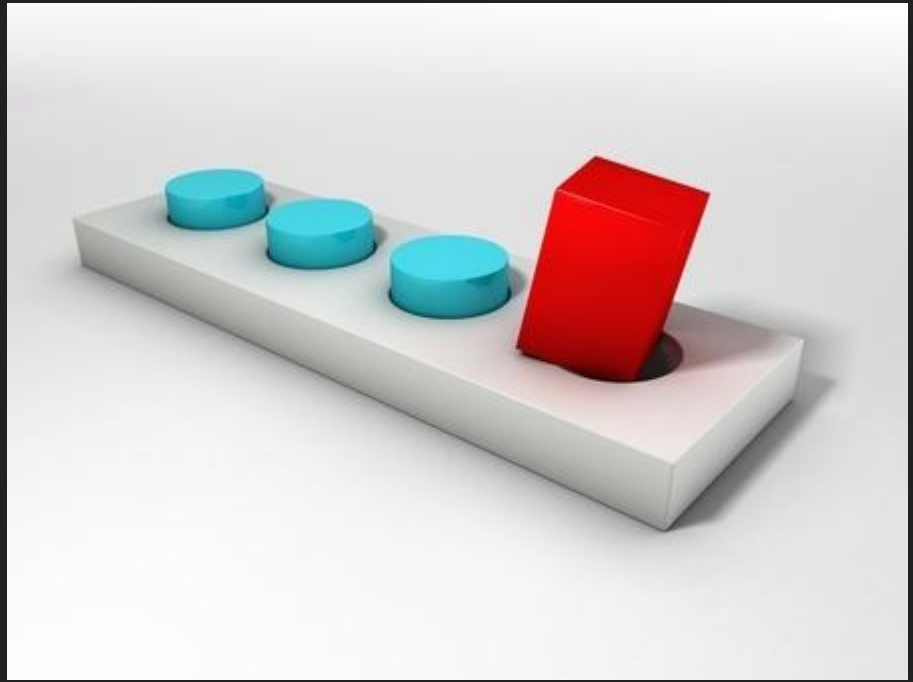
Research Report: On the Feasibility of Retrofitting Operating Systems with Generated Protocol Parsers

Wayne Wang, Peter C. Johnson
Middlebury College

Integrating generated protocol parsers into existing OS



Process

Operating Systems

FreeBSD
illumos
Linux

Protocols

IP
USB 3.0 (xHCI)
SCSI

Research Process

- Ingress & Egress points

Protocol	Ingress Point	Egress Point
IP	hand-off from link layer (e.g., Ethernet)	hand-off to transport layer (e.g., TCP, UDP)
USB	bus controller interrupt handler	hand-off to device-specific driver
SCSI	HBA controller interrupt handler	hand-off to filesystem or block I/O layer

- traced execution of functions
- Labeled each line of code into 9 categories

Category: Parsing

(IP - FreeBSD)

```
    if (m->m_flags & M_FASTFWD_OURS) {
        m->m_flags &= ~M_FASTFWD_OURS;
        /* Set up some basics that will be used later. */
        ip = mtod(m, struct ip *);
        hlen = ip->ip_hl << 2;
        ip_len = ntohs(ip->ip_len);
        goto ours;
    }
```

Category - Multiplexing

(USB - Linux)

```
switch (trb_type) {
case TRB_COMPLETION:
    handle_cmd_completion(xhci, &event->event_cmd);
    break;
case TRB_PORT_STATUS:
    handle_port_status(xhci, event);
    update_ptrs = 0;
    break;
case TRB_TRANSFER:
    ret = handle_tx_event(xhci, &event->trans_event);
    if (ret >= 0)
        update_ptrs = 0;
    break;
```

Category - State management

(SCSI - illumos)

```
/*
 * Only update state to IOCOMPQ if we were in the INTR state.
 * Any other state (e.g. TIMED_OUT, ABORTED) needs to remain.
 */
if (pwrk->state == PMCS_WORK_STATE_INTR) {
    pwrk->state = PMCS_WORK_STATE_IOCOMPQ;
}
```

