The Open Network Lab (Part 4)

Ken Wong
Applied Research Laboratory
Computer Science and Engineering Department
http://www.arl.wustl.edu/~kenw
kenw@arl.wustl.edu
http://www.onl.wustl.edu (ONL)
National Science Foundation ANI-023826, CNS-0551651, REL-0632580

Topics
1) Link Emulation
2) Delay Plugin and Callbacks
3) Resources
Emulating Link Rate $R = 8$ Mbps

- Pkt with length $L$ arrives at time $t$ ➔
  - Leaves at time $t + L/R$ if queue is empty
  - Leaves at time $t' + L/R$ where $t' = \text{time preceding pkt leaves if queue is NOT empty}$

Need
- Pkt queue
- Compute departure time of next pkt ➔ Need ...
  - clock
  - event scheduler

Periodic callback function
- a function that is called at a fixed time period
- need to register the callback function
  - `PLUGIN_MSR_CLOCK_HANDLER_FCT(pdelay_callback, MSR_CLOCK_HANDLER_PCU_ID, ticks);`
Processing Logic

- **_handle_packet**
  - Remove pkt from bufferList
  - Queue pkt with timestamp = current time
  - Forward all pkts with expired departure times

- **_callback**
  - Forward all pkts with expired departure times

**Issues**
- Pkt queue is shared by _handle_pkt and _callback
  - Need to atomically access queue ➔ disable-enable interrupts
- When should callback be registered and by whom?
- Once callback is registered, _callback will be called even if setup has not completed

pdelay-600 Processing Logic

**Features**
- pdelay_callback is registered with kernel when plugin is loaded, but doesn’t do anything until setup done
  - `global_pdelay_inst_ptr != NULL` when setup is done
- Interrupts are disabled while queue is processed
  - Prevents race between pdelay_callback and pdelay_handle_packet

- See Tutorial ➔ More Plugins ➔ Delay Packets for pseudo-code logic
pdelay Instance Variables

```c
#define DELAY_TIME 50 // Initial delay (msec)
static int cbPeriod = 500; // callback period in usec (500 usec)

struct pdelay_instance {
    struct rp_instance rootinstance;
    HDRQ_t qhead; // addr of 1st pkt buffer in delay queue
    int qlen; // #pkts in delay queue
    int pkt_count; // #pkts handled by instance
    int drop_count; // #pkts dropped by instance
    int fwd_count; // #pkts forwarded by instance
    int max_qlen; // maximum queue length
    int earliest_depart_time; // earliest time nxt pkt should leave (msec)
    int delay_time; // amount to delay (msec)
};
```

delay_time
- initialized to DELAY_TIME (50 msec)
- can be changed via control message from RLI

pdelay_handle_packet (1)

```c
void pdelay_handle_packet(
    struct rp_instance *this, // points to instance
    void *bufferList) // points to list of pkt buffers
{
    struct pdelay_instance *inst = (struct pdelay_instance *)this;
    u_int32_t curr_time; // Current time in ms
    struct msr_bufhdr_t *buffer; // incoming pkt buffer
    int s;

    s = PLUGIN_SPLCLOCK_FCT();
    buffer = msr_firstBuffer(bufferList);
    msr_removeBuffer(bufferList, buffer);

    // process pkt ...
    pdelay_fwdpkts (curr_time, inst, 0);
    PLUGIN_SPLX_FCT(s);
    return;
}
```

called when pkt arrives to SPC
pdelay_handle_packet (2)

```c
struct pdelay_instance *inst = (struct pdelay_instance *)this;

u_int32_t curr_time; // Current time in ms

struct msr_bufhdr_t *buffer; // incoming pkt buffer

... msr_removeBuffer(bufferList, buffer);

inst->pkt_count++;

curr_time = PLUGIN_CPU_CLOCK_1MSEC_FCT();

buffer->time_fwd = curr_time + inst->delay_time;

msr_addBuffer(&inst->qhead, buffer);  // insert pkt into queue

inst->qlen++;

if (inst->max_qlen < inst->qlen)
    inst->max_qlen = inst->qlen;

if (inst->qlen == 1) {
    inst->earliest_depart_time = curr_time + inst->delay_time;
}

... pdelay_init_class

```
pdelay_create_instance (1)

```c
struct rp_instance * pdelay_create_instance (
  struct rp_class *myclass,
  u_int32_t instanceid
) {
    struct pdelay_instance *myinst;
    int s;

    MSR_PLUGIN_MALLOC(myinst, struct pdelay_instance *,
                      sizeof(struct pdelay_instance), M_MSR, M_WAITOK);
    if (myinst == NULL) return NULL;

    ... standard instance initialization ...

    global_pdelay_inst_ptr = (struct rp_instance *) myinst;

    return (struct rp_instance *)myinst;
}
```

called when plugin instance is created

pdelay_create_instance (2)

```c
msr_initBuffer(&myinst->qhead); // Instance specific processing
  myinst->qlen = 0;
  myinst->pkt_count = 0;
  myinst->drop_count = 0;
  myinst->fwd_count = 0;
  myinst->max_qlen = 0;
  myinst->delay_time = DELAY_TIME;
  myinst->earliest_depart_time = 0;

  global_pdelay_inst_ptr = (struct rp_instance *) myinst;

  return (struct rp_instance *)myinst;
}
```
pdelay_callback

```c
1 void pdelay_callback(void) {
2     struct pdelay_instance *myinst;
3     u_int32_t curr_time;
4     int s;
5     
6     if (global_pdelay_inst_ptr == NULL) return;
7     myinst = (struct pdelay_instance *) global_pdelay_inst_ptr;
8     if (myinst == NULL) return;
9     
10    if (myinst->qlen > 0) { // check for expired delays
11      s = PLUGIN_SPLCLOCK_FCT(); // Block clock interrupts
12      curr_time = PLUGIN_CPU_CLOCK_1MSEC_FCT();
13      pdelay_fwdpkt (1, curr_time, myinst);
14      PLUGIN_SPLX_FCT(s);
15    }
16 }
```
called after plugin is registered

delay_fwdpkt

```c
1 void pdelay_fwdpkt ( 
2     u_int32_t curr_time, //
3     struct pdelay_instance *myinst, // instance addr
4     int caller // 0: called by pdelay_handle_packet
5     // 1: called by pdelay_callback
6     ) {
7     struct msr_bufhdr_t *buffer; // pkt buffer
8     
9     while ( (myinst->qlen > 0) &&
10              (curr_time >= myinst->earliest_depart_time) ) {
11         buffer = msr_firstBuffer(&myinst->qhead);
12         msr_removeBuffer(&myinst->qhead, buffer);
13         if (PLUGIN_IP_FWD_FCT(buffer) == 0) myinst->fwd_count++;
14         myinst->qlen--;
15         if (myinst->qlen > 0) { // check next packet
16             buffer = msr_firstBuffer(&myinst->qhead);
17             myinst->earliest_depart_time = buffer->time_fwd;
18         }
19     }
```
called from pdelay_handle_packet and pdelay_callback
Your Lab 3

- Four flows with parameters
  - S: unique session (Flow) id
  - R: #messages (pkts) remaining to be transferred

- Features
  - One queue for each flow
  - Choose pkt from flow with smallest R when FPX queue is empty

See Code For

- ~/onl/stdPlugins/pdelay-600/
  - delay plugin
- ~/onl/stdPlugins/fpxReads-987
  - reading various FPX registers (e.g., queue length)