Booklet of Network Modeling and Simulation

<u>CONTENT</u>

Course code CS432
Credits 3+0

Instructor

Dr. Ali Hammad Akbar

Lecturing style

Video lectures of short

duration (5-7 minutes)

Evaluation

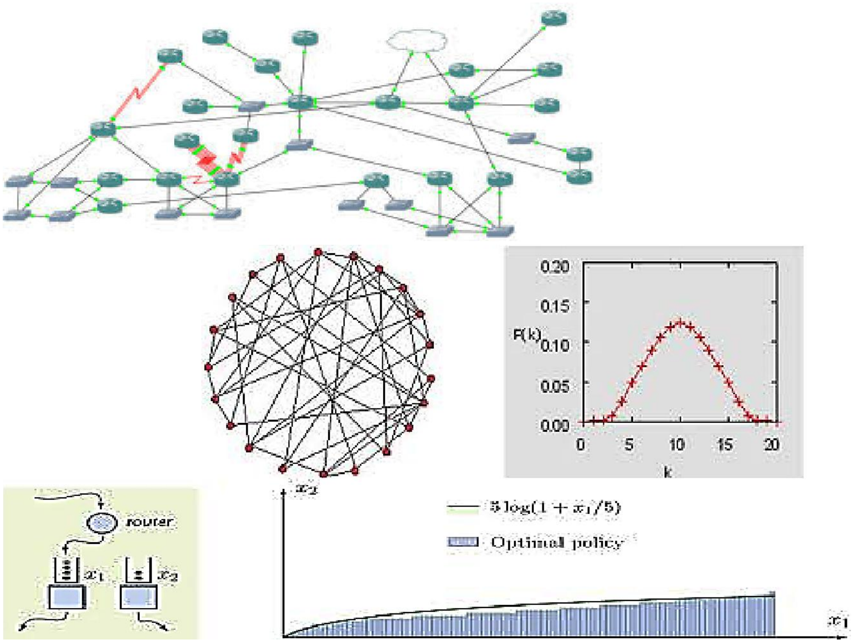
Quizzes

Assignment

Simulation modules

Mid-term and final

The complex world of networks
Networks are many and diverse. Can we simplify it? Or at least the discussion of it?



Need for NeMS
Let us explore the need and motivation to perform Network Modeling and Simulation (NeMS) by looking at the technology landscape. The landscape consists of the people, technology and their relationships.

Technology Landscape
1. Communications systems: Evolving rapidly
2. User demands: High performance networks
3. Service providers: Rapidly expanding their network infrastructure

Network researchers face the protocol war by developing new communications techniques, architectures and capabilities. Equipment vendors are releasing new devices with increasing capability and complexity. Technology developers and OEMs are developing NG equipment. Network designers and developers are working on how to satisfy the QoS demands of users amidst emerging technologies and techniques viz a viz legacy counterparts? Network Engineer in operations is thinking about what is the right approach to solving problems? Do I buy latest device from company X that claims to solve all my problems? Do I replace underlying technology of my system with the latest generation? Next Generation Network Architect wonders how do I know how this new approach will interact with already existing protocols? How do I build confidence in the utility of this approach without producing and deploying the technology? Is there one solution? Actually not! There are various ways to answer and satisfy the goal seekers. These include

- Prototyping & empirical testing
- Trial field deployment
- Modeling and Simulation (M & S)
- Analysis

The order represents decreasing costs but increasing abstraction. It is upto the network engineer to trade them off.

What is NeMS?
Network Modeling and Simulation is often considered a single term. In reality, it is not! Simulation is the imitation of behaviour of real-world system or Computational re-enactment according to rules described in model. Whereas modeling is a step that precedes simulations. Together they form an iterative process approximating the real world systems. Model is the logical representation of a complex entity, system, phenomena or a process. In communications, network model could be analytical representation, mathematical form as a state Machine or closed or approximate form. Computer simulation is the execution of computer software that reproduces behavior with a certain degree of accuracy to provide visual insight. It is basically a template on which a computer program runs. It has

- Inputs
- Outputs
- Behaviour

Formally, simulations are pieces of computer software that implement algorithms, take inputs and give outputs.

The model definition could be

- Descriptive
- Analytical
- Mathematical
- Algorithmic

The computer models can be described into various types

- Stochastic vs Deterministic
- Continuous vs discrete
- Steady state vs Dynamic
- Local or Distributed
- Linear or nonlinear
  Open or closed

These models must be applied according to the perspective. It is important to Model only what you understand. Likewise, understanding your model is equally necessary. Model what you need & no more so that it is neither underdefined nor overdefined.

Simulation Building Process
Consider a one hop communication scenario between two wireless notebooks connected through a WiFi AP. The simulation entities would include wireless computers and their packets (multiple instances), WiFi AP (single instance), and a traffic generator (single instance) that creates wireless computers and their packets. The states of the system would include WiFi AP (idle or busy). Each computer generates a number of packets with each packet successful/failed. The events would include wireless computer creation, packet generation, wireless AP activity. Queues would be needed to schedule the events. These would contain frames waiting in output queue of wireless computer and frames (packets) at WiFi AP input queue. To make the simulation exciting, dynamic and insightful, randomization has to be performed. These would include random realizations of packet lengths, no. of frames per wireless computer, no. of wireless computers, BER (and PER) and packet drop ratio in WiFi AP input queue. It is further needed to distribute various entities and events. The distributions could include Uniform and Gaussian etc for packet lengths and no. of frames per wireless computer.

How Does The Simulation Run?
Following steps are executed during the running of the simulation.

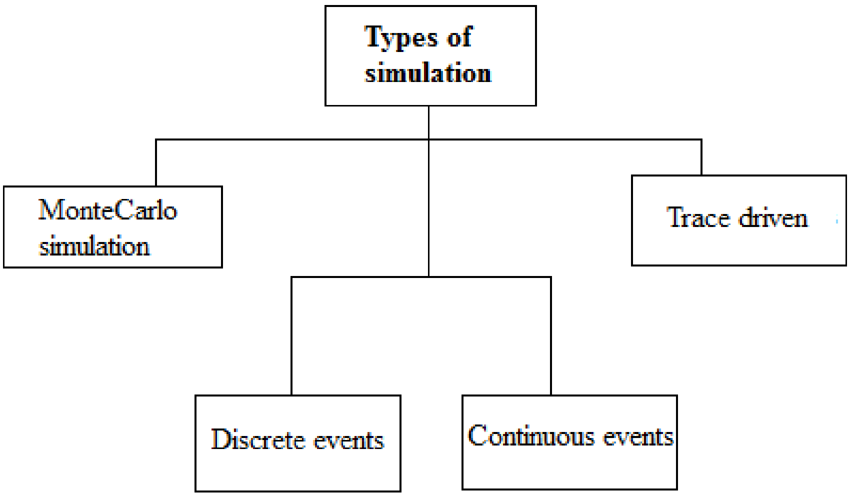
- Inputs created/initialized
- Events of transmission, reception and noise occur
- Randomness causes queues to behave and err
- Packet successes/failures
- Simulation logs are compiled and presented as the output in desirable formats

Components of a simulator

- A self-contained program
- Event queue
- Simulation clock
- State variables
- Event routines
- Input routine
- Report generation routine
- Initialization routine
- Main program

Types of simulations

- Monte Carlo simulation
- Trace driven
- Discrete events
- Continuous events



When to simulate

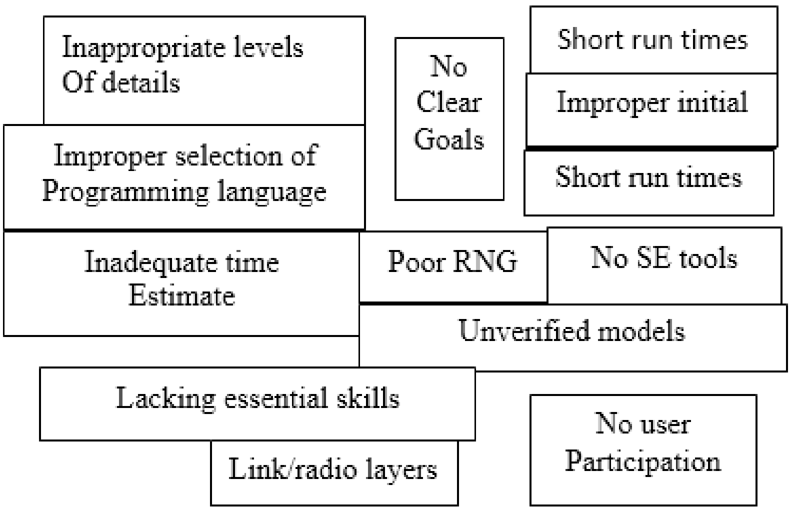
- Analytical model not feasible (complex)
- Analytical model not possible (too simple)
- Simulate to verify analysis
- Otherwise simulations are unnecessary

When not to simulate

- Analytical model gives good enough representation
- Simulation takes months
- Simulation is expensive
- Simulation is non-scalable

General mistakes

- Inappropriate levels of details
- Improper selection of programming language
- Unverified models
- Improper initial conditions
- Short run times
- Poor random number generators
- Inadequate time estimate
- No achievable goals
- Incomplete mix of essential skills
- Inadequate level of user participation
- Inability to manage simulation project



Inappropriate levels of details

- Include what is relevant
- Too fine simulations computationally heavy
  - Many interdependent parameters
  - Difficult to assess their interplay
- Tip: Necessity & sufficiency

Improper programming language

- Scope & type of simulation determine best choice
- Object oriented vs. procedural
  - Types/diversity of simulation parameters
- Interpreted vs. compiled
  - Machine dependence
  - Speed
  -

Unverified models
- Programming is non trivial
- Semantic mistakes
- make simulations get
- Wrong results
- Misleading results
- Modular verification a must

Improper initial conditions
- Initial condition not steady state
- Often a late realization
- Surprisingly wrong results
- May never converge

Short run times
- Strong dependence on Initial conditions
- Don't achieve true
- steady state

Poor random number generators
- Lacking pseudo-random sequence leads to predictability
- Wrong choice of seed value could cause inadvertent correlation between processes
  - Use celebrated RNGs

Inadequate time estimate
- Models overstate the simulations
  - Implementations get delayed
- Software development life cycle must assess model complexity

No achievable goals
- Goals not defined
  - Tangible output analysis
  - Logs and trace files
- Goals are unreal
  - Affects simulation complexity and implementation

Incomplete mix of essential skills
- Domain knowledge
- Statistics
- Programming
- Project management
- Past experience

Inadequate level of user participation

- From modeling to implementation
- UI design
- Output analysis

Inability to manage simulation project

- Simulations are not monolithic
- Need software engineering tools
  - Multivariate design
  - Code management
  - Track changes

Simulation inaccuracies

- Over reliance on link budget methods for abstraction
- Overly simplistic modeling of radio layers

Over reliance on link budget methods for abstraction

- Link budget losses overly static
  - Fair enough for steady state analysis
  - Dynamic analysis not possible
- Results are misleading

Overly simplistic modeling of radio layers

- Lowest layer often ignored
  - No bit level BER & delay
- Often the Achilles heel
- Wrong results in highly dynamic use cases

Development of Systems Simulation

A “Still I am not dead yet!” scenario

$$h = \frac{1}{2}at^2 + vt + s,$$

Available

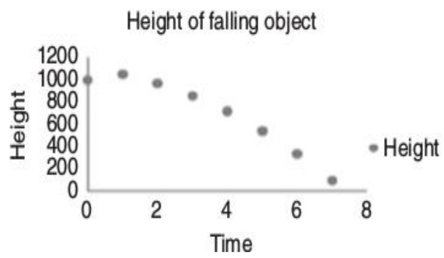
h = height (feet)
t = time in motion (seconds)
v = initial velocity (feet per second, + is up)
s = initial height (feet)
a = acceleration (feet per second per second)

Not available

Mass of object
Air resistance
Location of object

```
/* Height of an object moving under gravity. */
/* Initial height s and velocity v constants. */
main()
{
float h, v = 100.0, s = 1000.0;
int t;
```

t	v	h
0	100	1000
1	68	1052
2	36	972
3	4	860
4	−28	719
5	−60	540
6	−92	332
7	−124	92



Development Process

- Problem formulation
- Data collection & analysis
- Simulation development
- Model validation, verification, & calibration
- "What-if" analysis
- Sensitivity estimation

Problem formulation

- Identify controllable and uncontrollable inputs

Data collection & analysis

- What to collect
- How much to collect
- Cost and accuracy trade off

Simulation development

- Codify, codify and codify!

Model validation, verification, & calibration

- Validation
- Is it the right system?
- Emulates real phenomenon

Model validation, verification, & calibration

- Verification
- Are we building the system right?
- Implementation must correspond to the model

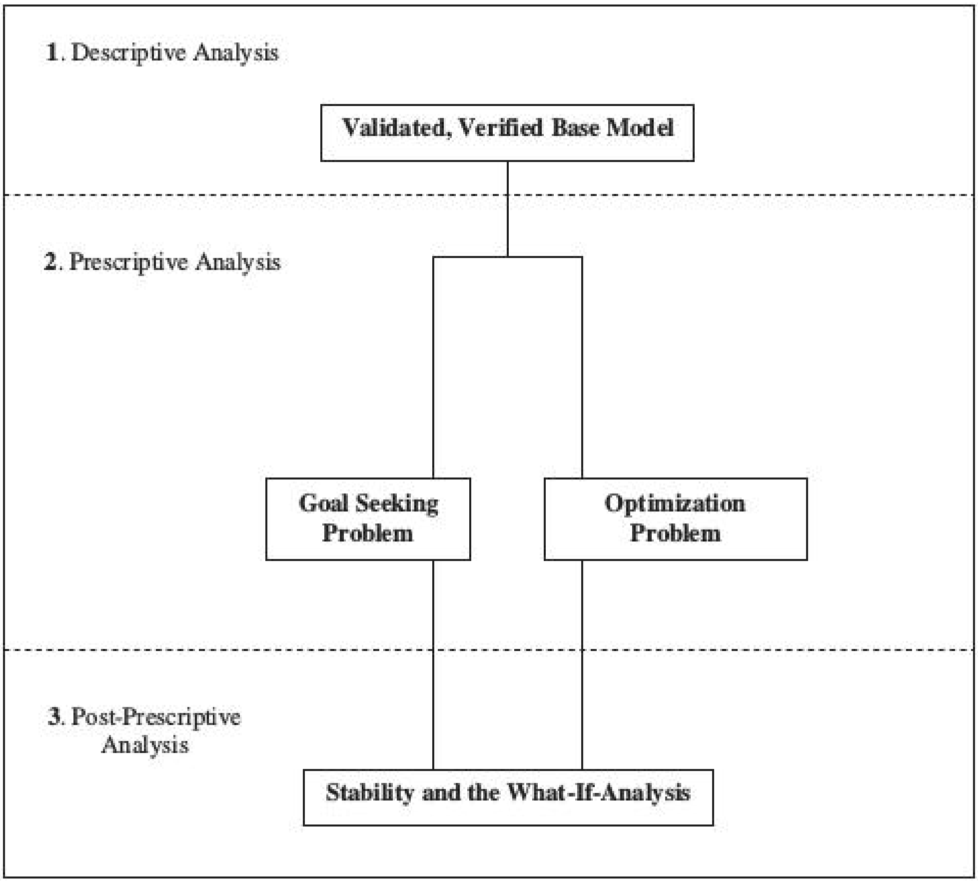
Model validation, verification, & calibration

- Calibration
- Parameter estimation
- Tweaking/tuning to ensure that simulated data follows real data

“What-if” analysis
- Performance measures with different inputs

Sensitivity analysis
- Relative importance of different parameters with respect to output
- Even with respect to each other

Life cycle of Simulation Development



Recommended Text and References

NeMS contents cover
- Well known mathematical models, equations and forms
- Widely used simulation tools and code reusability
- Their inter-relationship

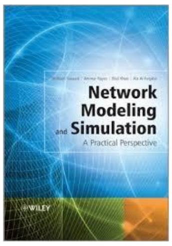
NeMS contents don't cover

- Mathematical derivations from scratch
- Programming dexterity

Uptill now

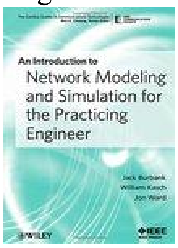
Basics of NeMS

- Mohsen Guizani et al, “Network Modeling and Simulation” John Wiley , 2010.



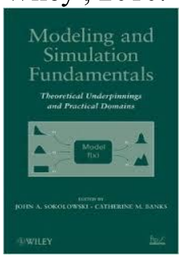
Basics of NeMS

- Jack Burbank et al, “An Introduction to Network Modeling & Simulation for the Practicing Engineer” John Wiley , 2011.

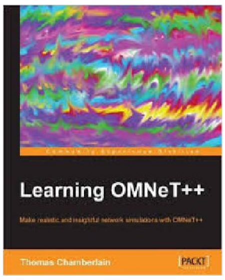


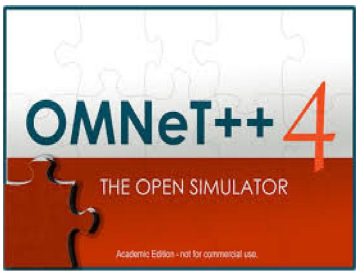
Basics of NeMS

- John A. Sokolowski & Catherine M. Banks, “Modeling and Simulation Fundamentals” John Wiley , 2010.



Next Roadmap





eclipse INET FRAMEWORK

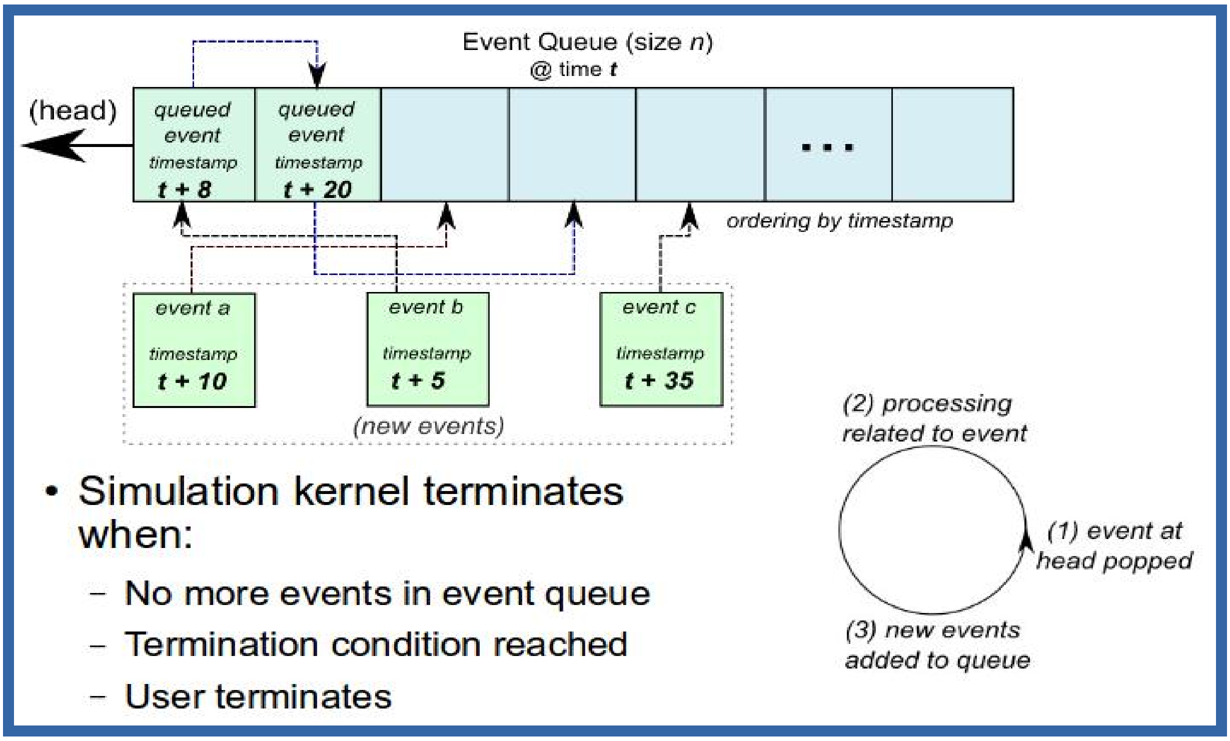
- TicToc tutorial

- OMNET++ Manual
- Website: https://omnetpp.org
- INET Framework for OMNeT++
- OMNET++ Wiki
- Mixim Sourceforge Page

Introduction to OMNET++
What is OMNET++

- Objective Modular Network Testbed in C++
  - Simulation kernel
  - Component-based simulation library
- A framework, not a simulator
- Designed to create & simulate any network

Simulation Kernel



Getting a free copy

- www.omnetpp.org
- Download the latest release (4.6 in our case)
  "Omnetpp-4.6-src-windows.zip"
- Complete folder
  - C++ compiler
  - CMD line build environment
- Download source code

Compile & Install

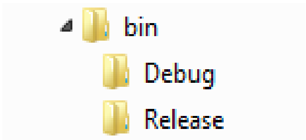
- Compiling and installing on Windows self-contained
- Enter OMNeT++ folder that you unzipped
- Run the file called Mingwenv.cmd

> Minimum GNU environment for Windows
> Compilers provide access to functionality
> Of Microsoft C runtime and some
> Language-specific runtimes

Compile & Install

- When terminal appears, enter the commands
  /.configure
  Make

> Build process produces
> Debug and release
> Binaries



> Debug is elaborate
> But slow

> Release is optimized
> & fast

Debug mode

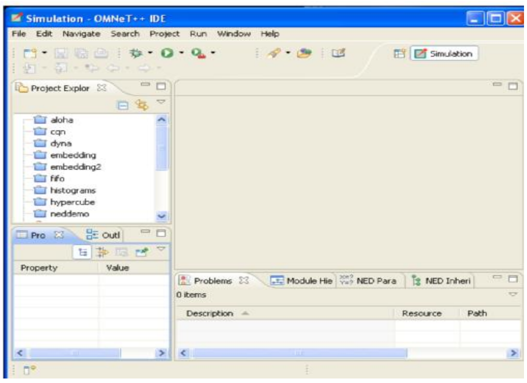
- Does not optimize the binary it produces
- Source code and generated instructions relationship is complex
- Allows accurate breakpoints setting
- Allows code step-through one line at a time
- Compiled with full symbolic debug information

Release mode

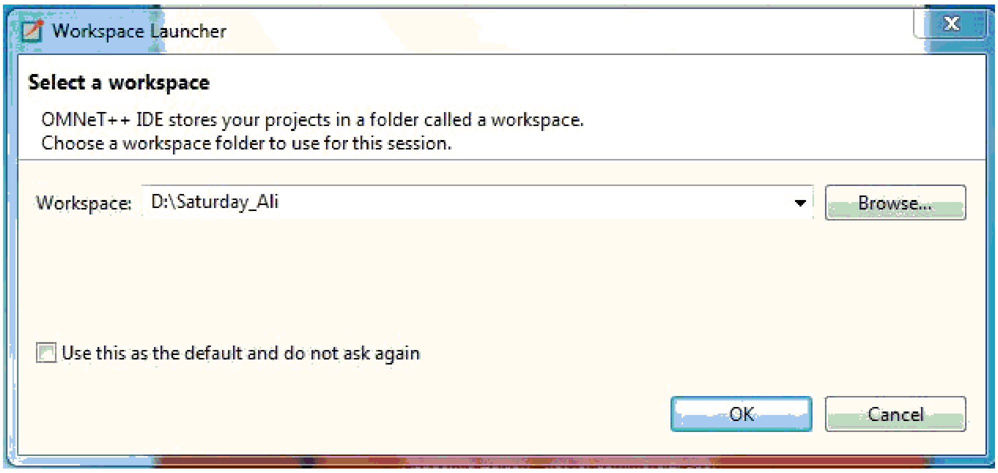
- Enables optimizations
- Generates instructions without any debug data
- Lots of code could be completely removed or rewritten
- Resulting executable may not match with written code

Running first time

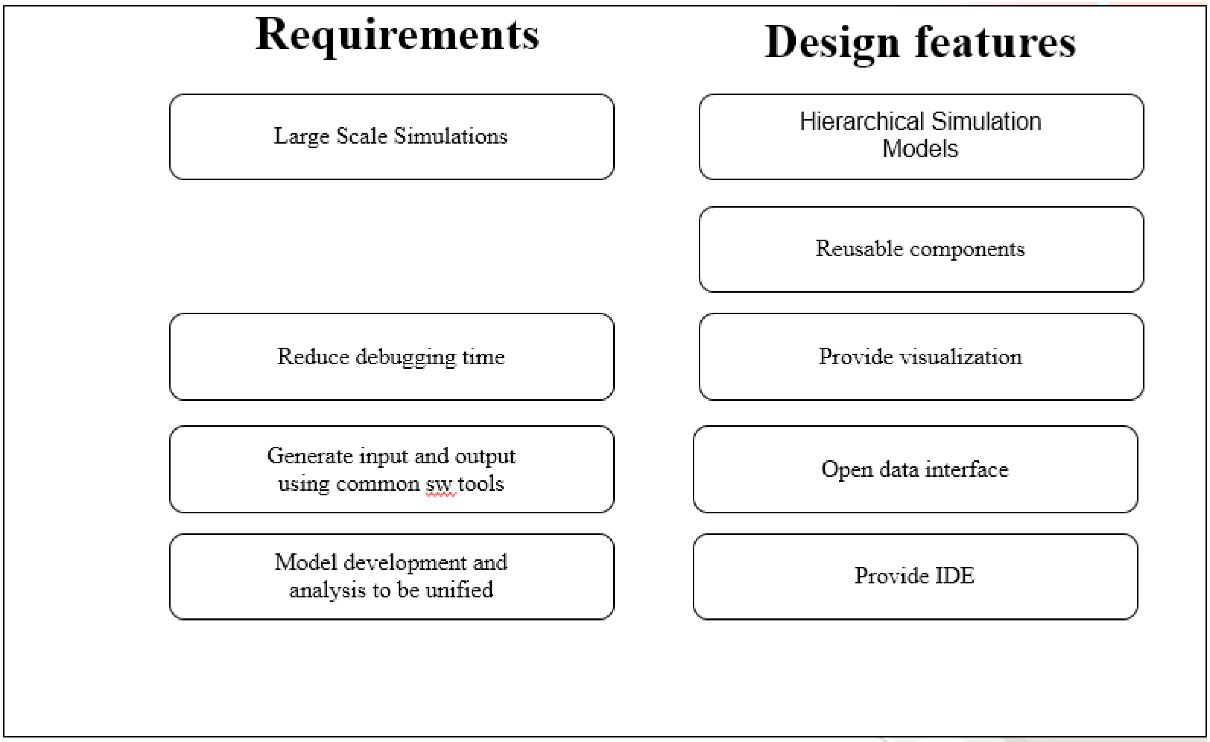
- OMNeT++ comes with an Eclipse-based Simulation IDE
- Type omnetpp



Select the default workspace

- A workspace is a logical collection of projects
- A workspace called p2p may contain only peer to peer applications
- 

Design of OMNET++

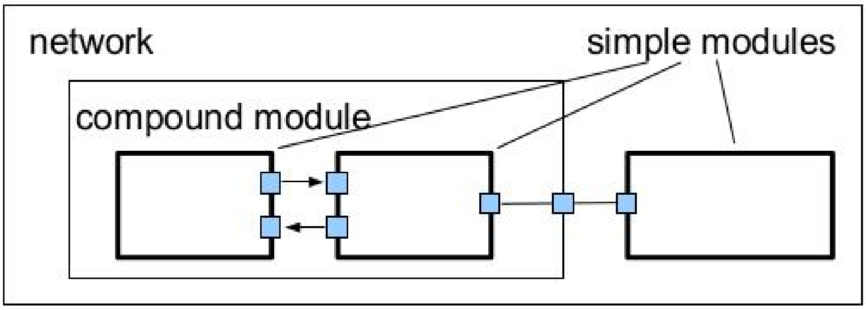


Model Structure

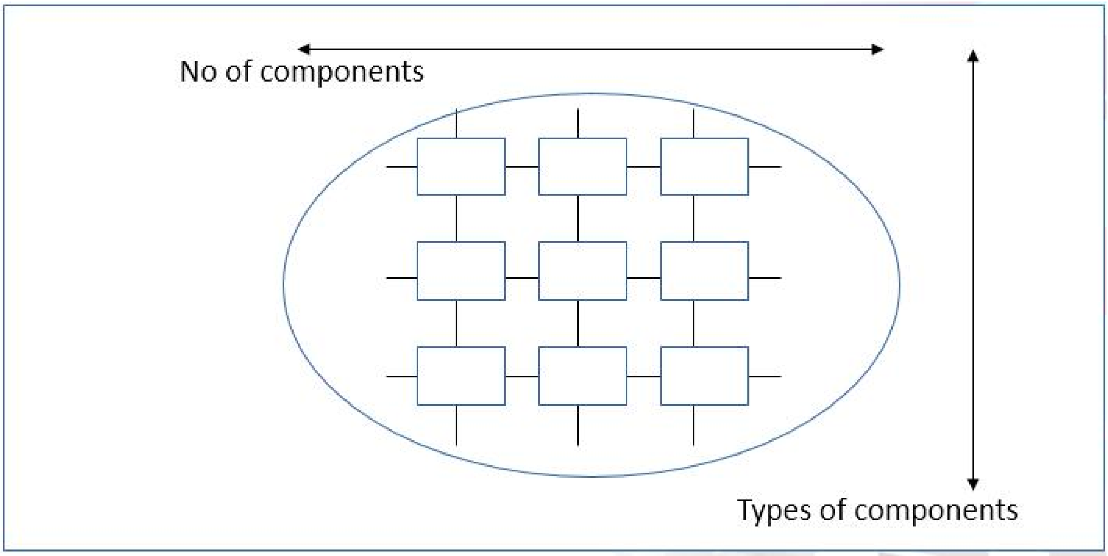
- Model consists of modules
- Modules communicate with message passing
- Modules are C++ files
  - Implement simulation class library
  - Run in simulation kernel

- Module types
  - Simple (active modules)
  - Compound

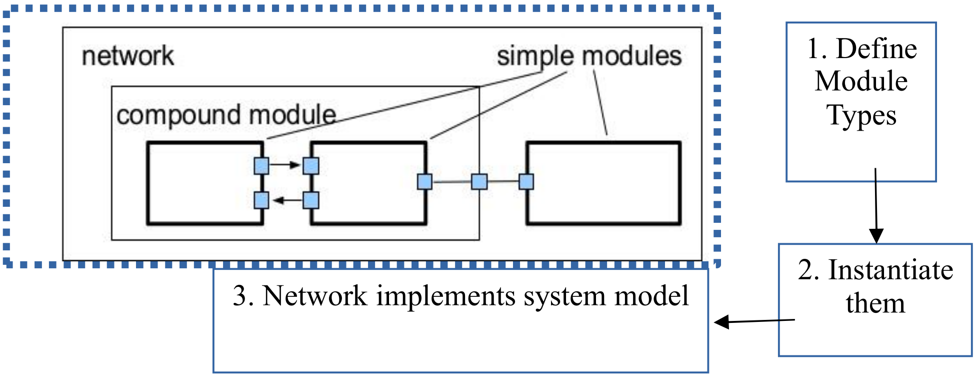
- Simple modules can be grouped into compound modules and so forth
- Modules communicate through gates (connections)
  - Directly between modules or through intermediaries



No. of hierarchy levels not limited



- **Gates**
  - **Gates**
  - Input output interfaces of modules
  - Allow message passing
  - Linked via connection (T_{PROP}, R_{DATA}, BER) Input

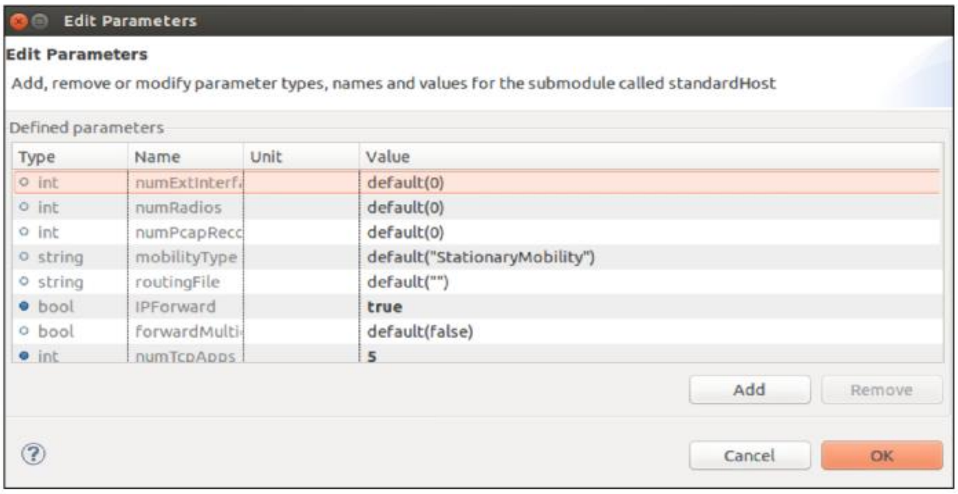


- **Channels**
  - Connection types with specific properties
  - Reusable at several places
  - Standard Host talking to another Standard Host via an Ethernet cable

- **Message**; tuple (time stamp, arbitrary data, ...)
- **Network**; A compound module with no external gates

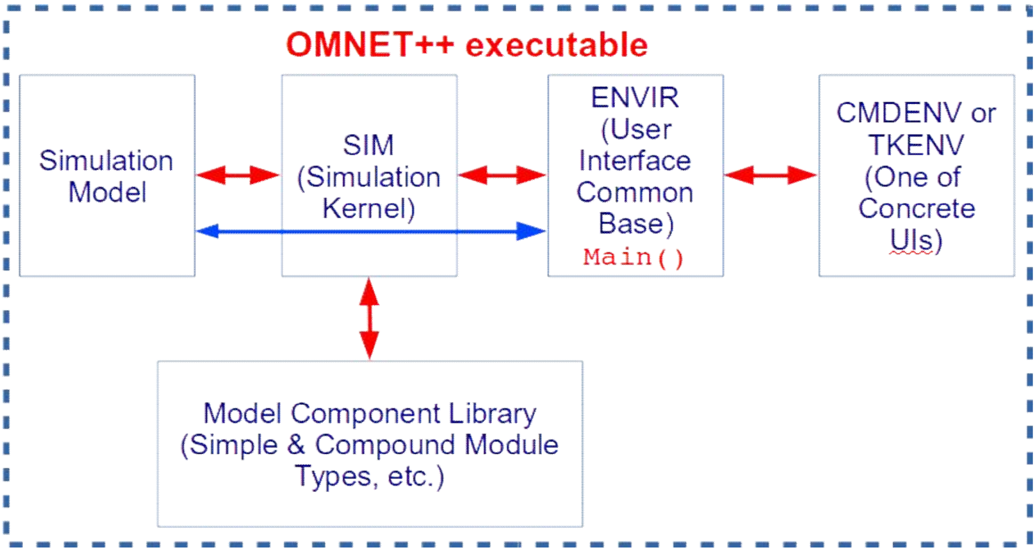
Module Parameters

- Pass configuration data to simple modules
- Define model topology
- String, numeric, boolean
- Constants, random numbers
- Expressions as references



Internal architecture of OMNET++

OMNeT++ simulation programs possess a modular structure.



Model Component Library

- Consists of the code of compiled simple and compound modules

Simulation Kernel & SIM Class Library

- Modules are instantiated and concrete simulation model is built by simulation kernel
- SIM covers most of the common simulation tasks through classes
  - Generate random number (distributions)
  - Queues (F IFO, priority)
- Messages (hold arbitrary data structures)
- Routing (explore topology, generate graph data structure)

Envir, Cmdenv and Tkenv Libraries

- Simulation executes in an environment
- Defines and determines
  - Where input data come from
  - Where simulation results go to
  - What happens to debugging output?
  - Controls the simulation execution
  - How model is visualized

What is NED Language?

- A network description language
- Creates network topologies in OMNeT++
- You create alternately create topology graphically as well
- Correspondingly NED source code is automatically generated

Typical Ingredients of NED description

- Network definitions
- Compound module definitions
- Simple module declarations

Network Definition

- Network definitions are compound modules
  - Self-contained simulation models

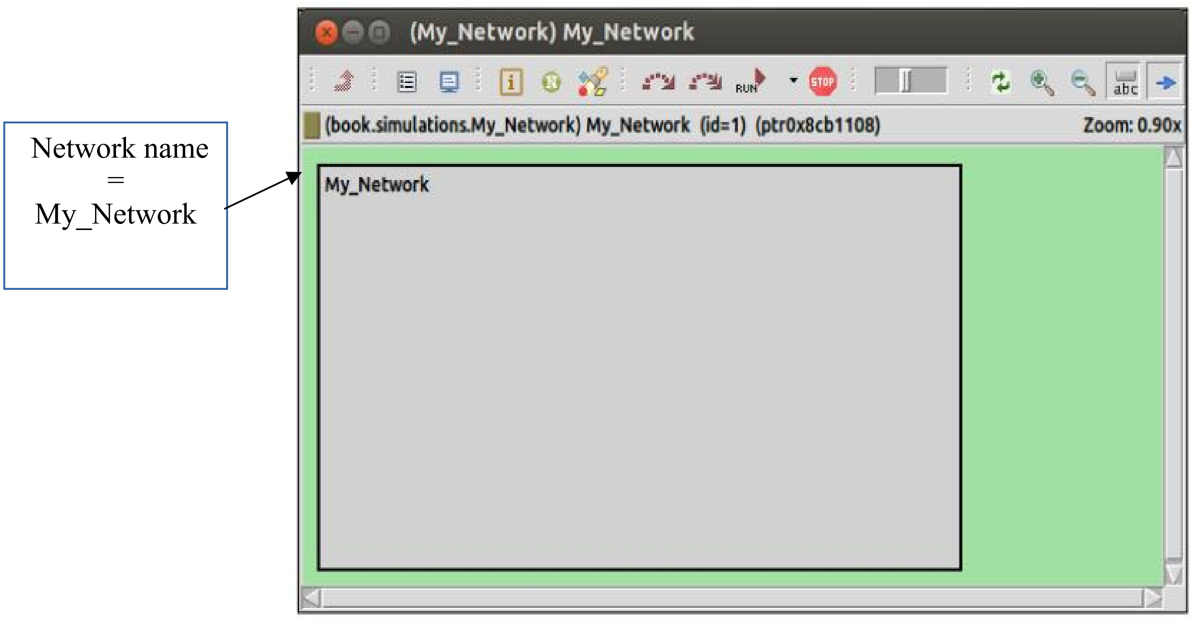
Simple Module Declaration

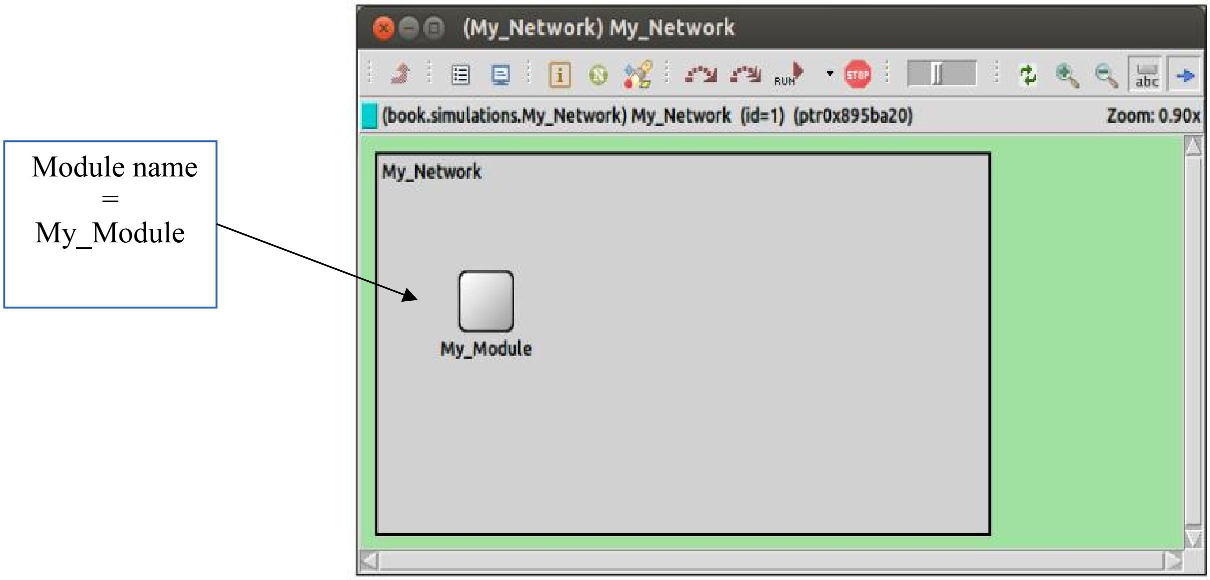
- Describes the interface of modules
  - Gates
  - Parameters

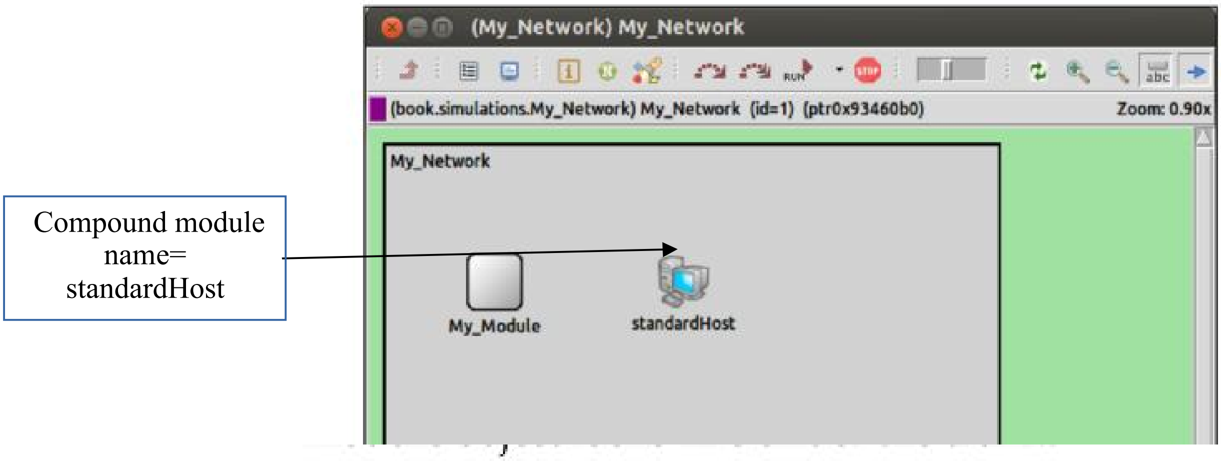
Compound module definitions

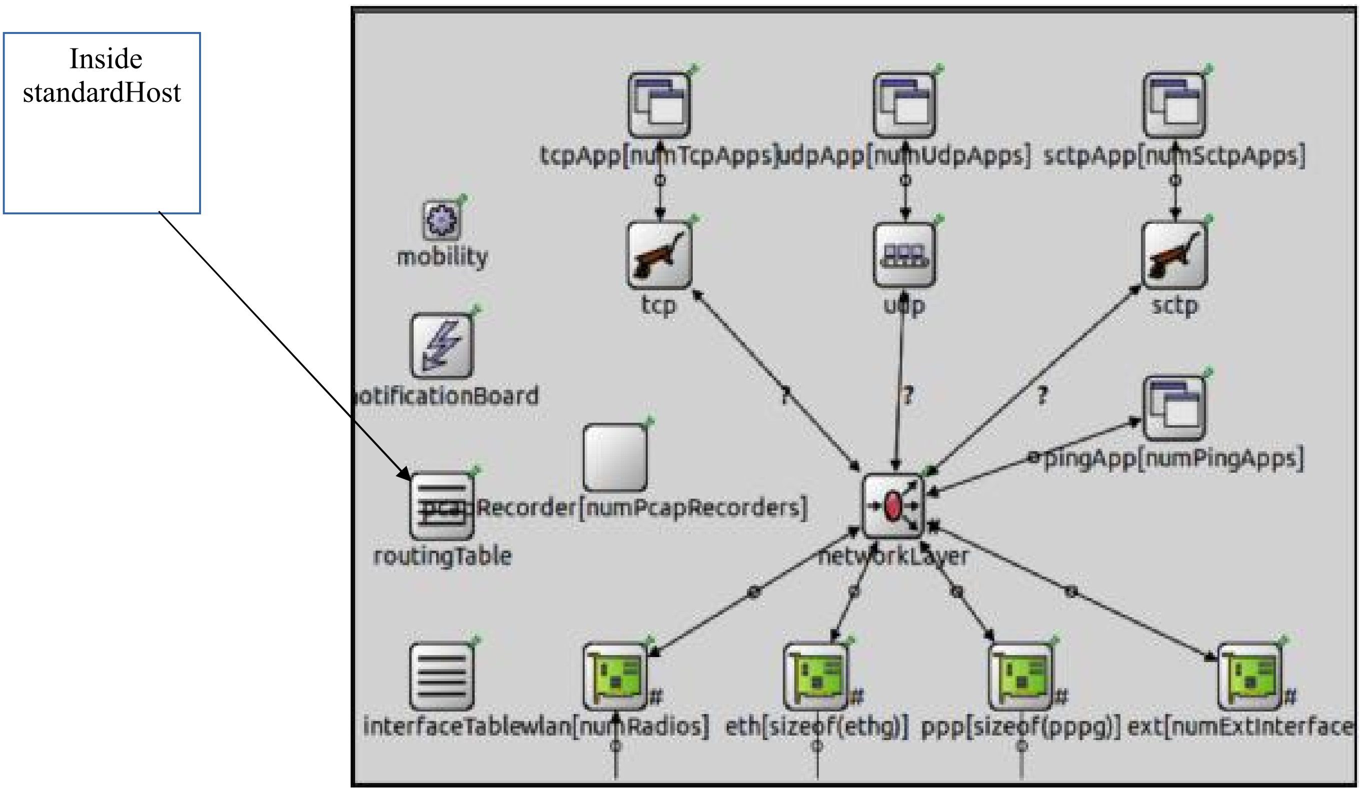
- Declaration of external interface
  - Gates
  - Parameters
- Definition of
  - Sub modules
  - Their interconnections

Let us create a topology called My_Network using Graphical Editor




Module name
=
My_Module
(My_Network) My_Network
(book.simulations.My_Network) My_Network (id=1) (ptr0x895ba20)
Zoom: 0.90x
My_Network
My_Module


Compound module
name=
standardHost
(My_Network) My_Network
(book.simulations.My_Network) My_Network (id=1) (ptr0x93460b0)
Zoom: 0.90x
My_Network
My_Module
standardHost

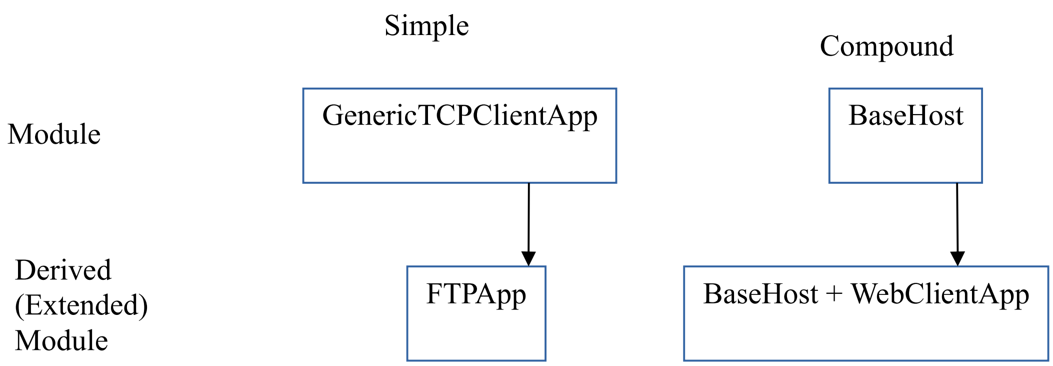

Inside
standardHost
tcpApp[numTcpApps]
udpApp[numUdpApps]
sctpApp[numSctpApps]
mobility
tcp
udp
sctp
otificationBoard
pingApp[numPingApps]
pcapRecorder[numPcapRecorders]
routingTable
networkLayer
interfaceTable
wlan[numRadios]
eth[sizeof(ethg)]
ppp[sizeof(pppg)]
ext[numExtInterface

\

More About NED Language

- Inheritance
- Modules and channels can be subclassed
- Derived modules and channels may add
  - New parameters
  - Gates
- Similarly compound modules may add
  - New parameters
  - Connections

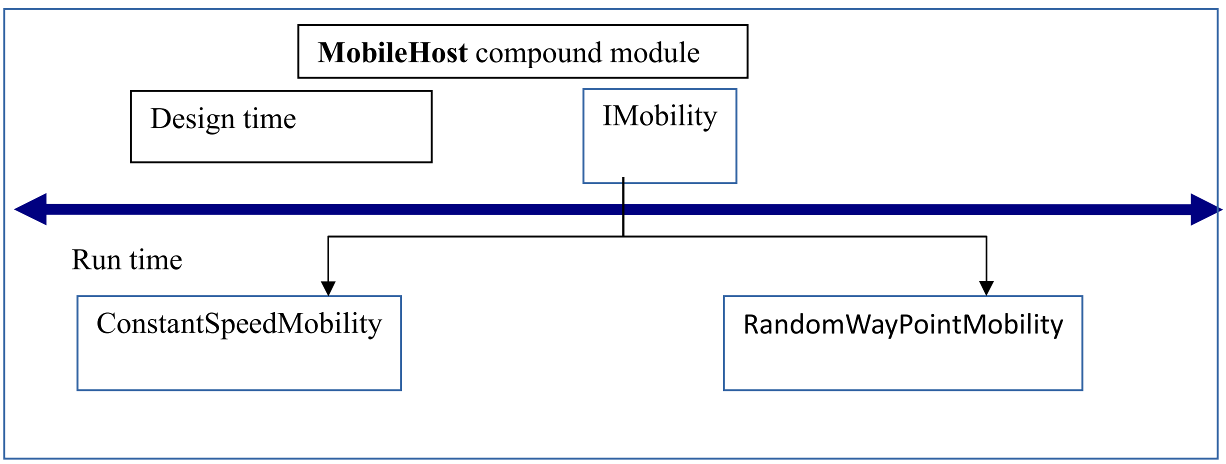
Example



Interface instantiation

- Module and channel interfaces can be used as a placeholder
  - where normally a module or channel type would be used
- Concrete module or channel type determined
  - At network setup time by a parameter

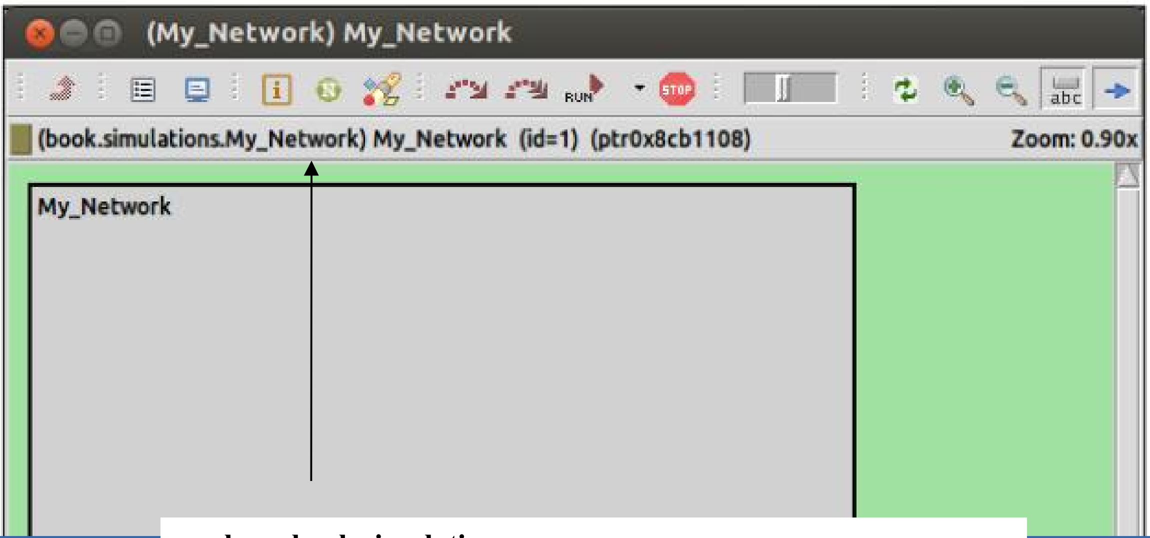
Example:



Packages

- Addresses name clashes between different models
- Simplifies specifying which NED files are needed by a specific simulation model

Example:



package book.simulations;

Package is a mechanism to organize various classes and files. The simulation project inside of OMNeT++ is called "**Book**" and this NED file is found in the "**simulations**" folder of the Project.

Separation of Model and Experiments

- Always good practice to try to separate different aspects of simulation
- **Model topology**
  - NED file
  - MSG file
- **Model behavior**
  - C++ code
- Provides cleaner model

Configuring simulations

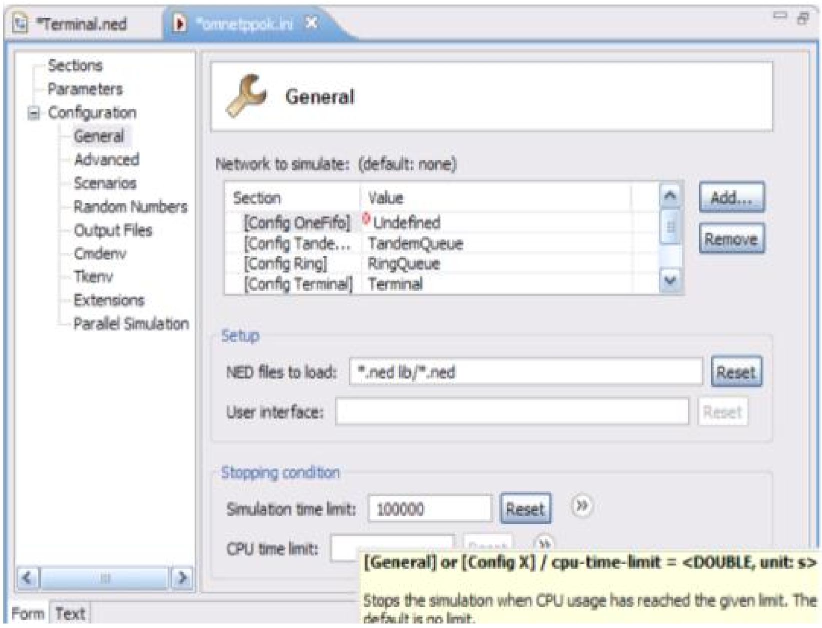
- How to capture the effect of different inputs?
  - Run to run variables
- C++ and NED code do not have such variables
- INI files provide a mechanism to specify these parameters
  - omnet.ini

INI File Syntax

- Basically an ASCII text file
- Consists of
  - Key-value pairs
    <key>=<value>

INI File Editor

- INI File Editor lets the user configure simulation models for execution
- Both form-based and source editing



INI File Editor

- Considers all NED declarations
  - Simple modules
  - Compound modules
  - Channels, etc
- Fully relates this information to the INI file contents
- Editor knows which INI file keys match which module parameters

Example omnet.ini

My_Network
wildcarded

No of Apps

```
[General]
network = book.simulations.My_Network
#We will make standardHost a TCP Session Application in
order for it to communicate#
**.standardHost.numTcpApps = 1
```

Example

Application Name

```
[General]
network = book.simulations.My_Network
#We will make standardHost a TCP Session Application in
order for it to communicate#
**.standardHost.numTcpApps = 1
**.standardHost.tcpApp[0].typename = "TCPSessionApp"
**.standardHost.tcpApp[0].connectAddress = "standardHost1"
**.standardHost.tcpApp[0].connectPort = 1000
#We will make standardHost1 a TCP Echo Application, this
means that it
will send #an echo packet once it receives a packet.
**.standardHost1.numTcpApps = 1
**.standardHost1.tcpApp[0].typename = "TCPEchoApp"
**.standardHost1.tcpApp[0].localPort = 1000
**.standardHost1.tcpApp[0].echoFactor = 3.0
#**.ppp[*].queueType = "DropTailQueue"
```

```
[General]
network = book.simulations.My_Network
#We will make standardHost a TCP Session Application in order for
it to communicate#
**.standardHost.numTcpApps = 1
**.standardHost.tcpApp[0].typename = "TCPSessionApp"
**.standardHost.tcpApp[0].connectAddress = "standardHost1"
**.standardHost.tcpApp[0].connectPort = 1000
#We will make standardHost1 a TCP Echo Application, this means
that it
will send #an echo packet once it receives a packet.
**.standardHost1.numTcpApps = 1
**.standardHost1.tcpApp[0].typename = "TCPEchoApp"
**.standardHost1.tcpApp[0].localPort = 1000
**.standardHost1.tcpApp[0].echoFactor = 3.0
#**.ppp[*].queueType = "DropTailQueue"
```

Who to connect with whom

Which port to connect to

```
[General]
network = book.simulations.My_Network
#We will make standardHost a TCP Session Application in order for it
to communicate
**.standardHost.numTcpApps = 1
**.standardHost.tcpApp[0].typename = "TCPSessionApp"
**.standardHost.tcpApp[0].connectAddress = "standardHost1"
**.standardHost.tcpApp[0].connectPort = 1000
#We will make standardHost1 a TCP Echo Application, this means
that it will send #an echo packet once it receives a packet.
**.standardHost1.numTcpApps = 1
**.standardHost1.tcpApp[0].typename = "TCPEchoApp"
**.standardHost1.tcpApp[0].localPort = 1000
**.standardHost1.tcpApp[0].echoFactor = 3.0
#**.ppp[*].queueType = "DropTailQueue"
#**.ppp[*].queue.frameCapacity = 10
#**.eth[*].queueType = "DropTailQueue"
```

Reply size = Echo Packet* EF

```
[General]
network = book.simulations.My_Network
#We will make standardHost a TCP Session Application in order for it to communicate
**.standardHost.numTcpApps = 1
**.standardHost.tcpApp[0].typename = "TCPSessionApp"
**.standardHost.tcpApp[0].connectAddress = "standardHost1"
**.standardHost.tcpApp[0].connectPort = 1000
#We will make standardHost1 a TCP Echo Application, this means that it will send #an echo packet once it receives a packet.
**.standardHost1.numTcpApps = 1
**.standardHost1.tcpApp[0].typename = "TCPEchoApp"
**.standardHost1.tcpApp[0].localPort = 1000
**.standardHost1.tcpApp[0].echoFactor = 3.0
#**.ppp[*].queueType = "DropTailQueue"
#**.ppp[*].queue.frameCapacity = 10
#**.eth[*].queueType = "DropTailQueue"
```

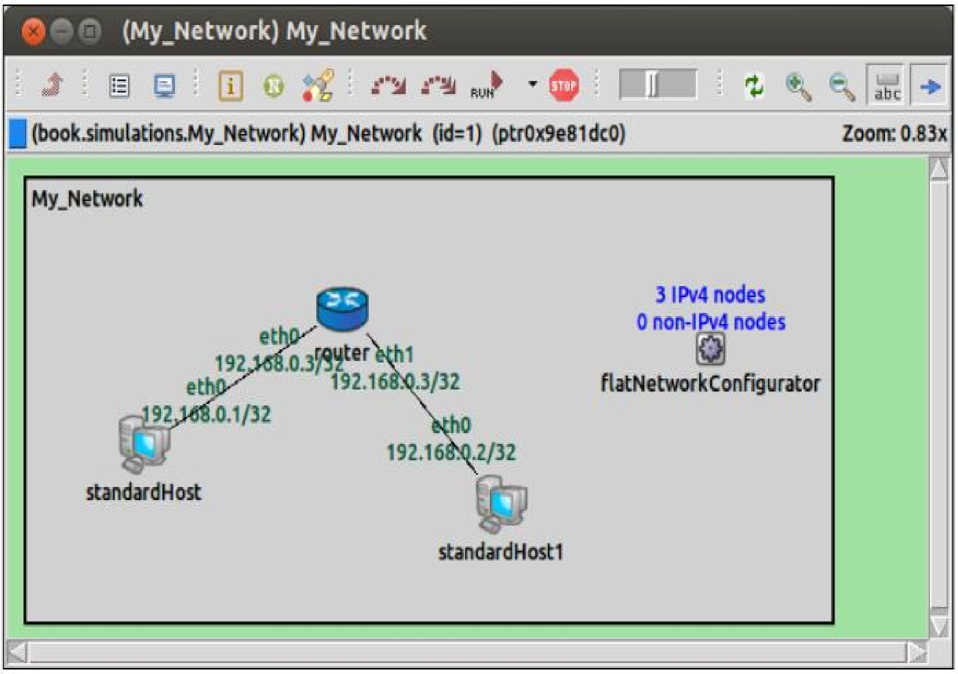
Queuing behaviour

```
[General]
network = book.simulations.My_Network
#We will make standardHost a TCP Session Application in order for it to communicate
**.standardHost.numTcpApps = 1
**.standardHost.tcpApp[0].typename = "TCPSessionApp"
**.standardHost.tcpApp[0].connectAddress = "standardHost1"
**.standardHost.tcpApp[0].connectPort = 1000
#We will make standardHost1 a TCP Echo Application, this means that it will send #an echo packet once it receives a packet.
**.standardHost1.numTcpApps = 1
**.standardHost1.tcpApp[0].typename = "TCPEchoApp"
**.standardHost1.tcpApp[0].localPort = 1000
**.standardHost1.tcpApp[0].echoFactor = 3.0
#**.ppp[*].queueType = "DropTailQueue"
#**.ppp[*].queue.frameCapacity = 10
#**.eth[*].queueType = "DropTailQueue"
```

Buffer Size

```
[General]
network = book.simulations.My_Network
#We will make standardHost a TCP Session Application in order for it to
communicate
**.standardHost.numTcpApps = 1
**.standardHost.tcpApp[0].typename = "TCPSessionApp"
**.standardHost.tcpApp[0].connectAddress = "standardHost1"
**.standardHost.tcpApp[0].connectPort = 1000
#We will make standardHost1 a TCP Echo Application, this means that it will
send #an echo packet once it receives a packet.
**.standardHost1.numTcpApps = 1
**.standardHost1.tcpApp[0].typename = "TCPEchoApp"
**.standardHost1.tcpApp[0].localPort = 1000
**.standardHost1.tcpApp[0].echoFactor = 3.0
#**.ppp[*].queueType = "DropTailQueue"
#**.ppp[*].queue.frameCapacity = 10
#**.eth[*].queueType = "DropTailQueue"
```

Example

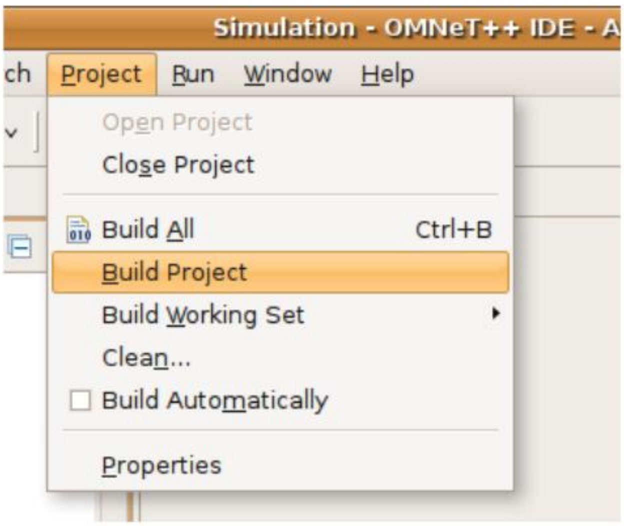


Building Simulation Programs

Using GUI Project Builder

- Initial build takes longer on indexing before building the project
- Dependency generation in the generated make files

– Classes, functions, methods, variables, macros



Using Mingwenv

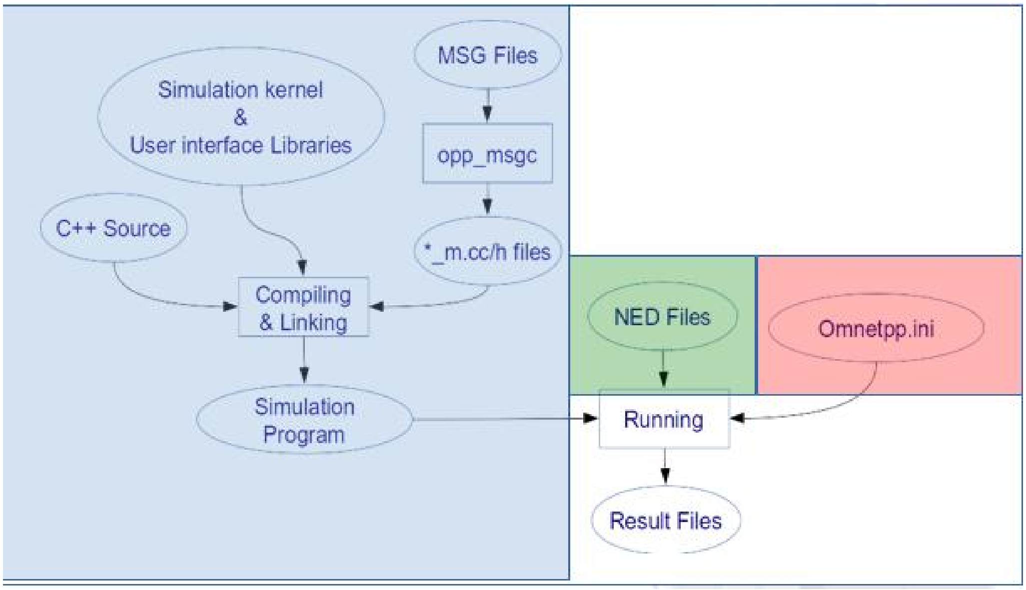
- Once you have the source files (*.ned, *.msg, *.cc, *.h) in a directory

– Change the working directory to there

- Type
  $ opp_makemake
- This will create a file named Makefile
- Type
  $ make
  Your simulation program should build

A makefile is used to tell the compiler which source files you want to compile. It'll also do things like name your executable and place it in a specific location.

Where to next!



Running Simulations

What is Simulation Run?

- Launch the built project make file

OMNET++ IDE Features

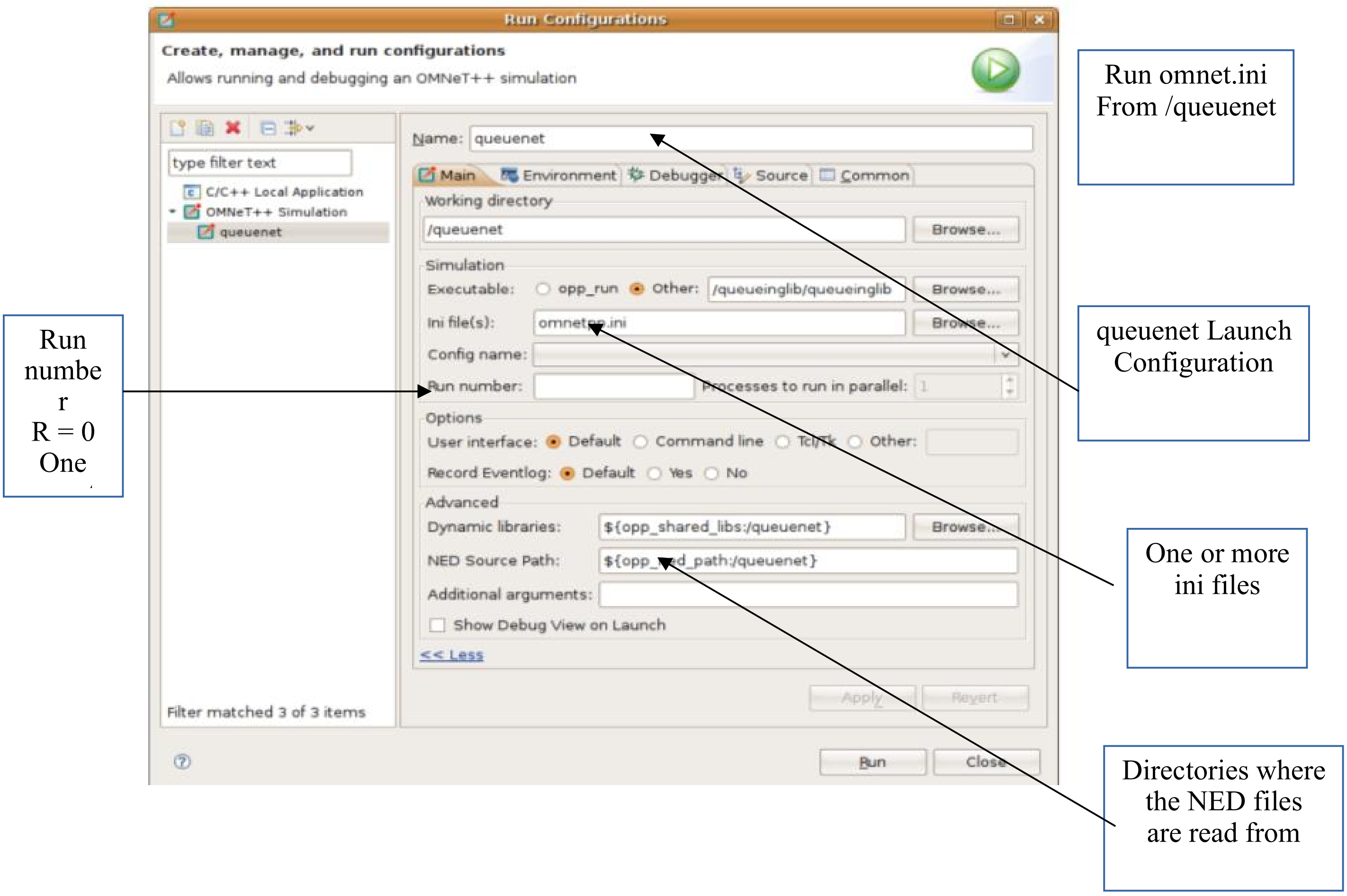
- Single runs
- Batch runs
- Run numbers
- Graphical mode (Tkenv)
- Command mode(Cmdenv)
- Simulation configuration
- Recording event logs
- Debug support

Quick Run

- In Project Explorer, select a project
- Clicking Run button on the toolbar
- Runs vary
  - Folder

- Runs if single ini file present
  - – ini file
    - Use *this* as the main ini file
  - – NED file
    - Scan for available ini file

Launch Configuration



Animation and Tracing

OMNeT++ is capable of

- Animating
  - – Flow of messages on network charts
- Reflecting
  - – State changes of the nodes in the display
- Animation is automatic
- No programming need for simulating engineer
- Suitable network simulations
  - – Rarely need fully customizable animation capabilities

Simulation Tracing

- Simple modules may write textual debug (trace) information like printf()
- OMNET++ provides Module output window
- – Special window to display output stream
- Eases following the module execution

Simulation Object Inspection

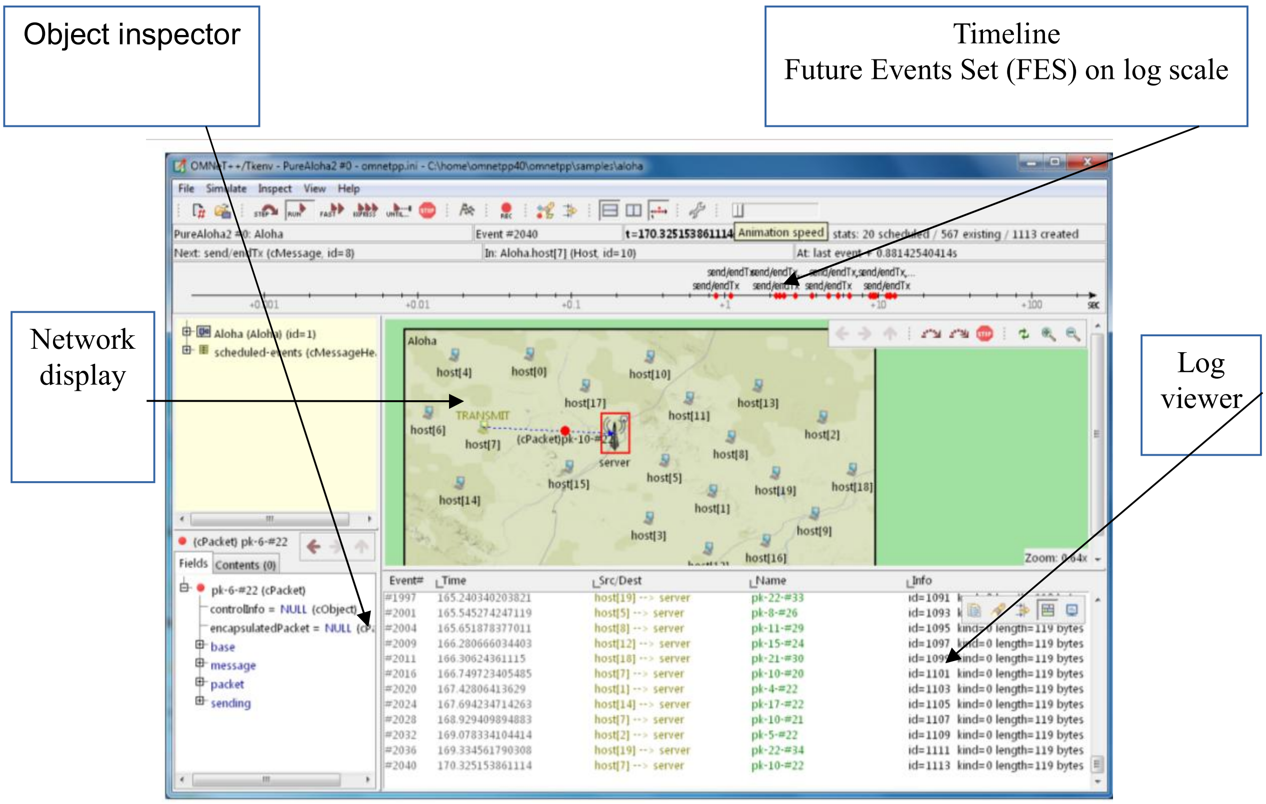
- An object inspector is a GUI window associated with a simulation object
  - Displays contents and properties
- Three types
  - Network Display
  - Log Viewer
  - Object Inspector

Tkenv

Tkenv is a graphical runtime interface for simulations

- It provides
  - Network visualization
  - Message flow animation
  - Log of message flow
  - Display of textual module logs
- Inspectors
- Visualization of statistics
  - Histograms, etc. during simulation execution
- Event log recording for later analysis

Tkenv in action



Organizing and Performing Experiments

Need for organizing experiments
Stuart Kurkow, "MANET Simulation
Studies:
The Incredibles," ACM's Mobile Computing and Communications
Review, 9(4):
50-61, 2005

Repeatable
- Fellow researcher should be able to repeat

Unbiased
- Results must not be specific to scenario used in experiment

Rigorous
- Scenarios & conditions for experiments must be truly representative

Statistically sound
- Experiments results must not violate mathematical principles

151 papers presented at MobiHoc (2000-2005)

114 of 151 (75.5%) are simulation-based papers

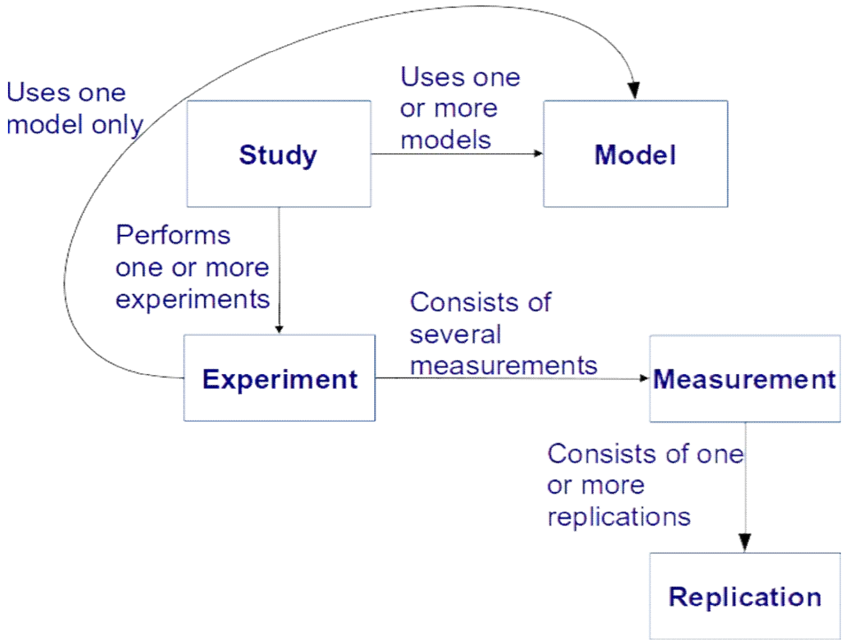
34 of 114 (29.8%) did not state simulator used

98 of 112 (87.5%) did not include confidence intervals

106 of 114 (93%) did not address initialization bias

etc.

