USB/IP protocol

Architecture

The USB/IP protocol follows a server/client architecture. The server exports the USB devices and the clients import them. The device driver for the exported USB device runs on the client machine.

The client may ask for the list of the exported USB devices. To get the list the client opens a TCP/IP connection to the server, and sends an OP_REQ_DEVLIST packet on top of the TCP/IP connection (so the actual OP_REQ_DEVLIST may be sent in one or more pieces at the low level transport layer). The server sends back the OP_REP_DEVLIST packet which lists the exported USB devices. Finally the TCP/IP connection is closed.

```
 virtual host controller                                 usb host
      "client"                                           "server"
  (imports USB devices)                             (exports USB devices)
          |                                                 |
          |                  OP_REQ_DEVLIST                 |
          | ----------------------------------------------> |
          |                                                 |
          |                  OP_REP_DEVLIST                 |
          | <---------------------------------------------- |
          |                                                 |
```

Once the client knows the list of exported USB devices it may decide to use one of them. First the client opens a TCP/IP connection to the server and sends an OP_REQ_IMPORT packet. The server replies with OP_REP_IMPORT. If the import was successful the TCP/IP connection remains open and will be used to transfer the URB traffic between the client and the server. The client may send two types of packets: the USBIP_CMD_SUBMIT to submit an URB, and USBIP_CMD_UNLINK to unlink a previously submitted URB. The answers of the server may be USBIP_RET_SUBMIT and USBIP_RET_UNLINK respectively.

```
 virtual host controller                                 usb host
      "client"                                           "server"
  (imports USB devices)                             (exports USB devices)
          |                                                 |
          |                  OP_REQ_IMPORT                  |
          | ----------------------------------------------> |
          |                                                 |
          |                  OP_REP_IMPORT                  |
          | <---------------------------------------------- |
          |                                                 |
          |                                                 |
          |            USBIP_CMD_SUBMIT(seqnum = n)         |
          | ----------------------------------------------> |
          |                                                 |
          |            USBIP_RET_SUBMIT(seqnum = n)         |
          | <---------------------------------------------- |
          |                        .                        |
          |                        :                        |
          |                                                 |
          |            USBIP_CMD_SUBMIT(seqnum = m)         |
          | ----------------------------------------------> |
          |                                                 |
          |            USBIP_CMD_SUBMIT(seqnum = m+1)       |
          | ----------------------------------------------> |
          |                                                 |
          |            USBIP_CMD_SUBMIT(seqnum = m+2)       |
          | ----------------------------------------------> |
          |                                                 |
          |            USBIP_RET_SUBMIT(seqnum = m)         |
          | <---------------------------------------------- |
          |                                                 |
          |            USBIP_CMD_SUBMIT(seqnum = m+3)       |
          | ----------------------------------------------> |
          |                                                 |
          |            USBIP_RET_SUBMIT(seqnum = m+1)       |
          | <---------------------------------------------- |
          |                                                 |
          |            USBIP_CMD_SUBMIT(seqnum = m+4)       |
          | ----------------------------------------------> |
          |                                                 |
          |            USBIP_RET_SUBMIT(seqnum = m+2)       |
          | <---------------------------------------------- |
          |                        .                        |
          |                        :                        |
```

For UNLINK, note that after a successful USBIP_RET_UNLINK, the unlinked URB submission would not have a corresponding USBIP_RET_SUBMIT (this is explained in function stub_recv_cmd_unlink of drivers/usb/usbip/stub_rx.c).

```
 virtual host controller                                 usb host
      "client"                                           "server"
  (imports USB devices)                             (exports USB devices)
          |                                                 |
          |            USBIP_CMD_SUBMIT(seqnum = p)         |
          | ----------------------------------------------> |
          |                                                 |
          |               USBIP_CMD_UNLINK                  |
          |         (seqnum = p+1, unlink_seqnum = p)       |
          | ----------------------------------------------> |
          |                                                 |
          |               USBIP_RET_UNLINK                  |
          |        (seqnum = p+1, status = -ECONNRESET)     |
          | <---------------------------------------------- |
          |                                                 |
          |         Note: No USBIP_RET_SUBMIT(seqnum = p)   |
          | <--X---X---X---X---X---X---X---X---X---X---X--- |
          |                        .                        |
          |                        :                        |
          |                                                 |
          |            USBIP_CMD_SUBMIT(seqnum = q)         |
          | ----------------------------------------------> |
          |                                                 |
          |            USBIP_RET_SUBMIT(seqnum = q)         |
          | <---------------------------------------------- |
          |                                                 |
          |                 USBIP_CMD_UNLINK                |
          |          (seqnum = q+1, unlink_seqnum = q)      |
          | ----------------------------------------------> |
          |                                                 |
          |                 USBIP_RET_UNLINK                |
          |            (seqnum = q+1, status = 0)           |
          | <---------------------------------------------- |
          |                                                 |
```

The fields are in network (big endian) byte order meaning that the most significant byte (MSB) is stored at the lowest address.

