

NS-2 Tutorial

Qing (Kenny) Shao (SFU/CNL)

Polly Huang (AT&T Labs Research)

Communication Networks Laboratory

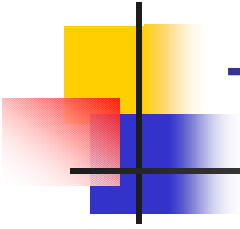
<http://www.ensc.sfu.ca/research/cnl>

School of Engineering Science

Simon Fraser University

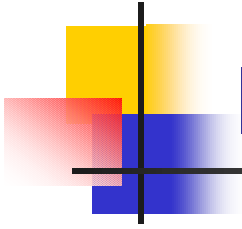


**communication
networks
laboratory**



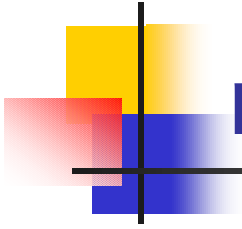
Tutorial Schedule

- Basic introduction (Leo)
- Ns fundamentals (Leo)
- ns-2 internal
- Extending ns-2 Simulator



Introduction (Review)

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

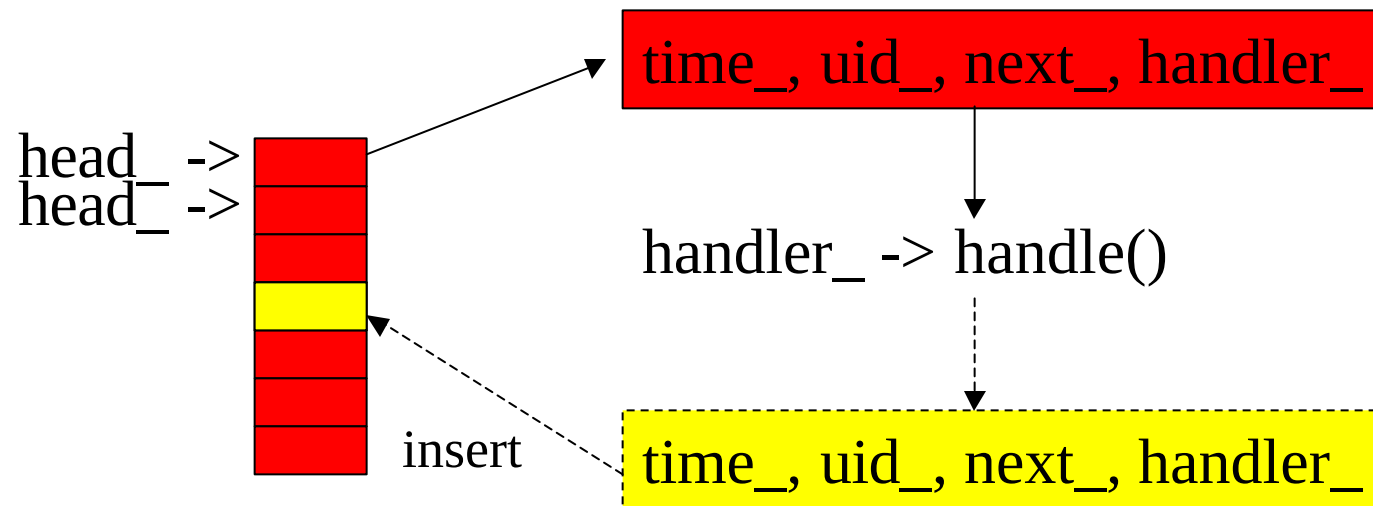


ns-internals

- Discrete Event Scheduler
- Network Topology
- Routing
- Transport
- Application
- Packet Flow
- Packet Format