Relationship between terminologies



How to organize experiments

Model

- The executable
- (C++ files & external libraries + NED files
- Invariant for the purpose of experimentation
- INI file not part of model

Study

- One or more experiments to investigate a phenomenon
- Usually many experiments
- One or more models

Experiment

- Exploration of a parameter space on a model
- Only and only one model

Measurement

- A set of simulation runs on the same model with same parameters
- Characterized by INI file
- But with different seeds
- May involve replication for averaging out

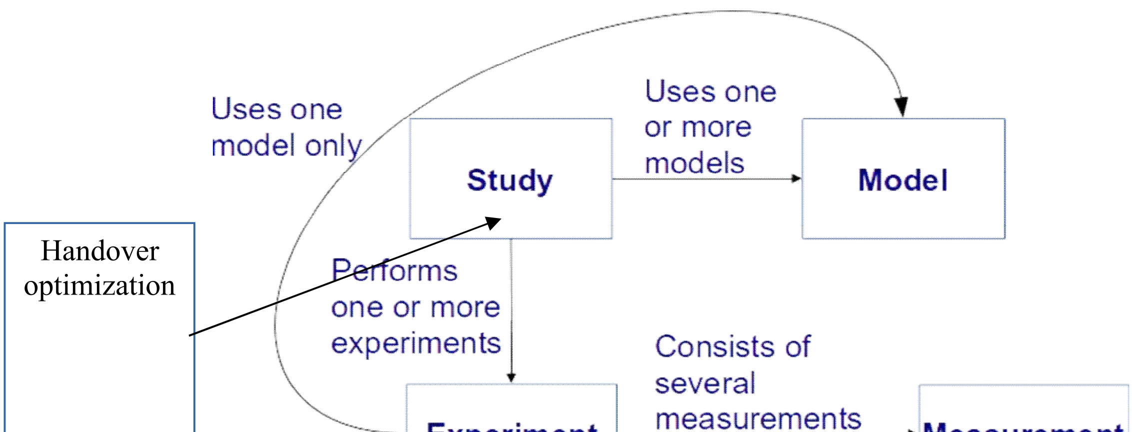
Replication

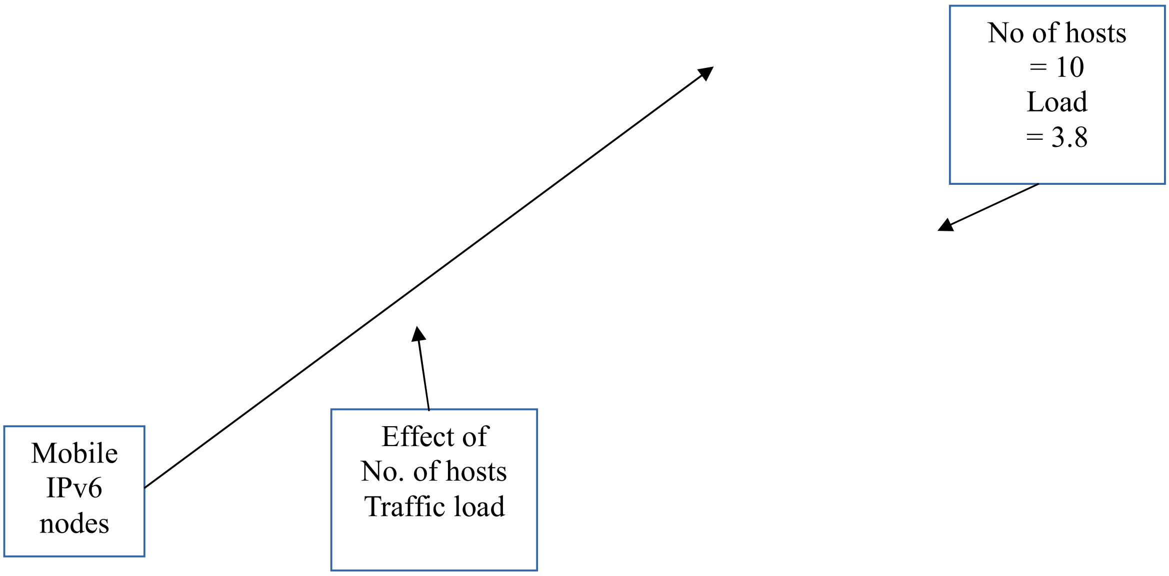
- One repetition of a measurement
- Replication can be characterized by the seed values it uses

Run

- One instance of running the simulation
- Characterized by exact time date
- computer (host name)

Example





Sequence Charts

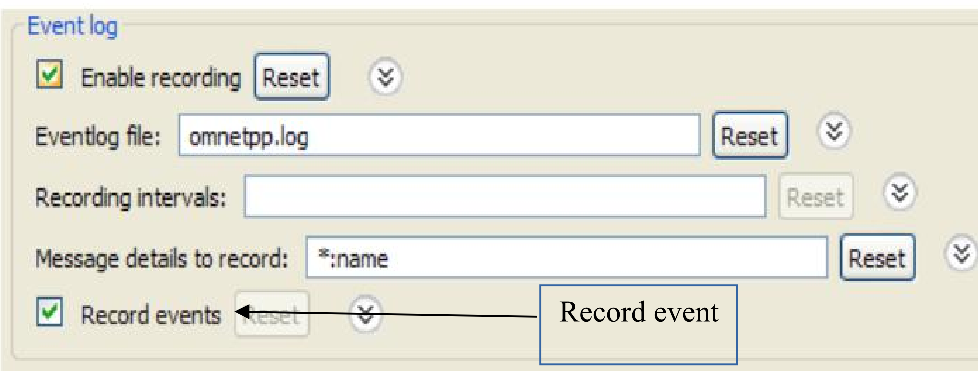
Event Log Tables

- An event log file contains
  - Tabulated log of messages sent during simulation
    - Between modules
    - Self-messages (timers)
  - Event details that prompts such sending or reception
- User can control
  - Amount of data recorded from messages
  - Start/stop time
  - Which modules to include in the log

Event Log File Creation (1 of 2)

- Type
  $ record-eventlog = true
- Output placed in
  /results directory
- Filename
  ${configname}-${runnumber}.elog

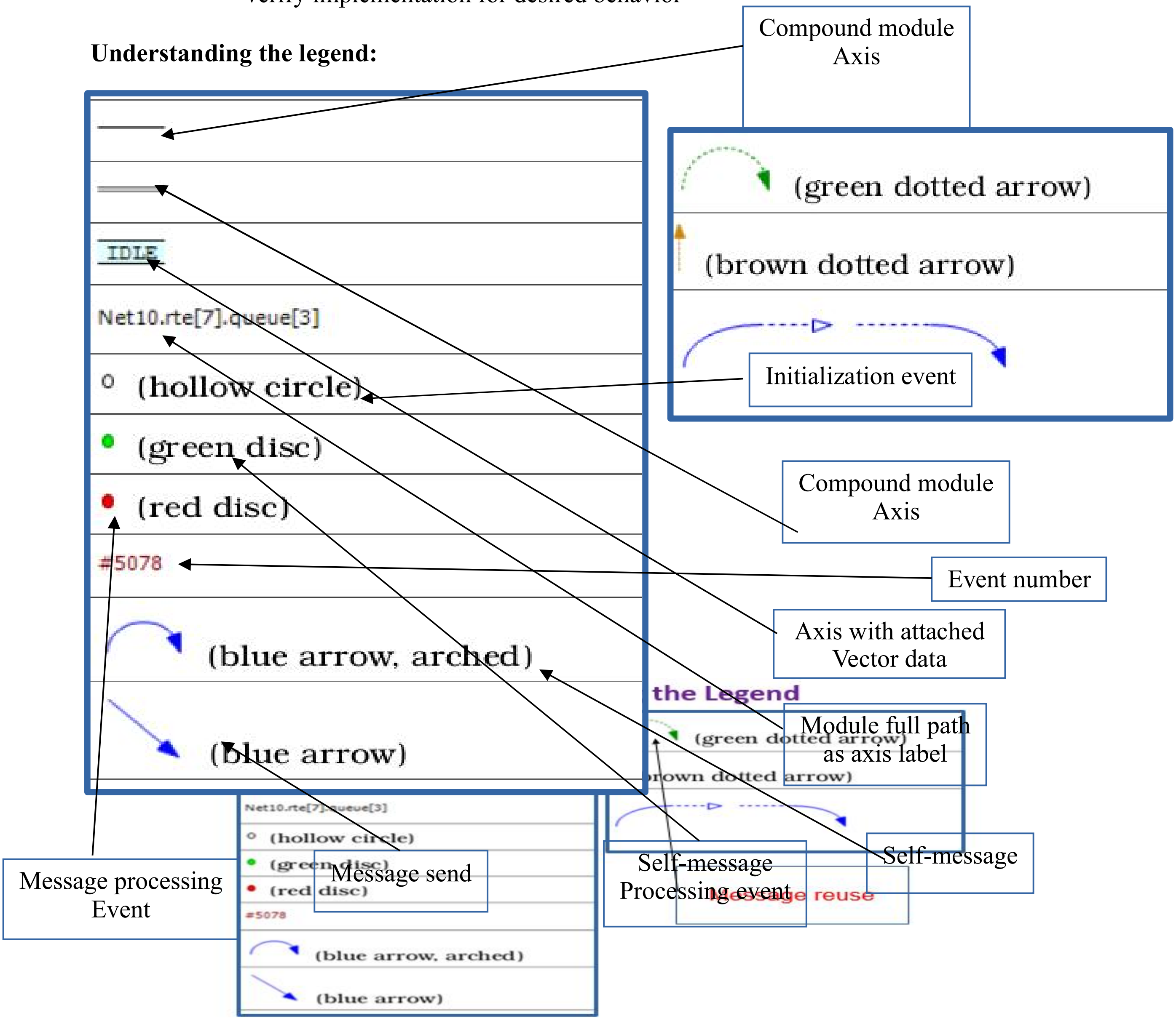
Using INI file event log configuration

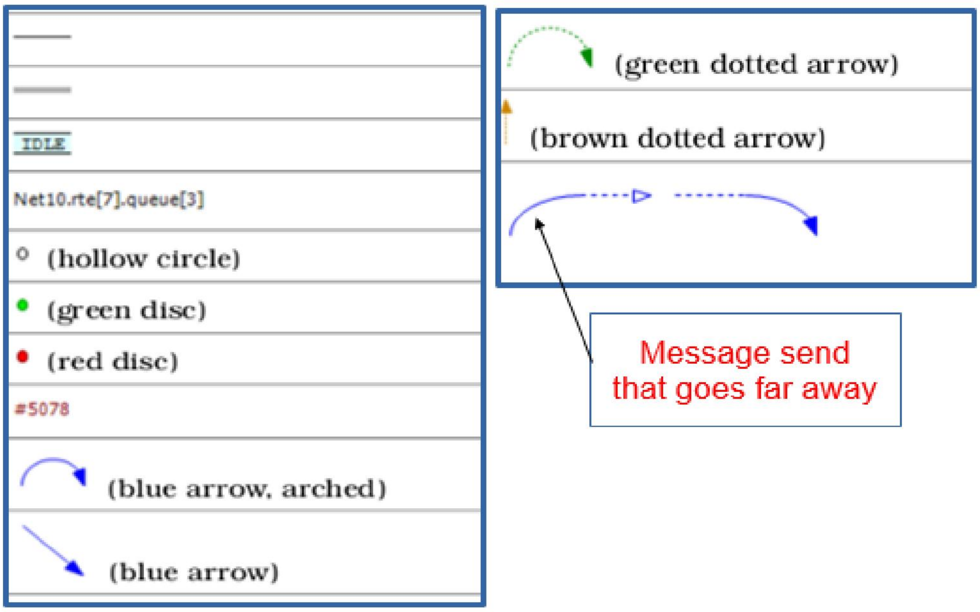


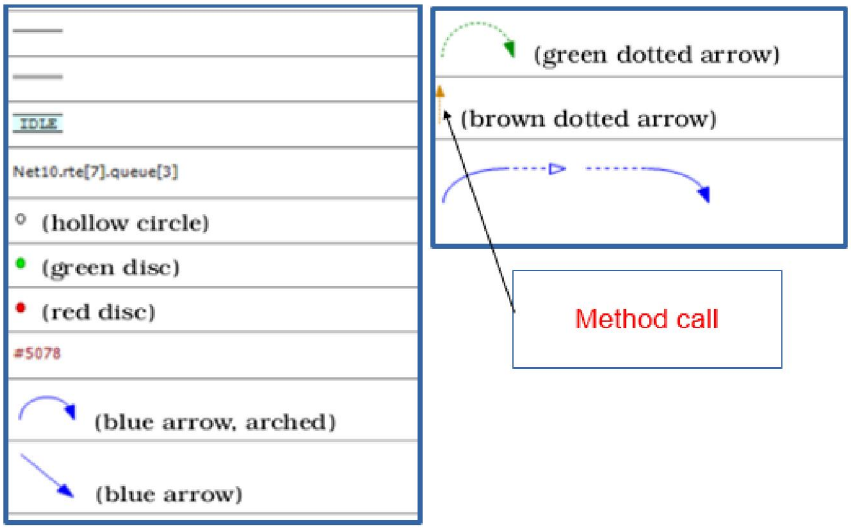
Sequence Chart

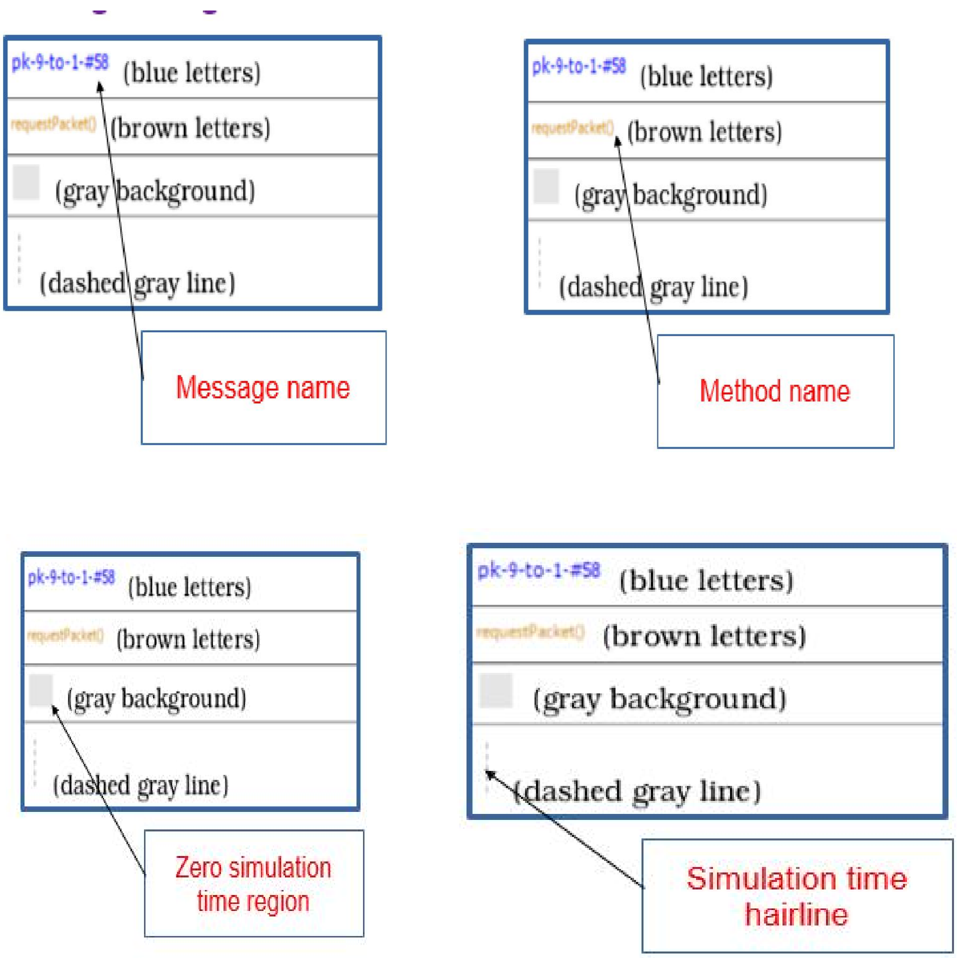
- Displays event log files in a graphical form
- Helps focus on causes & consequences of events/messages
- Helps users understand
  - Complex simulation models
  - Verify implementation for desired behavior

Understanding the legend:

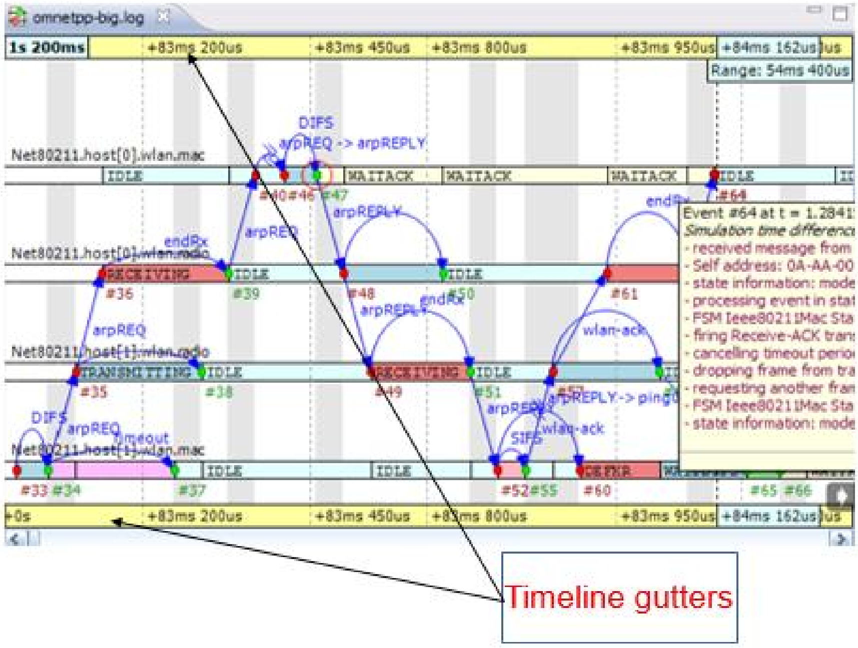


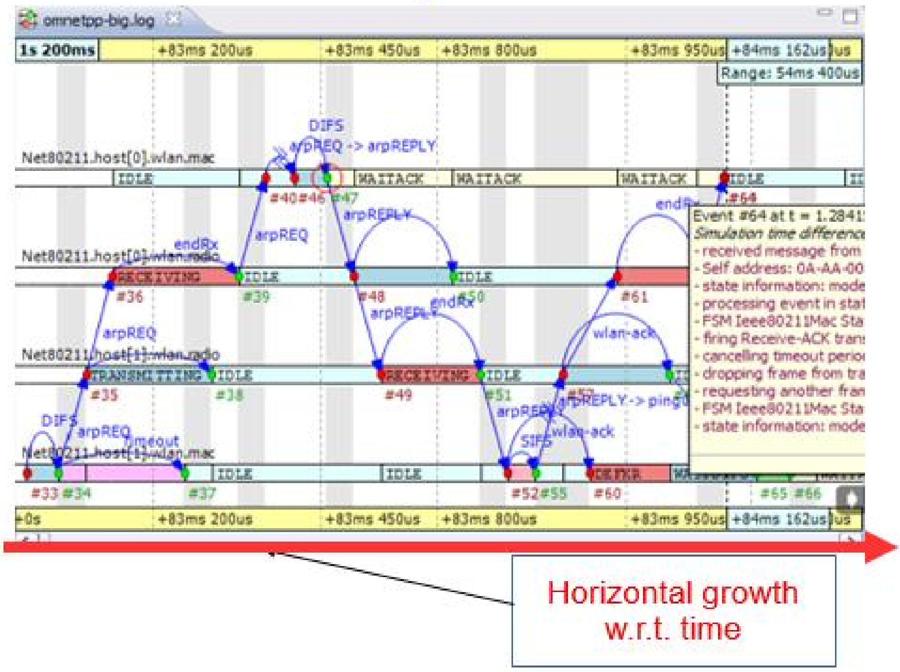

IDLE
Net10.rte[7].queue[3]
(hollow circle)
(green disc)
(red disc)
#5078
(blue arrow, arched)
(blue arrow)
(green dotted arrow)
(brown dotted arrow)
Message send that goes far away

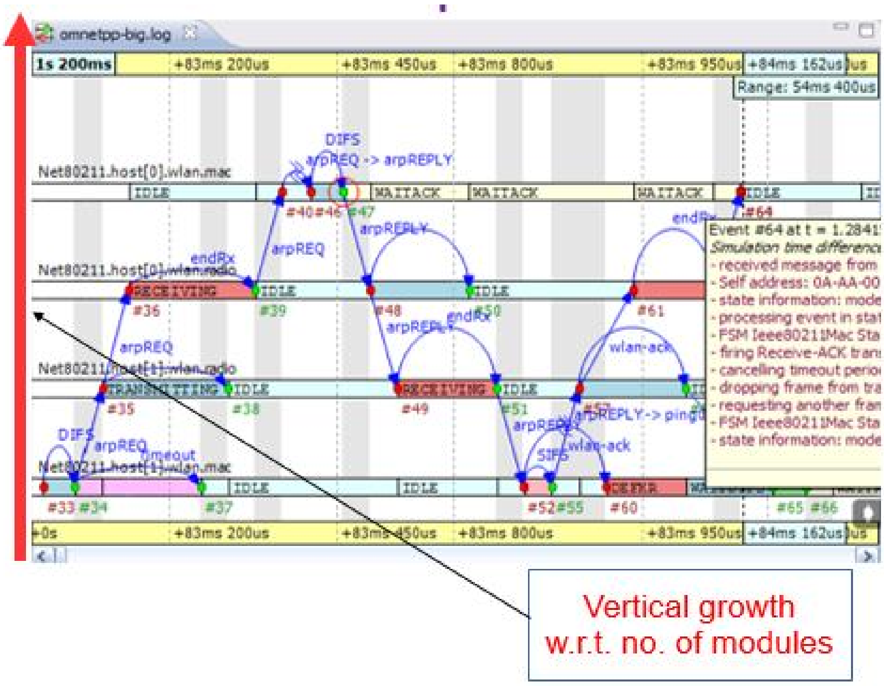

IDLE
Net10.rte[7].queue[3]
(hollow circle)
(green disc)
(red disc)
#5078
(blue arrow, arched)
(blue arrow)
(green dotted arrow)
(brown dotted arrow)
Method call

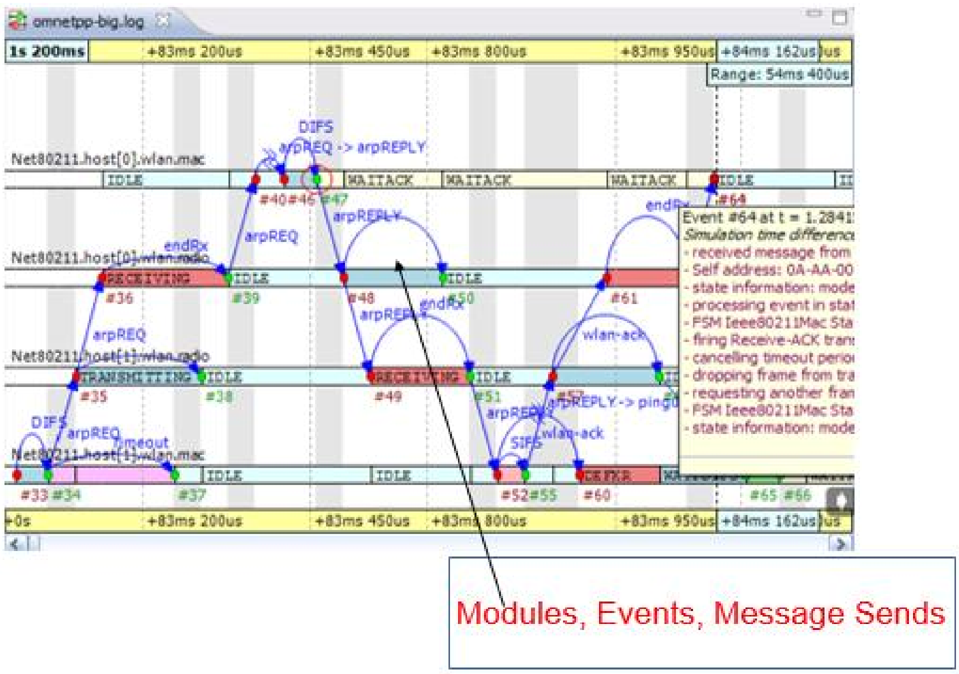


Parts of Sequence Charts


omnetpp-big.log
1s 200ms
+83ms 200us
+83ms 450us
+83ms 800us
+83ms 950us
+84ms 162us
Range: 54ms 400us
Net80211.host[0].wlan.mac
Net80211.host[0].wlan.radio
Net80211.host[1].wlan.radio
Net80211.host[1].wlan.mac
+0s
Timeline gutters


omnetpp-big.log
1s 200ms
+83ms 200us
+83ms 450us
+83ms 800us
+83ms 950us
+84ms 162us
Range: 54ms 400us
Net80211.host[0].wlan.mac
Net80211.host[0].wlan.radio
Net80211.host[1].wlan.radio
Net80211.host[1].wlan.mac
+0s
Horizontal growth w.r.t. time





What is Timeline?

- Simulation time mapped onto the horizontal axis
- Various ways
  – Intervals between interesting events often of different magnitudes
- Example
  – MAC (ms)
  – Higher layers (ms)

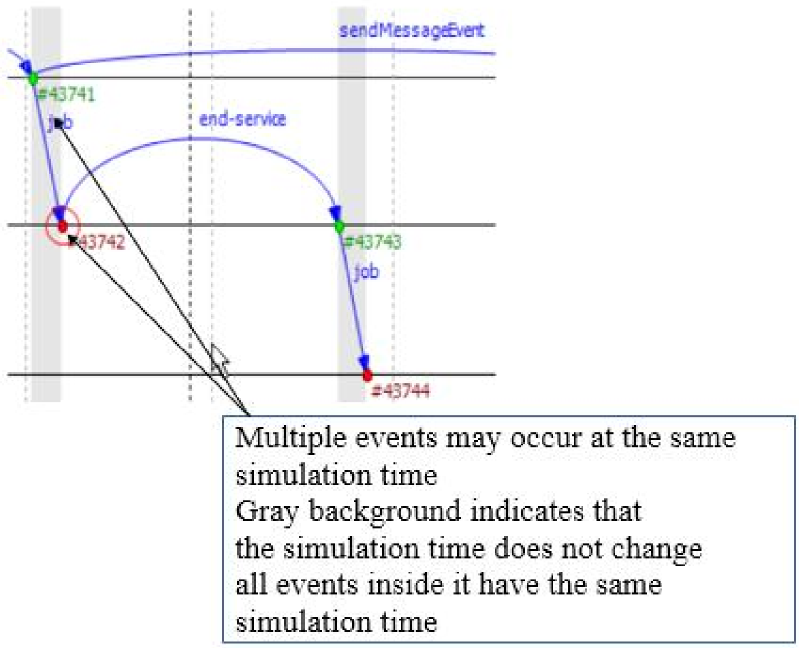
Types of Timeline

- **Linear**: simulation time proportional to distance measured in pixels
- **Event number**: event number proportional to the distance measured in pixels
- **Step**: distance between subsequent events is same
- **Nonlinear**: distance between subsequent events is nonlinear function of simulation time between them

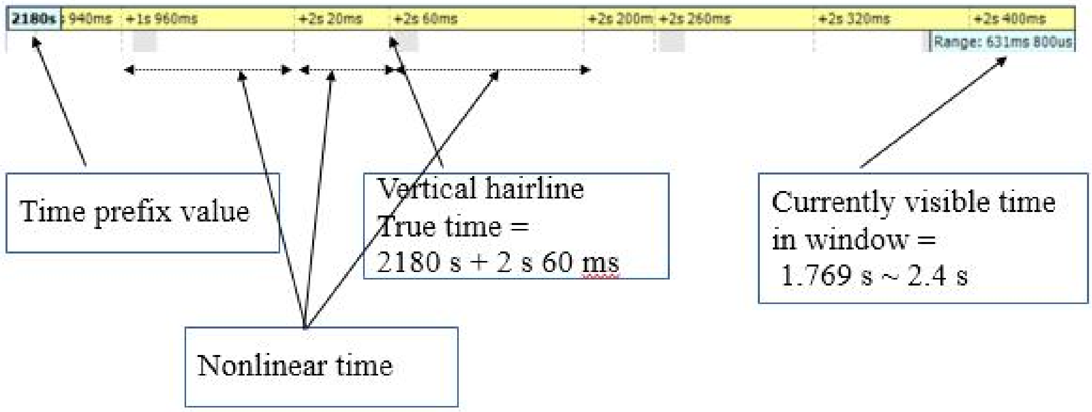
Interpreting Sequence Charts

- Zero Simulation Time Regions
- Gutter
- Events
- Messages
- Displaying Module State on Axes

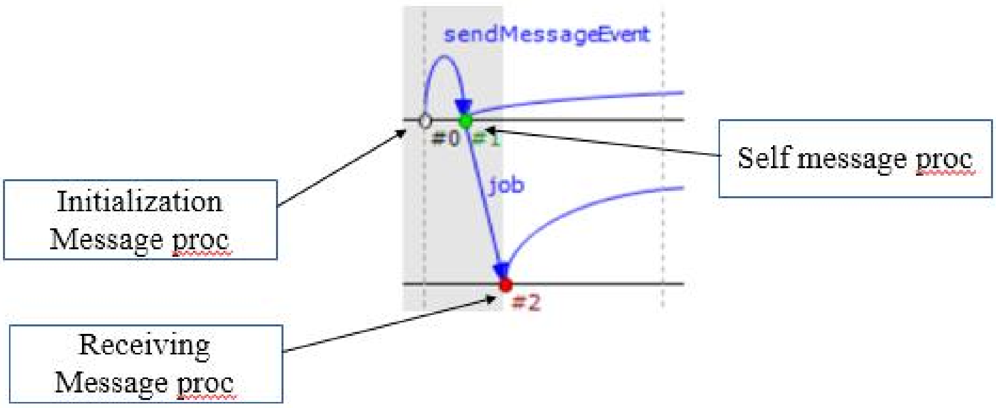
Zero Simulation Time Regions



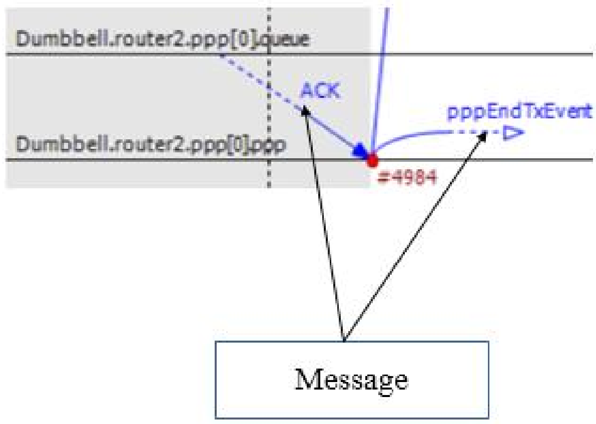
Gutter



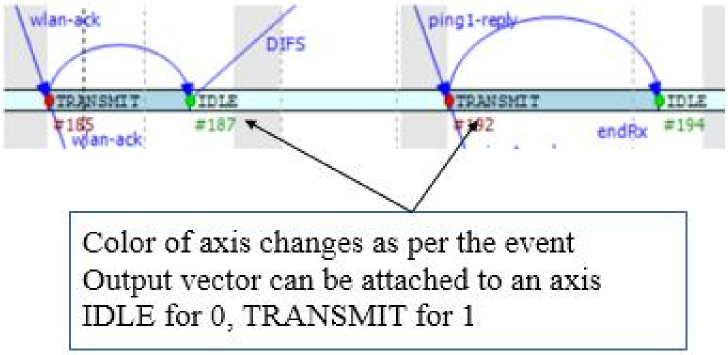
Events Processing



Messages



Displaying Module State on Axes



TicToc Tutorial

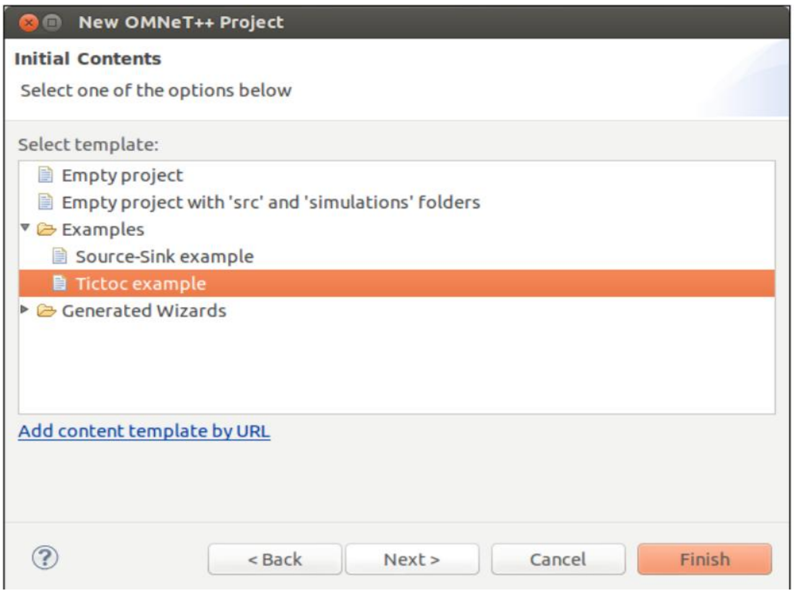
TicToc with 2-nodes

- Two nodes, Tic and Toc
- One node initializes by sending a message to the other
- Every time a node receives the message
    - Sends it back
- Continue indefinitely
    - Till user stops

Creating an empty project

- Open the OMNeT++ IDE
- Navigate to File | New | OMNeT++ Project
- Enter a Name for the project
- Next

Select the Tictoc example file in the Examples folder
You have created Tictoc example project



Opening NED file

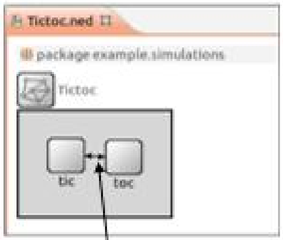
- In newly createdproject, navigate to the simulations folder in the Project Explorer

- Open Tictoc.ned

Understanding toctoc1.ned

```
package example.simulations;
import example.Txc;
//
// Two instances (tic and toc) of Txc connected.
//
network Tictoc
{
submodules:
tic: Txc;
toc: Txc;
connections:
tic.out --> {delay = 100ms;} --> toc.in;
tic.in <-- {delay = 100ms;} <-- toc.out;
}
```

Original simple module



Opening Simple Module

- Open project explorer
- Open src folder of this project
- Open Txc.ned

Understanding Txc.ned

```
package example;
//
// Immediately sends out any message it receives. It can optionally
generate
// a message at the beginning of the simulation, to bootstrap the
process.
//
simple Txc
{
parameters:
bool sendInitialMessage = default(false);
gates:
input in;
output out;
}
```

Implements Txc.cc

One input gate
One output gate

Opening Simple Module

- Open project explorer

- Open src folder of this project
- Open Txc.cc

Understanding Txc.ned

```
#include "Txc.h"
namespace example {
Define_Module(Txc);
void Txc::initialize()
{
if (par("sendInitialMessage").boolValue())
{
cMessage *msg = new cMessage("tictocMsg");
send(msg, ""out"");
}
}
void Txc::handleMessage(cMessage *msg)
{
// just send back the message we received
send(msg, "out"");
}
}; // namespace
```

OMNET++ Module

Initialize method

HandleMessage method

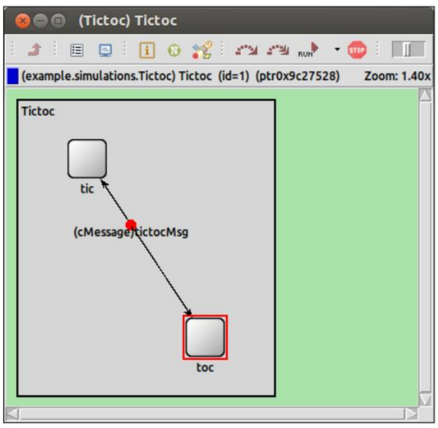
Echo back

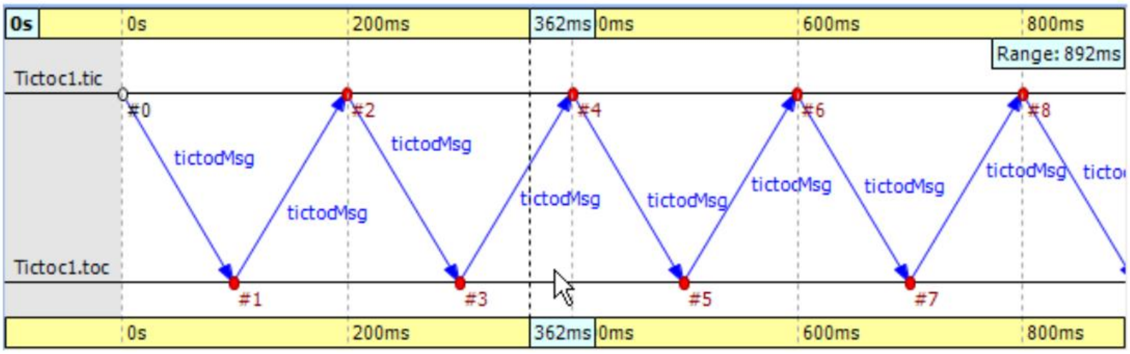
Understanding omnet.ini

```
[General]
network = Tictoc
cpu-time-limit = 60s
#debug-on-errors = true
**.tic.sendInitialMessage = true
```

Starts the activity

Compiling & Running on Tkenv





Extending TicToc

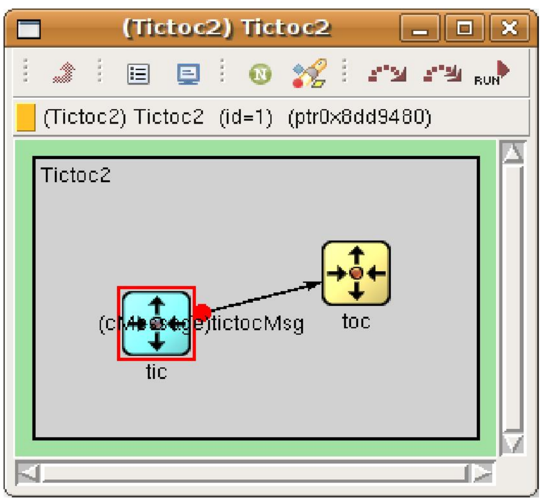
2. Refine graphics & add debugging output	7. Random numbers & parameters	12. Using two-way conanections
3. Add state variables	8. Timeout, Cancelling timers	13. Defining our message class
4. Adding parameters	9. Retransmitting same message	14. Displaying number of packets sent/received
5. Using inheritance	10. More than two nodes	15. Visualizing output scalars and vectors
6. Modeling processing delay	11. Channels & inner type definitions	16. Sequence charts and event logs

Refine graphics &

- Tictoc2.ned

Add debugging output

- Txc2.cc



Tictoc2.ned

```
// "block/routing" icon to the simple module. All
submodules of type
// Txc2 will use this icon by default
//
simple Txc2
{
    parameters:
        @display("i=block/routing"); // add a default
icon
    gates:
        input in;
        output out;
}
//
```

Add icon

```
// Make the two module look a bit different with
colorization effect.
// Use cyan for `tic', and yellow for `toc'.
//
network Tictoc2
{
    submodules:
        tic: Txc2 {
            parameters:
                @display("i=,cyan"); // do not change
the icon (first arg of i=) just colorize it
        }
        toc: Txc2 {
            parameters:
                @display("i=,gold"); // here too
        }
    connections:
```

Change color

```
class Txc2 : public cSimpleModule
{
  protected:
    virtual void initialize();
    virtual void handleMessage(cMessage *msg);
```

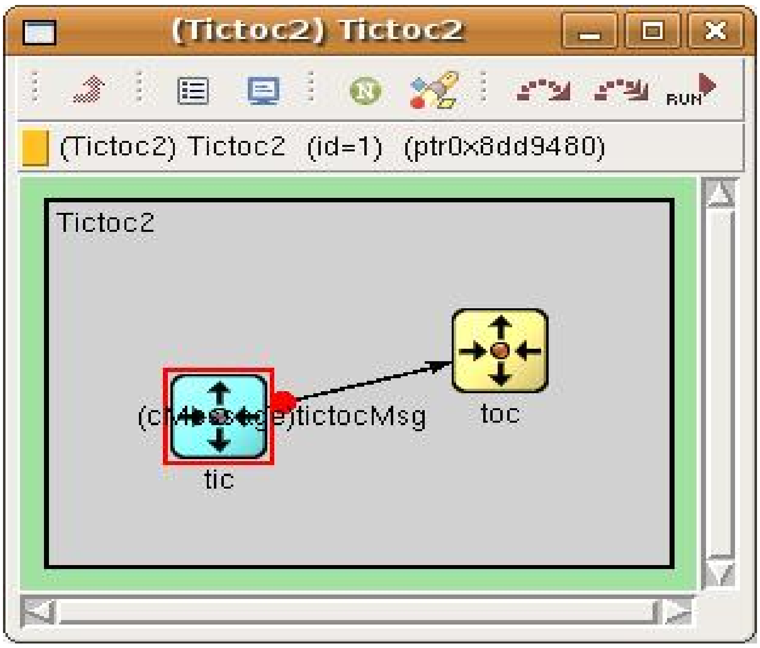
```
};
Define_Module(Txc2);

void Txc2::initialize()
{
    if (strcmp("tic", getName()) == 0)
    {
        // The 'ev' object works like 'cout' in C++.
        EV << "Sending initial message\n";
        cMessage *msg = new cMessage("tictocMsg");
        send(msg, "out");
    }
}

  void Txc2::handleMessage(cMessage *msg)
  {
      // msg->getName() is name of the msg object, here
  it will be "tictocMsg".
      EV << "Received message `" << msg->getName() <<
  "', sending it out again\n";
      send(msg, "out");
  }
```

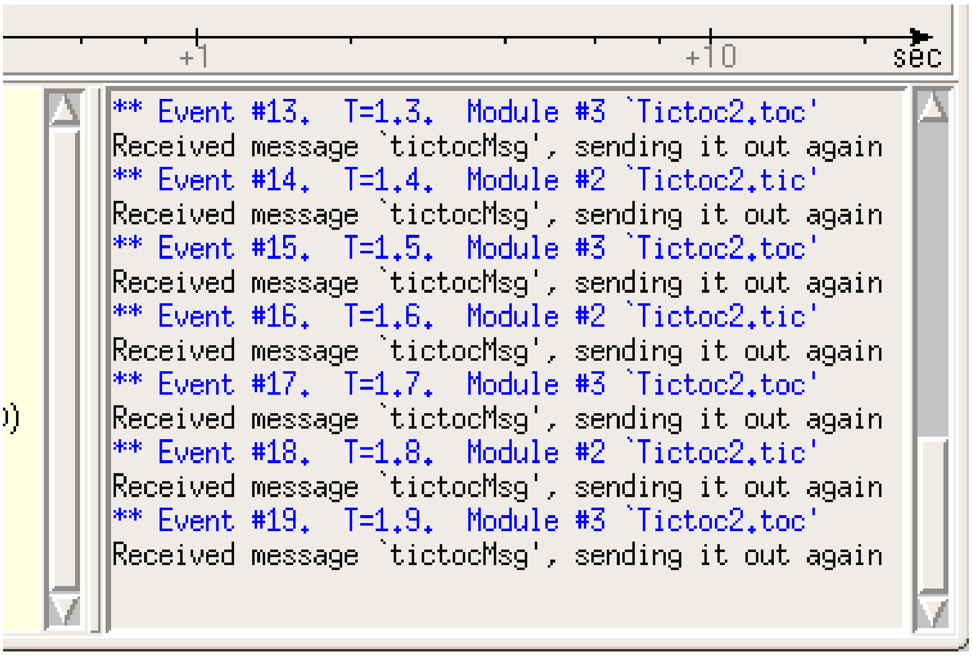
Debug information

Message name

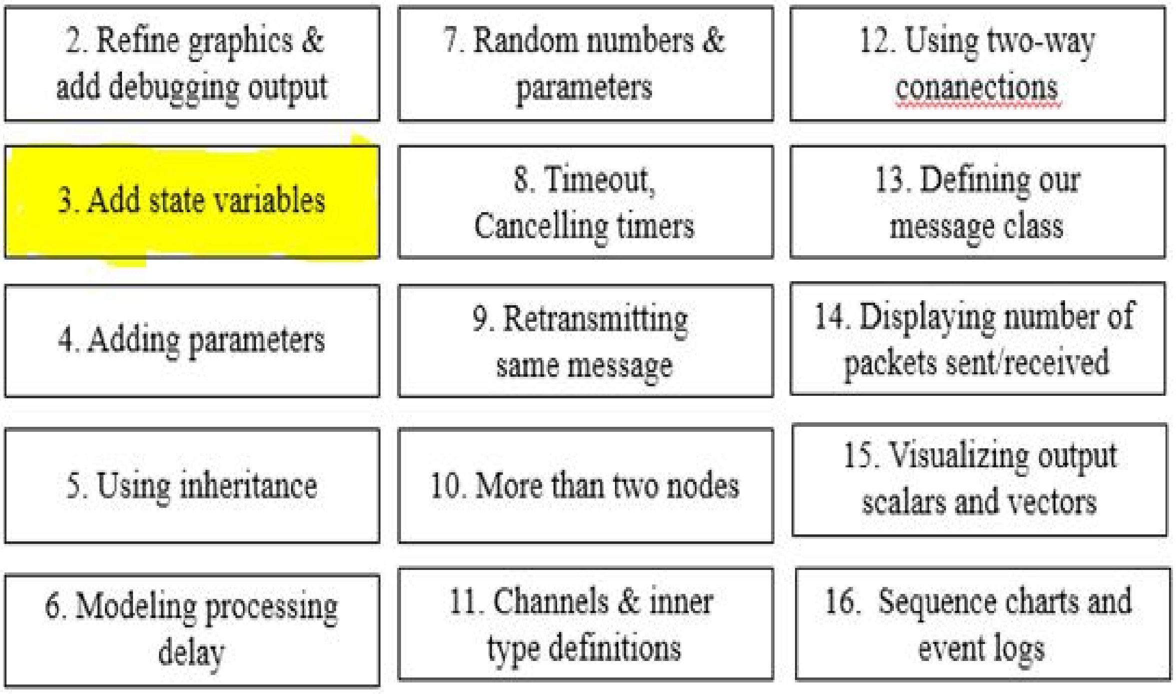
Debug information

Tkenv output





Extending TicToc



Add State Variables

- Add a counter as a class member to the module
- Delete the message after 10 exchanges
- Txc3.cc

Txc3.cc

```
class Txc3 : public cSimpleModule
{
  private:
    int counter;  // Note the counter here
  protected:
    virtual void initialize();
    virtual void handleMessage(cMessage *msg);
};
Define_Module(Txc3);

void Txc3::initialize()
{
// Initialize counter to ten.
   counter = 10;
   WATCH(counter);
```

Counter

Let you examine the variable under Tkenv.

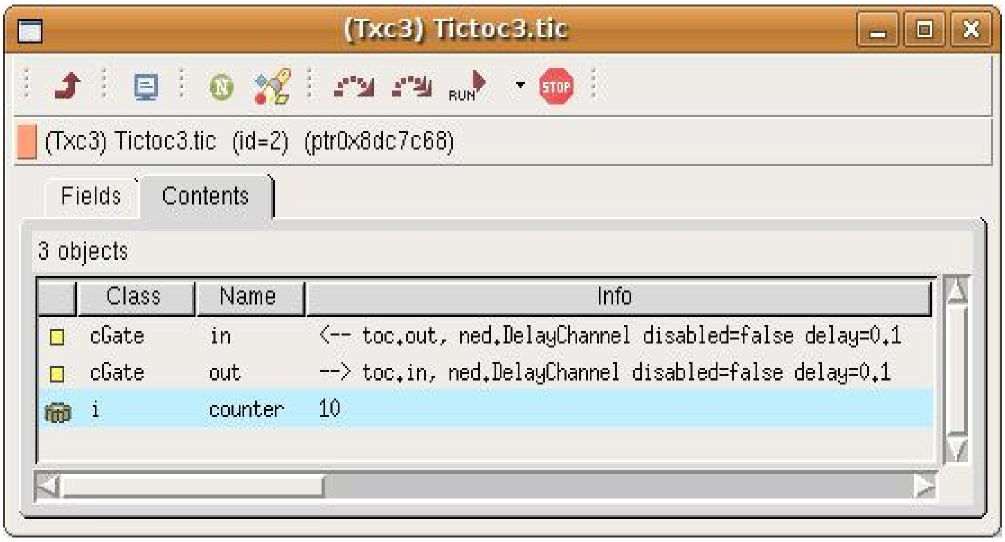
```
    if (strcmp("tic", getName()) == 0)
    {
        EV << "Sending initial message\n";
        cMessage *msg = new cMessage("tictocMsg");
        send(msg, "out");
    }
}
void Txc3::handleMessage(cMessage *msg)
{
    // Decrement counter and check value.
    counter--;
    if (counter==0)
    {
   EV << getName() << "'s counter reached zero,
       deleting message\n";
       delete msg;


          }
          else
          {
          EV << getName() << "'s counter is " <<
              counter << ", sending back message\n";
          send(msg, "out");
          }
      }
```

Decrement counter

If counter is zero, delete message

Or show current counter value

Output:



Adding parameters

2. Refine graphics & add debugging output	7. Random numbers & parameters	12. Using two-way conanections
3. Add state variables	8. Timeout, Cancelling timers	13. Defining our message class
4. Adding parameters	9. Retransmitting same message	14. Displaying number of packets sent/received
5. Using inheritance	10. More than two nodes	15. Visualizing output scalars and vectors
6. Modeling processing delay	11. Channels & inner type definitions	16. Sequence charts and event logs

Adding parameters

- Add input parameters to the simulations
  - Count = 10 now into a parameter that the user can define
- tictoc4.ned
- Txc4.cc
- Omnet.ini

Boolean parameter (decides if module should send out first message in its initialization code)
- tictoc4.ned
- Txc4.cc
- Omnet.ini

tictoc4.ned

```
// Adding module parameters.
//
network Tictoc4
{
    submodules:
        tic: Txc4 {
            parameters:
                sendMsgOnInit = true;
                @display("i=,cyan");
        }
        toc: Txc4 {
            parameters:
                sendMsgOnInit = false;
                @display("i=,gold");
        }
```

Network

Assign values to parameters

```
simple Txc4
{
    parameters:
        bool sendMsgOnInit = default(false); //
whether the module should send out a message on
initialization
        int limit = default(2);   // another parameter
with a default value
        @display("i=block/routing");
    gates:
        input in;
        output out;
}
```

Simple module

Parameters
Send message on Initialization
Limit parameter

```
void Txc4::initialize()
{
    // Initialize the counter with the "limit" module
parameter, declared in the NED file (tictoc4.ned).
    counter = par("limit");

    // we no longer depend on the name of the module
   to decide whether to send an initial message
    if (par("sendMsgOnInit").boolValue() == true)
    {
        EV << "Sending initial message\n";
        cMessage *msg = new cMessage("tictocMsg");
        send(msg, "out");
    }
}
```

Takes counter value
From limit
Makes initialization
Independent of tic & toc

```
Tictoc4.toc.limit = 5

// or Tictoc4.t*c.limit=5

// or Tictoc4.*.limit=5

// or **.limit=5
```

Value assignment to
limit parameter
Through ini file
(wildcard support)

Using Inheritance

2. Refine graphics & add debugging output	7. Random numbers & parameters	12. Using two-way conanections
3. Add state variables	8. Timeout, Cancelling timers	13. Defining our message class
4. Adding parameters	9. Retransmitting same message	14. Displaying number of packets sent/received
5. Using inheritance	10. More than two nodes	15. Visualizing output scalars and vectors
6. Modeling processing delay	11. Channels & inner type definitions	16. Sequence charts and event logs

Using Inheritance

- What is different between tic and toc?
  - Parameter values
  - Display string
- Inheritance allows to create a simple module
  - Then derive modules from ittictoc5.ned

tictoc5.ned

```
simple Toc5 extends Txc5
{
    parameters:
        @display("i=,gold");
        sendMsgOnInit = false;  // Tic modules should
                          // not send a message on init
}
```

Declare toc

Assign value

```
simple Tic5 extends Txc5
{
    parameters:
        @display("i=,cyan");
        sendMsgOnInit = true;  // Tic modules should
                              //send a message on init
}
```

Declare tic

Assign value

```
simple Txc5
{
    parameters:
        bool sendMsgOnInit = default(false);
        int limit = default(2);
        @display("i=block/routing");
    gates:
        input in;
        output out;
}
```

Base module

Generalized parameters

```
network Tictoc5
{
    submodules:
        tic: Tic5;   // the limit parameter is still unbound here.
We will get it from the ini file
        toc: Toc5;
    connections:
```

Create network

Use them as submodule

Modeling processing delay

2. Refine graphics & add debugging output	7. Random numbers & parameters	12. Using two-way connections
3. Add state variables	8. Timeout, Cancelling timers	13. Defining our message class
4. Adding parameters	9. Retransmitting same message	14. Displaying number of packets sent/received
5. Using inheritance	10. More than two nodes	15. Visualizing output scalars and vectors
6. Modeling processing delay	11. Channels & inner type definitions	16. Sequence charts and event logs

Modeling processing delay

- So far, no processing delay in tictoc
- We need timer in
- Tictoc module to send itself "Event" message
- tictoc6.ned
- txc6.c

Strategy

- Initialize after 5 seconds
- Hold the message for 1 simulated second
  - Send a message to itself
  - Send it back
- Need to add two variables to the class
  - event
  - tictocMessage

When to send

What to send

tx6.cc

```
void Txc6::initialize()
{
// Create the event object (ordinary message) for
//timing
event = new cMessage("event");
tictocMsg = NULL;
if (strcmp("tic", getName()) == 0)
EV << "Scheduling first send to t=5.0s\n";
tictocMsg = new cMessage("tictocMsg");
scheduleAt(5.0, event);
}
}
```

Defining event

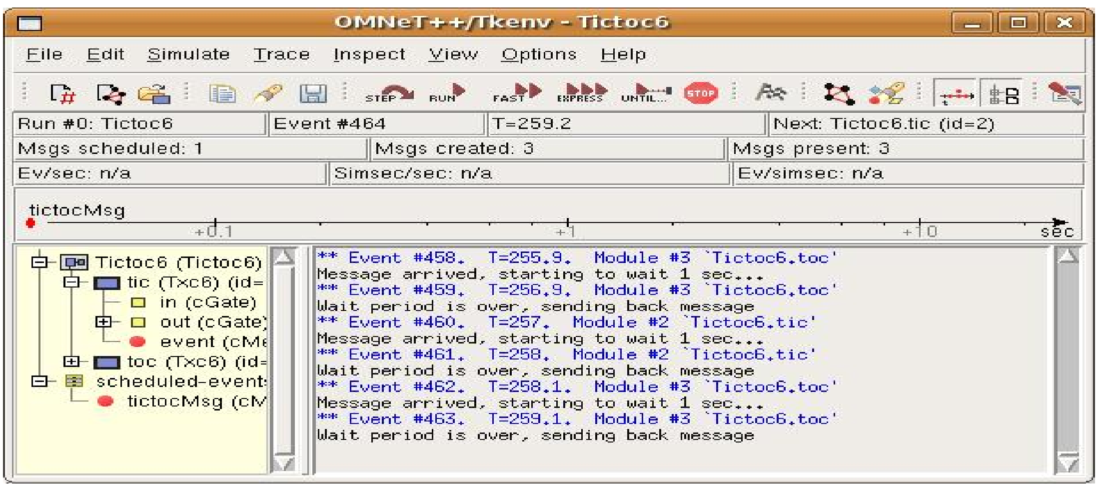
Operation of event

```
void Txc6::handleMessage (cMessage * msg)
{
if (msg==event)
   {EV << "Wait period is over, sending back message\n";
send(tictocMsg, "out");
tictocMsg = NULL;
}
Else
{
EV << "Message arrived, starting to wait 1 sec...\n";
tictocMsg = msg;
scheduleAt(simTime()+1.0, event);
}
}
```

Self message

External message
(from the other side)

Output:



Random numbers and parameters

2. Refine graphics & add debugging output	7. Random numbers & parameters	12. Using two-way conanections
3. Add state variables	8. Timeout, Cancelling timers	13. Defining our message class
4. Adding parameters	9. Retransmitting same message	14. Displaying number of packets sent/received
5. Using inheritance	10. More than two nodes	15. Visualizing output scalars and vectors
6. Modeling processing delay	11. Channels & inner type definitions	16. Sequence charts and event logs

Random numbers and parameters

- Introduce random numbers in simulation
  - Randomly lose packet
  - Change delay from 1s to a random value
- txc7.cc
- tictoc7.ned Or omnetpp.ini

txc7.cc

```
void Txc7::handleMessage(cMessage *msg)
{
    if (msg==event)
    {
        EV << "Wait period is over, sending back
message\n";
        send(tictocMsg, "out");
        tictocMsg = NULL;
    }
    else
    {
        if (uniform(0,1) < 0.1)
        {
            EV << "\"Losing\" message\n";

         delete msg;
        }
```

Lose the message with 0.1 probability

```
else
{
// The "delayTime" module parameter set
// to "exponential(5)" in tictoc7.ned so
// we'll get a different delay every time.

simtime_t delay = par("delayTime");

EV << "Message arrived, starting to wait "
<< delay << " secs...\n";
tictocMsg = msg;

scheduleAt(simTime()+delay, event);
}
```

delay parameter coming from .ned

```
network = Tictoc7
# argument to exponential() is the mean

Tictoc7.tic.delayTime = exponential(3s)
```

mean

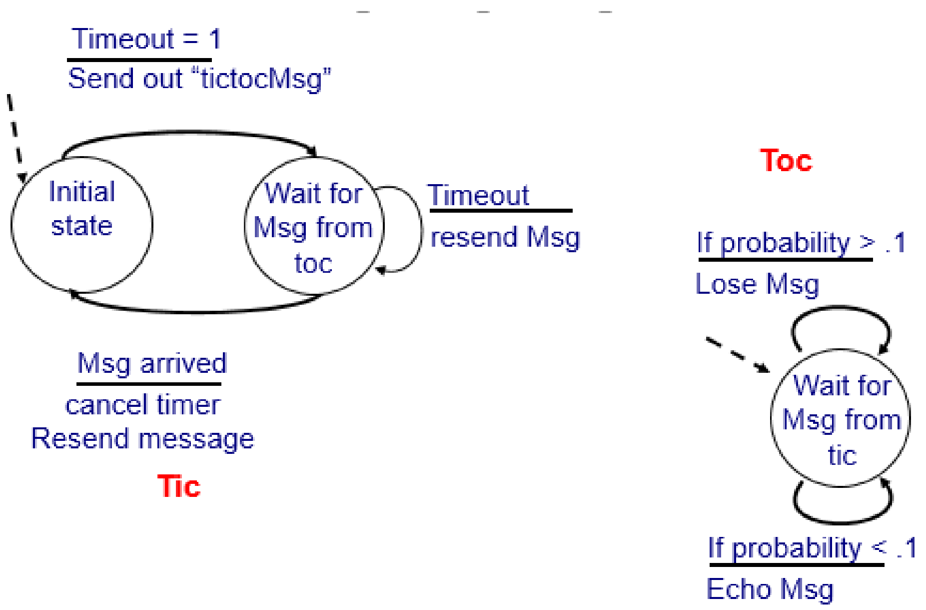
Timeout, cancelling timers

2. Refine graphics & add debugging output	7. Random numbers & parameters	12. Using two-way connections
3. Add state variables	8. Timeout, Cancelling timers	13. Defining our message class
4. Adding parameters	9. Retransmitting same message	14. Displaying number of packets sent/received
5. Using inheritance	10. More than two nodes	15. Visualizing output scalars and vectors
6. Modeling processing delay	11. Channels & inner type definitions	16. Sequence charts and event logs

Timeout, cancelling timers

- Getting closer to real world working protocols
- Stop-and-wait protocol
- txc8.cc
- tictoc8.ned
- omnetpp.ini

Strategy



txc8.cc

```
void Tic8::initialize()
{
// Initialize variables.

timeout = 1.0;
timeoutEvent = new cMessage("timeoutEvent");

// Generate and send initial message.

EV << "Sending initial message\n";
cMessage *msg = new cMessage("tictocMsg");
send(msg, "out");
scheduleAt(simTime()+timeout, timeoutEvent);
}
```

Initialize with timeout = 1 to start operation

```
void Tic8::handleMessage(cMessage *msg)
{
    if (msg==timeoutEvent)
```

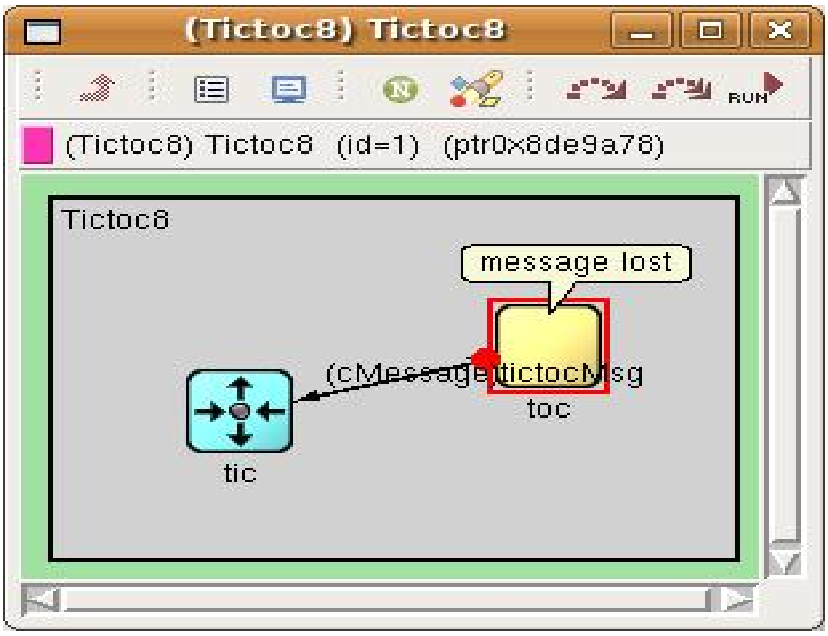
timeout means we have to re-send it

```
 {
  EV << "Timeout expired, resending and restarting
    timer\n";
  cMessage *newMsg = new cMessage("tictocMsg");
  send(newMsg, "out");
  scheduleAt(simTime()+timeout, timeoutEvent);
 }

else
{
 // delete received message & cancel timeout event.

 EV << "Timer cancelled.\n";
 cancelEvent(timeoutEvent);
 delete msg;
```

message arrived
Ack received

```
 //Ready to send another one.

 cMessage *newMsg = new cMessage("tictocMsg");
 send(newMsg, "out");
 scheduleAt(simTime()+timeout, timeoutEvent);

    }
}
```

Output



Retransmitting same message

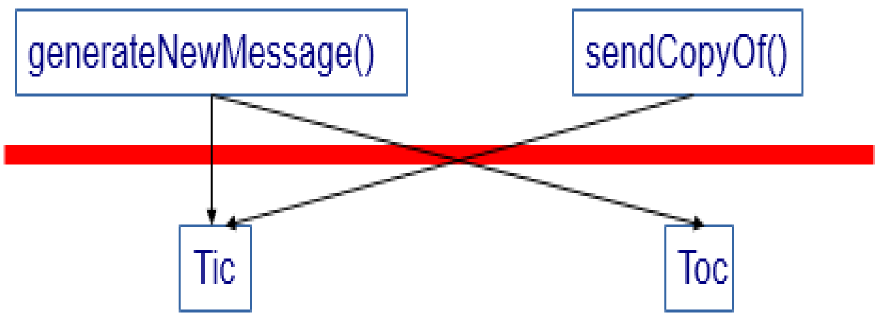
2. Refine graphics & add debugging output	7. Random numbers & parameters	12. Using two-way conanections
3. Add state variables	8. Timeout, Cancelling timers	13. Defining our message class
4. Adding parameters	9. Retransmitting same message	14. Displaying number of packets sent/received
5. Using inheritance	10. More than two nodes	15. Visualizing output scalars and vectors
6. Modeling processing delay	11. Channels & inner type definitions	16. Sequence charts and event logs

Retransmitting same message)

- So far we used “tictocMsg”
- It was created afresh everytime
  - At tic
  - At toc
- In reality, original packet needs to be retransmitted
- Solution: Keep a copy with tic
- txc9.cc
- tictoc9.ned
- omnetpp.ini

Strategy

- Create two new functions
- Conditionally call them in tic and toc



Txc9.cc

```
void Tic9::handleMessage(cMessage *msg)
{
    if (msg==timeoutEvent)
    {
     EV << "Timeout expired, resending message and
     restarting timer\n";

  sendCopyOf(message);

  scheduleAt(simTime()+timeout, timeoutEvent);
    }
```

Retransmit the same packet

```
  else // message arrived
     {
        // Acknowledgement received!
     // Ready to send another one.

     message = generateNewMessage();

     sendCopyOf(message);

     scheduleAt(simTime()+timeout, timeoutEvent);
     }
```

Transmit a new packet

generateNewMessage()

```
{
// Generate a message with a different name every time.
    char msgname[20];
    sprintf(msgname, "tic-%d", ++seq);
    cMessage *msg = new cMessage(msgname);
    return msg;
}
```

Increments **seq no** and is value of **%d**

Prints the string on Location of length 20 pointed by **msgname**

Displays string which is **seq no** as decimal

sendCopyOf(cMessage *msg)

```
{
 // Duplicate message and send the copy.

   cMessage *copy = (cMessage *) msg->dup();

    send(copy, "out");
}
```

Creates & returns an exact copy of **msg**

Value of **copy** taken from **msg**

Casts return value of **dup()** to a pointer of cMessage type

More than 2 nodes

2. Refine graphics & add debugging output	7. Random numbers & parameters	12. Using two-way connections
3. Add state variables	8. Timeout, Cancelling timers	13. Defining our message class
4. Adding parameters	9. Retransmitting same message	14. Displaying number of packets sent/received
5. Using inheritance	10. More than two nodes	15. Visualizing output scalars and vectors
6. Modeling processing delay	11. Channels & inner type definitions	16. Sequence charts and event logs

More than 2 nodes

- Create several tic modules
- Connect them into a network
- One of the nodes generates a message
- Others toss it around in random directions
- Until it arrives at a predetermined destination
- tictoc10.ned
- omnetpp.ini
- txc10.cc

Tictoc10.ned

```
simple Txc10
{
    parameters:
        @display("i=block/routing");
    gates:
        input in[];  // declare in[] and out[] to be
vector gates
        output out[];
}
```

[ ] turns the gates into gate vectors


```
simple Txc10
{
    parameters:
        @display("i=block/routing");
    gates:
        input in[];  // declare in[] and out[] to be
vector gates
        output out[];
}
```

[ ] turns the gates into gate vectors


```
network Tictoc10
{
submodules:
tic[6]: Txc10;
connections:
tic[0].out++ --> {  delay = 100ms; } --> tic[1].in++;
tic[0].in++ <-- {  delay = 100ms; } <-- tic[1].out++;
tic[1].out++ --> {  delay = 100ms; } --> tic[2].in++;
tic[1].in++ <-- {  delay = 100ms; } <-- tic[2].out++;
tic[1].out++ --> {  delay = 100ms; } --> tic[4].in++;
tic[1].in++ <-- {  delay = 100ms; } <-- tic[4].out++;
tic[3].out++ --> {  delay = 100ms; } --> tic[4].in++;
tic[3].in++ <-- {  delay = 100ms; } <-- tic[4].out++;
tic[4].out++ --> {  delay = 100ms; } --> tic[5].in++;
tic[4].in++ <-- {  delay = 100ms; } <-- tic[5].out++;
}
```

size of the vector (no. of gates) determined here

```
void Txc10::initialize()
{
    if (getIndex()==0)
    {
        // Boot the process scheduling the initial
message as a self-message.
        char msgname[20];
        sprintf(msgname, "tic-%d", getIndex());
        cMessage *msg = new cMessage(msgname);
        scheduleAt(0.0, msg);
    }
}
```

tic[0] generates the message to be sent around

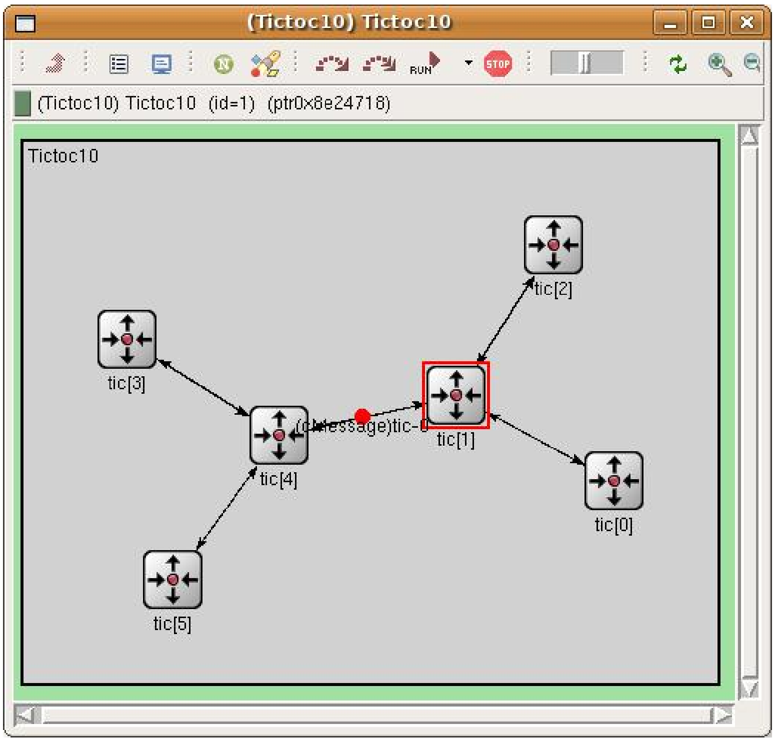
```
void Txc10::handleMessage(cMessage *msg)
{
    if (getIndex()==3)
    {
        // Message arrived.
        EV << "Message " << msg << " arrived.\n";
        delete msg;
    }
    else
    {
        // We need to forward the message.
        forwardMessage(msg);
    }
}
```

message arrives at tic[3] (final destination!)

```
void Txc10::forwardMessage(cMessage *msg)
{
    // In this example, we just pick a random gate to
send it on.
    // We draw a random number between 0 and the size of
gate `out[]'.
    int n = gateSize("out");
    int k = intuniform(0,n-1);

    EV << "Forwarding message " << msg << " on port
out[" << k << "]\n";
    send(msg, "out", k);
}
```

Uniform distribution with Probability = 1/6

Output



Channels & inner type definitions

2. Refine graphics & add debugging output	7. Random numbers & parameters	12. Using two-way conanections
3. Add state variables	8. Timeout, Cancelling timers	13. Defining our message class
4. Adding parameters	9. Retransmitting same message	14. Displaying number of packets sent/received
5. Using inheritance	10. More than two nodes	15. Visualizing output scalars and vectors
6. Modeling processing delay	11. Channels & inner type definitions	16. Sequence charts and event logs

Channels & inner type definitions

- With growing topology
  - We can improve connection section
- tictoc11.ned
- omnetpp.ini
- txc11.cc
- Connections with same delay parameter can be *typified* as channel
- Such channel can then be replicated between gates

Tictoc11.ned

```
network Tictoc11
{
types:
    channel Channel extends ned.DelayChannel {
              delay = 100ms;}
```

types section defines new channel type

types definition only visible inside network (local or inner type)

```
connections:
          tic[0].out++ --> Channel --> tic[1].in++;
          tic[0].in++ <-- Channel <-- tic[1].out++;

          tic[1].out++ --> Channel --> tic[2].in++;
          tic[1].in++ <-- Channel <-- tic[2].out++;

          tic[1].out++ --> Channel --> tic[4].in++;
          tic[1].in++ <-- Channel <-- tic[4].out++;

          tic[3].out++ --> Channel --> tic[4].in++;
          tic[3].in++ <-- Channel <-- tic[4].out++;

          tic[4].out++ --> Channel --> tic[5].in++;
          tic[4].in++ <-- Channel <-- tic[5].out++;
}
```

Delay parameter for whole network easily changed

Using two-way connections

2. Refine graphics & add debugging output	7. Random numbers & parameters	12. Using two-way conanections
3. Add state variables	8. Timeout, Cancelling timers	13. Defining our message class
4. Adding parameters	9. Retransmitting same message	14. Displaying number of packets sent/received
5. Using inheritance	10. More than two nodes	15. Visualizing output scalars and vectors
6. Modeling processing delay	11. Channels & inner type definitions	16. Sequence charts and event logs

Using two-way connections

- So far, each node pair is connected with two connections
- Two-way connection can reduce coding size
- tictoc12.ned
- txc12.cc
- omnetpp.ini
- We define two-way (inout) gates Instead of in and out gates

Tictoc12.ned

```
connections:
        tic[0].gate++ <--> Channel <--> tic[1].gate++;
        tic[1].gate++ <--> Channel <--> tic[2].gate++;
        tic[1].gate++ <--> Channel <--> tic[4].gate++;
        tic[3].gate++ <--> Channel <--> tic[4].gate++;
        tic[4].gate++ <--> Channel <--> tic[5].gate++;
```

```
simple Txc12
{
    parameters:
        @display("i=block/routing");
    gates:
        inout gate[];  // declare two way connections
}
```

inout gate defined for both incoming and outgoing messages

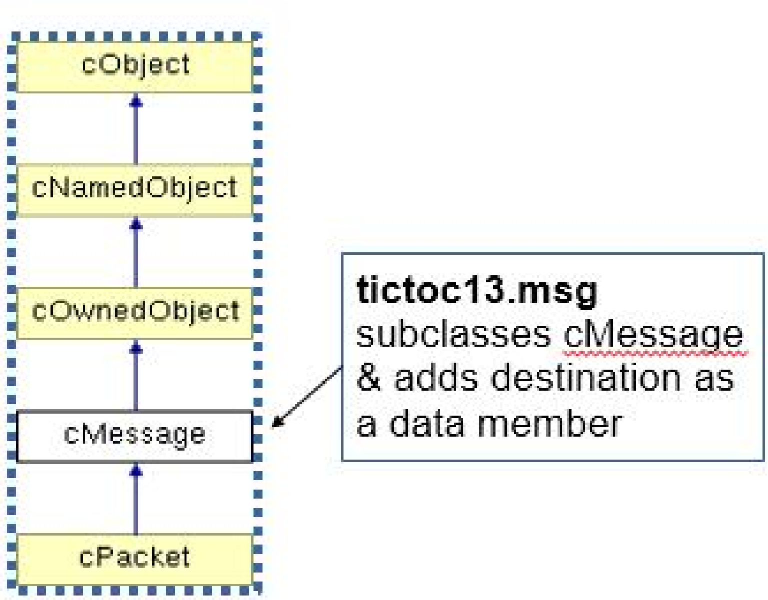
Defining our message class

2. Refine graphics & add debugging output	7. Random numbers & parameters	12. Using two-way conanections
3. Add state variables	8. Timeout, Cancelling timers	13. Defining our message class
4. Adding parameters	9. Retransmitting same message	14. Displaying number of packets sent/received
5. Using inheritance	10. More than two nodes	15. Visualizing output scalars and vectors
6. Modeling processing delay	11. Channels & inner type definitions	16. Sequence charts and event logs

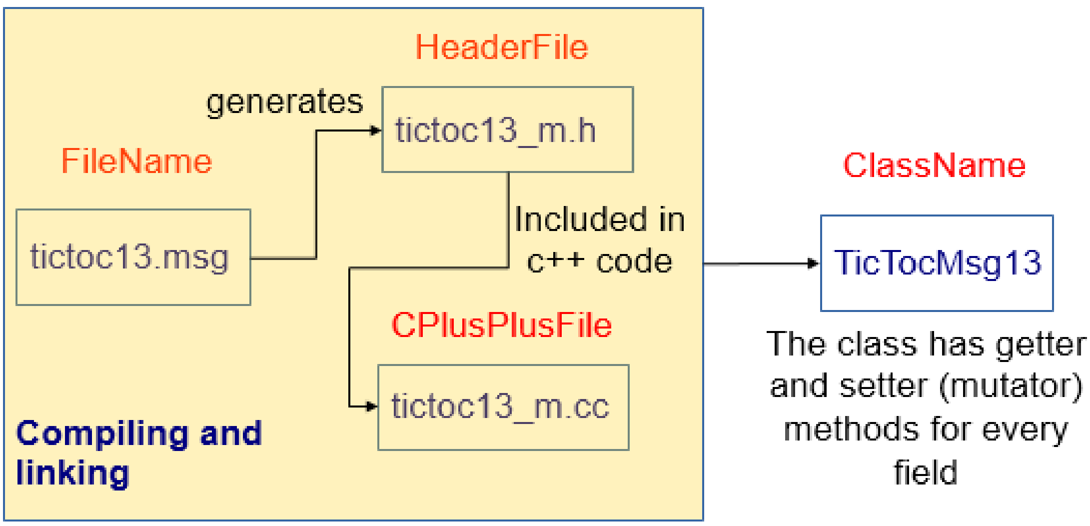
Defining our message class

- Instead of hardcoding tic[3], we need flexibility
- Draw out a random destination
- Add Destination address
- tictoc13.ned
- txc13.cc
- tictoc13.msg
- omnetpp.ini

Strategy : Avoid boilerplate code writing



Strategy :Avoid boilerplate code writing



\

Txc13.cc

```
void Txc13::handleMessage(cMessage *msg)
{
TicTocMsg13 *ttmsg = check_and_cast<TicTocMsg13 *>(msg);
if (ttmsg->getDestination()==getIndex())
 {// Message arrived.
    EV << "Message " << ttmsg << " arrived after " <<
  ttmsg->getHopCount() << " hops.\n";
  bubble("ARRIVED, starting new one!");
  delete ttmsg;
  // Generate another one.
  EV << "Generating another message: ";
  TicTocMsg13 *newmsg = generateMessage();
  EV << newmsg << endl;
  forwardMessage(newmsg);
 }
 else // We need to forward the message.
 { forwardMessage(ttmsg);
 }
}
```

Only destination address responds

Destination becomes source now

Its not the destination

\

```
TicTocMsg13 *Txc13::generateMessage()
{
// Produce source and destination addresses.

int src = getIndex();    // our module index
int n = size();        // module vector size
int dest = intuniform(0,n-2);
if (dest>=src) dest++;
char msgname[20];
   sprintf(msgname, "tic-%d-to-%d", src, dest);

// Create message object & set source and destination
field.
    TicTocMsg13 *msg = new TicTocMsg13(msgname);
    msg->setSource(src);
    msg->setDestination(dest);
    return msg;
```

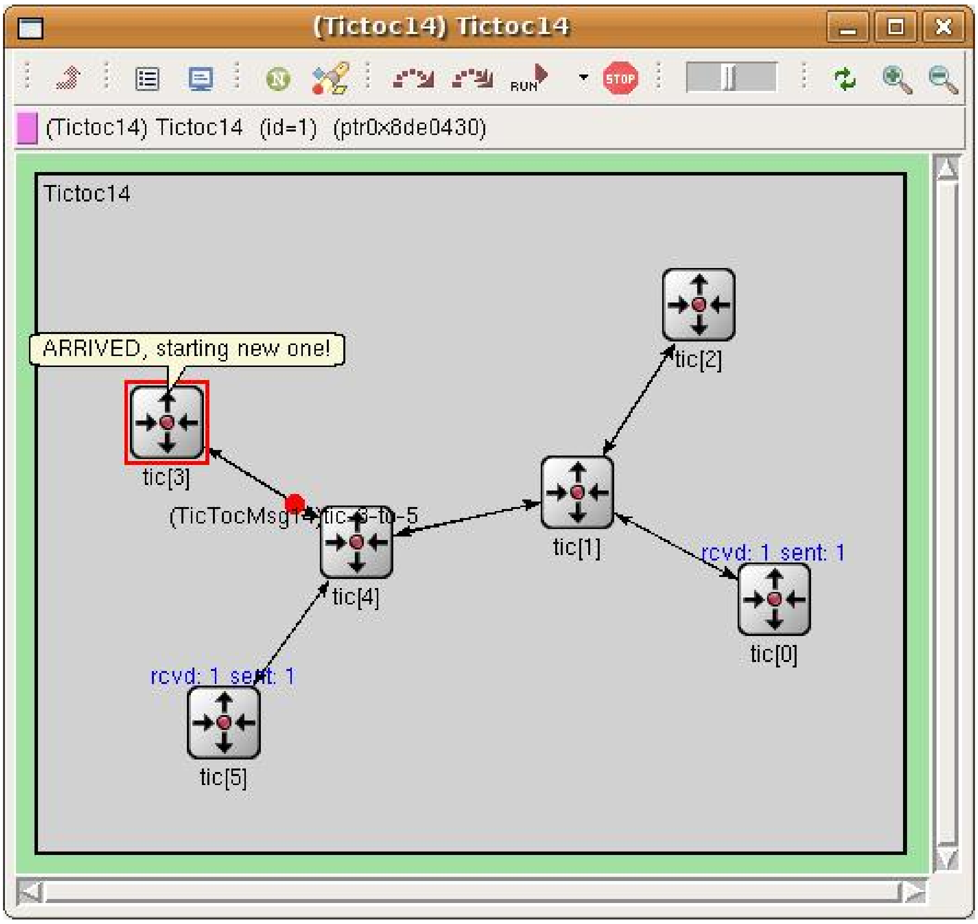
Except itself

Msg shows Addressing info

```
void Txc13::forwardMessage(TicTocMsg13 *msg)
{
// Increment hop count.

msg->setHopCount(msg->getHopCount()+1);

// Same routing as before: random gate.

int n = gateSize("gate");
int k = intuniform(0,n-1);

EV << "Forwarding message " << msg << "
on gate[" << k << "]\n";

send(msg, "gate$o", k);

}
```

Output gate of **inout** gate

Output



Displaying no. of packets sent/received

2. Refine graphics & add debugging output	7. Random numbers & parameters	12. Using two-way connections
3. Add state variables	8. Timeout, Cancelling timers	13. Defining our message class
4. Adding parameters	9. Retransmitting same message	14. Displaying number of packets sent/received
5. Using inheritance	10. More than two nodes	15. Visualizing output scalars and vectors
6. Modeling processing delay	11. Channels & inner type definitions	16. Sequence charts and event logs

Displaying no. of packets sent/received

- No. of messages at each node
- tictoc14.ned
- txc14.cc
- tictoc14.msg
- omnetpp.ini

Txc14.cc

```
class Txc14 : public cSimpleModule
{
  private:
    long numSent;
    long numReceived;
  protected:
 virtual void updateDisplay();
```

Declared

```
  void Txc14::initialize()
{
  // Initialize variables
  numSent = 0;
  numReceived = 0;
  WATCH(numSent);
  WATCH(numReceived);
```

set to zero & WATCH'ed in initialize() method

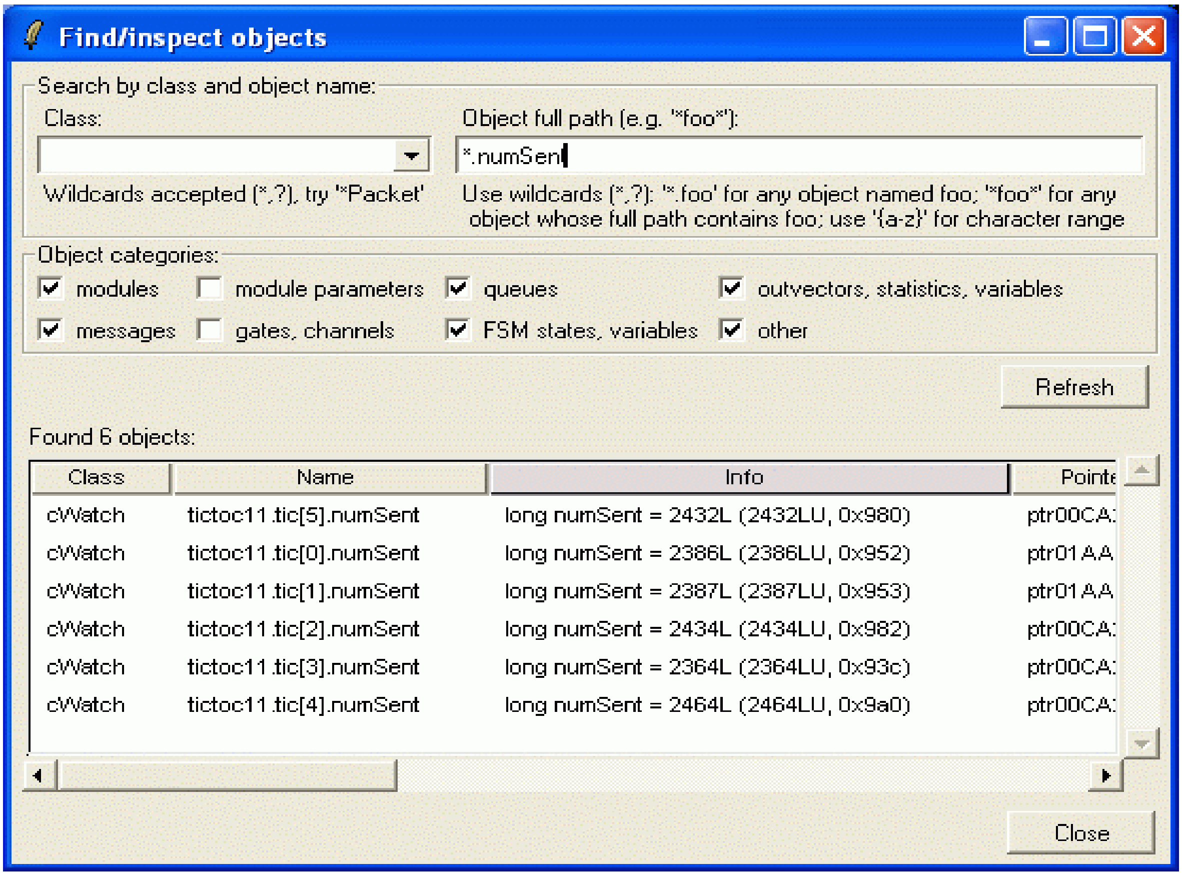
```
void Txc14::handleMessage(cMessage *msg)
{
 if (ttmsg->getDestination()==getIndex())
  {
   if (ev.isGUI())
    {
   updateDisplay();
    }
  }
}
```

