بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

==========

CS432 Handouts

Network Modeling and Simulation

Week [1 to 16]

Made by Mah jabeen
0321 2711298
[mahjabeen97869@gmail.com]

==========

Week 1

TOPIC 01

Network Modeling and Simulation

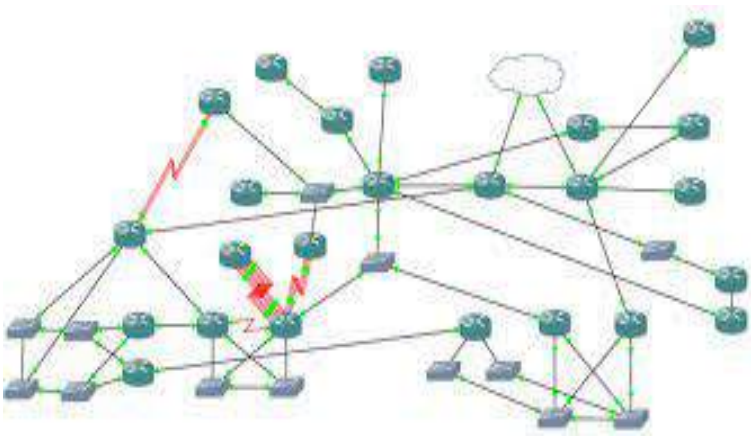
Course code CS432
Credits 3+0
Instructor
Dr. Ali Hammad Akbar
Lecturing style
Video lectures of short duration
(5-7 minutes)

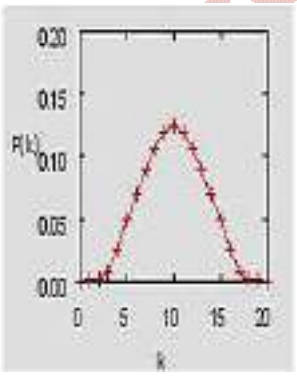
Evaluation

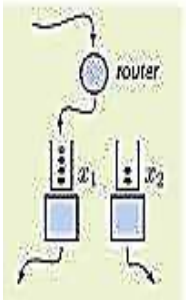
- Quizzes
- Assignment
- Simulation modules
- Mid-term and final

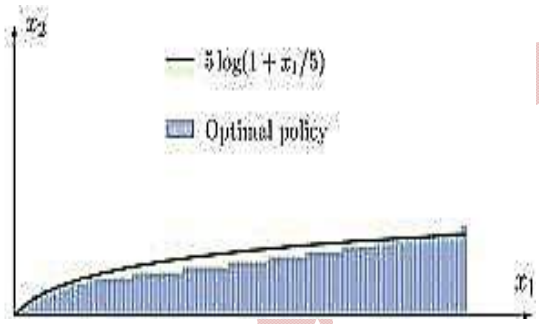
Enter the complex world of networks

- Can we simplify it?
- Or at least the discussion of it?



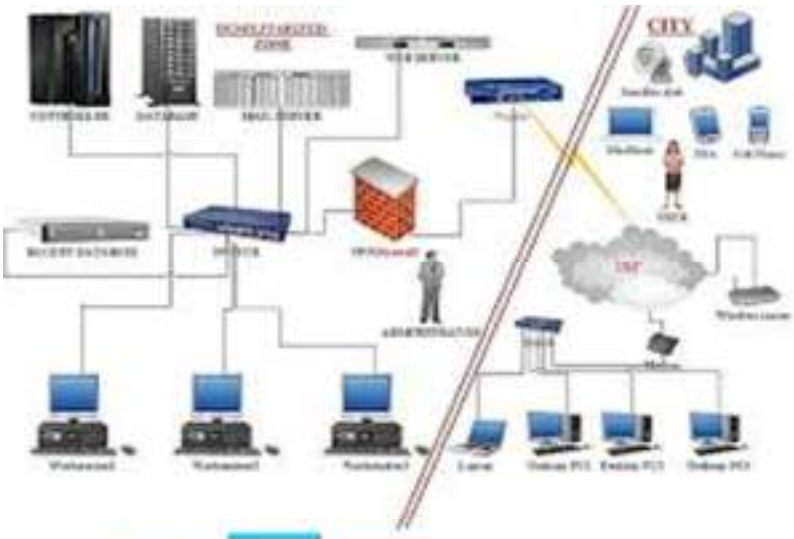




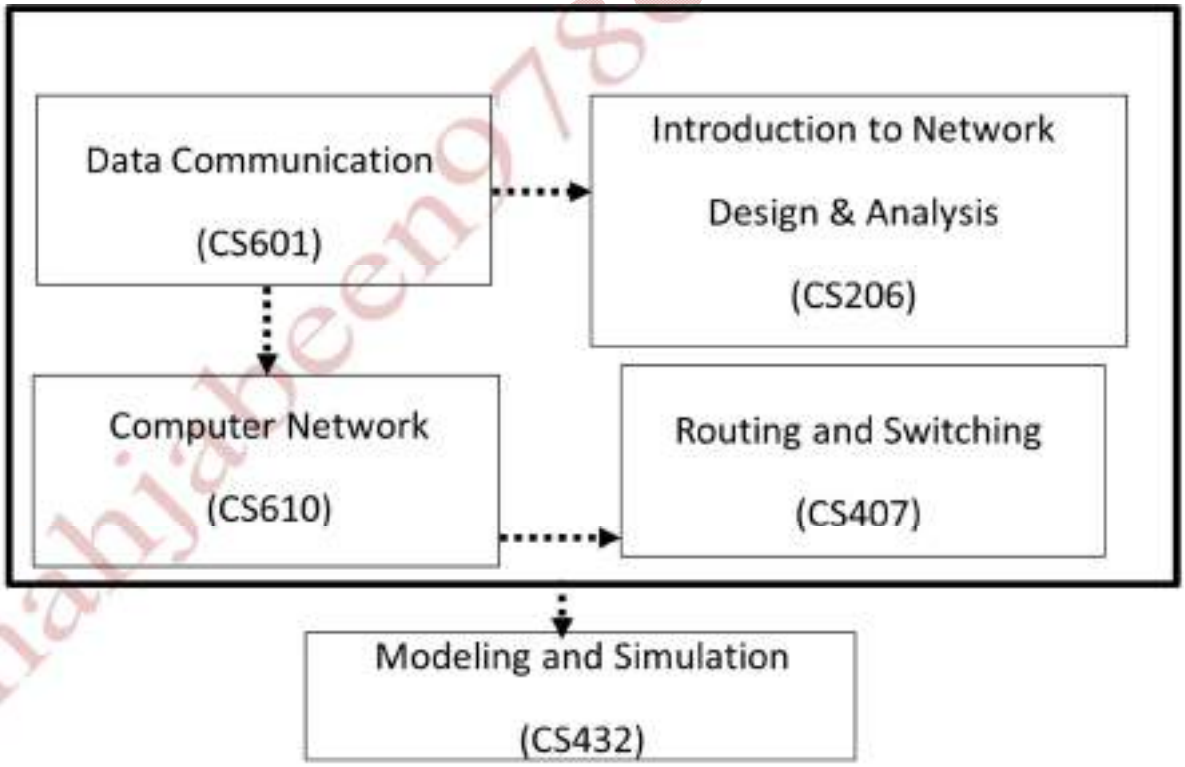


Knowing me

- Wireless Networks
- Distributed Systems
- Enterprise Network
- Broadband Access
- Machine-to-Machine
- Protocols and Models



Knowing you:



TOPIC 02

Need for NeMS

In this module

Let us explore the need and motivation to perform Network Modeling and Simulation (NeMS)

Technology Landscape (1 of 4)
Communications systems: Evolving rapidly
User demands: High performance networks
Service providers: Rapidly expanding their network infrastructure

Technology Landscape (2 of 4)
Network researchers face the protocol war
Developing new communications techniques, architectures and capabilities

Technology Landscape
(3 of 4)
Equipment vendors:
Release new devices with increasing capability and complexity

Technology Landscape
(4 of 4)
Technology developers and OEMs:
Developing NG equipment

Stakeholders challenges

- Network developer
- Network Designer
- Operational Engineer
- Architect

Network designer/developer

- How do I satisfy the QoS

demands of users amidst emerging technologies and techniques viz a viz legacy counterparts?

Network Engineer in operations

- What is the right approach to solving my problem?
- 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

- 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?

NO

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

COST **Abstraction**

END

TOPIC 03

What is NeMS?

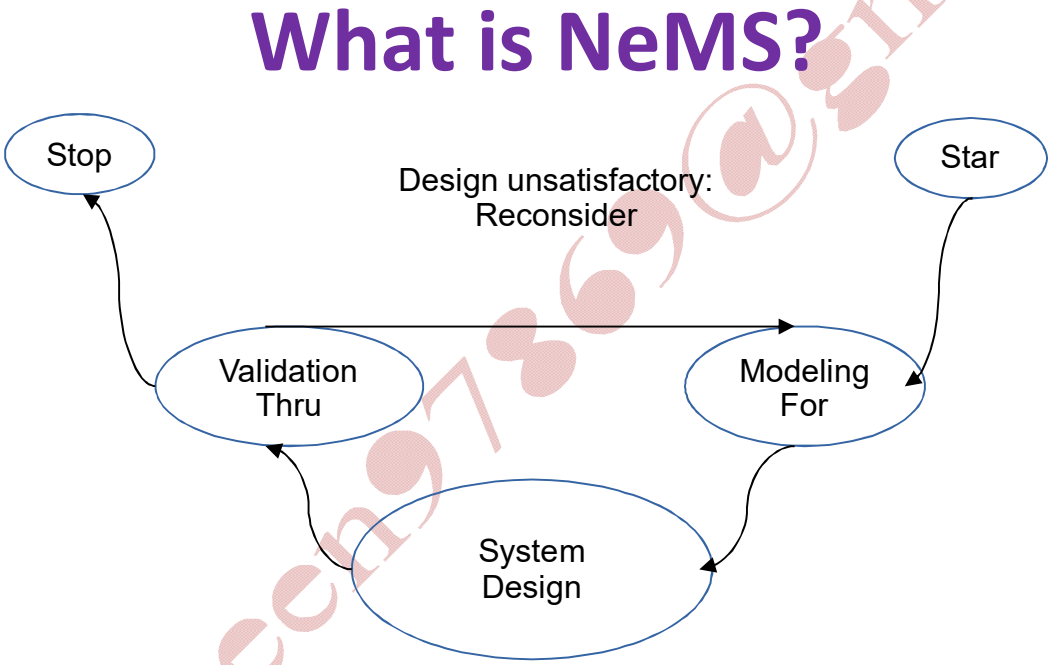
In this module

We shall understand NeMS as a single concept and break it further down

- What is modeling?
- What is simulation?
- Network Modeling and Simulation is often considered a single term.

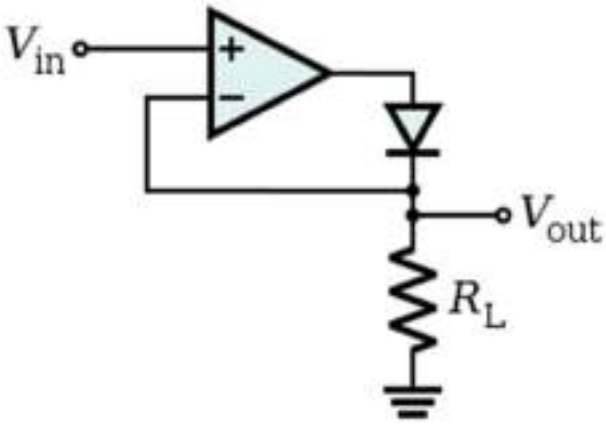
Simulation is

- Imitation of behaviour of real-world system
- Computational re-enactment
- According to rules described in **model**
- Modeling **precedes** simulations
- Together they form an **iterative** process
- **Approximate** the real world systems



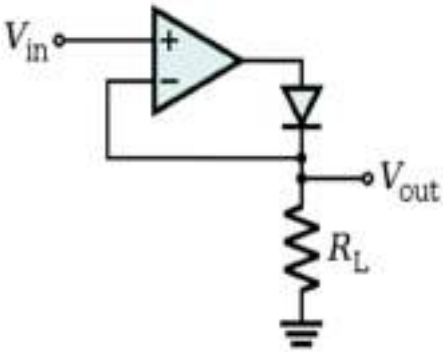
- Consider a simple signal detection circuit

of users with DBMS

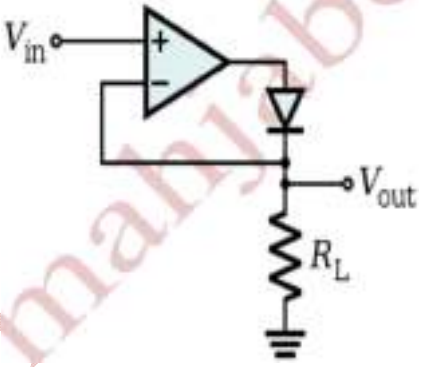


- Its **simulation** could give out
  - Correct result
  - Incorrect result
  - No result

Why!



- Because its behaviour has to be governed by its **model**
  - Transceiver design (Detection theory)
  - Digital circuit (Boolean algebra)



Model

- Logical representation
  - Complex entity
  - System

- Phenomena
- Process

Network Model

- In communications
  - Analytical representation
  - Mathematical form
  - State Machine
  - Closed or approximate form

Computer Simulation

- Computer software
  - Reproduces behavior
  - Certain degree of accuracy
  - Provides visual insight

END

TOPIC 04

What is Model?

In this module

We shall learn

- What is a computer model?
- What are types of models?

Computer Model

- A template on which a computer program runs
  - Inputs
  - Outputs
  - Behaviour
- Model definition
  - Descriptive
  - Analytical
  - Mathematical
  - Algorithmic

Types of computer models

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

- Open or closed

Stochastic vs Deterministic

- Deterministic models have no randomness
  - Given input always produces same output
  - Defined as a state machine Most common type of computer model

Stochastic vs Deterministic

- Stochastic models don't have unique input–output mapping
- Unpredictable execution
- Associated simulations are pseudo-random using random number generators
- Not widely employed

Continuous vs Discrete

- State variables assume

any value in continuous

- State variables assume only discrete values in discrete-state models
  - Continuous or discrete-time models
- Discrete computer models need clocks and timers

Steady state vs Dynamic

- Steady state models establish input-output relation in equilibrium
  - Simpler
- Dynamic models integrate internal changes to the system with changing inputs
  - Complex

Local or Distributed

- A distributed model would require multiple computing platforms
  - Distributed networking
  - Need synchronization
  - Better emulate real world
- Local models require single platform

Simpler to implement

Linear or Nonlinear

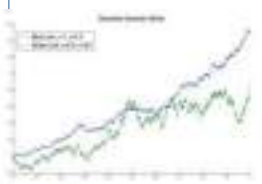
- A model that maps inputs to outputs linearly is simpler
- Nonlinear models involve complex functions
  - Difficult to implement

Open or Closed

- Open model define and require external inputs
- A model that does not take external inputs is closed
  - Automated systems

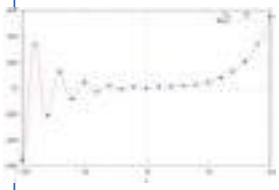
What is Model?

Stochastic
Deterministi



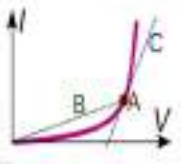
A

Continuous
Discrete



B

Linear
Nonlinear



C

Supra-modeling

A+B'+C
A'+C'

End

TOPIC 05

Modeling Perspective & Intra-Model Relation

In this module
We shall learn

- What needs to be modeled?
- Which is the most suitable model?
- How to do most appropriate modeling?

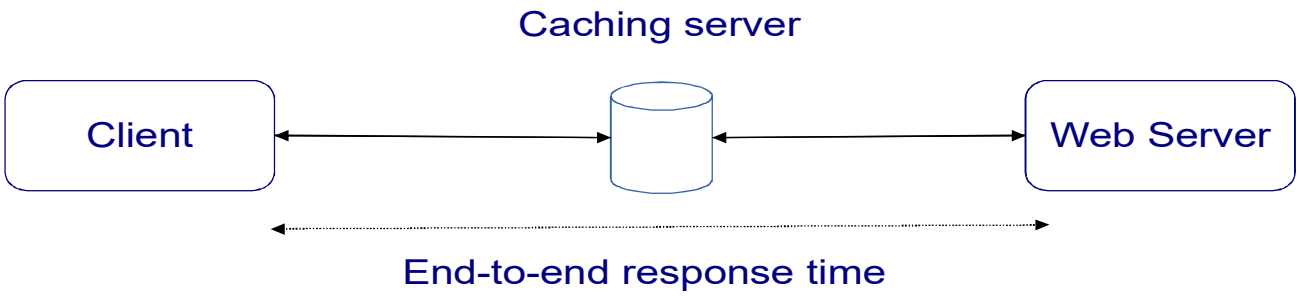
Model only what you understand

- Utility of model is as good as it mimicks real world system
- Knowledge of system is pre-requisite
- Wireless model requires

- Free space path loss
- Hidden terminal
- Absorption

Understand Your Model

- Your model has
  - Your perspective
  - Your assumptions
  - Your analytics/maths
- Caching server
  - Response time
  - Infinite buffer
  - Queuing theory

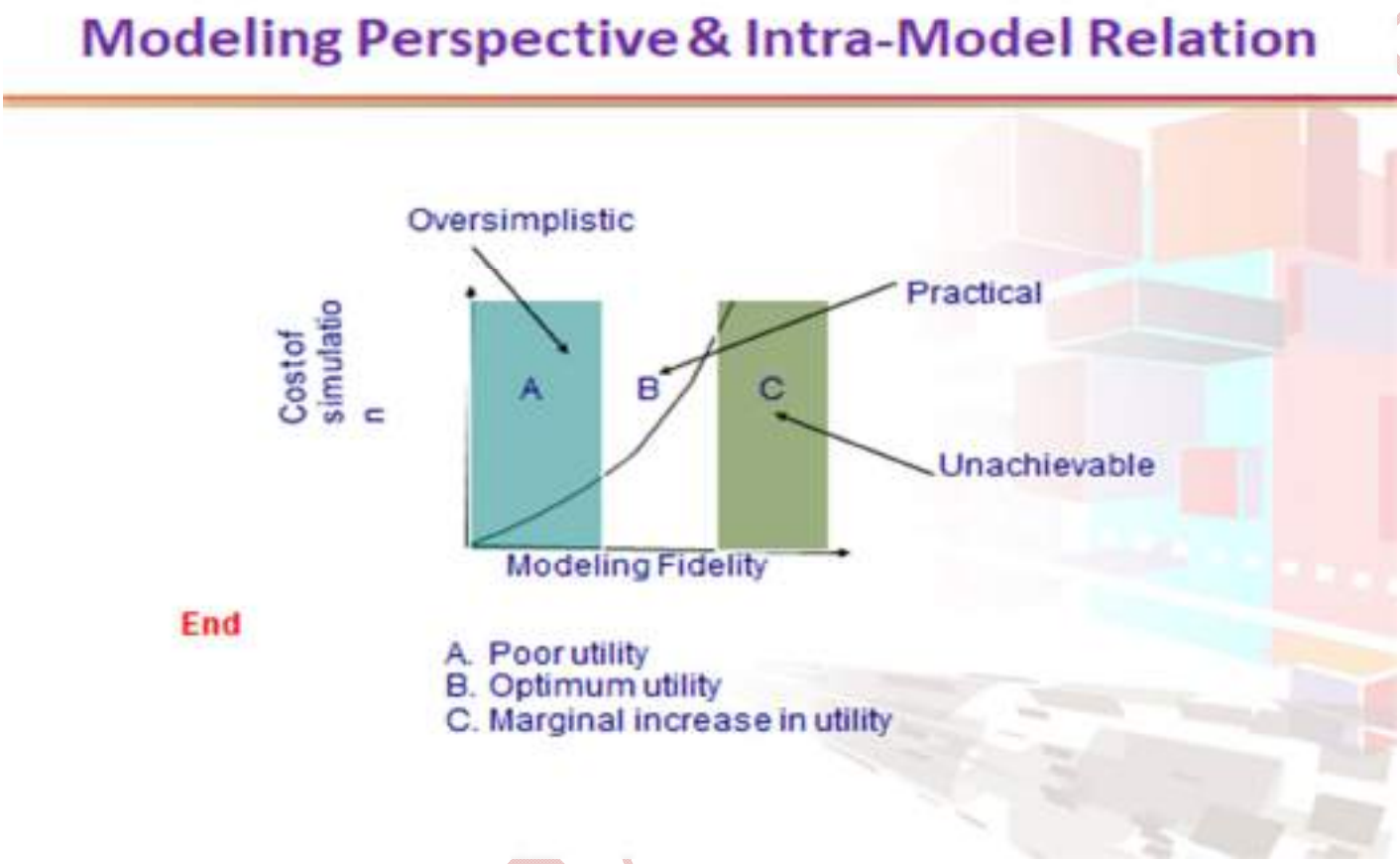


Average Response Time = Average Number in System / Average Throughput

Model what you need & no more

- Underdefined model
  - Simplified analysis
  - Easier Simulations
  - But, untrustworthy
- Overdefined model
  - Harder to realize
  - Complex analysis
  - Long run simulations
  - Very reliable
  - But, increased error

- A. Poor utility
- B. Optimum utility
- C. Marginal increase in utility



TOPIC 06

What is Simulation?

In this module

We shall understand

- Simulated outputs

What is Simulation?

Modeling and Simulation Terminologies

Parameter	Explanation
BER	PER, Throughput delay

Parameter	Explanation
Throughput	File transfer delay

Parameter	Explanation
Packet overhead	Network efficiency

Simulated outputs
END

TOPIC 07

What is Simulation?

In this module
We shall explore and learn

- A simple use-case
- What is its simulation?
- What are modeling and simulation terminologies

Please recall that

- Simulations are pieces of computer software
  - Implement algorithms
  - Take inputs
  - Give outputs

A simple use case: One-hop Communication Network

What is Simulation?

Modeling and Simulation Terminologies

Parameter	Explanation
Signal Power dBm	Received Power, BER, PER

Modeled inputs

Parameter	Explanation
Waveform Type Analog/Digital	BER, PER

Modeled inputs

Parameter	Explanation
FEC Hamming code	BER, PER, ReTX

Parameter	Explanation
Retransmission Protocol GoBack*N* Selective Repeat	Throughput, Delay

Parameter	Explanation
Channel Model AWGN, Rayleigh Rician	Received Power, BER, PER

Parameter	Explanation
Mobility Model Random waypoint Gauss Markov	MAC and IP routing

Parameter	Explanation
Channel Model AWGN, Rayleigh Rician	Received Power, BER, PER

Parameter	Explanation
Mobility Model Random waypoint Gauss Markov	MAC and IP routing

Parameter	Explanation
Traffic Model	Offered Load and queue behaviour

Parameter	Explanation
Network Topology	Connectivity, coverage

Parameter	Explanation
BER	PER, Throughput delay

Parameter	Explanation
Throughput	File transfer delay

Parameter	Explanation
Packet overhead	Network efficiency

Simulated outputs

END

TOPIC 08

Simulation Building Process

In this module

We shall explore and learn

- Simulation building process
- Inside computer

- Simulation run

Remember!
- One hop communication scenario

Simulation building process
- **Entities**
  - Wireless computers

and their packets (multiple instances)
  - WiFi AP (single instance)

Simulation building process
- **Entities (continued...)**
  - Traffic generator (single instance)
  - Creates wireless computers and their packets

Simulation building process
- **States**
  - WiFi AP (idle or busy)
  - Each computer generates a number of packets
  - Each packet successful/failed

Simulation building process
- **Events**
  - Wireless computer creation
  - Packet generation
  - Wireless AP activity
  - Each packet successful/failed

Simulation building process
- **Queues**
- Frames waiting in output queue of wireless computer
- Frames (Packet) at WiFi AP input queue

Simulation building process
- **Random realizations**
- Packet lengths
- No of frames per

wireless computer
- No. of wireless

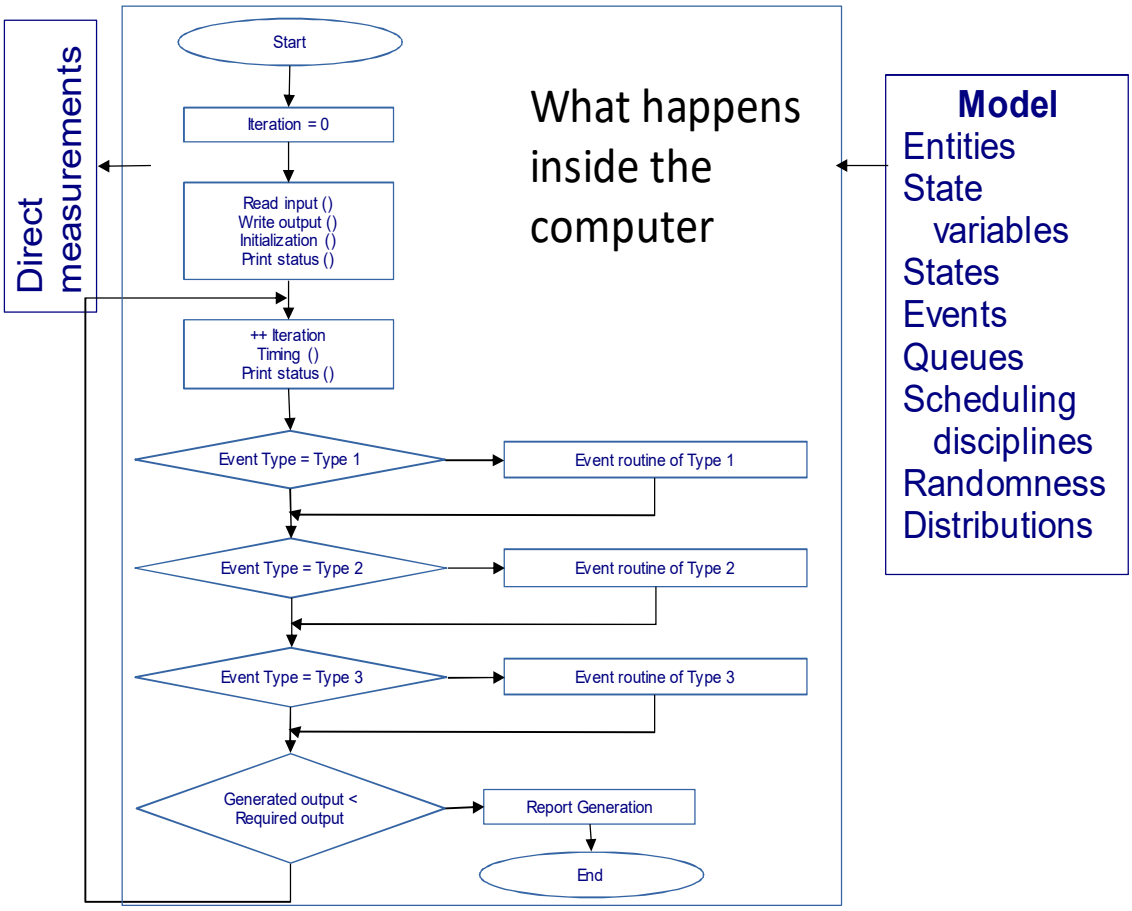
computers
- BER and PER
- Packet drop ratio

in WiFi AP input queue

Simulation building process

- **Distributions**
- Uniform/Gaussian
- Packet lengths
- No of frames per
- wireless computer

Simulation Building Process



Simulation run

- Inputs created/initialized
- Events of transmission, reception and noise occur

- Randomness causes queues to behave and err
- Packet successes/failures
- Simulation logs/compiles outputs

END

TOPIC 09

Components of Simulator

In this module
We shall determine
- Components of a typical simulator
- Their relationship and organization

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

Event queue
- Number of events in an ordered list
- Determines complexity
- Total simulation time

Simulation clock
- Global variable
- Clock ticks (constant and small increments)
- Time driven
    - Events occurring

within incrementing time are of interest
- Event driven
    - Time is fast forwarded to the occurrence of event

State variables

- Together define the state of the system

Event routines
- Handle the occurrence of event
- Event creation
- Event action
- Updates state variables
- Updates event queue

Input routines
- Provide user interface

(UI)
- Takes parameters
- Passes to main program

Report generation routine
- Calculates results
- Provides to user interface

(UI)

Initialization routine
- Initializes
  - State variables
  - Global variables
  - Statistical variables

Main program
- Calls other routines
- Initializes and terminates simulation

END

TOPIC 10

Types of Simulations

In this module

We shall summarize
- Performance evaluation techniques
- Types of simulations

Performance Evaluation Techniques (PETs)
- **Simulation**: behavior of interconnected subsystems & subprocesses

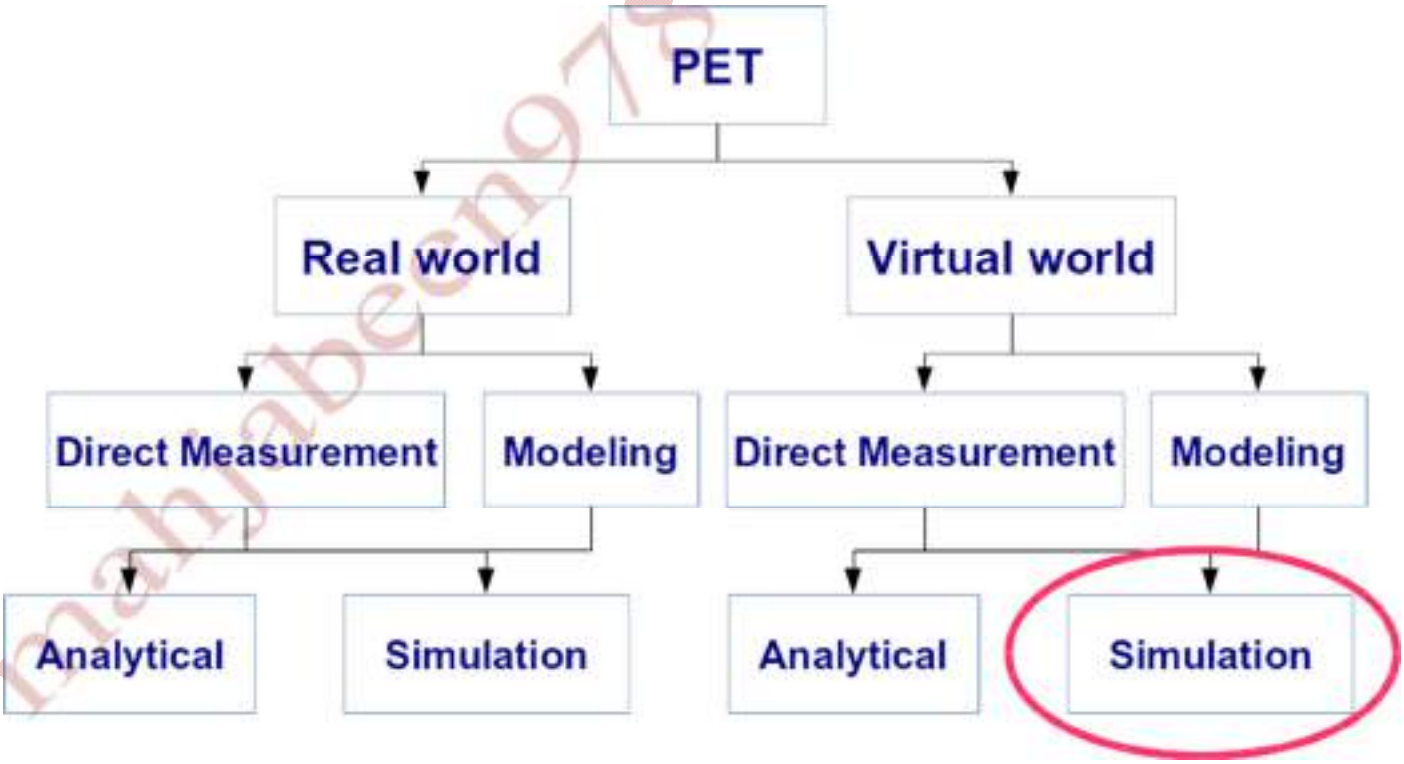
- **Evaluation**: measure of an aggregate systemic property
    - Efficient and confident

PETs

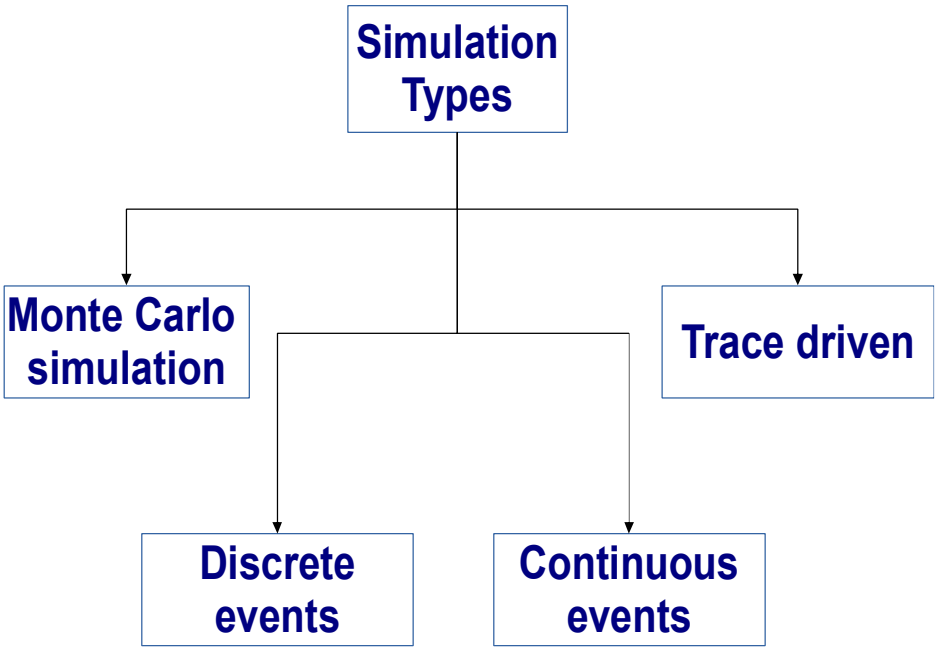
Direct measurement

- Not abstracted
- Incorporate real workload
- Maps to real world

- Only available for operational system
- Behavioral sensitivity
- End-to-end not possible



Types of Simulations



Monte Carlo

- No time axis
- Model probabilistic phenomena that do not change with time

Trace driven

- Ordered list of events

as input

- Expected outputs!

Continuous event

- States change to inputs

at non discrete times

- Consequently has a very large number of outputs and state changes
- Generally transformed into discrete event simulations

Discrete event

- No events between intervals

Finite events and output

END

TOPIC 11

Common Simulation Pitfalls

In this module
We shall understand
- General and programming mistakes
- Simulation inaccuracies
- Misleading results

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

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

END

TOPIC 12

Common Simulation Pitfalls

In this module
We shall understand
- General and programming mistakes
- Simulation inaccuracies
- Misleading results

General mistakes

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

END

TOPIC 13

Common Simulation Pitfalls

In this module
We shall understand
- General and programming mistakes
- Simulation inaccuracies
- Misleading results

General mistakes
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

END

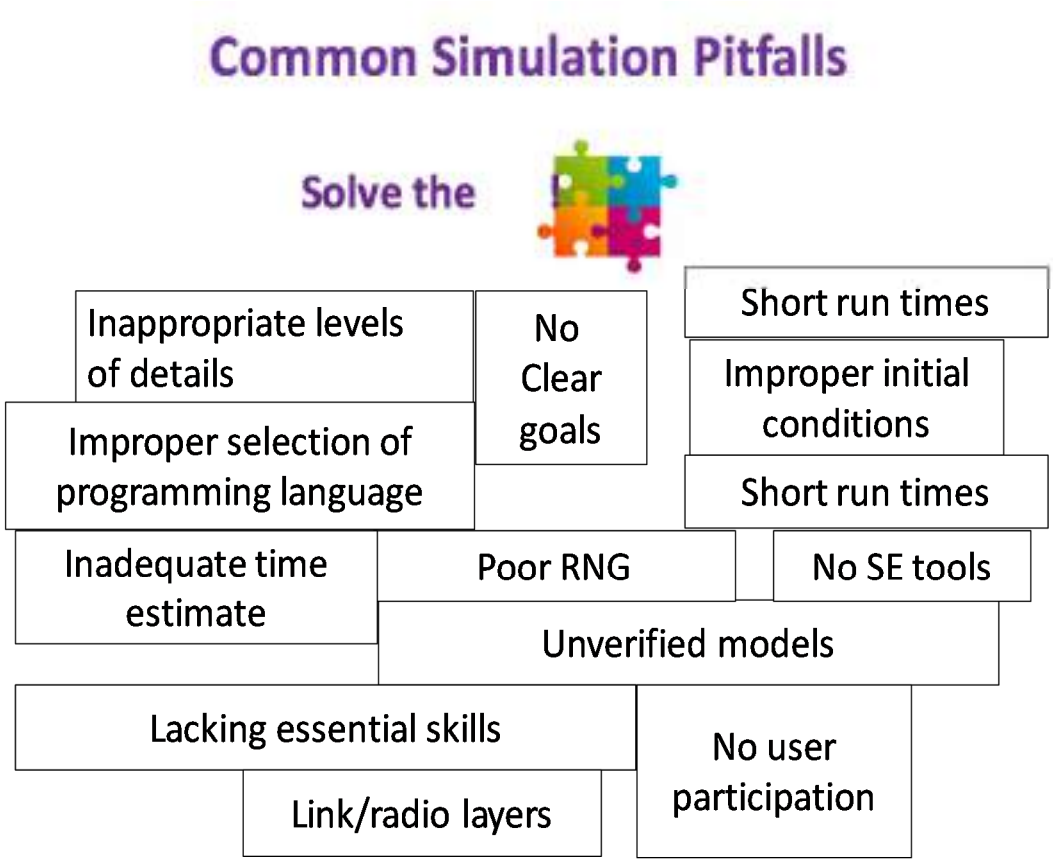
TOPIC 14

Common Simulation Pitfalls

In this module
We shall understand
- General and programming mistakes
- Simulation inaccuracies
- Misleading results

Simulation inaccuracies
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

Simulation inaccuracies
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



END

Week 2

TOPIC 15

Development of Systems Simulation

In this module

We shall understand

- The process of development of systems simulations
- Development life cycle

A “Still I am not dead yet!” scenario

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

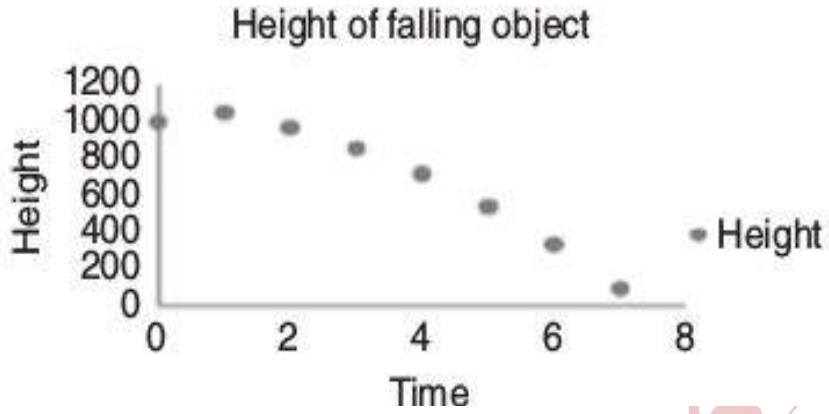
Available
h = height (feet)
t = time in motion (seconds)
v = initial velocity (feet per second, + is up)
s = initial height (feet)
a = acceleration (feet per second per second

Not available
Mass of object
Air resistance
Location of object

A “Still I am not dead yet!” scenario

```
/* 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;
        for (t = 0, h = s; h >= 0.0; t++)
        {
            h = ( -16.0 * t * t) + (v * t) + s;
            printf(“Height at time %d = %f\n ”, t, h);
        }
}
```

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

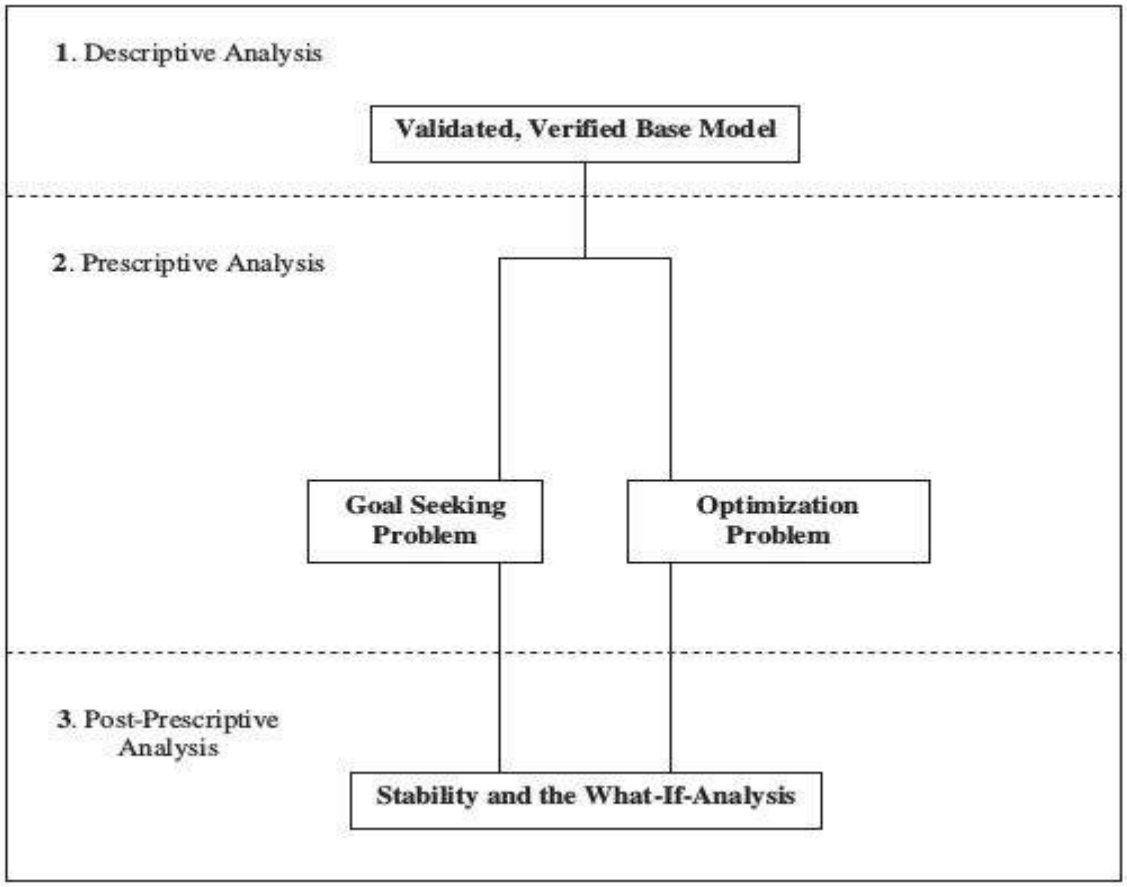
"What-if" analysis

- 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



END

TOPIC 16

Recommended Text and References

In this module
We shall assess

- NeMS coverage of contents
- Most suitable resources
    - Text books
    - References Books
    - Online Resources

NeMS contents cover

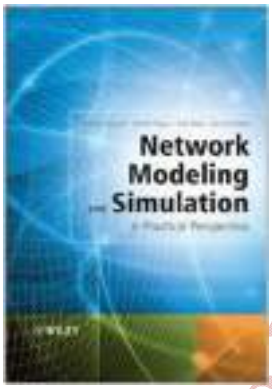
- Well known mathematical models, equations and forms
- Widely used simulation tools and code reusability
- Their inter-relationship

NeMS contents don't cover

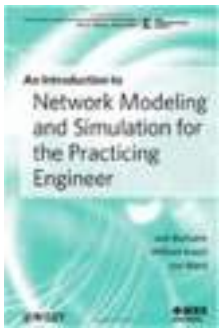
- Mathematical derivations from scratch
- Programming dexterity

Uptill now
Basics of NeMS

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



Uptill now
Basics of NeMS
Jack Burbank et al, “An Introduction to Network Modeling & Simulation for the Practicing Engineer” John Wiley , 2011



Uptill now
Basics of NeMS

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

Next Roadmap (2 of 2)

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

END

TOPIC 17

Introduction to OMNET++

In this module

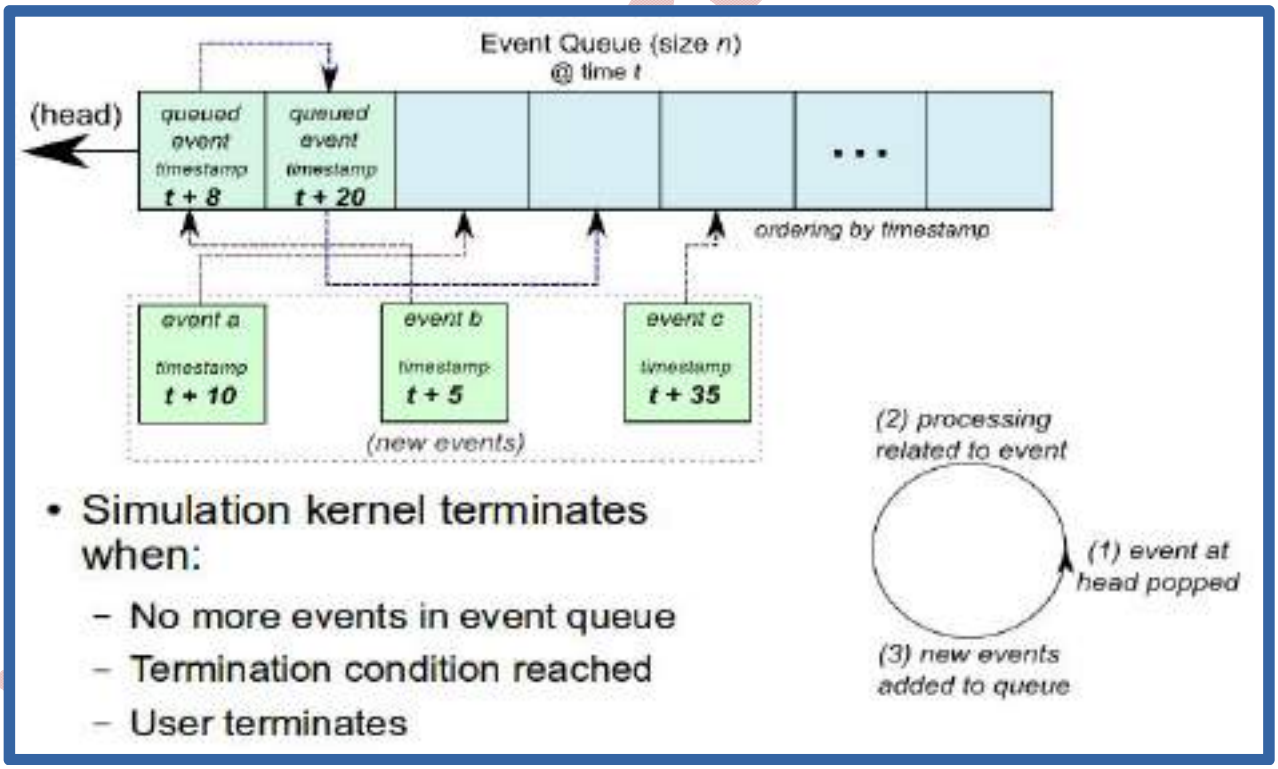
We shall cover

- What is OMNET++?
- How to get a free copy?
- How to compile and install on Windows?
- Running for the first time

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

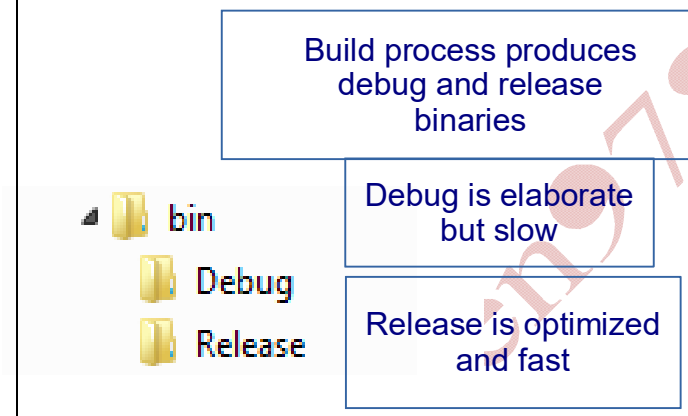
- Compiling and installing on Windows self-contained
- Enter OMNeT++ folder that you unzipped
- Run the file called Mingwenv.cmd

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



Debug mode

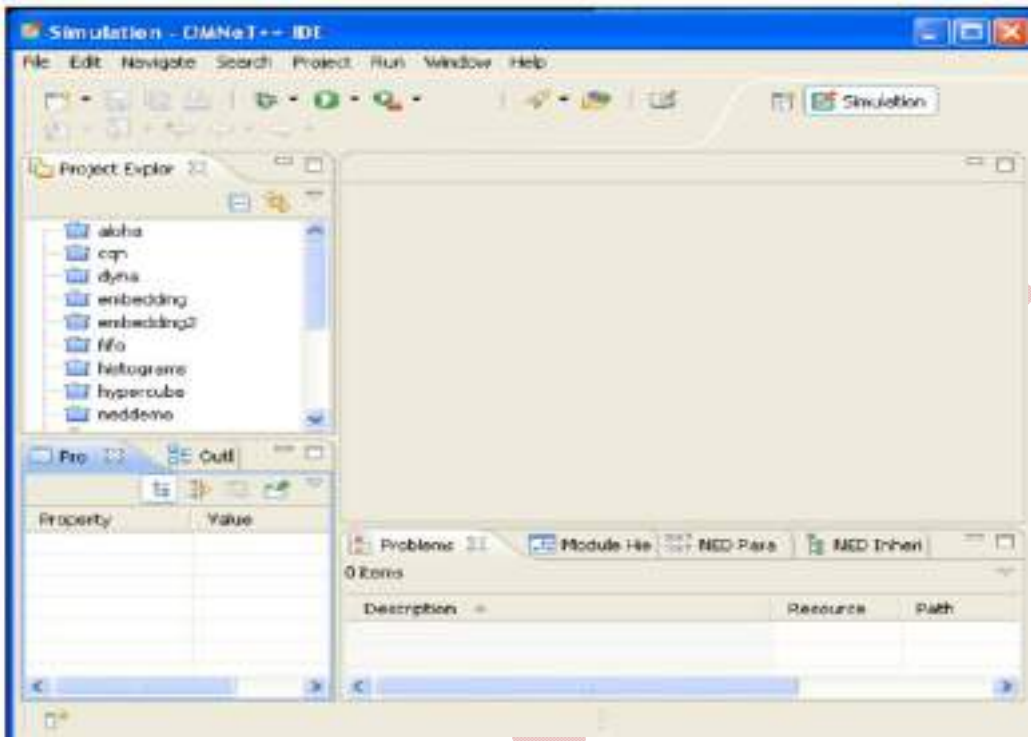
- Does not optimize the binary it produces
- Source code and generated instructions relationship is complex
- Allows accurate breakpoints setting
- Allows code step-through one line at a time
- Compiled with full symbolic debug information

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

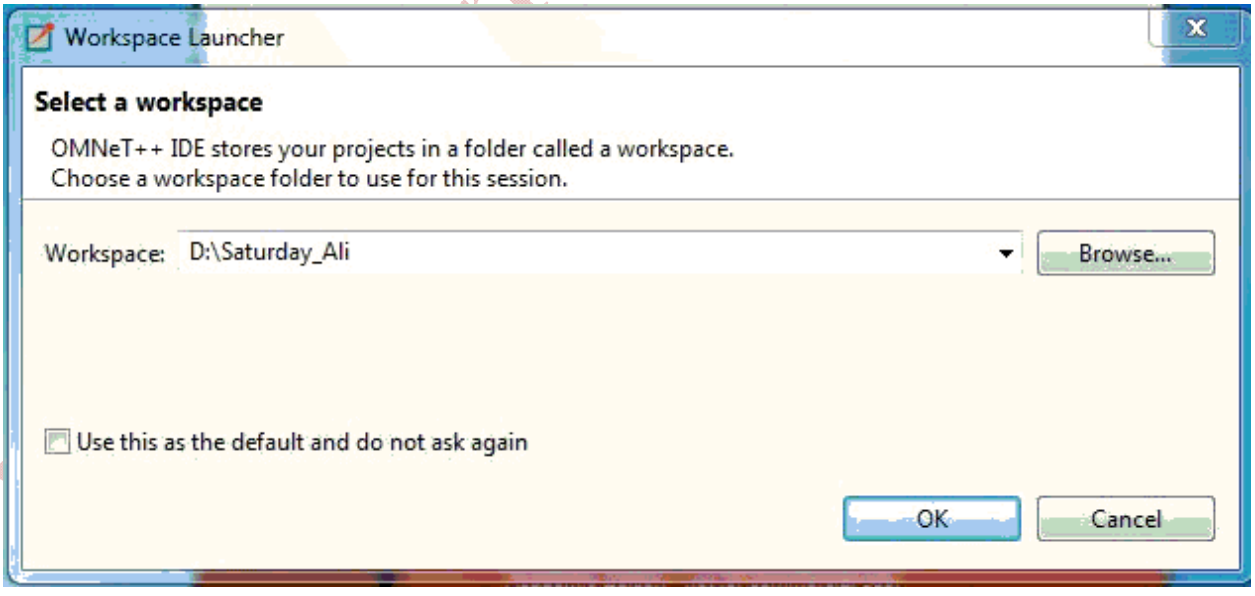
Running first time

- 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



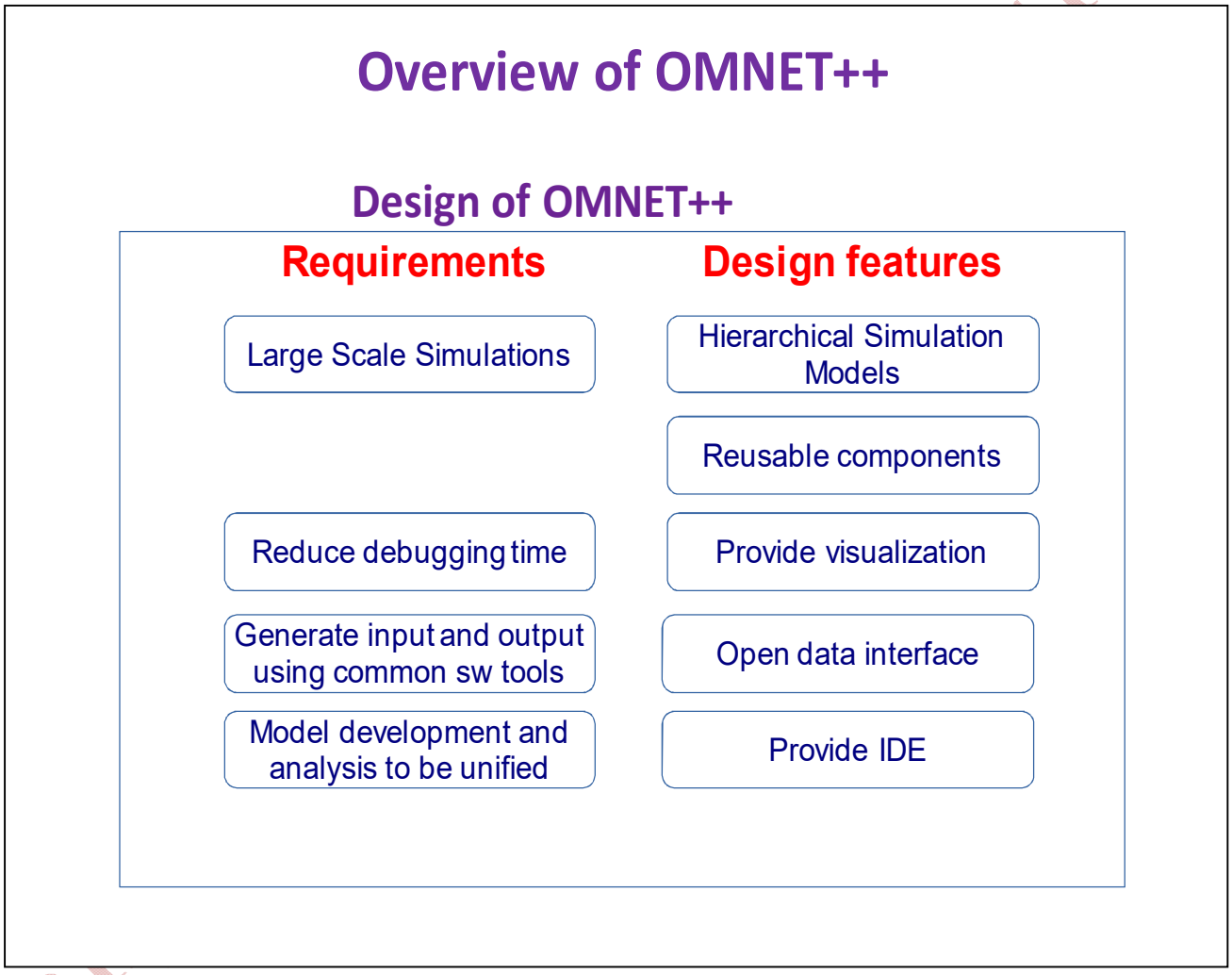
END

TOPIC 18

Overview of OMNET++

In this module
We shall cover

- Design of OMNET++
- Model structure



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

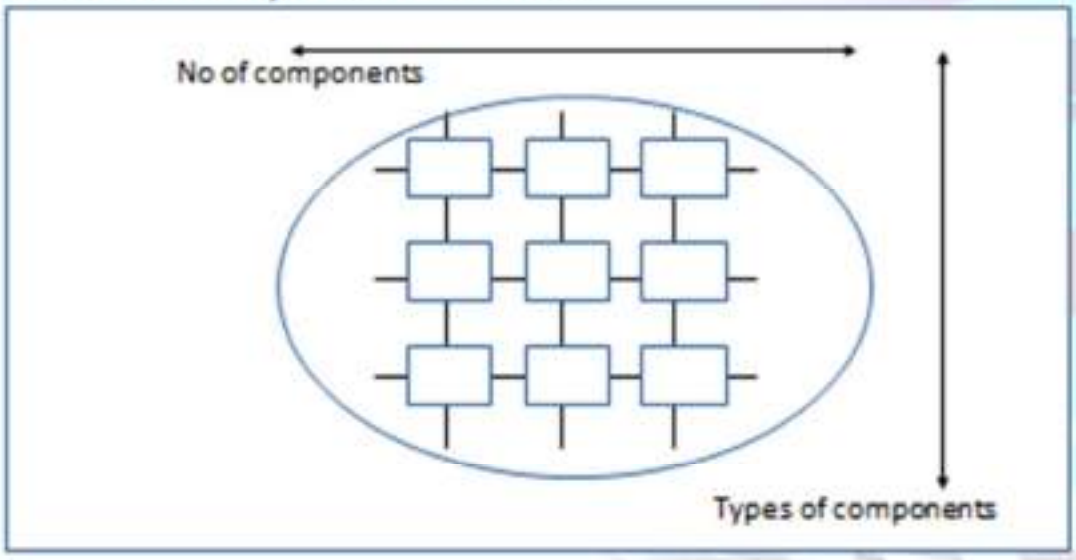
Model Structure

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

Overview of OMNET++

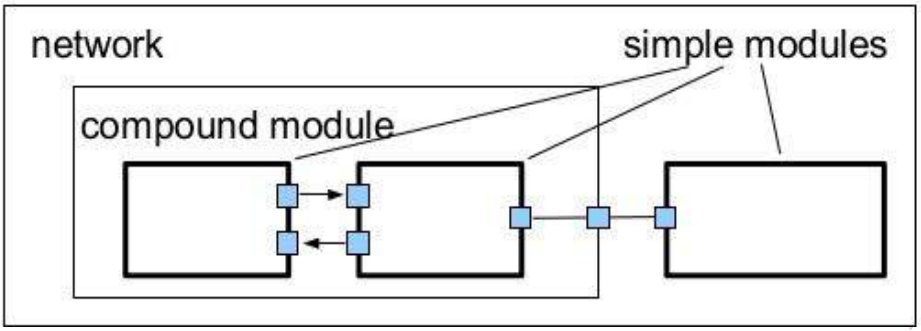
Model Structure

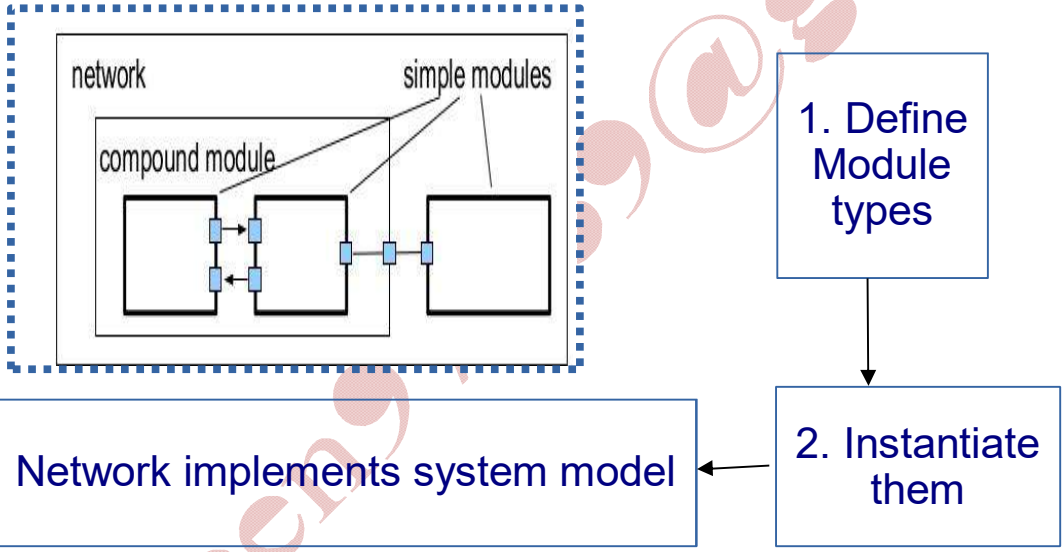
- No. of hierarchy levels not limited



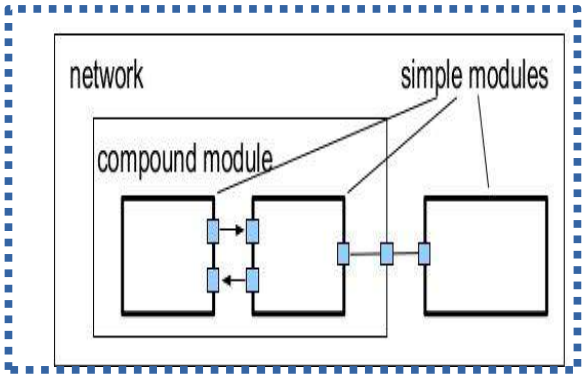
Model Structure

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





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

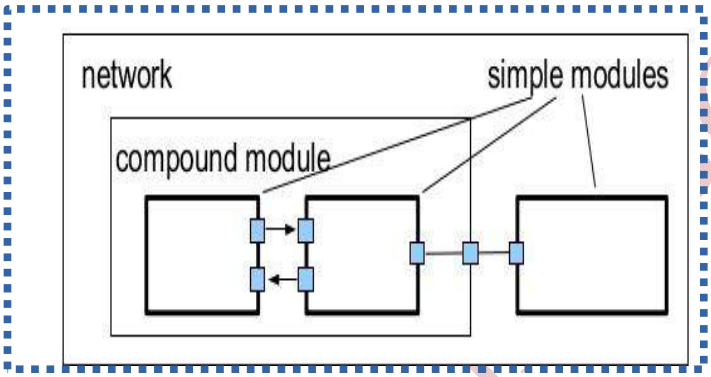


1. Define Module types

2. Instantiate them

3. Network implements system model

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



1. Define Module types

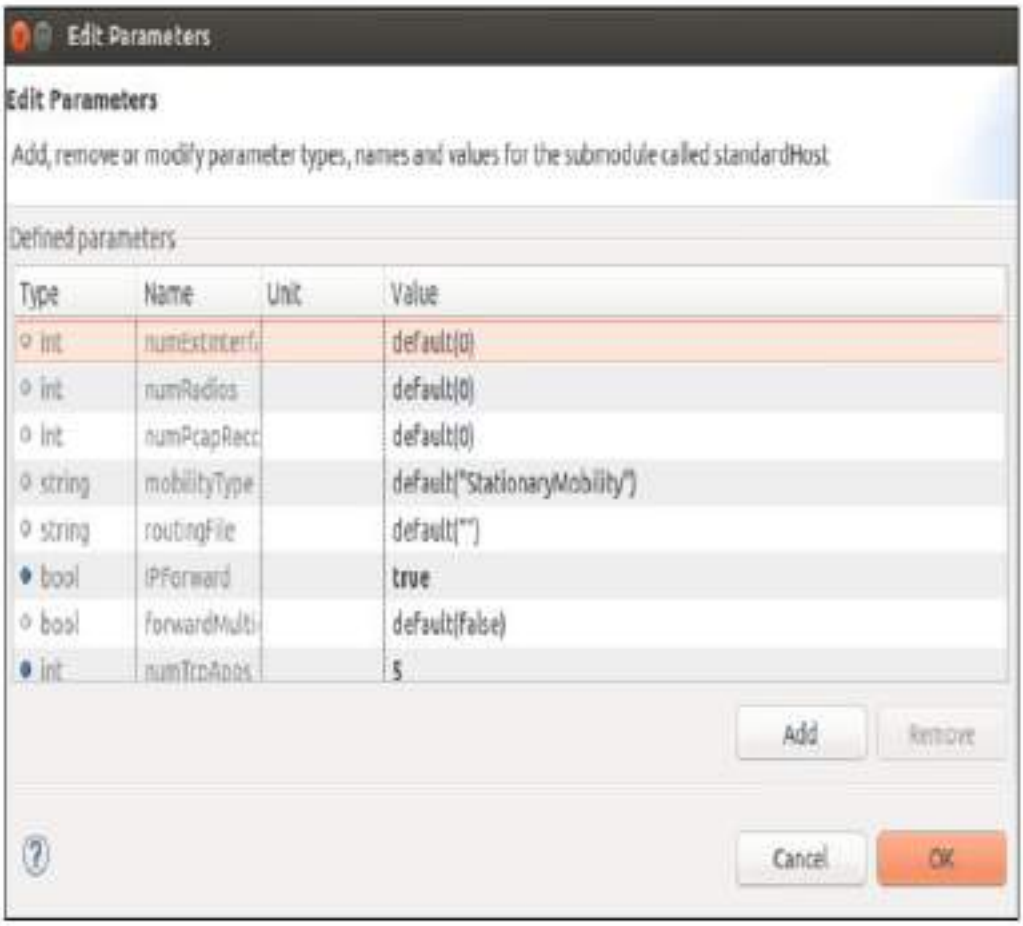
2. Instantiate them

3. Network implements system model

Module Parameters

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

Module Parameters



END

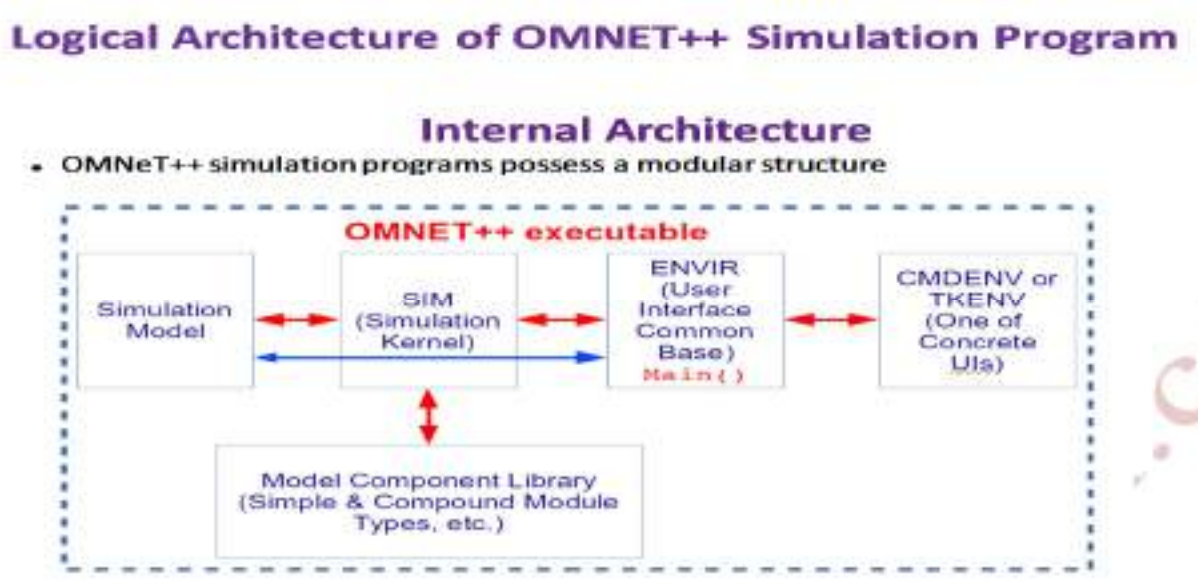
TOPIC 19

Logical Architecture of OMNET++ Simulation Program

In this module

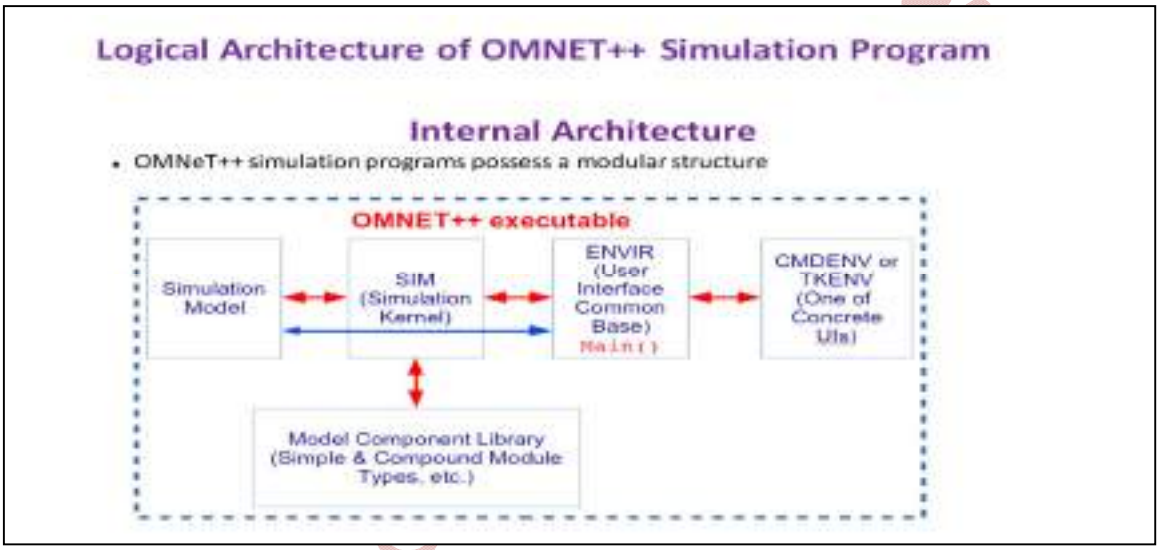
We shall cover

- Internal architecture of OMNET++
- Contents of the Simulation Library



Model Component Library

- Consists of the code of compiled simple and compound modules



Simulation Kernel & SIM Class Library (1 of 2)

- Modules are instantiated and concrete simulation model is built by the simulation kernel
- SIM covers most of the common simulation tasks through classes
  - Generate random number (distributions)

Queues (FIFO, priority)

Simulation Kernel & SIM Class Library (2 of 2)

- 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

TOPIC 20

Overview of OMNET++

In this module

- We shall cover
- Introduction to NED Language
- Graphical Editor

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

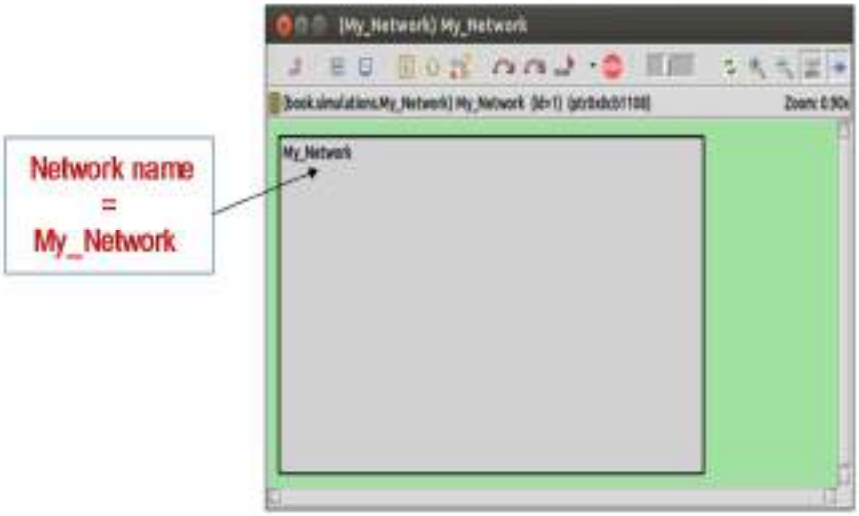
- Describes the interface of modules
  - Gates
  - Parameters

Compound module definitions

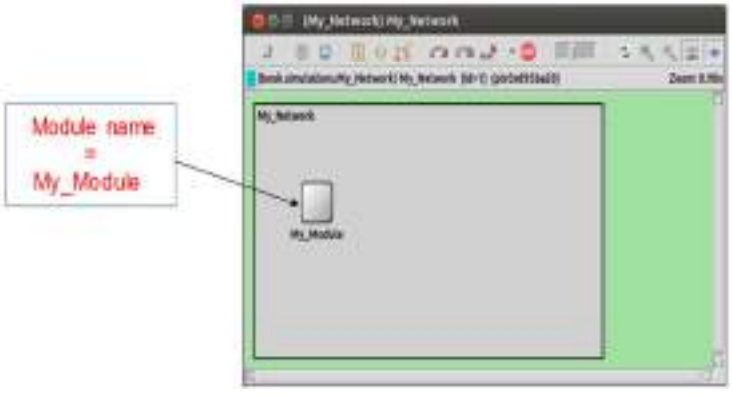
- Declaration of external interface
  - Gates
  - Parameters
- Definition of
  - Submodules
  - Their interconnections

Overview of OMNET++

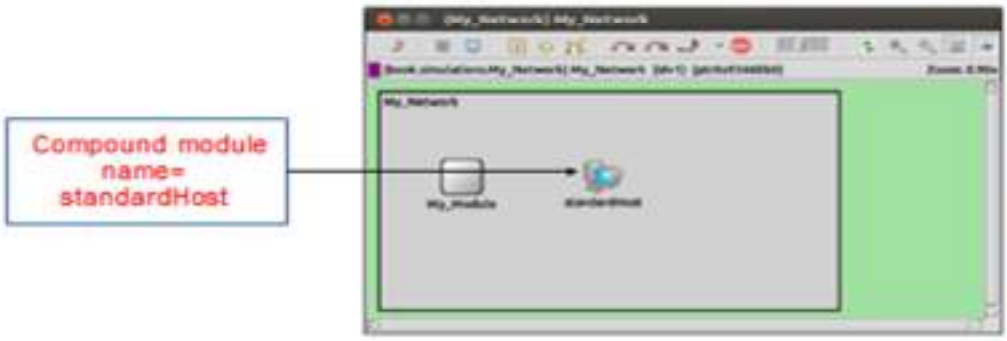
Let us create a topology called My_Network using Graphical Editor



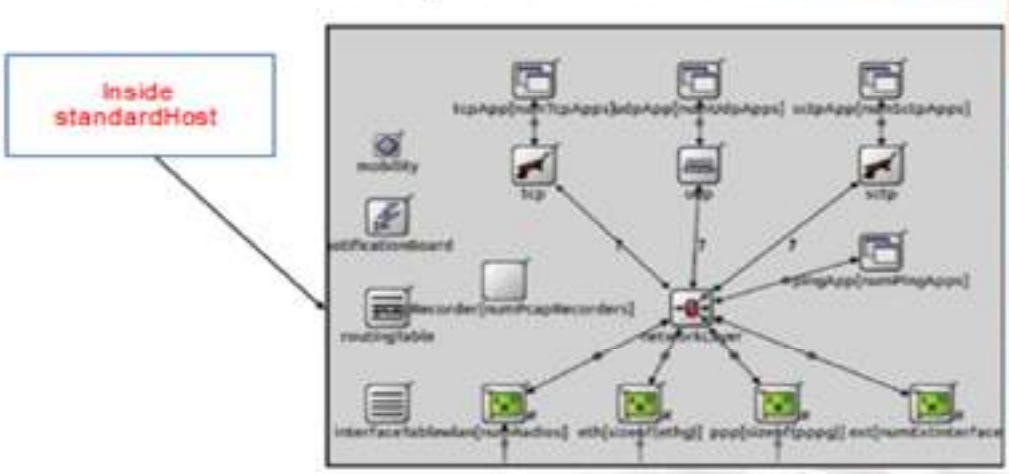
Overview of OMNET++



Overview of OMNET++



Overview of OMNET++



TOPIC 21

More About NED Language

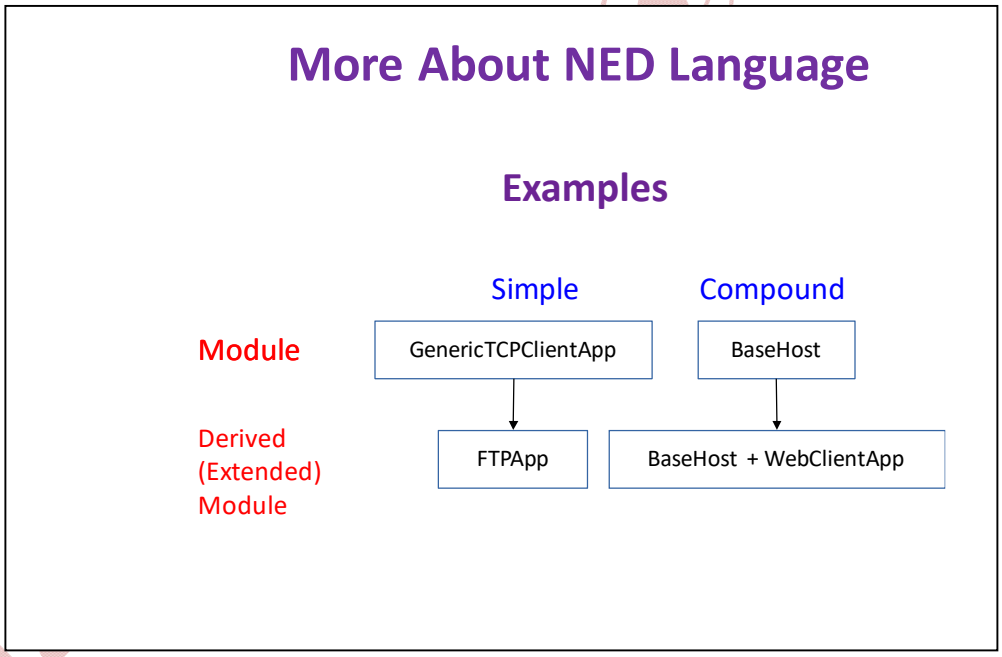
In this module

We shall cover more
Details about NED

- Inheritance
- Interfaces
- Packages
- Inner types

Inheritance

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

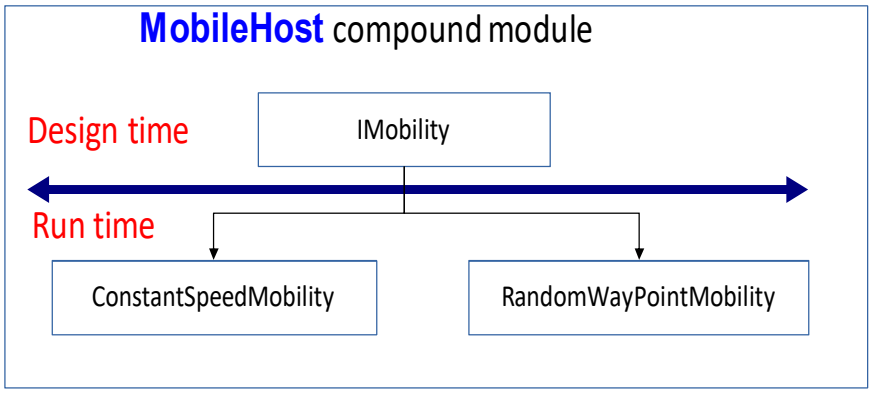


Interface instantiation

- 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
  -

More About NED Language

Example

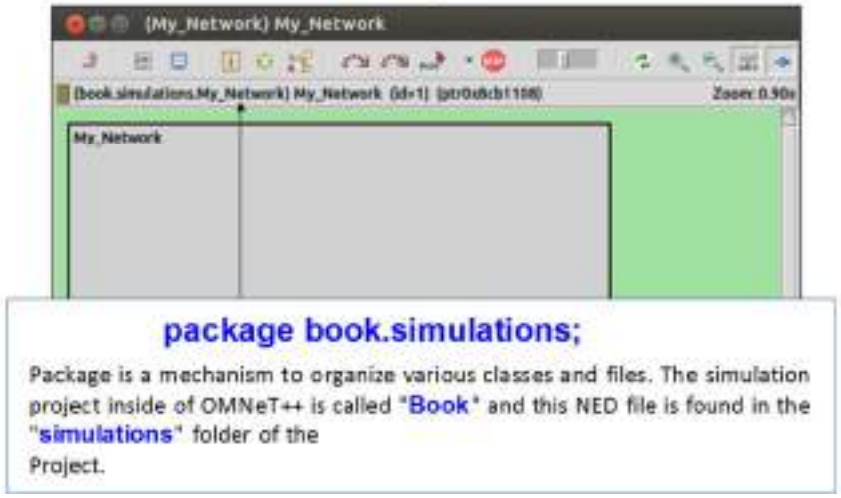


Packages

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

More About NED Language

Example



TOPIC 22

Configuring OMNET++ Simulations

In this module
We shall understand
- Need for separating models and experiments
- Configuring simulations
- INI files
- OMNET++ INI file Editor

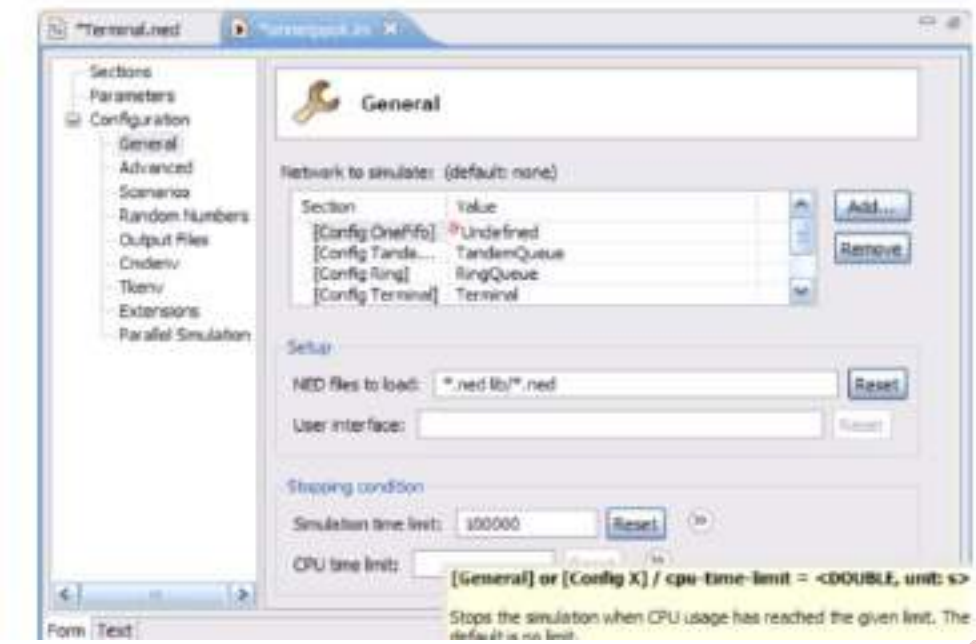
Separation of Model and Experiments
- Always good practice to try to separate different aspects of simulation
- **Model topology**
  - NED file
  - MSG file
- **Model behavior**
  - C++ code
- Provides cleaner model

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

Configuring OMNET++ Simulations

Example omnet.ini

My_Network *wildcarded* as **

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
**.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"
```

Configuring OMNET++ Simulations

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"
#**.ppp[*].queue.frameCapacity = 10
#**.eth[*].queueType = "DropTailQueue"
```


Configuring OMNET++ Simulations

Example

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"
```

Configuring OMNET++ Simulations

Example

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"
```

Configuring OMNET++ Simulations

Example

Who to connect with whom

```
[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"
```

Configuring OMNET++ Simulations

Example

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"
```

Configuring OMNET++ Simulations

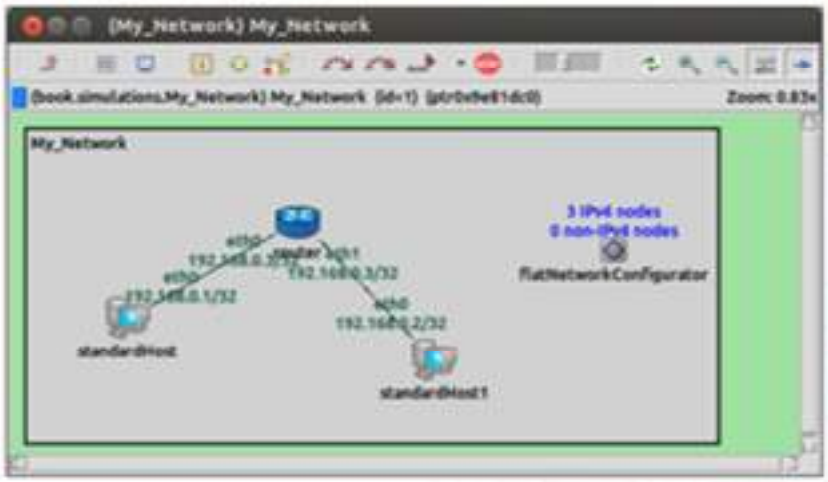
Example

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"
```

Configuring OMNET++ Simulations

Example



TOPIC 23

BUILDING SIMULATION:

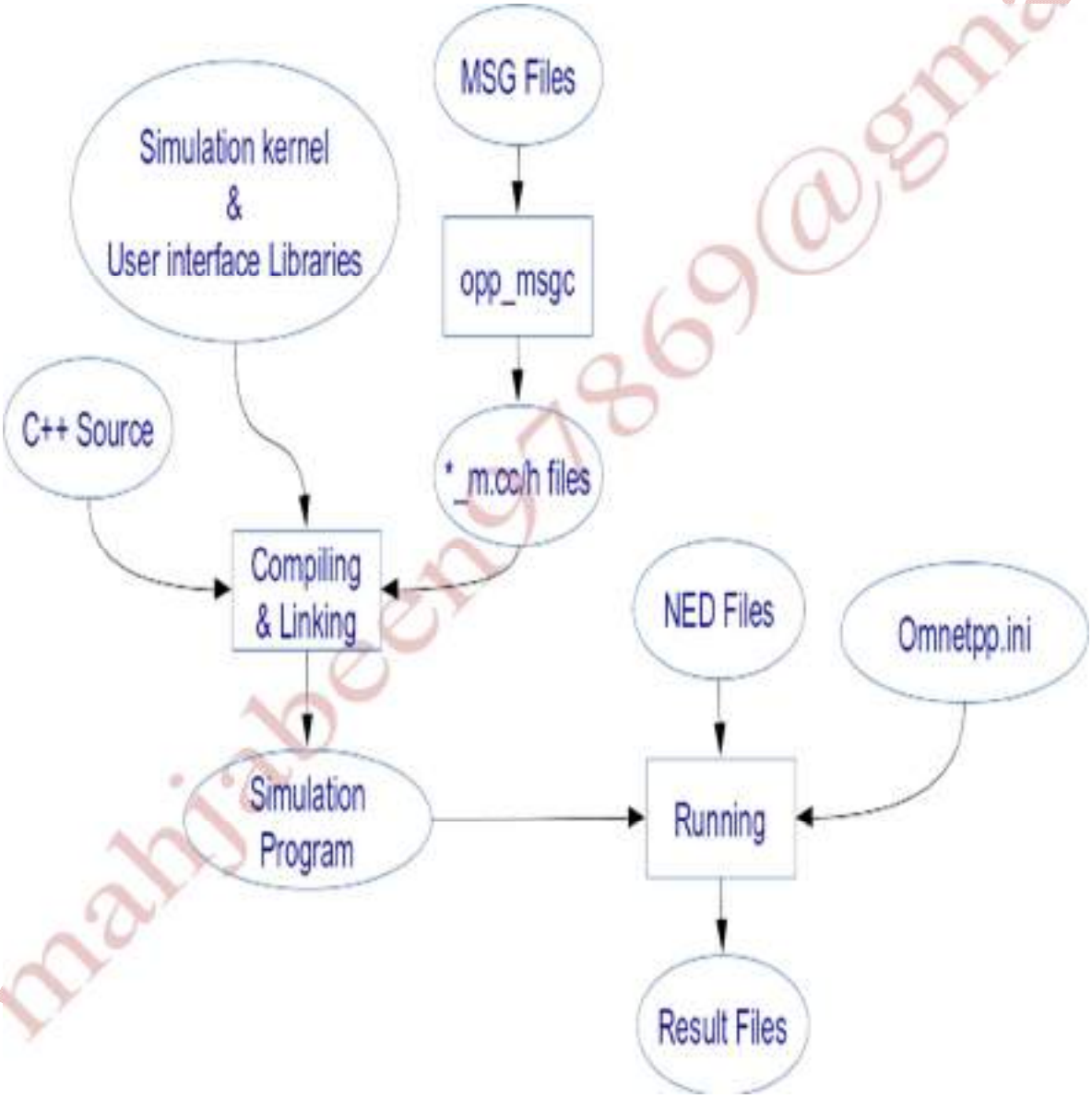
In this module

We shall cover

- Build Process
- How to build
- OMNET++ simulations

Building Simulation Programs

End-to-End Process
(Build + Configure+ Run + Analyze)



Process of Build

- Same as building

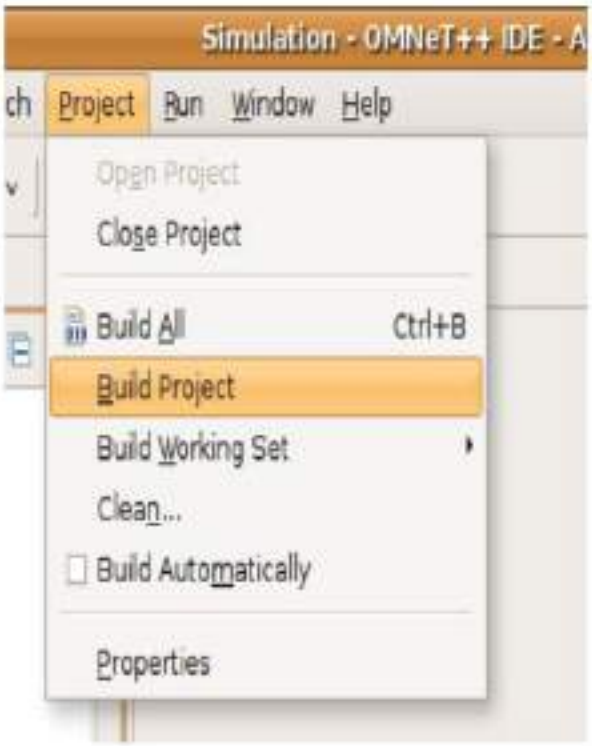
any C/C++ program from source

  - All C++ sources need to be compiled into object files
  - All object files need to be linked with

necessary libraries to get

    - Executable
    - Or shared library

- Initial build takes longer on indexing before building the project
- Dependency generation in the generated makefiles
    - Classes, functions, methods, variables, macros



Building Simulation Programs

Using Mingwenv

- Once you have the source files (*.ned, *.msg, *.cc, *.h) in a directory
  - Change the working directory to there
- Type

$ opp_makemake

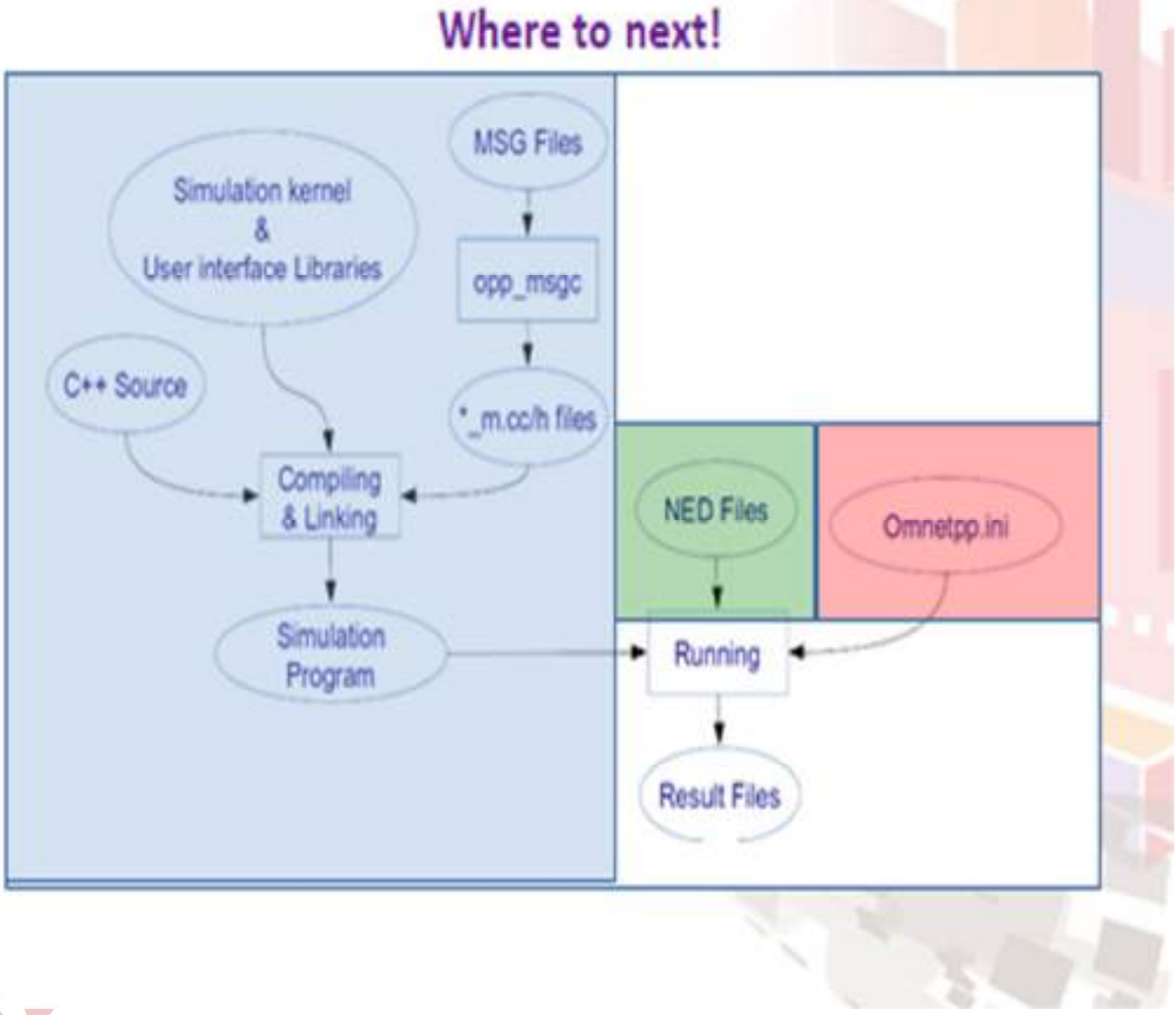
- This will create a file named Makefile
- Type

$ 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.

Building Simulation Programs



TOPIC 24

RUNNING SIMMULATION

In this module

We shall cover

- What is a simulation run?
- Quick Run
- Creating run configuration

What is Simulation Run?

- Launch the built project make file

OMNET++ IDE Features

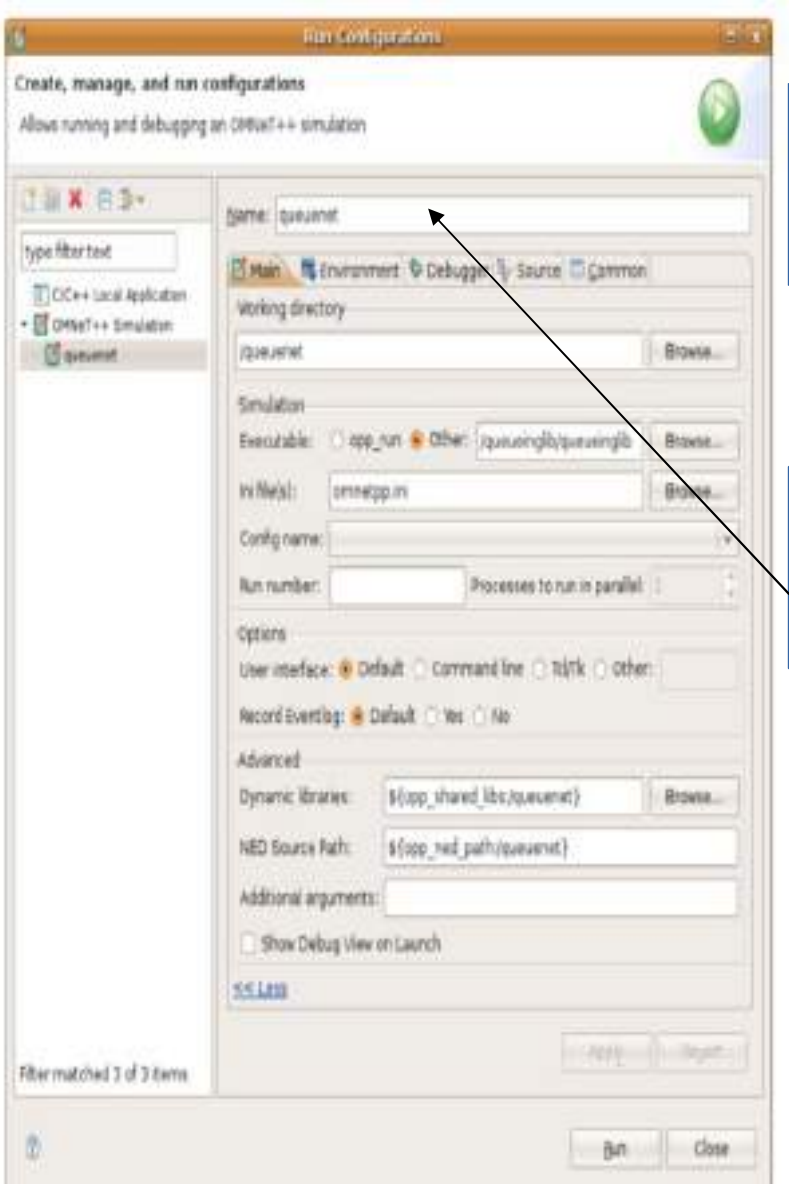
- Single runs
- Batch runs
- Run numbers
- Graphical mode (Tkenv)
- Command mode (Cmdenv)
- Simulation configuration
- Recording event logs
- Debug support

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

Running Simulations

Launch Configuration

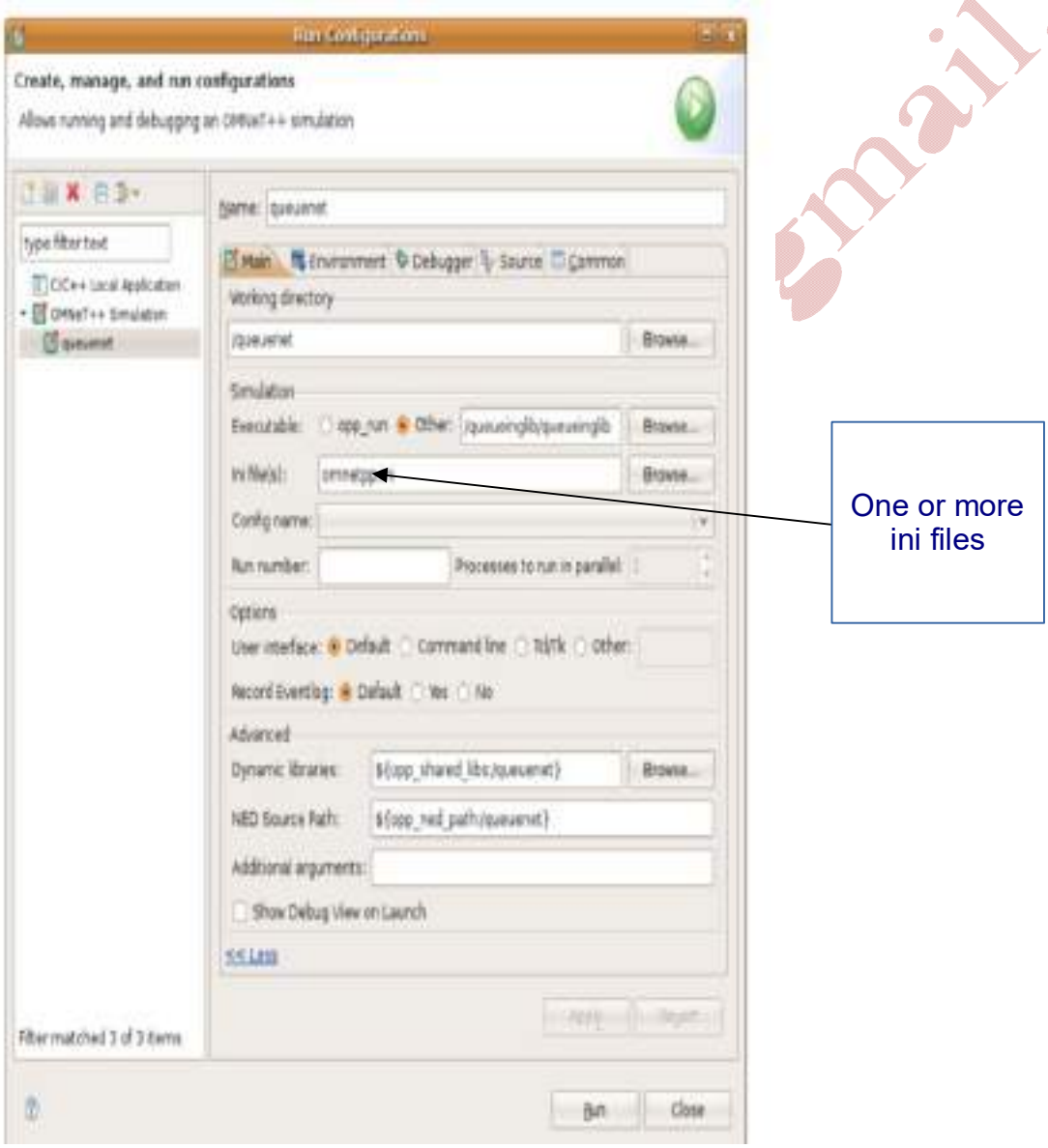


Run omnet.ini
From /queuenet

queuenet Launch
Configuration

Running Simulations

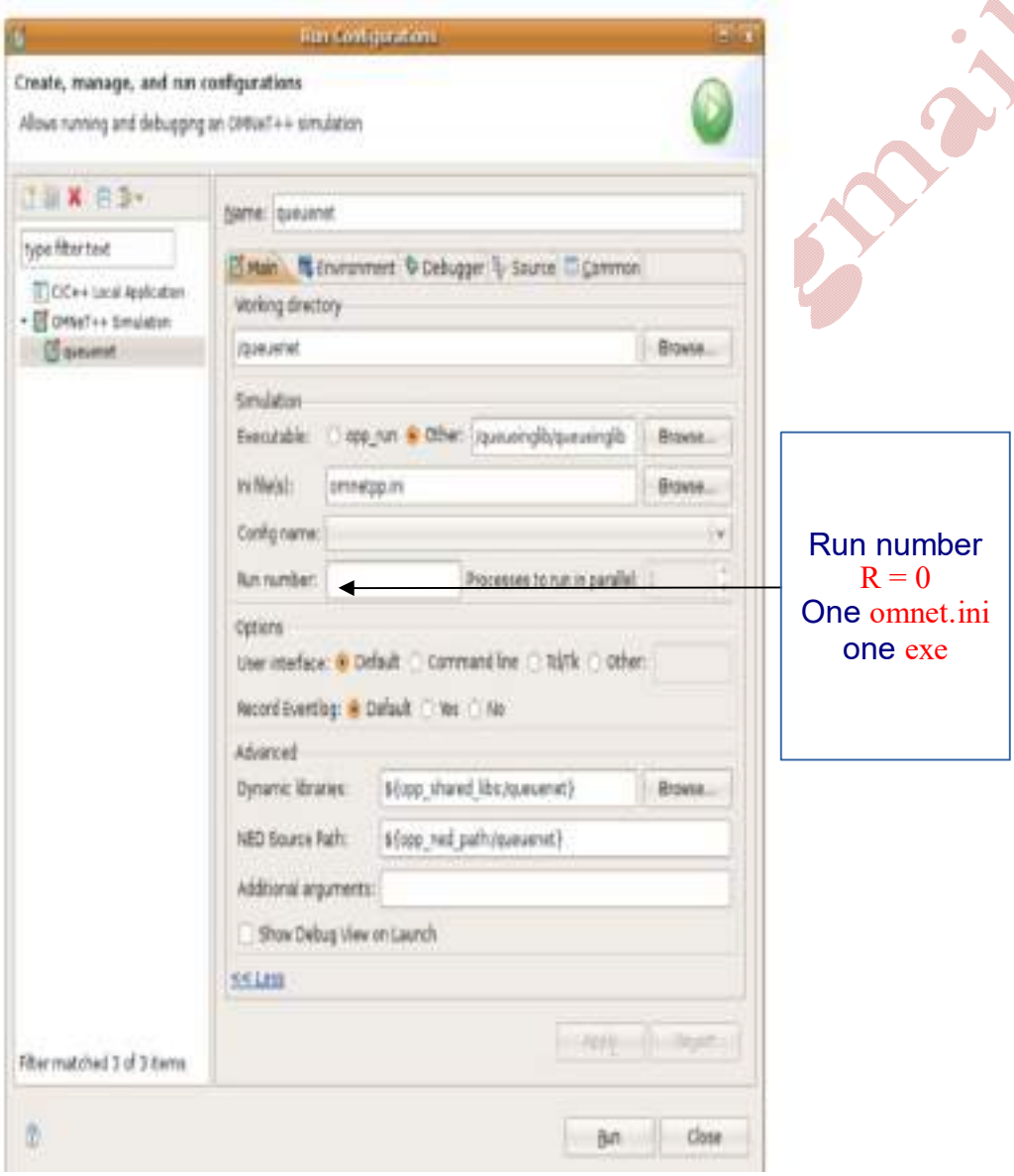
Launch Configuration



One or more ini files

Running Simulations

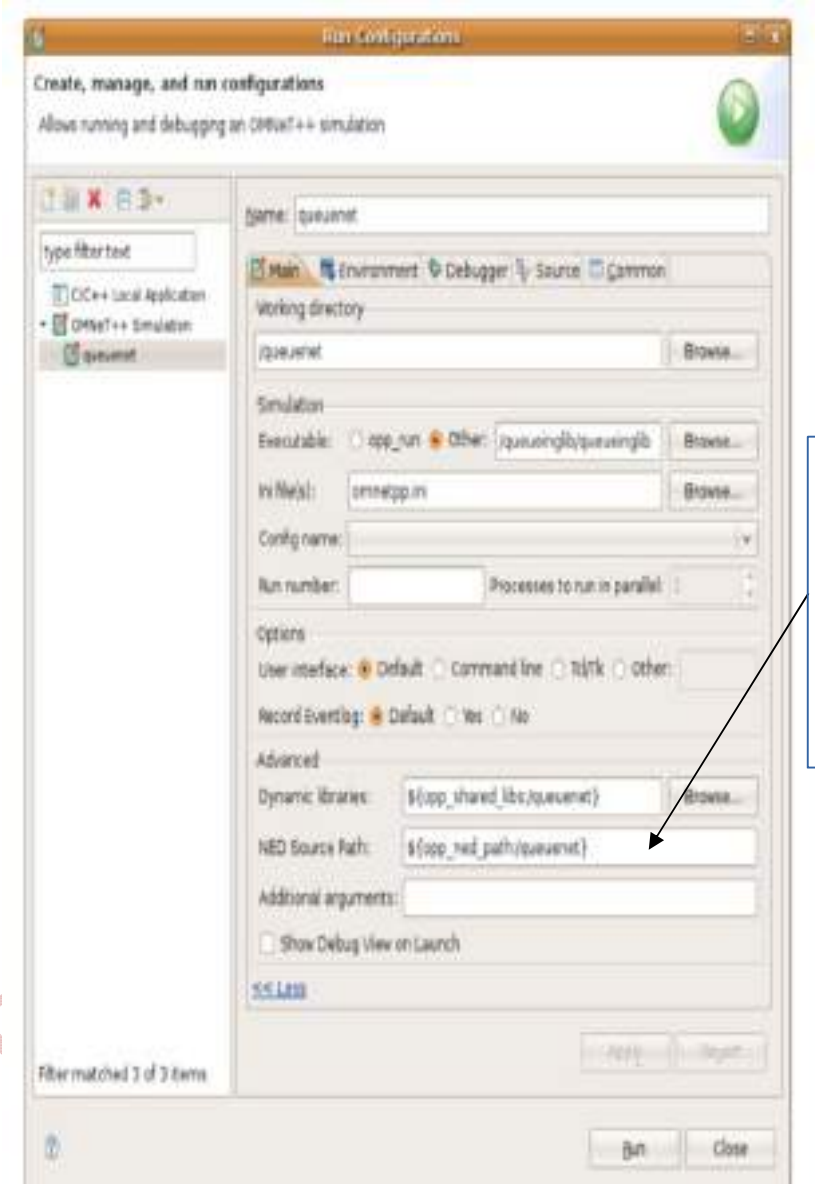
Launch Configuration



Run number
R = 0
One omnet.ini
one exe

Running Simulations

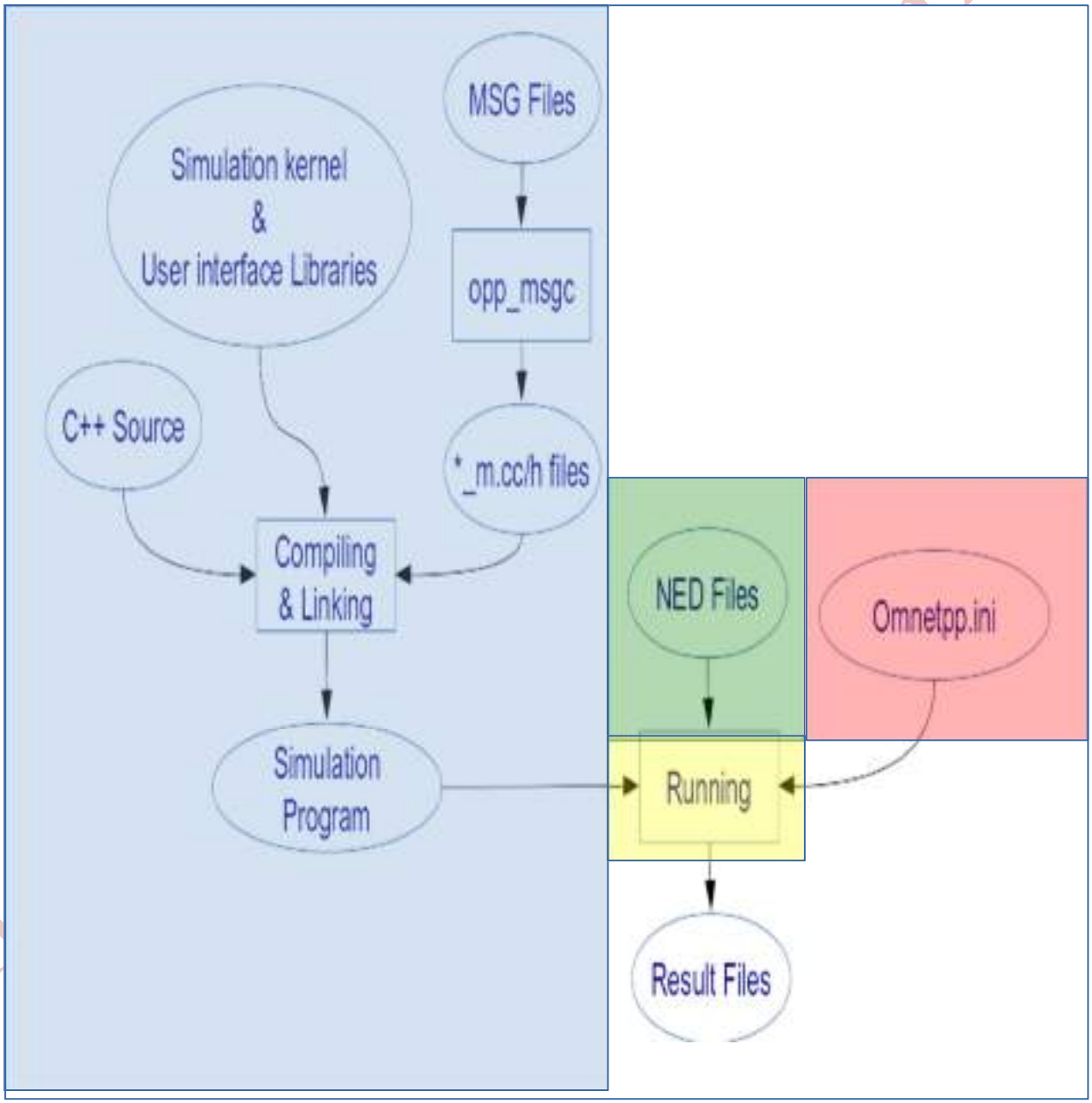
Launch Configuration



Directories where the NED files are read from

Running Simulations

Where to next!



