ONL Plugins

Charlie Wiseman
Applied Research Laboratory
Computer Science and Engineering Department
cgw1@arl.wustl.edu

http://www.onl.wustl.edu (ONL)
Basic NSP Architecture

- CP = control processor
- PP = port processor
- SPC = smart port card
- FPX = field programmable port extender
Adding Features with SPC Plugins

SPC uses qid to direct packet to plugin

plugins are kernel-resident software modules

on egress, packets mapped to per-flow queue

filters used to direct packets to SPC queue

on ingress, packets mapped to VOQ

to/from links

to/from switch core
Plugin Resources

- Tutorial Pages
  - Router Plugins
  - Writing Your First Plugin
  - Summary Information => Predefined Plugins
  - Summary Information => SPC Macros and Functions

- Code Examples
  - ~onl/stdPlugins/
    - COUNTER-432, stats-100, stringSub-102, nullPlugin-103

- Preparation

```bash
onlusr> mkdir ~/myplugins        # make your plugin dir
onlusr> cp -R ~onl/stdPlugins/bin myplugins  # newplugin.pl
onlusr> cp -R ~onl/stdPlugins/doc myplugins  # README
onlusr> cp -R ~onl/stdPlugins/template myplugins
```
Writing Your Own Plugins

- Plugins are netBSD kernel modules in SPC
- Written in C but follow OO-like pattern
  - plugin type is called a class – each class has a name and numerical id
  - a plugin class must be “loaded” into an SPC before it can be run
  - a class can be instantiated one or more times in an SPC
    - each instance is bound to a queue id, so it can receive packets from FPX
    - each instance may have private data that is retained across packets.
    - may also define class data that is accessible to all instances
- Each plugin class has a standard set of functions
  - `pluginName_handle_packet` – receive pkt; optionally return pkt(s)
  - `pluginName_handle_msg` – receive and respond to control msgs
  - `pluginName_create_instance` – used to initialize per instance variables
  - `pluginName_free_instance` – used to cleanup data structures
  - miscellaneous other functions – typically don’t require changes
Example 1

- Copy stats-100
  - Edit Makefile
    - Change KMOD to 'test'
    - Add 'CFLAGS += -DMSRDEBUG'
  - Edit test.h, test.c
    - Change any instance of 'stats' to 'test'
      - :%s/stats/test/g
  - Edit test.h
    - Change test_ID to 4242
Example 1 instance structure

```c
struct test_instance {
    struct rp_instance rootinstance; // do not touch

    // add declarations for per instance data here
    int icmpCnt; // number of icmp packets seen recently
    int tcpCnt; // number of tcp packets seen recently
    int udpCnt; // number of udp packets seen recently
    int icmpTot; // total number of icmp packets seen
    int tcpTot; // total number of tcp packets seen
    int udpTot; // total number of udp packets seen

};
```
Example 1 create instance

```c
struct rp_instance * test_create_instance( struct rp_class *theClass, u_int32_t instanceid) {
    struct test_instance *newInst;
    MSR_PLUGIN_MALLOC(newInst, struct test_instance *, sizeof(struct test_instance), M_MSR,M_WAITOK);
    if (newInst == NULL) return NULL;
    newInst->rootinstance.rpclass = &test_class;
    newInst->rootinstance.handle_packet = test_handle_packet;
    newInst->rootinstance.free_instance = test_free_instance;
    newInst->rootinstance.bind_instance = test_bind_instance;
    newInst->rootinstance.unbind_instance = test_unbind_instance;
    newInst->rootinstance.handle_msg = test_handle_msg;
    newInst->rootinstance.instanceid = instanceid;
    newInst->icmpCnt = 0; newInst->tcpCnt = 0; newInst->udpCnt = 0;
    newInst->icmpTot = 0; newInst->tcpTot = 0; newInst->udpTot = 0;
    return (struct rp_instance *) newInst;
}
```
Example 1 handle packet

```c
void test_handle_packet(struct rp_instance *this, void *bufferList) {
    struct test_instance *inst = (struct test_instance *) this;
    msr_bufhdr_t *buffer = msr_firstBuffer(bufferList);
    struct ip *iph = msr_pkt_iph(buffer);
    int len = msr_iplen(iph);
    int proto = msr_ipproto(iph);
    if (proto == 1) {
        inst->icmpCnt++; inst->icmpTot++;
    } else if (proto == 6) {
        inst->tcpCnt++; inst->tcpTot++;
    } else if (proto == 17) {
        inst->udpCnt++; inst->udpTot++;
    }
    msr_removeBuffer(bufferList, buffer);
    msr_freeBuffer(buffer);
}
```
Example 2

