The Network Simulator *ns–2*

Jörg Widmer
International Computer Science Institute
widmer@icsi.berkeley.edu
March 22, 2000

Overview

- Introduction
- Simulator architecture
- Example simulation
- Support for wireless scenarios
- Other interesting features

Why Network Simulations

- Necessity to perform both real-world experiments *and* simulations
- Standard platform for protocol development
- But: capturing all possible details of real-world scenarios is (nearly) impossible

Features

Simulator for the design of network protocols and architectures

- Event driven
- Extensible framework
- Multi-protocol simulator

- Object-Oriented
- Modular

- Developed by UCB, LBNL, ISI/USC, CMU, Sun, Xerox PARC, . . .
- About 60K lines of code + 40K lines of test suites/examples

Architecure

- Nodes
  - Agents
  - Classifiers
- Links
  - Queue
  - Delay
- Applications
- Protocols
- Packets
- Loss modules
- Math support (random number generation, integrals)
- . . .

C++ and OTcl

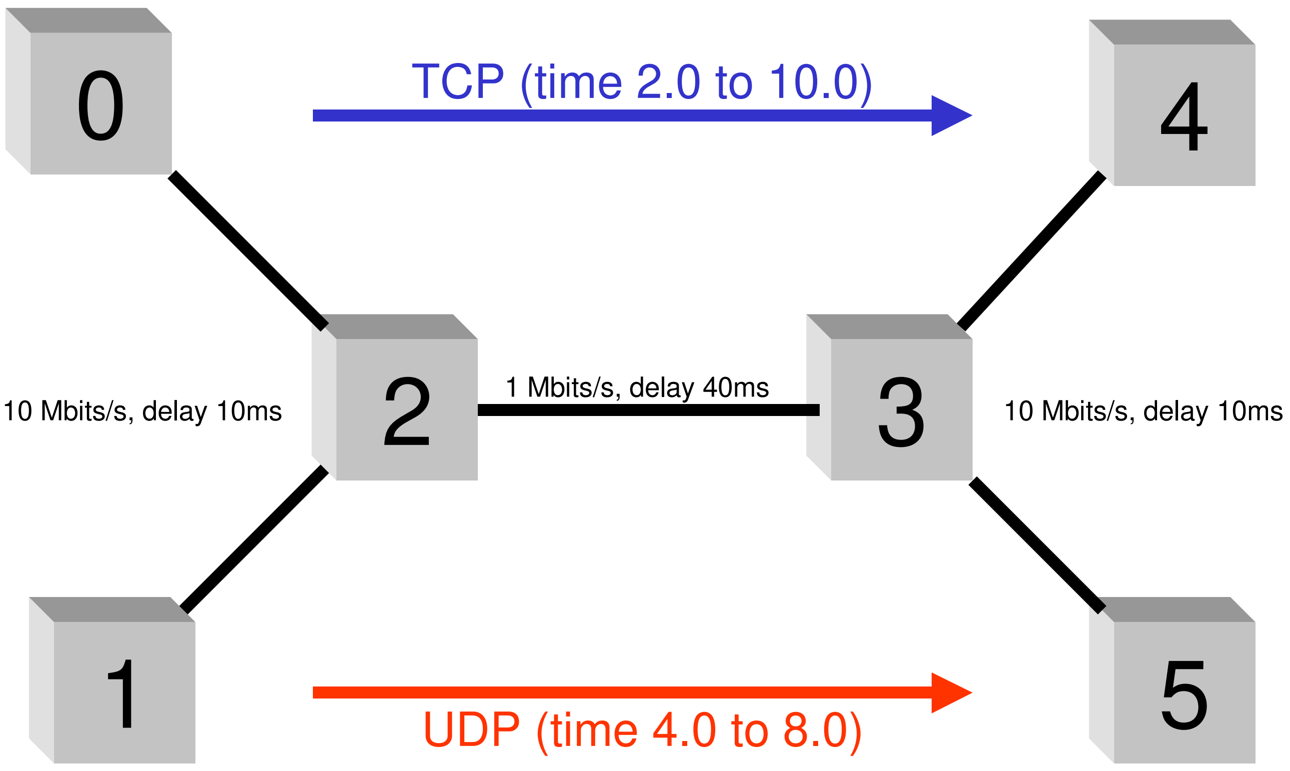
- Split-language programming
  (mapping of classes, split objects, . . .)
- C++ for the core components
  (low level event processing, packet forwarding, . . .)
- OTcl for control operations
  (to build the simulation scenario, model dynamic configurations, . . .)