TOPIC 25

Animation and tracing

In this module

We shall cover

- What is animation of simulation?
- What is traceability of simulation models?
- Object inspection
- Tkenv and its feature set

Animating Simulation(1 of 2)

OMNeT++ is capable of

- Animating
  - Flow of messages on network charts
- Reflecting
  - State changes of the nodes in the display

Animating Simulation(2 of 2)

- 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 (1 of 2)

Tkenv is a graphical runtime interface for simulations

- It provides
  - Network visualization
  - Message flow animation
  - Log of message flow
  - Display of textual module logs

Tkenv (2 of 2)

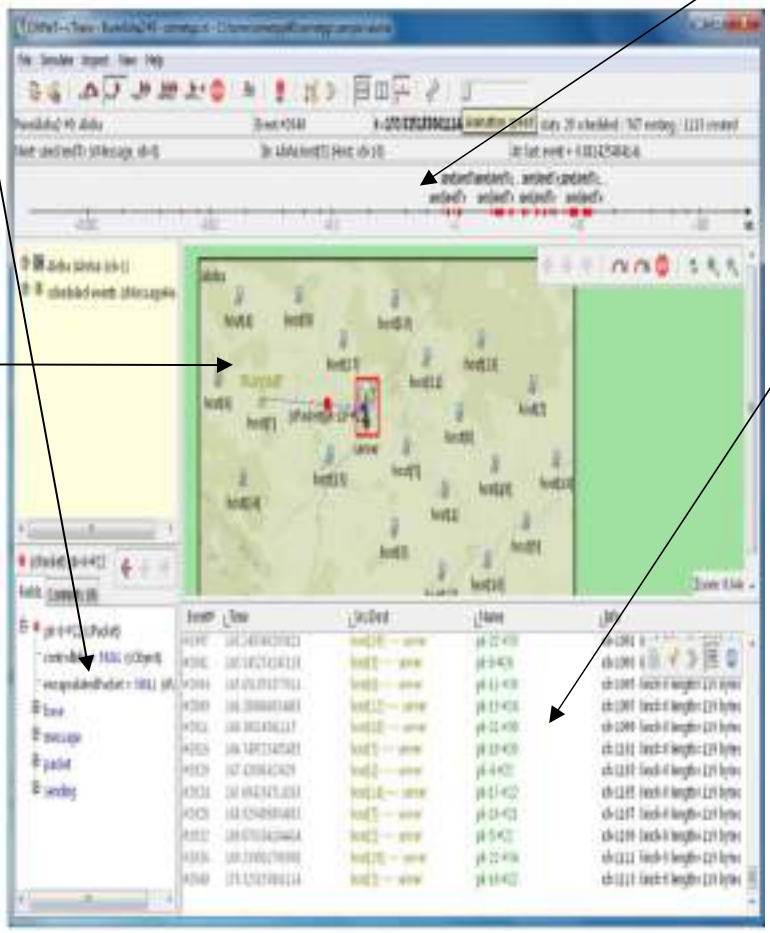
- Inspectors
- Visualization of statistics
  - Histograms, etc. during simulation execution
- Event log recording for later analysis

Animation and Tracing

Tkenv in action

Object inspector

Timeline
Future Events Set (FES) on log scale



Network display

Log viewer

TOPIC 26

Organizing and Performing Experiments

In this module
We shall understand
- Need for organizing experiments
- How to organize and perform experiments

Need for organizing experiments(1 of 4)
- Stuart Kurkow, "MANET Simulation
- Studies:
- The Incredibles," ACM's MobileComputing and Communications
- Review, 9(4):
- 50-61, 2005

Need for organizing experiments (2 of 4)

Repeatable
- Fellow researcher should be able to repeat

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

Need for organizing experiments (3 of 4)

Rigorous
- Scenarios & conditions for experiments must be truly representative

Statistically sound
- Experiments results must not violate mathematical principles

Organizing and Performing Experiments

Need for organizing experiments (4 of 4)

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

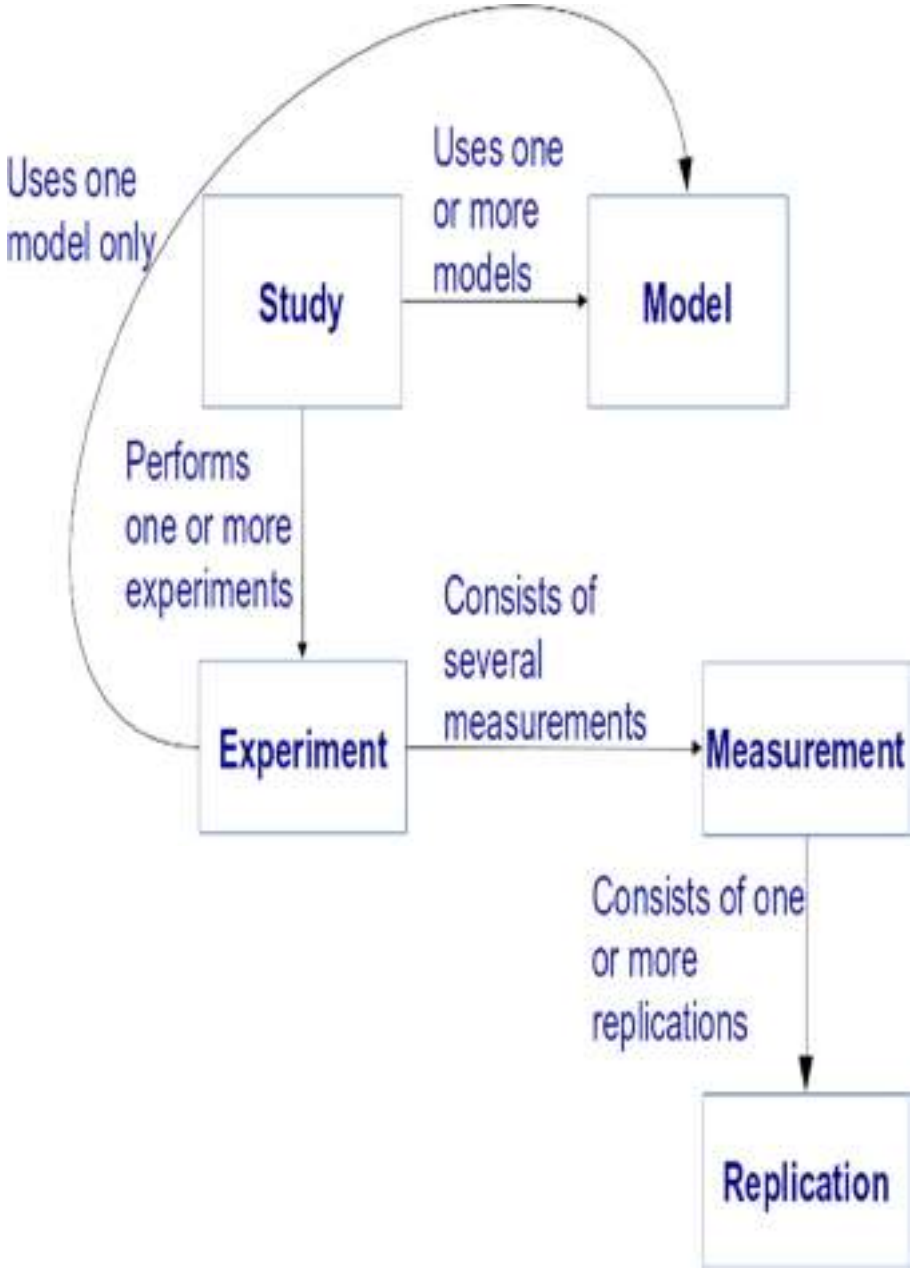
98 of 112 (87.5%) did not include confidence intervals

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

etc.

Organizing and Performing Experiments

Relationship between terminologies



How to organize experiments
(1 of 6)
Model

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

How to organize experiments
(2 of 6)
Study

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

How to organize experiments
(3 of 6)
Experiment

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

How to organize experiments
(4 of 6)
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

How to organize experiments
(5 of 6)
Replication

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

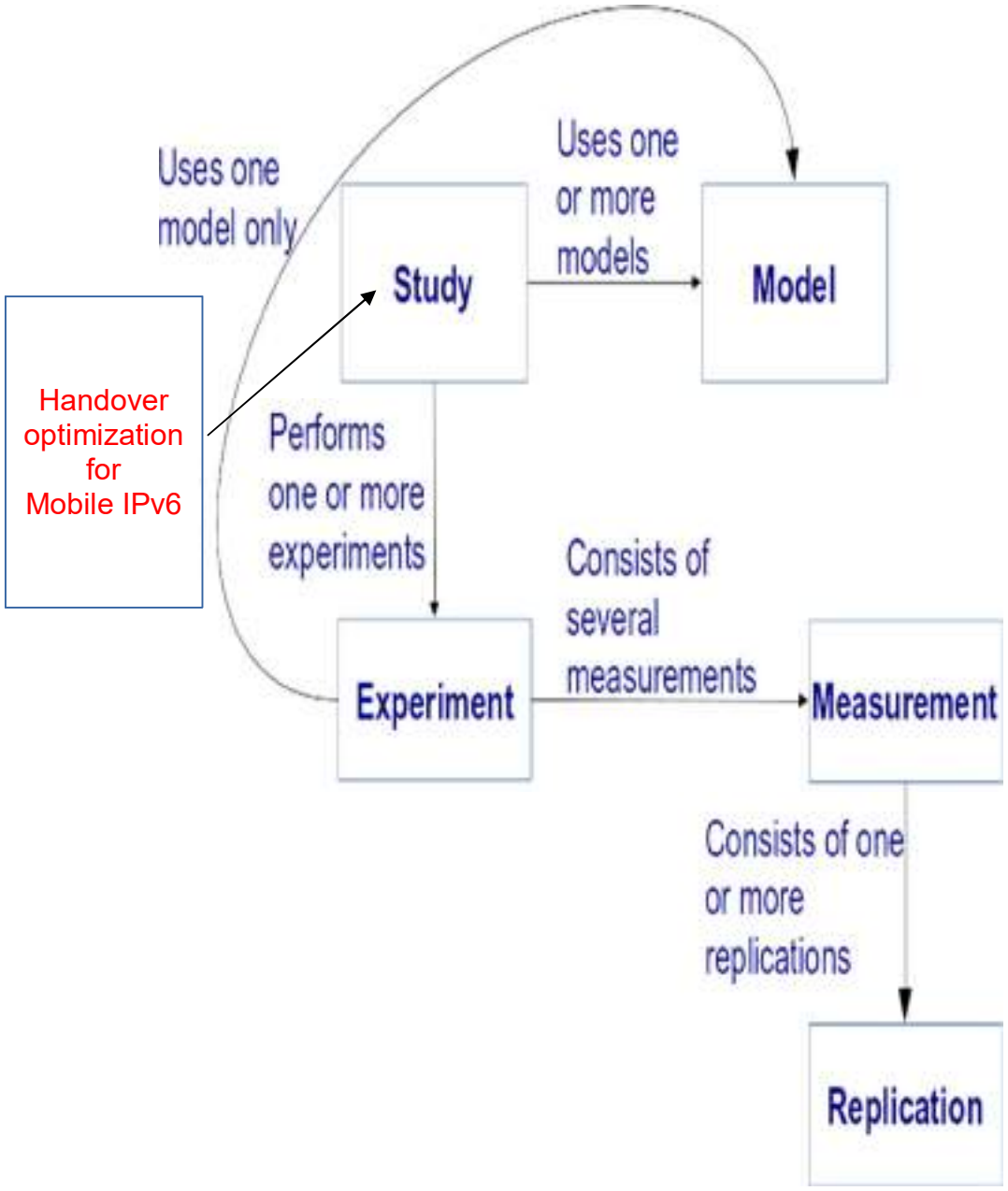
How to organize experiments
(6 of 6)
Run

- One instance ofrunning the simulation
- Characterized by
  - exact time/date

- computer (host name)

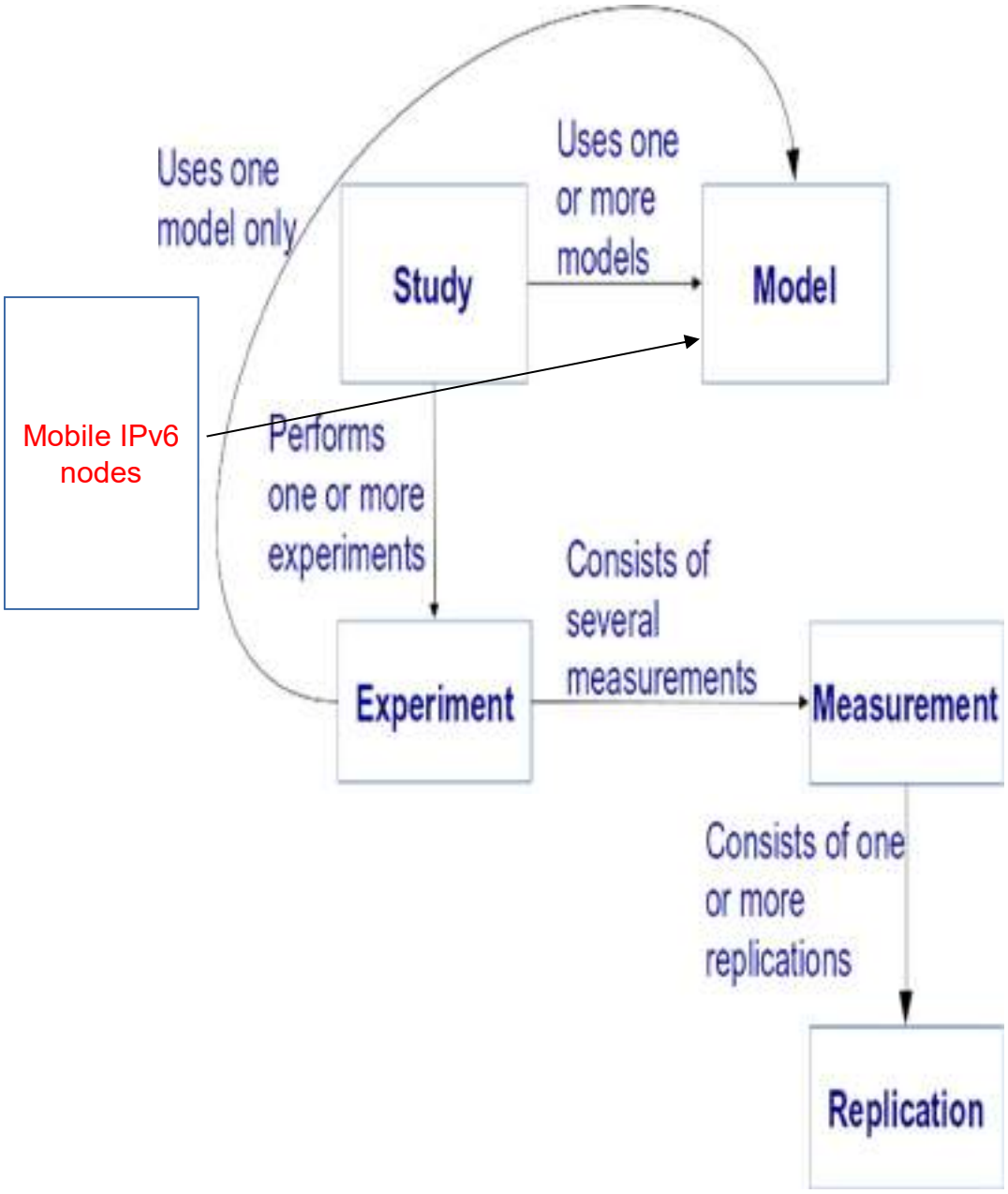
Organizing and Performing Experiments

Example



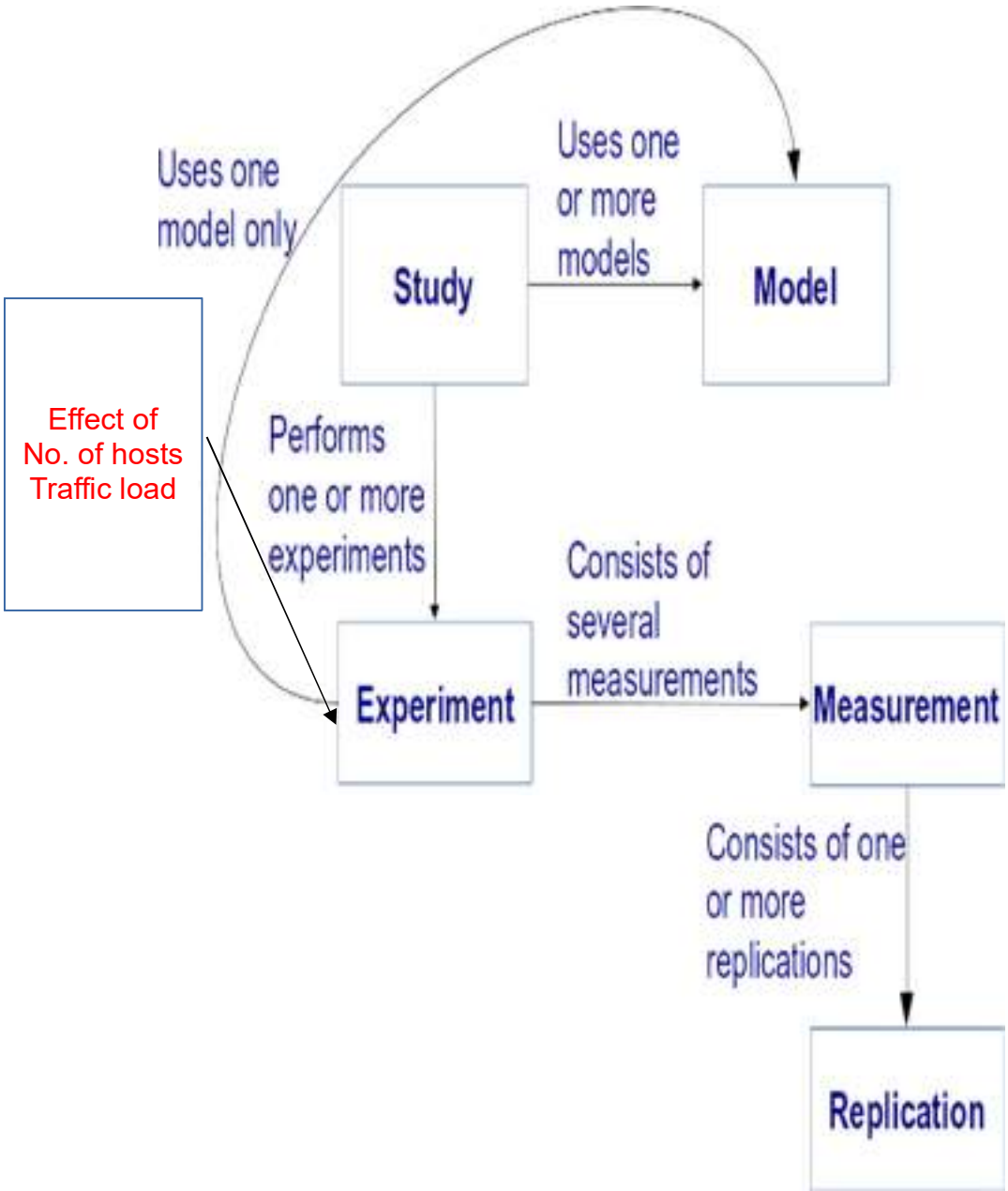
Organizing and Performing Experiments

Example



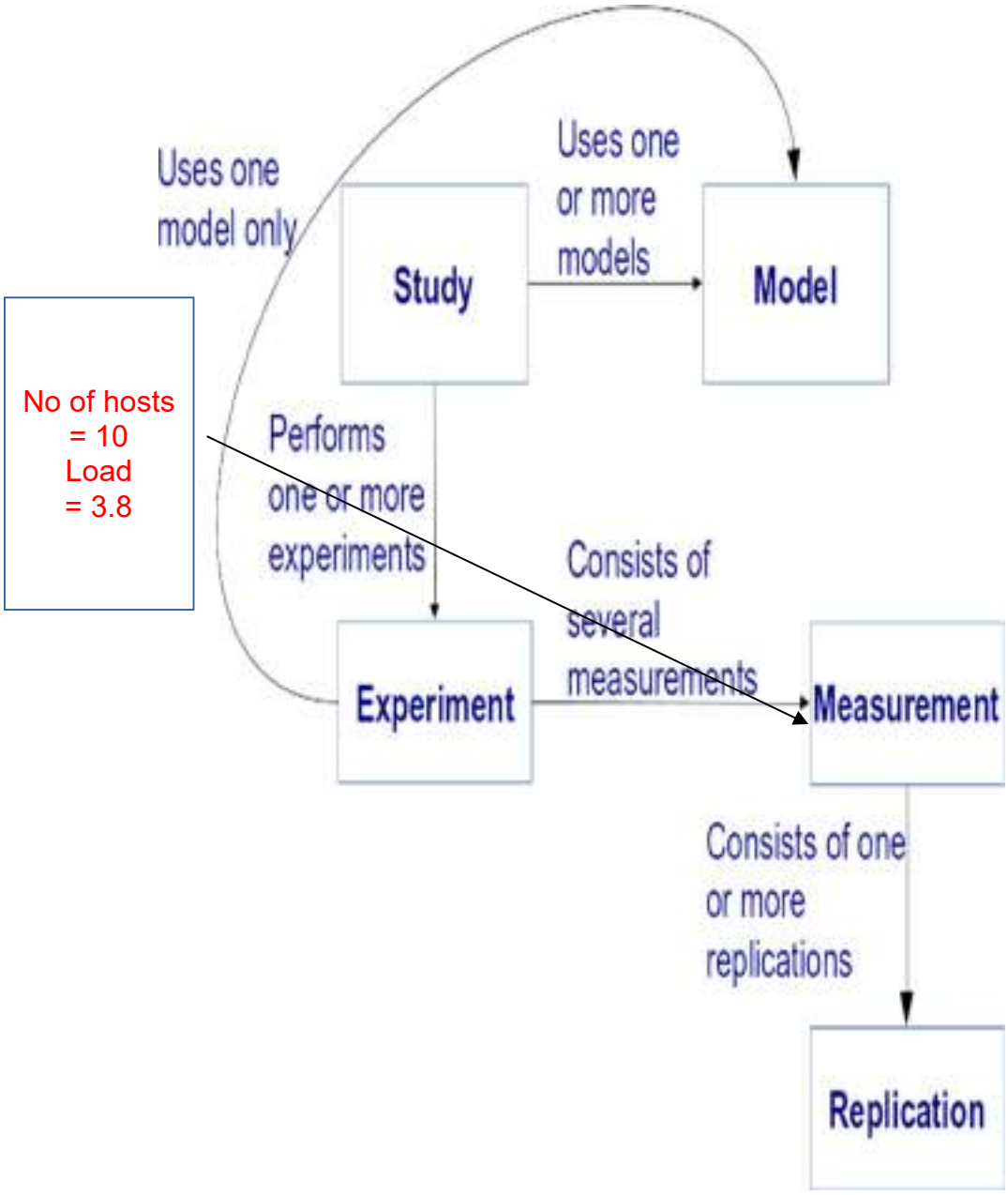
Organizing and Performing Experiments

Example



Organizing and Performing Experiments

Example



TOPIC 27

Sequence Charts

In this module
We shall explore
- What are event log tables?
- What are sequence
- charts?

Event Log Tables
(1 of 2)
- An eventlog file contains
  - — Tabulated log of messages sent during simulation
    - Between modules
    - Self-messages (timers)
  - — Event details that prompts such sending or reception

Event Log Tables
(2 of 2)
- User can control
  - — Amount of datamrecorded 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

Event Log File Creation (2 of 2)
Using INI file event log configuration

Record event

Sequence Chart

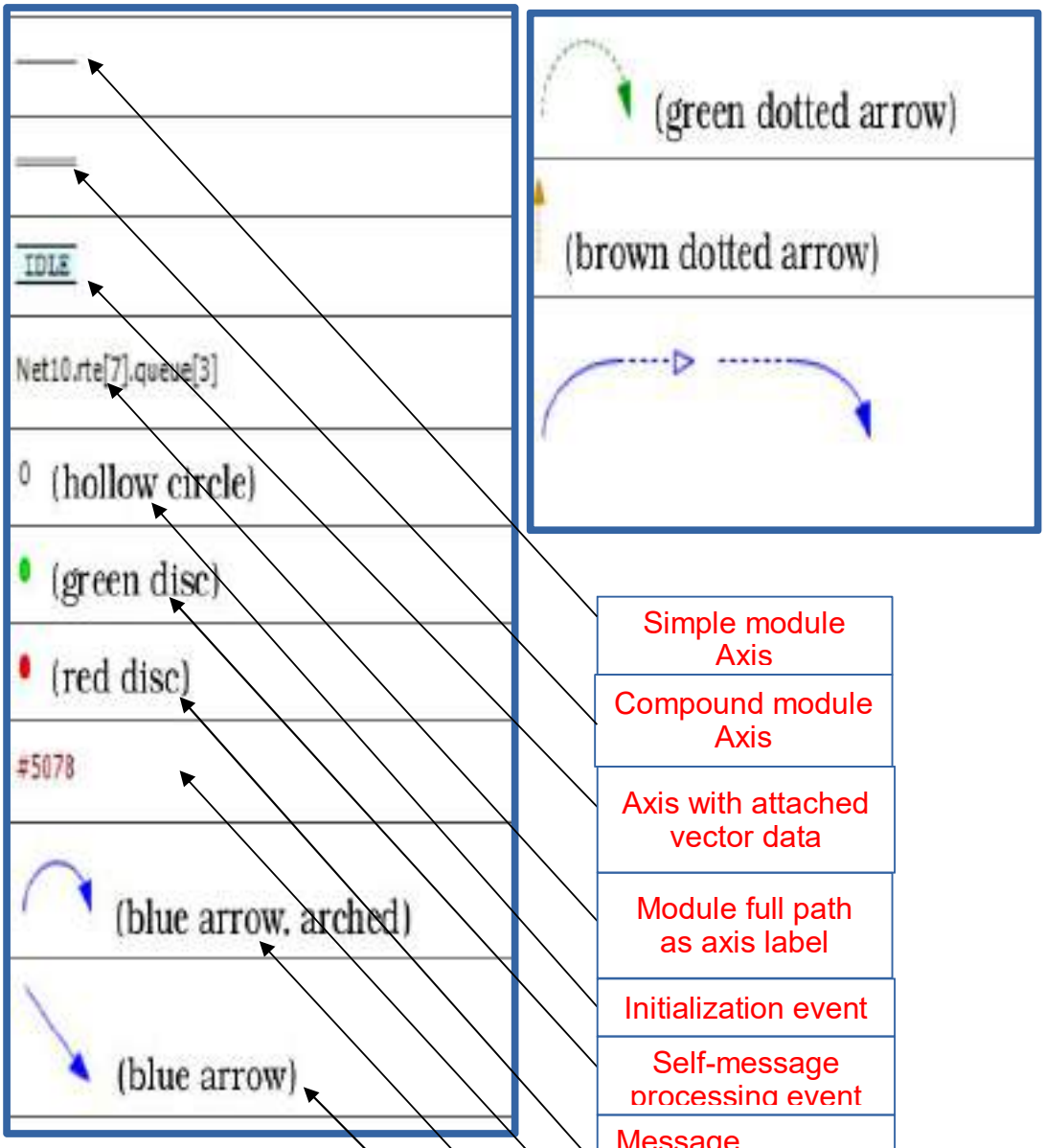
- Displays eventlog files

in a graphical form

- Helps focus on causes & consequences of events/messages
- Helps users understand
  - —Complex simulation models
  - —Verify implementation for desired behavior

Sequence Charts

Understanding the Legend



Simple module Axis

Compound module Axis

Axis with attached vector data

Module full path as axis label

Initialization event

Self-message processing event

Message processing event

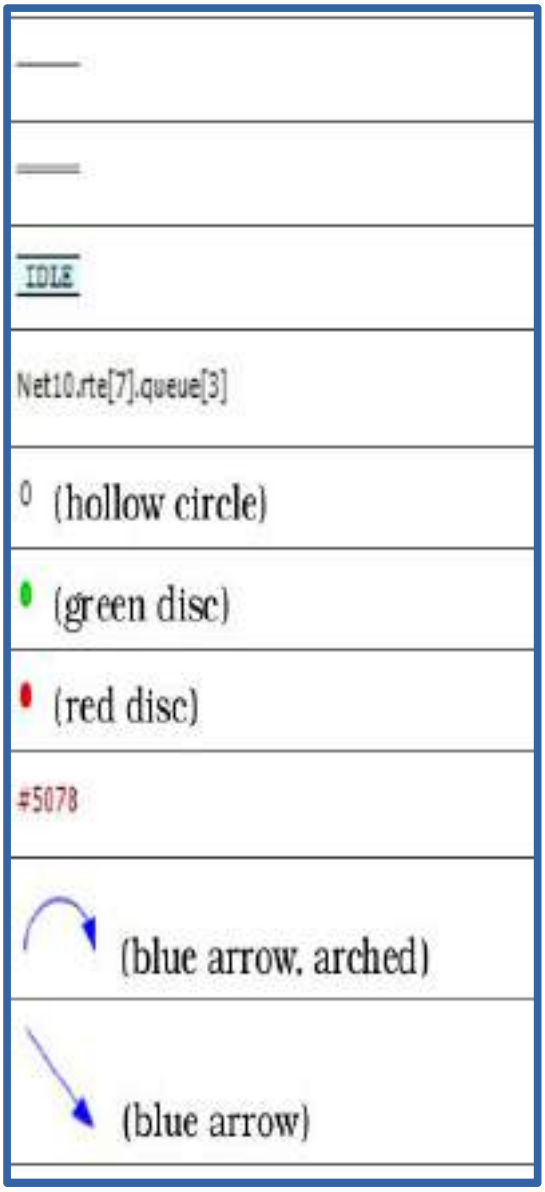
Event number

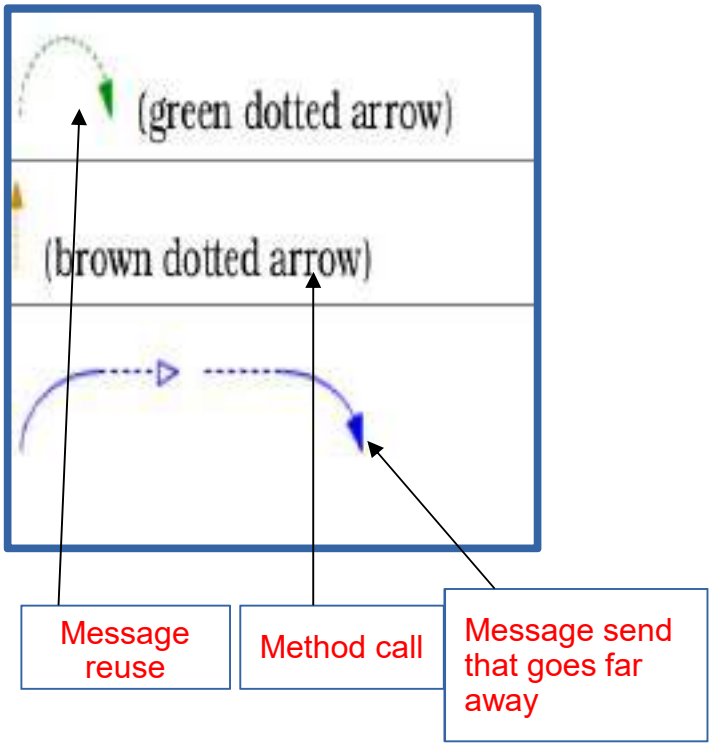
Self-message

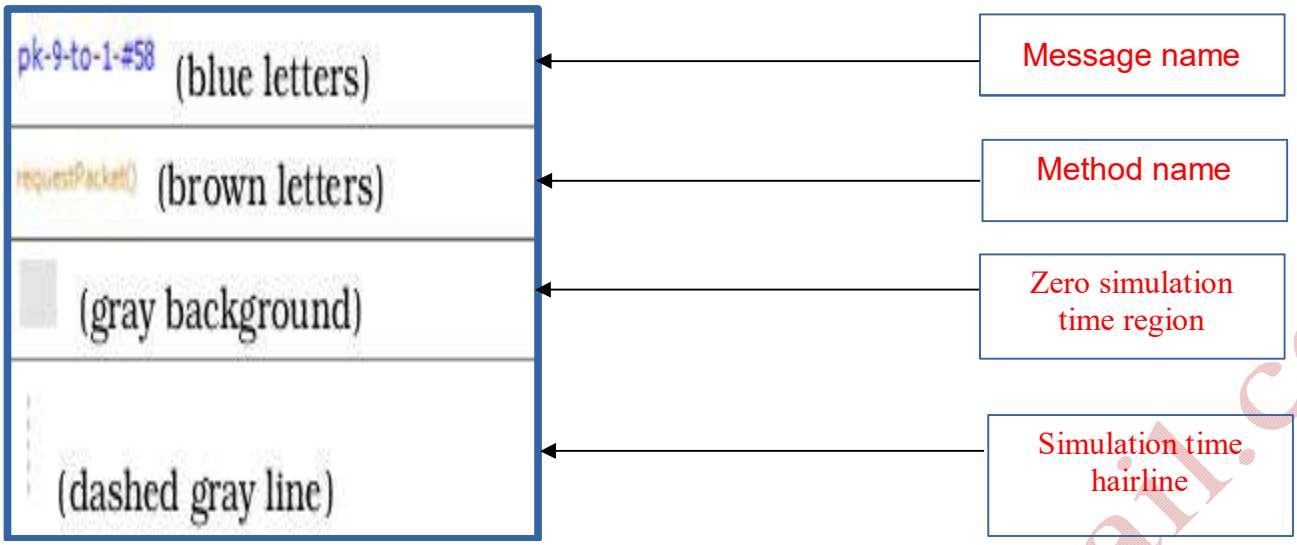
Message send

Sequence Charts

Understanding the Legend







END

TOPIC 28

Sequence Charts

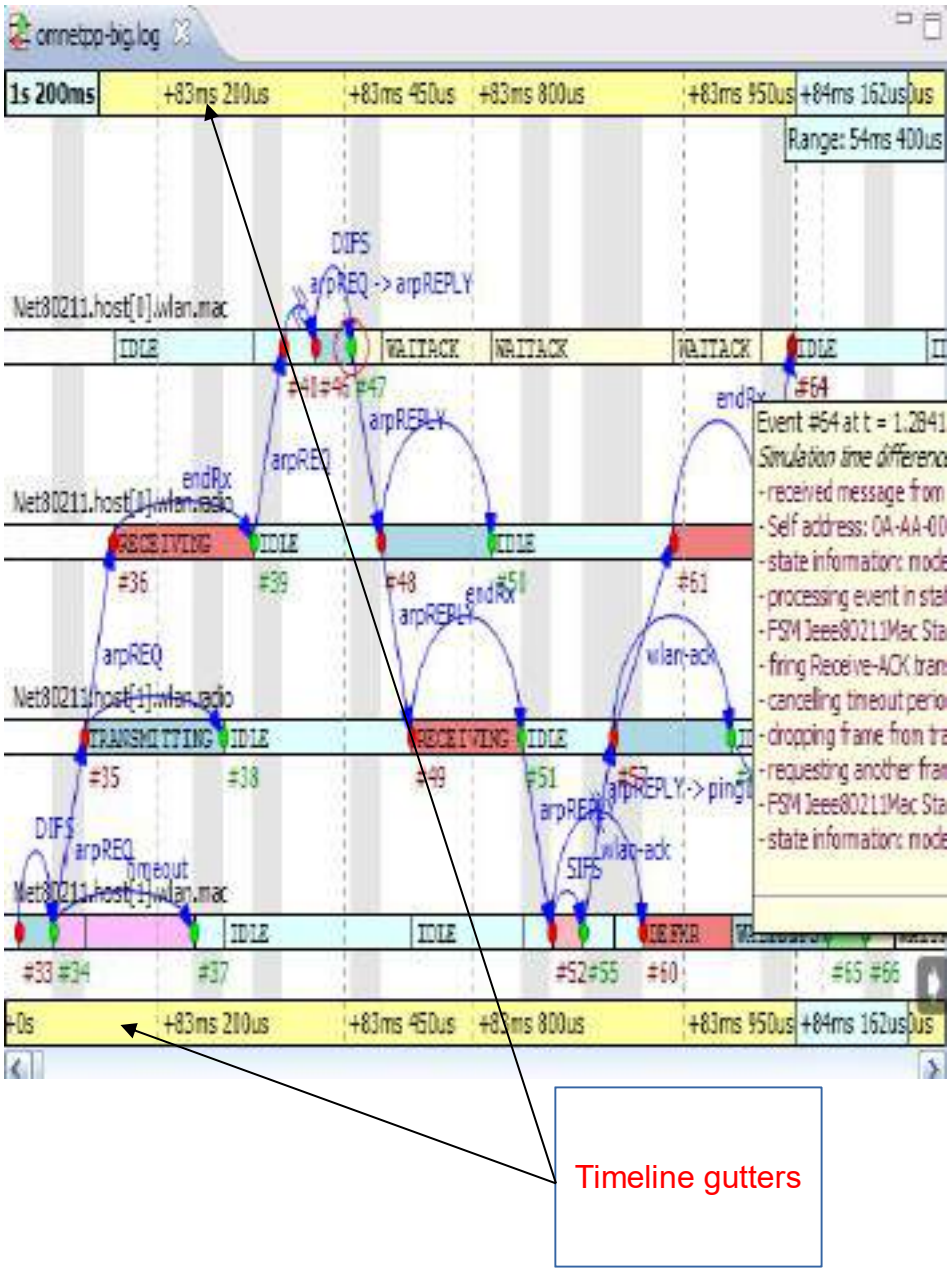
In this module

We shall understand

- Parts of sequence charts
- What is timeline?
- Types of timelines
- Interesting ways to interpret sequence charts

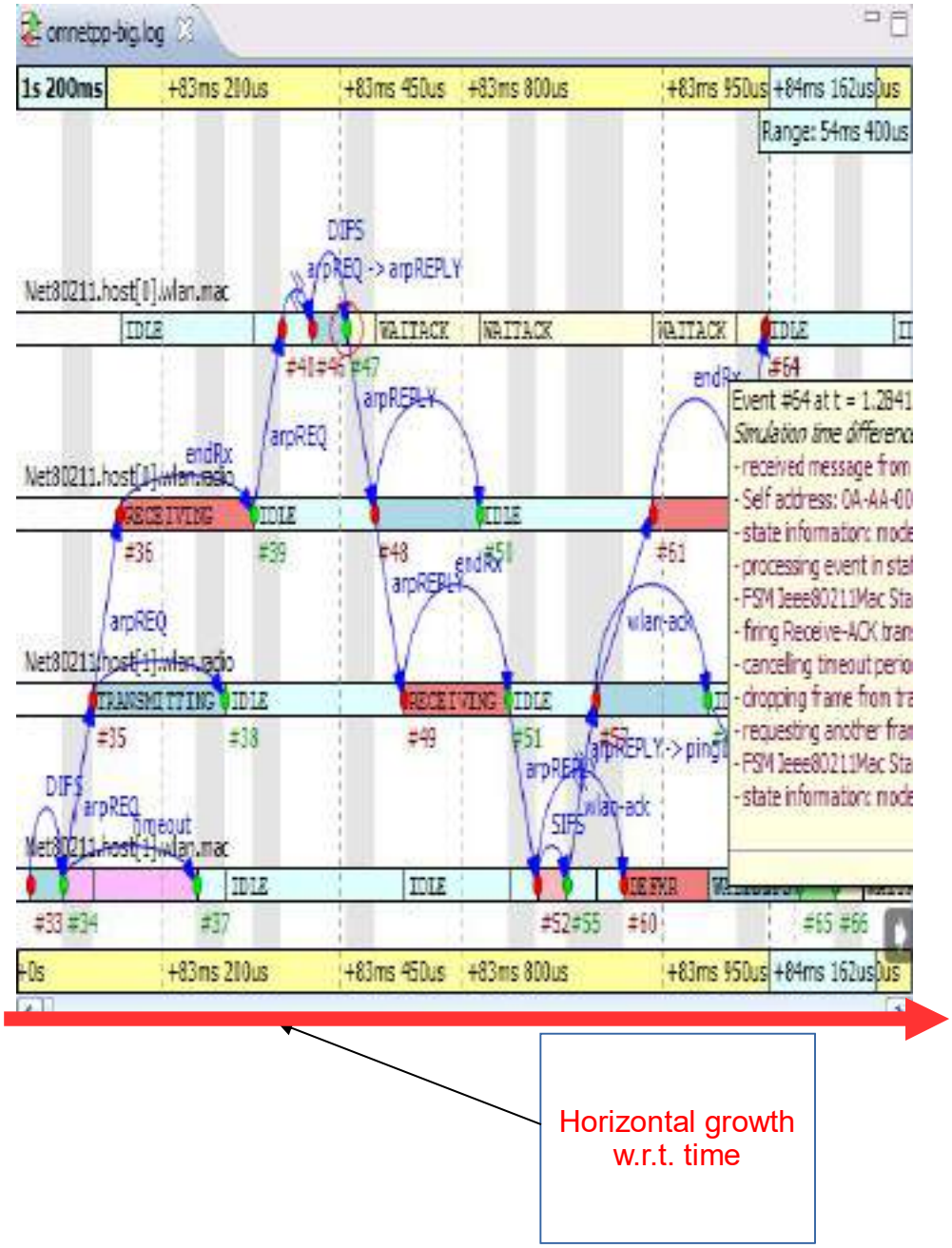
Sequence Charts

Parts of Sequence Charts



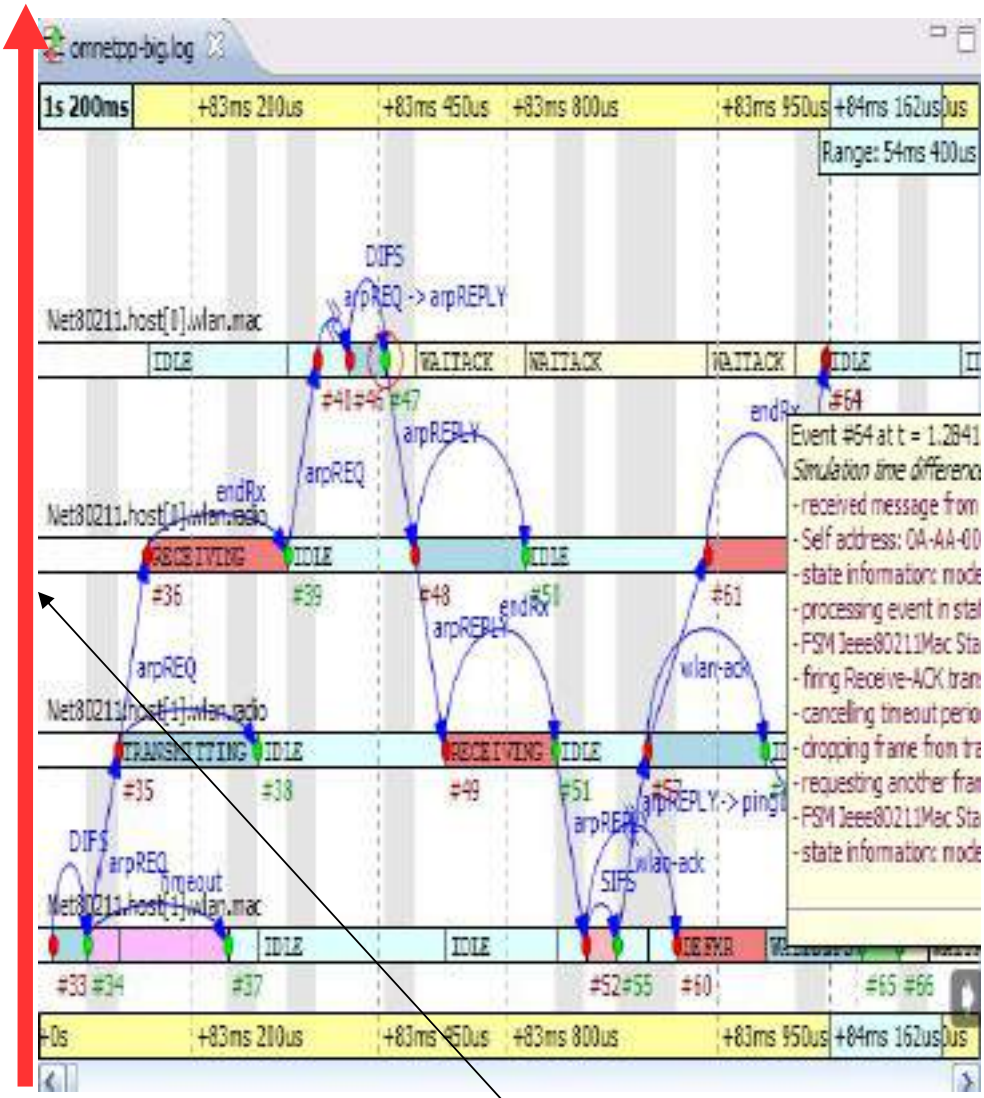
Sequence Charts

Parts of Sequence Charts



Sequence Charts

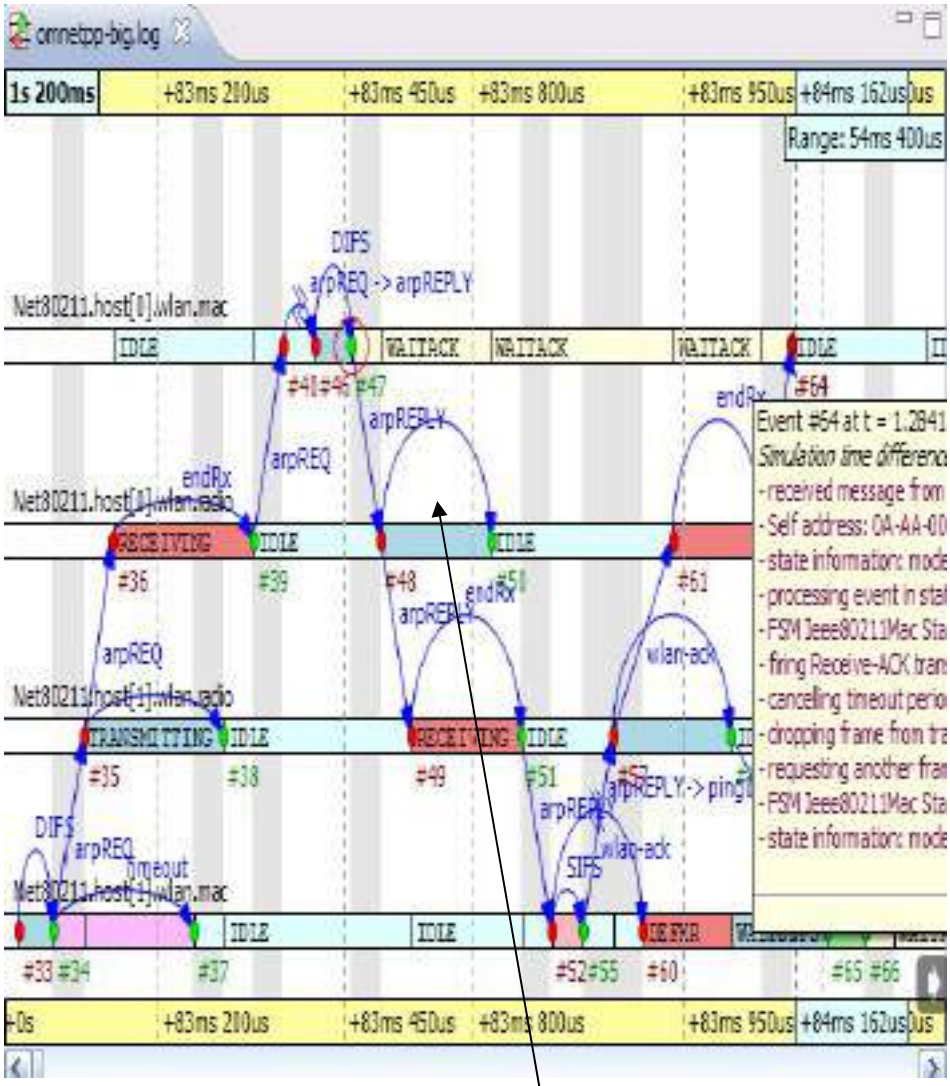
Parts of Sequence Charts



Vertical growth
w.r.t. no. of modules

Sequence Charts

Parts of Sequence Charts



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
(1 of 2)

- **Linear**: simulation time proportional to distance measured in pixels
- **Event number**: event number proportional to the distance measured in pixels

Types of Timeline
(2 of 2)

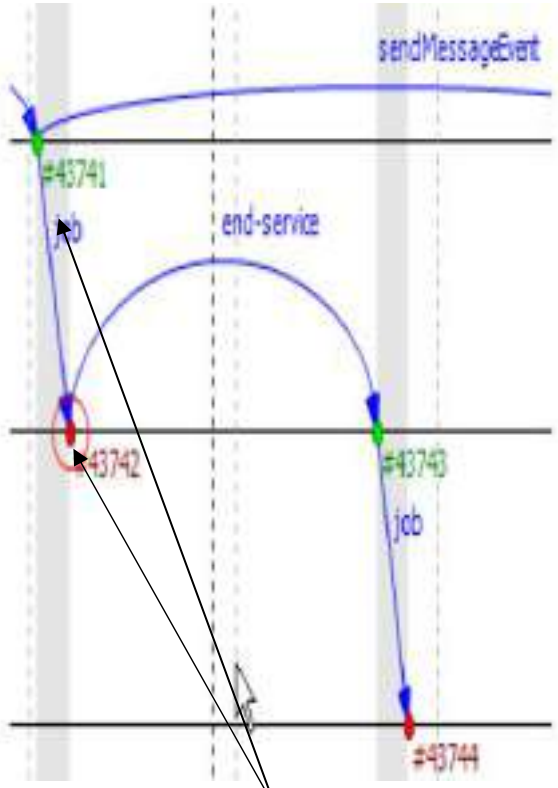
- **Step**: distance between subsequent events is same
- **Nonlinear**: distance between subsequent events is nonlinear function of simulation time between them

Interpreting Sequence Charts

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

Sequence Charts

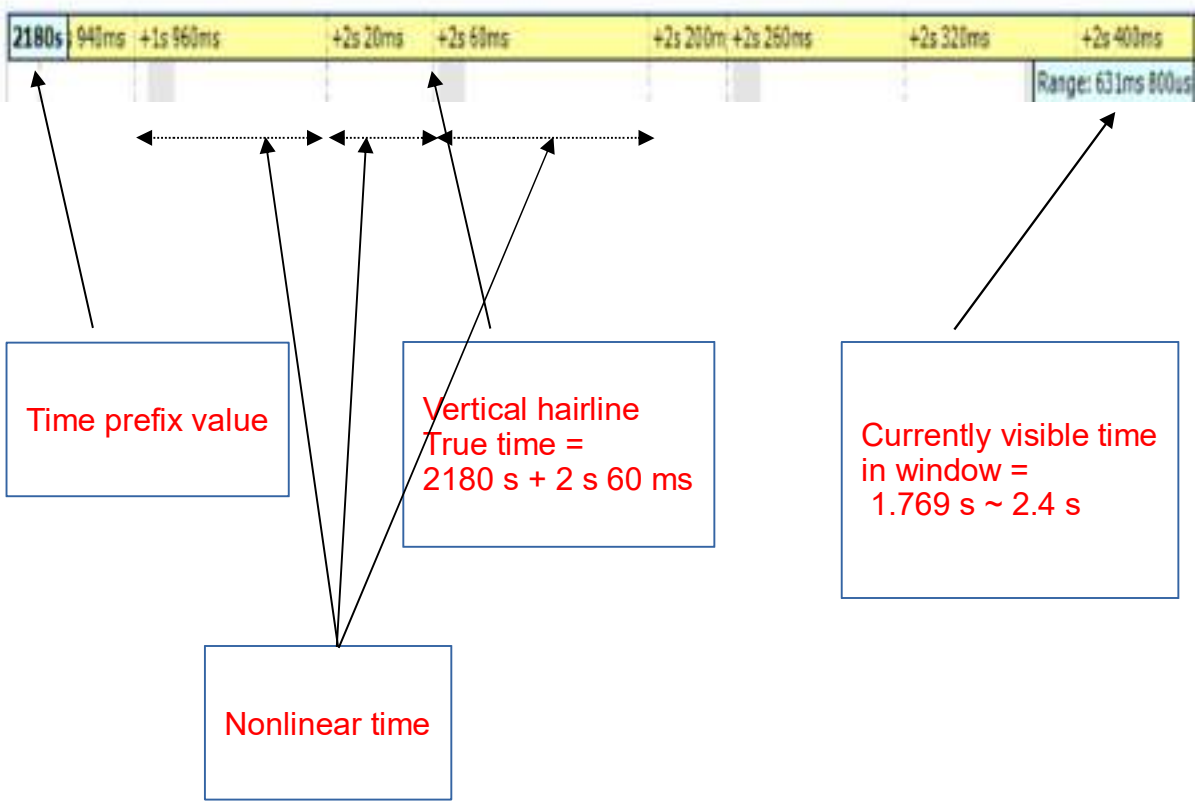
Zero Simulation Time Regions



Multiple events may occur at the same simulation time
Gray background indicates that the simulation time does not change all events inside it have the same simulation time

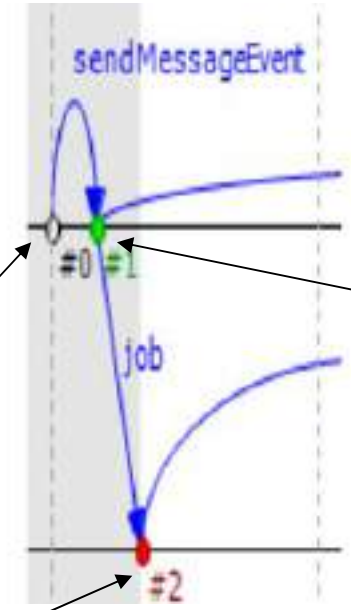
Sequence Charts

Gutter



Sequence Charts

Events Processing



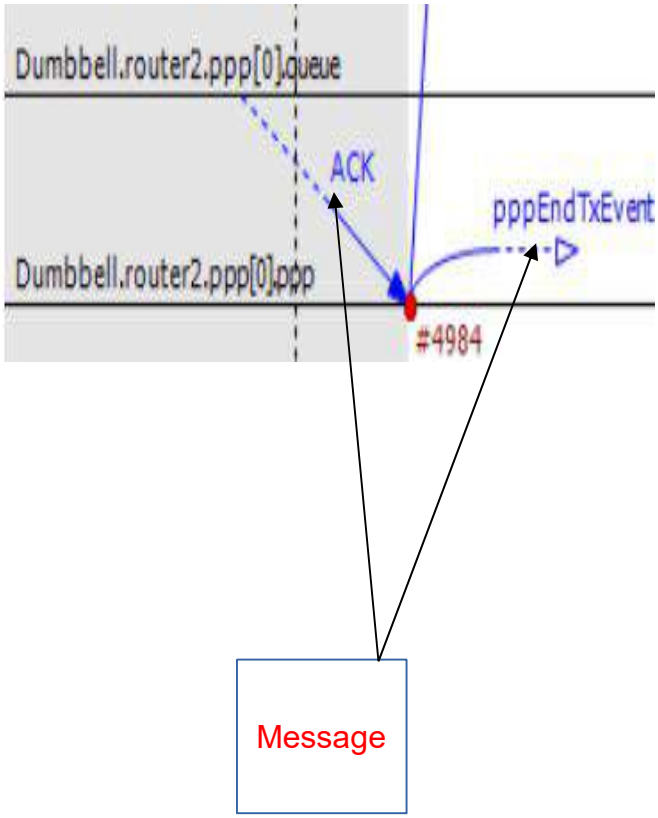
Initialization Message proc

Self message proc

Receiving Message proc

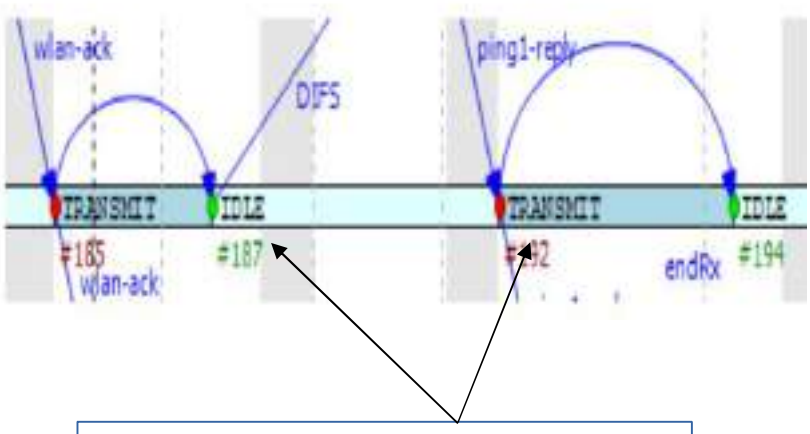
Sequence Charts

Messages



Sequence Charts

Displaying Module State on Axes



Color of axis changes as per the event
Output vector can be attached to an axis
IDLE for 0, TRANSMIT for 1

END

Week 03

TOPIC 29

Recap: TicToc Tutorial

In this module

We shall cover TicToc tutorial to recap

- Understanding NED file
- Understanding C++ file
- How to make the project?
- Preparing INI file
- Launch the simulation
- Sequence chart

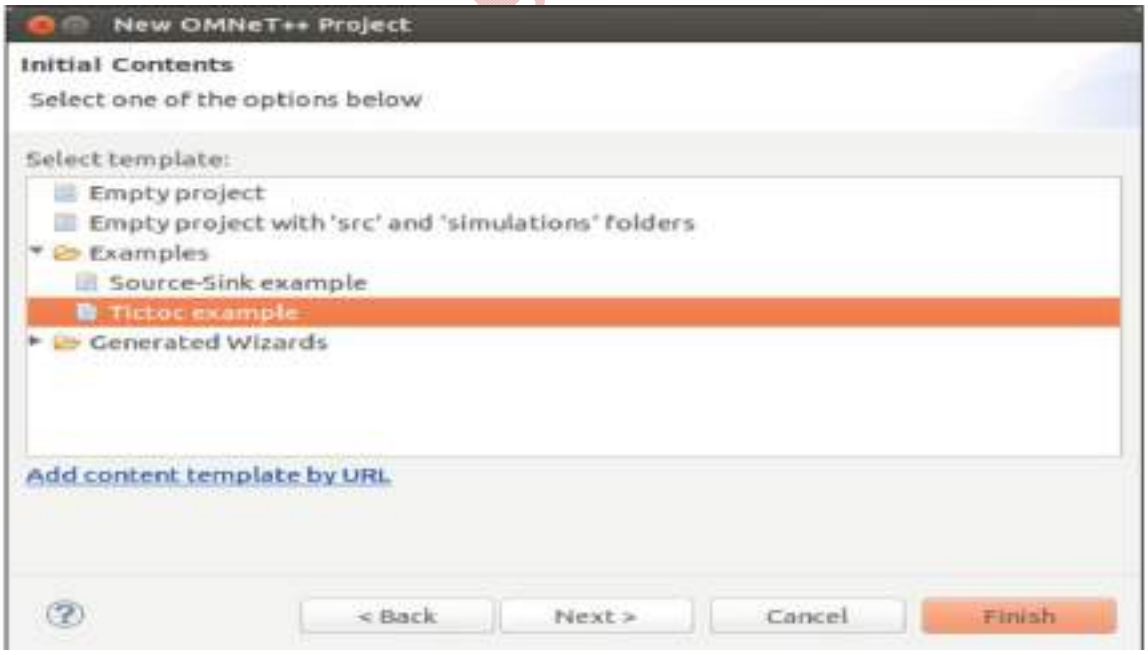
TicToc with 2-nodes

- 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 created

project, navigate to the simulations folder in the Project Explorer

- Open

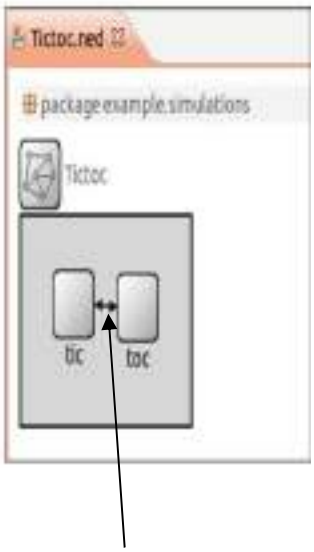
Tictoc.ned

Recap: TicToc Tutorial

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



Recap: TicToc Tutorial

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.ned

Opening Simple Module

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

Recap: TicToc Tutorial

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

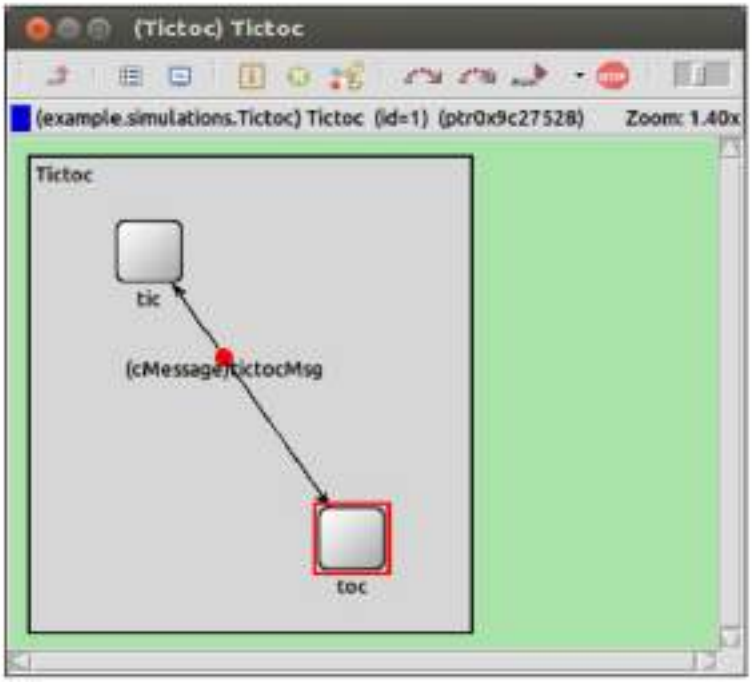
HandleMessage method

Echo back

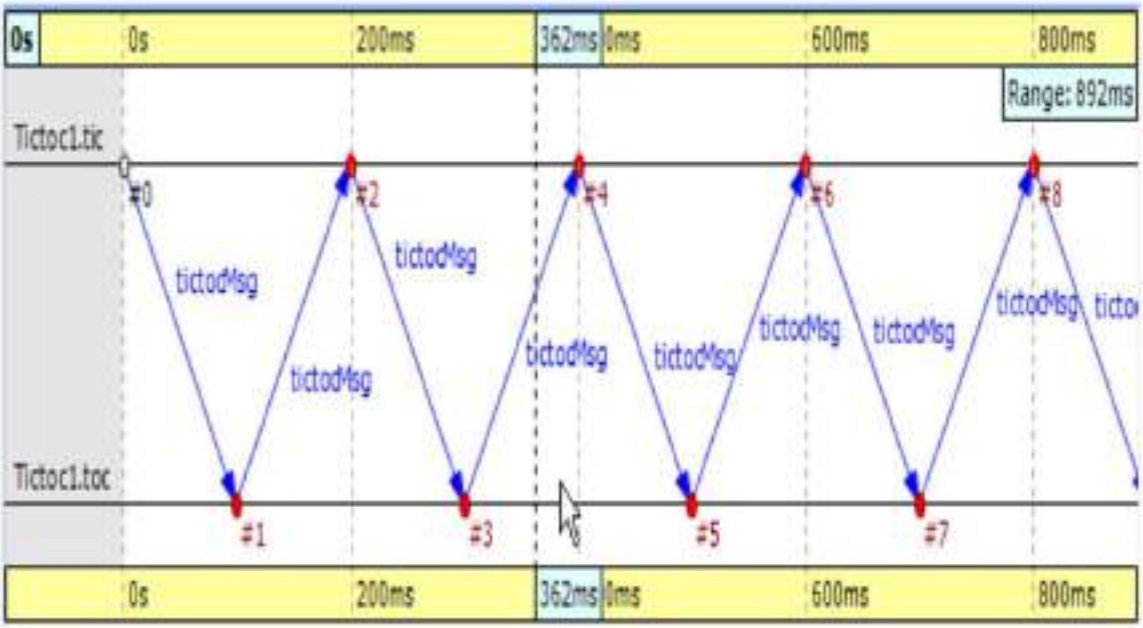
```
[General]
network = Tictoc
cpu-time-limit = 60s
#debug-on-errors = true
**.tic.sendInitialMessage = true
```

Starts the activity

Compiling & Running on Tkenv



Recap: TicToc Tutorial



TOPIC 30

Extending TicToc

In this module

We shall extend TicToc tutorial

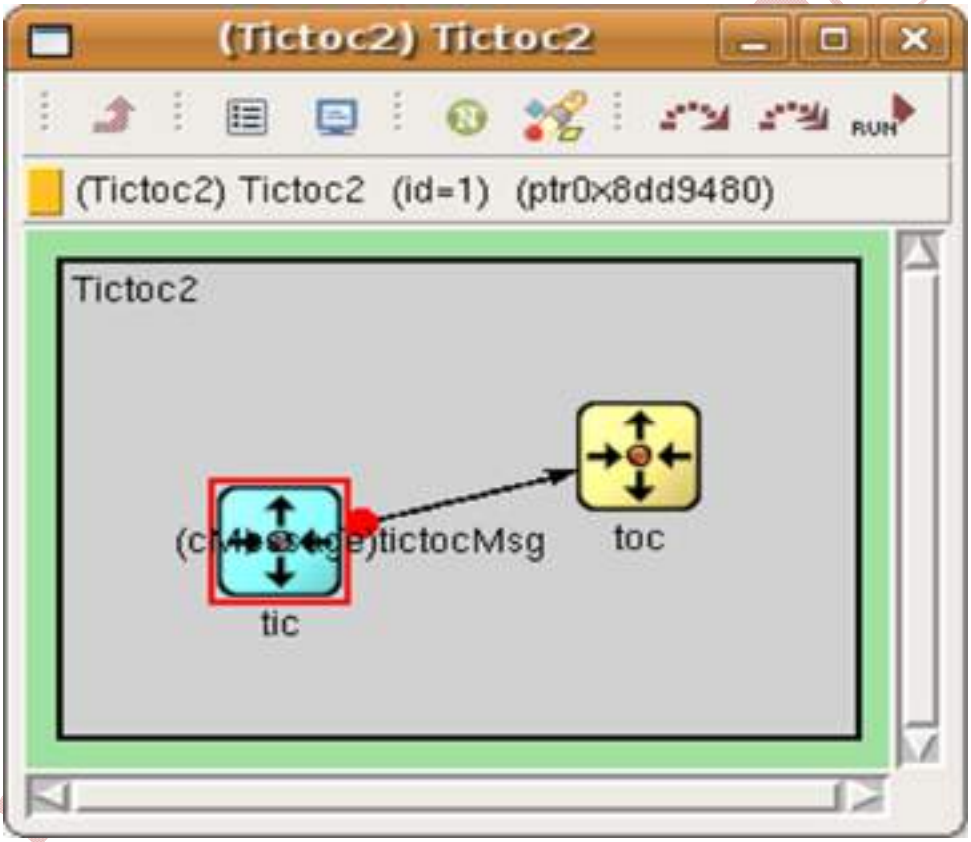
- Refine graphics, & add debugging output
- Add state variables
- Add parameters
- Model processing delay
- And more!

Refine graphics &

- Tictoc2.ned

Add debugging output

- Txc2.cc



Extending TicToc

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

Tictoc2.ned (1 of 2)

```
// "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

Tictoc2.ned (2 of 2)

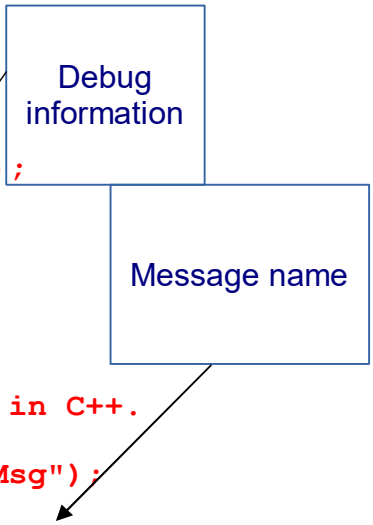
```
// Make the two module look a bit different with
colorization effect.
// Use cyan for `tic', and yellow for `toc'.
//
network Tictoc2
{
    submodules:
        tic: Txc2 {
            parameters:
                @display("i=,cyan"); // do not change the
icon (first arg of i=) just colorize it
  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

Change color

Extending TicToc

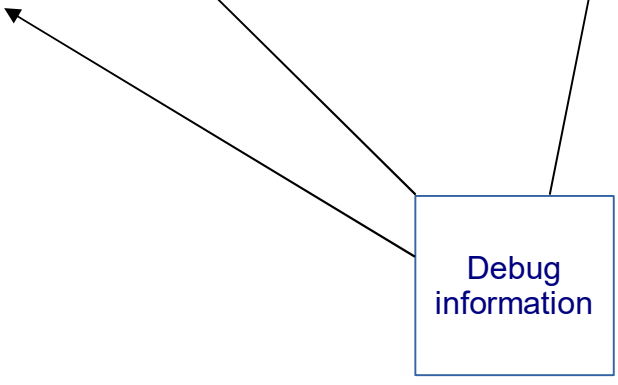
Txc2.cc (1 of 2)

```
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");
    }
}
```

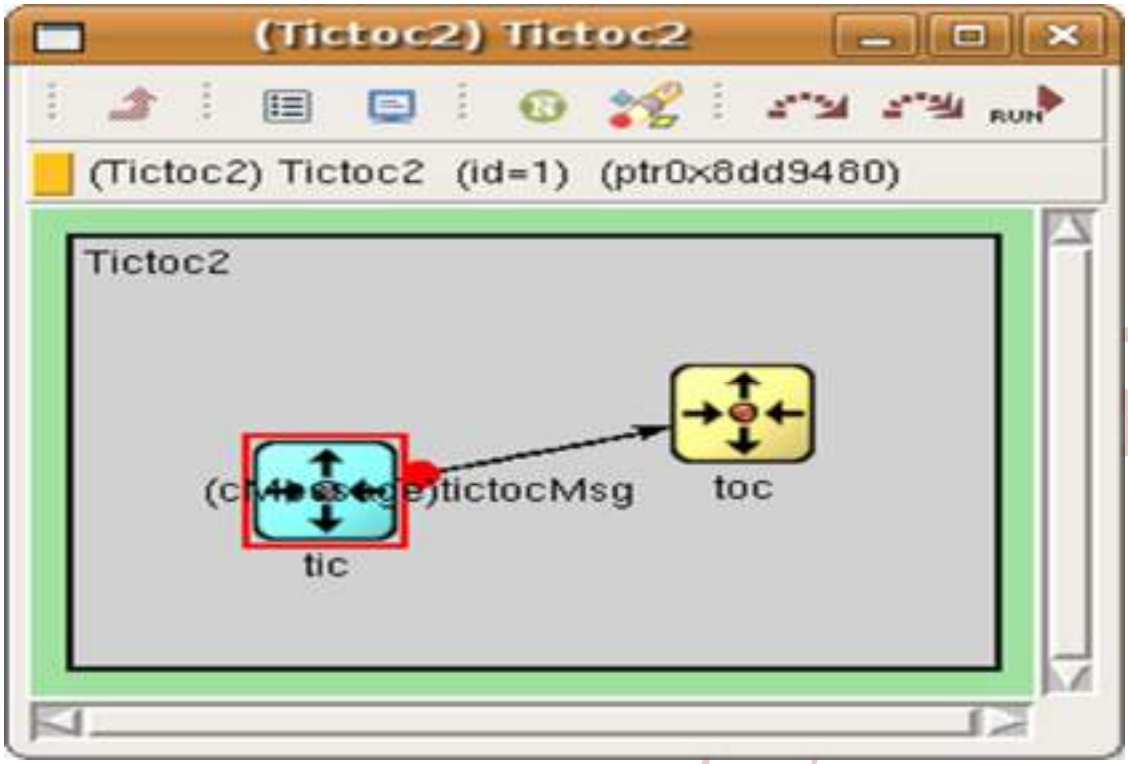


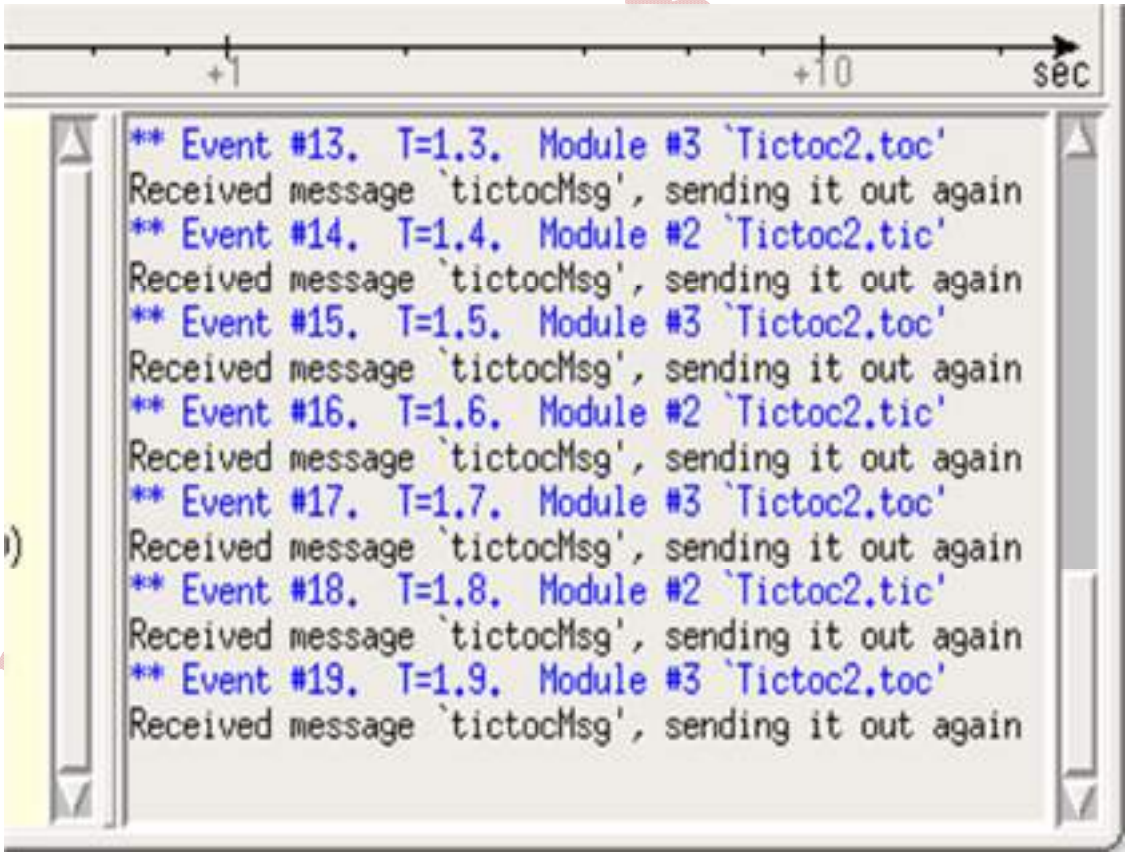
Txc2.cc (2 of 2)

```
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");
}
```



Tkenv output





TOPIC 31

Extending TicToc

In this module
We shall extend TicToc tutorial

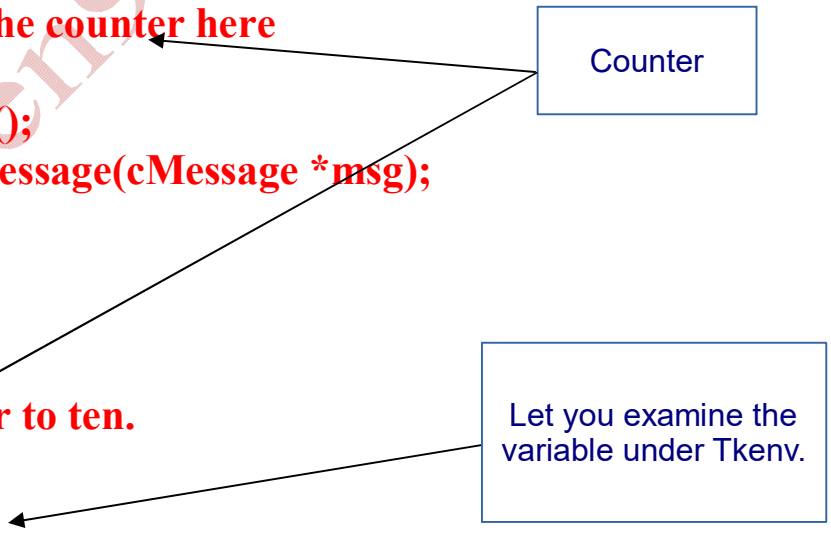
- Refine graphics, & add debugging output
- Add state variables
- Add parameters
- Model processing delay And more!

Add State Variables

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

Txc3.cc (1 of 3)

```
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);
```



Extending TicToc

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

Extending TicToc

Txc3.cc (2 of 3)

```
    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
```

Decrement counter

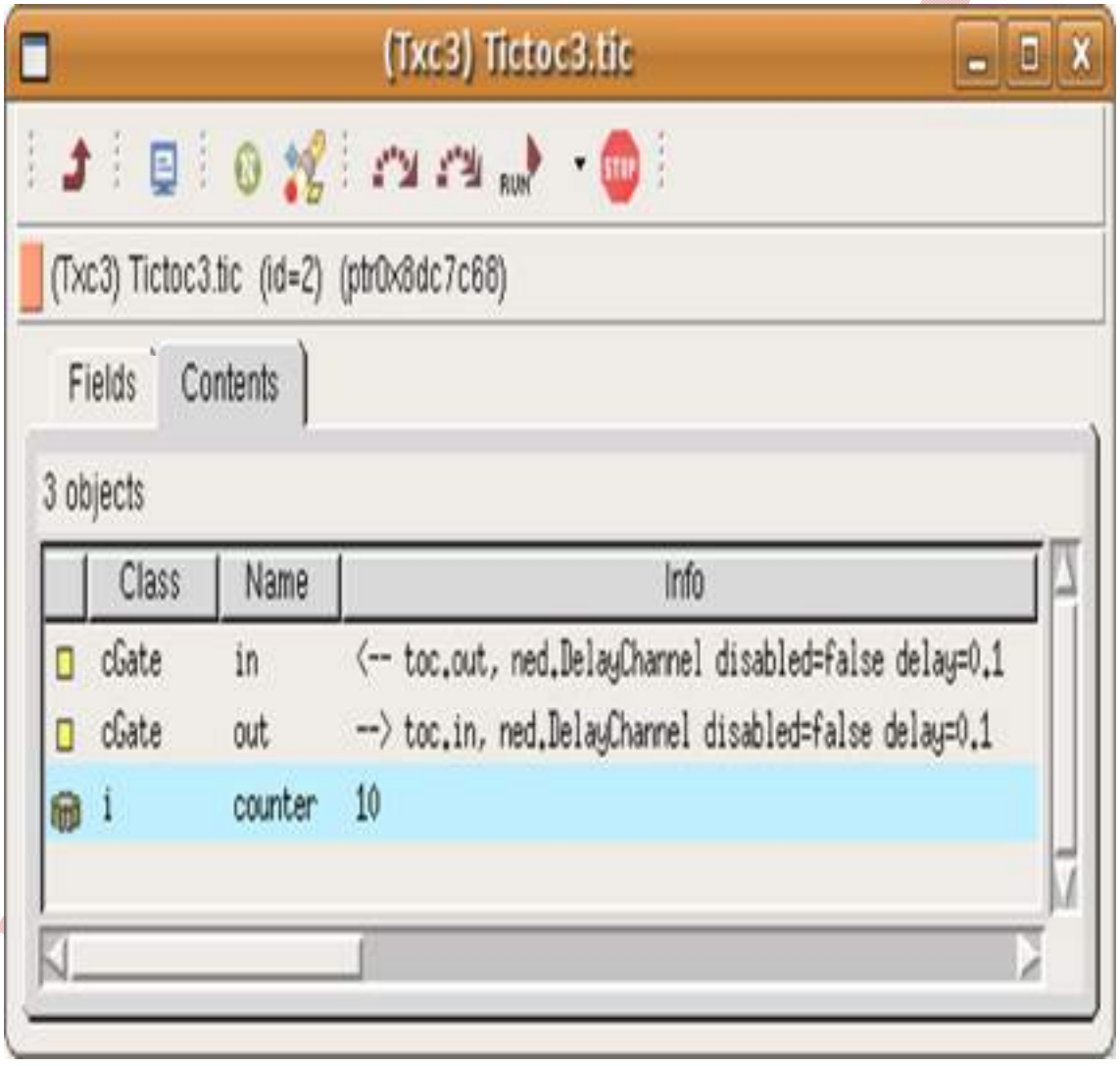
If counter is zero, delete message

Txc3.cc (3 of 3)

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

Or show current counter value

Extending TicToc



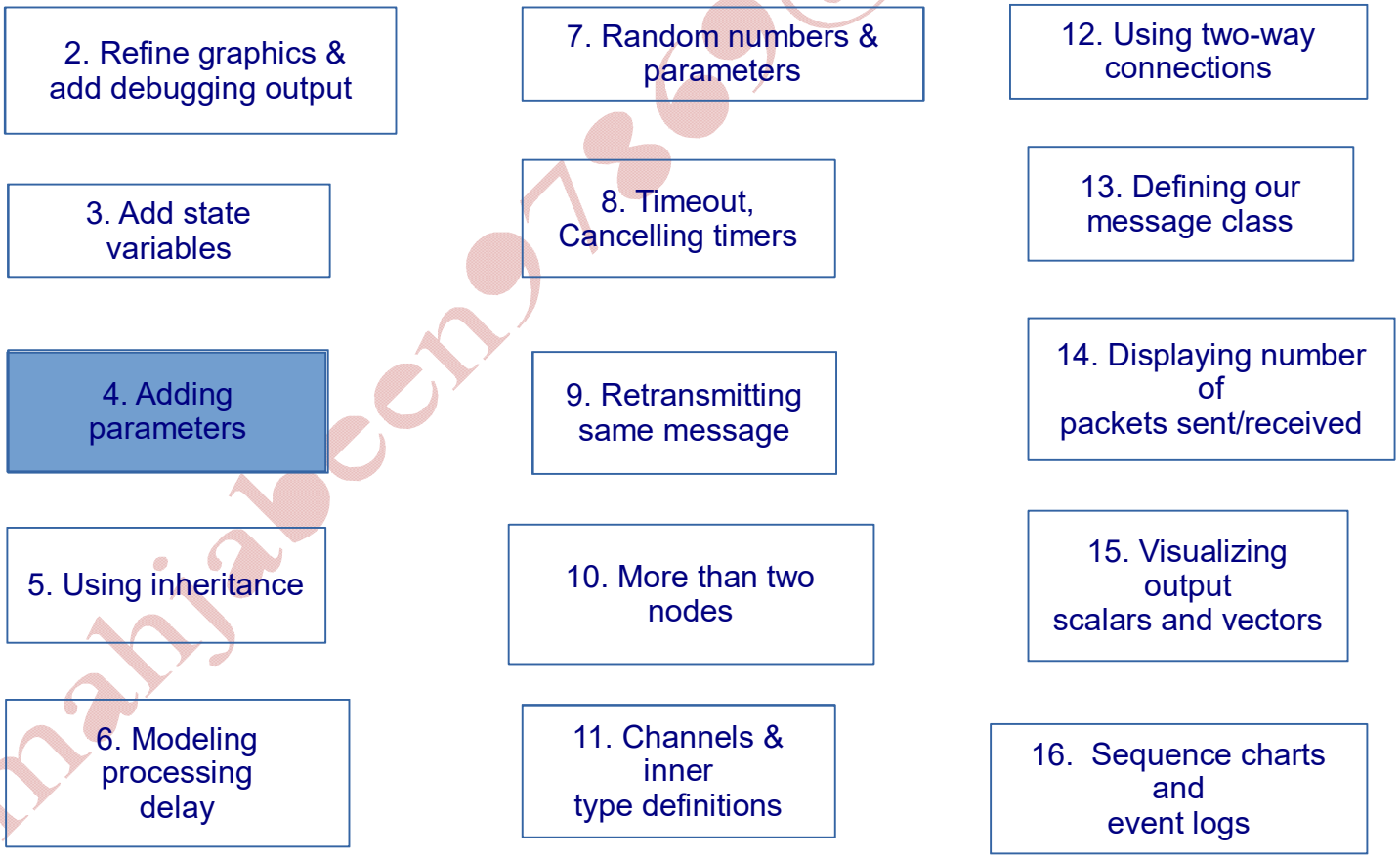
TOPIC 32

Extending TicToc

In this module

We shall extend TicToc tutorial

- Refine graphics, & add debugging output
- Add state variables
- Add parameters
- Model processing delay
- And more!



Adding parameters (1 0f 2)

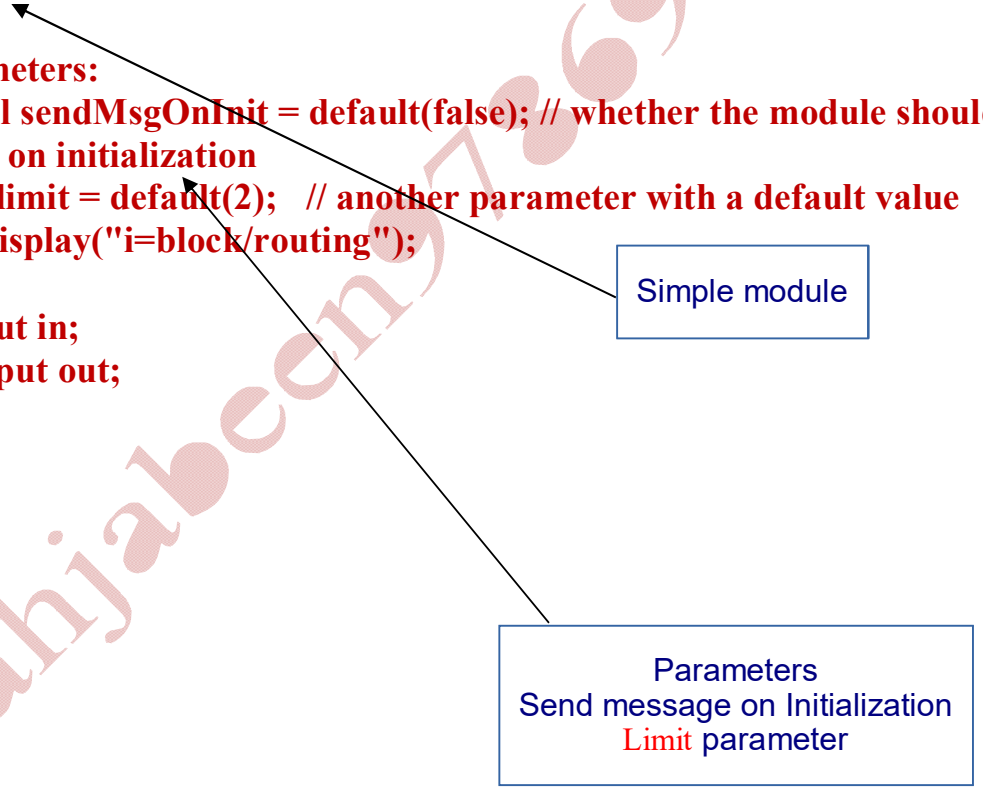
- **Add add input parameters to the simulations**
  - **Count = 10 now into a parameter that the user can define**
- **tictoc4.ned**
- **Txc4.cc**
- **Omnet.ini**

Adding parameters (1 0f 2)

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

tictoc4.ned (1 of 2)

```
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;
}
```



tictoc4.ned (2 of 2)

```
// 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

Txc4.cc

```
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

Omnet.ini
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)

TOPIC 33

Extending TicToc

In this module

We shall extend TicToc tutorial

- Refine graphics, & add debugging output
- Add state variables
- Add parameters
- Model processing delay
- And more!

2. Refine graphics & add debugging output

3. Add state variables

4. Adding parameters

5. Using inheritance

6. Modeling processing delay

7. Random numbers & parameters

8. Timeout, Cancelling timers

9. Retransmitting same message

10. More than two nodes

11. Channels & inner type definitions

12. Using two-way connections

13. Defining our message class

14. Displaying number of packets sent/received

15. Visualizing output scalars and vectors

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 it
- tictoc5.ned

tictoc5.ned (1 of 4)

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

Base module

Generalized parameters

tictoc5.ned (2 of 4)

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

Declare tic

Assign value

Extending TicToc

tictoc5.ned (3 of 4)

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

Declare toc

Assign value

Extending TicToc

tictoc5.ned (4 of 4)

```
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

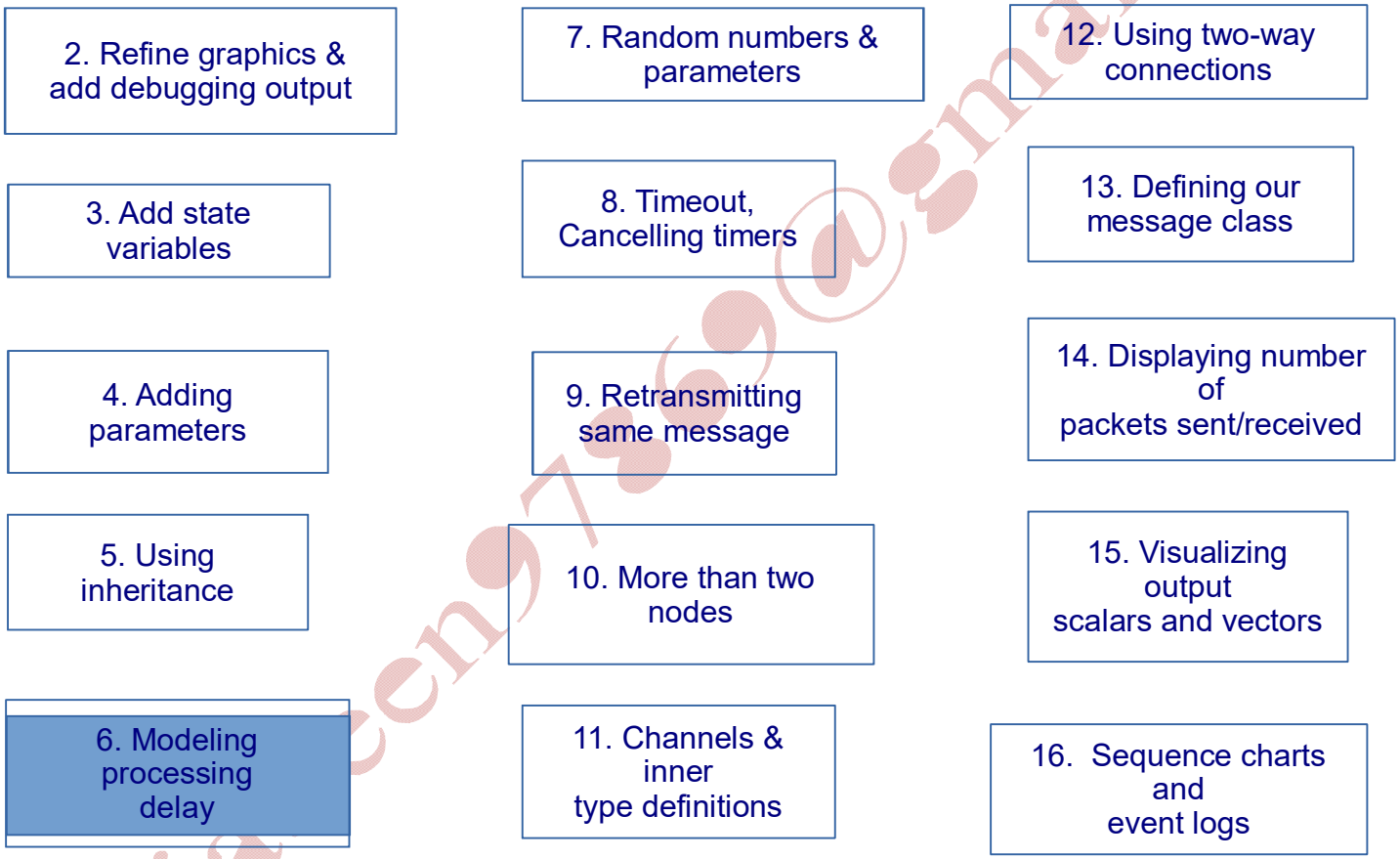
TOPIC 34

Extending TicToc

In this module

We shall extend TicToc tutorial

- Refine graphics, & add debugging output
- Add state variables
- Add parameters
- Model processing delay
- And more!



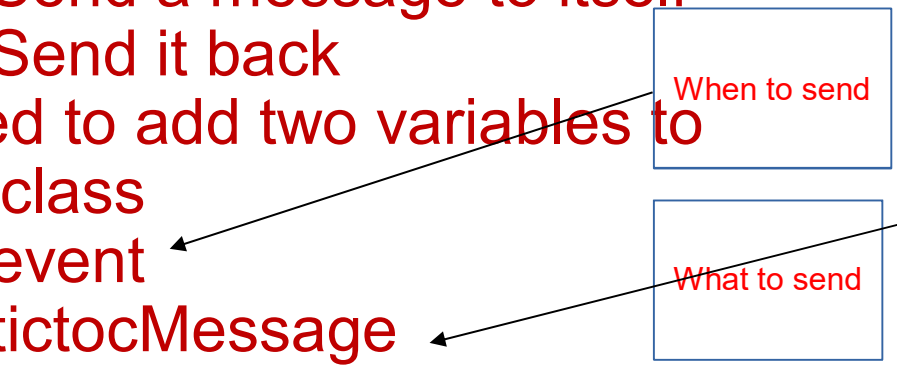
Modeling processing delay

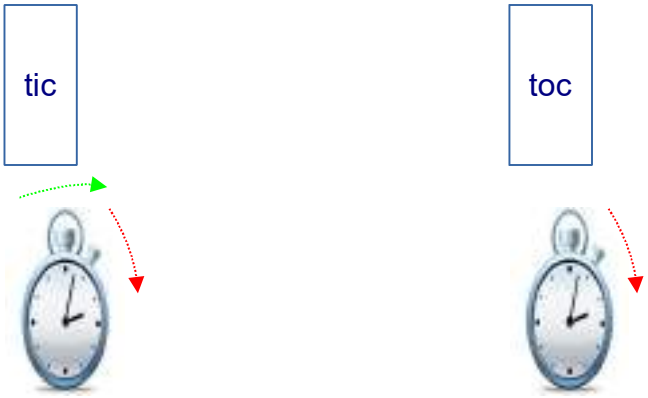
- **So far, no processing delay in tictoc**
- **We need timer in**
- **Tictoc module to send itself "Event" message**
- **tictoc6.ned**
- **txc6.cc**

Extending TicToc

Strategy

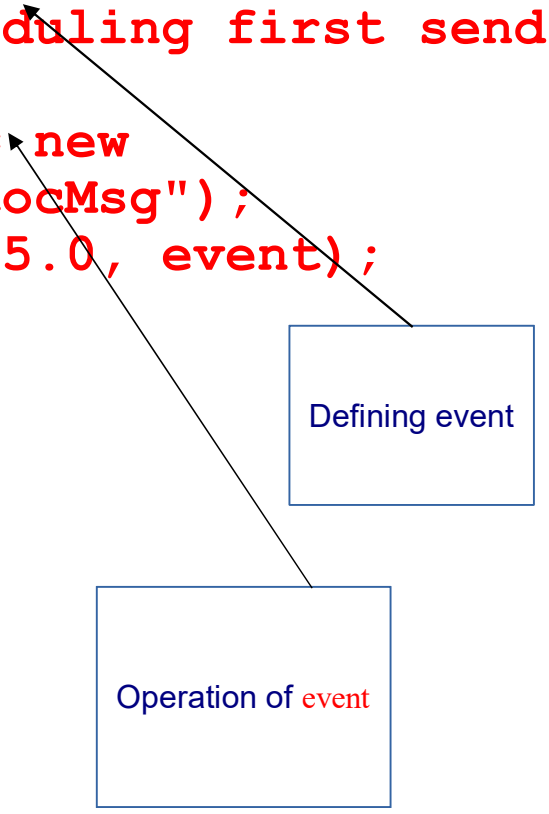
- Initialize after 5 seconds
- Hold the message for 1 simulated second
  - Send a message to itself
  - Send it back
- Need to add two variables to the class
  - event
  - tictocMessage





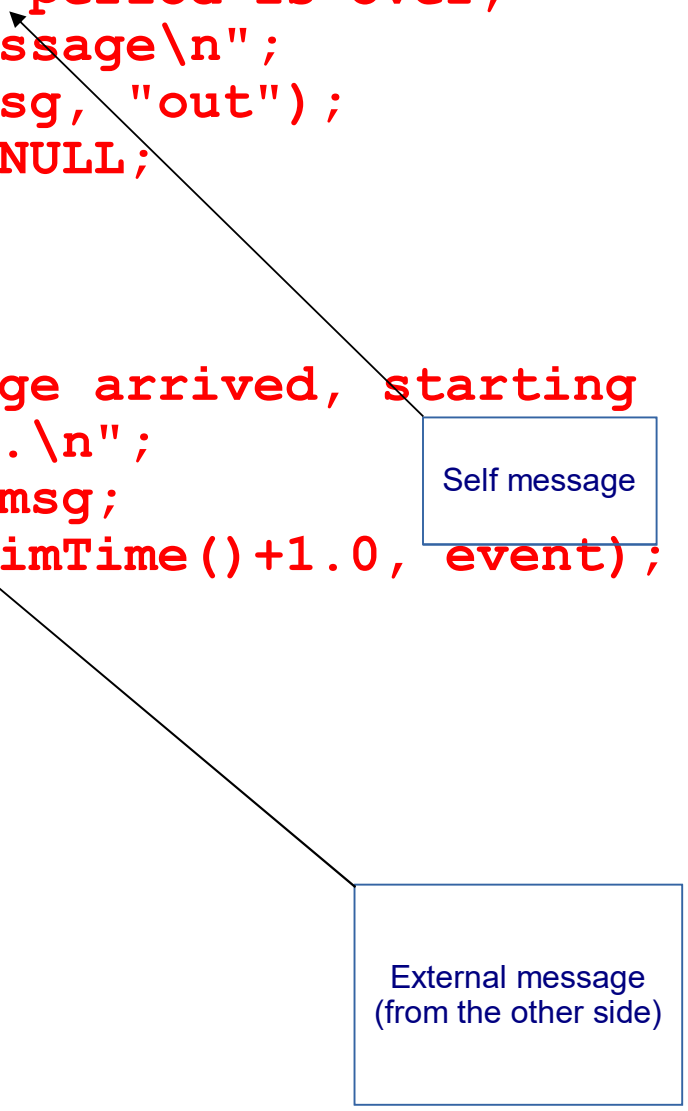
tx6.cc (1 of 2)

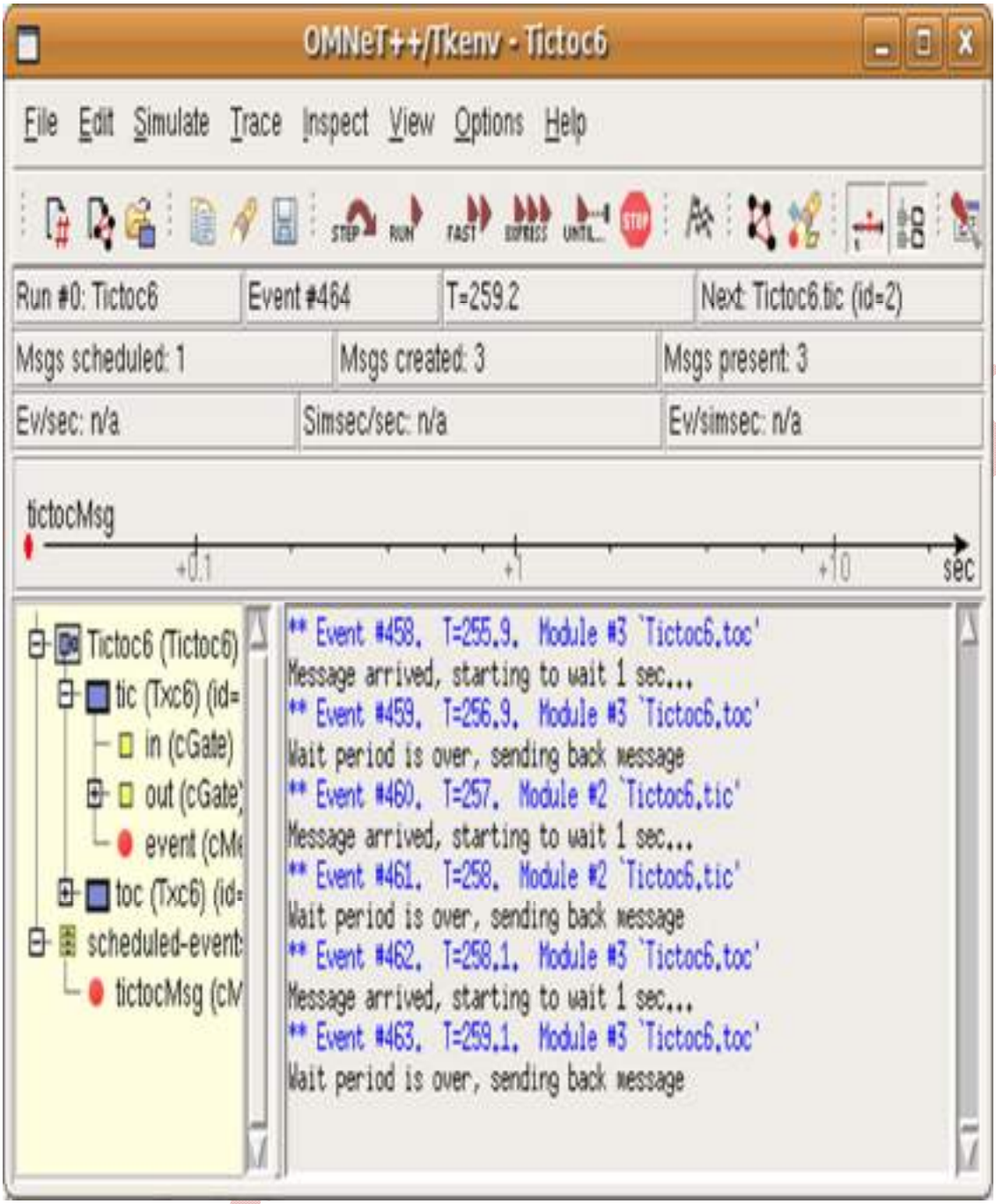
```
void Txc6::initialize()
{
   // Create the event object
(ordinary message) for  //timing
   event = new cMessage("event");
   tictocMsg = NULL;
   if (strcmp("tic", getName()) == 0)
   EV << "Scheduling first send to
t=5.0s\n";
   tictocMsg = new
cMessage("tictocMsg");
   scheduleAt(5.0, event);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



tx6.cc (2 of 2)

```
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);
    }
}
```





TOPIC 35

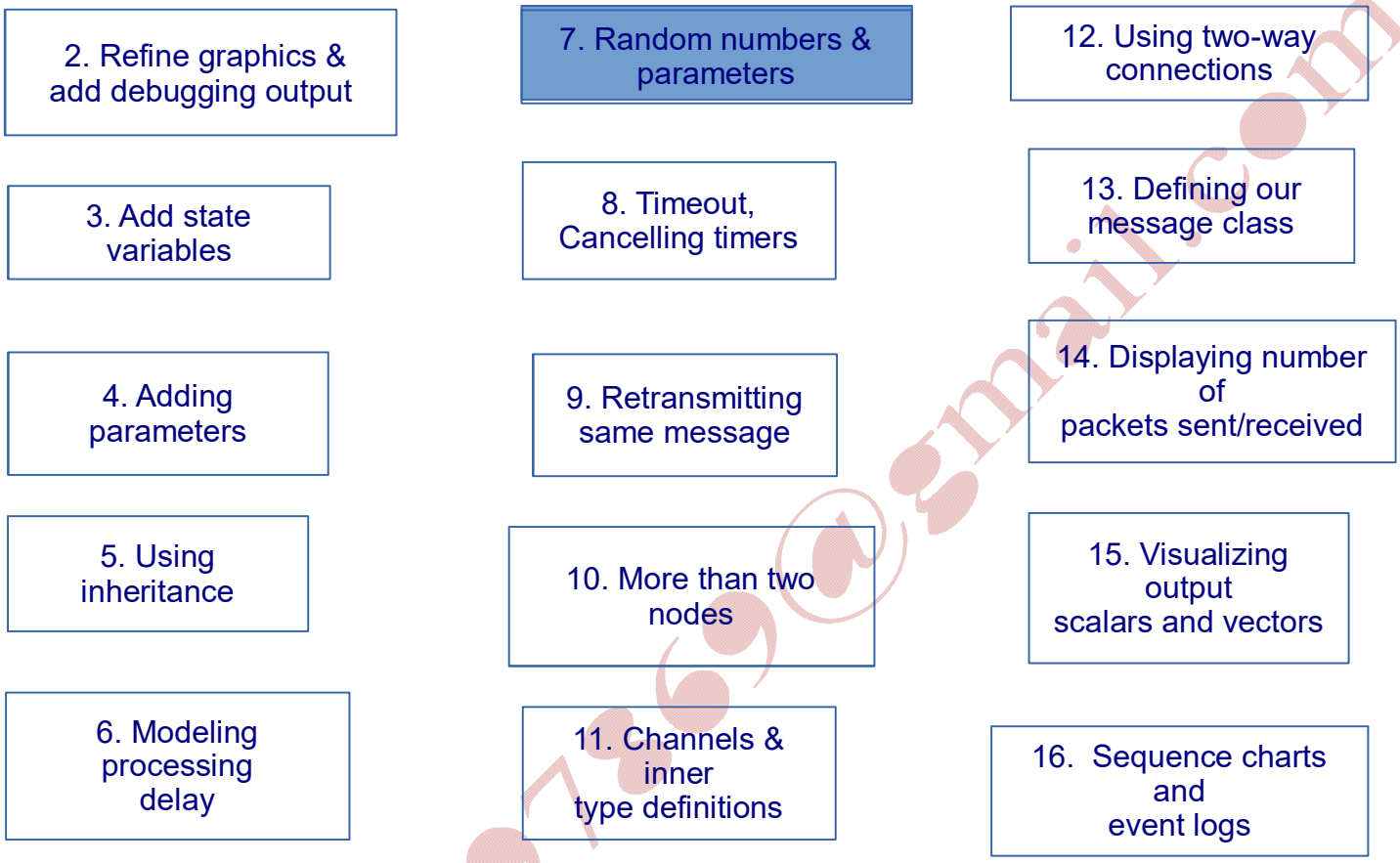
Extending TicToc

In this module

We shall extend TicToc tutorial

- Refine graphics, & add debugging output
- Add state variables

- Add parameters
- Model processing delay
- And more!