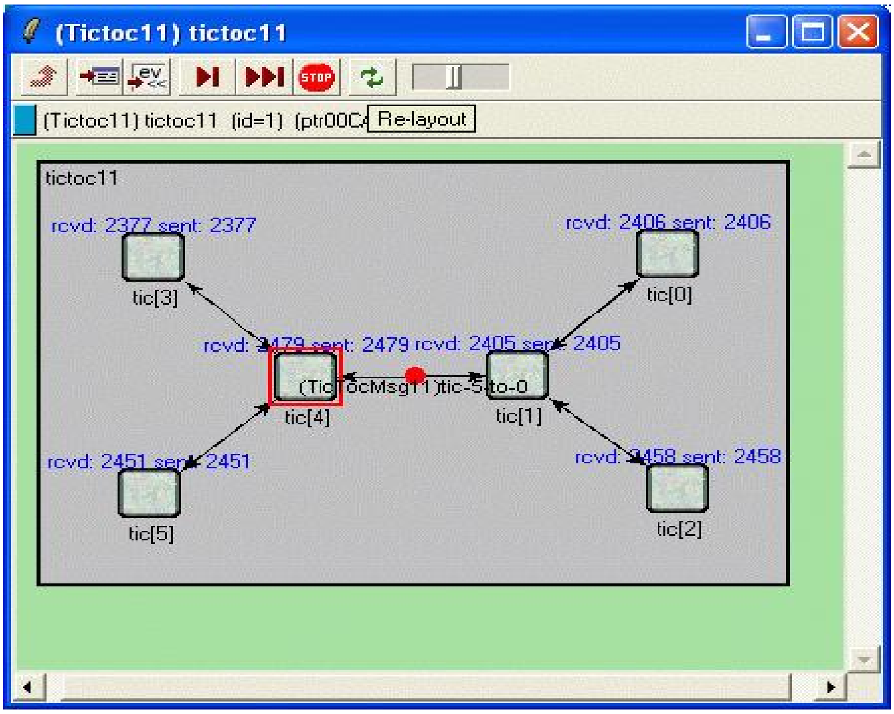
info appears above module icons

```
void Txc14::updateDisplay()
{
    char buf[40];
    sprintf(buf, "rcvd: %ld sent: %ld", numReceived,
numSent);
    getDisplayString().setTagArg("t",0,buf);
}
```

Similar to bubble text but without bubble

Object Inspector in Tkenv





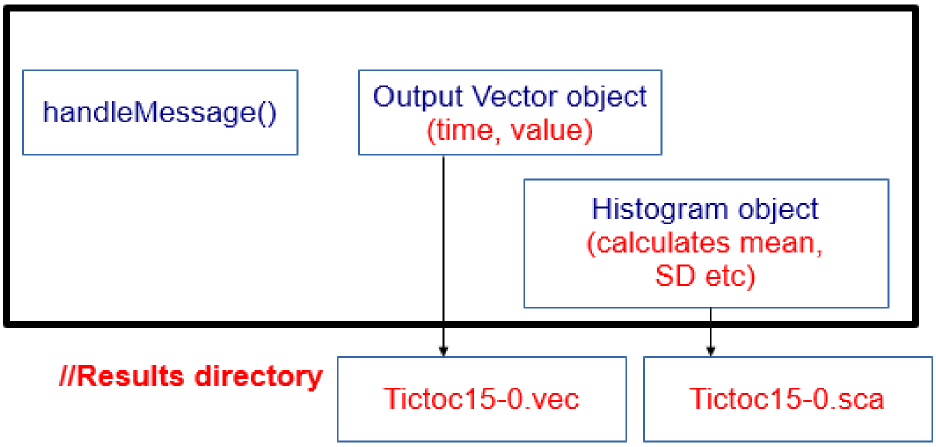
Adding statistics collection

2. Refine graphics & add debugging output	7. Random numbers & parameters	12. Using two-way connections
3. Add state variables	8. Timeout, Cancelling timers	13. Defining our message class
4. Adding parameters	9. Retransmitting same message	14. Displaying number of packets sent/received
5. Using inheritance	10. More than two nodes	15. Visualizing output scalars and vectors
6. Modeling processing delay	11. Channels & inner type definitions	16. Sequence charts and event logs

Adding statistics collection

- When packet traverses multiple hops, it becomes important to collect network statistics
  - Average hop count
  - Max, min etc
- tictoc15.ned
- txc15.cc
- tictoc15.msg
- omnetpp.ini

Strategy



Txc15.cc

```
class Txc15 : public cSimpleModule
{
  private:
    long numSent;
    long numReceived;
    cLongHistogram hopCountStats;
    cOutVector hopCountVector;
  virtual void finish();
```

The finish() function is called by OMNeT++ at the end of the simulation

HopCountVector.record outputs to vector file

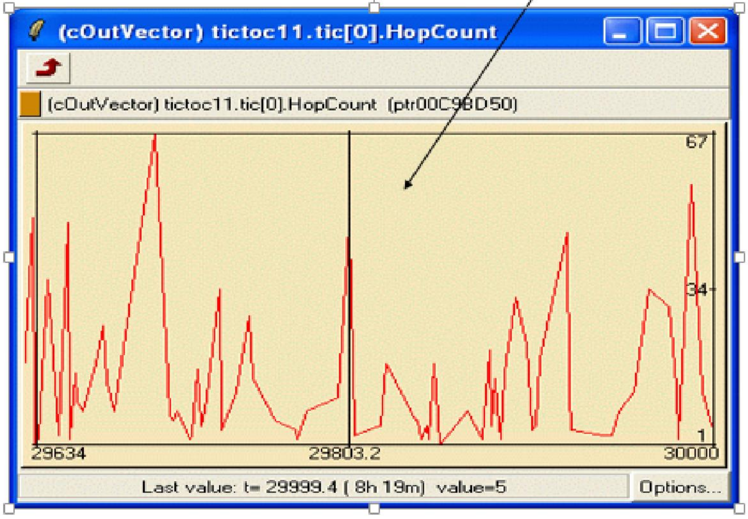
```
void Txc15::handleMessage(cMessage *msg)
{
   // Message arrived
         int hopcount = ttmsg->getHopCount();
         EV << "Message " << ttmsg << " arrived after "
<< hopcount << " hops.\n";
         bubble("ARRIVED, starting new one!");

         numReceived++;
         hopCountVector.record(hopcount);
         hopCountStats.collect(hopcount);
}
```

Update the statistics

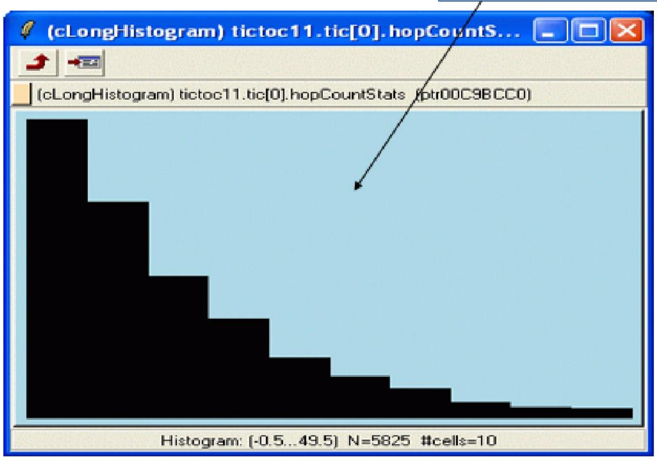
```
void Txc15::finish()
{    EV << "Sent:      " << numSent << endl;
     EV << "Received: " << numReceived << endl;
     EV << "Hop count, min:     " <<
hopCountStats.getMin() << endl;
     EV << "Hop count, max:     " <<
hopCountStats.getMax() << endl;
     EV << "Hop count, mean:    " <<
hopCountStats.getMean() << endl;
     EV << "Hop count, stddev: " <<
hopCountStats.getStddev() << endl;
     recordScalar("#sent", numSent);
     recordScalar("#received", numReceived);

     hopCountStats.recordAs("hop count");
}
```

Records the statistics into the scalar result file And displays "Hop count"

Records the vector as time series

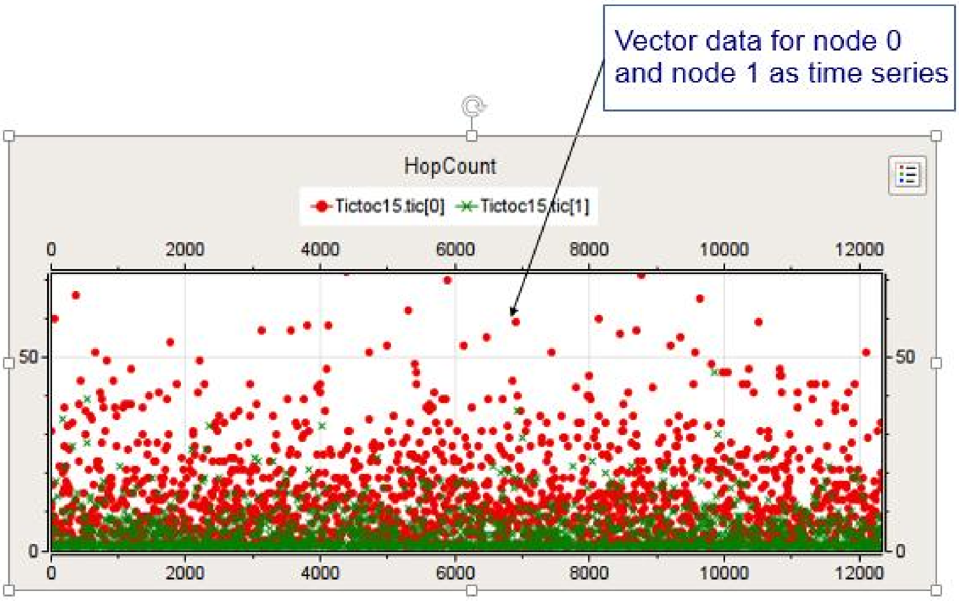


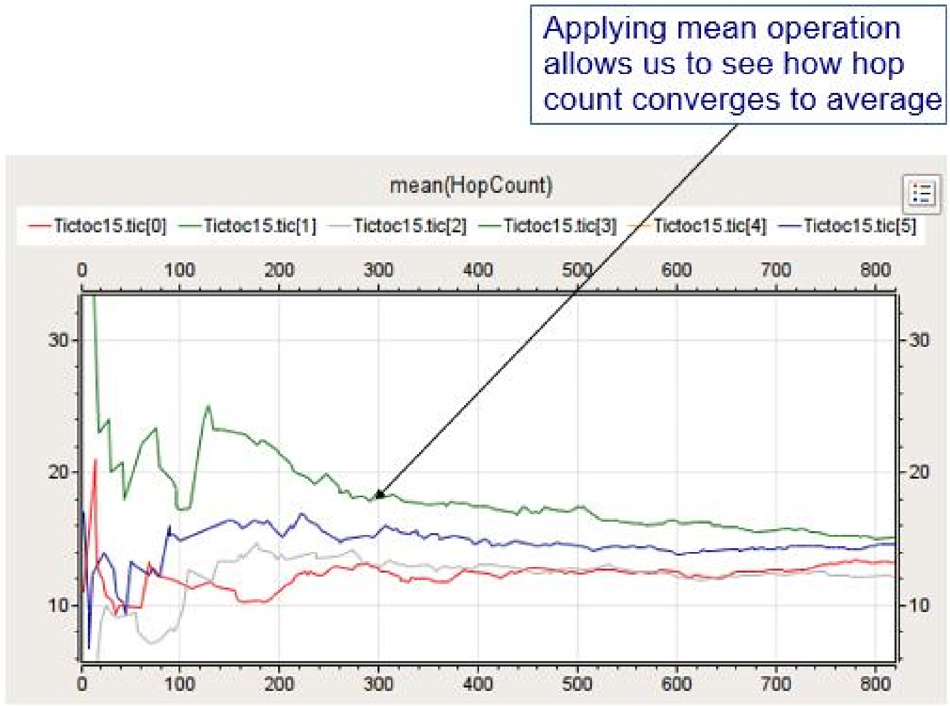
Histogram during the simulation run

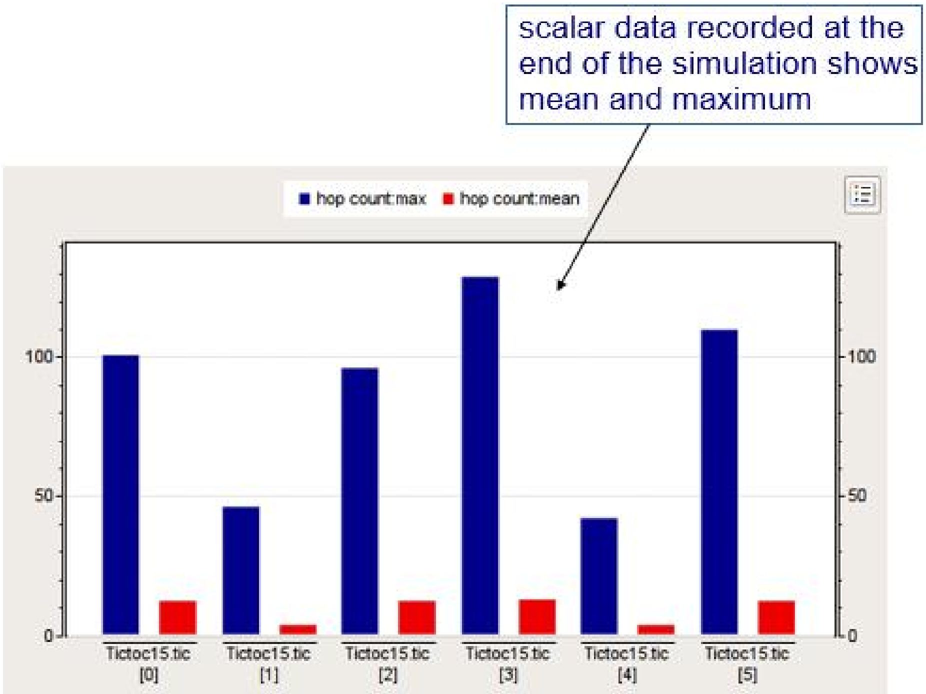


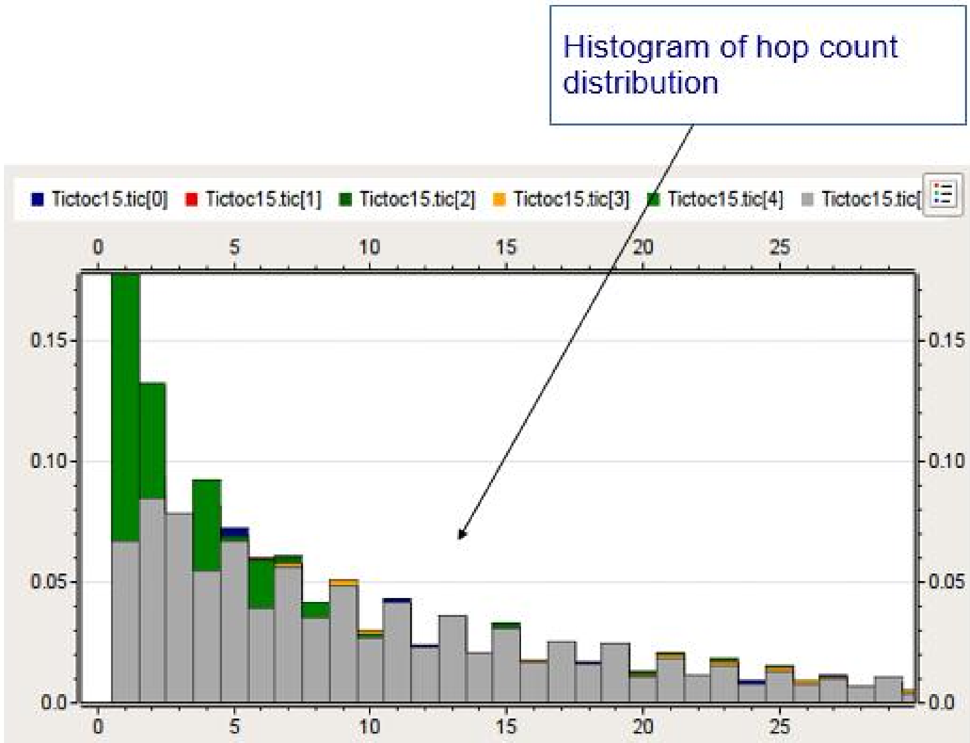
Visualizing output scalars & vectors**Visualizing output scalars & vectors**

- OMNET++ allows to visualize outputs of scalar and vector files
  - Filtering
  - Processing
  - Displaying






scalar data recorded at the end of the simulation shows mean and maximum
hop count:max
hop count:mean
0
50
100
Tictoc15.tic [0]
Tictoc15.tic [1]
Tictoc15.tic [2]
Tictoc15.tic [3]
Tictoc15.tic [4]
Tictoc15.tic [5]


Histogram of hop count distribution
Tictoc15.tic[0]
Tictoc15.tic[1]
Tictoc15.tic[2]
Tictoc15.tic[3]
Tictoc15.tic[4]
Tictoc15.tic[
0
5
10
15
20
25
0.0
0.05
0.10
0.15

Analyzing Results

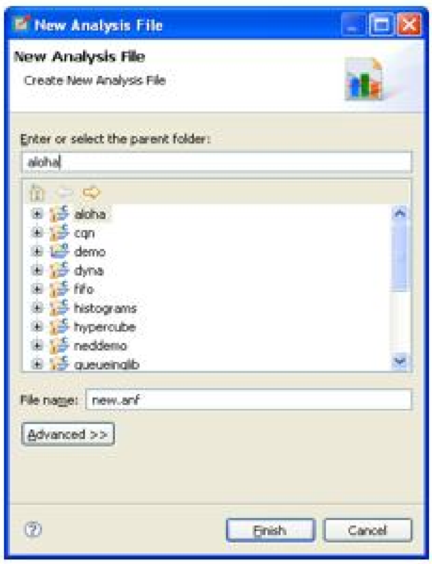
What is Simulation Analysis?

- Analyzing simulation results is lengthy and time consuming process
- Result are recorded as scalar values, vector values and histograms
- User can apply statistical methods
  - Extract the relevant information
  - Draw conclusions

Analysis File (.anf)

- A file that automates the steps to analyze the results
  - Loading result files
  - Filter them
  - Transform data

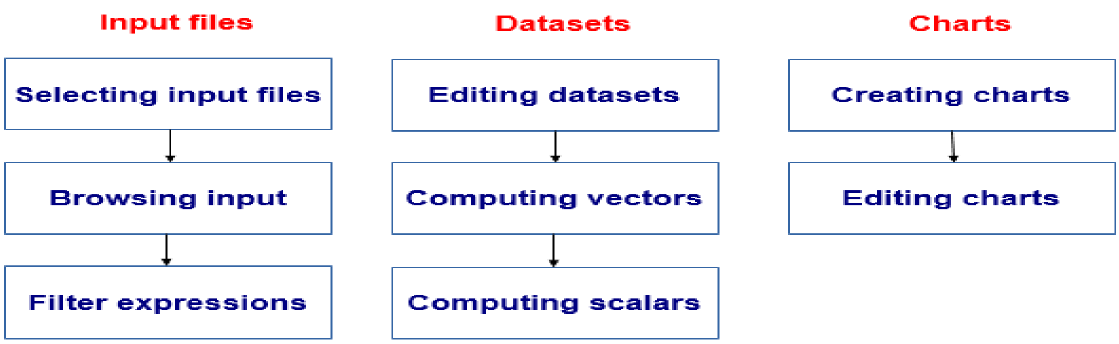
Creating Analysis File

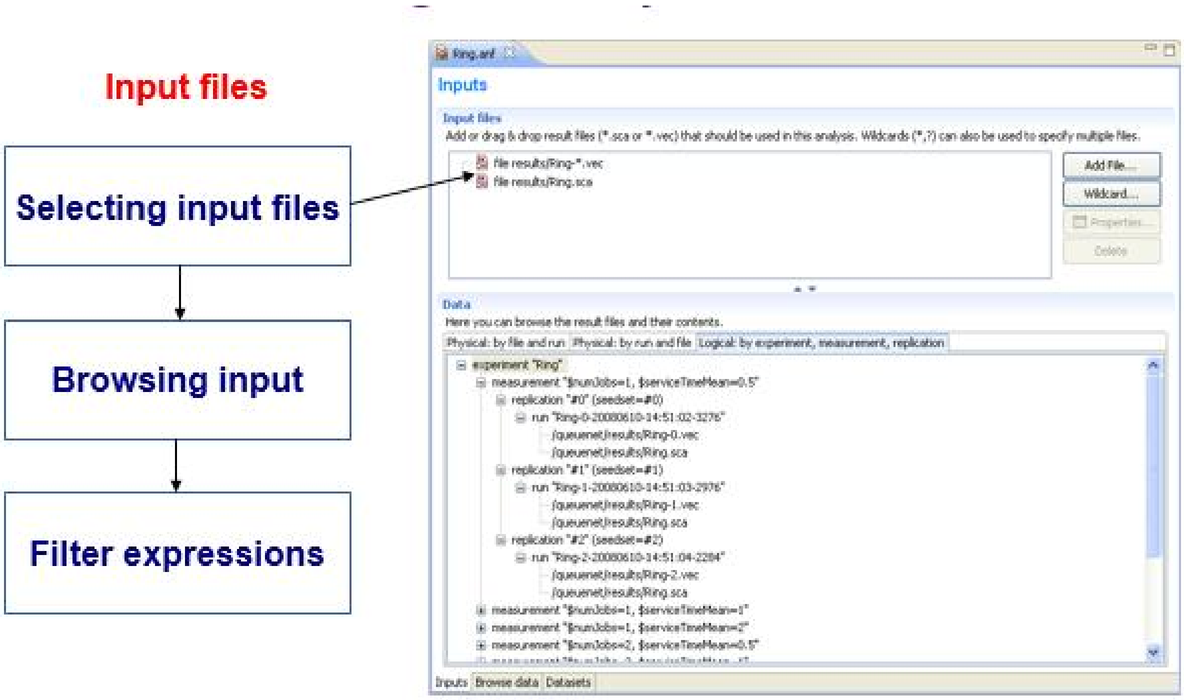


Quick way

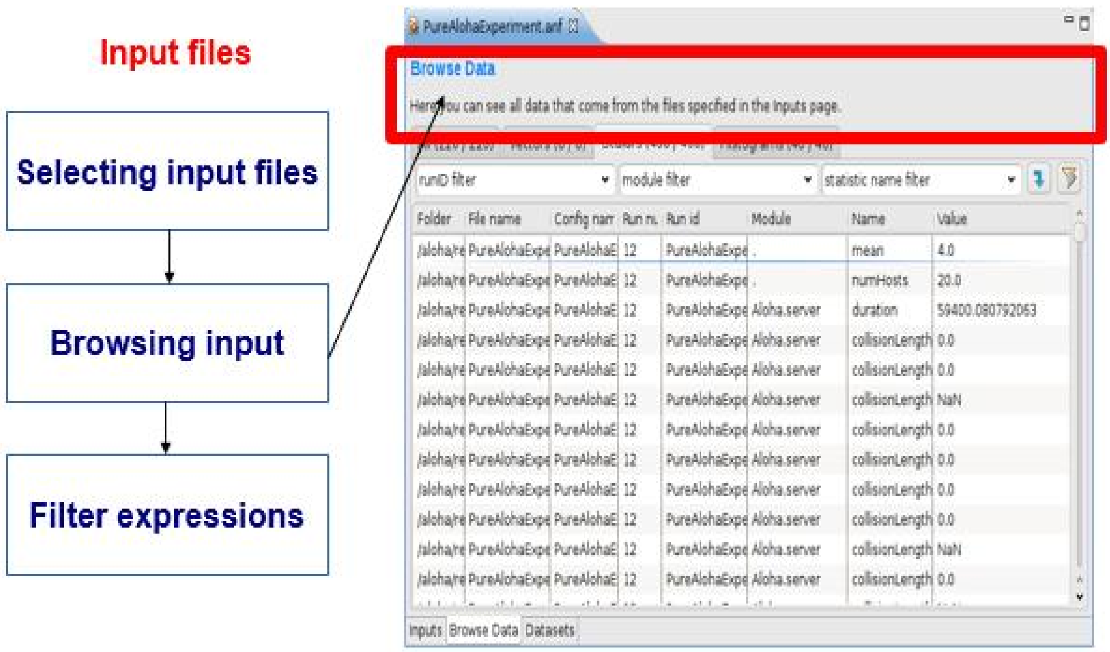
- Double-click on the **result file** in the **Project Explorer** View
- Open the **New Analysis File** dialog
  - Folder and file name get **prefilled** (according to location and name of result file)

Using the Analysis Editor



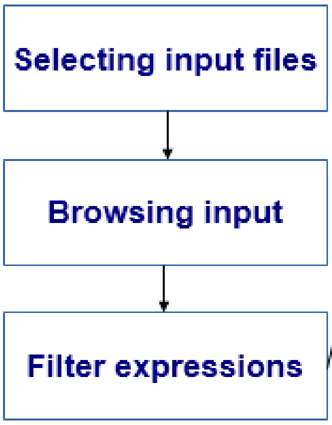


Input files
Selecting input files
Browsing input
Filter expressions
Ring.anf
Inputs
Input files
Add or drag & drop result files (*.sca or *.vec) that should be used in this analysis. Wildcards (*,?) can also be used to specify multiple files.
file results/Ring-*.vec
file results/Ring.sca
Add File...
Wildcard...
Properties...
Delete
Data
Here you can browse the result files and their contents.
Physical: by file and run
Physical: by run and file
Logical: by experiment, measurement, replication
Inputs
Browse data
Datasets



Input files
Selecting input files
Browsing input
Filter expressions
PureAlohaExperiment.anf
Browse Data
Here you can see all data that come from the files specified in the Inputs page.
runID filter
module filter
statistic name filter
Folder
File name
Config name
Run id
Module
Name
Value
Inputs
Browse Data
Datasets

Input files



A filter expression is composed of atomic patterns with the AND, OR, NOT operators

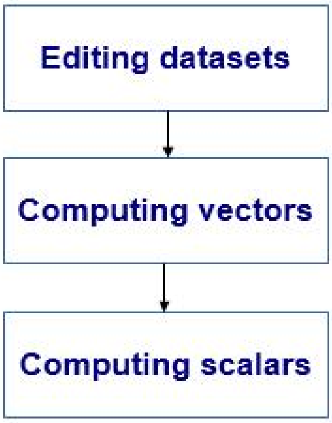
It has the form <field_name> (<pattern>)

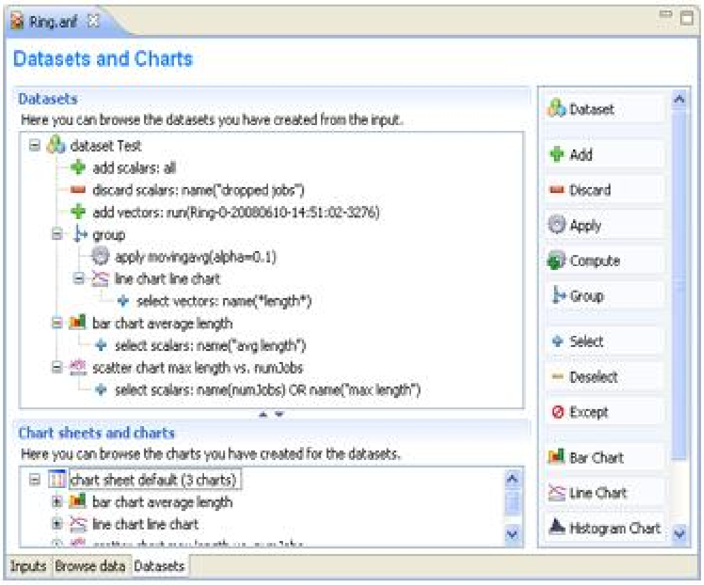
Example

module(**.sink) AND (name("queuing time") OR name("transmission time"))

Results in queuing times and transmission times that are written by modules named sink.

Datasets



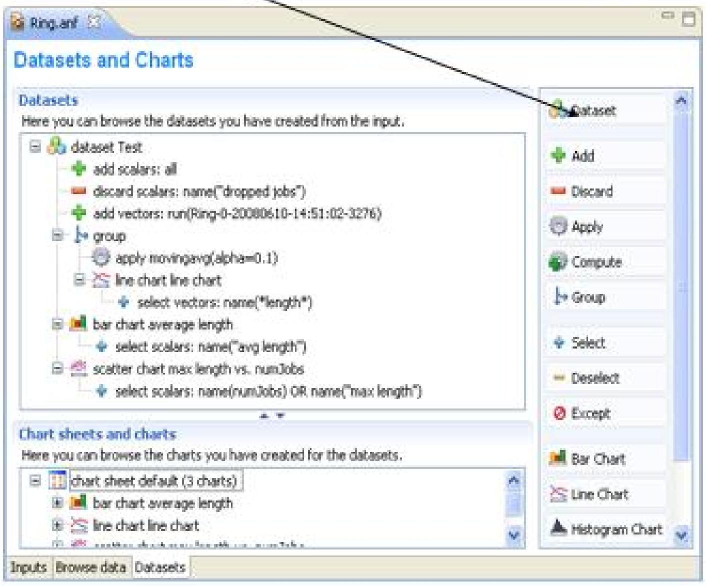


Datasets

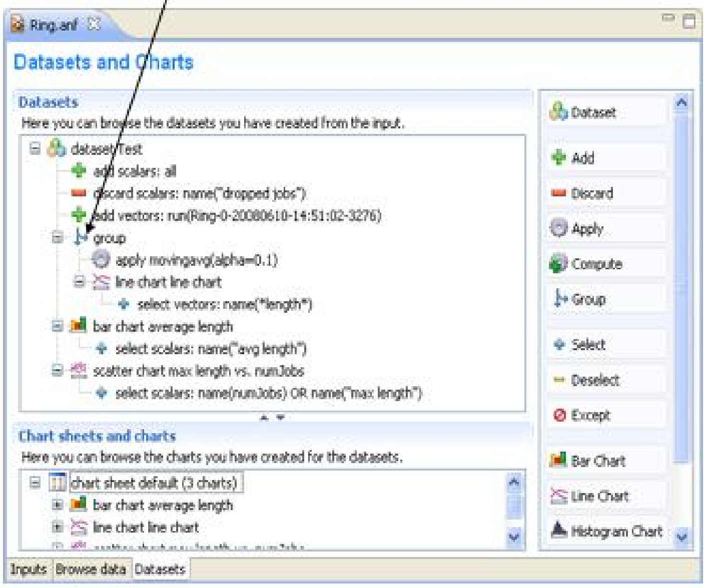
- Describe a set of input data, the processing applied to them and the charts
- Displayed as a tree of processing steps and charts
- Nodes are used for
  - Adding and discarding data
  - Applying processing to vectors and scalars
  - Selecting the operands of the operations
  - Content of charts, and for creating charts

Editing Datasets

New elements can be added by dragging elements from the palette on the right

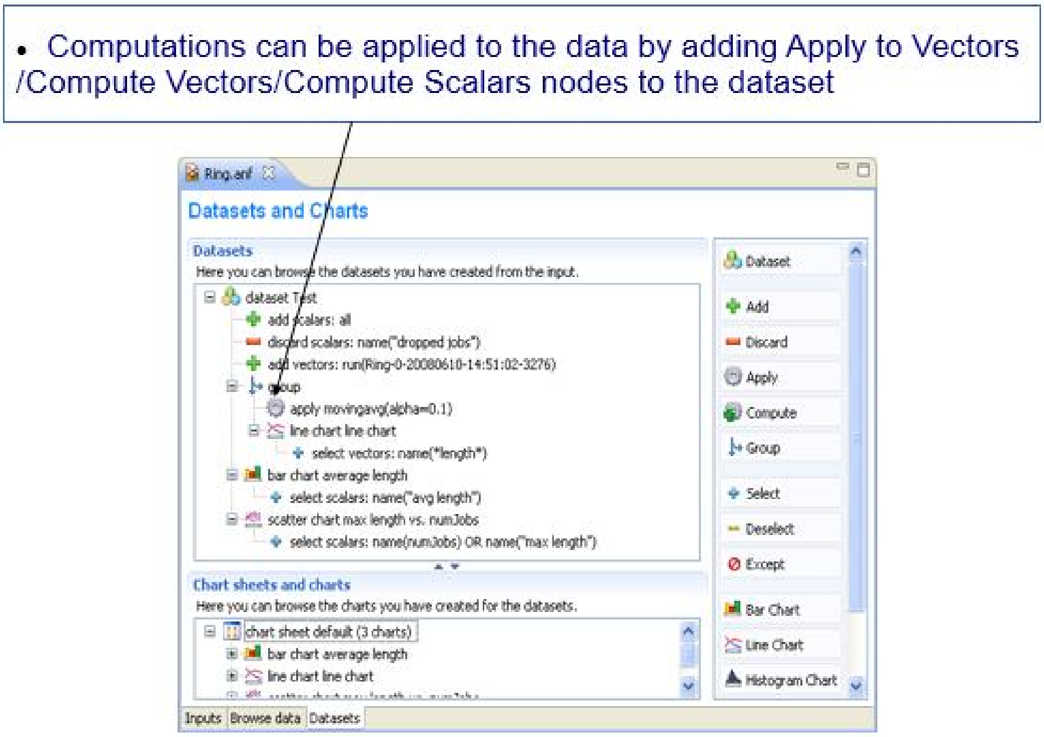


- Processing steps within a Group node only affect the group
- Allows branches to be created in the dataset
- A range of siblings can be grouped together by choosing “Group”



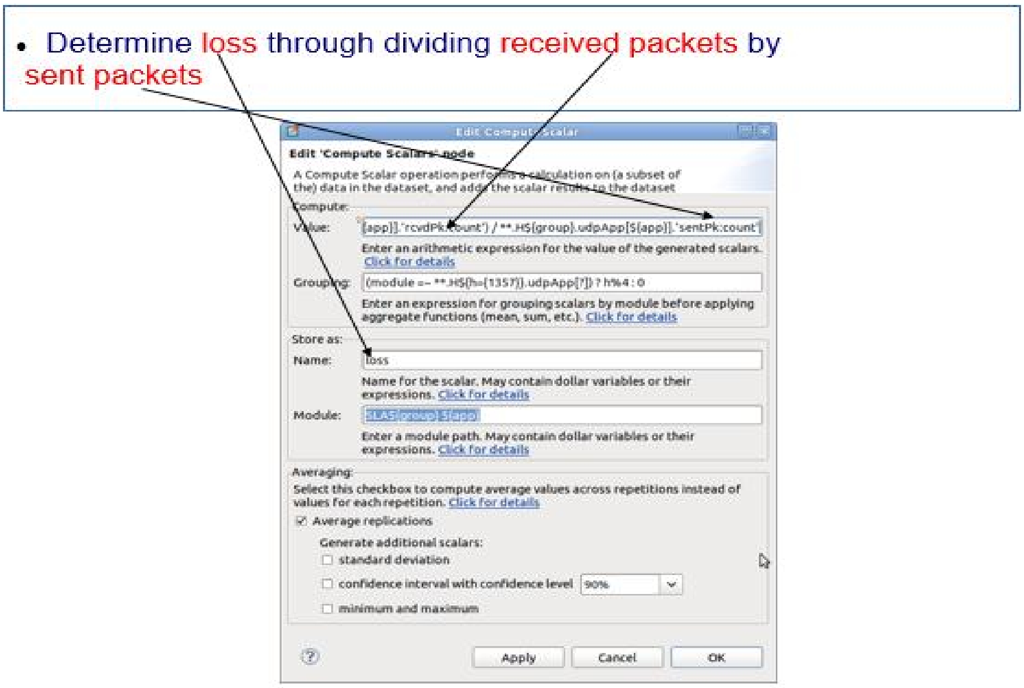
What is Compute Vectors?

- Both Compute Vectors and Apply to Vectors nodes compute new vectors from other vectors

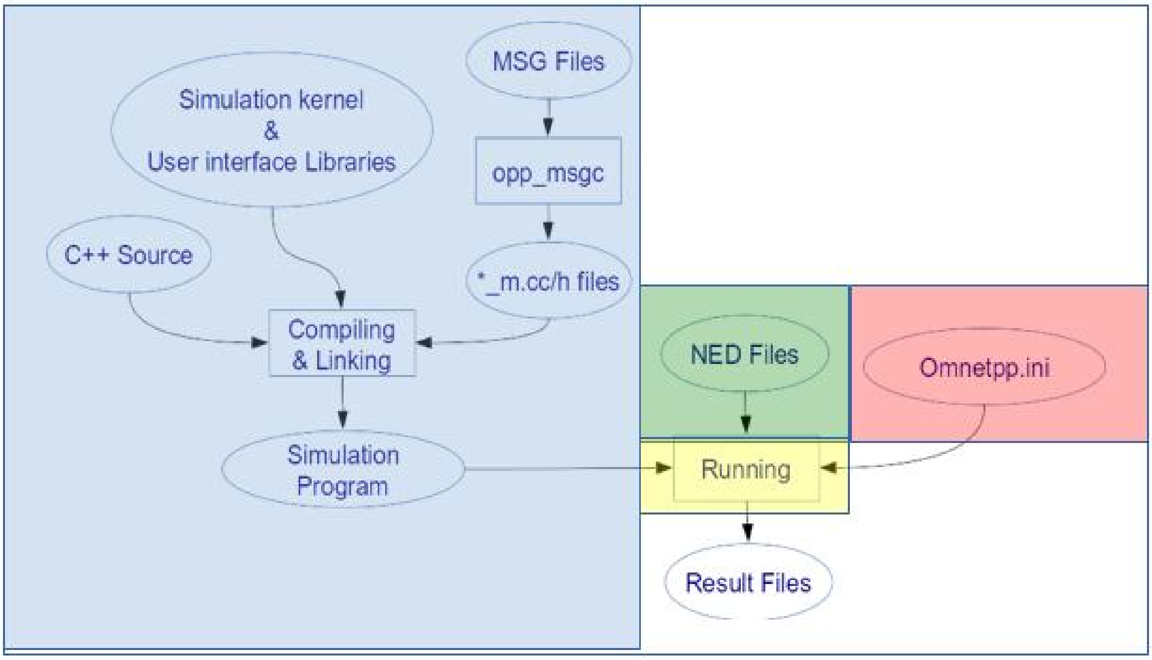


What is Compute Scalars?

- The Compute Scalars dataset node adds new scalars to the dataset whose values are computed from other statistics in the dataset



Finally we are done!



Computation Examples 1

Bit rate

- Assume several source modules in the network that generate CBR traffic
- Parameterized with **packet length (in bytes)** and **send interval (seconds)**
- Both parameters saved as scalars by each module (**pkLen, sendInterval**)
- To use the **bit rate** for further computations or charts
  - Add a **Compute Scalar node** with the following content to create an additional **bit rate** scalar for each source module

Value: pkLen*8/sendInterval
Name: bitrate

Throughput

- Assume several sink modules record **rcvdByteCount** scalars, and simulation duration is saved globally as the **duration** scalar of the top-level module.
- We are interested in the **throughput** at each sink module
- We need to refer to the **duration** scalar by its qualified name (prefix it with the full name of its module)
- **rcvdByteCount** can be left unqualified

Value:8*rcvdByteCount/Network.duration
Name: throughput

Total Received Bytes

- We are interested in the total number of **bytes received** in the network
- We can use the **sum()** function
- We store the result as a new scalar of the toplevel module, **Network**.

Value: sum(rcvdByteCount)
Name: totalRcvdBytes
Module: Network

Bytes Received by Hosts

- If several modules record scalars named **rcvdByteCount**
- We are only interested in the ones recorded from **network hosts**
- you can **qualify** the scalar name with a **pattern**

  Value: sum(**.host*.**.rcvdByteCount)
  Name: totalHostRcvdBytes
  Module: Network

Average of Peak Delay

- If several modules record vectors named **end-to-end delay**
- We are interested in **average of the peak end-to-end delays** experienced by each module
- We can use the **max()** function on the vectors to get the peak
- Then we need **mean()** to obtain their averages

  Value: mean(max('end-to-end delay'))
  Name: avgPeakDelay
  Module: Network

Computation Examples 2

Packet loss per client-server pair

- 3 clients (cli0, cli1, cli2) and 3 servers (srv0, srv1,

srv2) in the network

- Each client sends datagrams to the corresponding server
- **Packet loss per client-server pair** computed from the number of **sent** and **received** packets.
- We use the **i** variable to match the corresponding clients and servers.

  Value: Net.cli${i={0..2}}.pkSent -
  Net.srv{i}.pkRcvd
  Name: pkLoss
  Module: Net.srv${i}

Total No. of Transport Packets

- When input scalars are recorded by **different modules**
  - We need the host variable to **match TCP and UDP modules under the same host**
- Compute the **total number** of transport packets (the sum of the TCP and UDP packet counts) for each host

  Value: ${host=**}.udp.pkCount +
  ${host}.tcp.pkCount
  Name: transportPkCount
  Module: ${host}

Modules with largest RTT

- A network has various modules recording ping round-trip delays (RTT)
- We want to count the modules with large RTT values (where the average RTT is more than twice the global average in the network)
- We need to do it in steps

  Step 1:
  Value: mean('rtt:vector')
  Name: average
  Step 2:
  Value: average / mean(**.average)
  Name: relativeAverage

Step 3:
Value: count(relativeAverage)
Grouping: value > 2.0 ? "Above" : "Normal"
Name: num${group}
Module: Net

Simulation Models and INET
What is Simulation Model?
As we know that
OMNET++ is not a simulation itself

- It is a framework that allows other simulation frameworks
  - To be created
  - To be simulated
- Simulation frameworks are simulation libraries
  - Implement protocols
  -

Types of Simulation Model

- Domain-specific functionality is provided by model frameworks
  - WSNs
  - Ad-hoc networks
  - Internet protocols,
  - Performance modeling
  - Photonic networks, etc.,
- Developed as independent projects
- Reusability of models in OMNeT++ is due to its modular architecture
- Simulation models are easily integrated into OMNET++

Some Well-known Types

- INET Framework
- OverSim
- Veins
- INETMANET
- MIXIM
- Castalia

INET

- The INET Framework can be considered the standard protocol model library of OMNeT++
- Contains models for the Internet stack
  - TCP, UDP, IPv4, IPv6, OSPF, BGP, etc
- Wired and wireless link layers
- Ethernet, PPP, 802.11, etc)
- Support for mobility
- QoS support
- DiffServ, RSVP
- Several application models
- Maintained by OMNeT++ team officially

OverSim

- Overlay and peer-to-peer network simulation framework
- Contains several models for
- Structured
    - — Chord
    - — Kademlia
    - — Pastry
- Unstructured
    - — GIA

Veins

- Inter-Vehicular Communication (IVC) simulation framework
- It is a road traffic microsimulation model

INETMANET

- Fork of INET framework
- Simulation frAAamework for mobile ad-hoc networks
- Written and maintained by Alfonso Ariza.

MIXIM

- Modeling framework created for
    - — Mobile wireless
    - — Fixed wireless
    - — WSNs
    - — BANs and VANs
    - — Ad-hoc networks
- Radiowave propagation
- Interference estimation
- Power consumption
- Wireless MAC protocols

CASTALIA

- Simulation framework for networks of low-power embedded devices
- Offers models for
    - — Temporal path loss
    - — Fine-grain interference
    - — RSSI calculation
    - — Physical process model
    - — Node clock drift
    - — MAC protocols

Design Tour of INET 1

In this module
We shall take a guided
Tour of INET to

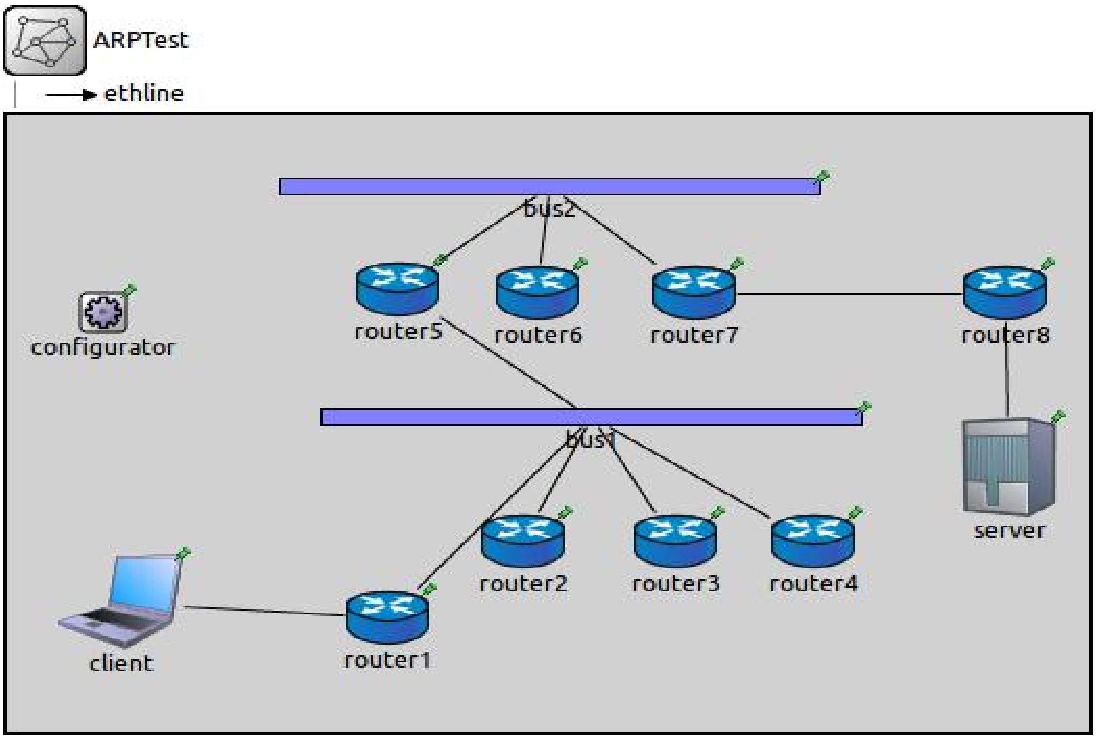
- Understand how ARP works in Ethernet environments
- Walk through features of INET
- Peek into various
  - Packets
  - Queues
  - Internal tables

Why ARP scenario?

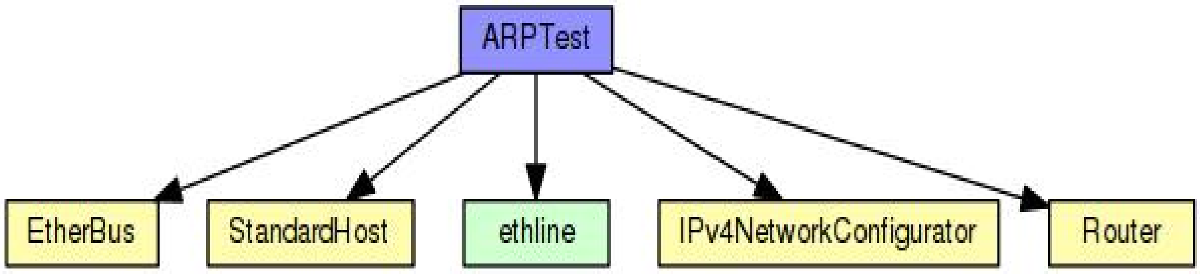
- While ARP is not the most important protocol, it is very interesting
- It relates to
  - Ethernet
  - IP
  - And other higher layer protocols

Scenario

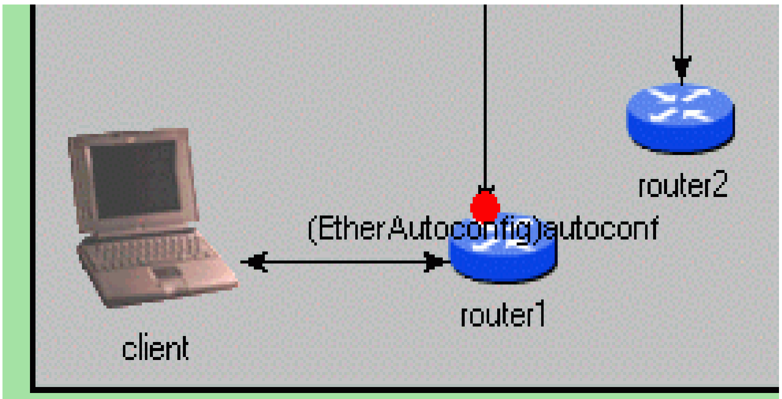
- Client computer opens TCP session with server
- Rest of operations (including ARP) follow
  - ARP has to learn the MAC address for the default router



Usage Diagram for ARP



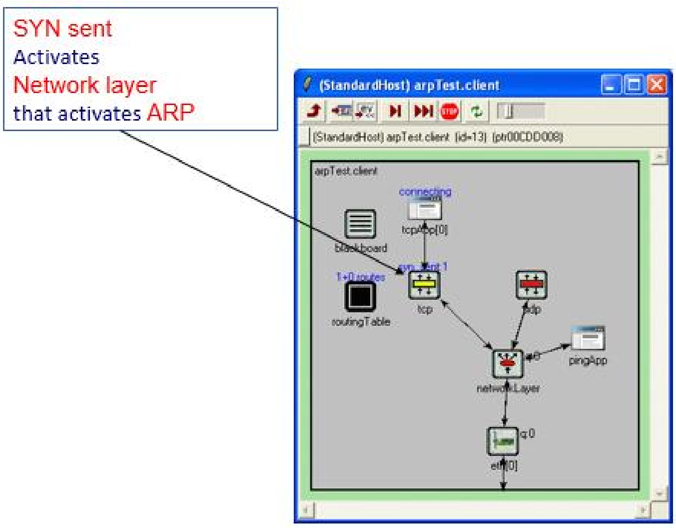
On simulation start
Ethernet autoconfiguration precedes ARP



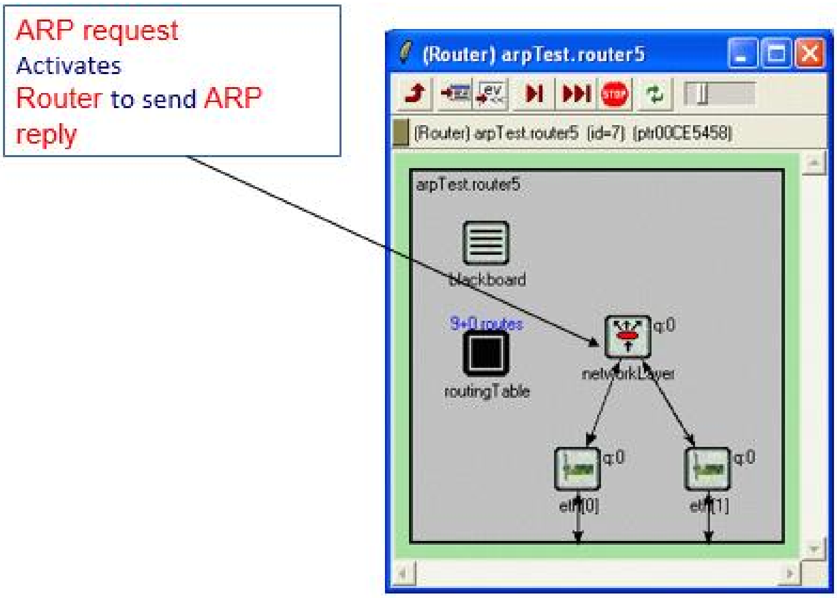
Entities at work

- Various compound modules interact with each other
- TCP host on Ethernet
- Router
- TCP server
- How end-to-end transmission takes place?

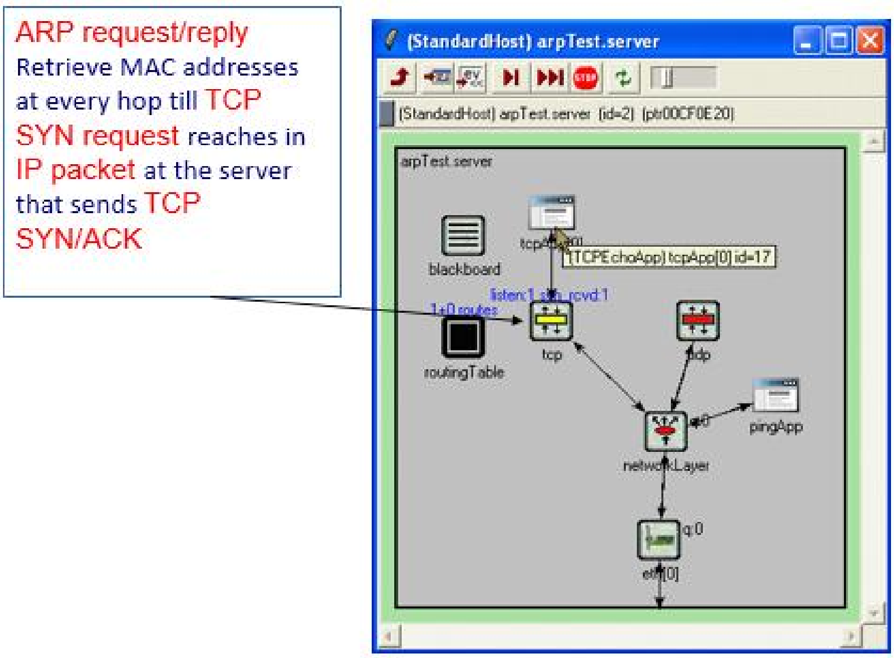
TCP Client



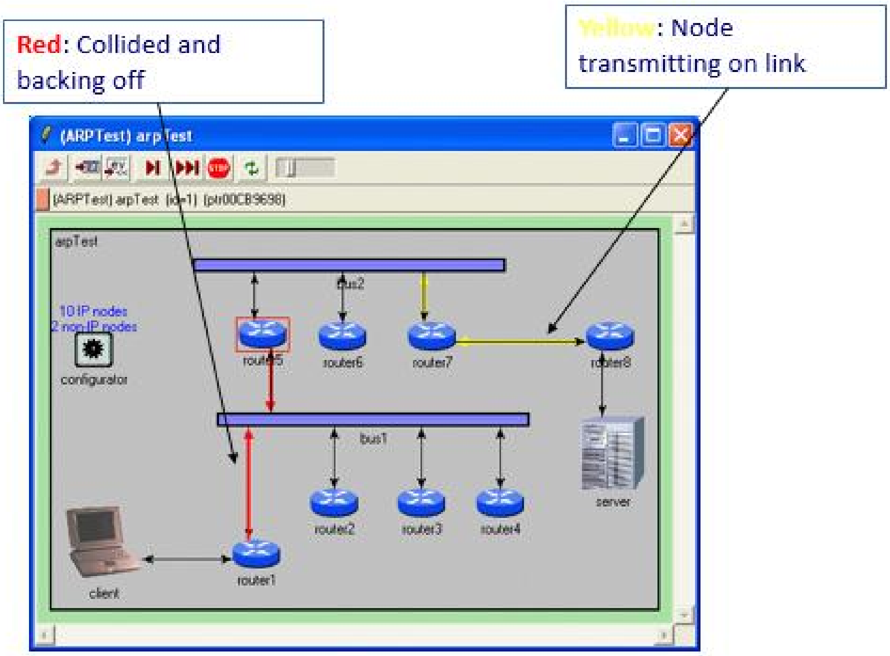
Router



TCP Server



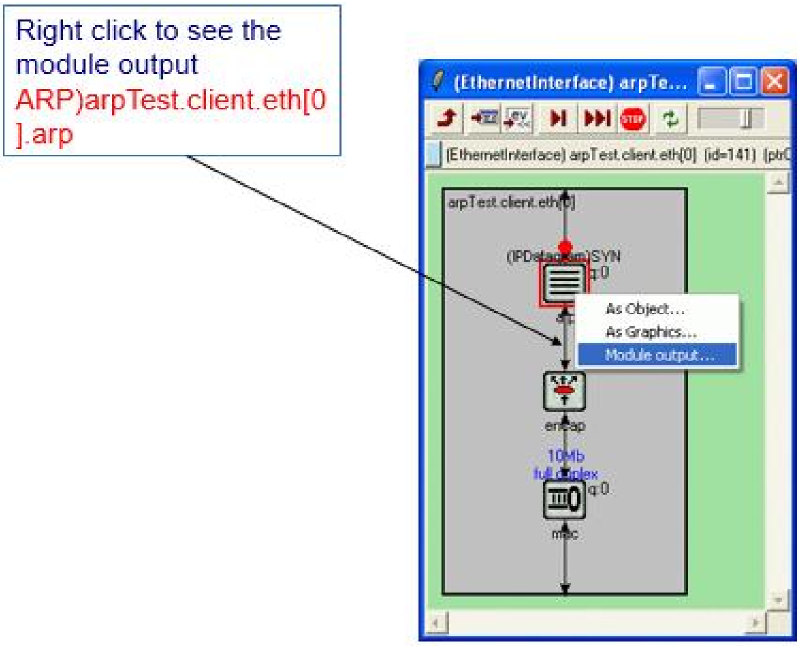
End-to-end transmission



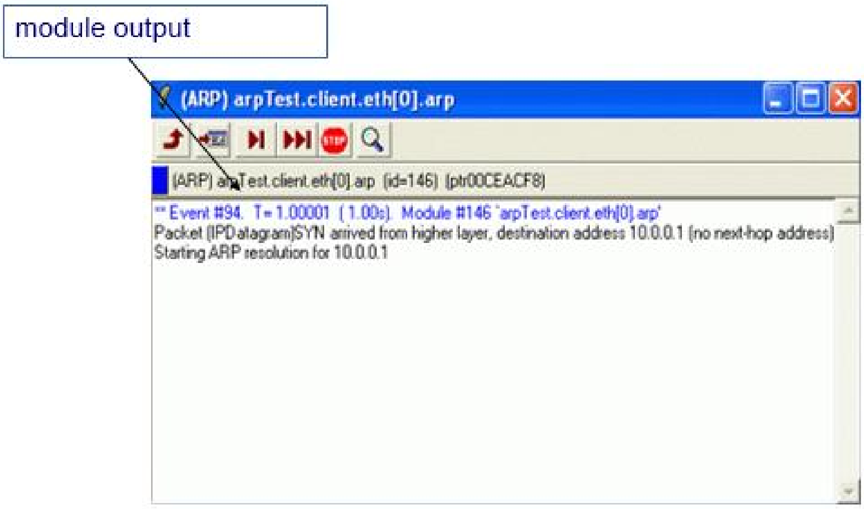
Ethernet Compound Module

- In order to further understand how INET works, let us explore Ethernet (Compound Module)
- Consists of
  - Arp
  - Encap
  - And Mac

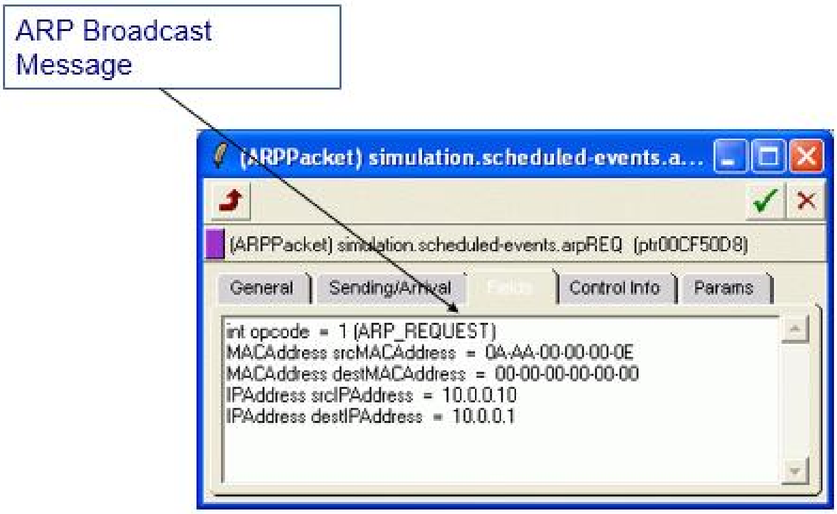
Ethernet Compound Module



arpTest.client.eth[0].arp



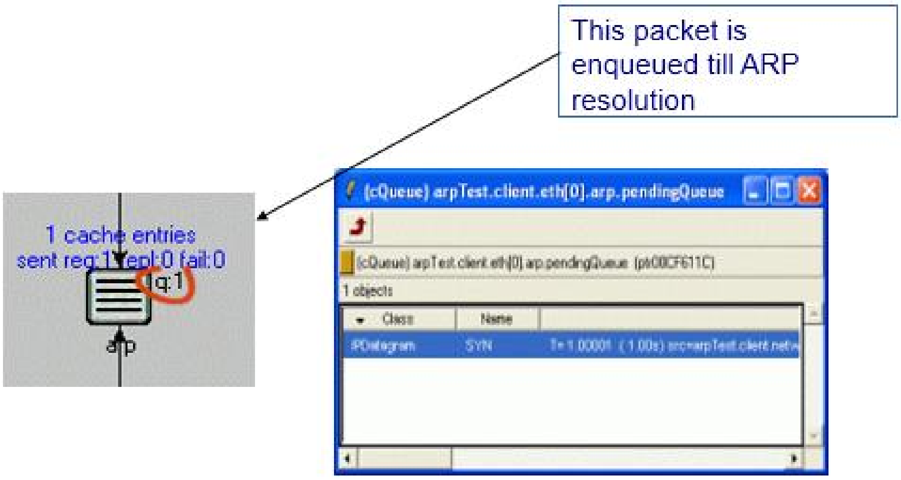
Inside ARP PaAcket



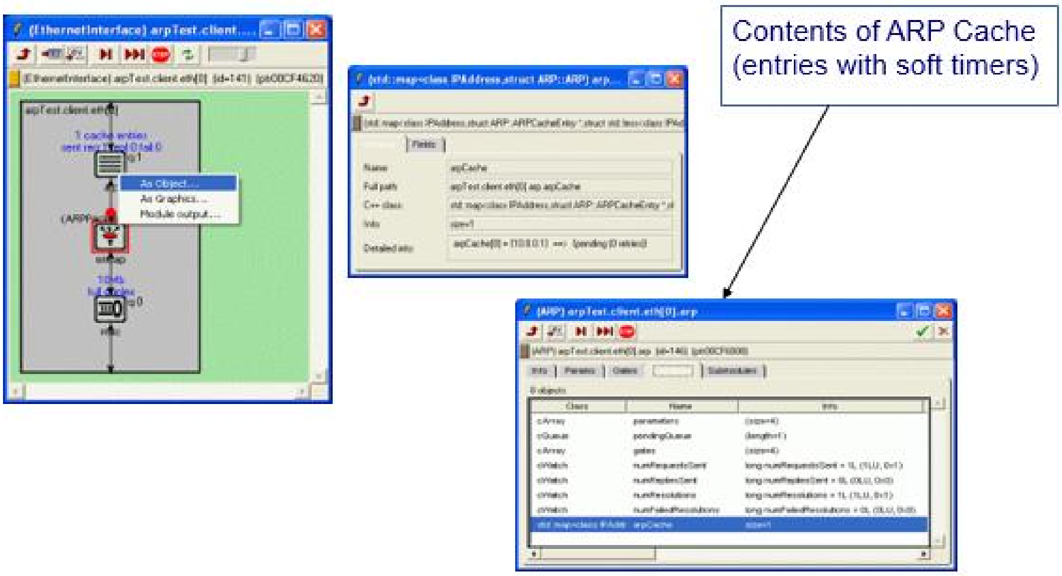
ARP Packet Class (Generated by .msg file)

```
// file: ARPPacket.msg
message ARPPacket
{
fields:
 int opcode enum(ARPOpcode);
 MACAddress srcMACAddress;
 MACAddress destMACAddress;
 IPAddress srcIPAddress;
 IPAddress destIPAddress;
};
```

Packet Queue (Contains IP Packet)



ARP Cache Build-up

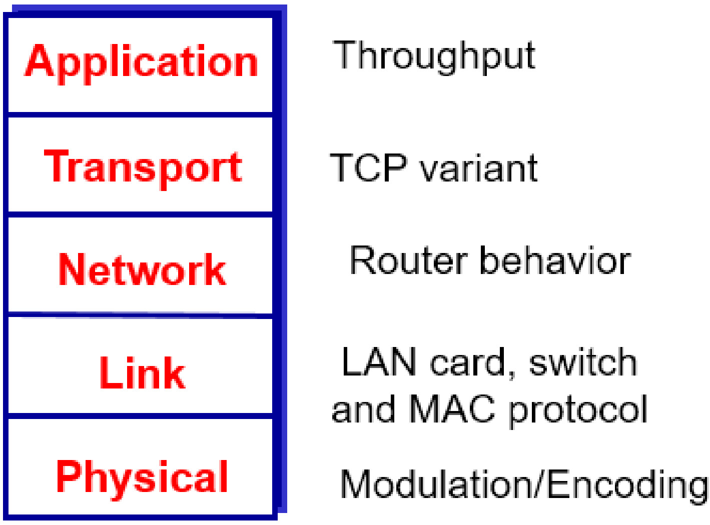


Introduction to top-down approach to modelling and simulation

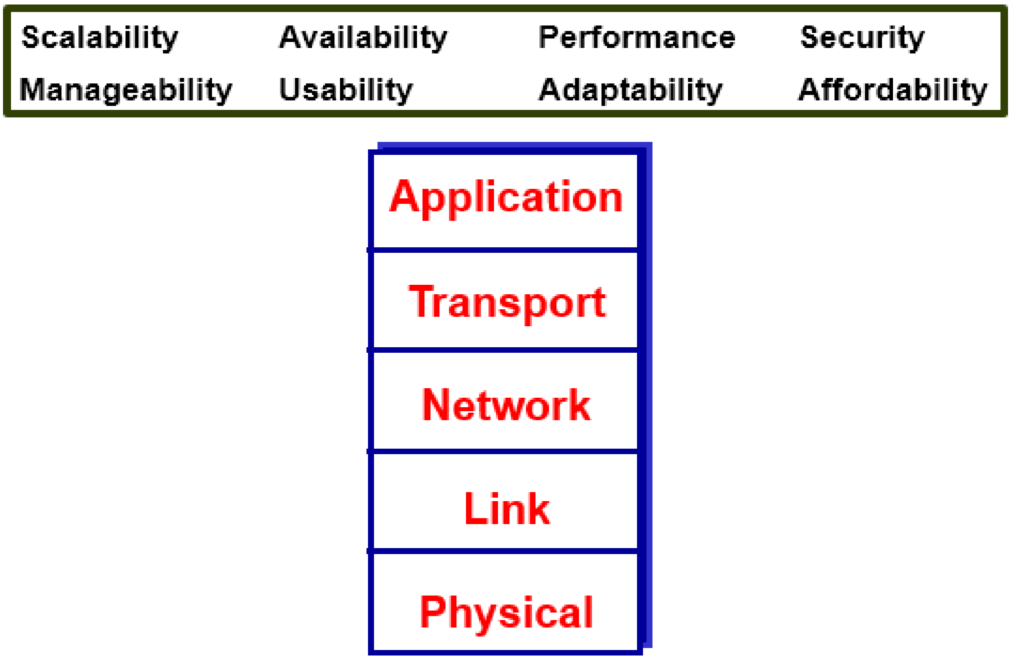
Top-down approach to NeMS

- Networks are complex to design
- One-time design of simulation is cumbersome
- **Top-down**: Phased roll-out of model-simulate cycle
    - — Iterative

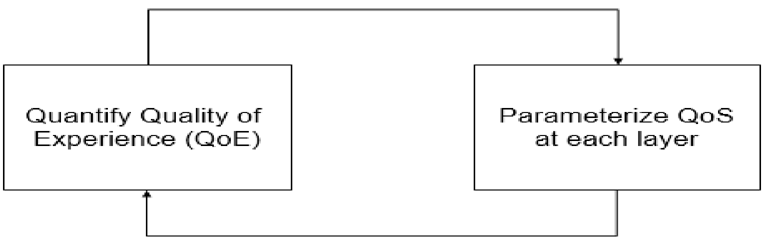
Rolling-out of model at every layer to Design a Network



Design goodness (QoE) is user-centric aspects



Strategy



Rules for Mathematical Reading

What is mathematical modeling?

A Representation of an object, a system, or an idea in some form other than that of the entity itself.
(Shannon)

Quantification

- The act of counting and measuring that maps human sense, observations and experiences into members of some set of numbers
- Facts represented as quantitative facts are the basis of science

Formalism

- Mathematics creates models that have certain relationships
- Statements of mathematics can be considered to be statements about the consequences of certain string manipulation rules

Best practices to read mathematical expressions

A) Understanding math is like understanding a foreign language
B) Learn the formulas you already understand
C) Always learn what the formula will give you and the conditions
D) Keep a chart of the formulas you need to know
E) Math is often written in different ways, but with the same meaning

What is an equation?

- A statement that the values of two mathematical expressions are equal
- indicated by "=" sign
- What is a formula then!

Constituents of an equation?

- Expressions consist of one or more of these arguments
  - Numerical constants
  - Symbolic names
  - Mathematical operators
  - Functions
  - Conditional expressions

Easy math writing

- 2-3-4 rule
- Consider splitting every
  - Sentence of more than 2 lines
  - Sentence with more than 3 verbs
  - Paragraph with more than 4 "long" sentences
- Use mnemonics
  - s for speed
  - v for velocity
  - t for time

Easy math writing

- Organize into segments
  - An entity intended to be read comfortably from beginning to end!
- Segments are standalone
  - Definite start
  - Definite end
- Segments should be represented **linearly**

QoE—Usability
Everything starts with "You"

Scalability	Availability	Performance	Security
Manageability	Usability	Adaptability	Affordability

What is usability?

- Usability (U_b) is defined as the ease of use with which network users can access the network and services
- Ergonomic and technological facilitation
  - Networks should make users' jobs easier
- Some design decisions have a negative affect on usability
  - Strict security
- Some choices are user friendly
  - WiFi
  - DHCP

Understanding usability
Sanjay Kumar Gupta, "Usability Models Based on Network Artifacts for Rural Development" Int. J. Computer Technology & Applications,Vol 4 (3),508-513

- U_b: Usability as ease of use
- U_e: Use effort
- U_b □ $1/U_e$

Usability expressions

$$U_b(NAD) \propto U_b(HB) + U_b(SWH) + U_b(BRG) + U_b(RTR) + U_b(GTW)$$

$$\sum_{I=1}^{N} \sum_{j=1}^{M} U_b(NAD)_{ij}$$

Connotations

- Usability (U_b) is expressed as a function of network devices
- The top-down approach implies that the assessment of overall usability has to be based on the performance of
  - Hubs/switches
  - Routers/gateways

QoE—Scalability
Ability to grow

What is scalability?

- Scalability refers to the ability to grow (or add)
- Factors to be added
    - — Number of applications
    - — Number of sites
    - — Addressing at sites
    - — No. of users
    - — No. of servers

Effects of growth

- Efficiency decreases with increasing factors
    - – But increases with increasing "other" factors
- Execution time increases with increasing factors
    - – But decreases with increasing "other" factors

Understanding efficiency & speed-up

- Execution time tends to vary with problem size
    - — Must be normalized when comparing network performance at different traffic volumes

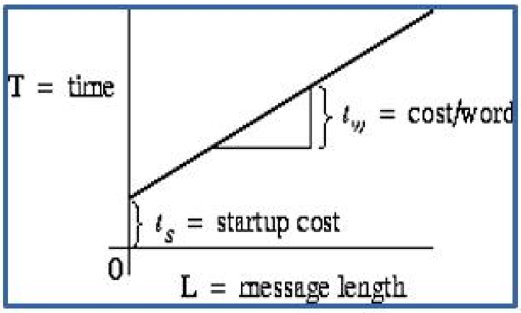
$E_{Relative} = T_1 \div (\text{No. of hosts} \times T_{\text{No of hosts}})$

$S_{Relative} = \text{No. of hosts} \times E_1$

Understanding execution time

$Time_{Execution} = T_{Compute} + T_{Comm} + T_{Idle}$

$T_{msg} = t_s + t_w L$



Connotations

- Scalability is expressed as a function of factors in the network
- This criteria affects the design choices made for the network model

QoE—Planning for Expansion
Need to expand is ever increasing
Why plan?

- Expansion is unavoidable
- Unplanned expansion causes performance degradation
  - Execution time
  - Efficiency
- Planning is necessary
  - Preemption is key
  - Late planning is no planning

Considerations for Planning

- Nodes and locations
  - End hosts
  - Switches
  - Routers
- Equipment scalability
  - No of ports
- Naming system
  - Extensible tuple
  (Node ID, Network ID)
- Application-specific protocol choices
  - Email
  - File transfer, sharing & access
  - DB access & updating
  - Web browsing
  - Network game
  - Remote terminal
  - Videoconferencing
  - Video on demand (VoD)

QoE—Expanding Access to Data
Scalability without continued access to data is futile

Access to data

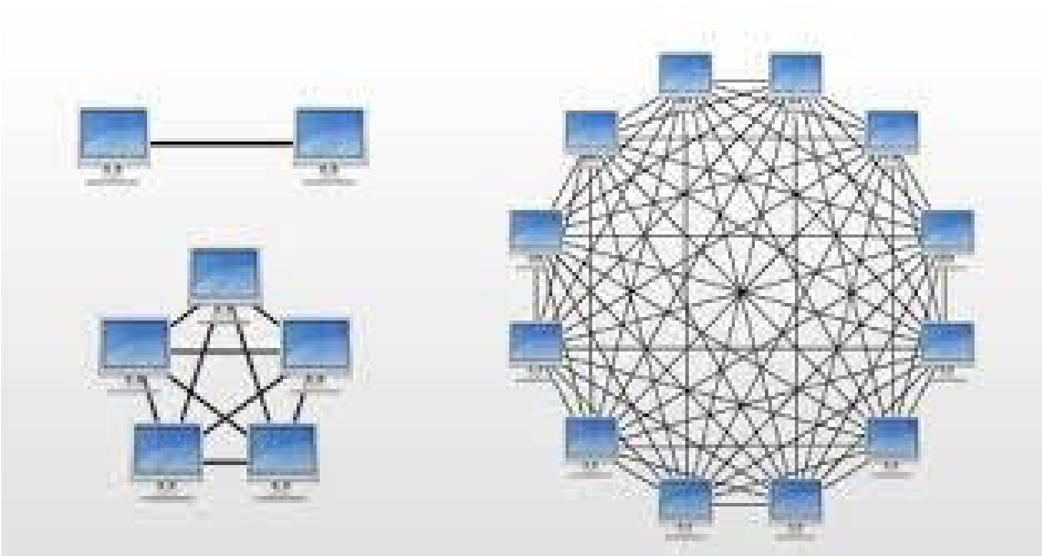
- Social networking has emerged
- Extranets need topology definitions & dedicated bandwidth allocation
  - Classic 80-20 rule ´
- Increased access
  - Data available to more departments
  - Increased utilization of network services

Metcalfe's Law

- Community value of a network grows as the square of the number of its users
- Often cited as an explanation for the rapid growth of the Internet

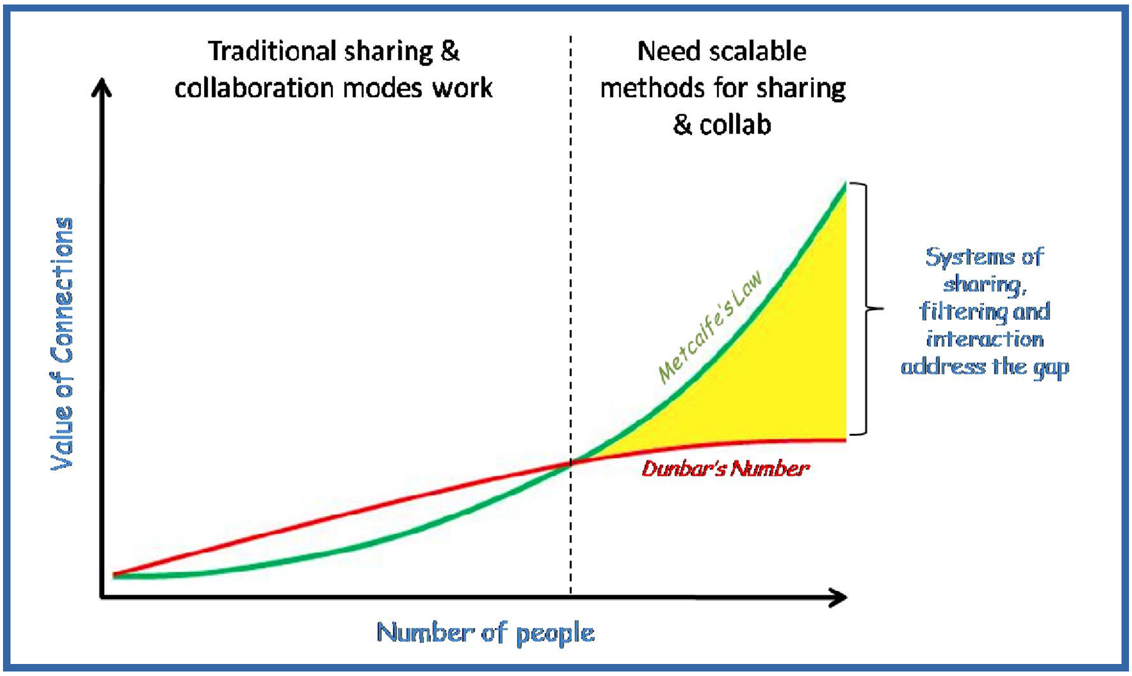
Expression for Metcalfe's Law

$n(n-1)/2$ or $O(n^2)$ connections between "n" nodes



Manifestation of Metcalfe's Law

- Can be seen in network applications



Connotations

- Network model is more scalable than the number of nodes and servers in the topology
- The total traffic load generated depends upon the user activity

QoE—Constraints on Scalability
The whole cannot be greater than the sum of its parts
(Apologies to Aristotle)

Recall parts of model!

- Nodes
  - Computing
  - Memory
- Protocols
  - Operation
  - Message formats
- Devices
  - Ports
  - Specifications
- And more!

Identify their upper bounds

- Nodes
  - N_{max}
- Protocols
  - Operation: O_{max}
  - Message formats: M_{max}
- Devices
  - Ports: P_{max}
  - Specifications: S_{max}

Maximum scaled up network

- Given by MinMax decision rule
  $Min(N_{max},O_{max},M_{max},S_{max})$
- The strength of the chain is determined by the weakest link

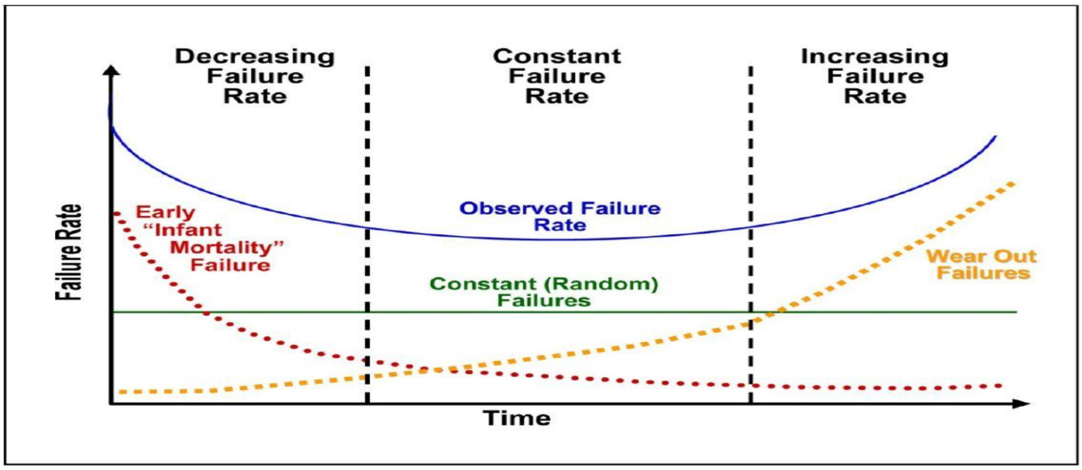
Specific example

- Constrained addressing
  - IPv4
  - Top-level exhaustion occurred on 31 Jan 2011
  - 24 Sep 2015 for North America
- Unconstrained addressing (for now!)
  - IPv6
- With everything as IoT, 2^{128} is the constraint

QoE—Availability
The degree to which a system, subsystem or equipment is in a specified operable and committable state

Everything may fail, not if but when!

- Networks Nodes Links
- Typical failure of components represented by the famous



Network Availability

- Percent uptime per year, month, week, day, or hour to total time in that period
  - For example:
    - 24/7 operation
- Network is up for 165 hours in the 168-hour week
  - Availability is 98.21%

Application perspective

- Applications may require different levels
  - Real time
    - Video/Audio
  - Commerce
    - Non-repudiable transactions
  - Non-real time
    - Email

Availability vs reliability

- Reliability is the ability of a system to complete its function
  - accuracy
  - error rates
  - Stability
- Even if a system is available does not mean its reliable

Availability vs capacity

- A system that runs out of capacity becomes unavailable
  - ATM connection admission control
  - Regulates no. of cells into network
- If capacity & QoS for connection unavailable
  - cells dropped

Availability vs redundancy

- Redundancy is not a goal
- It is provided to achieve a level of availability
  - Only a means!

Availability vs resiliency

- How much stress can be taken by network?
  - Availability difficult to maintain
  - No. of failures that make a system unavailable
- How soon can a network rebound?
  - Availability difficult to achieve

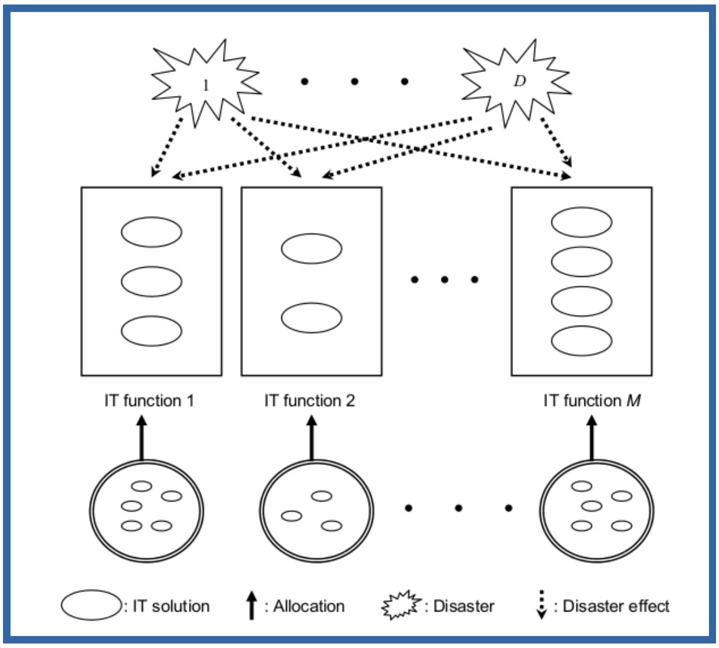
QoE—Disaster Recovery
Amat Victoria Curam (Latin)
Victory Loves Preparation

Benjamin B. M. Shao, "Allocating Redundancy to Critical Information Technology Functions for Disaster Recovery," *Proc. 10Th Americas Conference on Information Systems*, Aug. 2004

The question
How to allocate redundancy to IT functions such that the overall survivability of these IT functions against disasters is maximized and the cost remains under budget.

