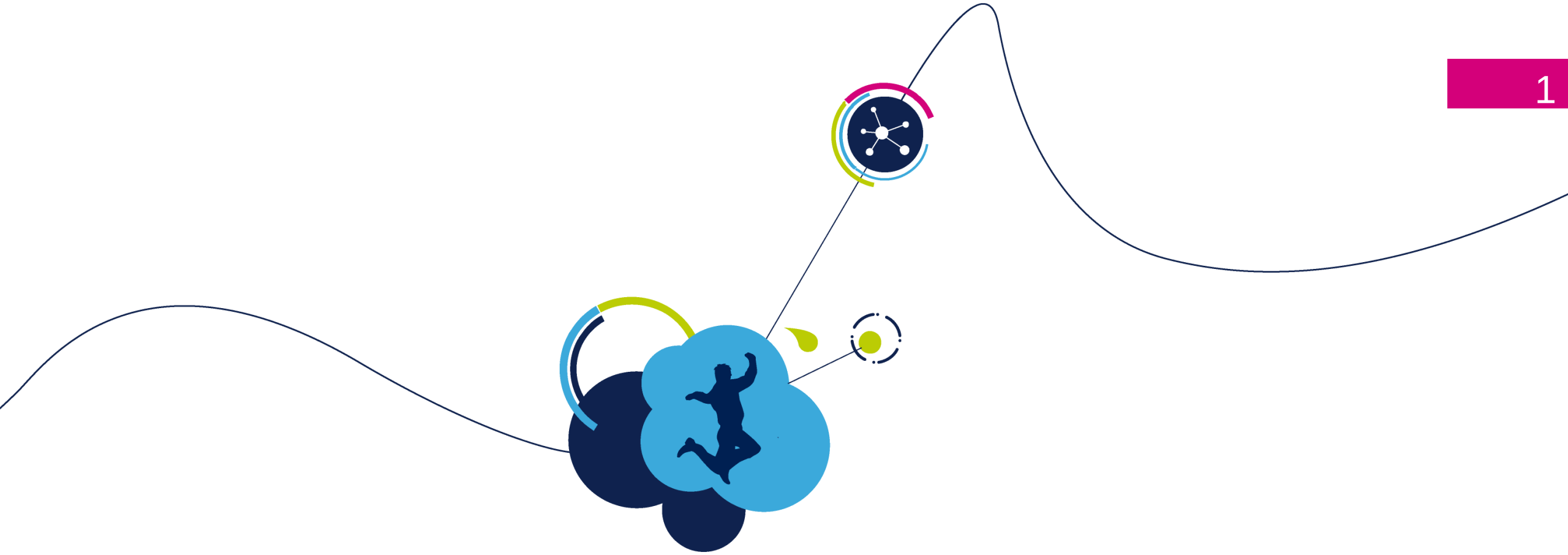


USB CDC throughput

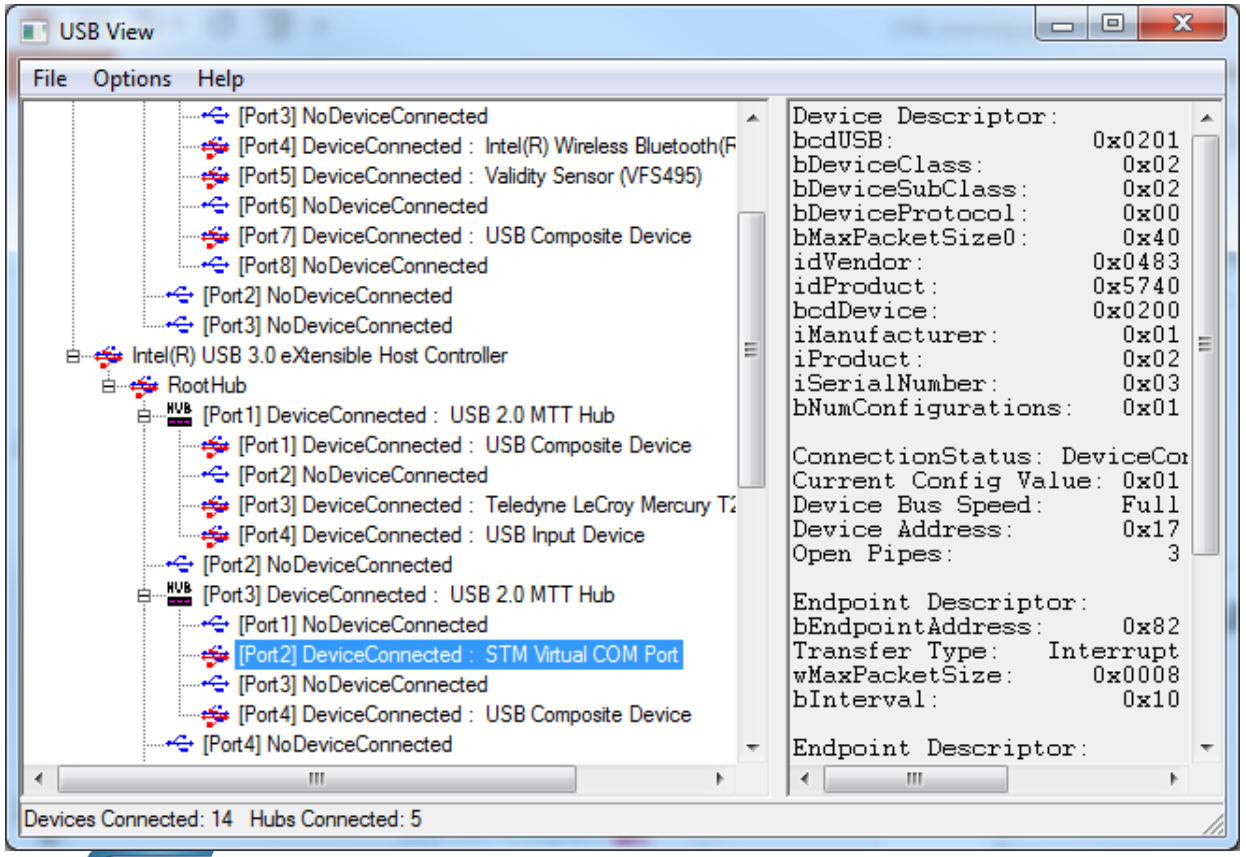
USB CDC throughput

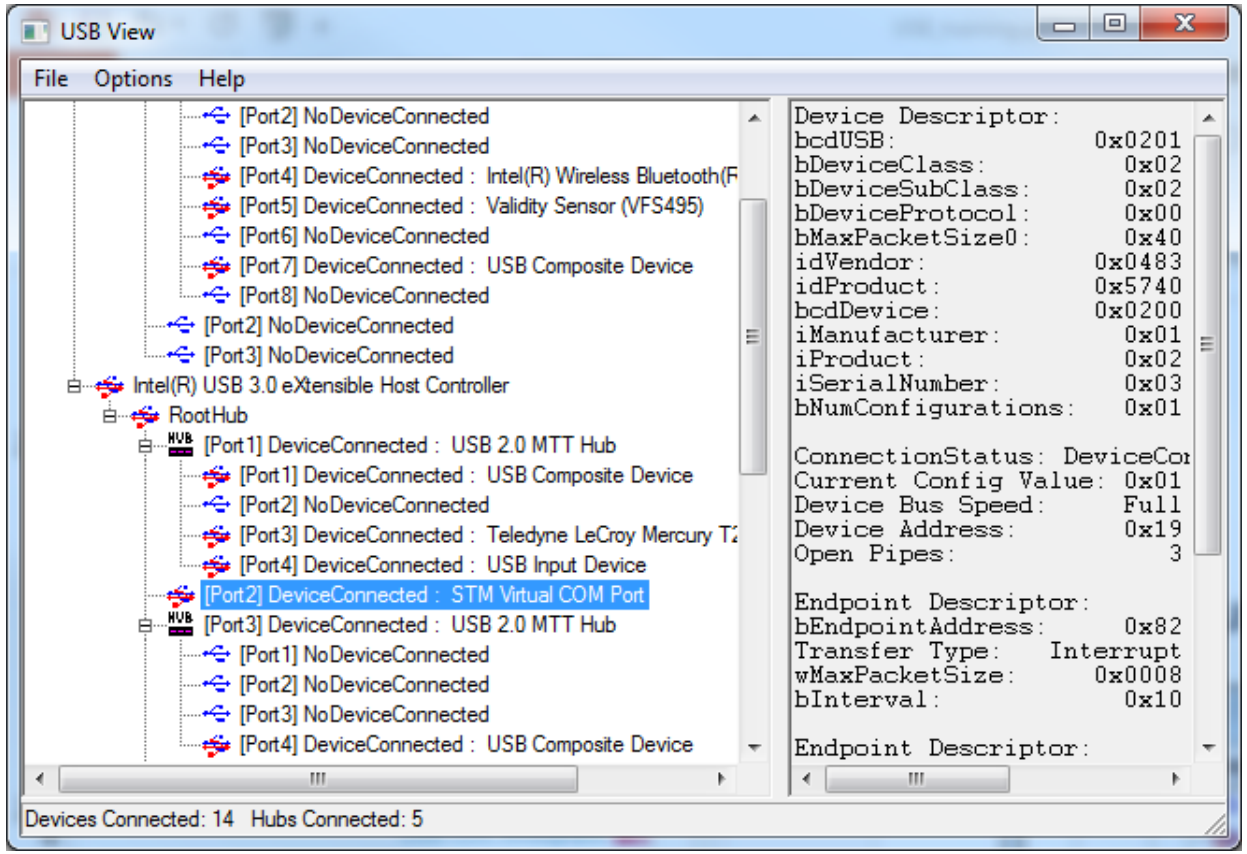
- Device throughput strongly depends on host
  - As host is responsible for communication initialization device throughput may be limited by host, that is not sending IN/OUT packet fast enough
  - Depend also on host OS and used software