Protocol Version

The documented USBIP version is v1.1.1. The binary representation of this version in message headers is 0x0111.

This is defined in tools/usb/usbip/configure.ac

Message Format

OP_REQ_DEVLIST:
Retrieve the list of exported USB devices.

Offset	Length	Value	Description
0	2		USBIP version
2	2	0x8005	Command code: Retrieve the list of exported USB devices.
4	4	0x00000000	Status: unused, shall be set to 0

OP_REP_DEVLIST:
Reply with the list of exported USB devices.

Offset	Length	Value	Description
0	2		USBIP version
2	2	0x0005	Reply code: The list of exported USB devices.
4	4	0x00000000	Status: 0 for OK
8	4	n	Number of exported devices: 0 means no exported devices.
0x0C			From now on the exported n devices are described, if any. If no devices are exported the message ends with the previous "number of exported devices" field.
	256		path: Path of the device on the host exporting the USB device, string closed with zero byte, e.g. "/sys/devices/pci0000:00/0000:00:1d.1/usb3/3-2" The unused bytes shall be filled with zero bytes.
0x10C	32		busid: Bus ID of the exported device, string closed with zero byte, e.g. "3-2". The unused bytes shall be filled with zero bytes.
0x12C	4		busnum
0x130	4		devnum
0x134	4		speed
0x138	2		idVendor
0x13A	2		idProduct
0x13C	2		bcdDevice
0x13E	1		bDeviceClass
0x13F	1		bDeviceSubClass
0x140	1		bDeviceProtocol
0x141	1		bConfigurationValue
0x142	1		bNumConfigurations
0x143	1		bNumInterfaces
0x144		m_0	From now on each interface is described, all together bNumInterfaces times, with the following 4 fields:
	1		bInterfaceClass
0x145	1		bInterfaceSubClass
0x146	1		bInterfaceProtocol
0x147	1		padding byte for alignment, shall be set to zero
0xC + i*0x138 + m_(i-1)*4			The second exported USB device starts at i=1 with the path field.

OP_REQ_IMPORT:
Request to import (attach) a remote USB device.

Offset	Length	Value	Description
0	2		USBIP version
2	2	0x8003	Command code: import a remote USB device.
4	4	0x00000000	Status: unused, shall be set to 0
8	32		busid: the busid of the exported device on the remote host. The possible values are taken from the message field OP_REP_DEVLIST.busid. A string closed with zero, the unused bytes shall be filled with zeros.

OP_REP_IMPORT:
Reply to import (attach) a remote USB device.

Offset	Length	Value	Description
0	2		USBIP version
2	2	0x0003	Reply code: Reply to import.
4	4	0x00000000	Status: • 0 for OK • 1 for error
8			From now on comes the details of the imported device, if the previous status field was OK (0), otherwise the reply ends with the status field.
	256		path: Path of the device on the host exporting the USB device, string closed with zero byte, e.g. "/sys/devices/pci0000:00/0000:00:1d.1/usb3/3-2" The unused bytes shall be filled with zero bytes.

Offset	Length	Value	Description
0x108	32		busid: Bus ID of the exported device, string closed with zero byte, e.g. "3-2". The unused bytes shall be filled with zero bytes.
0x128	4		busnum
0x12C	4		devnum
0x130	4		speed
0x134	2		idVendor
0x136	2		idProduct
0x138	2		bcdDevice
0x139	1		bDeviceClass
0x13A	1		bDeviceSubClass
0x13B	1		bDeviceProtocol
0x13C	1		bConfigurationValue
0x13D	1		bNumConfigurations
0x13E	1		bNumInterfaces

The following four commands have a common basic header called 'usbip_header_basic', and their headers, called 'usbip_header' (before transfer_buffer payload), have the same length, therefore paddings are needed.