Redundancy
- Redundancy in preparation for disasters provides disaster preparation
  - Proactive prevention
  - Reactive recovery
  - Backup facilities

Redundancy Allocation Scenario



Redundancy Allocation Model
- IT function can be implemented by a number of IT assets
  - Computing hardware
  - Communication links
  - IT personnel, and
- other infrastructure

Redundancy Allocation Model

D: number of potential disasters + 1 (the last one for no disaster occurring);

p_d: probability of disaster d occurring, $p_d \in (0, 1)$ and $\sum_{d=1}^{D} p_d = 1$;

M: number of IT functions the organization needs to perform;

w_m: importance weight (or frequency of usage) of IT function m, $w_m \in (0, 1)$ and $\sum_{m=1}^{M} w_m = 1$;

n_m: number of solutions (assets) available for IT function m to select from;

X_{mi}: 1 if solution i (= 1, ..., n_m) is selected for IT function m, or 0 otherwise;

C_{mi}: cost of selecting solution i for IT function m;

S_{mid}: survivability of solution i for IT function m against disaster d;

e_{mid}: failure probability of solution i for IT function m against disaster d, $e_{mid} = 1 - S_{mid}$;

B: available budget.

(RAP) $$\max S^* = \sum_{d=1}^{D} p_d \sum_{m=1}^{M} w_m \left[1 - \prod_{i=1}^{n_m} e_{mid}{}^{X_{mi}}\right]$$

subject to

$$\sum_{i=1}^{n_m} X_{mi} \geq 1, m = 1, \ldots, M$$

$$\sum_{m=1}^{M} \sum_{i=1}^{n_m} C_{mi} X_{mi} \leq B$$

$$X_{mi} = 0 \text{ or } 1, \quad \text{for } m = 1, \ldots, M \text{ and } i = 1, \ldots, n_m$$

At least one IT solution be selected and allocated to each IT function

$$\text{(RAP)} \quad \max S^* = \sum_{d=1}^{D} p_d \sum_{m=1}^{M} w_m \left[1 - \prod_{i=1}^{n_m} e_{mid}^{\ X_{mi}} \right]$$

subject to

$$\sum_{i=1}^{n_m} X_{mi} \geq 1, m = 1, \ldots, M$$

$$\sum_{m=1}^{M} \sum_{i=1}^{n_m} C_{mi} X_{mi} \leq B$$

$$X_{mi} = 0 \text{ or } 1, \quad \text{for } m = 1, \ldots, M \text{ and } i = 1, \ldots, n_m$$

Guarantees that the total costs of Redundancy allocation not exceed the budget limitation

Redundancy Allocation Model

- *m* fails against *d* only when all of its selected solutions fail at same time
  - As long as one of the selected solutions survives, *m* would still be operational

QoE—Specifying Requirements

Measurable is achievable

Availability in %age per annum

- Uptime of 99.70%
  - 30 mins downtime
- Uptime of 99.95%
  - 5 mins downtime
- Map onto totally deviant requirements

Availability in calendar year

- Downtime on weekdays
  - vs weekends
- Project deadlines

Availability in spurts

- Staggered vs onetime
- 99.70% uptime
  - 30 minutes per year
  - 10.70 sec per hour
- Acceptable for some users not to others
- Allowed for few applications

QoE—Five Nines Availability
The devil is in the details of availability

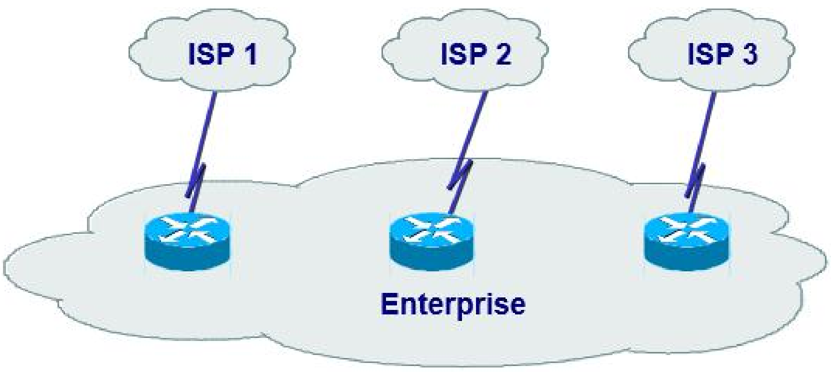
5 9s as best-case availability

- Some enterprises may want 99.999%
  - 5 minutes downtime per year
- Sometime or all the time?
  - A million $ worth question for managers
- Repair time inclusive or exclusive
  - In service upgrades (hot-swaps) possible?
- Hardware manufacturers provide 5 9s
- However sum is not equal to parts
  - Carrier and power outages
  - faulty software in routers & switches
- Unexpected and sudden increase in bandwidth or server usage
- Configuration problems, human errors (90% of all!)
- Security breaches, and software glitches

Shifting Impact of 9s on time

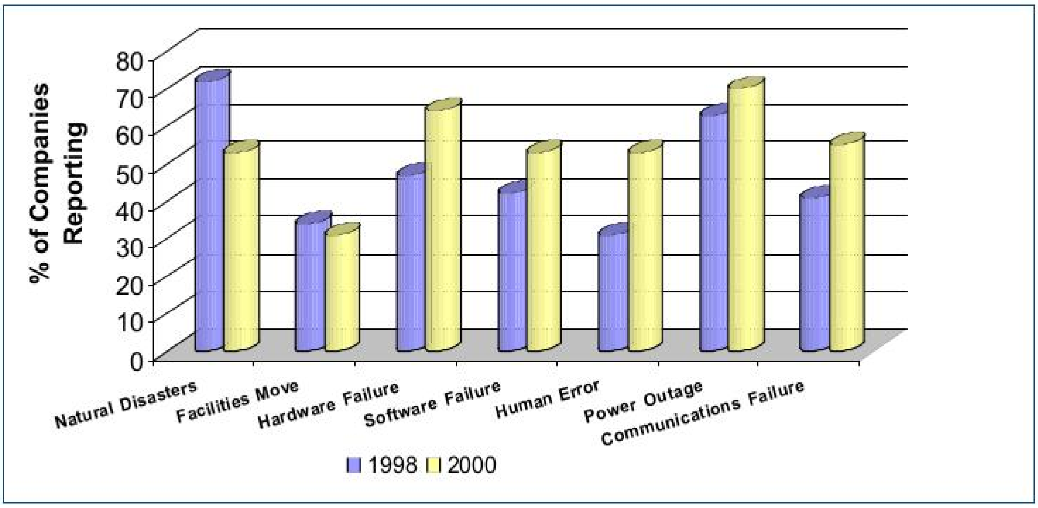
	Per Hour	Per Day	Per Week	Per Year
99.999%	.0006	.01	.10	$\boxed{5}$
99.98%	.012	.29	2	105
99.95%	.03	.72	$\boxed{5}$	263
99.90%	.06	1.44	10	526
99.70%	.18	$\boxed{4.32}$	30	1577

99.999% Availability might require triple redundancy

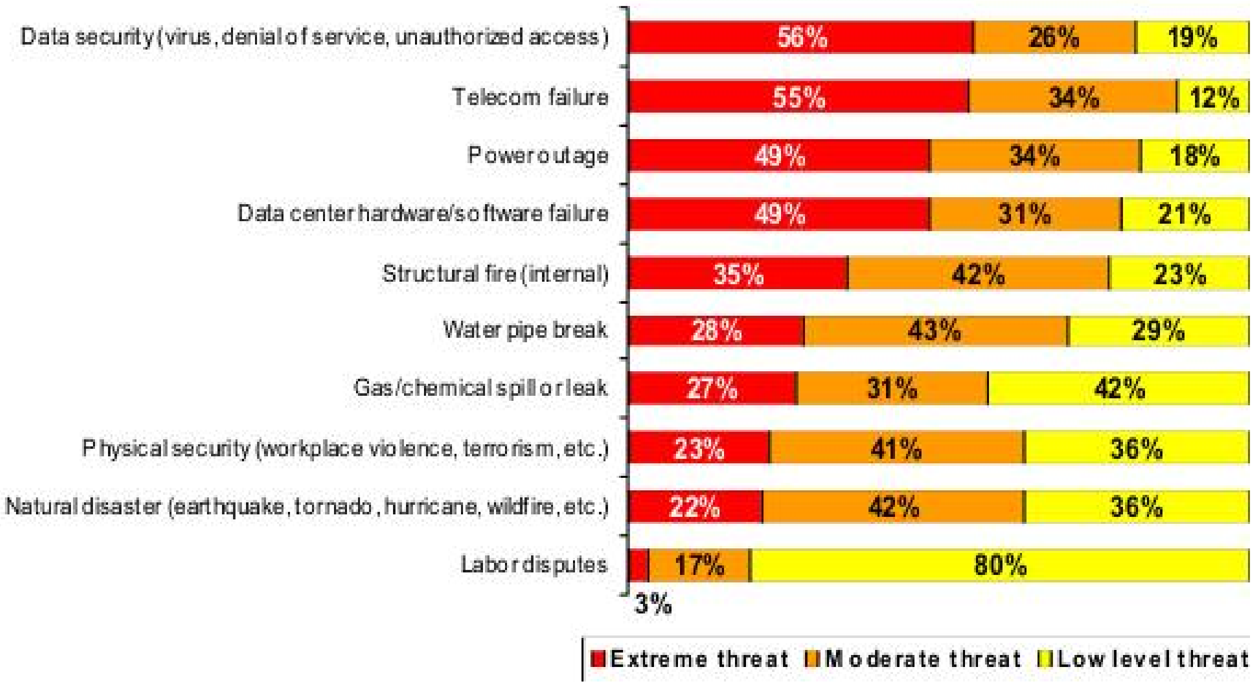


One being active, one in hot standby ready to be used immediately, one in standby or maintenance

QoE—Cost of downtime
40 percent of companies that
shut down for three days failed within 36 months (Contingency Planning and Management magazine)



Source: Top Business Continuity Priorities for 2004.©EnvoyWorldWide - February, 2004



Source: New England Disaster Recovery Information X-Change (NEDRIX)

Step-wise approach to measure downtime cost

1. Identify Business Continuity Components
2. Define What You Protect
3. Prioritize Business Functions
4. Classify Outage Types,
5. Calculate cost

Identify Business Continuity Components

- People
- Property
- Systems
- Data

Define What You're Protecting

- Define core competencies
  - — product, service, process, or methodology

Prioritize Business Functions

- Business functions necessary to sustain that core competency
  - — And associated IT infrastructure
- 80% of available resources restore 20 % systems, applications, and data

Outage Types, Frequencies, & Duration

- Branch Outage
- Regional outage
- Data center outage
- National outage

Calculate cost

Frequency x Duration x Hourly Cost = Lost Profits

Outage	Minimum Impact	Maximum Impact
Branch	1X	5X
Data Center	2X	10X
Regional	0.2X	1X
National	1.5X	1.5X
Total	3.5X	15X

Example

- If there were 90 branch outages in an average year
  - — Each lasting an average of one-and-a-half hours
  - — Costing $300/hour 90 outages x 1.5 hours x $300/hour = $ 40,500
- Cost of branch outages for a year =$40,500

QoE—MTBF AND MTTR
Averaging out the availability

Availability as MTBF

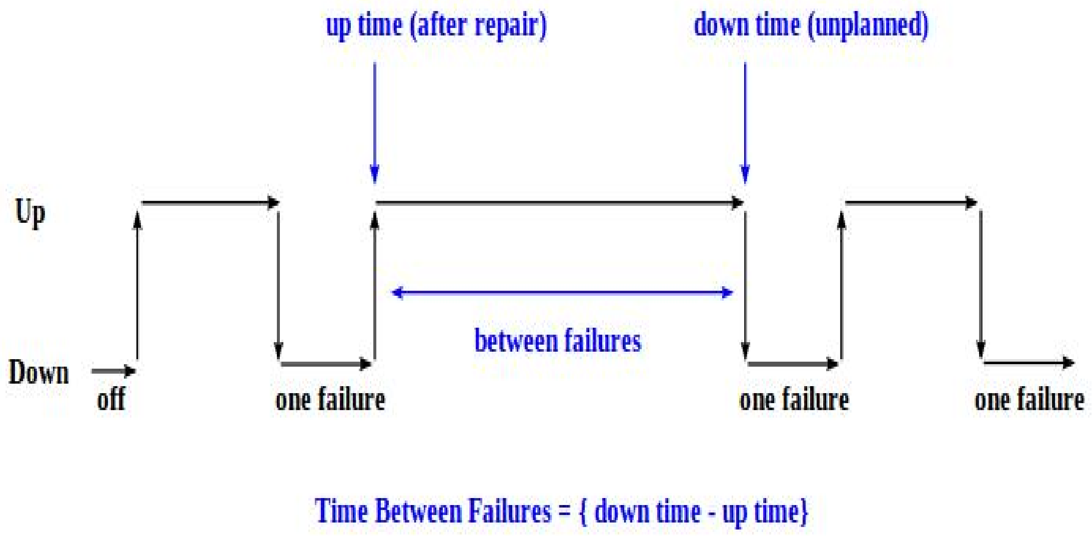
- Mean time bw failure (MTBF) & mean time to repair (MTTR)
- Component vs service
  - Mean time bw service outage (MTBSO)
  - Mean time to recover from service outage (MTTSO)
- Typical MTTF value is once per 4000 hrs or 166.7 days
- Typical acceptable MTTR value is one hour

Availability = MTBF/(MTBF + MTTR)

4,000/4,001 = 99.98% availability

- MTBF with MTTR help to assess frequency and length of service outage
  - Mean value must be supported with variance

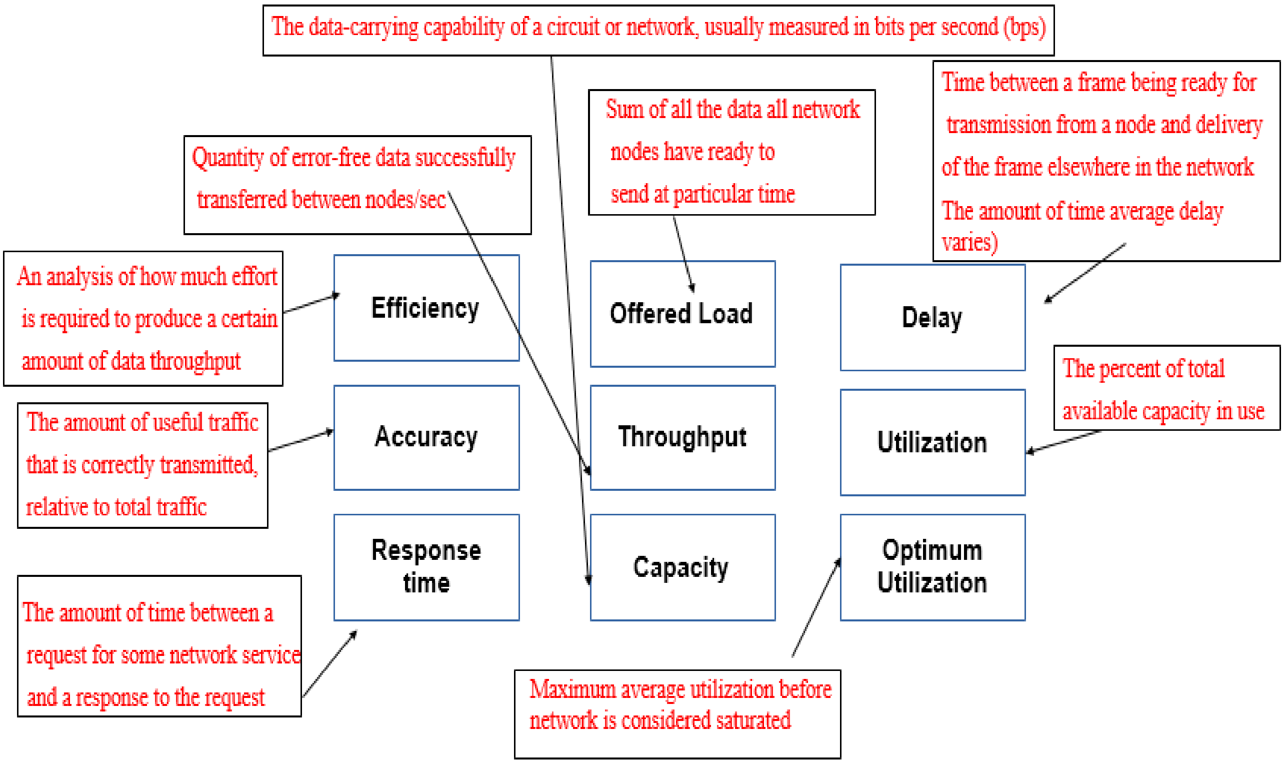
- The difference between MTTF and MTBF is the assumption of the former that the system shall be repaired while in the later the system is replace



QoE—Network Performance
Composite metric that is end-to-end

Definition

- An overall working
- Many different ways to measure the performance of a network
  - Each network is different in nature and design
- Modeled
- Simulated
- Measured



QoE—Optimum Network Utilization
Optimum is "As good as it gets"

Definition of optimum

- Selection of a best element (with regard to some criteria) from some set of available alternatives

Optimum network utilization

- How much % of bandwidth capacity in a specific time period?
- Time varying phenomenon
  - Instantaneous, averaged, weighted)
- Both goal & constraint
- Typical value is 70$%
  - Exceeding this results in performance degradation
- WAN links utilization is more crucial than LAN
  - Pay per packet
- Compression, caching and concatenation used to reduce WAN utilization
- LANs are over-budgeted
  - Fast Ethernet)
- Full-duplex vs. half duplex switches
- User activity levels
- LANs suffer from exceeding utilization in switch-to-switch

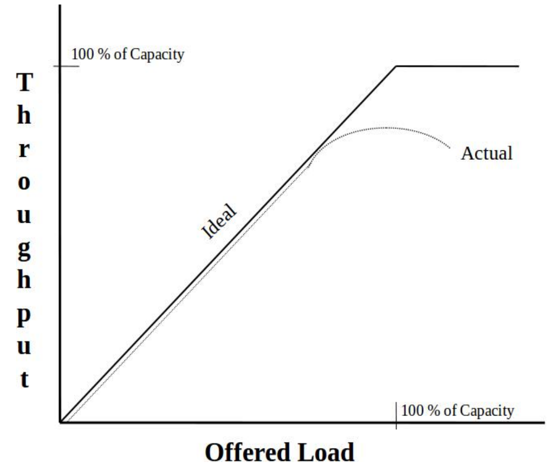
QoE—Throughput
Throughput = Goodput + Badput

Definition of throughput

- Quantity of error free data transmitted/ sec
  - Erroneous transmissions futile
- Ideally, should be the same as capacity

$$\boxed{C = B\log_2\left(1+\frac{S}{N}\right)}$$

Deviation indicates the limitations of media type, device and network



QoE—Throughput of devices
Simulation of devices and specifications is vendor specific

Types of device throughputs

- Inter-networking devices give throughput as in
  - TCP/IP: Packets per second
  - ATM: Cells per second
- Sizes vary from 53, 64 to 1518 Bytes

Example—CISCO devices

- Traffic generators-device-traffic checkers in tandem measure throughput
  - Smaller packets give better pps
- Cisco claims of 400 million pps for the Cisco Catalyst 6500 switch
  CISCO claims throughput; which in actual is the capacity

Table 2-1 *Theoretical Maximum Packets per Second (pps)*

Frame Size (in Bytes)	100-Mbps Ethernet Maximum pps	1-Gbps Ethernet Maximum pps
64	148,800	1,488,000
128	84,450	844,500
256	45,280	452,800
512	23,490	234,900
768	15,860	158,600
1024	11,970	119,700
1280	9610	96,100
1518	8120	81,200

QoE—Application Layer Throughput
Application layer uses lower layers unfairly

Definition

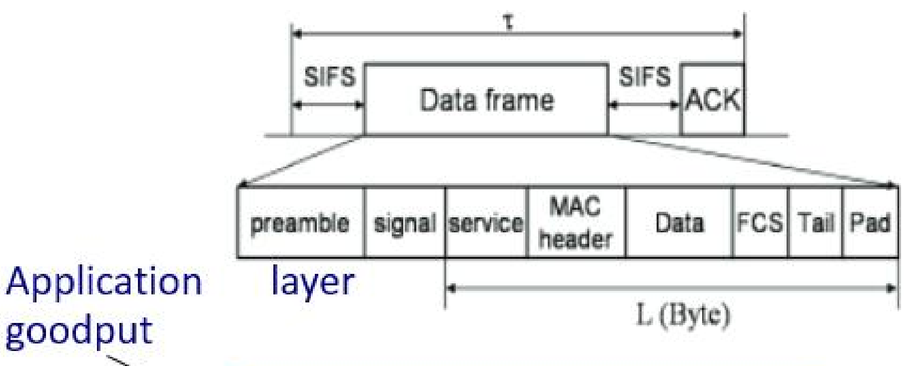
- Application layer throughput = goodput + badput
- Goodput vs badput
- Badput contributed by retxns, header etc
  - Fraction of packets that collided/lost
    - Fc = C/N
    - Fc = L/N

Factors affecting goodput

- End-to-end error rates
- Protocol functions (handshaking, windows, & acks)
- Protocol parameters (frame size, retx timers)
- pps rate of networking devices
- Lost packets at networking devices
- Workstation & server performance factors:
  - Disk-access speed
  - Disk-caching size
  - Device driver performance
- Computer bus performance (capacity/arbitration)
- Processor (CPU) performance
- Memory performance (access time for real and virtual memory)
- Operating system inefficiencies
- Application inefficiencies or bugs

An, Cheolhong, and Truong Q. Nguyen. "Error Resilient Video Coding using Cross-Layer Optimization Approach." IEEE Transactions on Multimedia 10 (2008): 1406-1418.

$$\boxed{\tau = T_{Preamble} + T_{Sig} + \frac{8LT_{symbol}}{N_{SD} r \log_2 M_o} + 2SIFS + \tau_{ack}}$$



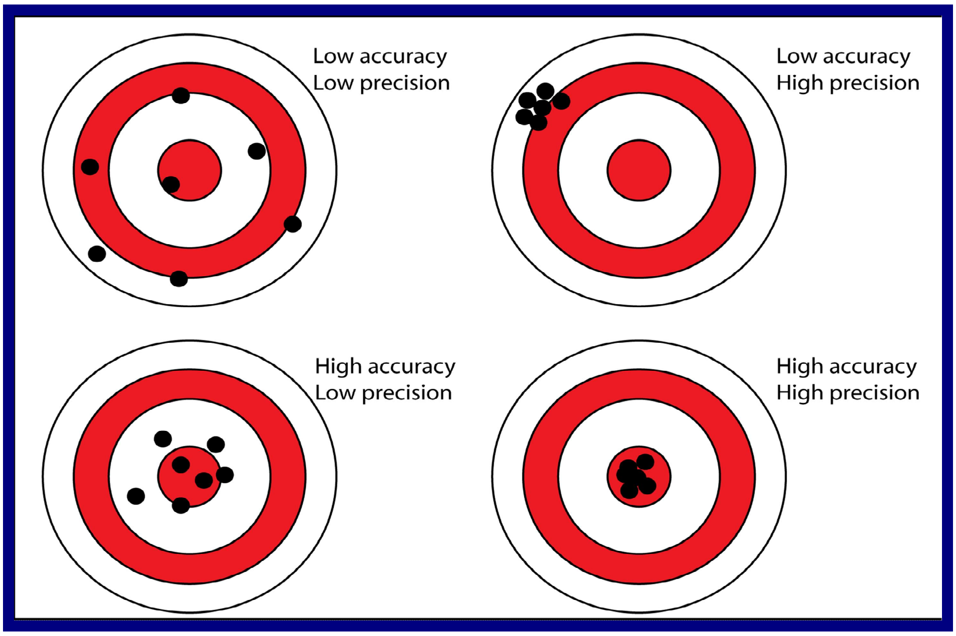
$$\boxed{x_{gp} = \frac{8(L - h_{ov})}{T_{avg}} = \frac{8(L - h_{ov})(1 - P_{fr})}{\tau}}$$

Connotations

- Application layer throughput provides insight into “useful' transmissions
  - It relates resource allocation down to physical layer throughput

QoE—Accuracy

Being accurate is not being precise



Definition

- Data sent and received should be the same
- Also referred as the number of error-free frames transmitted relative to the total number of frames transmitted

Factors affecting accuracy

- Packet reordering at routers
- Power surges
  - Lightning impulse of 1 s on 10 Mbps link
- Impedance mismatch problems
- Poor physical connections
- Failing devices
- Noise caused by electrical machinery
- WAN links give BER and SNR (10^{-5}~10^{-11})
- LANs specify erroneous frames per 10^{6} Bytes
- On shared Ethernet, collisions main cause of accuracy degradation
- First 64 Bytes collision (legal or runt frames)
- Typical acceptable value is .1% frames
- Late collisions are illegal

- Nahum, Erich M. "Validating an architectural simulator." Department of Computer Science, University of Massachusetts at Amherst. 1996.

Accuracy = [(Real value – Error) / Real value] * 100

The frequency of events plays a key role in the overall accuracy
- E_i is the event i in the system
- $freq(E_i)$ is the frequency of event i
- $real(E_i)$ is the desirable (real) cost of event i
- $sim(E_i)$ is the simulated (obtained) cost of event i

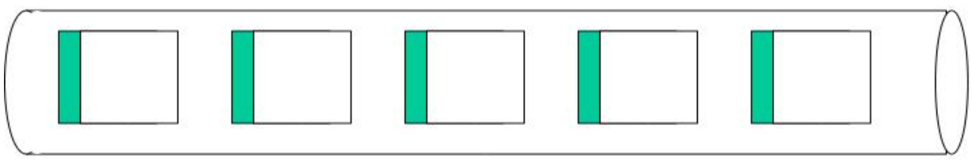
$$\boxed{Error = \sum_{i=1}^{n} freq(E_i) * (real(E_i) - sim\ (E_i))}$$

QoE—Efficiency
Boiling water analogy

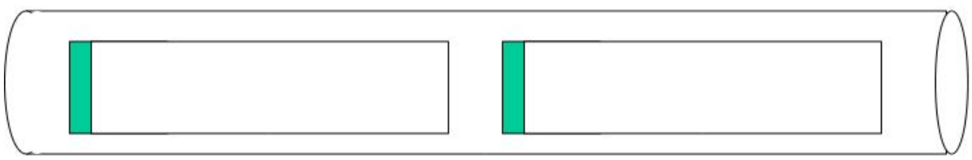
Definition
- Application layer throughput = goodput + badput
- Goodput vs badput
- Badput contributed by retxns, header etc
  - Fraction of packets that collided/lost
    - Fc = C/N
    - Fc = L/N

Factors affecting efficiency
- Access protocols
  - high number of users showing activity
  - Ethernet not efficient at high collision rates
- Frame size
  - Using large frame is useful for single user on WAN links
- Serialization delay on WAN links results in unfair treatment
  - for real-time shorter frames enquired in router

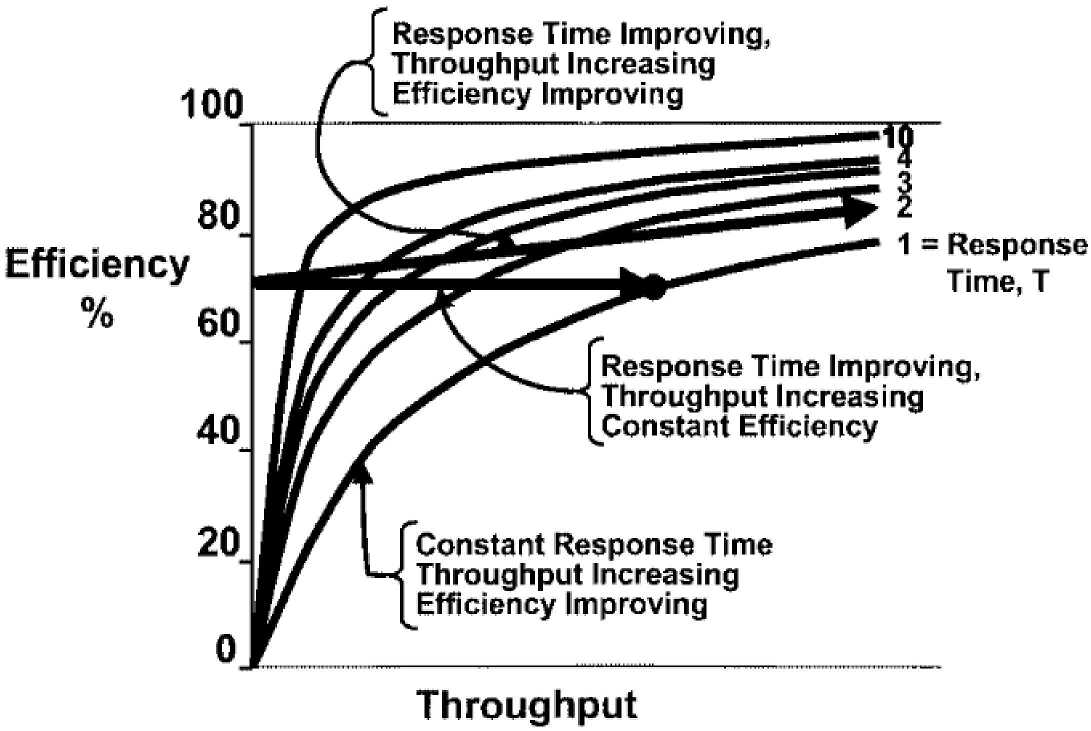
Small Frames (Less Efficient)



Large Frames (More Efficient)



Kleinrock, Leonard. "Creating a mathematical theory of computer networks." Operations Research 50.1 (2002): 125-131.



If you scale capacity more slowly than throughput while holding the average response time constant, then the channel efficiency (channel utilization) will increase

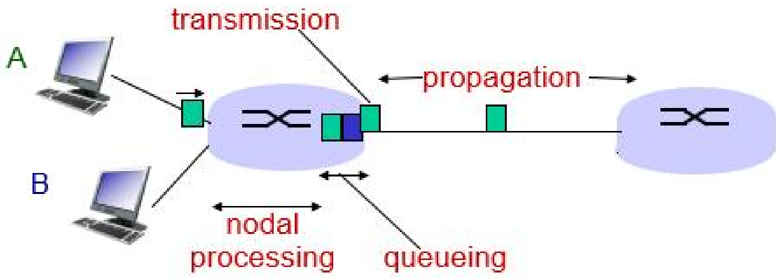
Average Efficiency
Latora, Vito, and Massimo Marchiori. "Efficient behavior of small-world networks." Physical review letters 87.19 (2001): 198701.

$$\boxed{E(G) = \frac{2}{n(n-1)} \sum_{i<j\in G}^{n} \frac{1}{d(i,j)}}$$

- E(G) is the average efficiency of a network G
- n denotes the total nodes in a network
- d(i,j) denotes the shortest path between a node i and a neighboring node j

QoE—Delay and Jitter
Applications might forgive delay but not jitter
Delay
- Voice and video applications (especially interactive) demand minimum delay
- Other applications such as Telnet remote echo need timed performance

Sources of packet delay

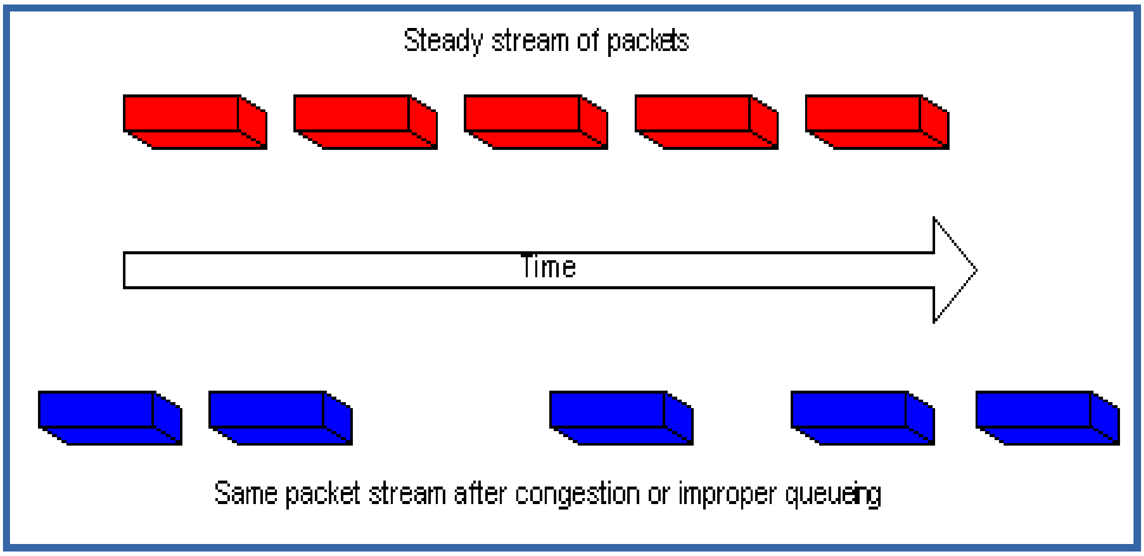


$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

Delay variation (jitter)

- The amount of time average delay varies
- Voice, video, and audio are intolerant of delay variation

Source of jitter



QoE
It is the small factors that matter the most

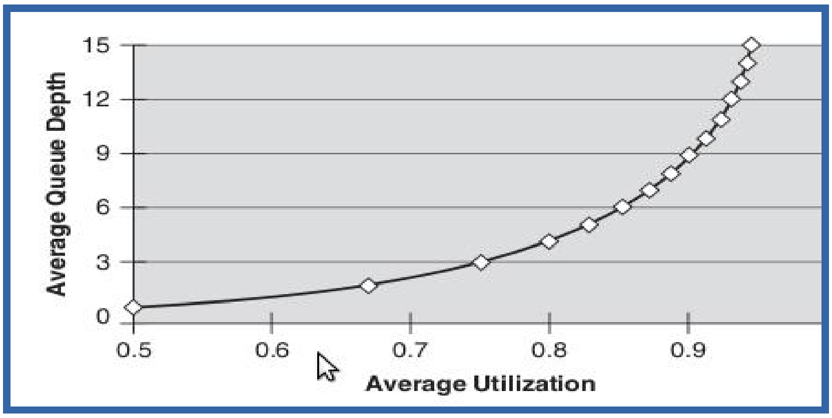
Causes of Delay

- Propagation
  - Media type
  - Length
- Transmission (serialization)
  - 1024 Bytes on T1
- Switching delay
  - upto 5-20 microsec for 64 Bytes frame

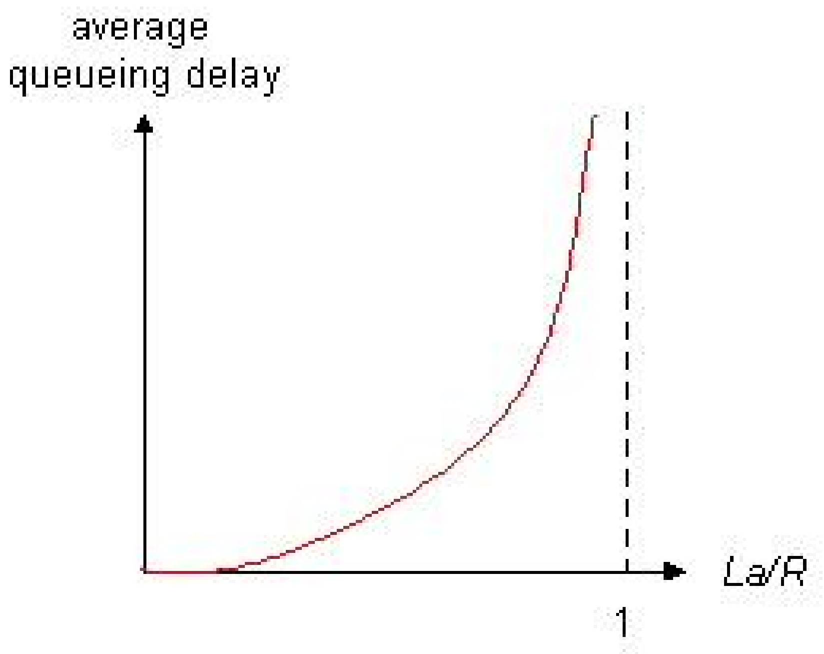
- Router delay
  - Look-up, router architecture, configuration
  - Software features that optimize the forwarding of packets
- NAT, IPSEC, QoS, ACL **Causes of Delay (3 of 3)**
- Queuing delay
  - Dependent upon utilization
- **Formula**

  Queue depth = Utilization/(1 – Utilization)

Queue Depth vs. Utilization



Implications of queuing delay



QoE—Delay variation
All animals are equal, but some animals are more equal than others (George Orwell)

Delay variation

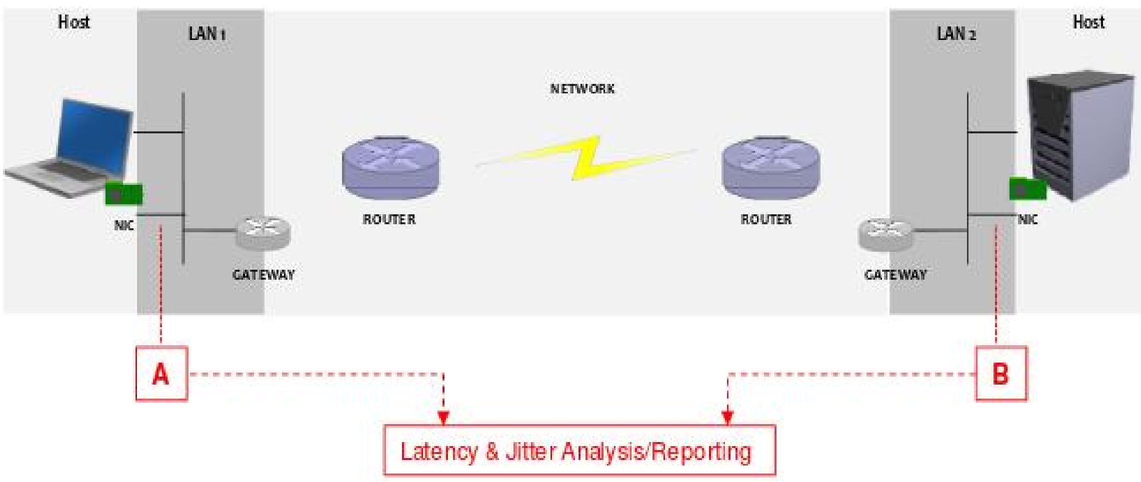
- Amount of time average delay varies
- Voice, video, and audio are intolerant of delay variation
- Tradeoffs needed for efficiency for high-volume applications versus lowConcept of jitter buffer to smoothen out the jitter
- Variations on the input side are smaller than the buffer
- Acceptable variation is 1-2% of the delay

Jitter types

- Jitter is quantified in two ways
- Delay jitter
  - bounds maximum difference in total delay of different packets
  - Assumes source is perfectly periodic
- Used for Interactive communication
  - voice and video teleconferencing
- Helps to translate to maximum buffer size needed at the destination Second measure is rate jitter
- Bounds difference in packet delivery rates at various times
- Measures difference between minimal and maximal inter-arrival times (reciprocal of rate)
- Useful measure for many real time applications
- Video broadcast over the net
- Slight deviation of rate translates to only a small deterioration in the perceived quality

Jitter Analysis Points

Kay, Rony. "Pragmatic network latency engineering fundamental facts and analysis." cPacket Networks, White Paper (2009): 1-31.



Measurement of jitter

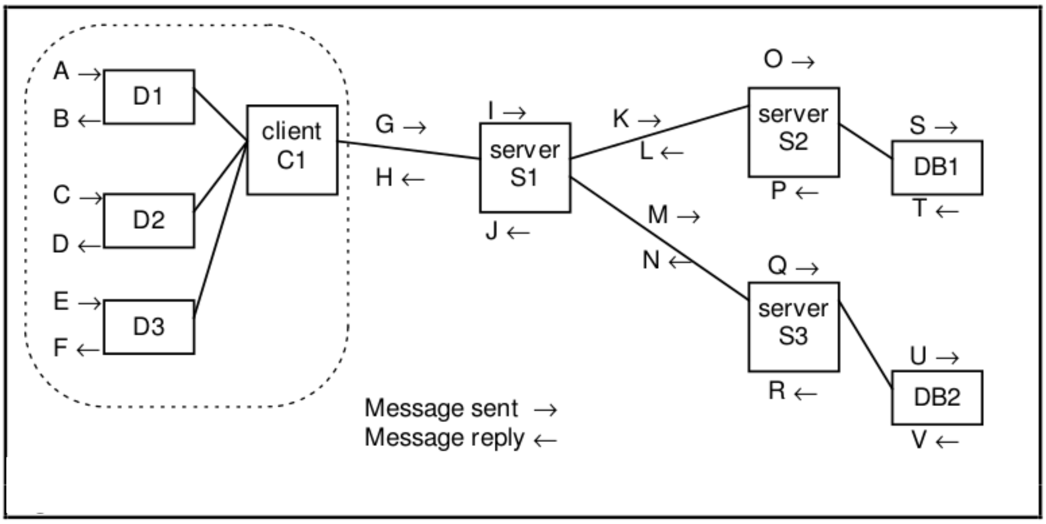
Packet ID	Time at Point A	Time at Point B	Latency	Jitter
1	TA_1	TB_1	$L_1 = TB_1 - TA_1$	---
2	TA_2	TB_2	$L_2 = TB_2 - TA_2$	$L_2 - L_1$
.	.	.	.	.
m	TA_m	TB_m	$L_m = TB_m - TA_m$	$L_m - L_{m-1}$
.	.	.	.	.
I	TA_i	TB_i	$L_i = TB_i - TA_i$	$L_i - L_{i-1}$
.	.	.	.	.
n-m+1	TA_{n-m+1}	TB_{n-m+1}	$L_{n-m+1} = TB_{n-m+1} - TA_{n-m+1}$	$L_{n-m+1} - L_{n-m}$
.	.	.	.	.
n-1	TA_{n-1}	TB_{n-1}	$L_{n-1} = TB_{n-1} - TA_{n-1}$	$L_{n-1} - L_{n-2}$
N	TA_n	TB_n	$L_n = TB_n - TA_n$	$L_n - L_{n-1}$

QoE—Response Time
Response time is relative phenomenon

Definition

- The amount of time between a request for some network service and a response to the request

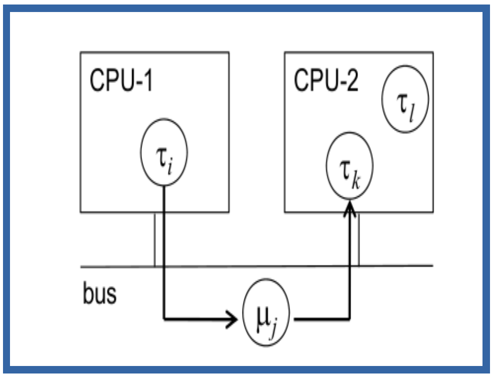
Measurement Points Locations
Tim R Norton. "End-To-End Response Time: Where to Measure?" Computer Measurement Group Conference Proceedings, 1999.



Measurement of Response Time

[1] Reinder J., Bril., System Architecture and Networking. TU/e Informatica
[2] Sjodin, Mikael, and Hans Hansson. "Improved response-time analysis calculations." Real-Time Systems Symposium, 1998. Proceedings., The 19th IEEE. IEEE, 1998.

Measurement of Response Time



1. a strictly periodic *system* event activates a task τ_i
2. task τ_i sends message μ_j
   - FJ_i causes AJ_j
3. message μ_j triggers task τ_k
   - FJ_j causes AJ_k
4. task τ_k generates a *system* response

AJ_k influences *system* response and response times of task τ_l with a lower priority

Measurement of Response Time
Ceiling function represents maximum number of pre-emptions by higher priority processes

$$R_i = B_i + C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i + J_j}{T_j} \right\rceil C_j$$

$\boxed{R_i\text{: worst case response (computation) time}}$
B_i: maximum blocking time from lower priority processes
C_i: the worst case computation time
$hp(i)$: the set of processes with higher priority than process i
J_j: the maximum jitter variation in activation times
(e.g. output of one task triggers a next task)
T_j: The period (or minimum inter-arrival time)
C_j:the worst case computation time

QoE—Security
Threat = Capability + Intention

Definition
- Protection of information systems from threat
  - Hardware
  - Software
  - Information on them
- Avoidance from
  - Disruption
  - Misdirection of the services they provide

Implementation
- Includes controlling physical access to the hardware
- Protecting against harm via
  - Network access
  - Data
  - Code injection

Trusted Computing Base
- Rainbow Series(orange book)
- Set of all hardware, firmware, and/or software components
- Critical to its security
- Bugs occurring inside jeopardize security of entire system

Bell-Lapadula Model
- Users as Subjects
- Predicates
  - Devices and data as Objects
- Process algebra provides the action (verb) of subject over predicates

Bell-Lapadula Model
- Users as Subjects
- Predicates
  - Devices and data as Objects
- Process algebra provides the action (verb) of subject over predicates

QoE—Reconnaissance Attacks
Prevention is better than cure

Definition
- Reconnaissance is a type of computer attack
- Intruder engages with the targeted system
  - Gathers information about vulnerabilities

Types
- Active reconnaissance
- Port scanning
- Passive reconnaissance
- Sniffing
- War driving
- War dialing

Targeted Threat Index

Hardy, Seth, et al. "Targeted threat index: Characterizing and quantifying politically-motivated targeted malware." Proceedings of the 23rd USENIX Security Symposium. 2014.

Targeted Threat Index

- Vulnerability of system
- Depends upon
  - Target feature set
  - Attacker methods
  - Attacker aggressiveness

  TTI = Method * Implementation

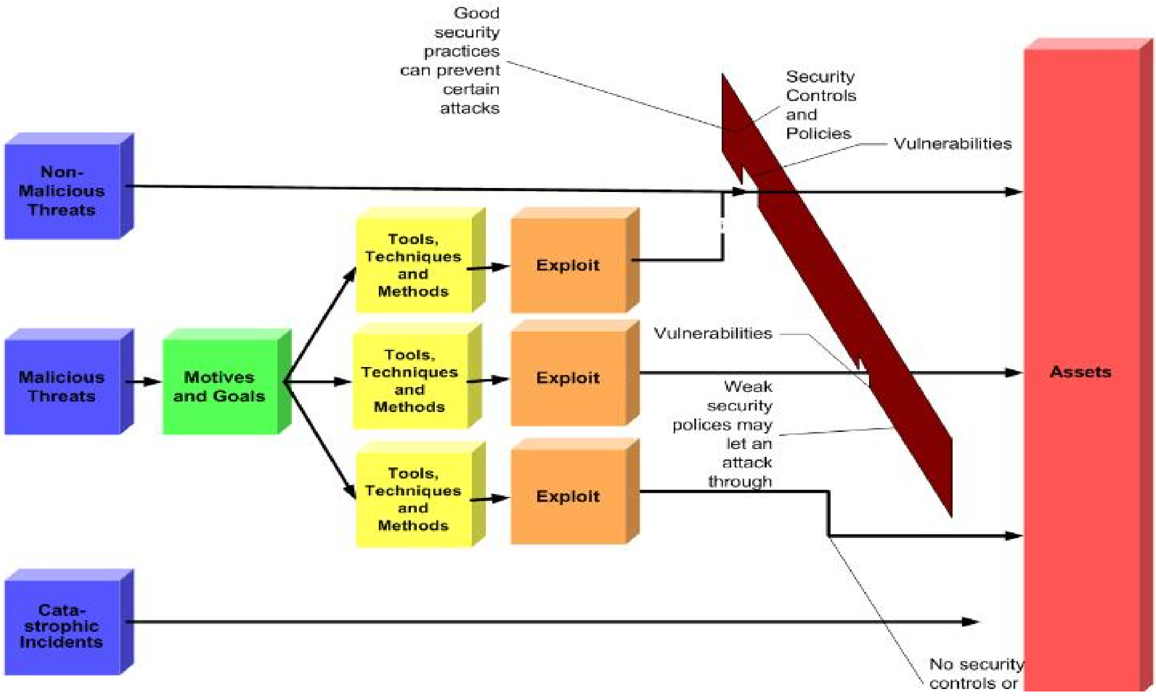
QoE—Security Requirements

Definition

- Enlist all the activities, actions, hardware/software
- Confidentiality
- Integrity
- Authorization
- Authenticity
- Availability
- Encryption

Assessing Security Levels

Burchett, Ian. "Quantifying Computer Network Security." (2011).



Common Vulnerability Scoring System

- Provides a repeatable quantitative score for computer security vulnerabilities

Vulnerability Compositing Method per Client

$$V(v) \rightarrow CVSS\ Base\ Score\ for\ Given\ Vulnerability$$

$$S(v) = 1 - {}^{V(v)}/_{10}$$

$$S(v_1, v_2, \ldots, v_n) = \prod_{i=1}^{n} S(v_i)$$

$$H(v_1, v_2, \ldots, v_n) = 10(1 - S(v_1, v_2, \ldots, v_n))$$

QoE—Manageability
Definition

- The level of human effort required to keep that system operating at a satisfactory level
  - Deployment
  - Configuration
  - Upgrading
  - Tuning
  - Backup
  - Failure recovery

Assessing Manageability
Candea, George. "Toward Quantifying System Manageability." UseNix HotDep. 2008.

Manageability Metric

$$Manageability = \frac{TotalTime_{\mathrm{eval}}}{\sum_{\mathrm{i}=1}^{n} Weight_{\mathrm{i}} \times Time_{\mathrm{i}} \times Steps_{\mathrm{i}}}$$

\

The notion of efficiency of management operations, which is approximated by the time $Time_i$ the system takes to complete $Task_i$
Approximate complexity of a management task by the number of discrete, atomic steps ($Steps_i$) required to complete $Task_i$

Commentary

- Manageability is reduced proportionally to how long the management tasks take
- And to how many atomic steps are involved in each such task
- The fewer steps there are, the lower the exposed complexity of the system
- The faster the management tasks can be completed, the lower the likelihood of trouble
- Less management a system requires (i.e., the longer $TotalTime_{eval}$ for the same N_{total}), the easier it is to manage
- Equivalently, the less the system needs to be managed, the better

QoE—DoS Attack
Definition

- An attempt to make a machine or network resource unavailable to its intended users,
- Temporarily
- Indefinitely

Implementation

- Transmit a large number of packets
  - TCP Syn attack
  - Ping attack
- Server crashing attack
  - Large computational load

A Simple Attack Analysis
He, Changhua. Analysis of security protocols for wireless networks. PhD Diss. Stanford University, 2005.

A Simple Attack Analysis

- Attack type: TCP SYN flooding DoS attacks
  - n packets are used for attack

- Counter: Random drop queue 'Q'
  - Q = queue depth

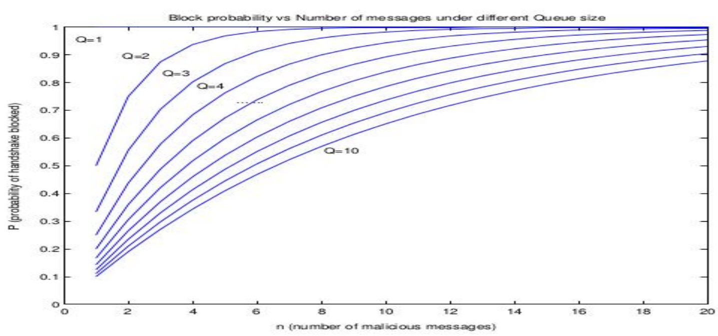
Attack success probability

- $P = 1 - (1 - 1/Q)^n$

Attack failure probability

- 1-P

A Simple Attack Analysis



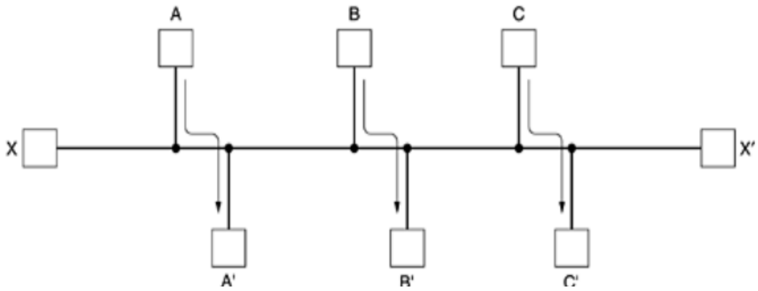
Making Network Design Tradeoffs
Definition

- Make balance between desirable & incompatible features
- A compromise
- Often conflicting technical goals
- Make tradeoff a necessity
  - Availability vs affordability
  - Usability vs security

A Simple Communication tradeoff
Compressing of an image

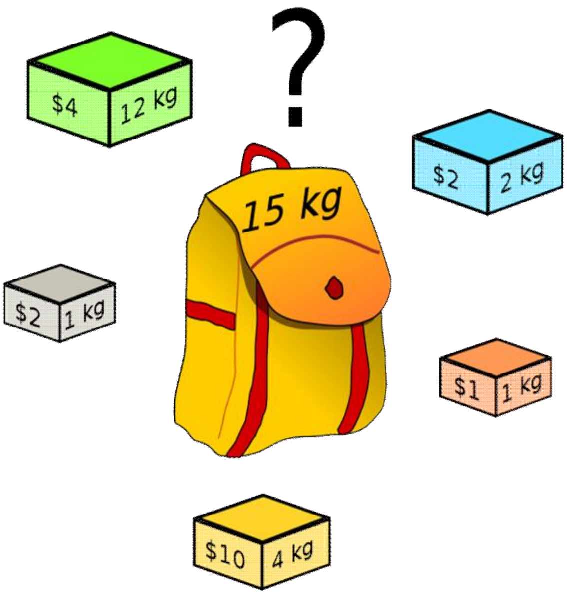
- Reduces transmission time/costs
- At the expense of CPU time
- Tradeoff between computation and communication

Tradeoff at Network Level



- Throughput is at conflict with fairness
- Tradeoff can be implemented through weighted scheduling

A child with Rs. 100 in a convenience store!
Handle it as a knapsack problem!



$$\text{maximize} \sum_{i=1}^{n} v_i x_i$$

$$\text{subject to} \sum_{i=1}^{n} w_i x_i \leq W \text{ and } x_i \in \{0,1\}.$$

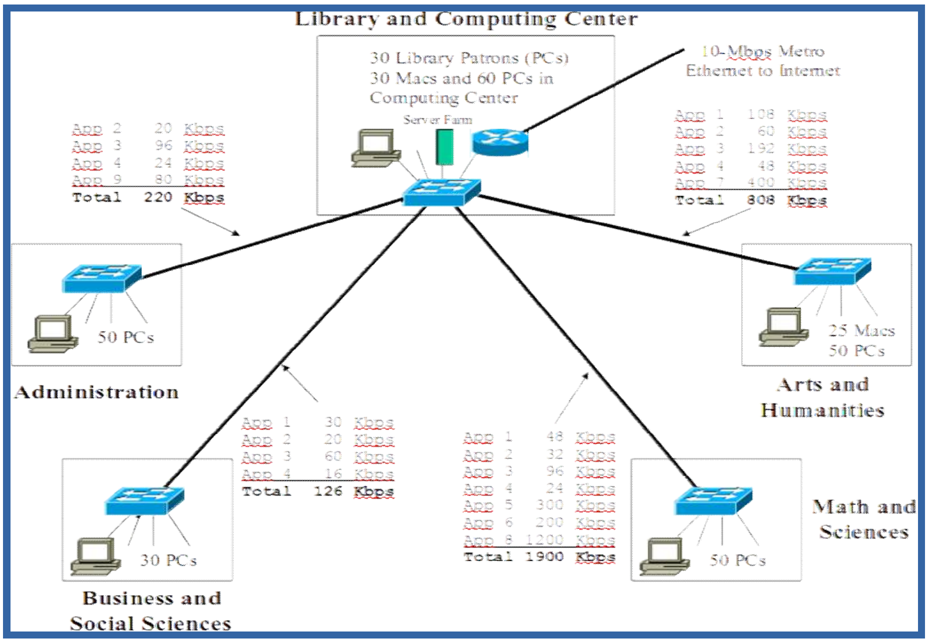
A child with Rs. 100 in a convenience store!

Scalability	20
Availability	30
Network performance	15
Security	5
Manageability	5
Usability	5
Adaptability	5
Affordability	15
Total	100
(Must add up to 100)	

$$\text{maximize} \sum_{i=1}^{n} v_i x_i$$

$$\text{subject to} \sum_{i=1}^{n} w_i x_i \leq W \text{ and } x_i \in \{0,1\}.$$

Problem Set 1



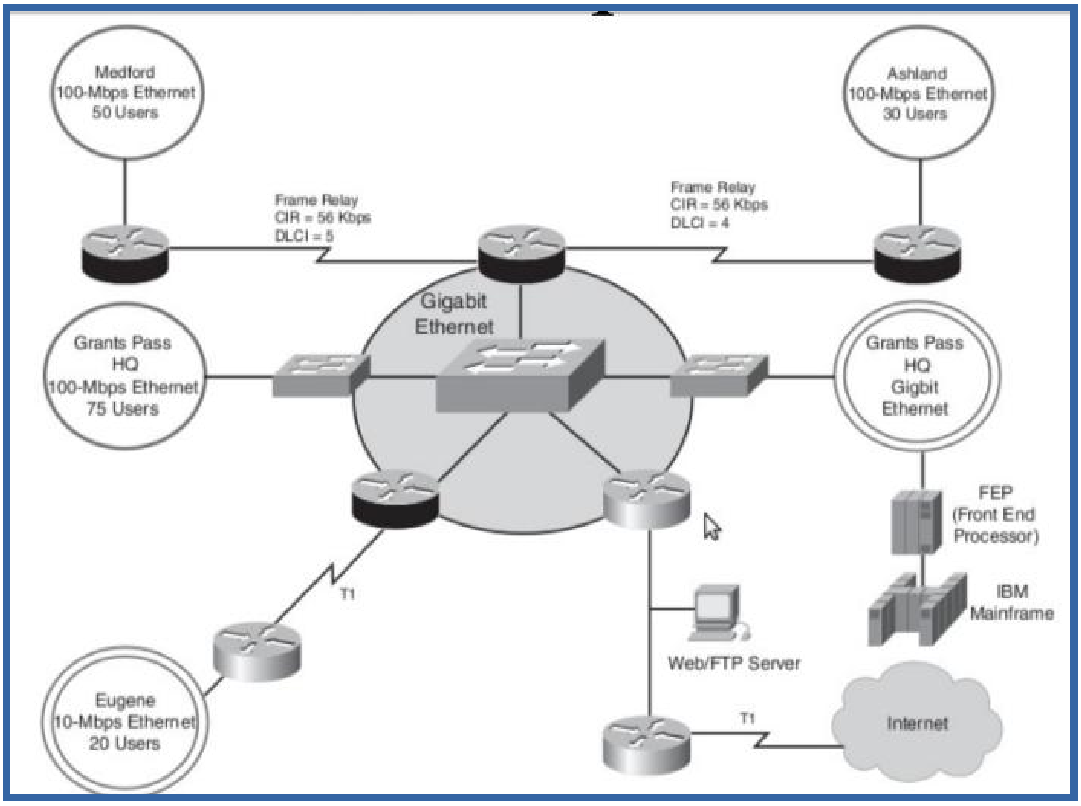
Effect of Topology Factors

1. What is the total data rate of the network?
2. What is the application that is generating the maximum load per user in Administration department?
3. What is the application that is generating the minimum load per user in Math and Science department?

Effect of Routing Protocols

1. If RIP sends a routing packet every 30 seconds and each packet contains 25 routes (Each route is 20B), what is the bit rate?

Problem Set 2

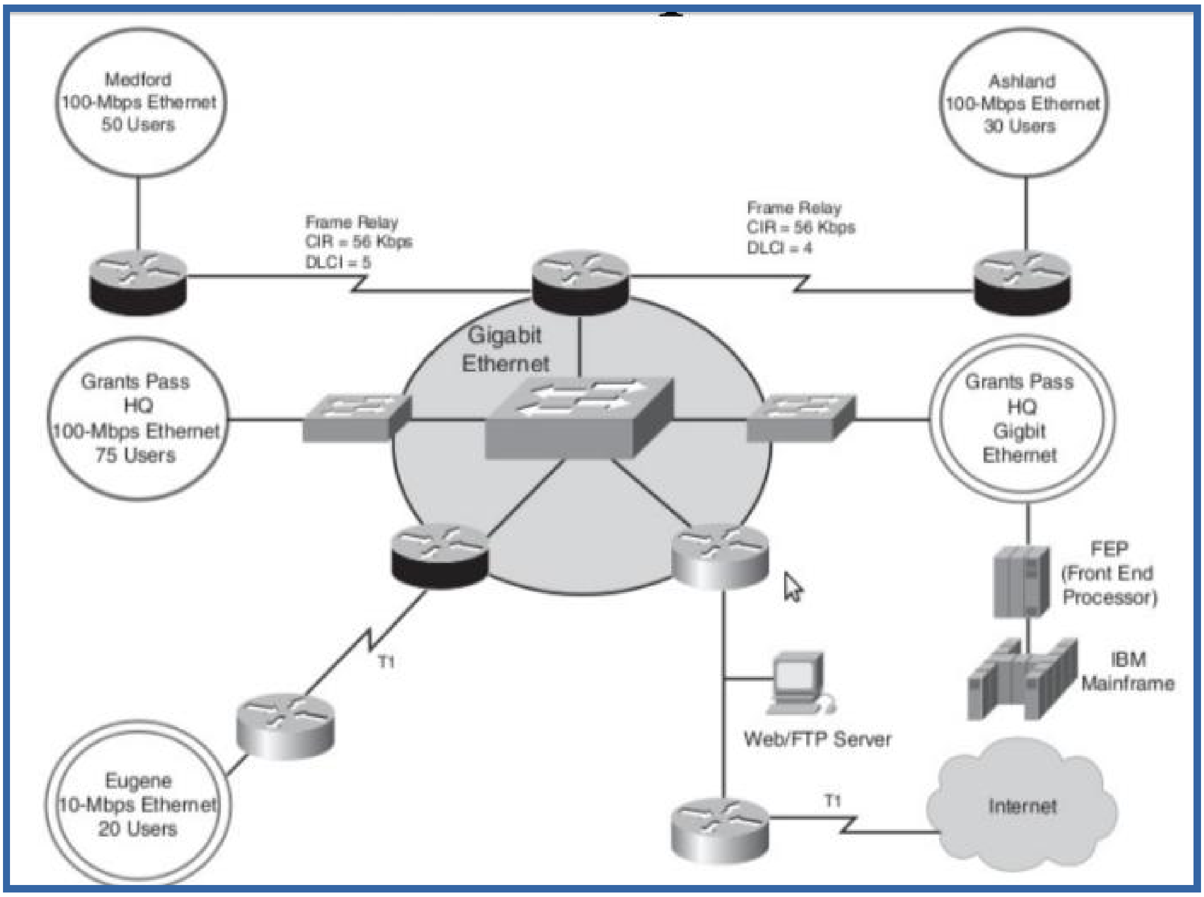


Effects of Deployment/Protocol Behaviors
1. Where is the data center?
2. What is the data rate available for users of Eugene?
3. What is the maximum Internet speed available to the users?
4. Label the router that needs to implement firewall.
5. If a user in Medford sends out a broadcast 255.255.255.255, what is the impact?

Queuing Behaviors
1. A CISCO switch has 20 users (clients and servers), each offering packets at a rate of 200 packets per second. If the average length of the packets is 64 Bytes, and the transmission rate of the switch is 10 Mbps measure the **load** of all the users and the LAN **utilization**. Then measure the **queue depth**

Understanding Network Design
1. Label the bastion host in the network.
2. Label the fastest end-to-end interoffice segment.
3. Label the slowest end-to-end interoffice segment.
4. How many total LAN segments are there?
5. Label at least one network where duplex auto-negotiation might help.
6. Label at least one segment where BERT can be used to measure BER.

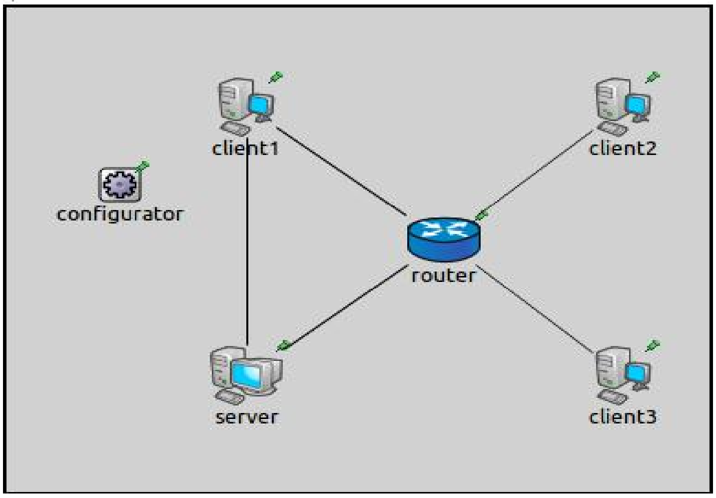
Simulate FTP Scenario
A Real World Scenario



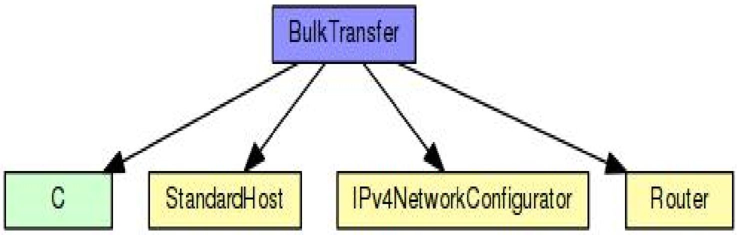
Factors affecting goodput

- End-to-end error rates
- Protocol functions (handshaking, windows, & acks)
- Protocol parameters (frame size, retx timers)
- pps rate of networking devices
- Lost packets at networking devices Workstation & server performance factors:
- Disk-access speed
- Disk-caching size
- Device driver performance
- Computer bus performance (capacity/arbitration)
- Processor (CPU) performance
- Memory performance (access time for real and virtual memory)
- Operating system inefficiencies
- Application inefficiencies or bugs

Implementation in INET
Source: https://omnetpp.org/doc/inet/api-current/neddoc/index.html
examples/inet/bulktransfer/BulkTransfer.ned



Usage diagram



Source: src/applications/tcpapp/TCPBasicClientApp.ned
numRequestsPerSession = exponential(3)
requestLength = truncnormal(20,5)
replyLength = exponential(1000000)

What to model?
1. Total time it takes to complete file transfer
2. Total goodput vs badput
3. Network utilization
4. Delay variation
5. Usability
6. Scalability
7. Availability

Parameters

Name	Type	Default value	Description
localAddress	*string*	""	may be left empty ("")
localPort	*int*	-1	port number to listen on
connectAddress	*string*	""	server address (may be symbolic)
connectPort	*int*	1000	port number to connect to
dataTransferMode	*string*	"bytecount"	
startTime	*double*	1s	time first session begins
stopTime	*double*	-1s	time of finishing sending, negative values mean forever
numRequestsPerSession	*int*	1	number of requests sent per session
requestLength	*int*	200B	length of a request
replyLength	*int*	1MiB	length of a reply
thinkTime	*double*		time gap between requests
idleInterval	*double*		time gap between sessions
reconnectInterval	*double*	30s	if connection breaks, waits this much before trying to reconnect

What to model?

Statistics:

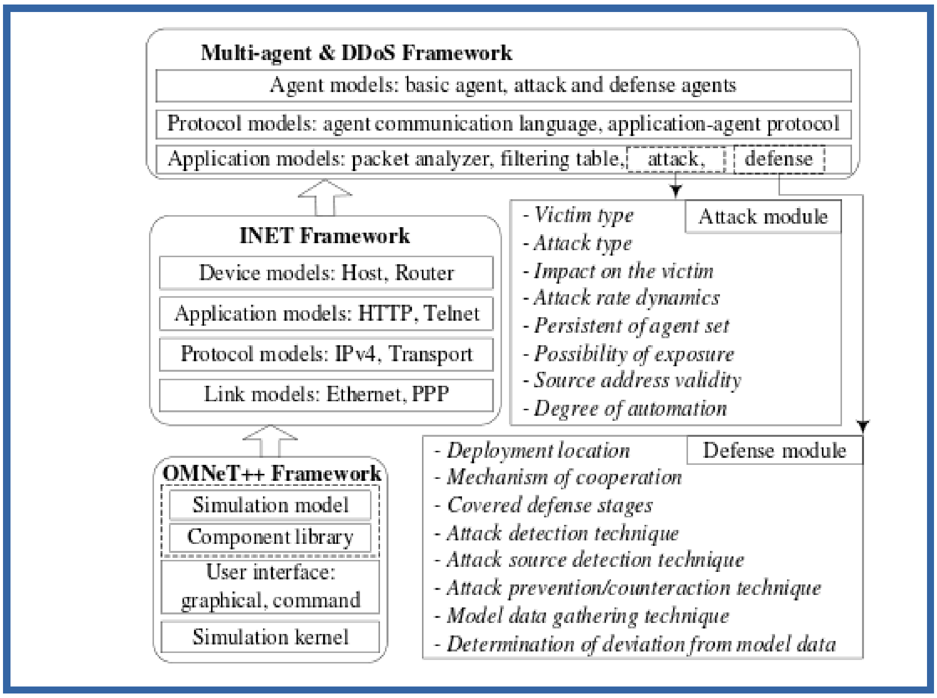
Name	Title	Source	Record	Unit	Interpolation Mode
numActiveSessions	number of active sessions	sum(connect)	max, timeavg, vector		sample-hold
sentPk	packets sent	sentPk	count, sum(packetBytes), vector(packetBytes)		none
endToEndDelay	end-to-end delay	messageAge(rcvdPk)	histogram, vector	s	none
rcvdPk	packets received	rcvdPk	count, sum(packetBytes), vector(packetBytes)		none
numSessions	total number of sessions	sum(connect+1)/2	last		

Simulating DoS Attack

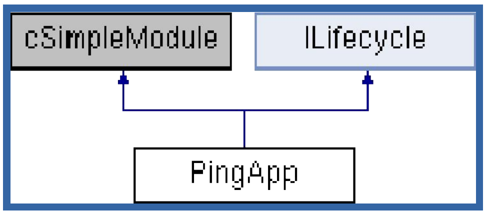
Igor Kotenko & Alexander Ulanov , "Simulation of Internet DDoS Attacks and Defense ," ISC 2006, LNCS 4176, pp. 327–342, 2006.

Kaur, Rupinderjit, Amrit Lal Sangal, and Kush Kumar. "Modeling and simulation of DDoS attack using Omnet++." Signal Processing and Integrated Networks (SPIN), 2014 International Conference on. IEEE, 2014.

What to model?

