Pearls in Life

Technical reflections and records of thoughts

- RSS

Simulate Random MAC Protocol in NS2 (Part III)

Dec 15th, 2013

Now we have the simulation script, and also added our protocol to the NS2 simulator, which is still a placeholder. Now we're going the actually implement our own random MAC protocol.

Protocol Description

According to the project specification, when sending out an packet, our protocol is supposed to send out `X` copies of the packet at random time before sending out next packet. As long as the receiver receive at least one of the `X` duplicates, we say this packet was successfully delivered.

Protocol Parameters

From the protocol description, it's obvious that we need to know:

- How many copies to send for one packet? I.e., the `X`
- The interval of sending packet from up layer, so that we can schedule resending before up layer pass down the next packet.

So we add two class variables in `mac/rmac.h`

```
1 int repeatTx_;
2 double interval_;
```

And in the constructor function of `RMAC` class, we need to bind the variables through TCL so that we can pass values in TCL script.

```
1 bind("repeatTx_", &repeatTx_);
2 bind("interval_", &interval_);
```

TCL Object Binding

We also need to let TCL runtime to *know* our protocol. That is, when we write this in TCL script.

```
1 set val(mac) Mac/RMAC
```

TCL runtime would have to know the corresponding class of the `Mac/RMAC`.

Since we copied code from `mac-simple.cc` and also made the changes, this part has been done, but let's just review the code snippet that does the binding.

```
1 static class RMACClass : public TclClass {
2     public :
3         RMACClass() : TclClass("Mac/RMAC") {}
4         TclObject* create(int, const char* const*) {
5             return (new RMAC());
6         }
7 } class_rmac;
```

Here the `Mac/RMAC` string will be our protocol name.

Interaction with Adjacent Layer

The most important function in any NS2 MAC protocol is the `recv` function. It's the interface to upper (Network Layer) and also lower (Physical Layer) layers.

The `recv` of our MAC protocol will look like this.

```
1  void
2  RMAC::recv(Packet *p, Handler *h) {
3
4    struct hdr_cmn *hdr = HDR_CMN(p);
5    /* let RMAC::send handle the outgoing packets */
6    if (hdr->direction() == hdr_cmn::DOWN) {
7        sendDown(p,h);
8    }
9      else {
10         sendUp(p,h);
11     }
12 }
```

Here we first get the header of the packet, and check it's directory. `hdr_cmn::DOWN` means this packet is from upper layer, and we need to send it out. `hdr_cmn::UP` means this packet is from lower layer (received packet), we need to deliver it to upper layer.

Repeat Sending

The key part of our MAC protocol is to repeated send multiple copies when sending out a packet. So we need to mainly modify the `sendDown` function.

```
1  double max_delay = 0;
2
3  // generate repeatTx_ number of random delays
4  double* delays = new double[repeatTx_];
5  for (int i = 0; i < repeatTx_; i++) {
6      delays[i] = (rand() % 100) / 100.0 * interval_;
7      if (delays[i] > max_delay) {
8          max_delay = delays[i];
9      }
10 }
11
12 // use dummy tx handler for first repeatTx_-1 packets
13 for (int i = 0; i < repeatTx_; i++) {
14     if (delays[i] != max_delay) {
15         Scheduler::instance().schedule(&resendHandler_, (Event*)p->copy(), delays[i]);
16     }
17 }
18 delete delays;
19
20 waitTimer->restart(max_delay);
21 if (rx_state_ == MAC_IDLE ) {
22     // we're idle, so start sending now
23     sendTimer->restart(max_delay + ch->txtime());
24 } else {
25     // we're currently receiving, so schedule it after
26     // we finish receiving
27     sendTimer->restart(max_delay + ch->txtime()
28             + HDR_CMN(pktRx_)->txtime());
29 }
```

We first generate `repeatTx_` number of delays before next interval. Except for the `max_delay`, which will be the last copy to send, we use the `Scheduler` to resend the duplicated packets, and for last packet, we just use the timer scheme of `SimpleMac`.

Here is the `resendHander_` looks like.

```
1 void
2 RMACResendHandler::handle(Event* p) {
3     mac_->resend((Packet*) p);
4 }
5
6 void
7 RMAC::resend(Packet* p) {
8     downtarget_->recv(p, NULL);
9 }
```

You can find the complete code for `rmac.cc` and `rmac.h` here.

Change History

- **Sun Dec 15 12:54:11 2013**

Posted by Jinghao Shi Dec 15th, 2013

Related Posts

- Simulate Random MAC Protocol in NS2

new post

View on Github

Tagged with: mac, ns2

In Category: network

(Part I)
- OS161 Swapping
- Simulate Random MAC Protocol in NS2 (Part II)
- OS161 fork System Call
- OS161 Tool Chain Setup

« Simulate Random MAC Protocol in NS2 (Part II) Simulate Random MAC Protocol in NS2 (Part IV) »

Comments

Did you know...

Only Chuck Norris can prevent forest fires.

Blog Stats

90 hits

Popular Posts

- OS161 TLB Miss and Page Fault
- OS161 Coremap
- OS161 fork System Call
- OS161 File System Calls
- OS161 execv System Call

Recent Posts

- Tap Notification to Send Email
- Sum of N Largest Numbers in Google Spreadsheet
- OS161 Tool Chain Setup
- Simulate Random MAC Protocol in NS2 (Part IV)
- Simulate Random MAC Protocol in NS2 (Part III)

Tags

AlertDialog C DownloadManager MLFQ SO_REUSEADDR addrspace backup beamer bind binutils bison bmake c c++ caption cat chrome close compare_xml coremap cron cscope ctags dispatch django dropbox dup2 email exit fd_set fdtable fedora figure function gcc gdb getifaddrs glic google calendar graphicspath graphicx grep heap intent jekyll lfs libpopller linux lock lseek mac migration notification ns2 open ota page fault page table pdflatex pid plugin popular posts prototype python query read readline rsync ruby rwlock sbrk scheduling scp sed select service signapk socket sort spreadsheet ssh stack swap synchronization

sys161 syscall table tcl texinfo tlb toolchain undodir vimrc vm waitpid wordpress write yacc yum sty zip

Categories

Android (6)
errors (9)
latex (8)
linux (7)
network (8)
octopress (5)
os161 (25)
tricks (1)
vim (4)

GitHub Repos

- latex.template.slides

  Beamer template

- dotfiles

  Various configuration files

- jhshi.github.com

  Personal Blog

- latex.template.acm-proc

  Latex template for course project proposals, reports.

- latex.template.homework

  Latex template for homework.

@jhshi on GitHub

Stack Overflow

Page Link