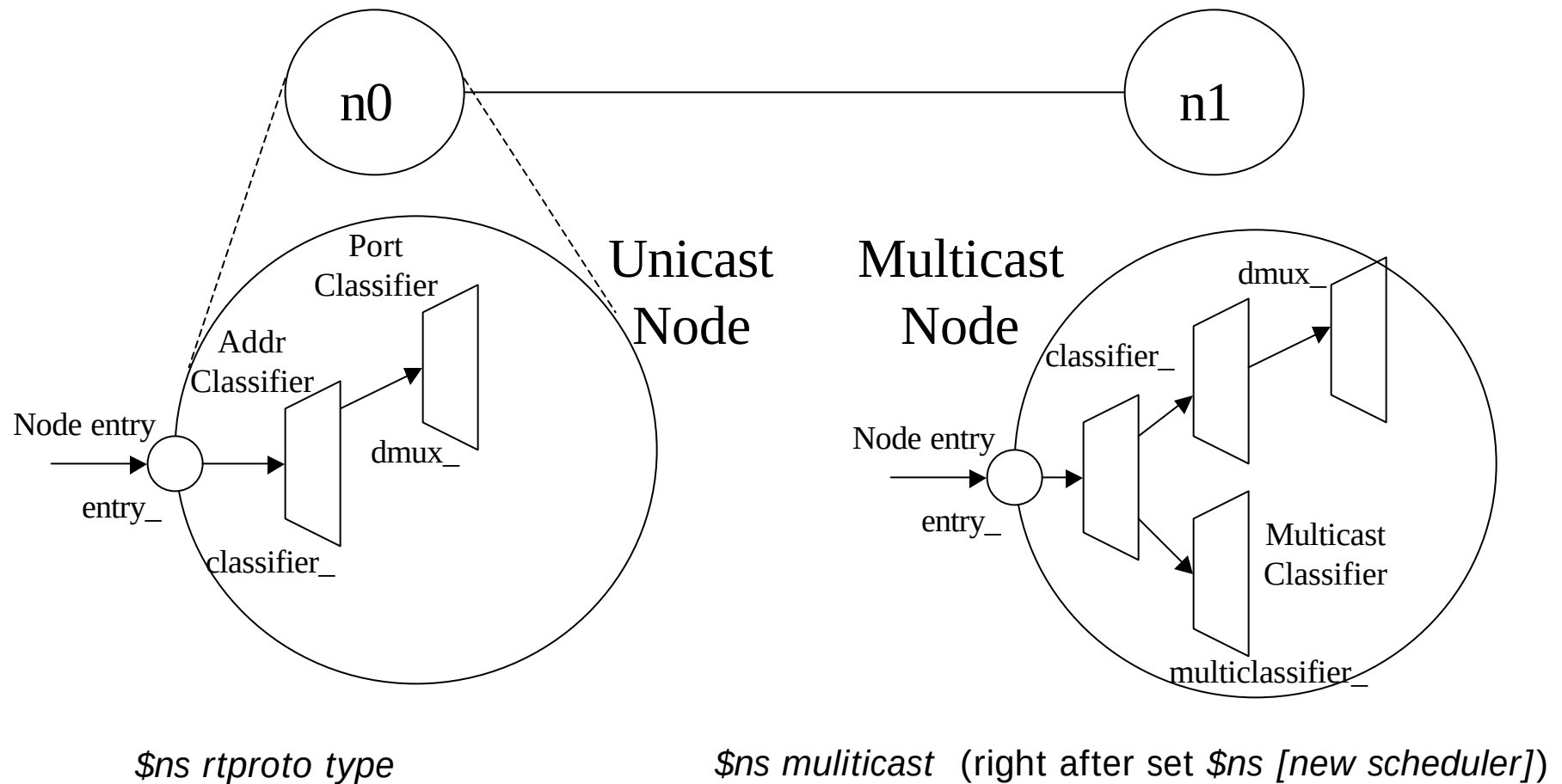


Discrete Event Scheduler



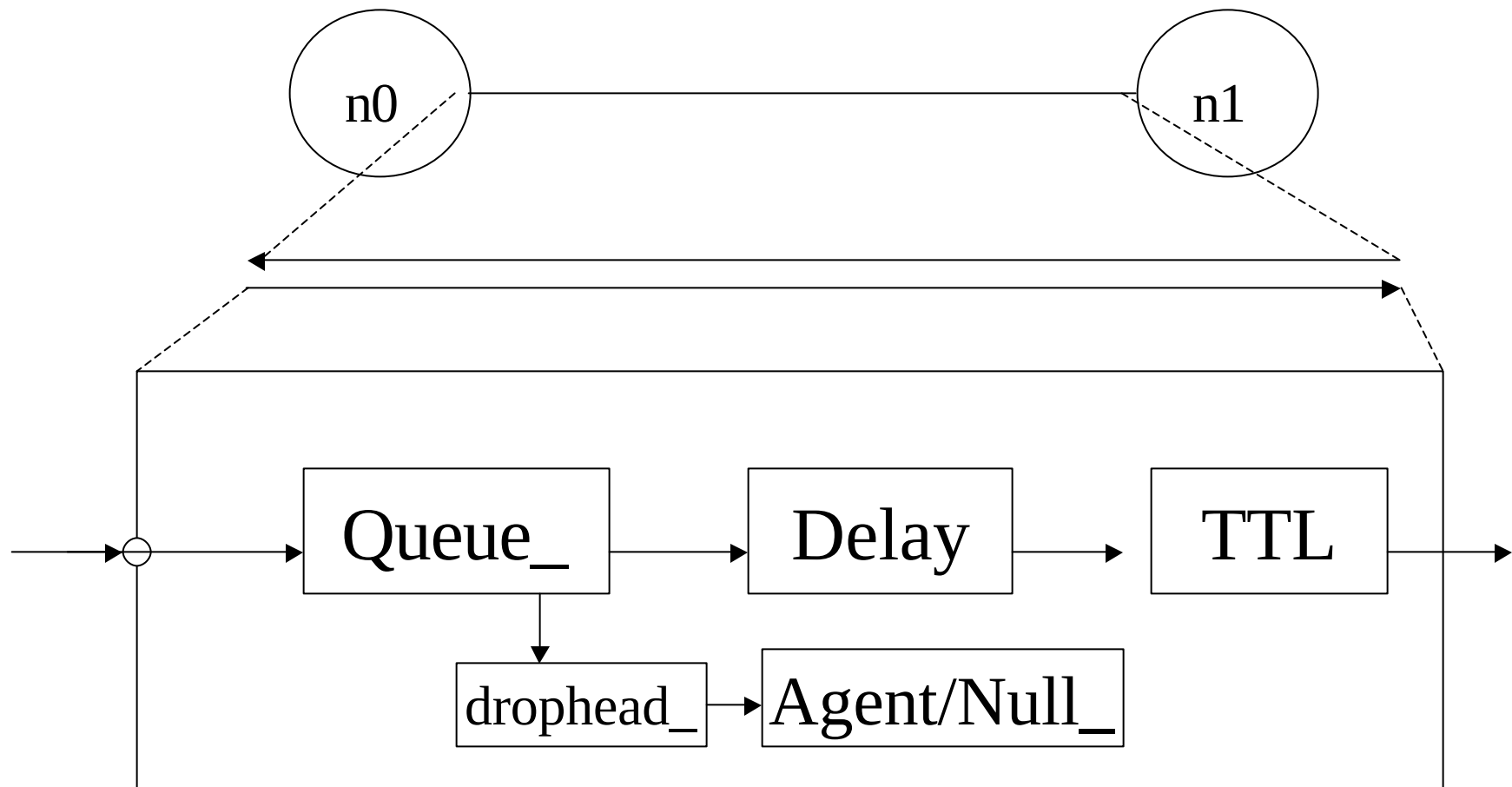


Network Topology - Node





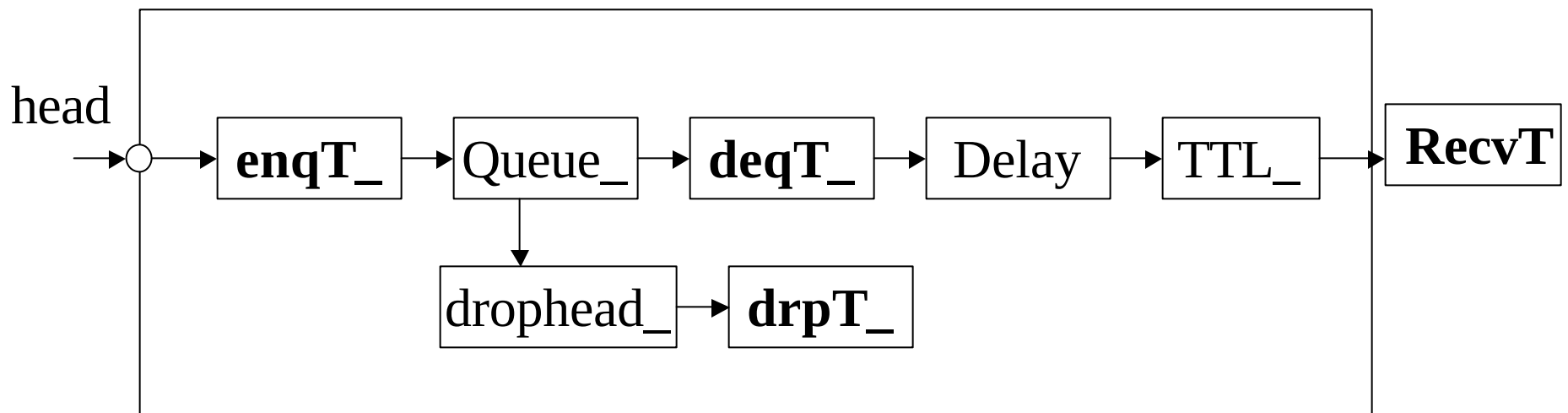
