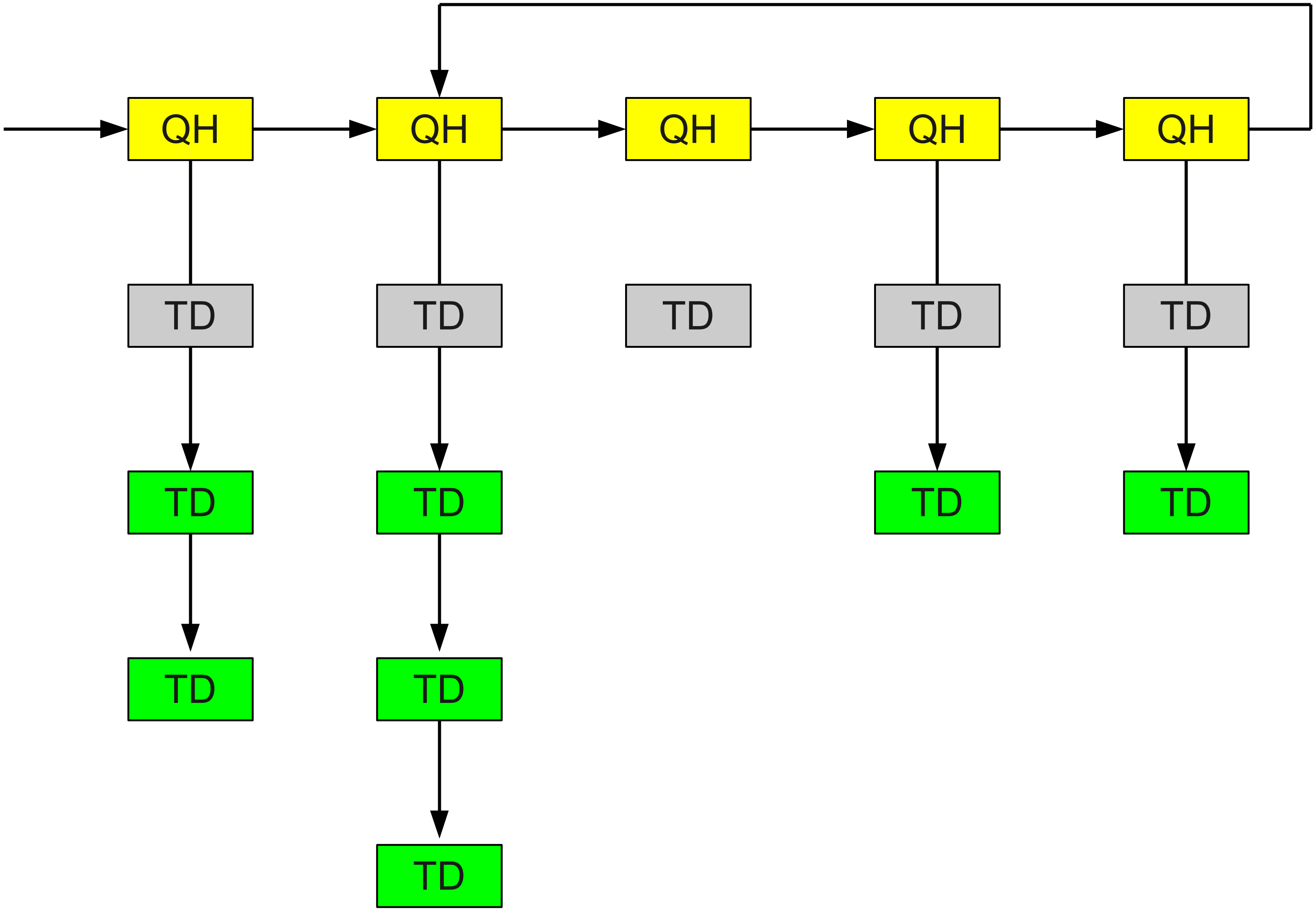
uhci host controller

- Bandwidth accounting (next slides).
- Support queuing & pipelining.
- Fix ich9 companion irq routing.
  - Use all 4 intx pins for multifunction device to reduce IRQ sharing.
- Emulation bugfixes.