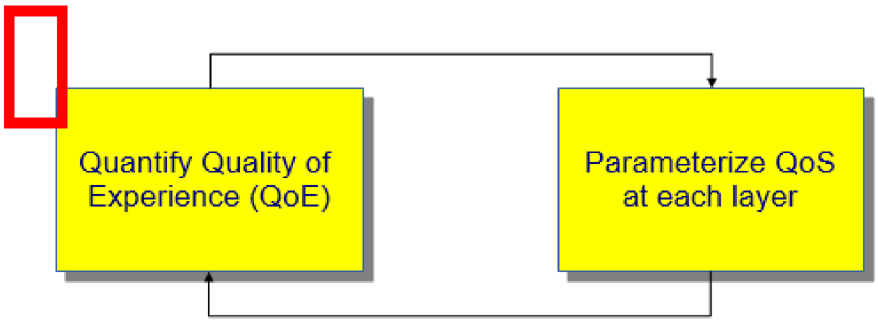


Configuring Ping of Death attack

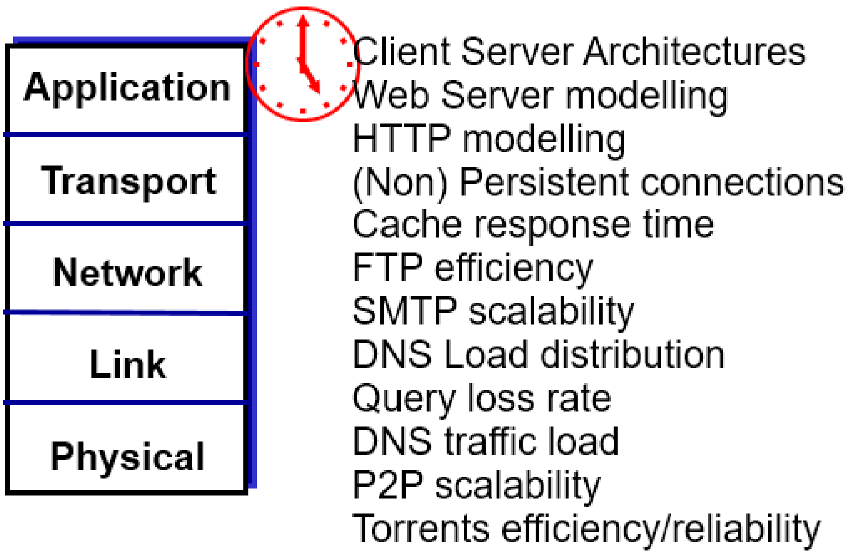


```
cSimpleModule::initialize();
packetSize = par("packetSize");
sendIntervalPar = &par("sendInterval");
hopLimit = par("hopLimit");
count = par("count");
startTime = par("startTime");
stopTime = par("stopTime");
```

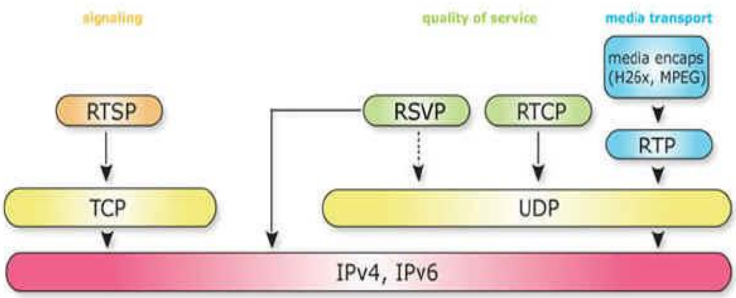
Summarizing top-down approach
Our Strategy



Application layer Roll-out for M&S



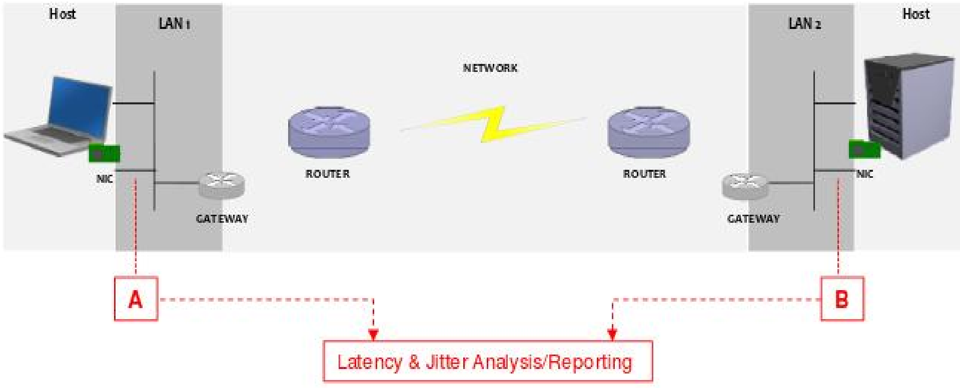
Simulate RTP with Packet Loss
Family of RTP



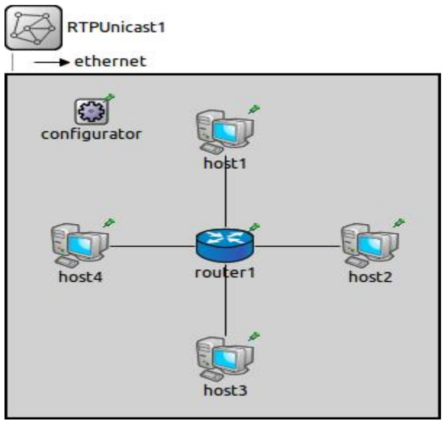
RTP

- Real-time Transport Protocol (RTP) is a network protocol
- Delivers audio/video over IP networks
- Streaming media
- Telephony
- Video teleconference
- Television service
- Push-to-talk over web

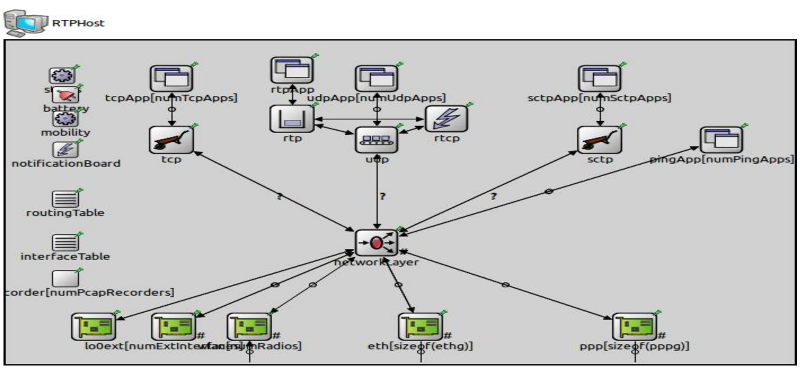
Delay/Jitter Analysis Points



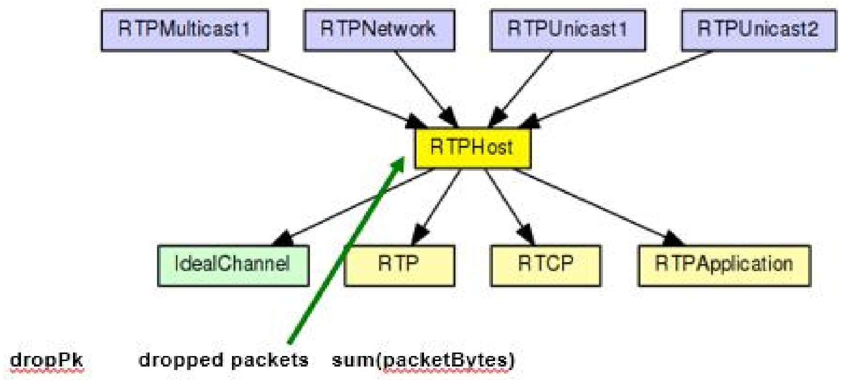
Inet for Simulating RTP (examples/rtp/unicast1/unicast1.ned)



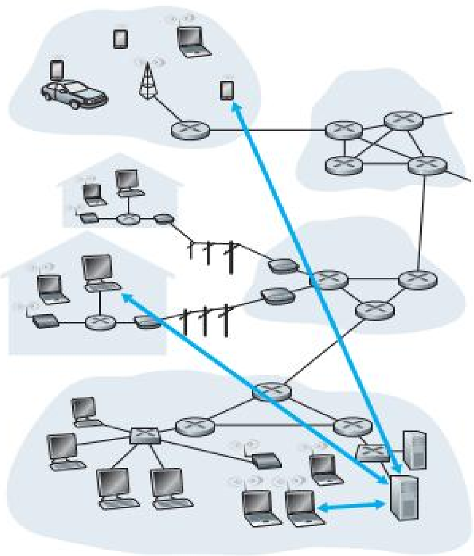
src/nodes/rtp/RTPHost.ned



Usage Diagram and Statistics



Client Server Architectures
An architecture for data exchange



Definition

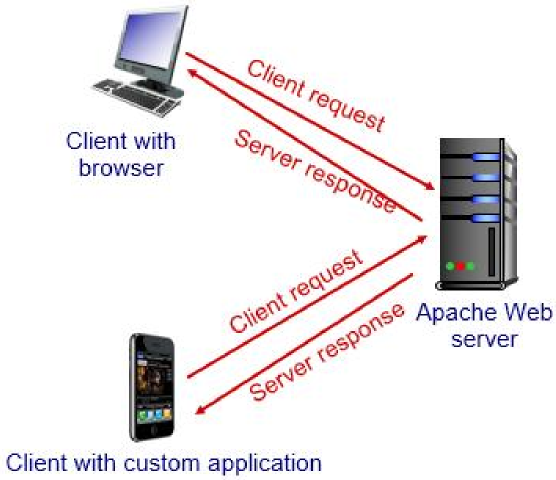
- One known server
- Always-on
- Permanent IP address
- Clients communicate with server
- Intermittently connected

Performance

$$D_{cs} \geq \max\left\{\frac{NF}{u_s}, \frac{F}{d_{min}}\right\}$$

- Distribution time for the client-server architecture denoted by D_{cs}
- Size of the file to be distributed (in bits) by F
- Number of peers that want to obtain a copy of the file is N
- d_{min} denotes the download rate of the peer with the lowest download rate
- Server upload rate is u_s

Web Server Modeling
Message Flow



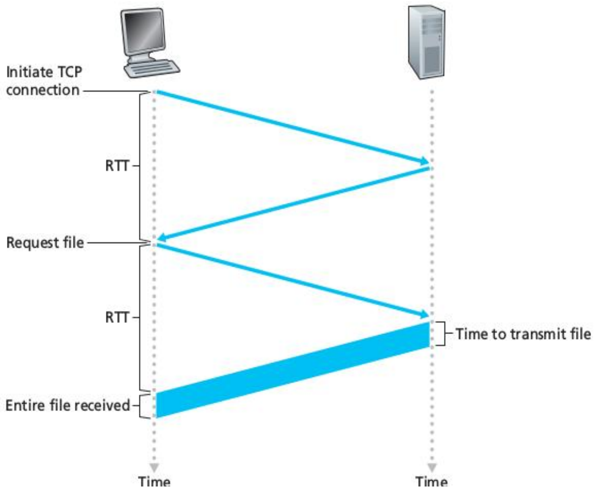
Operation

- Handles multiple HTTP requests
- Accepts and parses the HTTP request
- Gets the requested file from the server's file system
- Creates and sends an HTTP response message consisting of the requested file

Characterizing web server

- Buffer size per client
- Number of clients
- File size that it handles
- Processing time
- Time out interval

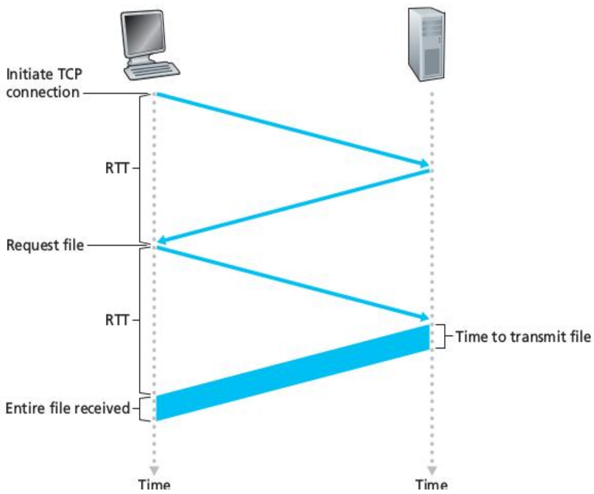
HTTP Modeling
Time line operation



Variants

- HTTP is based on sequenced messages
- Underlying TCP handshaking determines the overall performance
  - Persistent
  - Non-persistent
  - Pipelined
  - Caching

Non-Persistent Connections
TCP handshaking required for every object



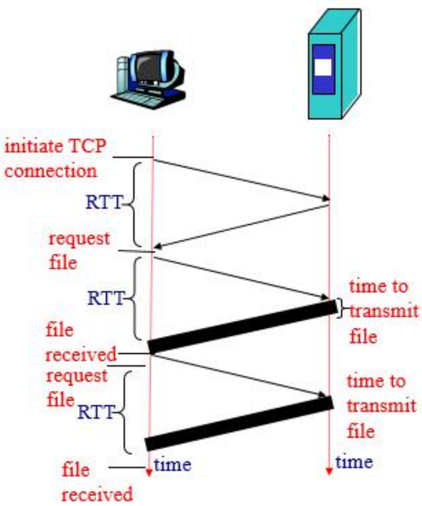
Modeling Non-persistence

- It requires 2 RTTs per object
- Total time for N objects

N*2RTT + N*Transmit time

- Consequent effect on simulated time is exacerbated in a multi-hop real world network

Persistent Connections
TCP handshaking required once



Modeling Persistence

- It requires 1 RTTs per object
- Total time for N objects

(N+1)*RTT + N*Transmit time

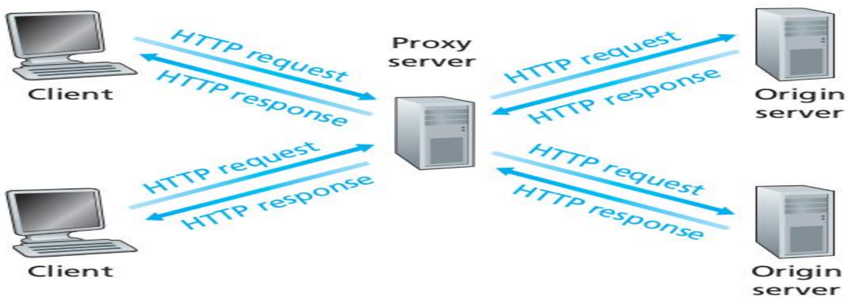
- Consequent effect on simulated time is noticed in a multi-hop real world network

Cache Response Time

Caching operation

- User sets browser: Web accesses via cache
- Browser sends all HTTP requests to cache
  - — Object in cache: cache returns object
  - — Else cache requests and returns object from origin server

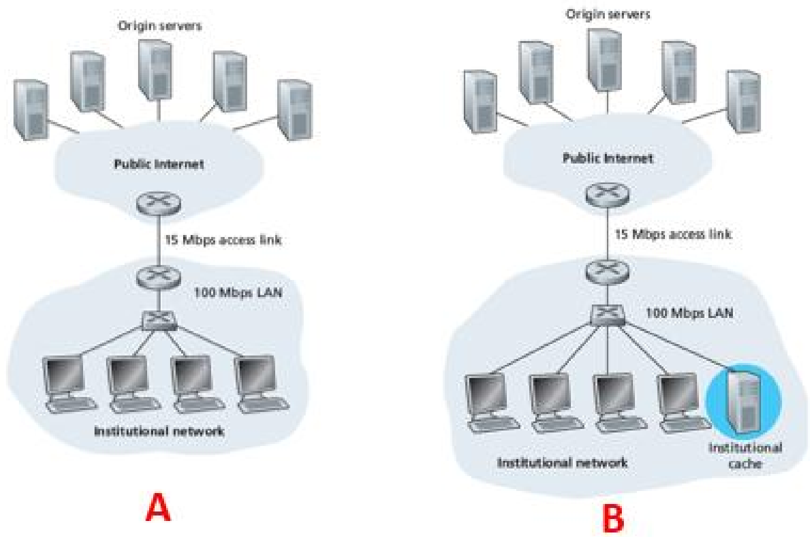
Clients requesting objects through cache



Advantages of caching

- Reduces response time for client request
- Reduce traffic on an institution's access link

Simulating Scenarios with and without cache



Factors affecting caching

- Average request rate from institution's browsers to servers
- Round trip delay from institutional router to server
- Correlation between requests
- Average object size

Example

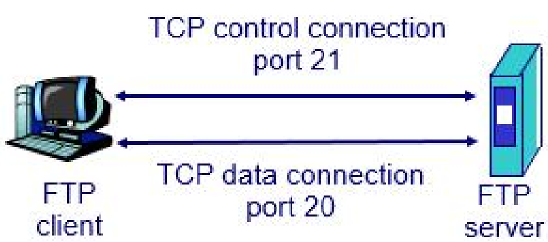
- Average object size = 100,000 bits
- Avg. request rate from institution's browsers to origin servers = 15/sec
- Delay from institutional router to any origin server and back to router = 2 sec Utilization on LAN = 15%
- Utilization on access link = 100%
- Total delay = Internet delay + access delay + LAN delay
  = 2 sec + minutes + milliseconds
- If hit rate is .4
- 40% satisfied locally
- 60% requests satisfied by server
- Utilization of access link reduced to 60% (say 10 ms)
- Avg delay = Internet + access + LAN
  = .6 * (2.01) s + ms < 1.4 secs

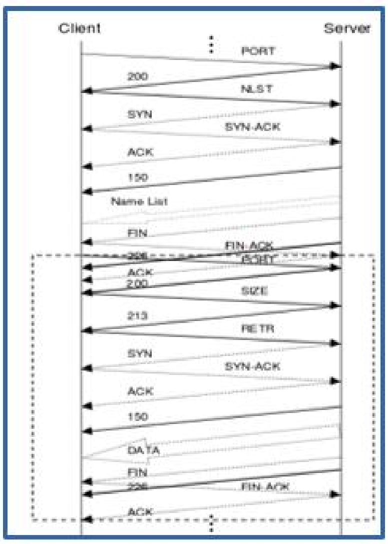
FTP Efficiency

FTP operation

- Client contacts FTP server at port 21
- Client obtains authorization
- Browses remote directory
- Server receives file transfer command
- Server opens TCP data connection to client
- After transfer connection closed

Control Signaling of FTP



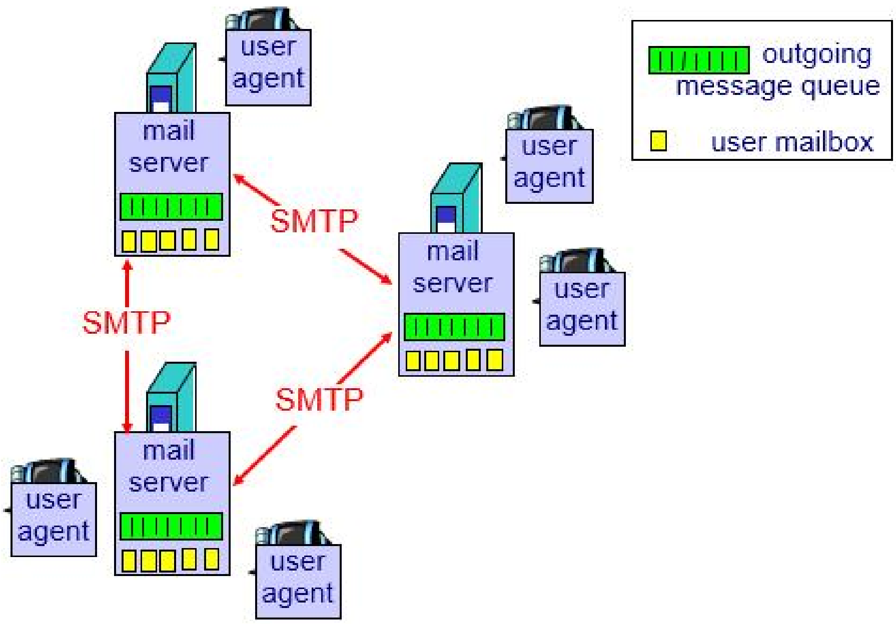


Computational Efficiency of FTP
(COURTESY: ALEBRA TECHNOLOGIES INC.)

$$\frac{((TCPU) - (ICPU)) \text{ X } MIPS}{TRATE} = \text{Millions of Instructions per Megabyte}$$

TCPU = Total CPU seconds recorded during the period of file transfer
ICPU = Measured CPU seconds when machine is idle for the equivalent period
MIPS = Machine performance rating in Millions of Instructions per second
TRATE = Transfer rate in megabytes per second

SMTP Scalability
Entities of SMTP Architecture



Recall scalability

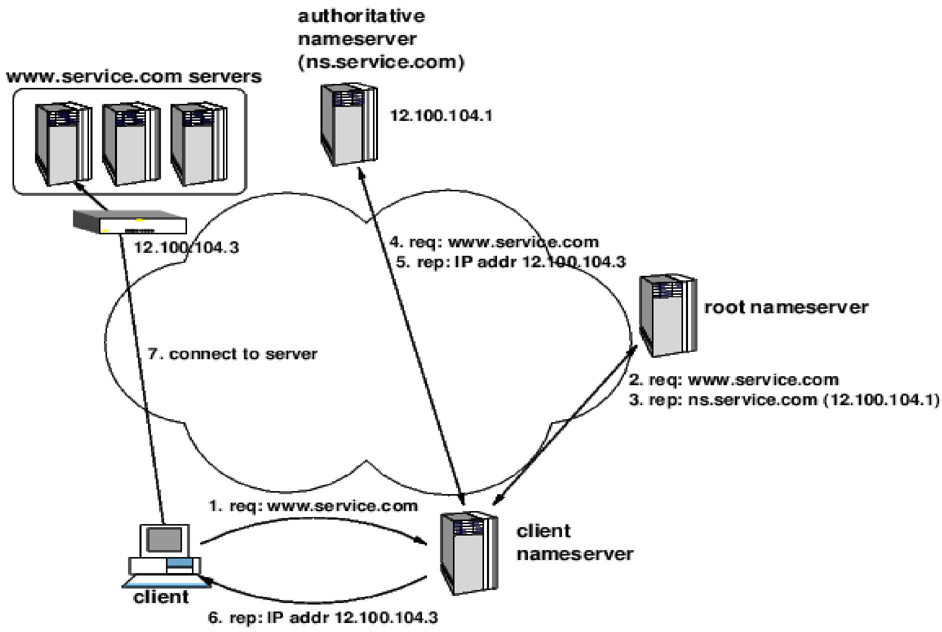
- Ability to grow
- Scaling may include
  - Number of user sites
  - Inter-site topology
  - No. of user agents
  - User mailbox size
  - No. of mail servers
  - Outgoing queue size

Efficiency & speed-up for SMTP
Mail delivery time tends to vary with scaling factors

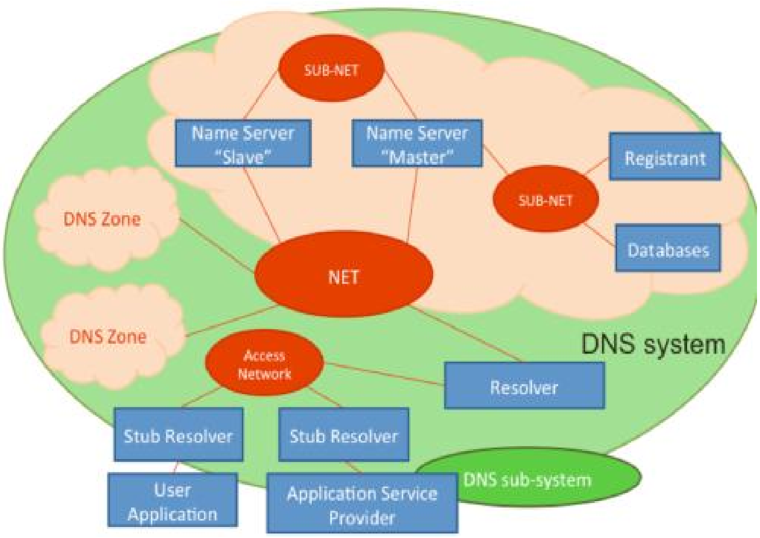
- Must be normalized when comparing SMTP performance at different traffic volumes
  - On single server
  - Servers confederation

$E_{Relative} = T_1$, (No. of hosts ´ $T_{No\ of\ hosts}$)
$S_{Relative}$ = No. of hosts ´ E_1

DNS Load Distribution & Loss
Typifying DNS operation



Casalicchio, E., Caselli, M., Coletta, A., & Fovino, I. N. Aggregation of DNS health indicators: issues, expectations and results



Health metrics
Incoming Bandwidth Consumption (IBC)

- Ratio between total amount of incoming data during a session over the duration of the session
- Range: [0, IBC max]
- measured in Mbit/s

$$\boxed{q(x) = 1 - \frac{x}{IBCMax}}$$

Health metrics

Incoming Traffic Variation (ITV)

- For each session *i*,

$(IBC_i - IBC_{i-1})/length_i$

- IBC_i is incoming bandwidth consumption in *ith* session
- $length_i$ is duration of that session

$$\boxed{q(x) = \begin{cases} e^{-2x/ITV_{max}} & x > 0 \\ 1 & x \leq 0 \end{cases}}$$

Traffic Tolerance (TT)

- Measures the Round Trip Time (RTT) of a IP packet flowing between end-user node and ISP's recursive resolver in seconds

$$\boxed{q(x) = \begin{cases} 1 & x \leq RTT_{avg} \\ -\frac{x}{RTT_{avg}} + 2 & RTT_{avg} \leq x \leq 2RTT_{avg} \\ 0 & x > 2RTT_{avg} \end{cases}}$$

DNS Requests per Seconds (DNSR)

- It gives the total number of DNS queries in the session

$$\boxed{q(x) = \begin{cases} 1 - \frac{x}{2 \cdot DNSR_{avg}} & 0 \leq x \leq 2 \cdot DNSR_{avg} \\ 0 & x > 2 \cdot DNSR_{avg} \end{cases}}$$

Rate of Repeated Queries (RRQ)

- In a single session a name is resolved only once due to caching
- The metric returns no. of repeated DNS queries in a session for same name if the query is lost
  - Or not cached

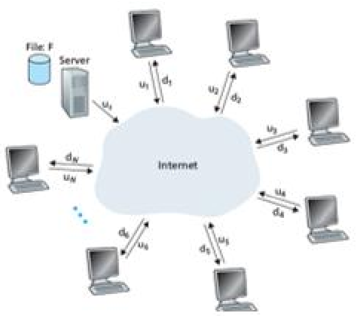
$$\boxed{q(x) = 1 - \frac{x}{R_{max}}}$$

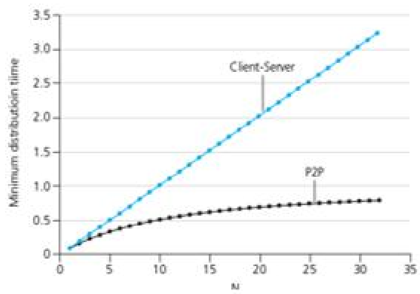
Peer to Peer Scalability

Operation

- *No* always-on server
- Arbitrary end systems directly communicate
- peers are intermittently connected
- Change IP addresses

File Distribution Problem





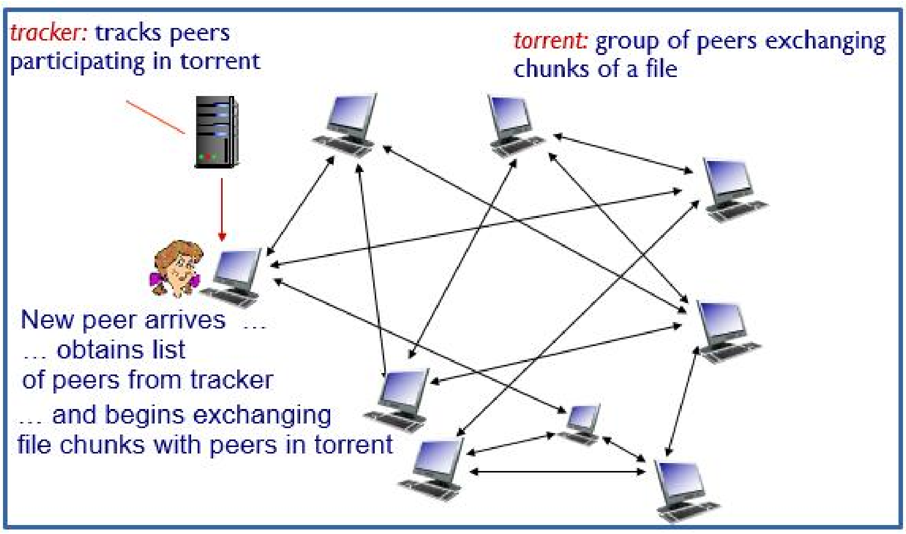
$$D_{cs} \geq \max\left\{\frac{NF}{u_s}, \frac{F}{d_{min}}\right\} \Rightarrow D_{\mathrm{P2P}} = \max\left\{\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_s + \sum_{i=1}^{N} u_i}\right\}$$

Performance

$$D_{\mathrm{P2P}} = \max\left\{\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_s + \sum_{i=1}^{N} u_i}\right\}$$

- Distribution time for the P2P architecture denoted by D_{P2P}
- Size of the file to be distributed (in bits) by F
- Number of peers that want to obtain a copy of the file is N
- d_{min} denotes the download rate of the peer with the lowest download rate
- Upload capacity of the system as a whole = the upload rate of the server **plus** the upload rates of each of the individual peers, that is, $u_{total} = u_s + u_1 + ... + u_N$
- Server upload rate is u_s

Torrents Efficiency
Basic Torrent Operation



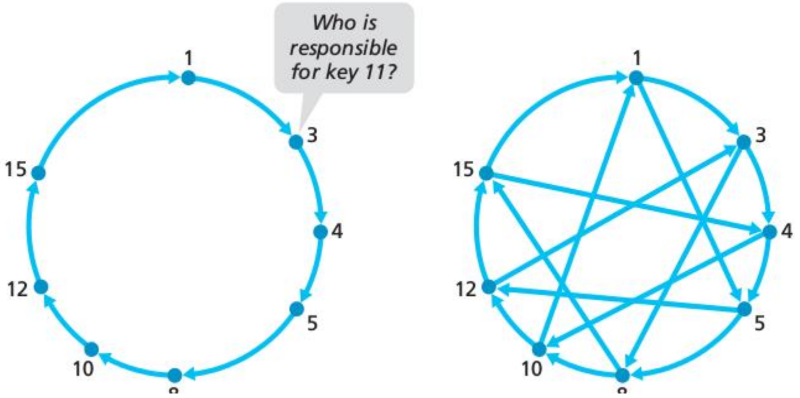
Factors affecting efficiency

- Heterogeneous upload capacity
- Diversities of neighbor selecting mechanisms
- Geographical distribution of peers
- Downloading rates of LocalBT clients
- Peer selection policy

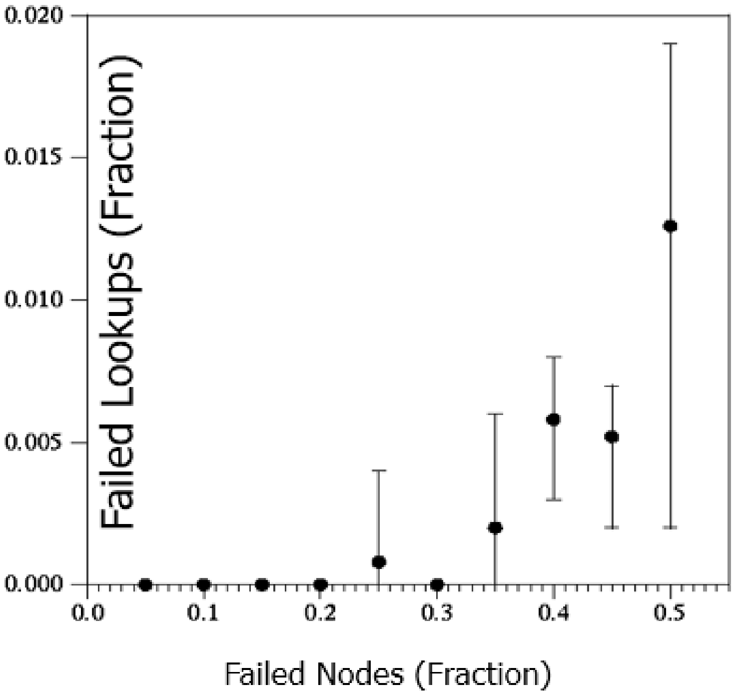
Performance

$$\boxed{\text{Efficiency of BitTorrent} = (T_{BitTorrent} - T_{CSFD})/T_{CSFD}}$$

Reliability of Circular DHT
Operation of Circular DHT



REDUNDANCY HANDLES FAILURES



1000 DHT nodes
Average of 5 runs
6 replicas for each key
Kill fraction of nodes
Measure how many lookups fail
All replicas must be killed for lookup to fail

Cost of Reliability

$$C = \frac{l}{T_{ls}} + \frac{2 \times \sum_{r=0}^{\frac{128}{b}} ((2^b - 1) \times (1 - b(0; N, \frac{1}{(2^b)^{(r+1)}})))}{T_{rt}}$$

l leaf-set keepalive messages every *T* seconds
2-messages for probe and response Routing table probes every T_{rt}
Summation computes expected number of routing table entries (128/*b* rows and 2^b columns)

- Last expression is a binomial distribution

Problem Set 1
Network Latencies

Consider an institutional network connected to the Internet. Suppose that the average object size is **850,000 bits** and that the average request rate from the institution's browsers to the origin servers is **16 requests per second**. Also suppose that the amount of time it takes from when the router on the Internet side of the access link forwards an HTTP request until it receives the response is **three seconds** on average.

Model the **total average response time** as the sum of the **average access delay** (that is, the delay from Internet router to institution router) and the **average Internet delay**. For the average access delay, use **Δ/(1 – Δb)**, where **Δ** is the average time required to send an object over the access link and **b** is the arrival rate of objects to the access link.

Now suppose a cache is installed in the institutional LAN. Suppose the miss rate is 0.4. Find the total response time.

HTTP Performance
Suppose that an HTML file on a web server references **eight (8)** very small objects. Neglecting transmission times, how much time it takes when non-persistent HTTP connection is used and the browser is configured for **five (5)** parallel connections?

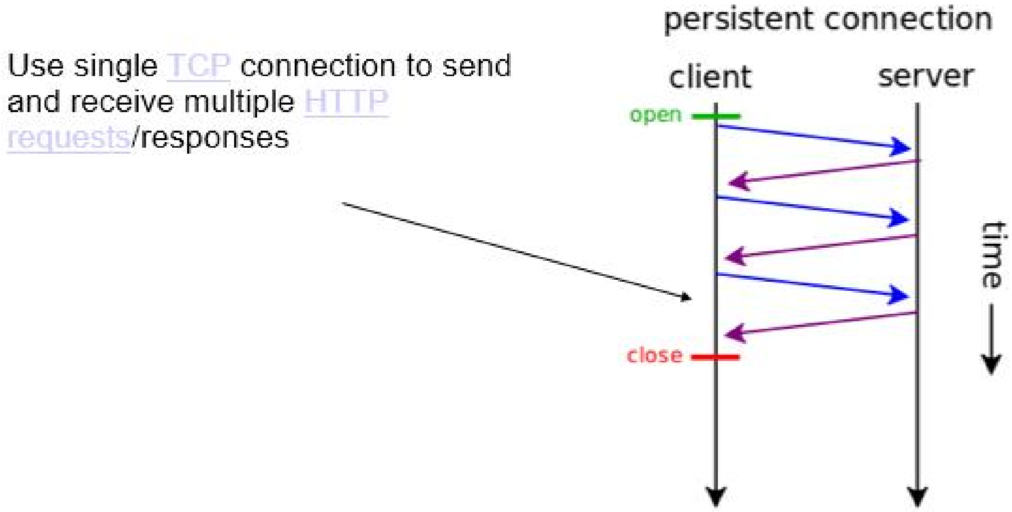
A. 18RTT B. 6RTT
C. 3RTT D. None of these

Problem Set 2
P2P Protocols

Suppose that **peer 3** learns that **peer 5** has left. How does peer 3 update its **successor** state information?
A. It asks peer 4 B. It asks peer 8
C. It asks peer 2 D. None

User Activity Monitoring
For a **1 Mbps** link, if each user generating **200 kbps** is active for **20%** of the time, what is the probability that out of a total of **100** users, more than **5** users be active?

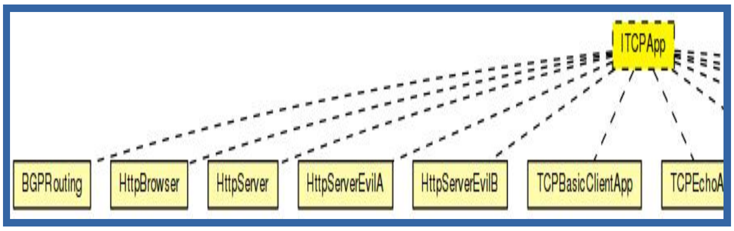
Simulate HTTP Persistence



HTTP Evolution

- RFC 793 does not support persistence
  - HTTP1.0
- Additional mechanism needed
  - Use keep-alive
- HTTP 1.1 is persistent by default

HTTP Support in OMNET++
Module Interface ITCPApp



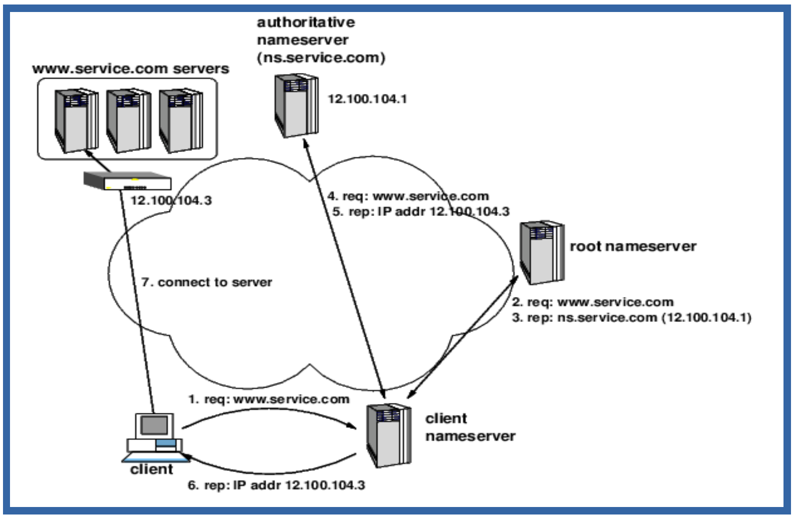
- Template for TCP applications (Inheritance)
- It shows what gates a TCP app needs
- to be able to be used in StandardHost etc

HTTP Browser in OMNET++
src/applications/httptools/HttpBrowser.ned
Default support is HTTP 1.1
simple HttpBrowser like ITCPApp
{ parameters:
int httpProtocol = default(11); }
Supported Modes

- Random request mode
- Browser uses statistical distributions generate requests to random web servers

- Scripted mode
- Browsing behavior determined by a list of predefined web sites to visit at specific times

Simulate DNS Query Response
Basic Operation



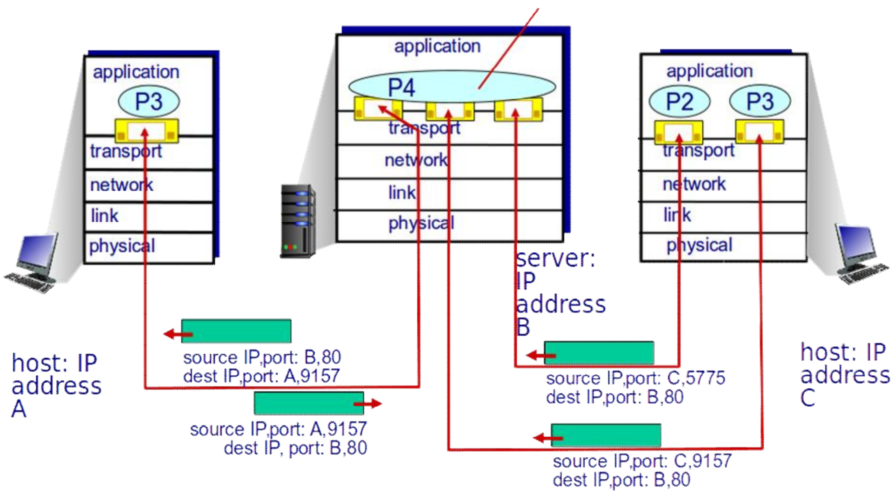
DNS Support in OMNET++

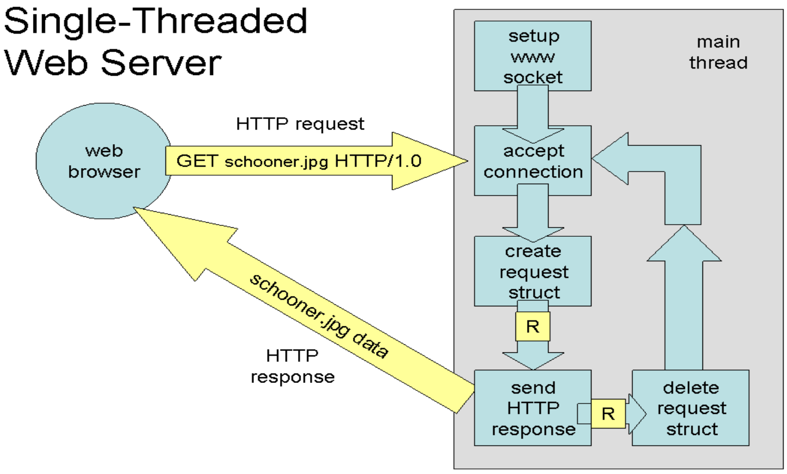
- Extensions provide classes and functions to simulate DNS and MDNS traffic
- Implement RFC 1035

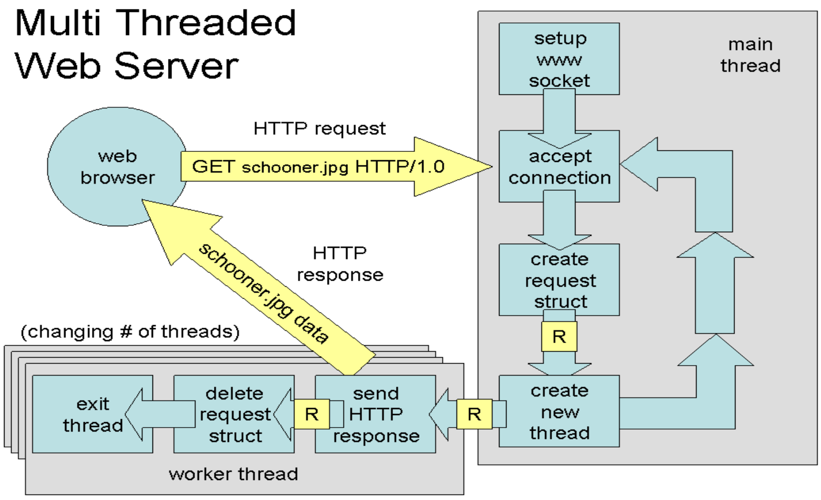
Supported DNS Operations

- Name servers with recursive resolving capabilities
- Authoritative servers with DNS zone configuration using master files
- Caching servers without zones
- Only recursively resolving
- DNS Cache base that can be extended
- Caches based on different policies possible
- DNS client that can query a DNS server

Simulate TCP Threading
Threaded Server







INET Support for TCP

- RFC 793 - Transmission Control Protocol
- RFC 896 - Congestion Control in IP/TCP Internetworks
- RFC 1122 - Requirements for Internet Hosts -- Communication Layers
- RFC 1323 - TCP Extensions for High Performance
- RFC 2018 - TCP Selective Acknowledgment Options
- RFC 2581 - TCP Congestion Control
- RFC 2883 - An Extension to the Selective Acknowledgement (SACK) Option for TCP

Features

- RFC 793 TCP states and state transitions
- Connection setup and teardown as in RFC 793
- Segment processing
- Receive buffer to cache above-sequence data
- Data not yet forwarded

Simulate HTTP Handshaking
HTTP Requests

HTTP/1.0:

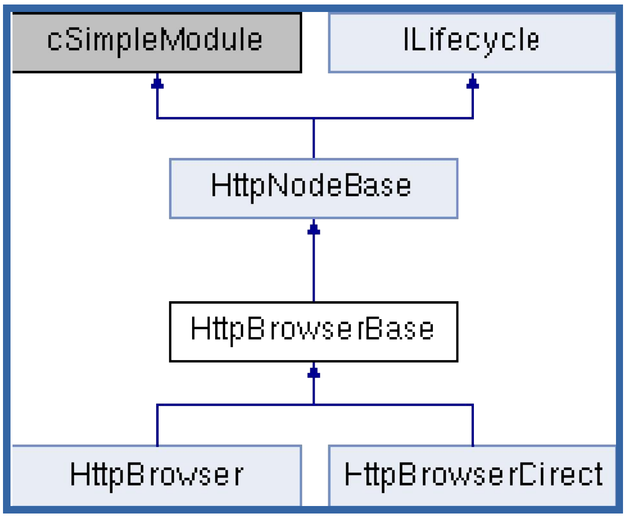
- GET
- POST
- HEAD
- asks server to leave requested object out of response

HTTP/1.1:

- GET, POST, HEAD
- PUT
- uploads file in entity body to path specified in URL field
- DELETE
- deletes file specified in the URL field

HTTP Response
200 OK
request succeeded, requested object later in this msg
301 Moved Permanently
requested object moved, new location specified later in this msg (Location:)
400 Bad Request
request msg not understood by server
404 Not Found
requested document not found on this server
505 HTTP Version Not Supported

HttpBrowser Class Reference
(Inheritance Diagram)



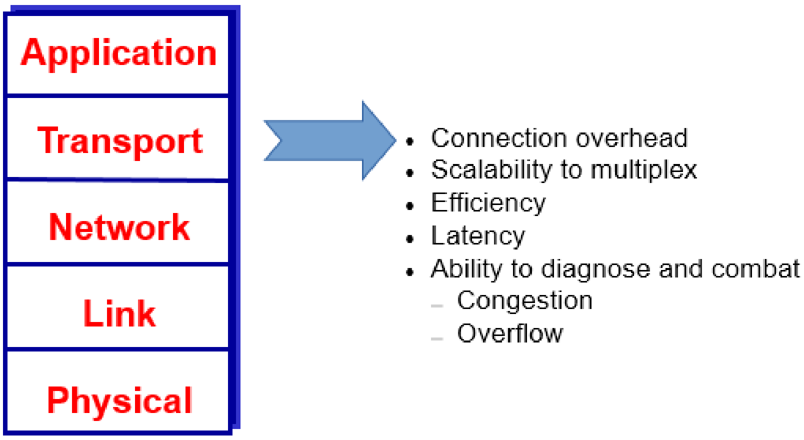
Intro & Transport Services
Introduction

- Transport layer is the big brother
- Manages end to end delivery of data
- Modeling of transport layer is pivotal to the overall performance

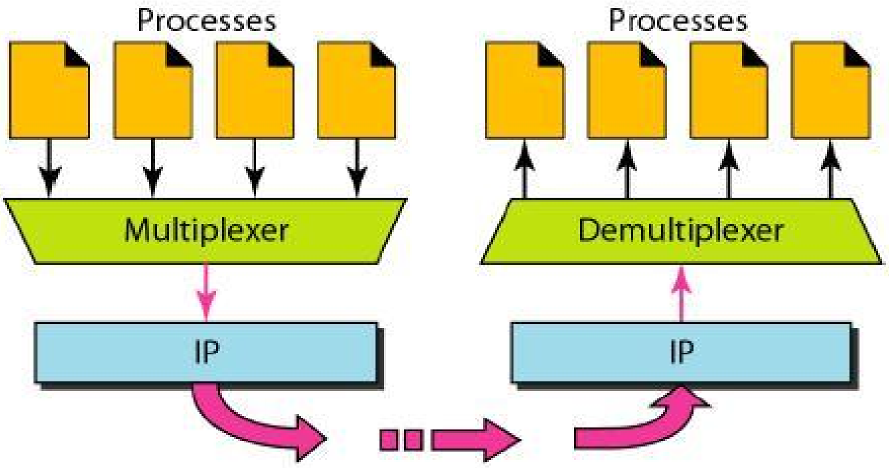
Transport Services

- Multiplexing and demultiplexing
- Reliable, in-order delivery (TCP)
- Congestion control
- Flow control
- Connection setup
- Unreliable, unordered delivery: UDP
- “best-effort” IP
- Services not available
- Delay guarantees
- Bandwidth guarantees

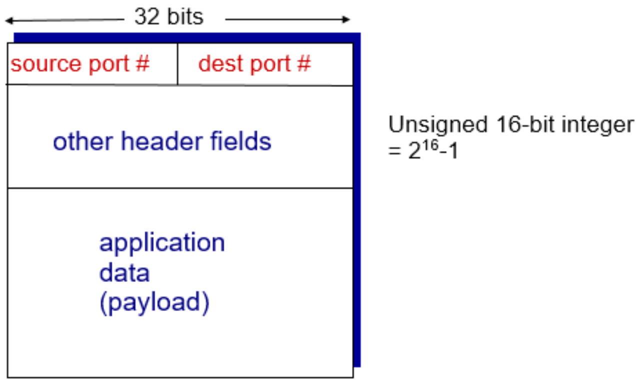
Modeling Approach



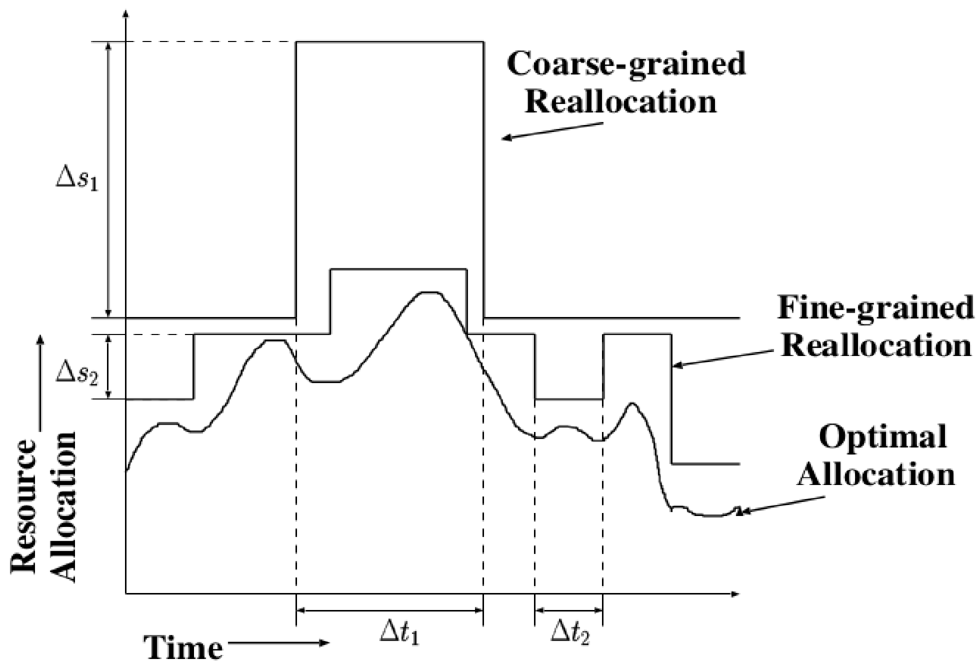
Multiplexing & Demultiplexing Basics



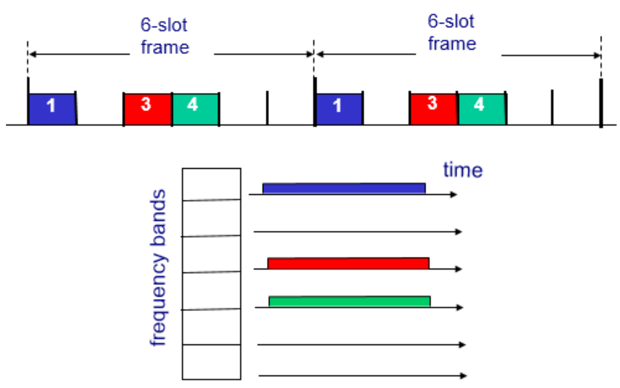
Capability of Port #



Cost of Multiplexing



Multiplexing Communication Link
Typical Multiplexing Techniques



Capacity Overhead

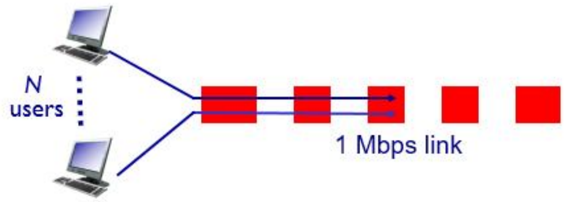
$$\rho = \left(\frac{R_{pract} - R_{opt}}{R_{opt}} \right) \cdot 100;$$

Capacity overhead ρ is defined as %age increase in the resource requirement of a practical multiplexing scheme when compared to the optimal

$$R^i_{opt}(t) \;\; R^i_{pract}(t)$$

Amount of resources allocated to application i at time t using the optimal and practical allocation scheme respectively

Gain in Statistical (Packet) Multiplexing



- Each user: 100 kb/s when "active"
- active 10% of time
- Strict Multiplexing: 10 users
- Statistical Multiplexing: with 35 users, probability > 10 active at same time is less than .0004

Checksum
Introduction

- Transport layer incorporates error detection
- Checksum is "checking the sum" both at the sender and receiver
- Performed at the header or the entire body

Operation in Brief & Performance

		1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
		1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
wraparound	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1
sum		1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	0
checksum		0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1

Overhead and Operational cost

- Divide the M-bit data into N-bit chunks
  - Total chunks M/N
- Checksum is also N-bit
- Total sums $M/N + 1$

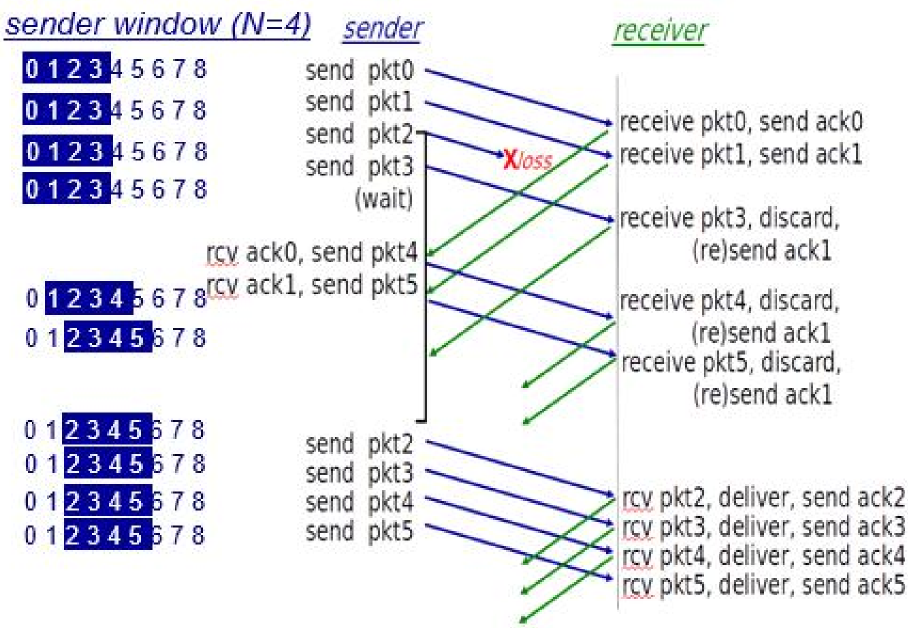
Undetected Errors

- Reordering of 2 byte words, i.e. 01 02 03 04 changes to 03 04 01 02
- Inserting zero-valued bytes i.e. 01 02 03 04 changes to 01 02 00 00 03 04
- Deleting zero-valued bytes i.e. 01 02 00 00 03 04 changes to 01 02 03 04
- Replacing a string of sixteen 0's with 1's or 1' with 0's
- Multiple errors which sum to zero, i.e. 01 02 03 04 changes to 01 03 03 03

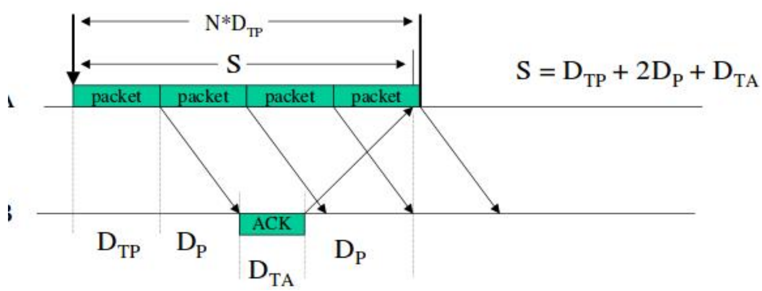
Go Back N
Introduction

- Retransmission strategy (ARQ)
- No need to buffer at receiver
- Wheat and rice analogy!
  - Go back N is wheat
  - Fresher is better

Performance Amidst Packet Loss



Efficiency without Errors



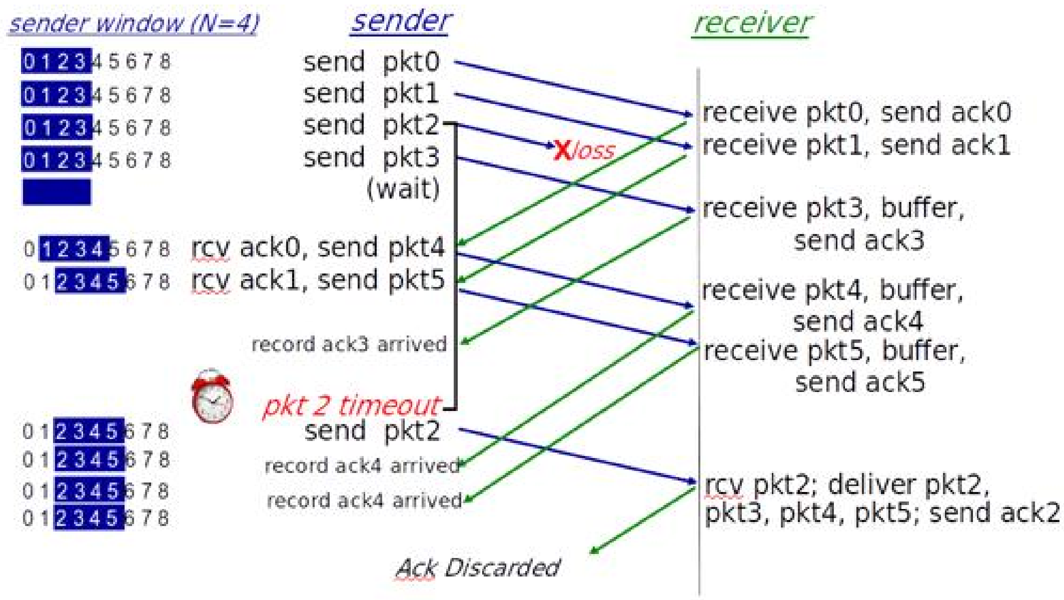
- Choose *N* large enough to allow continuous transmission while waiting for an ACK for the first packet of the window

$$If\ N > S/D_{TP} \qquad E = min\{1, N^*D_{TP}/S\}$$

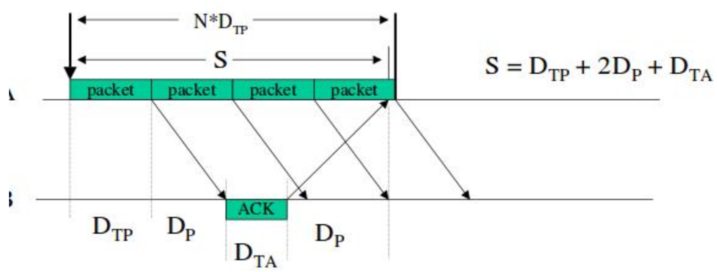
Selective Repeat
Introduction

- Retransmission strategy (ARQ)
- No need to retransmit all after loss
- Buffer requirements at receiver
- Wheat and rice analogy!
  - Selective repeat is wheat
  - Older is better

Performance Amidst Packet Loss



Efficiency without Errors



- Same as Go Back *N*
- *If* $N > S/D_{TP}$ $\quad E = min\{1, N*D_{TP}/S\}$

Efficiency with Errors

- Only packets containing errors will be retransmitted

$$E = 1 - P$$

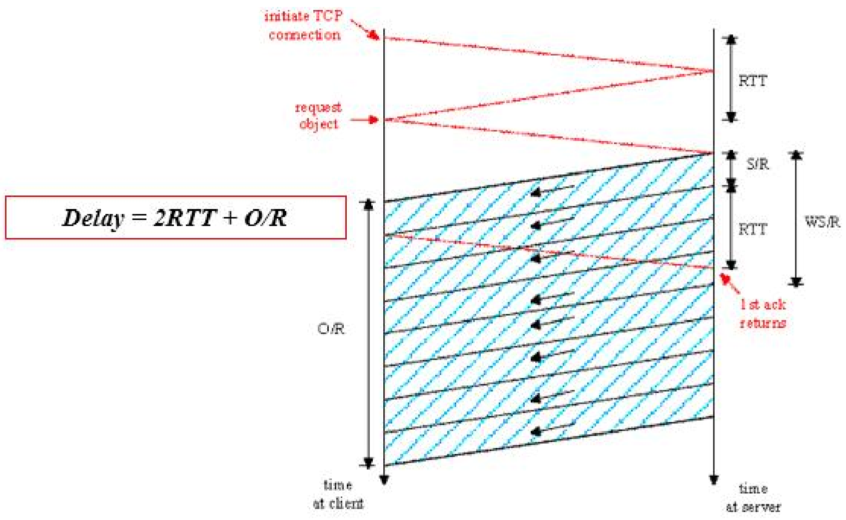
Implications of buffer size

- Buffer limit at sender
  - Number of un-ACKed packets at sender =< W
- Buffer limit at receiver
  - Number of un-ACKed packets at sender cannot differ by more than W

RTT Estimation and Timeout
Fixed Window
First case
WS/R > RTT + S/R

- ACK for first segment in window returns before window's worth of data sent

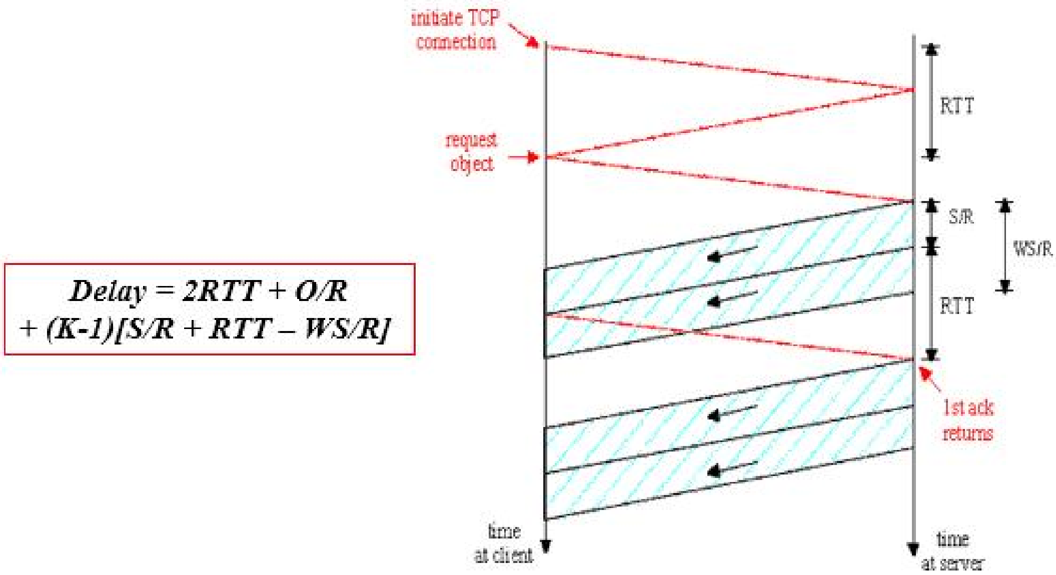
Delay Performance with Fixed Window



Fixed Window
Second case
WS/R < RTT + S/R

- Wait for ACK after sending window's worth of data sent
- *K* is the number of windows that cover the object

Delay Performance with Fixed Window



TCP Timeout Value

- Longer than RTT
- As RTT varies
    - — *Too short:* premature timeout,
    - — Unnecessary retransmissions
    - — *Too long:* slow reaction to segment loss

Estimating RTT

- **SampleRTT** Measured time from segment transmission until ACK receipt
- Ignores retransmissions
- **EstimatedRTT** is “smoother”
- Averages several recent measurements, not just current SampleRTT

Relationship Between TimeOut and Estimated RTT

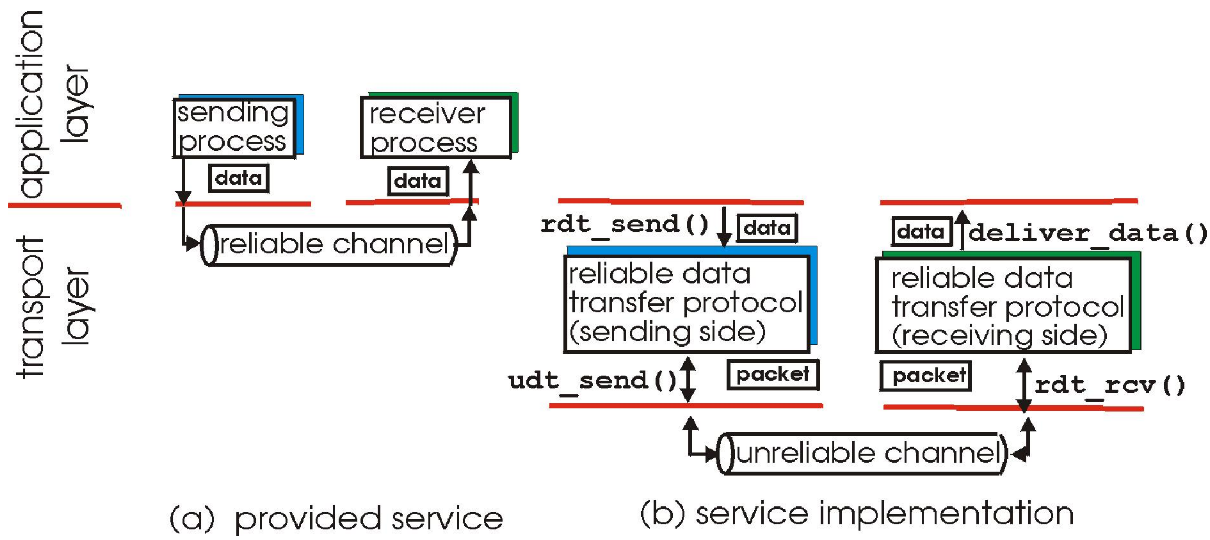
$$EstimatedRTT = (1- a)*EstimatedRTT + a*SampleRTT$$
$$TimeoutInterval = EstimatedRTT + 4*DevRTT$$

Reliable Data Transfer

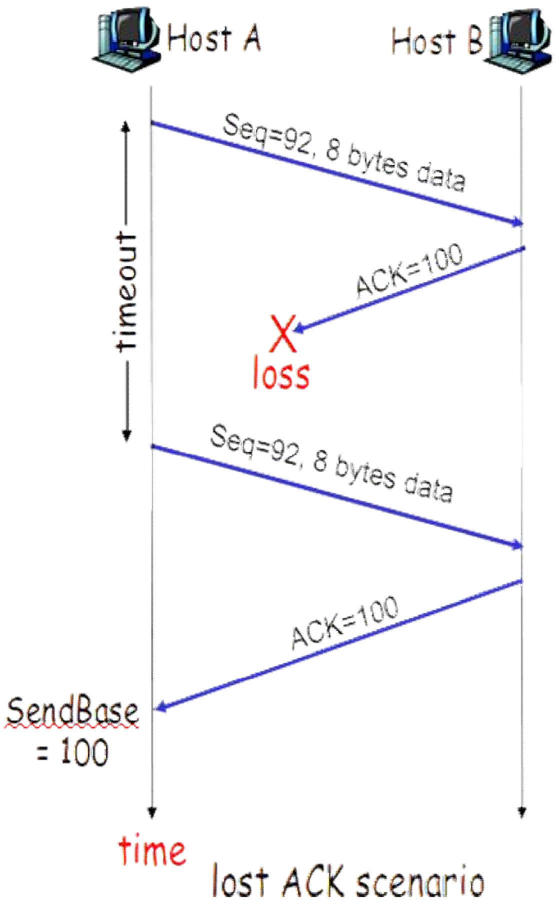
- TCP offers data reliability
- In case data is lost or corrupted
    - — Error Control
    - — Flow control
    - — Congestion control
- Reliability at the cost of throughput
- With slow start and FRFR, throughput is given by

$$\frac{1.22 \cdot MSS}{RTT\sqrt{L}}$$

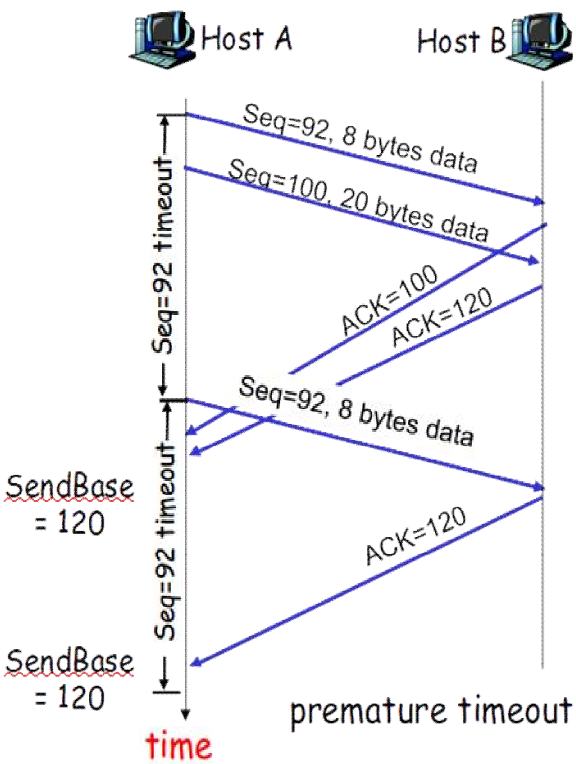
Reliability Services



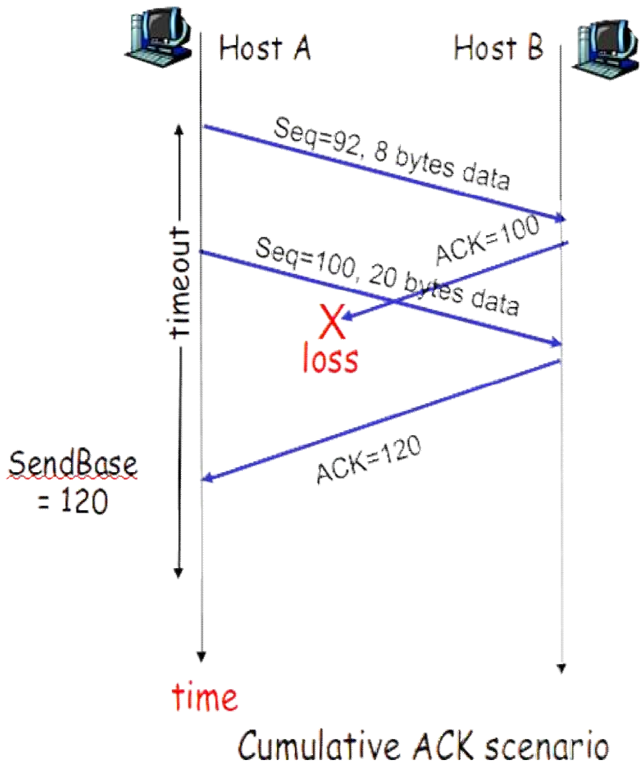
Handling Loss



Early TimeOut



Delayed Ack



Flow Control
Introduction

- Receiver throttles the sender by advertising a window
  - Not larger than the amount of data that it can buffer
- TCP on the receive side must keep

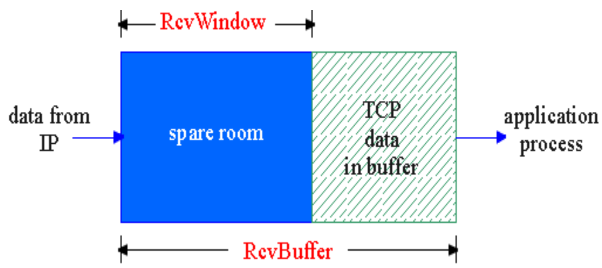
LastByteRcvd − LastByteRead ≤ MaxRcvBuffer

Implication

- If local process reads data just as fast as it arrives
- Causes LastByteRead to be incremented at the same rate as LastByteRcvd
- Advertised window stays open
  (AdvertisedWindow = MaxRcvBuffer)
- If receiving process falls behind, advertised window grows smaller with every segment that arrives, until it eventually goes to 0

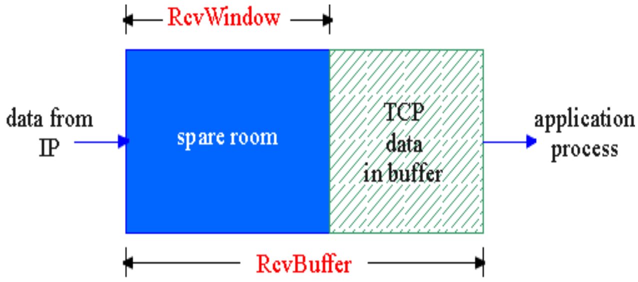
Advertised Window

AdvertisedWindow = MaxRcvBuffer – ((NextByteExpected – 1) – LastByteRead)



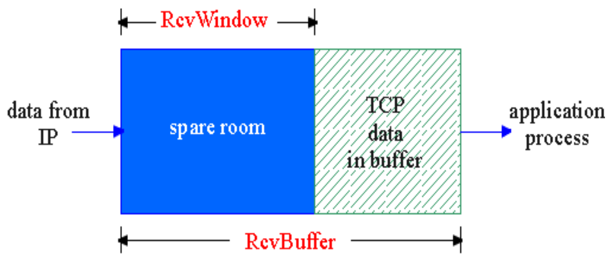
Sender Window

$$LastByteSent - LastByteAcked \le AdvertisedWindow$$



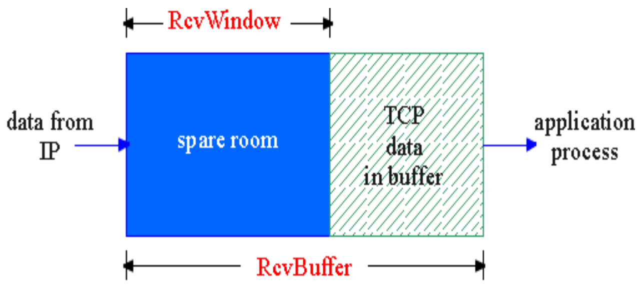
Effective Window

$$EffectiveWindow = AdvertisedWindow - (LastByteSent - LastByteAcked)$$



Relationship between Max_Send and Max_Receive Buffer

$$LastByteWritten - LastByteAcked \le MaxSendBuffer$$

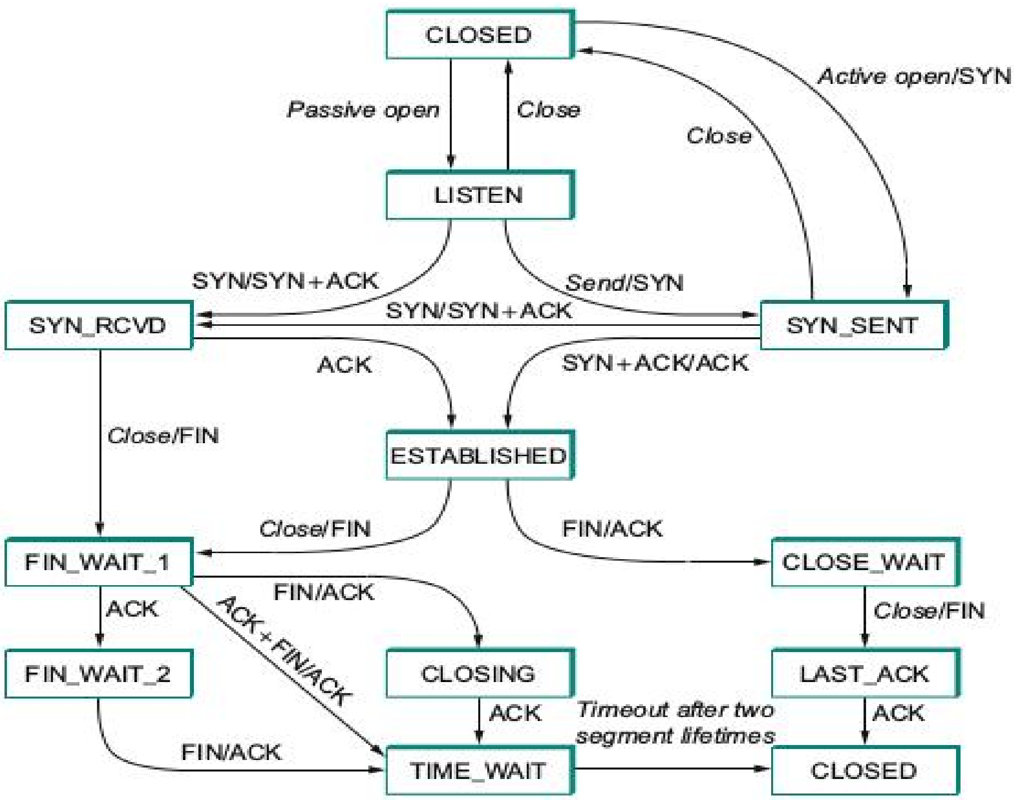


TCP Connection Management
Cost and Feasibility Model
Cohen, Edith, Haim Kaplan, and Jeffrey Oldham. "Managing TCP connections under persistent HTTP." Computer Networks 31.11 (1999): 1709-1723.
Kurose and Ross. "Computer Networking Top-Down Approach Featuring the Internet".

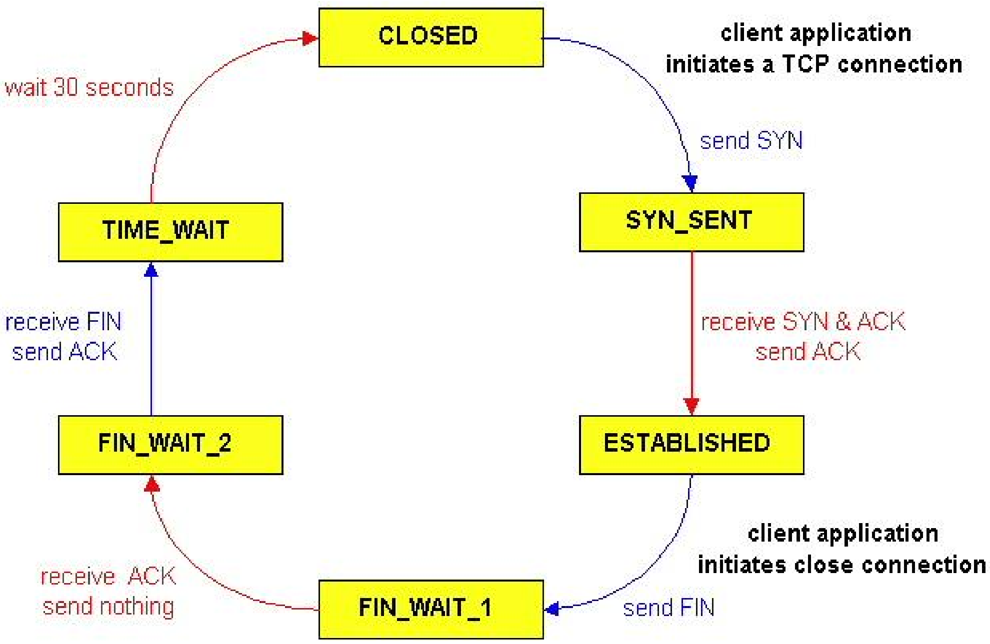
Holding Time

- Upon receiving an HTTP request r , the server decides on a holding-time interval T(r)
- The server then leaves the connection open for at most T(r) seconds from the moment it received r
- If a new request r' arrives within the next T(r) seconds, then a new holding-time interval T(r') is in effect
- Otherwise the connection is terminated after T(r) seconds

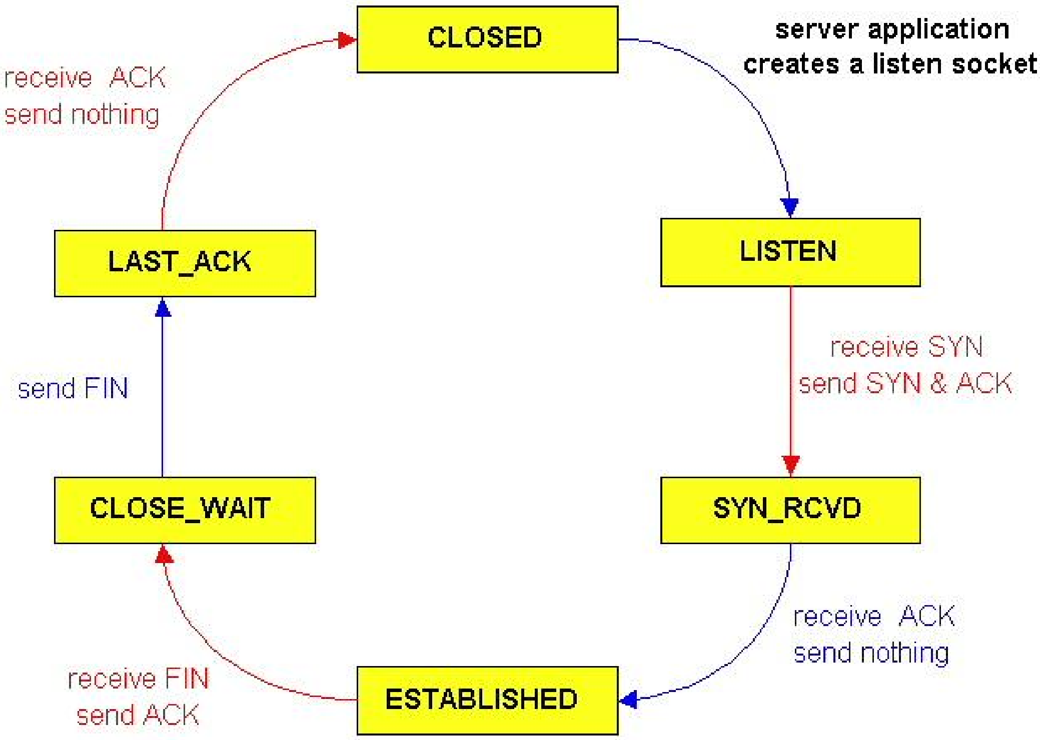
TCP State Transition



TCP Client Lifecycle



TCP Server Lifecycle



Connection Management Policy (1 of 3)

- Policy A is an algorithm that determines an interval T(r) for every request r
- Consider a request sequence s
- The profit (number of hits), PA of a policy A on s is the number of requests that did not require opening a new connection
- The number of misses, MA of A on s is number of requests that require opening a new connection
- The open-cost, HA of a policy A is total time connections are open

What to model?

- Trade-offs between open-cost and number of misses

Principles of Congestion Control

References

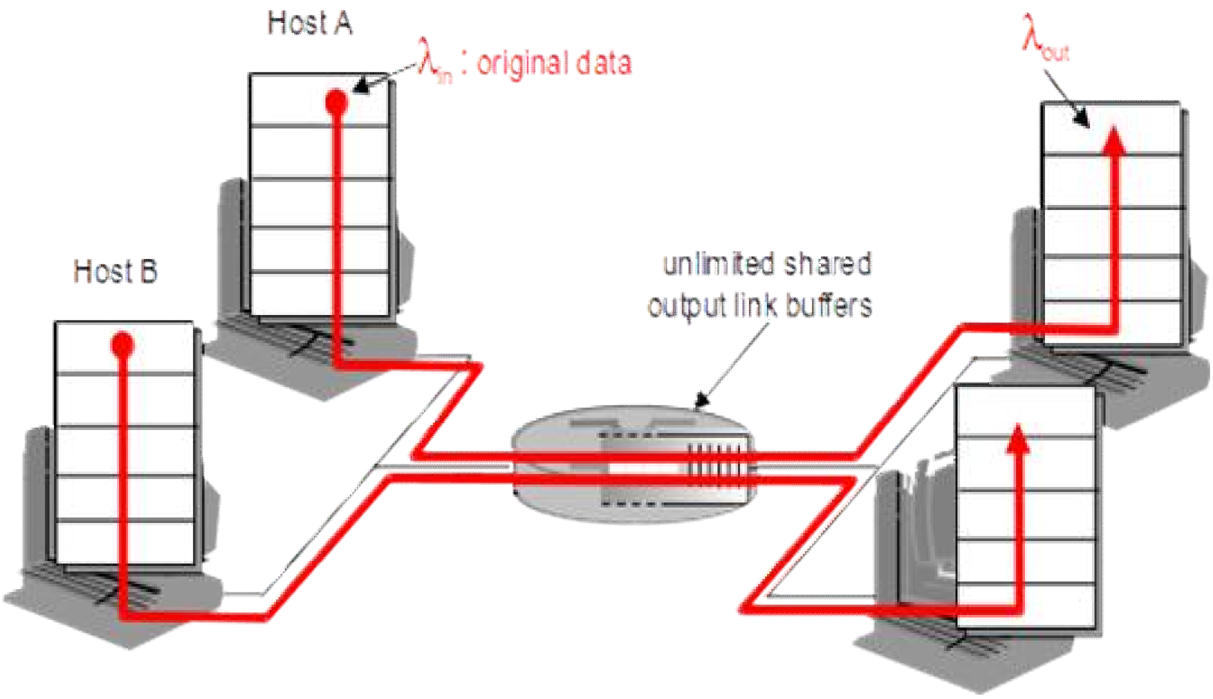
RFC 2914: Congestion Control Principles

Kurose and Ross. “Computer Networking Top-Down Approach Featuring the Internet”.

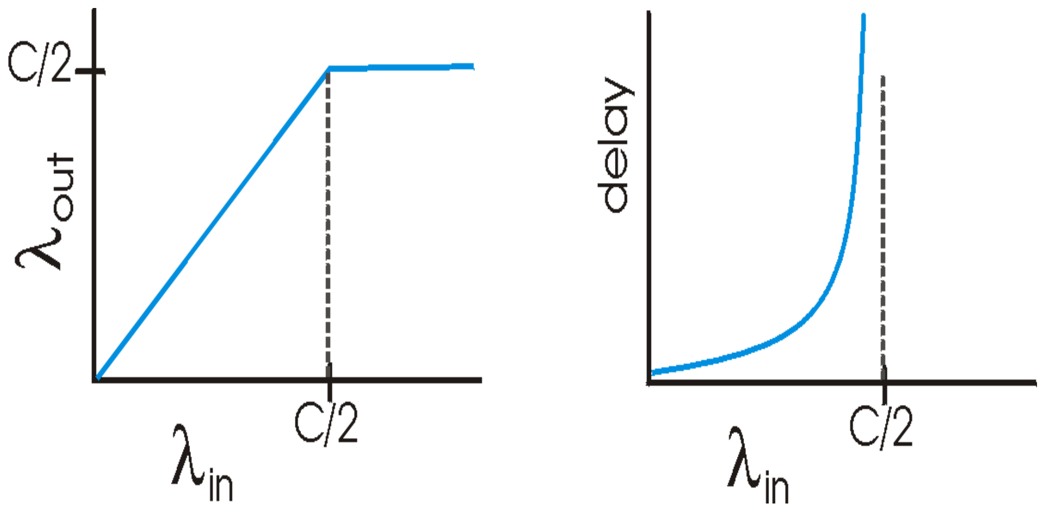
Introduction

- Too many sources sending too much data too fast for network to handle
- Manifestations
- Lost packets
  - buffer overflow at routers
- Long delays
  - Queuing in router buffers

Infinite Buffer Scenario

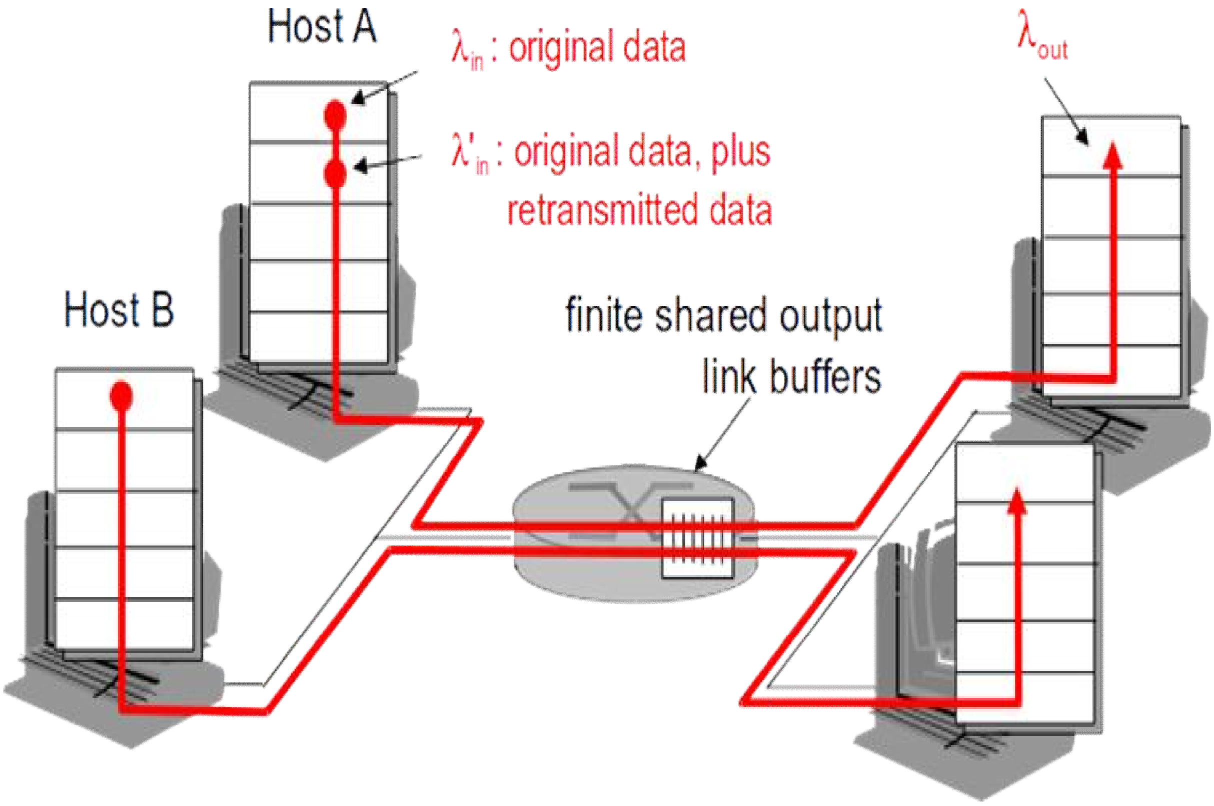


Effects of Congestion

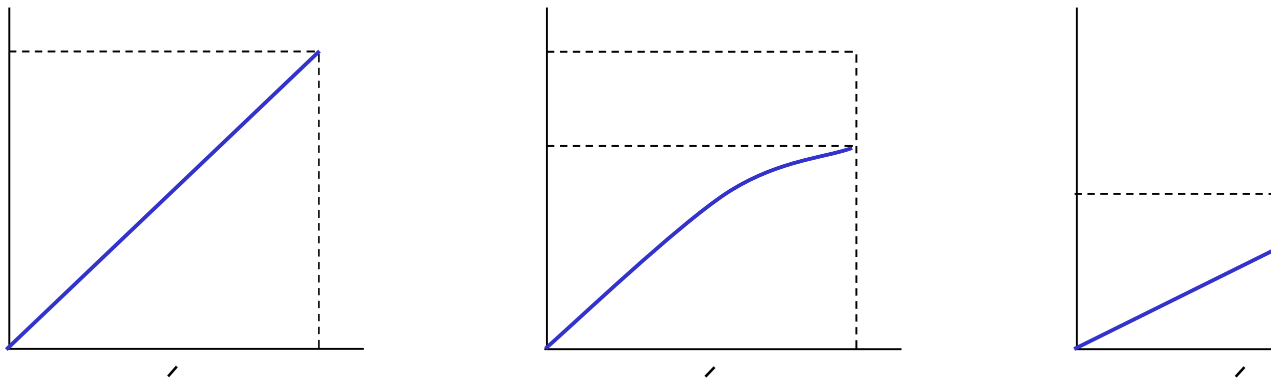


- Large delays when congested
- Maximum achievable throughput

Finite Buffer Scenario



Effects of Congestion



a. No loss
b. Perfect loss
c. Imperfect loss

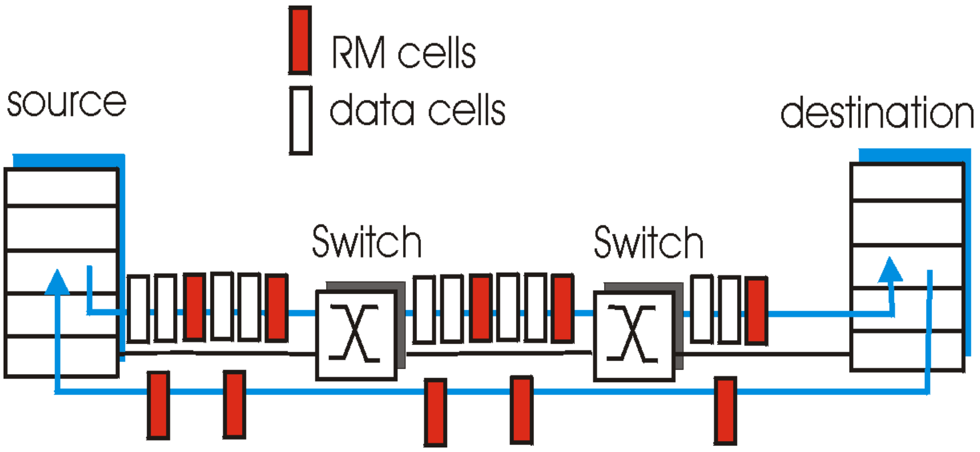
Combat Strategies (1 of 2)
End-end congestion control

- No explicit feedback from network
- Congestion inferred from end-system observed loss, delay
- Approach taken by TCP
- Network-assisted congestion control
- Routers provide feedback to end systems
- Single bit indicating congestion (SNA, DEC bit, TCP/IP ECN, ATM)
- Explicit rate sender should send at

ATM ABR Congestion Control
References
Kurose and Ross. “Computer Networking Top-Down Approach Featuring the Internet”.

