Cross Layer Interaction in NS2

Jiazi Yi
LIX, Ecole Polytechnique, France

November 18, 2012

Abstract

This document introduces how to obtain cross layer information in NS2 (layer 2 packet drop, queue length, etc.), with can be used for upper layer routing protocol. The routing protocol can be in NS2 native C++ or AgentJ Java code.

1 Layer 2 Packet Drop Notification

The packet drop at layer 2 in NS2 can be of two cases:

ns2.34/mac/arp.cc When a resolve of a destination address failed because maximum number of retires is exceeded.

ns2.34/mac/mac-802_11.cc The transmission failed because of maximum number of retries is exceeded. It can be that an RTS is sent out but no corresponding CTS is received, or no ACK is received.

1.1 Layer 2 Packet Drop Notification in C++

This subsection introduces how to enable layer 2 packet drop notification for protocol in C++.

1. In NS2, the common header of a packet has a field where one can specify a function that will be called if the packet can't be sent by layer-2 agent.

   In the protocol *yourProtocol.cc*, add the following two lines:

```
ch->xmit_failure_       =       yourProtocol_mac_failed_callback;
ch->xmit_failure_data_ = (void*)this;
```

2. Then one needs to implement the function which is registered inside the common header. In *YourProtocol.cc*, add:

```
static void
YourProtocol_mac_failed_callback(Packet *p, void *arg){
  ((YourProtocol*)arg)->mac_failed(p);
}
```

3. The function *mac_failed* needs to be added to *YourProtocol.h*:

```
class YourProtocol : public Agent{
 /*....*/
  public:
    YourProtocol(nsaddr_t);
    int          command(Packet*, Handler*);
    void         recv(Packet*, Handler*);
    void mac_failed(Packet*);
};
```

   and *YourProtocol.cc*:

```
void YourProtocol::mac_failed(Packet* p){
  struct hdr_ip* ih      = HDR_IP(p);
  struct hdr_cmn* ch     = HDR_CMN(p);
  debug("%f: Transmission from %d to %d failed at MAC layer\n",
    CURRENT_TIME, ra_addr(), ch->next_hop());

  /* do necessary operation here*/

  drop(p, DRO_RTR_MAC_CALLBACK);
}
```

1.2 Layer 2 Packet Drop Notification with AgentJ

AgentJ provides the ability to simulate real-world Java applications within NS2 [1]. This subsection describes how to enable layer 2 packet drop notification for AgentJ protocol in Java.

1. Add the function *notifyRoutingProtocol* to handle packet drop in mac layer.
   In ns2.34/mac/mac-802_11.cc, add the function:

```
void
Mac802_11::notifyRoutingProtocol(Packet* p)
{
        struct hdr_mac802_11 *mac = HDR_MAC802_11(p);
        if (GET_ETHER_TYPE(mac->dh_body) != ETHERTYPE_IP)
                return; // ignore non-IP packets

        Tcl& tcl = Tcl::instance();
        tcl.evalf("$node_(%d) set ragent_", addr());

        const char* tclResult = tcl.result();
        Agentj* agent = (Agentj*)TclObject::lookup(tclResult);
        if (agent == NULL || agent->getRouterAgent() == NULL){
                printf("ERROR (mac-802_11.cpp):   agentj or router agent is
                    NULL\n");
                abort();
        }

        struct hdr_cmn *ch = HDR_CMN(p);
        struct hdr_ip *ip = HDR_IP(p);
        if (ip->daddr() == IP_BROADCAST)
                return; // ignore control traffic

        //get the right type of protocol
        YourProtocolRouter* yourprotocolRouter = dynamic_cast<
            YourProtocolRouter*>(agent->getRouterAgent());
        if (yourprotocolRouter){
                PLOG(PL_TRACE, "%d ms: Node %d: MAC notifying routing protocol
                    of unsuccessfully sent packet (%d) to %d\n", (int) (
                    Scheduler::instance().clock() * 1000), addr(), ch->uid_,
                    ip->daddr());
                yourprotocolRouter->notifyRoutingProtocol(p);
        }
        /****else if (rplRouter){
            maybe other protocol types goes here...
        } */
        else{
                printf("ERROR (mac-802_11.cpp): Router agent is unknown!!\n");
                abort();
        }
}
```

2. Add the trigger of *notifyRoutingProtocol* in the link layer.

[1] http://cs.itd.nrl.navy.mil/work/agentj

The defined function *notifyRoutingProtocol* should be called when a transmission failed either because of 802.11 retransmission timeout or ARP resolution failure.

For 802.11 retransmission timeout, call the *notifyRoutingProtocol* in the function of *void Mac802_11::RetransmitDATA()*:

```
void
Mac802_11::RetransmitDATA()
{ /*..... */
        if(*rcount >= thresh) {
                /* IEEE Spec section 9.2.3.5 says this should be greater than
                   or equal */
                macmib_.FailedCount++;
                /* tell the callback the send operation failed
                   before discarding the packet */
                hdr_cmn *ch = HDR_CMN(pktTx_);
                if (ch->xmit_failure_) {
                        ch->size() -= phymib_.getHdrLen11();
                        ch->xmit_reason_ = XMIT_REASON_ACK;
                        ch->xmit_failure_(pktTx_->copy(),
                                          ch->xmit_failure_data_);
                }
        //trigger the notification process
                notifyRoutingProtocol(pktTx_);
                discard(pktTx_, DROP_MAC_RETRY_COUNT_EXCEEDED);
                pktTx_ = 0;
                *rcount = 0;
                rst_cw();
        }
        /*......*/
}
```

The same should be added for ARP packet drop.

