Network Simulator 2: Introduction

Presented by Ke Liu
Dept. Of Computer Science
SUNY Binghamton

Spring, 2006
NS-2 Overview
NS-2

- Developed by UC Berkeley
- Maintained by USC
- Popular simulator in scientific environment
- Other popular network simulators
  - QualNet: based on GloMoSim
  - Others: GloMoSim, OPNET, etc

© Ke Liu, SUNY-Binghamton, 2006
NS2 Goals

• To support networking research and education
  – Protocol design, traffic studies, etc.
  – Protocol comparison;
  – New architecture designs are also supported.

• To provide collaborative environment
  – Freely distributed, open source;
  – Increase confidence in result
Two Languages: C++, OTcl

OTcl: short for MIT Object Tcl, an extension to Tcl/Tk for object-oriented programming.

- Used to build the network structure and topology which is just the surface of your simulation;

- Easily to configure your network parameters;

- Not enough for research schemes and protocol architecture adaption.

© Ke Liu, SUNY-Binghamton, 2006
Two Languages (Con’t)

C++: Most important and kernel part of the NS2

- To implement the kernel of the architecture of the protocol designs;
- From the packet flow view, the processes run on a single node;
- To change or “comment out” the existing protocols running in NS2;
- Details of your research scheme.

© Ke Liu, SUNY-Binghamton, 2006
Why 2 Languages?

- 2 requirements of the simulator
  - Detailed simulation of Protocol: Run-time speed;
  - Varying parameters or configuration: easy to use.

- C++ is fast to run but slower to code and change;

- OTcl is easy to code but runs slowly.
Protocols/Models supported by NS2

• Wired Networking
  – Routing: Unicast, Multicast, and Hierarchical Routing, etc.
  – Transportation: TCP, UDP, others;
  – Traffic sources: web, ftp, telnet, cbr, etc.
  – Queuing disciplines: drop-tail, RED, etc.
  – QoS: IntServ and Diffserv

• Wireless Networking
  – Ad hoc routing and mobile IP
    – Routing Protocol: AODV, DSDV, DSR, etc.
    – MAC layer Protocol: TDMA, CDMA(?), IEEE Mac 802.x, etc.
    – Physical layers: different channels(?), directional attena

• Sensor Networks

© Ke Liu, SUNY-Binghamton, 2006
Researcds based on NS2

- Intserv/Diffserv (QoS)

- Multicast: Routing, Reliable multicast

- Transport: TCP Congestion control

- Application: Web caching Multimedia

- Sensor Networks: LEACH, Directed Diffusion, etc.

- etc.
NS2 research actions

- **NS2**: the simulator itself, now version: ns-2.29
  We will work with the part mostly.

- **NAM**: Network animator. Visualized trace tool (not really).
  My recommendation is that "Don’t use nam at all".

- **Pre-processing**:
  Traffic and topology generators

- **Post-processing**:
  Simple trace analysis, often in Awk, Perl (mostly), or Tcl
Living under NS2
The NS2 Directory Structure
Warning

- Try to avoid using ns2 with version before 2.27

- DO NOT use gcc 4.x, suggestion: gcc3.3

- If you work with MAC layer protocols, please be careful for the versions
A Simple Simulation

We just need one Tcl script to do so.
A Simple Simulation, part 1: set up

```tcl
set ns [new Simulator]
$ns use-scheduler Heap
$ns color 1 Blue
$ns color 2 Red

set nf [open out.nam w]
$ns namtrace-all $nf
set tf [open trace.tr w]
$ns trace-all $tf

proc finish {} {
    global ns nf tf
    $ns flush-trace
    #Close the NAM trace file
    close $nf
    #Close the ns2 trace file
    close $tf
    exit 0
}
```

© Ke Liu, SUNY-Binghamton, 2006
A Simple Simulation, part 2: n/w structure

#Create four nodes

set n0 [\$ns node]
set n1 [\$ns node]
set n2 [\$ns node]
set n3 [\$ns node]

#Create links between the nodes

\$ns duplex-link \$n0 \$n2 2Mb 10ms DropTail
\$ns duplex-link \$n1 \$n2 2Mb 10ms DropTail
\$ns duplex-link \$n2 \$n3 1.7Mb 20ms DropTail

#Set Queue Size of link (n2-n3) to 10
\$ns queue-limit \$n2 \$n3 10

#Give node position (for NAM)

\$ns duplex-link-op \$n0 \$n2 orient right-down
\$ns duplex-link-op \$n1 \$n2 orient right-up
\$ns duplex-link-op \$n2 \$n3 orient right