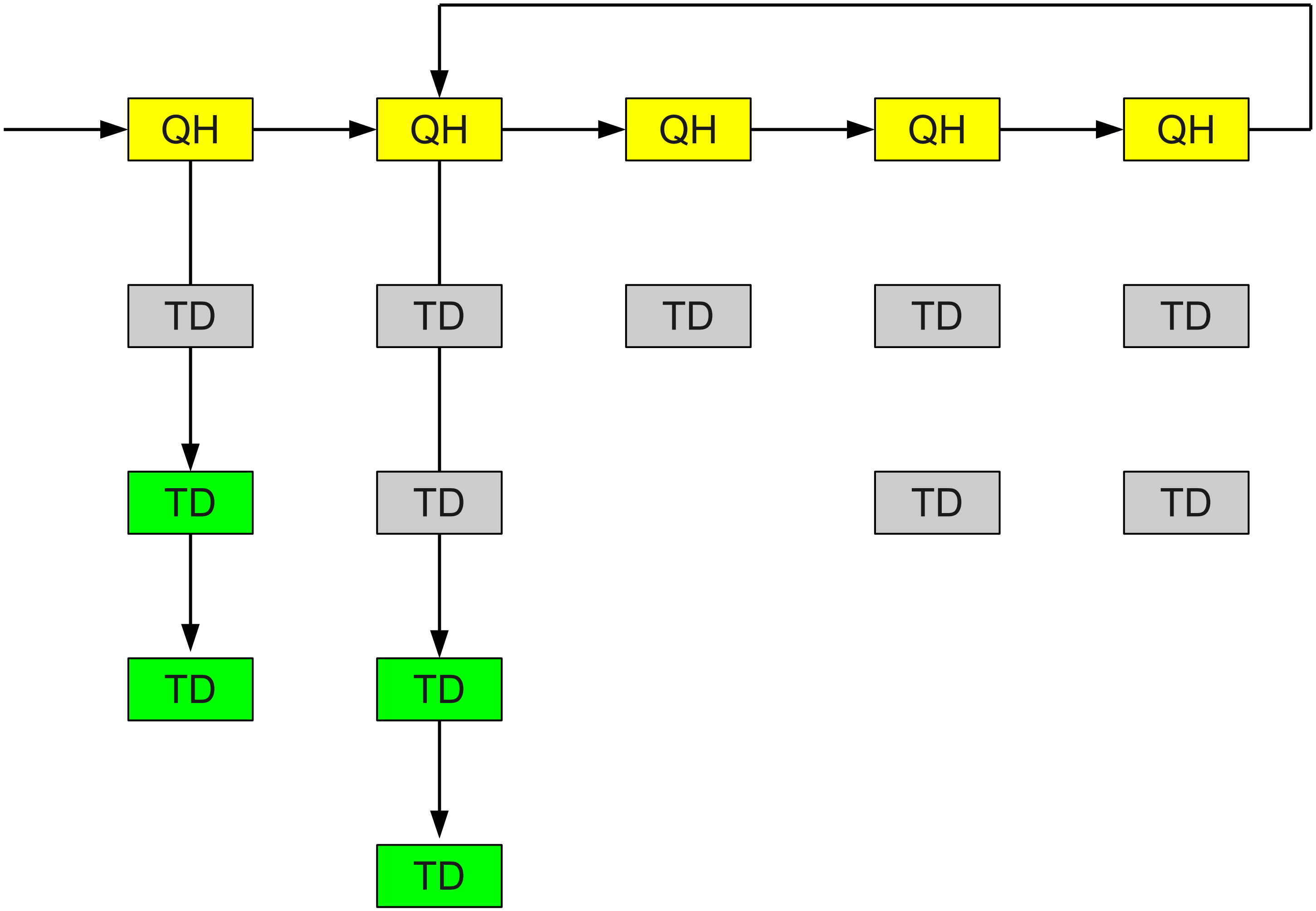
uhci bandwidth management, start



uhci bandwidth management, first round



uhci bandwidth management, second round



ohci host controller

- Emulation bugfixes.
- Added sysbus variant.

