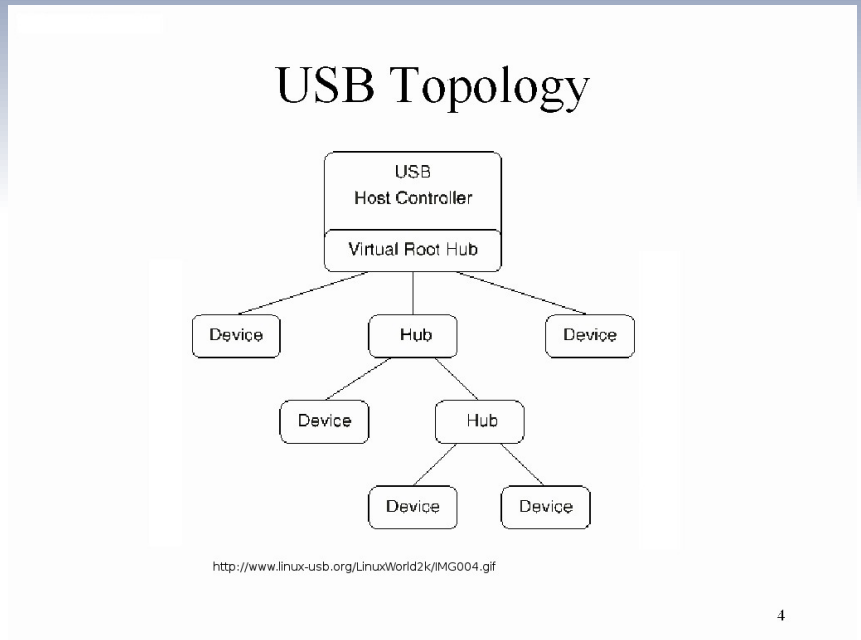
Easy USB device access with PyUSB

Wander Lairson Costa
Mozilla Corporation

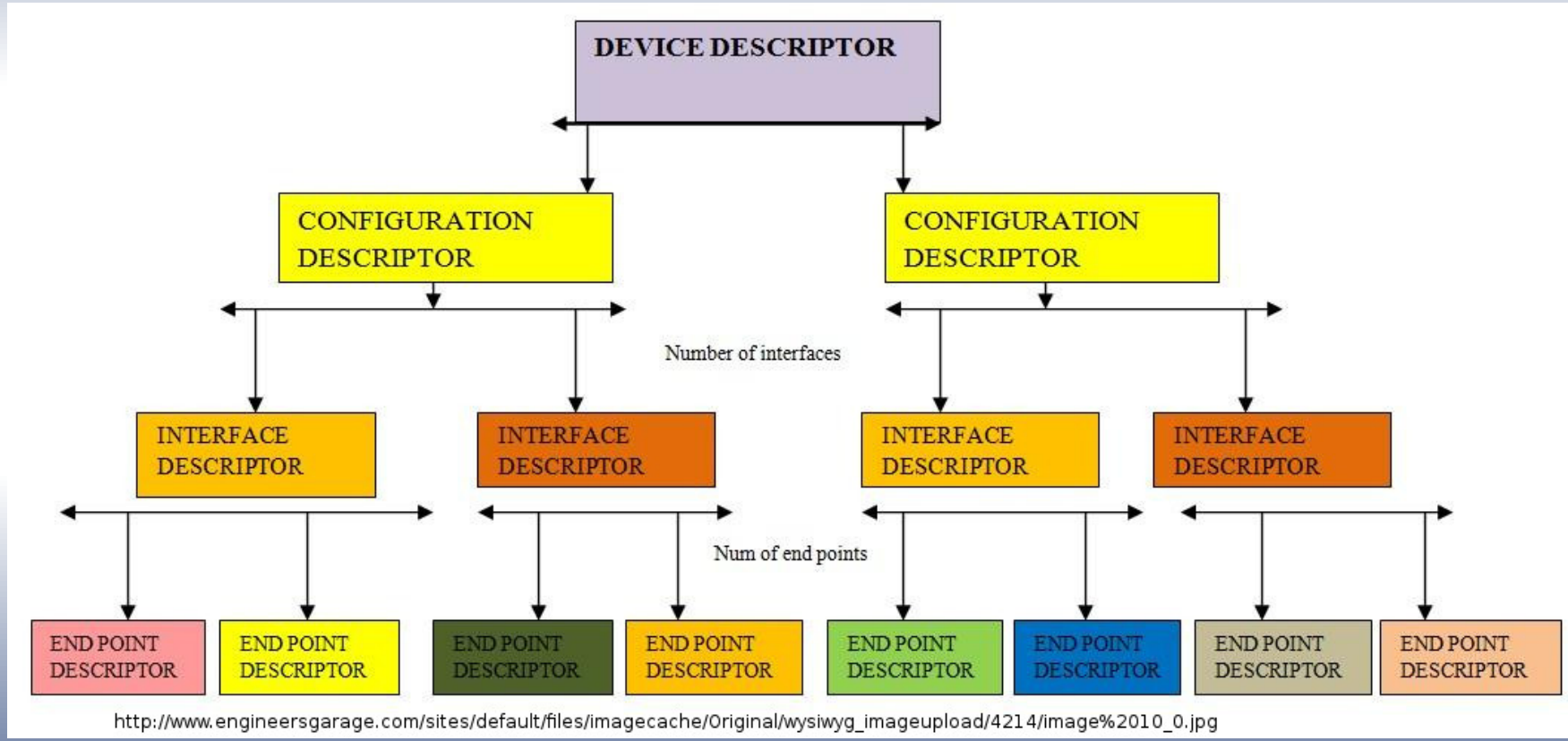
https://twitter.com/walac00
https://github.com/walac
https://br.linkedin.com/in/walac