#Monitor the queue for link (n2-n3). (for NAM)

\$ns duplex-link-op \$n2 \$n3 queuePos 0.5

© Ke Liu, SUNY-Binghamton, 2006
A Simple Simulation, part 3: Transport and Traffic

#Setup a TCP connection: from node 0 to node 3

set tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
$tcp set fid_ 1

#Setup a FTP over TCP connection

set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP ;#for Nam

#Setup a UDP connection: from node 1 to node 3

set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $udp $null
$udp set fid_ 2
A Simple Simulation, part 4: traffic

#Setup a CBR over UDP connection

set cbr [new Application/Traffic/CBR]
cbr attach-agent $udp $cbr
set type_ CBR $cbr ;#used by Nam

set packet_size_ 1000 $cbr
set rate_ 1mb $cbr
set random_ false ;#generating traffic periodically

#Schedule events for the CBR and FTP agents

$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 4.0 "$ftp stop"
$ns at 4.5 "$cbr stop"

#Detach tcp and sink agents (not really necessary)

$ns at 4.5 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n3 $sink"

#Call the finish procedure after 5 seconds of simulation time

$ns at 5.0 "finish"

#Run the simulation

$ns run

© Ke Liu, SUNY-Binghamton, 2006
Steps in writing a simulating script

- Create the event scheduler
- Turn on tracing
- Create network
- Setup routing
- Insert errors
- Create transport connection
- Create traffic
- Transmit application-level data

© Ke Liu, SUNY-Binghamton, 2006
The trace file

- Turn on tracing on specific links
  
  ```bash
  $ns_ trace-queue $n0 $n1
  ```

- Each line in the trace file is in the format:
  
  `<event> <time> <from> <to> <pkt-type> <pkt-size> <flags> <fid> <src.port> <dst.port> <seq> <unique pkt id>`

- Trace example:
  
  ```
  + 1 0 2 cbr 210 ------- 0 0.0 3.1 0 0
  - 1 0 2 cbr 210 ------- 0 0.0 3.1 0 0
  r 1.00234 0 2 cbr 210 ------- 0 0.0 3.1 0 0
  ```

- Event: `s` send, `r` receive, `+` enqueue, `−` dequeue, `d` drop, `f` forward,
The network Topology

© Ke Liu, SUNY-Binghamton, 2006
The Node Architecture
The Packet Flow
Extending to NS2
Class Hierarchy in NS2 (Partial, C++ code)
Create New Component for NS2

Your research needs you to do so, no escaping(crying!!!).

- Extending ns in Otcl
  source your changes in your simulation scripts

- Extending ns in C++
  - Change Makefile (if created new files)
  - make depend
  - recompile
  - Makefile is in the ”ns-2.29” directory
Adding New Class

otcl

bind()     TclClass()     command()

C++

© Ke Liu, SUNY-Binghamton, 2006
C++ Code Architecture of a Mobile Node (DSDV)
A component at some layer of each node

- Up target: except the highest layer component
- Down target: even the lowest layer component
- Timers: triggering some actions
- Send: either to up or down neighbor
- Recv: from either up or down neighbor
Assignments

• Finish the following sections of the tutorial on link http://nile.wpi.edu/NS/
  – Purpose
  – Overview
  – Basics
  – Post Simulation
  – Extending NS: where to Find What?

• Understand how to simulate the DSR and DSDV routing with ns2. You may use the NS2 manual.

Project Assignment 1: requirement

- Wireless simulation: using **BOTH** DSDV and DSR routing;

- You should trace **Agent, Routing Agent, Mac**;

- You DON’T need to create nam trace;

- Create 9 **CBR** flows upon **UDP** transportation to one node.

- CBR Packet Size: 1024 Bytes, interval 1.0 sec, Simulation(Traffic) Time: 120 sec;

- No other parameter’s default value should be changed;

- All 9 flows start at the same time, so as end.
How to code a new Routing Agent
Steps

• Packet header design

• Packet header globalizing

• Routing Agent design

• Timer Design

• Tcl script commands Design

• Tcl linkage

© Ke Liu, SUNY-Binghamton, 2006
Objective: GPSR routing

- MobiCom 2000

- B. Karp and H.T. Kung, Harvard University

- **GPSR: Greedy Perimeter stateless Routing for Wireless Networks**

- Stateless Point-to-Point Routing based on Location information
GPSR: Greedy Forwarding