Category - Memory management

(USB - FreeBSD)

```
usbd_get_page(&sc->sc_hw.root_pc, 0, &buf_res);
```

Category - Hardware manipulation

(SCSI - linux)

```
if (intstat & CMDCMPLT) {
    ahd_outb(ahd, CLRINT, CLRCMDINT);

    /*
     * Ensure that the chip sees that we've cleared
     * this interrupt before we walk the output fifo.
     * Otherwise, we may, due to posted bus writes,
     * clear the interrupt after we finish the scan,
     * and after the sequencer has added new entries
     * and asserted the interrupt again.
     */
    if ((ahd->bugs & AHD_INTCOLLISION_BUG) != 0) {
        if (ahd_is_paused(ahd)) {
            /*
             * Potentially lost SEQINT.
             * If SEQINTCODE is non-zero,
             * simulate the SEQINT.
             */
            if (ahd_inb(ahd, SEQINTCODE) != NO_SEQINT)
                intstat |= SEQINT;
        }
    } else {
        ahd_flush_device_writes(ahd);
    }
```

Category - Synchronization

(SCSI - FreeBSD)

```
   cam_periph_lock(periph);
```

Categories

- **Queueing**

```
/*
 * If we are waiting on a queue, just remove the USB transfer
 * from the queue, if any. We should have the required locks
 * locked to do the remove when this function is called.
 */
usbd_transfer_dequeue(xfer);
```

- **Diagnostic**

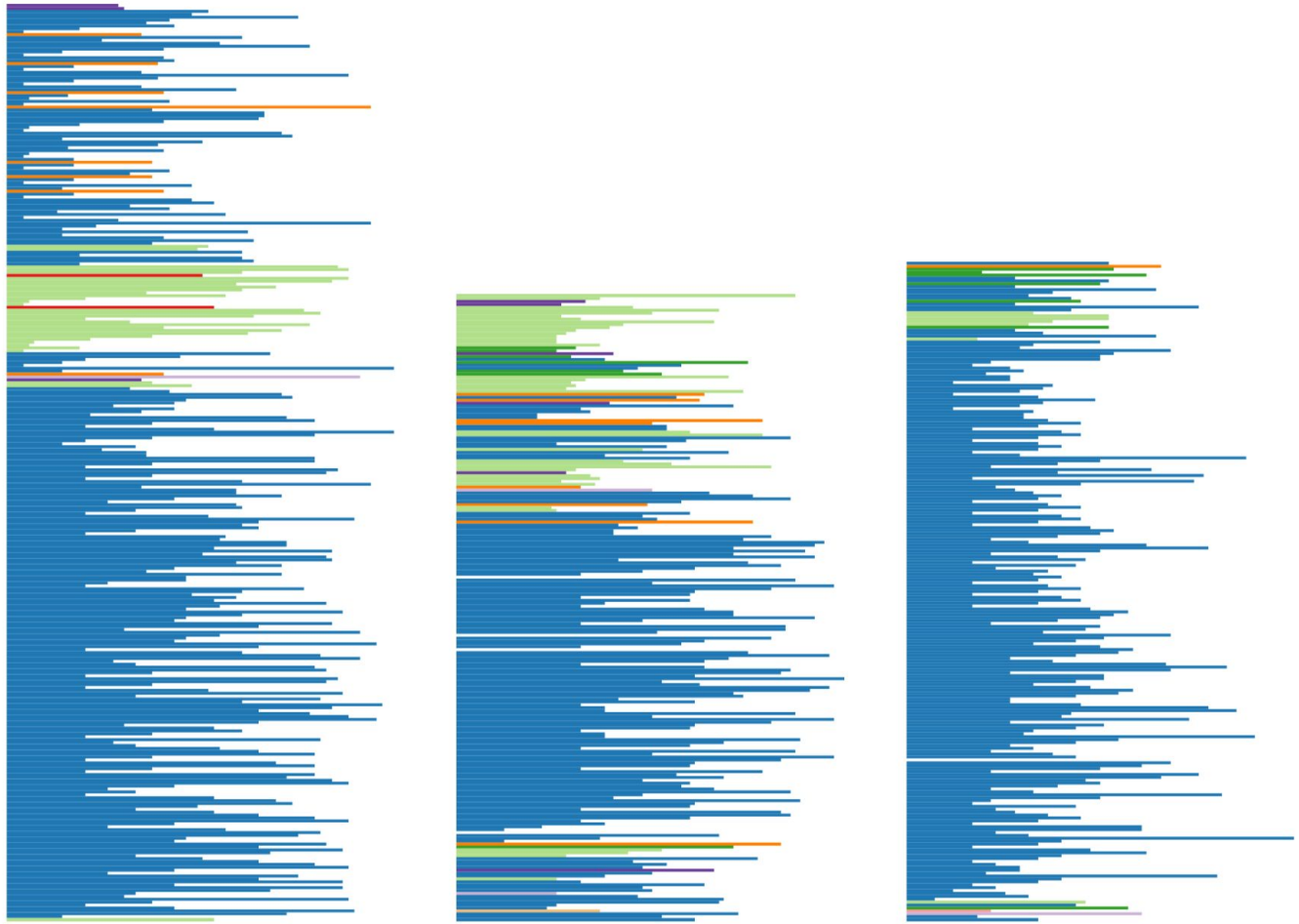
```
SCSI_LOG_HLCOMPLETE(1, scmd_printk(KERN_INFO, SCpnt,
                    "sd_done: completed %d of %d bytes\n",
                    good_bytes, scsi_bufflen(SCpnt)));
```

