Project 3: ns Simulation of Application-Level Routing

- Improve the application performance by user-level / application-level routing.

- Understand the use of network measurement and user decisions in network route selection.
Scenario

UDP Background Traffic at 6.1 Mbps

User sends
UDP Traffic at 6 Mbps

UDP Background Traffic at 7.3 Mbps
Project Description

5 tasks:

- **Task0**: Understand the C++ and OTcl Integration issues (0.5%)
- **Task1**: Writing a OTcl script (2%)
- **Task2**: Specification of measurement and selection algorithm (2%)
- **Task3**: Write the C++ modules (6.5%)
- **Task4**: Modify the OTcl script of task1 and test (4%)
- **Task5** (optional): Make algorithm adaptive to bandwidth and RTT (1%)
ns Setup

Follow the instructions:
http://www-networking.eecs.berkeley.edu/~altmann/lectures/ee122/copy-ns_v03.htm

The process:
- Download ns source code: ns.tar
- Install in ~/proj3 directory in your home directory
- Make changes to the Makefile (as described) or take Ye’s Makefile from the ee122 project Q&A web page
- Compile ns

Make sure that you are using your ns version from now on
Simulation System

tcl script (input)

```tcl
# Create a simulator object
set ns [new Simulator]

# Define different colors for data flows
$ns color 1 Blue
$ns color 2 Red

# Open the nam trace files
set nf [open project_b_task_1I.nam w]
$ns namtrace-all $nf

# Open the measurement output files
set f0 [open project_b_task_1I_out0.tr w]
set f1 [open project_b_task_1I_out1.tr w]
set f2 [open project_b_task_1I_out2.tr w]
```

NAM (output)

Gnuplot (output)
**Source Routing**

- #enables source routing
  - $n1 ns src_rting 1

- #set routes
  - set temp [$n1 set src_agent_]
  - $temp install_connection 2 1 5 1 3 5

- #connect transport agent with src routing agent
  - $udp0 target [$n1 set src_agent_]

- #start simulation
  - $ns at 6.0 "$udp1 set fid_ 2"
ns Architecture

- Control operations in Otcl
- Class hierarchy (extract):

```
Agent/TCP/Reno
```

![Class hierarchy diagram](Image)
**ns Simulator Basics**

- *ns* is written in C++ with an OTcl interpreter as front end

- *ns* supports a class hierarchy in C++ (called **compiled hierarchy**) and a similar class hierarchy within OTcl (called **interpreted hierarchy**).

- There is a **one-to-one correspondence** between classes in both hierarchies. But, there are some hierarchies in C++ and OTcl that are not mirrored.

- When a user **instantiates a class in a OTcl script**, an object of the **corresponding compiled class** is also instantiated through methods in the class **TclObject**.
Build New Agents / Applications: OTcl

- `set udp1 [new Agent/UDP/UDPar]`
  - define new subclass of the `TclClass`
- `$ns attach-agent $n5 $udp1`
- `set ar1 [new Application/AppRouter]`
  - define new subclass of the `TclClass`
- `$ar1 set packetSize_ 500`
  - Set default values by using `bind()` method
- `$ar1 attach-agent $udp1`
  - define the sequence of actions that implement the command "attach-agent" in `AppRouter::command()` method
Build New Agents / Applications: C++

- Implementation of the Application/AppRouter class
  - Name example: *ar-app.cc* and *ar-app.h*

- Implementation of the Agent/UdpAgent/UdpARAgent class
  - Name example: *udp-ar.cc* and *udp-ar.h*

- Put them in *ns-2.1b9a/mycode*

- Based on *mm-app.cc, mm-app.h, udp-mm.cc* and *udp-mm.h*
  - [http://nile.wpi.edu/NS/](http://nile.wpi.edu/NS/)
  - see "Add New Application and Agent"
ns Files Modification

- Depending on the implementation, some existing files have to be modified

- Make the new protocol (of AppRouter) and its name known
  - common/packet.h
  - tcl/lib/ns-packet.tcl

- Set the default value of variables
  - tcl/lib/ns-default.tcl

- If necessary, add method definitions:
  - apps/app.h
  - common/agent.h
Literature and Help for \textit{ns}

- **NS by Example**
  - http://nile.wpi.edu/NS/
  - "\textit{Add New Application and Agent}"

- **NS Manual**
  - http://www.isi.edu/nsnam/ns/
  - Chapter 3 of the Manual