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 (1 of 2)

```
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

txc7.cc (2 of 2)

```
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

tictoc7.ned

```
network = Tictoc7
# argument to exponential() is the mean
Tictoc7.tic.delayTime = exponential(3s)
```

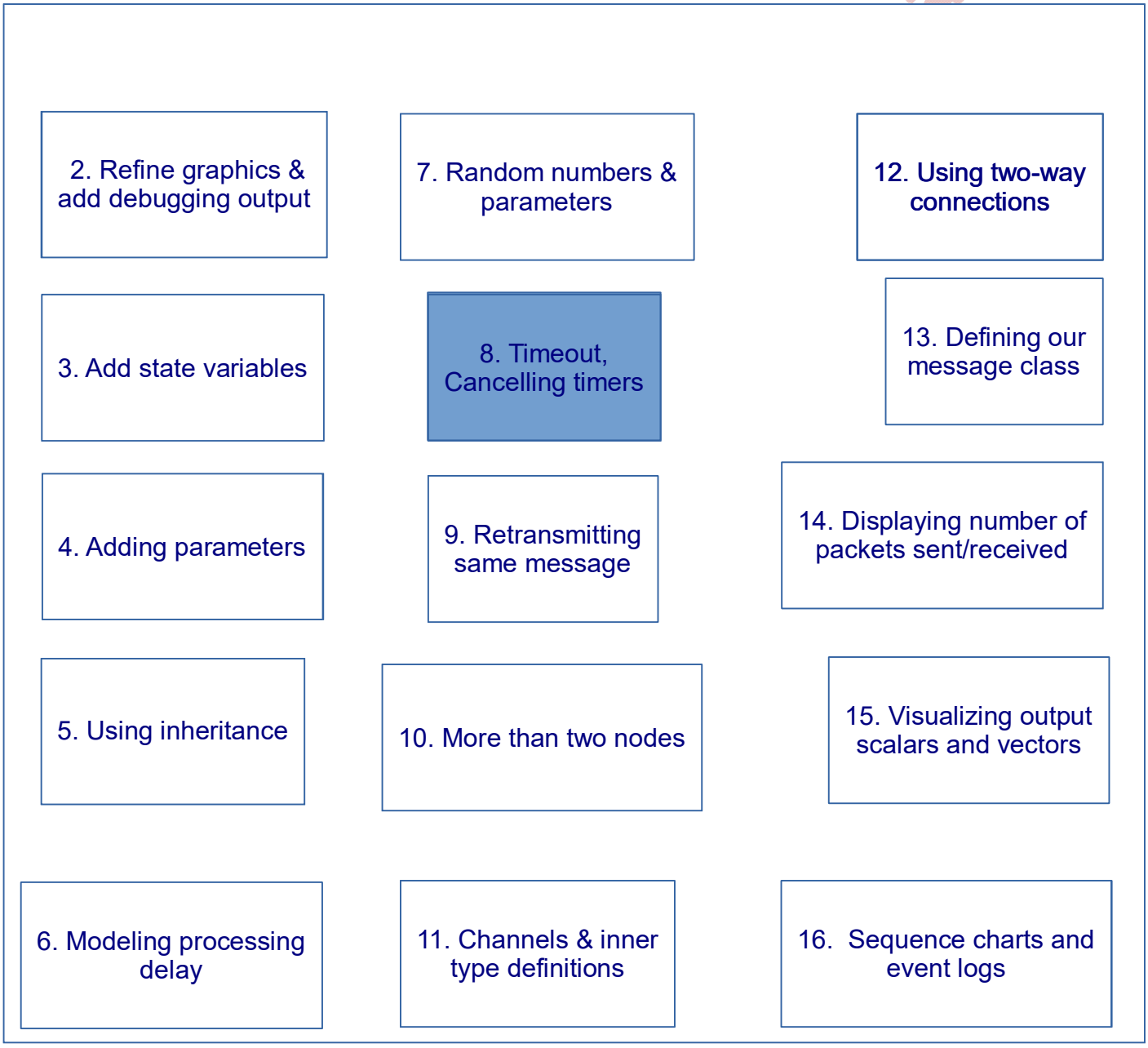
mean

TOPIC 36

In this module

We shall extend TicToc tutorial

- Refine graphics, & add debugging output
- Add state variables
- Add parameters
- Model processing delay
- And more!



Timeout, cancelling timers

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

Extending TicToc

Strategy

Extending TicToc

txc8.cc (1 of 3)

```
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

Extending TicToc

txc8.cc (2 of 3)

```
void Tic8::handleMessage(cMessage *msg)
{
    if (msg==timeoutEvent)
 {
  EV << "Timeout expired, resending and restarting
    timer\n";
  cMessage *newMsg = new cMessage("tictocMsg");
  send(newMsg, "out");
  scheduleAt(simTime()+timeout, timeoutEvent);
 }
```

timeout means we have to re-send it

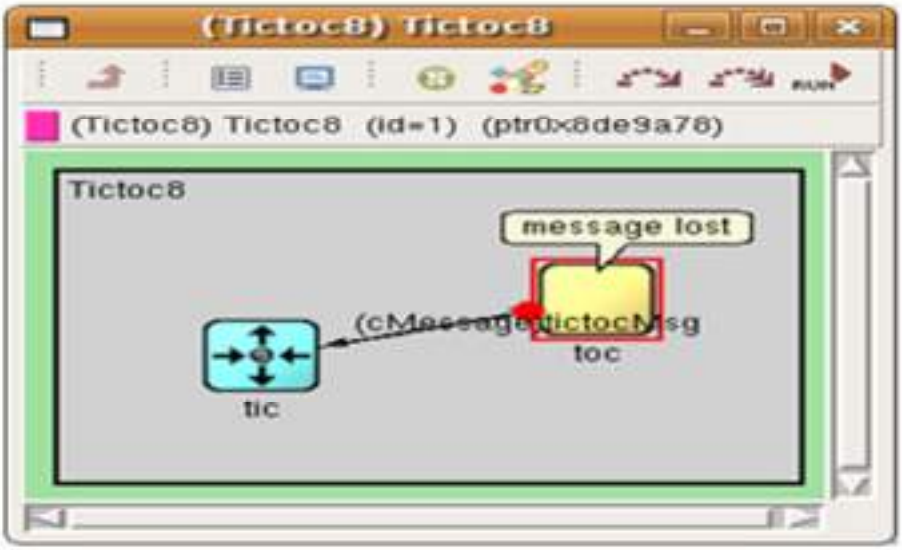
Extending TicTac

txc8.cc (3 of 3)

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

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

 //Ready to send another one.

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

   }
}
```

message arrived
Ack received

Extending TicToc



TOPIC 37

In this module

We shall extend TicToc tutorial

- Refine graphics, & add debugging output
- Add state variables
- Add parameters
- Model processing delay
- And more!

Retransmitting same message (1 of 2)

- **So far we used "tictocMsg"**
- **It was created afresh everytime**
  - **At tic**
  - **At toc**

Extending TicToc

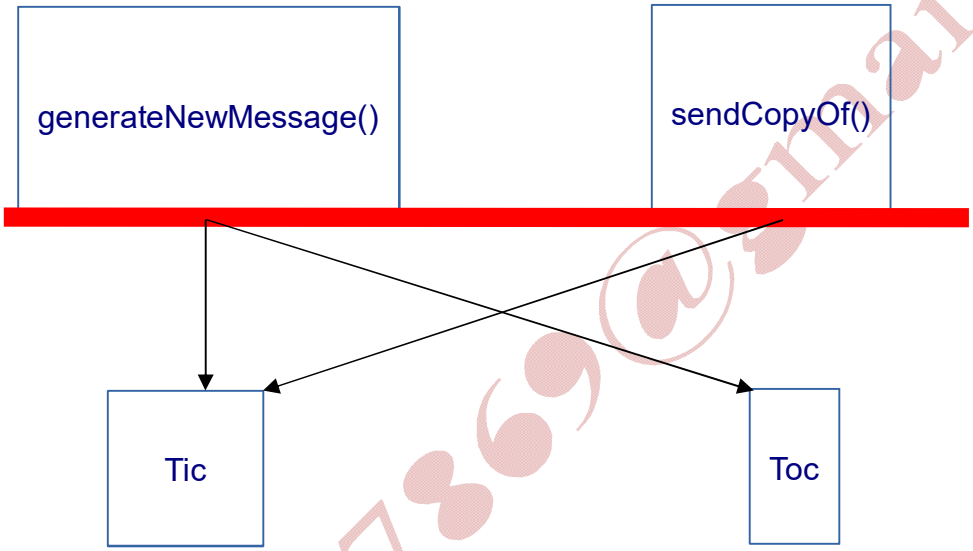
2. Refine graphics & add debugging output	7. Random numbers & parameters	12. Using two-way connections
3. Add state variables	8. Timeout, Cancelling timers	13. Defining our message class
4. Adding parameters	9. Retransmitting same message	14. Displaying number of packets sent/received
5. Using inheritance	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

Retransmitting same message (2 of 2)

- **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



Extending TicToc

Txc9.cc (1 of 2)

```
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

Extending TicToc

Txc9.cc (2 of 2)

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

    message = generateNewMessage();

    sendCopyOf(message);

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

Transmit a new packet

Extending TicToc

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;
}
```

Prints the string on Location of length 20 pointed by msgname

Increments seq no and is value of %d

Displays string which is seq no as decimal

Extending TicToc

```
sendCopyOf(cMessage *msg)
{
  // Duplicate message and send the copy.
    cMessage *copy = (cMessage *) msg-
>dup();
    send(copy, "out");
}
```

Value of copy taken from msg

Creates & returns an exact copy of msg

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

TOPIC 38

In this module

We shall extend TicToc tutorial

- Refine graphics, & add debugging output
- Add state variables
- Add parameters
- Model processing delay
- And more!

Extending TicToc

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 (1 of 2)

- Create several tic modules
- Connect them into a network
- One of the nodes generates a message

More than 2 nodes (2 of 2)

- Others toss it around in random directions
- Until it arrives at a predetermined destination
- tictoc10.ned
- omnetpp.ini
- txc10.cc

Extending TicToc

Tictoc10.ned (1 of 2)

```
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

Extending TicToc

Tictoc10.ned (2 of 2)

```
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

Extending TicToc

Txc10.cc (1 of 3)

```
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

Extending TicToc

Txc10.cc (2 of 3)

```
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!)

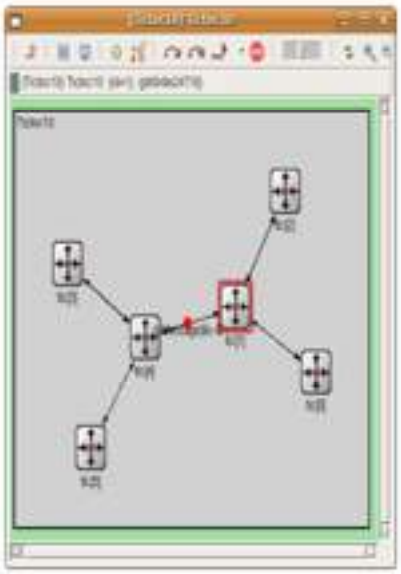
Extending TicToc

Txc10.cc (3 of 3)

```
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

Extending TicToc



TOPIC 39

In this module

We shall extend TicToc tutorial

- Refine graphics, & add debugging output
- Add state variables
- Add parameters
- Model processing delay
- And more!

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

Channels & inner type definitions (1 of 2)

- **With growing topology**
  - **We can improve connection section**
- **tictoc11.ned**
- **omnetpp.ini**
- **txc11.cc**

Channels & inner type definitions (2 of 2)

- **Connections with same delay parameter can be *typified* as channel**
- **Such channel can then be replicated between gates**

Extending TicToc

Tictoc11.ned (1 of 2)

```
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)

Extending TicToc

Tictoc11.ned (2 of 2)

```
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

TOPIC 40

In this module

We shall extend TicToc tutorial

- Refine graphics, & add debugging output
- Add state variables
- Add parameters
- Model processing delay
- And more!

Using two-way connections (1 of 2)

- So far, each node pair is connected with two connections
- Two-way connection can reduce coding size
- tictoc12.ned
- txc12.cc
- omnetpp.ini

Using two-way connections (2 of 2)

- We define two-way (inout) gates
  - Instead of in and out gates

Extending TicToc

Tictoc12.ned (1 of 2)

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

inout gate defined for both incoming and outgoing messages

Extending TicToc

Tictoc12.ned (2 of 2)

```
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++;
```

END

CS432 Handouts Made by Mahjabeen
mahjabeen97869@gmail.com
contact # 0321 2711298

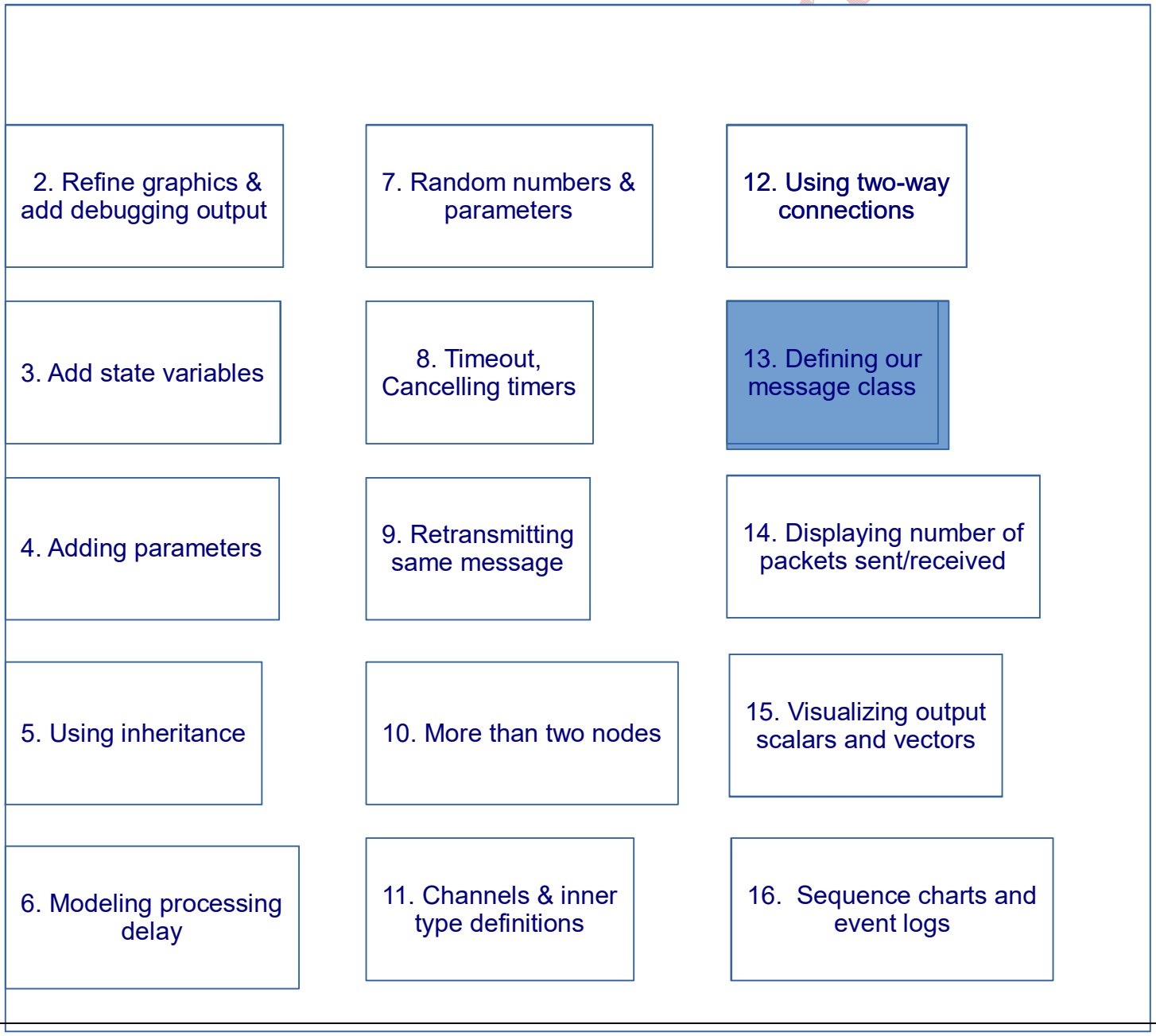
TOPIC 41

In this module

We shall extend TicToc tutorial

- Refine graphics, & add debugging output
- Add state variables
- Add parameters
- Model processing delay
- And more!

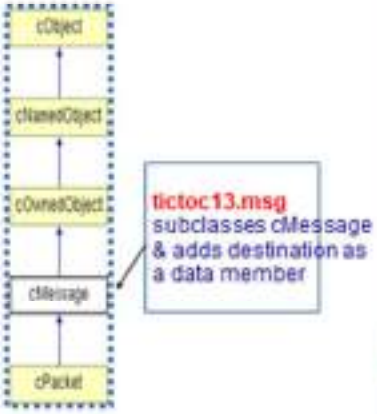
Extending TicToc



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

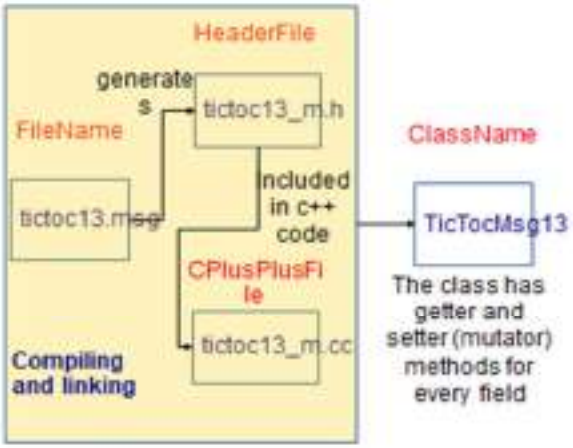
Extending TicToc

Strategy (1 of 2): Avoid boilerplate code writing



Extending TicToc

Strategy (2 of 2): Avoid boilerplate code writing



Extending TicToc

Txc13.cc (1 of 3)

```
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

Extending TicToc

Txc13.cc (2 of 3)

```
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

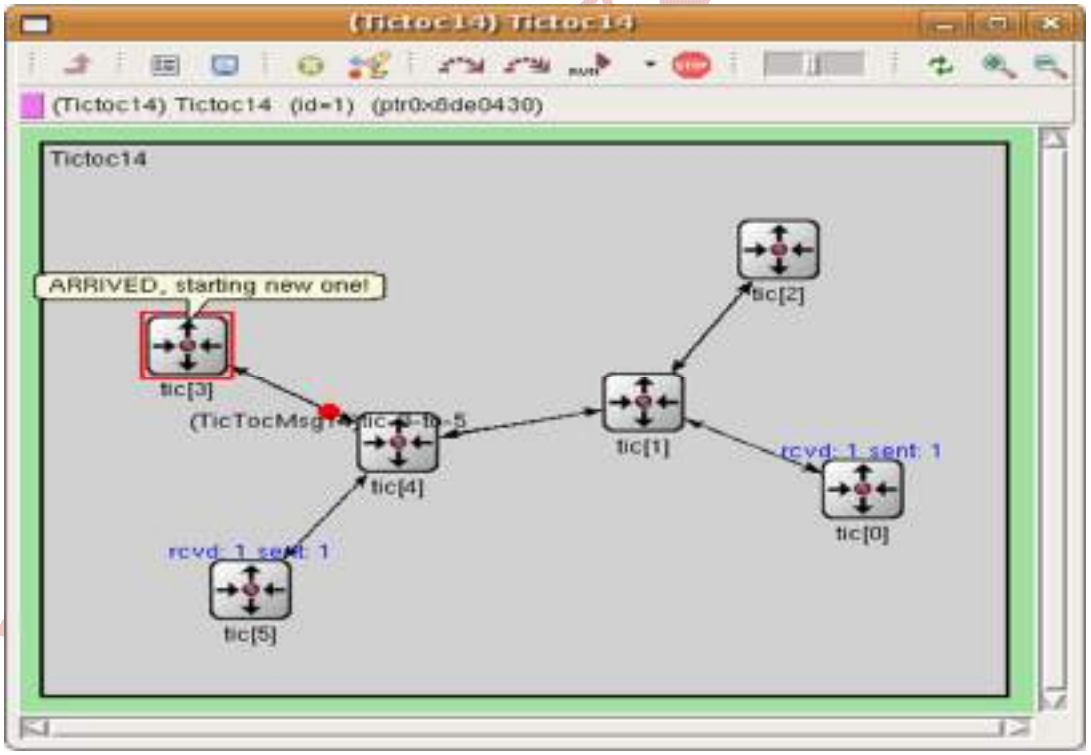
Destination becomes source now

Its not the destination

Extending TicTocTxc13.cc (3 of 3)

```
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



Week 04

TOPIC 42

In this module

We shall extend TicToc tutorial

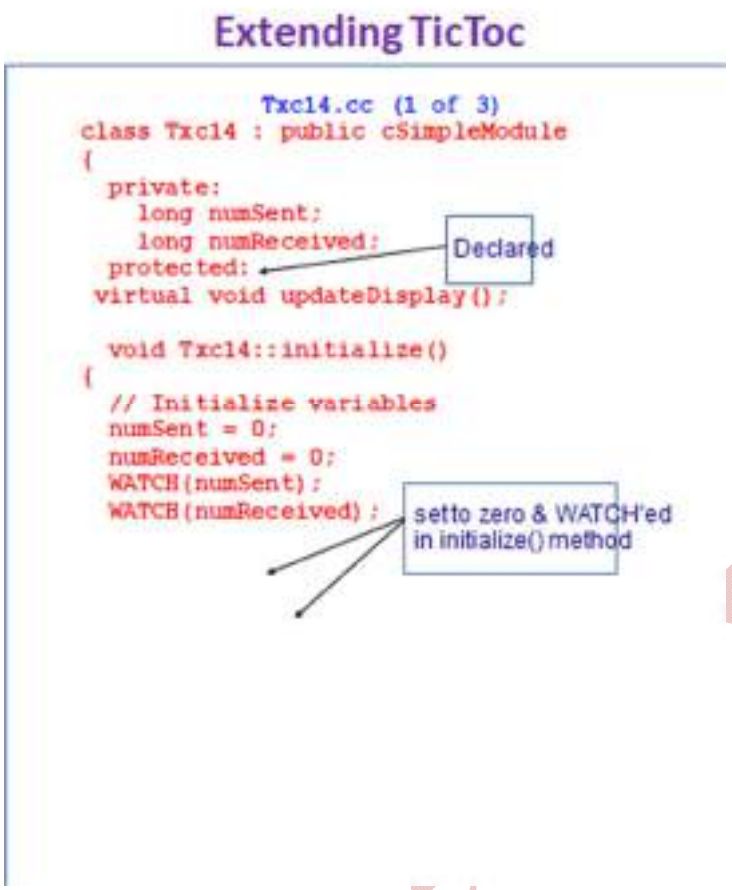
- Refine graphics, & add debugging output
- Add state variables
- Add parameters
- Model processing delay
- And more!

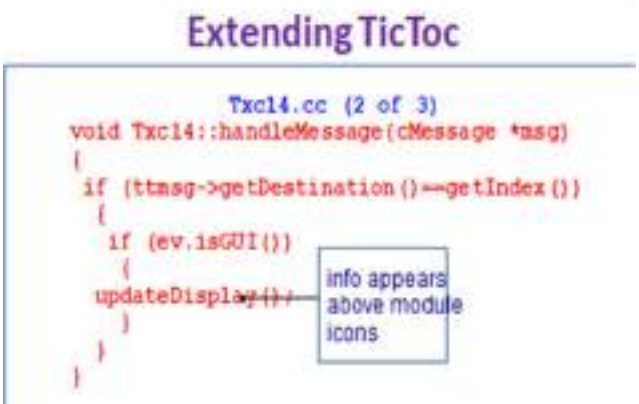
Extending TicToc

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

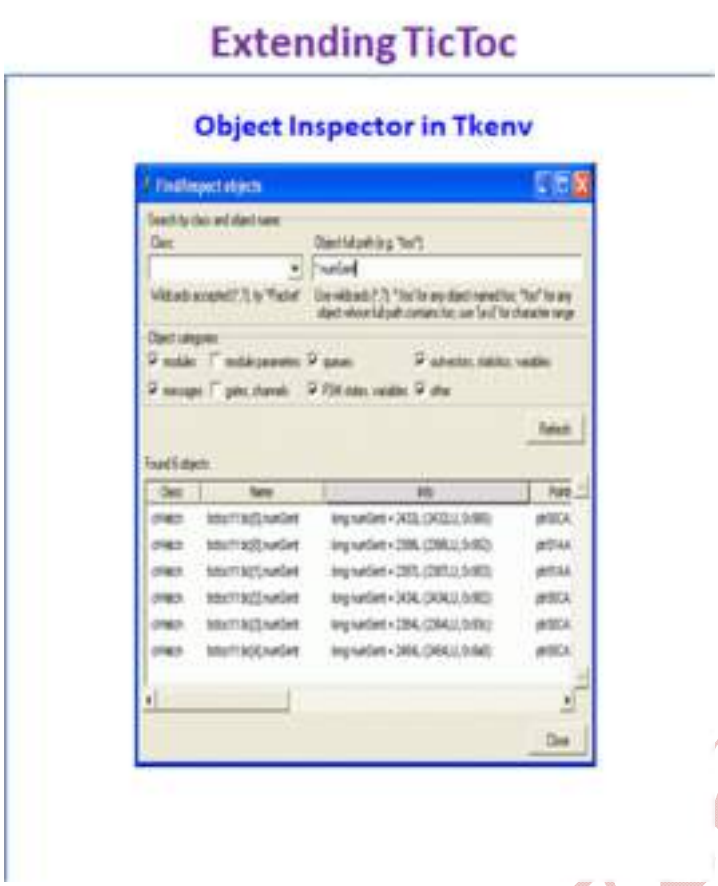


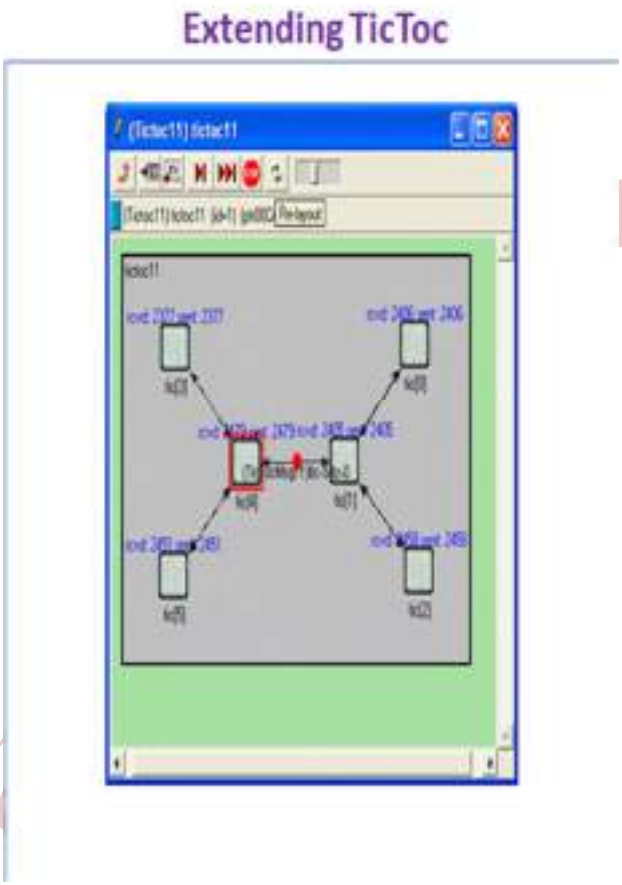


Txc14.cc (3 of 3)

```
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





TOPIC 43

In this module

We shall extend TicToc tutorial

- Refine graphics, & add debugging output
- Add state variables
- Add parameters
- Model processing delay
- And more!

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

Extending TicToc

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

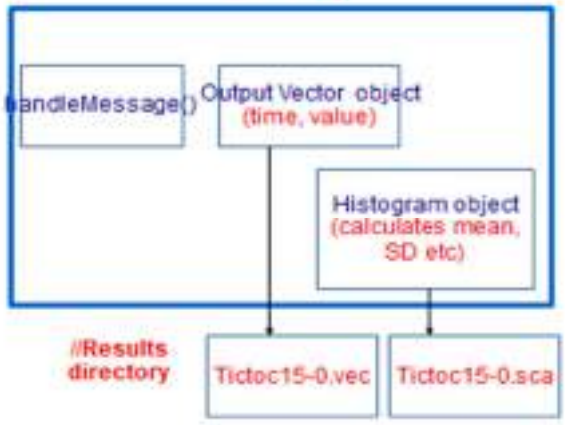
6. Modeling processing delay

11. Channels & inner type definitions

16. Sequence charts and event logs

Extending TicToc

Strategy



Extending TicToc

Txc15.cc (1 of 3)

```
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

Extending TicToc

Txc15.cc (2 of 3)

```
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

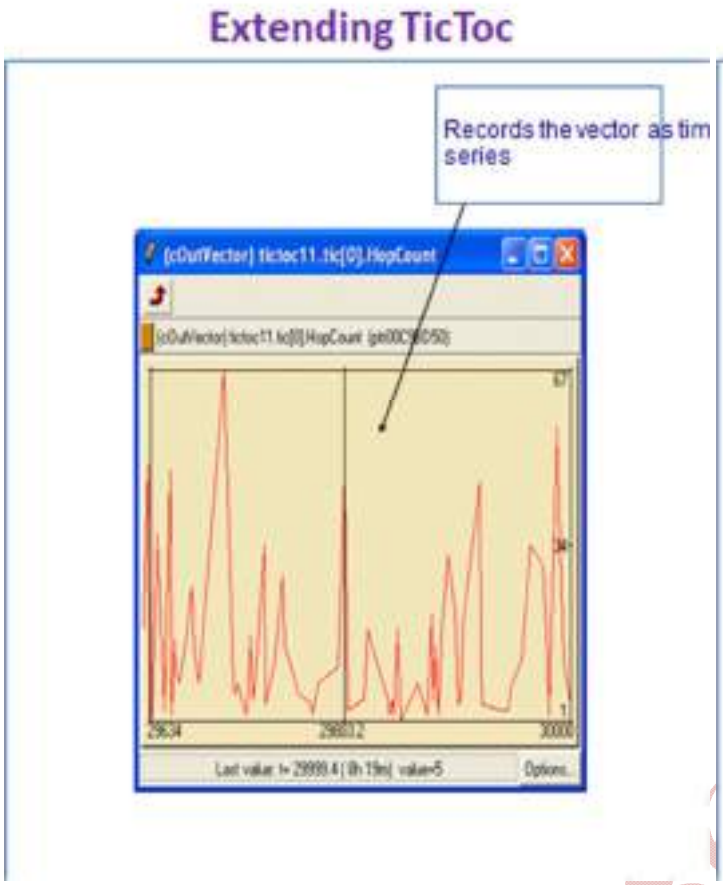
Extending TicToc

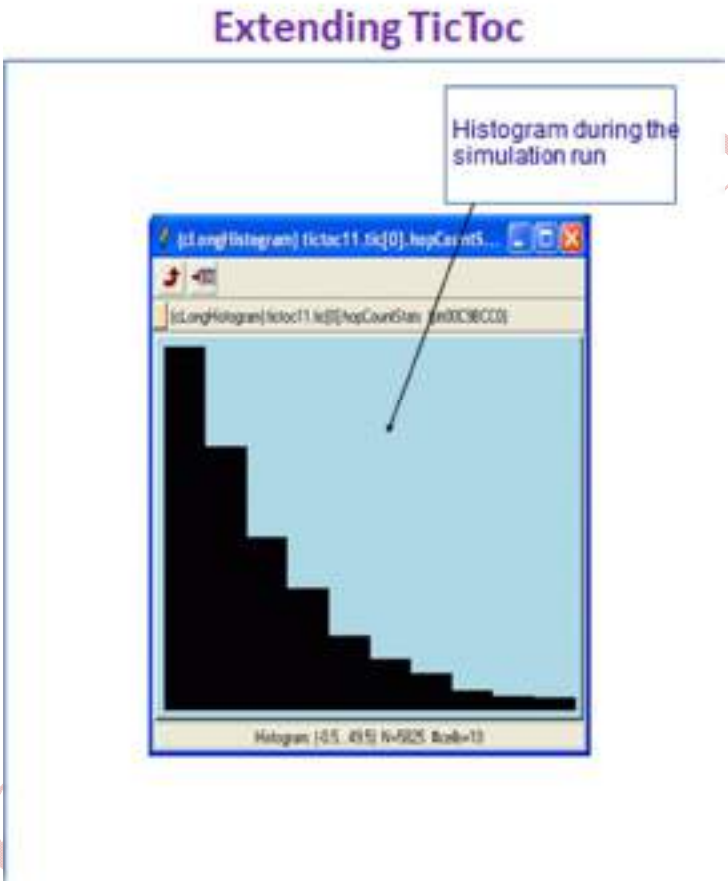
Txc15.cc (3 of 3)

```
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"





TOPIC 44

In this module

We shall extend TicToc tutorial

- Refine graphics, & add debugging output
- Add state variables
- Add parameters
- Model processing delay
- And more!

Visualizing output scalars & vectors

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

Extending TicToc

2. Refine graphics & add debugging output

3. Add state variables

4. Adding parameters

5. Using inheritance

6. Modeling processing delay

7. Random numbers & parameters

8. Timeout, Cancelling timers

9. Retransmitting same message

10. More than two nodes

11. Channels & inner type definitions

12. Using two-way connections

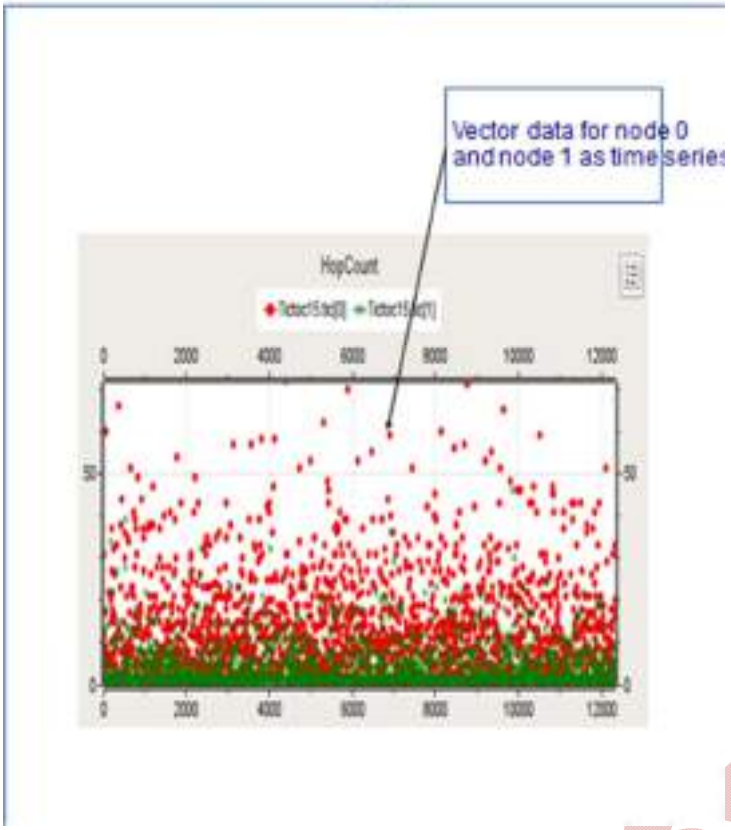
13. Defining our message class

14. Displaying number of packets sent/received

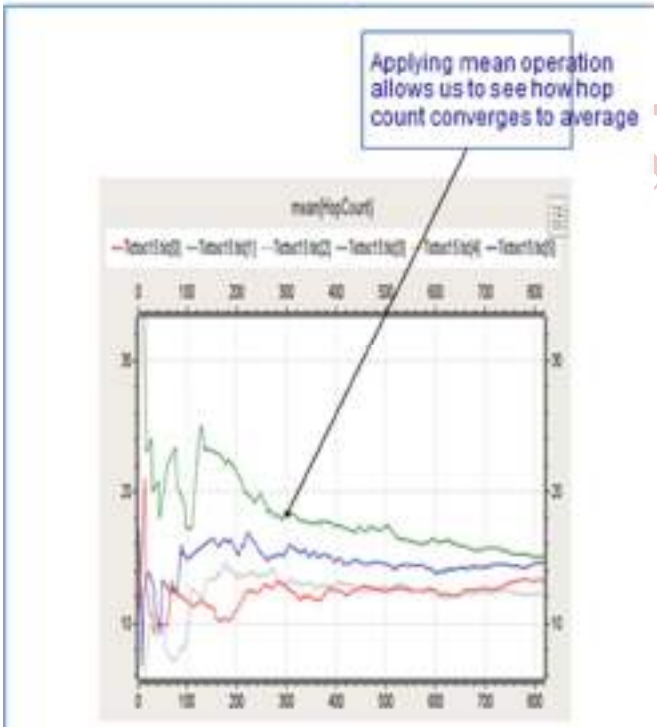
15. Visualizing output scalars and vectors

16. Sequence charts and event logs

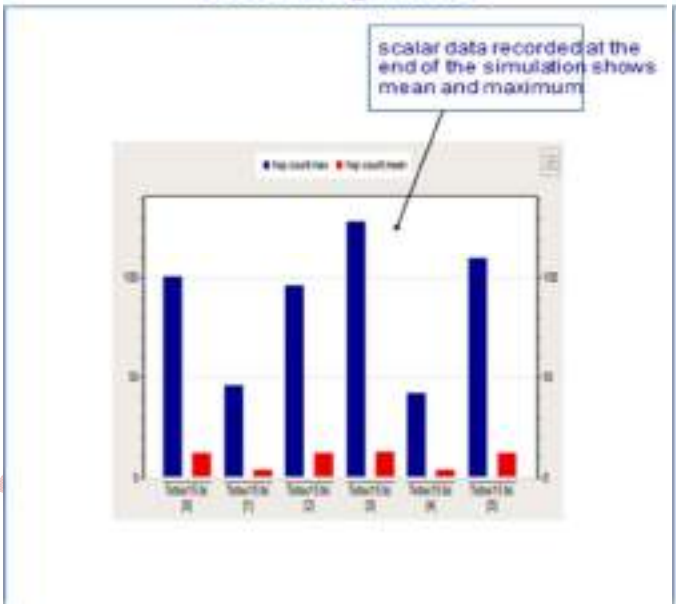
Extending TicToc



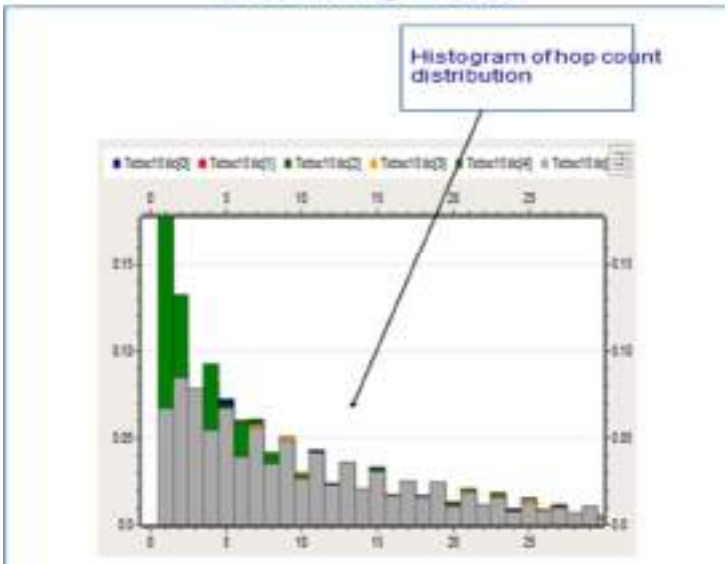
Extending TicToc



Extending TicToc



Extending TicToc



TOPIC 45

In this module

We shall cover

- What is simulations analysis?
- Analysis files
- Creating analysis files
- Using analysis editor

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
  - Chartify the results

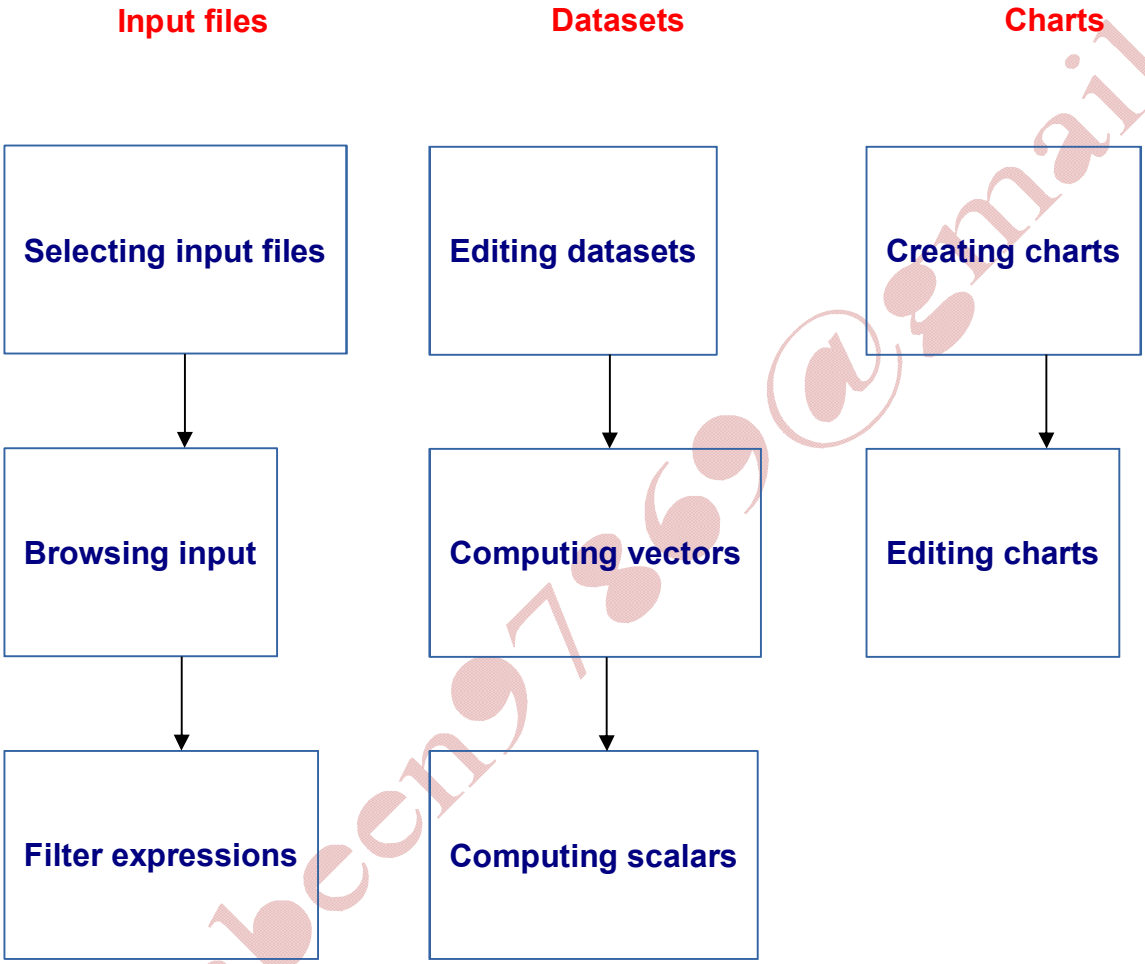
Creating Analysis File

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)

Analyzing Results

Using the Analysis Editor



CS432 Handouts Made by Mahjabeen
mahjabeen97869@gmail.com
contact # 0321 2711298

Analyzing Results

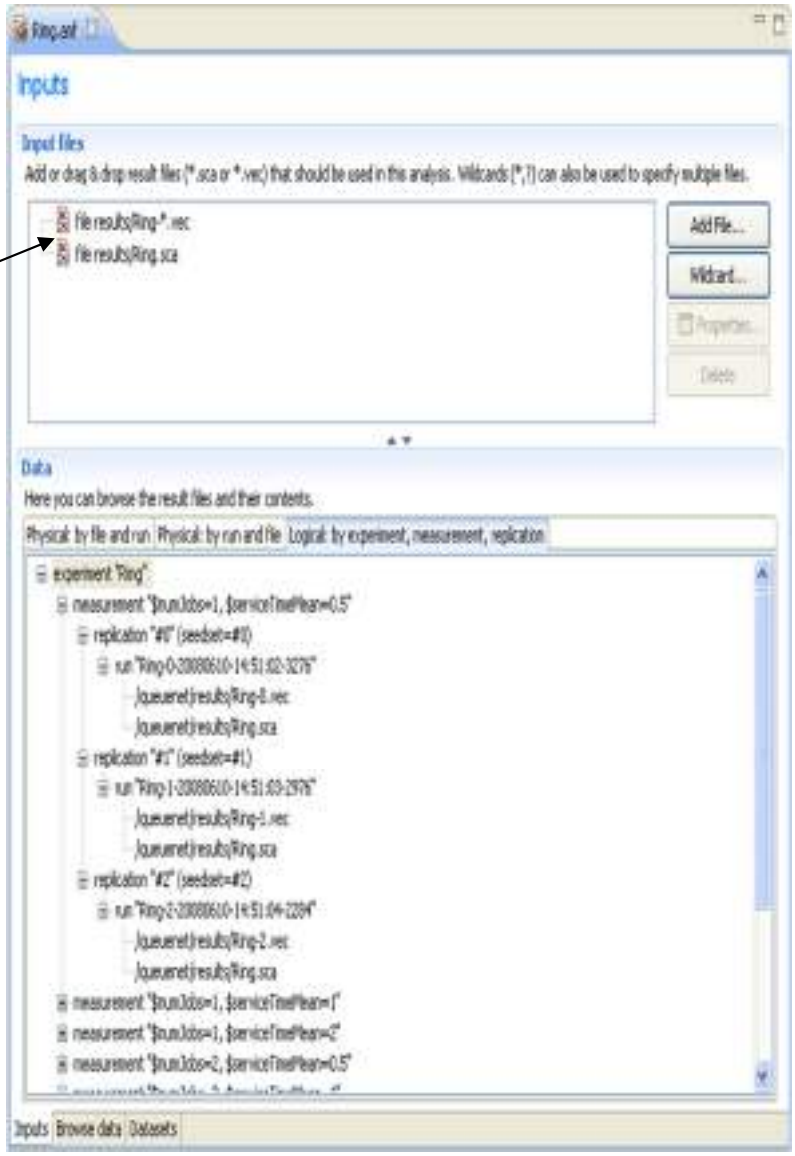
Using the Analysis Editor

Input files

Selecting input files

Browsing input

Filter expressions



Analyzing Results

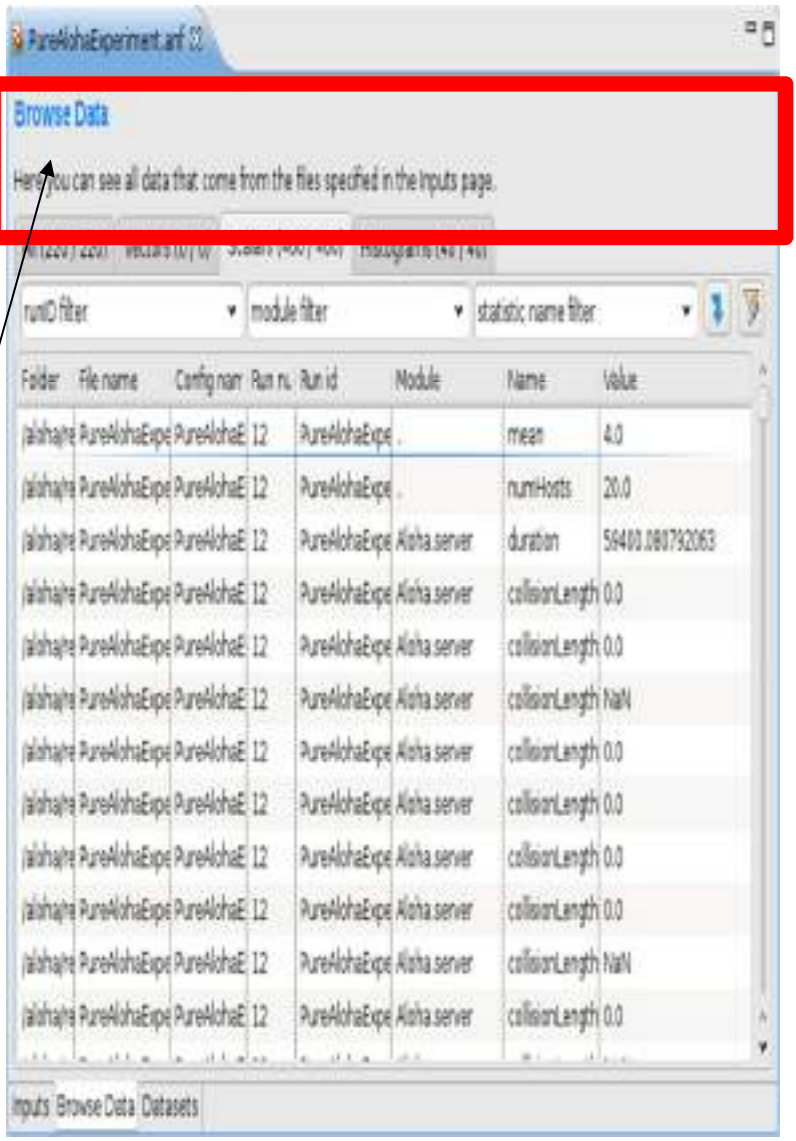
Using the Analysis Editor

Input files

Selecting input files

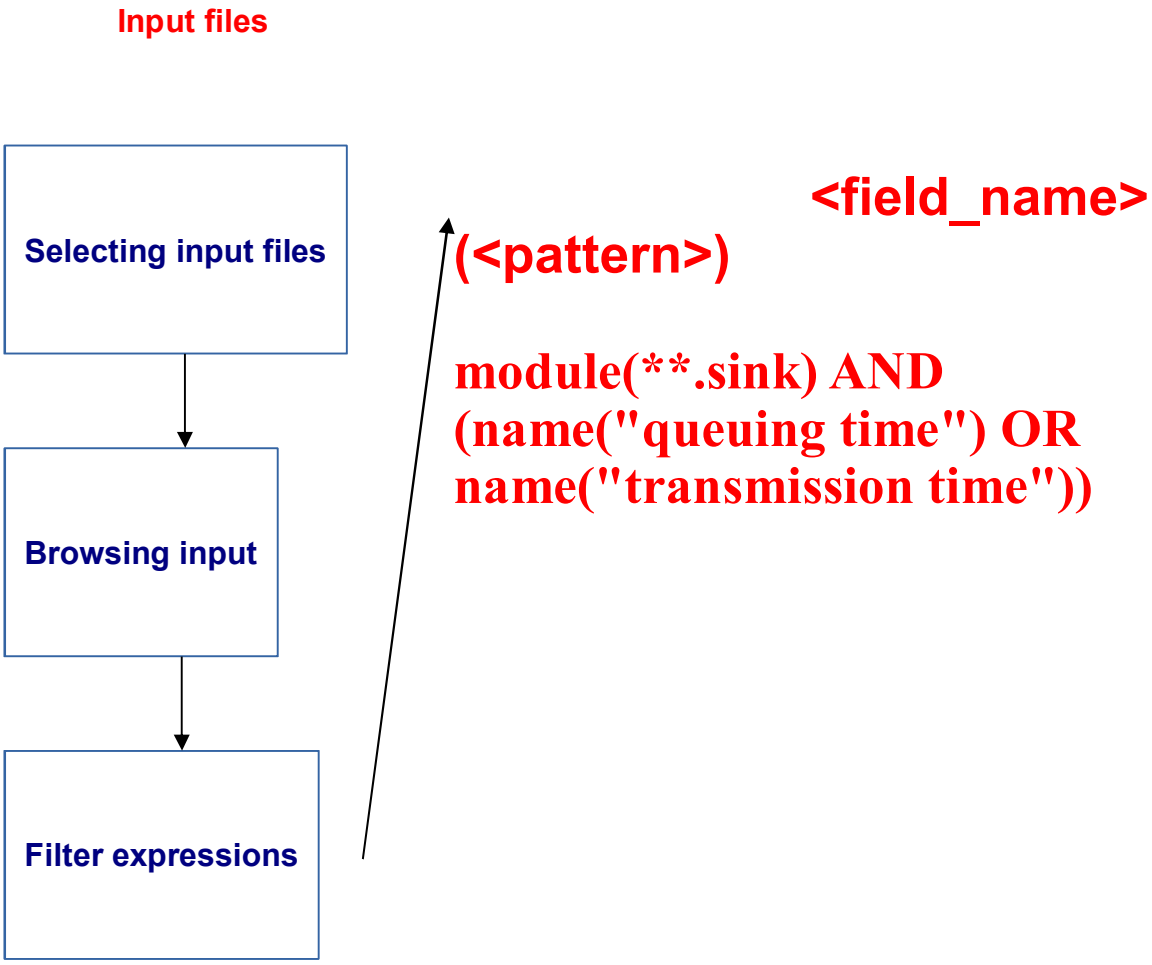
Browsing input

Filter expressions



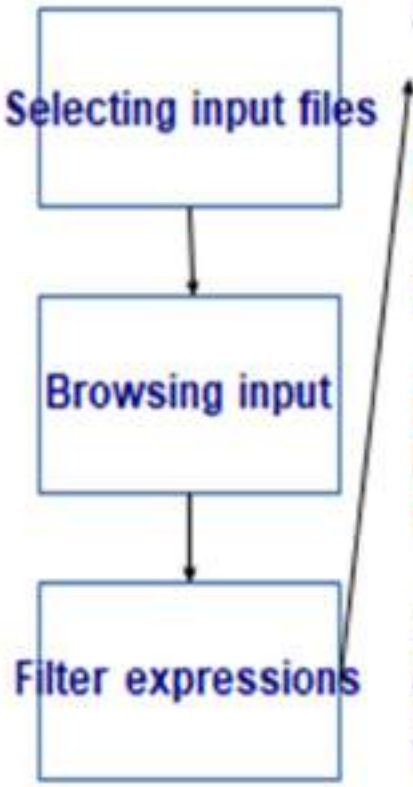
Analyzing Results

Using the Analysis Editor



Analyzing Results

Using the Analysis Editor



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.

Analyzing Results

Using the Analysis Editor

Datasets

Editing datasets

Computing vectors

Computing scalars

Using the Analysis Editor

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

TOPIC 46

In this module

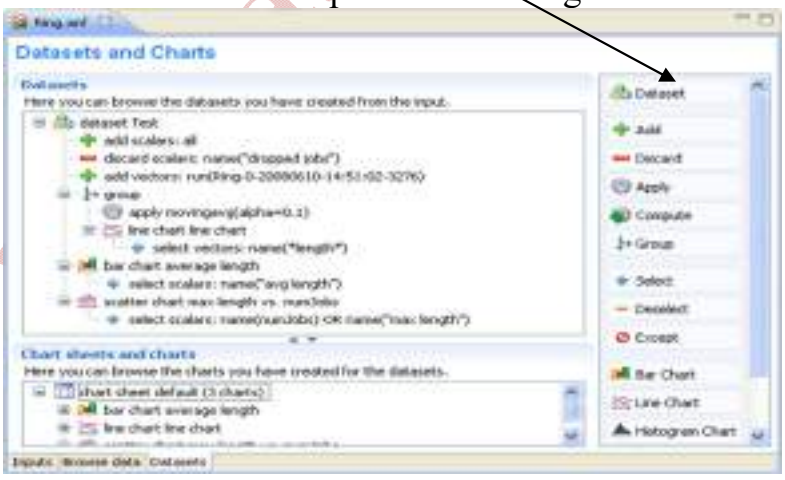
We shall understand

- How to edit datasets?
- Computing vectors
- Computing scalars
- Computation examples

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



Editing Datasets

- 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?

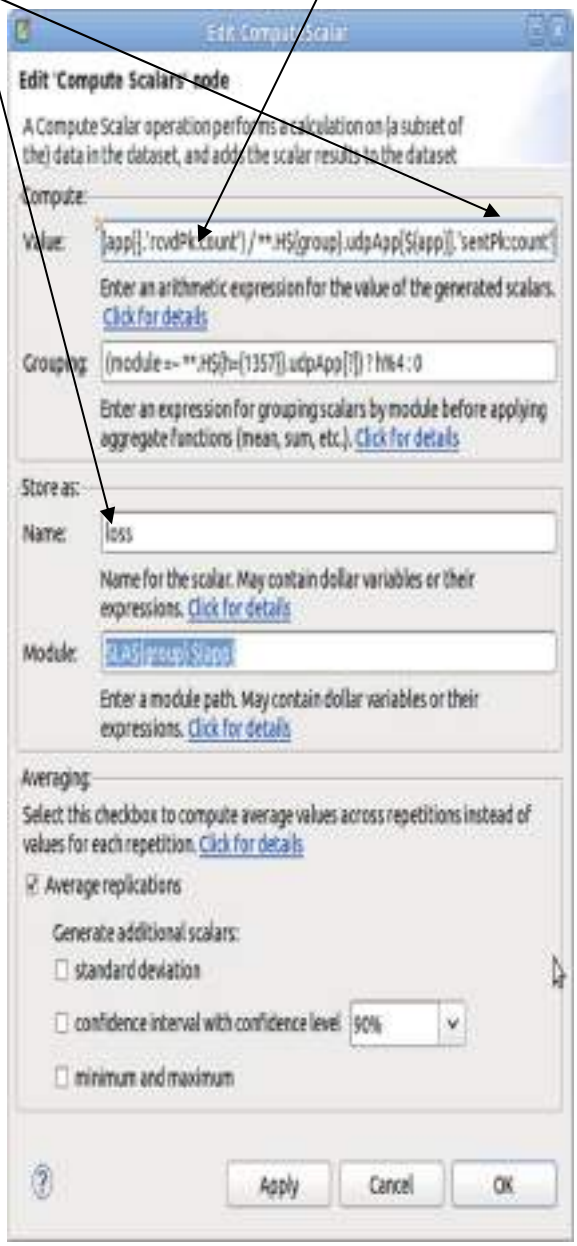
- Both Compute Vectors and Apply to Vectors nodes compute new vectors from other vectors

- Computations can be applied to the data by adding Apply to Vectors /Compute Vectors/Compute Scalars nodes to the dataset

What is Compute Scalars?

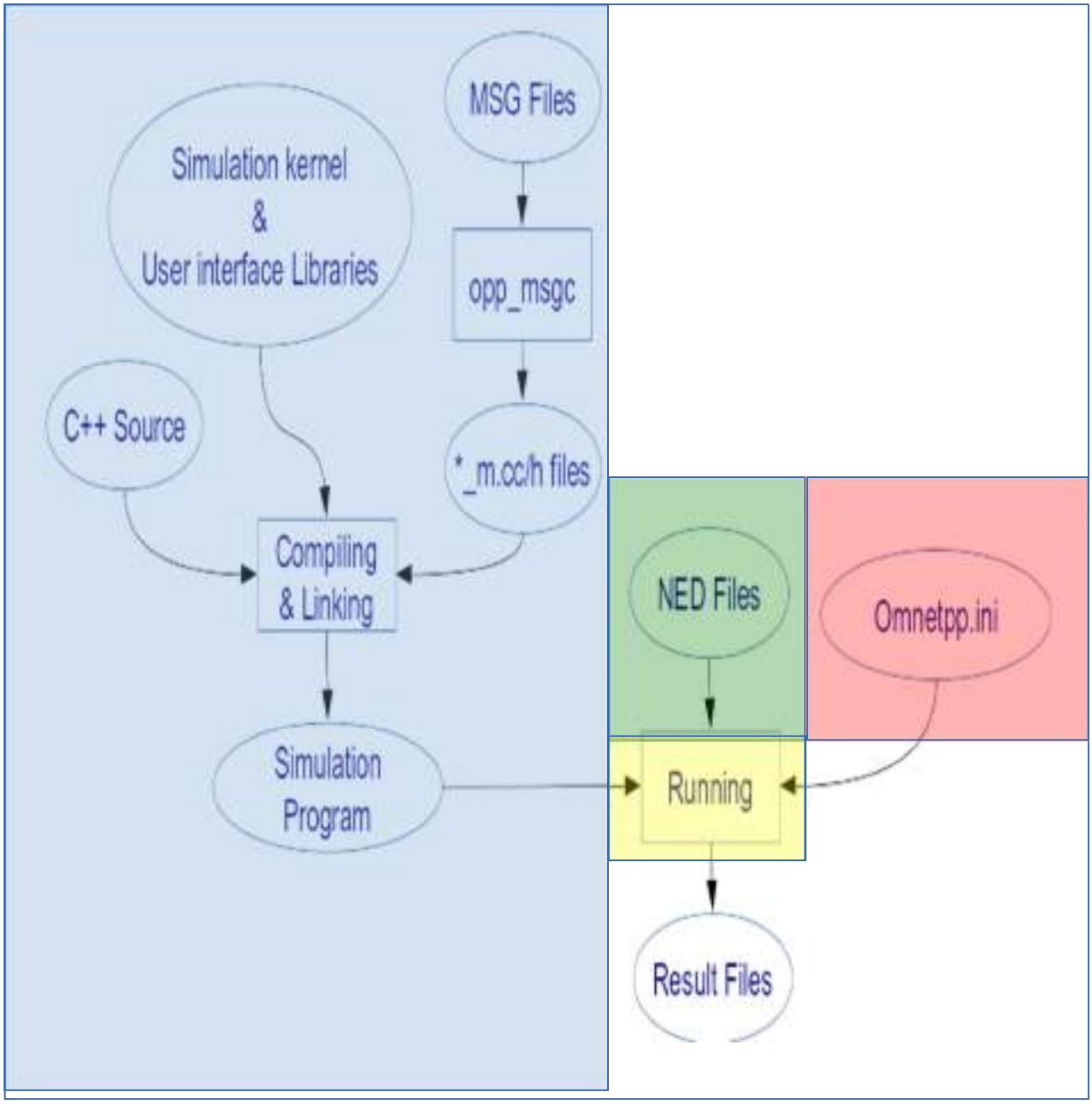
- The Compute Scalars dataset node adds new scalars to the dataset whose values are computed from other statistics in the dataset

- Determine loss through dividing received packets by sent packets



Editing Datasets

Finally we are done!



TOPIC 47

In this module
We shall take various computation examples

- Bit rate
- Throughput
- Total received bytes
- Bytes received by hosts
- Average of peak delay

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

TOPIC 48

In this module

We shall take various computation examples

- Packet loss per client-server pair
- total number of

transport packets

- Modules with largest

RTTs

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 (1 of 2)

- 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

Modules with largest RTT (2 of 2)

Step 2:
Value: average / mean(**.average)
Name: relativeAverage
Step 3:
Value: count(relativeAverage)
Grouping: value > 2.0 ? "Above" : "Normal"
Name: num${group}
Module: Net

TOPIC 49

In this module

We shall cover

- What is a simulation model?
- Types of Simulation Models
- 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 (1 of 2)

- Domain-specific functionality is provided by model frameworks
  - WSNs
  - Ad-hoc networks
  - Internet protocols,
  - Performance modeling
  - Photonic networks, etc.,
- Developed as independent projects

Types of Simulation Model (2 of 2)

- Reusability of models in OMNeT++ is due to its modular architecture
- Simulation models are easily integrated into OMNET++

https://omnetpp.org/models

Some Well-known Types

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

Simulation Models and INET

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 framework 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

TOPIC 50

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

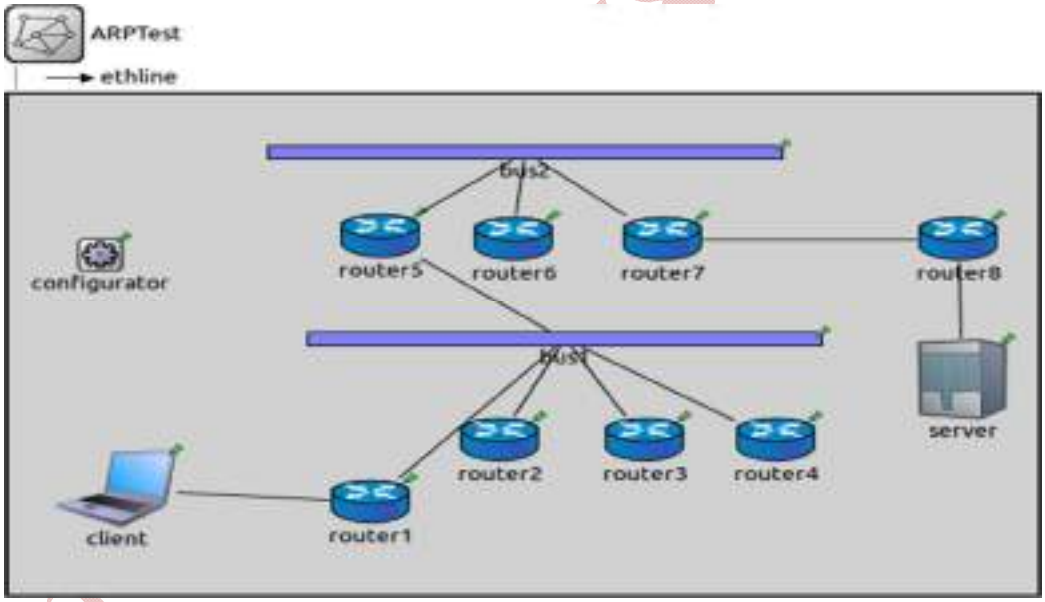
Why ARP scenario?

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

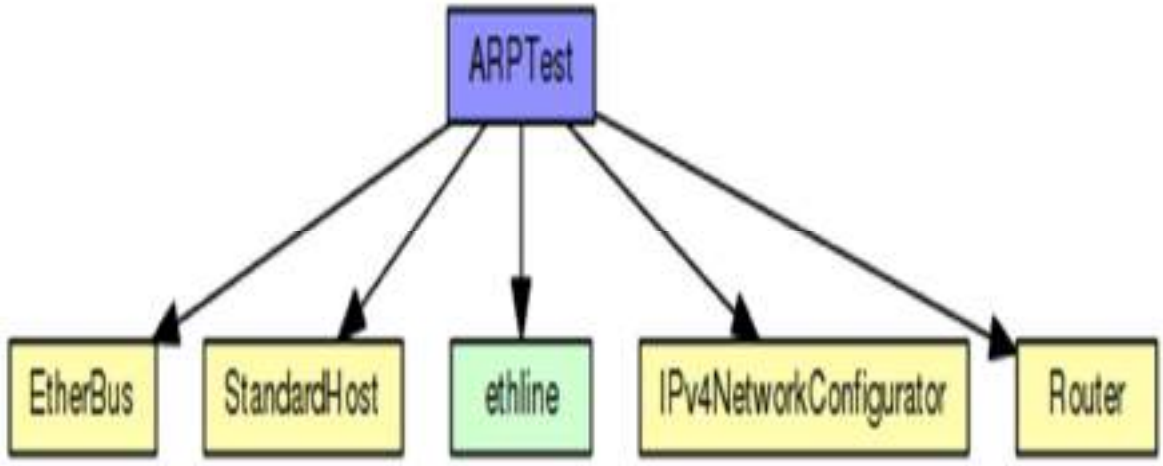
https://omnetpp.org/doc/inet/api-current/neddoc/index.html

Scenario

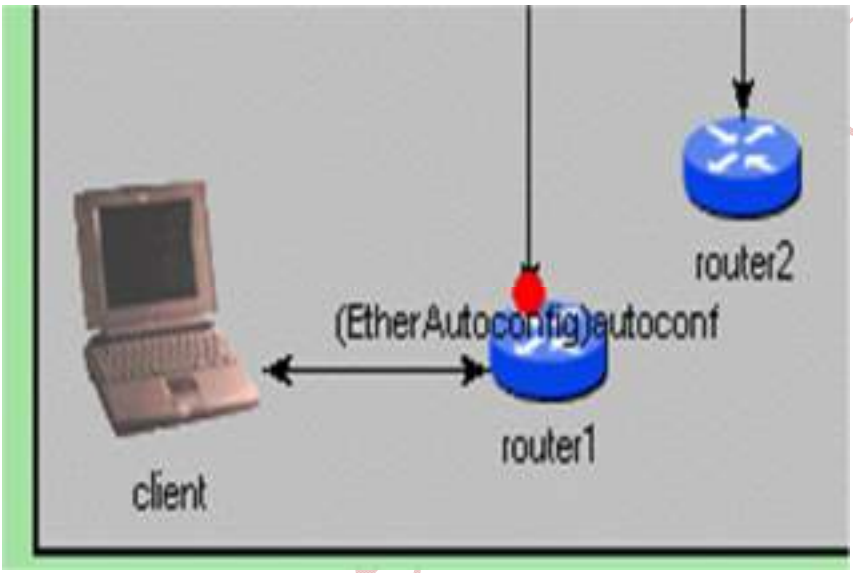
- Client computer opens TCP session with server
- Rest of operations (including ARP) follow
  - ARP has to learn the MAC address for the default router



Usage Diagram for ARP



On simulation start
Ethernet autoconfiguration precedes ARP



TOPIC 51

Design Tour of INET 2

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

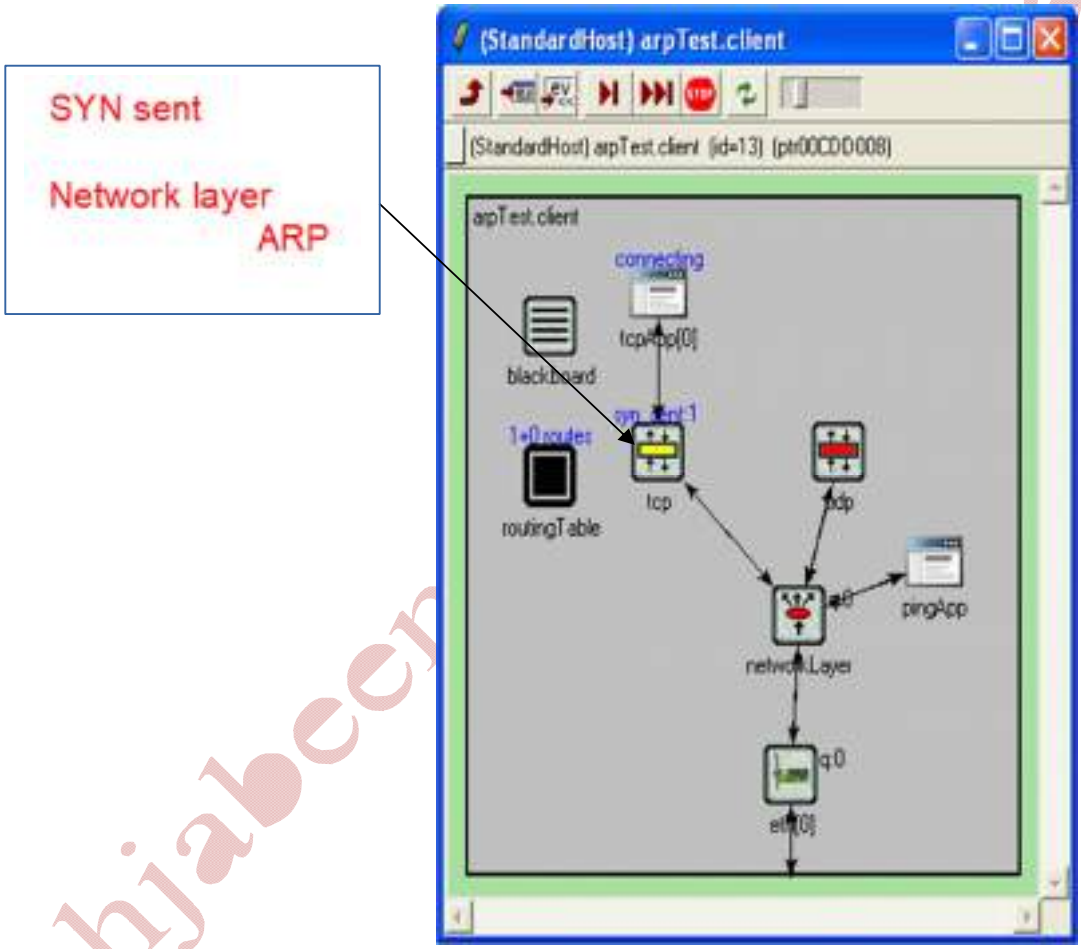
- Understand how ARP works in Ethernet environments
- Walk through features of INET
- Peek into various
  - Packets

- Queues
- Internal tables

Entities at work

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

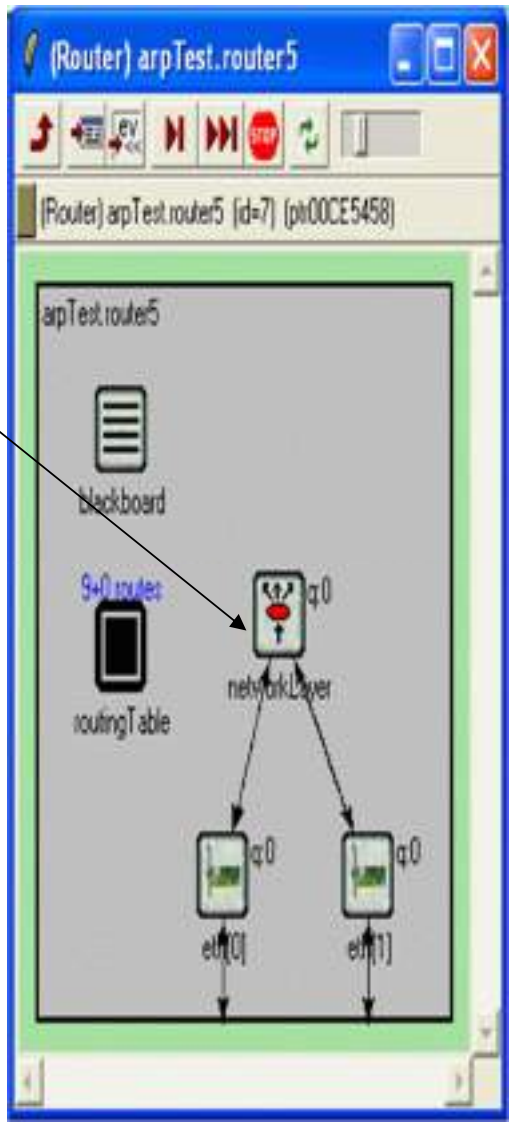
TCP Client



Design Tour of INET 2

Router

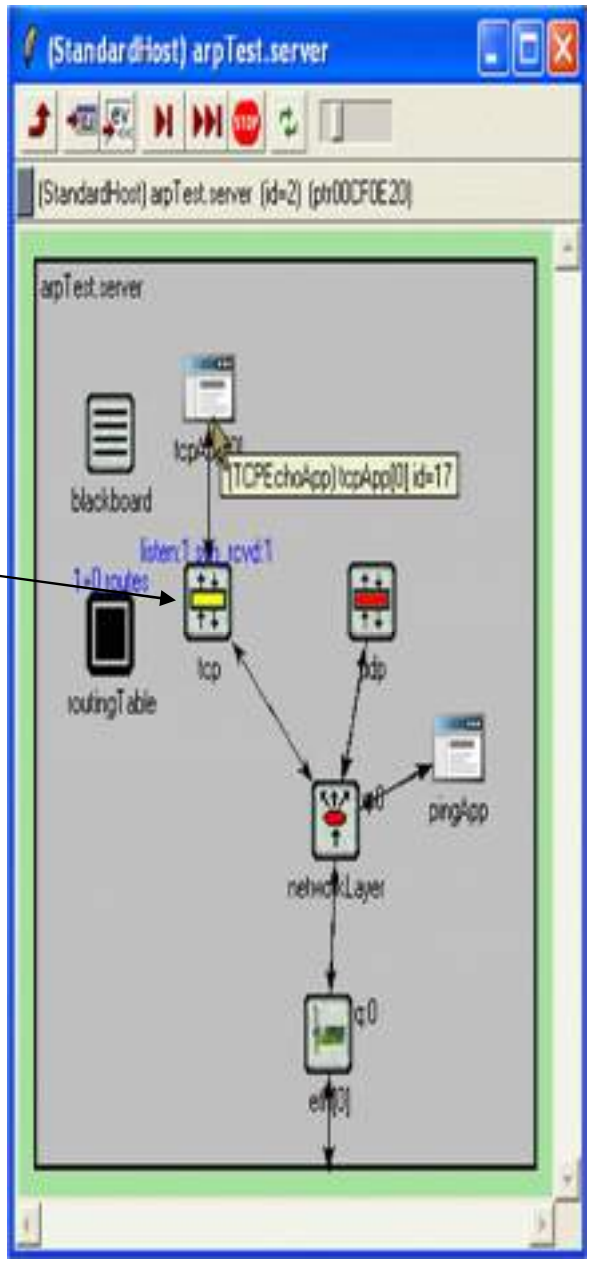
ARP request
Activates
Router to send
ARP reply



Design Tour of INET 2

TCP Server

ARP request/reply Retrieve MAC addresses at every hop till TCP SYN request reaches in IP packet at the server that sends TCP SYN/ACK

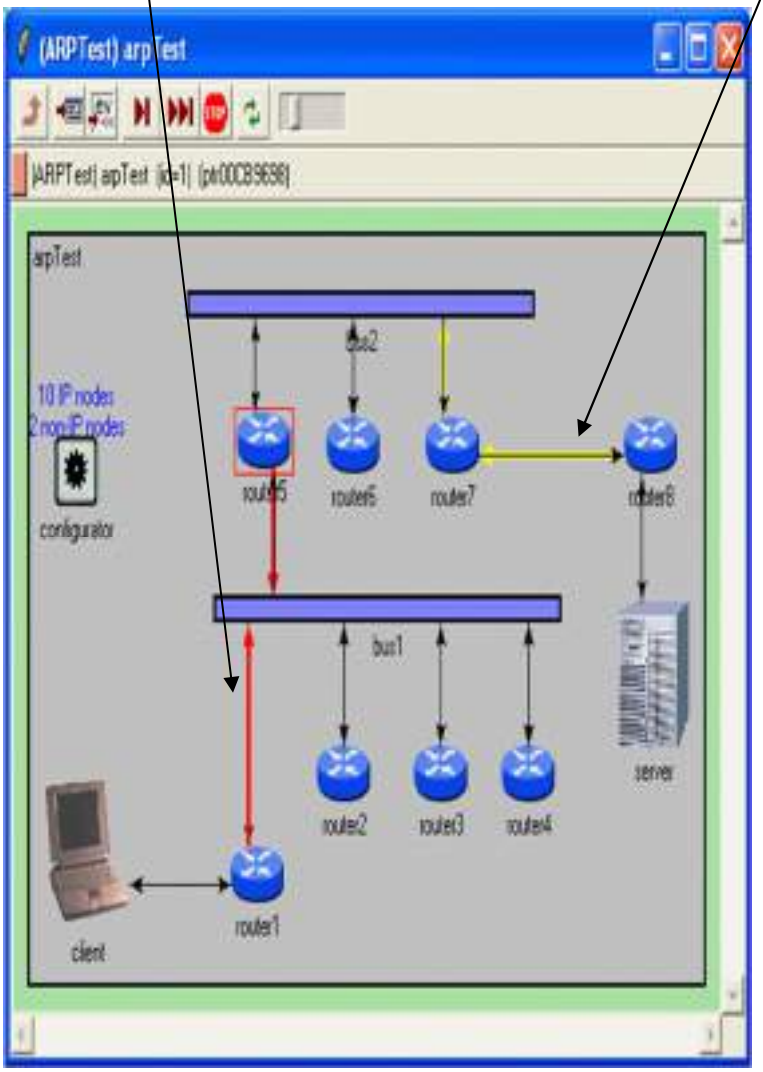


Design Tour of INET 2

End-to-end transmission

Red: Collided and backing off

Yellow: Node transmitting on link



TOPIC 52

Design Tour of INET 3

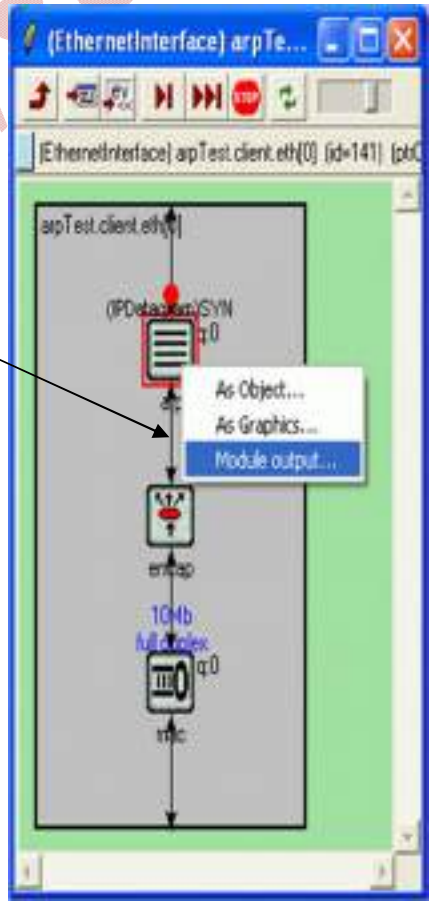
In this module
We shall take a guided
Tour of INET to

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

Ethernet Compound Module

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

Right click to see the module output ARP)arpTest.client .eth[0].arp



Design Tour of INET 3

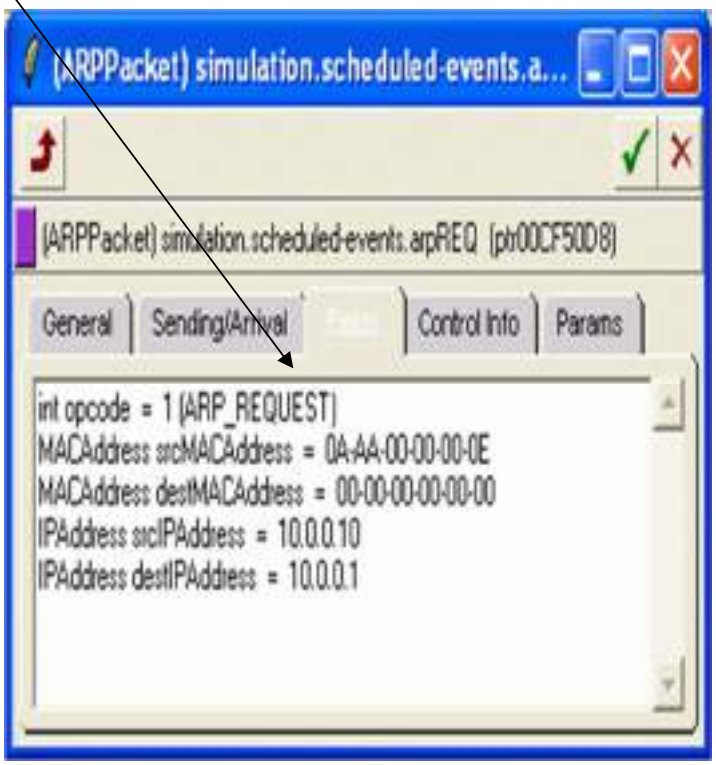
arpTest.client.eth[0].arp

module output

Design Tour of INET 3

Inside ARP Packet

ARP Broadcast Message



Design Tour of INET 3

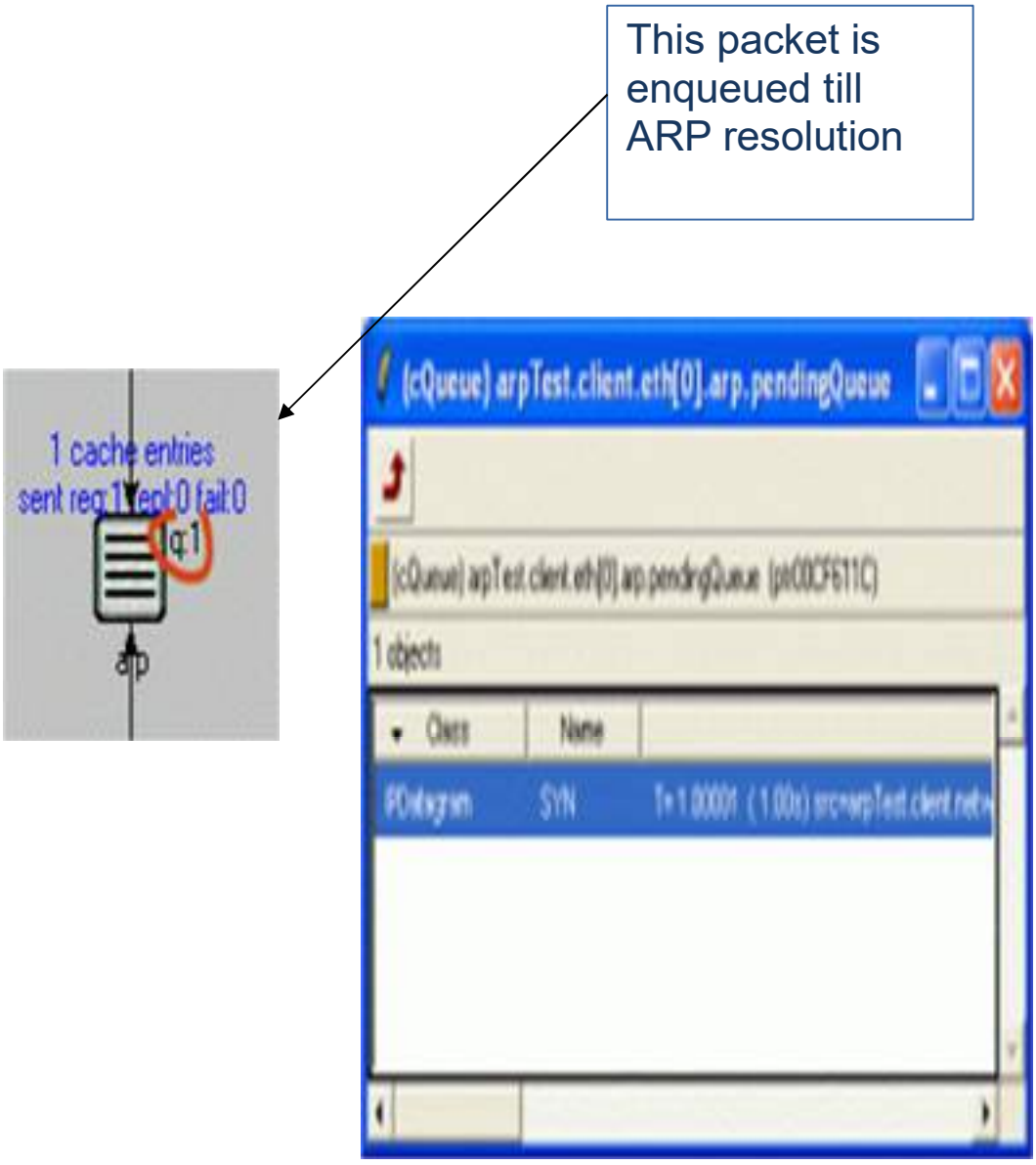
ARP Packet Class (Generated by .msg file)

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

This packet is appended with broadcast address in control info (a small data structure)

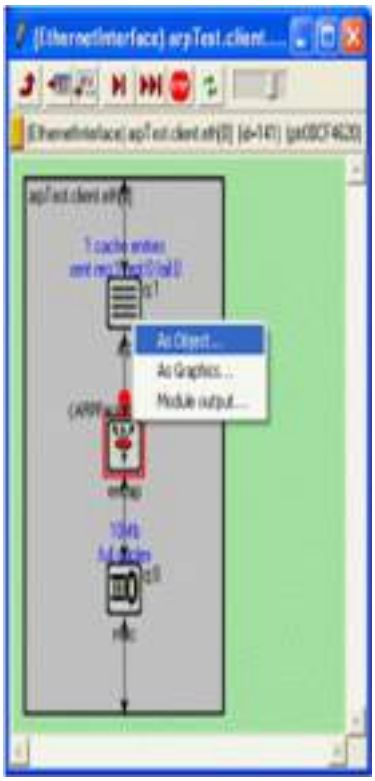
Design Tour of INET 3

Packet Queue (Contains IP Packet)

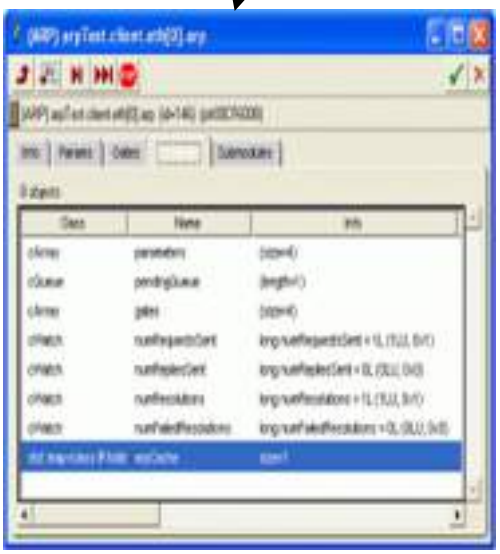


Design Tour of INET 3

ARP Cache Build-up



Contents of ARP Cache (entries with soft timers)



TOPIC 53

Introduction to top-down approach to modelling and simulation

In this module

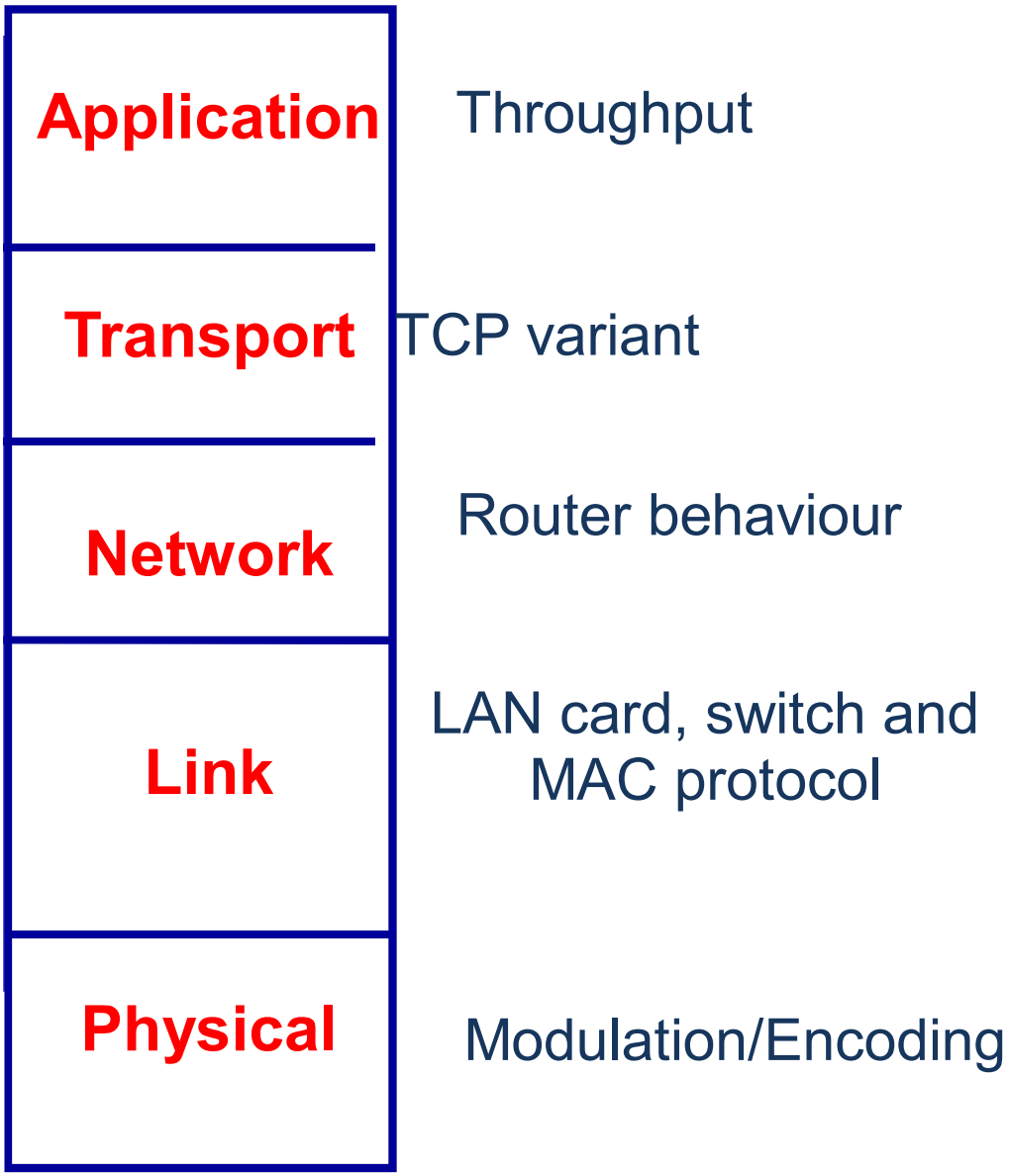
We shall understand

- What is top-down approach to NeMS?
- Phased roll-out
- User-centric aspects

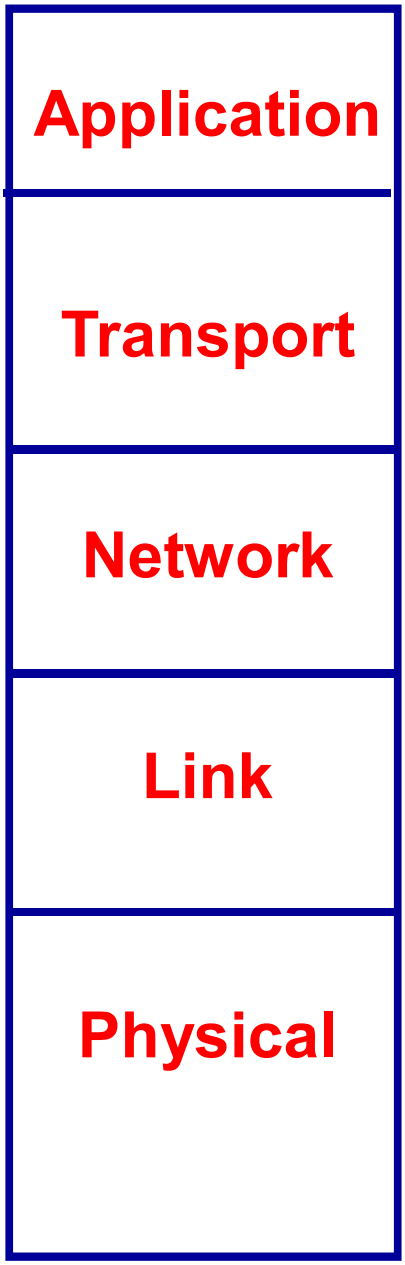
- **Top-down approach to NeMS**
- Networks are complex to design
- One-time design of simulation is cumbersome
- **Top-down**: Phased roll-out of model-simulate cycle
  - Iterative

Introduction to top-down approach to modelling and simulation

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



Scalability	Availability	Performance	Security
Manageability	Usability	Adaptability	Affordability



Throughput

TCP variant

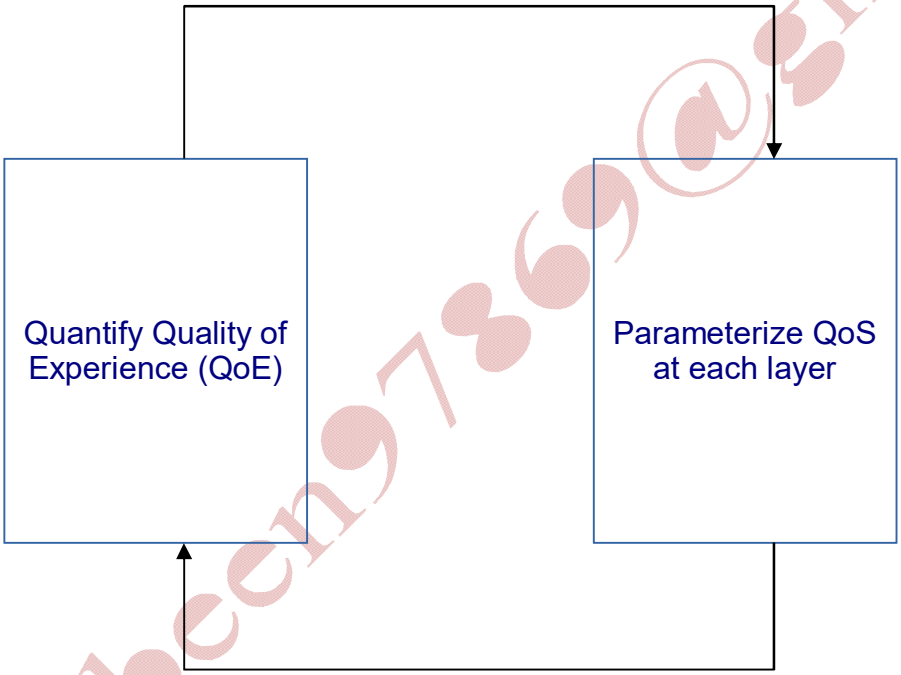
Router behaviour

LAN card, switch and MAC protocol

Modulation/Encoding

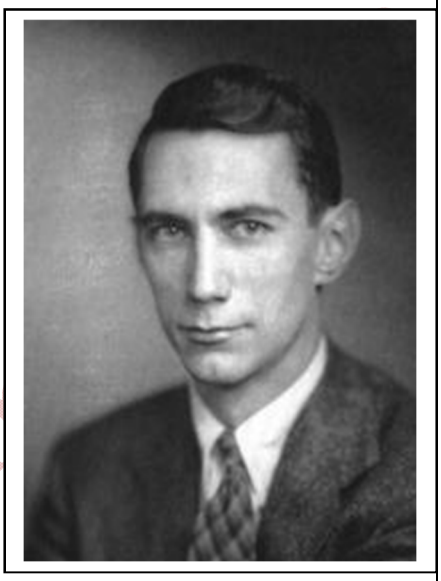
Introduction to top-down approach to modelling and simulation

Strategy



TOPIC 54

Rules for Mathematical Reading



In this module
We develop an understanding of

- Quantification
- Formalism
- Best practices to read mathematical expressions

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

END

TOPIC 55

Rules for Mathematical Writing

In this module
We shall know
- Constituents of an equation
- Easy math writing

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

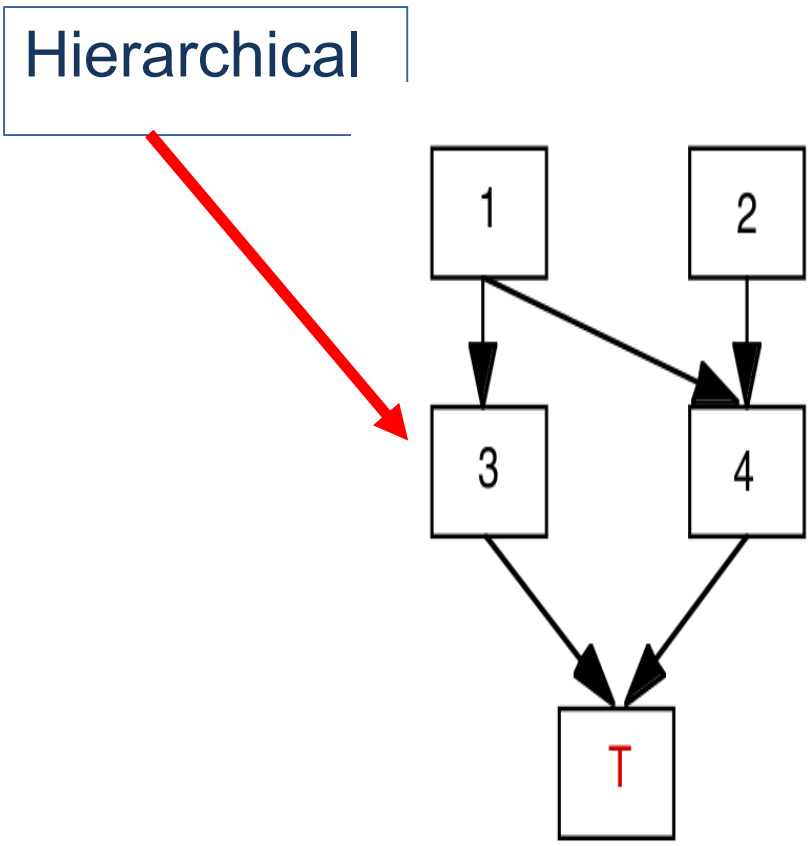
Constituents of an equation?
- Expressions consist of one or more of these arguments
  - Numerical constants
  - Symbolic names
  - Mathematical operators
  - Functions
  - Conditional expressions

Easy math writing (1 of 3)
- 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
- 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**

Rules for Mathematical Writing

Relationship between arguments of a segment (1 of 3)

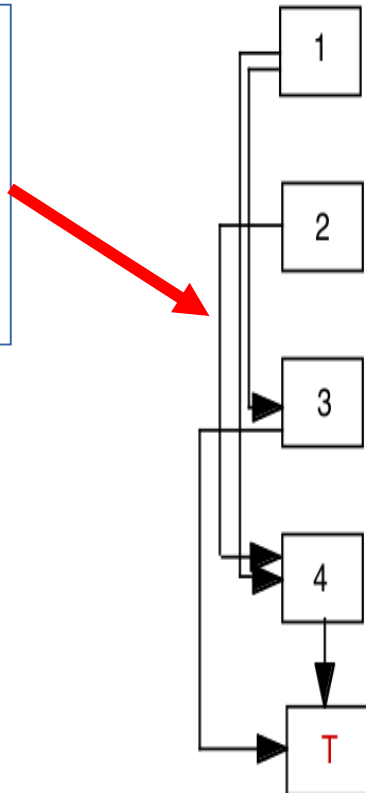


CS432 Handouts Made by Mahjabeen
mahjabeen97869@gmail.com
contact # 0321 2711298

Rules for Mathematical Writing

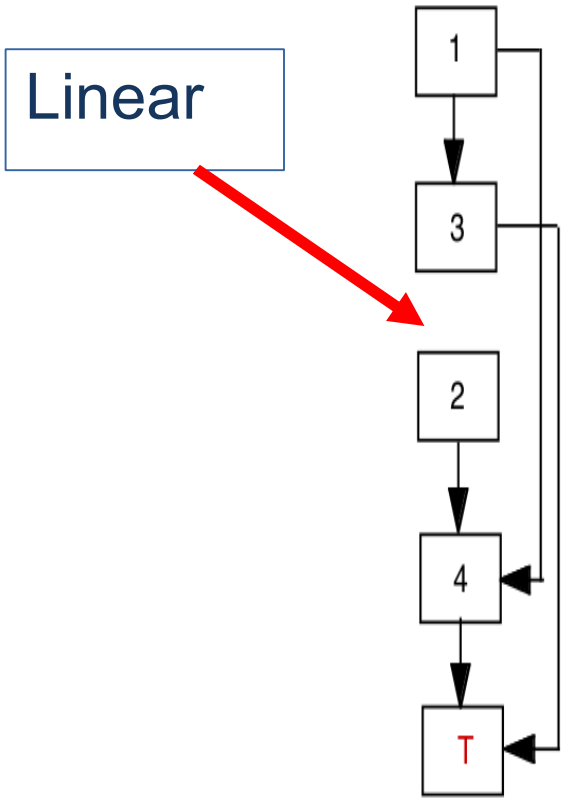
Relationship between arguments of a segment (2 of 3)

Nonlinear with crossing



Rules for Mathematical Writing

Relationship between arguments of a segment (3 of 3)



END

Week 05

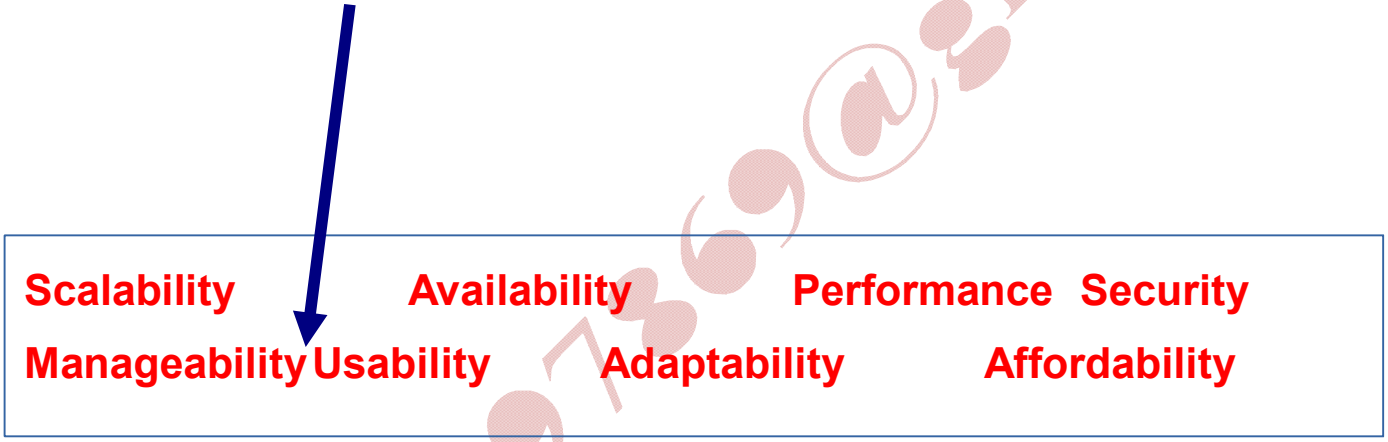
TOPIC 56

QoE—Usability

In this module
We shall explore

- What is usability?
- Some usability expressions
- Connotations of usability on network model