- CDC bulk endpoints have no bandwidth guarantee
  - It’s very convenient to use host port with no other function connected on the hub
  - With more functions connected on one hub bandwidth may be limited

USB CDC throughput

- USB bus interconnection in your PC can be checked using USB view or similar tool





life.augmented

USB CDC throughput

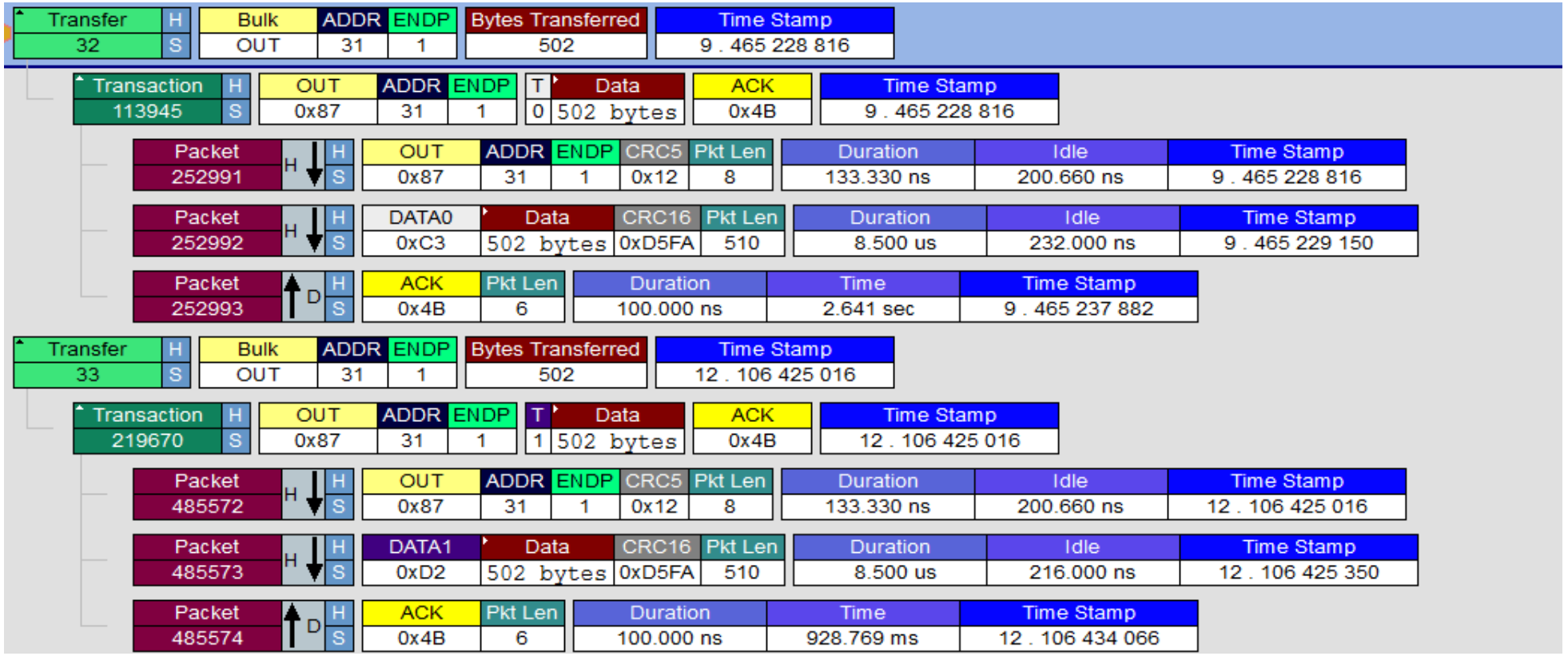
- What is the limiting the bandwidth if not optimal yet?
- If you can observe NAKs from device, throughput is limited by device

