NS2 discrete event-driven philosophy (Scheduler, Handler, Event, Timer)

ytzhang1979

Posts: n/a

Tag: NS2, tools, timer, random, class, delay, working

NS2 is a discrete event-driven simulation mechanism, this literature everywhere are talking about but never talked about the idea. This paper attempts to explore discrete event-driven departure from the several **NS2** Basic class how it was.

First, the relationship between the Scheduler, Handler and Event class.
NS2 in the event (Event) is a basic scheduling unit, for example, to send a packet, receives a Packet. Each Event has its own processing tools, this tool is a Handler class object handler_. Handler contains only one function, described the Event processing methods, namely handle (Event * e).

Given an event, the Scheduler will schedule (Handler * h, Event * e, double delay) function is called, the function set Event The uid_, and set up to handle the event handler: e-> handler_ = h, then the event into the event queue maintained by the Scheduler. In general, to handle the event Handler are generated the event of the entity's own, so we can often see the schedule function when calling * h use this pointer.

NS2 run the simulation, Scheduler :: run () function constantly running to deal with the events in the queue, the queue in the event out individually dequeue (), and then run Scheduler :: dispatch (Event * p, double time) function, an event from the queue popped, call its corresponding Handler handle () function handles it.

This completes an event sent from generation to queue to be processed in the process.

Next, look at the the role of TimerHandler.
Timer **(Timer)** is a key means of **NS2** simulation event, which is used to set up a future event, when it will dispatch for processing. **Timer** are TimerHandler base class derived class. TimerHandler interact with the Scheduler function is sched (double delay), it calls the Scheduler :: schedule (Handler * h, Event * e, double delay) insert events into the queue after a delay time, Handler set for TimerHandler itself (with this) .

after the delay time, the Scheduler will dequeue from the event queue of the event, call processing functions, that is, TimerHandler :: handle (). The handle () In addition to the state to do some timer is set, the core of the processing is done by the virtual function expire (Event * e). The dynamic characteristics of C + + is not difficult to understand expire () will be in a variety of custom the **Timer** (also is TimerHandler the derived class) be rewritten to achieve the different treatment of the **Timer.**

Look at a specific example
 NIST Wimax module DlTimer example, the **Timer** is used to trigger a downlink frame generation:

```
class DlTimer: public TimerHandler {
public:
DlTimer (Mac802_16 * m): TimerHandler () {m_ = m;}

void expire (Event * e);
private:
Mac802_16 * m_;
};
```

MAC entities using the **Timer** Mac802_16BS :: init () function to open the **Timer:**

```
double stime = getFrameDuration () + Random :: uniform (0, getFrameDuration ());
DL_ timer _-> sched (stime);
```

This timer is timing stime after the trigger event. Scheduler when to do about it? Know by the above will be by

Schedule () function is called the event handler handles the event. This event handler, what is it? Sched (), the code can be traced back to an inline function:

```
of inline void _sched (double delay) {
(Void) Scheduler :: instance (). Schedule (this, & event_, delay);
}
```

The function set event_ Handler for this, is to call schedule () object, and the object back to go back to exactly dl_ a **timer** _! So we know that, after the timer expires will call dl_ **timer** _ handle () method of doing things, the DlTimer is the direct successor to the TimerHandler handle () method, and TimerHandler :: handle () function by virtual function expire (Event * e) to do things. In DlTimer inherited TimerHandler just rewrite expire function:

```
the void DlTimer :: expire (Event * e)
{
m_-> start_dlsubframe ();
}
```

Things became clear dl_ **Timer** _ when Scheduler trigger an event, this event triggered MAC802_16 class start_dlsubframe () action, this function is the "Start downlink subframe. This completes the completion of the whole process of a scheduling a **Timer.**

« The NS2 in packet Packet Analysis | VS2010 call the the MATLAB method (reproduced + Fixed) »

Related

NS2 discrete event-driven principle Scheduler, Handler, Event	NS2	NS2, discrete event-driven philosophy, Scheduler	whatsao
MySQL scheduled task (event scheduler) (Event Scheduler)	web	web, mysql	cindyqh
Analysis of NS2 event scheduling process	Uncategorized	expansion, Insert, tool, delay	wwwkkkyyy
JavaScript event handler for the event parameters	js	js	qq396667738
J2ME event handler	J2ME	j2me, programming, phone game, action, Mobile	sunjunlove

Contact Us-Disclaimer-Top