Everything starts with "You"



What is usability? (1 of 2)

- 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

END

TOPIC 57

QoE—Scalability

In this module

We shall explore

- What is Scalability?
- Scalability analysis

- Connotations of scalability on network model

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

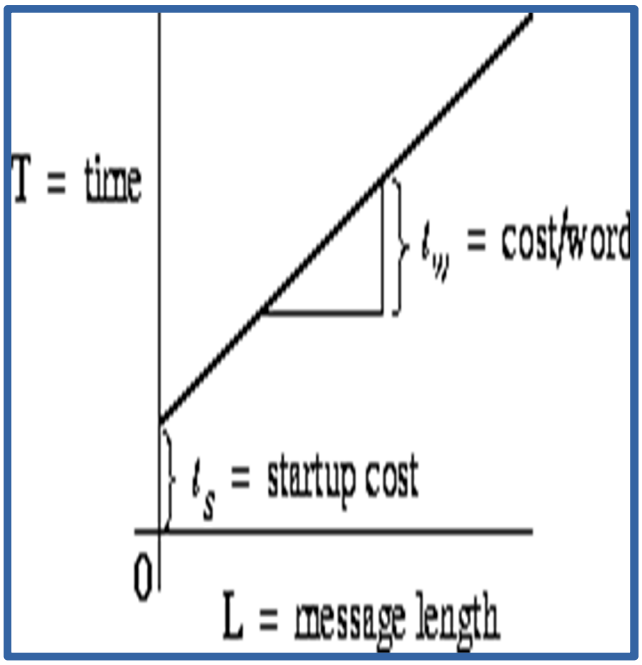
$E_{Relative} = T_1 \text{✿} (\text{No. of hosts} \diamond T_{\text{No of hosts}})$

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

Understanding execution time

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

$T_{msg} = t_s + t_wL$



Connotations

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

END

TOPIC 58

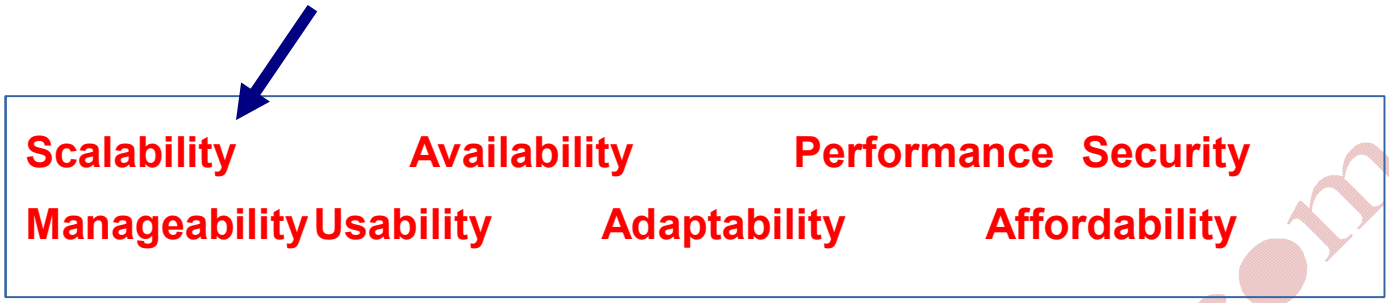
QoE—Planning for Expansion

In this module

We shall understand

- Why plan to expand?
- Considerations for planning

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
- Nodes and locations
  - End hosts
  - Switches
  - Routers
- Equipment scalability
  - No of ports
- Naming system
  - Extensible tuple

(Node ID, Network ID)

Considerations for Planning (2 of 2)

- Application-specific protocol choices

Email	File transfer, sharing & access
DB access & updating	Web browsing
Network game	Remote terminal
Videoconferencing	Video on

TOPIC 59

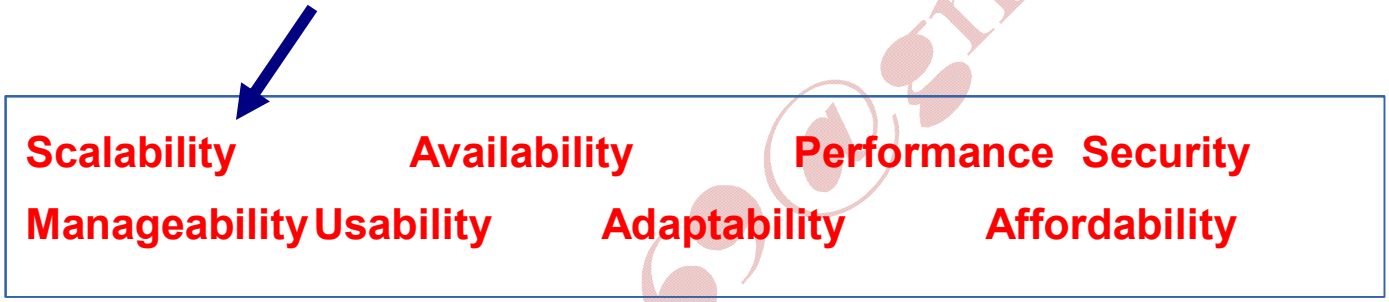
QoE—Expanding Access to Data

In this module

We shall understand

- What is data access?
- Metcalfe's law
- Application of the law
- Connotations

Scalability without continued access to data is futile



Access to data

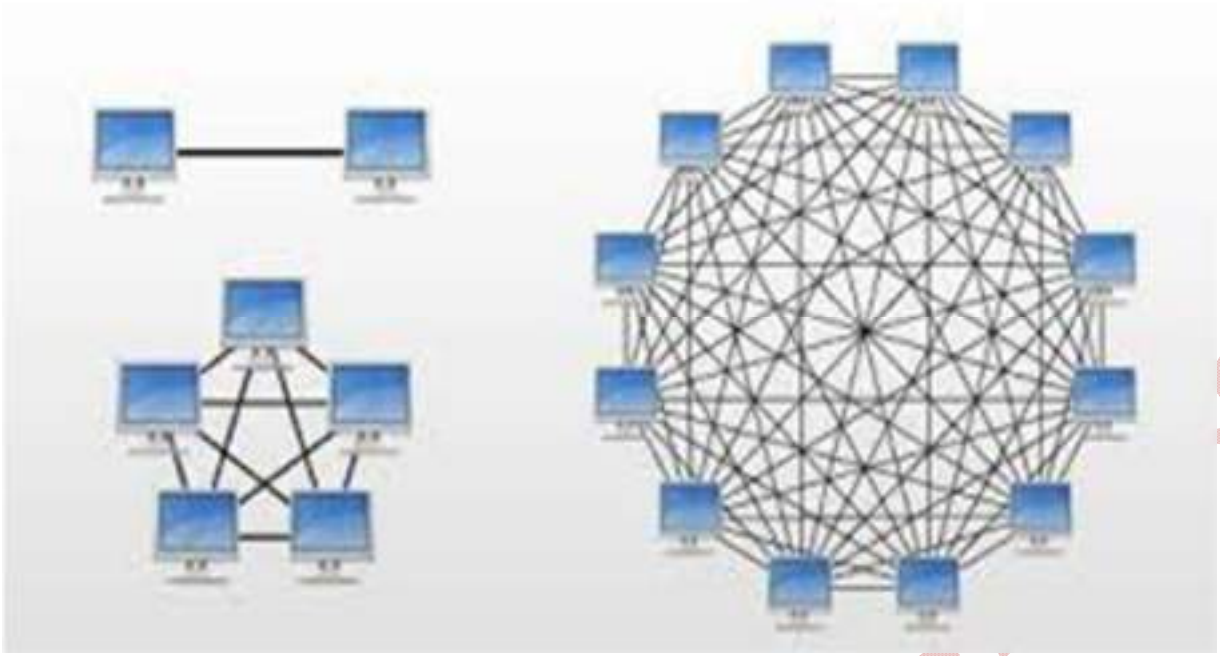
- Social networking has emerged
- Extranets need topology definitions & dedicated bandwidth allocation
  - Classic 80-20 rule <?>
- Increased access
  - Data available to more departments
  - Increased utilization of network services

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

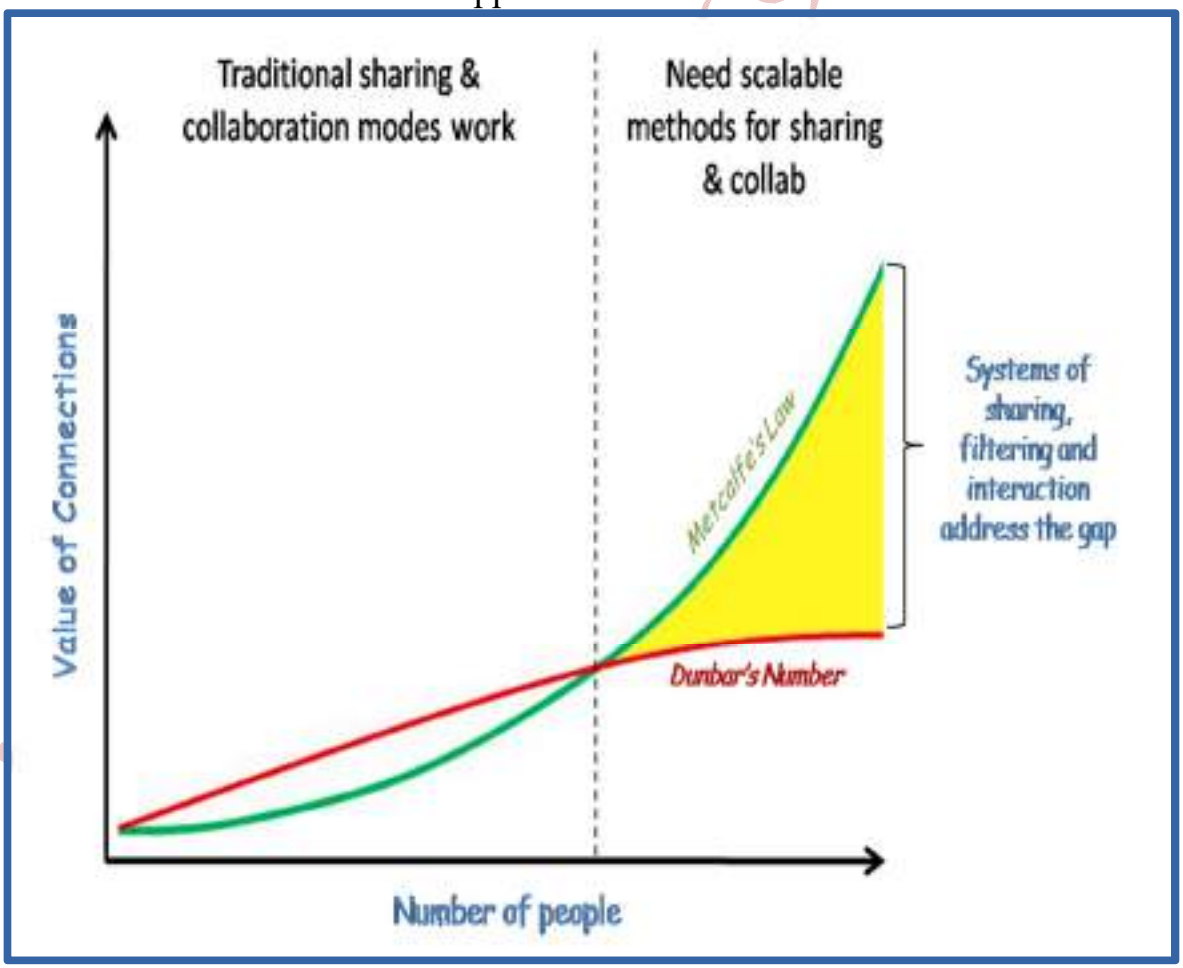
Expression for Metcalfe's Law

- $n(n-1)/2$ 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

END

TOPIC 60

QoE—Constraints on Scalability

In this module

We shall understand

- Parts of model
- Constraints of scalability
- Connotations

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

END

TOPIC 61

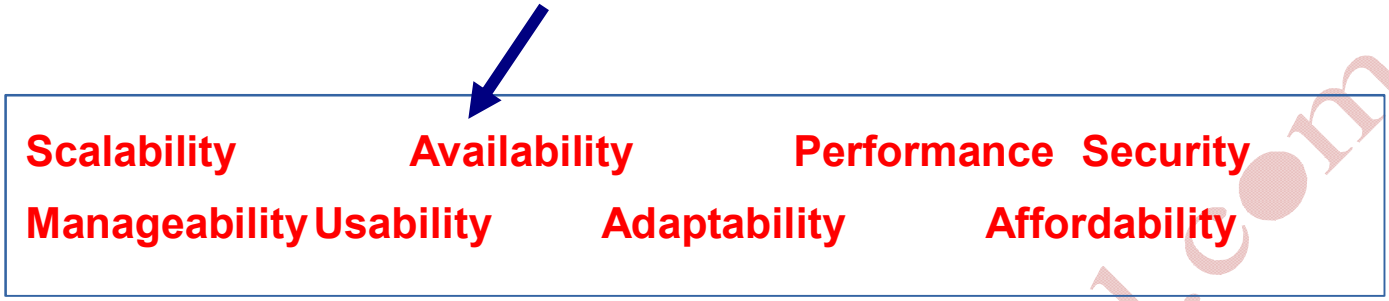
QoE—Availability

In this module

We shall understand

- What is availability?
- Vs reliability
- Vs capacity
- Vs Redundancy
- Specifying availability requirements

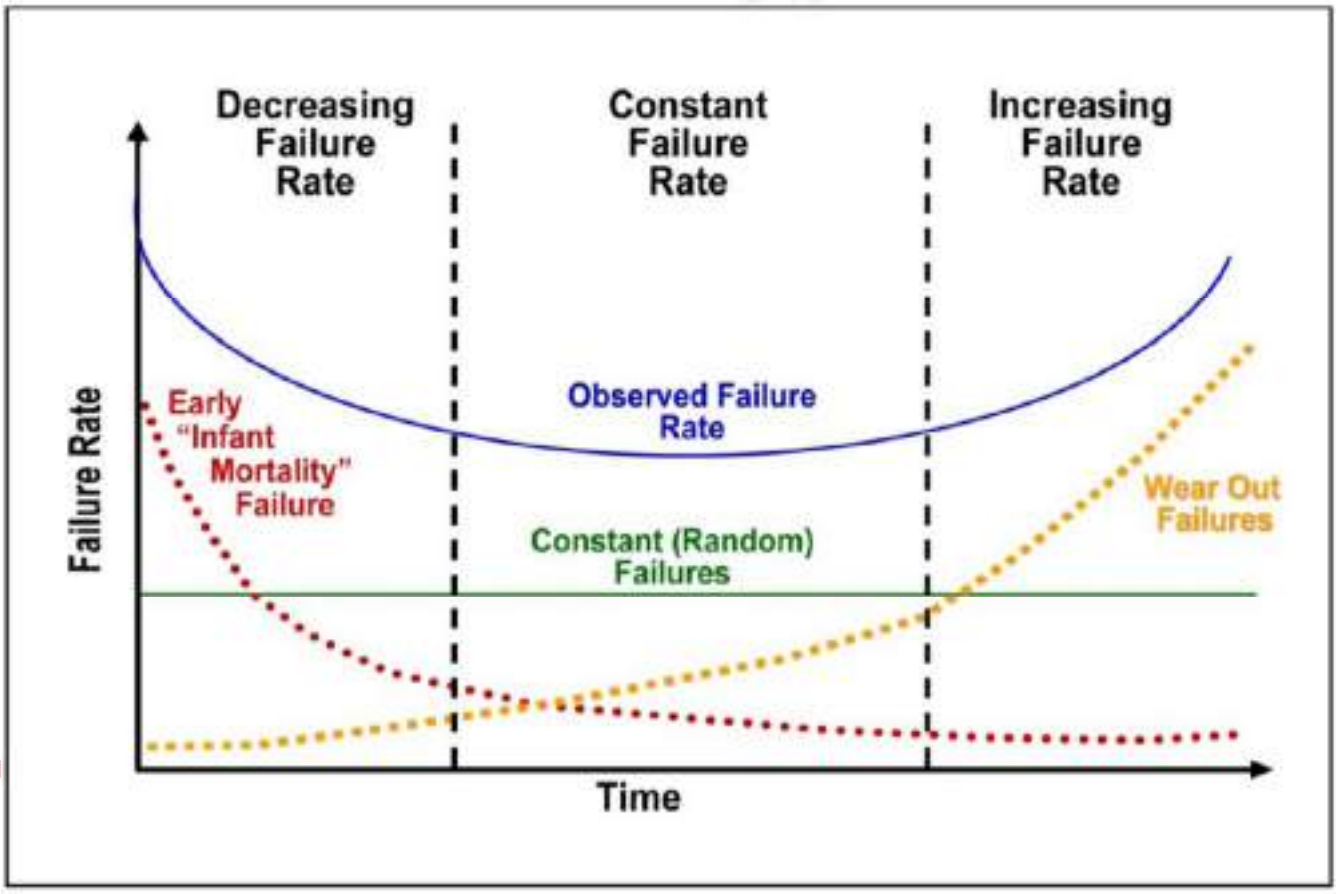
The degree to which a system, subsystem or equipment is in a specified operable and committable state



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

END

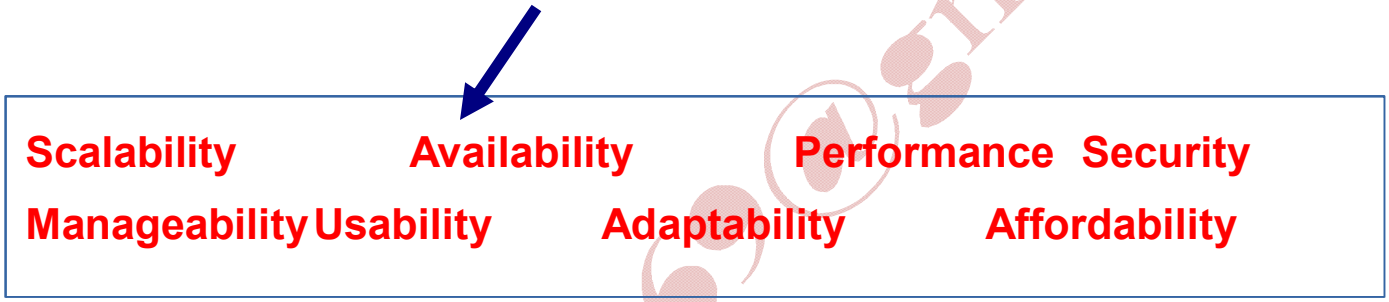
TOPIC 62

QoE—Disaster Recovery

In this module
We shall understand

- A disaster recovery scenario
- Redundancy for recovery
- Allocation model

Amat Victoria Curam (Latin) Victory Loves Preparation



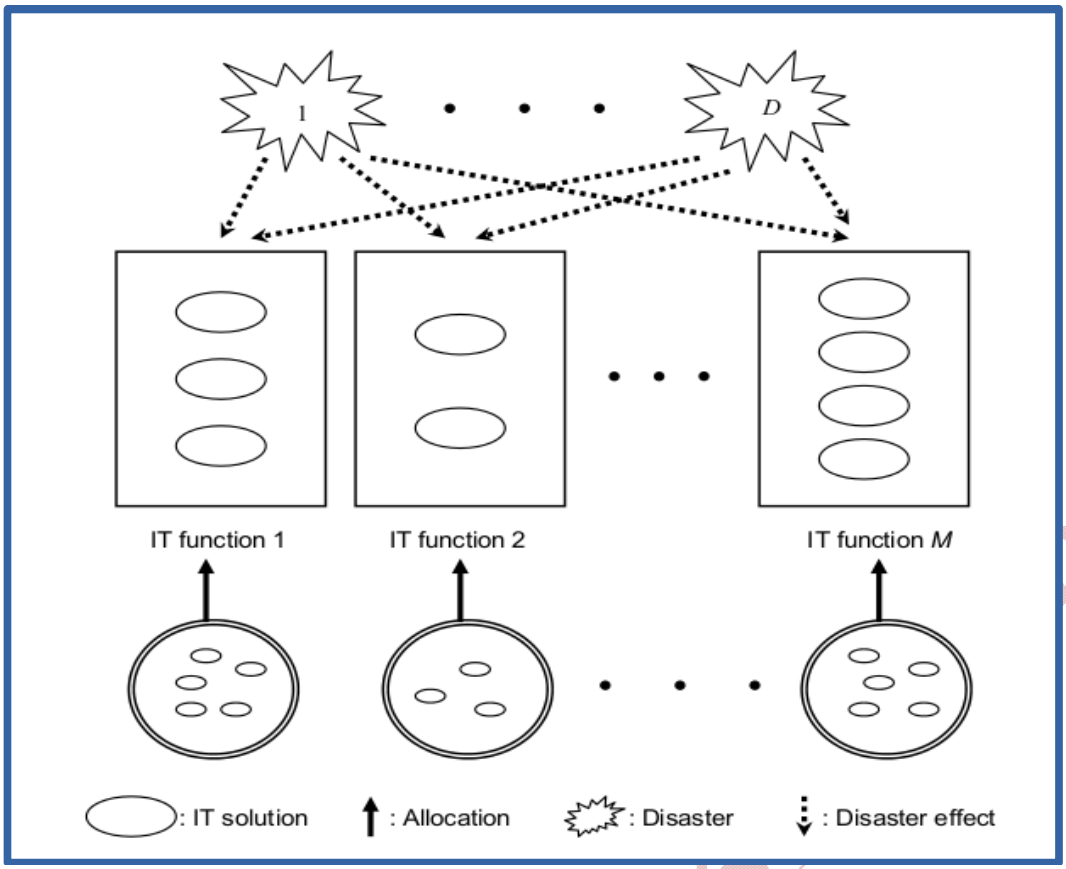
Benjamin B. M. Shao, "Allocating Redundancy to Critical Information Technology Functions for Disaster Recovery," *Proc. 10Th Americas Conference on Information Systems*, Aug. 2004

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 (1 of 6)

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

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 $(= 1, \ldots, n_m)$ 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.

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

subject to

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

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

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

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 (6 of 6)

- *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

END

TOPIC 63

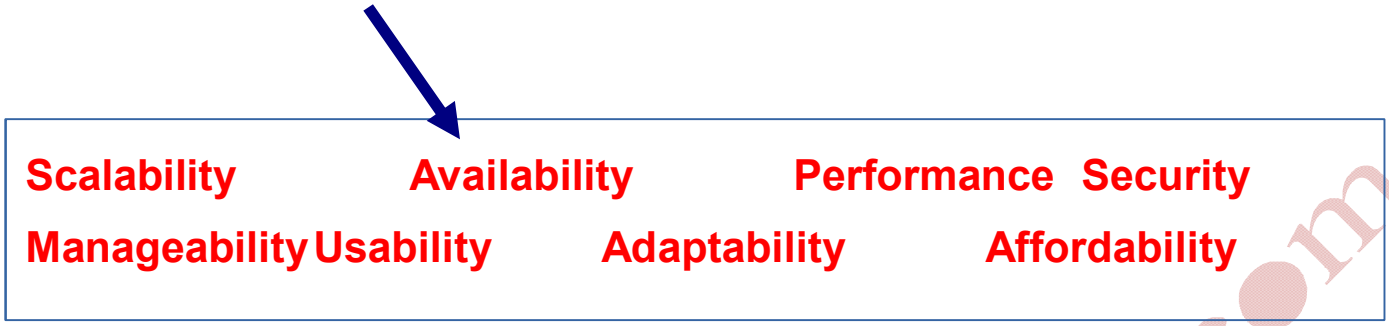
QoE—Specifying Requirements

In this module

We shall understand

- What is availability specification?
- Per year (365 days)
- Per calendar year
- Per spurt

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

END

TOPIC 64

QoE—Five Nines Availability

In this module

We shall understand

- What is 5 9s?
- Impact on costs
- Example

The devil is in the details of availability

Scalability	**Availability**	**Performance**	**Security**
Manageability	**Usability**	**Adaptability**	**Affordability**

5 9s as best-case availability (1 of 4)

- 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

QoE—Five Nines Availability

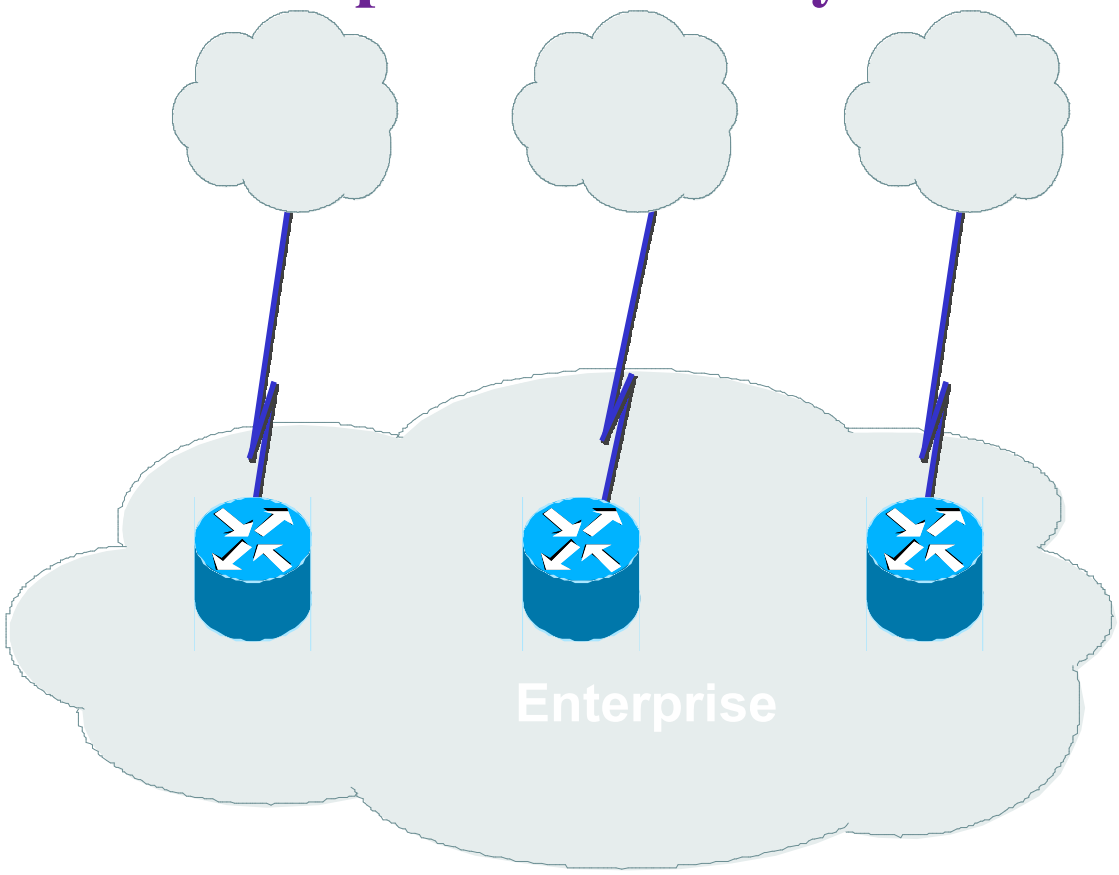
Shifting Impact of 9s on time

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

QoE—Five Nines Availability

99.999% Availability might require

triple redundancy



TOPIC 65

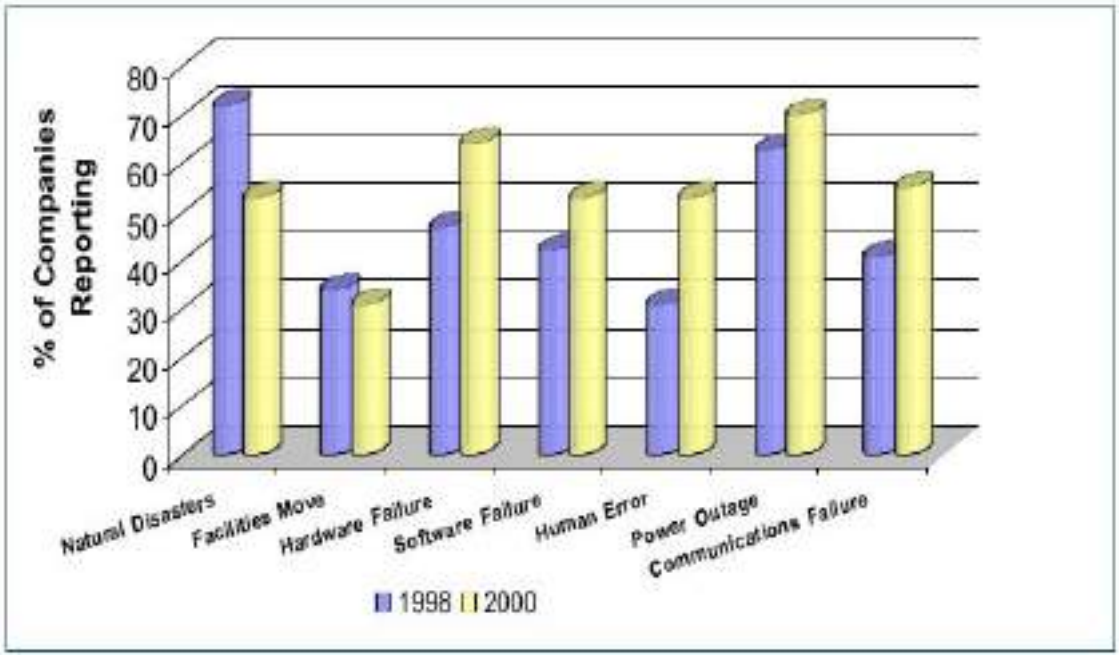
QoE—Cost of downtime

In this module
We shall understand

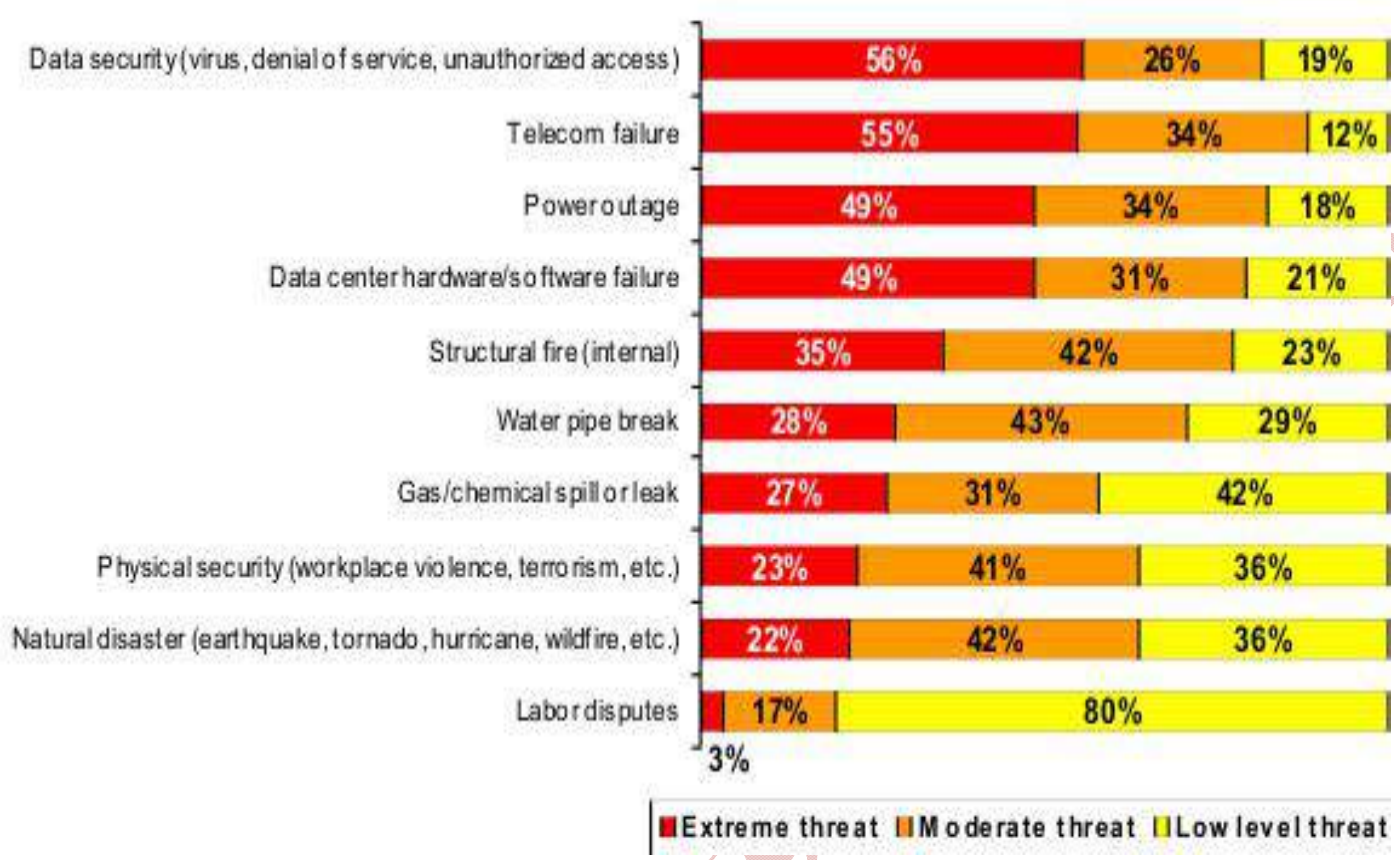
- What is the impact of downtime?
- Step-wise approach

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

Scalability Availability Performance Security
Manageability Usability Adaptability Affordability



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

I. Identify Business Continuity Components
II. Define What You Protect
III. Prioritize Business Functions
IV. Classify Outage Types,
V. 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

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

END

TOPIC 66

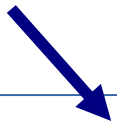
QoE—MTBF AND MTTR

In this module

We shall understand

- What is typical representation of availability?
- Availability of components and service
- Example

Averaging out the availability



Scalability	**Availability**	**Performance**	**Security**
Manageability	**Usability**	**Adaptability**	**Affordability**

Availability as MTBF (1 of 3)

- 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)

Availability as MTBF (2 of 3)

- Typical MTTF value is once per 4000 hrs or 166.7 days
- Typical acceptable MTTR value is one hour

Availability = MTBF/(MTBF + MTTR)

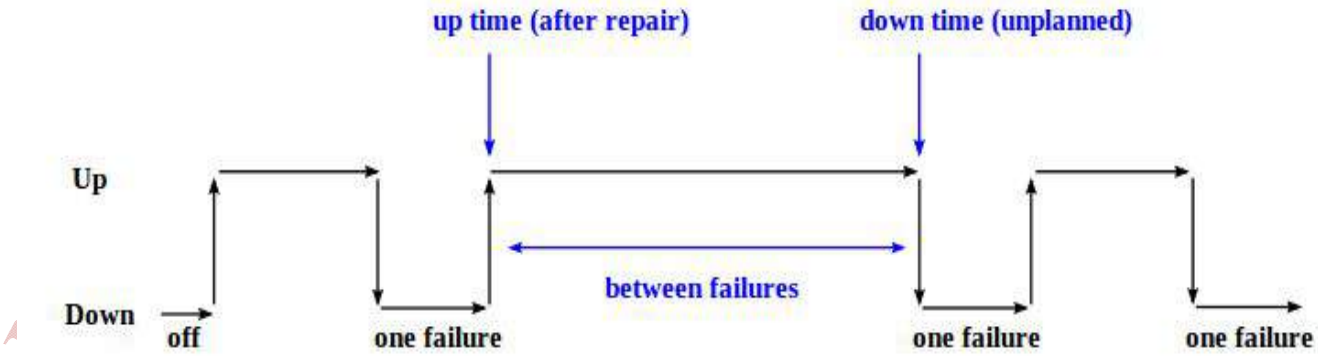
4,000/4,001 = 99.98% availability

Availability as MTBF (3 of 3)

- 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 replaced



Time Between Failures = { down time - up time}

TOPIC 67

QoE—Network Performance

In this module

We shall understand

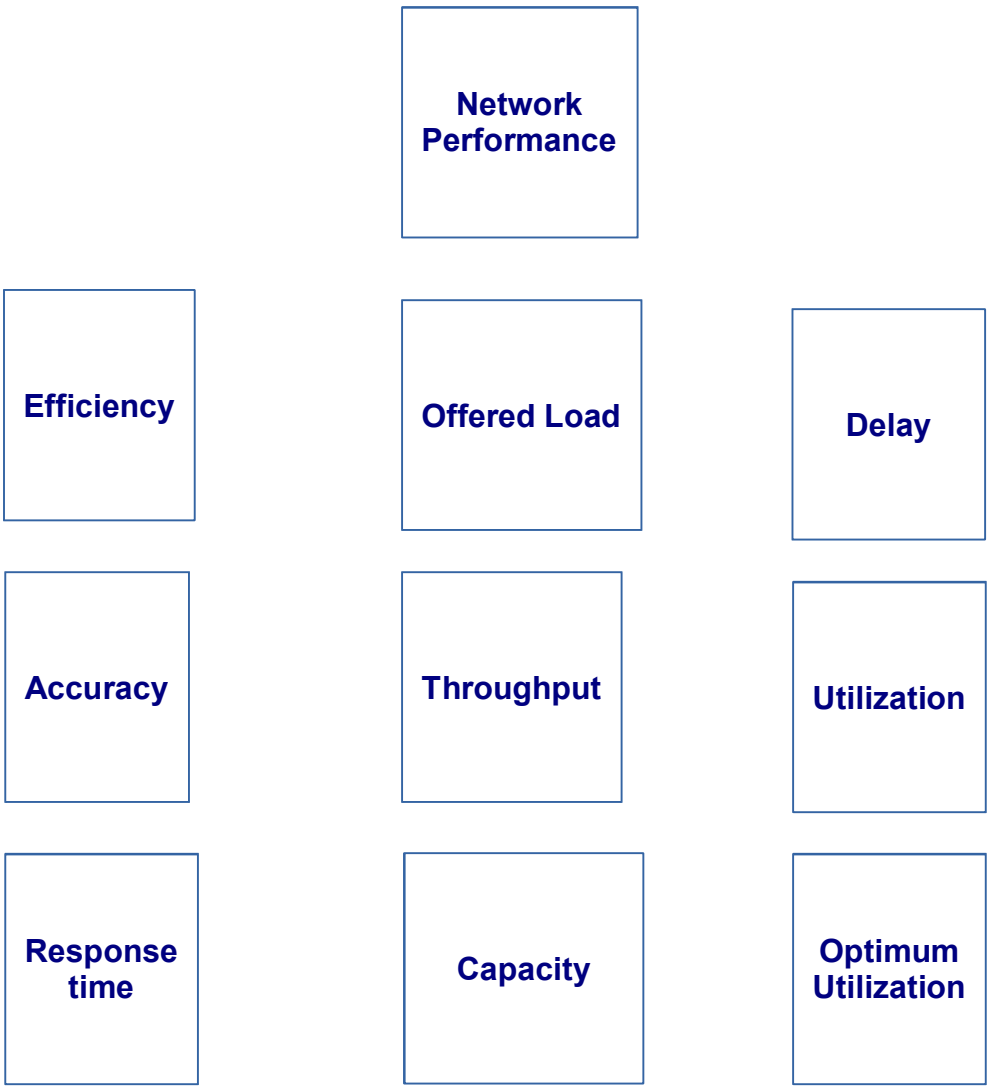
- How to define network performance?
- Performance factors

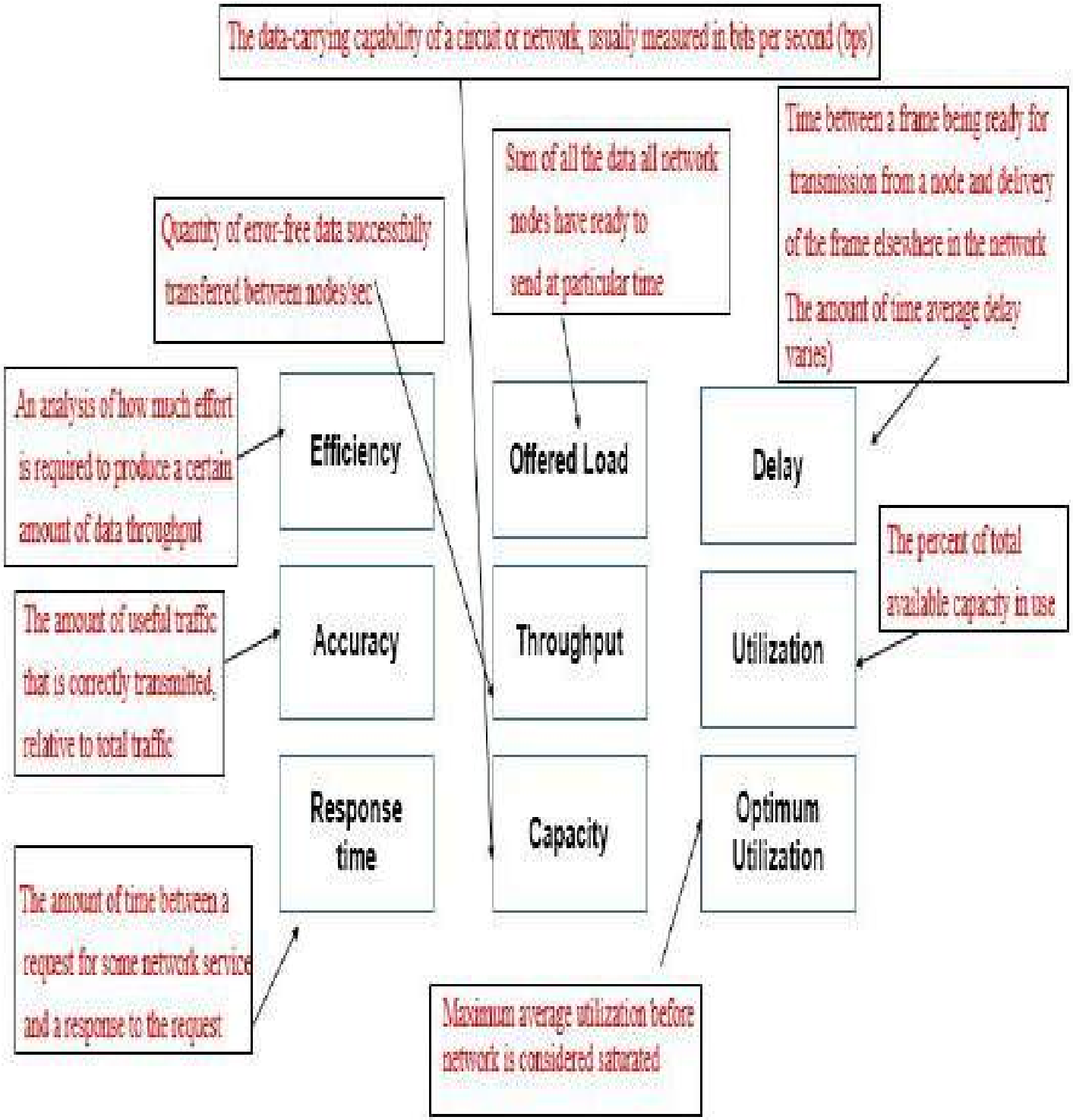
Composite metric that is end-to-end

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

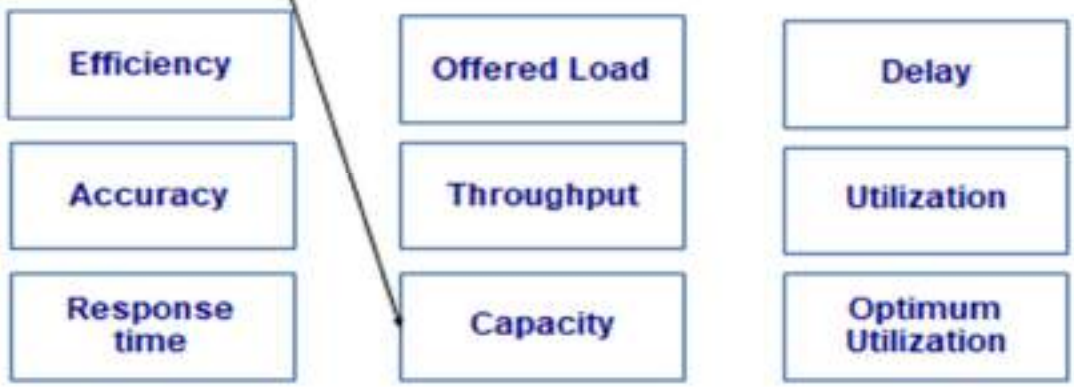
QoE—Network Performance




The data-carrying capability of a circuit or network, usually measured in bits per second (bps)
Quantity of error-free data successfully transferred between nodes/sec
Sum of all the data all network nodes have ready to send at particular time
Time between a frame being ready for transmission from a node and delivery of the frame elsewhere in the network
The amount of time average delay varies)
An analysis of how much effort is required to produce a certain amount of data throughput
Efficiency
Offered Load
Delay
The percent of total available capacity in use
The amount of useful traffic that is correctly transmitted, relative to total traffic
Accuracy
Throughput
Utilization
Response time
Capacity
Optimum Utilization
The amount of time between a request for some network service and a response to the request
Maximum average utilization before network is considered saturated

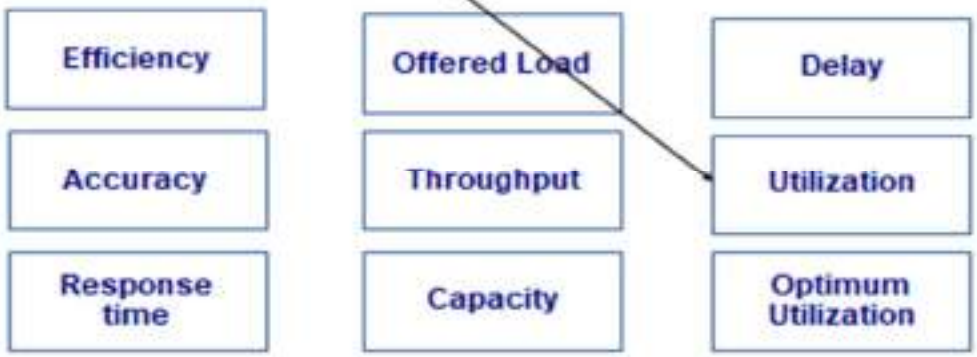
QoE—Network Performance

The data-carrying capability of a circuit or network, usually measured in bits per second (bps)



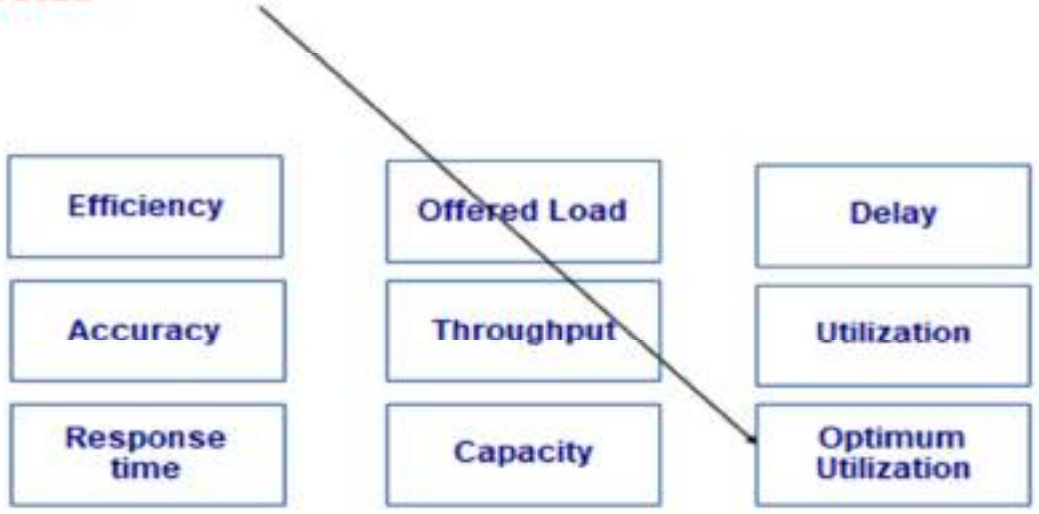
QoE—Network Performance

The percent of total available capacity in use



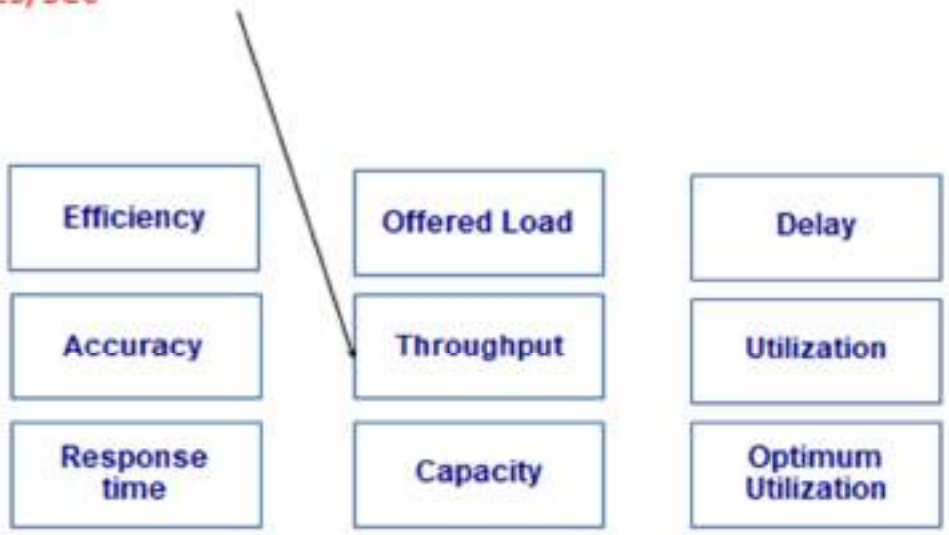
QoE—Network Performance

Maximum average utilization before network is considered saturated



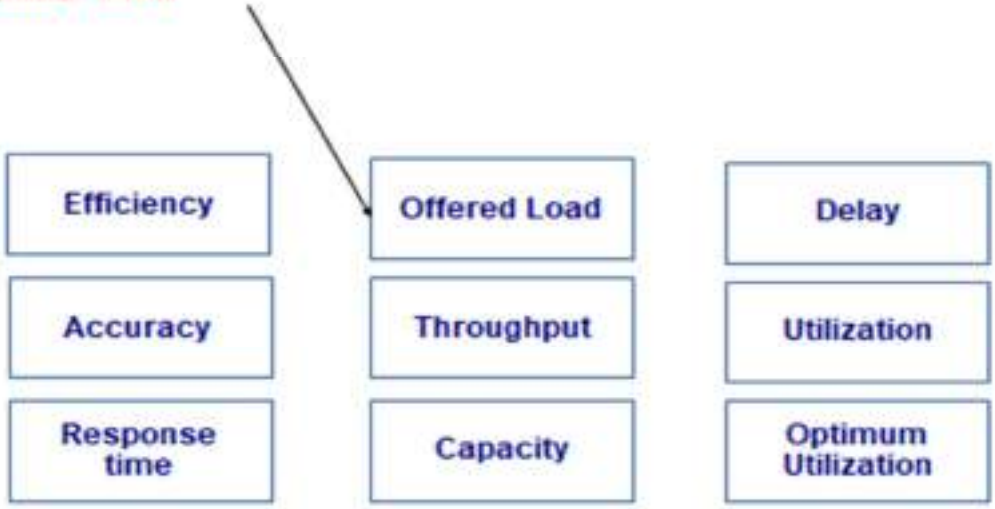
QoE—Network Performance

Quantity of error-free data successfully transferred between nodes/sec



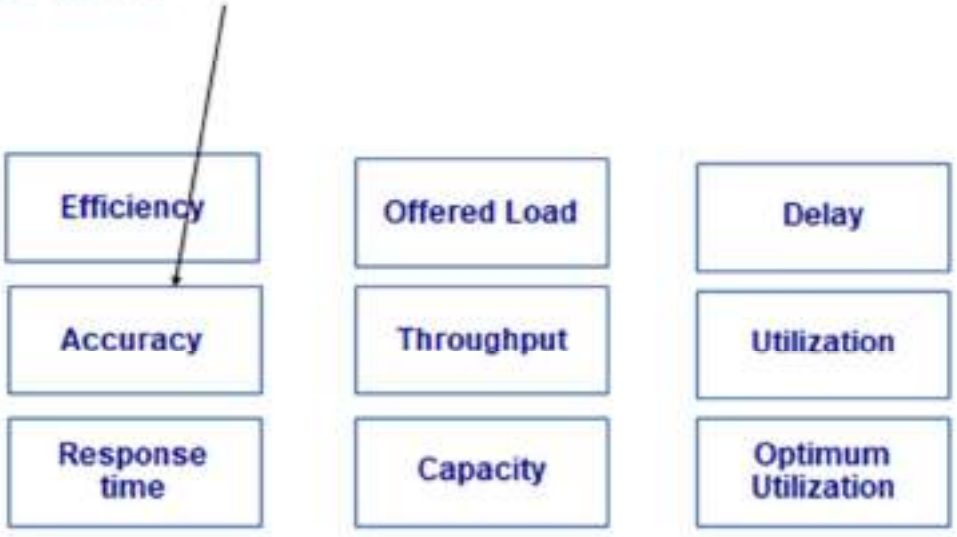
QoE—Network Performance

Sum of all the data all network nodes have ready to send at particular time



QoE—Network Performance

The amount of useful traffic that is correctly transmitted, relative to total traffic



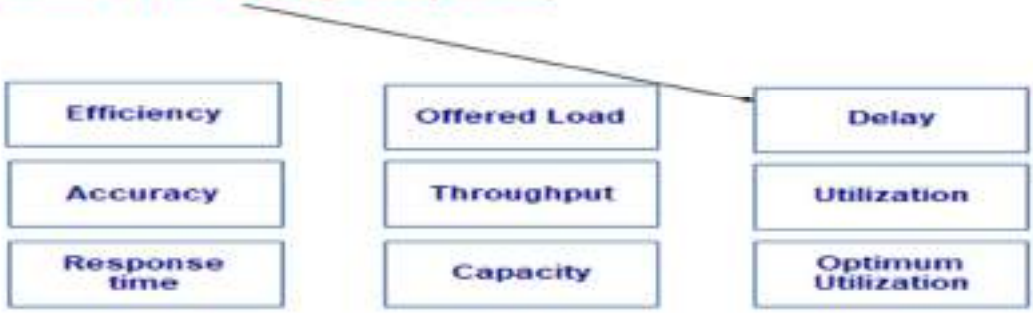
QoE—Network Performance

An analysis of how much effort is required to produce a certain amount of data throughput

QoE—Network Performance

Time between a frame being ready for transmission from a node and delivery of the frame elsewhere in the network

The amount of time average delay varies)



QoE—Network Performance

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

TOPIC 68

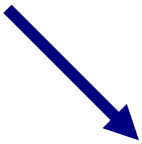
QoE—Optimum Network Utilization

In this module

We shall understand

- What is optimum?
- What is optimum network utilization?
  - At WANs
  - At LANs
  - Typical values

Optimum is "As good as it gets"



Scalability	**Availability**	**Performance**	**Security**
Manageability	**Usability**	**Adaptability**	**Affordability**

Definition of optimum

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

Optimum network utilization (1 of 4)

- How much % of bandwidth capacity in a specific time period?
- Time varying phenomenon
  - Instantaneous, averaged, weighted)
- Both goal & constraint

Optimum network utilization (2 of 4)

- Typical value is 70$%
  - Exceeding this results in performance degradation

Optimum network utilization (3 of 4)

- WAN links utilization is more crucial than LAN
  - Pay per packet
- Compression, caching and concatenation used to reduce WAN utilization

Optimum network utilization (4 of 4)

- LANs are over-budgeted
  - Fast Ethernet)
- Full-duplex vs half duplex switches
- User activity levels
- LANs suffer from exceeding utilization in switch-to-switch

END

TOPIC 69

QoE—Throughput

In this module

We shall understand

- What is throughput?
- Throughput vs offered load

Throughput = Goodput + Badput

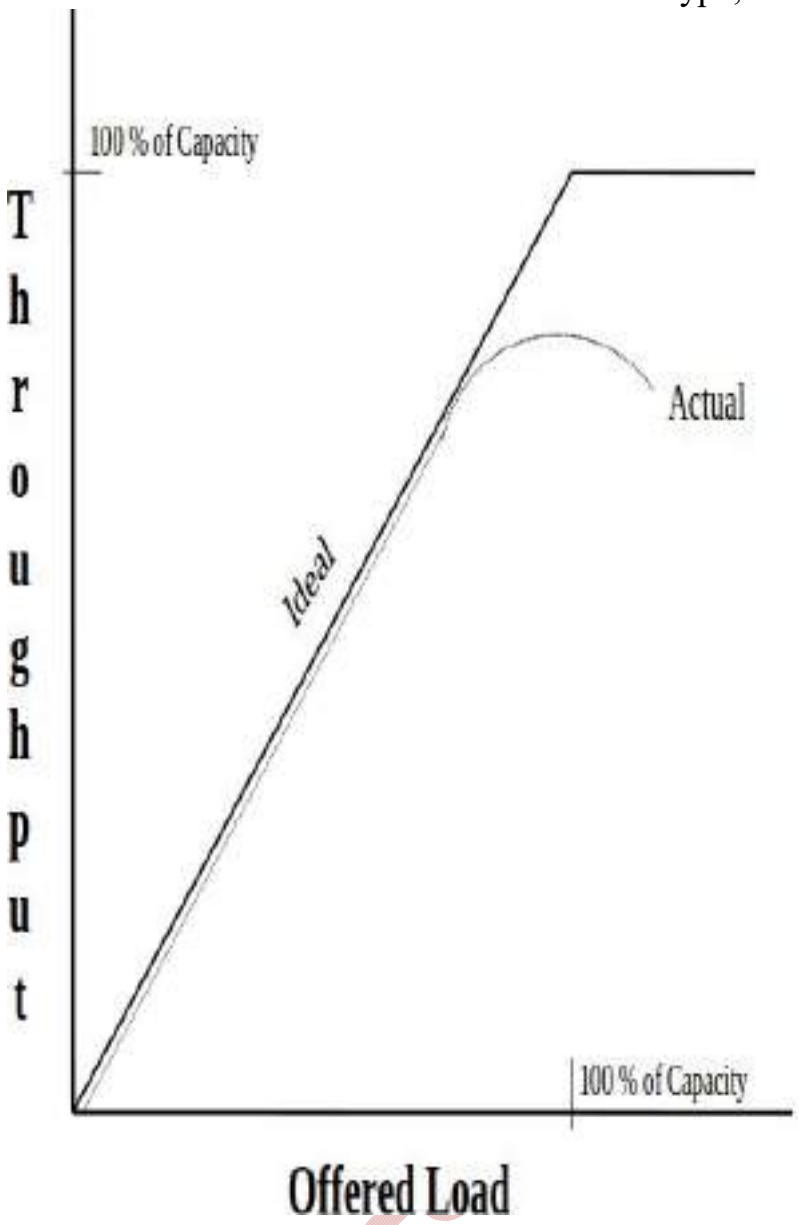
Scalability	**Availability**	**Performance**	**Security**
Manageability	**Usability**	**Adaptability**	**Affordability**

Definition of throughput

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

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

Deviation indicates the limitations of media type, device and network



END

TOPIC 70

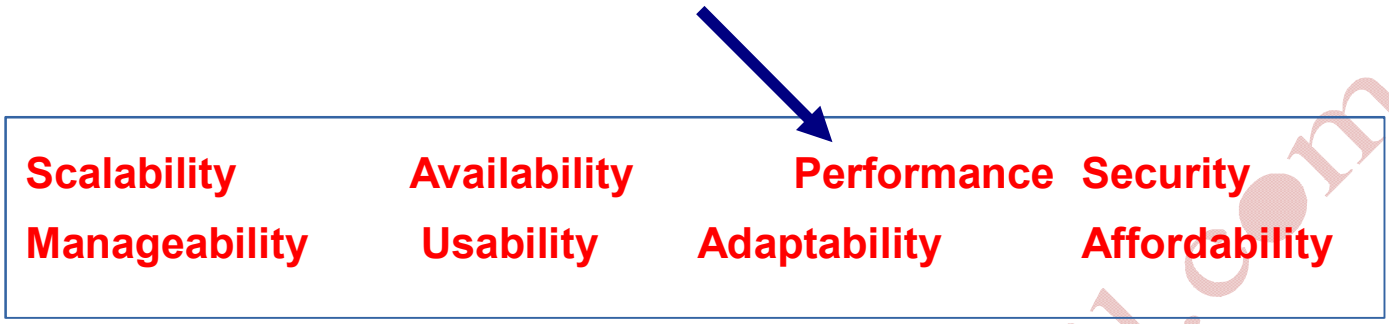
QoE—Throughput of devices

In this module

We shall understand

- What is throughput?
- Throughput vs offered load

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

END

WEEK 6

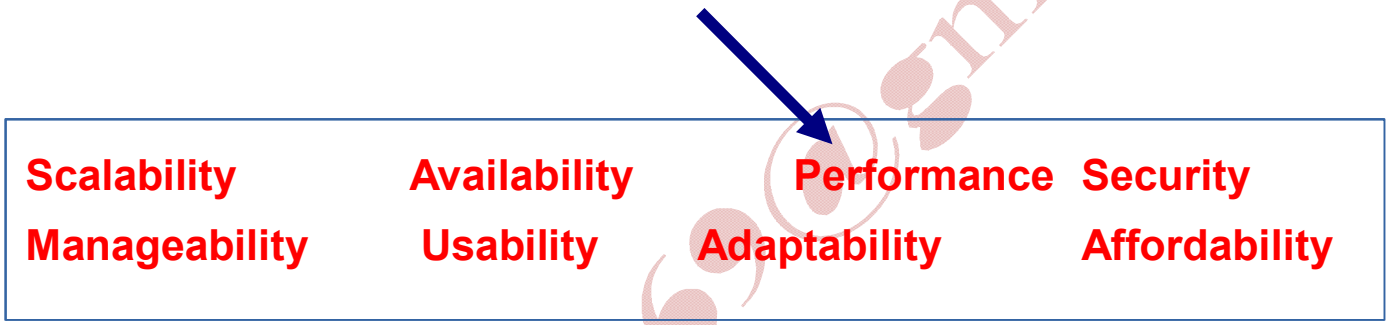
TOPIC 71

QoE—Application Layer Throughput

In this module
We shall understand

- Definition
- Factors affecting goodput
- Mathematical formulation

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 (1 of 3)

- 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
-

Factors affecting goodput (2 of 3)

- Workstation & server performance factors:
  - Disk-access speed
  - Disk-caching size
  - Device driver performance

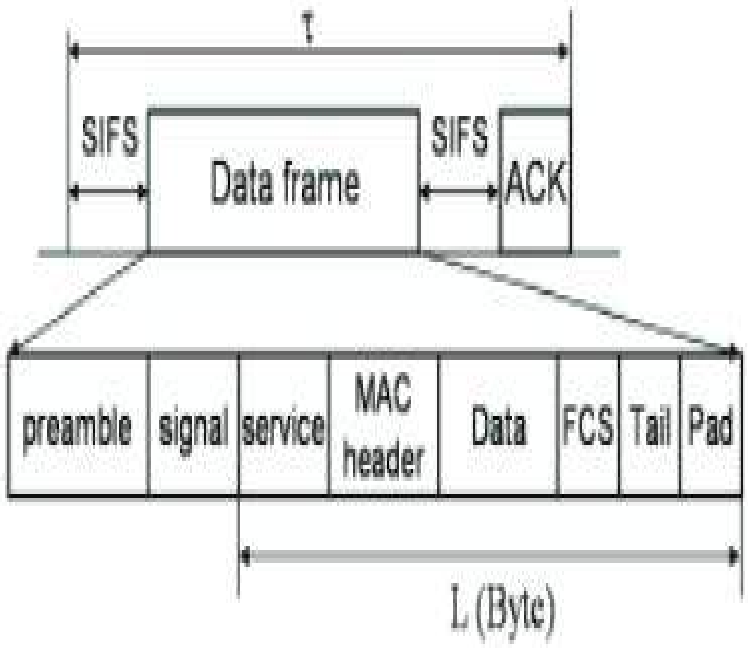
- Computer bus performance (capacity/arbitration)

Factors affecting goodput (3 of 3)

- 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.

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



Application layer goodput

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

Connotations

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

END

TOPIC 72

QoE—Accuracy

In this module

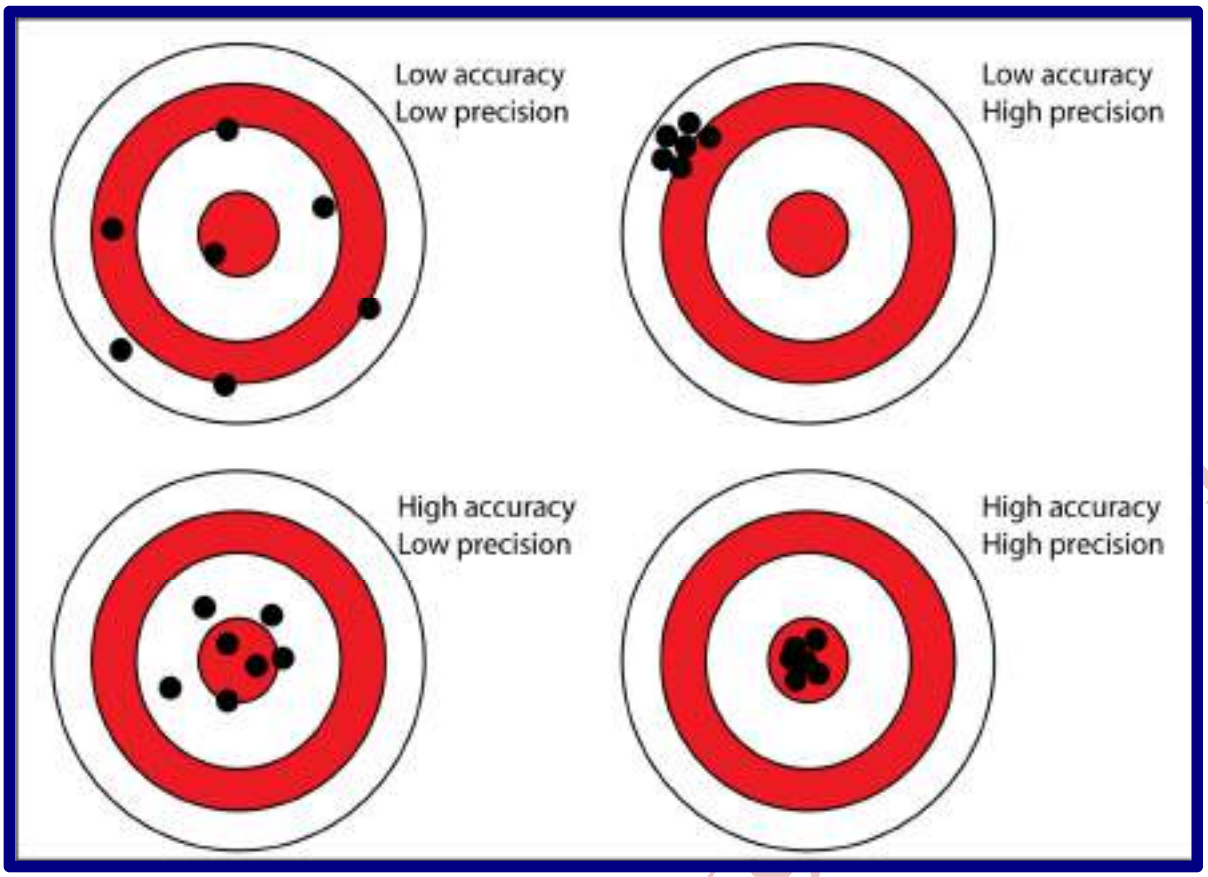
We shall understand

- Definition
- Factors affecting accuracy

Being accurate is not being precise

Scalability	**Availability**	**Performance**	**Security**
Manageability	**Usability**	**Adaptability**	**Affordability**

CS432 Handouts Made by Mahjabeen
mahjabeen97869@gmail.com
contact # 0321 2711298



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 (1 of 3)

- Packet reordering at routers
- Power surges
  - Lightning impulse of 1 s on 10 Mbps link
- Impedance mismatch problems

Factors affecting accuracy (2 of 3)

- 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

Factors affecting accuracy (3 of 3)

- 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

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

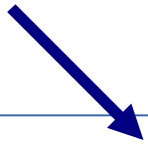
END

TOPIC 73

QoE—Efficiency
In this module
We shall understand

- Definition
- Factors affecting efficiency
- Throughput vs efficiency
- Average efficiency

Boiling water analogy



Scalability	**Availability**	**Performance**	**Security**
Manageability	**Usability**	**Adaptability**	**Affordability**

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

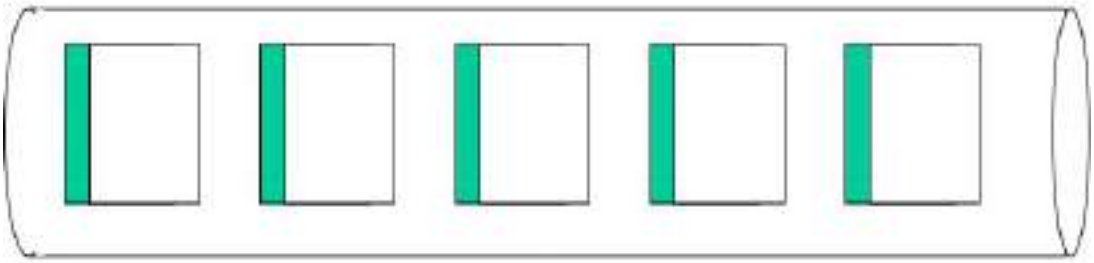
Factors affecting efficiency (1 of 3)

- Access protocols
  - high number of users showing activity
  - Ethernet not efficient at high collision rates

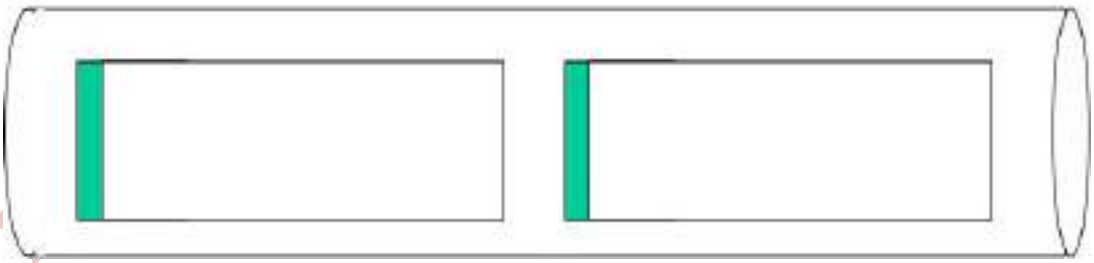
Factors affecting efficiency (2 of 3)

- 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 enqueued in router
- **Factors affecting efficiency (3 of 3)**
-

Small Frames (Less Efficient)

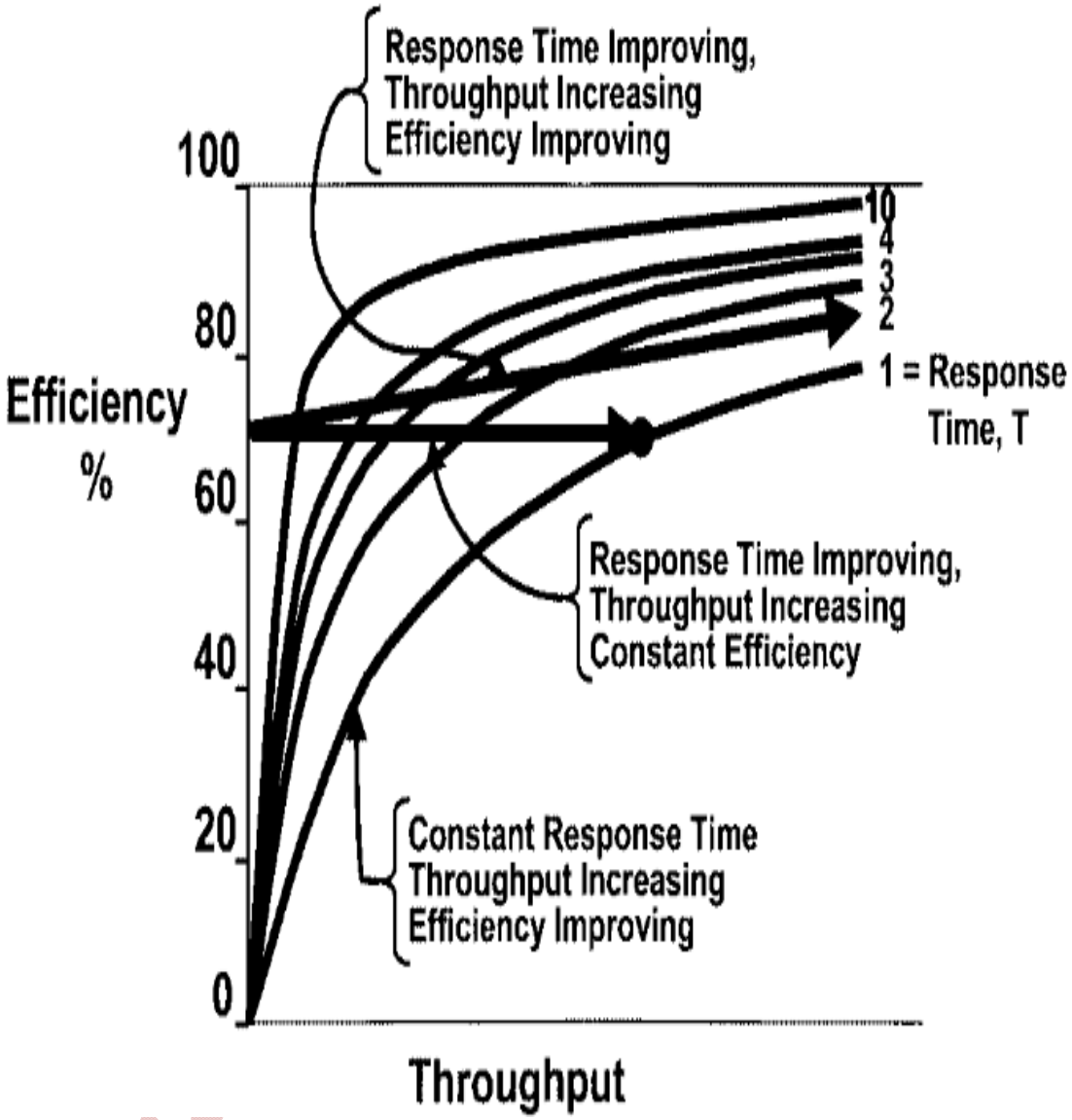


Large Frames (More Efficient)



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

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

- 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

END

TOPIC 74

QoE—Delay and Jitter

In this module
We shall understand

- Definition of delay
- Definition of delay variation

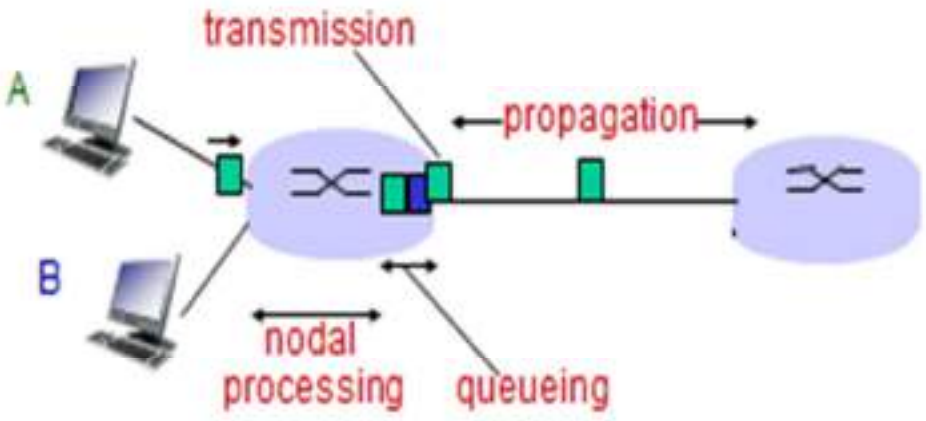
Applications might forgive delay but not jitter

Delay

- Voice and video applications (especially interactive) demand minimum delay
- Other applications such as Telnet remote echo need timed performance

QoE—Delay and Jitter

Sources of packet delay

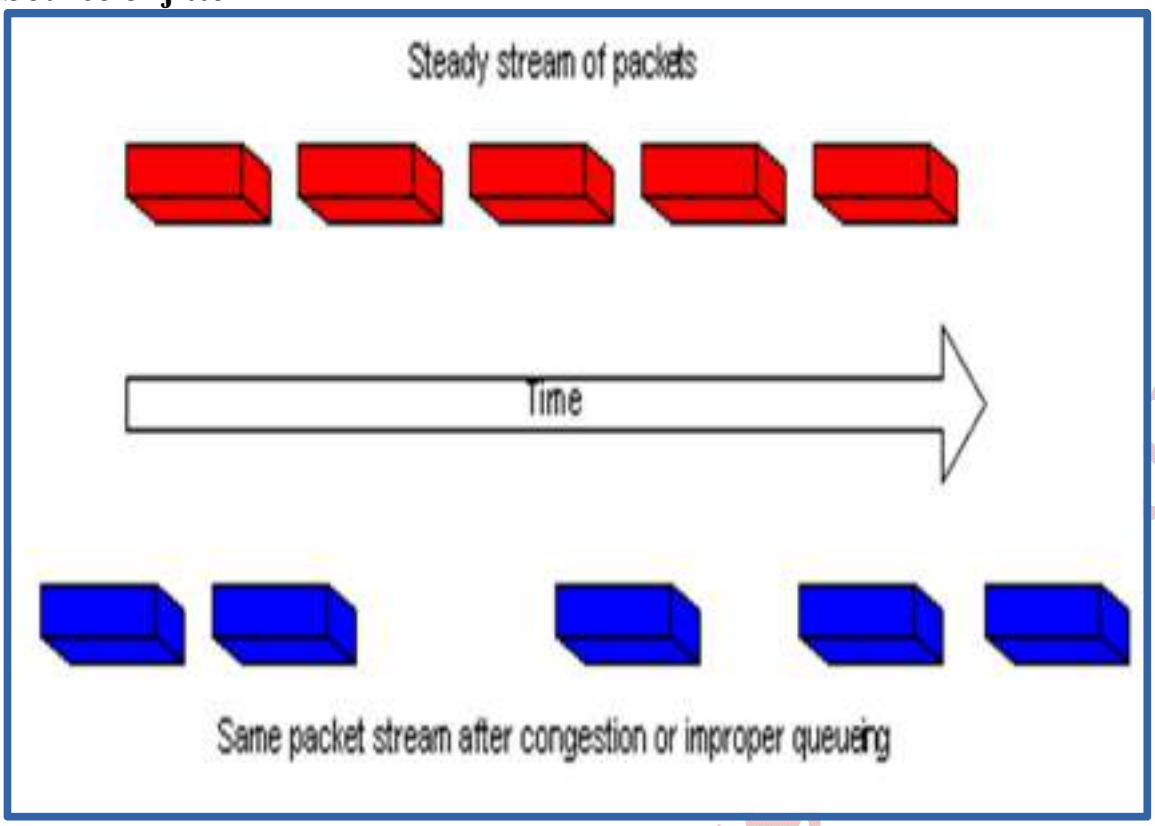


$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

Delay variation (jitter)

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

Source of jitter



END

TOPIC 75

QoE—Causes of Delay

In this module

We shall understand

- Causes of delay
- Effect of utilization on delay

It is the small factors that matter the most

Scalability	**Availability**	**Performance**	**Security**
Manageability	**Usability**	**Adaptability**	**Affordability**

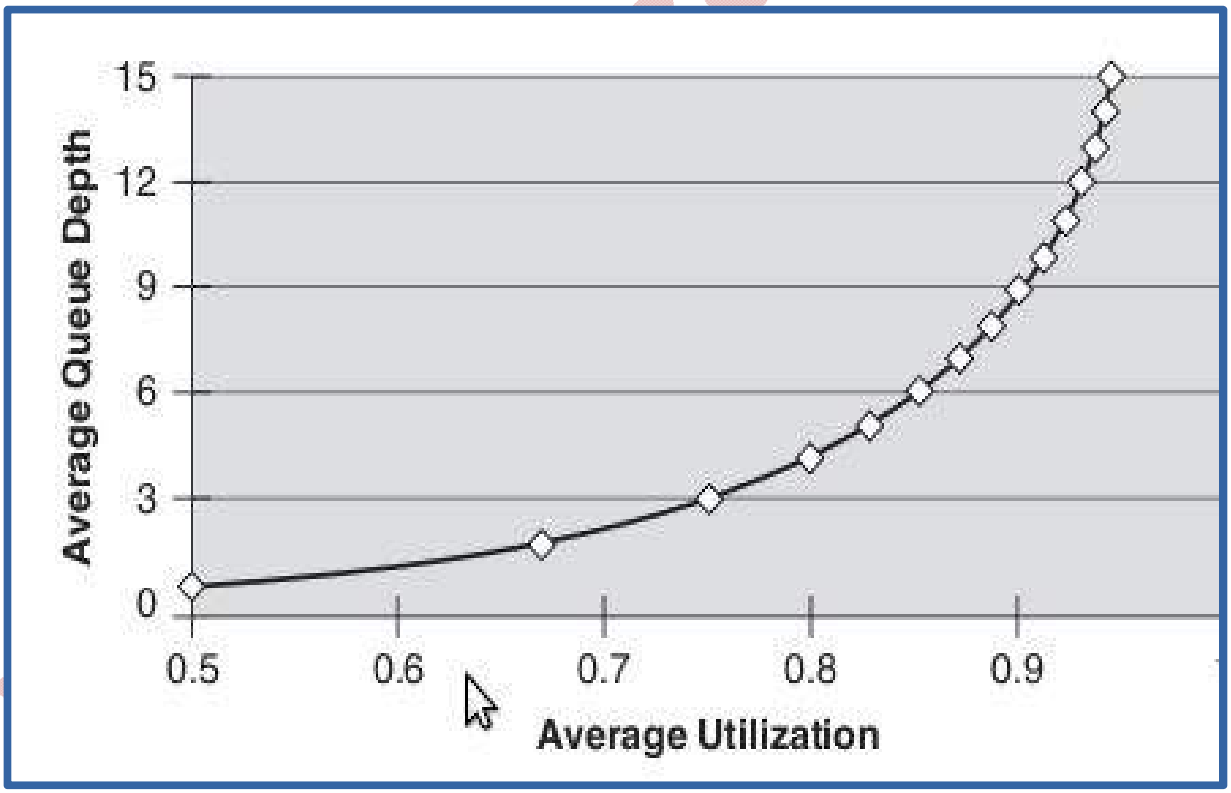
Causes of Delay

- 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
- Queuing delay
  - Dependent upon utilization
-

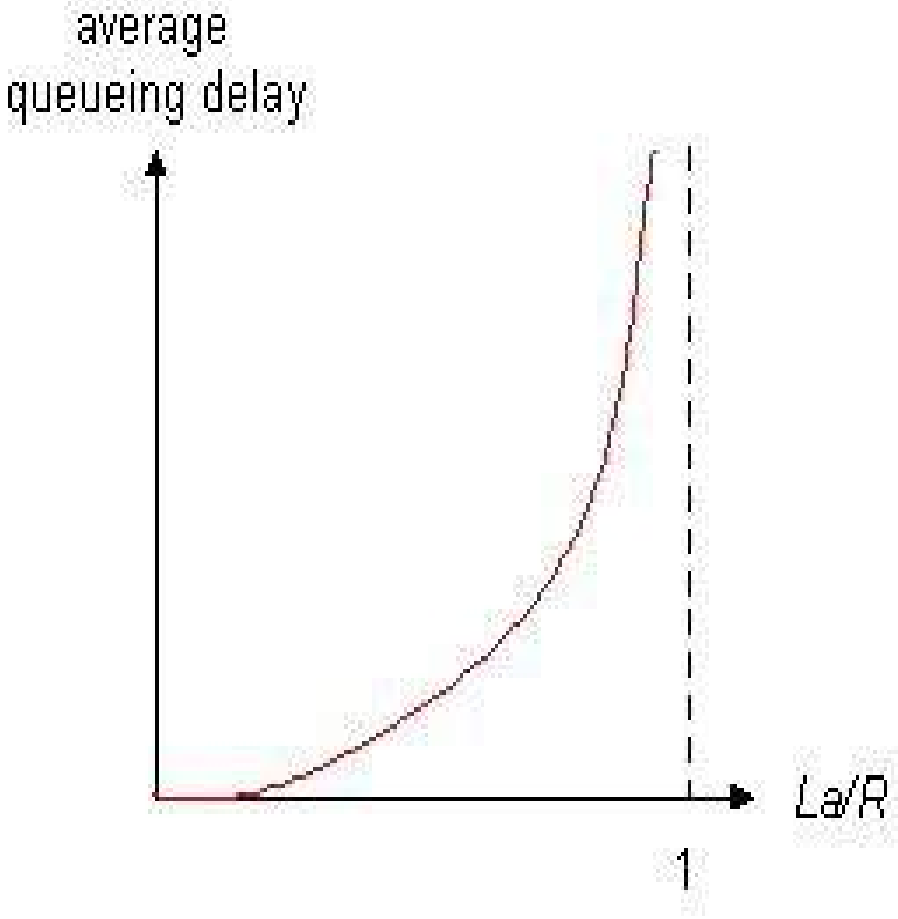
Formula

Queue depth = Utilization/(1 – Utilization)

Queue Depth vs Utilization



Implications of queuing delay



END

TOPIC 76

QoE—Delay variation

In this module

We shall understand

- Definition of delay variation (jitter)
- Types
- Measurement of jitter

All animals are equal, but some animals are more equal than others (George Orwell)

Scalability	**Availability**	**Performance**	**Security**
Manageability	**Usability**	**Adaptability**	**Affordability**

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 low

Delay variation

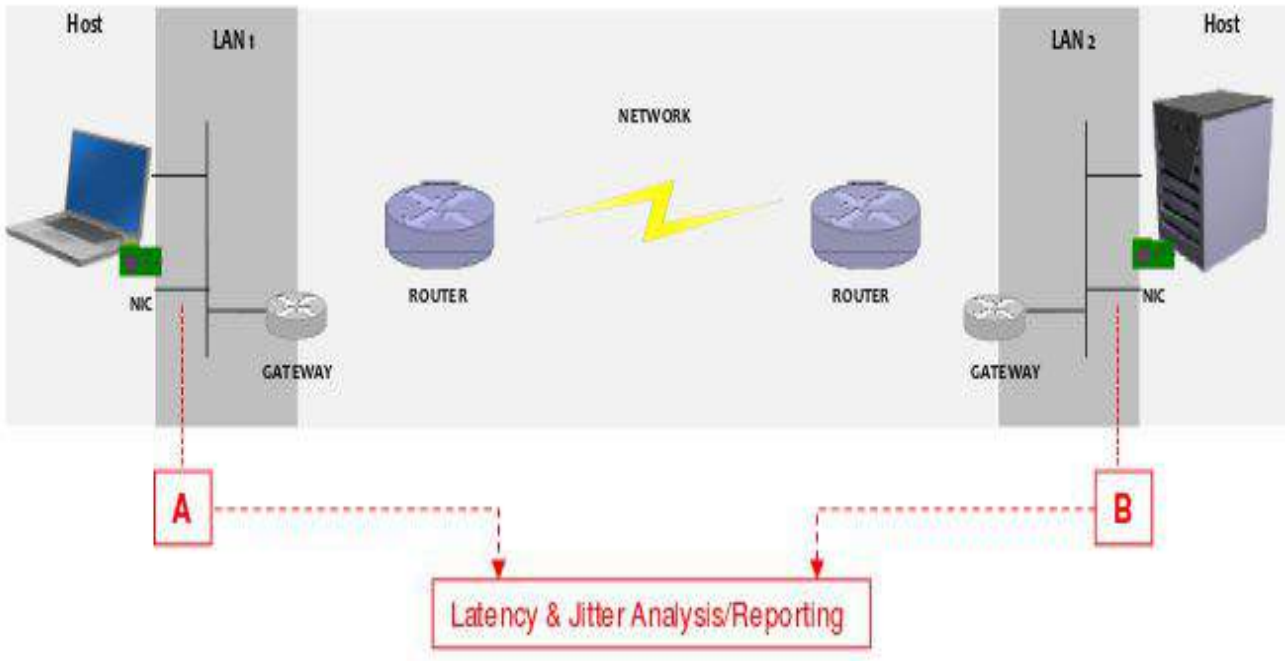
- Concept of jitter buffer to smoothen out the jitter
- Variations on the input side are smaller than the buffer
- Acceptable variation is 1-2% of the delay

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	$L_n = TB_n - TA_n$	$L_n - L_{n-1}$

END

TOPIC 77

QoE—Response Time

In this module

We shall understand

- Definition of response time
- Measuring points locations
- Mathematical form

Response time is relative phenomenon

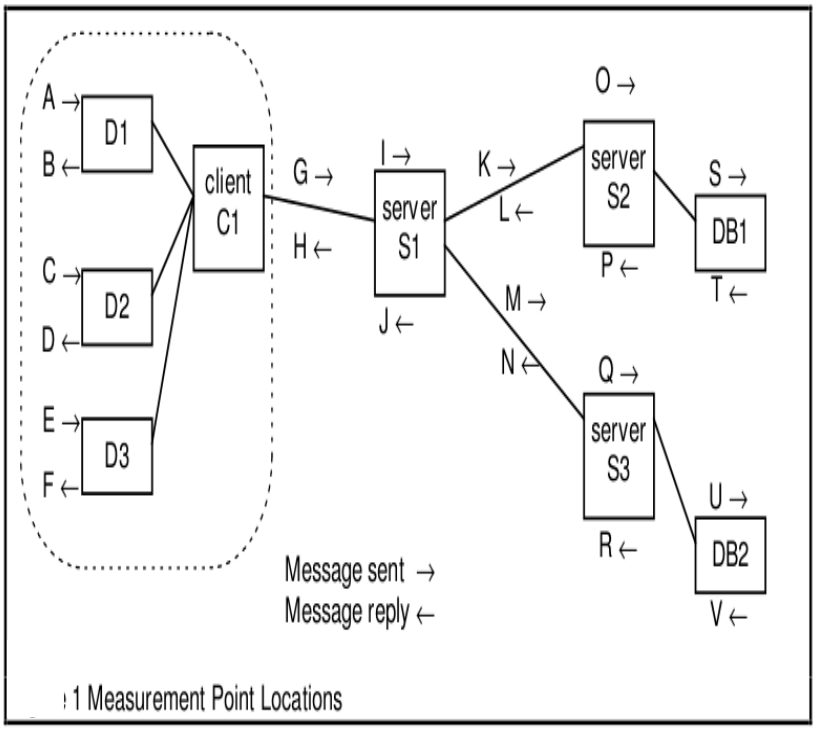
Scalability	**Availability**	**Performance**	**Security**
Manageability	**Usability**	**Adaptability**	**Affordability**

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.

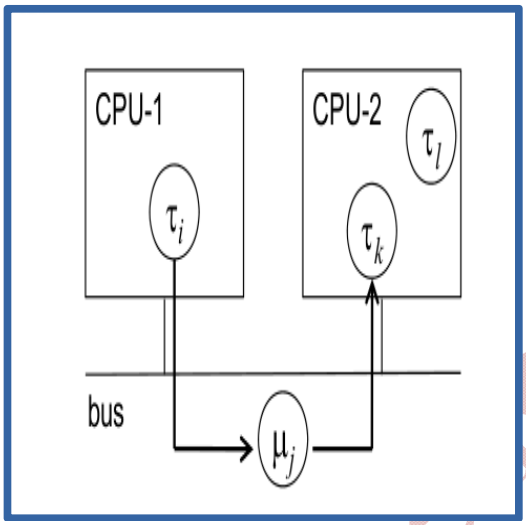


1 Measurement Point Locations

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

Ceiling function represents maximum number of
pre-emptions by higher priority

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

R_i: 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

END

TOPIC 78

QoE—Security

In this module
We shall understand

- What is security?
- Implementation

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

Tradeoff

- Security and deployment
- Operational ease
- Too much secure: bothersome inviting workarounds
- Thumb rule: cost of security implementation << cost of recovering from security lapse

END

TOPIC 79

QoE—Identifying Network Assets

In this module

We shall understand

- What are network assets?
- How to identify them?

Know thy self

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

END

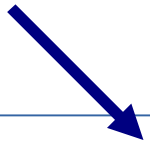
TOPIC 80

QoE—Reconnaissance Attacks

In this module
We shall understand

- What is reconnaissance?
- How to measure and quantify recy attack?

Prevention is better than cure



Scalability	**Availability**	**Performance**	**Security**
Manageability	**Usability**	**Adaptability**	**Affordability**

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

CS432 Handouts Made by Mahjabeen
mahjabeen97869@gmail.com
contact # 0321 2711298

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

END

TOPIC 81

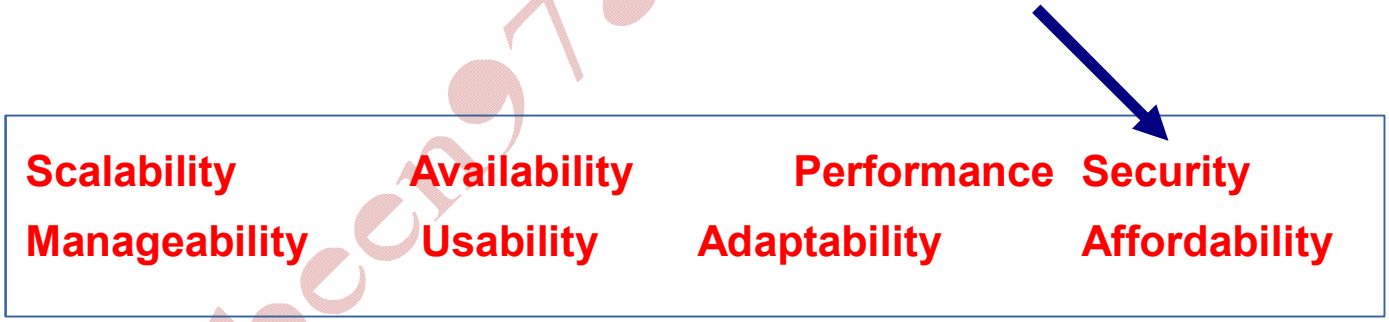
QoE—Security Requirements

In this module

We shall understand

- How to quantify security requirements?
- What is CVSS?
- How to use it?

Comprehensive planning is half success

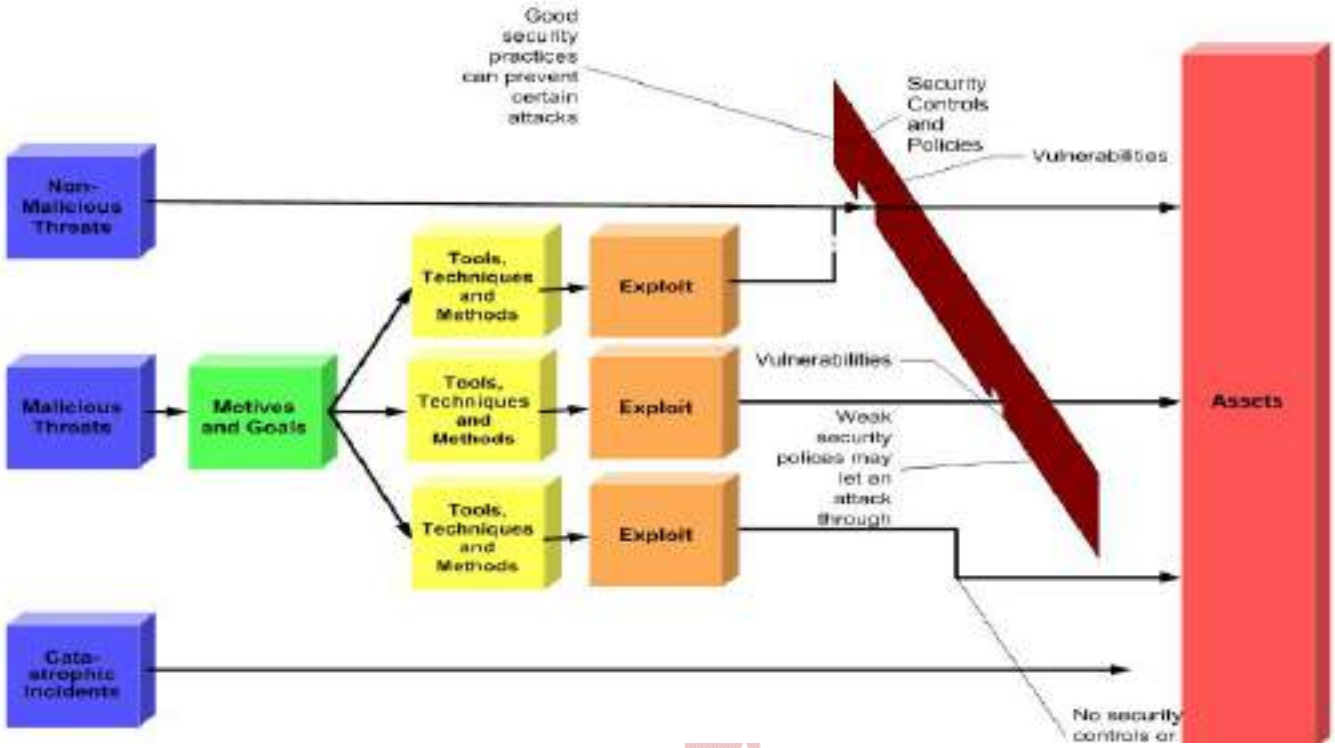


Definition

- 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

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

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

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

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

TOPIC 82

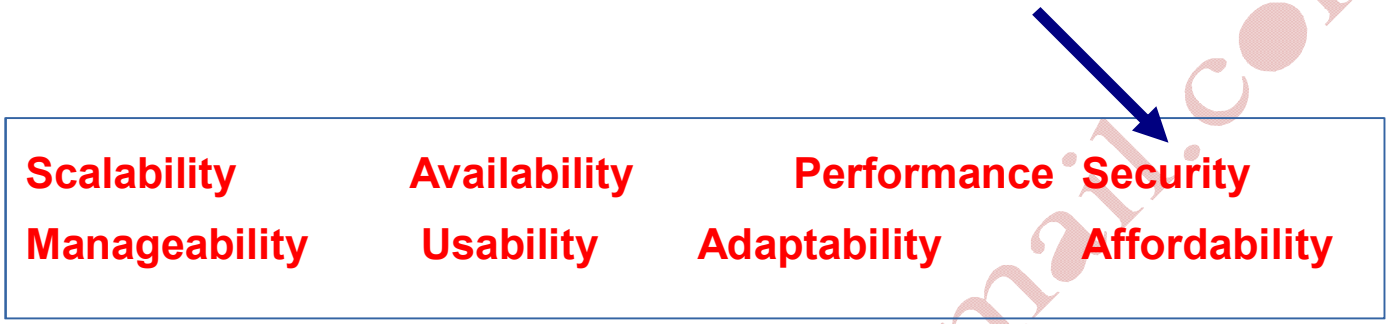
QoE—Manageability

In this module

We shall understand

- What is manageability?
- Manageability metric

SMART goals - Specific, Measurable, Achievable, Realistic and Timely



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

$$Manageability = \frac{TotalTime_{\text{eval}}}{\sum_{i=1}^{n} Weight_i \times Time_i \times Steps_i}$$

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 (1 of 3)

- Manageability is reduced proportionally to how long the management tasks take
- And to how many atomic steps are involved in each such task

Commentary (2 of 3)

- 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

Commentary (3 of 3)

- 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

END

TOPIC 83

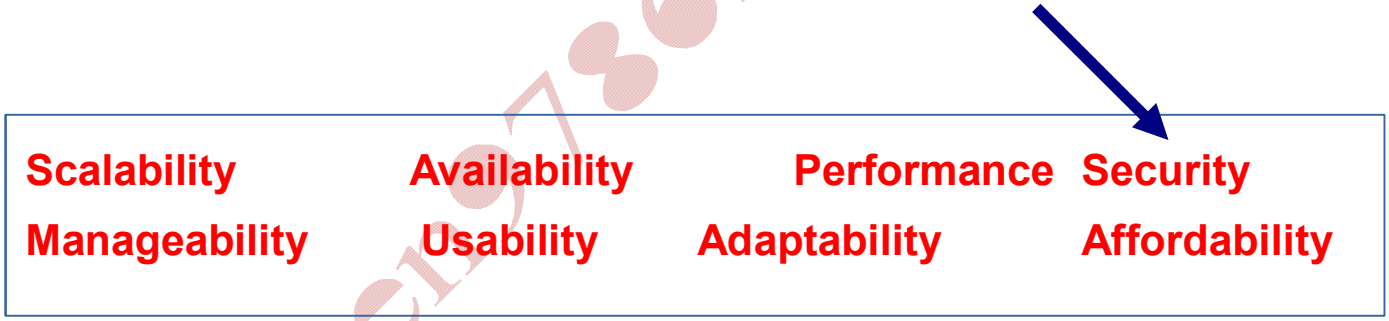
QoE—DoS Attack

In this module

We shall understand

- A simple DoS attack
- Counter strategy
- Analysis

DoS attacker = uncouth talkative person



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.

- Attack type: TCP SYN flooding DoS attacks
  - n packets are used for attack
- Counter: Random drop queue 'Q'

Q = queue depth

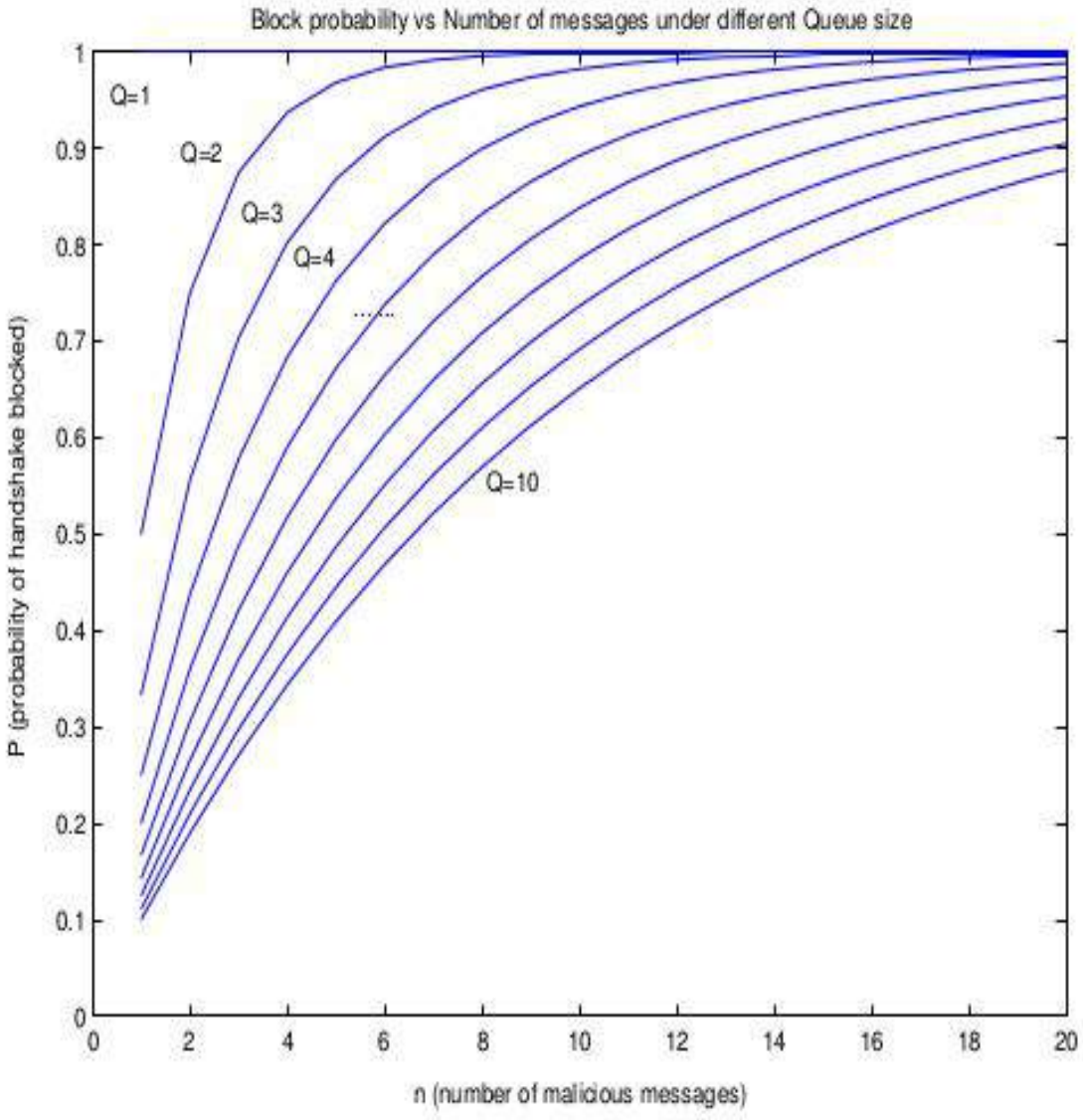
Attack success probability

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

Attack failure probability

- 1-P

A Simple Attack Analysis



END

WEEK 7

TOPIC 84

Making Network Design Tradeoffs

In this module
We shall understand

- Why is tradeoff necessary?
- Making tradeoff
- A usecase

Holistic solution requires a fine balance

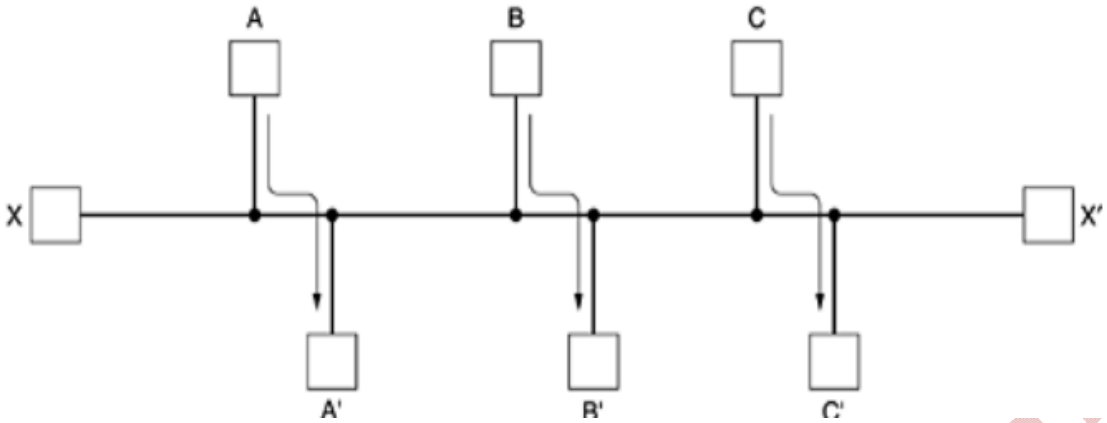
Scalability	**Availability**	**Performance**	**Security**
Manageability	**Usability**	**Adaptability**	**Affordability**

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

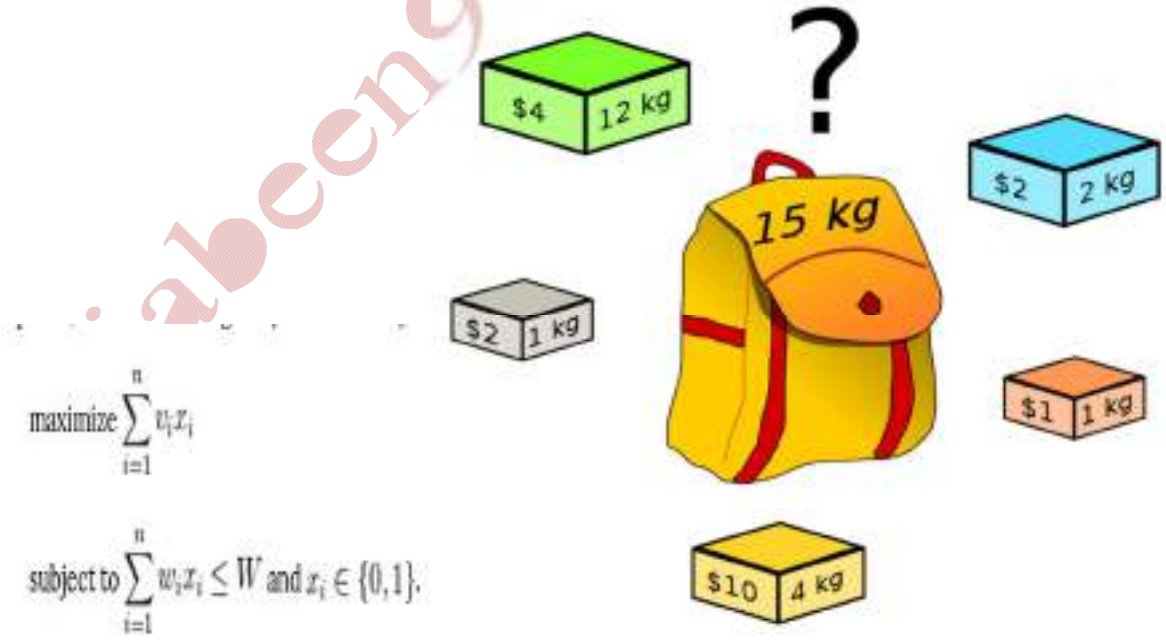


Tradeoff at Network Level

- 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!