Source

Sink

© Ke Liu, SUNY-Binghamton, 2006
GPSR: GF Failure
GPSR: Void Problem

© Ke Liu, SUNY-Binghamton, 2006
GPSR: Perimeter (Face) Routing

Planarizing the Graph (mapped from network), routing around faces to sink
Packet Header

struct hdr_gpsr {
  u_int8_t type_;  

static int offset_;  
inline static int& offset() {return offset_;} 
inline static struct hdr_gpsr* access(const Packet *p) {
    return (struct hdr_gpsr*) p->access(offset_);  
}  
};
Packet Header

```c
struct hdr_gpsr_hello {
    u_int8_t type_;  // My geo info
    float x_;  // My geo info
    float y_;  
    inline int size() {  
        int sz =
            sizeof(u_int8_t) +
            2*sizeof(float);
        return sz;
    }
};
```

© Ke Liu, SUNY-Binghamton, 2006
struct hdr_gpsr_query {
    u_int8_t type_;           //The sink geo info
    float x_;                
    float y_;                
    float ts_;               //time stampe
    int hops_;               
    u_int8_t seqno_;         //query sequence number

    inline int size() {
        int sz =
            2*sizeof(u_int8_t) +
            3*sizeof(float) +
            sizeof(int);
        return sz;
    }
};

© Ke Liu, SUNY-Binghamton, 2006
Packet Header

```c
struct hdr_gpsr_data {
    u_int8_t type_;  // Greedy forwarding or Perimeter Routing
    u_int8_t mode_;  // Greedy forwarding or Perimeter Routing
    float sx_;       // the geo info of src
    float sy_;       // the geo info of src
    float dx_;       // the geo info of dst
    float dy_;       // the geo info of dst
    float ts_;       // the originating time stamp

    inline int size(){
        int sz =
            2*sizeof(u_int8_t) +
            5*sizeof(float);
        return sz;
    }
};
```

© Ke Liu, SUNY-Binghamton, 2006
union hdr_all_gpsr {
  hdr_gpsr gh;
  hdr_gpsr_hello ghh;
  hdr_gpsr_query gqh;
  hdr_gpsr_data gdh;
};
Routing Agent

class GPSRAgent : public Agent {
    MobileNode *node_;  
    PortClassifier *port_dmux_;  
    ...
    GPSRNeighbors *nblist_; //neighbor list (routing table)
    ...
    GPSRHelloTimer hello_timer_;  
    double hello_period_;  
    void hellomsg();  
    void recvHello(Packet*);  
    ...
    void GetLocation(double*, double*);  
    ...
    void forwardData(Packet*);  

© Ke Liu, SUNY-Binghamton, 2006
Routing Agent

protected:
    void hello tout();
    ...

public:
    GPSRAgent();
    int command(int, const char* const*);
    void recv(Packet*, Handler*);
};
Routing Table

- Maintaining the routing information;
- Able to decide a next hop when required;
- It is a part of the routing agent;
- It may just a local object, not a global object (NsObject)
Initializing: variables binding

- The configuration of routing agent can be done through the tcl script by calling "bind" function, which does not requiring re-compiling when we change the value of it (Example Later);

```cpp
GPSRAgent::GPSRAgent() : Agent(PT_GPSR),
    hello_timer_(this){
    ...
    bind("hello_period_", &hello_period_);
    ...
}
```
Packet forwarding: on Receiving

- According to the packet type, taking different actions

```cpp
void GPSRAgent::recv(Packet *p){
    if(iph->saddr() == my_id_){ // a packet generated by myself
        if(cmh->num_forwards() == 0){ // coming from higher layer
            ... // append some routing information
            forwardData(p); // forward it
            return;
        }
    }
    else if(cmh->num_forwards() > 0){ // routing loop
        ... // typically drop it
    }
}
```
Packet forwarding: Classifying

...  
if(cmh->ptype() == PT_GPSR){
    struct hdr_gpsr *gh = HDR_GPSR(p);
    switch(gh->type_){
    case GPSRTYPE_HELLO:
        recvHello(p);
        break;
    ...
    }
} else { //Data packet coming from others
    iph->ttl_--;
    if(iph->ttl_ == 0){ //ttl expire
        drop(p, DROP_RTR_TTL);
        return;
    }
    forwardData(p);
}
Packet forwarding: on forwarding

void GPSRAgent::forwardData(Packet *p){
  struct hdr_cmn *cmh = HDR_CMN(p);
  struct hdr_ip *iph = HDR_IP(p);

  if(cmh->direction() == hdr_cmn::UP &&
      ((nsaddr_t)iph->daddr() == IP_BROADCAST ||
       iph->daddr() == my_id_)){ //a packet sent to me !
    port_dmux_->recv(p, 0); //forward to higher layer
    return;
  }
}
Packet forwarding: on forwarding

else{
    ...
    //update the routing info in the packet header
    cmh->direction() = hdr_cmn::DOWN;  //to lower layers
    cmh->addr_type() = NS_AF_INET;
    cmh->next_hop_ = nexthop;  //change the next hop
    send(p, 0);
}