usbip_header_basic:

Offset	Length	Description
0	4	command
4	4	seqnum: sequential number that identifies requests and corresponding responses; incremented per connection
8	4	devid: specifies a remote USB device uniquely instead of busnum and devnum; for client (request), this value is ((busnum << 16) | devnum); for server (response), this shall be set to 0
0xC	4	direction: • 0: USBIP_DIR_OUT • 1: USBIP_DIR_IN only used by client, for server this shall be 0
0x10	4	ep: endpoint number only used by client, for server this shall be 0; for UNLINK, this shall be 0

USBIP_CMD_SUBMIT:
    Submit an URB

Offset	Length	Description
0	20	usbip_header_basic, 'command' shall be 0x00000001
0x14	4	transfer_flags: possible values depend on the URB transfer_flags (refer to URB doc in Documentation/driver-api/usb/URB.rst) but with URB_NO_TRANSFER_DMA_MAP masked. Refer to function usbip_pack_cmd_submit and function tweak_transfer_flags in drivers/usb/usbip/ usbip_common.c. The following fields may also ref to function usbip_pack_cmd_submit and URB doc
0x18	4	transfer_buffer_length: use URB transfer_buffer_length
0x1C	4	start_frame: use URB start_frame; initial frame for ISO transfer; shall be set to 0 if not ISO transfer
0x20	4	number_of_packets: number of ISO packets; shall be set to 0xffffffff if not ISO transfer
0x24	4	interval: maximum time for the request on the server-side host controller
0x28	8	setup: data bytes for USB setup, filled with zeros if not used.
0x30	n	transfer_buffer. If direction is USBIP_DIR_OUT then n equals transfer_buffer_length; otherwise n equals 0. For ISO transfers the padding between each ISO packets is not transmitted.
0x30+n	m	iso_packet_descriptor

USBIP_RET_SUBMIT:
    Reply for submitting an URB

Offset	Length	Description
0	20	usbip_header_basic, 'command' shall be 0x00000003
0x14	4	status: zero for successful URB transaction, otherwise some kind of error happened.
0x18	4	actual_length: number of URB data bytes; use URB actual_length
0x1C	4	start_frame: use URB start_frame; initial frame for ISO transfer; shall be set to 0 if not ISO transfer
0x20	4	number_of_packets: number of ISO packets; shall be set to 0xffffffff if not ISO transfer
0x24	4	error_count
0x28	8	padding, shall be set to 0
0x30	n	transfer_buffer. If direction is USBIP_DIR_IN then n equals actual_length; otherwise n equals 0. For ISO transfers the padding between each ISO packets is not transmitted.
0x30+n	m	iso_packet_descriptor

USBIP_CMD_UNLINK:
    Unlink an URB

Offset	Length	Description
0	20	usbip_header_basic, 'command' shall be 0x00000002
0x14	4	unlink_seqnum, of the SUBMIT request to unlink
0x18	24	padding, shall be set to 0

USBIP_RET_UNLINK:
    Reply for URB unlink

Offset	Length	Description
0	20	usbip_header_basic, 'command' shall be 0x00000004
0x14	4	status: This is similar to the status of USBIP_RET_SUBMIT (share the same memory offset). When UNLINK is successful, status is -ECONNRESET; when USBIP_CMD_UNLINK is after USBIP_RET_SUBMIT status is 0
0x18	24	padding, shall be set to 0

EXAMPLE

The following data is captured from wire with Human Interface Devices (HID) payload

```
CmdIntrIN:  00000001 00000d05 0001000f 00000001 00000001 00000200 00000040 ffffffff 00000000 00000004 00000000 00000000
CmdIntrOUT: 00000001 00000d06 0001000f 00000000 00000001 00000000 00000040 ffffffff 00000000 00000004 00000000 00000000
            ffffffff860008a784ce5ae21237630000000000000000000000000000000000000000000000000000000000000000000000000000000
RetIntrOut: 00000003 00000d06 00000000 00000000 00000000 00000000 00000040 ffffffff 00000000 00000000 00000000 00000000
RetIntrIn:  00000003 00000d05 00000000 00000000 00000000 00000000 00000040 ffffffff 00000000 00000000 00000000 00000000
            ffffffff860011a784ce5ae2123763612891b102010000040000000000000000000000000000000000000000000000000000000000000
```