Making Network Design Tradeoffs

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)	

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

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

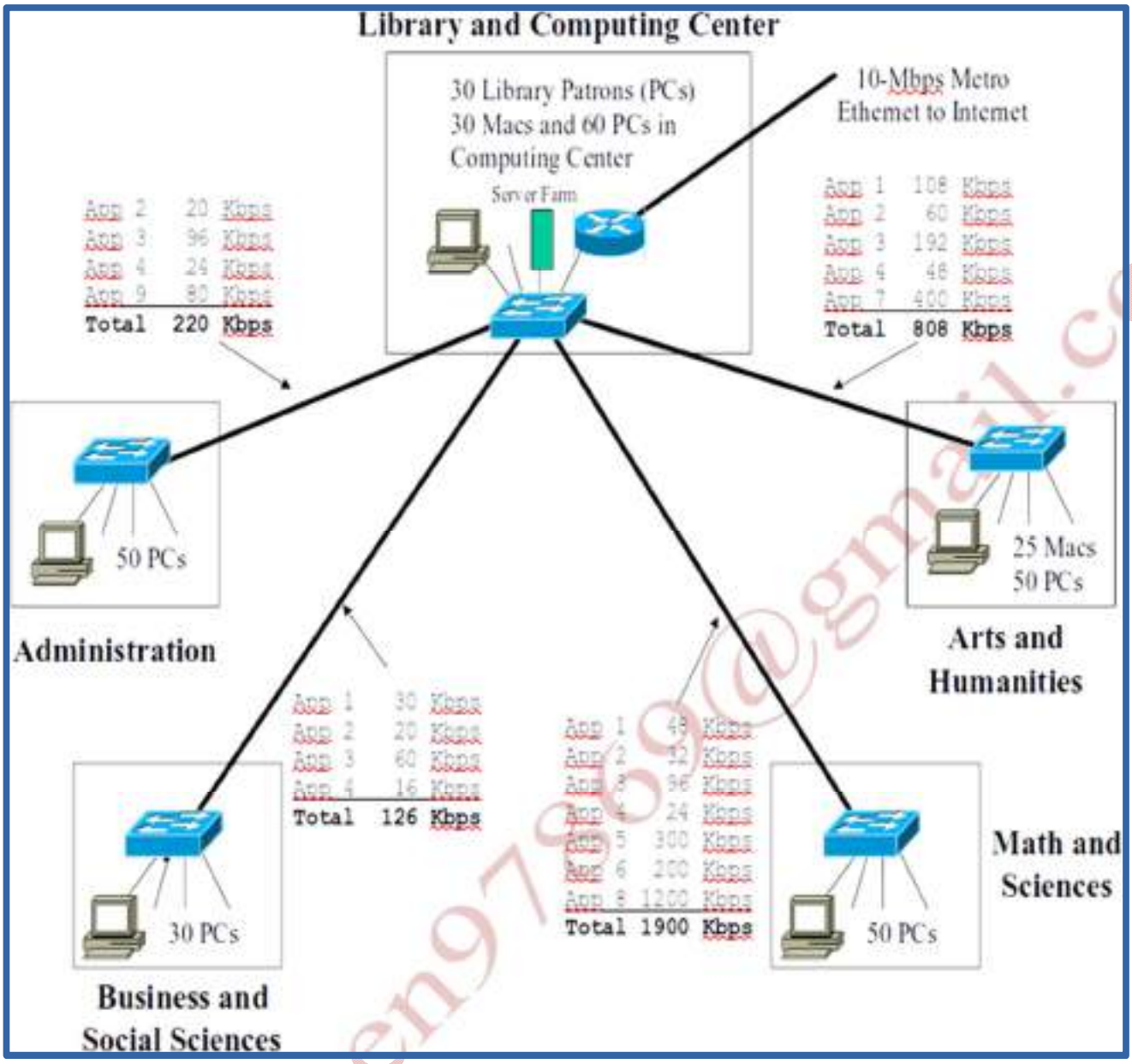
END

TOPIC 85

Problem Set 1

In this module
We shall understand

- Intertwined role of users/network size and topology
- Some insightful questions
- Effect of routing protocols



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?

END

TOPIC 86

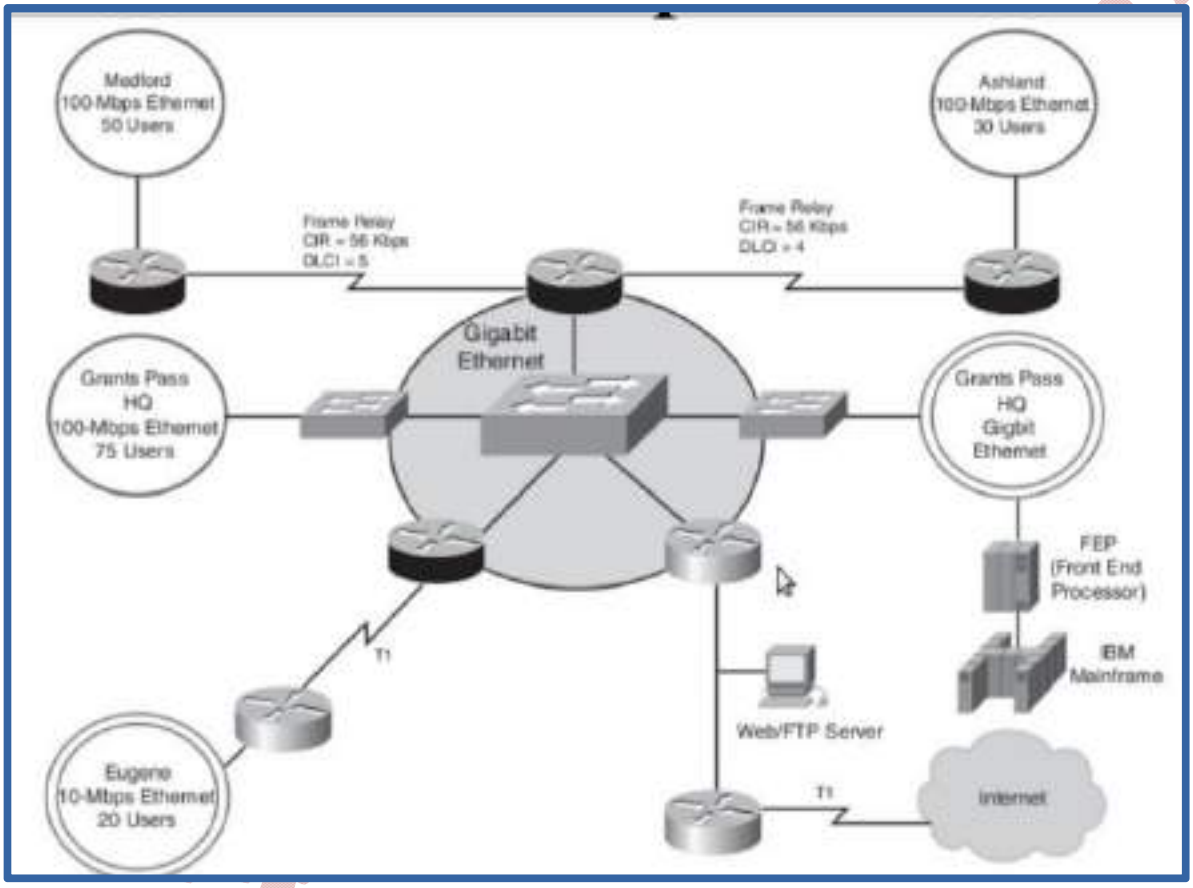
FINAL TERM

Problem Set 2

In this module

We shall understand

- Effect of deployment
- Effect of protocol behaviour
- Queuing behaviour
- Considerations for network design



Effects of Deployment/Protocol Behaviour

1. Where is the data centre?
2. What is the data rate available for users of Eugene?
3. What is the maximum Internet speed available to the users? 100 mbps ethernet 75 user
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 Behaviour

1. A CISCO switch has 20 users (clients ans 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.

END

TOPIC 87

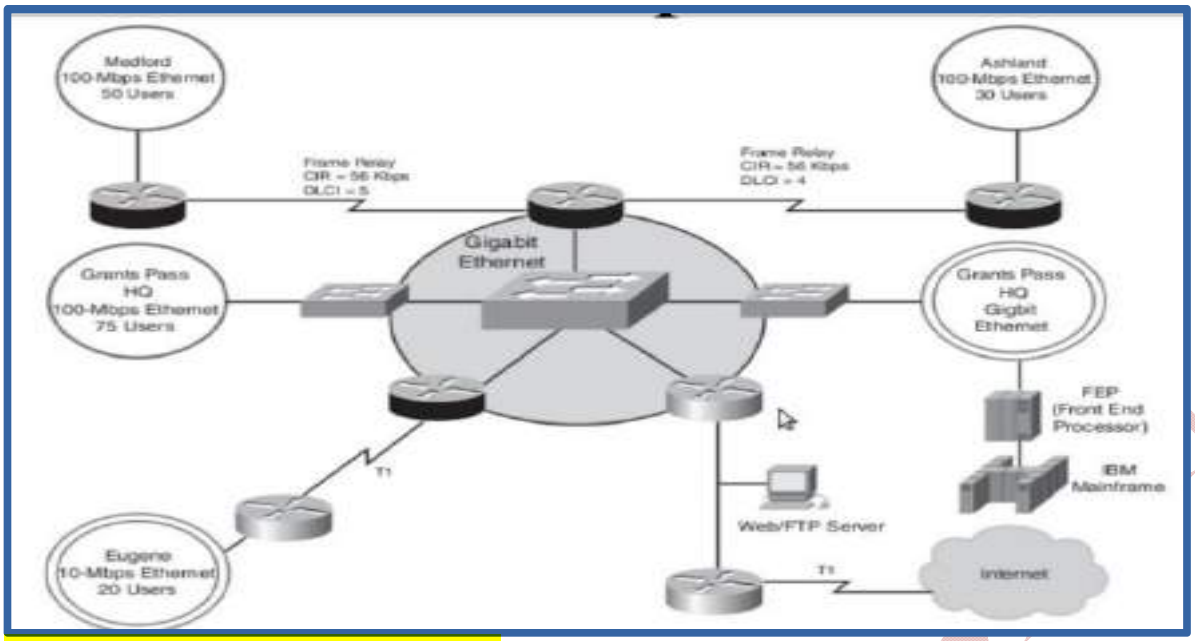
Simulate FTP Scenario

In this module
We shall recall and use
- Factors affecting goodput
- Using Inet framework
- Expectations in "events log"
- What to model?

Factors affecting goodput (1 of 3)
- 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

Factors affecting goodput (2 of 3)
- Workstation & server performance factors:
  - Disk-access speed
  - Disk-caching size
  - Device driver performance
- Computer bus performance (capacity/arbitration)

A Real World Scenario

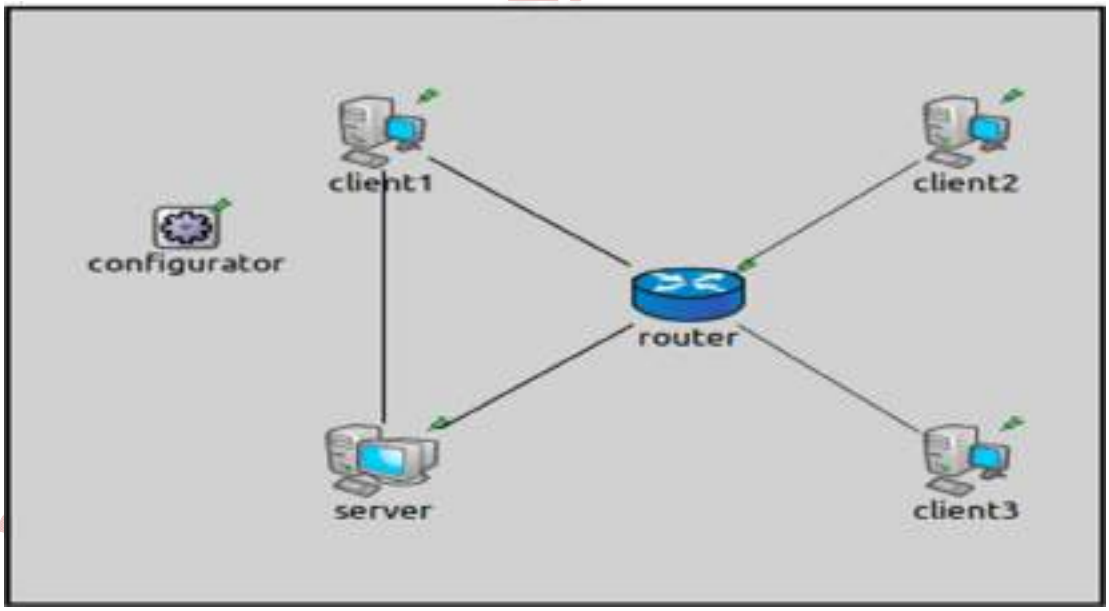


Factors affecting goodput (3 of 3)

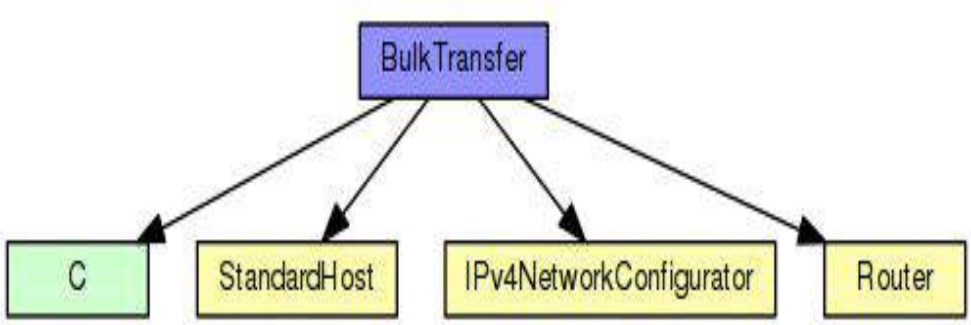
- Processor (CPU) performance
- Memory performance (access time for real and virtual memory)
- Operating system inefficiencies
- Application inefficiencies or bugs

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		

END

TOPIC 88

Summarizing top-down approach

In this module

We shall recap

- Top-down approach to NeMS
- Start with application layer

Summarizing top-down approach

Application layer Roll-out for M&S

Client Server Architectures
Web Server modelling
HTTP modelling
(Non) Persistent connections
Cache response time
FTP efficiency
SMTP scalability
DNS Load distribution
Query loss rate
DNS traffic load
P2P scalability
Torrents efficiency/reliability

Client Server Architectures
Web Server modelling
HTTP modelling
(Non) Persistent connections
Cache response time
FTP efficiency
SMTP scalability
DNS Load distribution
Query loss rate
DNS traffic load
P2P scalability
Torrents efficiency/reliability

END

TOPIC 89

Simulating DoS Attack

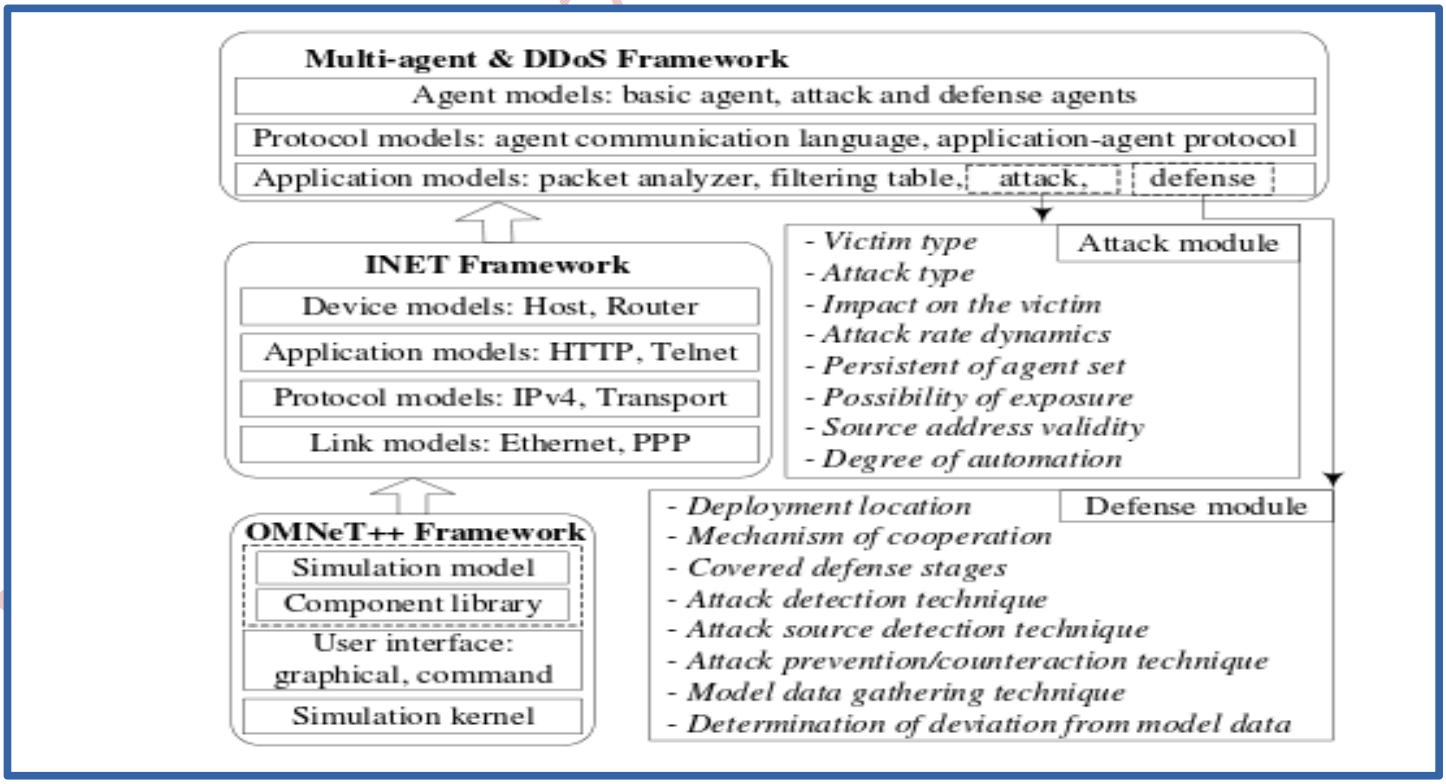
In this module
We shall understand

- Attack and defense module
- Implementing Ping of death attack in OMNET++

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");

END

TOPIC 90

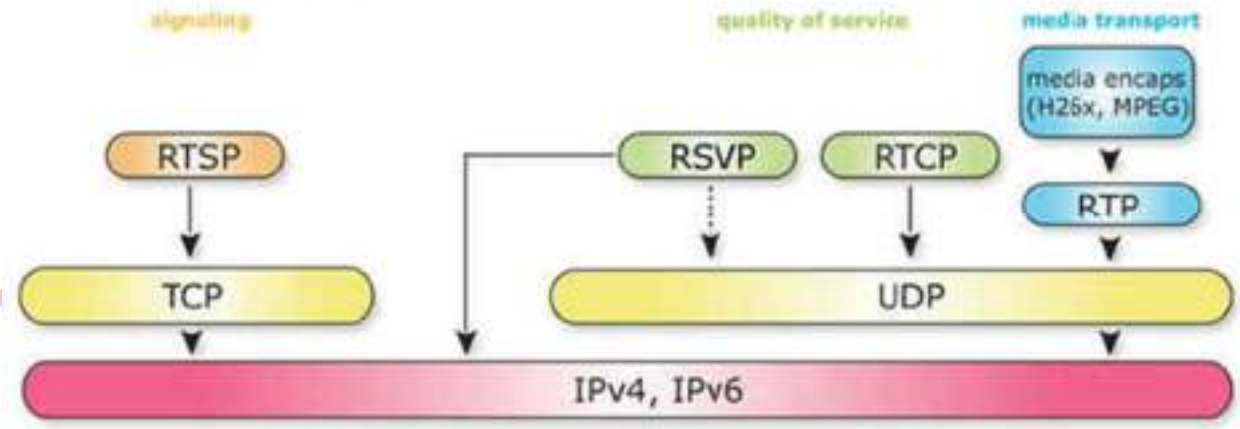
Simulate RTP with Packet Loss

In this module

We shall understand

- A quick RTP round-up
- Recalling delay/jitter
- Inet for simulating RTP
- Determining packet loss

Family of RTP



RTP

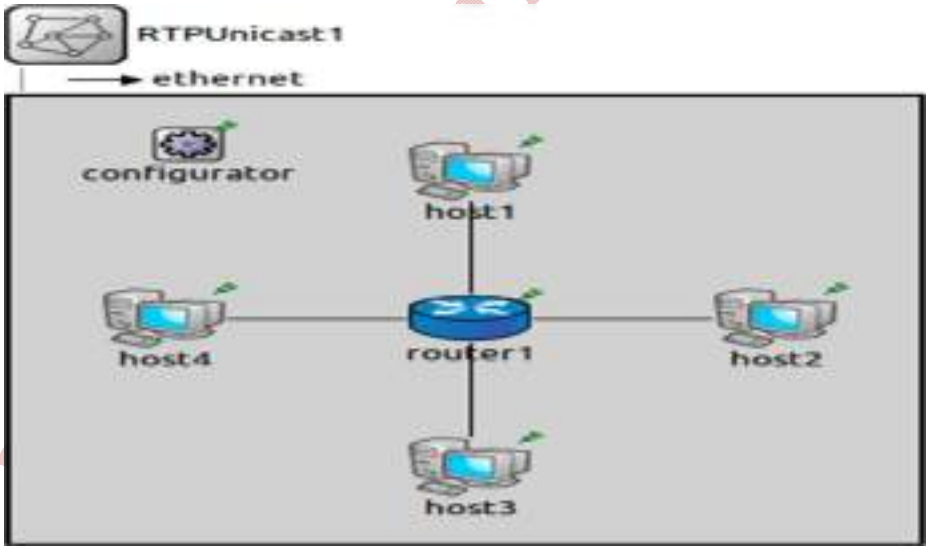
- Real-time Transport Protocol (RTP) is a network protocol

- Delivers audio/video over IP networks
- Streaming media
- Telephony
- Video teleconference
- Television service
- Push-to-talk over web

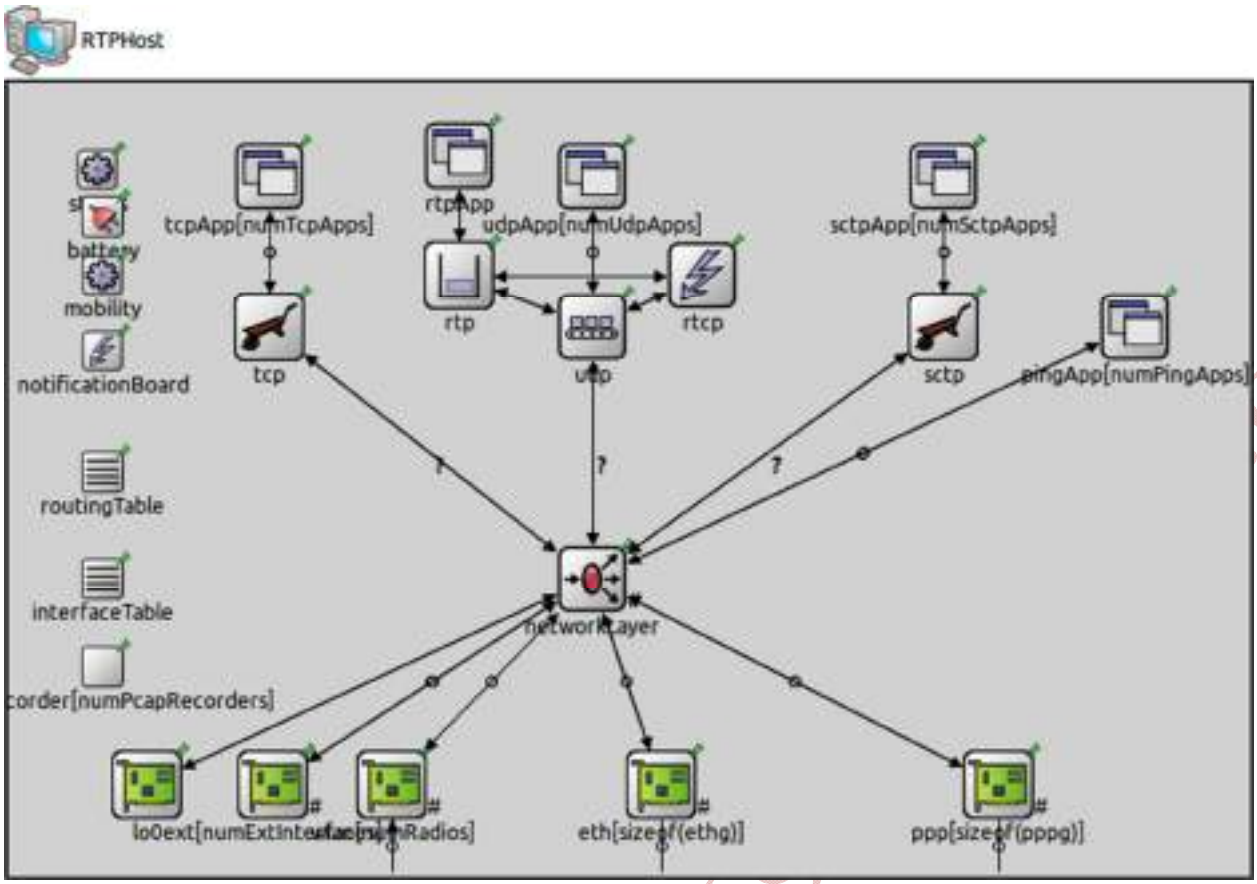
Simulate RTP with Packet Loss

Delay/Jitter Analysis Points

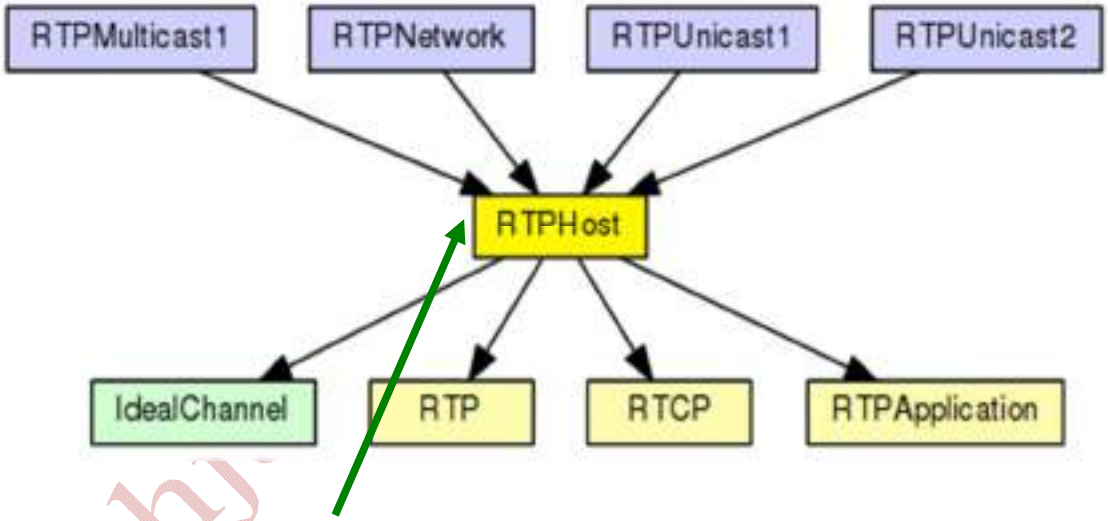
Inet for Simulating RTP
(examples/rtp/unicast1/unicast1.ned



src/nodes/rtp/RTPHost.ned



Usage Diagram and Statistics



dropPk dropped packets sum(packetBytes)

END

TOPIC 91+92

Reading material

Real-time Transport Protocol (RTP) is a network protocol used to transmit real-time media data, such as audio and video, over IP networks. RTP provides end-to-end delivery services for data with real-time characteristics, including the following:

- Provision of timing
- Loss detection
- Correction mechanisms

RTP is widely used in applications such as:

- Streaming media
- Telephony
- Video teleconference
- Television service
- Push-to-talk over web

How Real-time Transport Protocol (RTP) work in a network?

It works by breaking the data into small packets and sending them over the network to the intended receiver. RTP also includes a control protocol called RTCP (Real-time Transport Control Protocol), which provides feedback to the sender about the quality of the transmission and enables the receiver to control the flow of data.

Overall, RTP is essential for ensuring that real-time media data is delivered efficiently and reliably over IP networks, providing a smooth and seamless user experience for applications such as video conferencing and streaming media.

On which layer of network Real-time Transport Protocol (RTP) works?

Real-time Transport Protocol (RTP) operates at the application layer of the OSI (Open Systems Interconnection) model, which is Layer 7.

While RTP operates at the application layer, it is often used in conjunction with lower-level transport protocols such as User Datagram Protocol (UDP) or Transmission Control Protocol (TCP), which operate at the transport layer (Layer 4) of the OSI model. **UDP is commonly used with RTP** because it provides low-latency, connectionless delivery of data, which is important for real-time applications.

Overall, RTP provides a standardized framework for the transmission of real-time audio and video data over IP networks, regardless of the underlying transport layer protocols used.

END

TOPIC 93

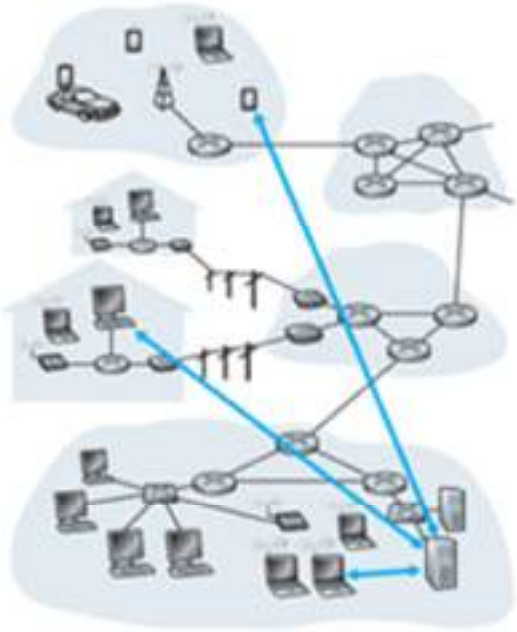
Client Server Architectures

In this module
We shall understand

- Definition of client server architecture
- Factors affecting architectures
- Performance

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

END

TOPIC 94

Web Server Modeling

In this module
We shall understand

- What does a web sever do?
- Ways to characterize it

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

END

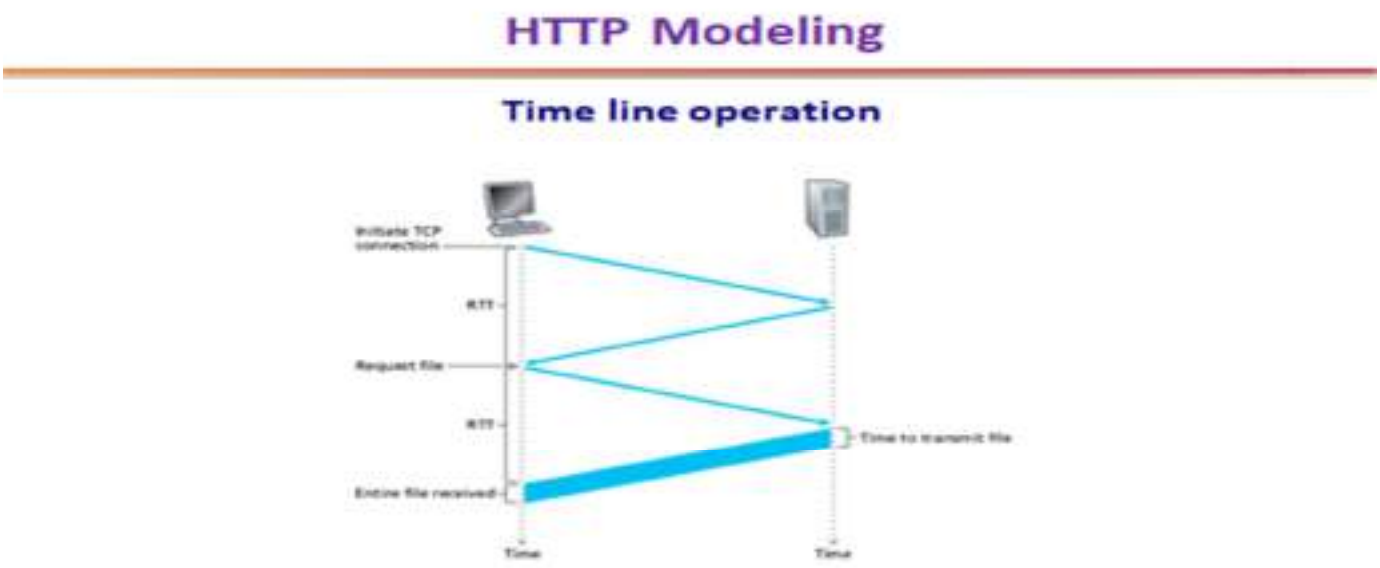
TOPIC 95

HTTP Modeling

In this module

We shall understand

- HyperText Transfer Protocol
- Its variants



Variants

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

– Caching

END

TOPIC 96

Non-Persistent Connections

In this module
We shall understand

- What is non-persistence
- Modeling non-persistence

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

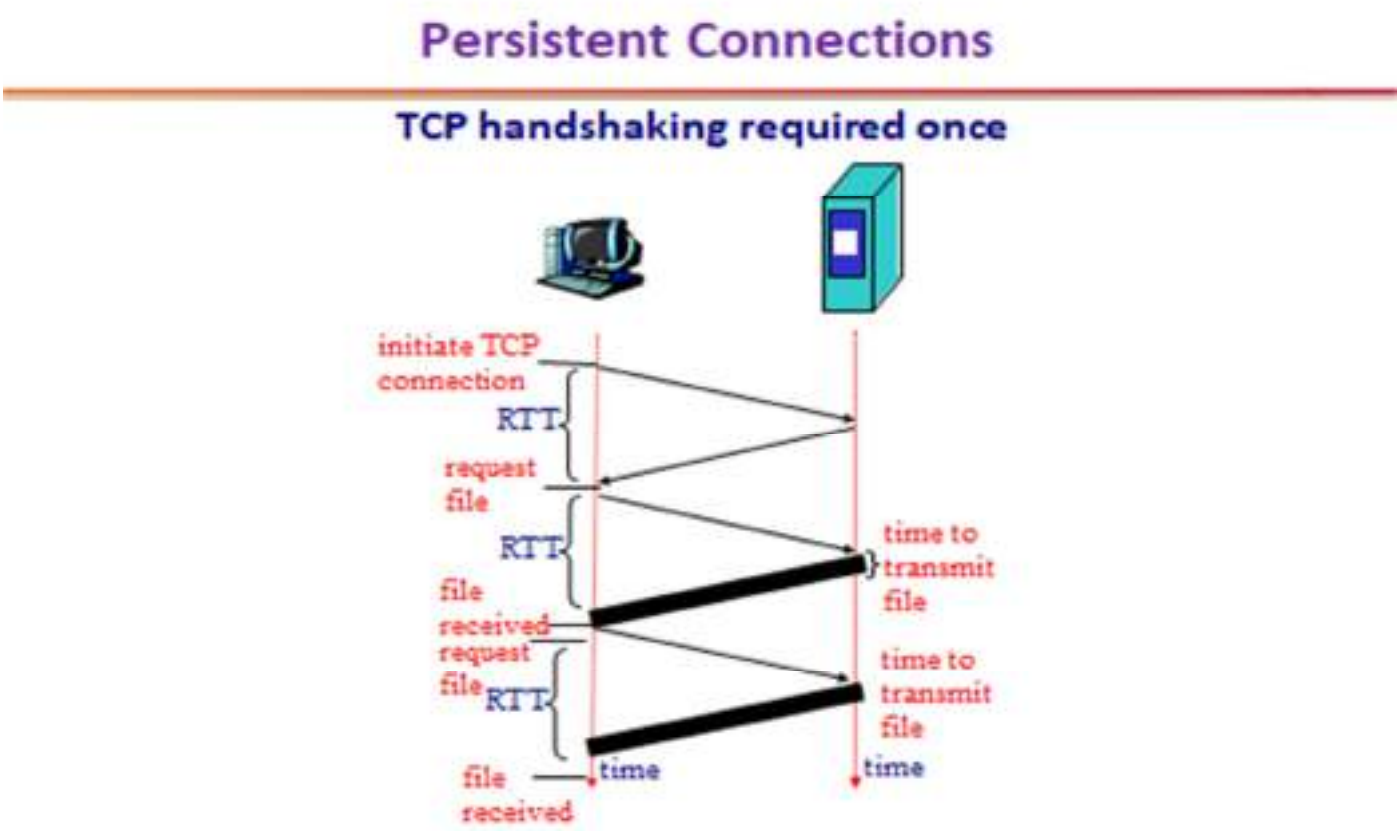
END

TOPIC 97

Persistent Connections

In this module
We shall understand

- HTTP 1.1 as persistent HTTP
- Its performance gain



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

END

TOPIC 98

Cache Response Time

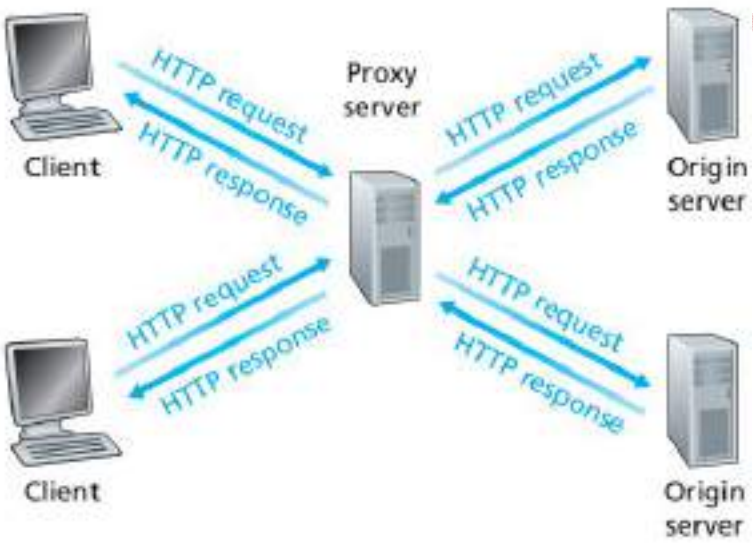
In this module
We shall understand

- Operation of cache
- Performance gain due to web cache

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

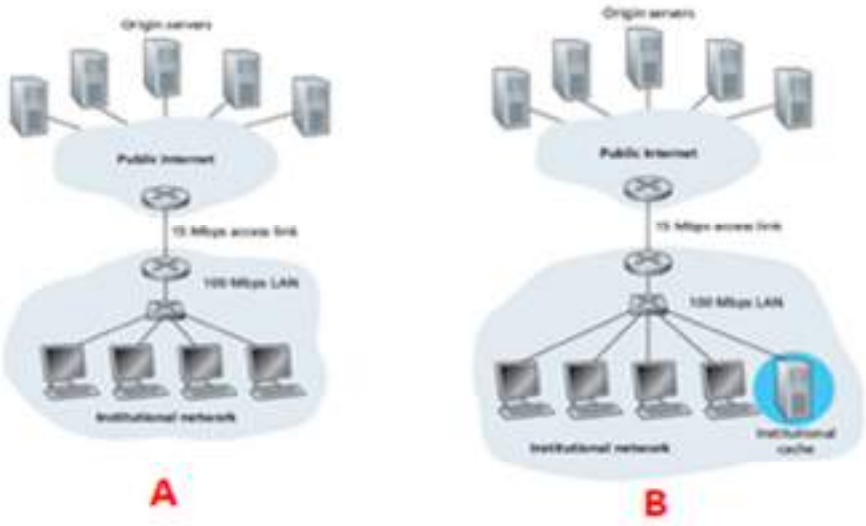


Advantages of caching

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

Cache Response Time

Simulating Scenarios with and without cache



Factors affecting caching

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

Example (1 of 3)

- 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

Example (2 of 3)

- Utilization on LAN = 15%
- Utilization on access link = 100%
- Total delay = Internet delay + access delay + LAN delay

= 2 sec + minutes + milliseconds

Example (3 of 3)

- 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

END

Week 08

TOPIC 99

FTP Efficiency

In this module

We shall understand

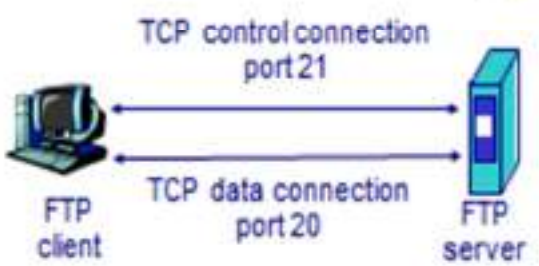
- Basic operation of FTP
- Evaluating file transfer

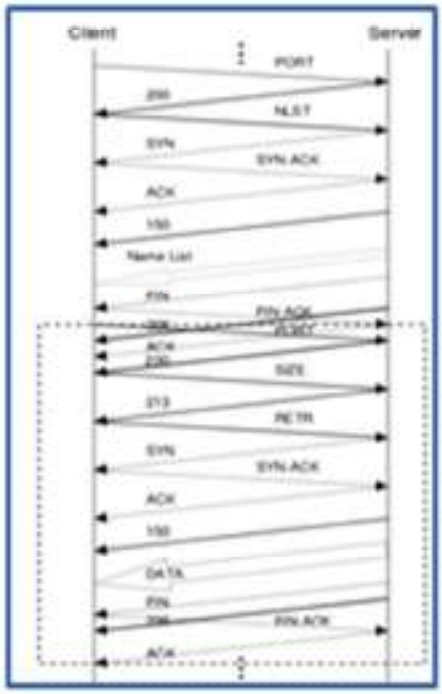
FTP operation

- 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

FTP Efficiency

Control Signaling of FTP





Computational Efficiency of FTP
(COURTESY: ALEBRA TECHNOLOGIES INC

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

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

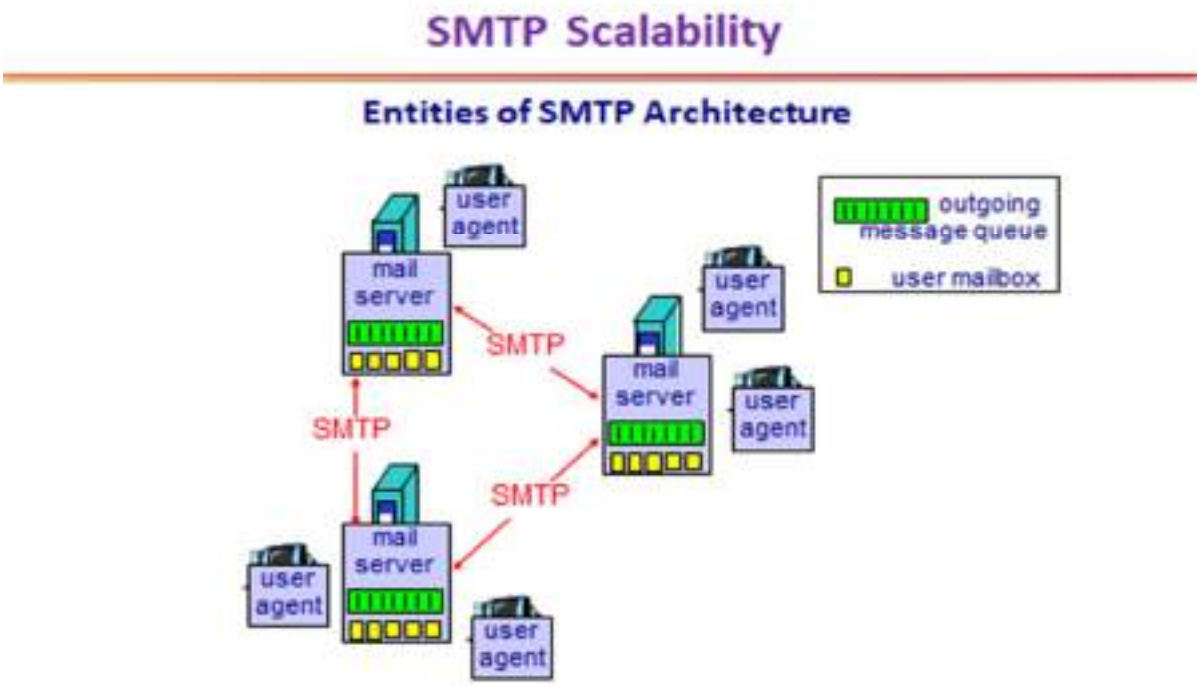
END

TOPIC 100

SMTP Scalability

In this module
We shall understand

- Entities of SMTP architecture
- Assessing scalability
- Efficiency and speedup



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
    - Discipline

Efficiency & speed-up for SMTP

CS432 Handouts Made by Mahjabeen
mahjabeen97869@gmail.com
contact # 0321 2711298

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

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

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

END

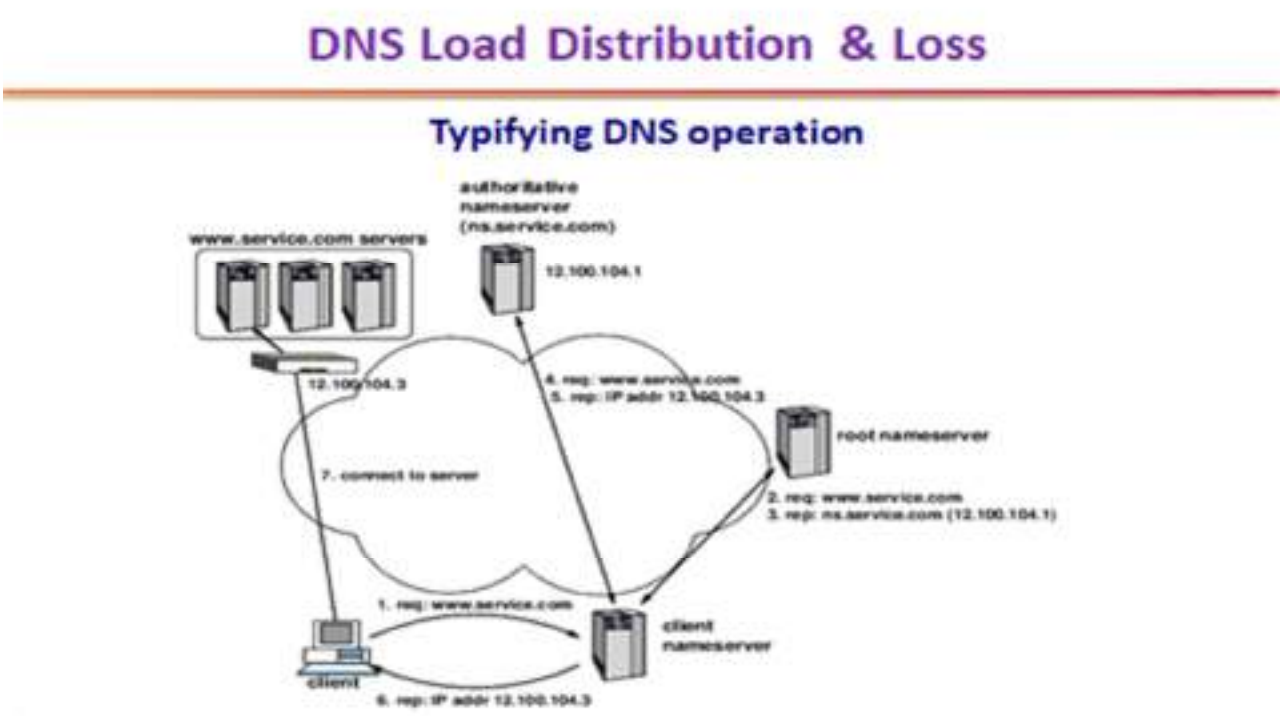
TOPIC 101

DNS Load Distribution & Loss

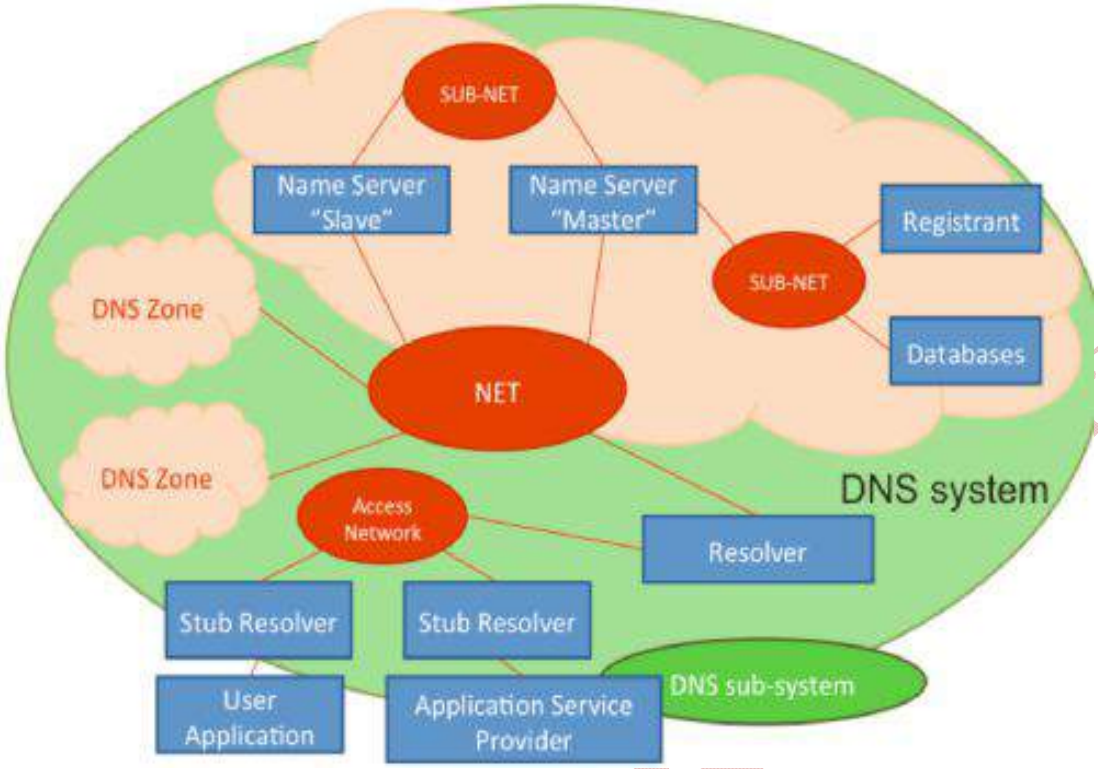
In this module

We shall understand

- Operational overview of DNS
- Quantifying load distribution metrics



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

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

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

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

Mcqs

Health metrics

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

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

Health metrics

DNS Requests per Seconds (DNSR)

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

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

Health metrics

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

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

END

TOPIC 102

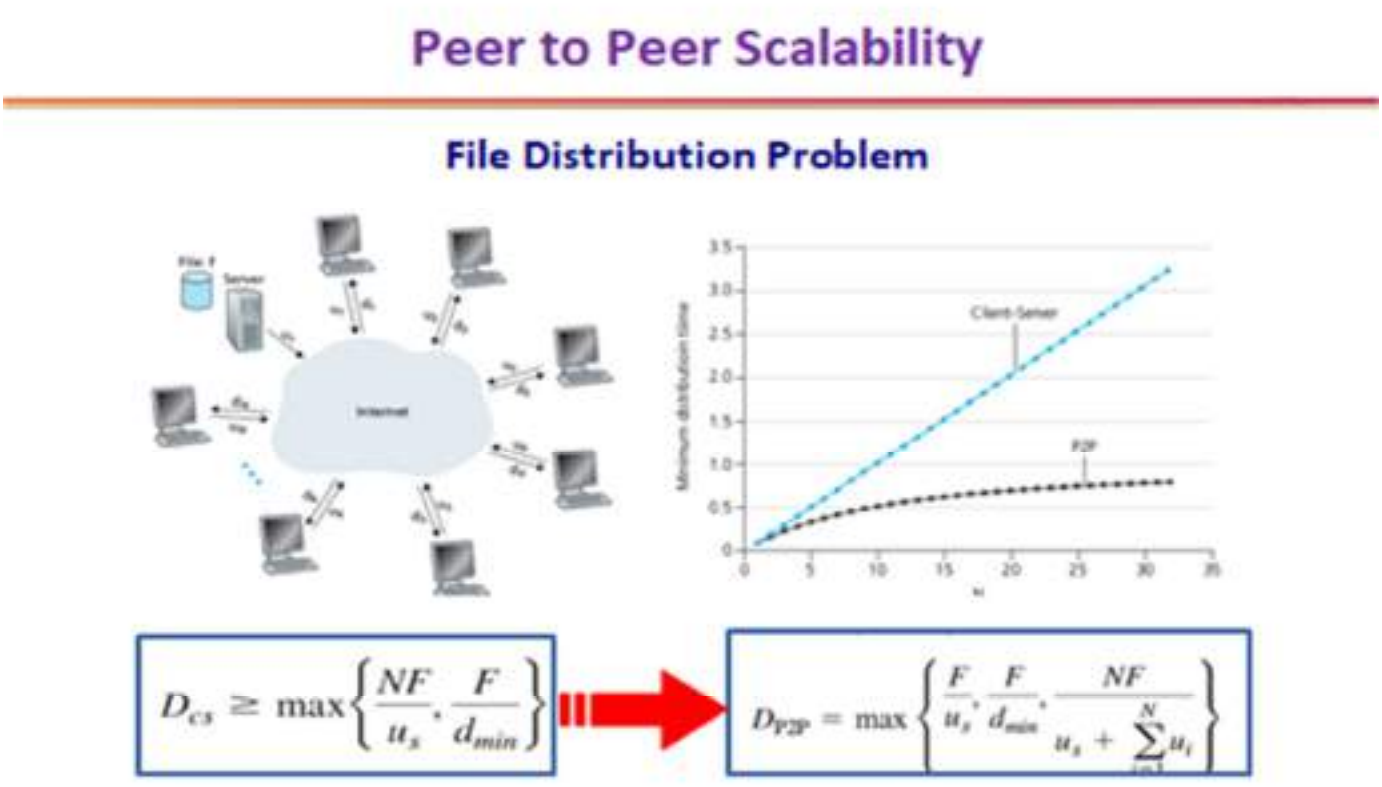
Peer to Peer Scalability

In this module
We shall understand

- P2P operation
- File distribution problem
- P2P performance

Operation

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



Performance

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

- 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

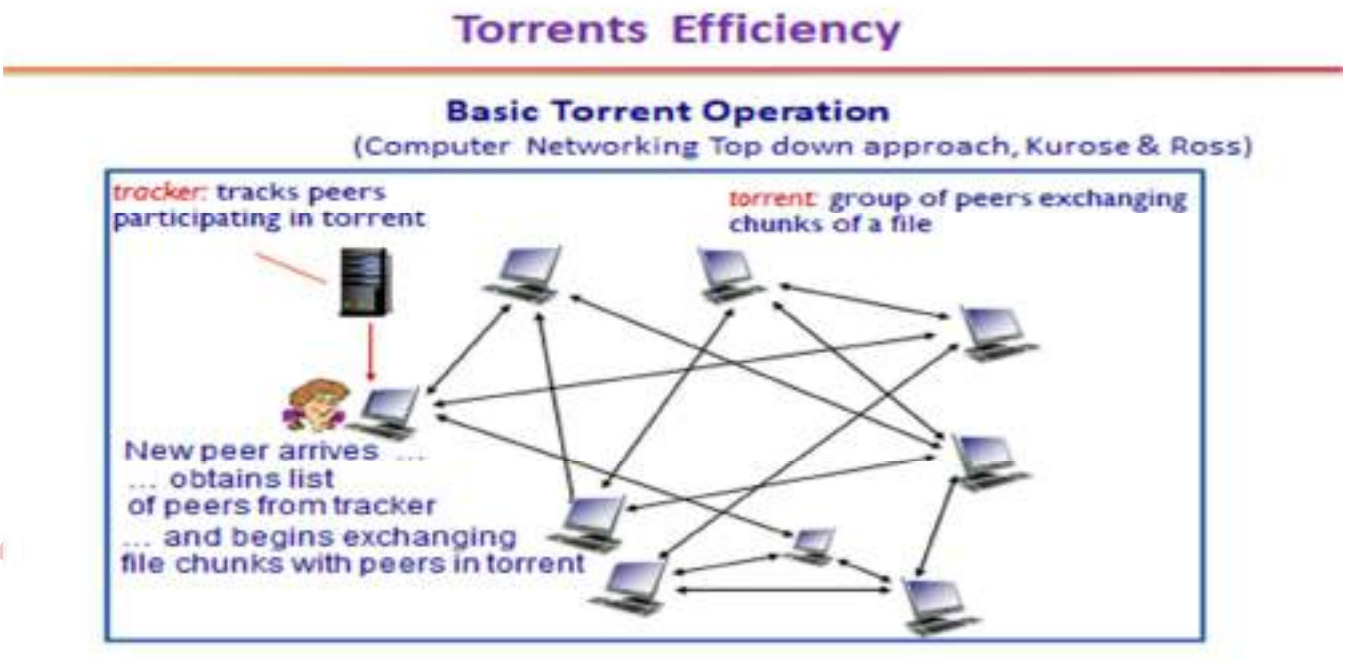
END

TOPIC 103

Torrents Efficiency

In this module
We shall understand

- Basic operation of BitTorrents
- Performance of BitTorrents



Factors affecting efficiency

- Heterogeneous upload capacity

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

Performance

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

Wu, Gang, and Tzi-cker Chiueh. "How efficient is BitTorrent?." Electronic Imaging 2006. International Society for Optics and Photonics, 2006.

Yu, Lidong, Ming Chen, and Changyou Xing. "Quantifying downloading performance of locality-aware bittorrent protocols." Computational Science and Its Applications-ICCSA 2011. Springer Berlin Heidelberg, 2011. 562-576.

END

TOPIC 104

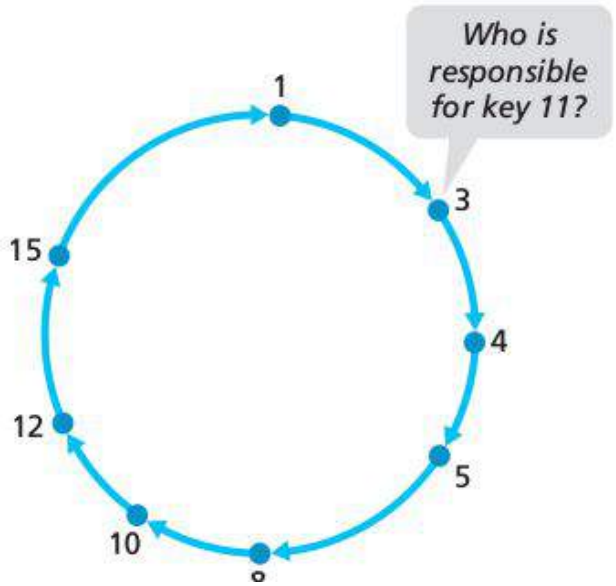
Reliability of Circular DHT

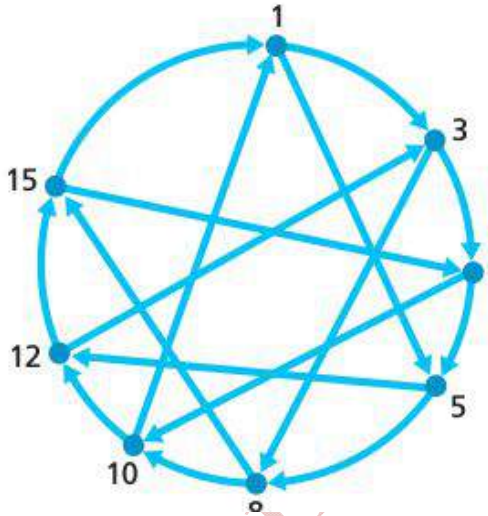
In this module

We shall understand

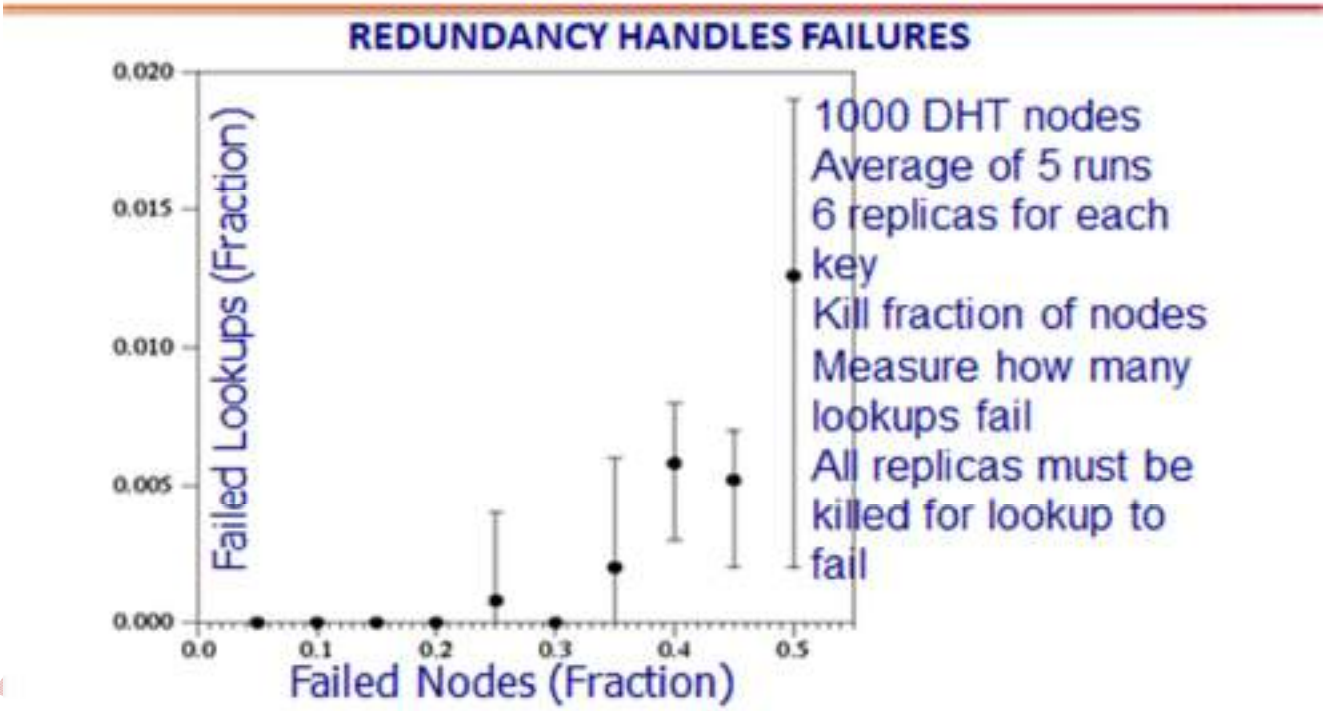
- Basic operation of circular DHT
- Redundancy for handling failures
- Cost of reliability expression

Operation of Circular DHT





Reliability of Circular DHT



Cost of Reliability

Russ Cox, “A Backup System built from a
Peer-to-Peer Distributed Hash Table.” http://pdos.lcs.mit.edu/chord

Mahajan, Ratul, Miguel Castro, and Antony Rowstron. "Controlling the cost of reliability in peer-to-peer overlays." Peer-to-Peer Systems II. Springer Berlin Heidelberg, 2003. 21-32.

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

END

TOPIC 105

Problem Set 1

In this module
We shall solve problems to

- Recap network latencies
- Recall HTTP performacne

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

END

TOPIC 106

Problem Set 2

In this module
We shall solve
problems to

- Refresh P2P operation
- Understand user activity

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?

END

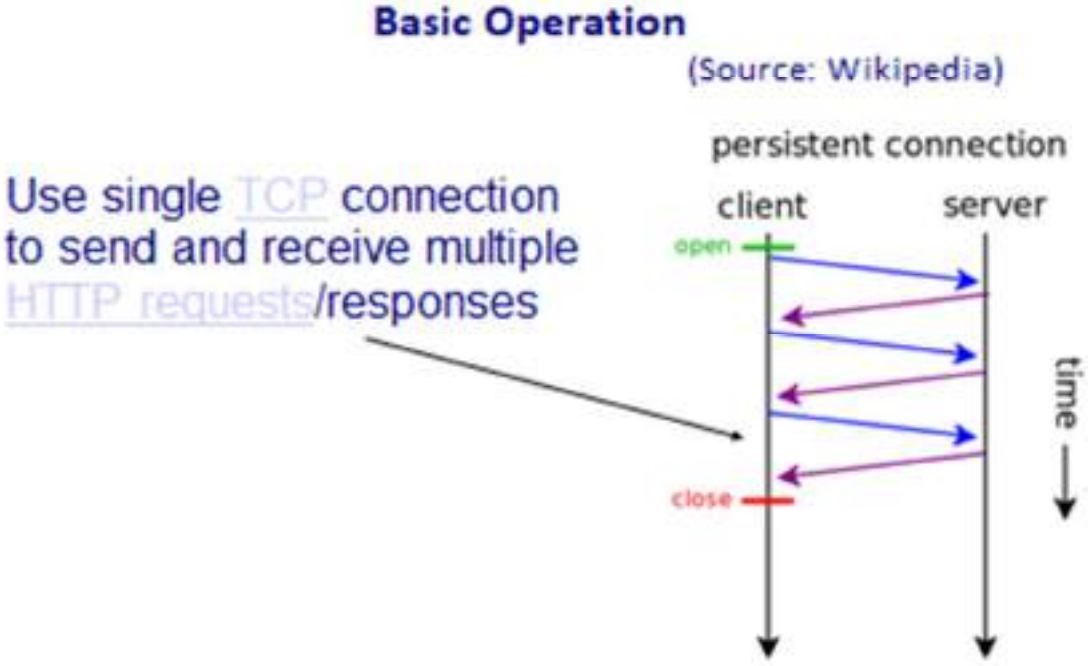
TOPIC 107

Simulate HTTP Persistence

In this module
We shall understand

- Basic operation of HTTP Persistence
- Support in OMNET++
- Customization

Simulate HTTP Persistence



HTTP Evolution

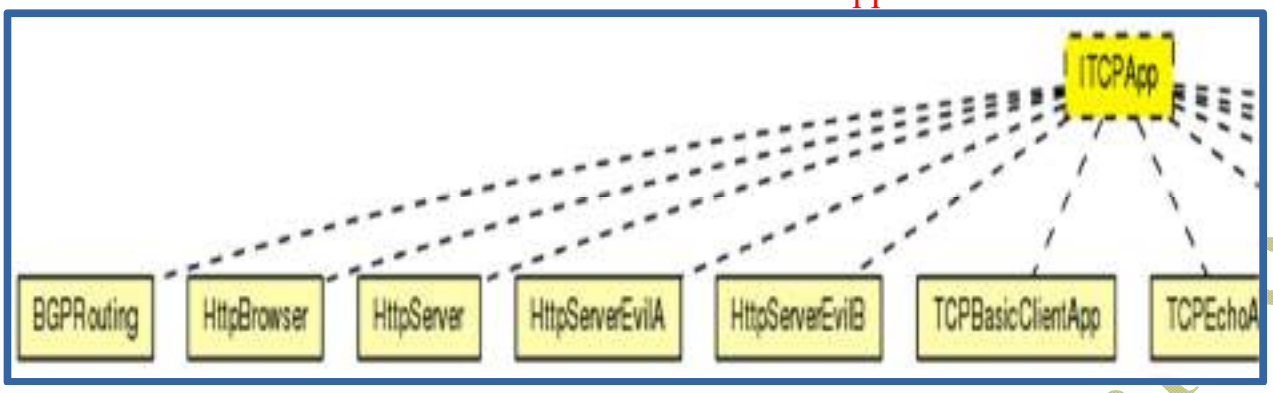
- RFC 793 does not support persistence
  - HTTP1.0
- Additional mechanism needed
  - Use keep-alive
- HTTP 1.1 is persistent by default

HTTP 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 Difference between random and scripted mode

- 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

END

TOPIC 108

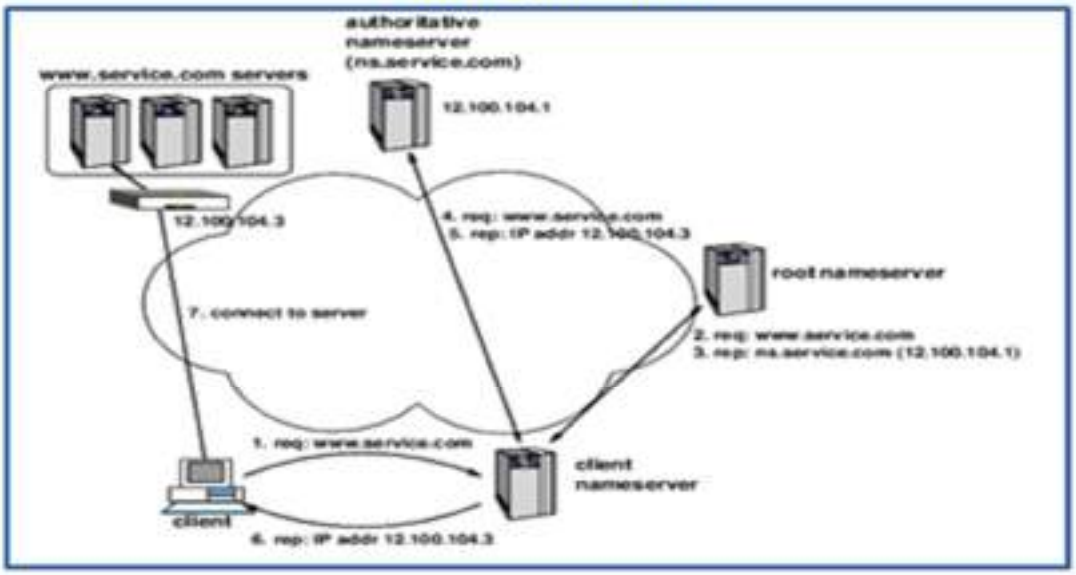
Simulate DNS Query Response

In this module
We shall understand

- Basic operation of DNS
- Newly added support in OMNET++

Simulate DNS Query Response

Basic Operation



DNS Support in OMNET++
https://github.com/saenridanra/inet-dns-extension
(Courtesy: Andreas Rain)

- 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

END

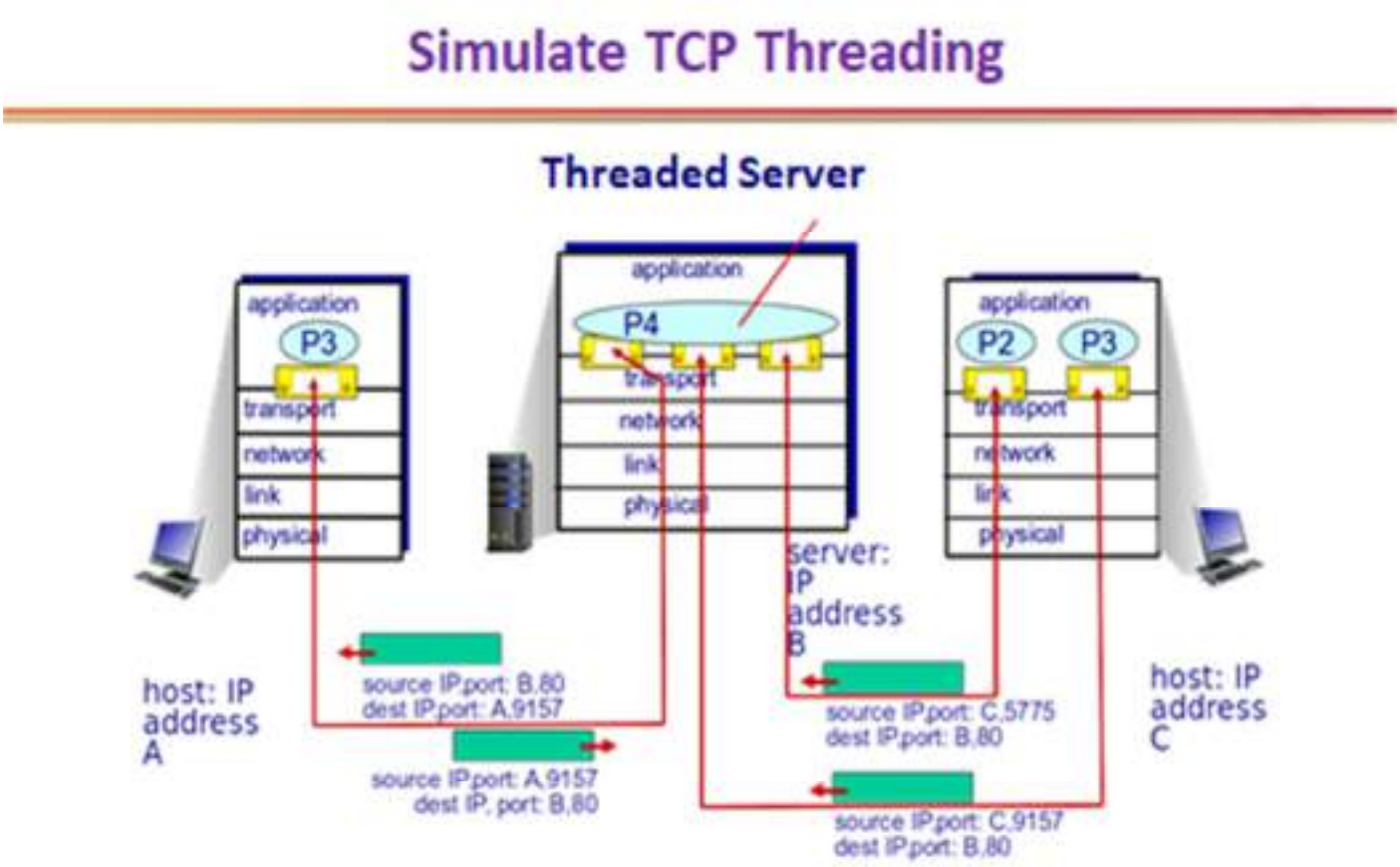
TOPIC 109

Simulate TCP Threading

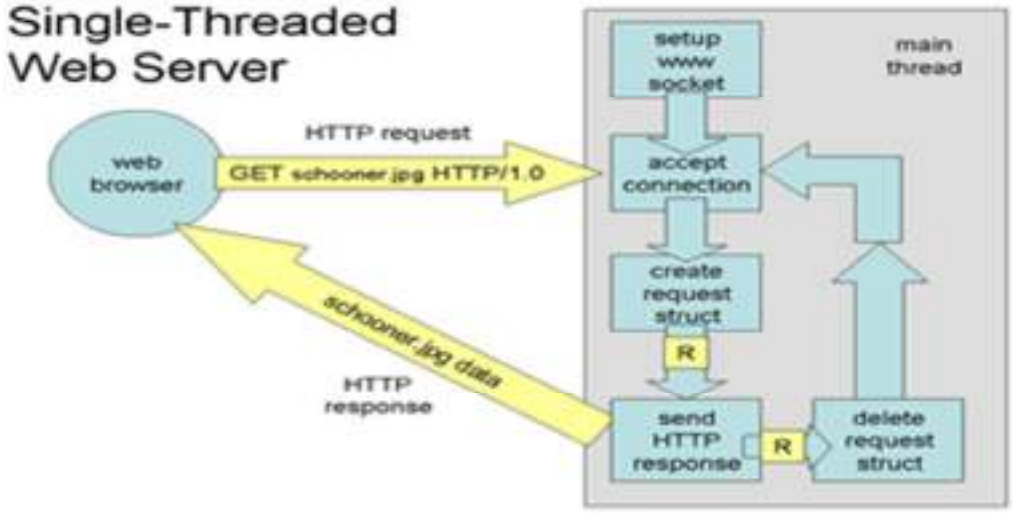
In this module

We shall understand

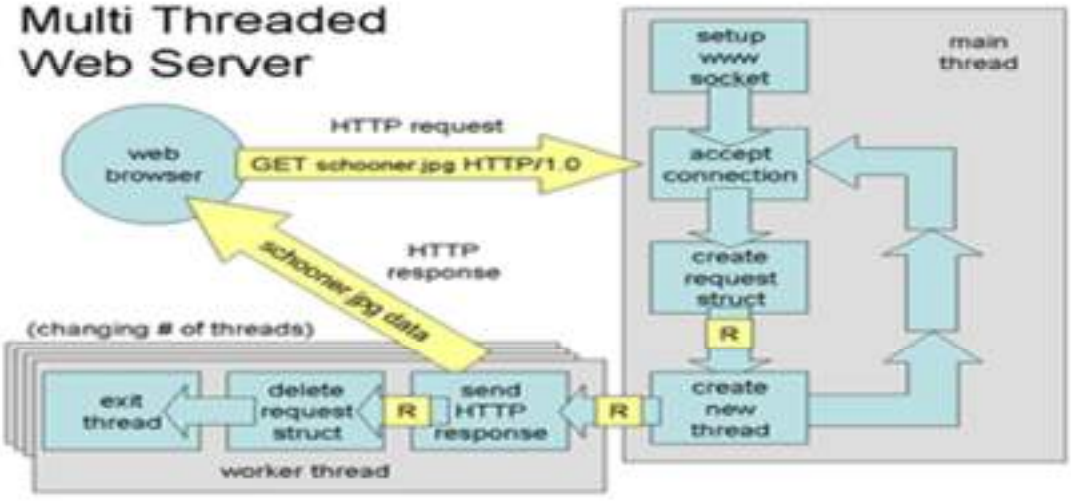
- Operation of threading in TCP
- Variants of threading
- Support in HTTP



Simulate TCP Threading



Simulate TCP Threading



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**

END

TOPIC 110

Simulate HTTP Handshaking

In this module

We shall understand

- HTTP Messaging

HTTP Requests

HTTP/1.0:

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

HTTP/1.1:

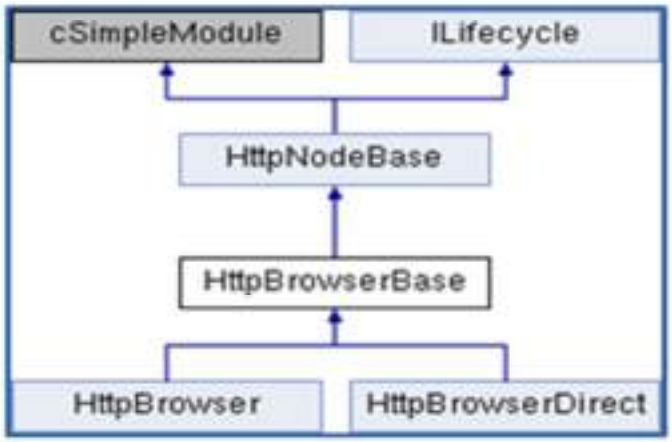
- GET, POST, HEAD MCQS
- PUT
- uploads file in entity body to path specified in URL field
- DELETE
- deletes file specified in the URL field

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

Simulate HTTP Handshaking

HttpBrowser Class Reference (Inheritance Diagram)



END

TOPIC 111

Intro & Transport Services
In this module
We shall understand
- Transport services
- Modeling approach to transport layer

Introduction
- Transport layer is the big brother

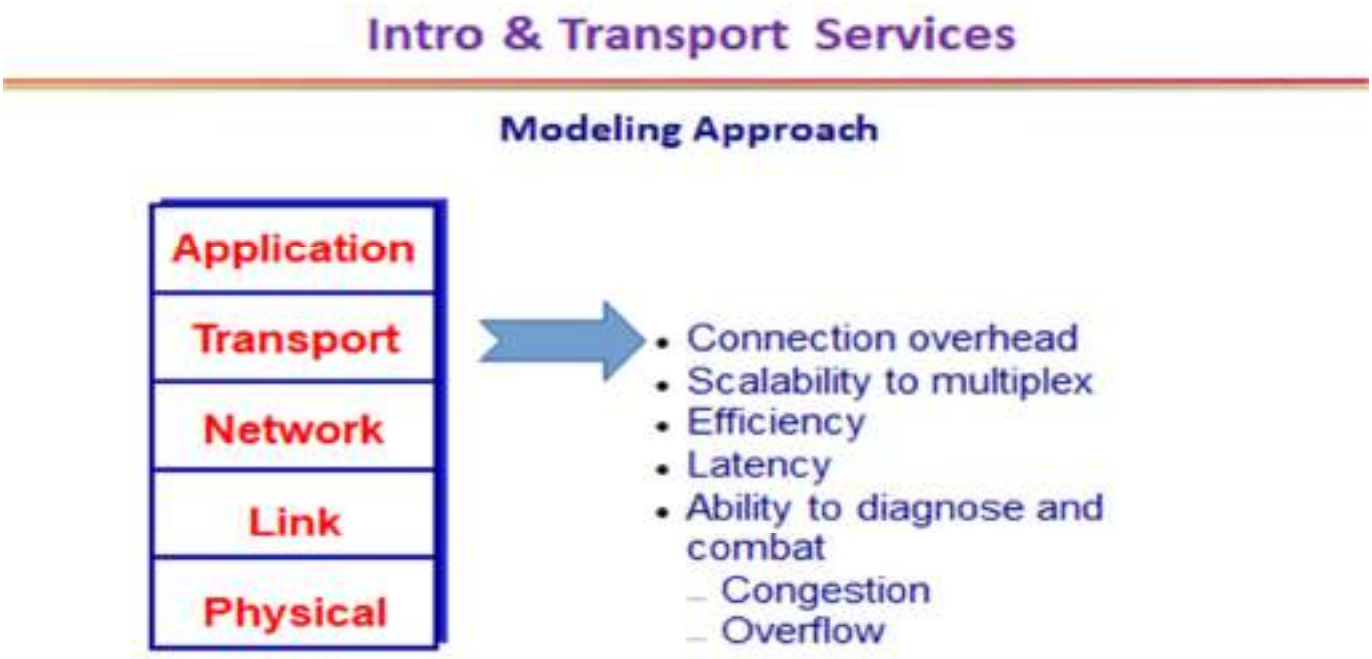
- Manages end to end delivery of data
- Modeling of transport layer is pivotal to the overall performance

Transport Services (1 of 2)

- Multiplexing and demultiplexing
- Reliable, in-order delivery (TCP)
- Congestion control
- Flow control
- Connection setup

Transport Services (2 of 2)

- Unreliable, unordered delivery: UDP
- "best-effort" IP
- Services not available
  - Delay guarantees
  - Bandwidth guarantees



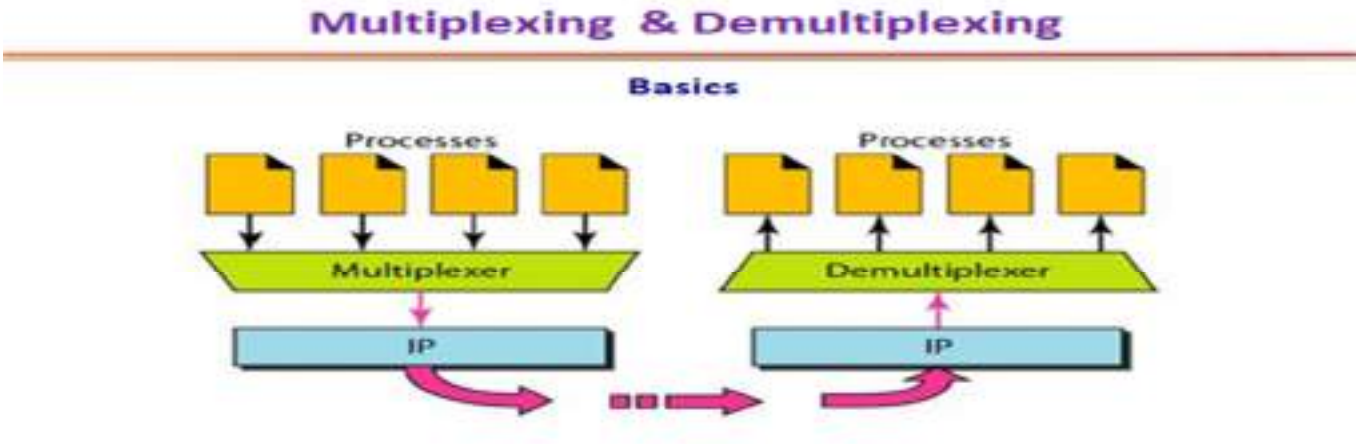
END

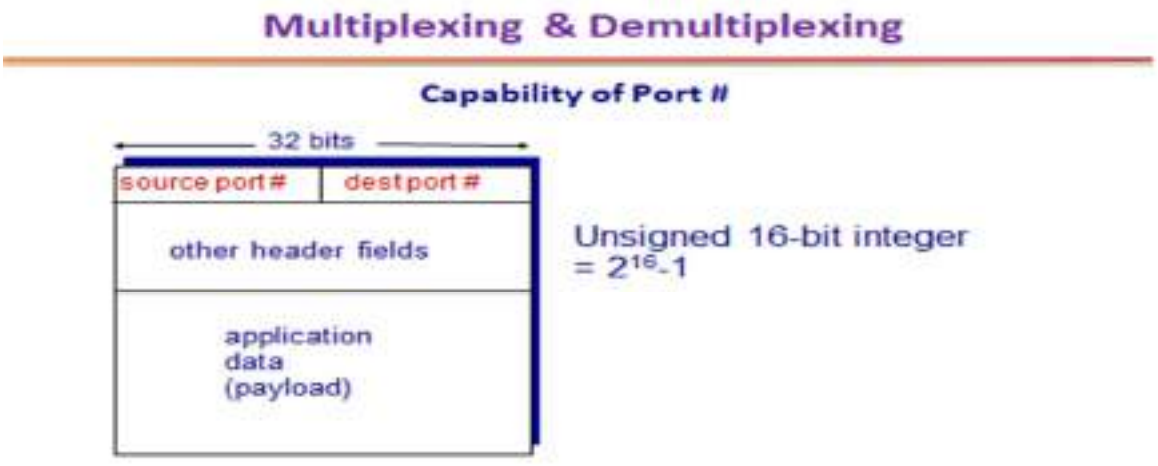
TOPIC 112

Multiplexing & Demultiplexing

In this module
We shall understand

- Basics of DeMux
- Capability of port numbers
- Cost of multiplexing

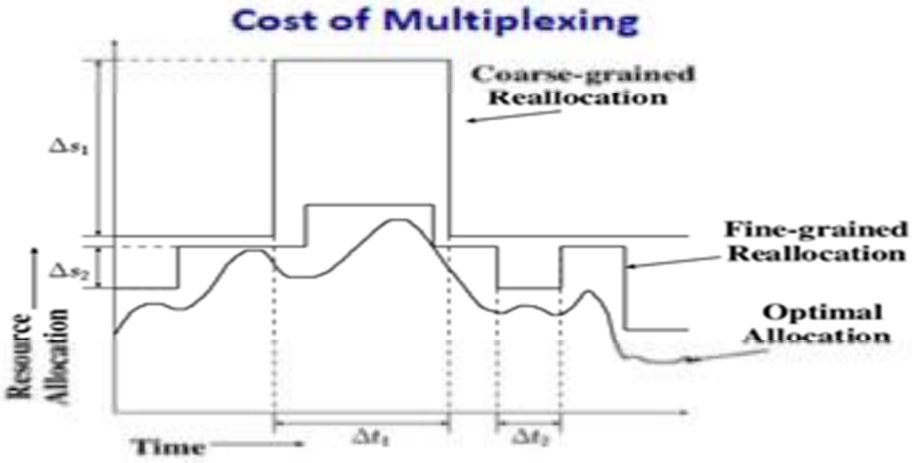




Cost of Multiplexing

Chandra, Abhishek, et al. "Quantifying the benefits of resource multiplexing in on-demand data centers." Computer Science Department Faculty Publication Series University of Massachusetts, Amherst (2003): 20.

Multiplexing & Demultiplexing



END

TOPIC 113

Intro & Transport Services

In this module

We shall understand

- Transport services
- Modeling approach to transport layer

Introduction

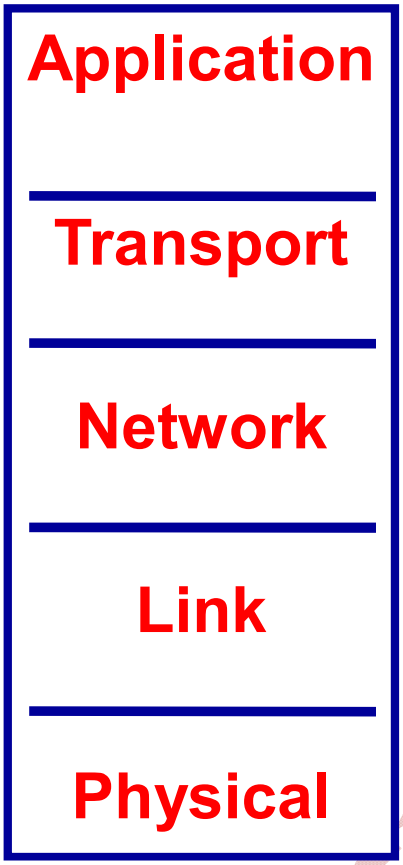
- Transport layer is the big brother
- Manages end to end delivery of data
- Modeling of transport layer is pivotal to the overall performance

Transport Services (1 of 2)

- Multiplexing and demultiplexing
- Reliable, in-order delivery (TCP)
- Congestion control
- Flow control
- Connection setup

Transport Services (2 of 2)

- Unreliable, unordered delivery: UDP
- "best-effort" IP
- Services not available
  - Delay guarantees
  - Bandwidth guarantees



Connection overhead
Scalability to multiplex
Efficiency
Latency
Ability to diagnose and combat
Congestion
Overflow

END

TOPIC 114

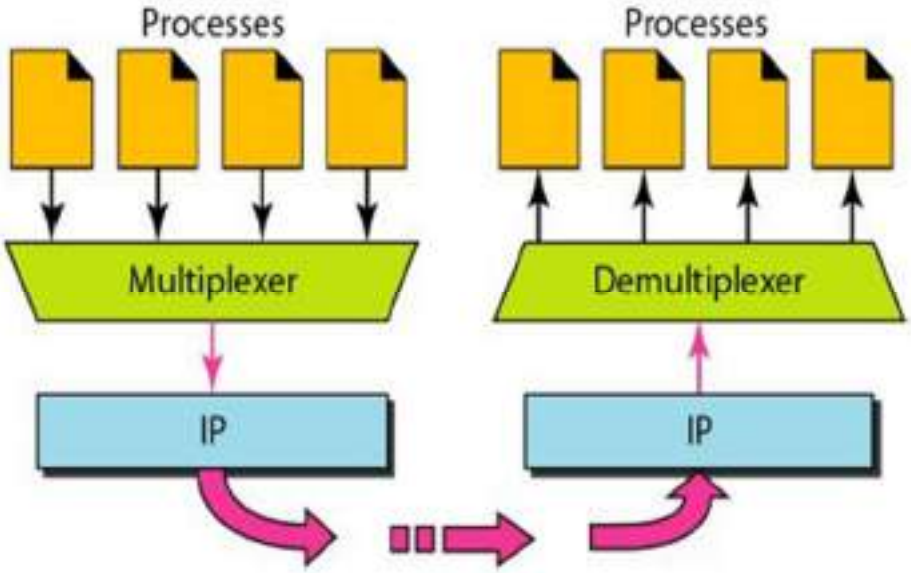
Multiplexing & Demultiplexing

In this module

We shall understand

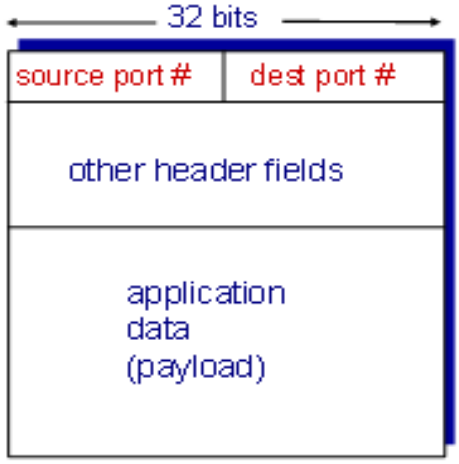
- Basics of DeMux
- Capability of port numbers
- Cost of multiplexing

BASIC



Multiplexing & Demultiplexing

Capability of Port #

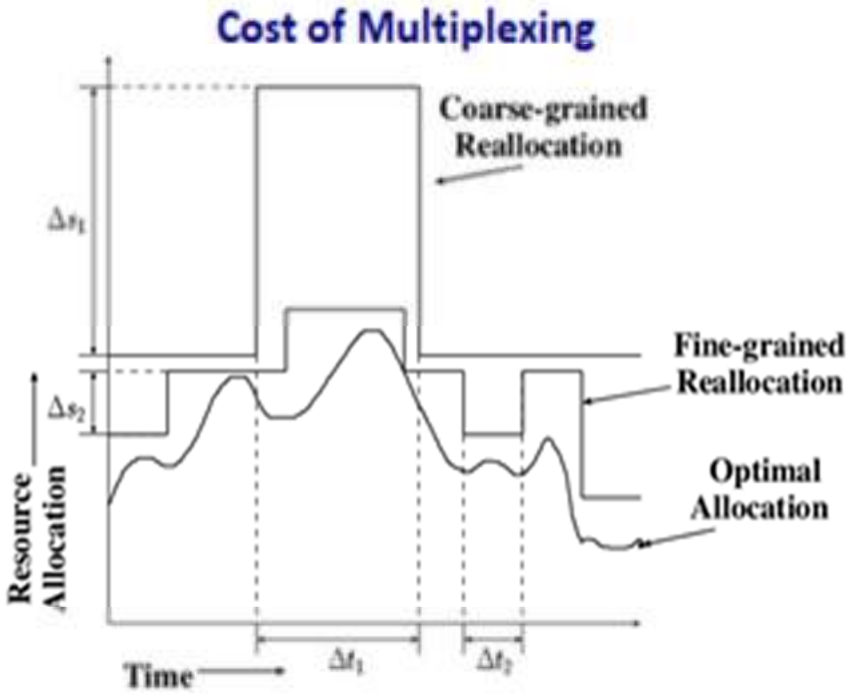


Unsigned 16-bit integer
$= 2^{16}-1$

Cost of Multiplexing

Chandra, Abhishek, et al. "Quantifying the benefits of resource multiplexing in on-demand data centers." Computer Science Department Faculty Publication Series University of Massachusetts, Amherst (2003): 20.

Multiplexing & Demultiplexing



END

TOPIC 115

Multiplexing Communication Link

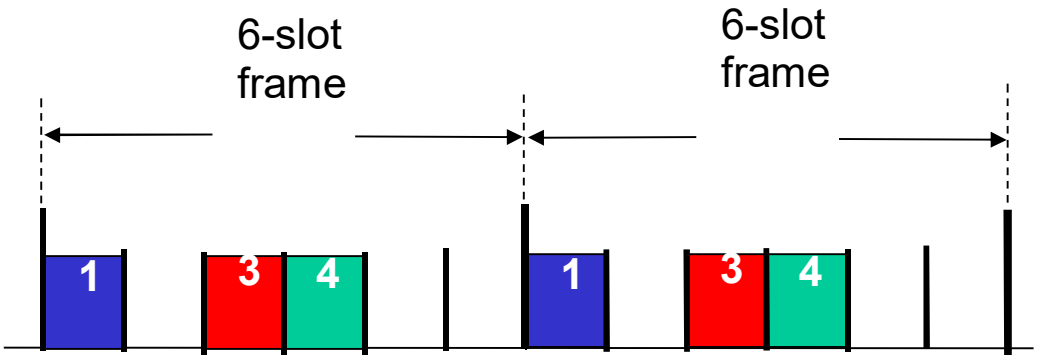
In this module

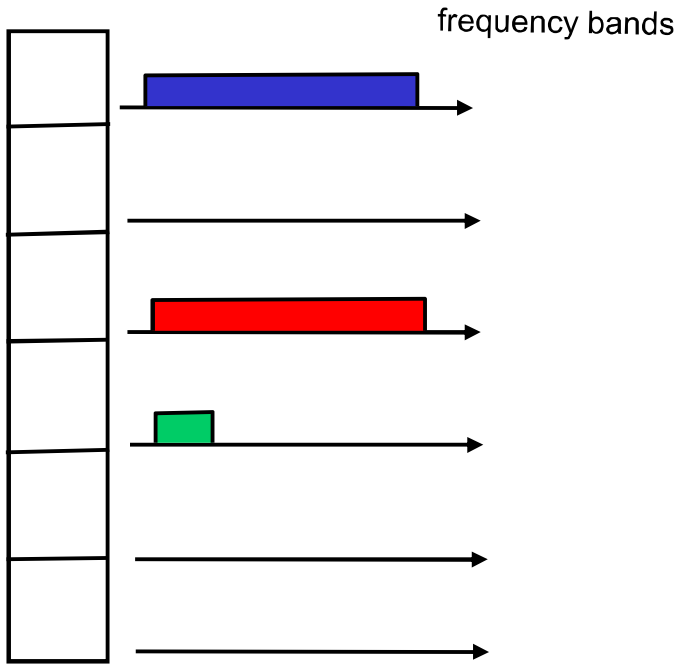
We shall understand

- Typical multiplexing techniques
- Capacity overhead
- Statistical (packet) multiplexing

Multiplexing Communication Link

Typical Multiplexing Techniques





Capacity Overhead

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

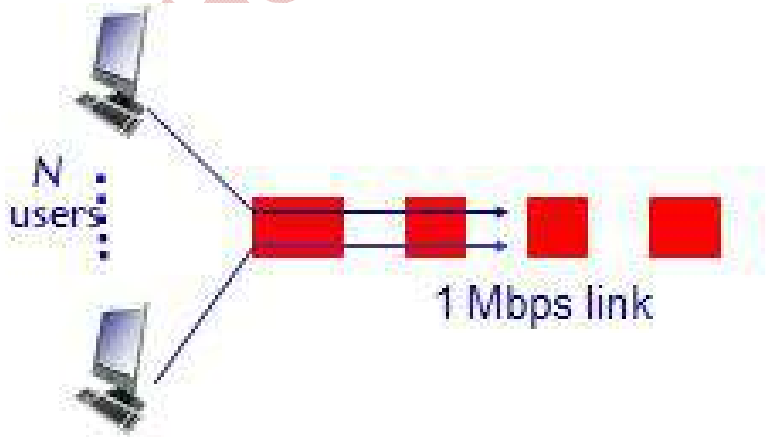
Capacity overhead ρ is defined as %age increase in the resource requirement of a practical multiplexing scheme when compared to the optimal

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

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



END

TOPIC 116

Checksum

In this module

We shall understand

- A brief of checksum
- How to measure its strength?

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

Stone, Jonathan, et al. "Performance of checksums and CRCs over real data." Networking, IEEE/ACM Transactions on 6.5 (1998): 529-543.
http://noahdavids.org/self_published/CRC_and_checksum.html

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
(1 of 2)

- 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

Undetected Errors
(2 of 2)

- 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

END

TOPIC 117

Go Back N

Retransmission strategy

In this module

We shall understand

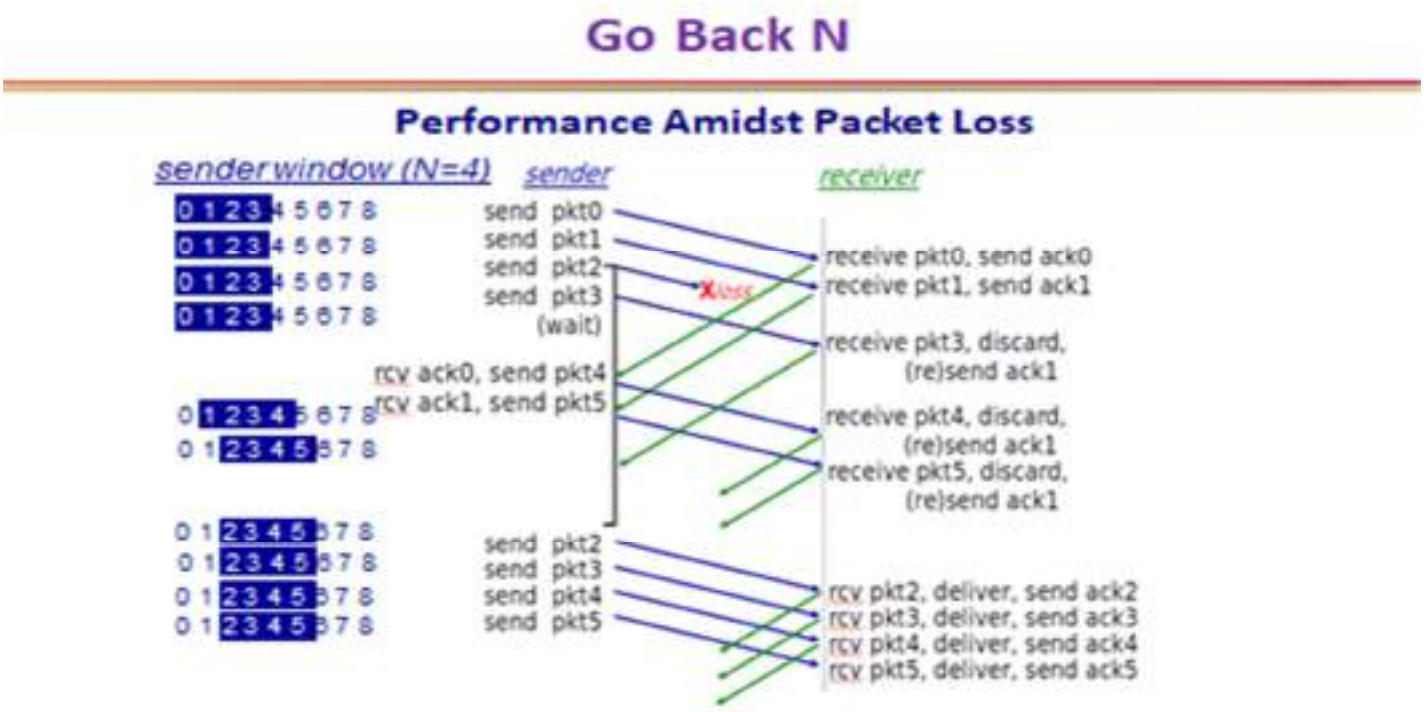
- How Go Back N

works?

- Efficiency vs buffer tradeoff

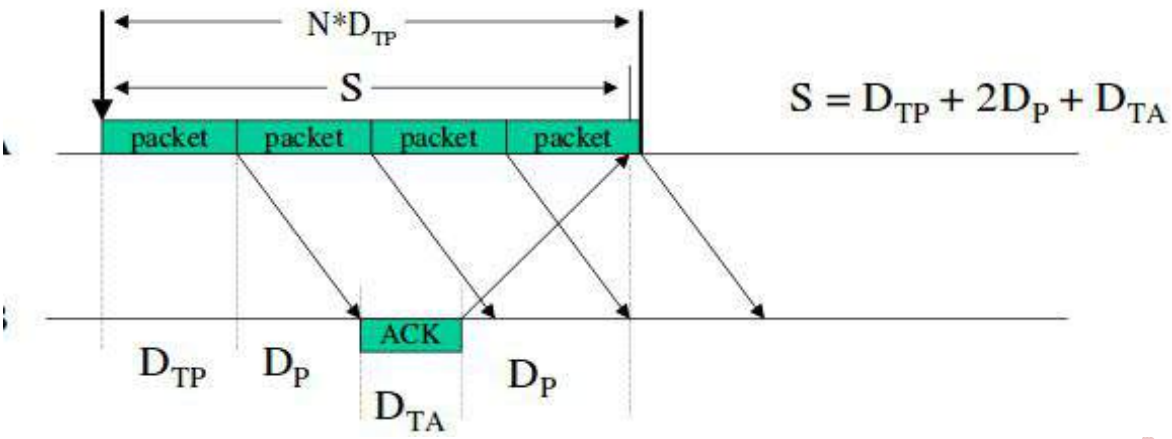
Introduction

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



Efficiency without Errors

(Courtesy: Eytan Modiano



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

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

- When an error occurs the entire window of N packets must be retransmitted
- X = Number of packets sent per successful transmission
- *Assume* $N = S/D_{TP}$ and $TO = N*D_{TP}$
- $E[X] = 1*(1-P) + (X+N)*P$

$E = 1/E[X]$

END

TOPIC 118

Selective Repeat

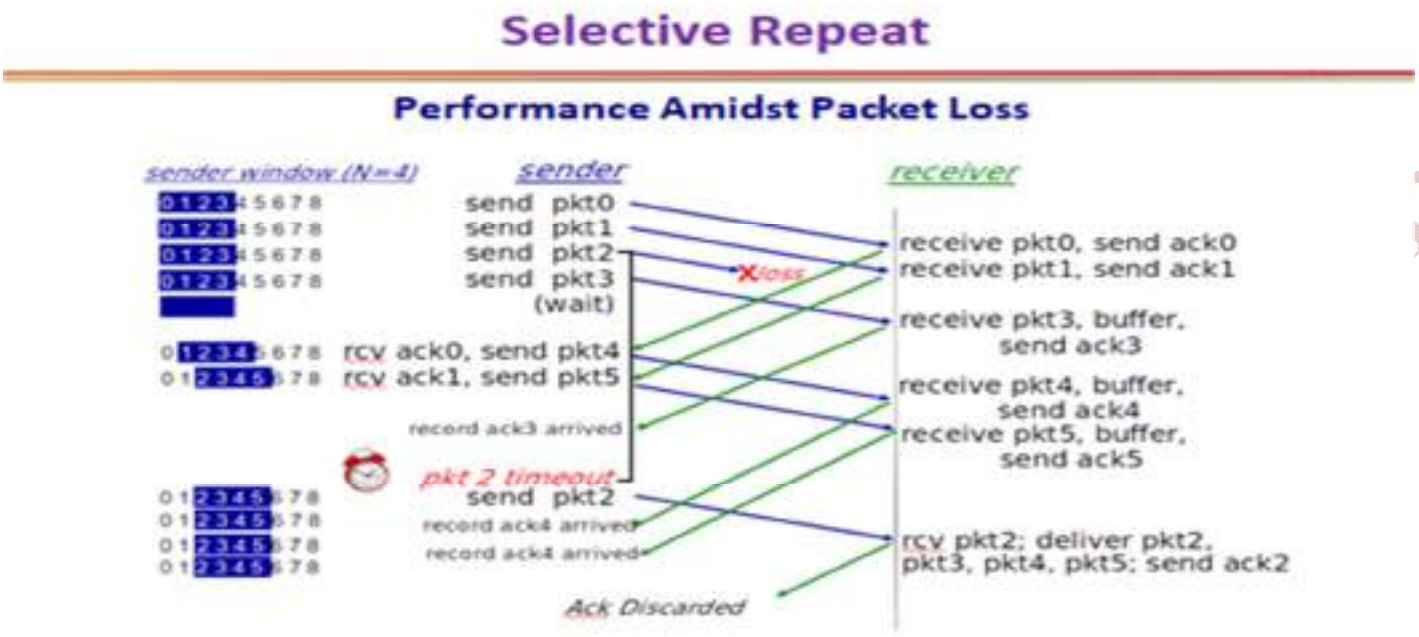
In this module

We shall understand

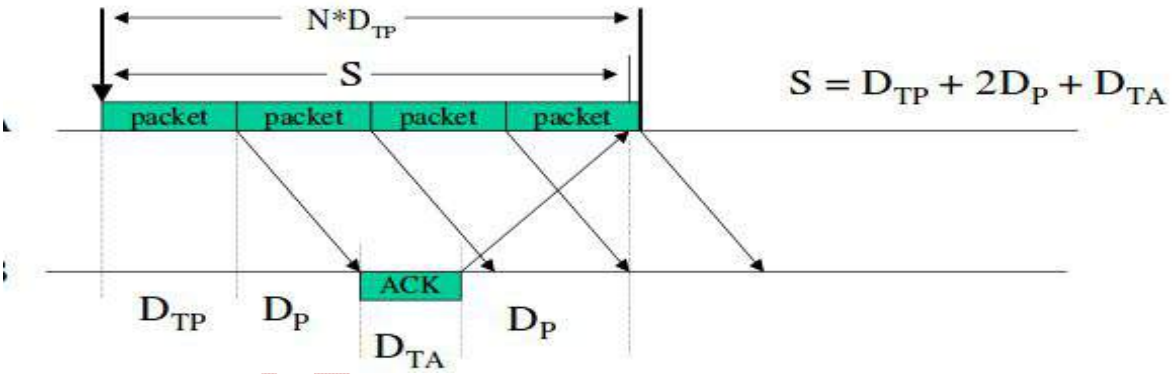
- How Selective Repeat works?
- Efficiency vs buffer tradeoff

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



Efficiency without Errors
(Courtesy: Eytan Modiano



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

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

END

TOPIC 119

RTT Estimation and Timeout

In this module

We shall understand

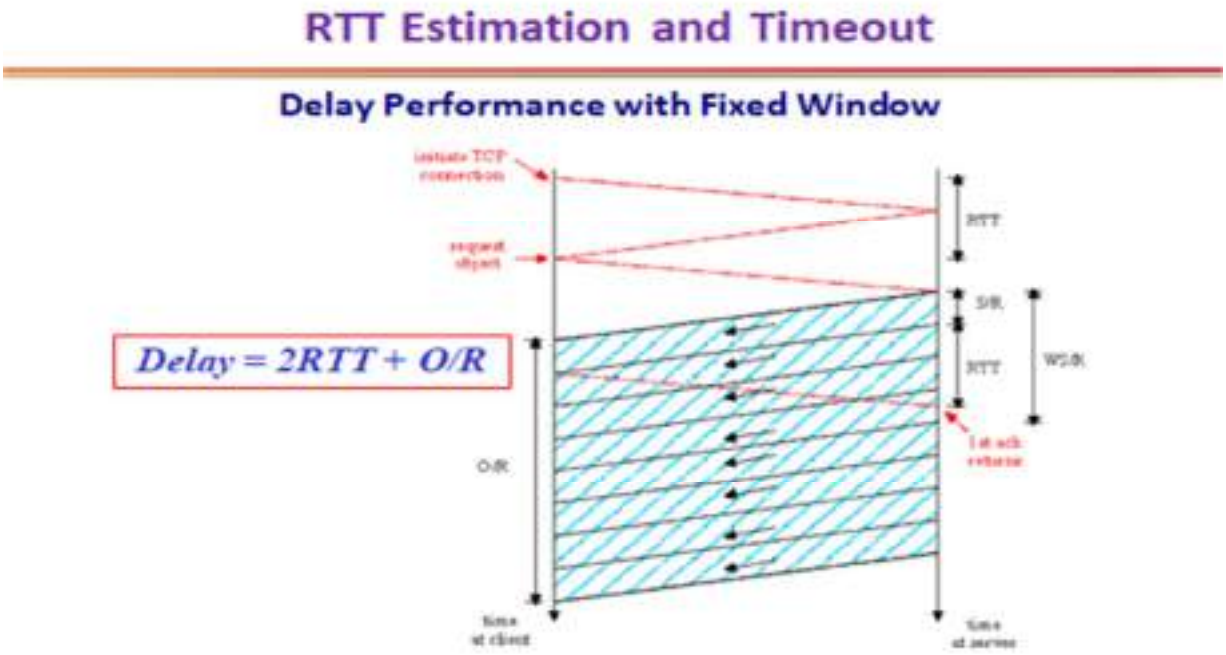
- Measuring RTT
- TCP TimeOut
- What is RTT estimation?