- Add debugging
  » MSR_DEBUG statements
- Allow packets to continue through router
  » Simply don't drop packets
- test.c changes:

```c
// test.c

void test_handle_packet() {
    ...
    MSR_DEBUG( (MSR_DEBUG_PLUGIN | MSR_DEBUG_LEVEL_INFO,
        "test_handle_packet: instance %d got %d packet\n",
        this->instanceid, proto) );

    // remove packet from input list
    //msr_removeBuffer(bufferList, buffer);
    //msr_freeBuffer(buffer);
}
```
Example 3

- Modify packet output port
  - Change extra 'shim' header fields
  - See multicast-101 plugin
  - See Tutorial -> Summary Information -> Summary of SPC Macros and Functions

- test.c changes:
  
  ```c
  test_handle_packet() {
      ...
      MSR_Shim_t *shim = msr_pkt_shim(buffer);
      uint ovin = msr_shim_get_ovin(shim);
      ...
      if(proto == 1 && msr_vin_get_pn(ovin) == 3) {
          msr_shim_set_ovin(shim,msr_vin_make(2,0));
      }
      ...
  }
  ```
Example 4

- Add internal queue
  » Need queue structure
  » See pdelay-600
  » See Tutorial -> More Plugins -> Delay Packets

- test.h changes to instance structure:

  ```
  struct test_instance {
    ...
    HDRQ_t qhead;
  };
  ```

- test.c changes to create instance:

  ```
  test_create_instance() {
    ...
    msr_initBuffer(&newInst->qhead);
    ...
  }
  ```
Example 4

- test.c changes to handle packet:

```c
void test_handle_packet() {
    ...
    mshr_bufhdr_t *buf_intq;
    ...
    buf_intq = mshr_firstBuffer(&inst->qhead);
    if(buf_intq != NULL) {
        mshr_removeBuffer(&inst->qhead, buf_intq);
        if (PLUGIN_IP_FWD_FCT(buf_intq) == 0) {
            MSR_DEBUG(... ok message ...);
        } else {
            MSR_DEBUG(... not ok ...);
        }
    }
    mshr_removeBuffer(bufferList, buffer);
    mshr_addBuffer(&inst->qhead, buffer);
    ... REMOVE shim output port changes ...
    ...
}
```
Example 5

- Adding a callback function
  » Useful for adding delays and periodic scheduling
  » See pdelay-600, dropdelay-610
  » See Tutorial -> More Plugins -> Delay Packets

- test.c changes
  » Add global instance pointer:
    · struct rp_instance *global_test_inst_ptr;
  » Add callback function declaration:
    · void test_callback(void);
Example 5

• test.c changes to create instance:

```c

void test_create_instance() {
    ...
    int ticks = PLUGIN_MSR_USEC2TICKS_FCT(500*10);
    ...
    global_test_inst_ptr = (struct rp_instance *) newInst;
    if (PLUGIN_MSR_CLOCK_HANDLER_FCT(test_callback,
        MSR_CLOCK_HANDLER_PCU_ID, ticks) < 0) {
        MSR_DEBUG(... error ....);
    }
    ...
}
```
Example 5

- test.c changes to handle packet:

```c
void test_handle_packet()
{
    ...
    int s;
    ...
    ... Remove de-queueing step ...
    s = PLUGIN_SPLCLOCK_FCT();
    msr_removeBuffer(bufferList, buffer);
    msr_addBuffer(&inst->qhead, buffer);
    PLUGIN_SPLX_FCT(s);
    ...
}
```
Example 5

- test.c addition, test_callback:

```c
void test_callback(void) {
    int s;
    msr_bufhdr_t *buffer;
    struct test_instance *inst = (struct test_instance *) global_test_inst_ptr;
    s = PLUGIN_SPLCLOCK_FCT();
    buffer = msr_firstBuffer(&inst->qhead);
    if(buffer != NULL) {
        msr_removeBuffer(&inst->qhead, buffer);
        if (PLUGIN_IP_FWD_FCT(buffer) == 0) {
            MSR_DEBUG(... ok ...);
        } else {
            MSR_DEBUG(... error ...);
        }
    } else {
        MSR_DEBUG(... error ...);
    }
    PLUGIN_SPLX_FCT(s);
}
```
Summary

- Read the tutorial!
- /users/onl/stdPlugins contains all of the standard plugin code
- /users/wiseman/myplugins/test-4242 contains the five examples in these slides as test.{c,h}.1 – test.{c,h}.5
- Build in small increments and debug at each step
  » Always try just a few packets at first then move on to larger bandwidth streams
  » Turn debugging off for anything more than a few packets!
The End