© Ke Liu, SUNY-Binghamton, 2006
Agent Tcl script Commands

int GPSRAgent::command(int argc, const char*const* argv){
    ... if(strcasecmp(argv[1], "log-energy") == 0){
        if(node_ && node_->energy_model()){
            //record the energy
        }
        return TCL_OK;
    }
    TclObject *obj;
    if ((obj = TclObject::lookup (argv[2])) == 0){
        fprintf(stderr, "%s: %s lookup of %s failed\n", 
            __FILE__, argv[1], argv[2]);
        return (TCL_ERROR);
    }
}

© Ke Liu, SUNY-Binghamton, 2006
Agent Tcl script Commands

```c
if (strcasecmp (argv[1], "node") == 0) {
    node_ = (MobileNode*) obj;
    return (TCL_OK);
}
else if (strcasecmp (argv[1], "port-dmux") == 0) {
    port_dmux_ = (PortClassifier*) obj; // (NsObject *) obj;
    return (TCL_OK);
}
```

© Ke Liu, SUNY-Binghamton, 2006
Globalizing

static class
GPSRHeaderClass : public PacketHeaderClass{
public:
  GPSRHeaderClass() : PacketHeaderClass("PacketHeader/GPSR",
      sizeof(hdr_all_gpsr)){
      bind_offset(&hdr_gpsr::offset_);
  }
}class_gpsrhdr;

static class GPSRAgentClass : public TclClass {
private:
  GPSRAgentClass() : TclClass("Agent/GPSR"){}
  TclObject *create(int, const char*const*){
      return (new GPSRAgent());
  }
}class_gpsr;

© Ke Liu, SUNY-Binghamton, 2006
Routing Protocol Control: Timers

class GPSRHelloTimer : public TimerHandler {
public:
    GPSRHelloTimer(GPSRAgent *a) : TimerHandler() {a_=a;}
protected:
    virtual void expire(Event *e);
    GPSRAgent *a_;
};
void GPSRHelloTimer::expire(Event *e){
    a_->hellomsg();
}
void GPSRAgent::hellomsg(){
    ... 
    hello_timer_.resched(hello_period_);
}
Changes to NS2: Packet type: *common/packet.h*

```c
enum packet_t {
    PT_TCP,
    ...
    PT_GPSR, //add
    ...
};

class p_info{
    p_info(){
        name_[PT_TCP] = "tcp";
        ...
        name_[PT_GPSR] = "gpsr"; //add
        ...
    }
};
```

© Ke Liu, SUNY-Binghamton, 2006
Changes to NS2: Trace support *trace/cmu-trace.cc*

```cpp
void CMUTrace::format(Packet *p, const char* why) {
    ... 
    case PT_TCP:
        break; 
    ... 
    case PT_GPSR:  //add
        break;  //add
    default: 
        ... 
}
```

© Ke Liu, SUNY-Binghamton, 2006
Changes to NS2: Tcl Library `tcl/lib/ns-packet.tcl`

```tcl
foreach prot {
    AODV
    DSR
    ...
    GPSR ;# add
    ...
}
```
Make it running: Makefile

OBJ_CC =

...$(OBJ_STL) \n
gpsr/gpsr.o   #add
Simulation: *wireless-gpsr.tcl*

... Agent/GPSR set hello_period_ 5.0 ;#Hello message period ...

for {set i 0} {i < $opt(nn)} {incr i} {
    $ns_ at [expr $opt(stop) - 0.1] "$ragent_($i) log-energy"
}
Trace Analyzing

s  5.054902765   _54_   RTR    ---   112   gpsr 29  [0 0 0 0]
    [energy 999.975614]  -------  [54:255 -1:255 32 0]

r  5.055969912   _44_   RTR    ---   112   gpsr 29  [0 ffffffff 36 800]
    [energy 999.975102]  -------  [54:255 -1:255 32 0]
Thank You!

Any Questions?

My email: kliu@cs.binghamton.edu