Network Topology - Link





Network Topology - Tracing

\$ns trace-all filename or \$ns namtrace-all filename

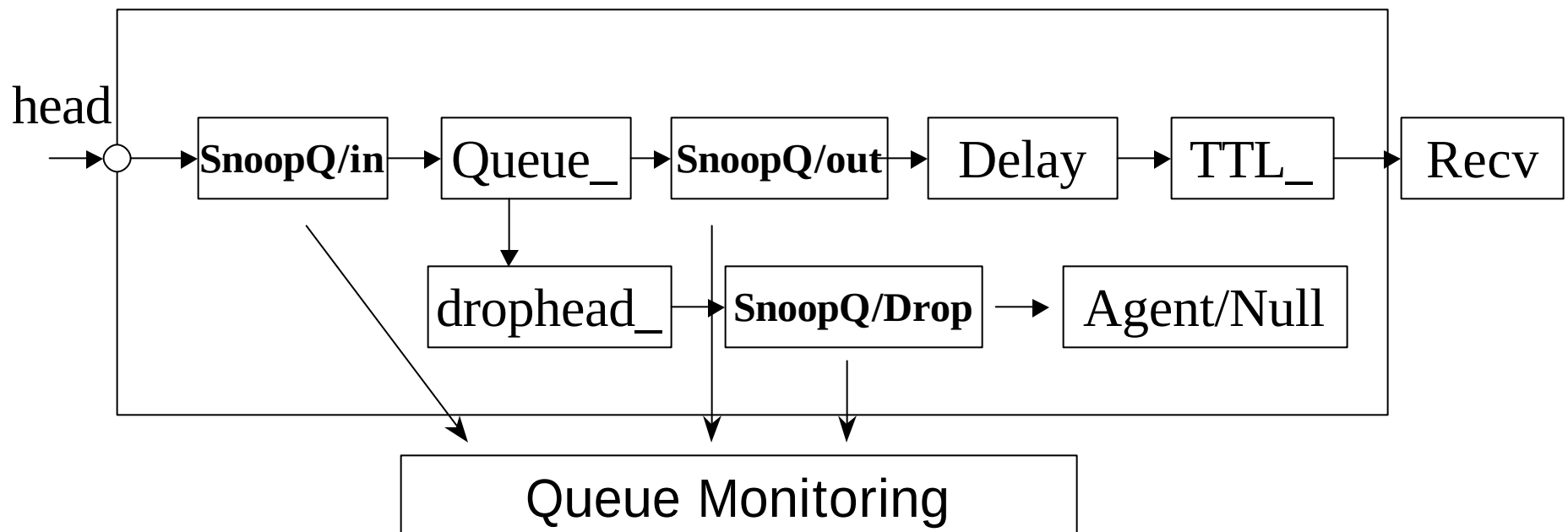


Inserting trace object