- **Assertion**

```
M_ASSERTPKTHDR(m);
NET_EPOCH_ASSERT();
```

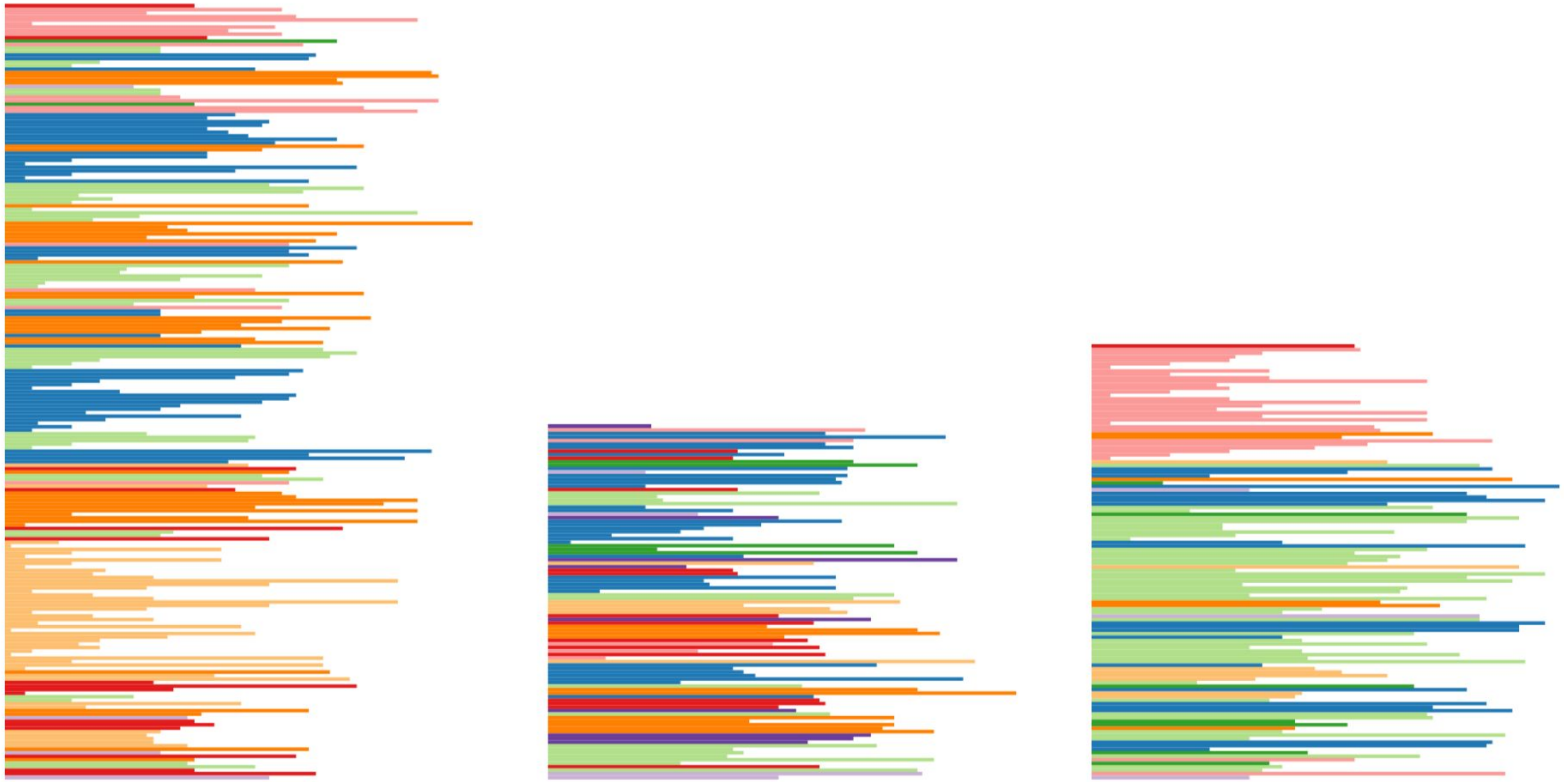
Results - categorization

	IP			USB			SCSI		
	FreeBSD	Illumos	Linux	FreeBSD	Illumos	Linux	FreeBSD	Illumos	Linux
parsing	85.1%	70.8%	92.5%	23.1%	31.7%	17.7%	23.6%	17.0%	22.5%
multiplexing	0.3%	0.9%	0.4%	1.8%	4.0%	2.4%	1.4%	1.9%	2.5%
state	10.4%	18.1%	3.1%	19.9%	14.9%	36.3%	30.6%	19.8%	41.7%
memory management	0.0%	3.2%	3.1%	0.9%	5.0%	5.6%	4.2%	4.7%	5.0%
hardware manipulation	0.0%	0.0%	0.4%	7.7%	5.0%	25.8%	5.6%	2.8%	9.2%
synchronization	0.6%	0.0%	0.0%	7.2%	13.9%	0.8%	8.3%	11.3%	0.8%
queueing	0.0%	0.5%	0.0%	21.7%	5.9%	6.5%	12.5%	7.5%	2.5%
diagnostics	2.5%	3.7%	0.4%	17.6%	10.9%	4.8%	9.7%	8.5%	15.8%
assertion	0.9%	2.8%	0.0%	0.0%	8.9%	0.0%	4.2%	26.4%	0.0%
total	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%



(a) FreeBSD (b) Illumos (c) Linux

Fig. 1. Categorized code diagram for IP. Colors correspond to categorizations shown in Tables IV and V; line width corresponds to the length of the individual lines of code.



(a) FreeBSD (b) Illumos (c) Linux

Fig. 2. Categorized code diagram for USB. Colors correspond to categorizations shown in Tables IV and V; line width corresponds to the length of the individual lines of code.

Results - multiplexing

IP

	FreeBSD	Illumos	Linux
destination class		callback	
ip.protocol	jump table	switch	jump table

USB

	FreeBSD	Illumos	Linux
OS request handler		callback	callback
device driver	callback	callback	callback
storage layer			callback

SCSI

	FreeBSD	Illumos	Linux
XHCI event type	switch	switch	switch
transfer	callback		
endpoint type		switch	if / else
endpoint	callback		
transfer	callback	switch, callback	callback

Results - multiplexing

Netgraph in FreeBSD

&

NetFilter in Linux

What is done

- Survey
- Categorization
- Summarize

What is next

- enumerate general kernel functions used
- analyze their similarity within a single OS & across OS
- define an API for a universal shim