Transfer	H	Bulk	ADDR	ENDP	Bytes Transferred	Time Stamp
20	S	OUT	32	1	393216	15 . 343 093 332

Transaction	H	OUT	ADDR	ENDP	T	Data	ACK	Time	Time Stamp
104961	S	0x87	32	1	0	512 bytes	0x4B	9.818 us	15 . 343 093 332
104962	S	0x87	32	1	1	512 bytes	0x5A	47.616 us	15 . 343 103 150
104964	S	0x2D	32	1		ACK 0x4B	Time 9.484 us	Time Stamp 15 . 343 150 766	
104966	S	0x87	32	1	1	512 bytes	0x4B	11.716 us	15 . 343 160 250
104967	S	0x87	32	1	0	512 bytes	0x5A	30.800 us	15 . 343 171 966
104968	S	0x2D	32	1		NAK 0x5A	Time 22.516 us	Time Stamp 15 . 343 202 766	
104970	S	0x2D	32	1		ACK 0x4B	Time 2.884 us	Time Stamp 15 . 343 225 282	
104972	S	0x87	32	1	0	512 bytes	0x4B	11.700 us	15 . 343 228 166

USB CDC throughput

- What is the limiting the bandwidth if not optimal yet?
  - If there are no NAKs from device side, bulk transactions smaller then maximum size (512 bytes for HS) and long delays between packets, throughput is limited by host



Transfer	H/S	Bulk	ADDR	ENDP	Bytes Transferred	Time Stamp
32		OUT	31	1	502	9 . 465 228 816

Transaction	H/S	OUT	ADDR	ENDP	T	Data	ACK	Time Stamp
113945		0x87	31	1	0	502 bytes	0x4B	9 . 465 228 816

Packet	H/S	OUT	ADDR	ENDP	CRC5	Pkt Len	Duration	Idle	Time Stamp
252991	H	0x87	31	1	0x12	8	133.330 ns	200.660 ns	9 . 465 228 816

Packet	H/S	DATA0	Data	CRC16	Pkt Len	Duration	Idle	Time Stamp
252992	H	0xC3	502 bytes	0xD5FA	510	8.500 us	232.000 ns	9 . 465 229 150

Packet	H/S	ACK	Pkt Len	Duration	Time	Time Stamp
252993	D	0x4B	6	100.000 ns	2.641 sec	9 . 465 237 882

Transfer	H/S	Bulk	ADDR	ENDP	Bytes Transferred	Time Stamp
33		OUT	31	1	502	12 . 106 425 016

Transaction	H/S	OUT	ADDR	ENDP	T	Data	ACK	Time Stamp
219670		0x87	31	1	1	502 bytes	0x4B	12 . 106 425 016

Packet	H/S	OUT	ADDR	ENDP	CRC5	Pkt Len	Duration	Idle	Time Stamp
485572	H	0x87	31	1	0x12	8	133.330 ns	200.660 ns	12 . 106 425 016

Packet	H/S	DATA1	Data	CRC16	Pkt Len	Duration	Idle	Time Stamp
485573	H	0xD2	502 bytes	0xD5FA	510	8.500 us	216.000 ns	12 . 106 425 350

Packet	H/S	ACK	Pkt Len	Duration	Time	Time Stamp
485574	D	0x4B	6	100.000 ns	928.769 ms	12 . 106 434 066

USB CDC throughput

- Virtual COM port application usually limit communication speed
- For test is the fastest option choose command line interface
- With Windows OS

```
OUT direction – send a file in binary form to COM port 10
copy /B file \\.\COM10
IN direction – store communication from COM port to file.txt
type COM10 > file.txt
```

- With Linux OS
  - Access to USB need administrator privileges – add sudo or check udev rules

```
OUT direction – send a file in binary form to COM port ttyACM0
bash -c 'cat filename >/dev/ttyACM0'
IN direction – receive data from COM port and discard immediately
cat /dev/ttyACM0 >/dev/null
```

USB CDC throughput

- Our test (default cube generated project – no library optimization), IN direction – STM32F723 OTG_HS is acting as device
  - Win10, usbser.sys, command line – 4 MB/s
  - Win10, libusb, command line – 6 MB/s
  - Win10, usbser.sys, C# application – 8.75 MB/s
  - Linux command line – 12.5 MB/s

USB CDC throughput

- CDC device throughput improvement
- Maximize FIFO size
- USB device is handled in interrupt in STM32 Cube HAL library – avoid other extensive high priority interrupts
- Optimize Cube HAL library
  - Avoid checking of unused flags – lose universality
  - Time critical events inside USB interrupt handler shall be handled first