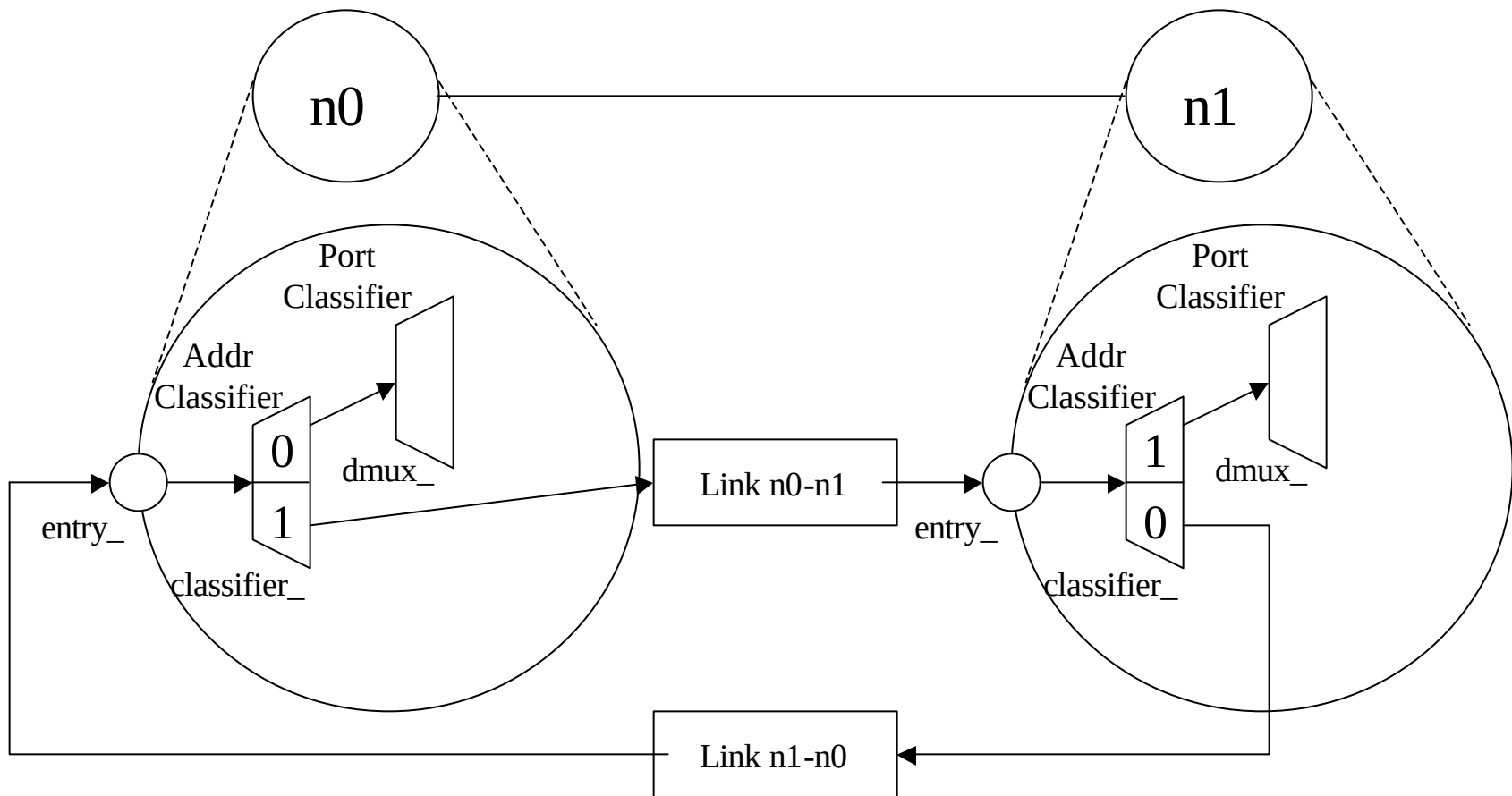


Network Topology - Tracing

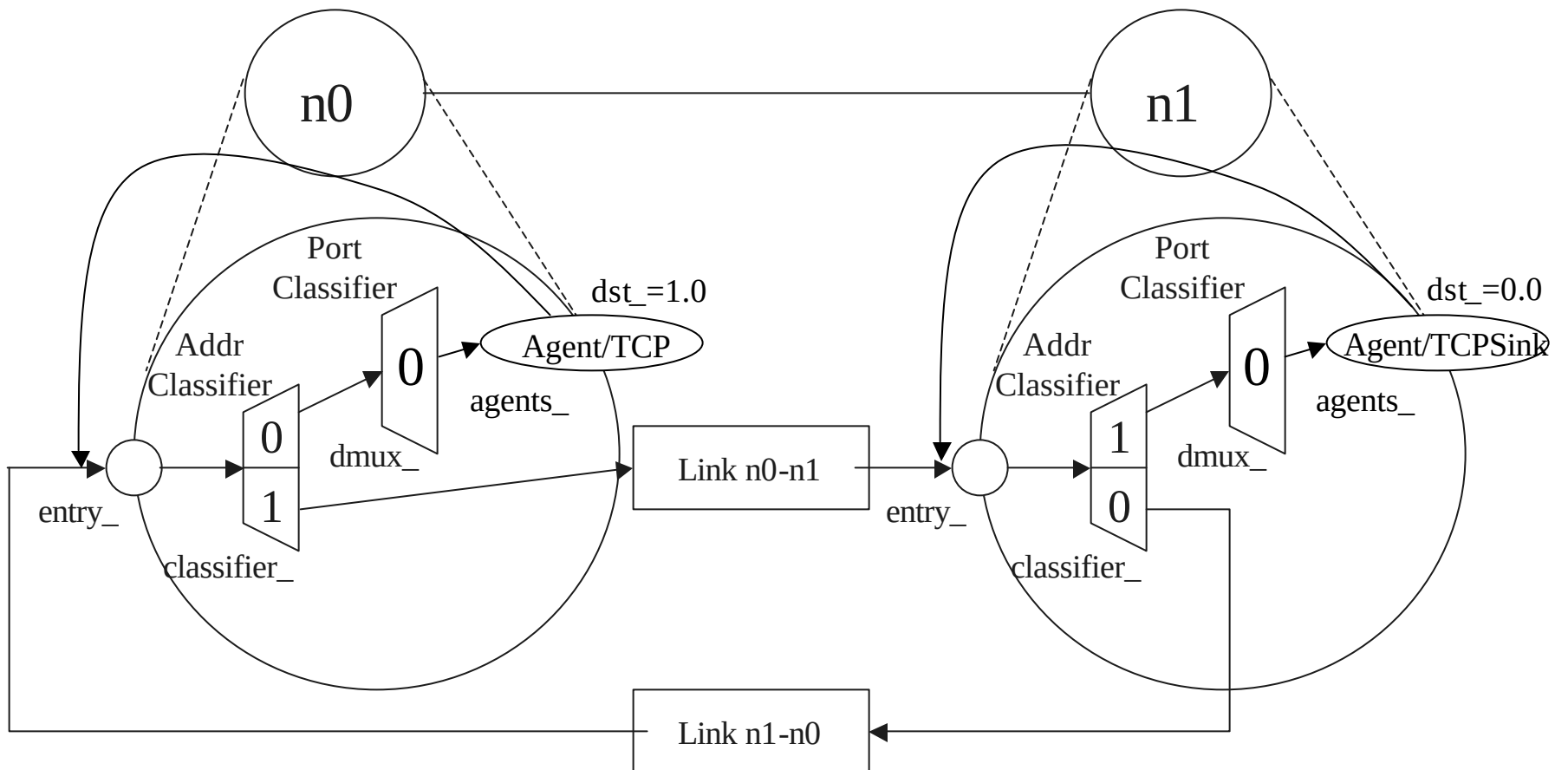
```
$ns monitor-queue node1 node2  
$ns at 0.0 qmon trace $filename
```