Introduction
- Available Bit Rate (ABR), a service used in ATM networks
- Source and destination don't need to be synchronized
- ABR does not guarantee against delay or data loss
- Allow network to allocate available bandwidth fairly over present ABR sources

Operation
- Elastic service
- If sender’s path is under loaded
- Use available bandwidth
- If sender’s path congested
- Sender throttled to minimum guaranteed rate



Combat Congestion
- Two-byte ER (explicit rate) field in RM cell
- Congested switch may lower ER value in cell
- Sender’s send rate thus minimum supportable rate on path
- EFCI bit in data cells is set to 1 in congested switch
- If data cell preceding RM cell has EFCI set, sender sets CI bit in returned RM cell

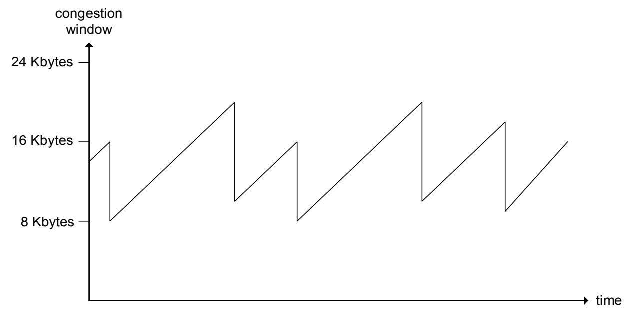
TCP Congestion Control
References
Kurose and Ross. “Computer Networking Top-Down Approach Featuring the Internet”.

Introduction
- End-end control (no network assistance)
- Sender limits transmission

  LastByteSent-LastByteAcked<= CongWin
- CongWin is dynamic, function of perceived network congestion

Operation

- Loss event = timeout *or* 3 duplicate acks
- TCP sender reduces rate (CongWin) after loss event
- Three mechanisms
  - — AIMD
  - — Slow start
  - — Conservative after timeout events



$$\text{Rate} = \frac{\text{CongWin}}{\text{RTT}} \text{ Bytes/sec}$$

Leaky Bucket and Token Bucket

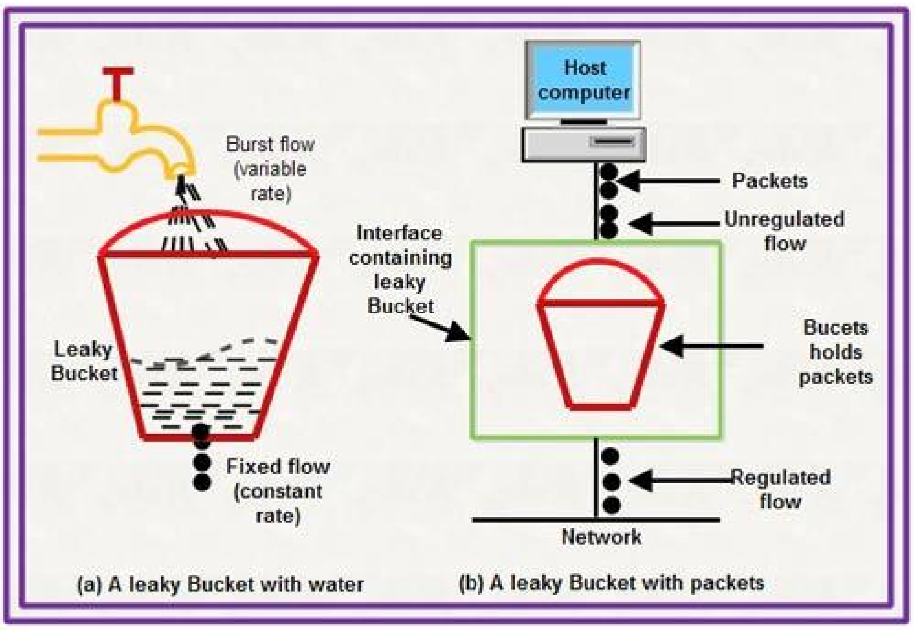
Leaky Bucket

- Buffering of the traffic to help manage and control the flow of traffic onto and through the network
- "Leaky" means buffer that is constantly flowing

Operation

- Traffic enters into the buffers and is tagged, based on the amount of packets allowed by the carrier
- If the user exceeds the amount of packets flow per increment then the buffer is filled and begins to empty out the bottom side at a constant rate

Leaky Bucket Algorithm

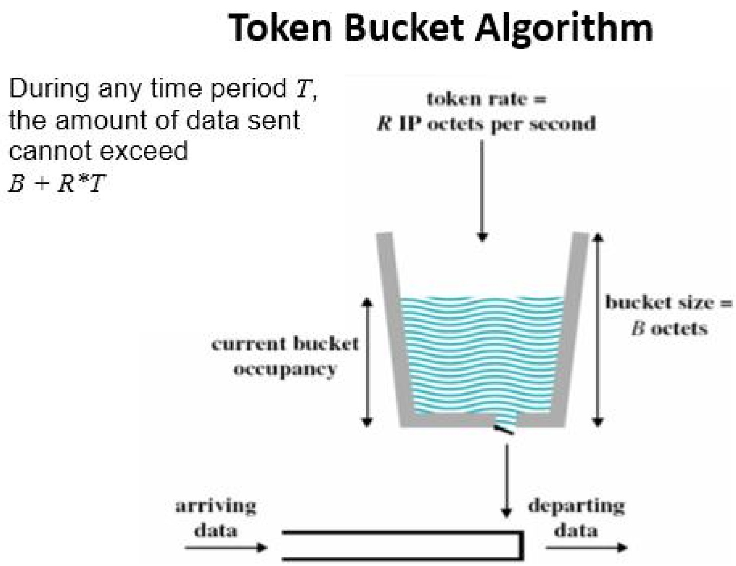


Token Bucket

- Many traffic sources can be defined by token bucket scheme
- Provides concise description of load imposed by flow
- Easy to determine resource requirements

Operation

- Provides input parameters to policing function
- IP packet may be processed if sufficient octet tokens to match the IP data number of tokens
- If insufficient tokens available, the packet is relegated to best-effort service
- To transmit a packet through router, one token must be removed
- If token bucket is empty, packet is queued waiting for next token
- If there is backlog of packets & an empty bucket, packets emitted smoothly



Quality of Service
Background

- Broadband IP packet networks are multiservice, all-purpose communications platforms
- Spurred QoS efforts
- Simplest strategy to the one-size-fits-all best-effort service in today's Internet: divide traffic into classes
- Provide different levels of service to these different classes of traffic

Introduction

- QoS a non-issue for circuit-switched networks
- Layer 2 and 3 QoS approaches
- ATM and Frame Relay provide L2 QoS
- Provide circuit-like emulation
- Traffic agreements
- Traffic control
- Connection admission control
- Congestion notification
- Fragmentation

QoS at Network Layer

- IP QoS is concerned with end-to-end internetwork
- With every hop L3 QoS parameters mapping to L2 QoS
- Type of Service (TOS) field provides initial IP network class of service mechanism
- Three precedence bits classify eight categories of services
- Lower precedence dropped for higher precedence in congestion
- Network equipment vendors rarely provide precedence bits usage

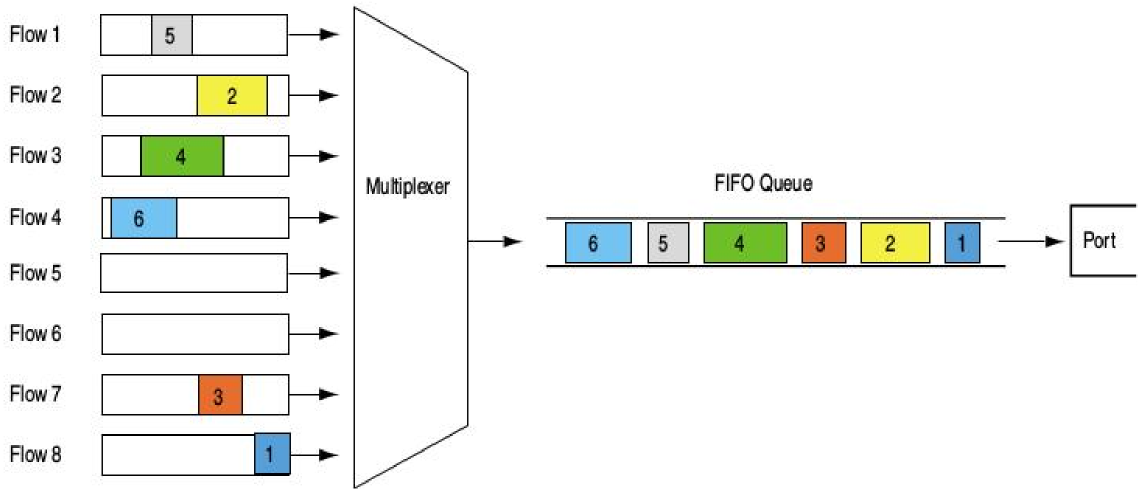
QoS Models

- Two QoS models for IP packet networks
- IntServ
  - Simulate "virtual circuit" of ATM or frame relay on L3
  - Sets up an end-to-end route with fixed QoS parameters
- DiffServ
  - Defining several common classes of service
  - Each with associated queue priorities and drop precedence on a per-hop basis

Hard vs Soft QoS

- Hard guarantee applications will receive its requested quality of service (QoS) with certainty
- Soft guarantee application will receive its requested quality of service with high probability

Fair Queues
First In First Out



Motivation for FQ

- During periods of congestion, FIFO queuing benefits UDP flows over TCP flows
- A bursty flow can consume the entire buffer space of a FIFO queue
- PQ totally favours TCP over UDP

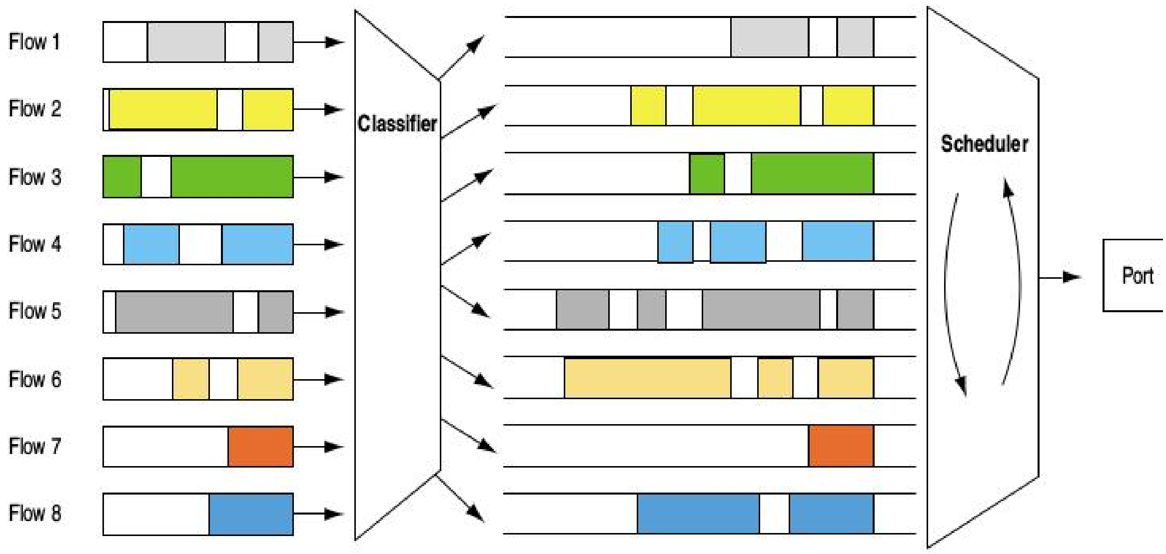
Introduction

- FQ is foundation for a scheduling disciplines designed to ensure that each flow has fair access to network resources
- Prevents a bursty flow from consuming undue bandwidth share
- Also called per-flow or flow-based queuing

Operation

- Packets are first classified into flows by the system
- Assigned to a queue that is specifically dedicated to that flow
- Queues are then serviced one packet at a time in round-robin order
- Empty queues are skipped

Fair Queuing with Classifier



Benefits

- Primary benefit of FQ is extremely bursty or misbehaving flow does not degrade QoS delivered to other flows
- Each flow is isolated into its own queue
- If a flow attempts to consume more than its share of BW, its queue is affected

Performance

- Allocation of single resource amongst N users
- Total resource μ_{Total}
- Each user i requests ρ_i
- Each user i receives μ_i Conditions:
- No user receives more than its request
- No other user satisfying condition 1 has a higher minimum allocation
- Above condition remains recursively true as we remove the minimal user & reduce total resource
- $\mu_{Total} \ll \mu_{Total} - \mu_i$
- Conditions:
- $\mu_i = Min(\mu_{Fair} - \rho_i)$
- Above condition remains recursively true as we remove the minimal user & reduce total resource

$$\mu_{Total} = \sum \mu_i$$

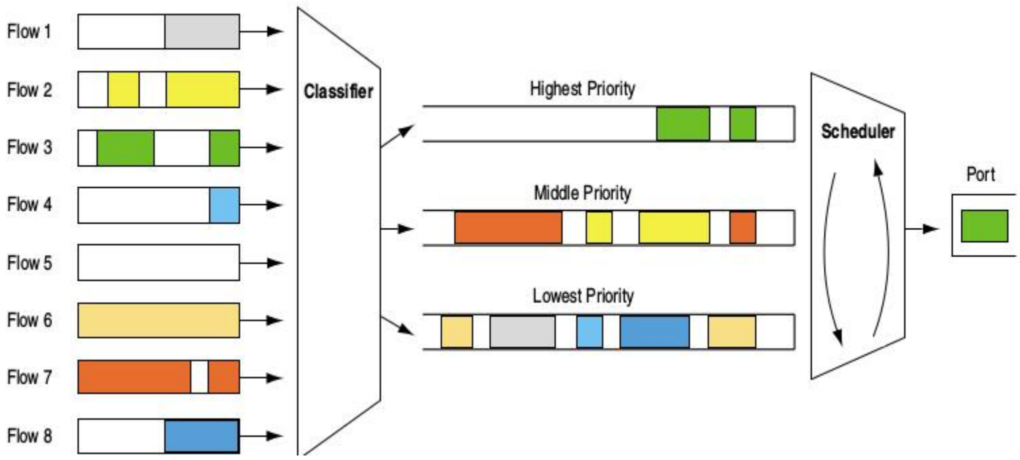
Priority Queues
Motivation

- Designed to provide a relatively simple method of supporting differentiated service classes
- To provide respective services to
  - Interactive traffic
  - Voice
  - Video
  - And best effort

Operation

- Packets classified and placed into different priority queues
- Packets scheduled from the head of a queue only if all queues of higher priority are empty
- Within each of the priority queues, packets are scheduled in FIFO order

Priority Queuing with Classifier



Priority Queuing with Classifier

Departure time of i^{th} packet through k^{th} queue =

$$T_i^k = \sum_{j=1}^{i-1} T_{r_j} + T_{s_i}$$

T_{rj} : Resident time of an item j in queue k
T_{si} : Service time of an item i i.e., processing time by the system

Variants

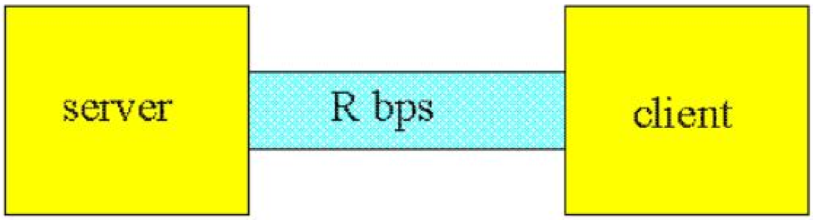
- Strict priority queuing
  - packets in a high-priority queue are always scheduled before packets in lower-priority queues
- Rate-controlled priority queuing
  - High-priority queue scheduled before lower-priority queues
  - Only if the amount of traffic in the high-priority queue stays below a user-configured threshold

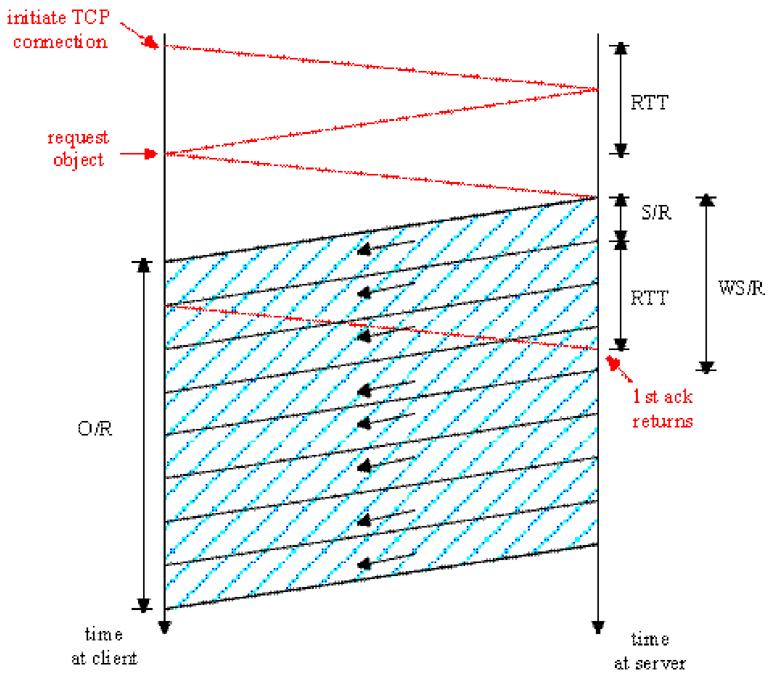
Static Window Modeling

Assumption

- Assume one link between client and server of rate R
- S: MSS (bits)
- O: object size (bits)
- No retransmissions (no loss, no corruption)
- Fixed congestion window, W segments

A simple one-link network connecting a client and a server





Operation (1 of 2)

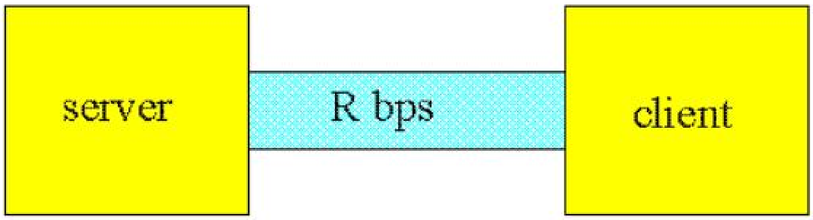
- Server not permitted to have more than W unacknowledged outstanding segments
- Server receives request from client
- Server sends W segments back-to-back to the client
- . Server then sends one segment into the network for each acknowledgement it receives
- Server continues to send one segment for each acknowledgement until all of the segments of the object have been sent

First Case

- Server receives ACK for first segment of first window before completing transmission of first window
- WS/R > RTT + S/R

$$\text{Delay} = 2\text{RTT} + O/R$$

Static Window Modeling—2
A simple one-link network connecting a client and a server

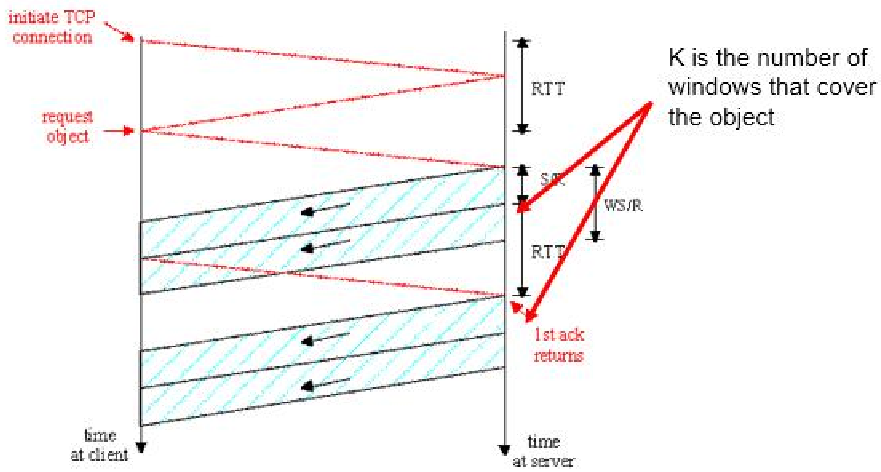


Second Case

- Server transmits first window's worth of segments before the server receives ACK for first segment in the window

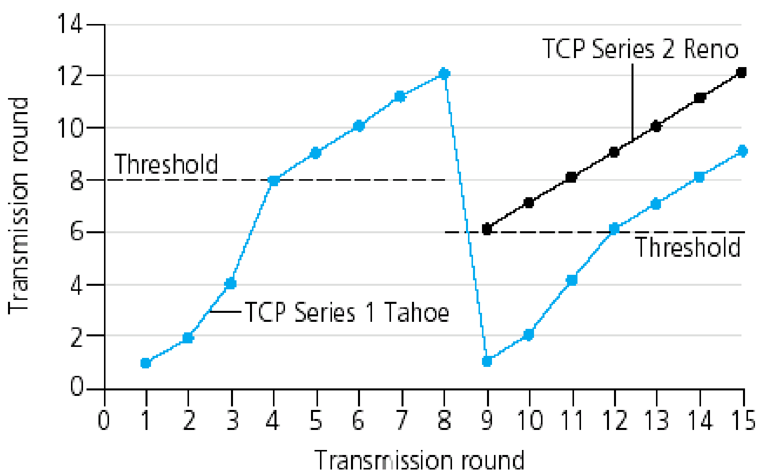
WS/R < RTT + S/R

- It is a scenario where the propagation delay dominates transmission time



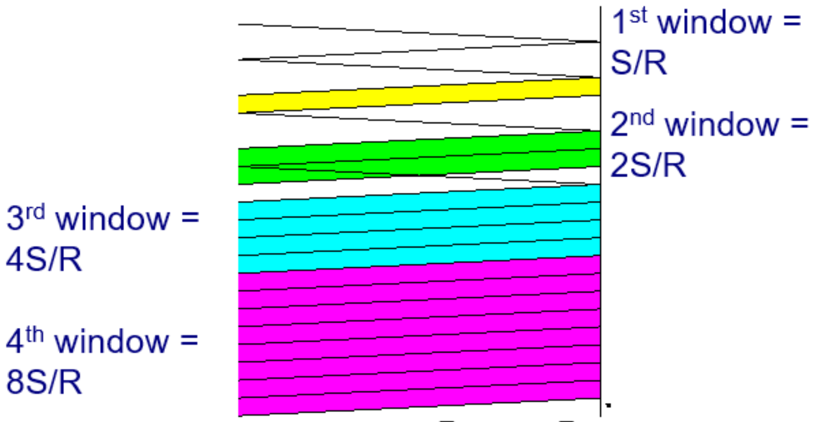
Delay = 2RTT + O/R+ (K-1)[S/R + RTT – WS/R]

Dynamic Window Modeling
TCP Congestion Dynamics



Assumptions

- Server starts with congestion window of one segment
- When it receives an ACK for segment, it increases its congestion window to two segments
- Sends two segments to the client
- Congestion window doubles every RTT



$$Latency = 2RTT + \frac{O}{R} + P\left[RTT + \frac{S}{R}\right] - (2^P - 1)\frac{S}{R}$$

$$\frac{S}{R} + RTT = \text{time from when server starts to send segment until server receives acknowledgement}$$

$$2^{k-1}\frac{S}{R} = \text{time to transmit the kth window}$$

$$\left[\frac{S}{R} + RTT - 2^{k-1}\frac{S}{R}\right]^{+} = \text{idle time after the kth window}$$

Example

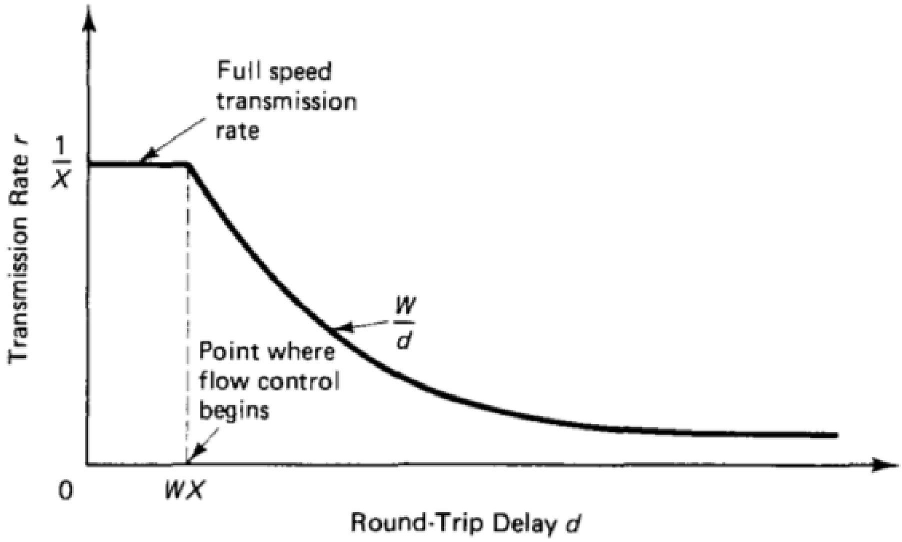
- O/S = 15 segments
- K = 4 windows
- Q = 2
- P = min{K-1,Q} = 2

Server idles P=2 times

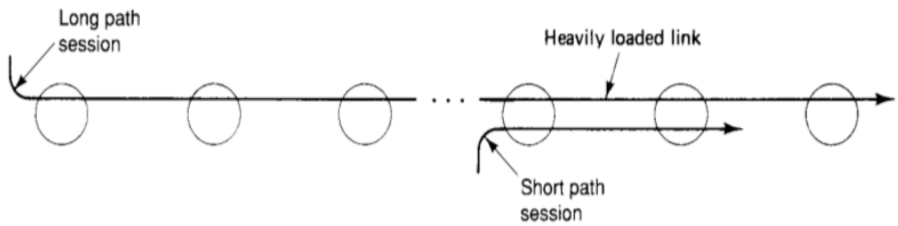
End-to-End Windows
Limitations

- Cannot guarantee a minimum rate for a session
- Not suited for
  - Voice and video
- Window size tradeoff requirements
  - Limit no. of packets in subnet
  - Full-speed transmission and max throughput

Delay-Throughput Trade-off



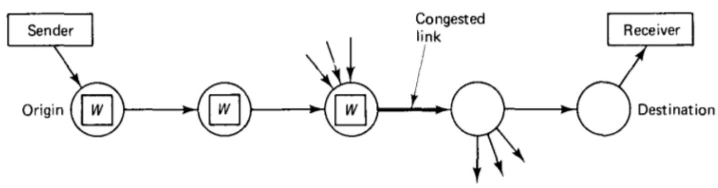
Node-by-Node Windows
Unfairness Problem in End-to-end
Long sessions with larger windows take precedence in intermediate devices



Virtual Circuit Windowing

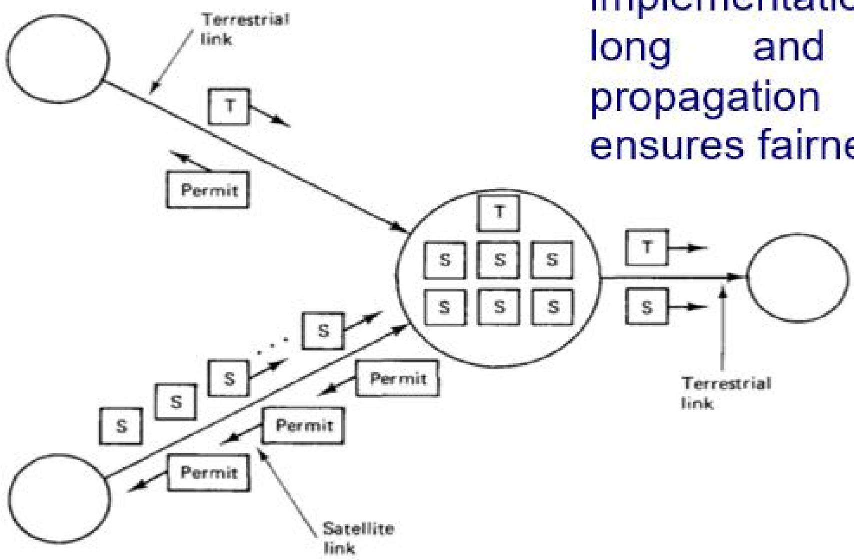
- A separate window for every VC & pair of adjacent nodes along path of VC Main idea
- Receiver avoids accumulation of large no. of packets into its memory
  - Slows down permit returns to sender

Backpressure Effect in VCs



Round Robin + Node-by-node

Round Robin implementation on long and short propagation delays ensures fairne ss



Little's Theorem
Big Questions

- What is the avg no. of customers in the system?
  - The "typical" no. of packets either waiting in queue or undergoing service
- What is the avg delay per customer?
- The "typical" time a packet spends waiting in queue plus the service time

Definition
N= l´T

- N = No. of customers
- l = Arrival rate
- T = Time spent by customers (packets) in the system

Interpretation

- Little's Theorem expresses crowded systems
- Large N associated with long customer delays (T) & vice versa
- Not influenced by arrival process distribution, service distribution, service order, etc.

Probabilistic Little's Theorem
Time average

- The time average of a function is found by evaluating a measure space with the average taken over a time, ΔT
- $P_n(t)$ = Probability of n customers in the system at time t

Statistical (Ensemble) average

- Defined as the number that measures the central tendency of a given set of numbers
- A number of different averages
- Mean, median, mode and range

Probabilistic interpretation

- Little's Theorem admits also a probabilistic interpretation for stationary process
  - Time avg replaceable with statistical avg

$$\overline{N}(t) = \sum_{n=0}^{\infty} n p_n(t)$$

Application

- Little's Theorem becomes applicable to deterministic and probabilistic systems
  - a situation does not exist where the theorem does not hold
  - Often termed as law

Little's Theorem; Applications
End-to-end flow control

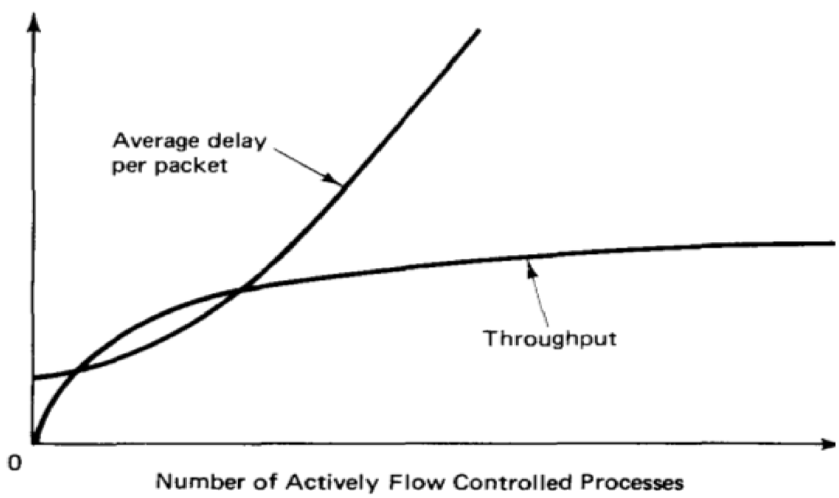
- Recall that end-to-end windows fail to provide adequate control of packet delay
- Little's theorem helps understand the relation
  - Window size
  - Delay
  - Throughput

Average delay per packet

- n flow controlled sessions in the network with fixed window sizesW_1,...Wn
- b = whether piggybacking supported or not
- l = throughput (total accepted input rate of sessions)

$$T = \frac{\sum_{i=1}^{n} \beta_i W_i}{\lambda}$$

Throughput and Delay vs Active Flows
When network is heavily loaded,avg delay per packet increases approximately linearly with the number of active sessions—the total throughput stays approximately constant



Arrivals as Poisson
M/M/1 system

- The M/M/1 queuing system consists of a single queuing station with a single server
  - Communication context: a single transmission line
- Probability distribution of the service time is exponential with mean 1/m sec

Arrivals

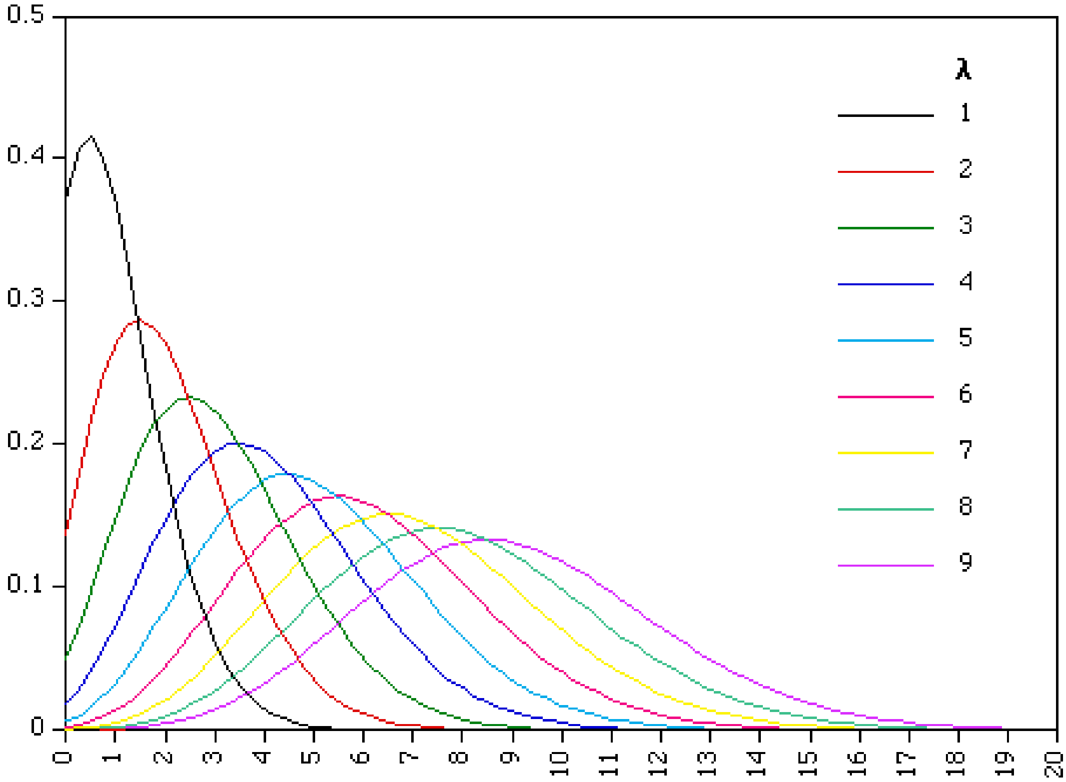
- Customers (packets) arrive according to a Poisson process
- $A(t)$ is a counting process that represents the total number of arrivals that have occurred from to time t

Poisson Process

- A Poisson process is generally considered to be a good model for the aggregate traffic of a large number of
  - Similar and
  - Independent users
- Merges n independent & identically distributed arrival processes
- Each process has arrival rate l/n
- So the aggregate process has arrival rate l
- No. of arrivals occurring in disjoint time intervals are independent
- No. of arrivals in any interval of length t is Poisson distributed with parameter lt

$$P\{A(t+\tau) - A(t) = n\} = e^{-\lambda\tau}\frac{(\lambda\tau)^n}{n!}, \qquad n = 0, 1, \ldots$$

Poisson Distribution



Service Statistics
What is service?

- The set of activities performed at the receiving device
- Router
  - MAC processing
  - Lookup
  - Forwarding decision
- Switch
  - Header processing
  - Port allocation table

Service distribution

- S_n is the service time of the nth customer
- Customer (packet) service times have an exponential distribution with parameter m
- m is also called service rate
- Represents the rate (in customers served per unit time) at which the server operates when busy
- Service times are mutually independent
- Also independent of all inter-arrival times
- Density function
- Service distribution

$$P\{s_n \leq s\} = 1 - e^{-\mu s}, \qquad s \geq 0$$

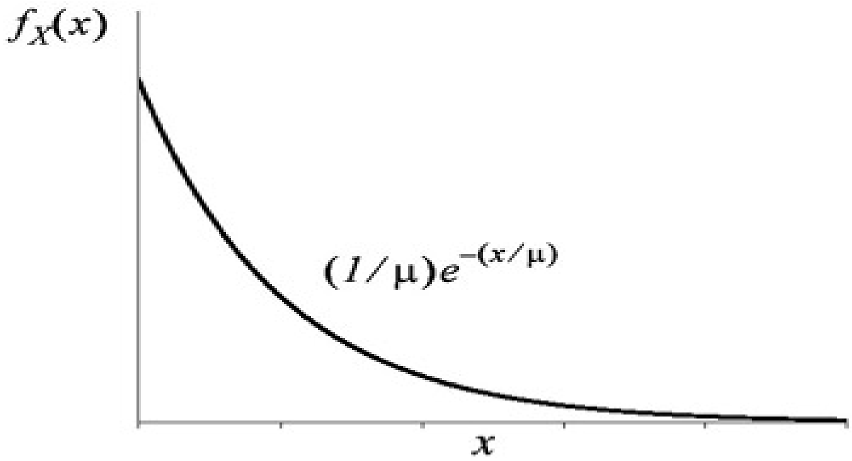
y function of s_n is $p(s_n) = \mu e^{-\mu s_n}$ and its m

Commentary

- In the context of a packet transmission, independence of inter-arrival and service times implies,
  - Length of an arriving packet does not affect the arrival time of the next packet

Exponential Distribution
Memorylessness

- Additional time needed to complete a customer's service in progress is independent of when the service started
- Time up to the next arrival is independent of when the previous arrival occurred



Arrival Occupancy Distribution
System under change

- Users (packets) come and leave the system
  - System under continuous change of occupancy
- It is possible that the times of customer arrivals are in some sense nontypical

Non-Typical Arrival

$$a_n = \lim_{t \to \infty} P\{N(t) = n \mid \text{an arrival occurred just after time } t\}$$

Typical Arrival

$$p_n = \lim_{t \to \infty} P\{N(t) = n\}$$

Occupancy distribution

- For M/M/1 systems
- $p_n = a_n$ for n =0,1,...
- Arriving customer finds the system in a "typical" state
- Future arrivals are independent of the current number in the system

Simulating TCP Receive Buffer
Operation

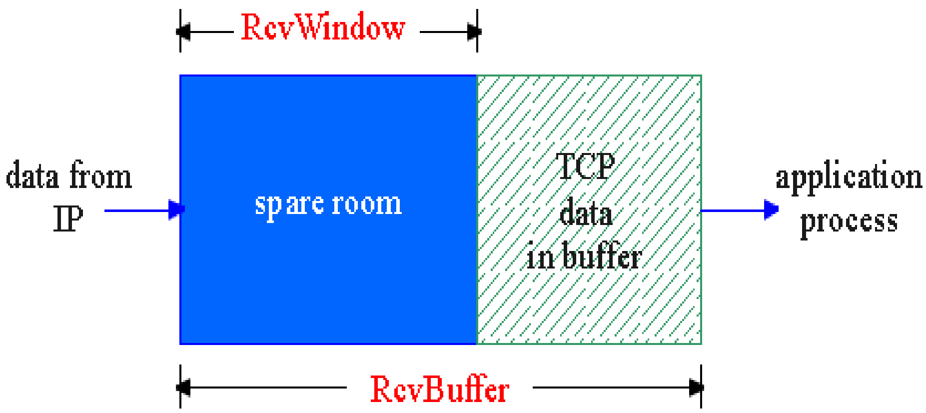
- RFC 1122 identifies host implementation requirements
  - Includes receive buffer to cache sequenced data not yet forwarded

INET Support

- It stores bytes and not segments
- Few implementations store segments on the retransmission queue, and others store only the data bytes

Receiver Window

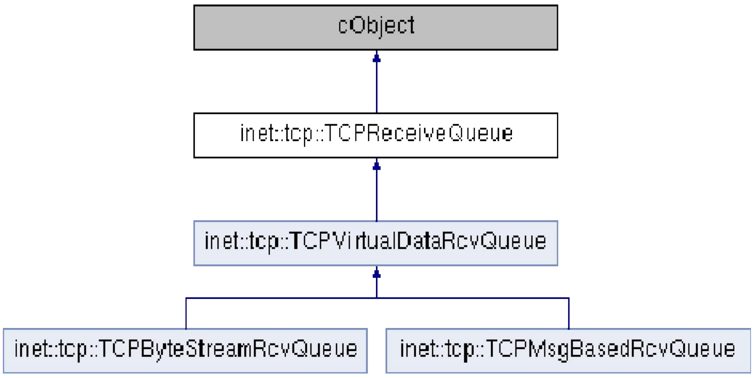
LastByteSent – LastByteAcked



Receive Buffer Support

- inet::tcp::TCPReceiveQueue::getAmountOfBufferedBytes ()
- Returns the number of bytes currently buffered in queue
- inet::tcp::TCPReceiveQueue::getAmountOfFreeBytes (uint32 maxRcvBuffer)
- Returns the number of bytes currently free (=available) in queue

inet::tcp::TCPReceiveQueue Class



Departure Occupancy Distribution

System under change

- Users (packets) come and leave the system
  - System under continuous change of occupancy
- It is possible that the times of customer departures are in some sense nontypical

Non-Typical Departure

$$d_n(t) = P\{N(t) = n \mid \text{a departure occurred just before time } t\}$$

Typical Departure

$$d_n = \lim_{t \to \infty} d_n(t)\,, \qquad n = 0, 1, \ldots$$

Occupancy distribution

- For M/M/1 systems
- $d_n = a_n$ for n =0,1,...
- For each time the number in the system increases from n to n+1 due to an arrival, there will be corresponding decrease from n+1 to n due to departure

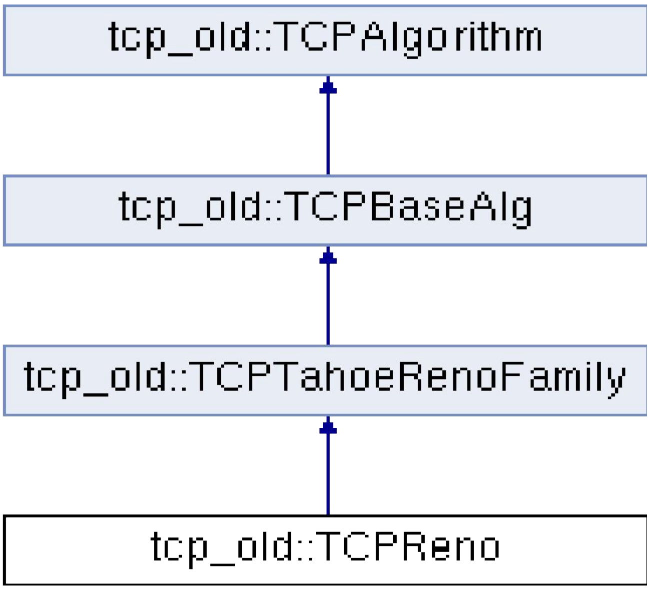
TCP BER Performance
Operation

- RFC 2581 identifies identifies the operation in the wake of TimeOut

TCPReno::recalculateSlowStartThreshold() [protected, virtual]

```
{
   // set ssthresh to flight size/2, but at least 2 MSS
   // (the formula below practically amounts to ssthresh=cwnd/2 most of the time)
   uint flight_size = std::min(state->snd_cwnd, state->snd_wnd);
   state->ssthresh = std::max(flight_size/2, 2*state->snd_mss);
   if (ssthreshVector) ssthreshVector->record(state->ssthresh);
}
```

tcp_old::TCPReno Class Reference



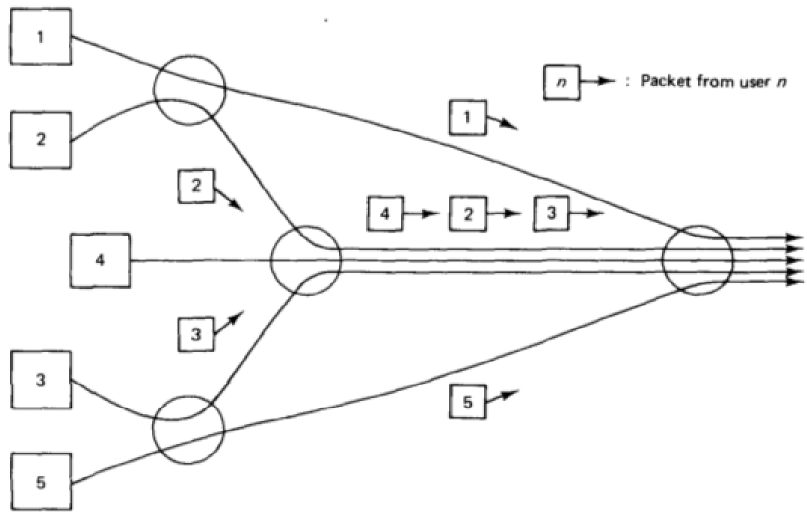
Problem Set 2
TCP TimeOut
Suppose that the five measured SampleRTT values are 106, 120, 140, 90 & 115 ms. Compute the EstimatedRTT after each of these SampleRTT values is obtained, using a value of $\alpha = 0.125$ & assuming that the value of EstimatedRTT was 100 ms just before the first of these five samples were obtained. Compute also the DevRTT after each sample is obtained, assuming a value of $\beta = 0.25$ and assuming the value of DevRTT was 5 ms just before the first of these five samples was obtained. Last, compute the TCP TimeoutInterval after each of these samples is obtained.

TCP Flow and Congestion Control
Host A is sending an enormous file to Host B over a TCP connection. Over this connection there is never any packet loss and the timers never expire. Denote the transmission rate of the link connecting Host A to the Internet by R bps. Suppose that the process in Host A is capable of sending data into its TCP socket at a rate S bps, where $S = 10 \cdot R$. Further suppose that the TCP receive buffer is large enough to hold the entire file, and the send buffer can hold only one percent of the file. What would prevent the process in Host A from continuously passing data to its TCP socket at rate S bps? TCP flow control? TCP congestion control? Or something else? Elaborate.

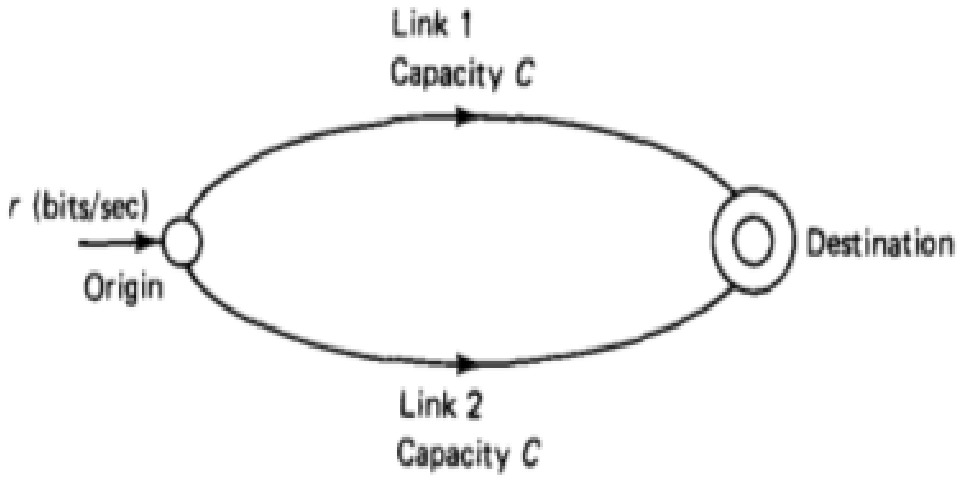
Virtual Circuit Networks
Basics
- Source-to-destination paths behave much like telephone circuit
- Performance guaranteed
- Network actions along source-to-dest path needed

Operation
- Call setup, teardown for each call *before* data can flow
- Each packet carries VC identifier
- Every router on source-dest path maintains “state” for each passing connection
- Resources (bandwidth, buffers) allocated to VC

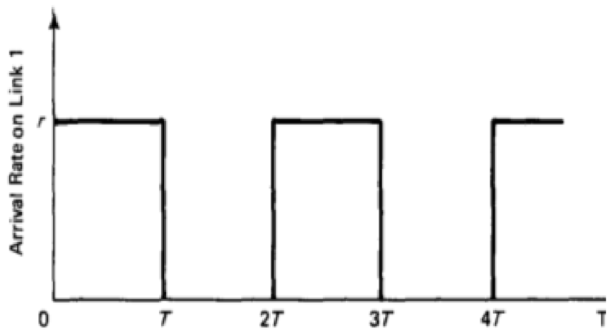
Packets Along the Same Path



Two Links Network Example



Stability Issues in VCs

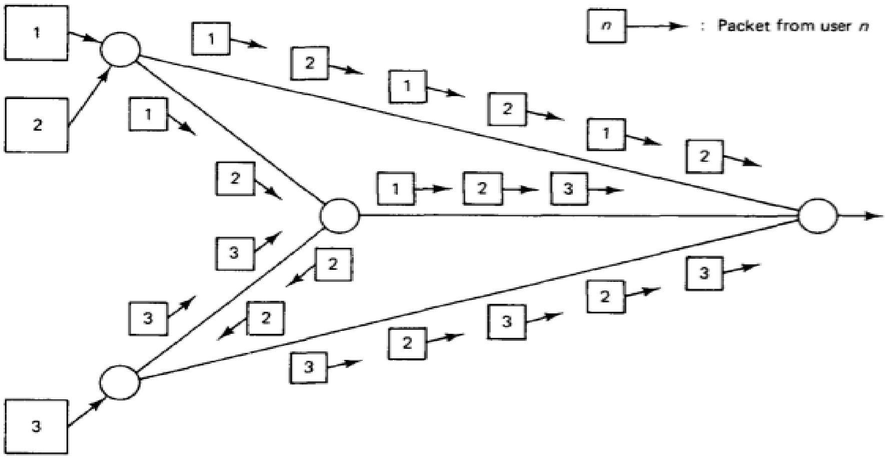


- Arrival rate on link 1 using the shortest path
- Only one path is used for routing at anyone time if the shortest path update period is much larger than the time required to empty the queue of waiting packets at the time of an update

Datagram Networks
Basics

- Two packets of the same user pair can travel along different routes
- A routing decision is required for each individual packet

Packets Along Different Paths

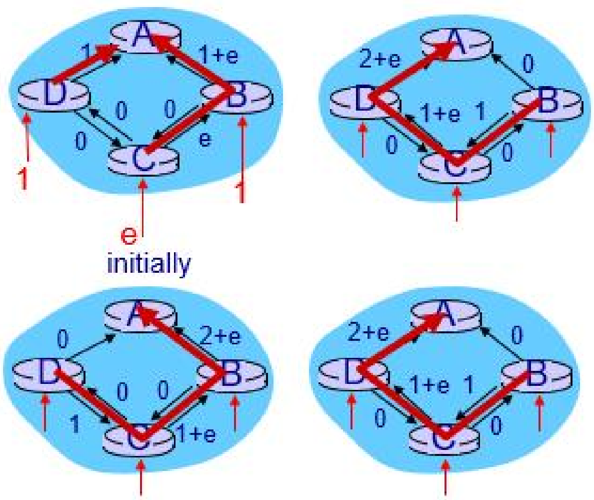


Complexity

- Each iteration of link state routing protocols
- n(n+1)/2 comparisons: $O(n^2)$
- More efficient implementations possible: O(nlogn)

Oscillations

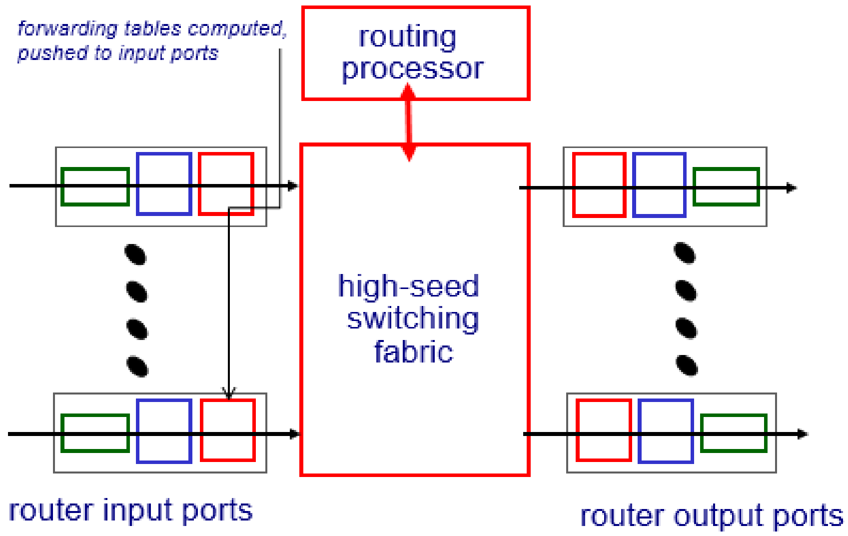
- Given these costs, finding new routes resulting in new costs



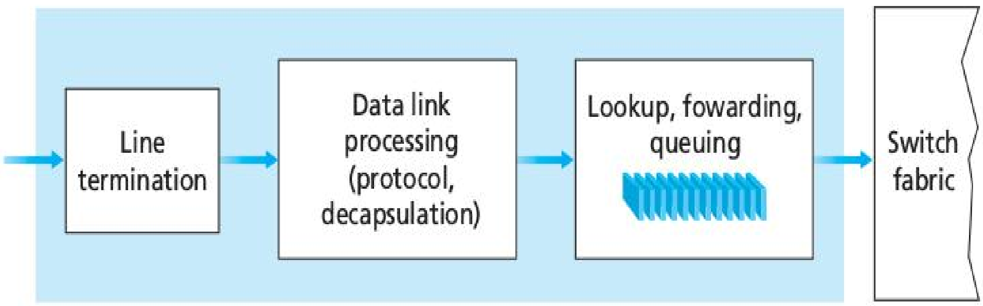
Input Processing
Basics

- Two key router functions:
- Run routing algorithms/protocol (RIP, OSPF, BGP)
- Forwarding datagrams from incoming to outgoing link

Router Functionality



Router Input



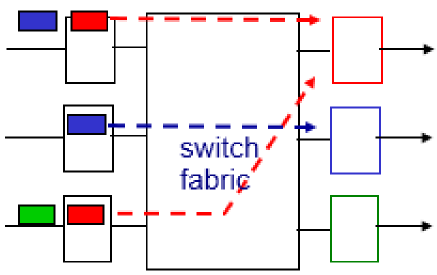
Distributed Switching

- Given datagram dest., lookup output port using forwarding table in input port memory
- Complete input port processing at 'line speed'
-

Input port queuing

- Fabric slower than input ports combined
- Queuing may occur at input queues

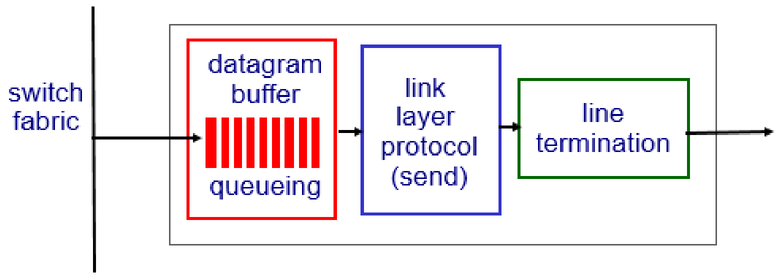
Input Port Queuing



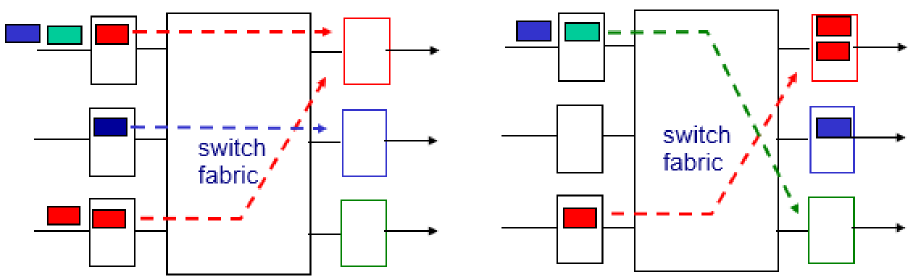
Output Processing
Operations

- Buffering required when datagrams arrive from fabric faster than the transmission rate
  - If R_{switch} is N times faster than R_{line}
- Scheduling discipline chooses among queued datagrams for transmission

Router Output Interface



Output Port Buffering



at *t*, packets more from input to output

one packet time later

How much to Buffer?

- RFC 3439: average buffering equal to "typical" RTT (say 250 msec) times link capacity C
- C = 10 Gpbs link
- 2.5 Gbit buffer
- With N flows, buffering equal to

$$\frac{RTT \cdot C}{\sqrt{N}}$$

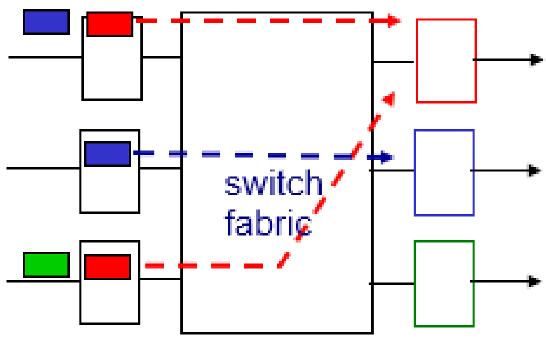
Head of Line Blocking
Input Port Overflow

- Fabric slower than input ports combined   queuing may occur at input queues
- Queuing delay and loss due to input buffer overflow!

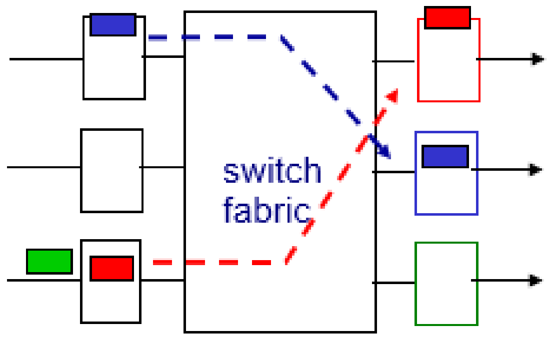
Head of Line

- Queued datagram at front of queue prevents others in queue from moving forward

Scenario



output port contention:
only one red datagram can be transferred.
lower red packet is blocked



one packet time later:
green packet experiences HOL blocking

Random Early Detection
Drop Tail

- Conventional tail drop algorithm
- A router buffers as many packets as it can
- Simply drops the ones it cannot buffer
- If buffers constantly full, network is congested
- Tail drop distributes buffer space unfairly among traffic flows

Active Queue Management

- When buffer becomes full or gets close to becoming full
- AQM is intelligent drop network congestion of network packets inside a buffer of NIC
- Often with the larger goal of reducing

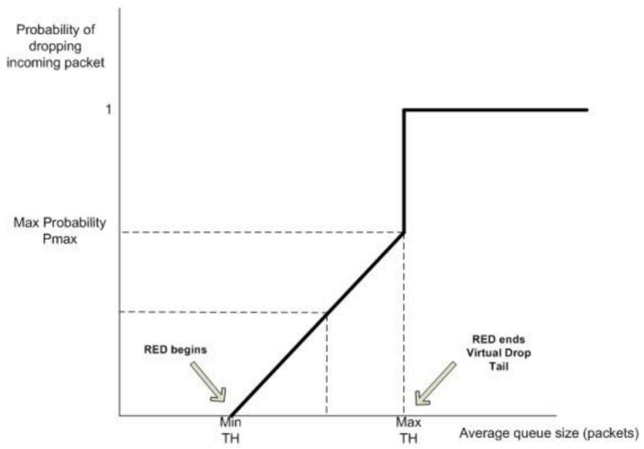
Drop Tail

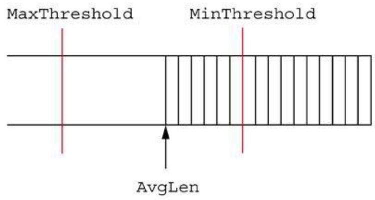
- Conventional tail drop algorithm
- A router buffers as many packets as it can
- Simply drops the ones it cannot buffer
- If buffers constantly full, network is congested
- Tail drop distributes buffer space unfairly among traffic flows

RED Operation

- Monitor avg queue size & drop packets based on probabilities
- If buffer empty, all incoming packets accepted
- As queue grows, P for dropping incoming packet grows
- When buffer full, $P = 1$ all incoming packets dropped

Operation:



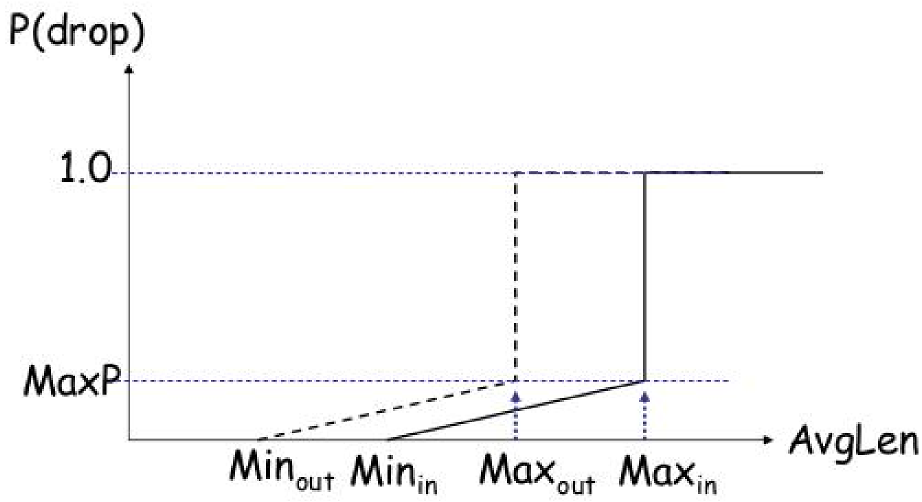


RED with In & Out (RIO)
Background

- Similar to RED, but with two separate probability curves
- Has two classes, "In" and "Out" (of profile)
- "Out" class has lower minimum threshold
- Packets are dropped from this class first
- As avg queue length increases, "In" packets are dropped
- Since best-effort is included in the "Out" class, assured traffic can starve best-effort

Operation
For each packet arrival
if it is an In packet
calculate the average In queue size avg_in ;
calculate the average queue size avg_total ;
If it is an In packet.
if min_in < avg_in < max_in
calculate probability P in
with probability P in , drop this packet;
else if max_in < avg_in
drop this packet.
If it is an Out packet
if min_out < avg_total < max_out
calculate probability Pout;
with probability Pout drop this packet;
else if max_out < avg_total
drop this packet

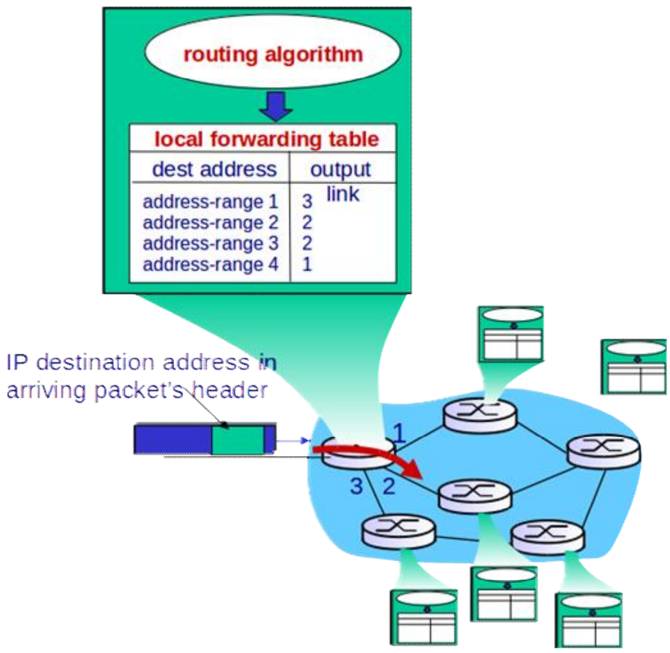
Operation



Routing Algorithms

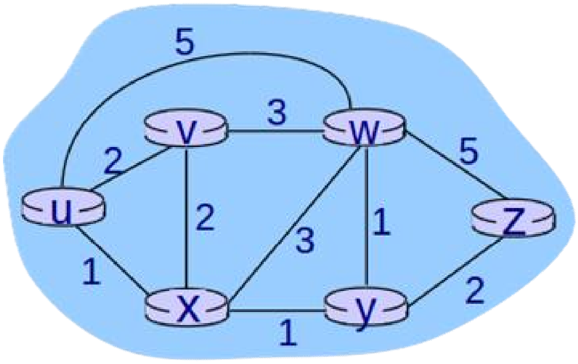
Interplay

- Routing algorithm determines end-end-path through network
- Forwarding table determines local forwarding
  - at this router
  - for IP destination address in arriving packet's header



Graph abstraction

- Graph: G = (N,E)
- N = set of routers = { u, v, w, x, y, z }
- E = set of links ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }



Cost

- Cost could always be 1
- Or inversely related to bandwidth
- Or inversely related to congestion
- Cost of path
- $(x_1, x_2, x_3, \ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

Algorithms
Key question: What is the least-cost path between u and z?
Routing algorithm: Algorithm that finds that least cost path

Complexity of Link State
Global Routing

- All routers have complete topology, link cost information
- Every node constructs a map of the connectivity to the network in the form of a graph Shows which nodes are connected to which other nodes

Link State

- Each node independently calculates best path from it to every possible destination in the network
- The collection of best paths will then form the node's routing tables
- Iterative: After *k* iterations, know path to k destination

Complexity

- For n nodes
- Each iteration: need to check all nodes, w, not in route discovered set N
- Full-mesh: n(n+1)/2
- Omega Notation: $O(n^2)$

Complexity of Distance Vector
Distributed Routing

- Router knows physically-connected neighbors + link costs to neighbors
- Iterative process of computation
- Exchange of info with neighbors

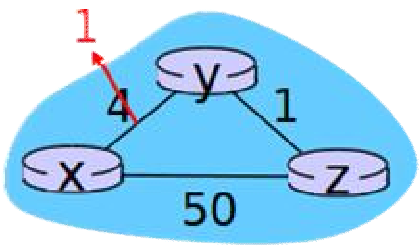
Key Idea

- From time-to-time, each node sends its own distance vector estimate to neighbors
- when x receives new DV estimate from neighbor, it updates its own DV using B-F equation

Count to Infinity Problem
Link Cost Changes

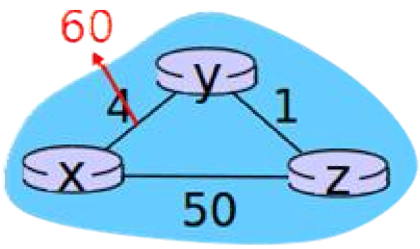
- Node detects local link cost change
- Updates routing info
- Recalculates distance vector
- If DV changes, notify neighbours

Good news



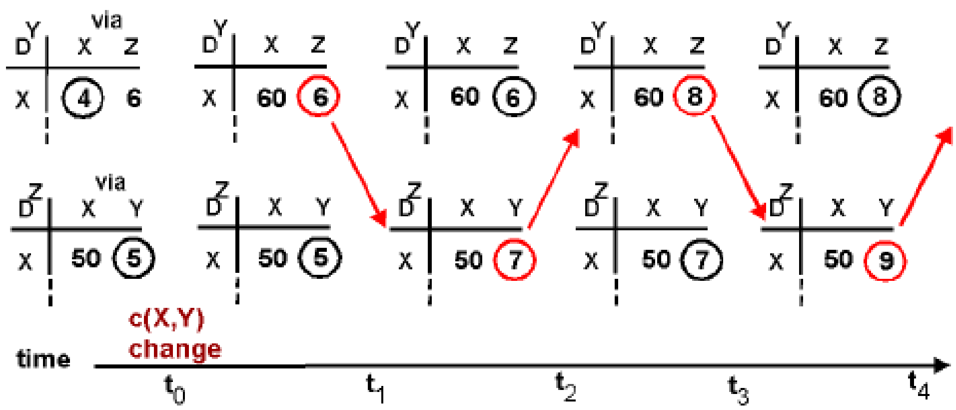
- At time t_0, *y* detects the link-cost change, updates its DV,
- & informs neighbors
- At time t_1, *z* receives the update from *y* and updates its table
- It computes new least cost to *x* & sends neighbors its DV
- At time t_2, *y* receives *z*'s update, updates
- *y*'s least costs do not change, *y* does *not* send message to *z*

Bad News!

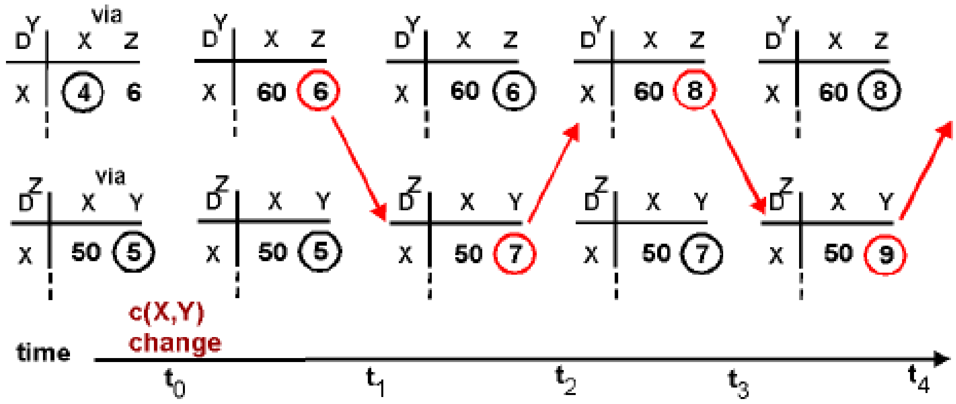


- Good news travels fast
- Bad news travels slow
- Takes 44 iterations before Z eventually computes its path via Y to be larger than 50

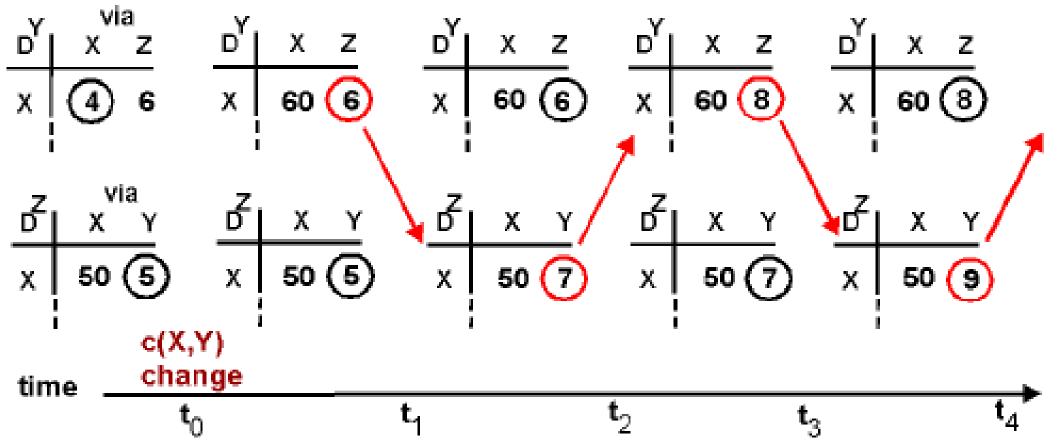
Bad News Causes Loops



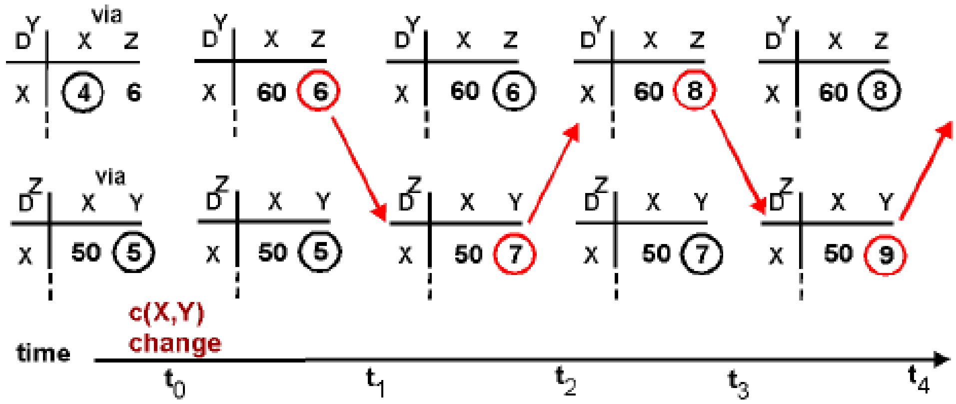
At time t0 Y detects the link cost change (the cost has changed from 4 to 60). Y computes its new minimum cost path to X to have a
cost of 6 via node Z. Of course, we can see that this new cost via Z is wrong



But the only information node Y has is that its direct cost to X is 60 and that Z has last told Y that Z could get to X with a cost of 5. So in order to get to X, Y would now route through Z, fully expecting that Z will be able to get to X with a cost of 5



So in order to get to X, Y would now route through Z, fully expecting that Z will be able to get to X with a cost of 5. As of t1 we have a routing loop—in order to get to X, Y routes through Z, and Z routes through Y.

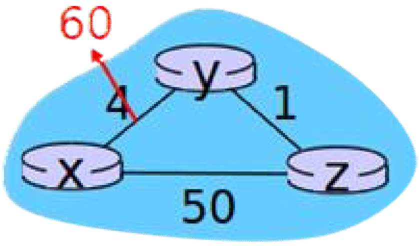


A routing loop is like a black hole—a packet arriving at Y or Z as of t1 will bounce back and forth between these two nodes forever or until the routing tables are changed

Poisoned Reverse
Need

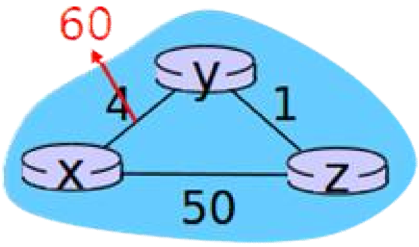
- Bad news travels very slow, especially if the cost change is large
- Ping-pong effect due to looping is undesirable
- Nodes are *blindly* following what is told to them
- Solution: Tell a small lie!
    - Poison the link

Operation



- If Z routes through Y to get to X
- Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z) table
- This lie prevents the loop

Performance



- Poisoned reverse does not work if more than 3-neighbors are involved in looping
- Other techniques such as packet or broadcast ID are incorporated

Hierarchical Routing; Complexity
Need

- All routers identical with a flat network is not true in practice
- Routers vary
    - Connectivity
    - Bandwidth
    - Resources & Cost

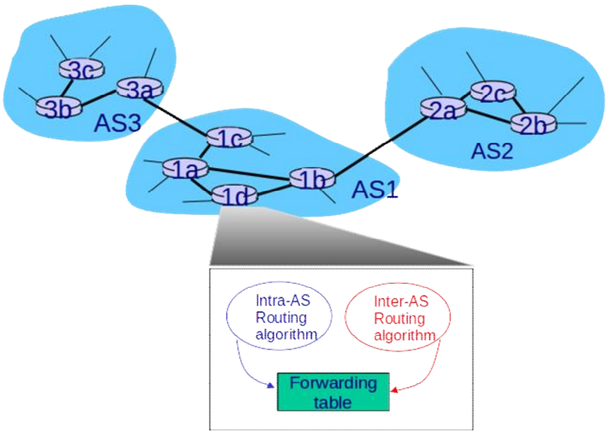
Each network admin wants autonomy
Solution: Make a hierarchical relationship between them.

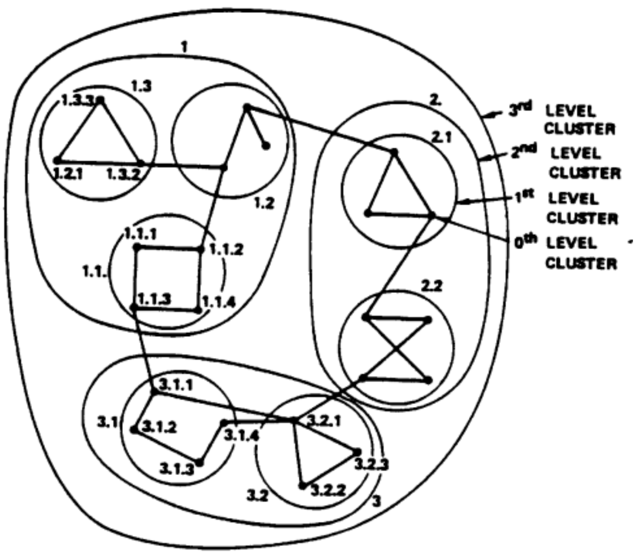
Methodology

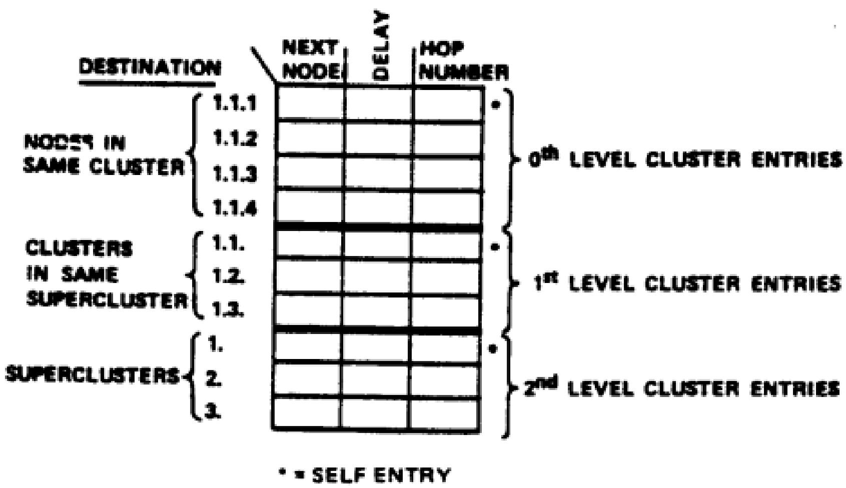
- Collect routers into regions, "autonomous systems" (AS)
- Each AS within an ISP
- ISP may consist of one or more ASes
- In same AS run same routing protocol
- "intra-AS" routing proutersrotocol
    - routers in different

  AS run differentintra-AS routing protocol
- Gateway router:
    - At "edge"
    - Has link to router in another AS

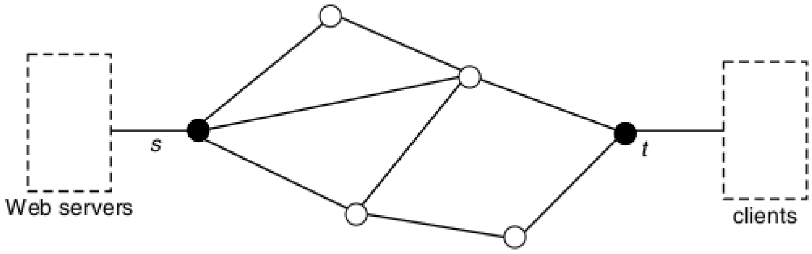
- Forwarding table configured by both intra- and inter-AS routing algorithm
- intra-AS sets entries for internal dests
- inter-AS & intra-AS sets entries for external dests







Elastic Aggregates & TE
A Generalized Scenario



Traffic Aggregate

- Suppose that a request arrives for downloading a file of size V bytes
- V bytes must be transferred from s to t
- Number of download requests arriving over T interval is N(T)

$V_1, V_2, \ldots, V_{N(T)}$

$$V(T) := \sum_{i=1}^{N(T)} V_i.$$

Average Requests

- Over the interval T, if EV is the average file size
- Average requests for an aggregate amount

$$\overline{V(T)} = (EV)\overline{N(T)}$$

Offered Load

- Dividing both sides by T, we get
  - Avg rate at which V(T) grows with time
  - Avg rate at which download requests arrive

$\rho = \lambda\ EV$

- ρ = Offered load expressed in bytes/sec
- λ = Average arrival rate of download requests

Optimal Routing
Feasible Routing

- The sum of all flows on a link should stay below the link capacity

$x(1) + x(2) + \cdots + x(K) \le C$

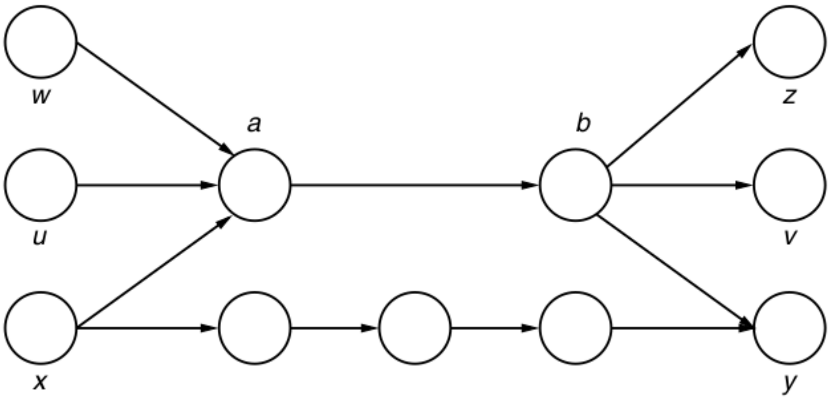
- Spare capacity

$z = C - (x(1) + x(2) + \cdots + x(K))$

Optimization Problem

- Given a network and a set of demands, there may be many feasible routes
- To choose one route from a set, define an objective function
- Choose the route that optimizes the objective function
- Optimal routing is the one that maximizes the smallest spare capacity
- Reasonable, because any link in the network has a spare capacity of at least z
- Increases chance that a future demand between any pair of nodes finds sufficient free capacity.