- Necessary to know both languages
- More difficult to debug

Creating a Simulation

- Create the event scheduler
- Create the network topology
- Specify traffic patterns
- Insert errors, modify network conditions, . . .
- Tracing

Example Simulation



Simulation Script

```
set ns [new Simulator]

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

   ...

$ns at 10.0 "$ns flush-trace; close $f; close $nf; \
            exit 0"
$ns run
```

Creating the Network Topology

```
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

$ns duplex-link $n0 $n2  10Mb 10ms DropTail
$ns duplex-link $n1 $n2  10Mb 10ms DropTail
$ns duplex-link $n2 $n3   1Mb 40ms DropTail
$ns duplex-link $n3 $n4  10Mb 10ms DropTail
$ns duplex-link $n3 $n5  10Mb 10ms DropTail
```

Defining the Communication Pattern

```
set tcp [new Agent/TCP]
$tcp set packetSize_ 1000
set sink [new Agent/TCPSink]
$ns attach-agent $n1 $tcp
$ns attach-agent $n5 $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 2.0 "$ftp start"
```

Similar: CBR flow from time 4.0 to 8.0 with a bitrate of 900KBit/s

Trace file format

event	time	nodes: from	to	packet: type	size	flags	flow ID	address: src	dest	seq-no	uid

```
+ 4.053333 1 2 cbr 1000 ------- 1 1.0 5.0 6 415
- 4.053333 1 2 cbr 1000 ------- 1 1.0 5.0 6 415
r 4.054704 0 2 tcp 1000 ------- 0 0.0 4.0 209 411
+ 4.054704 2 3 tcp 1000 ------- 0 0.0 4.0 209 411
r 4.055244 1 2 cbr 1000 ------- 1 1.0 5.0 5 412
+ 4.055244 2 3 cbr 1000 ------- 1 1.0 5.0 5 412
r 4.05552 3 4 tcp 1000 ------- 0 0.0 4.0 198 384
+ 4.05552 4 3 ack 40 ------- 0 4.0 0.0 198 416
- 4.05552 4 3 ack 40 ------- 0 4.0 0.0 198 416
r 4.057552 4 3 ack 40 ------- 0 4.0 0.0 197 413
```

Visualization with *NAM*

- Packet traces presented as graphical animation
- Additional NAM information in tcl trace files (node color, . . .)
- Captures simulation dynamics
- “Intuitive” feel for what the protocol is doing
- Trace files, time sequence graphs, etc. are still necessary for in-depth analysis

Creating Your Own Components

- Look at existing components
- Try to reuse existing modules
- Decide about inheritance, fill in functions
- Linkage to OTcl, implement complementary OTcl classes/functions