Routing

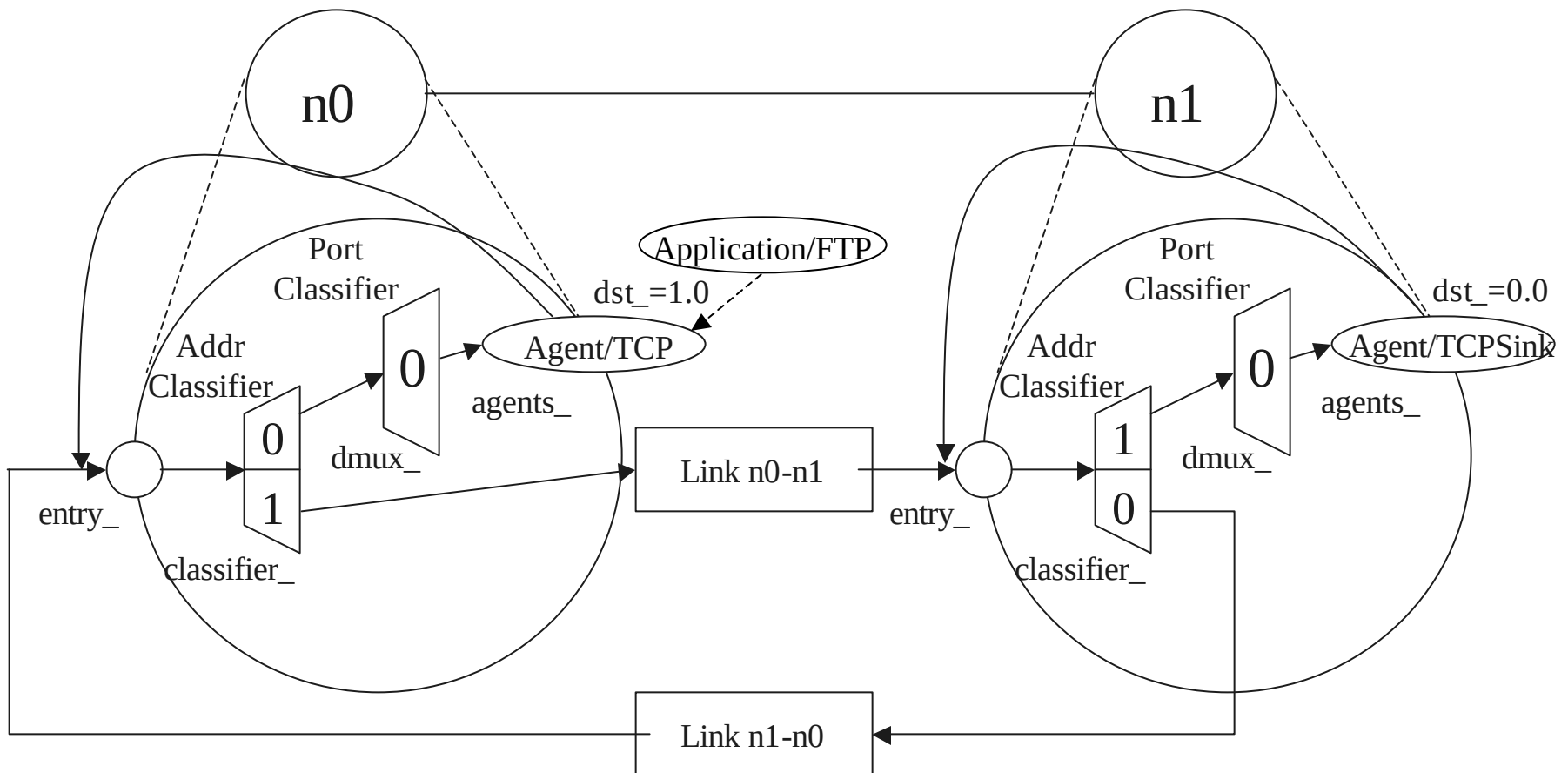


Transport



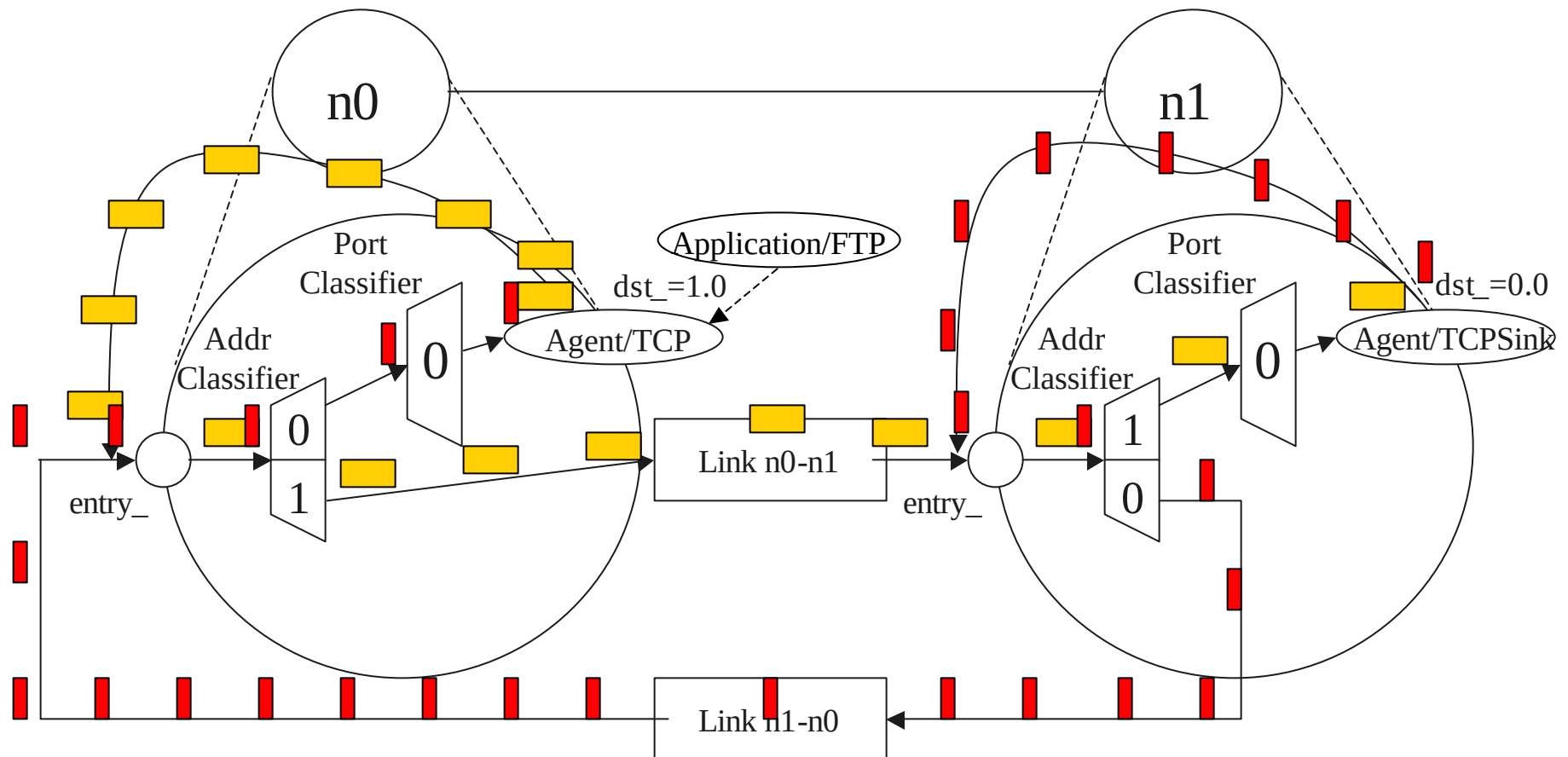


Application



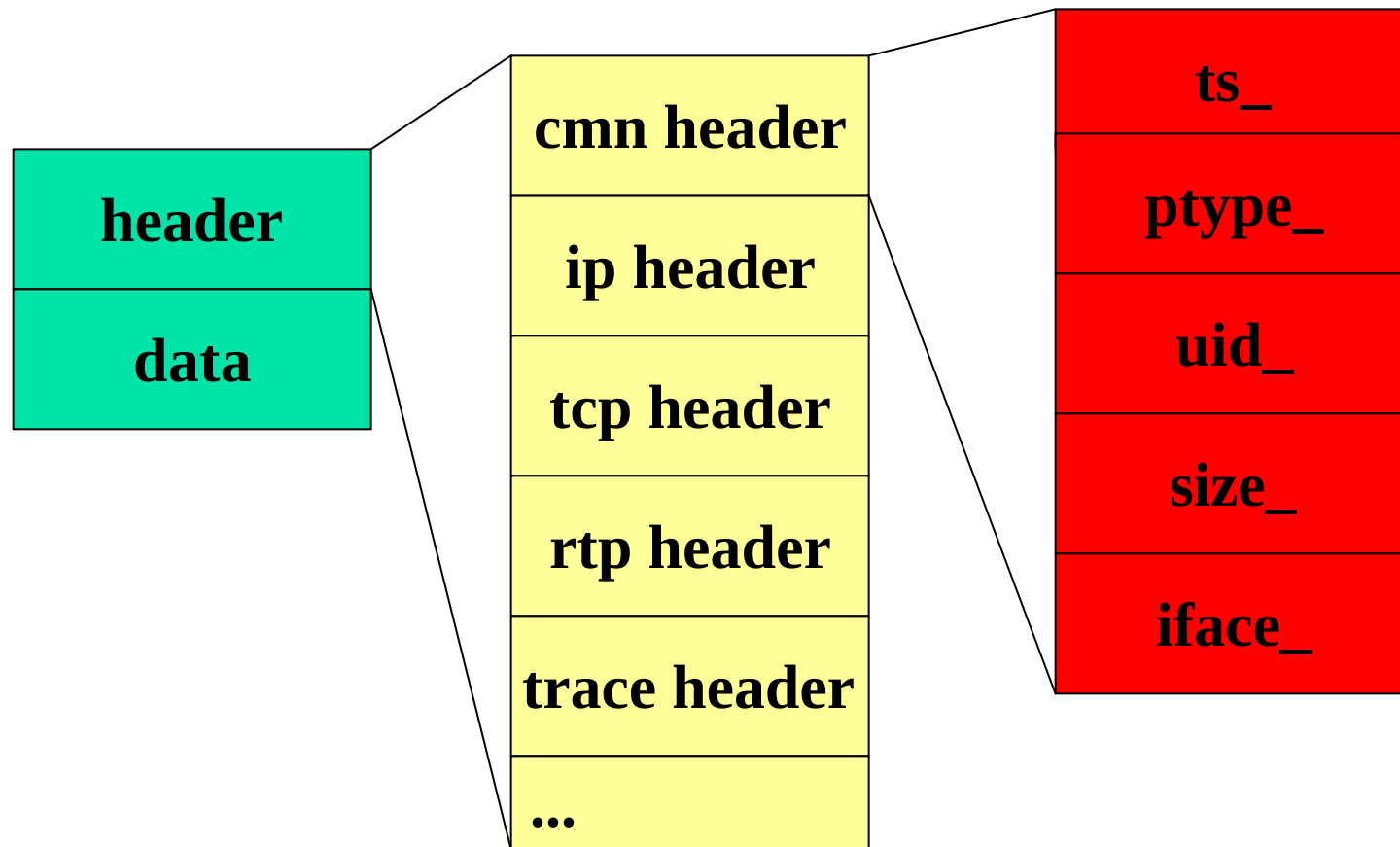


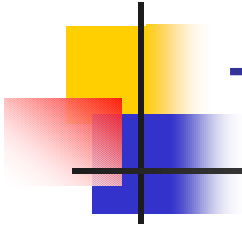
Packet Flow





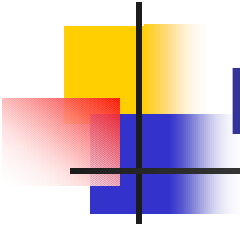
Packet Format





Tutorial Schedule

- Basic introduction (Leo)
- Ns fundamentals (Leo)
- ns-2 internal
- Extending ns-2 Simulator

