USB bulk streams

Background

Bulk endpoint streams were added in the USB 3.0 specification. Streams allow a device driver to overload a bulk endpoint so that multiple transfers can be queued at once.

Streams are defined in sections 4.4.6.4 and 8.12.1.4 of the Universal Serial Bus 3.0 specification at https://www.usb.org/developers/docs/ The USB Attached SCSI Protocol, which uses streams to queue multiple SCSI commands, can be found on the T10 website (https://t10.org/).

Device-side implications

Once a buffer has been queued to a stream ring, the device is notified (through an out-of-band mechanism on another endpoint) that data is ready for that stream ID. The device then tells the host which "stream" it wants to start. The host can also initiate a transfer on a stream without the device asking, but the device can refuse that transfer. Devices can switch between streams at any time.

Driver implications

```
int usb_alloc_streams(struct usb_interface *interface,
              struct usb_host_endpoint **eps, unsigned int num_eps,
              unsigned int num_streams, gfp_t mem_flags);
```

Device drivers will call this API to request that the host controller driver allocate memory so the driver can use up to num_streams stream IDs. They must pass an array of usb_host_endpoints that need to be setup with similar stream IDs. This is to ensure that a UASP driver will be able to use the same stream ID for the bulk IN and OUT endpoints used in a Bi-directional command sequence.

The return value is an error condition (if one of the endpoints doesn't support streams, or the xHCI driver ran out of memory), or the number of streams the host controller allocated for this endpoint. The xHCI host controller hardware declares how many stream IDs it can support, and each bulk endpoint on a SuperSpeed device will say how many stream IDs it can handle. Therefore, drivers should be able to deal with being allocated less stream IDs than they requested.

Do NOT call this function if you have URBs enqueued for any of the endpoints passed in as arguments. Do not call this function to request less than two streams.

Drivers will only be allowed to call this API once for the same endpoint without calling usb_free_streams(). This is a simplification for the xHCI host controller driver, and may change in the future.

Picking new Stream IDs to use

Stream ID 0 is reserved, and should not be used to communicate with devices. If usb_alloc_streams() returns with a value of N, you may use streams 1 though N. To queue an URB for a specific stream, set the urb->stream_id value. If the endpoint does not support streams, an error will be returned.

Note that new API to choose the next stream ID will have to be added if the xHCI driver supports secondary stream IDs.

Clean up

If a driver wishes to stop using streams to communicate with the device, it should call:

```
void usb_free_streams(struct usb_interface *interface,
              struct usb_host_endpoint **eps, unsigned int num_eps,
              gfp_t mem_flags);
```

All stream IDs will be deallocated when the driver releases the interface, to ensure that drivers that don't support streams will be able to use the endpoint.