ehci host controller

- Support queuing & pipelining.
- Adaptive sleep time.
  - Poll less frequently when the bus is idle.
  - Reduce wakeup rate & burn less cpu time.
  - async schedule (bulk + control) done.
  - sync schedule (iso + intr) wip.
- Emulation bugfixes.
- Added sysbus variant.

New: xhci host adapter

- Based on the code from Hector Martin.
- Virtualization-friendly hardware design.
  - Guest must ring doorbell after queuing up requests in the transfer rings
  - No polling needed, can go for a fully event driven design.
- USB3 support.
  - Streams are still on the TODO list.
- MSI(-X) support.

Direct pass-through: usb-host

- Support queuing & pipelining.
- Emulation bugfixes.
- Live migration / vmsave support.
  - Well, sort of, can't be made guest transparent.
  - Quite useful for savevm / loadvm.
  - Not so for actual live migration (unless you have a robot which migrates the usb device too).

Networked pass-through: usb-redir

- Support queuing & pipelining.
- Full live migration support (with spice).

New: usb-uas

- USB attached SCSI.
  - Modern USB-based HBA.
  - Supports TMF & TCQ.
  - Supports USB3 streams (not implemented yet).
- Not widely used yet.
  - Even USB3 sticks use the old BOT (bulk only transport) protocol.
- Guest side support is cutting edge too.
  - Had to patch the linux kernel to have a stable guest for testing.

Experimental: usb-mtp

- Media Transfer Protocol.
- Easy, network-less filesharing between host + guest.
  - more sane than vvfat.
- Newer android phones use this too.
- Plug & Play with windows guests.
- Proof-of-concept state, needs more polishing.
  - Does syncronous I/O -> adds latencies.
  - Burns alot of cpu time.

TODO: usb3 streams

- Not widely used yet.
- Needs changes on the whole stack
  - qemu xhci emulation.
  - qemu usb core.
  - usb-host / usb-redir / usb-uas.
  - libusbx.
  - linux kernel (usbfs).

TODO: libusbx for usb-host

- Will offload portability issues to libusbx.
- Opportunity to cleanup the historical grown codebase.
- Complex job with high risk of regressions.
  - Probably we'll have both usbfs and libusb implementations living side by side for a while.

TODO: improve xhci

- testing, testing, testing.
  - and fixing the bugs found of course ;)
- add xhci support to seabios.
  - so you can boot from usb sticks.

Hands on: use xhci, part 1

- Add xhci host controller:

```
<controller type='usb' index='0'>
   <address type='pci' slot='0x01' function='0x2'/>
</controller>
<controller type='usb' index='1' model='nec-xhci'>
   <address type='pci' slot='0x0e' function='0x0'/>
</controller>
```

- Attach tablet to xhci:

```
<input type='tablet' bus='usb'>
   <address type='usb' bus='1' port='2'/>
</input>
```

Hands on: use xhci, part 2

- Attach usb stick to xhci:

```
<disk type='file' device='disk'>
   <driver name='qemu' type='qcow2' cache='none'/>
   <source file='/path/to/stick.img'/>
   <target dev='sda' bus='usb'/>
   <address type='usb' bus='1' port='1'/>
</disk>
```

  - Libvirt accepts syntax but ignores specified address.
  - Device will show up on the last host controller added.
  - Fix is being worked on already.

Ressources

- git tree:
  `http://www.kraxel.org/cgit/qemu/log/?h=rebase/usb-next`
- Documentation (qemu src tree)
  `docs/usb2.txt`
  `docs/usb-storage.txt`