Limitations of Min Hop Routing
Scenario



Shortest path = Min hop routing

- If weight along each edge (link) set to 1
- Total bandwidth on a route is d × H
- Hmin takes min resources
- Least resource consumption
- H = no. of hops on chosen route
- Demand requires bandwidth d

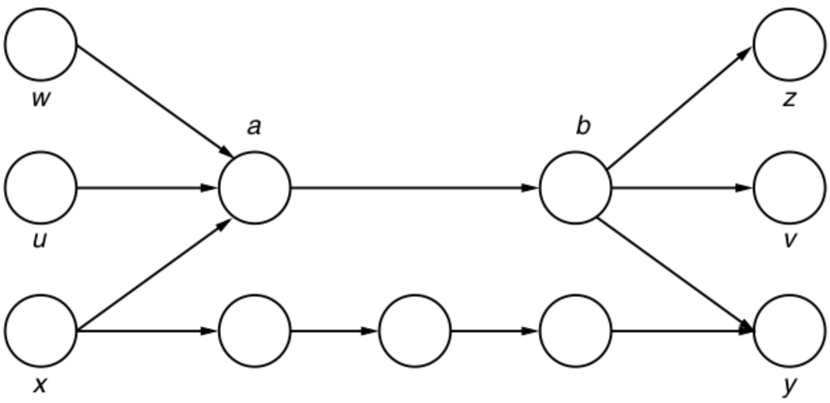
Disadvantages

- Consider that *x* uses *a* and b to reach *y*
- It results in non-utilization of direct hops between them
- Other source-destination pairs would never use these resources
- Network is partitioned

Formulation of Routing Problem
Shortest path is most congested

- One or more links in a network get congested
  - Form sub-paths on shortest path
- Unused bandwidth is available on other links

Routing as a User-Network-Traffic-QoS Phenomenon



Defining Routing Problems

- Shortest-widest path
- Widest-shortest path
- Least-loaded routing
- Maximally loaded routing
- Profile-based routing

.

Minimum Interference Routing

Route interference

- Any chosen route from a router *a* to another router *b* can possibly reduce the capacity available for demands between other node pairs
  - Often a phenomenon in ISP backbone sharing

Max Flow

- Maxflow (*s*, *t*) is a scalar
- Indicates the maximum amount of traffic that can be sent from *s* to *t*
- Exploits all possible paths through the network
  - An upper bound on the total bits/sec that can be sent from *s* to *t*

Minimum interference

- Idealy zero interference
- If maxflow (*s*, *t*) remains unchanged
- Path used for the (*a*, *b*) demand does not share any link with the set of paths available for (*s*, *t*)
- Non-zero minimum interference
- Paths share minimum hops

Problem Formulation

- After the (*a*, *b*) demand has been routed, the smallest maxflow value among all other (*s*, *t*) pairs is maximized

Example

Consider four flows, w.r.t (a, b).

- (30, 15, 6) corresponds to path P1 for (a, b)
- (12, 19, 8) corresponds to path P2 for (a, b)
- (3, 12, 16) corresponds to path P3
- Route P 2 is the minimum interference route for the (a, b) demand

QoS Routing

Single Stream

- A single stream session comes with
- A given bandwidth requirement
- A specified end-to-end delay requirement
- Arrives at the network
- QoS routing is to find a “good” route for the session

Network Operator

- Wider and holistic objectives
  - Minimization of total bandwidth consumed
  - Maximization of the smallest spare capacity on the links of the network

Tradeoff

- End to end
- Hop by hop
- Two QoS models for IP packet networks
- IntServ
  - Simulate the “virtual circuit” of ATM or frame relay on layer-3
- Sets up an end-to-end route with fixed QoS parameters
- DiffServ
- Defining several common classes of service with associated queue priorities and drop precedence on a per-hop basis hops

Nonadditive Metrics
Definition

- Nonadditive link metrics cannot be summed over the links of a path to obtain the path metric
- Must be aggregated through another way
- Example: Bandwidth
- Requires *d* units of BW
- The least available link bandwidth along the path should be *d*

Application

- Wider and holistic objectives
  - Minimization of total bandwidth consumed
  - Maximization of the smallest spare capacity on the links of the network

Implications

- What if no path exists?
  - S-D get isolated
- What if more than one path exists?
- BW measurement freq & accuracy is a tradeoff
- BW measurement is not exact

Solution

- Choose path with highest Prob of having *d* units

Additive Metrics; RMB
Definition

- Additive link metrics are summed over the links of a path to obtain the path metric
- Example: end-to-end delay
- If eah link offers *t* units of delay
- The total links *N* delay is *Nt*

Rate-based Mux

- A multiplexer takes input from various streams of traffic and puts them out on a single line
  - Used fixed sized frames
- Rate matching of heterogeneous sources is required
- Example: WFQ

Weighted Fair Queuing

- WFQ supports fair distribution of BW for variable-length packets
  - Weighted bit-by-bit round-robin scheduling
- Fair allocation of bandwidth
- Each queue receives its configured share of output port bandwidth

Finding Feasible Routes
Network Model

- G(N , L) is the network
- N is the set of nodes
- L is the set of links
- ξ_1 = sum of prop delay & maximum TXN time on link l
- C_l be the available capacity on link
- K = source–destination pairs in the network
- Consider a path P through the network between a source router and a destination router
- Capacity on path P = $\min_{l \in P} C_l$
- $H(P)$ = No. of hops (i.e., links) on path P
- Consider a path P through the network between a source router and a destination router
- Capacity on path P = $\min_{l \in P} C_l$
- $H(P)$ = No. of hops (i.e., links) on path P

Problem

- Find, on connection arrival, a route connecting the SD pair
  - Rate to be allocated on that route
  - Connection's delay and rate requirements are satisfied
  - Capacity constraints are not violated

Upper bound on delay

- Required end-to-end delay
- If all the paths are computed

$$D_k(P, r) = \frac{\sigma_k + H(P)\; c_k}{r} + \sum_{l \in P} \xi_l$$

  - Multi-commodity problem
  - NP hard

Upper Bound on Performance
Route and Rate Allocation (RRA)

- λ connections distrib over I classes given to G(N , L) network
- $\rho_{ik}\lambda$ = number of class i connections for SD pair k

$$\sum_{i=1}^{I} \sum_{k=1}^{K} p_{ik} = 1$$

- If not all connections can be admitted due to capacity, select a subset for admission

Problem Formulation

- What is the maximum value (revenue) of the minimum weighted carried traffic (W_{min}) that any RRA algorithm can extract from the network?

Offline Routing (Integer Linear Program)

$$I(\lambda, \mathrm{p}) = \max \min_{f \in \mathcal{F}} \left(\sum_{i=1}^{I} \alpha_i^f \sum_{k=1}^{K} s_{ik} \right)$$

$$s_{ik} = \sum_{j=1}^{J} n_{ij} \mathbf{G}_{j,k} \quad \forall i \in \mathcal{C}, \forall k \in \mathcal{K}$$

$$\sum_{i=1}^{C} \sum_{j=1}^{J} r_{ij} n_{ij} \mathbf{B}_{j,l} \le C_l \quad \forall l \in \mathcal{L}$$

$$s_{ik} \le p_{ik} \lambda \quad \forall i \in \mathcal{C}, k \in \mathcal{K}$$

- s_{ik} = carried traffic of class i for SD pair k
- n_{ij} = No. of class i connections carried on path j

Non-Rate-Based Multiplexers
Additive Metric

- Non-rate muxes are unlike rate-based
  - Rate requirement is relieved
- Other requirements emerge
  - Bit error rate
  - Packet Loss Probabilities
  - Preferential links or paths

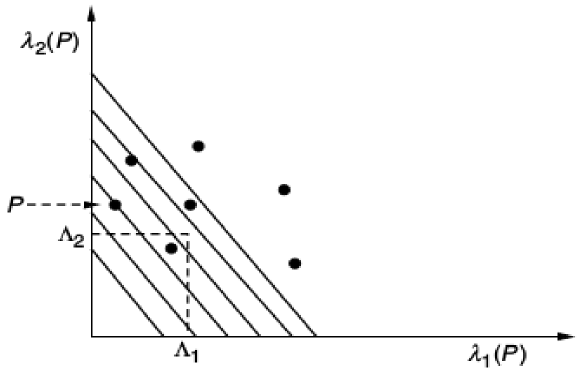
Multi-constrained Feasibility Problem

- m additive constraints are given
- Objective: find a path that satisfies all m constraints

Multi-constrained Feasibility Problem

- m additive constraints are given
- Objective: find a path that satisfies all m constraints
- In case several paths satisfying all m constraints are available
- No criterion specified for choosing one from this set of paths
  - Not defined as an objective function in the optimization problem

Heuristic Interpretation As Constrained Region



$\lambda(P) = \alpha_1\lambda_1(P) + \alpha_2\lambda_2(P)$

m path metric values $\lambda_1(P), ..., \lambda_i(P),...,\lambda_m(P)$, map to a single real value that represents the effective path length.

Efficient Longest Prefix Match
Operation at Router

- Perform a logical AND of netmask and 32-bit destination IP address in the packet
- If result matches network prefix in the forwarding table entry,
- Next hop is the corresponding entry in table.
- Route lookup a search problem

Longest Prefix Match

- Multiple matches of forwarding table entries to a destination IP address are handled through LPF
- If there are multiple matches to an IP address
  - One matching longest network prefix is returned by the lookup function

Binary Trie

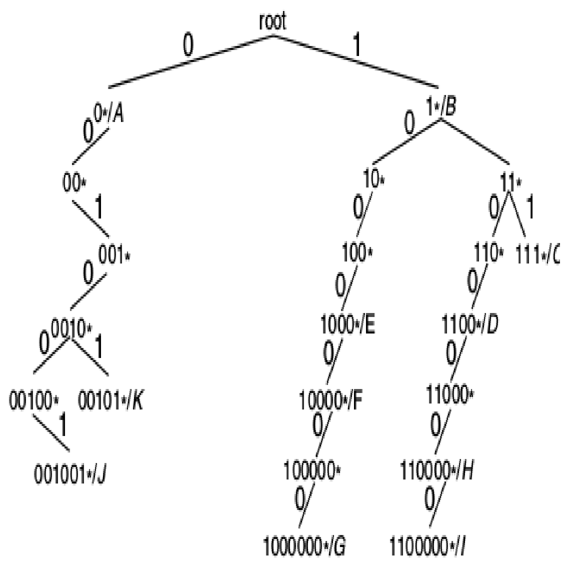
- Forwarding table organized as binary trie
  - Essentially a binary tree
- Each vertex at level *k* corresponds *k* bits prefix
- Each vertex has 2 children
  - *k* bit prefix expanded to (*k* + 1) bit prefix
- Route lookup essentially involves tracing 32-bit destination address in the trie to find the vertex
- The entry in the forwarding table that matches the longest prefix

Sample Forwarding Table & Representation

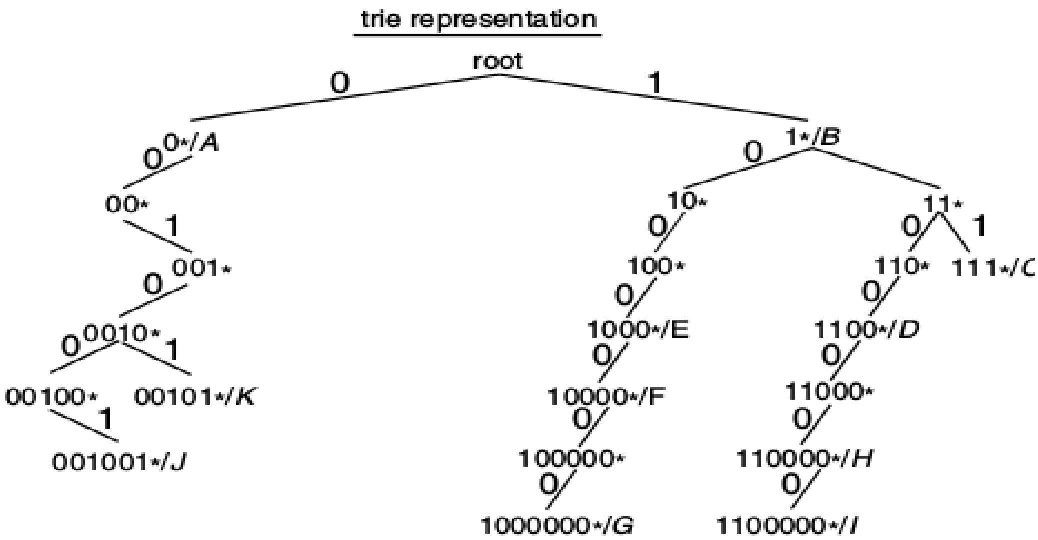
routing table

Net addr	Port
0*	*A*
1*	*B*
111*	*C*
1100*	*D*
1000*	*E*
10000*	*F*
1000000*	*G*
110000*	*H*
1100000*	*I*
001001*	*J*
00101*	*K*

trie representation



Trie Traversal for 1000000



Level-Compressed Tries
Traversal Time

- Binary Tree is a graph
- Complexity of depth-first traversals is O(n+m)
- Complexity then becomes O(n + n-1), which is O(n)

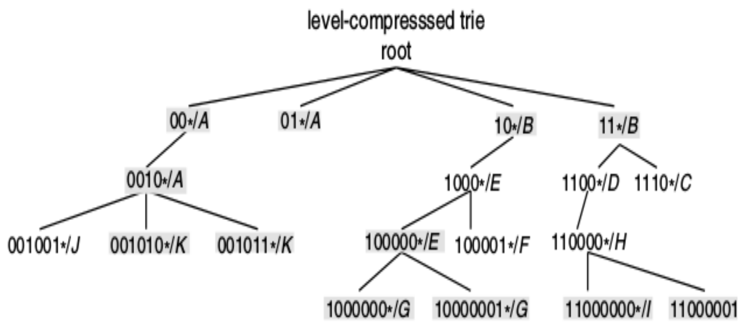
Level Compress

- Rather than define a level for each bit of the address
  - Define a level for groups of contiguous bits
- A simple case of level compression is to have a level for every *K* bits
- For N bits in address, then the number of levels is N/K
- Instead of two-way branch from each vertex of the trie 2 *K*-way branch
- Another view of level compression is to say that a subtree of height *k* is compressed into one level

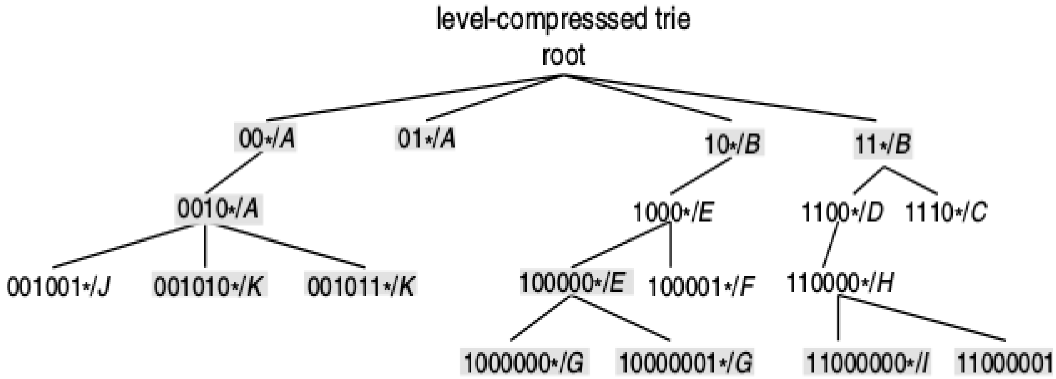
Level Compression or Prefix Expansion

routing table

Net addr	Port
0*	*A*
1*	*B*
111*	*C*
1100*	*D*
1000*	*E*
10000*	*F*
1000000*	*G*
110000*	*H*
1100000*	*I*
001001*	*J*
00101*	*K*



Trie (Retrieval) Traversal for 1000000



Flooding; ARPANET Algorithm
Usage of Flooding

- An algorithm whereby a node broadcasts a topological update message to all nodes
- Sending the message to its neighbors
  - Which in tum send the message to their neighbors, and so on

Indefinite Flood

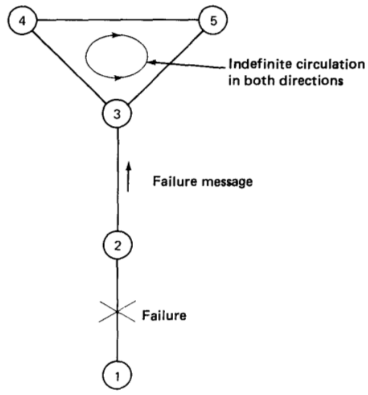
- Transmission of messages never terminates
  - Rule: node that receives a message relays it to all of its neighbors except from which it received

Level Compress

- Instead of two-way branch from each vertex of the trie
  - 2 *K*-way branch
- Another view of level compression is to say that a subtree of height *k* is compressed into one level

Indefinite Flood Problem

- A failure of link (1-2) is communicated to node 3 which triggers an indefinite circulation of the failure message along the loop (3,4,5) in both directions



ARPANET Solution

- Store enough information in update messages and network nodes
- To ensure that each message is transmitted by each node only a finite number of times
  - Preferably only once

- ARPANET used Sequence Numbers

Operation

- When a node *j* receives a message that originated at some node *i*
- Check if its seq no. > seq no. the message last received from *i*
- Yes: message stored in memory
- Transmit to all its neighbors except sender
- No: discard

Flooding w/o Periodic Updates
Redundancy of Periodicity

- Periodic updates needed because if some updates are sent but not incorporated
- Node crashes
- Transmission errors
  - Routing tables become inconsistent
- However under normal circumstances
  - Not needed

Need-based Updates

- Zero seq no allowed only when node is recovering from a crash
  - Situation where all of the node's incident links are down
  - And it is in the process of bringing links up
- Separate seq no. for each origin node

The Problem

- Link (2,3) goes down, then link (1,2) goes down, and then link (2,3) comes up while node 2 resets its sequence number to zero
- Nodes 2 and 3 exchange their (conflicting) view of the status of the directed links (1 ,2) and (2, 1)
- Both nodes discard each other's update message since it carries a sequence number zero which is equal to the one stored in their respective memories.

The Solution

- Depending on the lexicographic rule used
  - Either the (correct) view of node 2 regarding link (2, 1) will prevail right away
  - Or else node 2 will issue a new update message with sequence number 1 and its view will again prevail

Broadcast without Seq. Nos.
Redundancy of Periodicity

- Periodic updates needed because if some updates are sent but not incorporated
- Node crashes
- Transmission errors
  - Routing tables become inconsistent
- However under normal circumstances
  - Not needed

Need-based Updates

- Zero seq no allowed only when node is recovering from a crash
  - Situation where all of the node's incident links are down
  - And it is in the process of bringing links up
- Separate seq no. for each origin node

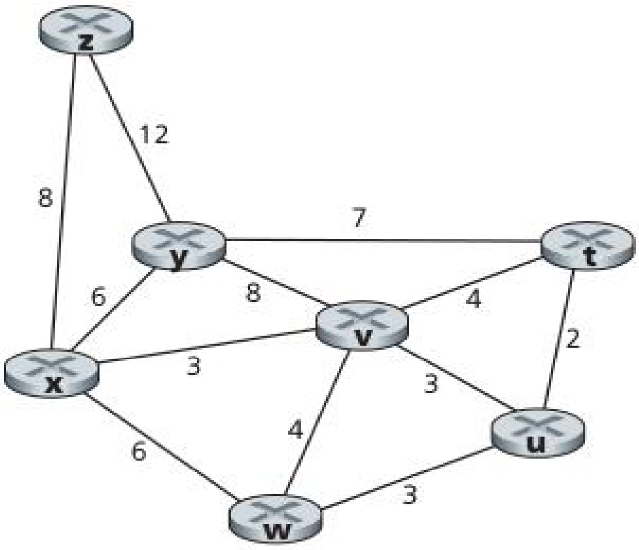
The Problem

- Link (2,3) goes down, then link (1,2) goes down, and then link (2,3) comes up while node 2 resets its sequence number to zero
- Nodes 2 and 3 exchange their (conflicting) view of the status of the directed links (1 ,2) and (2, 1)
- Both nodes discard each other's update message since it carries a sequence number zero which is equal to the one stored in their respective memories.

The Solution

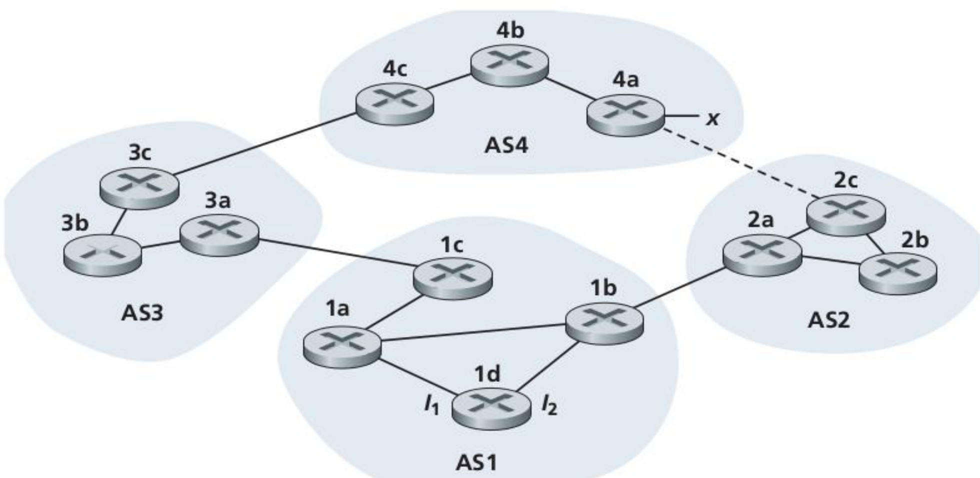
- Depending on the lexicographic rule used
  - Either the (correct) view of node 2 regarding link (2, 1) will prevail right away
  - Or else node 2 will issue a new update message with sequence number 1 and its view will again prevail

Problem Set 1
Topology for SPF Algorithm



Routing Algorithm Complexity
With the indicated link costs, use Dijkstra's shortest-path algorithm to compute the shortest path from x to all network nodes.

Inter-AS Connectivity



Operation of Inter-AS Routing Protocols
Consider the network. Suppose AS3 and AS2 are running OSPF for their intra-AS routing protocol. Suppose AS1 and AS4 are running RIP for their intra-AS routing protocol. Suppose eBGP and iBGP are used for the inter-AS routing protocol. Initially suppose there is no physical link between AS2 and AS4.

Operation of Inter-AS Routing Protocols
a. Router 3c learns about prefix x from which routing protocol: OSPF, RIP, eBGP, or iBGP?
b. Router 3a learns about x from which routing protocol?
c. Router 1c learns about x from which routing protocol?
d. Router 1d learns about x from which routing protocol?

Problem Set 1
Switching Fabric Performance in Routers
If the maximum queuing delay is $(n-1)D$ for a switching fabric n times faster than the input line rates. Suppose that all packets are of the same length, n packets arrive at the same time to the n input ports, and all n packets want to be forwarded to different output ports. What is the maximum delay for a packet for the (a) memory, (b) bus, and (c) crossbar switching fabrics?

Subnetting
Consider a subnet with prefix 128.119.40.128/26. Give an example of one IP address that can be assigned to this network.

Subnet Prefixes
Suppose an ISP owns the block of addresses of the form 128.119.40.64/26. Suppose it wants to create four subnets from this block, with each block having the same number of IP addresses. What are the prefixes (of form a.b.c.d/x) for the four subnets?

Fragmentation & Reassembly
Consider sending a 2400-byte datagram into a link that has an MTU of 700 bytes. Suppose the original datagram is stamped with the identification number 422. How many fragments are generated? What are the values in the various fields in the IP datagram(s) generated related to fragmentation?

Simulate QoS Routing
Operation of QoS Routing
Each class of traffic needs a minimum bandwidth path. To avoid oscillations by, QoS routing modifies the routing algorithms.

Key idea: Established routes continues to use the previous links till new paths (or links) are discovered

Assumptions

Let b_{min} = 50 kbps, b_{max} = 100 kbps, t = 1 min, w_1D = 0.2, w_2PLR = 0.3 and i=65 kbps

Then threshold s:

$s = \max [\{1 - 0.5\ (w1\ D + w2\ PLR)\}\ b_{max}, b_{min}]$

$s = \max [\{1 - 0.5\ (0.2 + 0.3)\}\ 100, 50] = 75$ kbps

Reserve the resources along the path equal to 75 kbps (s)

Else reserve the resources equal to 65 Kbps (i)

Otherwise reserve the resources equal to 50 Kbps (b_{min})

Decision

If resources are reserved equal to 75 kbps

when destination node receives data rate < 75 kbps for 1 min

It will notify the source

Source will establish a new route with data rate > s kbps

If no such route is available, a route with data rate > i or minimum bandwidth b_{min} condition will be set up

If resources are reserved equal to 75 kbps

when destination node receives data rate < 75 kbps for 1 min

It will notify the source

Source will establish a new route with data rate > s kbps

If no such route is available, a route with data rate > i or minimum bandwidth b_{min} condition will be set up

If resources are reserved equal to 75 kbps

when destination node receives data rate < 75 kbps for 1 min

It will notify the source

Source will establish a new route with data rate > s kbps

If no such route is available, a route with data rate > i or minimum bandwidth b_{min} condition will be set up

Support in INET

- Basic DiffServ support
- Current queue modules
  - DropTailQueue,
  - DropTailQoSQueue
  - REDQueue
- Classifier class: BasicDSCPClassifier
- classifyByDSCP() creates new packet classifiers

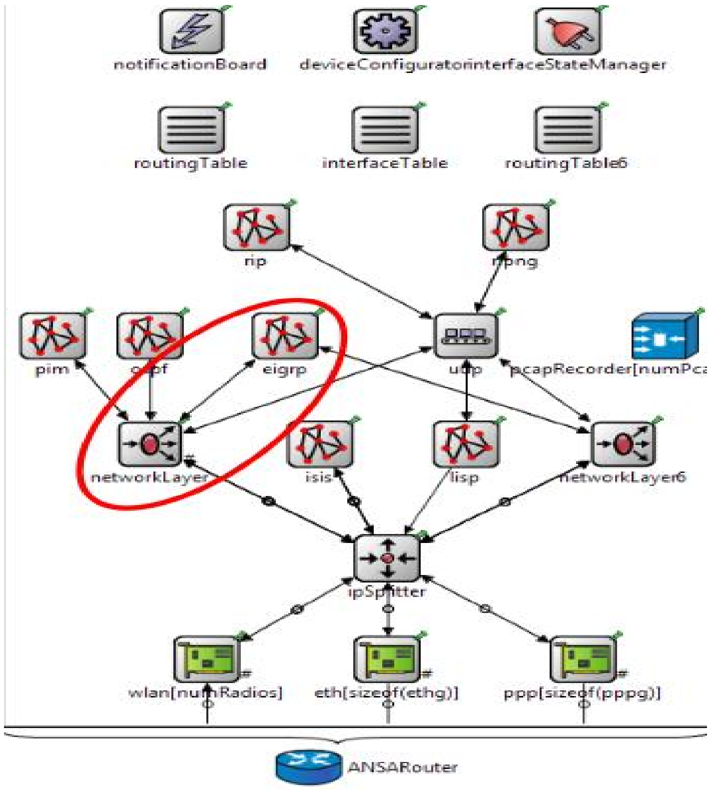
Simulate Routing Updates
EIGRP

- Cisco's EIGRP is a hybrid routing protocol between distance vector and link state routing protocols
- EIGRP offers routing based on composite metric
- Cisco released EIGRP specs as IETF's RFC draft in 2013

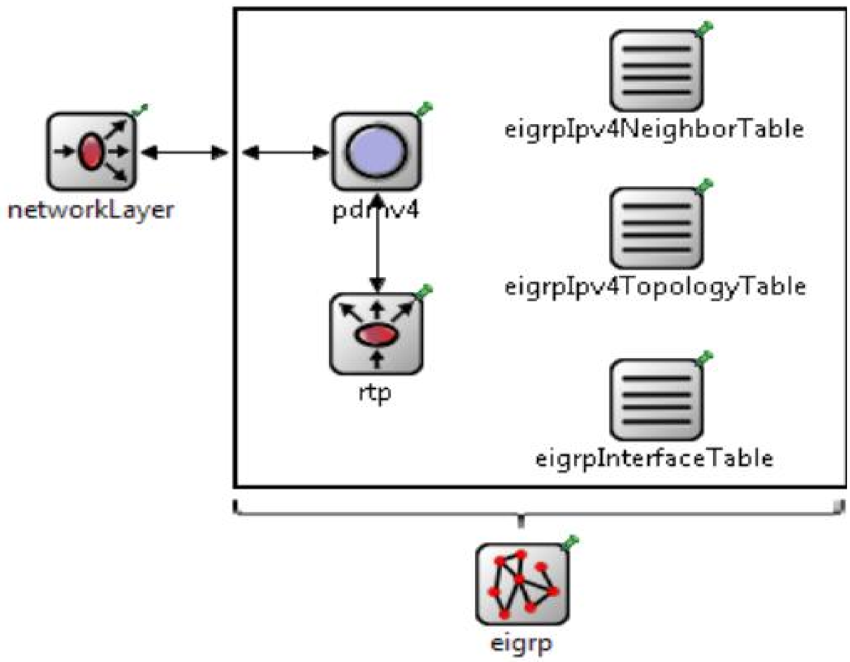
Basic Operation

- EIGRP employs Diffusing Update Algorithm (DUAL)
- Propagates topology change minimizing path compute time
- Sends event-driven partial bound updates
    - Weighted (Bandwidth + Delay)

ANSA
Automated Network Simulation and Analysis
@Brno University of Technology
Czech Republic



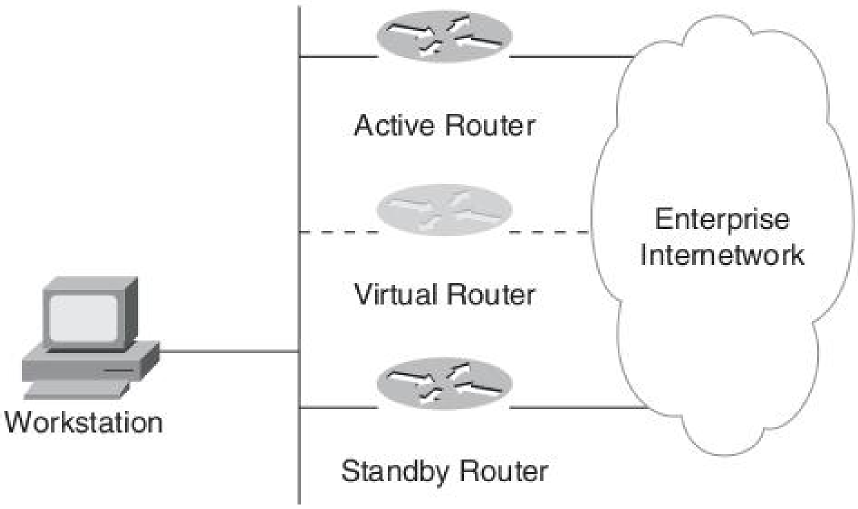
EIGRP Simulation Module Structure



Simulate HSRP
Background

- Allows PC to keep communicating on an internetwork even if its default gateway becomes unavailable
- Works by creating a virtual (phantom) router
  - Virtual router has its own IP and MAC addresses

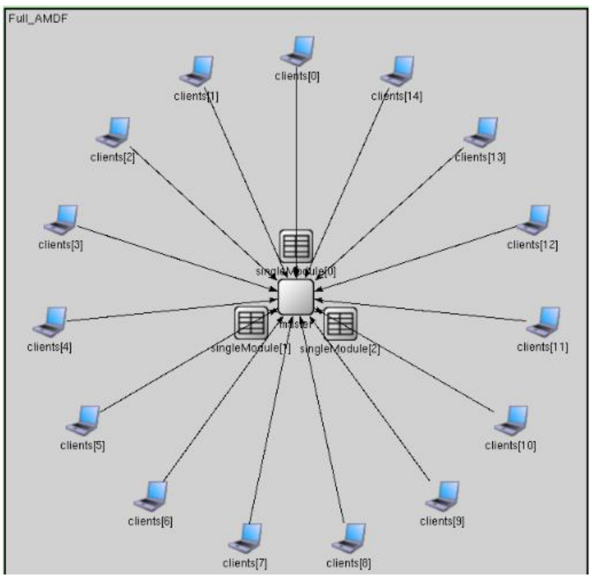
Hot Standby Router Protocol (HSRP)



Basic Operation

- Each PC is configured to use the virtual router as its default gateway
- When a PC broadcasts an ARP frame to find its default gateway, the active HSRP router responds with virtual router's MAC address
- Active router sends out HELLO periodically
- If the active router goes offline, a standby router takes over
- HSRP also works for proxy ARP

Automated Main Distribution Frame Housing Routers



Simulate Flooding
Message Complexity

- Flooding is a simple routing algorithm in which every incoming packet is sent through every outgoing link except the one it arrived on
- Complexity
  - $M = \Omega/(N-1)$

Displaying no. of packets sent/received

- No. of messages at each node
- tictoc14.ned
- txc14.cc
- tictoc14.msg

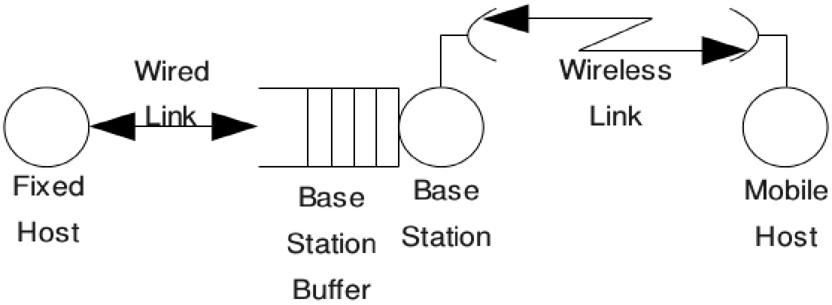
Txc14.cc

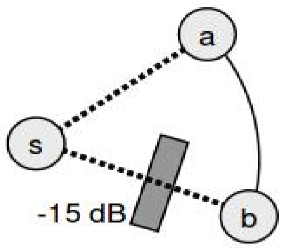
```
class Txc14 : public cSimpleModule
{
  private:
    long numSent;
    long numReceived;
  protected:
 virtual void updateDisplay();

  void Txc14::initialize()
{
  // Initialize variables
  numSent = 0;
  numReceived = 0;
  WATCH(numSent);
  WATCH(numReceived);
```

Declared

set to zero & WATCH'ed in initialize() method

Simulate TCP with BER
Reference Topology





TCP with BER

- A cross layer paradox
- TCP is for congestion control
- Packet loss due to PHY layer
  - BER
- TCP wrongly interprets
  - Goes into starvation

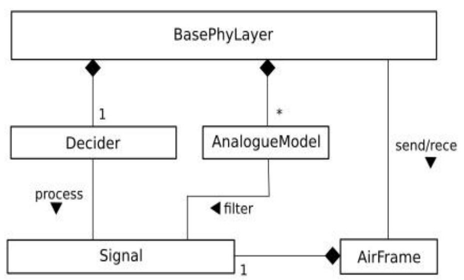
Line of Sight effect

- An object within the line-of-sight between two nodes s and b yields a weaker received signal than that of a non obstructed pair s and a at the same distance

Support in Mixim

- Decider module
  - Classifies incoming messages into receivable messages or noise
  - Calculates the bit errors for the message
  - Info. about current state of channel

PHY Layer Class Graph



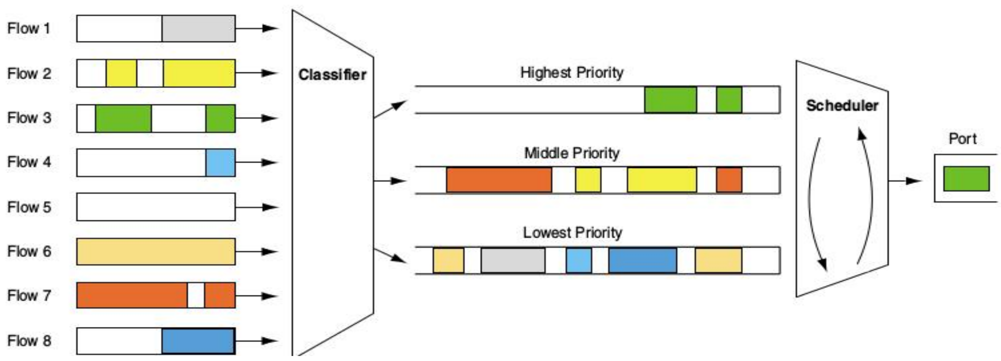
Simulate Priority Queues
Recall!

- Designed to provide a relatively simple method of supporting differentiated service classes
- Cqueue provides FIFO by default
- Need to be modified for priority queuing

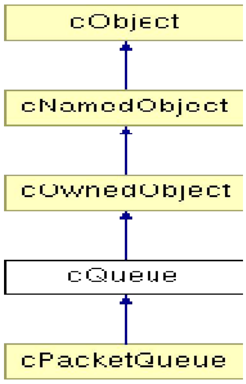
Operation

- Packets classified and placed into different priority queues
- Packets scheduled from the head of a queue only if all queues of higher priority are empty
- Within each of the priority queues, packets are scheduled in FIFO order

Priority Queuing with Classifier



Support in INET



Member Functions

- simple PriorityQueue extends Queue

{   @class(PriorityQueue);
}

- void setSchedulingPriority(short p);

DLL Services
Need

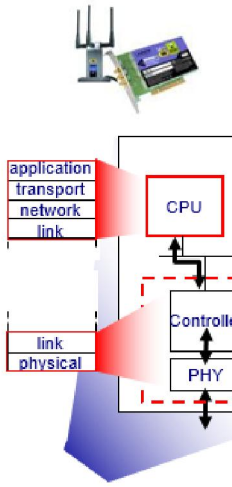
- The datalink layer is to the link what the transport layer is to the path
- Upper layer necessitates its behaviour
  - Reliability
  - Flow control
  - Error control
- Corresponding services must exist

Services Models

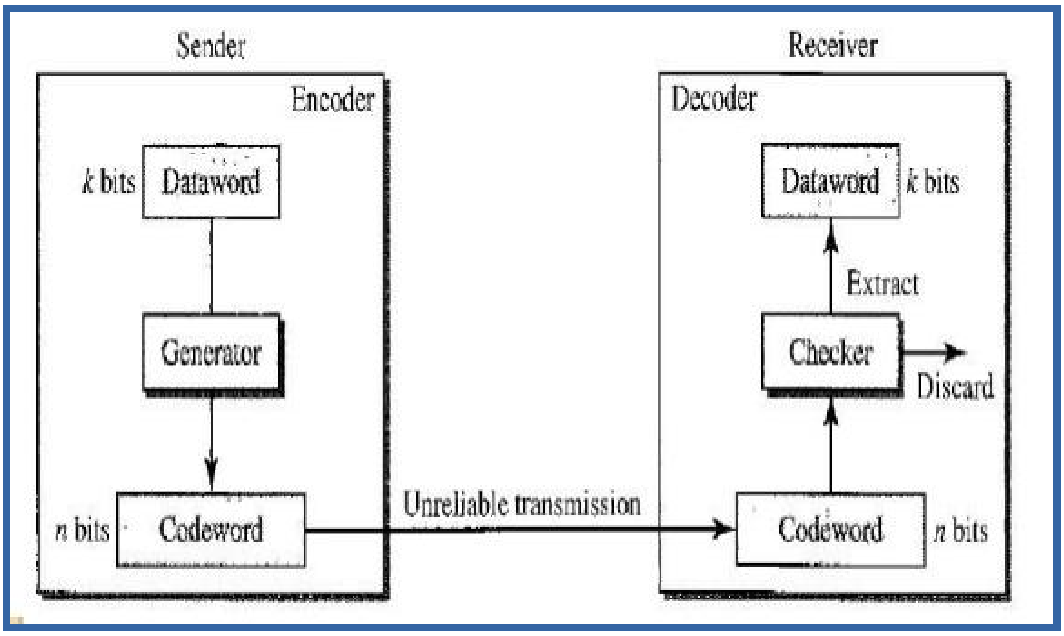
- Services offered
  - Reliable (PPP)
  - Unreliable (Ethernet)
- Point to point
- Multiaccess

Services

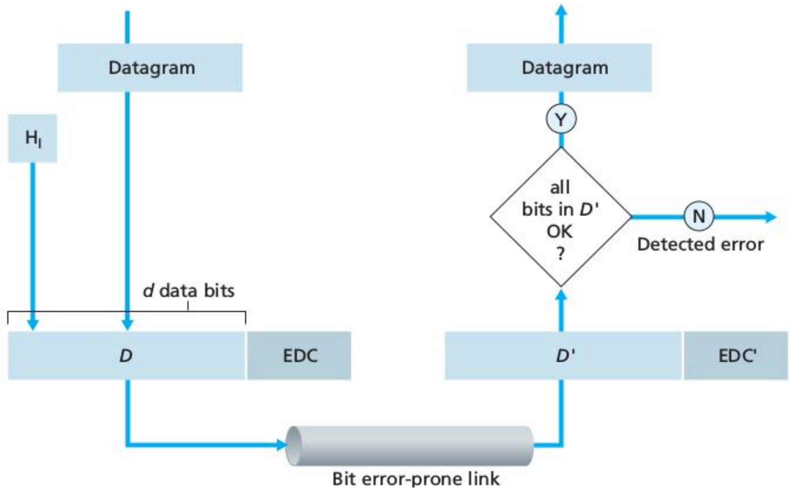
- Framing
- Link access
- Error control
- Contention control



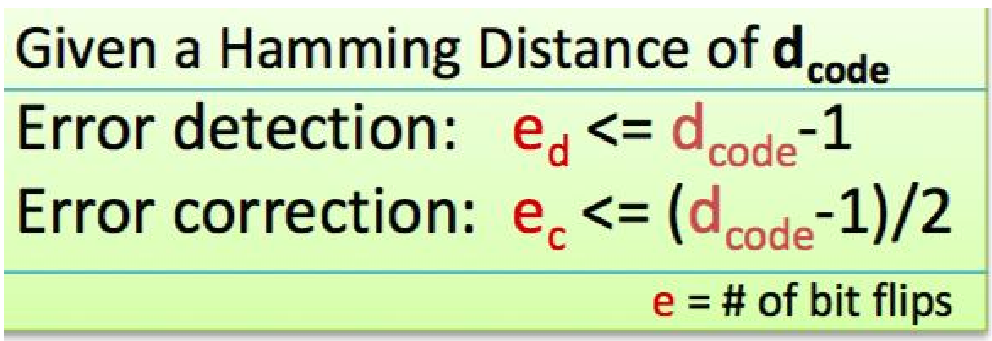
EDEC Techniques
Block Diagram



Strategy



Capabilities of EDEC

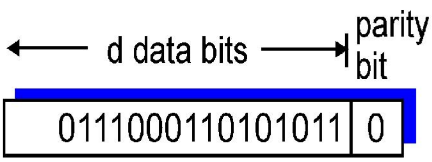


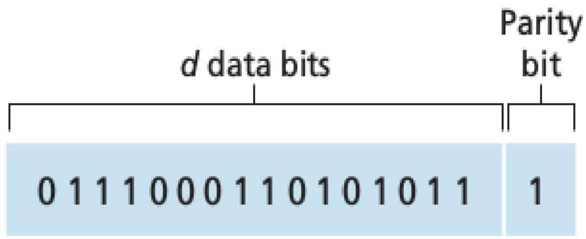
Constraints

- All EDEC methods only work below a certain error rate
- If we allow any no. of errors in data bits and in check bits, then no EDEC method can guarantee to work
    - Any valid pattern can transform into any other valid pattern

Parity Checks
Operation

- Single bit parity detect single bit errors
    - Even
    - Odd





Limitations

- Probability of undetected errors in a frame protected by single-bit parity
    - can approach 50 percent
- Burst errors cause such nondetections

Checksumming at DLL
Overhead of Parity schemes

- Single bit parity schemes provide little protection
- To provide enough resilience, redundancy increases linearly
- Solution:
    - Treat data as k-bit integers
    - Generate k-bit overhead

Operation

- RFC 1071 addresses Internet checksum algorithm
- 1s complement of all sums of k-bit integers forms the Internet checksum
    - 16-bit for TCP/UDP
- Carried in the segment header

Variants

- TCP and UDP: checksum computed over all fields
    - Header + data
- IP: IP header
- XTP: one checksum is computed over the header and another checksum computed over entire packet.

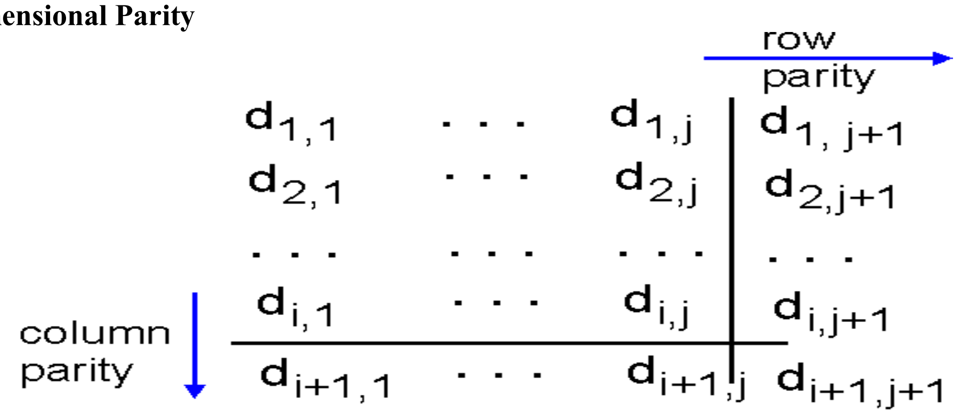
DLL vs Transport

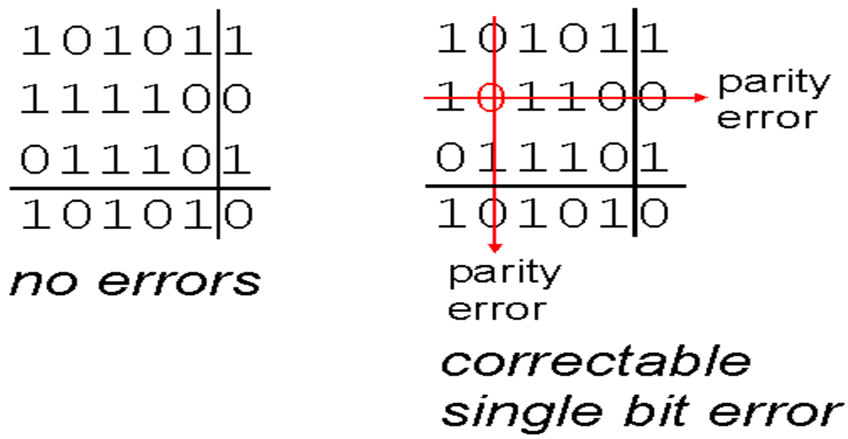
- Transport layer is typically implemented in software
- Error detection has to be simple and fast
    - Checksumming
- DLL implemented in NIC
    - CRC is more robust

Horizontal & Vertical Parity
2D Generalization

- d bits in D are divided into i rows and j columns
- Parity value computed for each row and for column
- i + j + 1 parity bits comprise DLL frame's error-detection bits
    - 17 bits for 64 bits
        - ~27%

Two-Dimensional Parity





Error Correction

- A single error is detectable
    - And correctable
- Even an error in the parity bits themselves is also detectable and correctable
    - Forward error correction (FEC)

Limitations

- Two-dimensional parity can also detect (but not correct!) any combination of two errors in a packet

Cyclic Redundancy Check
Principle

- Checksum becomes weak
    - Limited illegal rep
- CRC more powerful error-detection code
    - Views data bits, D, as a binary number
    - Choose r+1 bit G
    - Goal: choose r CRC bits, R, so <D,R> exactly divisible by G (modulo 2)
- Receiver knows G,
- Divides <D,R> by G
- All zeros
- No error
- If non-zero remainder: error detected!

Modulo 2

- Modulo-2 arithmetic
- Addition & subtraction are identical
- Both equivalent to bitwise exclusive-or (XOR) of operands
- 1011 XOR 0101 = 1110
- 1001 XOR 1101 = 0100

Operation

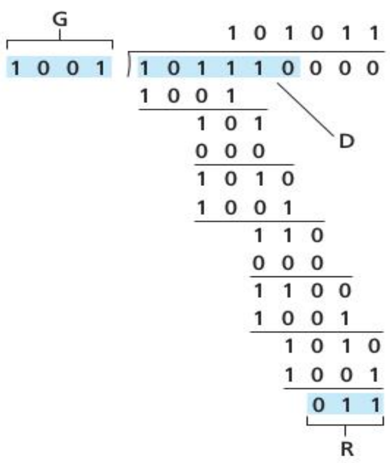
- $D \cdot 2^r$ XOR R = nG
    - Left shift by r then append R
    - Multiple of Generator

Mathematical manipulation

- $D \cdot 2^r$ = nG XOR R
- If we divide $D \cdot 2^r$ by G, want remainder R to satisfy

$$R = \text{remainder}\left[\frac{D \cdot 2^r}{G}\right]$$

D = 101110, d = 6, G = 1001, r =3

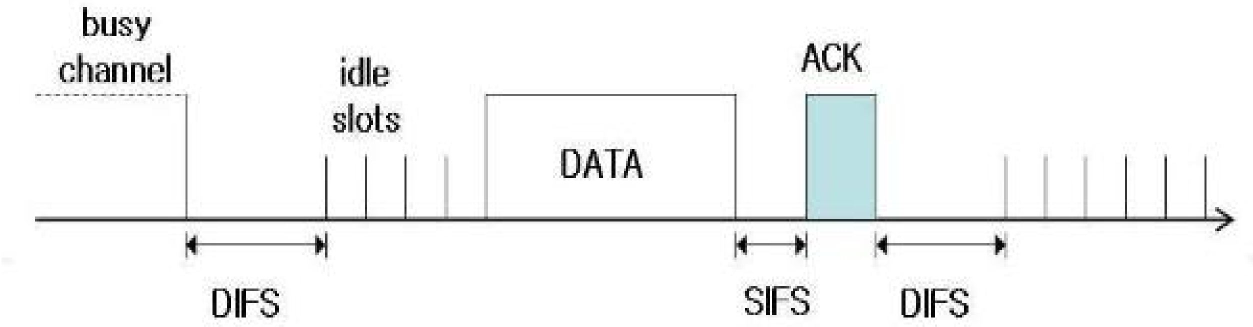


9 bits transmitted in this case are 101110 011

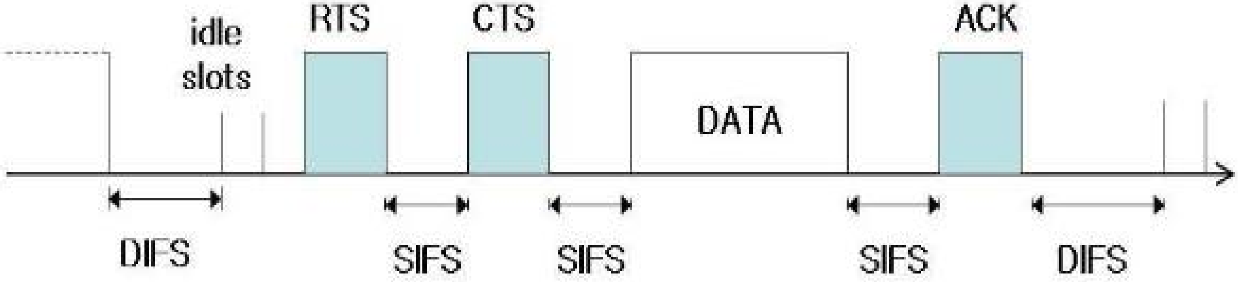
Throughput of MAC
Ideal MAC

- Broadcast channel of rate R bps
- When one node wants to transmit, it can send at rate R
    - M nodes transmit
    - Each sends at average rate R/M
- Fully decentralized
- No special node to coordinate TXNs
- No synchronization of clocks
- No slots
- Simple

Access Methods



(a) Basic access method



(b) Four-way handshaking access method

Analysis

- Effective throughout depends upon various factors
    - No. of active users
    - No of resources
    - Channel access methods
    - Traffic volumes
- Probabilistic in nature

Channel Partitioning
Basic Idea

- Divide channel into smaller “pieces”
    - Time slots
    - Frequency
    - Code
    - Space
- Exclusive use

TDM

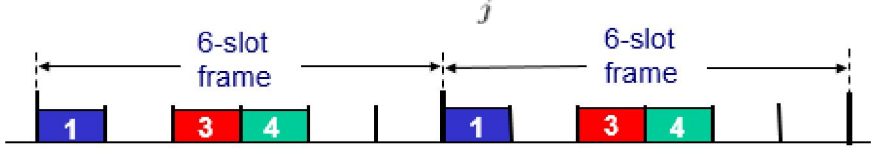
- Time divion multiplexing
- Access to channel in "rounds"
- Each station gets fixed length slot
- Length = packets trans time) in each round
    - Unused slots go idle

TDM Example

Example: 6-station LAN, 1,3,4 have pkt, slots 2,5,6 idle

- Fraction of time slots being used
    - Depends upon the frame size

$$f = \sum_j \alpha_j x_j$$



FDM

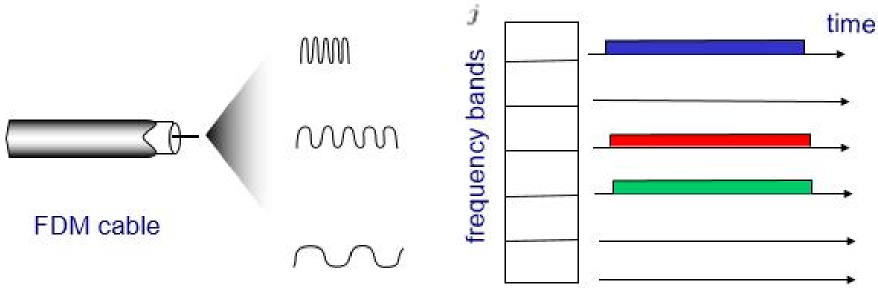
- Frequency divion multiplexing
    - Channel spectrum divided into frequency bands
- Each station assigned fixed frequency band
    - Unused transmission time in frequency bands go idle

FDM Example

Example: 6-station LAN, 1,3,4 have pkt, frequency bands 2,5,6 idle

- Fraction of frequency bands being used
    - Depends upon the transmission times of each user

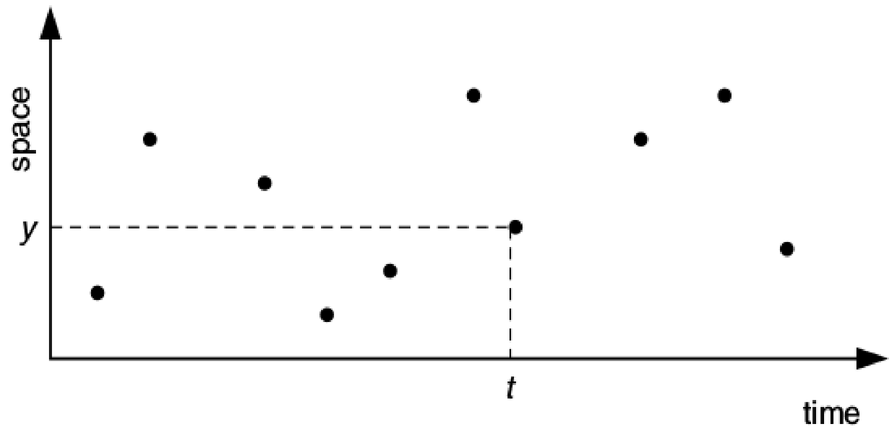
$$f = \sum_j \alpha_j x_j$$



Random Access Protocols
Basic Idea

- When node has packet to send
  - transmit at full channel data rate R
  - No a priori coordination among nodes
- Collisions are legal
  - Two or more transmitting nodes cause collision

Packet attempt instants in space and time



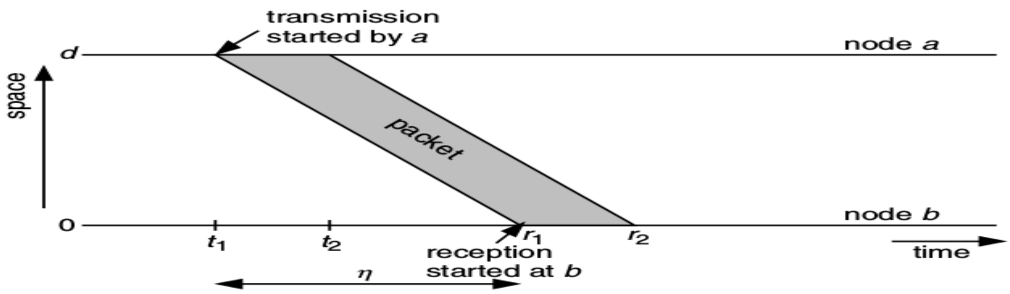
Managing Collisions

- How to detect collisions?
  - Voltage change
- How to recover from collisions?
  - Wait &
  - Retransmit

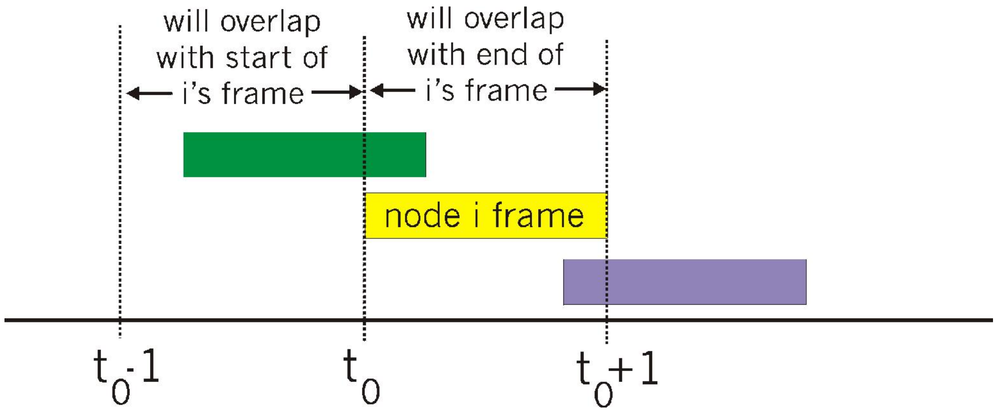
ALOHA
Basic Idea

- Just say as you like!
- Whenever and wherever
  - Simplest
  - No synchronization

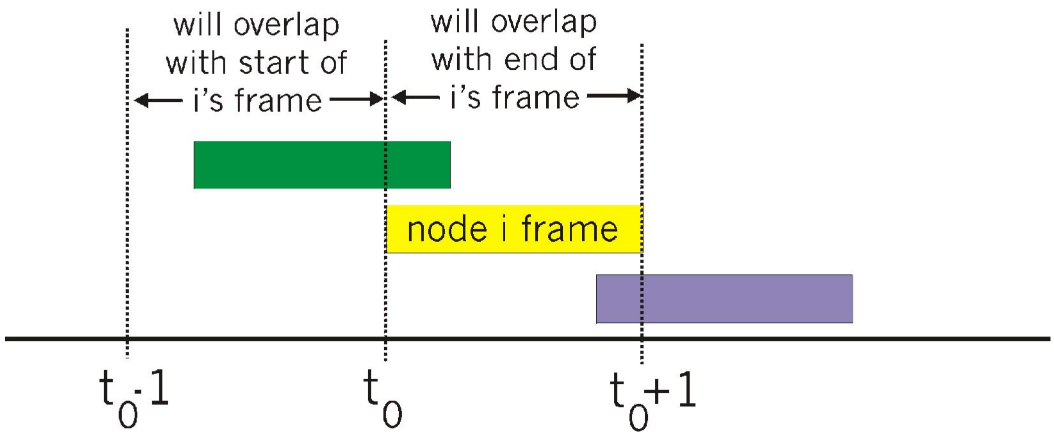
Packet transmissions are independent
Packet reception success dependent upon others not transmitting



Probability of Success



P(success by given node) = P(node transmits) * P(no other node transmits in [t_0-1,t_0] * P(no other node transmits in [t_0-1,t_0]



$p \cdot (1-p)^{N-1} \cdot (1-p)^{N-1}$
$= p \cdot (1-p)^{2(N-1)}$
[choosing optimum p and n very large]
= 1/(2e) = .18

Slotted ALOHA
Basic Idea

- Minimize collisions
  - Through synchronization
  - Through frame size delimiting

Assumption

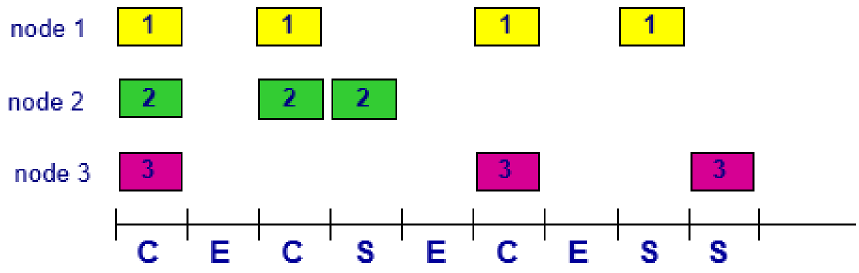
- All frames same size
- Time divided into equal size slots
  - Time to transmit 1 frame
- Nodes start to transmit only slot beginning
- Nodes are synchronized
  - If 2 or more nodes transmit in slot, all nodes detect collision

Operation

- when node obtains fresh frame, transmits in next slot
  - *if no collision:* node can send new frame in next slot
  - *if collision:* node retransmits frame in each subsequent slot with prob. p until success

Performance of Network with 3-Nodes

- 30% success
- How many collisions?
- How many empty slots?



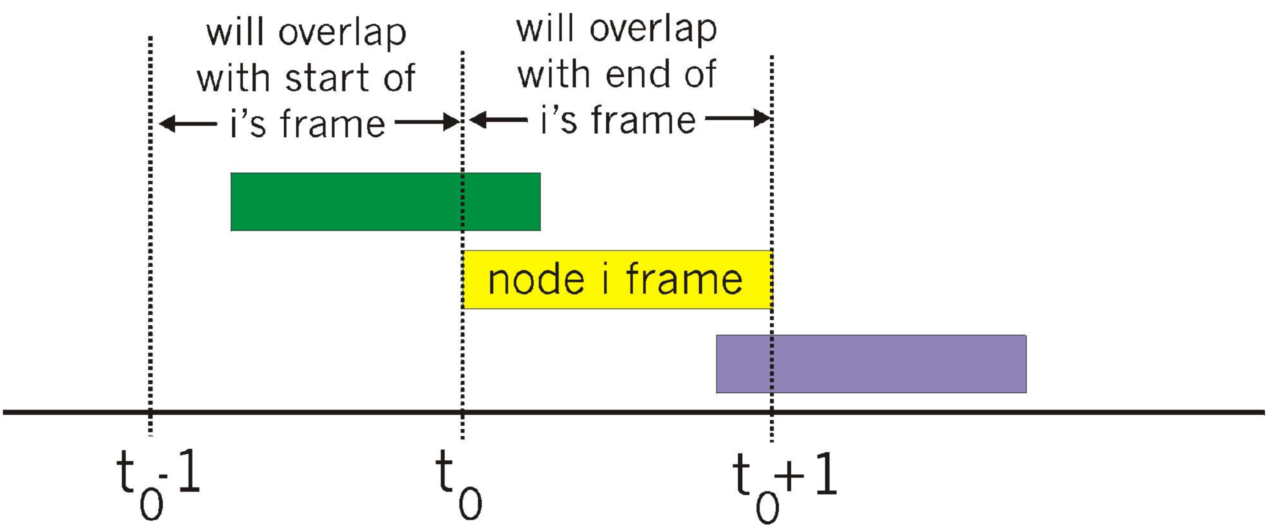
Pros

- Single active node can continuously transmit at full rate of channel
- Highly decentralized: only slots in nodes need to be in sync (master clock)
- Simple to implement

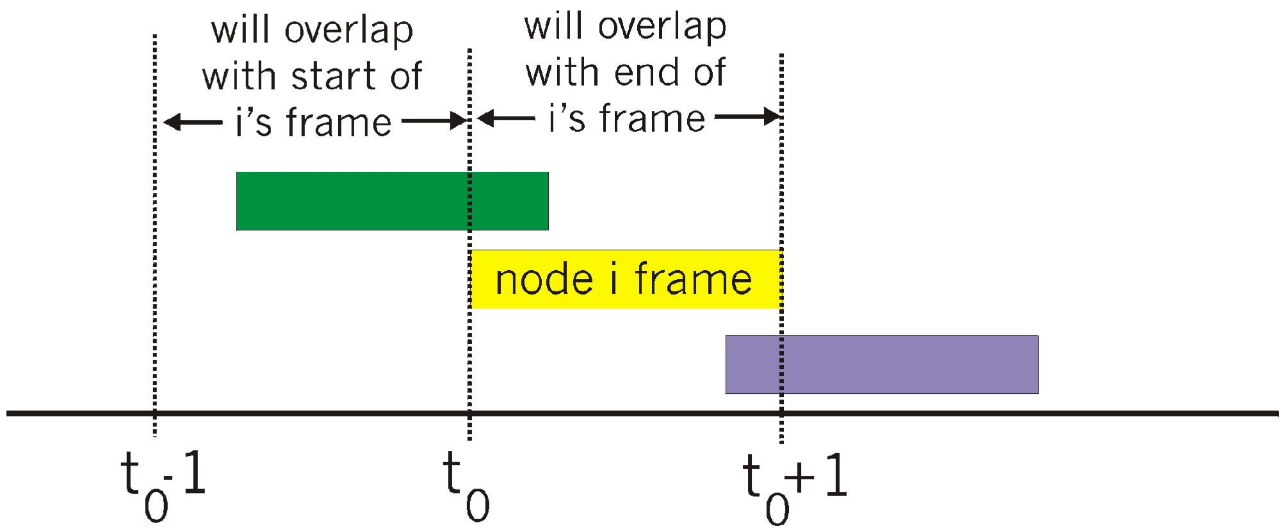
Cons

- Collisions, wasting slots idle slots
- Nodes may be not able to detect collision in time
- Clock synchronization needed

Probability of Success



- N nodes with many frames to send, each transmits in slot with probability p
- Prob that given node has success in a slot $= p(1-p)^{N-1}$
- Prob that *any* node has a success $= Np(1-p)^{N-1}$

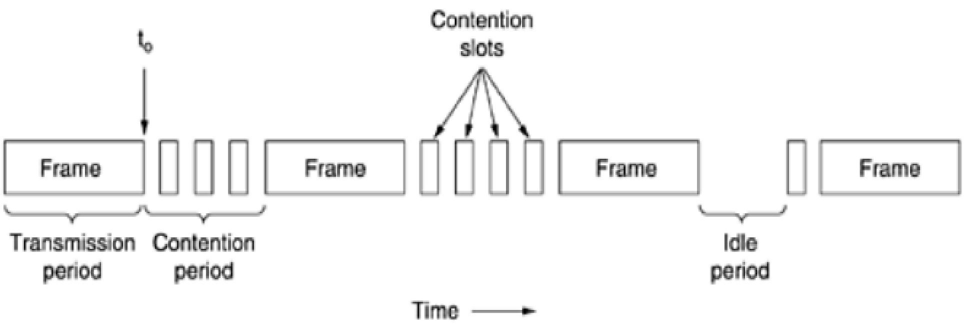


- Max efficiency: find p* that maximizes $Np(1-p)^{N-1}$
- for many nodes, take limit of $Np^*(1-p^*)^{N-1}$
  — N goes to infinity
- Max efficiency = *1/e* = .37

CSMA/CD
Basic Idea
Carrier Sensing
- Listen before transmit
- If channel sensed idle
  - Transmit entire frame
- If sensed busy
  - Defer transmission

Collisions Detection
  - Within short time
- Colliding transmissions aborted
- Reduces channel wastage

CSMA/CD States
- Contention
- Transmission
- Idle



Binary (Exp) backoff

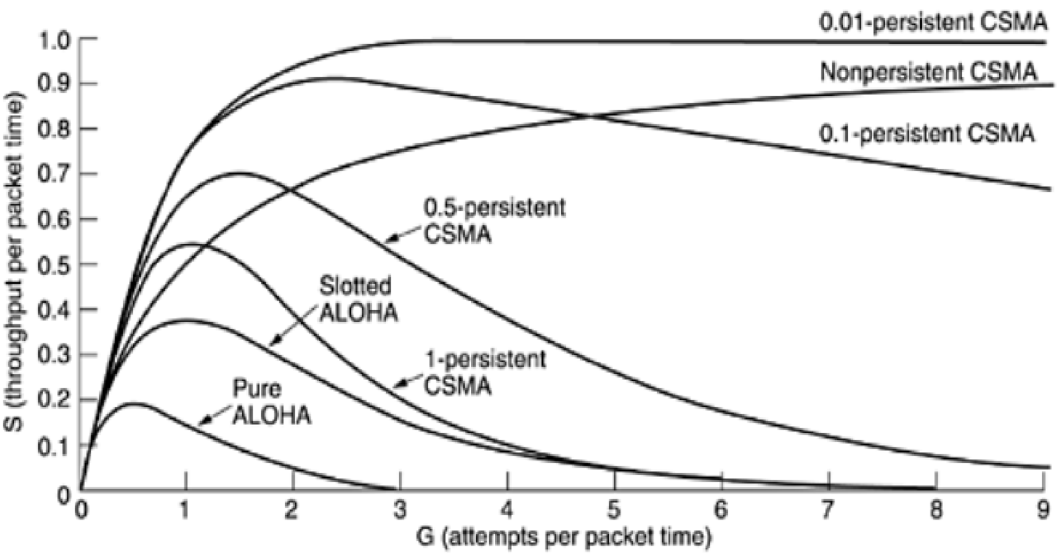
- After *m*th collision, NIC chooses *K* at random from $\{0,1,2, \ldots, 2^m-1\}$
- NIC waits K·512 bit times, returns to Step 2
  - If idle, start trans
  - If busy wait until idle, then transmits
- Longer backoff interval with more collisions

CSMA/CD Efficiency
Factors Affecting (1 of 2)

- T_{prop} = max prop delay between 2 nodes in LAN
- t_{trans} = time to transmit max-size frames
- Full load
  - Worst
- Partial load
  - Increases till a range
- No load
  - Poor performance
- Efficiency goes to 1
- As t_{prop} goes to 0
- As t_{trans} goes to infinity
- Efficiency goes to 0 vice versa

$$Efficiency = \frac{1}{1 + 5t_{prop}/t_{trans}}$$

Performance with Variants

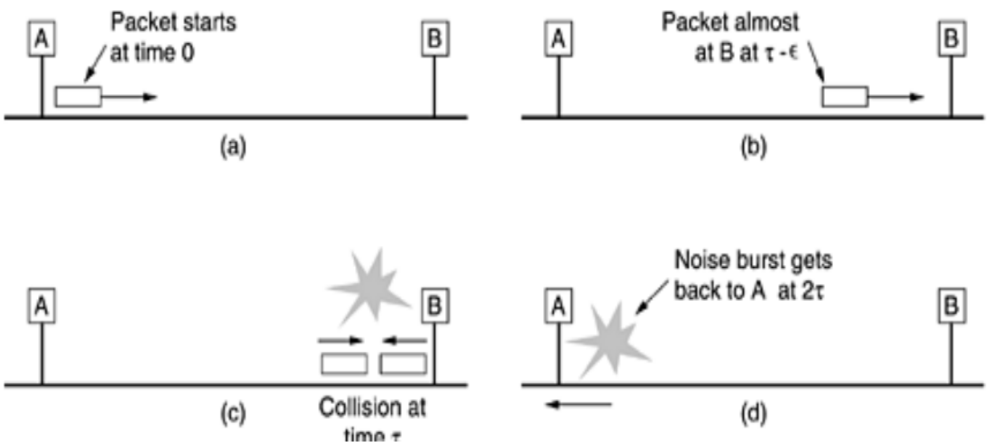


Min Frame Size Computation
Min Frame Size

- Ethernet recommends 64 Bytes
  – Inclusive of headers
- If the data portion of a frame < 46 bytes
- Pad field is used to fill out the frame to min. size
- Wireless networks necessitate lesser size
  – ReTx cost to be minimized
- Optical networks require longer frames

Reasons

- Data field of 0 bytes is sometimes useful
  – When a transceiver detects a collision, it truncates the current frame
- Stray bits and pieces of frames appear on the cable all the time
- To distinguish valid frames from garbage
- Collision detection can take as long as 2 τ



Reasons

- Prevent a station from completing the transmission of a short frame before the first bit has even reached the far end of the cable
  – where it may collide with another frame

Ethernet Calculation

- 10-Mbps LAN
- Max length = 2500 m (four repeaters: 802.3 specs)
- RTT = 50 µsec in the worst case
  — Therefore, the minimum frame must take at least this long to transmit
- At 10 Mbps, a bit takes 100 nsec
  — 500 bits is the smallest frame that is guaranteed to work
- To add some margin of safety, round up to 512 bits or 64 bytes

Max Frame Size Computation
Background

- Ethernet recommends 1500 bytes as Max
- This limit was chosen arbitrarily for DIX standard
- Transceiver needs enough RAM to hold an entire frame
  – More expensive transceivers

Factors

- Overhead
- Pipelining
- Transmission errors

Overhead of Variable Length Packet

- Each frame contains V as overhead bits
- K_{max}:Max length of packet
- Message of length M
  - Broken down into M/K_{max} packets

$$\text{total bits} = M + \left\lceil \frac{M}{K_{max}} \right\rceil V$$

Overhead of Variable Length Packet (2 of 2)

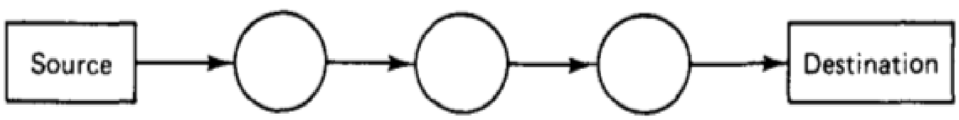
- As K_{max} decreases, the number of frames increases
  - Thus the total overhead in the message
- Processing load in a multiple hop network increases

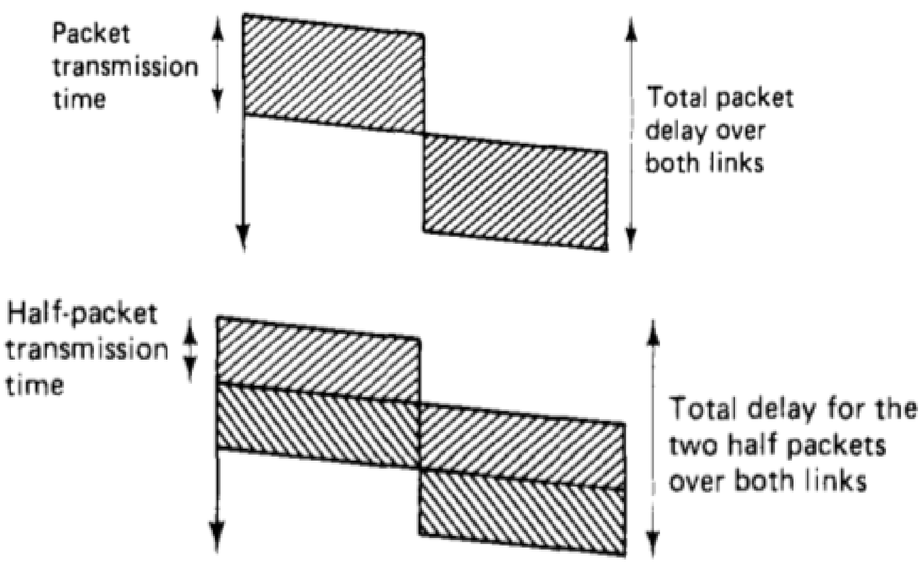
Pipelining

- Split the message into smaller packets
  - While the later packets arrive on the input queue of the node
  - Former packets are leaving or may have already left the output queue

Pipelining Scenario

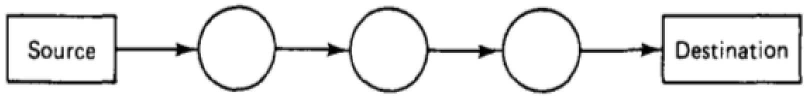
- Decreasing delay by shortening packets to take advantage of pipelining

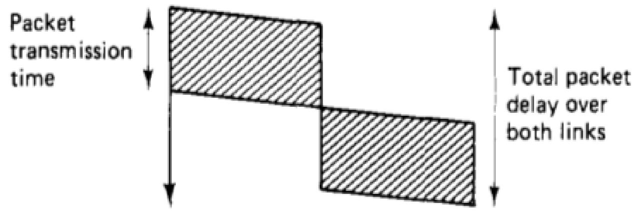




Pipelining Scenario

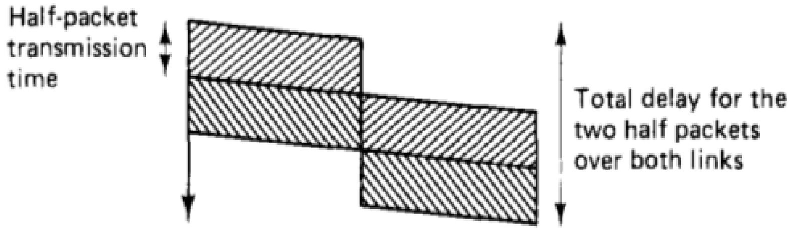
- The total packet delay over two empty links equals twice the packet transmission time on a link plus the overall propagation delay





Pipelining Scenario

- When each packet is split in two, a pipe lining effect occurs
- The total delay for the two half packets equals 1.5 times the original packet transmission time on a link plus the overall propagation delay



Tradeoff between Overhead & Pipelining

- As the overhead V increases, K_{max} should be increased
- As the path length j increases, K_{max} should be reduced

Transmission Errors

- Large frames have a somewhat higher error probability than small frames
- Probability of error on reasonable-sized
- frames is on the order of 10^{-4} or less
  - This effect is typically less important than the other effects

Fixed Frame Size Computation
Why Fixed Frame?