Fixed Window

First case

WS/R > RTT + S/R

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



Fixed Window

Second case

WS/R < RTT + S/R

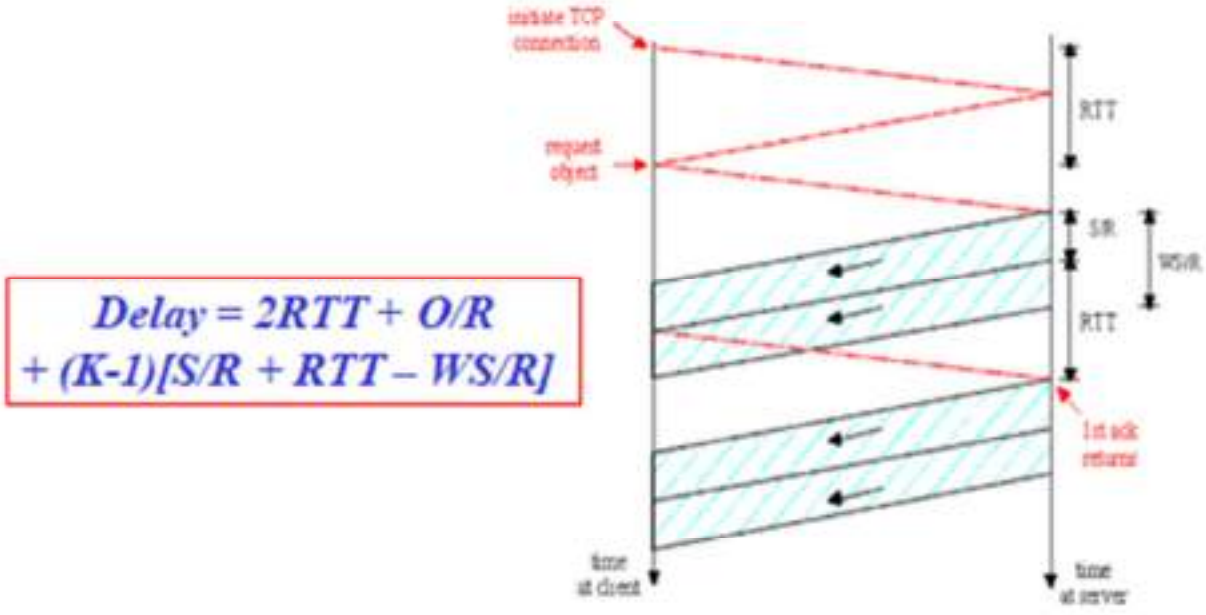
- Wait for ACK after sending window's worth of data sent

K is the number of windows that cover the object

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

RTT Estimation and Timeout

Delay Performance with Fixed Window



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

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

END

TOPIC 120

Reliable Data Transfer

In this module

We shall understand

- Data reliability
- Relationship between throughput and loss
- TCP reliability operations
- 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

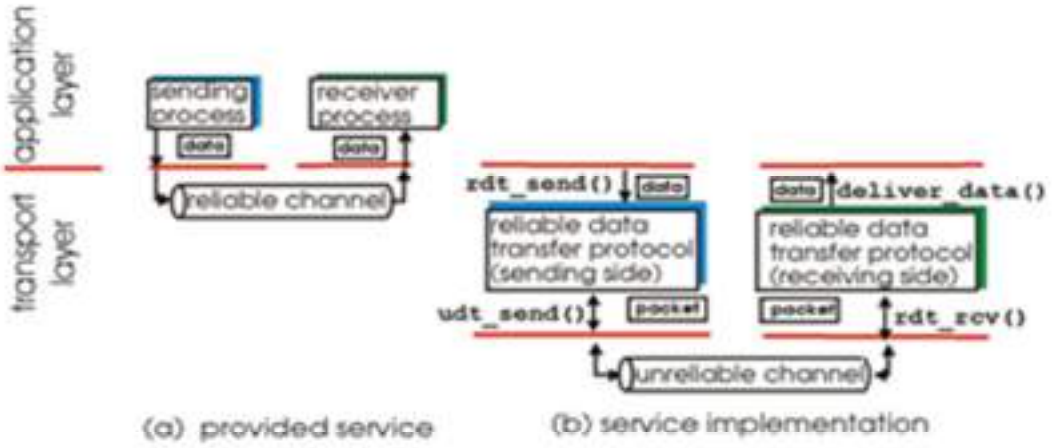
$$\frac{1.22 \cdot MSS}{RTT\ \sqrt{L}}$$

Fast recovery and fast retransmission

Reliable Data Transfer

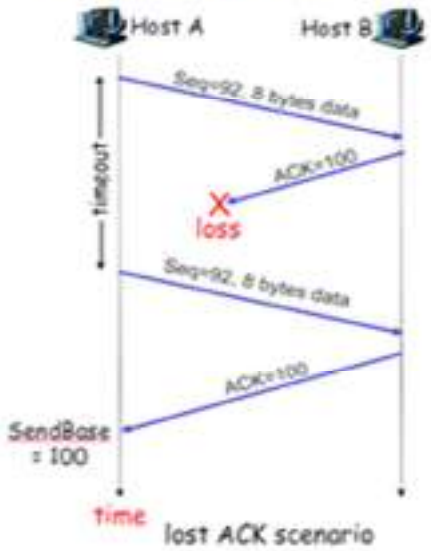
Reliability Services

(Kurose and Ross)



Reliable Data Transfer

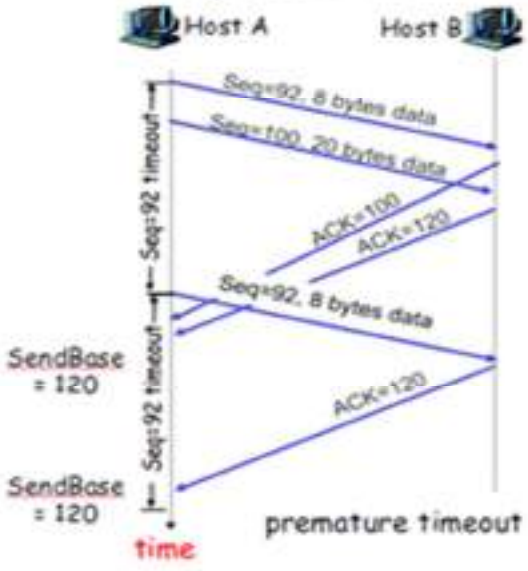
Handling Loss



lost ACK scenario

Reliable Data Transfer

Early TimeOut



premature timeout

END

TOPIC 121

Flow Control

In this module

We shall understand

- What is flow control in TCP?
- Advertised (receive) window
- Sender window
- Effective window

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 (1 of 2)

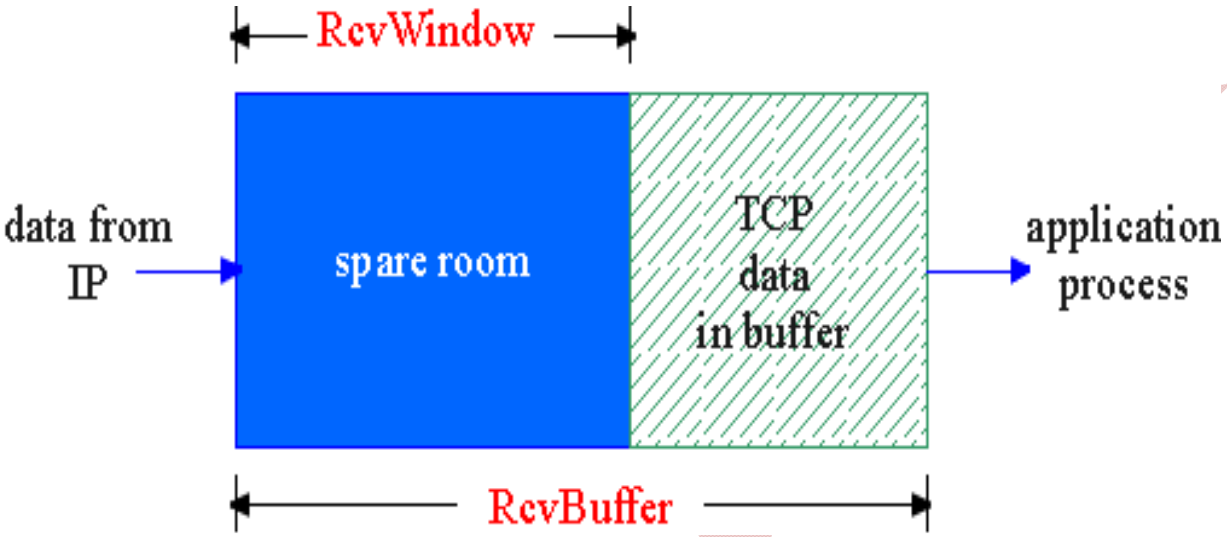
- If local process reads data just as fast as it arrives

- Causes LastByteRead to be incremented at the same rate as LastByteRcvd
- Advertised window stays open

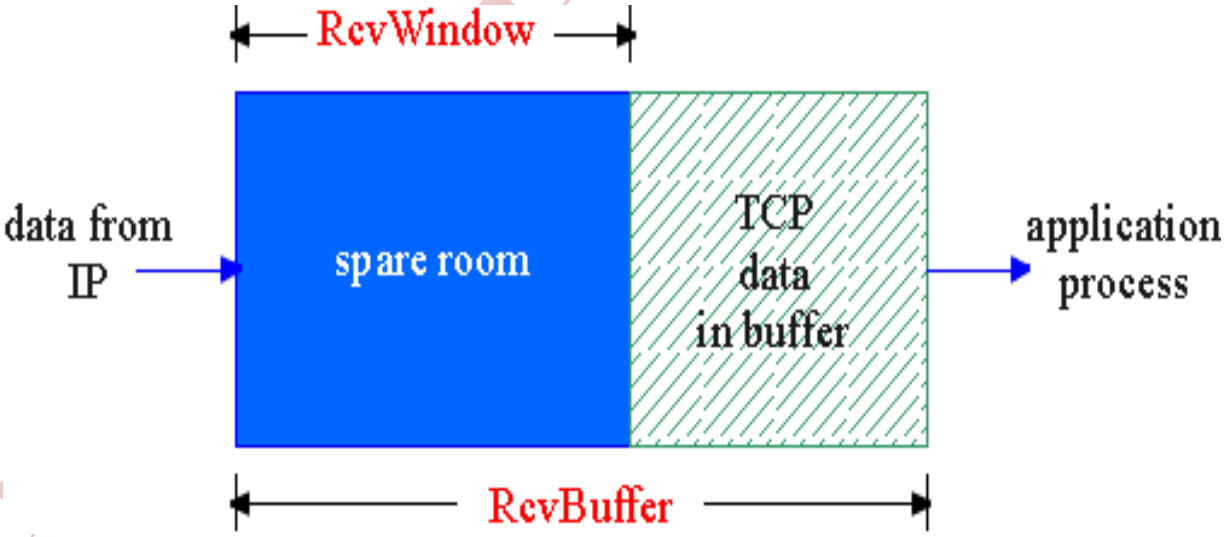
AdvertisedWindow = MaxRcvBuffer)

- If receiving process falls behind, advertised window grows smaller with every segment that arrives, until it eventually goes to 0

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



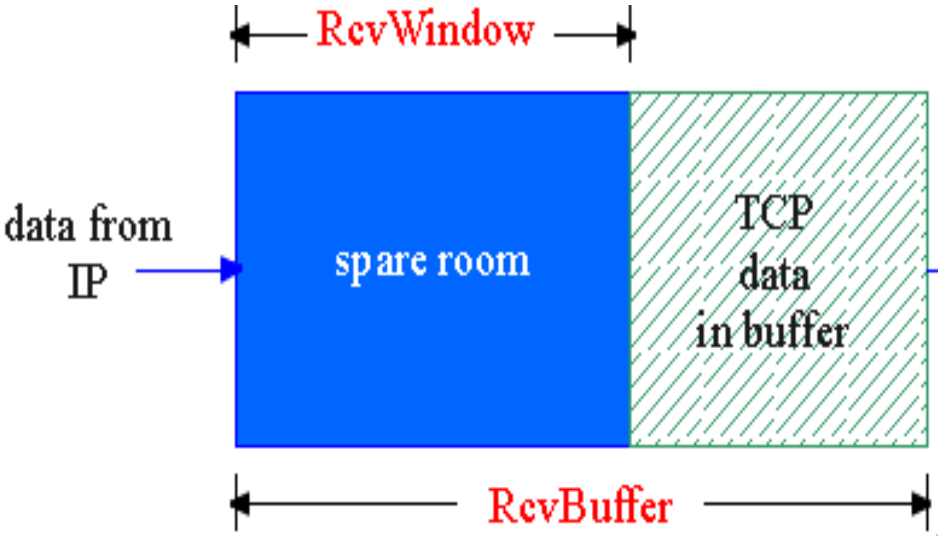
$$LastByteSent - LastByteAcked \leq AdvertisedWindow$$



Data send kia ja rha h

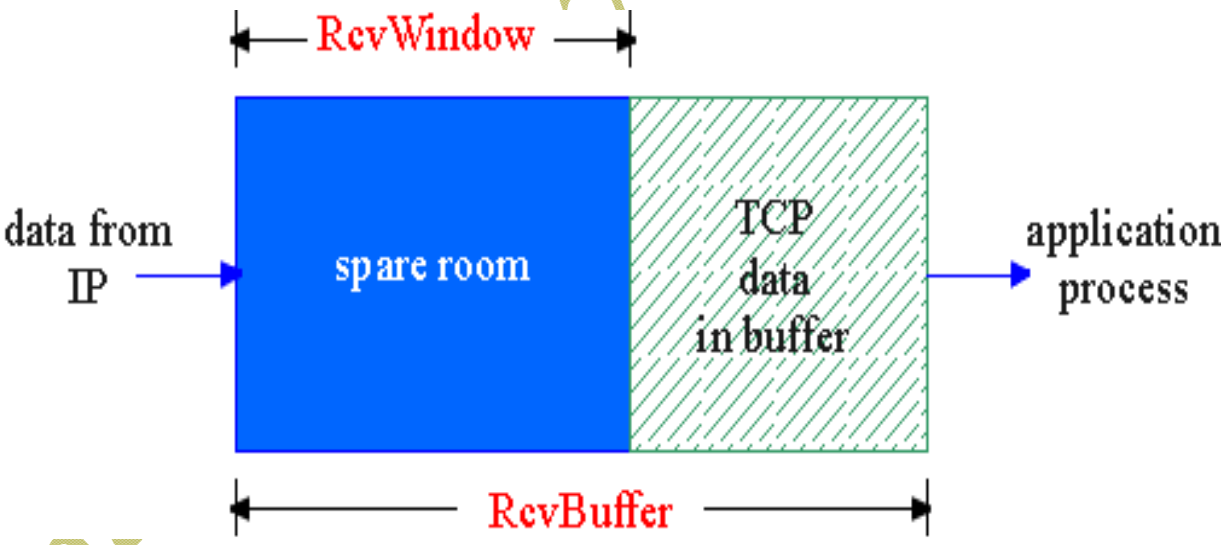
maximum data jo send kia ja skta h

$$\text{EffectiveWindow} = \text{AdvertisedWindow} - (\text{LastByteSent} - \text{LastByteAcked})$$



Relationship between Max_Send and Max_Receive Buffer

$$\text{LastByteWritten} - \text{LastByteAcked} \leq \text{MaxSendBuffer}$$



END

TOPIC 122

TCP Connection Management

In this module

We shall understand

- TCP connection management viz HTTP 1.1
- State Transition Diagrams
- What to model?

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”.

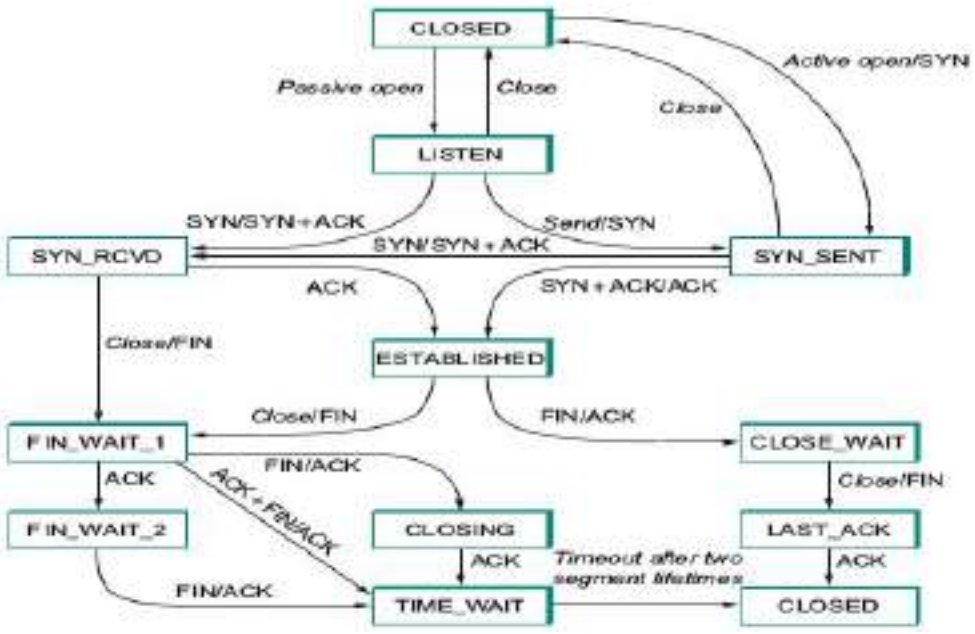
Holding Time (1 of 2)

- 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

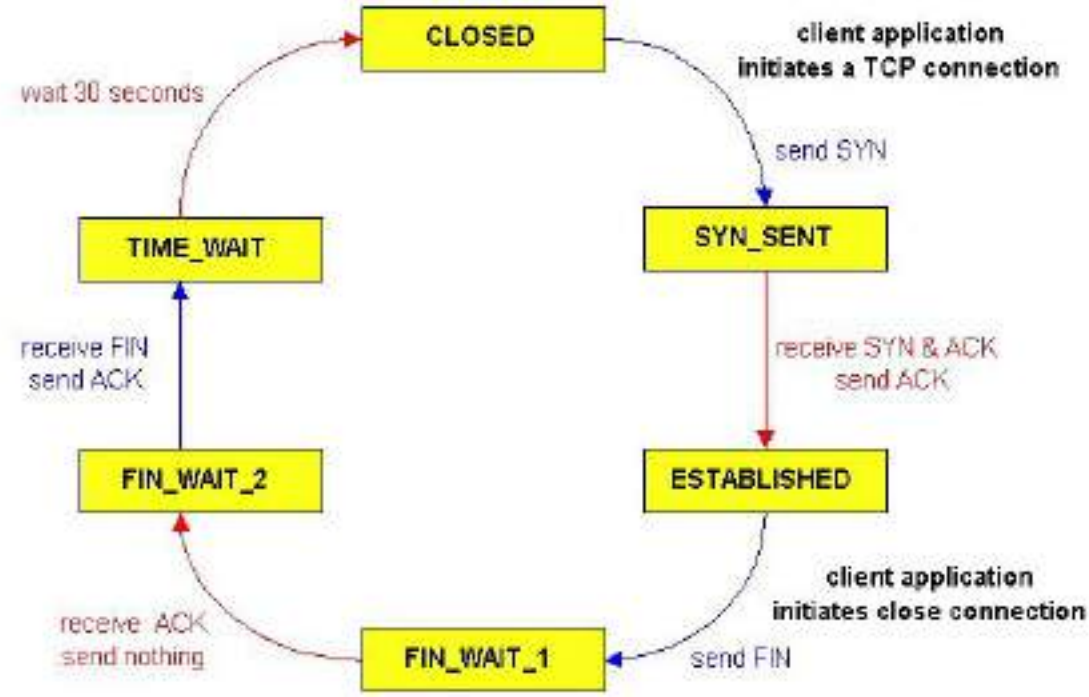
Holding Time (2 of 2)

- 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



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

Open connection

END

Network Modeling and simulation
Made by Mahjabeen
mahjabeen97869@gmail.com
contact # 0321 2711298

TOPIC 123

Principles of Congestion Control

In this module

We shall understand

- What is congestion?
- Effects of buffer
- Combat strategies

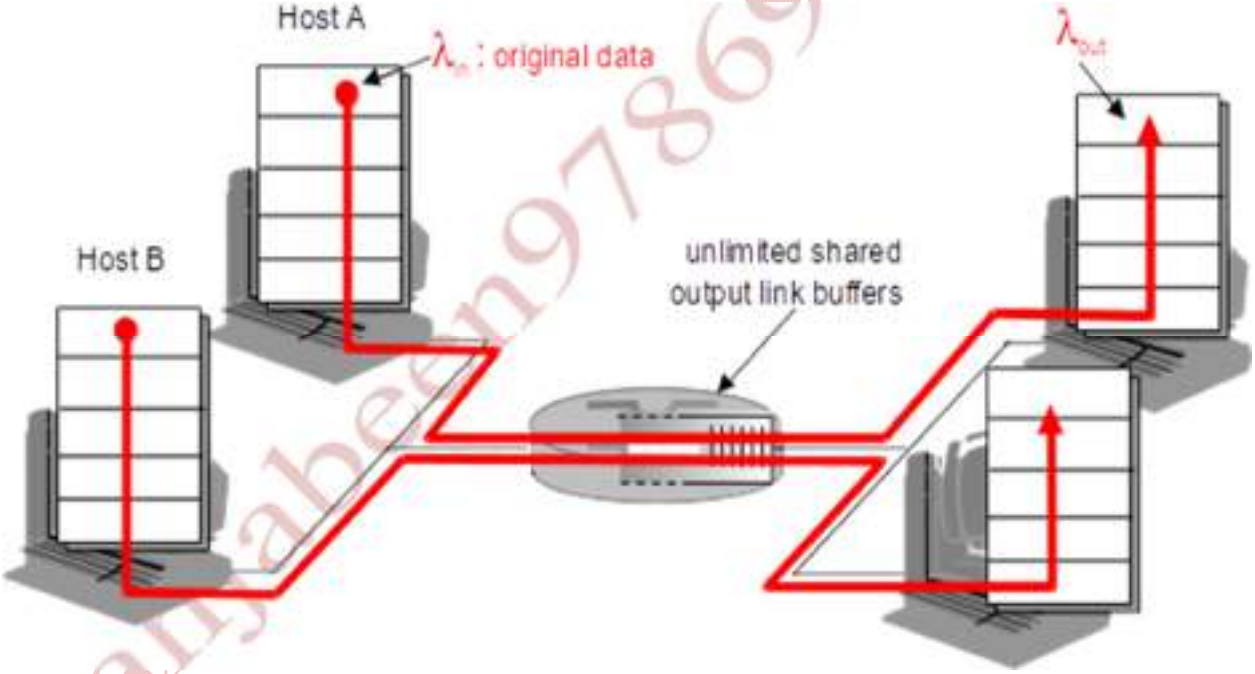
REFRENCES

RFC 2914: Congestion Control Principles
Kurose and Ross. “Computer Networking Top-Down Approach Featuring the Internet

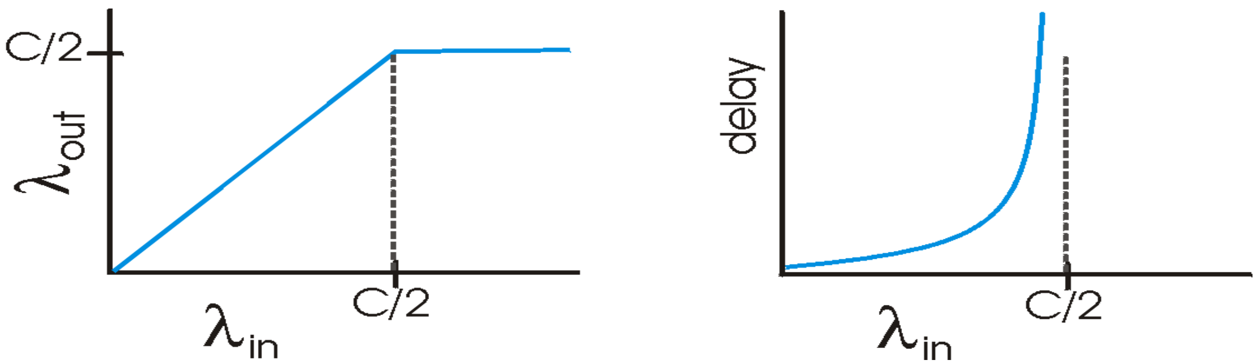
Introduction

- 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

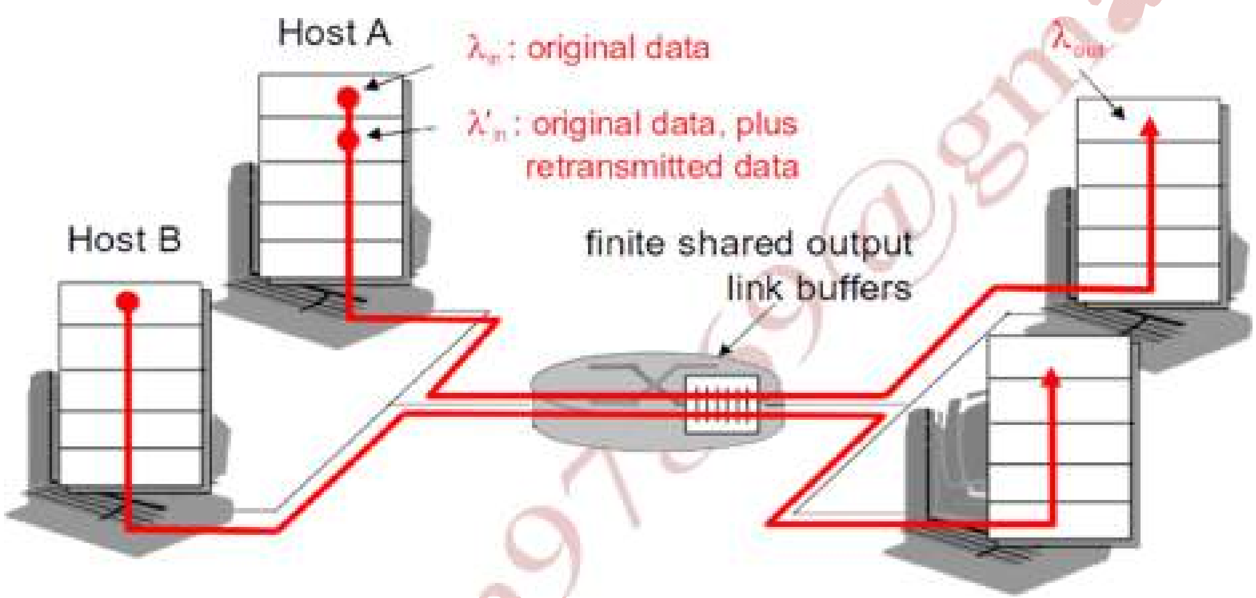


Effects of Congestion

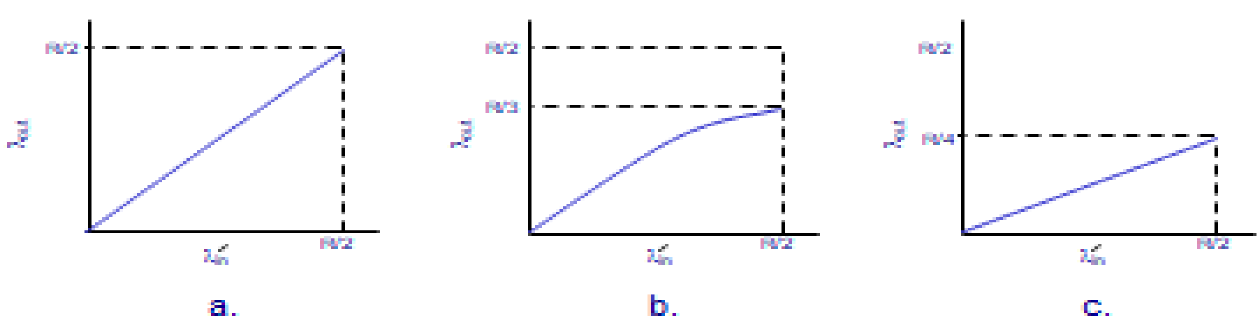


- Large delays when congested
- Maximum achievable throughput

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

Combat Strategies (2 of 2)

Network-assisted congestion control

- Routers provide feedback to end systems
- Single bit indicating congestion (SNA, DECbit, TCP/IP ECN, ATM)
- Explicit rate sender should send at

END

TOPIC 124

ATM ABR Congestion Control

In this module

We shall understand

- Available Bit Rate
- Operation
- Combat strategies

REFRENCE

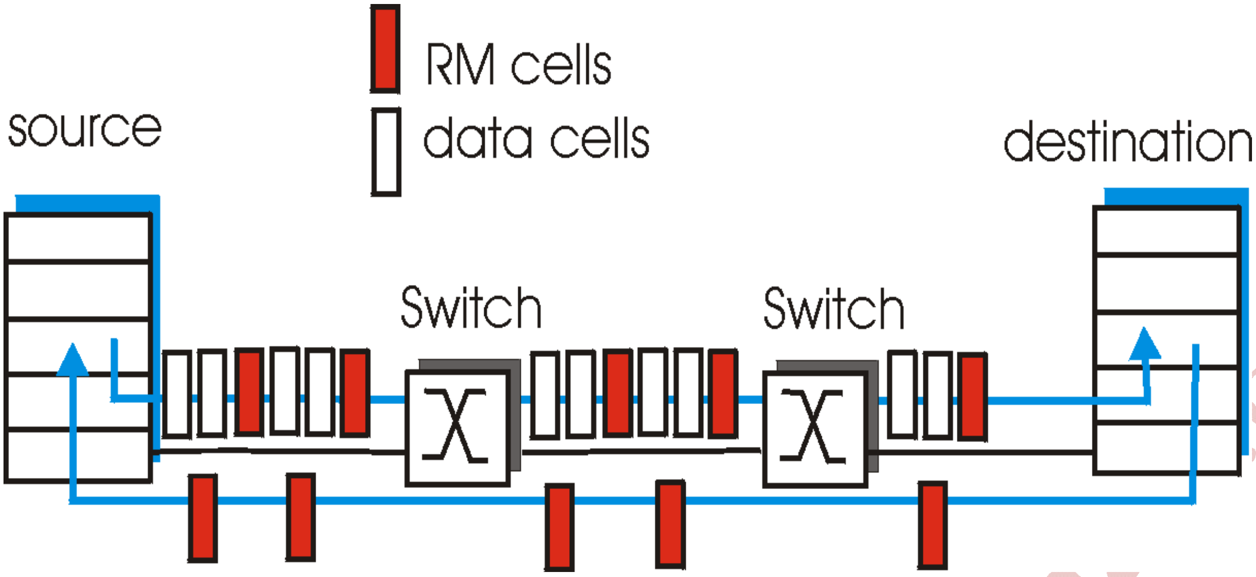
Kurose and Ross. “Computer Networking Top-Down Approach Featuring the Internet”.

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 underloaded
  - Use available bandwidth
- If sender’s path congested
  - Sender throttled to minimum guaranteed rate



Combat Congestion (1 of 2)

- 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

Combat Congestion (2 of 2)

- 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

END

TOPIC 125

TCP Congestion Control

In this module

We shall understand

- TCP Window operation
- Combating loss events
- Rate as congestion window

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

Introduction

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

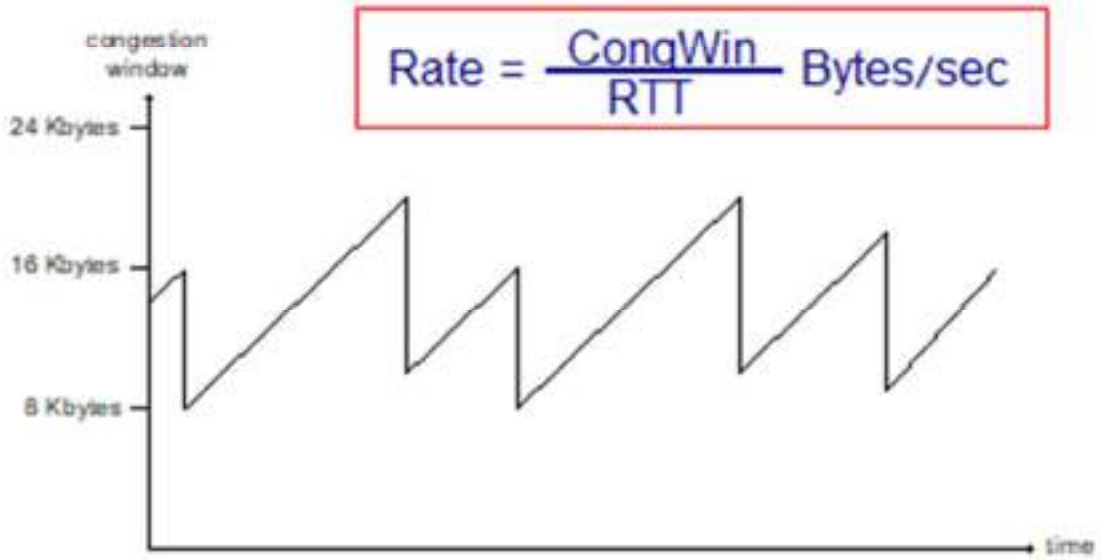
LastByteSent-LastByteAcked
<= CongWin

- 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

TCP Congestion Control



END

TOPIC 126

Leaky Bucket and Token Bucket

In this module

We shall understand

- Leaky bucket
- Operation
- Token bucket
- Operation

References

Regis J. Bates, Broadband Telecommunications Handbook, McGraw-Hill 2000.

Stallings, William. High-speed networks and internets: performance and quality of service. Pearson Education India, 2002.

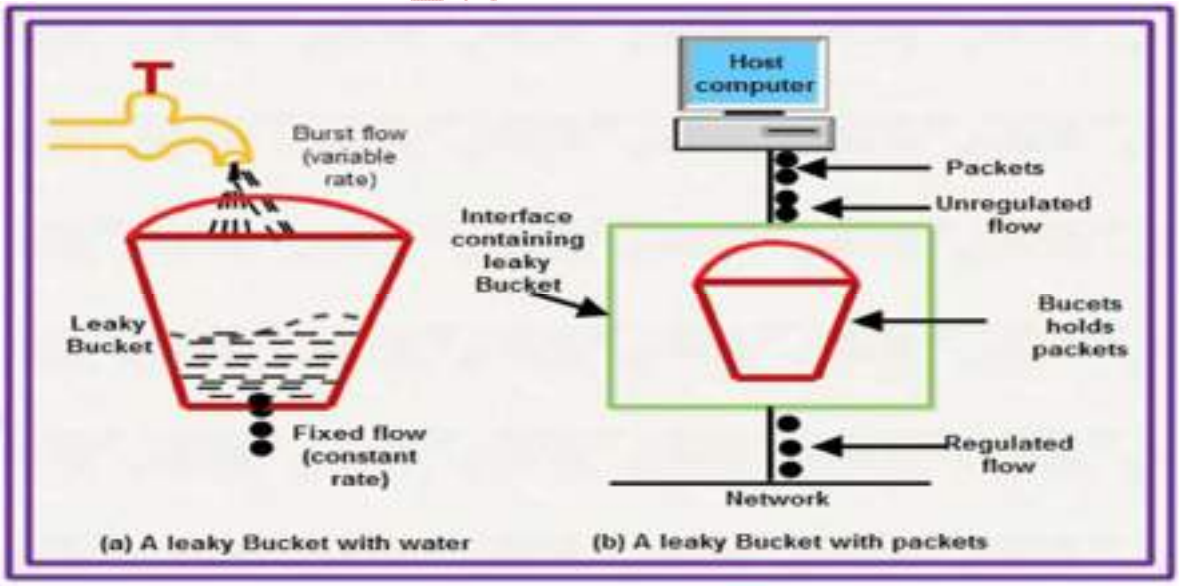
Leaky Bucket

- 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 (1 of 2)

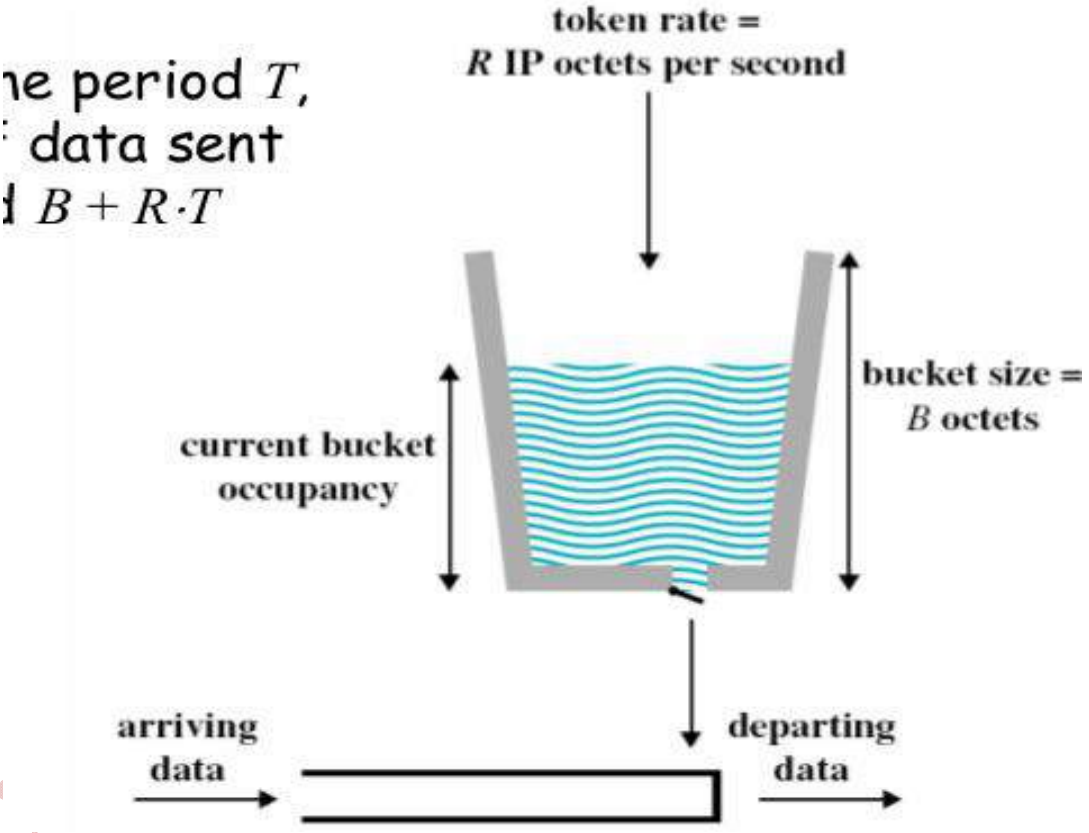
- Provides input parameters to policing function
- IP packet may be processed if sufficient octet tokens to match the IP data number of tokens
- If insufficient tokens available, the packet is relegated to best-effort service

Operation (2 of 2)

- 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

Token Bucket Algorithm

During any time period *T*,
the amount of data sent
cannot exceed
$B + R*T$



END

TOPIC 127

Quality of Service

In this module

We shall understand

- Definition of QoS
- Implications on various layers
- QoS Models
- Hard vs soft QoS

References

Kurose and Ross. "Computer Networking Top-Down Approach Featuring the Internet" 2012.
Haojin Wang. "Packet Broadband Network Handbook" McGraw-Hill 2002

Background (1 of 2)

- Broadband IP packet networks are multiservice, all-purpose communications platforms
- Spurred QoS efforts

Background (2 of 2)

- 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 (1 of 2)

- 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

Introduction (2 of 2)

- Traffic agreements
- Traffic control
- Connection admission control
- Congestion notification
- Fragmentation

QoS at Network Layer (1 of 2)

- 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

QoS at Network Layer (2 of 2)

- 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 (1 of 2)

- 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

QoS Models (2 of 2)

- 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

END

TOPIC 128

Fair Queues

In this module

We shall understand

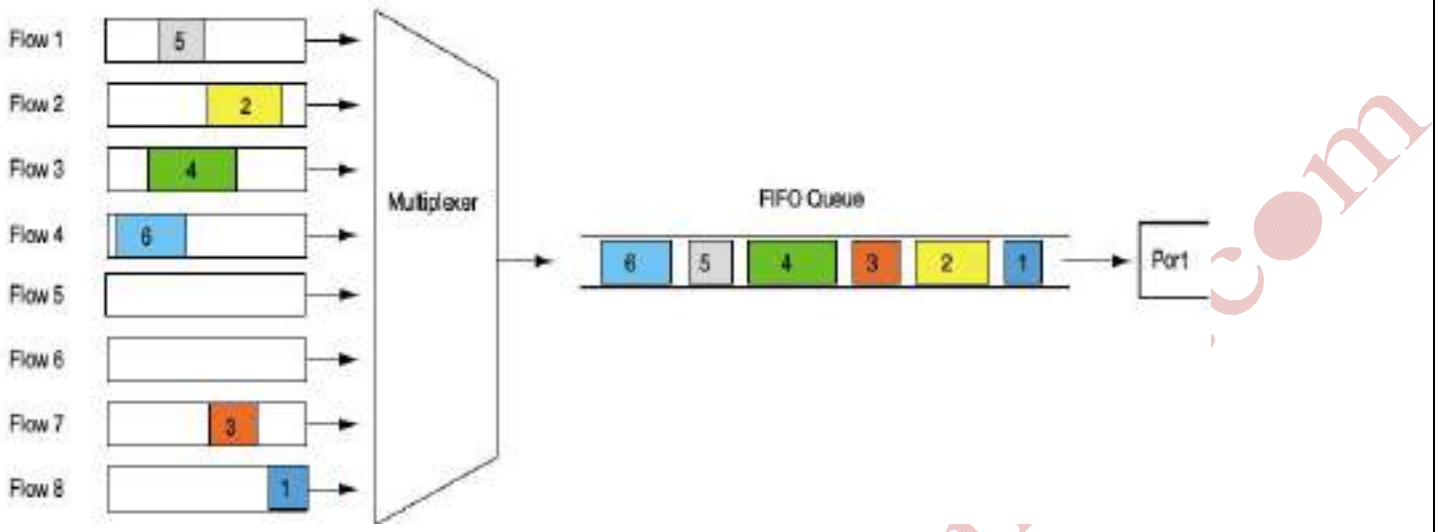
- FIFO
- Motivations for FQs
- Operation
- Benefits

References

Chuck Semeria, "Supporting Differentiated Service Classes: Queue Scheduling Disciplines," Juniper Networks, White Paper 2001.

CS432 Handouts Made by
Mahjabeen
mahjabeen97869@gmail.com
contact # 0321 2711298

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

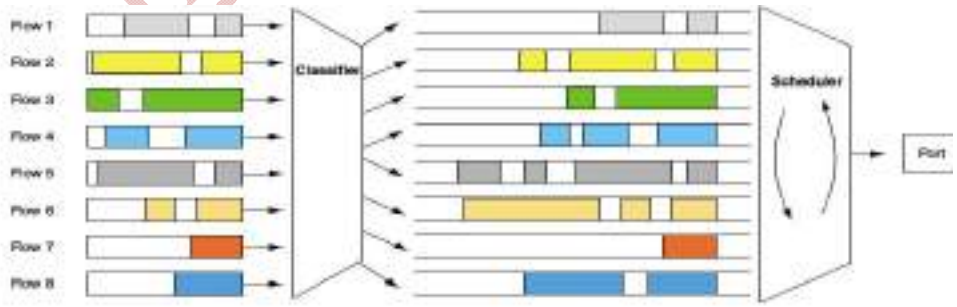
share

- **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 (1 of 3)

- 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

Performance (2 of 3)

Conditions:

- 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
- $\mu_{Total} \ll \mu_{Total} - \mu_i$

Performance (3 of 3)

- Conditions:
- $\mu_i = Min(\mu_{Fair} - \rho_i)$
- Above condition remains recursively true as we remove the minimal user & reduce total resource

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

END

WEEK10

TOPIC 129

Priority Queues

In this module

We shall understand

- **Motivations for PQs**
- **Operation**

- **Variants**
- **Delay Bounds**

References

Chuck Semeria, "Supporting Differentiated Service Classes: Queue Scheduling Disciplines," Juniper Networks, White Paper 2001.

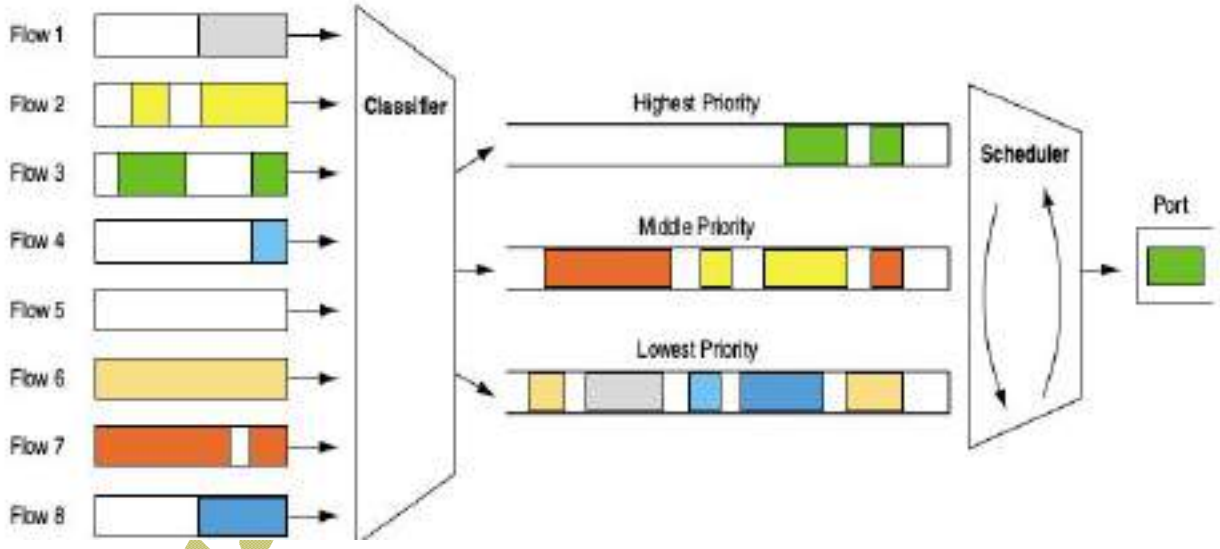
Motivation

- Designed to provide a relatively simple method of supporting differentiated service classes
- To provide respective services to PQs
  - 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 =

$$T_i^k = \sum_{j=1}^{i-1} T_{r_j} + T_{s_i} \qquad (2)$$

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 (1 of 2)

- Strict priority queuing
  - packets in a high-priority queue are always scheduled before packets in lower-priority queues

Variants (2 of 2)

- 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

END

TOPIC 130

Static Window Modeling

In this module

We shall understand

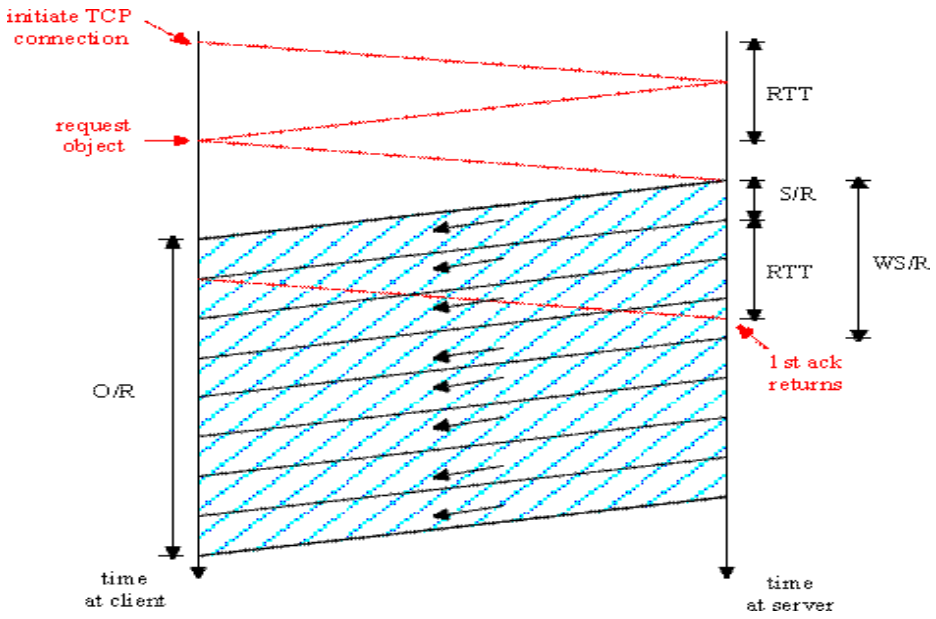
- What is static window?
- Fixed window variants

- **References**
- Kurose, J., and K. Ross. "A top-Down Approach." Computer Networking (2010):

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 Static window



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

Operation (2 of 2)

- . 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

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

END

TOPIC 131

Static Window Modeling—2

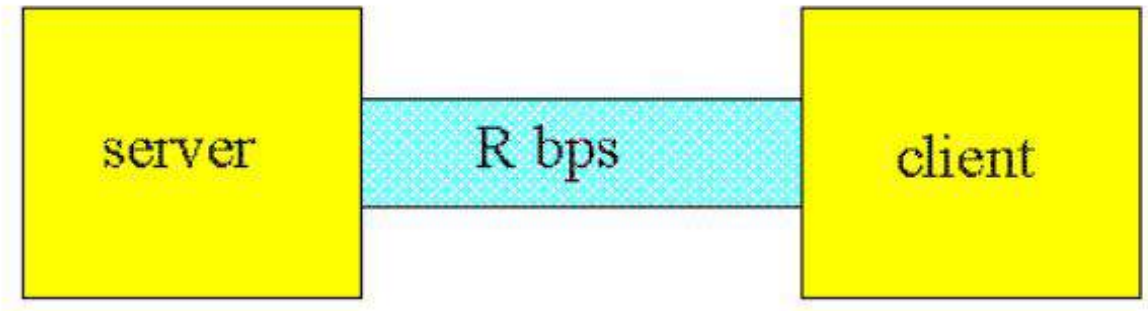
In this module

We shall understand

- What is static window?
- Fixed window variants

- **References**
- Kurose, J., and K. Ross. "A top-Down Approach." Computer Networking (2010):

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



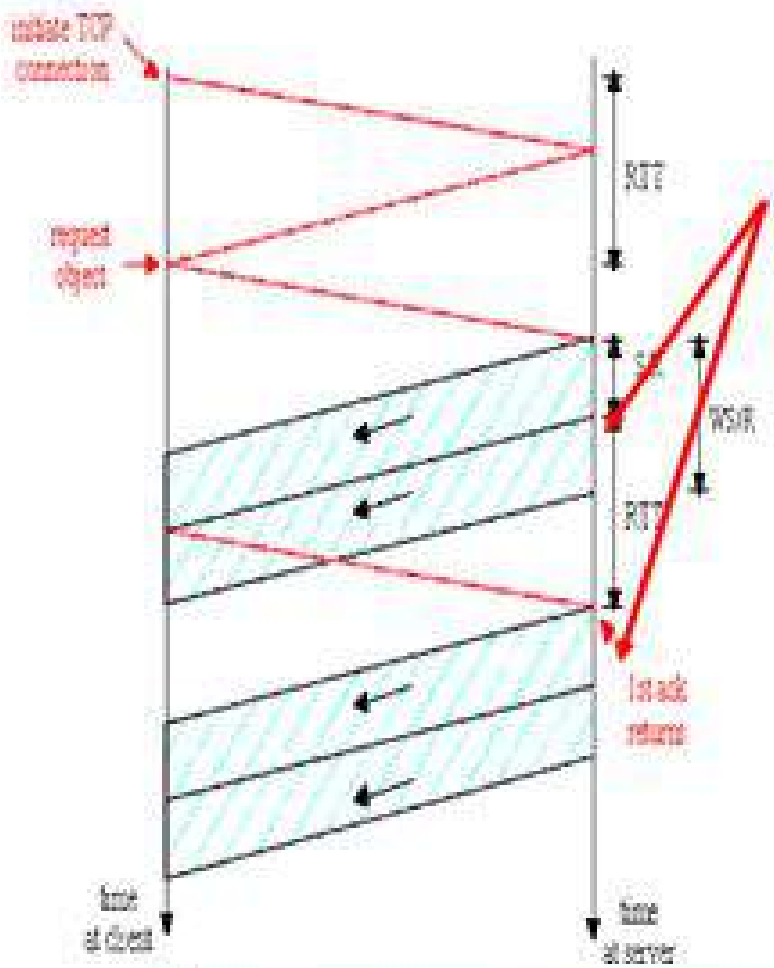
Second Case

- 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

Static Window Modeling—2



K is the number of windows that cover the object

$$\text{Delay} = 2RTT + O/R + (K-1)[S/R + RTT - WS/R]$$

END

TOPIC 132

Dynamic Window Modeling

In this module

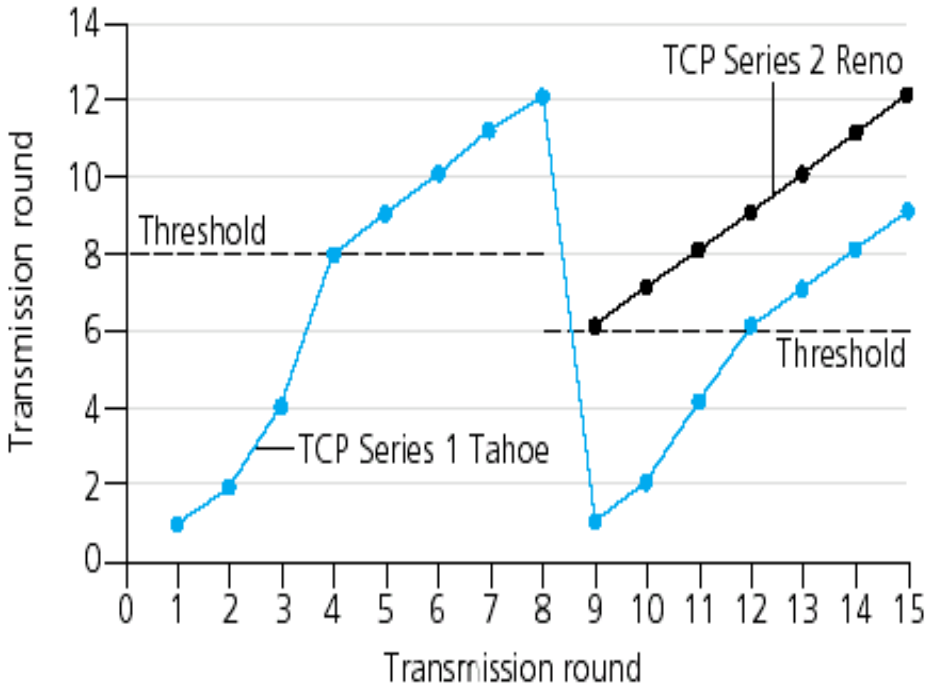
We shall investigate

- TCP congestion control dynamics
- Slow start modeling

References

Kurose, J., and K. Ross. "A top-Down Approach." Computer Networking (2010):

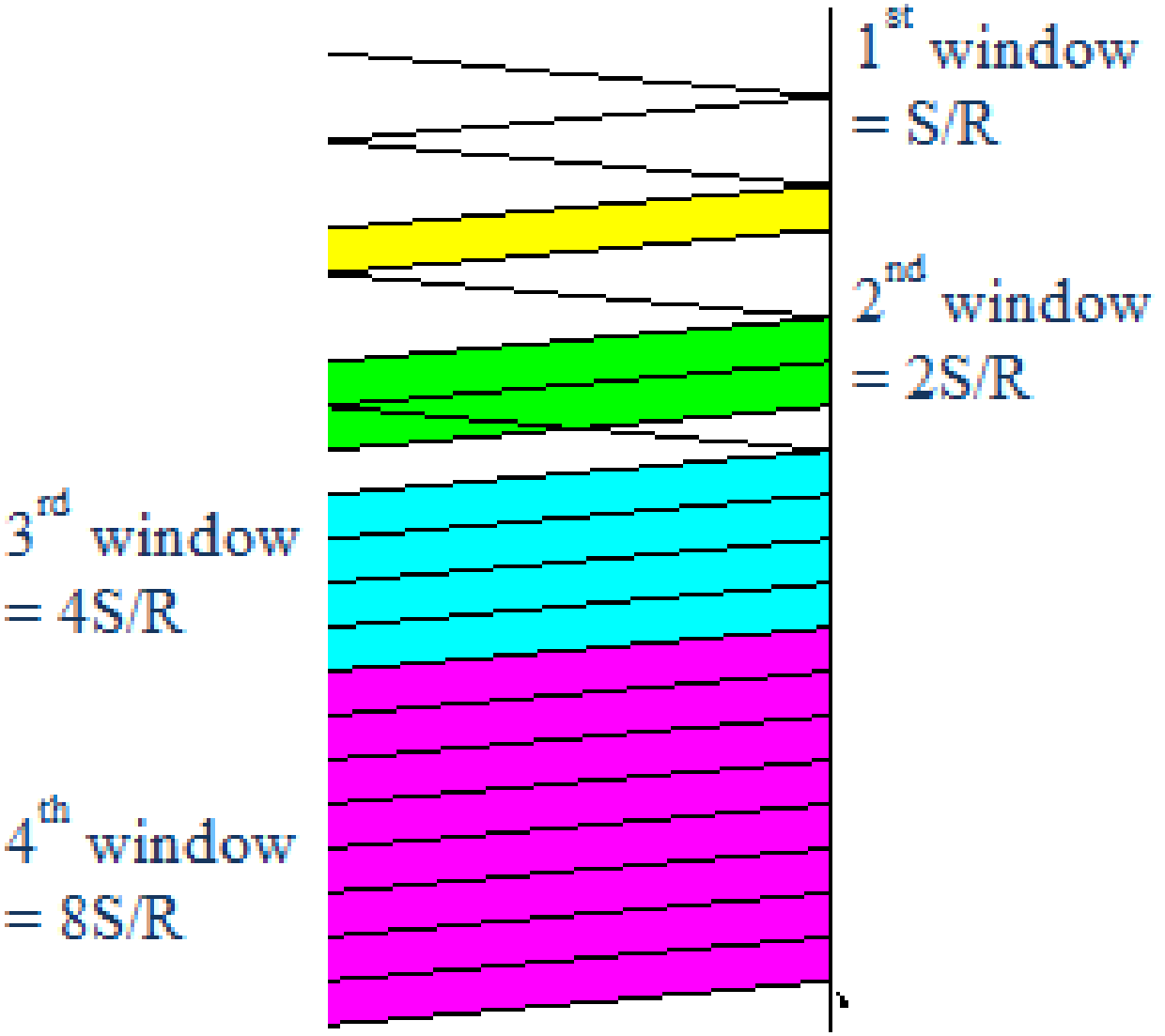
TCP Congestion Dynamics



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

Dynamic Window Modeling



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

Dynamic Window Modeling

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

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

$$\frac{S}{R} + RTT - 2^{k-1}\frac{S}{R}\Big]^{+} = \text{idle time after the } k\text{th window}$$

Example

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

4-1,Q
3,2

minimum value is 2

Server idles P=2 times

END

End-to-End Windows

In this module

We shall investigate

- Limitations of end-to-end windows
- Effect of throughput and delay

References

Bertsekas, Dimitri P., Robert G. Gallager, and Pierre Humblet. Data networks. Vol. 2. New Jersey: Prentice-Hall International, 1992.

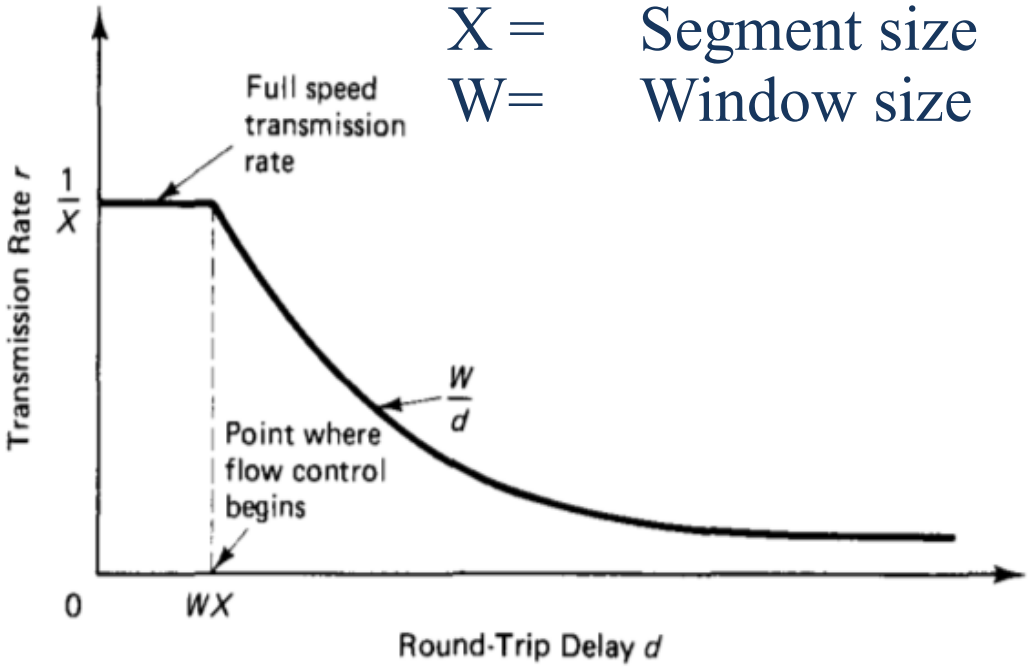
Limitations

- 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

X = Segment size

W= Window size



END

TOPIC 134

Node-by-Node Windows

In this module

We shall investigate

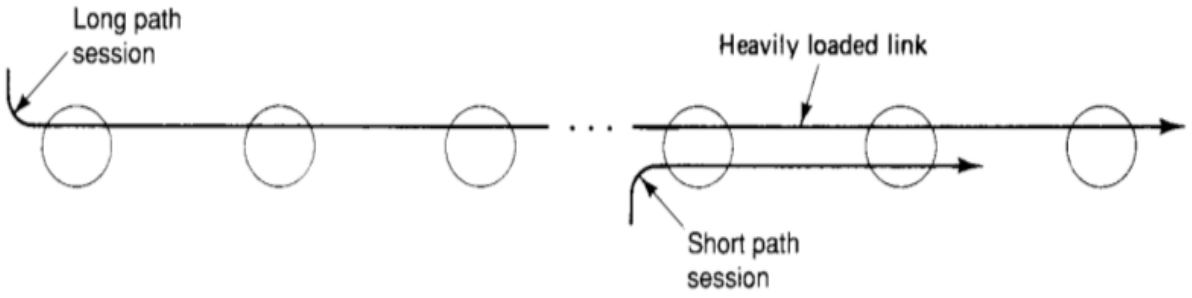
Overcoming limitations of end-to-end flow control

- Windowing for virtual circuits

References

Bertsekas, Dimitri P., Robert G. Gallager, and Pierre Humblet. Data networks. Vol. 2. New Jersey: Prentice-Hall International, 1992.

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

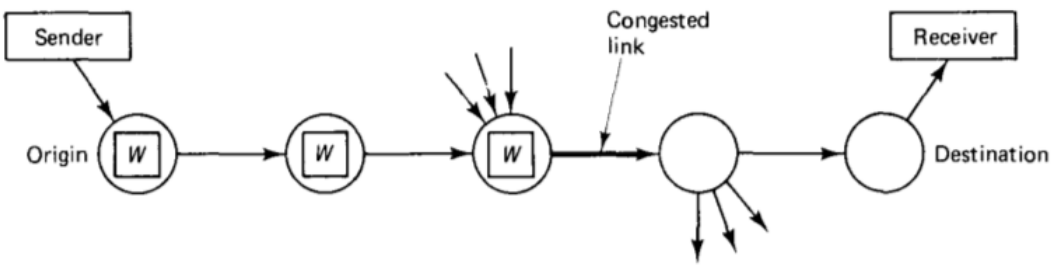
Main idea

- Receiver avoids accumulation of large no. of packets into its memory
  - Slows down permit returns to sender

Backpressure Effect in VCs

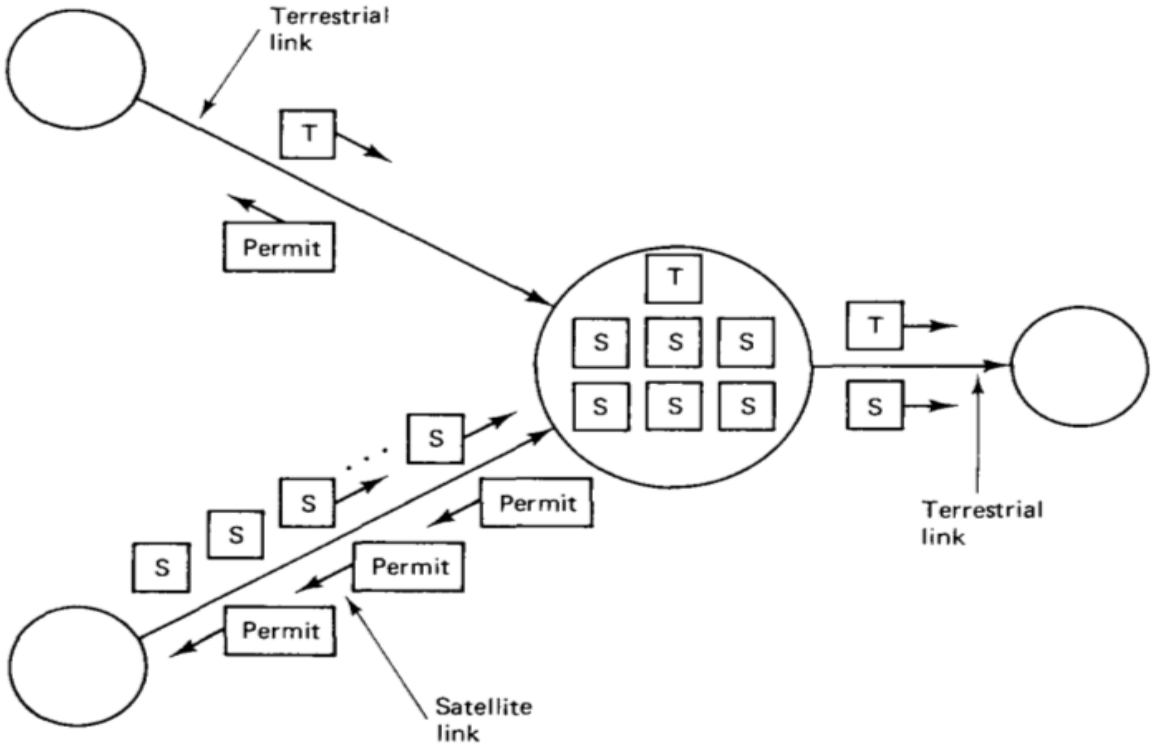
N = No. of nodes along VC

W= Window size



Round Robin + Node-by-node

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



END

TOPIC 135

Little's Theorem

In this module

We shall understand

- The big questions
- Definition of Little's Theorem
- Intuitive example

References

Bertsekas, Dimitri P., Robert G. Gallager, and Pierre Humblet. Data networks. Vol. 2. New Jersey: Prentice-Hall International, 1992.

John D. C. Little, A Proof for the Queuing Formula: L =λW. Operations Research 9(3):383-387 (1961)

Big Questions (1 of 2)

- What is the avg no. of customers in the system?

- The "typical" no. of packets either waiting in queue or undergoing service

Big Questions (2 of 2)

- What is the avg delay per customer?
- The "typical" time a packet spends waiting in queue plus the service time

Definition

$N = \lambda \times T$

N = No. of customers

λ = 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.

END

TOPIC 136

Probabilistic Little's Theorem

In this module

We shall understand

- Time average
- Statistical average
- Probabilistic interpretation

References

Bertsekas, Dimitri P., Robert G. Gallager, and Pierre Humblet. Data networks. Vol. 2. New Jersey: Prentice-Hall International, 1992.

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

END

TOPIC 137

Little's Theorem; Applications

In this module

We shall understand

- Network model as a Little's theorem application
- Interpretation of results
- **References**
- Bertsekas, Dimitri P., Robert G. Gallager, and Pierre Humblet. Data networks. Vol. 2. New Jersey: Prentice-Hall International, 1992.

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

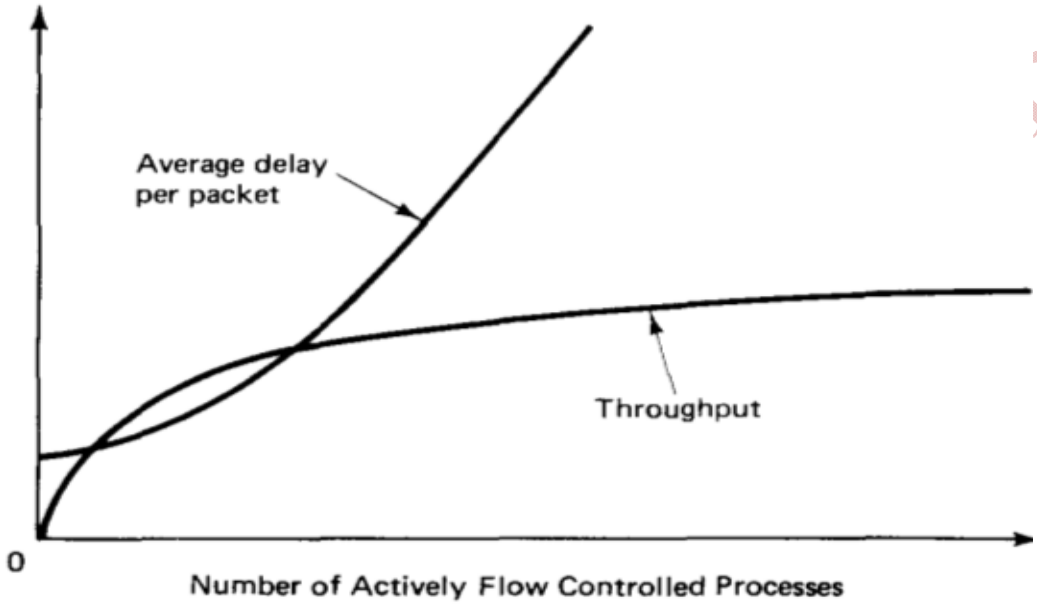
Average delay per packet (1 of 2)

- n flow controlled sessions in the network with fixed window sizes $W_1,...Wn$
- b = whether piggybacking supported or not
- l = throughput (total accepted input rate of sessions)

Average delay per packet (2 of 2)

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

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



END

TOPIC 138

Arrivals as Poisson
In this module
We shall understand

- M/M/1 systems
- Arrivals
- Poisson Process
- Distribution

References
Bertsekas, Dimitri P., Robert G. Gallager, and Pierre Humblet. Data networks. Vol. 2. New Jersey: Prentice-Hall International, 1992.

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 $1/\mu$ 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 (1 of 3)

- A Poisson process is generally considered to be a good model for the aggregate traffic of a large number of
  - Similar and
  - Independent users

Poisson Process (2 of 3)

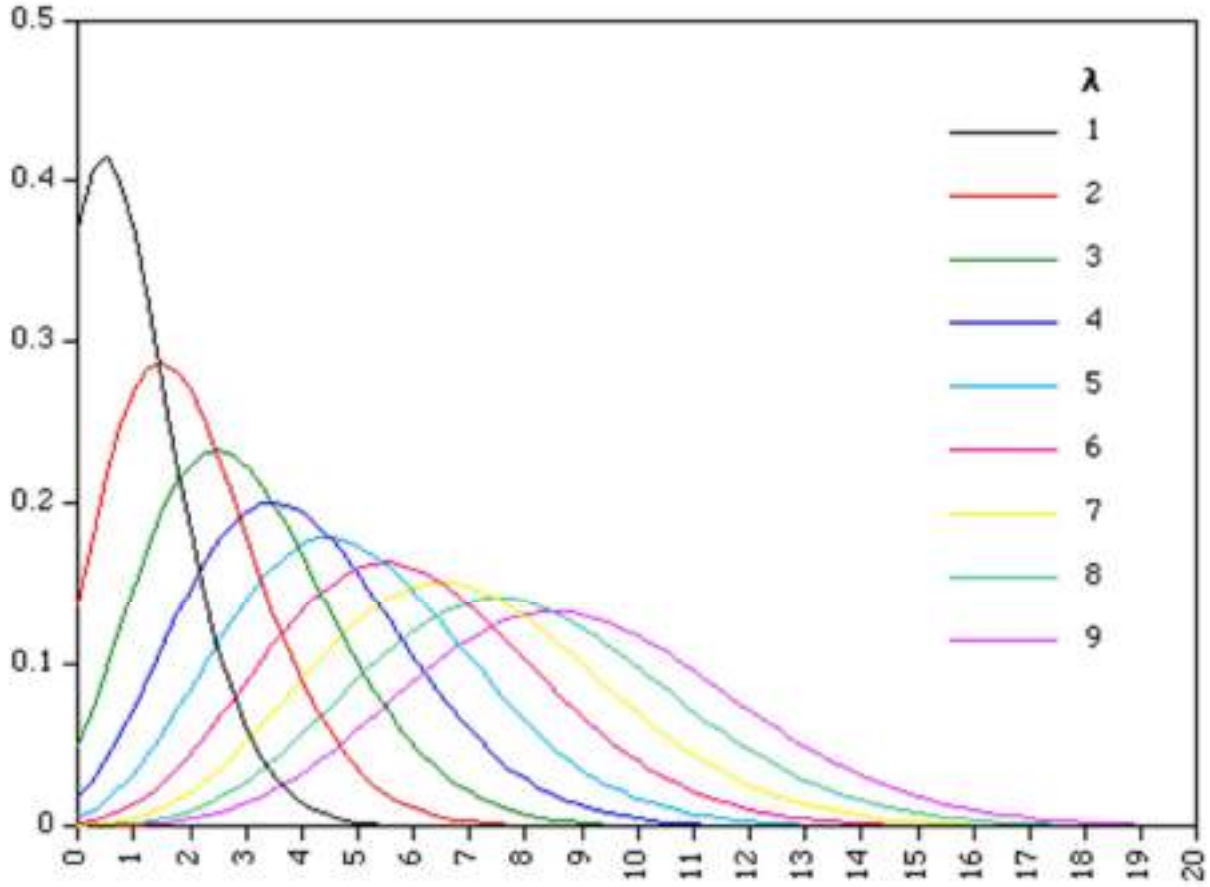
- Merges *n* independent & identically distributed arrival processes
- Each process has arrival rate λ/n
- So the aggregate process has arrival rate λ

Poisson Process (3 of 3)

- No. of arrivals occurring in disjoint time intervals are independent
- No. of arrivals in any interval of length t is Poisson distributed with parameter λt

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

Poisson Distribution



END

TOPIC 139

Service Statistics

In this module

We shall understand

- What is service?
- Service distribution
- Exponential distribution
- **References**
- Bertsekas, Dimitri P., Robert G. Gallager, and Pierre Humblet. Data networks. Vol. 2. New Jersey: Prentice-Hall International, 1992.

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 (1 of 3)

- S_n is the service time of the nth customer
- Customer (packet) service times have an exponential distribution with parameter m

Service distribution (2 of 3)

- m is also called service rate
- Represents the rate (in customers served per unit time) at which the server operates when busy

Service distribution (3 of 3)

- Service times are mutually independent
- Also independent of all inter-arrival times
- Density function
- Service distribution

$$p(s_n) = \mu e^{-\mu s_n}$$

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

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

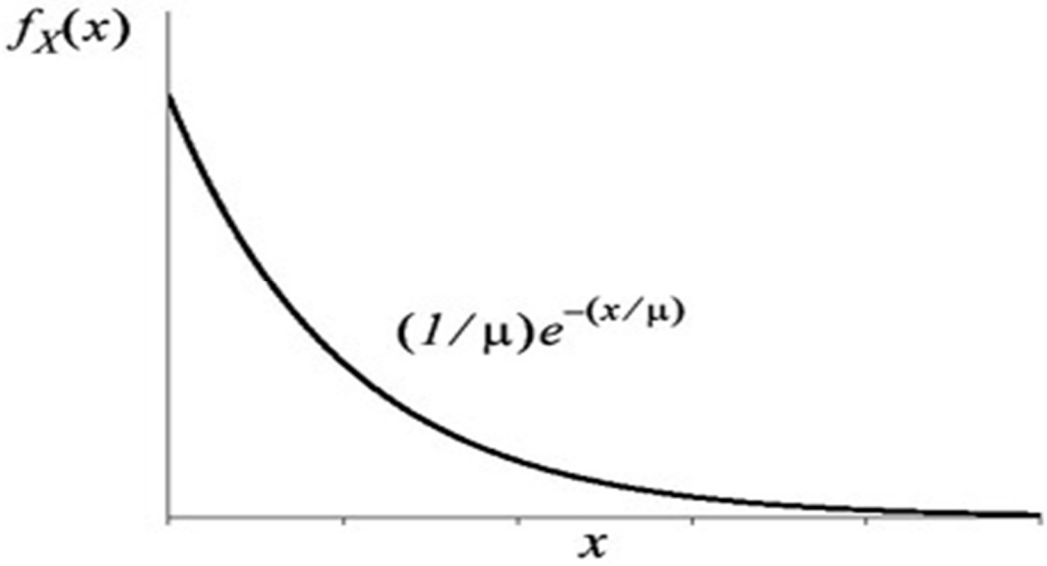
Commentary

- 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



END

TOPIC 140

Arrival Occupancy Distribution

In this module

We shall understand

- System under change
- Non-typical arrival
- Typical arrival
- Occupancy distribution

References

Bertsekas, Dimitri P., Robert G. Gallager, and Pierre Humblet. Data networks. Vol. 2. New Jersey: Prentice-Hall International, 1992.

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

y-state occupancy probabilities upon arrival,

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

Typical Arrival

responding unconditional steady

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

END

TOPIC 141

Simulating Slow Start

In this module

We shall understand

- Operation of Slow Start
- TCP Reno Support
- INET Class Reference

Operation

- RFC 2581 identifies the operation in the wake of TimeOut
  - Activate Slow Start
  - Bring window down to 1 MSS
  - Start exponential increase

Simulating Slow Start

tcp_old::TCPReno Class Reference

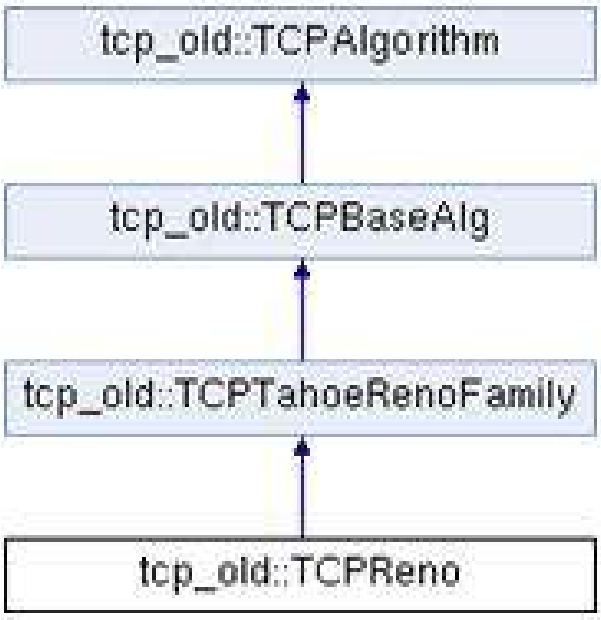
```
void TCPReno::processRexmitTimer     (
TCPEventCode &      event    )   [protected, virtual]
if (event==TCP_E_ABORT)
return;
// begin Slow Start (RFC 2581)
recalculateSlowStartThreshold();
state->snd_cwnd = state->snd_mss;
if (cwndVector) cwndVector->record(state->snd_cwnd);
tcpEV << "Begin Slow Start: resetting cwnd to " << state-
>snd_cwnd << ", ssthresh=" << state->ssthresh << "\n";
```

Simulating Slow Start

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



END

TOPIC 142

Network Service Models

In this module

We shall understand

- Need for service models
- Types of service models

Need

- The network layer is workhorse
- Upper layer necessitates its behaviour
  - — TCP
  - — UDP
  - — ATM
  - — Proprietary
- Corresponding services must exist

Services Models
(1 of 2)

- Services for individual datagrams
  - — Guaranteed delivery
  - — Guaranteed delivery with less than 40 msec delay

Services Models
(2 of 2)

- Services for a flow of datagrams:
  - — In-order datagram delivery
  - — Guaranteed minimum bandwidth to flow
  - — Restrictions on changes in inter-packet spacing

Network Service Models
Types

Network Architecture	Service Model	Guarantees ?				Congestion feedback
		Bandwidth	Loss	Order	Timing	
Internet	best effort	none	no	no	no	no (inferred via loss)
ATM	CBR	constant rate	yes	yes	yes	no congestion
ATM	VBR	guaranteed rate	yes	yes	yes	no congestion
ATM	ABR	guaranteed minimum	no	yes	no	yes
ATM	UBR	none	no	yes	no	no

Network Architecture	Service Model	Guarantees ?				Congestion feedback
		Bandwidth	Loss	Order	Timing	
Internet	best effort	none	no	no	no	no (inferred via loss)
ATM	CBR	constant rate	yes	yes	yes	no congestion
ATM	VBR	guaranteed rate	yes	yes	yes	no congestion
ATM	ABR	guaranteed minimum	no	yes	no	yes
ATM	UBR	none	no	yes	no	no

Network Architecture	Service Model	Guarantees ?				Congestion feedback
		Bandwidth	Loss	Order	Timing	
Internet	best effort	none	no	no	no	no (inferred via loss)
ATM	CBR	constant rate	yes	yes	yes	no congestion
ATM	VBR	guaranteed rate	yes	yes	yes	no congestion
ATM	ABR	guaranteed minimum	no	yes	no	yes
ATM	UBR	none	no	yes	no	no

Network Architecture	Service Model	Guarantees ?				Congestion feedback
		Bandwidth	Loss	Order	Timing	
Internet	best effort	none	no	no	no	no (inferred via loss)
ATM	CBR	constant rate	yes	yes	yes	no congestion
ATM	VBR	guaranteed rate	yes	yes	yes	no congestion
ATM	ABR	guaranteed minimum	no	yes	no	yes
ATM	UBR	none	no	yes	no	no

Network Architecture	Service Model	Guarantees ?				Congestion feedback
		Bandwidth	Loss	Order	Timing	
Internet	best effort	none	no	no	no	no (inferred via loss)
ATM	CBR	constant rate	yes	yes	yes	no congestion
ATM	VBR	guaranteed rate	yes	yes	yes	no congestion
ATM	ABR	guaranteed minimum	no	yes	no	yes
ATM	UBR	none	no	yes	no	no

END

WEEK 11

TOPIC 143

Virtual Circuit Networks

In this module

We shall understand

- What are virtual circuits?
- Path establishment
- Path traversal
- Stability issues

Basics

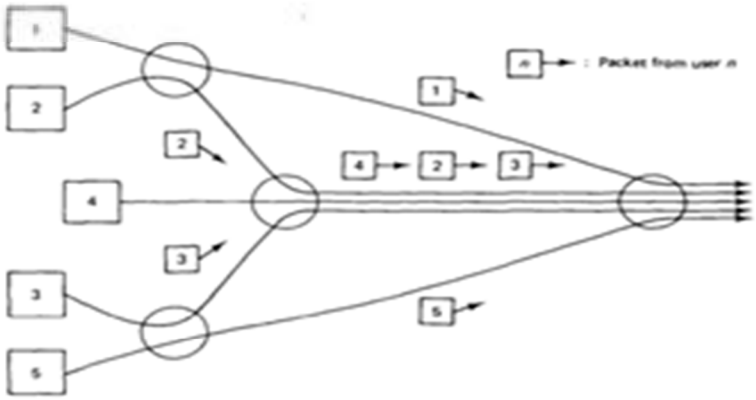
- Source-to-destination paths behave much like telephone circuit
- Performance guaranteed
- Network actions along source-to-dest path needed

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

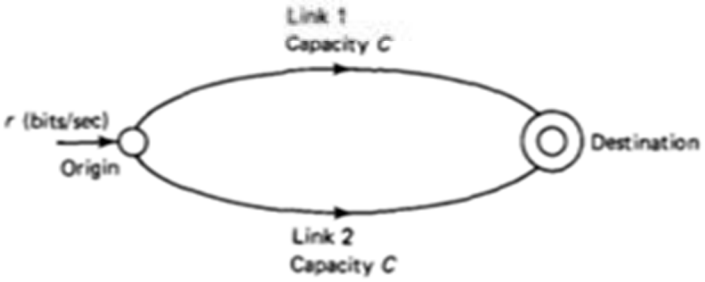
Virtual Circuit Networks

Packets Along the Same Path



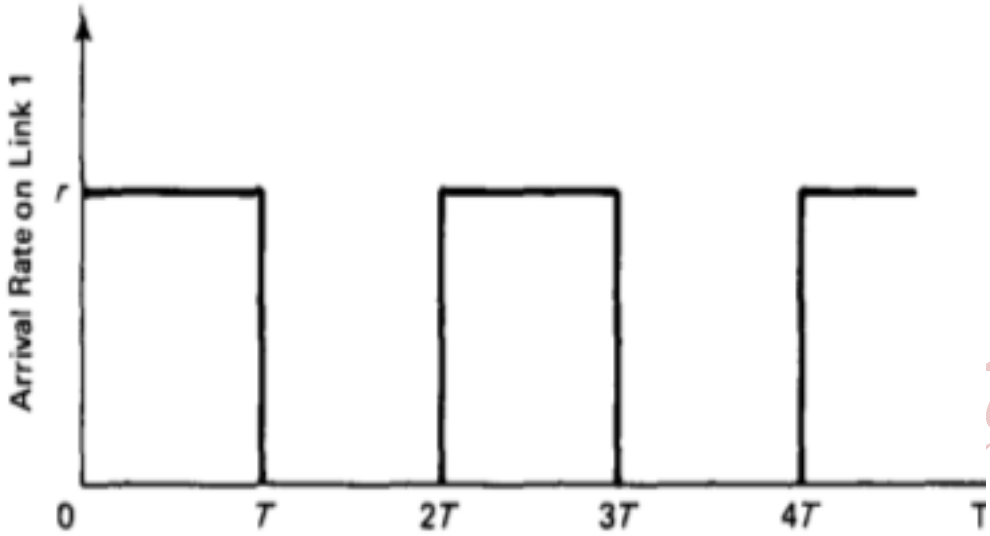
Virtual Circuit Networks

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

END

TOPIC 144

Datagram Networks

In this module

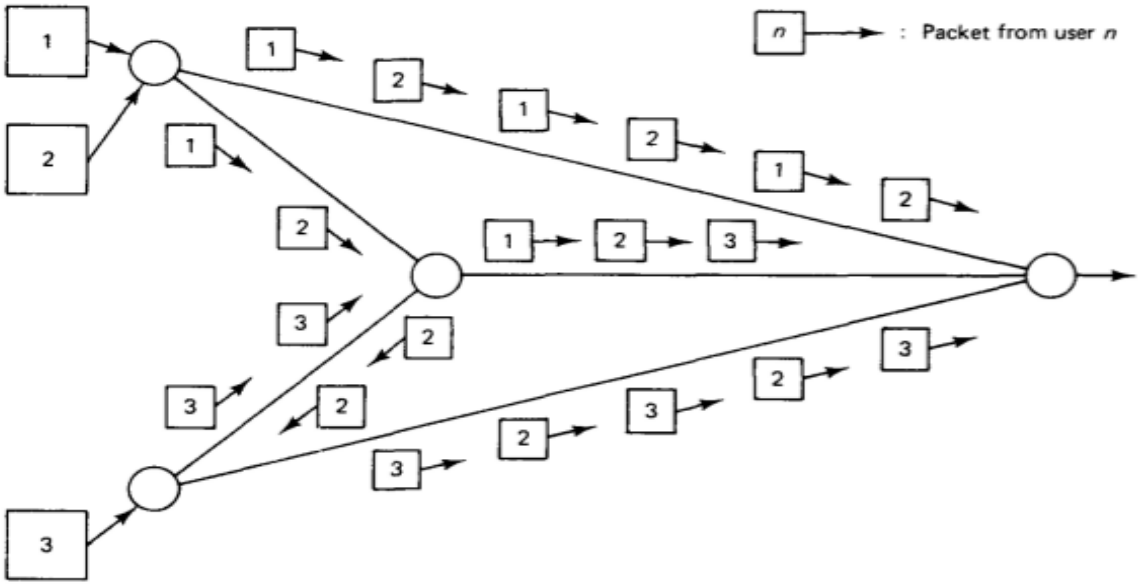
We shall understand

- How datagram networks operate?
- Complexity
- Oscillatory behaviour

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

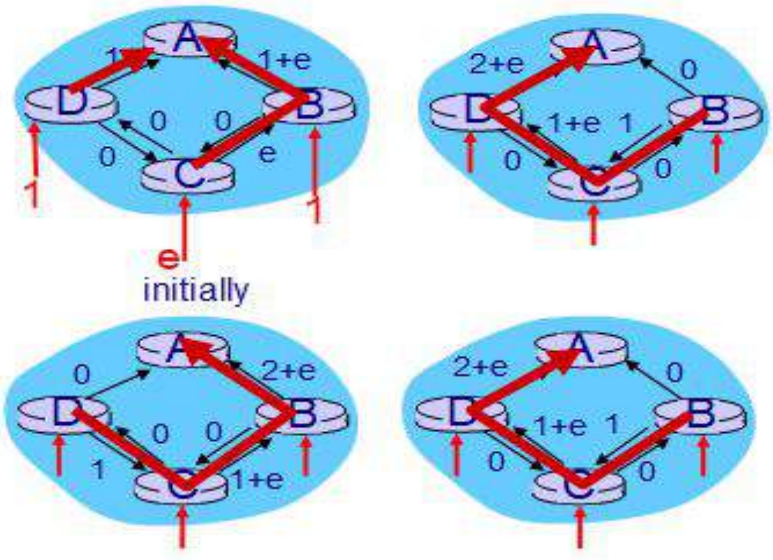


Complexity

- 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



END

TOPIC 145

Input Processing

In this module

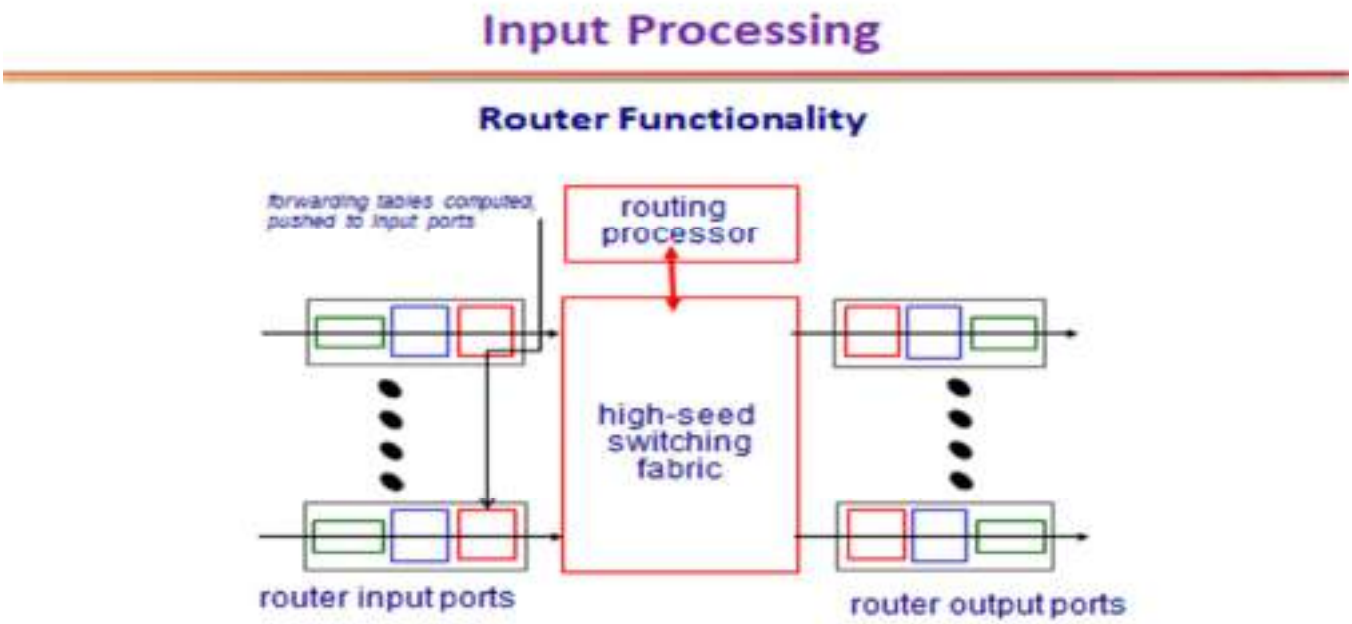
We shall understand

- How routers process information?
- Router input

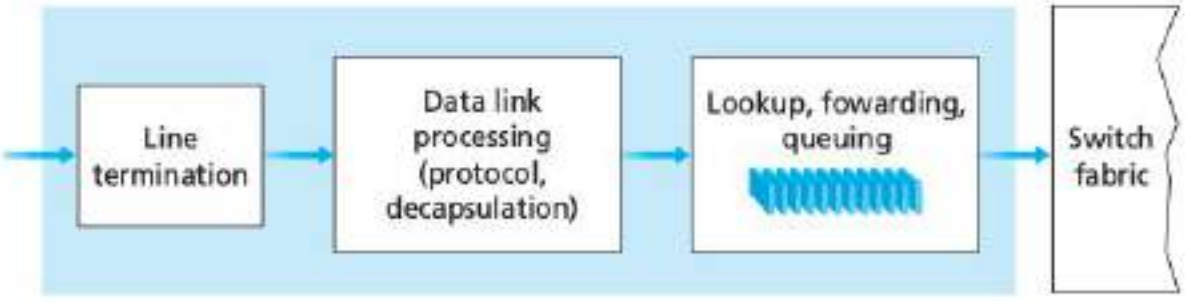
Basics

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

Router Functionality



Router Input



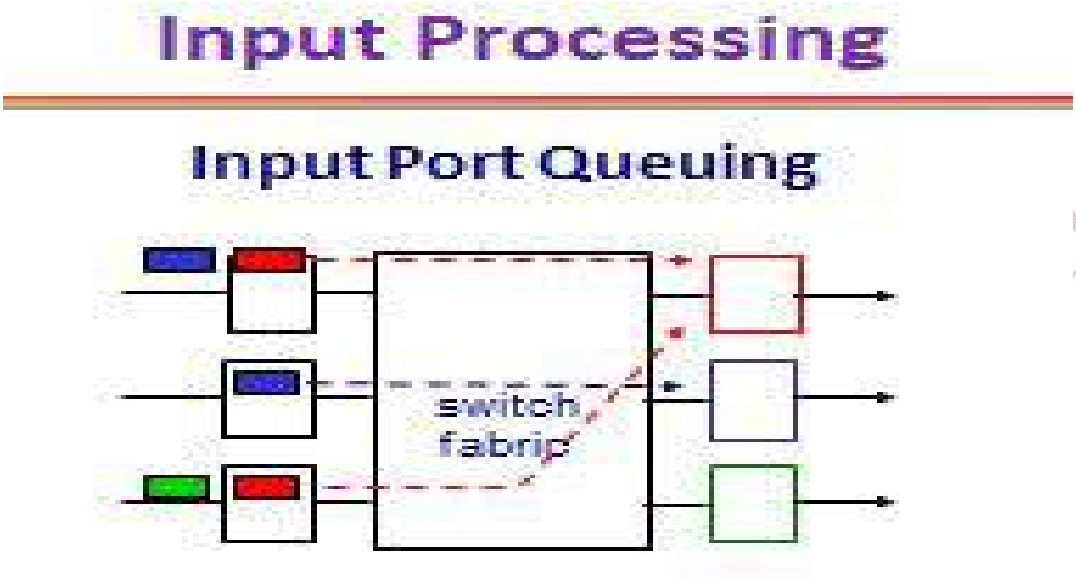
Distributed Switching

- 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 processing



END

TOPIC 146

Output Processing

In this module

We shall understand

- How routers output information?
- Output port buffering
- Buffering load

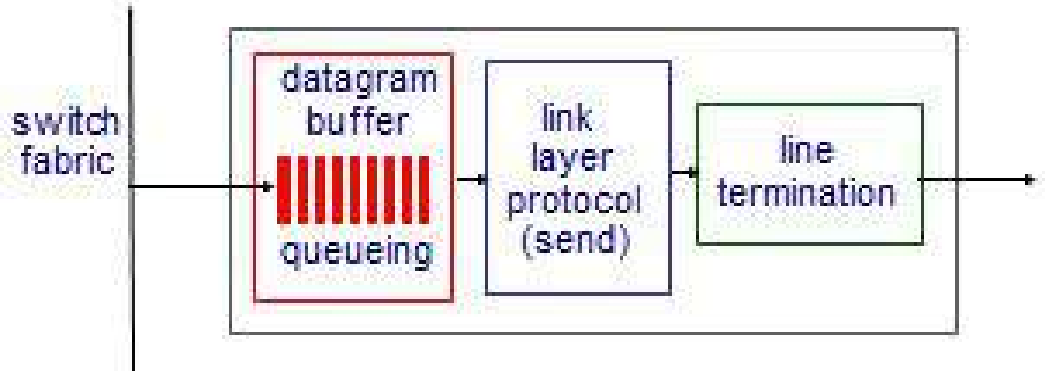
Operations

- 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

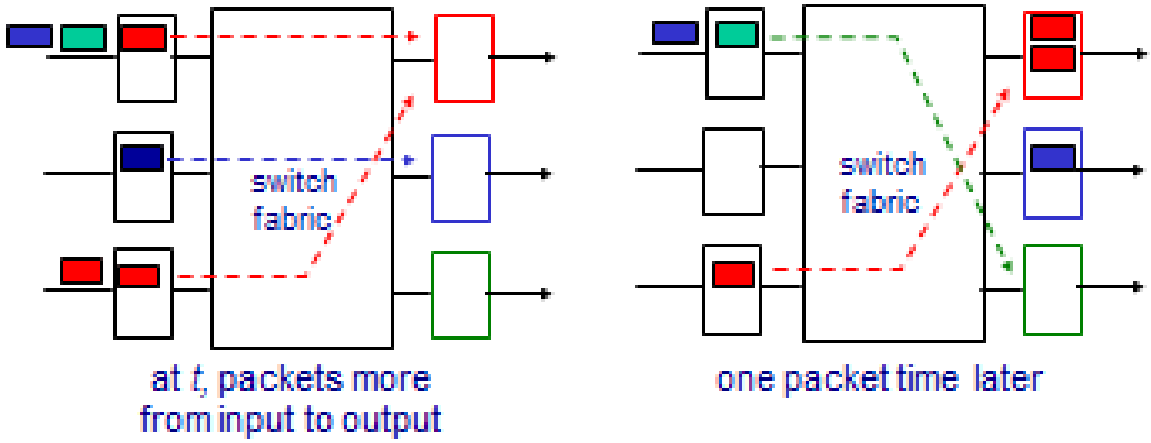
Output processing

Output Processing

Router Output Interface



Output Port Buffering



- **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

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

END

TOPIC 147

Head of Line Blocking

In this module

We shall understand

- Input port overflow
- Head of Line blocking
- Scenario

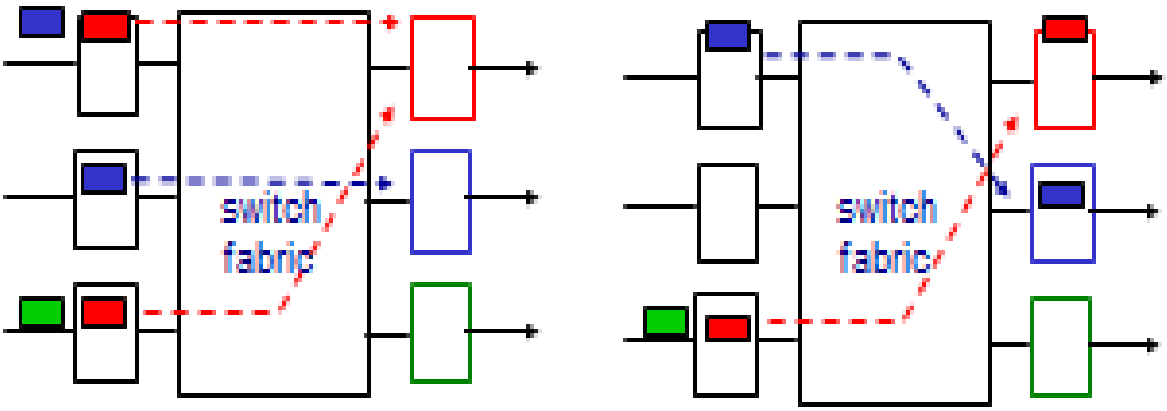
Input Port Overflow

- 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



one packet time later: green packet experiences HOL blocking

END

TOPIC 148

Random Early Detection

In this module

We shall understand

- Active Queue Management

- Drop Tail
- RED

Reference

Floyd, Sally, and Van Jacobson. "Random early detection gateways for congestion avoidance." Networking, IEEE/ACM Transactions on 1.4 (1993): 397-413.