USB Introduction

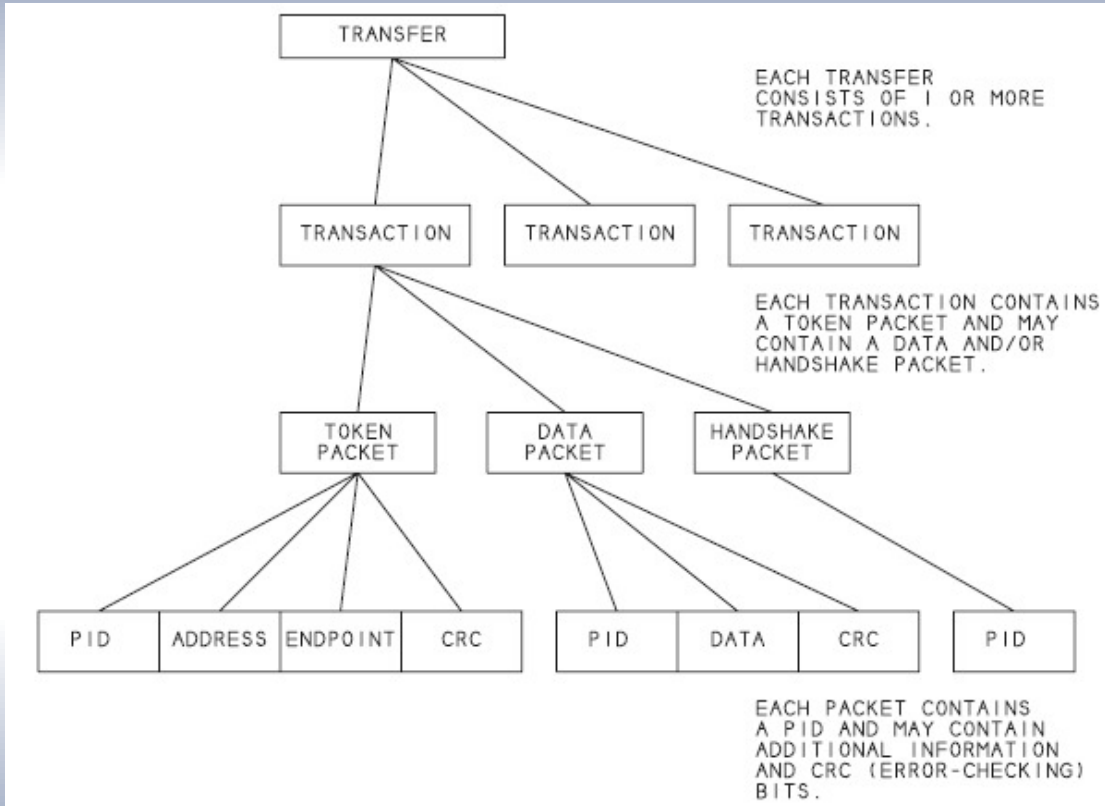


- Universal Serial Bus
- Created to replace several slow connections (serial, parallel, etc)
- It is hot plug and play bus. You don't need to turn off your computer connect a new device.
- It works as a Master/Slave. All requests are initiated by the Host (polling).

USB Descriptors

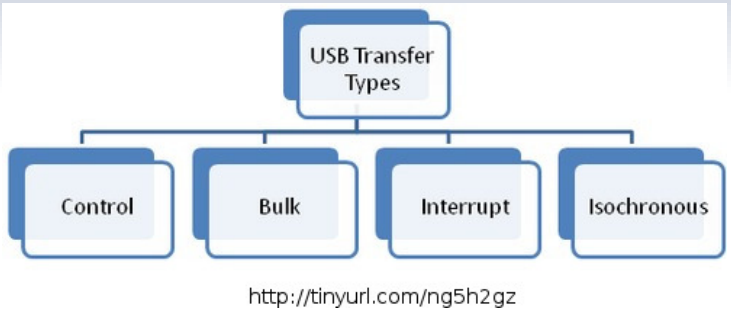


USB Transfers



http://www.crifan.com/files/doc/docbook/usb_basic/release/htmls/images/usb_transfer_transaction.png

Transfer Types



http://tinyurl.com/ng5h2gz

Control used by the Host to configure devices. It is the only transfer which supports bidirectional endpoints and which has its format defined by the USB spec.

Bulk typically used to transfer large amount of data, such as printers and disks. The bandwidth allocation can vary, according to the bus availability.

Interrupt typically used for lower latency, short data transfers. It is often used by input devices suchs as mouse and keyboards. Despite its name, it works by device polling from the Host.

Isochronous used for real time streaming. It prioritizes date rate, and it is the only transfer which does not guarantee data delivery, no CRC check or retry is performed.

Accessing USB devices

Accessing USB devices

Possible solutions:

Accessing USB devices

Possible solutions:

1. Write a kernel device driver.

Accessing USB devices

Possible solutions:

1. Write a kernel device driver.
2. Write a user mode device driver.

Accessing USB devices

Possible solutions:

1. Write a kernel device driver.
2. Write a user mode device driver.
3. Use a generic USB library:

Accessing USB devices

Possible solutions:

1. Write a kernel device driver.
2. Write a user mode device driver.
3. Use a generic USB library:
   - libusb 0.1
   - libusb 1.0
   - ~~libusbx~~
   - OpenUSB
   - libusb-win32
   - libusbK

Accessing USB devices

Possible solutions:

1. Write a kernel device driver.
2. Write a user mode device driver.
3. Use a generic USB library:
   - libusb 0.1
   - libusb 1.0
   - ~~libusbx~~
   - OpenUSB
   - libusb-win32
   - libusbK

   - Which one to use?

Accessing USB devices

Possible solutions:

1. Write a kernel device driver.
2. Write a user mode device driver.
3. Use a generic USB library:
   - libusb 0.1
   - libusb 1.0
   - ~~libusbx~~
   - OpenUSB
   - libusb-win32
   - libusbK

   - Which one to use?
   - PyUSB anwser: anyone!

PyUSB

- Up to version 0.4, PyUSB was thin wrapper for libusb 0.1
- Starting at version 1.0, PyUSB was redesigned to be a platform agnostic, library neutral and easy to use USB access package for Python.
- PyUSB detaches its API from the backend library used.
- You can select the backend you want, but in general PyUSB selects the most suitable backend for you.
- It works on any Python version $\geq$ 2.4!
- 100% written in Python!