- Expectability of performance
  - Latency
  - Throughput
  - Cell loss
- Resource pre-emption

Considerations

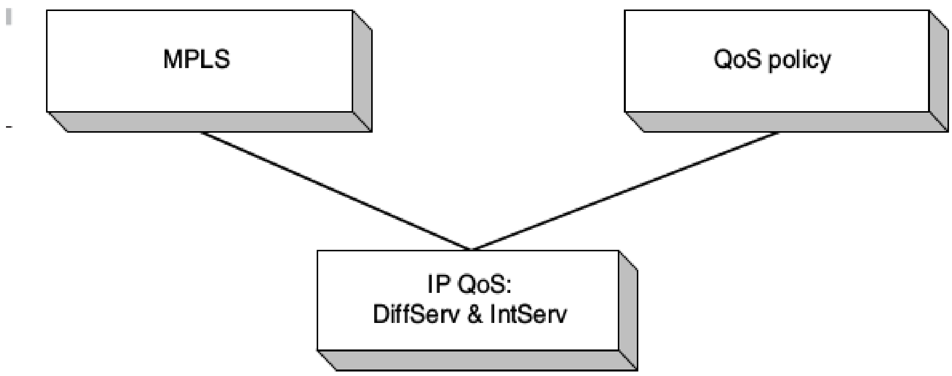
- How much should be the fixed size?
  - Processing at the nodes
- Header to payload efficiency
  - Padding requirement
- Applications (Voice/video)
  - Achieve a small delay for stream-type traffic
- Assume an arrival rate of R and a packet length K
  - First bit in a packet is then held up for a time K/R
  - Waiting for the packet to be assembled

Fixed Frame (cell) networks

- ATM recommends 53 bytes (424 bits) as Max
  - 48 bytes payload
  - 5 bytes header
- Emulates circuit-like behaviour
  - Good for interactive
  - Bad for file transfer

Multi-Protocol Label Switching
Intro

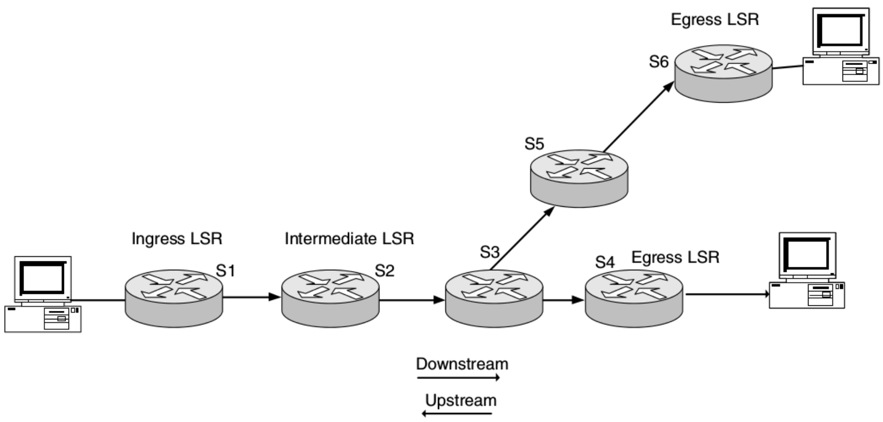
- Multi Protocol Label Switching (MPLS)
  - Fast packet switching & routing
  - Provides designation, routing, & switching of traffic flows through MPLS domain
- All packets labelled before being forwarded
- Network layer header not processed
- Although idea was to facilitate fast packet switching
- Main goal: support traffic engineering and QoS



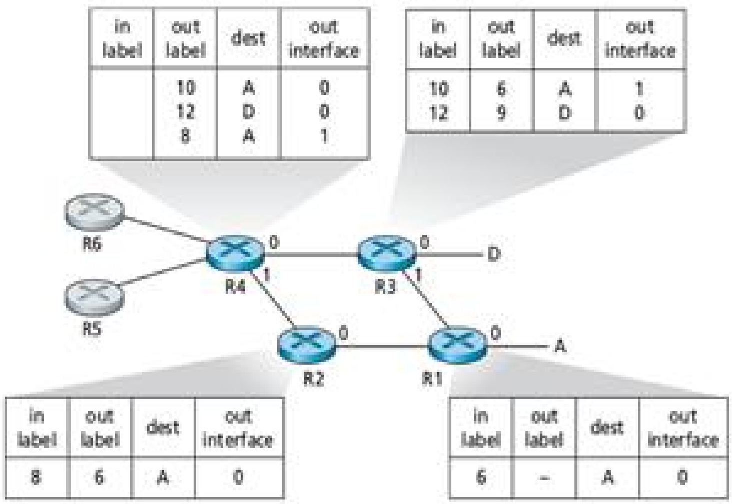
Basic Idea

- Route once and switch many times
- Set of packets that have the same traffic characteristics are forwarded in the same manner
  - Along the route that starts from an ingress node and ends at an egress node of an MPLS network

MPLS Network Components



MPLS Enhanced Forwarding



Important Parameters

- Link utilization
- Voice jitter
- End to end delay
- Traffic Received when FRR vs link failures

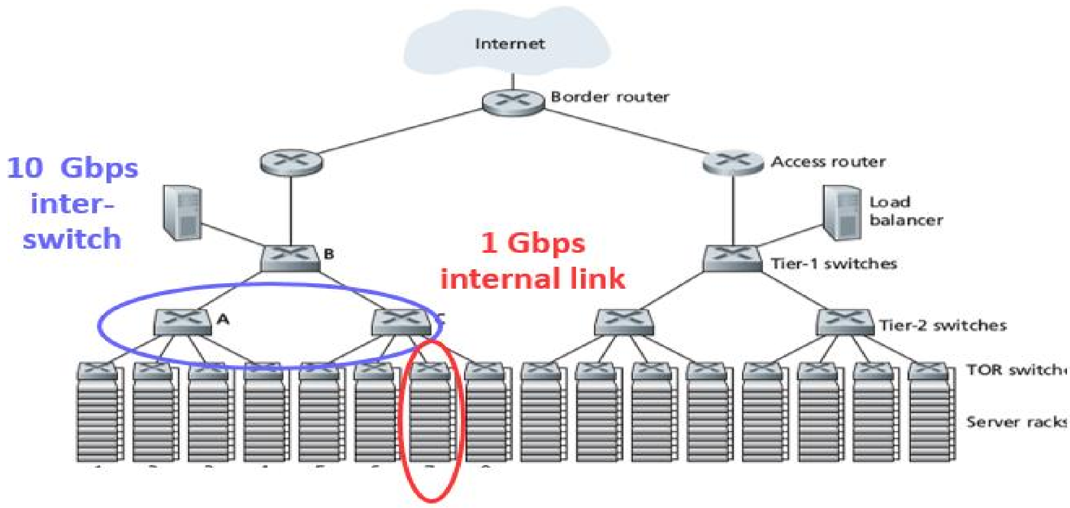
Load Balancing in Data Centre
Intro

- Google, Microsoft, Facebook, and Amazon have built massive data centers
- Each houses tens to hundreds of thousands of hosts
- Concurrently support many distinct cloud applications
- Search, email, social networking, and e-commerce
- Top of Rack (TOR) switch interconnects the hosts in the rack
- With each other
- With other switches in the data center
- Form a data centre network

Basic Idea

- Each application is associated with a publicly visible IP address
- Clients send their requests and receive responses
- Inside, external requests first directed to load balancer
- Distributes and balances requests to hosts
  - Also called L4 switch (with NAT)

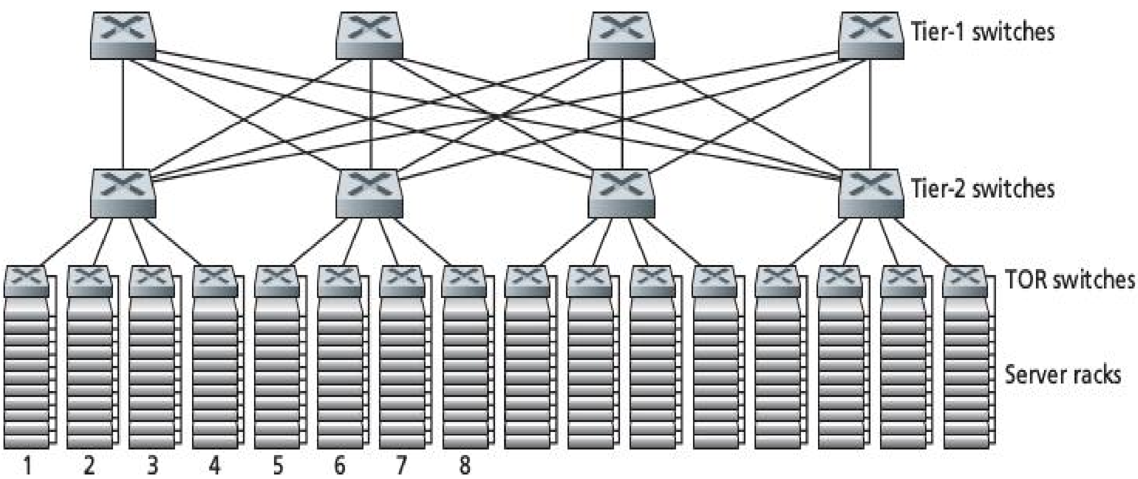
Problems in Hierarchical Topology



Limited Host to Host Connectivity (1 of 2)

- 40 simultaneous flows between 40 pairs of hosts in different racks
  - 10 hosts in rack 1 sends a flow to a corresponding host in rack 5
  - 10 flows between pairs of hosts in racks 2 and 6, 3 and 7, and 4 and 8
- 40 flows crossing the 10 Gbps A-to-B link (and B-to-C link) each only receive 10 Gbps / 40 = 250 Mbps

Solution: Fully Connected Topology

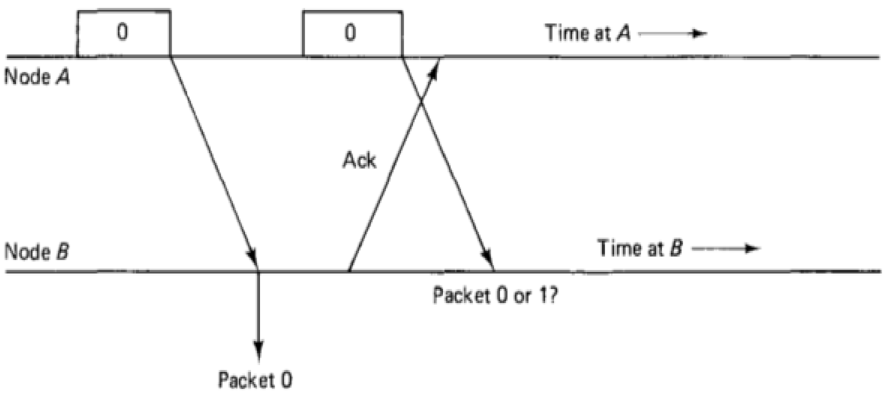


Correctness of Stop and Wait

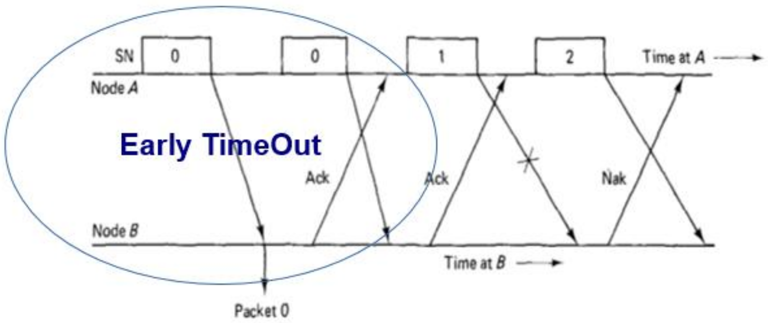
Stop and Wait Operation

- Client sends request
  - Waits for response
- Server sends response
  - Waits for ack
- Step-locked communication
- Most web and other servers based upon it
  - Pipelining is deviation

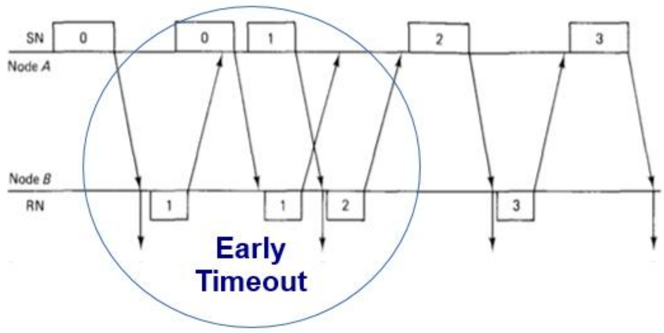
Problems with Unnumbered Packets



Problems with Unnumbered ACKs



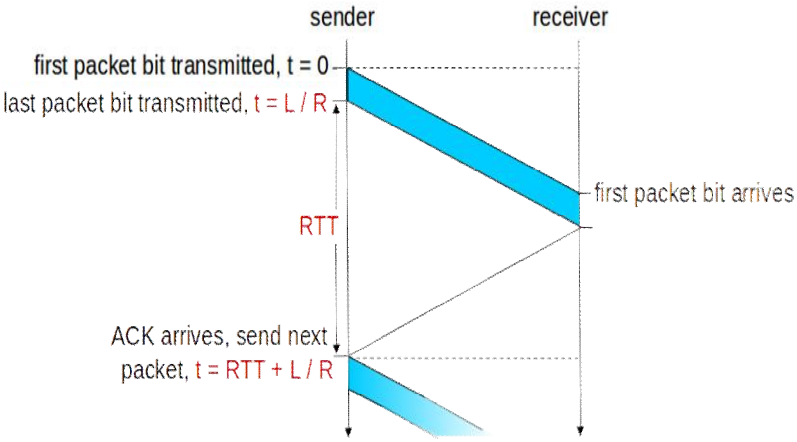
Problems Management using Seq. Nos.



Efficiency of Go Back N
Stop and Wait Operation

- Client sends request
  - Waits for response
- Sends next request
- Each request travels all along the way to server
- Response travels backwards

Efficiency of Stop & Wait Operation

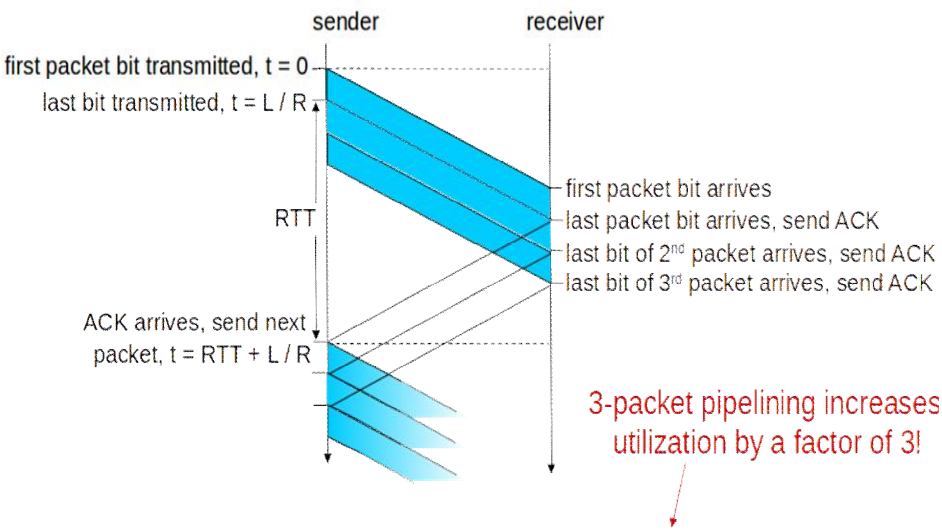


$$D_{trans} = \frac{L}{R} = \frac{8000 \text{ bits}}{10^9 \text{ bits/sec}} = 8 \text{ microsecs}$$

$$U_{sender} = \frac{L/R}{RTT + L/R} = \frac{.008}{30.008} = 0.00027$$

- 8000 bit packet
- If RTT=30 msec, 1KB pkt every 30 msec 33kB/sec throughput over 1 Gbps link

Utilization of Go-Back N under No Loss



$$U_{sender} = \frac{3L/R}{RTT + L/R} = \frac{.0024}{30.008} = 0.00081$$

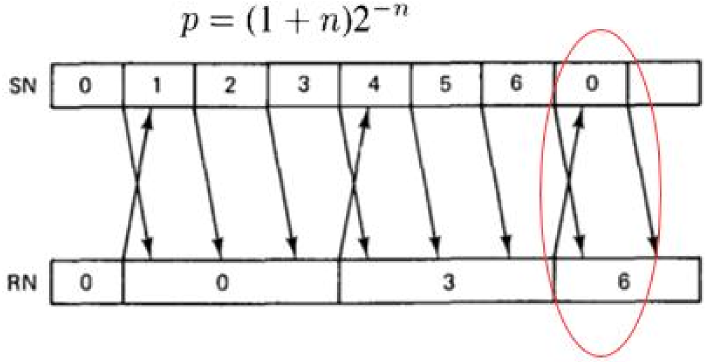
Limitations of Go Back N

- Retransmissions, or delays waiting for time outs, occur in go back N due to following
- Errors in the forward direction
- Errors in the feedback direction
- Longer frames in the feedback than in the forward directions

Effect of Long Frames in Reverse Direction

- Ack for packet 1 does not arrive at the sending side by the time packet 6 finishes transmission, thereby causing a retransmission of packet 0
- Probability that a frame is not acked by the time the window is exhausted is given by

$$p = (1 + n)2^{-n}$$



Character-based Framing

Character Codes

- Character codes such as ASCII provide binary representations
  - Keyboard characters and terminal control characters
  - Also for various communication control characters

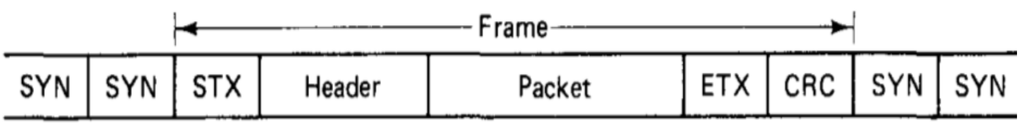
SYN Idle

- A string of SYN characters provides idle fill between frames when a sending DLC has no data to send
  - But a synchronous modem requires bits

STX and ETX

- STX (start of text) and ETX (end of text) are two other communication control characters
  - Used to indicate the beginning and end of a frame

Simplified Frame Structure



SYN = Synchronous idle

STX = Start of text

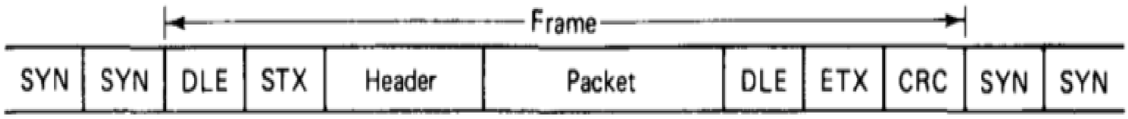
ETX = End of text

Problem

- The header or the CRC might (through chance) contain a communication control character
  - Since these always appear in known positions after STX or ETX, (no problem for the receiver)
- The payload might contain ETX character
  - Interpreted as ending the frame

Transparent Mode

- The transparent mode uses a special control character called DLE (data link escape)
  - Inserted before the STX character to indicate the start of a frame in transparent mode
  - Also inserted before intentional uses of communication control characters within such frame



DLE = Data link escape

Bit-oriented Framing
Bit-oriented Protocols

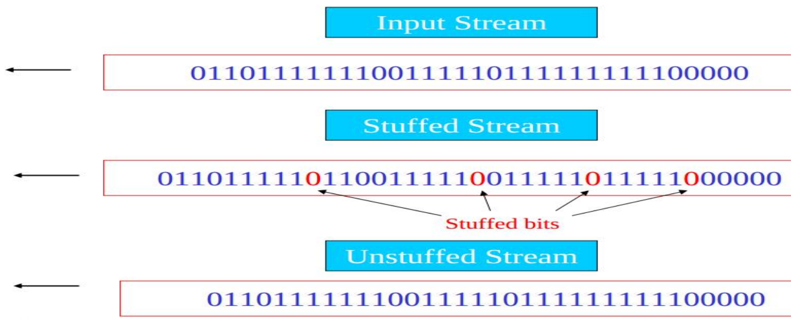
- Bit-oriented synchronous protocol pass variable-length frames
  - Image/voice data
  - Web data
- Dedicated or switched Simplex, half and full duplex

Flags

- 8-bit sequence (01111110) that delimits a frame's
  - Start and End
- **Procedure**
  - When DLL detects seq of 5 1s in a row in user data
  - Inserts a 0 immediately after the 5^{th} 1 in transmitted stream
- DLL at receiver removes inserted 0s by looking for seq of 5 1s followed by stuffed 0s
- **Problem**
- Confusion between possible appearances of the flag as a bit string within frame and actual flag indicating end of the frame

Bit-Stuffing Example

- The frame after stuffing never contains more than five consecutive 1's
  - Hence flag at the end of the frame is uniquely recognizable



Framing with Errors

Problems with framing

- Several peculiar problems arise
- When errors corrupt the framing information on the communication link
  - Flagging
  - CRC
  - Length field

Flags

- If an error occurs in flag at end of a frame
  - The receiver will not detect the end of frame
  - Does not check the cyclic redundancy check (CRC)
- When next flag detected, receiver assumes CRC to be in position preceding flag
- This perceived CRC might be the actual CRC for the following frame
- But the receiver interprets two frames as one
- Receiver fails to detect the errors with a probability 2^{-L}
- L is the length of the CRC

False Flag Example

Bits before the perceived flag are interpreted by the receiver as a CRC

- Accepting a false frame

0 1 0 0 1 1 0 1 1 1 0 0 1 ... (sent)

0 1 0 0 1 1 1 1 1 1 0 0 1 ... (received)

- Called the data sensitivity problem of DLC
  - Even though the CRC is capable of detecting any combination of three or fewer errors
  - A single error that creates or destroys a flag plus a special combination of data bits to satisfy the perceived preceding CRC, causes an undetectable error

Length Fields

Purpose of Length Field

- Basic problem in framing is to inform the receiving DLC where each idle fill string ends
  - Where each frame starts
  - Where each frame ends
- Include length field in the frame header

Overhead of Length Field

- If the length is represented by ordinary binary numbers
- No. of bits in the length field has to be at least
- $L = \log_2[K_{max}+1]$
  - K_{max} is the maximum frame size

Problems with Length Fields

- An error in this length field causes receiver to look for the CRC in the wrong place
  - An incorrect frame is accepted with probability 2^{-L}
  - L is the length of the Length field
- Receiver does not know where to look for subsequent frames

Partial Solution-1

- DECNET uses a fixed-length header for each frame
  - Places length of frame in header
  - Header has its own CRC
- Limitation: transmitter must still resync after such an error
- Receiver will not know when next frame starts

Partial Solution-2

- A similar approach is to put the length field of one frame into the trailer of preceding frame
  - Avoids inefficiency of the DECNET approach
  - Requires special synchronizing seq after each detected error

Topology and Connectivity
Topology

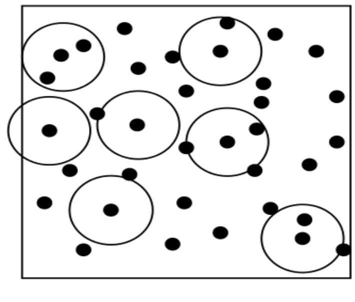
- Physical connectivity
  - Star
  - Hub
  - Mesh
  - Bus
  - Tree
- Connectivity is implied
- Wireless networks have constrained
  - Topology
  - Connectivity

Ad hoc Networks

- No infrastructure
- Nodes themselves
  - Transmit
  - Receive
  - Relay (forward)
- An operational area in which nodes randomly placed
- Locations follow a spatial distribution
- Must communicate with neighbors
  - Certain power

Spatial Reuse vs Connectivity

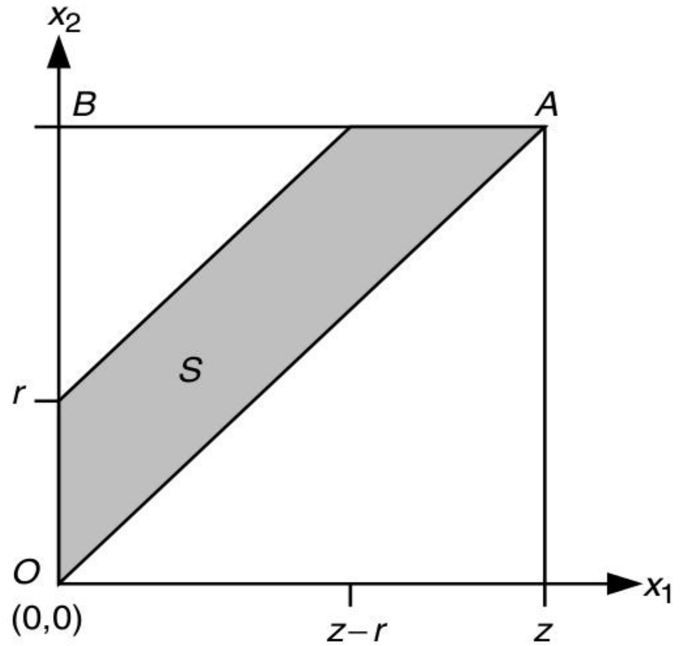
- The transmission range in the network is large
  - At a time at most one transmission occurs
- With smaller transmission ranges, many transmissions can occur simultaneously
  - Spatial reuse
  - Multihop



Feasibility Region

- x_1: location of the first node
- x_2: location of second node
- Nodes distributed uniformly in [0, z]

$x_1 \le x_2$



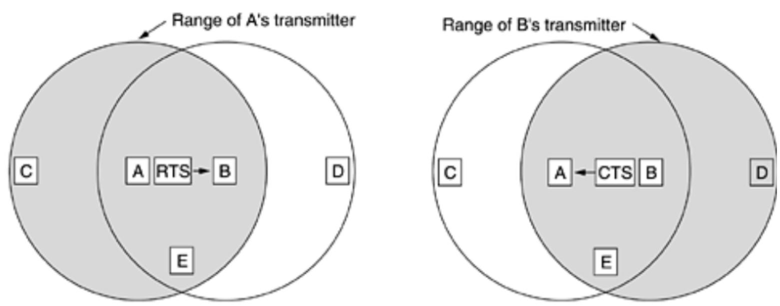
- Two-node network connected if $x_2 - x_1 \le r$
- Transmission range of every node: r(n) , where n is the number of nodes in network

Link Scheduling & Capacity
Hidden Terminal Problem

- Wireless nodes are blind
- Carrier sensing is hard
- Collision detection is harder

Link Scheduling

- MACA
- MACAW



Network Capacity

- Sum of all active connections
  - Simultaneous
  - Non interfering
- Varies with time
- Protocol design determines the effectiveness

Scheduling Constraints
Underlying Assumptions

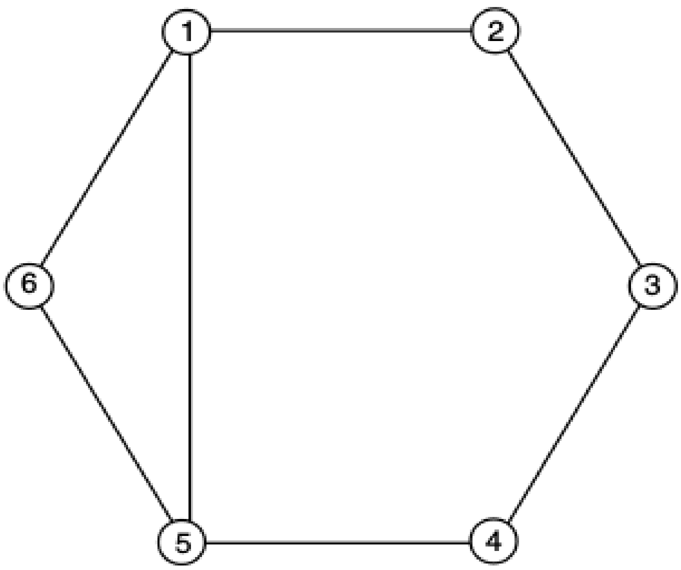
- Multihop wireless network
- Topology has already been discovered
- Directed graph G(N , E)
  - N is the set of nodes
  - E is the set of directed edges
- An edge $(i, j) \in E$
- Transmission from i , addressed to j
- Decoded by j, provided that the SIR at j is adequately high

Constraints

- The edges can be grouped into subsets
  - Edges in a subset can be activated in the same slot
  - Receiver in each edge can decode the transmission from the tail (TX) node of the edge
- Slotted time
- When such a set, S is activated one packet can be sent across each edge in S

Independent Sets

- S1 = {(1, 2), (5, 6), (3, 4)}
- S2 ={(2, 3), (1, 5)}
- S3 = {(2, 3), (4, 5), (1, 6)}



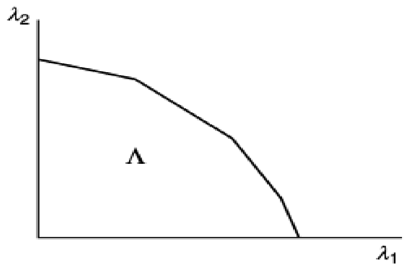
Centralized Scheduling
Scheduling Problem

- Schedule specifies a seq of independent sets to be activated
- Static link activation schedule
- Allocates M_S slots to independent set S
- BW allocation follows

$$b_e = \frac{\sum_{\{S \in \mathcal{S}\}} m_S I_{\{e \in S\}}}{M}$$

Maximum Schedulable Region

- Set of all such flow rates λ by L
  - Flow on each link be less than the average link capacity under the schedule



Bluetooth Example

- Piconet is a centralized TDM system
- Master controls the clock
- Determining which device gets to communicate in which time slot

Marginal Buffering at Every Hop
Definition

- Multiplexer has no buffer to store data arriving in a slot but cannot be served in that slot
- Performance depends only on marginal distribution of arrival process
- Doesn't depend on correlations b/w arrivals in slots

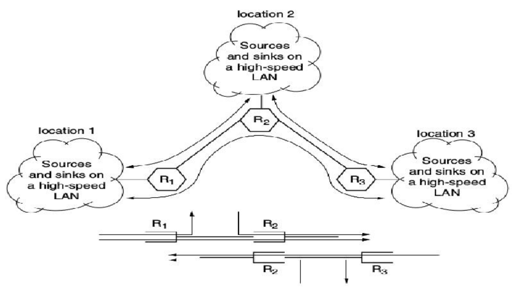
Simple Analogy

- The basic idea of "bufferless" multiplexing/routing is
  - Always forward a packet to an output port regardless of success

Multiplexer Network Scenario

- Traffic flow from location 1 to locations 2 and 3
- And from location 2 to location 3
  - Old and new traffic

causes superposition



Comments

- Packet switching is unachievable with zero buffering
- At least the header of a packet needs buffer
  - Cut through
- Mostly store-and-forward switching
  - An arriving packet entirely copied into switch from input to output links

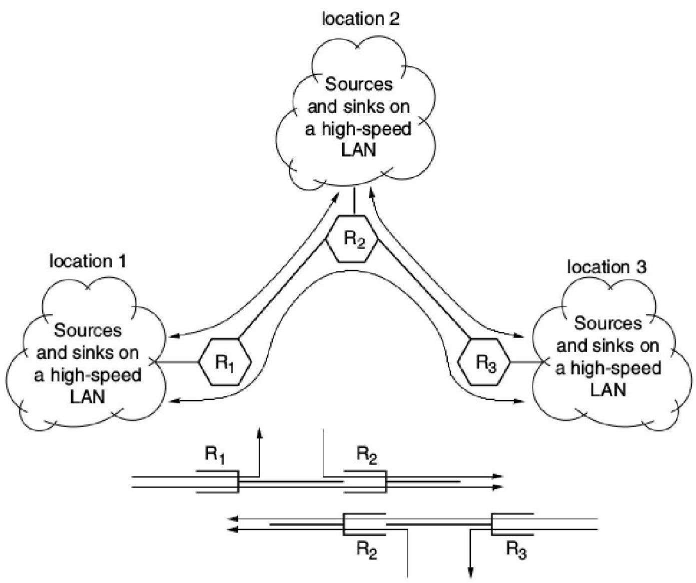
Arbitrary Buffering at Every Hop
Arbitrary Buffering

- Connection admission control with burst scale buffering
- Leaky bucket shaped sources and QoS requirements

Buffering constraints

- An arriving stream connection may or may not be admitted, if traffic is already being carried by the link
- Problem is exacerbated for multihop links

Scenario for Arbitrary Buffering



- Voice at loc 1 destined for loc 2 enters network at router 1 & leaves at router 2
- Voice at loc 1 destined for loc 3 leaves the link from router 1 to router 2 and enters the link from router 2 to router 3
- Here two-hop traffic multiplexed with data from loc 2 to loc 3

Comments

- Traffic from a source may be well characterized at the point where it enters the network
- After multiplexing at the first hop, the flows become dependent
  - This dependence is very difficult to characterize

Problem Set 1
Effect of BER on Channel Performance
Suppose that an 11-Mbps 802.11b LAN is transmitting 64-byte frames back-to-back over a radio channel with a bit error rate of 10-7 . How many frames per second will be damaged on average?

Ethernet Framing
A 1-km-long, 10-Mbps CSMA/CD LAN (not 802.3) has a propagation speed of 200 m/μsec. Repeaters are not allowed in this system. Data frames are 256 bits long, including 32 bits of header, checksum, and other overhead. The first bit slot after a successful transmission is reserved for the receiver to capture the channel in order to send a 32-bit acknowledgement frame. What is the effective data rate, excluding overhead, assuming that there are no collisions?

CSMA/CD Backoff Algo Performance
Two CSMA/CD stations are each trying to transmit long (multiframe) files. After each frame is sent, they contend for the channel, using the binary exponential backoff algorithm. What is the probability that the contention ends on round k, and what is the mean number of rounds per contention period?

Problem Set 1
Operation of MAC Addressing
Suppose nodes A, B, and C each attach to the same broadcast LAN (through their adapters). If A sends thousands of IP datagrams to B with each encapsulating frame addressed to the MAC address of B, will C's adapter process these frames? If so, will C's adapter pass the IP datagrams in these frames to the network layer C? How would your answers change if A sends frames with the MAC broadcast address?

Performance of ALOHA
Suppose four active nodes—nodes A, B, C and D—are competing for access to a channel using slotted ALOHA. Assume each node has an infinite number of packets to send. Each node attempts to transmit in each slot with probability p. The first slot is numbered slot 1, the second slot is numbered slot 2, and so on.
a. What is the probability that node A succeeds for the first time in slot 5?
b. What is the probability that some node (either A, B, C or D) succeeds in slot 4?
c. What is the probability that the first success occurs in slot 3?
d. What is the efficiency of this four-node system?

Switch Learn-ability
Consider a network in which 6 nodes labeled A through F are star connected into an Ethernet switch. Suppose that (i) B sends a frame to E, (ii) E replies with a frame to B, (iii) A sends a frame to B, (iv) B replies with a frame to A. The switch table is initially empty. Show the state of the switch table before and after each of these events. For each of these events, identify the link(s) on which the transmitted frame will be forwarded, and briefly justify your answers.

Simulate Parity Scheme Failure
Support in INET (Channel Behaviour)
- BER & PER allow basic error modelling
- When channel decides (based on RN) that an error occurred during transmission of packet
- Sets an error flag in the packet object

Support in INET (Rx Behaviour)
- The receiver module is expected to check the flag
- Discard the packet as corrupted if it is set
  - Default BER and PER are zero

Typical Example

- channel Ethernet100 extends ned.DatarateChannel

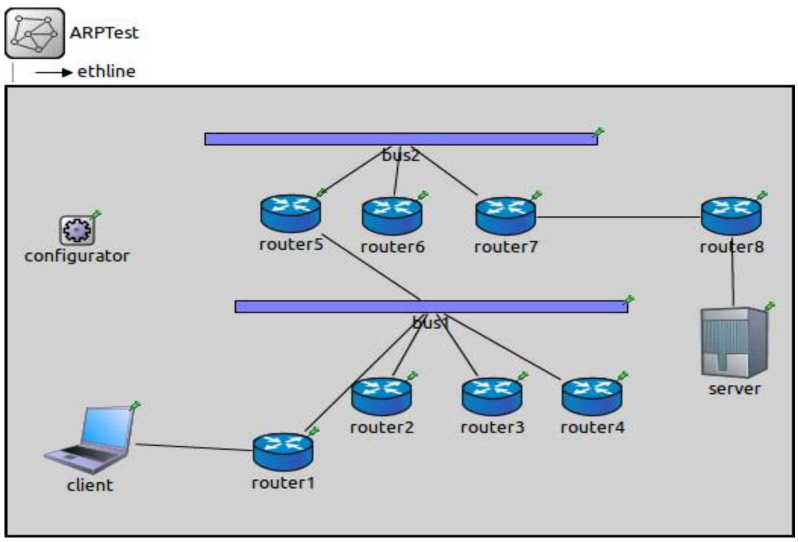
{
 datarate = 100Mbps;
 delay = 100us;
ber = 1e-10;
}

Failing Parity Scheme

- Need to hardcode the pattern that fails parity scheme
- The data pattern must be known
  - So that a corresponding error model can be designed

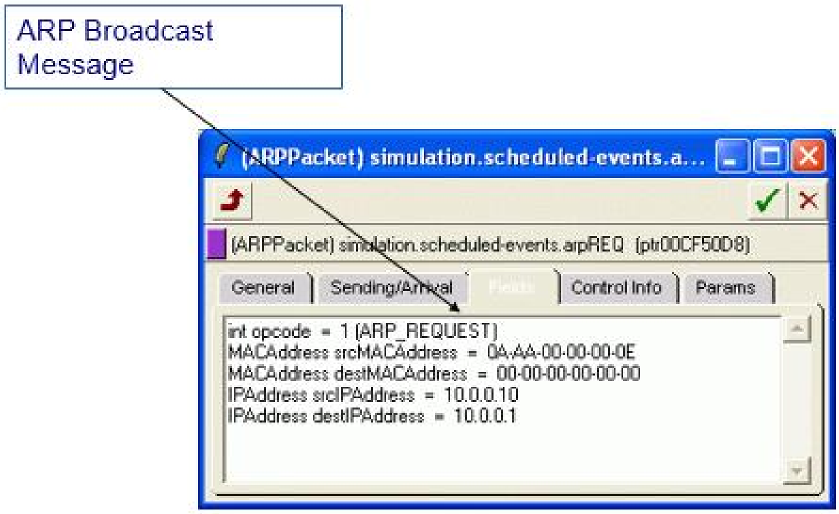
Simulate ARP Behaviour
Scenario

- Client computer opens TCP session with server
- Rest of operations (including ARP) follow
  - ARP has to learn the MAC address for the default router



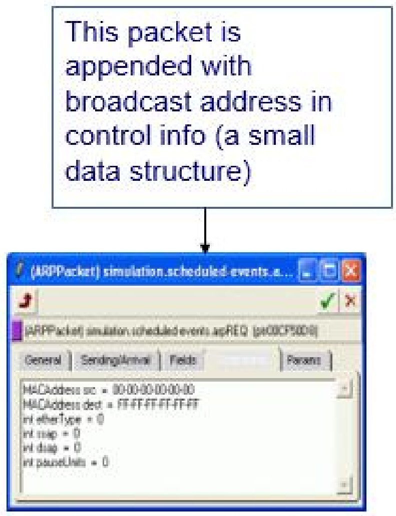
Design Tour of INET 3
arpTest.client.eth[0].arp

Inside ARP Packet

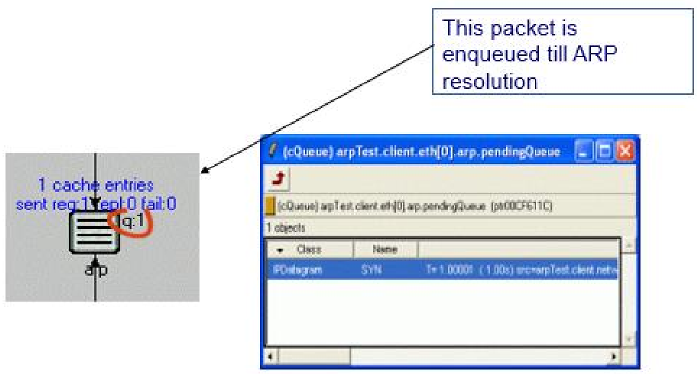


ARP Packet Class (Generated by .msg file)

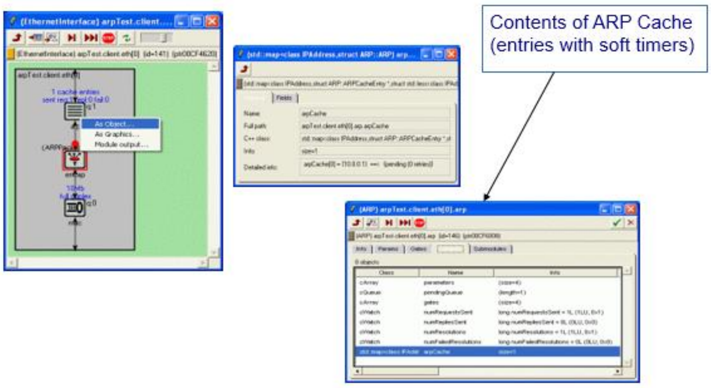
```
// file: ARPPacket.msg
message ARPPacket
{
fields:
 int opcode enum(ARPOpcode);
 MACAddress srcMACAddress;
 MACAddress destMACAddress;
 IPAddress srcIPAddress;
 IPAddress destIPAddress;
};
```



Packet Queue (Contains IP Packet)



ARP Cache Build-up



ARP Variants

- ARP Broadcast-unicast behaviour
- Proxy ARP
- Gratuitous ARP
- Reverse ARP

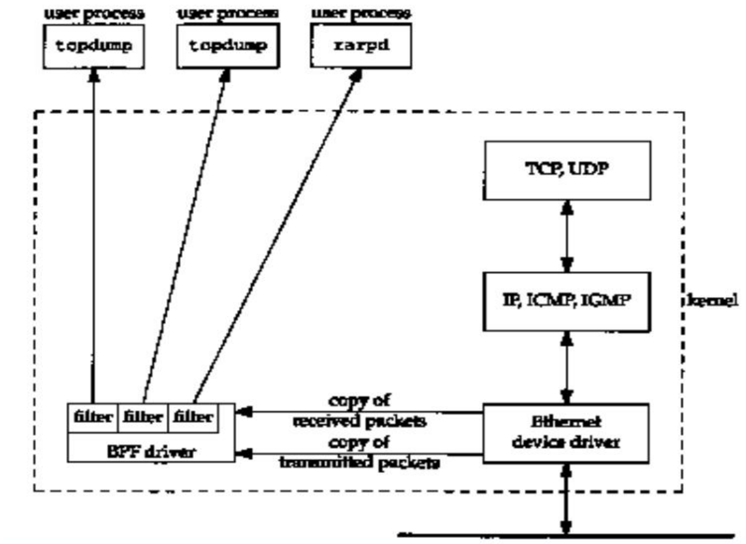
Performance

- No of broadcast attempts
- No of successes
- Effect of network size
- Multihop performance

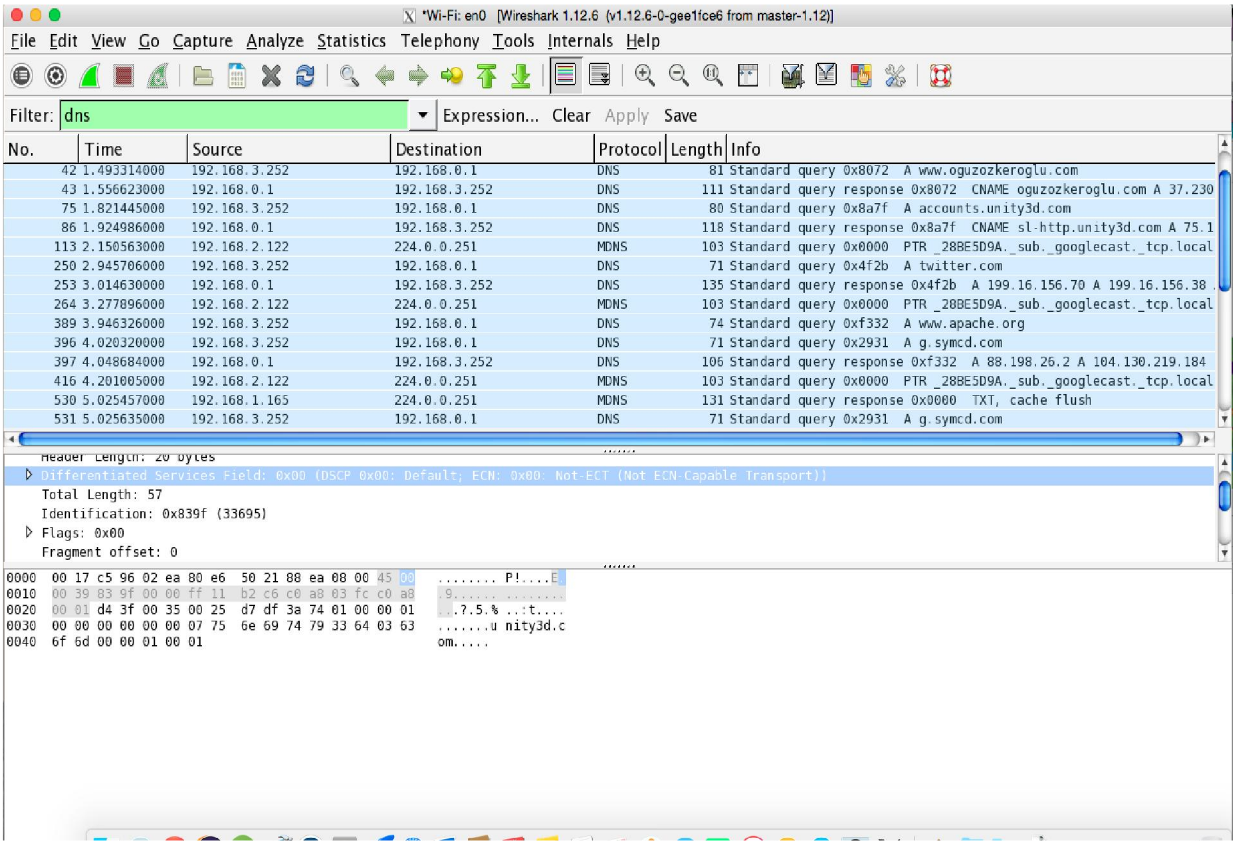
Output Analysis on WireShark
Wireshark

- A packet capturing & analysis tool
  - Work in promiscuous mode
- Presents output in Binary, Hex and ASCII
- Saves files as .pcap

Packet Capture Process



Wireshark Interface



Example

inet/examples/inet/tcpsack

- Sets up a flow between two hosts with TCP Sack
- Outputs files in multiple formats,
- Including the pcap format

Simulate Switching vs Routing

Why compare!

- Routing is inter-network phenomenon
  — It is pre-forwarding
- Switching is intra-network
  — It is forwarding
- Apparently no comparison
- Comparison at the device level
  — Router vs switch

Router vs Switch

- Routing process
  — Forwarding process
- Switching process
  — Port-based MAC learning
- ID-based behaviour
  — Unicast
  — Broadcast

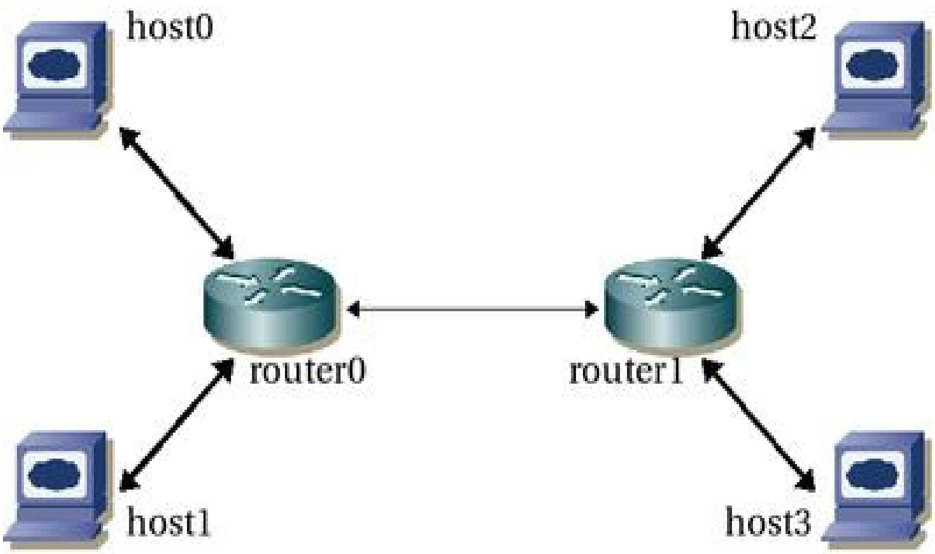
Basis of Comparison

- Cost
  - All router
  - All switch
  - Hybrid
- Isolation
  - Traffic
  - Domain
- Speed
- Complexity

Parameters

- Output queue lengths
- Output queue length distribution
- Output queue length Vs time plots
- Number of packets generated and received by hosts
- Packet size distribution
- Hop count distribution
- End to end delay

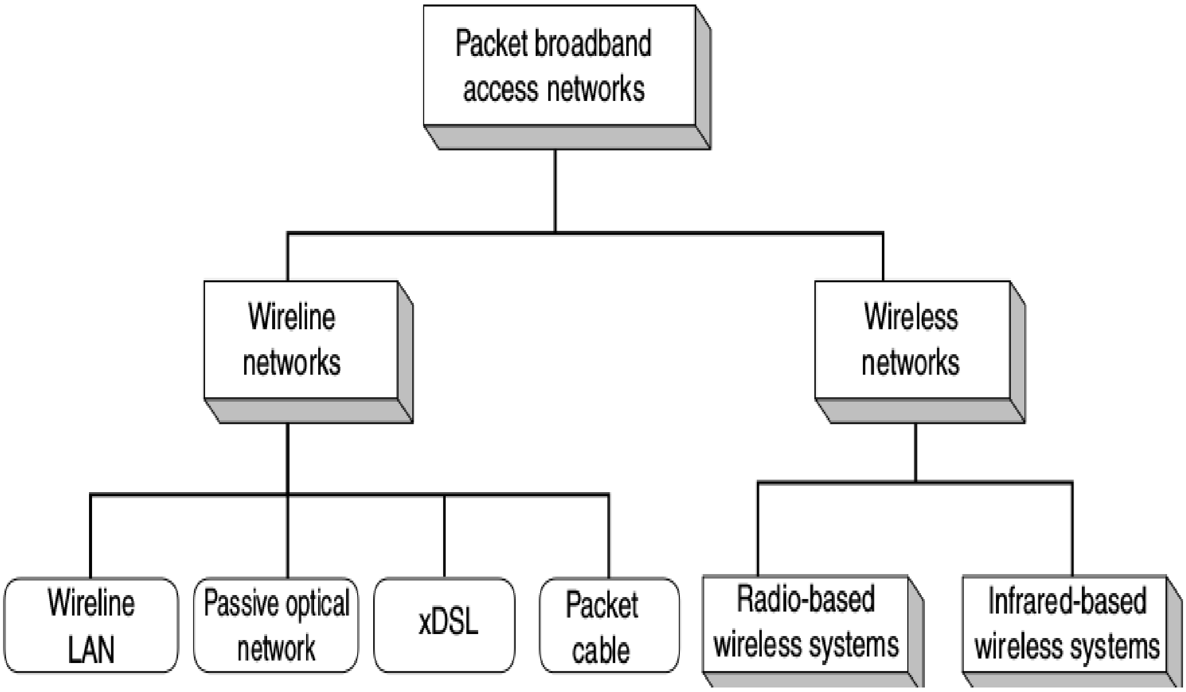
A Router (or Switch) Package



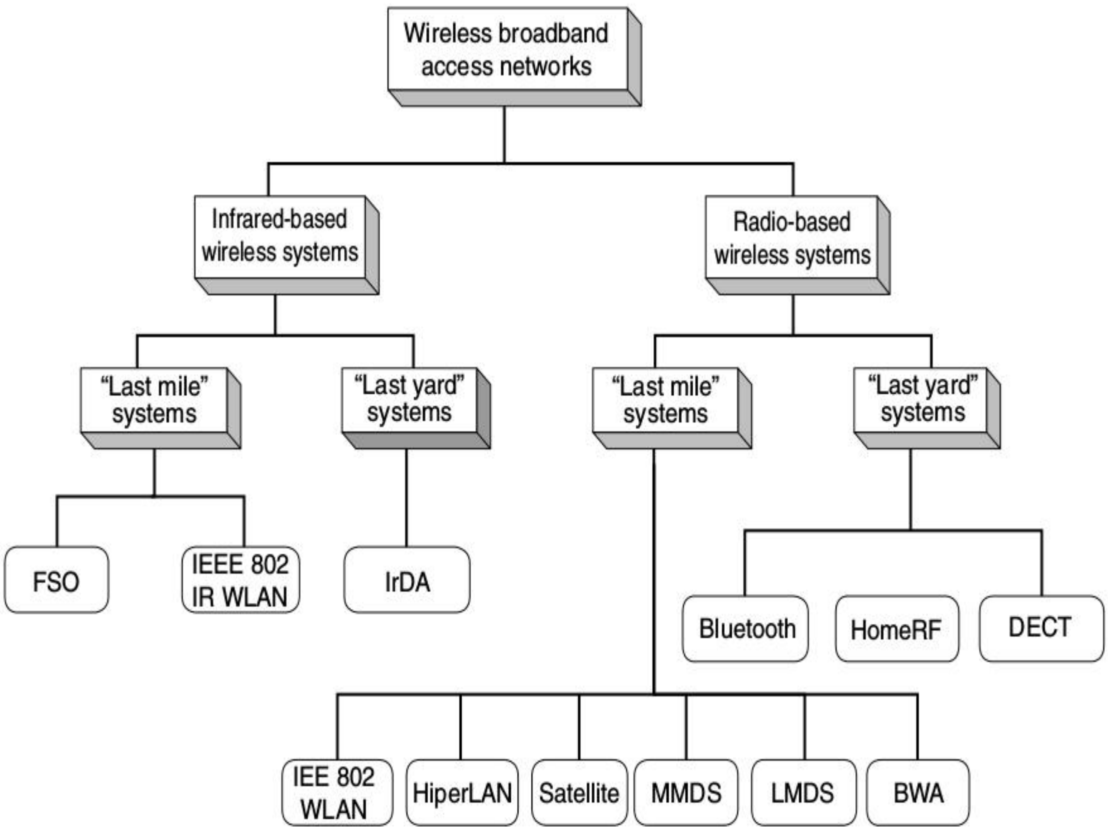
Overview of Access Technologies
Broadband Access

- Broadband is longhaul (backhaul)
  - Shared medium
  - Long distance
- Vs access side (baseband)
- Lastmile (first mile)
  - User-connecting technologies

Taxonomy of Packet Technologies

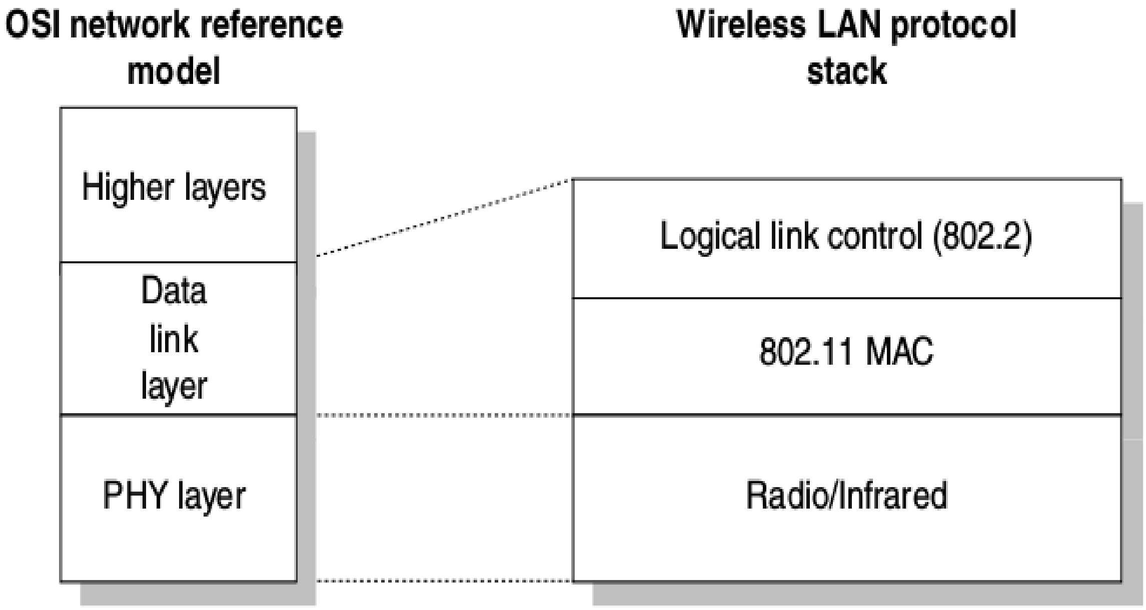


Taxonomy of Wireless Technologies

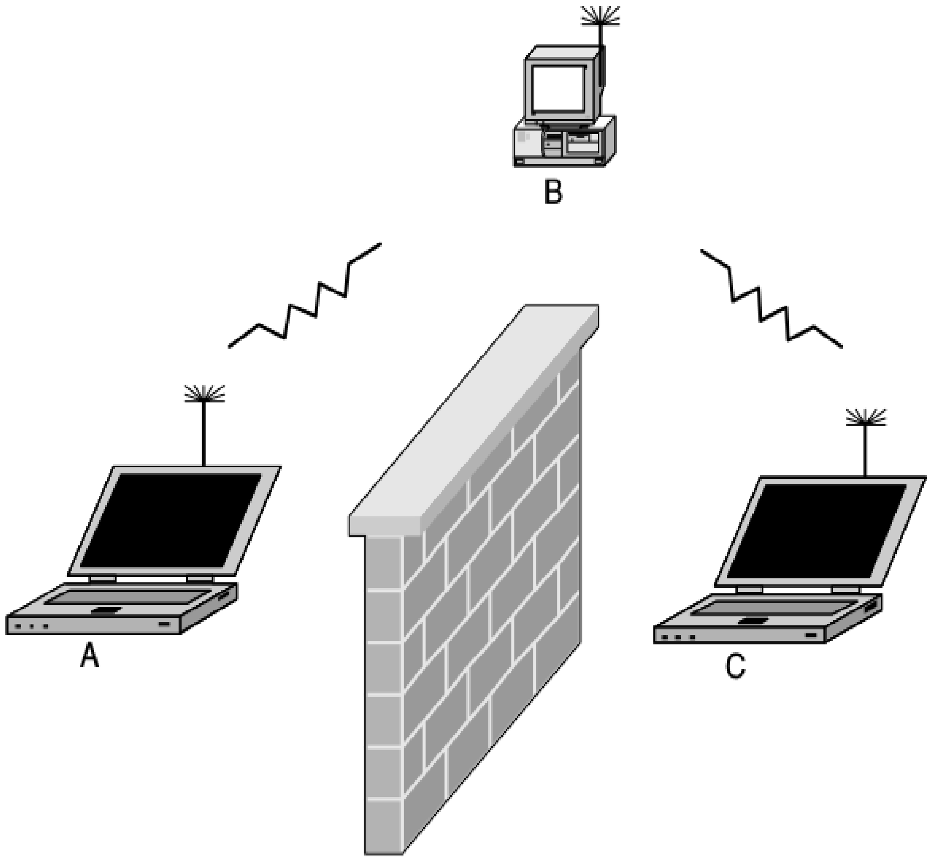


WiFi

WLAN Protocol Stack



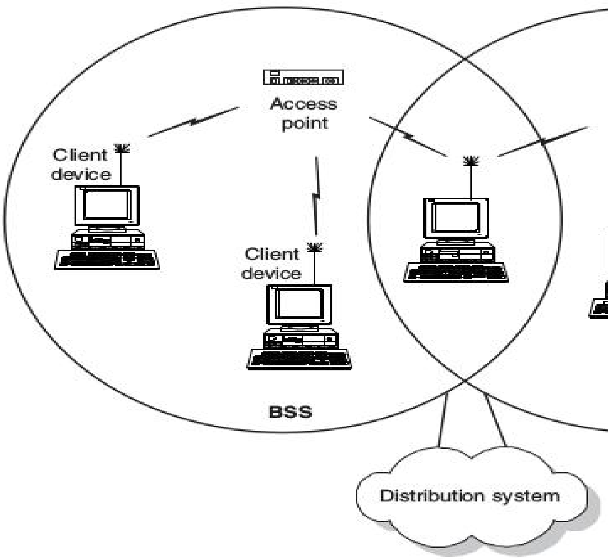
The Hidden (Exposed) Station Problem



RTS CTS Mechanism

- Sender sends request to send
- Receiver acknowledges as clear
  - Overhearing neighborhood cautioned

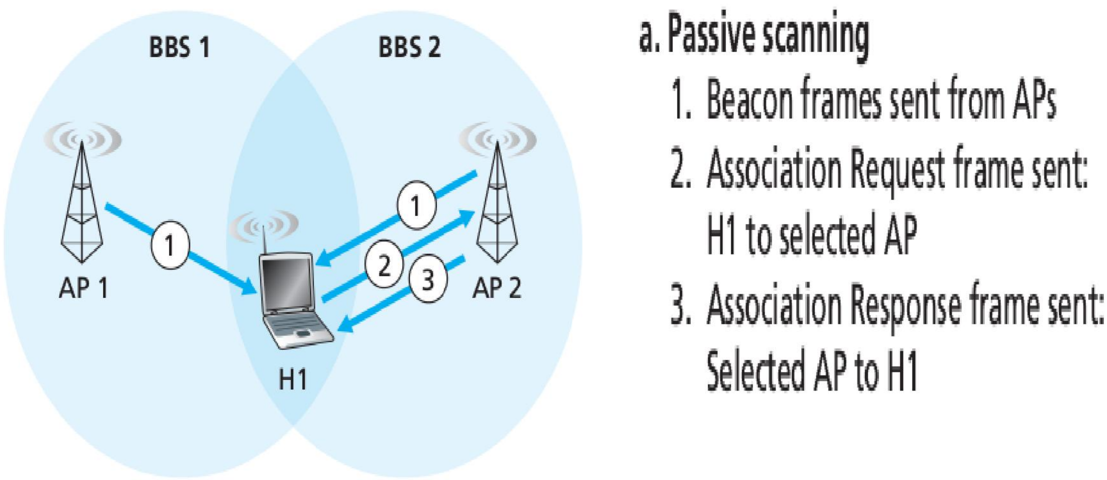
WLAN Configuration

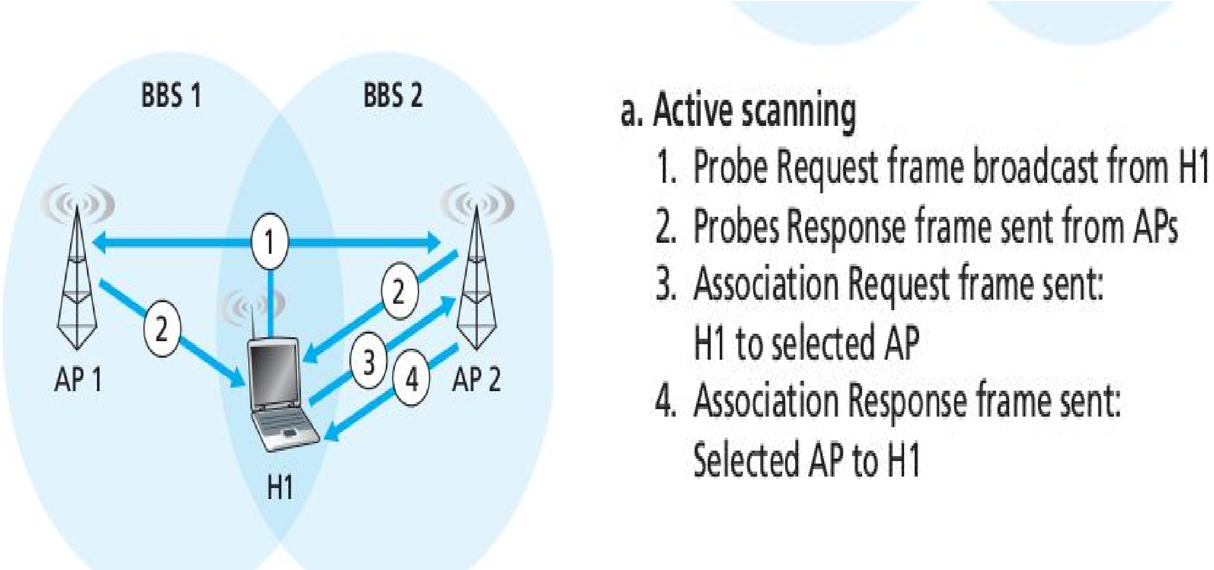


WiFi Operations
Operations

- Synchronization
- Authentication
- Association
- Data Transmission
- Handoff
- Power management

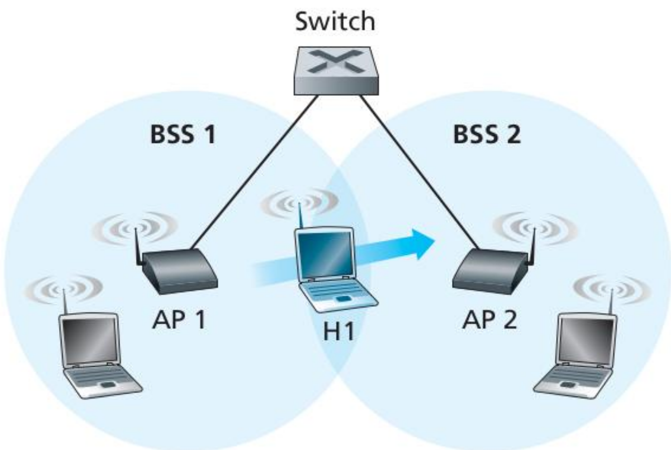
Scanning for APs





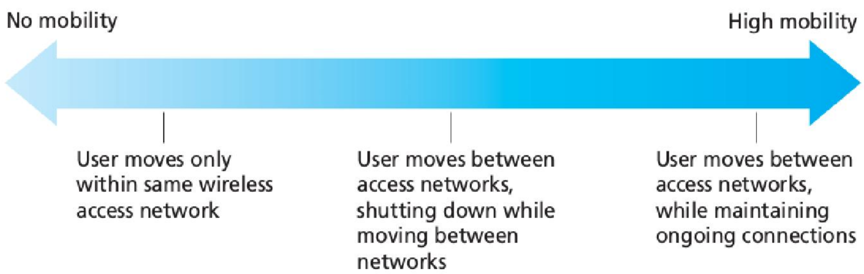
Mobility in the Same IP Subnet

- H1 moves from BSS1 to BSS2
- Keeps its IP address
    - — And all of its ongoing TCP connections



Mobile IP

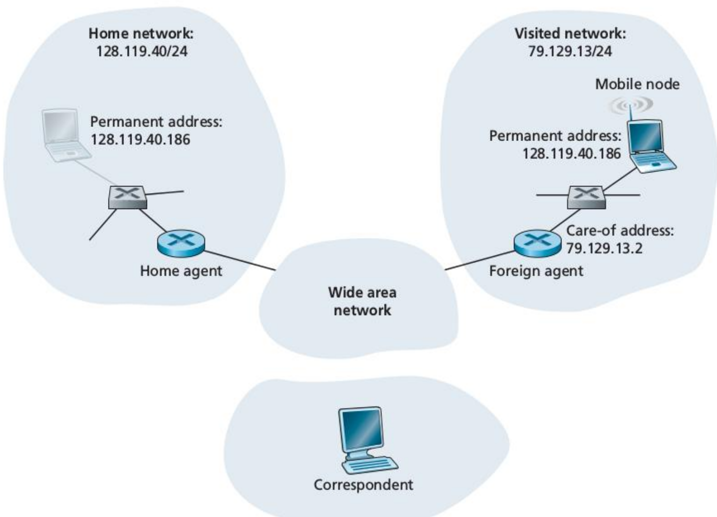
Degrees of Mobility



Mobile IP Standard

- RFC 3344
- Elements
  - — Home agents,
  - — Foreign agents,
- Foreign-agent registration
- Care-of-addresses
- Encapsulation (packet-within-a-packet)

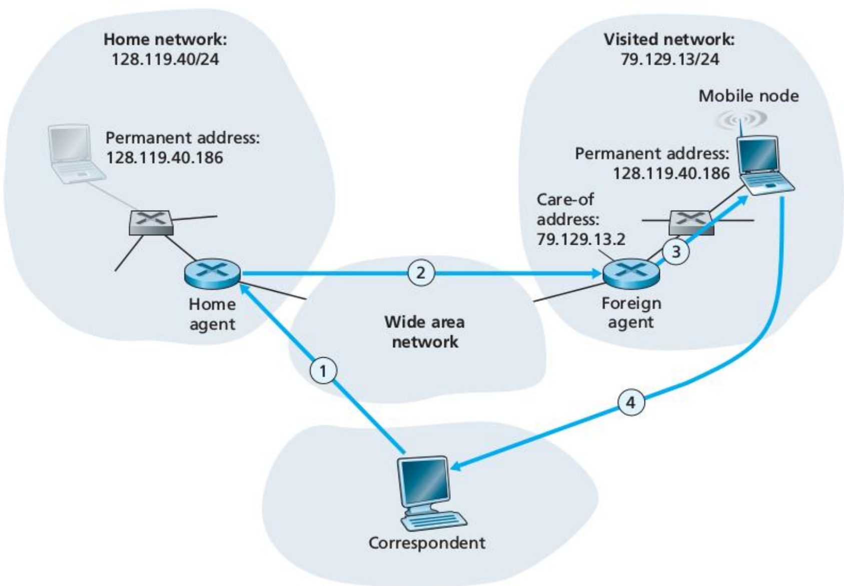
Elements of Mobile IP System



Procedures

- Agent discovery
- Registration with home agent
- Indirect routing of datagrams

Indirect Routing



Packet Cable Networks

Background

- Packet broadband cable network
  - — Built on existing broadcast cable TV (CATV) networks
- Hybrid fiber coax (HFC) cable networks
  - — Deployment of optical fiber
  - — New amplifier technology
- Alternative to DSL

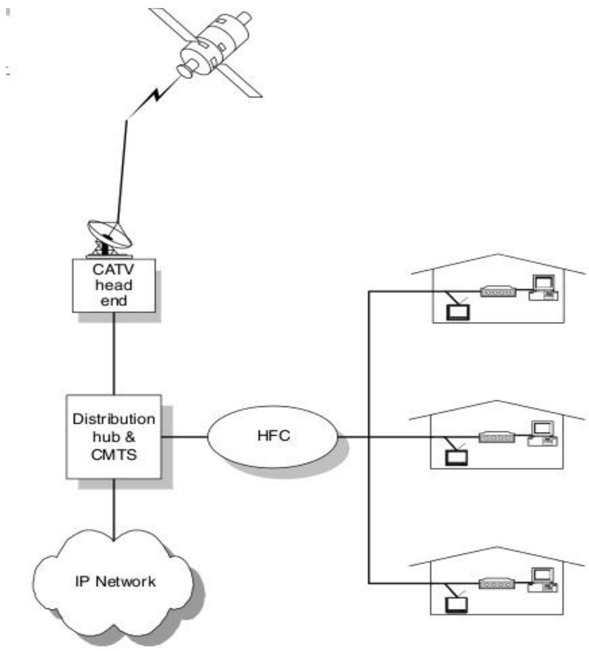
Architecture

- Tree topology
- One-way broadcast
- Headend and cable modems

Headend

- Operational center of a CATV cable access network
- Connected to many distribution nodes via trunk cables
  - — Coax cable or fiber

Components



Functions of Headend

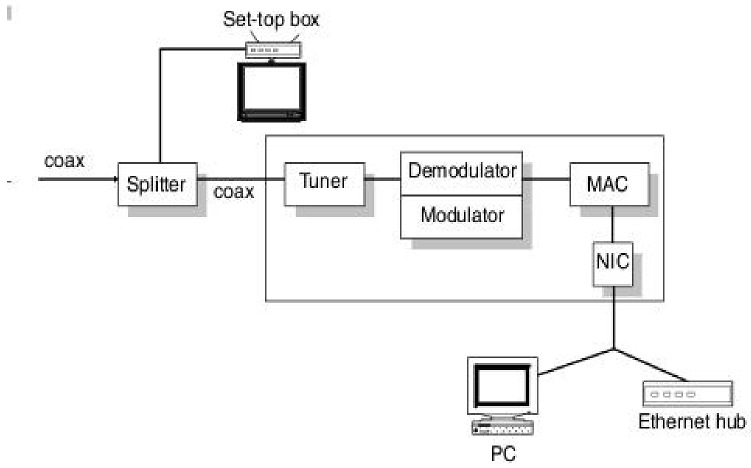
- Receiving broadcast signals from satellite or microwave dishes
- Mixing local or recorded TV programming
- Assigning channel frequencies to all signals destined for cable distribution

Functions of CMTS

- Controlling bandwidth allocation for data traffic to each modem
- Enforcing bandwidth allocation policy
- Assigning a time slot to each cable modem for transmitting upstream messages
- Enforcing QoS policies such as traffic shaping and policing (packet classification based on QoS classes)

Cable Modem Network Configuration

- Cable Model Systems accommodates two way communication
- DOCSIS (data over cable service interface specification)



WiMax

Background

- IEEE 802.16 is an emerging wireless MAN technology
- Originally designed to provide wireless last mile/first mile deployment in a MAN
- Also end-user access an alternative to 802.11 family
- Mobility support provided

Introduction

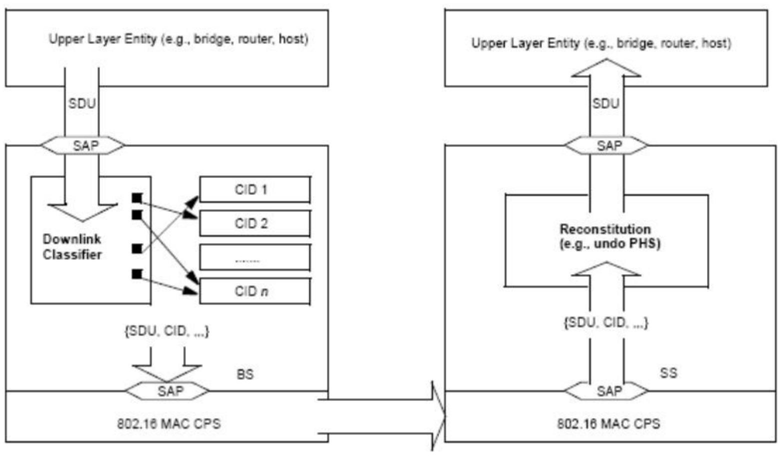
- Worldwide Interoperability for Microwave Access (WiMAX)
- Many basic ideas of 802.16 borrowed from DOCSIS/HFC applied to the wireless setting
- Good analogy : Wi-Fi : Ethernet :: WiMAX : DOCSIS/HFC

Architecture

- Line-of-Sight(LOS) and tens of Ghz spectrum
- Severe atmospheric attenuation
  - Suitable in operator network between two nodes with high bandwidth

Many base stations deployed at elevated positions

Components



Digital Subscriber Line

Background

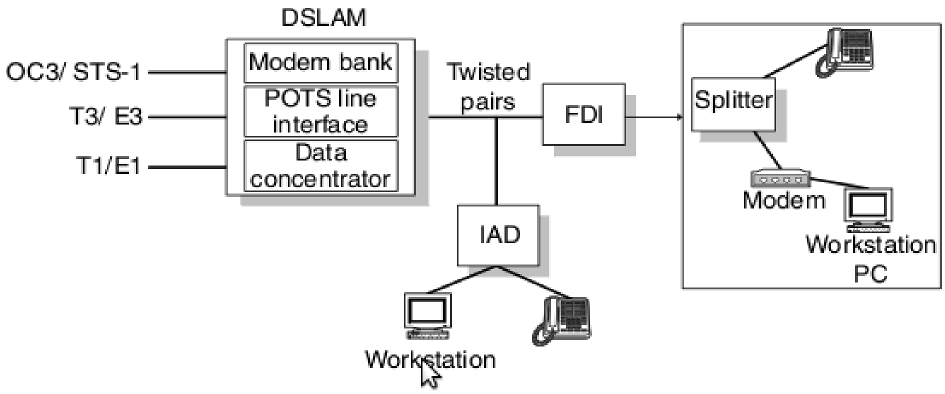
- A family of technologies for broadband last-mile solution using existing copper wires

Introduction

- Based on two premises
  - Discrete multitone (DMT) line code
  - Widely deployed twisted pair
- Provides upto 7 Mbps (suitable for Internet)
- Flexible bandwidth allocation per user demand
- Dedicated vs CATV

Architecture

- Enterprise CPE includes an integrated access device (IAD)
- Or connected through Feeder Distribution Interface



DSL Family

DSL type	Data transmission rate	Distance limit	Main applications
IDSL	128 Kbps in both directions	18,000 ft on 24-gauge wire	Similar to ISDN BRI but no voice service
CDSL	1 Mbps downstream; less upstream	18,000 ft on 24-gauge wire	Splitterless home and small office data service
G.Lite	1.544 to 6 Mbps downstream	18,000 ft on 24-gauge wire	Splitterless DSL; simplified ADSL
HDSL	1.544 Mbps duplex on 2 twisted pair lines; 2.04 Mbps duplex on 3 twisted pair lines	12,000 ft on 24-gauge wire	T1/E1 service replacement
SDSL	1.544 Mbps duplex (North America); 2.04 Mbps (Europe) on a single duplex line downstream	12,000 ft on 24-gauge wire	T1/E1 service replacement
ADSL	1.544 to 6.1 Mbps downstream; up to 640 kbps upstream	1.54 Mbps at 18,000 ft; 6.312 Mbps at 12,000 ft	Residential and small business Internet access and multimedia services
RADSL	640 Kbps to 2.2 Mbps downstream; 272 Kbps to 1.88 Mbps upstream	...	Internet access service for residential and small enterprise customers
VDSL	12.9 to 52.8 Mbps downstream; 1.6 Mbps to 2.3 Mbps upstream	4500 ft at 12.96 Mbps	Connections to fiber-based networks

Wireless Personal Area Networks

Introduction to LR-WPANs

- Low-rate low-power wireless personal area networks
  - Types of wireless sensor networks
- Applications
  - Industrial control & monitoring

- Environmental & health monitorinG

- Home automation, entertainment & toys
- Security, location and asset tracking
- Emergency and disaster response

Comparison

- IEEE 802.15.4
  - A new MAC for LR-WPAN
- IEEE 802.11: an “overkill technology”
- Bluetooth: High data rate for multimedia applications
- Small size network
- High power consumption

ZigBee vs Bluetooth

- Smaller packets over large network
- Mostly Static networks with many, infrequently used devices
- Larger packets over small network
- Ad-hoc networks

	Bluetooth	ZigBee
AIR INTERFACE	FHSS	DSSS
PROTOCOL STACK	250 kb	28 kb
BATTERY	rechargeable	nonrecharge
DEVICES/NETWORK	8	255
LINK RATE	1 Mbps	250 kbps
RANGE	~10 meters (w/o pa)	~30

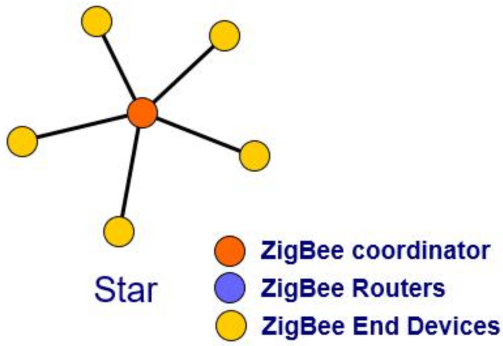
IEEE802.15.4
Features

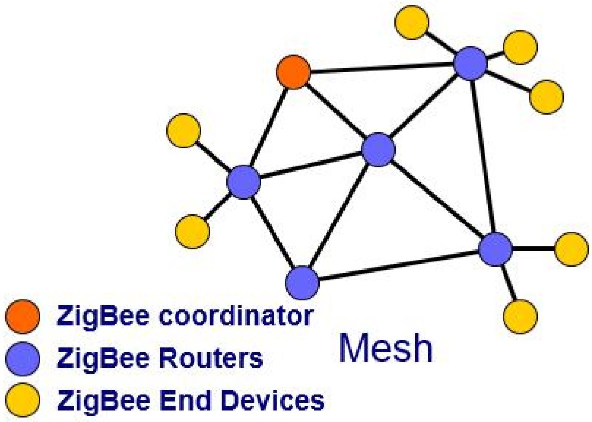
- Channels
  - 16 channels in 2450 MHz band
  - 10 channels in 915 MHz
  - 1 channel in 868 MHz
- Over-the-air rates of 250,40& 20 kb/s
- Addressing
- 16 bit short
- 64 bit extended

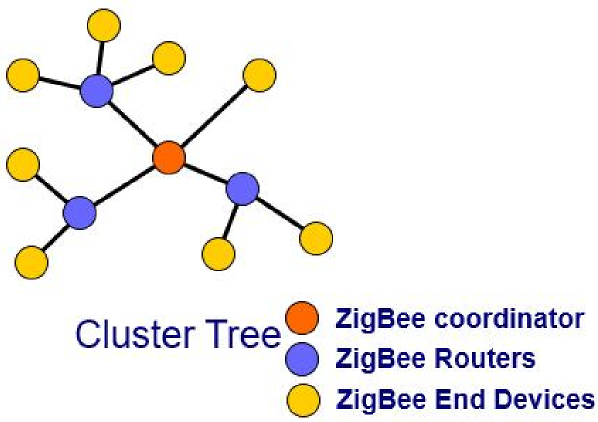
- Allocation of guaranteed time slots (GTSs)
- CSMA-CA channel access
- Fully acknowledged data transfer
- Low power consumption
- Energy detection (ED)

Link quality indication (LQI)

Topology Models





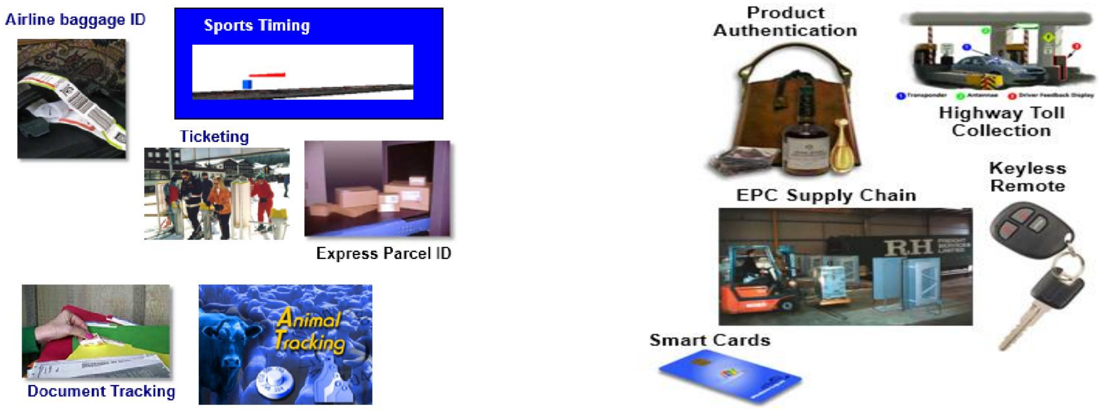


Radio Frequency Identification

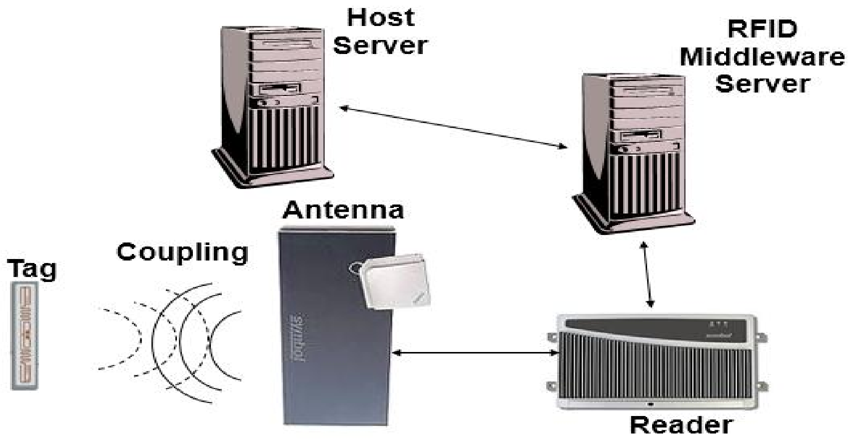
Introduction

- Presence known if within a certain radius
  - — Object identified
- Do not know exactly the position

Application Areas



Architecture



Traffic Flow