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 of network packets inside a buffer of NIC
- Often with the larger goal of reducing network congestion

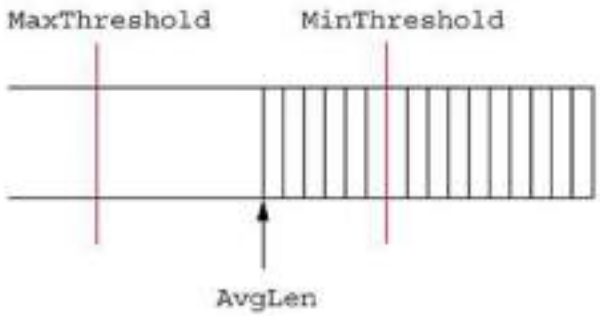
Drop Tail

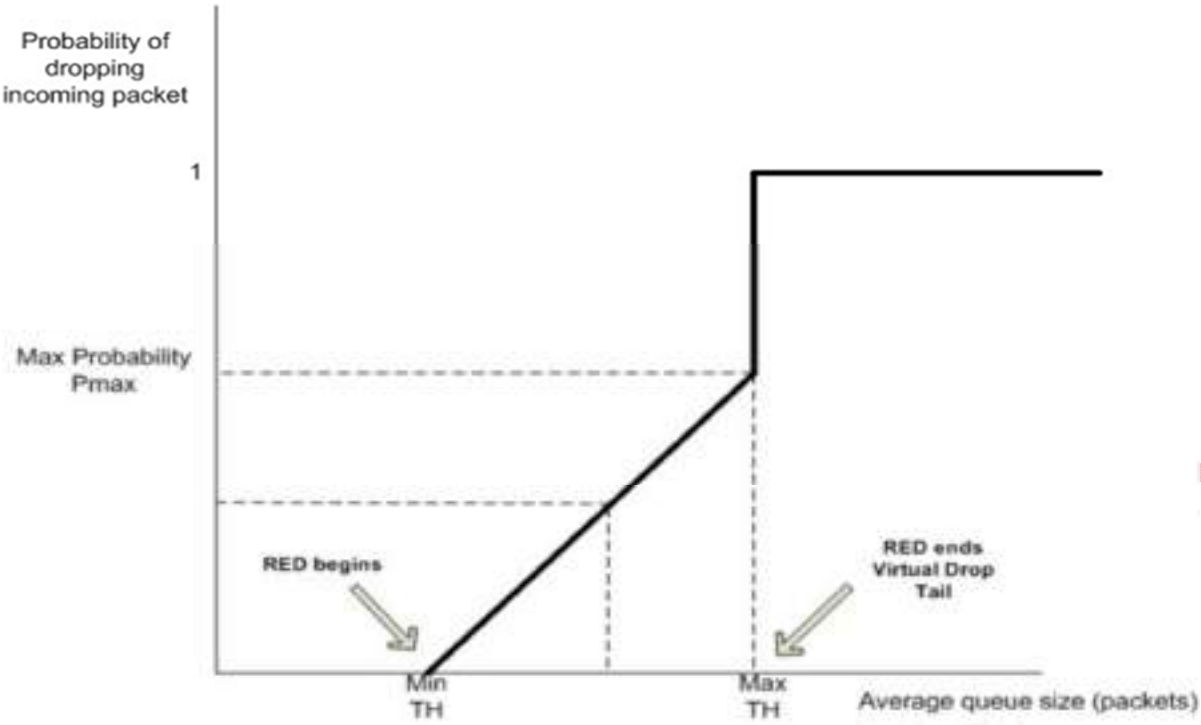
- Conventional tail drop algorithm
- A router buffers as many packets as it can
- Simply drops the ones it cannot buffer
- If buffers constantly full, network is congested
- Tail drop distributes buffer space unfairly among traffic flows

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





END

TOPIC 149

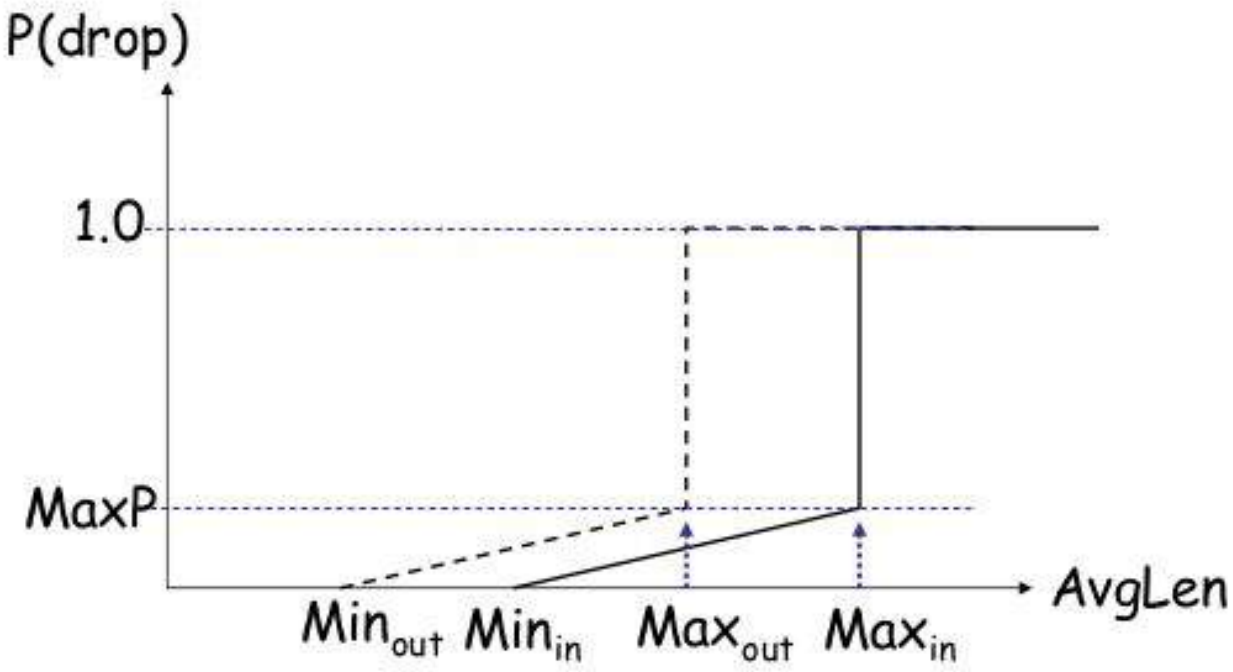
RED with In & Out (RIO)

In this module
We shall understand
- Background for RIO
- Operation
- **Reference**
- https://www.ee.iitb.ac.in/~prakshep/IBMA_lit/manual/manual223.html

Background (1 of 2)
- 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



END

TOPIC 150

Routing Algorithms

In this module

We shall understand

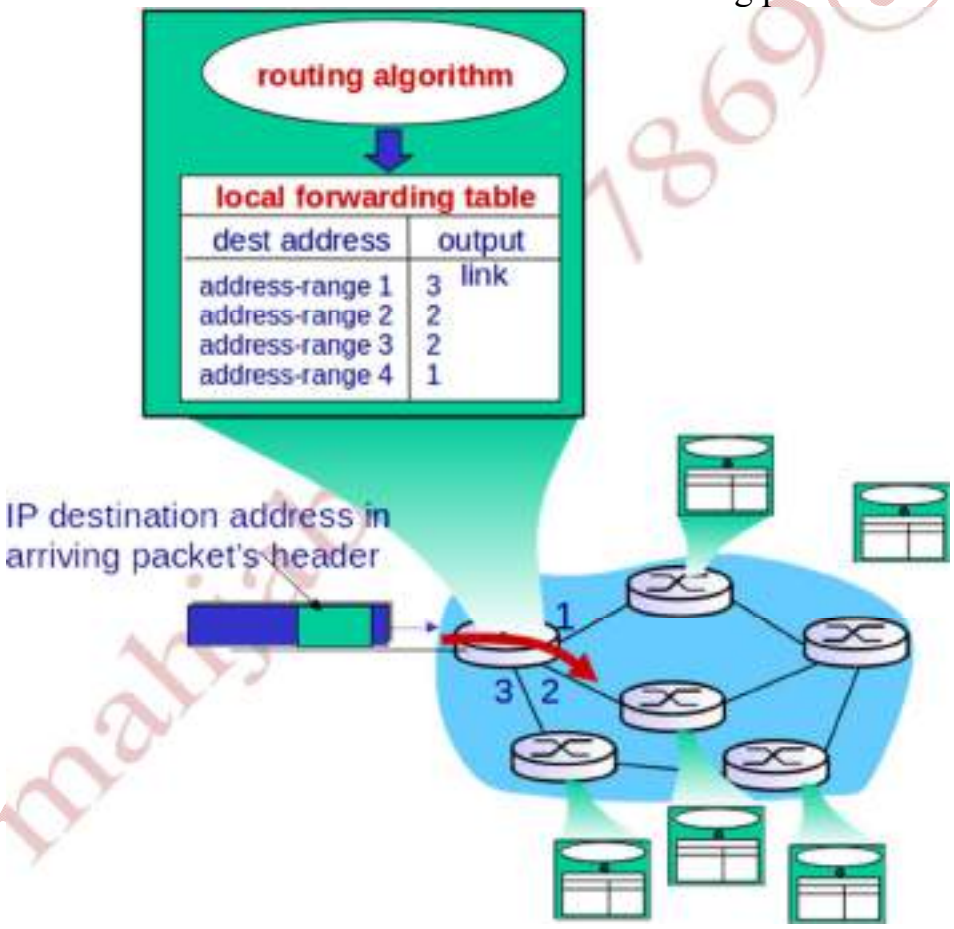
- Interplay between routing and forwarding
- Graph abstraction
- Key question

Reference

Kurose, James F. Computer Networking: A Top-Down Approach Featuring the Internet, 3/E. Pearson Education India, 2005.

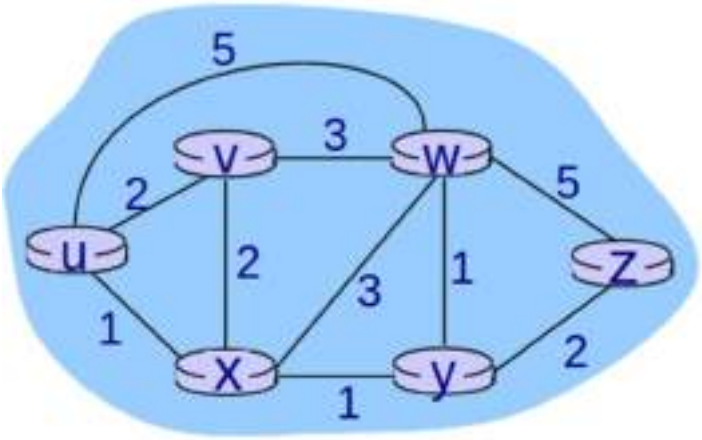
Interplay

- 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

END

TOPIC 151

Complexity of Link State

In this module

We shall understand

- Global algorithms
- Link state algorithms
- **Reference**
- Kurose, James F. Computer Networking: A Top-Down Approach Featuring the Internet, 3/E. Pearson Education India, 2005.

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: $O(n^2)$

END

TOPIC 152

Complexity of Distance Vector

In this module

We shall understand

- Distributed algorithms
- Distance vector algorithms
- Complexity

Reference

Kurose, James F. Computer Networking: A Top-Down Approach Featuring the Internet, 3/E. Pearson Education India, 2005.

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

Distance Vector

- Based on Bellman-Ford equation
- Dynamic programming model
- Let
- $d_x(y)$:= cost of least-cost path from x to y then
- $d_x(y) = \min \{c(x,v) + d_v(y)\}$

Complexity

- **Message**: exchanges between neighbours only
  - Linear
- **Speed**: convergence time varies due to potential routing loops

END

TOPIC 153

Count to Infinity Problem

In this module

We shall understand

- Link cost change
- Good news!
- Bad news!
- The problem
- Routing loops

Reference

Kurose, James F. Computer Networking: A Top-Down Approach Featuring the Internet, 3/E. Pearson Education India, 2005.

Link Cost Changes

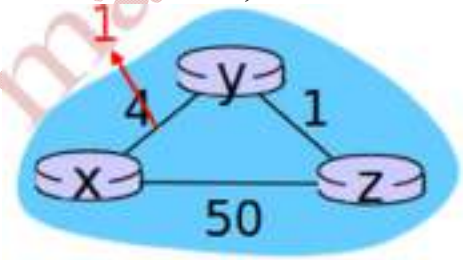
- Node detects local link cost change
- Updates routing info
- Recalculates distance vector
- If DV changes, notify neighbours

Good news! (1 of 2)

- At time t_0, y detects the link-cost change, updates its DV,

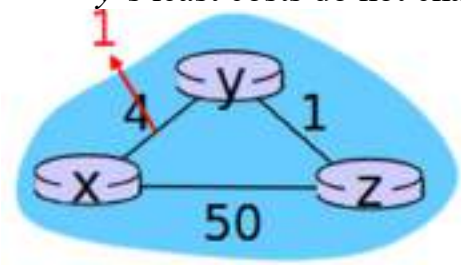
& informs neighbors

- At time t_1, z receives the update from y and updates its table



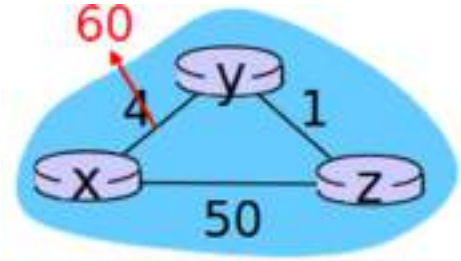
Good news! (2 of 2)

- 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

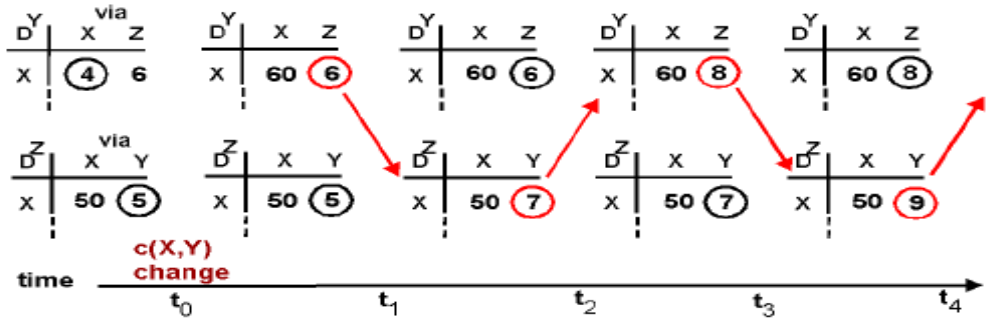


Bad News! (1 of 2)

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

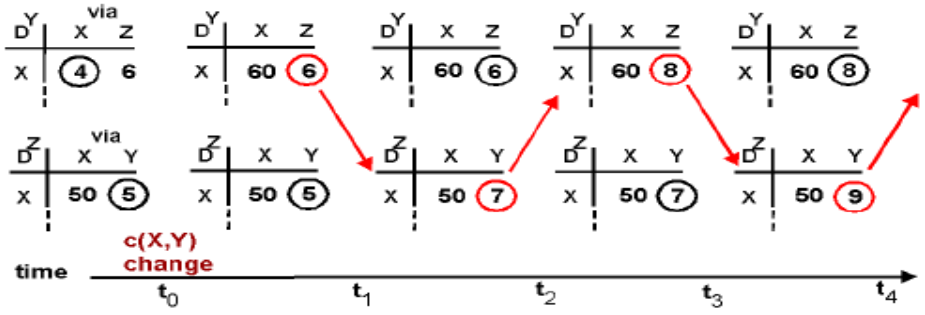


Bad News Causes Loops (1 of 4)



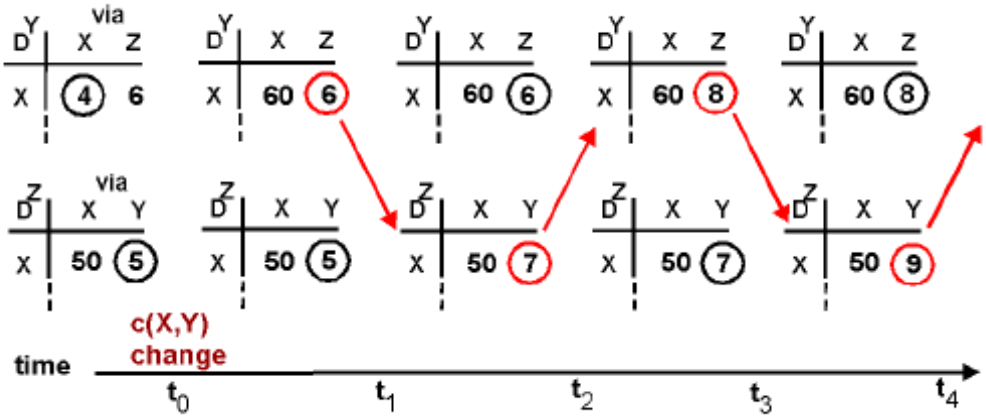
At time t0 Y detects the link cost change (the cost has changed from 4 to 60). Y computes its new minimum cost path to X to have a
cost of 6 via node Z. Of course, we can see that this new cost via Z is wrong

Bad News Causes Loops (2 of 4)



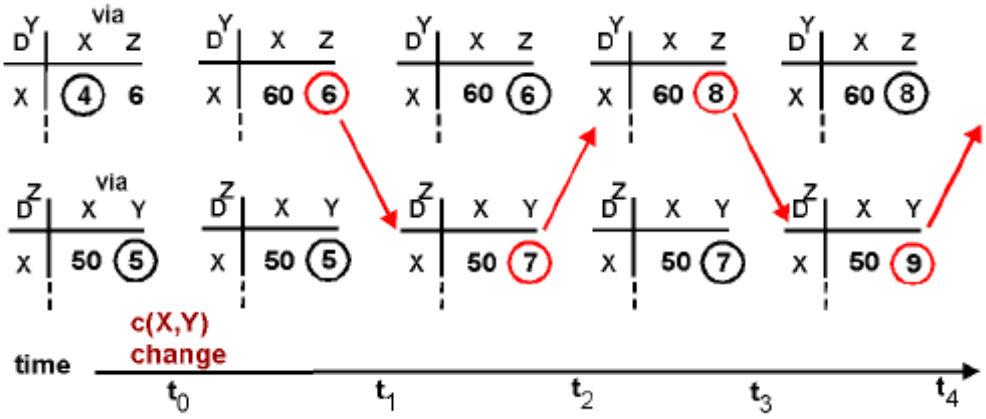
But the only information node Y has is that its direct cost to X is 60 and that Z has last told Y that Z could get to X with a cost of 5. So in order to get to X, Y would now route through Z, fully expecting that Z will be able to get to X with a cost of 5

Bad News Causes Loops (3 of 4)



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.

Bad News Causes Loops (4 of 4)



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

END

TOPIC 154

Poisoned Reverse

In this module

We shall understand

- Need for poisoned reverse
- Operation

Reference

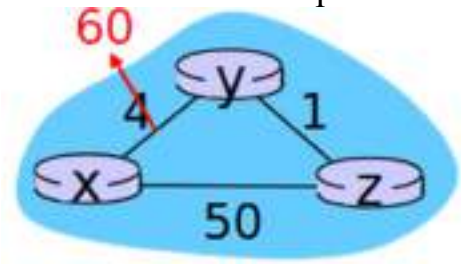
- Kurose, James F. Computer Networking: A Top-Down Approach Featuring the Internet, 3/E. Pearson Education India, 2005.

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

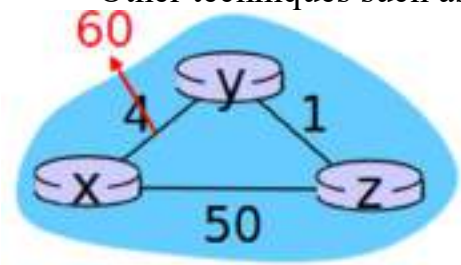
- If Z routes through Y to get to X
  - Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z) table
  - This lie prevents the loop



Performance

Mcqs

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



END

TOPIC 155

Hierarchical Routing; Complexity

In this module

We shall understand

- Why do we need hierarchical routing?
- Implementation methodology

Reference

- Kurose, James F. Computer Networking: A Top-Down Approach Featuring the Internet, 3/E. Pearson Education India, 2005.

Need

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

Each network admin wants autonomy

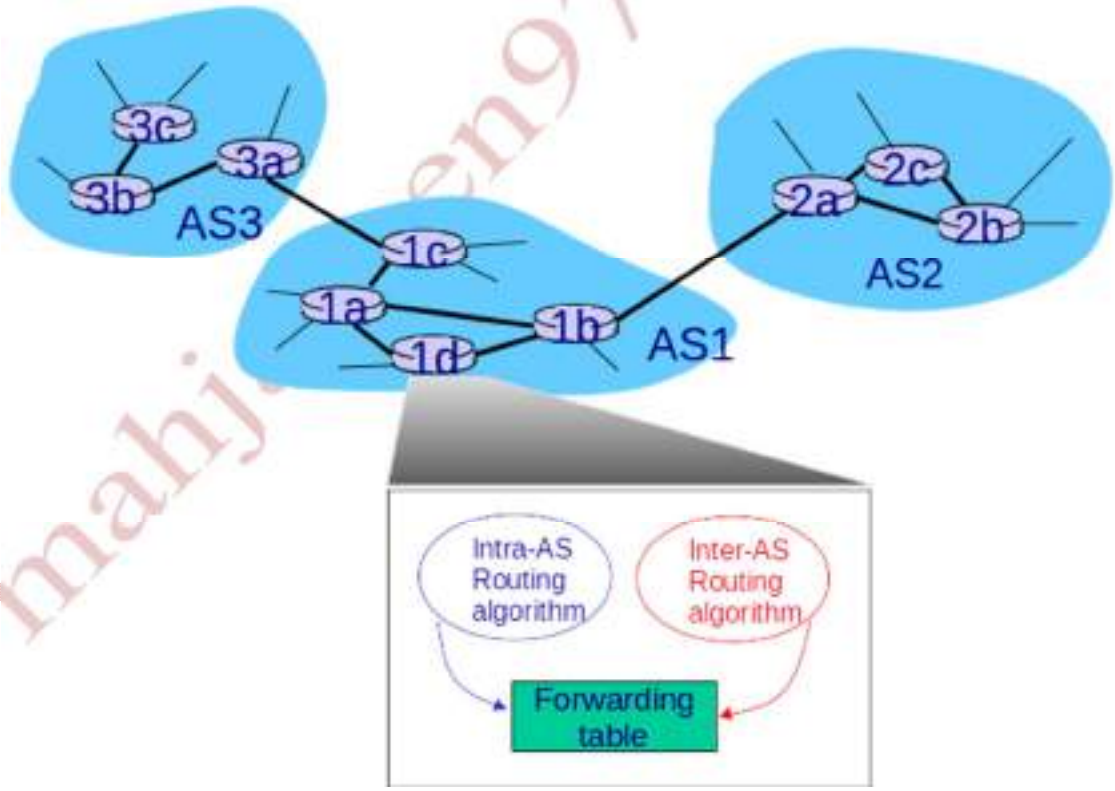
Solution: Make a hierarchical relationship between them

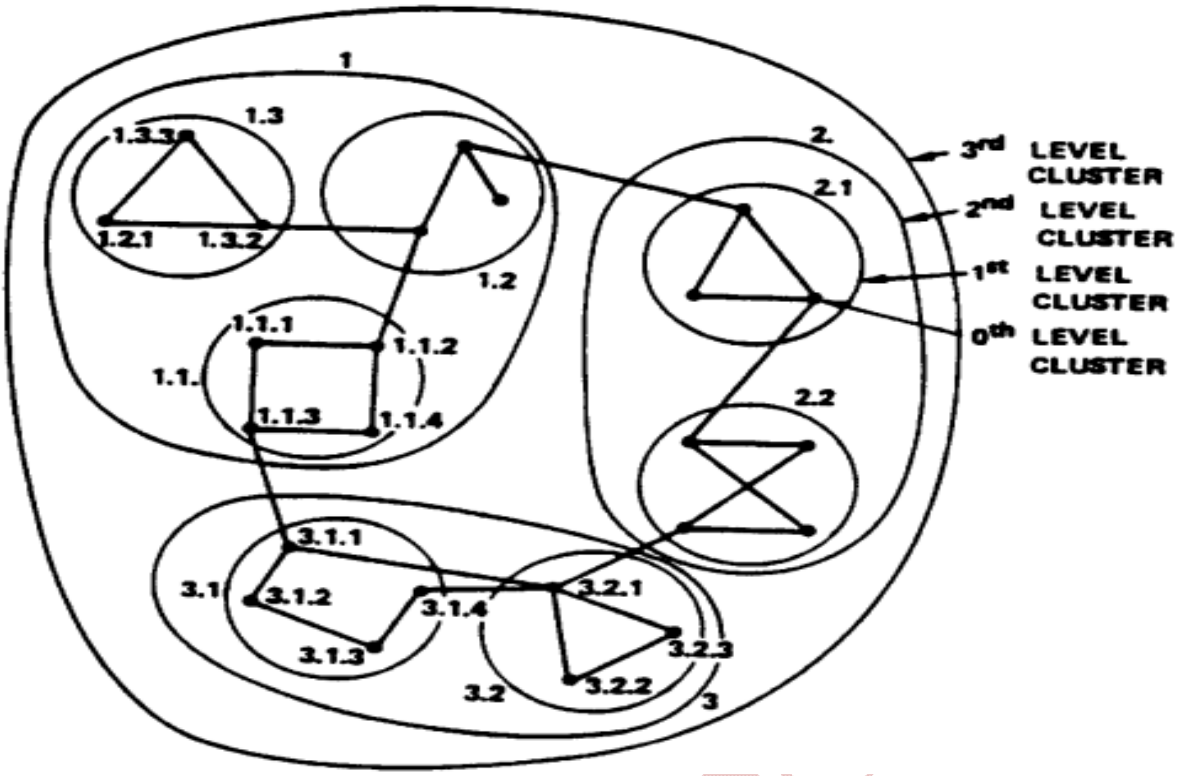
Methodology

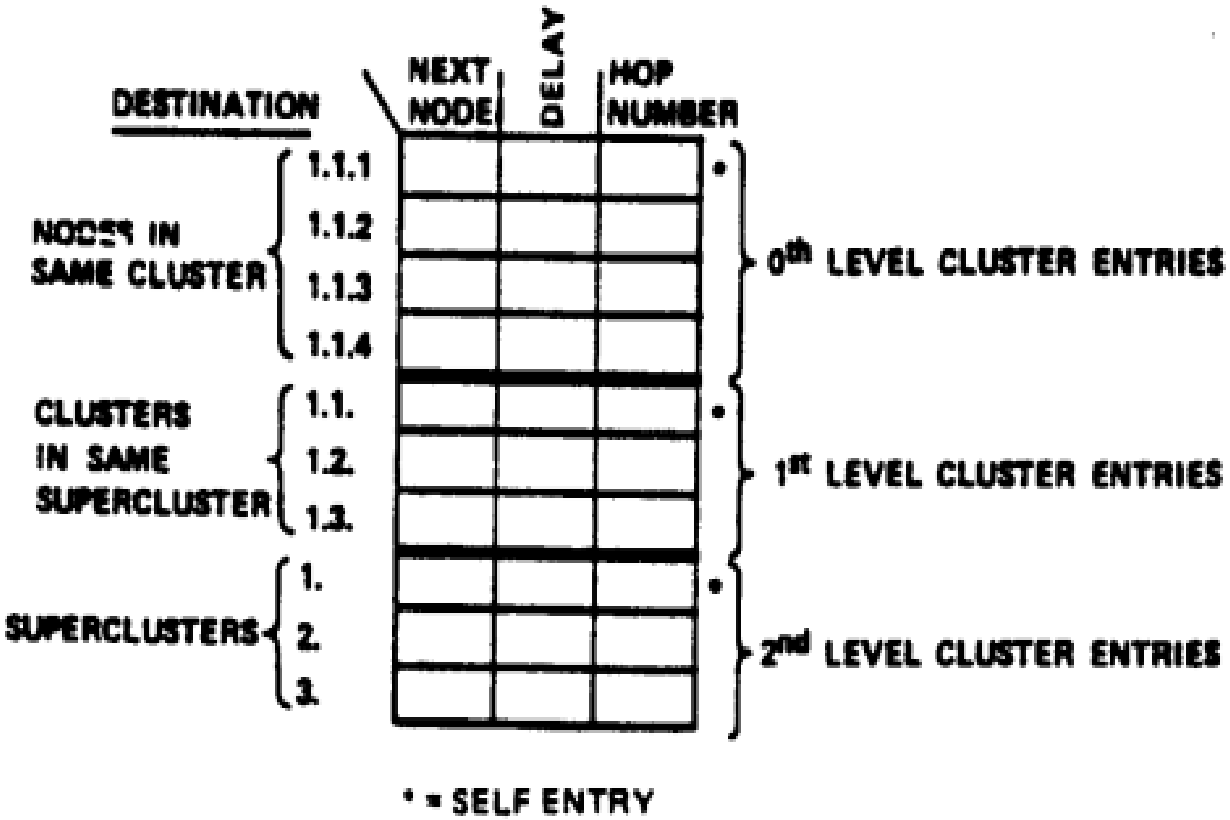
- Collect routers into regions, "autonomous systems" (AS)
- Each AS within an ISP
  - ISP may consist of one or more ASes
- In same AS run same routing protocol
- "intra-AS" routing proutersrotocol
  - routers in different

AS run different
intra-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




1
1.3
1.3.3
1.2.1
1.3.2
1.2
1.1.1
1.1.2
1.1.
1.1.3
1.1.4
2.
2.1
2.2
3.1.1
3.1
3.1.2
3.1.3
3.1.4
3.2.1
3.2.3
3.2.2
3.2
3
3rd LEVEL CLUSTER
2nd LEVEL CLUSTER
1st LEVEL CLUSTER
0th LEVEL CLUSTER


DESTINATION
NEXT NODE
DELAY
HOP NUMBER
NODES IN SAME CLUSTER
1.1.1
1.1.2
1.1.3
1.1.4
CLUSTERS IN SAME SUPERCLUSTER
1.1.
1.2.
1.3.
SUPERCLUSTERS
1.
2.
3.
0th LEVEL CLUSTER ENTRIES
1st LEVEL CLUSTER ENTRIES
2nd LEVEL CLUSTER ENTRIES
* = SELF ENTRY

END

TOPIC 156

Elastic Aggregates & TE

In this module

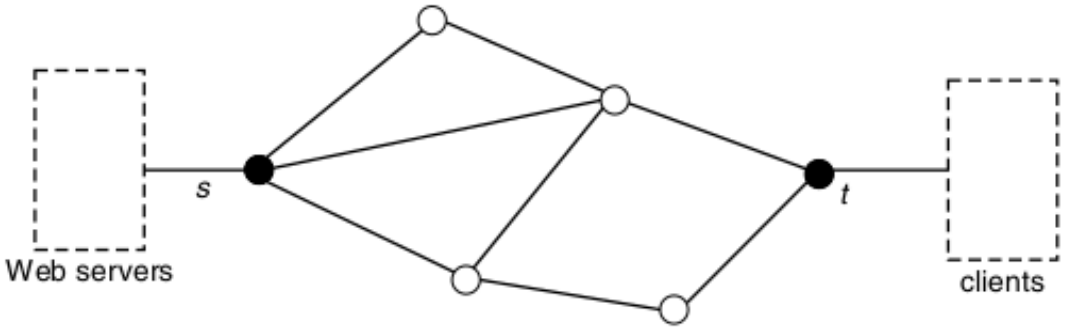
We shall understand

- A generalized scenario
- Demand characterization

Reference

- Anurag Kumar et al. Communication Networking—An Analytical Approach. Featuring the Internet,Morgan Kaufmann Publishers. 2004.

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

END

WEEK 12

TOPIC 157

Optimal Routing

In this module

We shall understand

- Feasible routing
- Optimization problem

Reference

- Anurag Kumar et al. Communication Networking—An Analytical Approach. Featuring the Internet,Morgan Kaufmann Publishers. 2004.

Feasible Routing

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

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

- **Spare capacity**

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

Optimization Problem (1 of 2)

- 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

Optimization Problem (2 of 2)

- 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

END

TOPIC 158

Limitations of Min Hop Routing

In this module

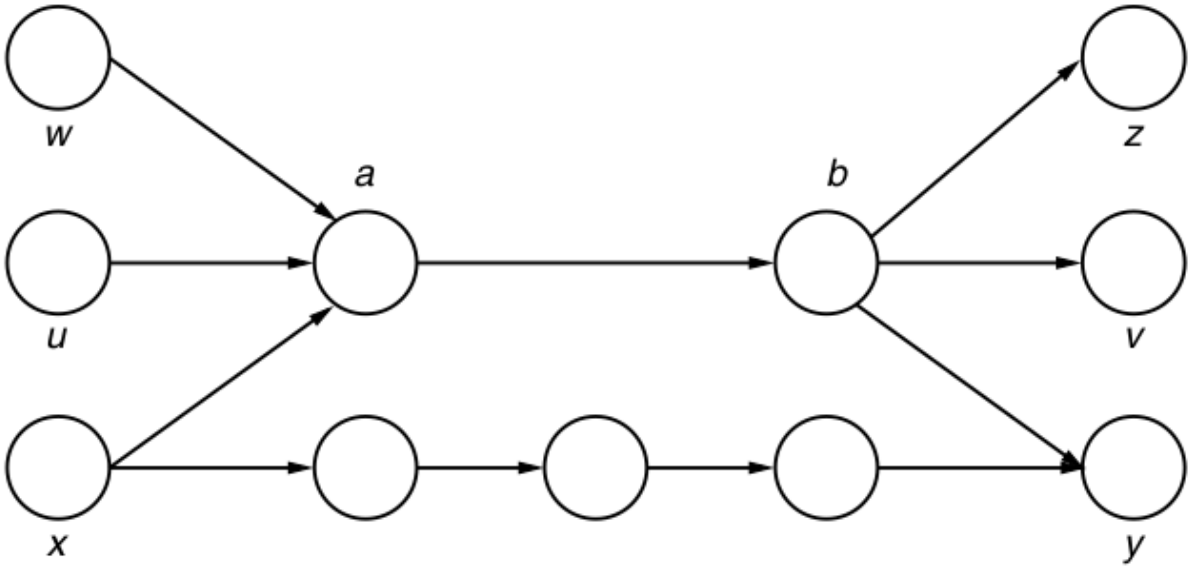
We shall understand

- Shortest path as min hop
- Disadvantages

Reference

- Anurag Kumar et al. Communication Networking—An Analytical Approach. Featuring the Internet,Morgan Kaufmann Publishers. 2004.

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

END

TOPIC 159

Formulation of Routing Problem

In this module

We shall understand

- Min hop routing as congestion source
- Routing problem definition

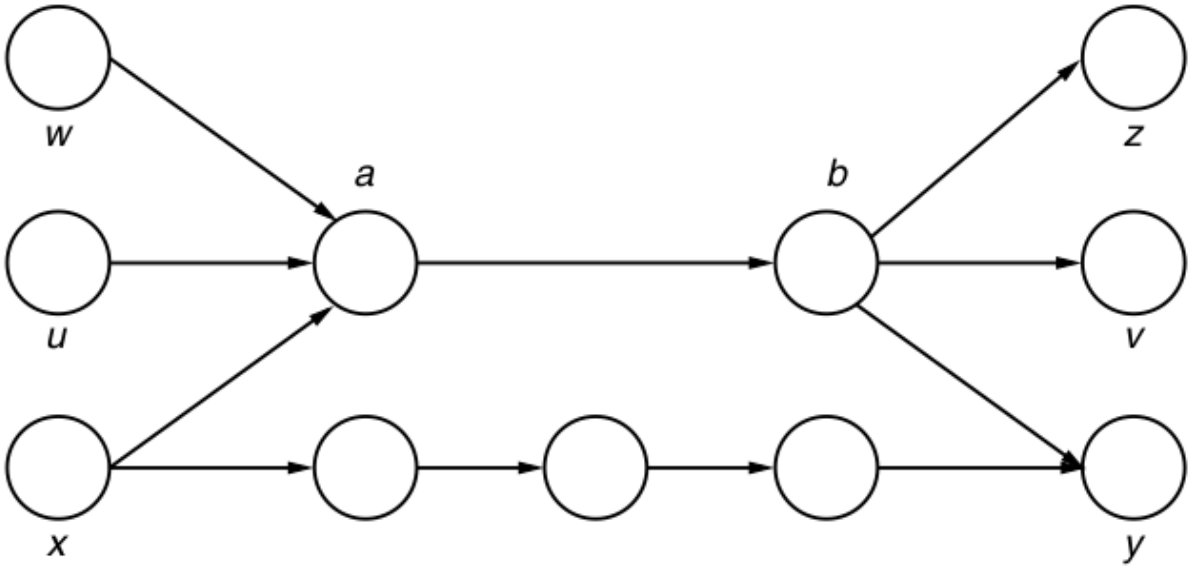
Reference

- Anurag Kumar et al. Communication Networking—An Analytical Approach. Featuring the Internet,Morgan Kaufmann Publishers. 2004.

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

END

TOPIC 160

Minimum Interference Routing

In this module

We shall understand

- What is route interference?

- Max Flow
- Minimum interference

Reference

- Anurag Kumar et al. Communication Networking—An Analytical Approach. Featuring the Internet,Morgan Kaufmann Publishers. 2004.

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

    - Ideally 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

END

TOPIC 161

QoS Routing

In this module

We shall understand

- QoS goals of single session

- QoS goals of network operator
- Tradeoff

Reference

Anurag Kumar et al. Communication Networking—An Analytical Approach. Featuring the Internet,Morgan Kaufmann Publishers. 2004.

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 (1 of 2)**
- 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
- **Tradeoff (2 of 2)**
- 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

END

TOPIC 162

Nonadditive Metrics

In this module

We shall understand

- Definition of nonadditive metrics
- Applying nonadditive metrics
- Implications

Reference

Anurag Kumar et al. Communication Networking—An Analytical Approach. Featuring the Internet, Morgan Kaufmann Publishers. 2004.

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

END

TOPIC 163

Additive Metrics; RMB

In this module

We shall understand

- Definition of additive metrics
- Rate-based multiplexers
- Implications

Reference

- PC Wong et. al. A Programmable rate-based multiplexer (PRS) for ATM switches and multiplexers. IEEE GlobeCom 1997.
- Anurag Kumar et al. Communication Networking—An Analytical Approach. Featuring the Internet, Morgan Kaufmann Publishers. 2004.

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*

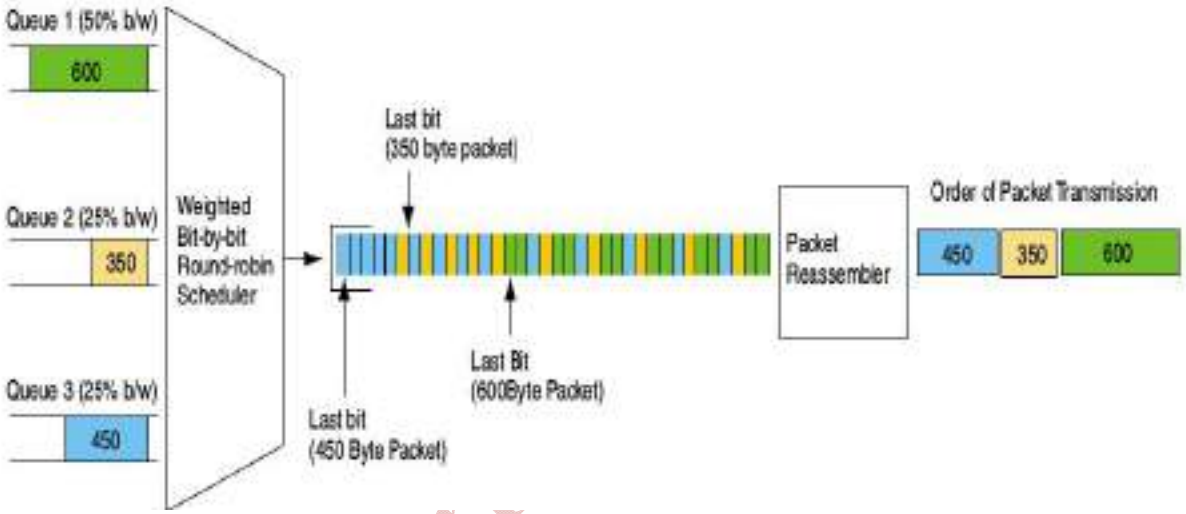
Rate-based Mux

- 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

Weighted Bit-by-bit Round-robin Scheduler with Packet Reassembler



END

TOPIC 164

Finding Feasible Routes

In this module

We shall understand

- Network model
- Problem formulation
- Multi-commodity problem

Reference

Anurag Kumar et al. Communication Networking—An Analytical Approach. Featuring the Internet, Morgan Kaufmann Publishers. 2004.

Network Model (1 of 3)

- 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

Network Model (1 of 3)

- 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

Network Model (2 of 3)

- 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

Network Model (3 of 3)

- Traffic/flow management discipline = Leaky Bucket
- σ_k = maximum burst size
- ρ_k = average rate of the source
- c_k = maximum packet length corresponding to session k

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

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

- If all the paths are computed
  - Multi-commodity problem
  - NP hard

END

TOPIC 165

Upper Bound on Performance

In this module

We shall understand

- Route and Rate Allocation (RRA)
- Problem Formulation

Reference

Anurag Kumar et al. Communication Networking—An Analytical Approach. Featuring the Internet, Morgan Kaufmann Publishers. 2004.

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)**

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

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

$$\sum_{i=1}^{C} \sum_{j=1}^{J} r_{ij} n_{ij} B_{j,l} \leq C_l \quad \forall l \in \mathcal{L}$$

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

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

END

TOPIC 166

Non-Rate-Based Multiplexers

In this module

We shall understand

- Non-Rate-Based Multiplexers
- Multi-constrained feasibility problem

Reference

Anurag Kumar et al. Communication Networking—An Analytical Approach. Featuring the Internet, Morgan Kaufmann Publishers. 2004.

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

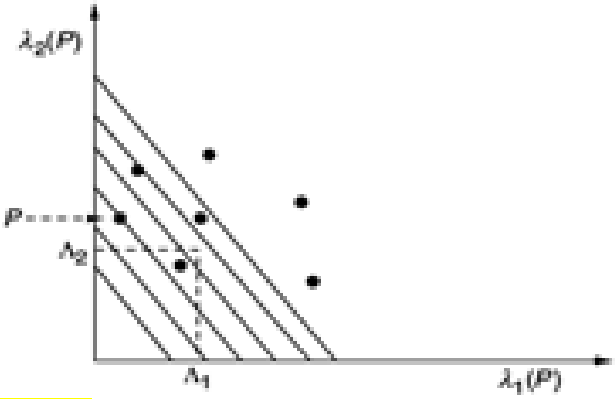
Multi-constrained Feasibility Problem (1 of 2)

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

Multi-constrained Feasibility Problem (2 of 2)

- 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

END

TOPIC 167

Efficient Longest Prefix Match

In this module

We shall understand

- Operation at router
- Longest prefix match
- Increasing efficiency

Reference

Anurag Kumar et al. Communication Networking—An Analytical Approach. Featuring the Internet, Morgan Kaufmann Publishers. 2004.

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

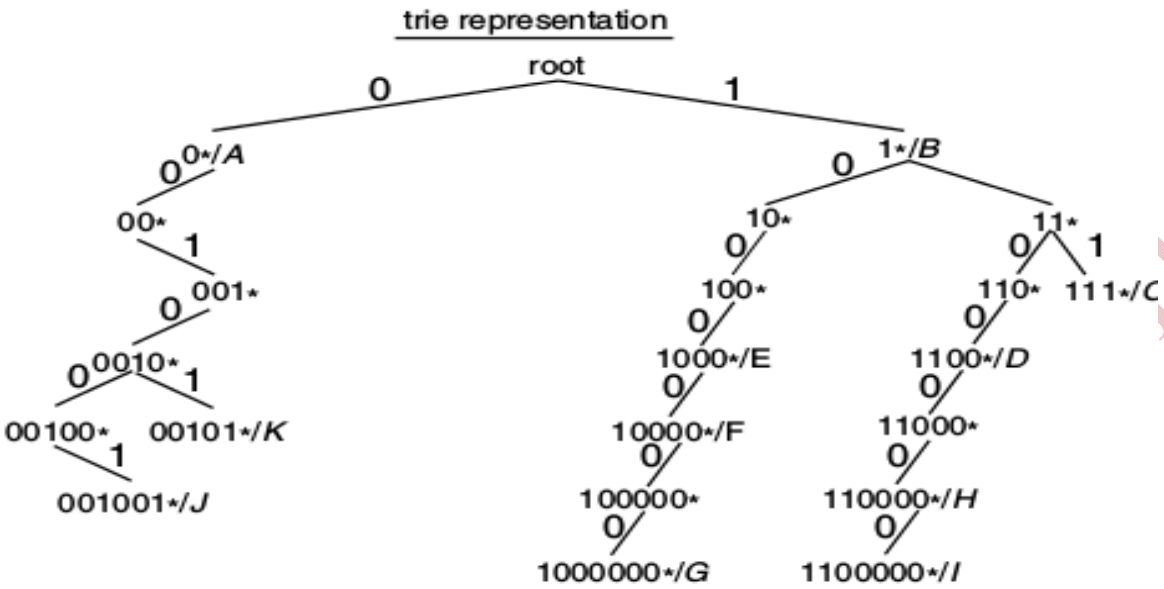
Binary Trie (1 of 2)

- 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

Binary Trie (2 of 2)

- 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

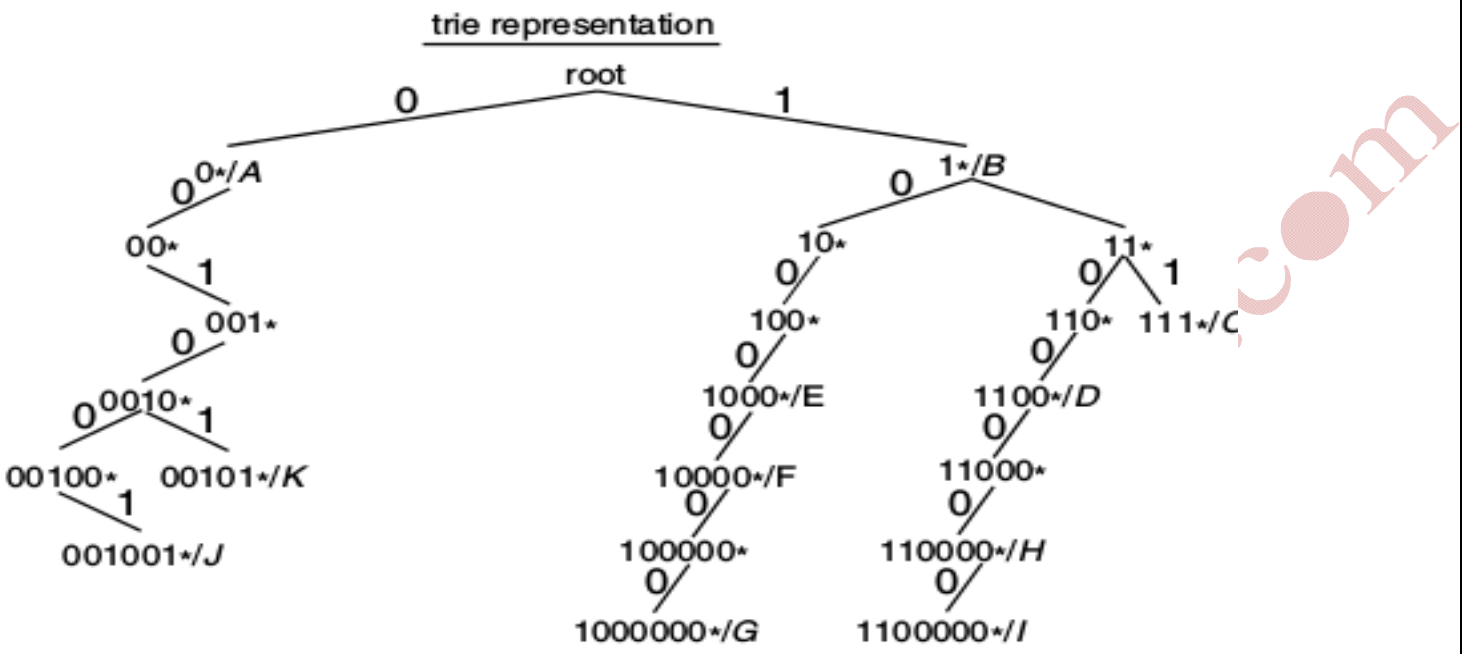


routing table

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*

CS432 Handouts Made by Mahjabeen
mahjabeen97869@gmail.com
contact # 0321 2711298

Trie Traversal for 1000000



END

TOPIC 168

Level-Compressed Tries

In this module

We shall understand

- Traversal time

Reference

- Anurag Kumar et al. Communication Networking—An Analytical Approach. Featuring the Internet, Morgan Kaufmann Publishers. 2004.

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 (1 of 2)

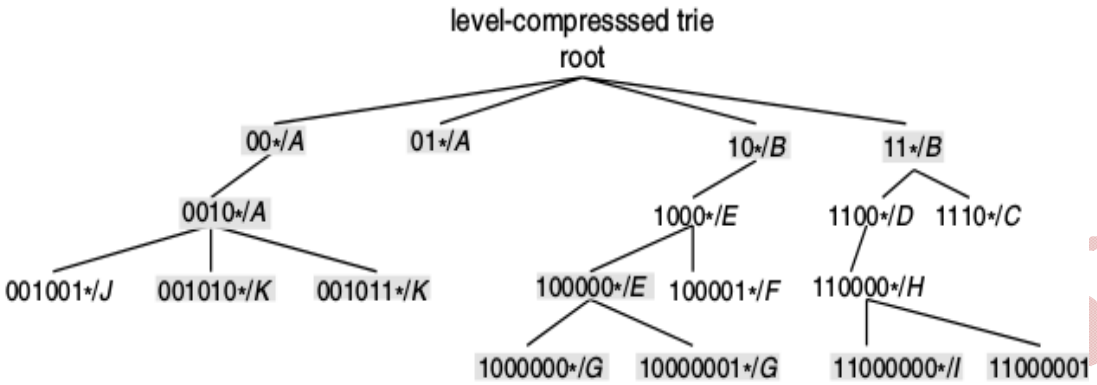
- Rather than define a level for each bit of the address
  - Define a level for groups of contiguous bits
- A simple case of level compression is to have a level for every *K* bits
- For N bits in address, then the number of levels is N/K

Level Compress (2 of 2)

- 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

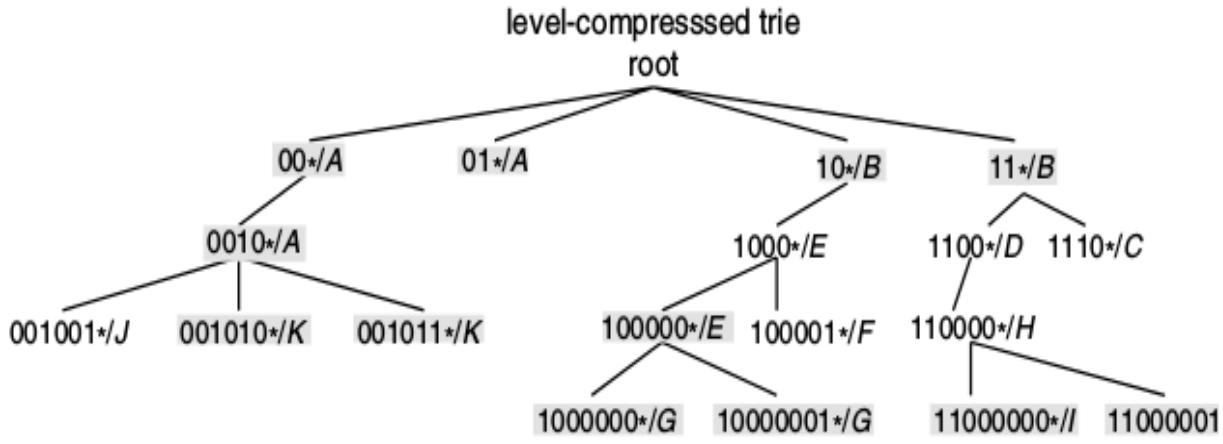
Level Compression or Prefix Expansion



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



TOPIC 169

Flooding; ARPANET Algorithm

In this module

We shall understand

- Usage of flooding
- Indefinite flood
- ARPANET solution

Reference

Bertsekas, Dimitri P., Robert G. Gallager, and Pierre Humblet. Data networks. Vol. 2. New Jersey: Prentice-Hall International, 1992.

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

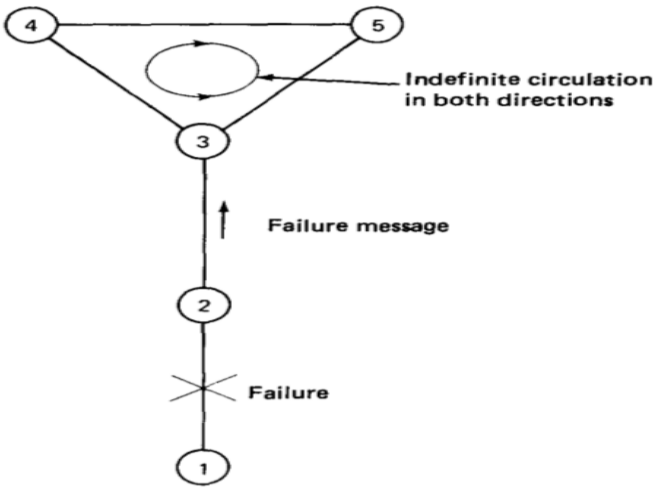
- Transmission of messages never terminates
  - Rule: node that receives a message relays it to all of its neighbors except from which it received

Level Compress (2 of 2)

- 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

END

TOPIC 170

Flooding w/o Periodic Updates

In this module

We shall understand

- Redundancy of periodicity
- Need-based updates
- The problem
- The solution

Reference

Bertsekas, Dimitri P., Robert G. Gallager, and Pierre Humblet. Data networks. Vol. 2. New Jersey: Prentice-Hall International, 1992.

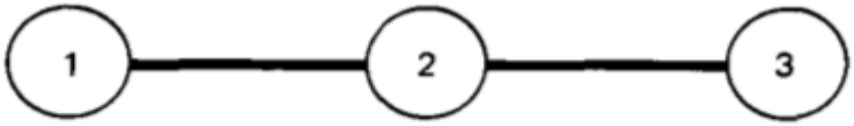
Redundancy of Periodicity

- 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

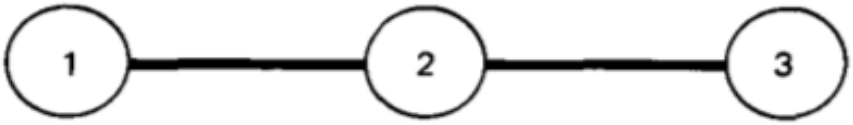
- Zero seq no allowed only when node is recovering from a crash
  - Situation where all of the node's incident links are down
  - And it is in the process of bringing links up
- Separate seq no. for each origin node

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

END

TOPIC 171

Broadcast without Seq. Nos.

In this module

We shall understand

- Redundancy of periodicity
- Need-based updates
- The problem
- The solution

Reference

Bertsekas, Dimitri P., Robert G. Gallager, and Pierre Humblet. Data networks. Vol. 2. New Jersey: Prentice-Hall International, 1992.

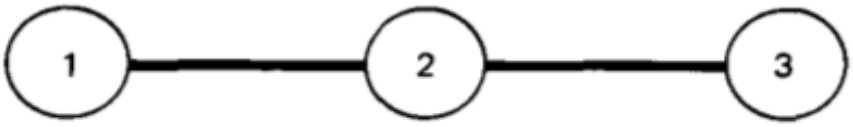
Redundancy of Periodicity

- 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

END

WEEK 13

TOPIC 172

Problem Set 1

In this module

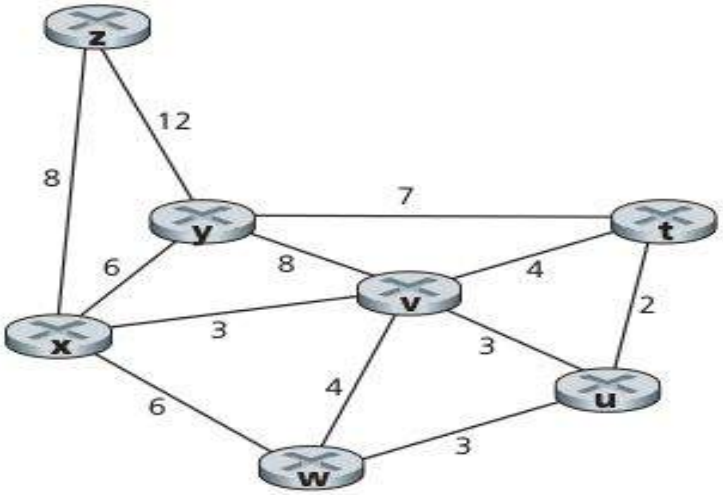
We shall understand

- Complexity of routing algorithms
- Operation of inter-AS protocols

Reference

Kurose, J., and K. Ross. "Computer networking: a top-down approach." 6th Edition (2013), Pearson

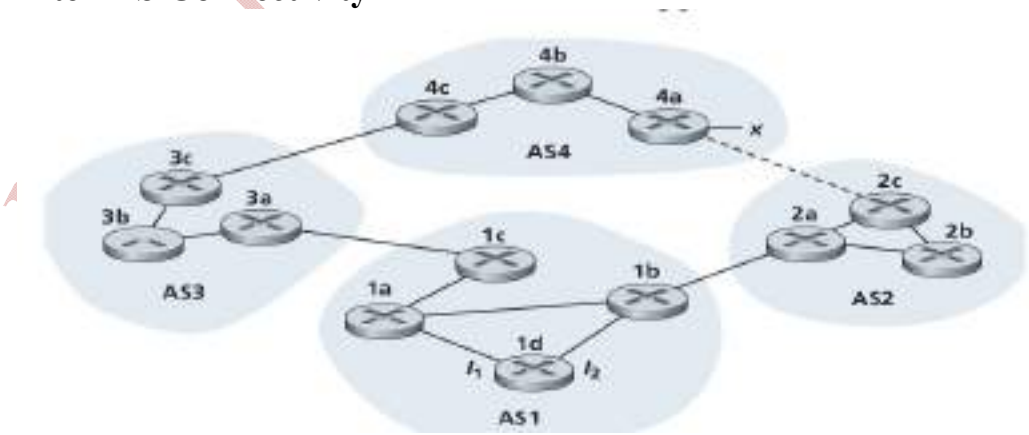
Topology for SPF Algorithm



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?

END

TOPIC 173

Problem Set 1

In this module

We shall understand

- Switching fabric performance
- Inter-AS connectivity
- Subnetting a network
- Fragmentation & reassembly

Reference

Kurose, J., and K. Ross. "Computer networking: a top-down approach." 6th Edition (2013), Pearson

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?

END

TOPIC 174

Simulate QoS Routing

In this module

We shall understand

- What is QoS routing?
- An example scenario
- Implementation example
- Support in Inet

References

Chaudhry, Shafique Ahmad, et al. "NETSAQ: Network state adaptive QoS provisioning for MANETs." Management of Convergence Networks and Services. Springer Berlin Heidelberg, 2006. 170-179.

"Design and Simulation of a Quality of Service (QoS) Guaranteed Model for Mobile Ad Hoc Networks (MANets)". SATNAC 2006.

https://omnetpp.org/pmwiki/index.php?n=Main.INETDiffServ

Operation of QoS Routing

simple quality of services

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 (1 of 3)

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

Decision (2 of 3)

If resources are reserved equal to 65 kbps

when destination node receives data rate < 65 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

Decision (3 of 3)

If resources are reserved equal to 50 kbps

when destination node receives data rate < 50 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

END

TOPIC 175

Simulate Routing Updates

In this module

We shall understand

- CISCO EIGRP
- Basic operation
- ANSA
- Module structure in Inet

Reference

Vesely, Vladimir, Jan Bloudicek, and Ondrej Rysavy. "Enhanced interior gateway routing protocol for OMNeT++." Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH), 2014 International Conference on. IEEE, 2014.

EIGRP

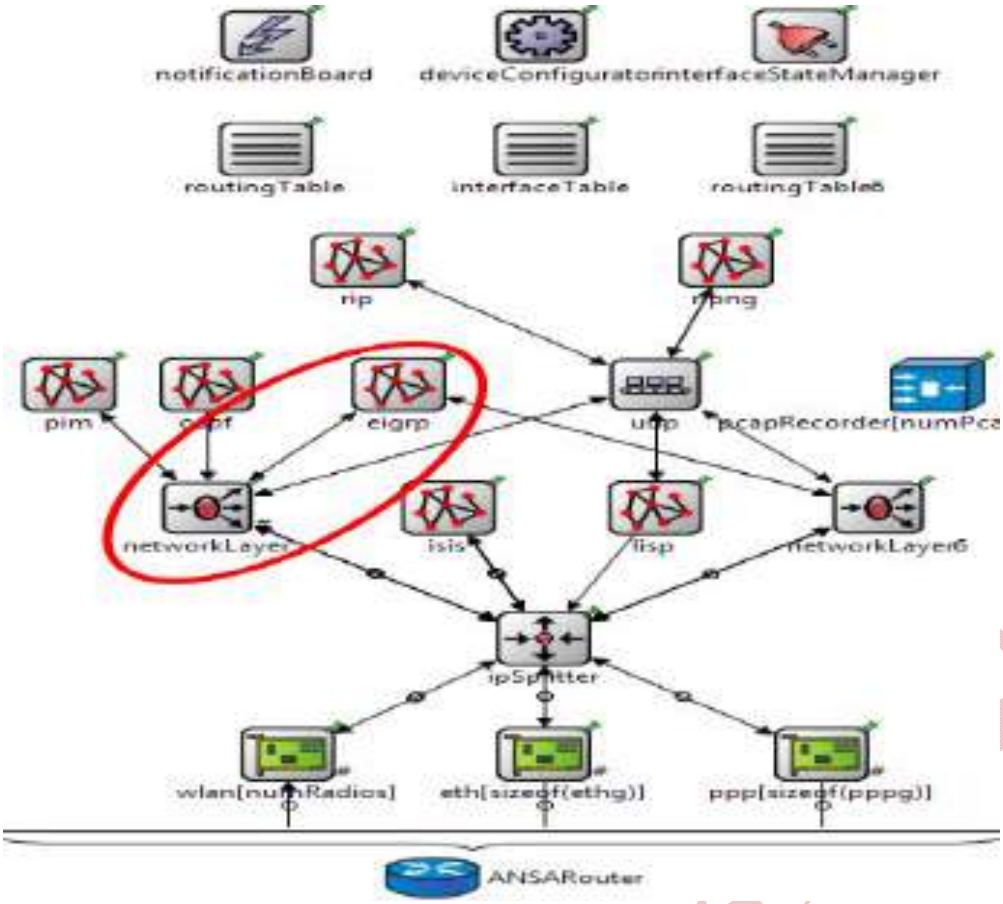
- Cisco's EIGRP is a hybrid routing protocol between distance vector and link state routing protocols
- EIGRP offers routing based on composite metric
- Cisco released EIGRP specs as IETF's RFC draft in 2013

Basic Operation

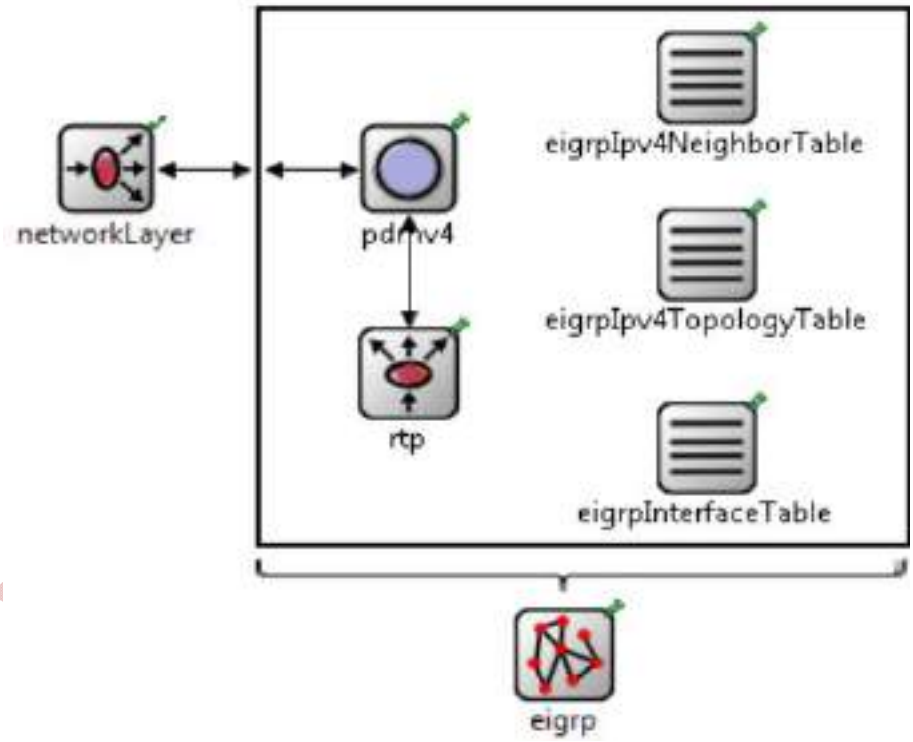
- EIGRP employs Diffusing Update Algorithm (DUAL)
- Propagates topology change minimizing path compute time
- Sends event-driven partial bound updates
  - Weighted (Bandwidth + Delay)

ANSA

- Automated Network Simulation and Analysis
- @Brno University of Technology
- Czech Republic



EIGRP Simulation Module Structure



END

TOPIC 176

Simulate HSRP

In this module

We shall understand

- Background for failover
- CISCO Hot standby

Basic operation

- AMDF Simulation

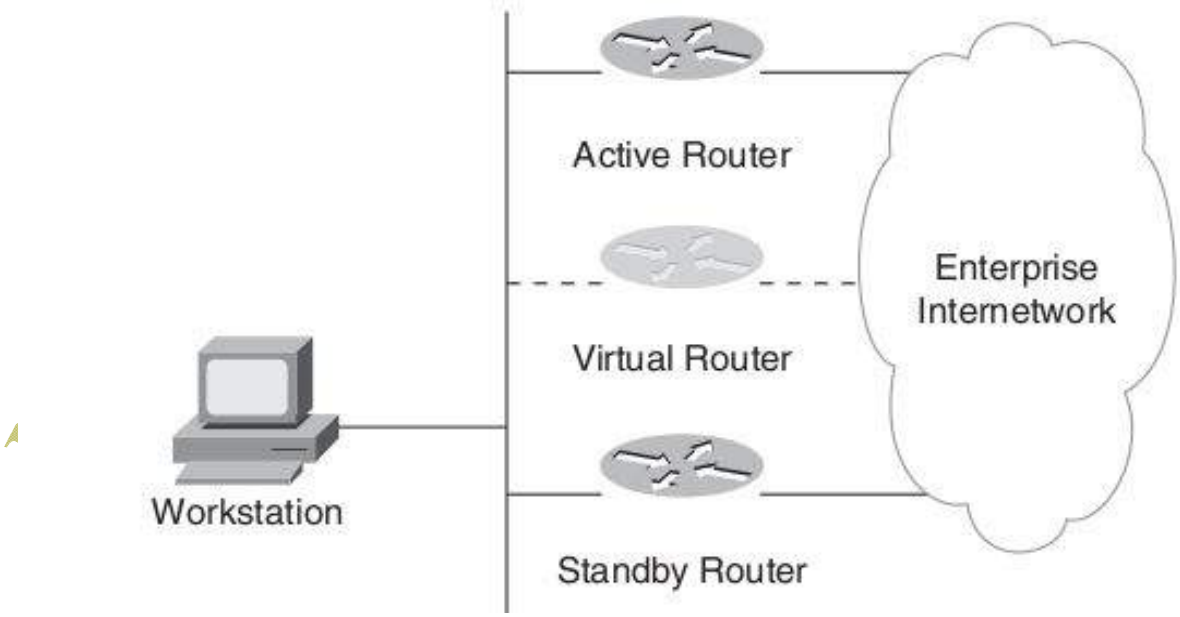
Reference

Oppenheimer, Priscilla. Top-down network design. Cisco Press, 2004.
Botha, Marthinus Ignatius. Modelling and Simulation Framework Incorporating Redundancy and Failure Probabilities for Evaluation of a Modular Automated Main Distribution Frame. Diss. University of Pretoria, 2012.

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
  - Virtual router has its own IP and MAC addresses

Hot Standby Router Protocol (HSRP)



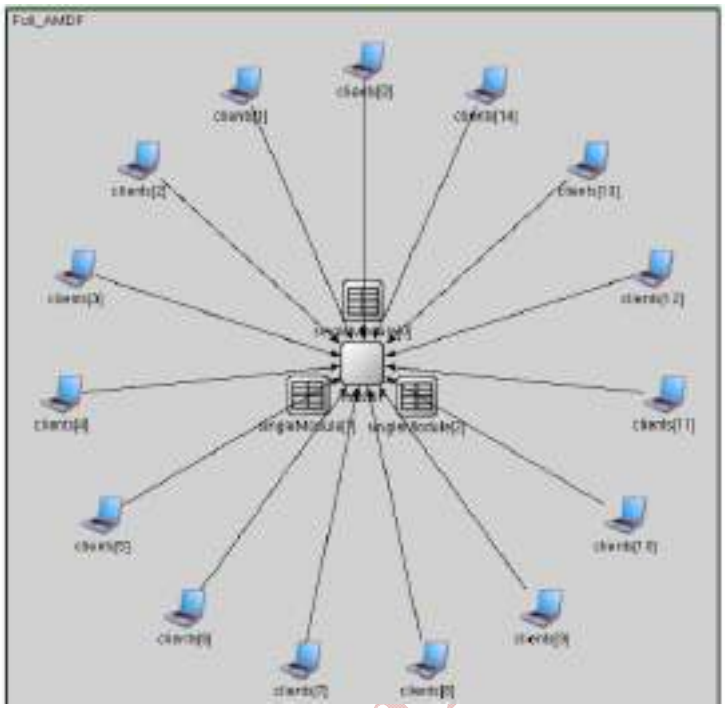
Basic Operation (1 of 2)

- 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

Basic Operation (2 of 2)

- 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



END

TOPIC 177

Simulate Flooding

In this module

We shall understand

- Message complexity in simple flooding
- Recalling TicToc14

Reference

Hu, Ruijing. "Efficient Probabilistic Information Broadcast Algorithm over Random Geometric Topologies." GLOBECOM. 2015.

Chelloug, Samia Allaoua. "Energy-Efficient Content-Based Routing in Internet of Things." Journal of Computer and Communications 3.12 (2015): 9.
Ruijing Hu. “Fair Comparison of Gossip Algorithms over Large-Scale Random Topologies.” International Symposium on Reliable Distributed Systems 2012.

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

Simulate Flooding

Txc14.cc (1 of 3)

```
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

setto zero & WATCH'ed in initialize() method

END

TOPIC 178

Simulate TCP with BER

In this module

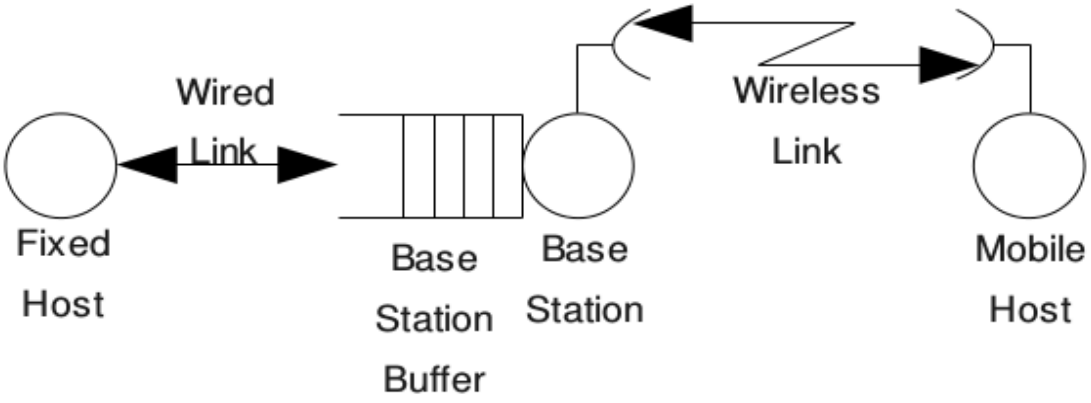
We shall understand

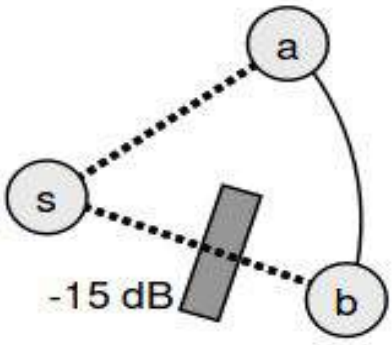
- Reference topology
- The paradox
- Support in Mixim

Reference

Xylomenos, George, et al. "TCP performance issues over wireless links." Communications Magazine, IEEE 39.4 (2001): 52-58.
Tian, Ye, Kai Xu, and Nirwan Ansari. "TCP in wireless environments: problems and solutions." Communications Magazine, IEEE 43.3 (2005): S27-S32.
Köpke, Andreas, et al. "Simulating wireless and mobile networks in OMNeT++ the MiXiM vision." Simulation tools and techniques, 2008.

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

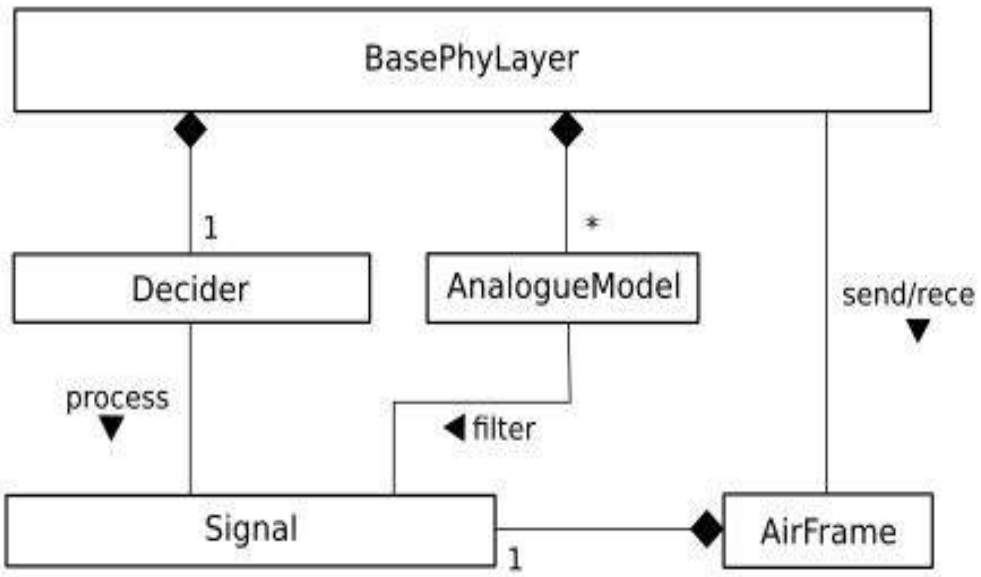
Line of Sight effect

- 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



END

TOPIC 179

Simulate Priority Queues

In this module

We shall understand

- Motivations for PQs
- Operation
- Variants

- Delay Bounds

References

Varga, András. "The OMNeT++ discrete event simulation system." Proceedings of the European simulation multiconference (ESM'2001). Vol. 9. No. S 185., 2001.
Chuck Semeria, "Supporting Differentiated Service Classes: Queue Scheduling Disciplines," Juniper Networks, White Paper 2001.

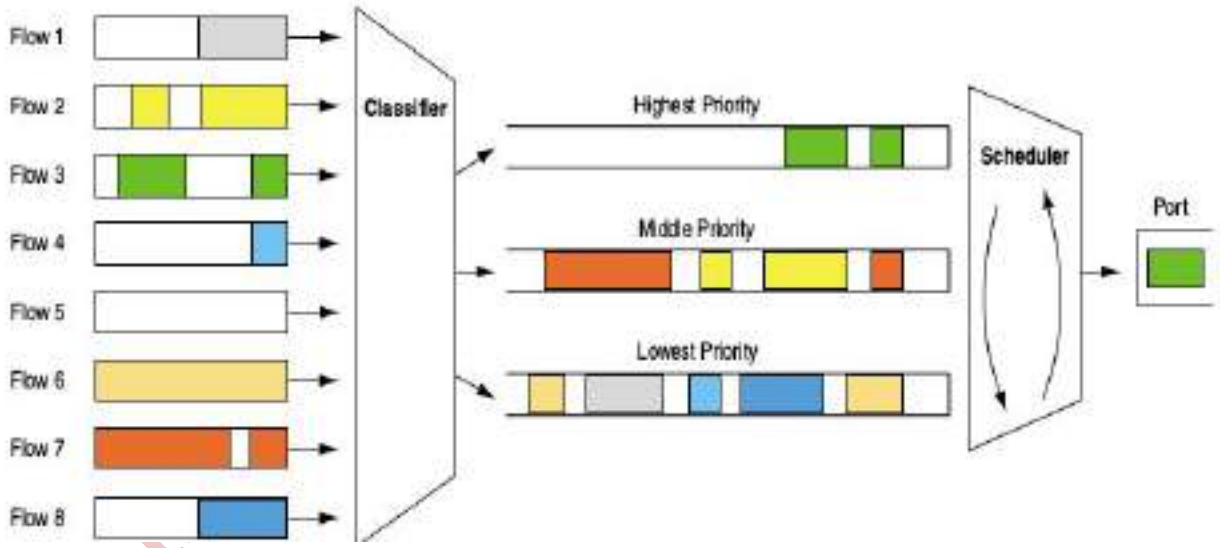
Recall!

- Designed to provide a relatively simple method of supporting differentiated service classes
- Cqueue provides FIFO by default
- Need to be modified for priority queuing

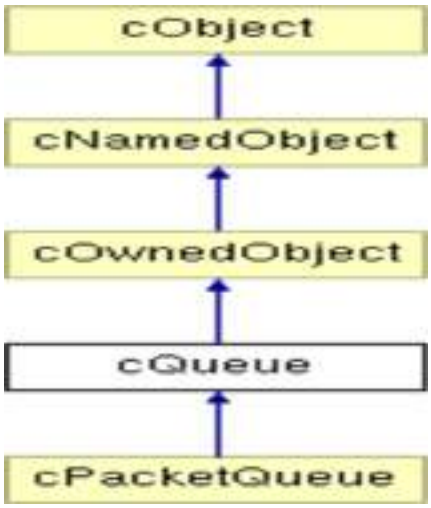
Operation

- 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



CS432 Handouts Made by
Mahjabeen
mahjabeen97869@gmail.com
contact # 0321 2711298

Member Functions

- simple PriorityQueue extends Queue