Demo: Getting the device serial number (C Version)

Demo: Getting the device serial number (C Version)

```c
#include <libusb.h>
#include <string.h>

int main(void)
{
  libusb_device_handle *handle;
  libusb_device *dev;
  struct libusb_device_descriptor desc;
  int transfered;
  char serial_number[256];

  /* initialized the library */
  libusb_init(NULL);

  /* open the device */
  handle = libusb_open_device_with_vid_pid(NULL, 0x4d8, 0xfa2e);

  /* get the serial number */
  dev = libusb_get_device(handle);
  libusb_get_device_descriptor(dev, &desc);
  libusb_get_string_descriptor_ascii(handle, desc.iSerialNumber, serial_number, 256);
  printf("Serial number = %s\n", serial_number);

  /* cleanup resources */
  libusb_close(handle);
  libusb_exit(NULL);
}
```

Demo: Getting the device serial number (Python version)

Demo: Getting the device serial number (Python version)

```
from usb.core import find
dev = find(idVendor=0x4d8, idProduct=0xfa2e)
print(dev.serial_number)
```

Demo: Getting the device serial number (Python version)

```
from usb.core import find
dev = find(idVendor=0x4d8, idProduct=0xfa2e)
print(dev.serial_number)
```

- It works with libusb 0.1 and libusb 1.0

Demo: Getting the device serial number (Python version)

```
from usb.core import find
dev = find(idVendor=0x4d8, idProduct=0xfa2e)
print(dev.serial_number)
```

- It works with libusb 0.1 and libusb 1.0
- The `find` function accepts any device descriptor field

Demo: Getting the device serial number (Python version)

```
from usb.core import find
dev = find(idVendor=0x4d8, idProduct=0xfa2e)
print(dev.serial_number)
```

- It works with libusb 0.1 and libusb 1.0
- The `find` function accepts any device descriptor field
- `find` can also return all devices that match the criteria (`find_all=True` argument).

```
devices = find(find_all=True) # all devices
printers = find(bDeviceClass=7, find_all=True) # all
printers
```

Demo: Writing data to an endpoint (C version)

Demo: Writing data to an endpoint (C version)

```c
#include <libusb.h>
#include <string.h>

int main(void)
{
  libusb_device_handle *handle;
  const char data[] = "test";
  int transfered;

  /* initialized the library */
  libusb_init(NULL);

  /* open the device */
  handle = libusb_open_device_with_vid_pid(NULL, 0x4d8, 0xfa2e);

  /* setup device */
  libusb_set_configuration(handle, 1);
  libusb_claim_interface(handle, 0);

  /* transfer the data */
  libusb_bulk_transfer(handle, 1, data, strlen(data), &transfered, 1000);

  /* cleanup resources */
  libusb_release_interface(handle, 0);
  libusb_close(handle);
  libusb_exit(NULL);
}
```

Demo: Writing data to an endpoint (Python version)

Demo: Writing data to an endpoint (Python version)

```
from usb.core import find
dev = find(idVendor=0x4d8, idProduct=0xfa2e)
dev.set_configuration()
dev.write(1, "test")
```

Demo: Writing data to an endpoint (Python version)

```
from usb.core import find
dev = find(idVendor=0x4d8, idProduct=0xfa2e)
dev.set_configuration()
dev.write(1, "test")
```

- You don't need to know the endpoint type, PyUSB takes care of it!

Demo: Writing data to an endpoint (Python version)

```
from usb.core import find
dev = find(idVendor=0x4d8, idProduct=0xfa2e)
dev.set_configuration()
dev.write(1, "test")
```

- You don't need to know the endpoint type, PyUSB takes care of it!
- Reading is as easy as writing:

```
data = dev.read(0x81, 4) # return an array.array
object
```

More info

Additional resources:

Project page https://walac.github.io/pyusb

Source code https://github.com/walac/pyusb

Tutorial https://github.com/walac/pyusb/blob/master/docs/tutorial.rst

Thank you!

Twitter	@walac00
Github	https://github.com/walac
Blog	https://walac.github.io
Linkedin	https://br.linkedin.com/in/walac