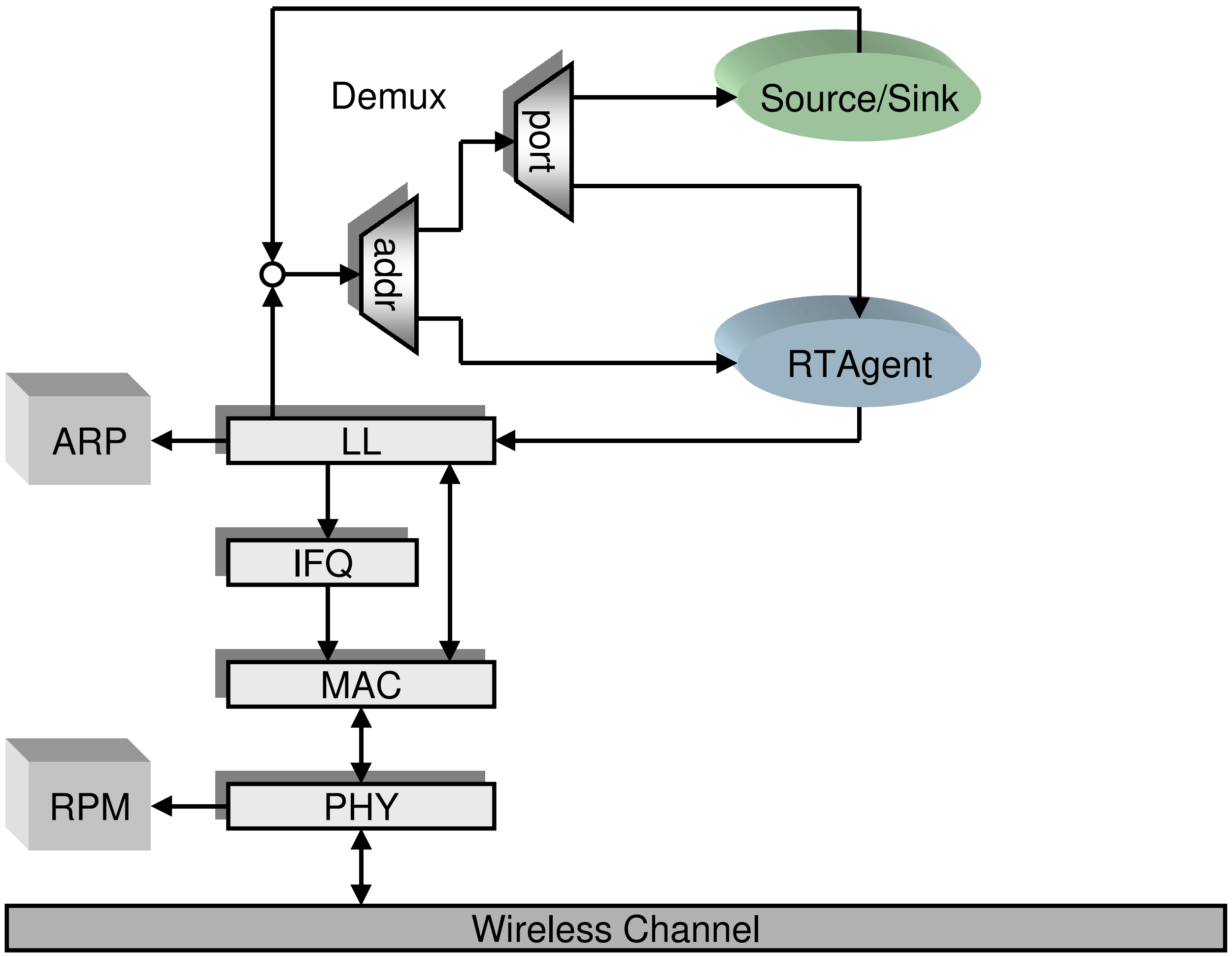
Simulation of Wireless Networks . . .

- Node movement
- Ad-hoc routing protocols: DSDV, DSR, AODV, TORA
- Support for Mobile IP
- Elaborate wireless model
  - Radio Propagation Model (Friss-space, two ray ground)
  - Antenna (omnidirectional)
  - MAC Layer (IEEE 802.11, CSMA)
  - Address Resolution Protocol (ARP)