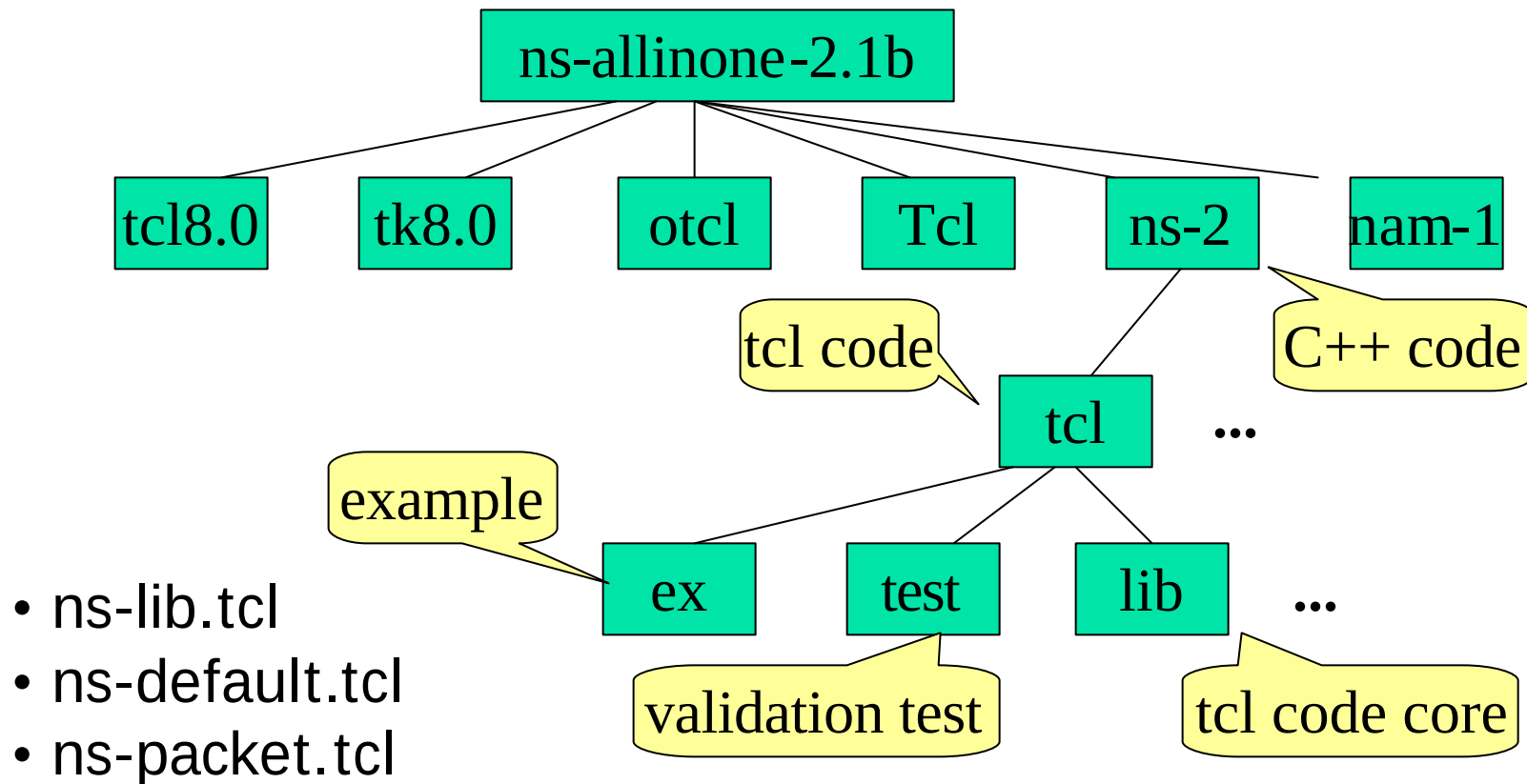
A decorative graphic consisting of overlapping yellow, red, and blue squares with a black crosshair.

Extending ns-2 Simulator

- **Making changes**
- Creating new components



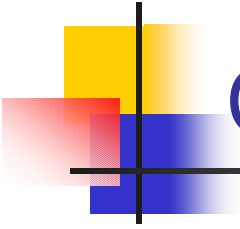
ns-2 Directory Structure





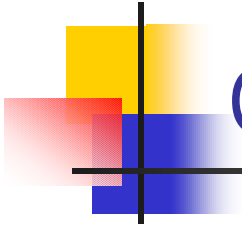
Making Changes in C++/Otcl Space

- Existing code
 - recompile
- Addition
 - change Makefile (NS_TCL_LIB), tcl/ns-lib.tcl (source) and recompile