3. Add C++ code for AgentJ.

   The method of *notifyRoutingProtocol* of *YourProtocol* should be (or be created) in ns2.34/agentj/core/src/main/c/agentj/, with name YourProtocolRouter.h:

```
#ifndef _YOURPROTOCOLROUTER
#define _YOURPROTOCOLROUTER

#include <set>
#include "AgentJRouter.h"

class YourProtocolRouter : public AgentJRouter {
    public:
        YourProtocolRouter();
        ~YourProtocolRouter();

        void receivePacket(Packet *p,Handler *handle);
        bool ProcessCommands(int argc, const char* const* argv);
        void notifyRoutingProtocol(Packet* p);
};  // end class

#endif
```

   and YourProtocolRouter.cpp:

```
void YourProtocolRouter::notifyRoutingProtocol(Packet *p) {
     JavaVM *jvm =AgentjVirtualMachine::getVM(NULL);
     JNIEnv *currentEnv;
     int   res = jvm->AttachCurrentThreadAsDaemon((void**)&currentEnv, NULL);

     if (res < 0) {
          PLOG(PL_FATAL, "YourProtocolRouter::notifyRoutingProtocol, Attach
               failed. Aborting\n");
          exit(0);
     }
```

```
    jclass routerClass = currentEnv->GetObjectClass(javaRouter_);
    if (routerClass == NULL){
        PLOG(PL_FATAL, "routerClass not found\n");
        abort();
    }

    //to retrieve nextHop
    struct hdr_cmn *ch = HDR_CMN(p);

    //the payload of the ns2 packet = YourProtocol message
    unsigned char* yourprotocolPacket = p->accessdata();
    jmethodID methodID = currentEnv->GetMethodID(routerClass, "deliveryFailed", "([BI)V");
    if (methodID == NULL){
        PLOG(PL_FATAL, "deliveryFailed method not found\n");
        abort();
    }

    int l = p->datalen();
    //need to convert jbyte* = unsigned char* into an array for java
    jbyteArray buf = currentEnv->NewByteArray(l);
    currentEnv->SetByteArrayRegion(buf, 0, l, (jbyte*) yourprotocolPacket);

    PLOG(PL_DETAIL, "Calling deliveryFailed of Java agent %d\n", agentjref_->
        addr());
    currentEnv->CallIntMethod(javaRouter_, methodID, buf, (int) ch->next_hop_
        );
    if (currentEnv->ExceptionCheck()){
        currentEnv->ExceptionDescribe();
        currentEnv->ExceptionClear();
        abort();
    }
    currentEnv->DeleteLocalRef(routerClass);
}
```

4. Add Java code for AgentJ-NS2 interface.

   In the AgentJ agent of Java code (normally *Ns2YourProtocolAgent.java*), add the *deliveryFailed* method to deal with the dropped packet:

```
public void deliveryFailed(byte[] message, int nextHop){
  AgentJVirtualMachine.setCurrentNodeforAgent(this);
  logger.fine("Delivery failed, packet sent back by ns2");

  /*** handle the dropped packet here*/
}
```

2 Information of Queue Length and other Layers

This section introduces how to obtain the queue length information for the routing protocol.

1. In YourProtocol.h, declare the Queue and Link Layer object

```
NsObject *ll; //link layer
CMUPriqueue* ifq; //outgoing queue
```

2. In YourProtocol.cc, initiate the related object in YourProcol.cc::command.

```
int
YourProtocol::command(int argc, const char*const* argv) {
        /* ...*/
        else if (strcasecmp(argv[1], "add-ll") == 0) {
                TclObject* obj;
                if( (obj = TclObject::lookup(argv[2])) == 0) {
                        fprintf(stderr, "YourProtocolAgent: %s lookup of %s
                            failed\n", argv[1], argv[2]);
```

```
                    return TCL_ERROR;
                }
                ll = (NsObject*) obj;
                if( (obj = TclObject::lookup(argv[3])) == 0) {
                        fprintf(stderr, "YourProtocolAgent: %s lookup of %s
                            failed\n", argv[1], argv[3]);
                    return TCL_ERROR;
                  }
                ifq = (CMUPriQueue *) obj;
                return TCL_OK;
        }
}
```

3. Initiate the object in tcl script.

   In the simulation .tcl script, add the following code during node initialization:

```
set rt($i) [$node_($i) agent 255] #get the routing protocol
$rt($i) add_ll [$node_($i) set ll_(0)] [$node_($i) set ifq_(0)]; #initiate the
     ifq info
```

4. Get the queue length information from the routing protocol.

   The length can be obtained in the routing protocol YourProtocol.cc by using

```
ifq->prq_length();
```

The same method can be applied to any other layers, such as MAC (mac-802_11.h/cc), LL, NETIF

Acknowledgment

Thanks Antonin Bas and Ulrich Herberg for providing helpful information in refining this document.