. . . But

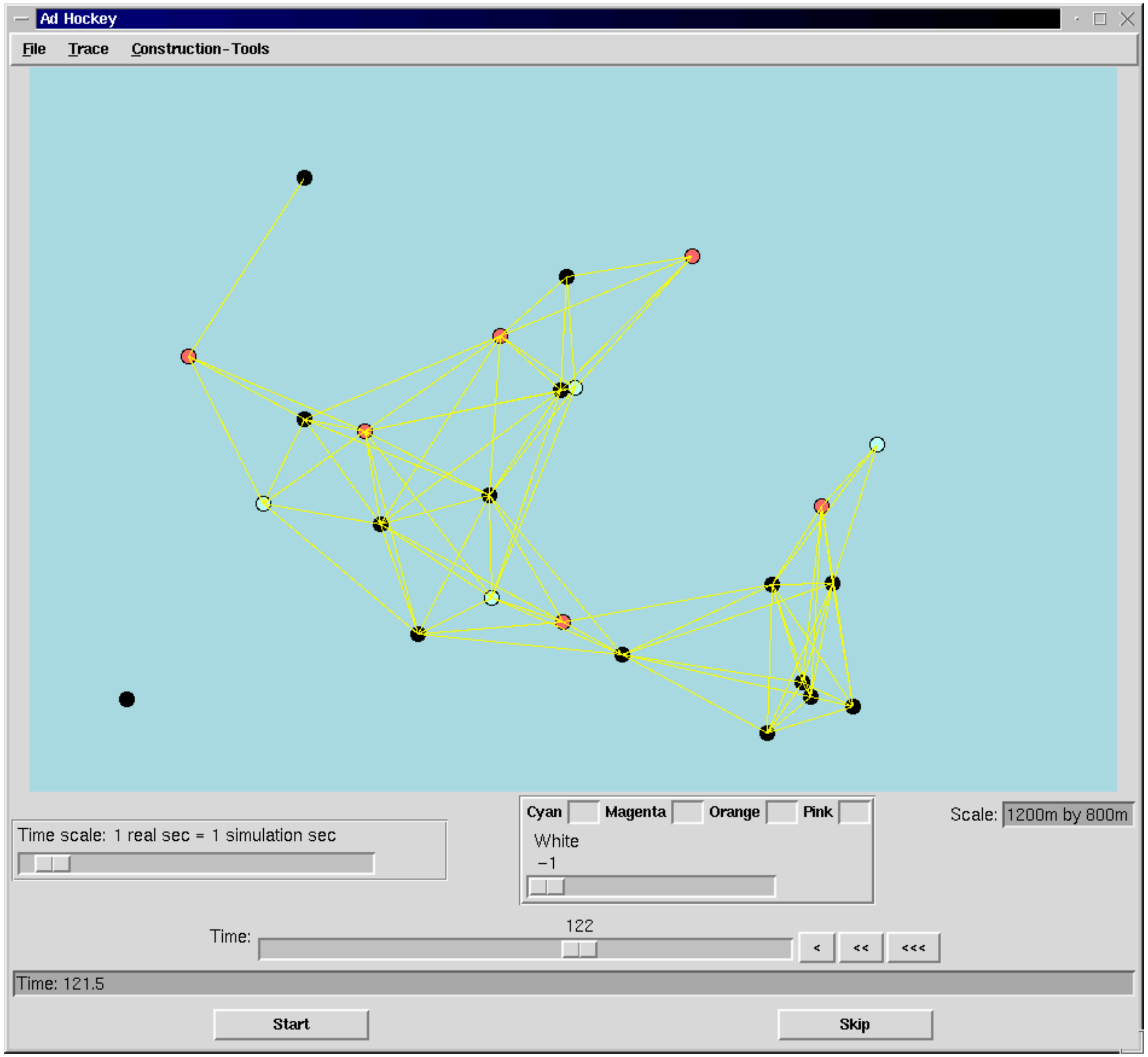
- Ported from different Mobile-IP implementations (Sun, CMU, . . .)
- Partly incomplete, buggy, ad-hoc solutions
  - Mobile IP requires ad-hoc routing protocol
  - *ns* doesn't handle overlapping service areas correctly
- Different trace file format
- Missing features
  - No route optimization
  - No support of IPv6
  - No / insufficient visualization

Wireless Node



Ad-Hockey (CMU)

- Visualization of ad-hoc networks
- No support for Mobile-IP / wired networks (*y*et)
- Special trace-file format
- Tool makes many assumptions / simplifications


Ad Hockey
File
Trace
Construction-Tools
Time scale: 1 real sec = 1 simulation sec
Cyan
Magenta
Orange
Pink
White
-1
Scale:
1200m by 800m
Time:
122
<
<<
<<<
Time: 121.5
Start
Skip

Further Features

- Abstraction (speed, reduction of resource requirements)
- Multicast (DVMRP, PIM, . . .)
- RTP/RTCP, SRM
- QoS (IntServ, DiffServ)
- Automated scenario generation (topologies, traffic patterns,. . .), scenario library, conversion of Internet topology maps
- Automated test suites (verification of correct behavior)
- Network emulation: interaction of network simulator and operational network node

ns Resources

- Website: `http://mash.CS.Berkeley.EDU/ns/`
- Code available for nearly all platforms
  (current version ns-2.1b6, released 01/18/2000)
- *ns* documentation, tutorials
- Overview of *ns* class structure (inheritance, . . .)
  `http://www-sop.inria.fr/rodeo/personnel/`
  `Antoine.Clerget/ns/ns/ns/ns-current/HIER.html`
- Mailing lists
  `ns-users@mash.CS.Berkeley.EDU`
  `ns-announce@mash.CS.Berkeley.EDU`
- Research using *ns*
  `http://www-mash.cs.berkeley.edu/ns/ns-research.html`