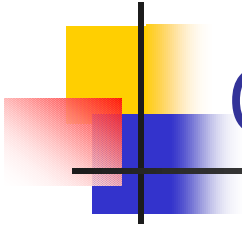
Outline

- Making changes
- **Creating new components**



Creating New Components

- Guidelines
- Inheritance Hierarchy
- C++ and otcl Interface
- Debugging

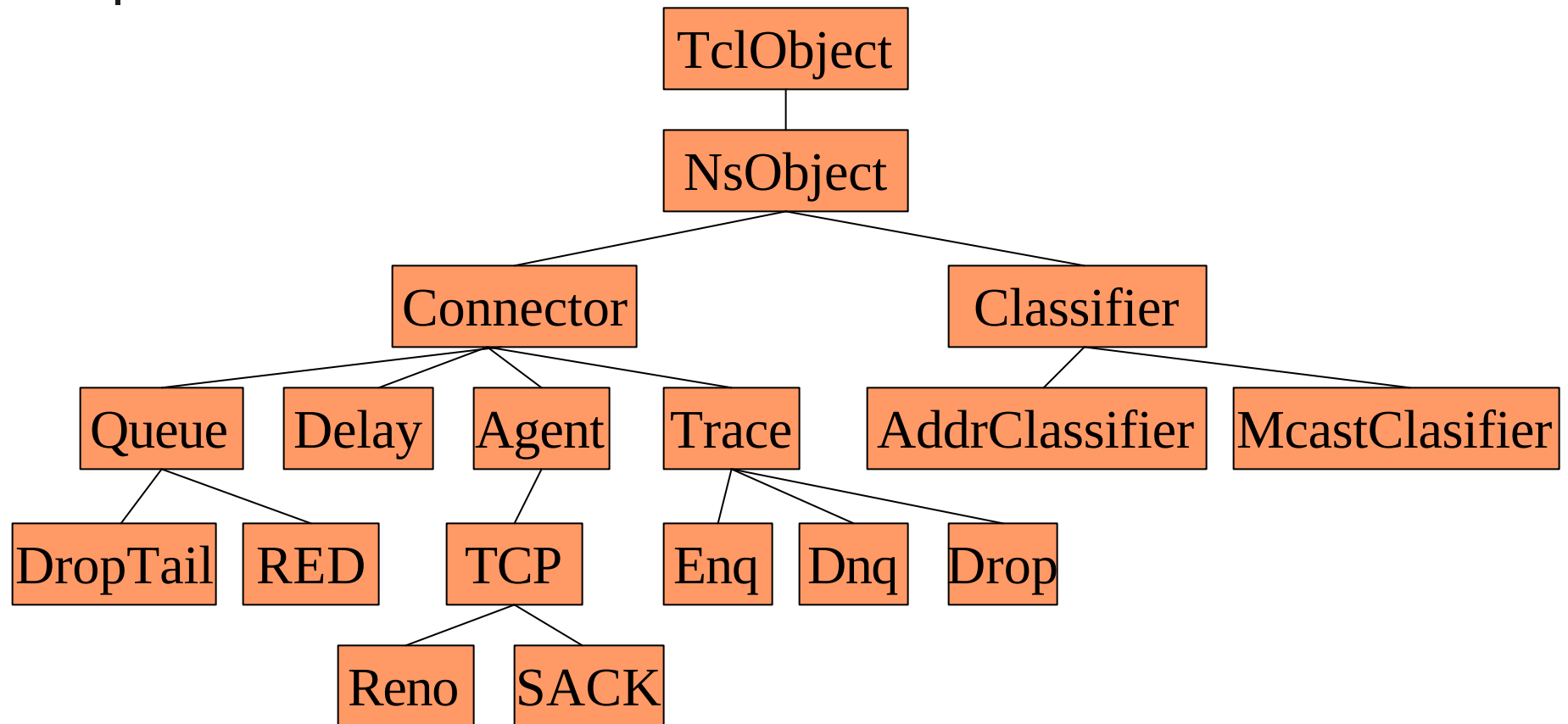


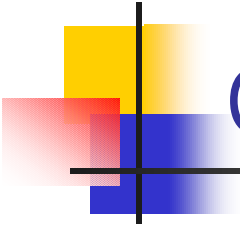
Guidelines

- Decide its inheritance structure
- Create the class and fill in the virtual functions
- Define otcl linkage functions
- Write the necessary otcl code to access your agent



Class Hierarchy (Partial)



A decorative graphic consisting of overlapping yellow, red, and blue squares with a black crosshair.

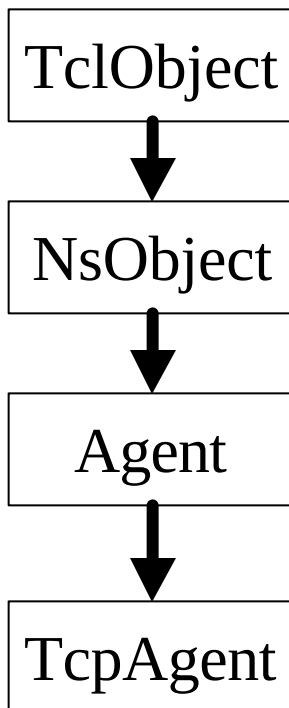
C++ and otcl Linkage

- TclClass
- TclObject: bind() method
- TclObject: command() method



TclClass

C++



```
Static class MyAgentClass : public TclClass {  
public:  
    MyagentClass() :  
        TclClass("Agent/MyagentOtcl") {}  
        TclObject* create(int, const char*const*) {  
            return (new MyAgent());  
        }  
} class_my_agent;  
  
$ns new Agent/myAgentOtcl
```



TclObject: bind()

- C++

```
MyAgent::MyAgent() {  
    bind("my_var1_otcl", $my_var1);  
    bind("my_var2_otcl", $my_var2);  
    ...  
}
```

- otcl

```
$myagent set my_var1_otcl 2  
$myagent set my_var2_otcl 3.14
```



TclObject: command()

- C++

```
Int MyAgent::command (int argc, const char*const* argv){  
    if (argc == 2) {  
        if (strcmp(argv[1], "call_my_priv_func") == 0) {  
            MyPrivFunc();  
            ...  
            return(TCL_OK);  
        }  
    }  
    return (Agent::command(argc, argv);  
}
```

- otcl

```
set myagent [new Agent/MyagentOtcl]  
$myagent call_my_priv_func
```



Debugging

- `printf()` and `puts ""`
- `gdb`
- tcl debugger
 - <http://expect.nist.gov/tcl-debug/>
 - place debug 1 at the appropriate location
 - trap to debugger from the script
 - single stepping through lines of codes
 - examine data and code using Tcl-ish commands