```
{   @class(PriorityQueue);
}
```

- void setSchedulingPriority(short p);

END

TOPIC 180

DLL Services

In this module

We shall understand

- Need for DLL
- Service models
- Services

Need

- The datalink layer is to the link what the transport layer is to the path

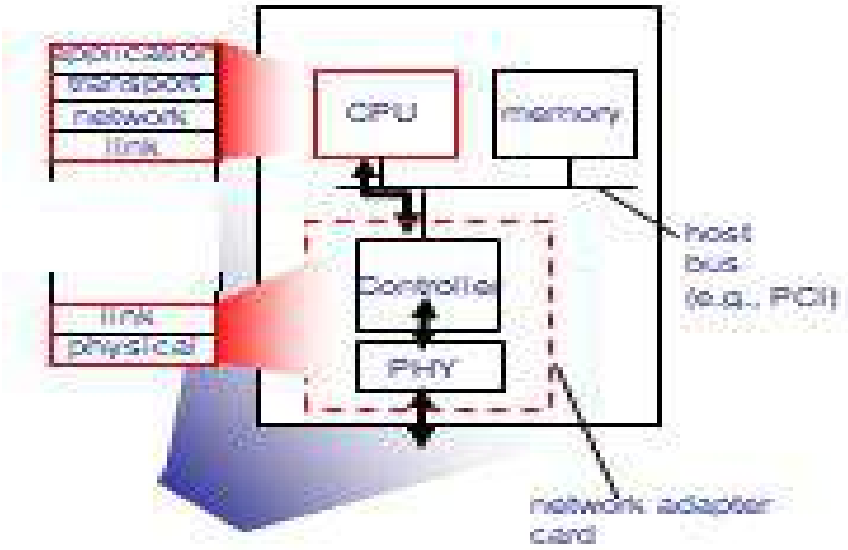
- Upper layer necessitates its behaviour
  - — Reliability
  - — Flow control
  - — Error control
- Corresponding services must exist

Services Models

- Services offered
  - — Reliable (PPP)
  - — Unreliable (Ethernet)
- Point to point
- Multiaccess

Services

- Framing
- Link access
- Error control
- Contention control



END

TOPIC 181

EDEC Techniques

In this module

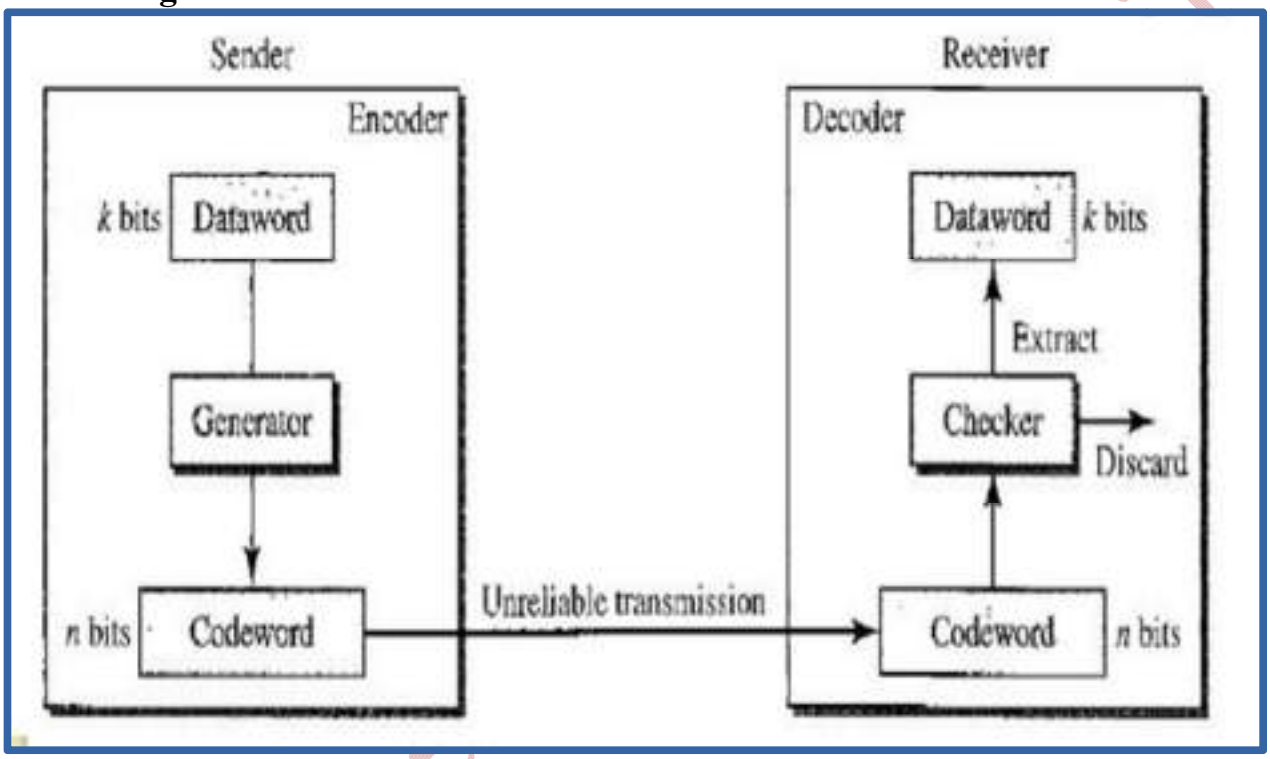
We shall understand

- Strategy of EDEC
- Capability of EDEC
- Constraints

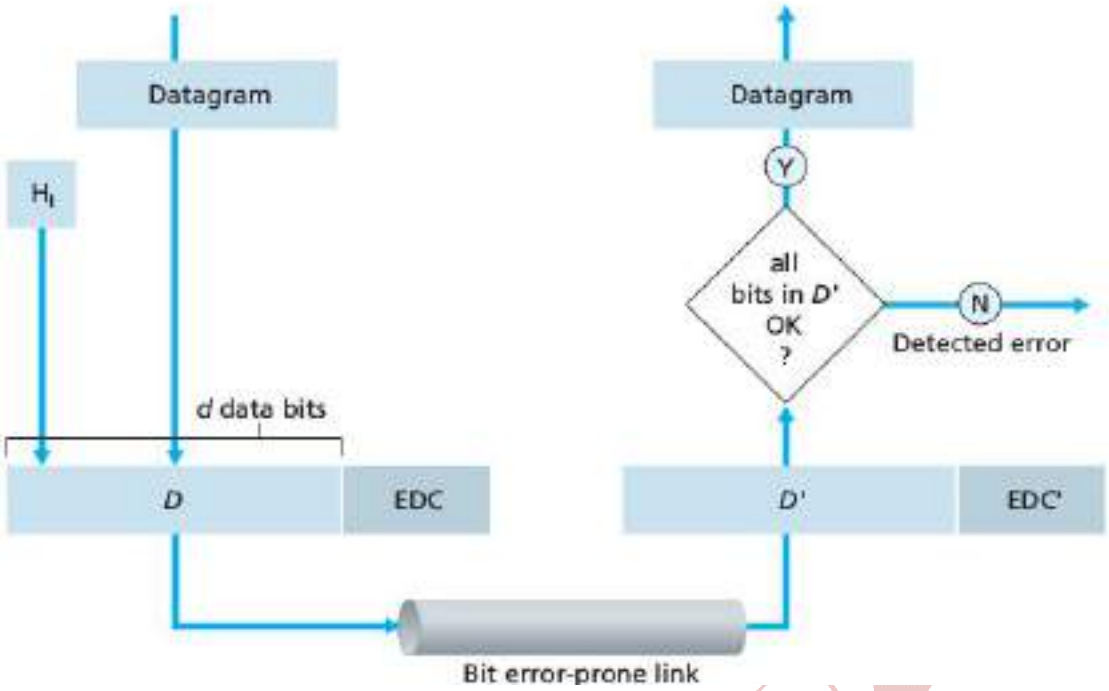
References

http://computing.dcu.ie/~humphrys/Notes/Networks/data.error.html
Graves, Michael. The Complete Guide to Networking And Network+. Cengage Learning, 2003.

Block Diagram



Strategy



Capabilities of EDEC

Given a Hamming Distance of d_{code}

Error detection: $e_d <= d_{code}-1$

Error correction: $e_c <= (d_{code}-1)/2$

e = # of bit flips

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

END

TOPIC 182

Parity Checks

In this module

We shall understand

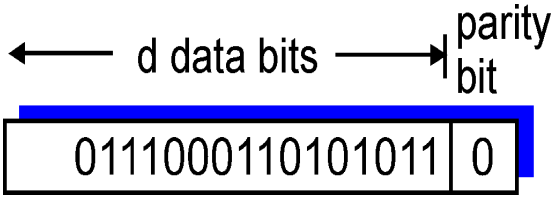
- Operation of single bit parity
- Limitations

References

Bertsekas, Dimitri P., Robert G. Gallager, and Pierre Humblet. Data networks. Vol. 2. New Jersey: Prentice-Hall International, 1992.
http://computing.dcu.ie/~humphrys/Notes/Networks/data.error.html
Graves, Michael. The Complete Guide to Networking And Network+. Cengage Learning, 2003.

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

END

TOPIC 183

Checksumming at DLL

In this module

We shall understand

- Overhead of parity
- Basic idea of checksum
- Operation at DLL vs Transport layer

References

- Bertsekas, Dimitri P., Robert G. Gallager, and Pierre Humblet. Data networks. Vol. 2. New Jersey: Prentice-Hall International, 1992.
- http://computing.dcu.ie/~humphrys/Notes/Networks/data.error.html
- Graves, Michael. The Complete Guide to Networking And Network+. Cengage Learning, 2003.

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

END

TOPIC 184

Horizontal & Vertical Parity

In this module

We shall understand

- 2-D Generalization
- Error correction
- Limitations

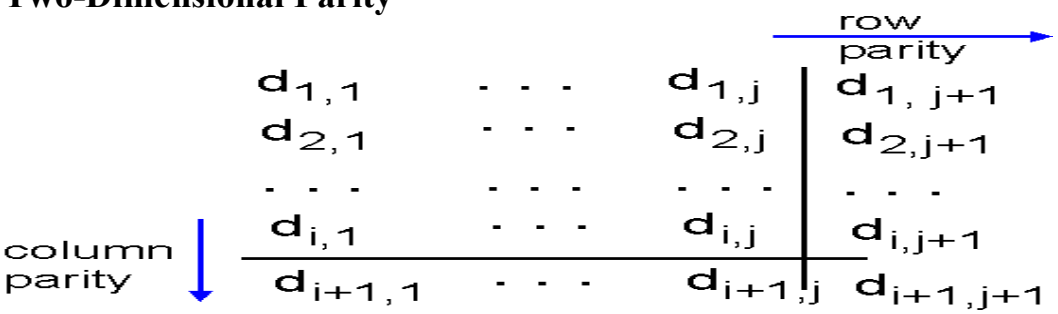
References

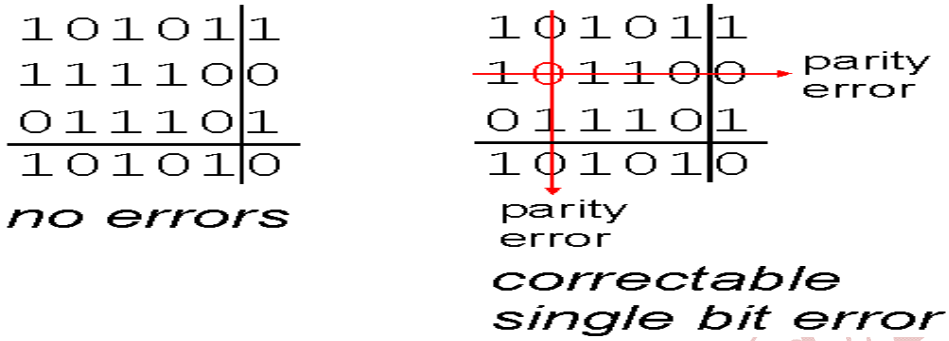
- Bertsekas, Dimitri P., Robert G. Gallager, and Pierre Humblet. Data networks. Vol. 2. New Jersey: Prentice-Hall International, 1992.
- Kurose, James F. Computer Networking: A Top-Down Approach Featuring the Internet, 3/E. Pearson Education India, 2005.

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

END

TOPIC 185

Cyclic Redundancy Check

In this module

We shall understand

- Principle
- Modulo 2
- Operation
- Applications

References

Bertsekas, Dimitri P., Robert G. Gallager, and Pierre Humblet. Data networks. Vol. 2. New Jersey: Prentice-Hall International, 1992.
Kurose, James F. Computer Networking: A Top-Down Approach Featuring the Internet, 3/E. Pearson Education India, 2005.

Principle (1 of 2)

- 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)

Principle (2 of 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

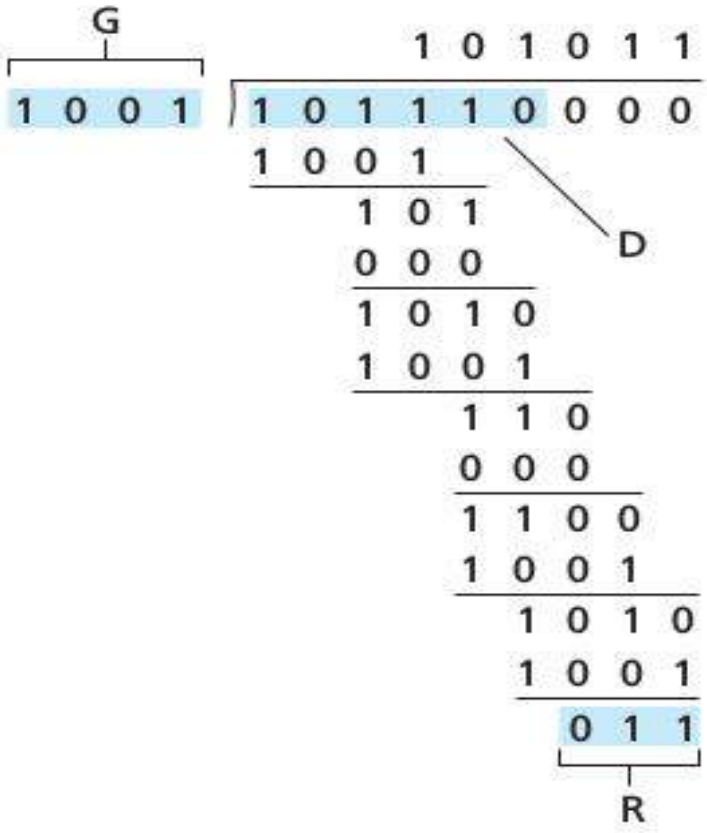
- $D \cdot 2^r$ XOR R = nG
  - — Left shift by r then append R
  - — Multiple of Generator

Mathematical manipulation

- $D \cdot 2^r$ = nG XOR R
- If we divide $D \cdot 2^r$ by G, want remainder R to satisfy
- R = remainde r[$\frac{D \cdot 2^r}{G}$]

D = 101110, d = 6, G = 1001, r =3

9 bits transmitted in this case are 101110 011



Applications

- CRC can detect all burst errors less than r+1 bits
- Widely used in practice
  - — Ethernet
  - — 802.11 WiFi
  - — ATM

END

TOPIC 186

Throughput of MAC

In this module

We shall understand

- An ideal MAC channel
- Contention issues

References

Lee, Ha Cheol. A MAC Throughput in the Wireless LAN. INTECH Open Access Publisher, 2012.

Kurose, James F. Computer Networking: A Top-Down Approach Featuring the Internet, 3/E. Pearson Education India, 2005.

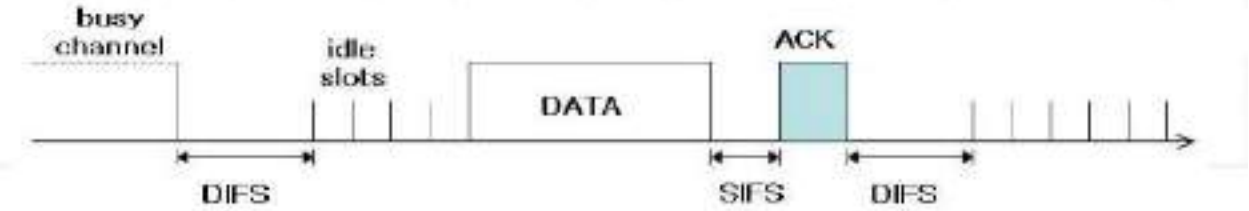
Ideal MAC (1 of 2)

- 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

Ideal MAC (2 of 2)

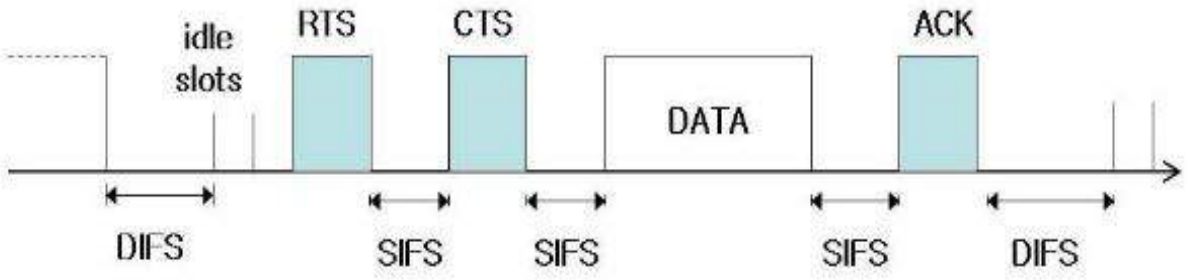
- Fully decentralized
  - No special node to coordinate TXNs
- No synchronization of clocks
- No slots
- Simple

Access Methods (1 of 2)



(a) Basic access method

Access Methods (2 of 2)



(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

END

TOPIC 187

Channel Partitioning

In this module

We shall understand

- Basic idea
- TDM
- FDM

References

Lee, Ha Cheol. A MAC Throughput in the Wireless LAN. INTECH Open Access Publisher, 2012.

Kurose, James F. Computer Networking: A Top-Down Approach Featuring the Internet, 3/E. Pearson Education India, 2005.

Basic Idea

- Divide channel into smaller “pieces”
  - — Time slots
  - — Frequency
  - — Code
  - — Space
- Exclusive use

TDM

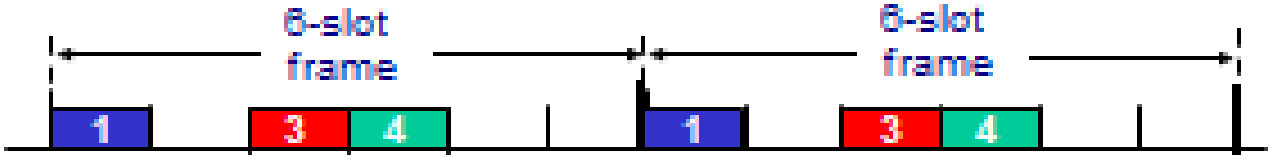
- Time divion multiplexing
- Access to channel in "rounds"
- Each station gets fixed length slot
- Length = packets trans time) in each round
  - Unused slots go idle

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

$$f = \sum_j \alpha_j x_j$$



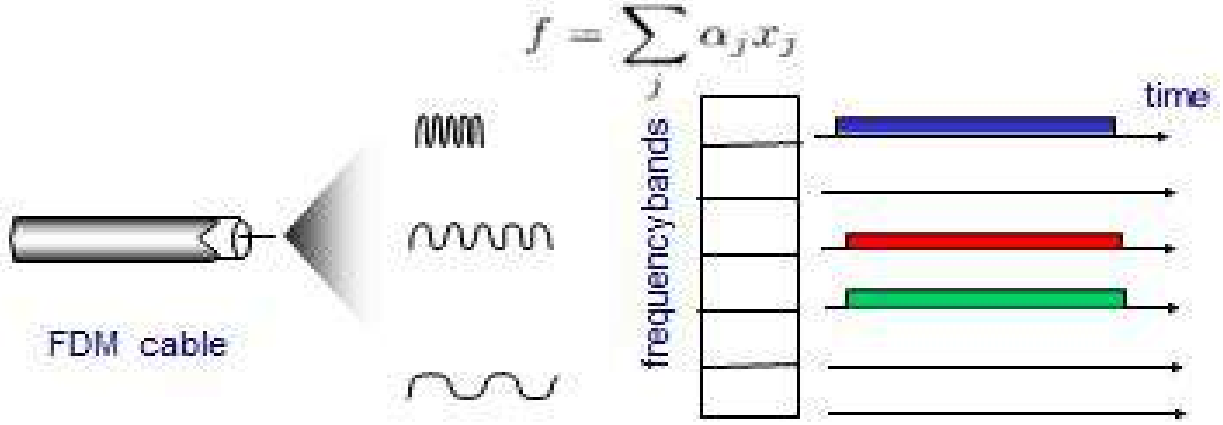
FDM

- 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



WEEK 14

TOPIC 188

Random Access Protocols

In this module

We shall understand

- Basic idea
- Packet attempts
- Managing collisions

References

Anurag Kumar et al. Communication Networking—An Analytical Approach. Morgan Kaufmann Publishers. 2004.

Kurose, James F. Computer Networking: A Top-Down Approach Featuring the Internet, 3/E. Pearson Education India, 2005.

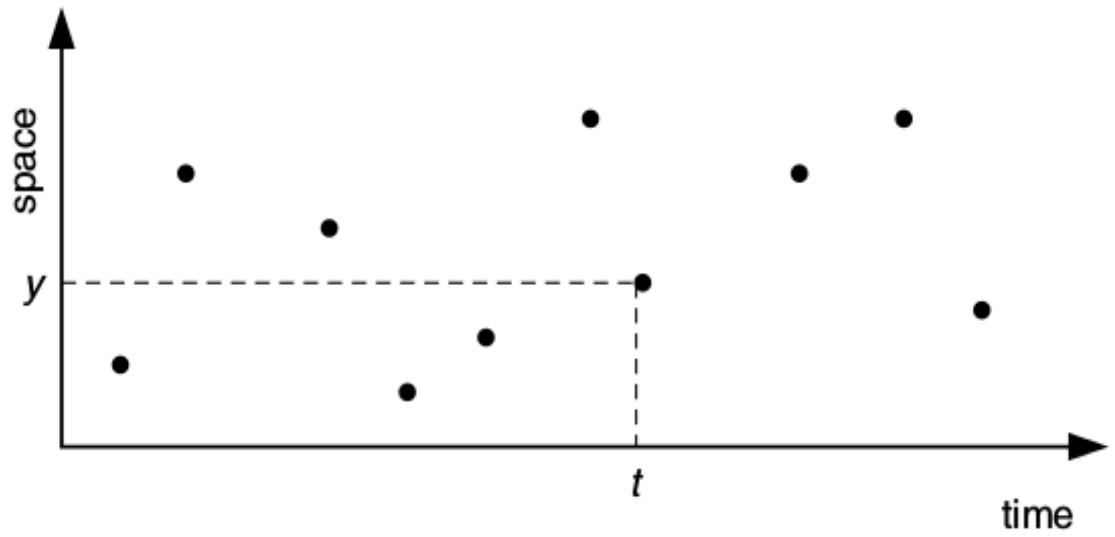
Basic Idea

- 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

END

TOPIC 189

ALOHA

In this module

We shall understand

- Basic idea
- Packet transmissions
- Probability of success

References

Anurag Kumar et al. Communication Networking—An Analytical Approach, Morgan Kaufmann Publishers. 2004.

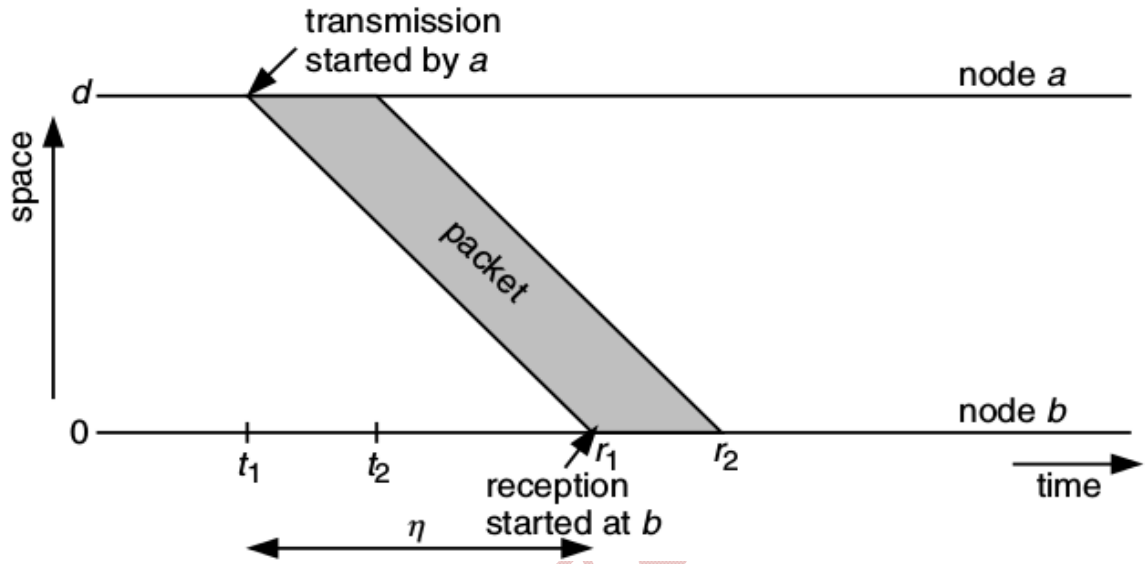
Kurose, James F. Computer Networking: A Top-Down Approach Featuring the Internet, 3/E. Pearson Education India, 2005.

Basic Idea

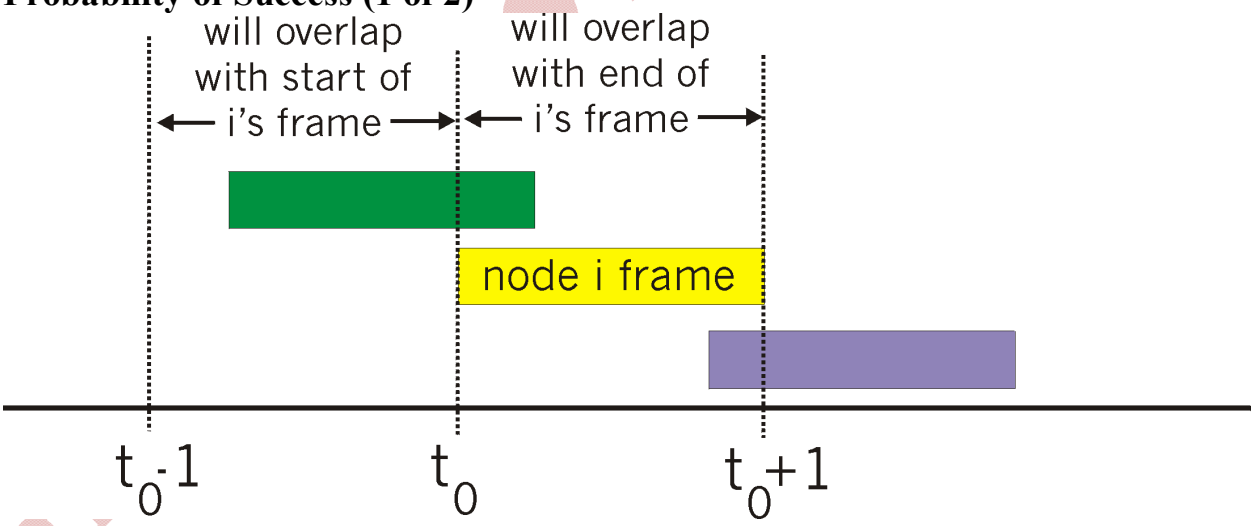
- Just say as you like!
- Whenever and wherever
  - — Simplest
  - — No synchronization

Packet transmissions are independent

- Packet reception success dependent upon others not transmitting

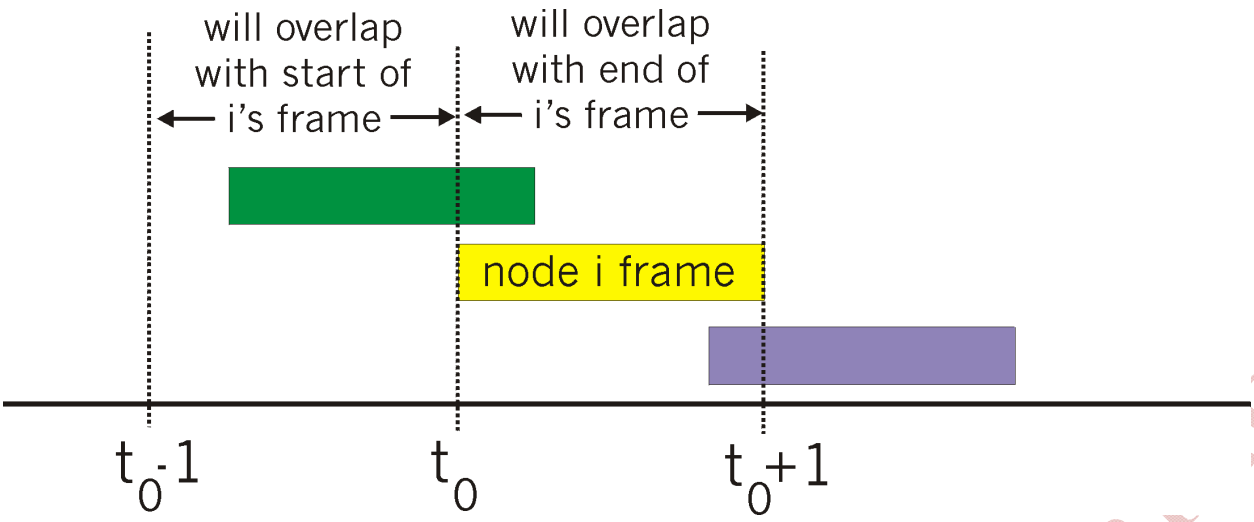


Probability of Success (1 of 2)



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 $[t_0-1,t_0$

Probability of Success (2 of 2)



$p \cdot (1-p)^{N-1} \cdot (1-p)^{N-1}$
$= p \cdot (1-p)^{2(N-1)}$
[choosing optimum p and n very large]
$= 1/(2e) = .18$

END

TOPIC 190

Slotted ALOHA

In this module

We shall understand

- Assumptions
- Operation
- Performance & limitations
- Efficiency

References

Anurag Kumar et al. Communication Networking—An Analytical Approach, Morgan Kaufmann Publishers. 2004.
Kurose, James F. Computer Networking: A Top-Down Approach Featuring the Internet, 3/E. Pearson Education India, 2005.

Basic Idea

- Minimize collisions
  - Through synchronization
  - Through frame size delimiting

Assumption

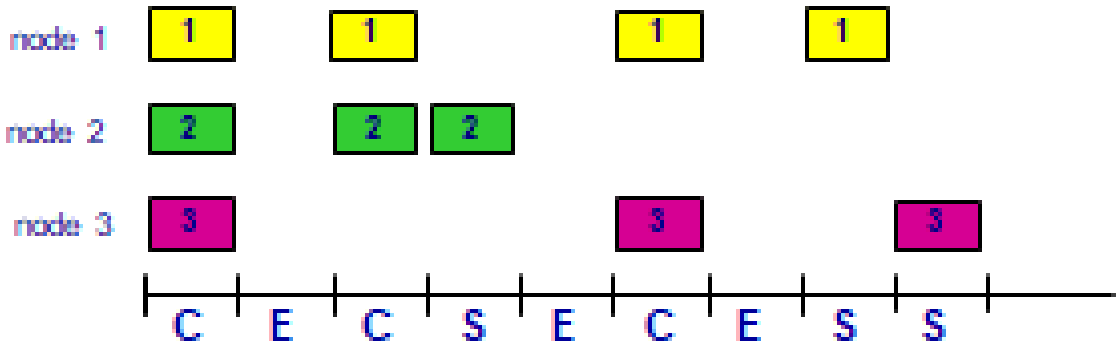
- All frames same size
- Time divided into equal size slots
  - Time to transmit 1 frame
- Nodes start to transmit only slot beginning
- Nodes are synchronized
  - If 2 or more nodes transmit in slot, all nodes detect collision

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?



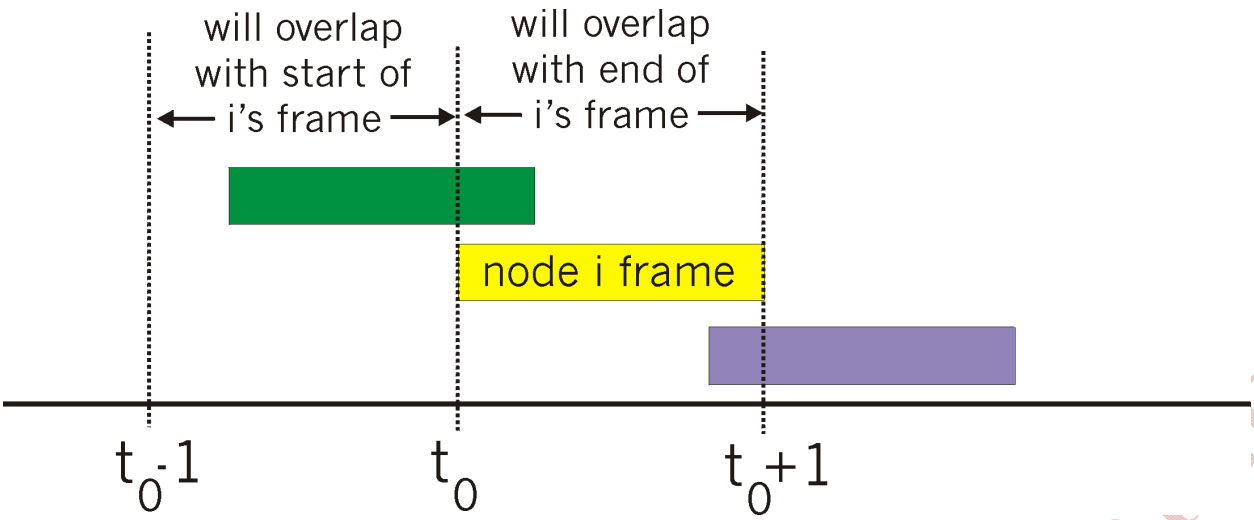
Pros

- 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

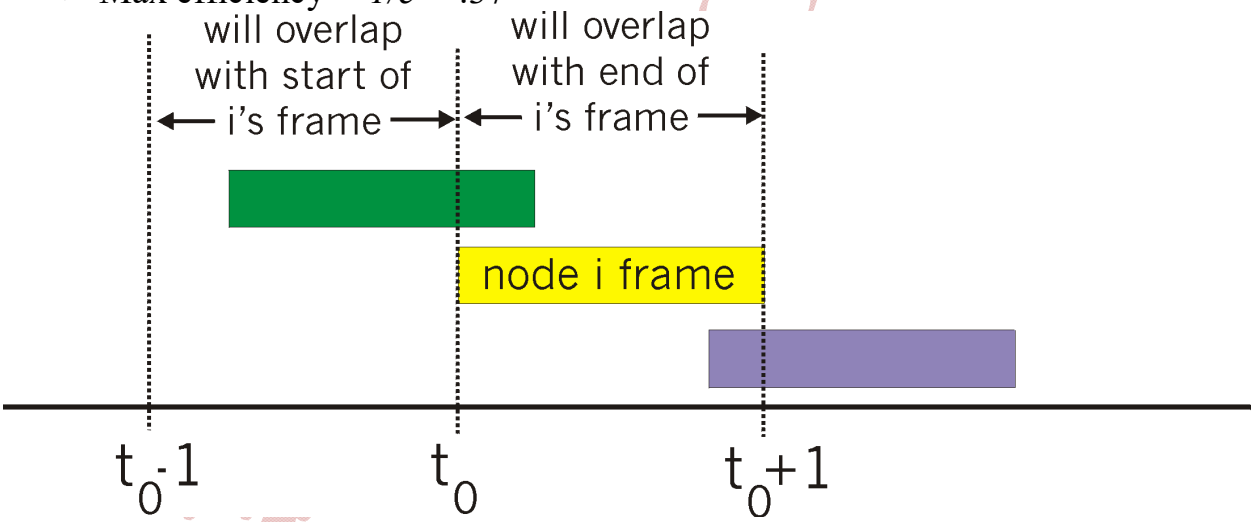
Probability of Success (1 of 2)



- *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 = $Np(1-p)^{N-1}$

Probability of Success (2 of 2)

- 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



END

TOPIC 191

CSMA/CD

In this module

We shall understand

- Basic idea
- CSMA/CD states
- BEBA

References

Andrew S Tanenbaum. Computer Networks (4th Edition). Pearson Education
Kurose, James F. Computer Networking: A Top-Down Approach Featuring the Internet, 3/E. Pearson Education India, 2005.

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

carrier sense multiple access with collision detection

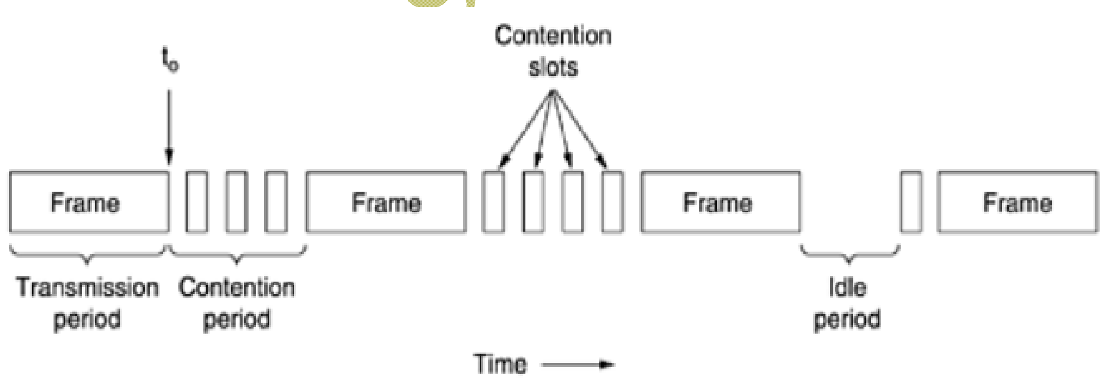
CSMA/CD States

this is protocol used in ethernet network to manage access

- Contention
- Transmission

to the networkchannel and prevent collision between device

- 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

END

TOPIC 192

CSMA/CD Efficiency

In this module

We shall understand

- Factors affecting efficiency
- Performance of variants

References

Andrew S Tanenbaum. Computer Networks (4th Edition). Pearson Education
Kurose, James F. Computer Networking: A Top-Down Approach Featuring the Internet, 3/E. Pearson Education India, 2005.

Factors Affecting (1 of 2)

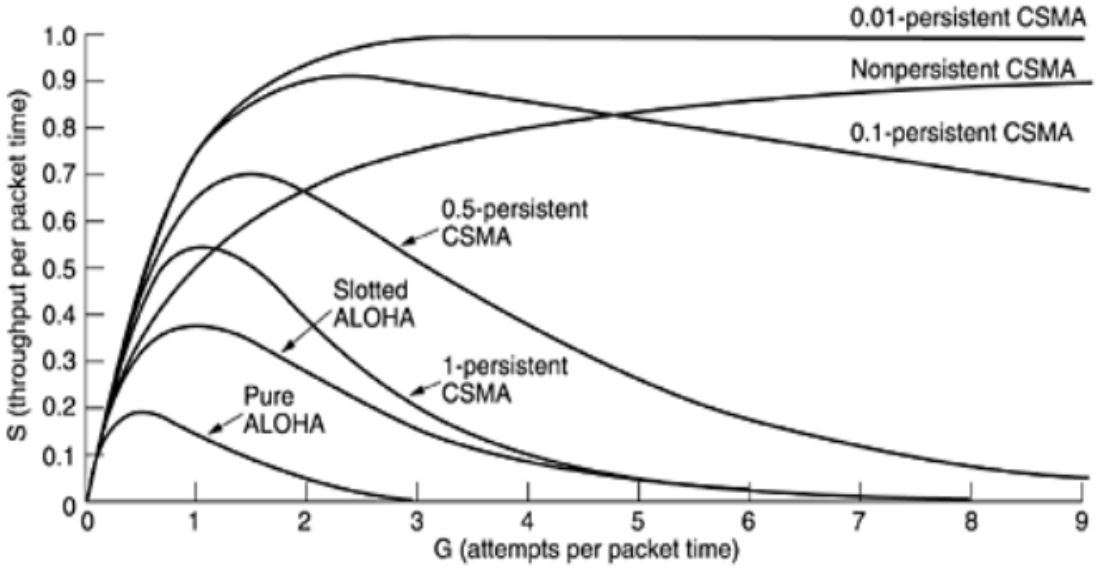
- T_{prop} = max prop delay between 2 nodes in LAN
- t_{trans} = time to transmit max-size frames
- Full load
  - Worst
- Partial load
  - Increases till a range
- No load
  - Poor performance

Factors affecting (2 of 2)

$$Efficiency = \frac{1}{1+5t_{prop}/t_{trans}}$$

- Efficiency goes to 1
  - As t_{prop} goes to 0
  - As t_{trans} goes to infinity
- Efficiency goes to 0 vice versa

Performance with Variants



END

TOPIC 193

Min Frame Size Computation

In this module

We shall understand

- What is minimum size?
- Factors for frame size computation
- Ethernet calculation example

References

Andrew S Tanenbaum. Computer Networks (4th Edition). Pearson Education
Kurose, James F. Computer Networking: A Top-Down Approach Featuring the Internet, 3/E. Pearson Education India, 2005.

Min Frame Size

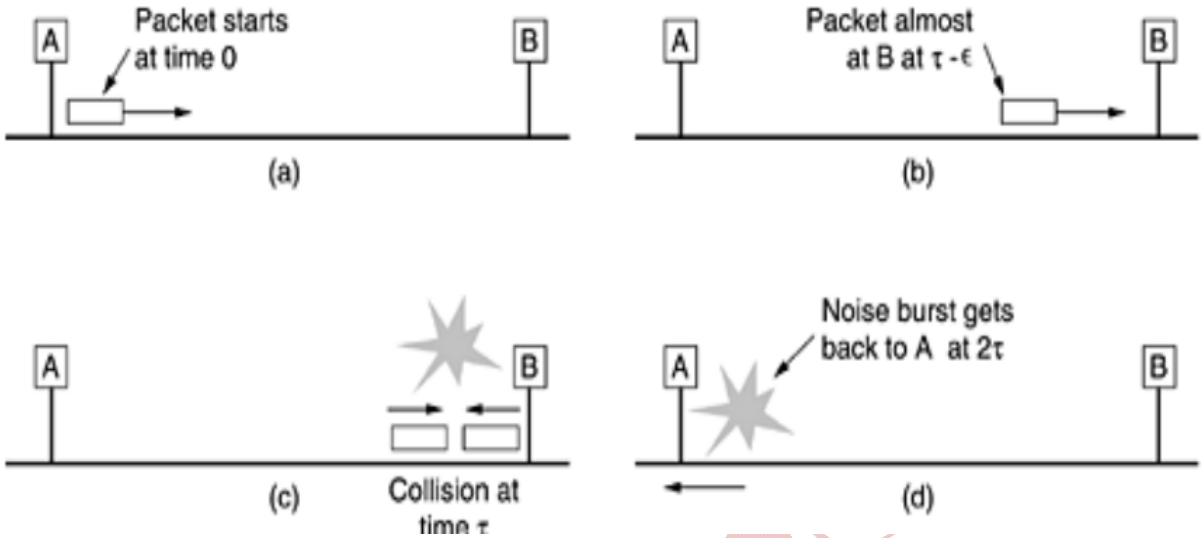
- Ethernet recommends 64 Bytes
  - Inclusive of headers
- If the data portion of a frame < 46 bytes
- Pad field is used to fill out the frame to min. size
- Wireless networks necessitate lesser size
  - ReTx cost to be minimized
- Optical networks require longer frames

Reasons (1 of 3)

- 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

Reasons (2 of 3)

- Collision detection can take as long as 2 τ



Reasons (3 of 3)

- 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

END

TOPIC 194

Max Frame Size Computation

In this module

We shall understand

- Ethernet max frame size
- Variable length packet
  - Overhead

- Pipelining
- Transmission errors

References

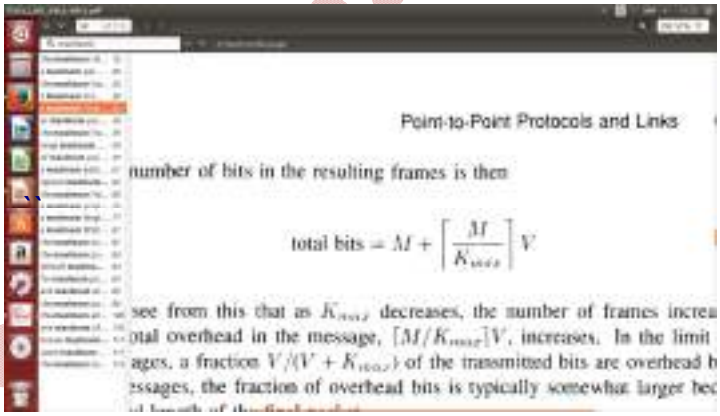
Andrew S Tanenbaum. Computer Networks (4th Edition). Pearson Education
Bertsekas, Dimitri P., Robert G. Gallager, and Pierre Humblet. Data networks. Vol. 2. New Jersey: Prentice-Hall International, 1992.

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 (1 of 3)

- Overhead
- Pipelining
- Transmission errors



Overhead of Variable Length Packet (1 of 2)

- Each frame contains V as overhead bits
- K_{max}:Max length of packet
- Message of length M
  - Broken down into M/K_{max} packets

Overhead of Variable Length Packet (2 of 2)

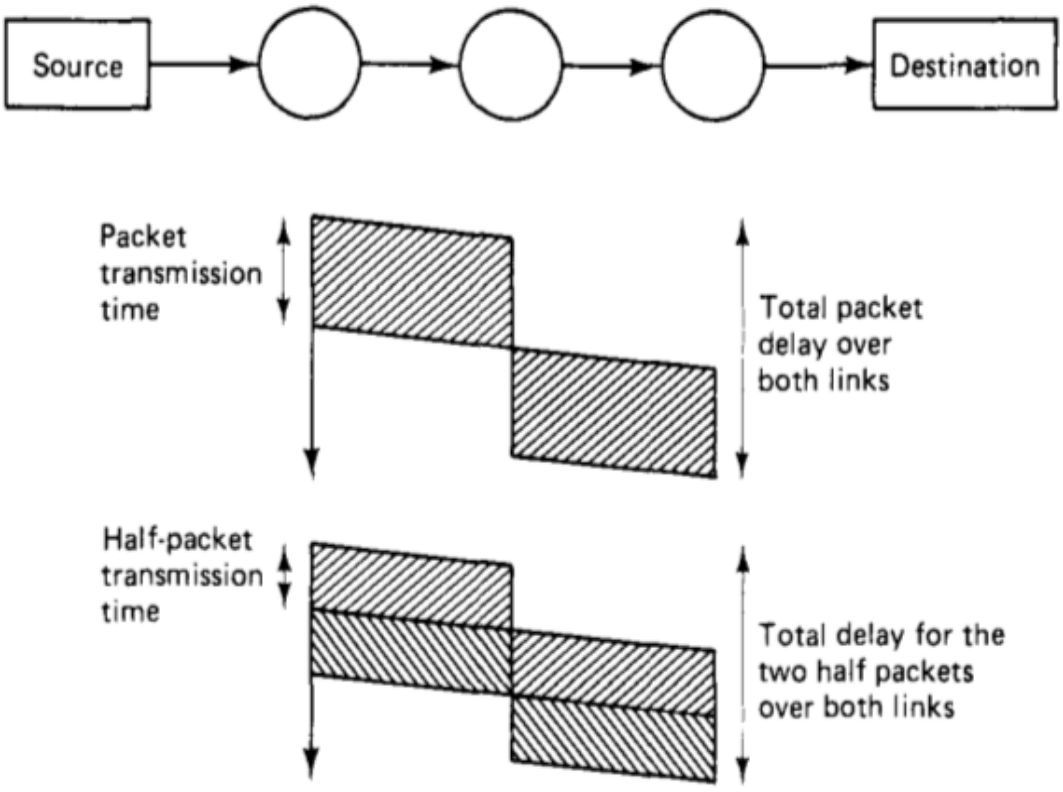
- As K_{max} decreases, the number of frames increases
  - Thus the total overhead in the message
- Processing load in a multiple hop network increases
-

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

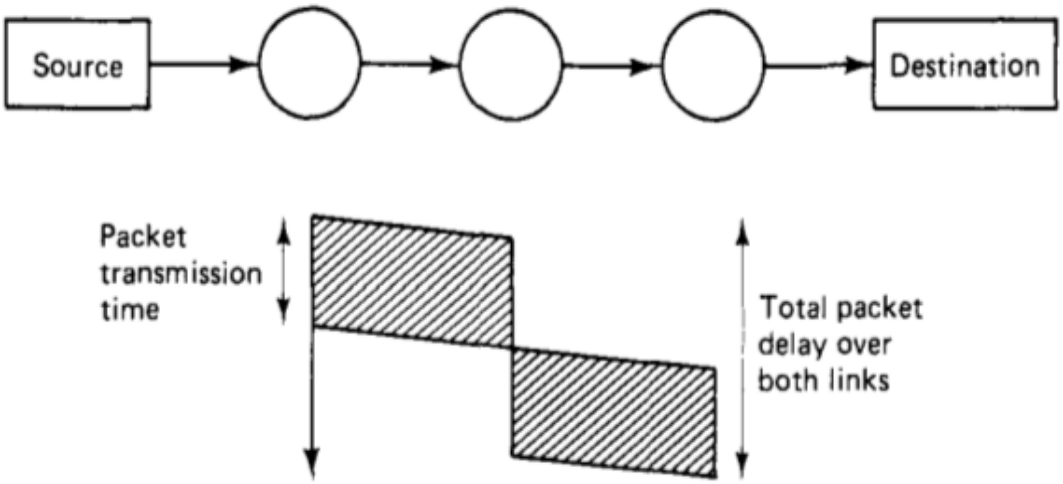
Pipelining Scenario

- Decreasing delay by shortening packets to take advantage of pipelining



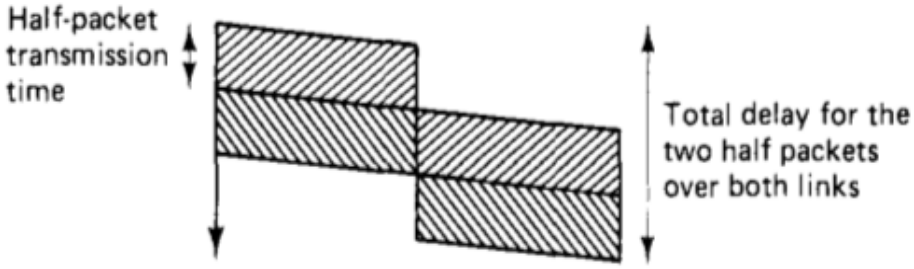
Pipelining Scenario

- 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

END

TOPIC 195

Fixed Frame Size Computation

In this module

We shall understand

- Why fixed frame?
- Fixed frame (cell) networks
- Considerations

References

Bertsekas, Dimitri P., Robert G. Gallager, and Pierre Humblet. Data networks. Vol. 2. New Jersey: Prentice-Hall International, 1992.

Maglaris, Basil S., and Mischa Schwartz. "Optimal fixed frame multiplexing in integrated line- and packet-switched communication networks." Information Theory, IEEE Transactions on 28.2 (1982): 263-273.

Why Fixed Frame?

- Expectability of performance
  - Latency
  - Throughput

  - Cell loss
- Resource pre-emption

Considerations (1 of 2)
- How much should be the fixed size?
  - Processing at the nodes
- Header to payload efficiency
  - Padding requirement

Considerations (2 of 2)
- 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

END

TOPIC 196

Multi-Protocol Label Switching

In this module

We shall understand
- Introduction
- Basic idea
- Network components
- Enhanced forwarding
- Important parameters

References

El-Sharkawy, Aamani Nemturand Mohamed. "Performance Evaluation of MPLS Network with Failure Protection using OPNET Modeler." Performance Evaluation 4.11 (2015).

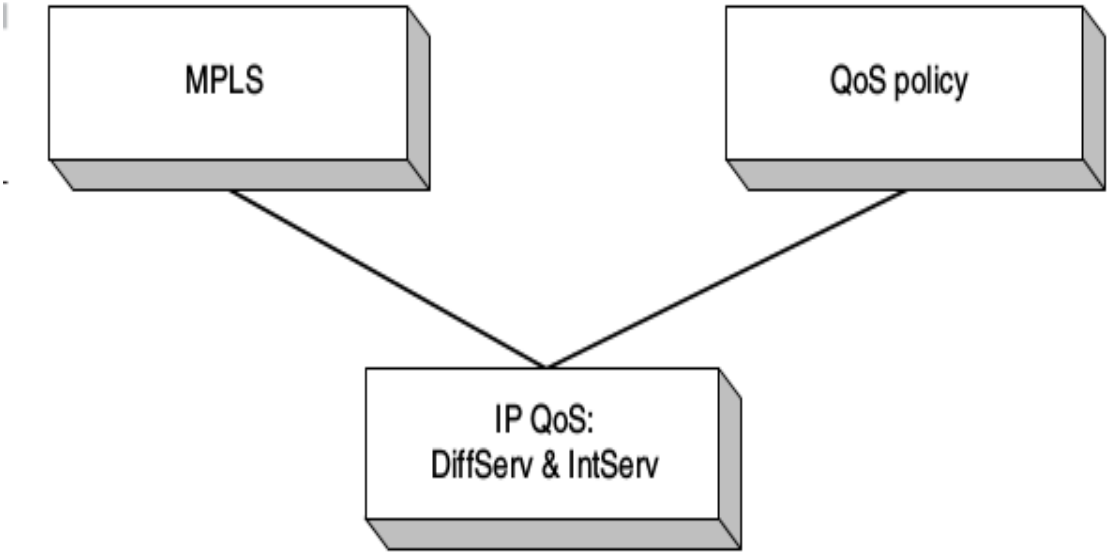
Ahmad, Suhail, Wajid Ali Hamdani, and Mohsin Hassan Magray. "Performance Evaluation of IPv4 and IPv6 over MPLS using OPNET." International Journal of Computer Applications 125.3 (2015).

Intro (1 of 2)

- 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

Intro (2 of 2)

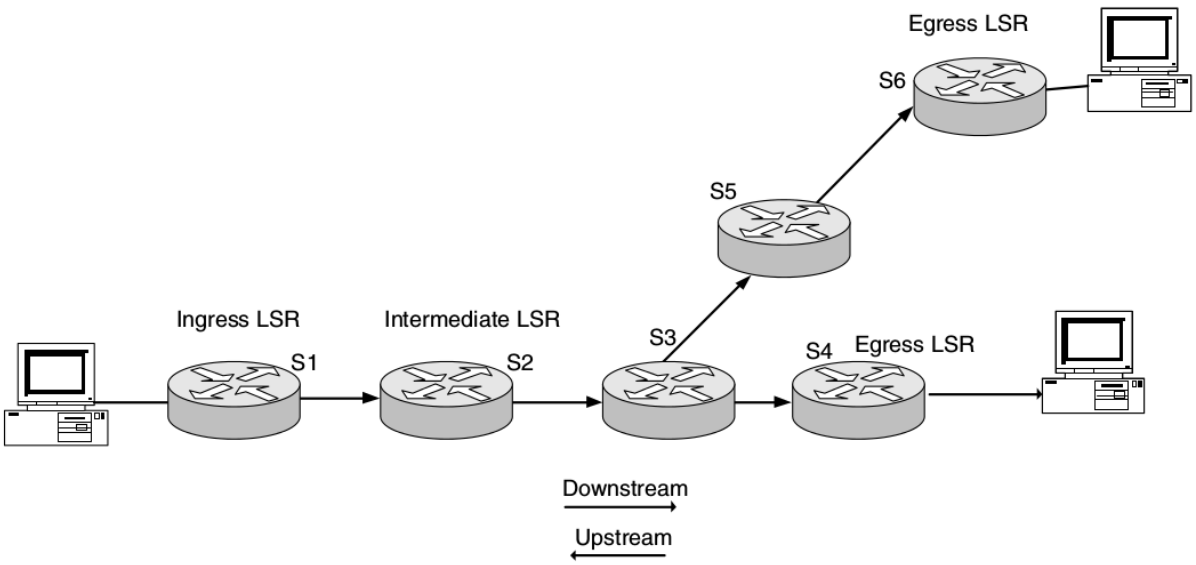
- Although idea was to facilitate fast packet switching
  - Main goal: support traffic engineering and QoS



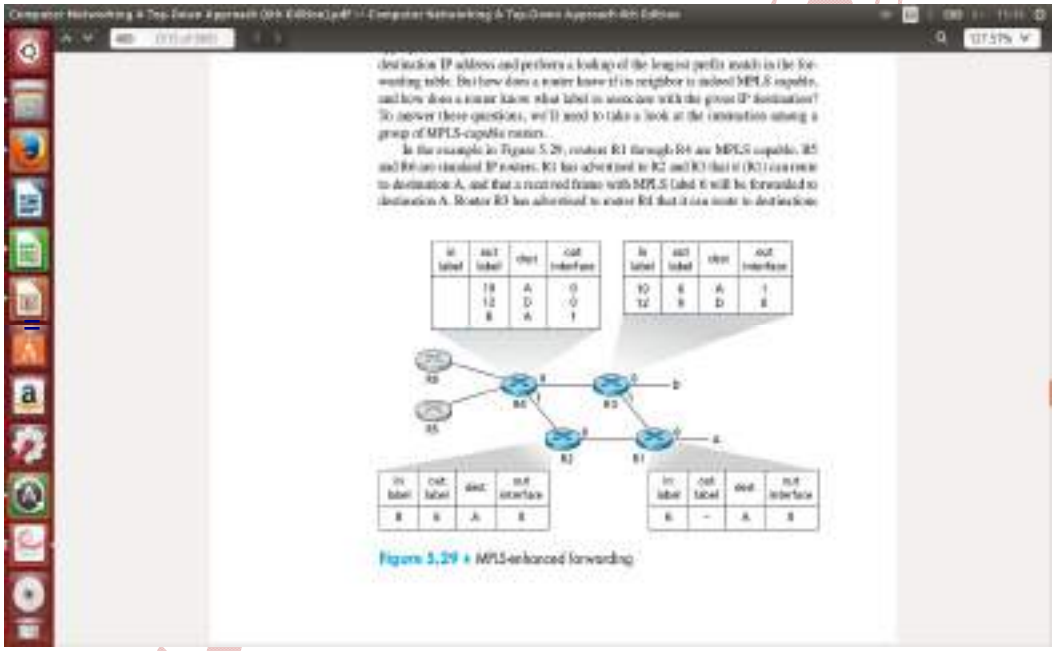
Basic Idea

- 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

END

TOPIC 197

Load Balancing in Data Centre

In this module

We shall understand

- Introduction
- Basic idea
- Hierarchical design
- Limited host-to-host capacity
- Fully connected topology

References

Greenberg, Albert, et al. "VL2: a scalable and flexible data center network." ACM SIGCOMM computer communication review. Vol. 39. No. 4. ACM, 2009.
Gandhi, Rohan, et al. "Duet: Cloud scale load balancing with hardware and software." ACM SIGCOMM Computer Communication Review 44.4 (2015): 27-38.

Intro (1 of 2)

- 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

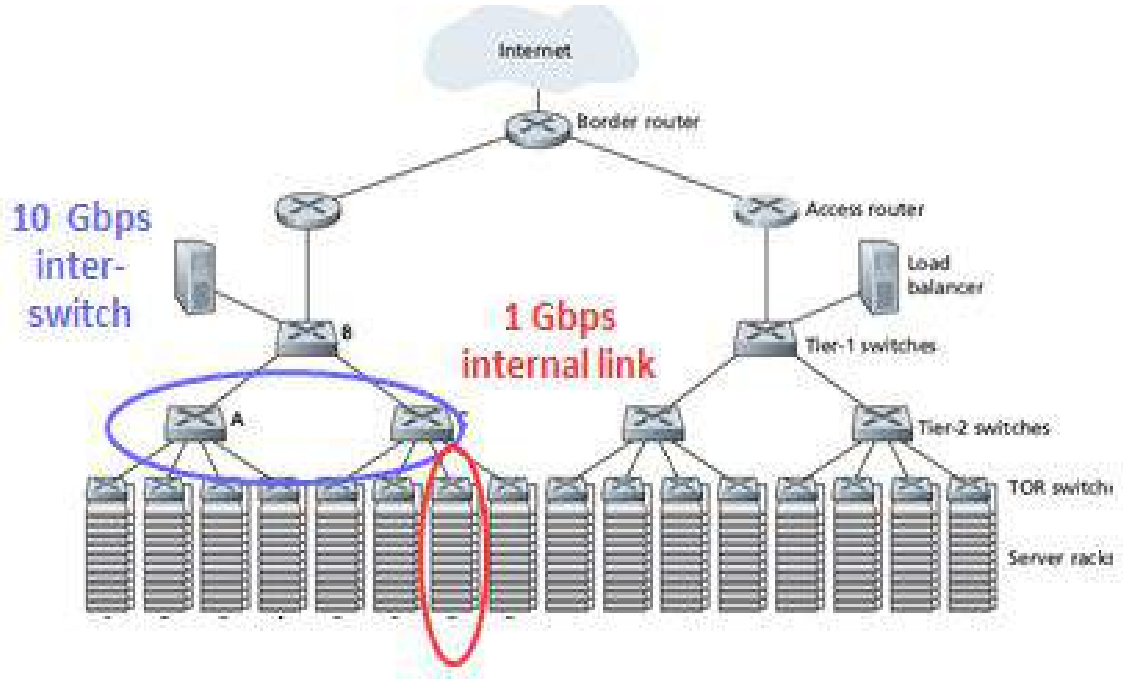
Intro (2 of 2)

- 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



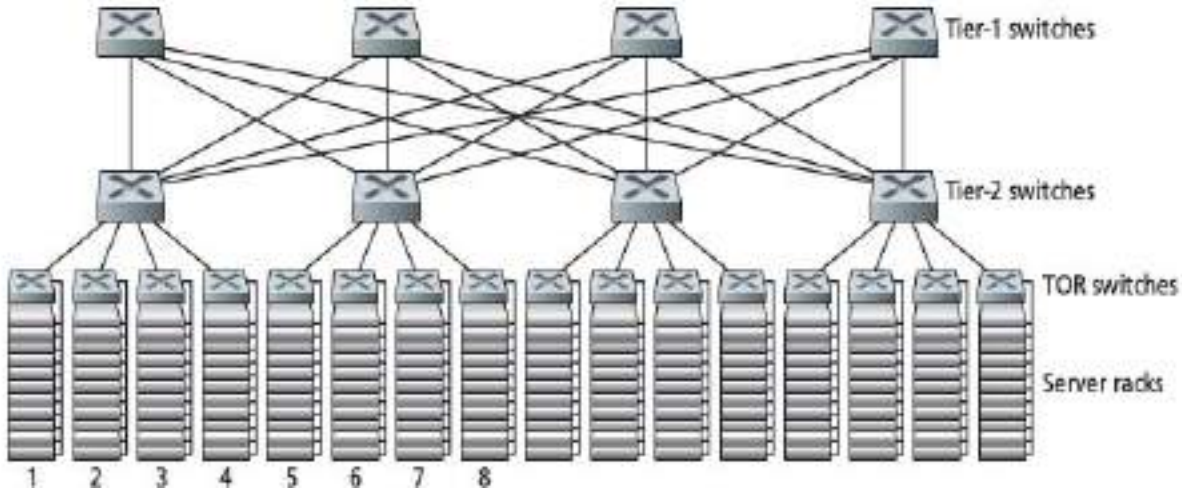
Limited Host to Host Connectivity (1 of 2)

- 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

Limited Host to Host Connectivity (2 of 2)

- 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



END

TOPIC 198

Correctness of Stop and Wait

In this module

We shall understand

- Stop and wait operation
- Unnumbered frames and ACKs Using seq nos

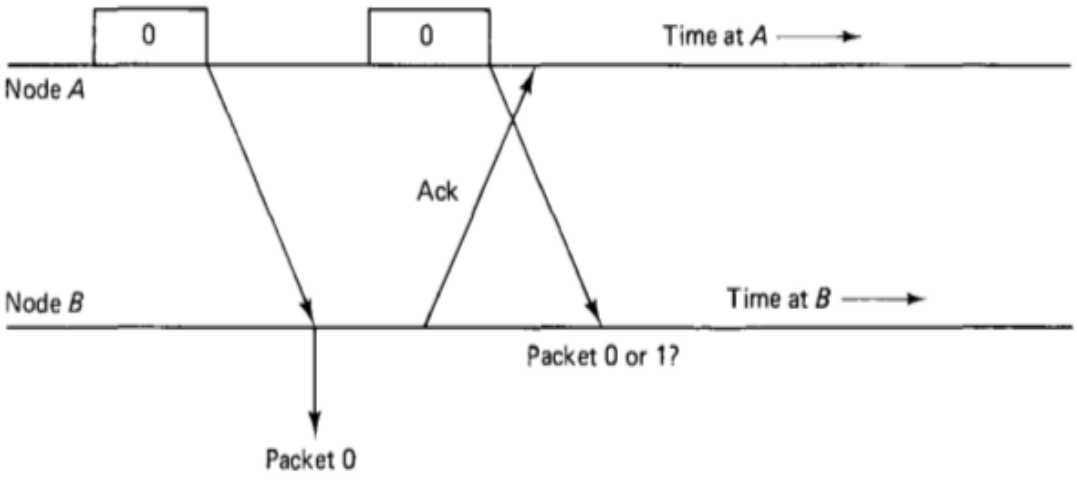
References

- Anagnostou, Miltiades E., and Emmanuel N. Protonotarios. "Performance analysis of the selective repeat ARQ protocol." Communications, IEEE Transactions on 34.2 (1986): 127-135.
- Bertsekas, Dimitri P., Robert G. Gallager, and Pierre Humblet. Data networks. Vol. 2. New Jersey: Prentice-Hall International, 1992.

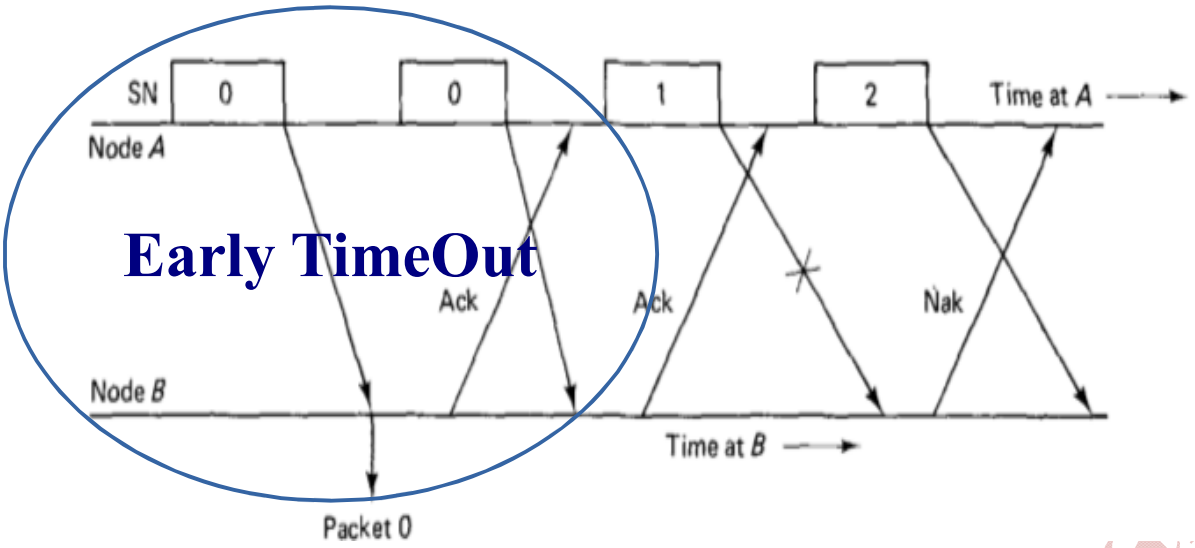
Stop and Wait Operation

- 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

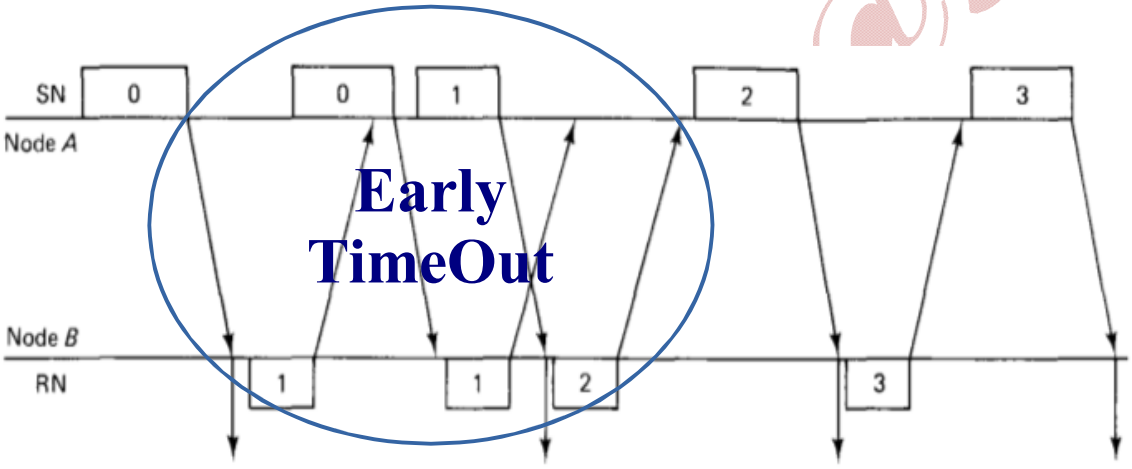
Problems with Unnumbered Packets

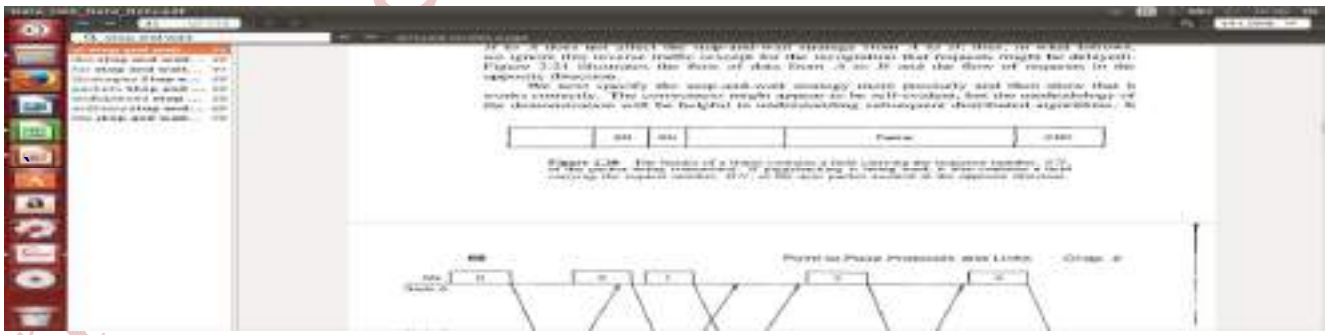


Problems with Unnumbered ACKs



Problems Management using Seq. Nos.





END

TOPIC 199

Efficiency of Go Back N

In this module

We shall understand

- Recalling Go back N
- Efficiency of Stop and wait
- Utilization (efficiency) of Go back N
- Limitations

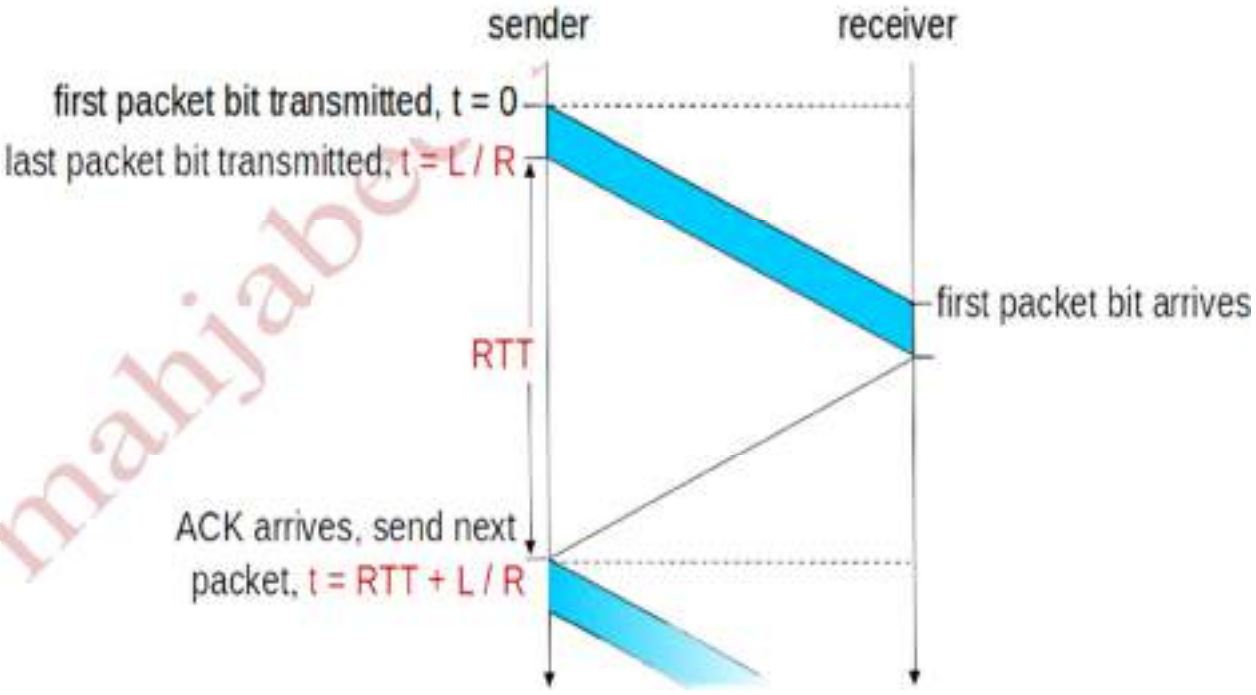
References

Anagnostou, Miltiades E., and Emmanuel N. Protonotarios. "Performance analysis of the selective repeat ARQ protocol." Communications, IEEE Transactions on 34.2 (1986): 127-135.
Bertsekas, Dimitri P., Robert G. Gallager, and Pierre Humblet. Data networks. Vol. 2. New Jersey: Prentice-Hall International, 1992.

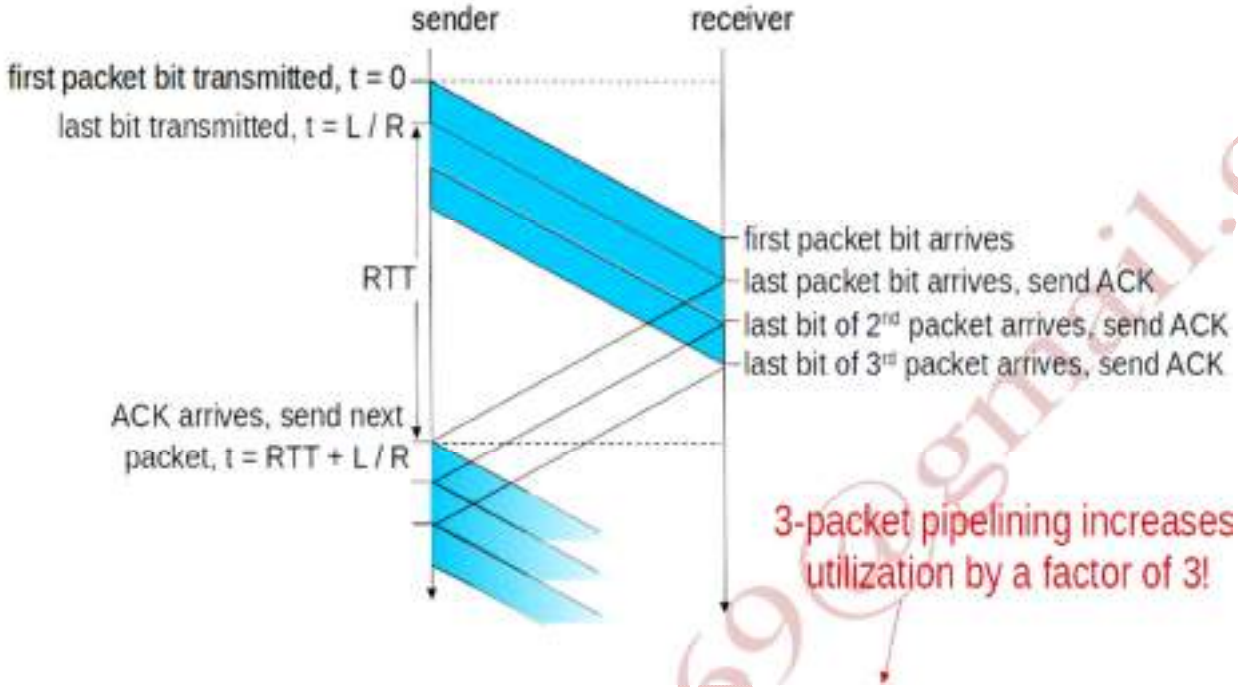
Stop and Wait Operation

- 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 (1 of 2)



Utilization of Go-Back N under No Loss



$$U_{sender} = \frac{3L/R}{RTT + L/R} = \frac{.0024}{30.008} = 0.00081$$

END

TOPIC 200

Character-based Framing

In this module

We shall understand

- Usage of character codes
- Synchronous idle
- Legality of control characters within data Transparent Mode

References

Bertsekas, Dimitri P., Robert G. Gallager, and Pierre Humblet. Data networks. Vol. 2. New Jersey: Prentice-Hall International, 1992.

Character Codes

- Character codes such as ASCII provide binary representations

- Keyboard characters and terminal control characters
- Also for various communication control characters

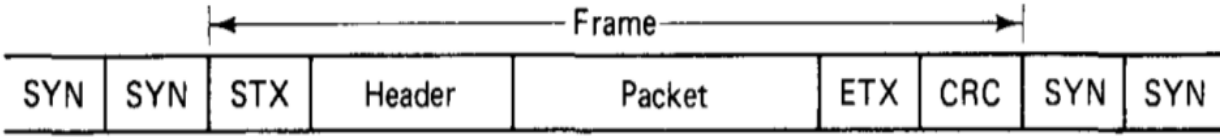
SYN Idle

- 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



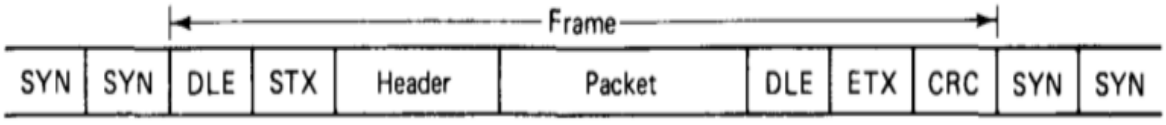
SYN = Synchronous idle
STX = Start of text
ETX = End of text

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

END

TOPIC 201

Bit-oriented Framing

In this module

We shall understand

- Bit-oriented frames
- Flags
- Bit stuffing example

References

Bertsekas, Dimitri P., Robert G. Gallager, and Pierre Humblet. Data networks. Vol. 2. New Jersey: Prentice-Hall International, 1992.

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 (1 of 2)

- 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 5th 1 in transmitted stream

Flags (2 of 2)

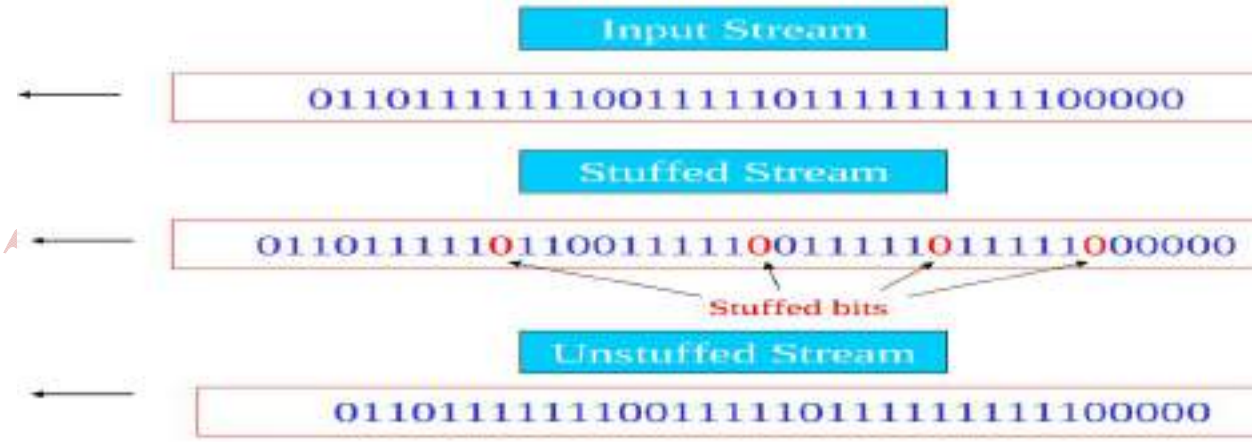
- DLL at receiver removes inserted 0s by looking for seq of 5 1s followed by stuffed 0s

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



END

WEEK 15

TOPIC 202

Framing with Errors

In this module

We shall understand

- Problems with framing
- Errors with flagging
- False flag example

References

Bertsekas, Dimitri P., Robert G. Gallager, and Pierre Humblet. Data networks. Vol. 2. New Jersey: Prentice-Hall International, 1992.

Problems with framing

- Several peculiar problems arise
- When errors corrupt the framing information on the communication link
  - Flagging
  - CRC
  - Length field

Flags (1 of 2)

- 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

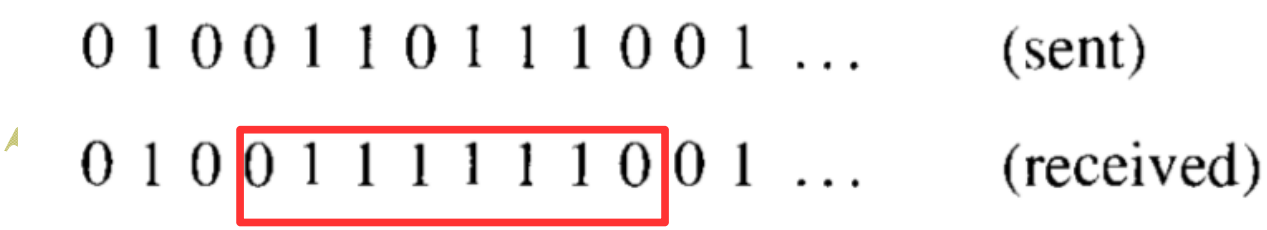
Flags (2 of 2)

- 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



- Called the data sensitivity problem of DLC

is a protocol used for establishing and managing connections between devices

- 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

END

TOPIC 203

Length Fields

In this module

We shall understand

- Purpose of length field
- Overhead of length field
- Problems with length field
- Partial solutions

References

Bertsekas, Dimitri P., Robert G. Gallager, and Pierre Humblet. Data networks. Vol. 2. New Jersey: Prentice-Hall International, 1992.

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

END

TOPIC 204

Topology and Connectivity

In this module

We shall understand

- Topology
- Ad hoc Networks
- Spatial reuse vs connectivity
- Feasibility region-based representation

References

Kumar, Anurag, D. Manjunath, and Joy Kuri. Communication networking: an analytical approach. Elsevier, 2004.

Calvert, Kenneth L., Matthew B. Doar, and Ellen W. Zegura. "Modeling internet topology." Communications Magazine, IEEE 35.6 (1997): 160-163.

Topology

- Physical connectivity
  - Star
  - Hub
  - Mesh
  - Bus
  - Tree
- Connectivity is implied
- Wireless networks have constrained
  - Topology
  - Connectivity

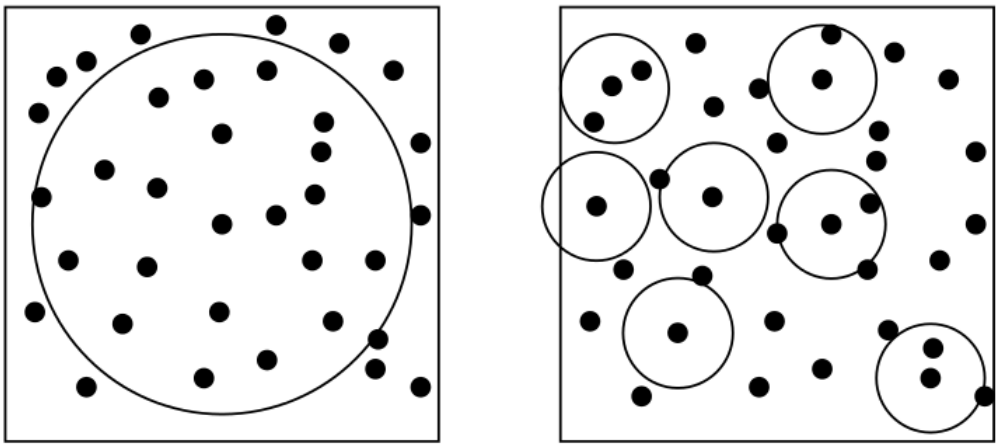
Ad hoc Networks

- 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

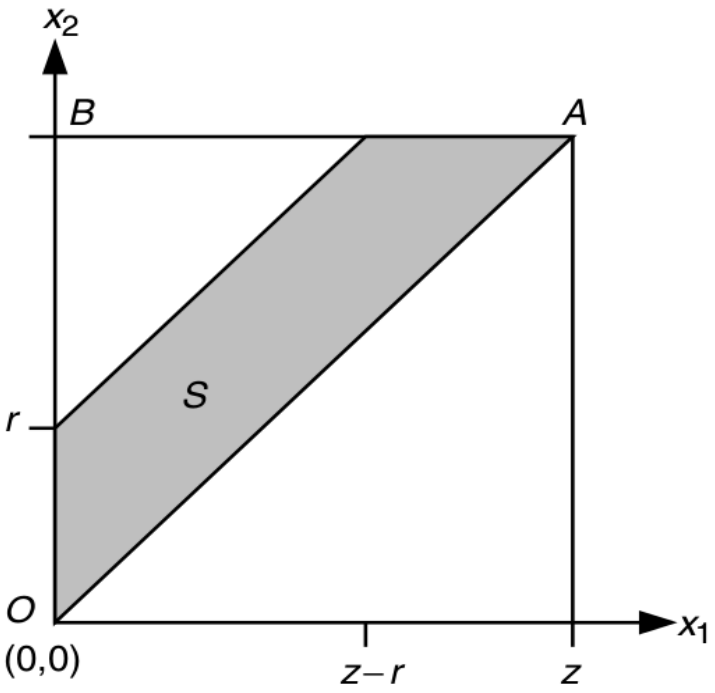


Feasibility Region

- x_1: location of the first node
- x_2: location of second node
- Nodes distributed uniformly in [0, z]

$x_1 \leq x_2$

- Two-node network connected if $x_2 - x_1 \leq r$
- Transmission range of every node: r(n) , where n is the number of nodes in network



TOPIC 205

Link Scheduling & Capacity

In this module

We shall understand

- Hidden terminal problem
- Link scheduling in wireless networks
- Network capacity

References

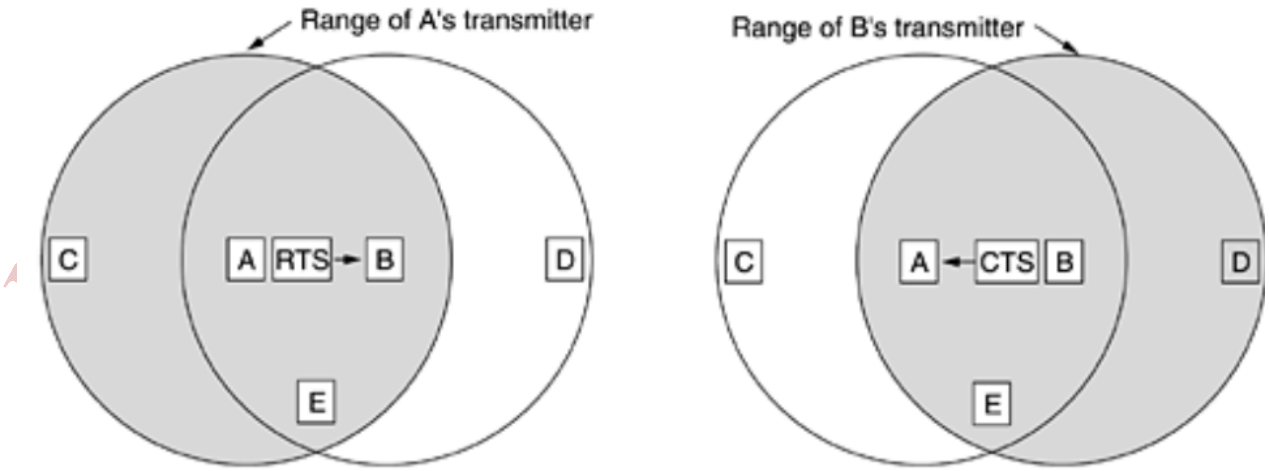
- Kumar, Anurag, D. Manjunath, and Joy Kuri. Communication networking: an analytical approach. Elsevier, 2004.
- Calvert, Kenneth L., Matthew B. Doar, and Ellen W. Zegura. "Modeling internet topology." Communications Magazine, IEEE 35.6 (1997): 160-163.

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

END

TOPIC 206

Scheduling Constraints

In this module

We shall understand

- Underlying assumptions

References

Kumar, Anurag, D. Manjunath, and Joy Kuri. Communication networking: an analytical approach. Elsevier, 2004.

Calvert, Kenneth L., Matthew B. Doar, and Ellen W. Zegura. "Modeling internet topology." Communications Magazine, IEEE 35.6 (1997): 160-163.

Underlying Assumptions (1 of 2)

- 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

Underlying Assumptions (2 of 2)

- An edge (i, j) ∈ E
- Transmission from i , addressed to j
- Decoded by j, provided that the SIR at j is adequately high

Constraints (1 of 2)

- 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

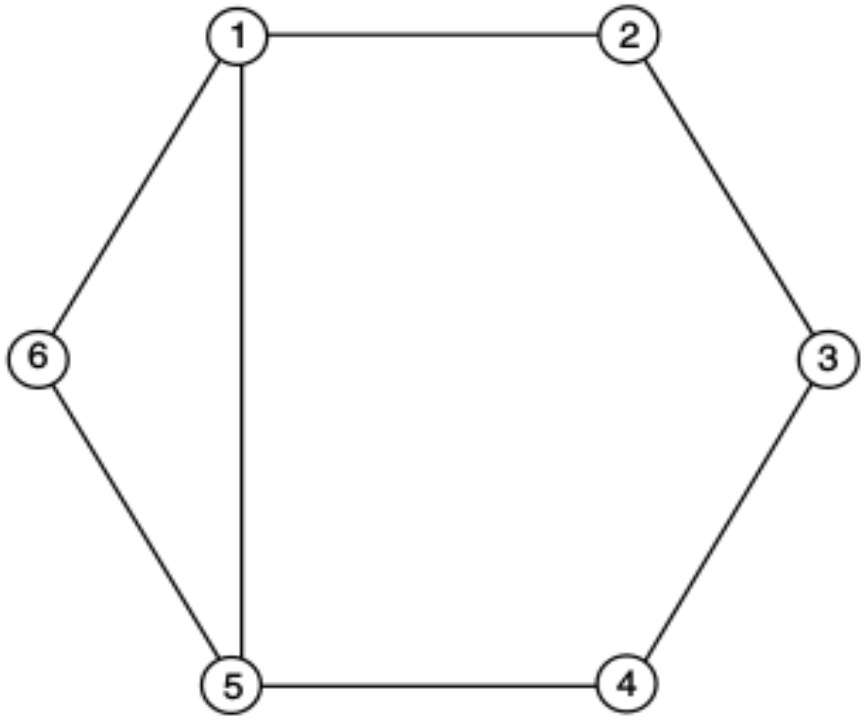
Constraints (2 of 2)

- 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)}



END

TOPIC 207

Centralized Scheduling

In this module

We shall understand

- Scheduling problem
- Schedulable region
- Bluetooth

References

- Kumar, Anurag, D. Manjunath, and Joy Kuri. Communication networking: an analytical approach. Elsevier, 2004.
- Calvert, Kenneth L., Matthew B. Doar, and Ellen W. Zegura. "Modeling internet topology." Communications Magazine, IEEE 35.6 (1997): 160-163.

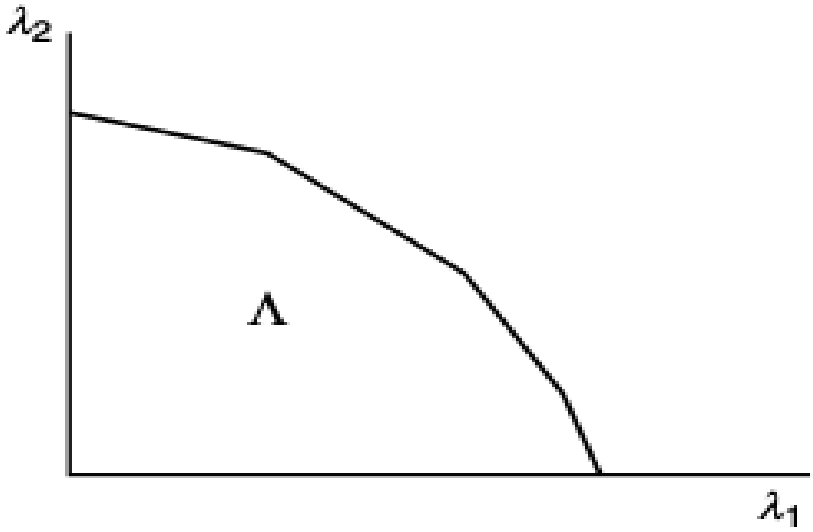
Scheduling Problem

- 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

END

TOPIC 208

Hot Backup Concepts

Hot Backup Concepts

TOPIC 209

Marginal Buffering at Every Hop

In this module

We shall understand

- Definition of marginal buffering
- Effect at every hop
- Simple analogy
- Example

References

Kumar, Anurag, D. Manjunath, and Joy Kuri. Communication networking: an analytical approach. Elsevier, 2004.

Moscibroda, Thomas, and Onur Mutlu. "A case for bufferless routing in on-chip networks." ACM SIGARCH Computer Architecture News. Vol. 37. No. 3. ACM, 2009

Definition

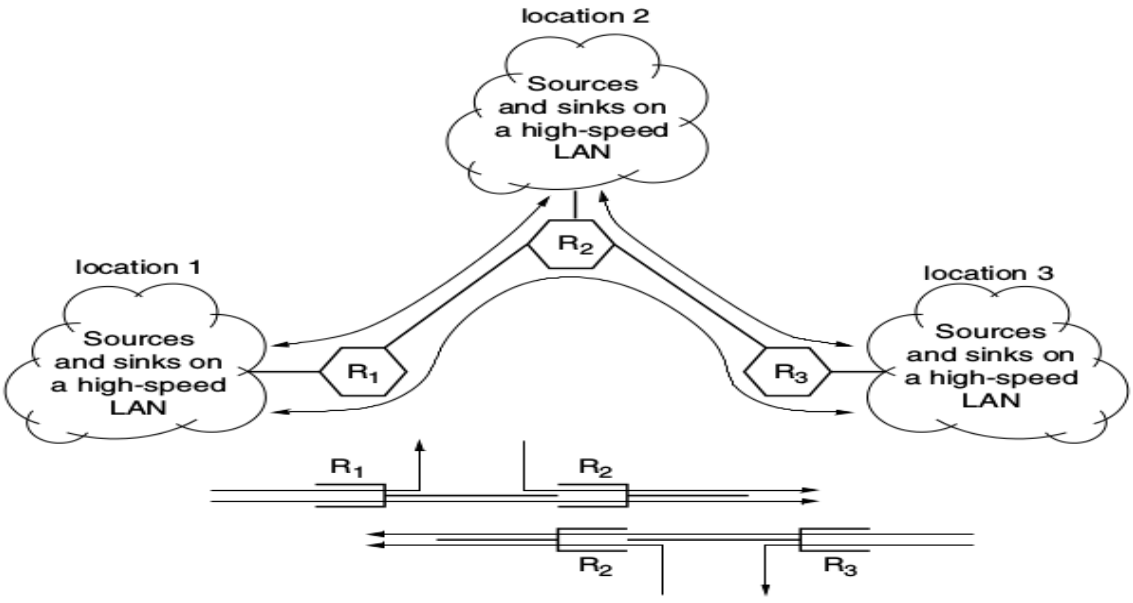
- Multiplexer has no buffer to store data arriving in a slot but cannot be served in that slot
- Performance depends only on marginal distribution of arrival process
- Doesn't depend on correlations b/w arrivals in slots

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

END

TOPIC 210

Problem Set 1

In this module

We shall understand and recall

- BER on channel performance
- Ethernet Framing
- Backoff Algorithm

Reference

Tanenbaum, Andrew S. "Computer networks, 4-th edition." ed: Prentice Hall (2003).

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?

END

TOPIC 211

Problem Set 1

In this module
We shall understand and recall
- Operation of MAC Addressing
- Performance of ALOHA
- Switch learn-ability

Reference
Kurose, James F. Computer Networking: A Top-Down Approach Featuring the Internet, 3/E. Pearson Education India, 2005.

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 (1 of 2)
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?

Performance of ALOHA (2 of 2)
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.

END

TOPIC 212

Simulate Parity Scheme Failure

In this module
We shall understand
- Injecting channel errors (channel)
- Receiver behaviour

- Failing parity scheme

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

END

TOPIC 213

Simulate ARP Behaviour

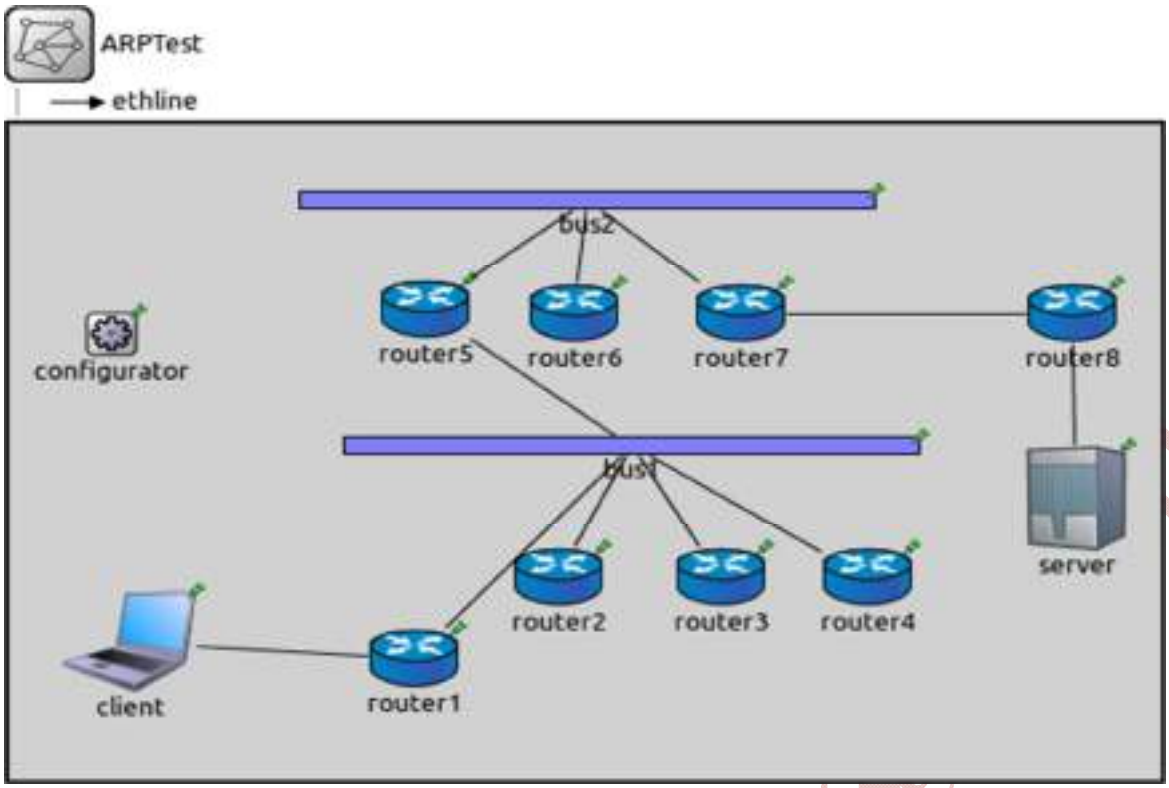
In this module

We shall understand

- How OMNET handles ARP? (recall!)
- Variants
- Performance

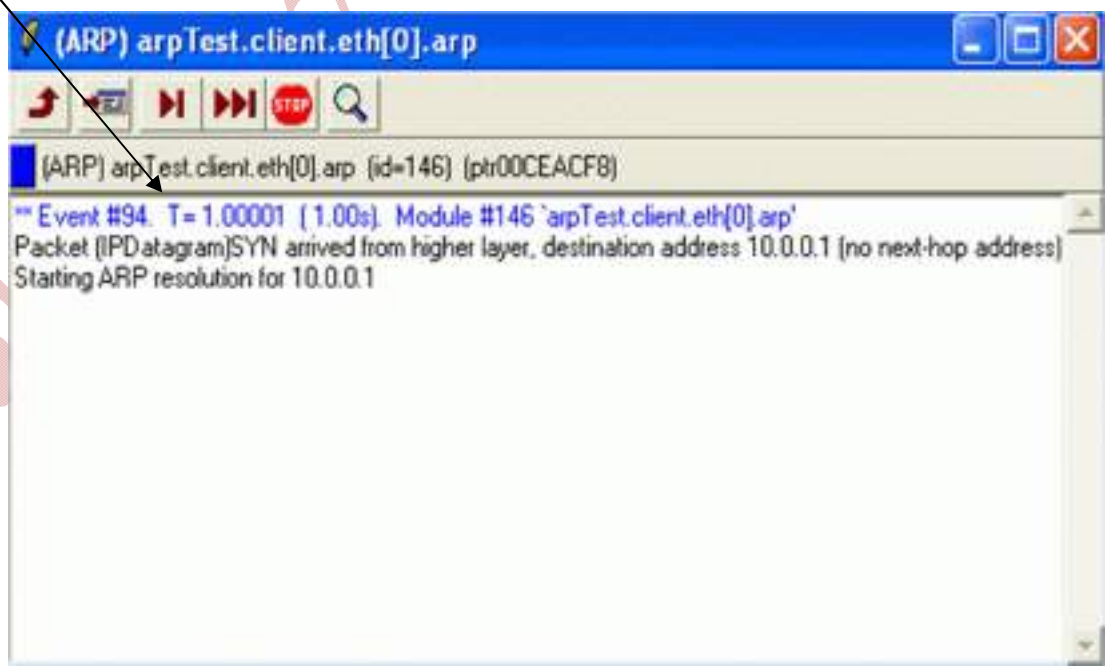
Scenario

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



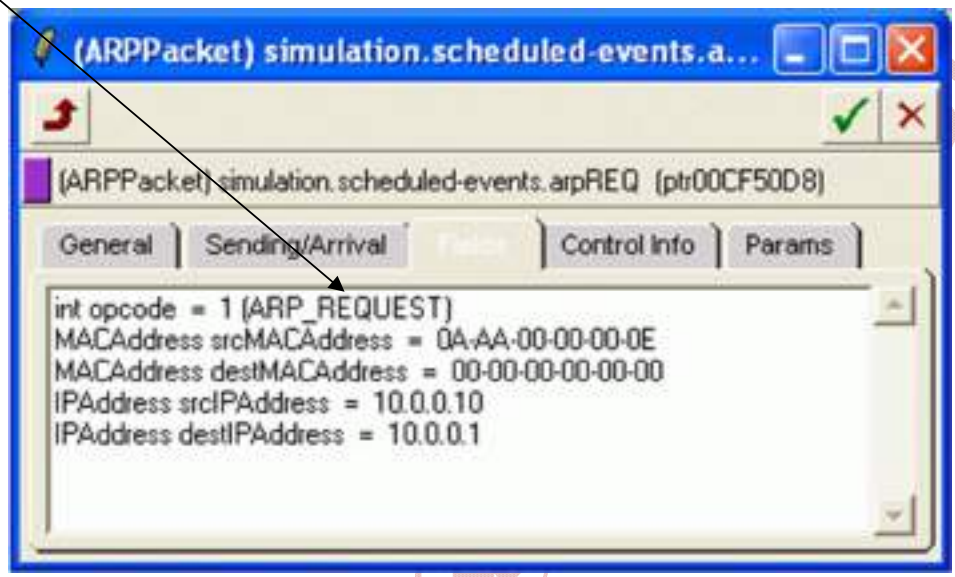
arpTest.client.eth[0].arp

module output



Inside ARP Packet

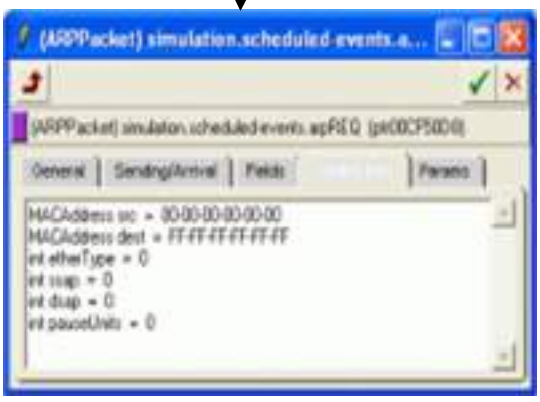
ARP Broadcast Message



ARP Packet Class (Generated by .msg file)

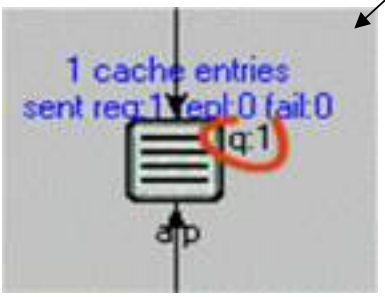
```
// file: ARPPacket.msg
message ARPPacket
{
fields:
 int opcode enum(ARPOpcode);
 MACAddress srcMACAddress;
 MACAddress destMACAddress;
 IPAddress srcIPAddress;
 IPAddress destIPAddress;
 };
```

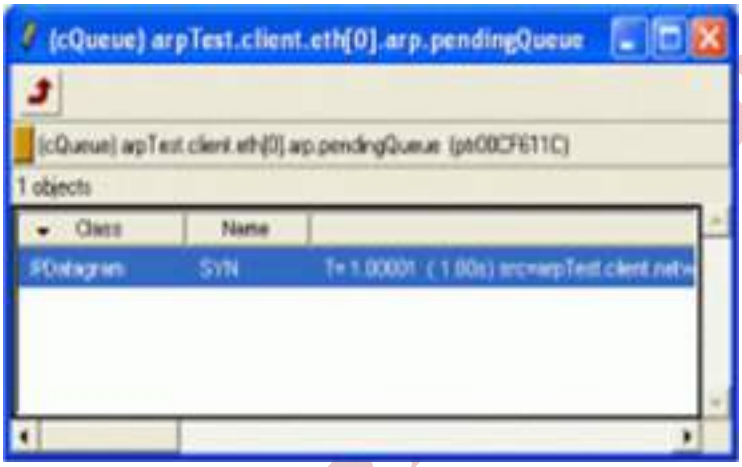
This packet is appended with broadcast address in control info (a small data structure)



Packet Queue (Contains IP Packet)

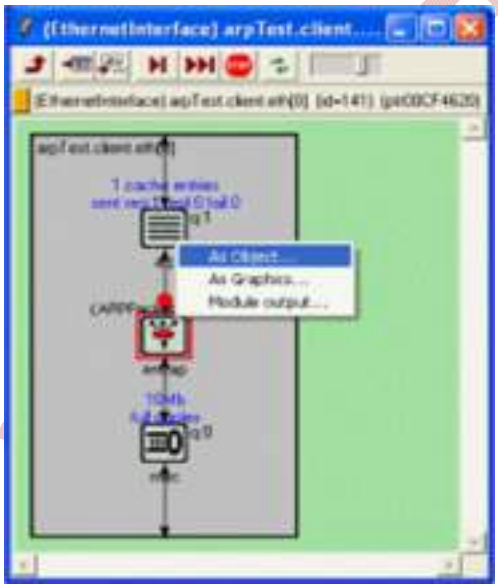
This packet is enqueued till ARP resolution

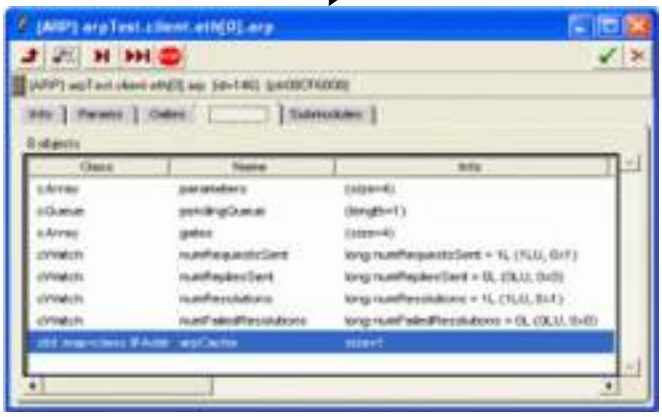




ARP Cache Build-up

Contents of ARP Cache (entries with soft timers)





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

END

TOPIC 214

Output Analysis on WireShark

In this module

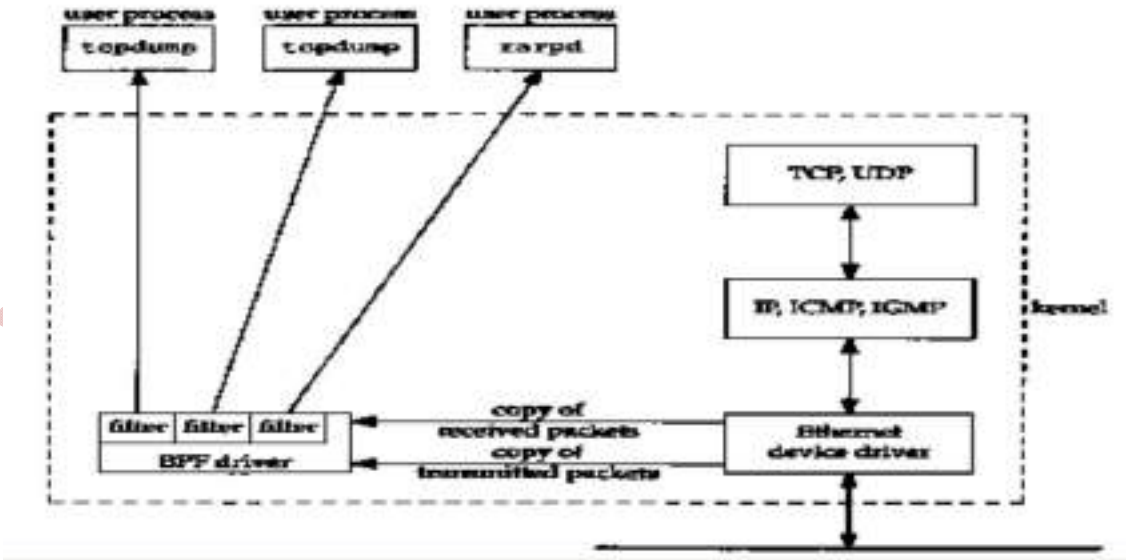
We shall understand

- Introduction to wireshark
- Packet capture process
- Example

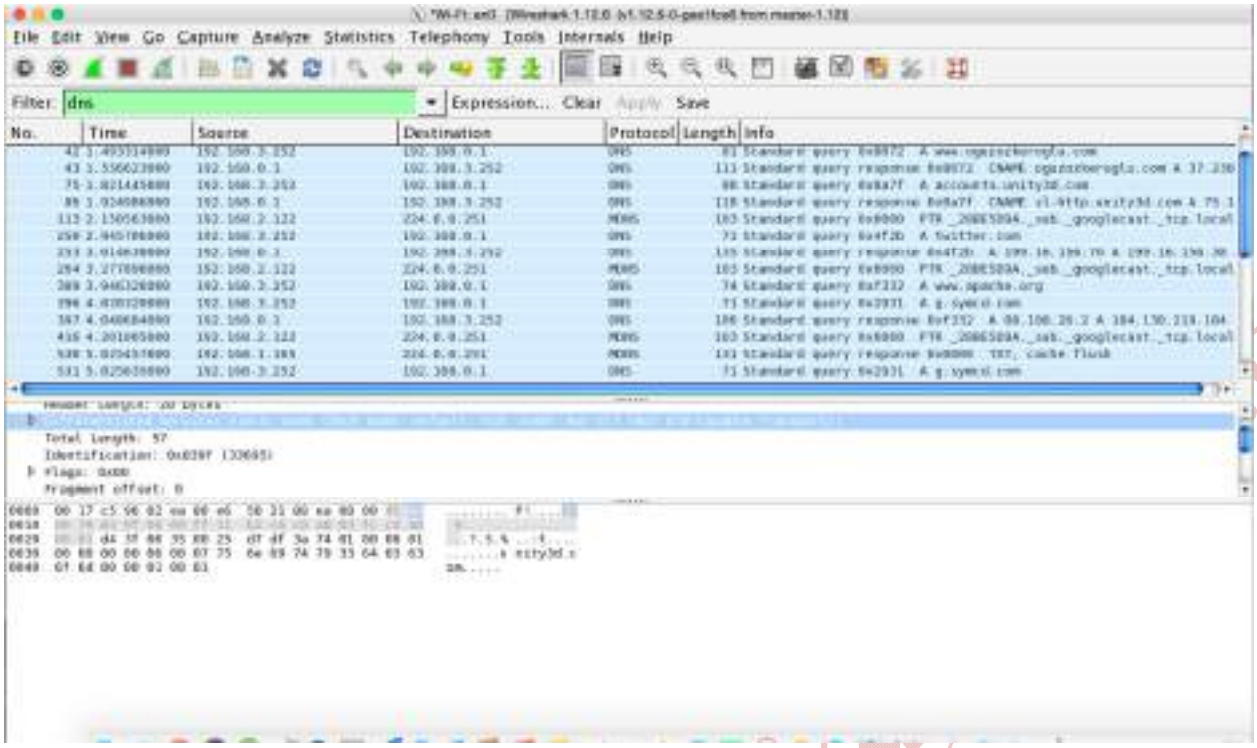
Wireshark

- 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

END

TOPIC 215

Simulate Switching vs Routing

In this module

We shall understand

- Need for comparison
- Router vs switch
- Basis of comparison

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

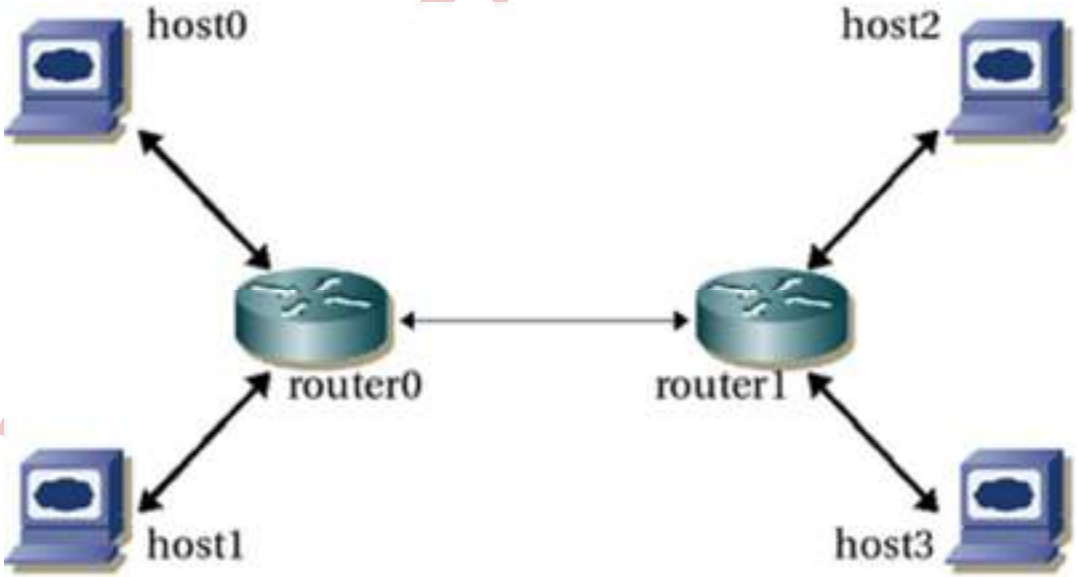
Basis of Comparison

- 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



END

WEEK 16

TOPIC 216

Overview of Access Technologies

In this module

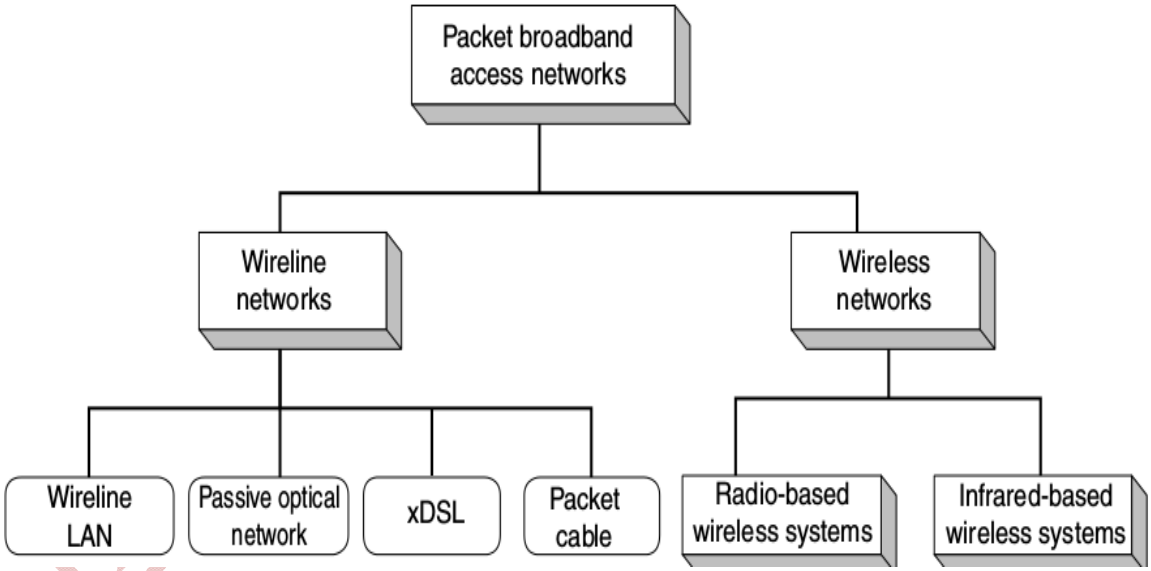
We shall understand

- What are broadband access technologies?
- Overall taxonomy
- Wireless technologies

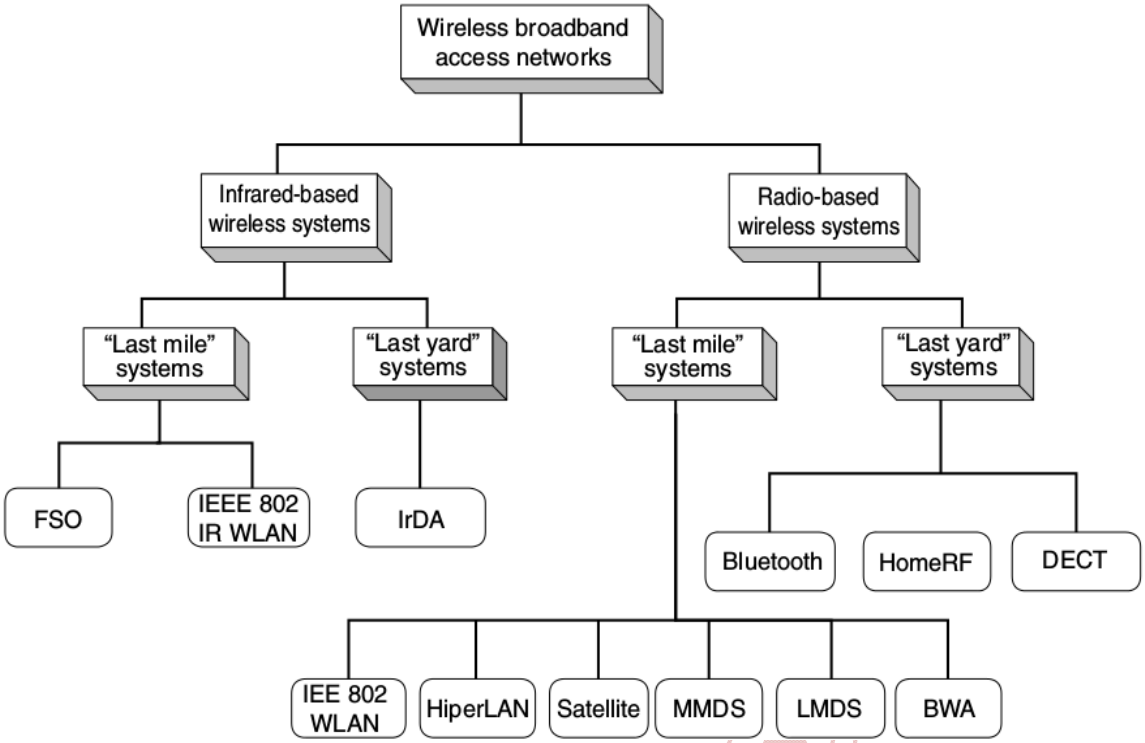
Broadband Access

- 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



END

TOPIC 217

WiFi

In this module

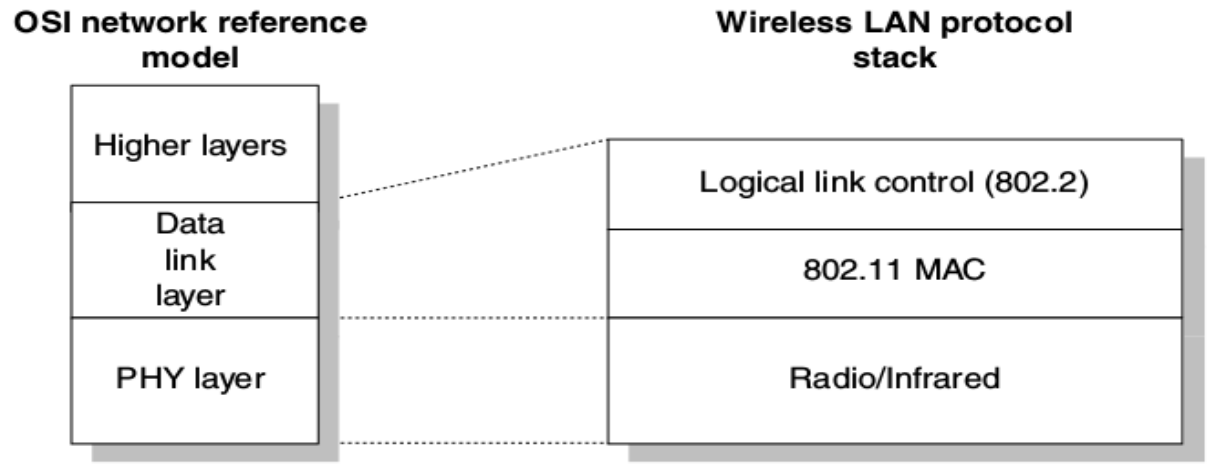
We shall understand

- Introduction to WLANs
- Protocol stack
- Hidden terminal problem
- RTS CTS Mechanism

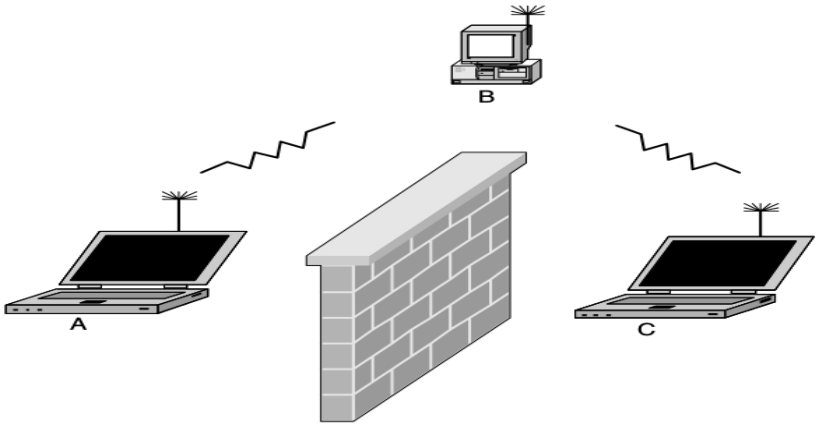
Wireless Fidelity Network

- Wireless LAN offers mobility and increased flexibility
  - Portable computers coming to a meeting forms a wireless LAN
  - More economical
- 802.11b uses a HF band
  - Up to 11 Mbps

WLAN Protocol Stack



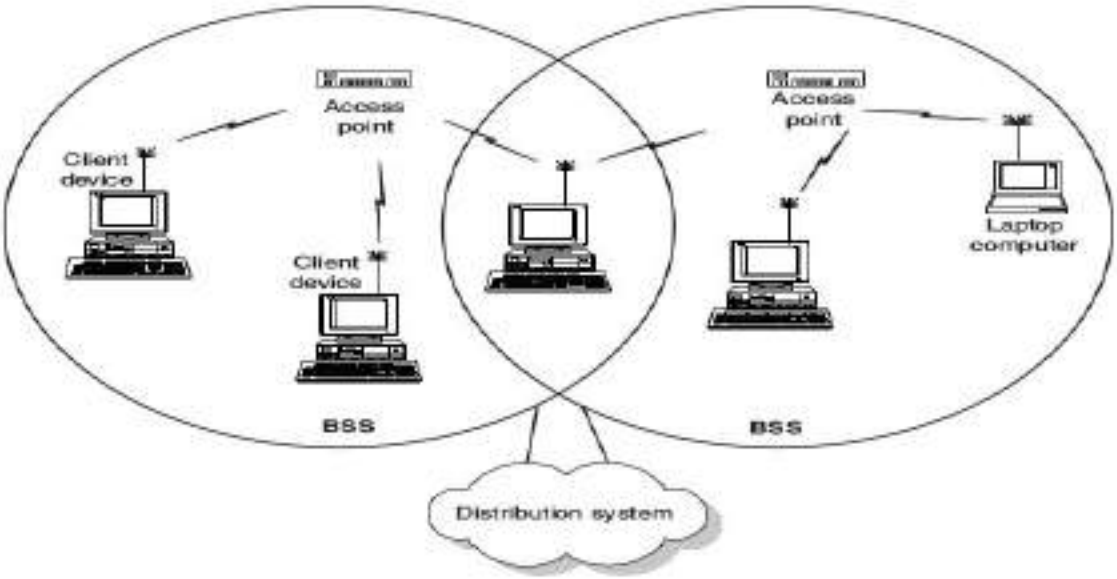
The Hidden (Exposed) Station Problem



RTS CTS Mechanism

- Sender sends request to send
- Receiver acknowledges as clear
  - Overhearing neighborhood cautioned

WLAN Configuration



TOPIC 218

WiFi Operations

In this module

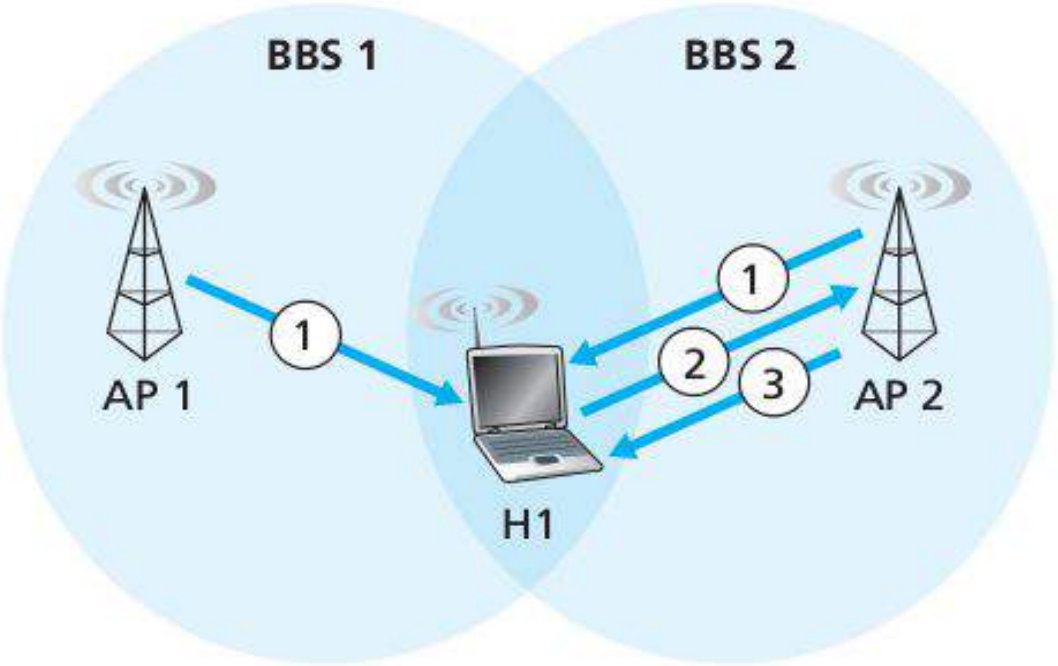
We shall understand

- Brief on operations
- Scanning for APs
- Mobility

Operations

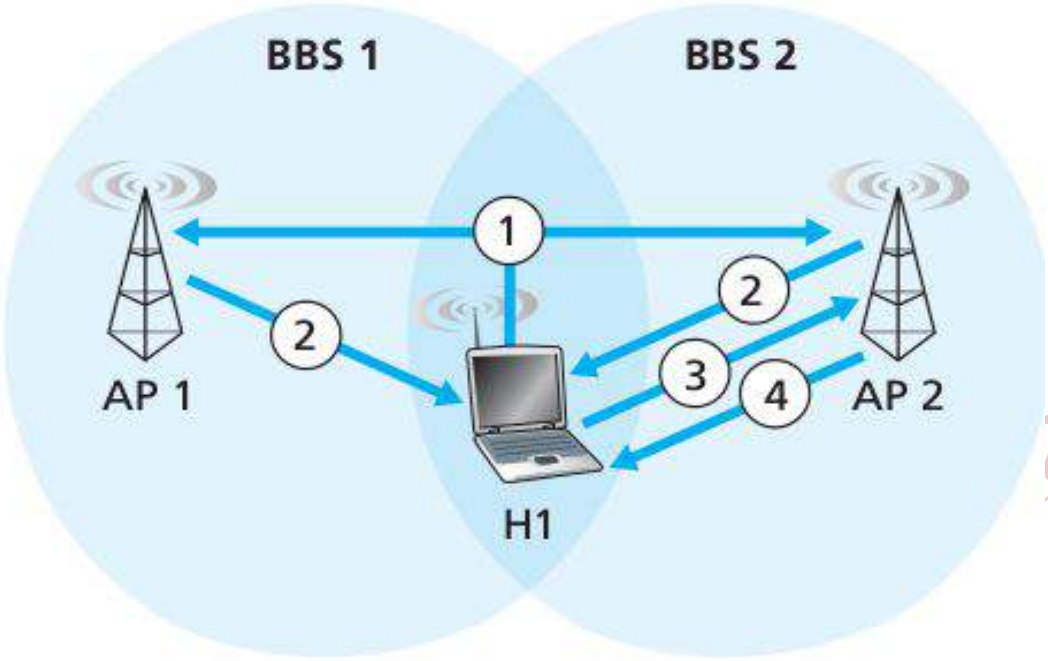
- Synchronization
- Authentication
- Association
- Data Transmission
- Handoff
- Power management

Scanning for APs [Passive scanning]



a. Passive scanning
1. Beacon frames sent from APs
2. Association Request frame sent: H1 to selected AP
3. Association Response frame sent: Selected AP to H1

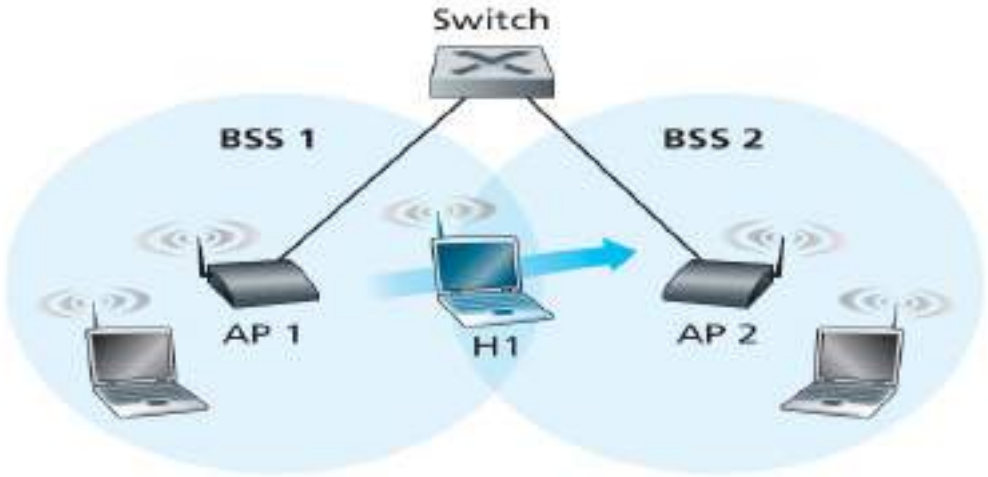
Scanning for APs [Active scanning]



a. Active scanning
1. Probe Request frame broadcast from H1
2. Probes Response frame sent from APs
3. Association Request frame sent: H1 to selected AP
4. Association Response frame sent: Selected AP to H1

Mobility in the Same IP Subnet

- H1 moves from BSS1 to BSS2
- Keeps its IP address
  - — And all of its ongoing TCP connections



END

TOPIC 219

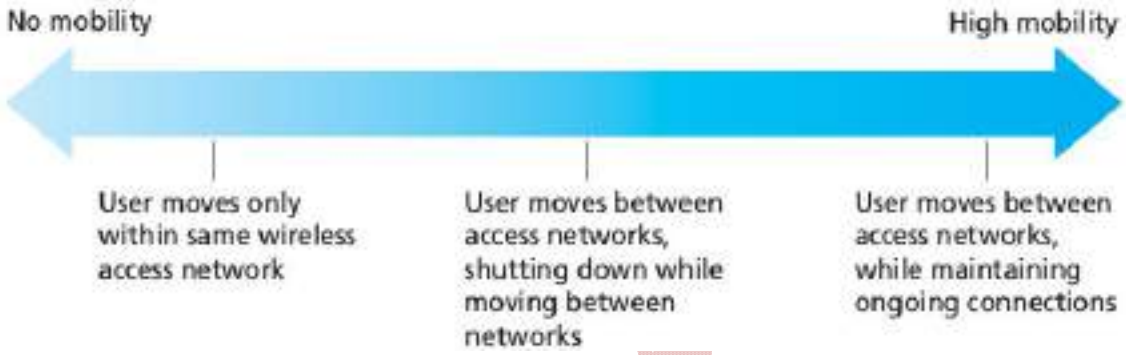
Mobile IP

In this module

We shall understand

- Degrees of mobility
- Mobile IP standard
- Elements
- Procedures

Degrees of Mobility

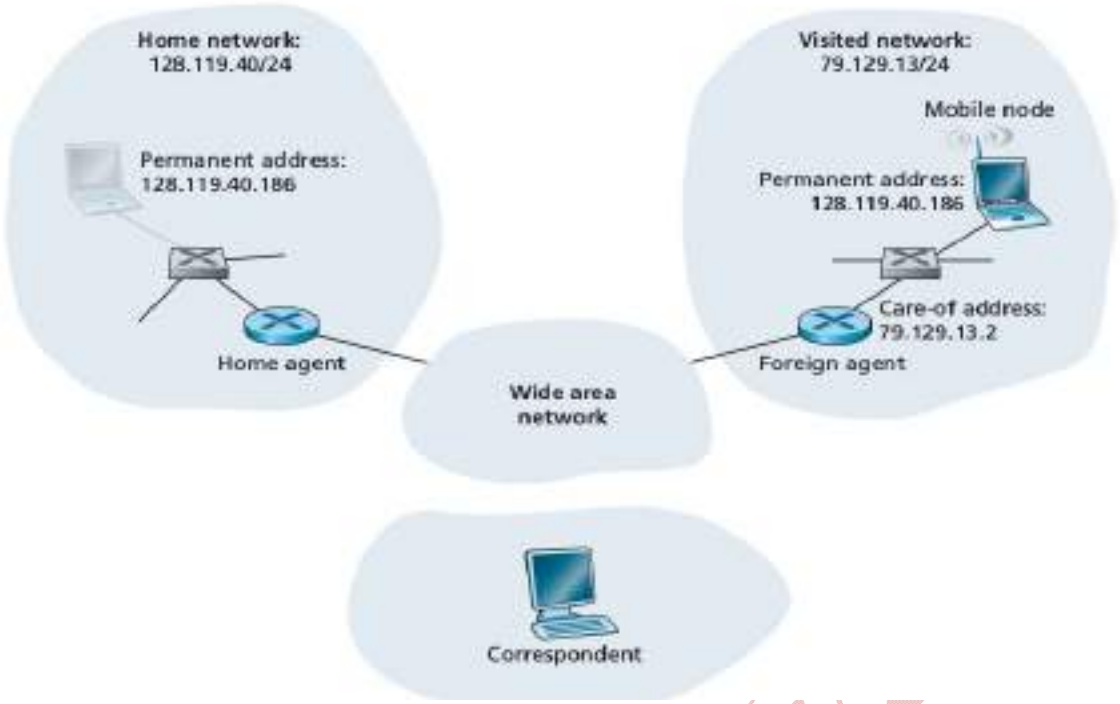


Mobile IP Standard

- RFC 3344
- Elements
  - — Home agents,
  - — Foreign agents,
- Foreign-agent registration
- Care-of-addresses
- Encapsulation (packet-within-a-packet)

Elements of Mobile IP System

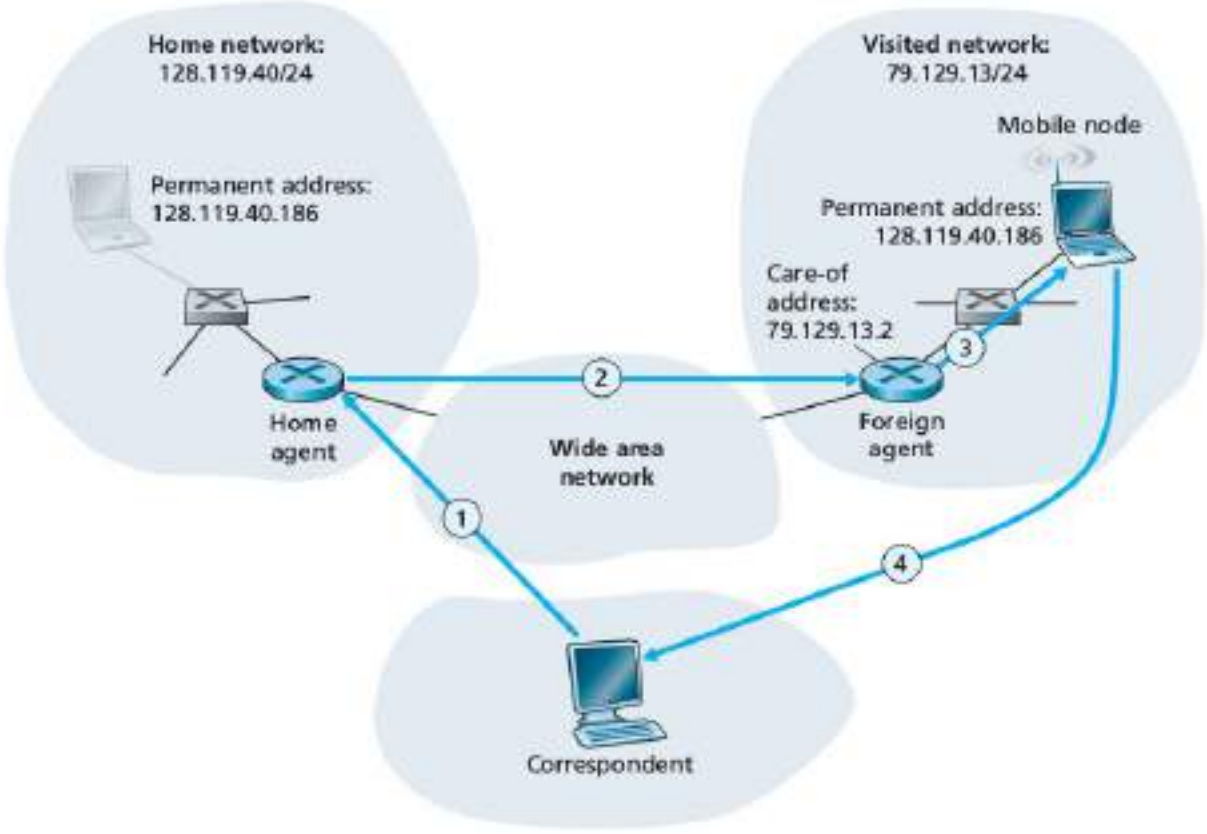
CS432 Handouts Made by Mahjabeen
mahjabeen97869@gmail.com
contact # 0321 2711298



Procedures

- Agent discovery
- Registration with home agent
- Indirect routing of datagrams

Indirect Routing



END

TOPIC 220

Packet Cable Networks

In this module

We shall understand

- Background
- Components
- Headend
- CMTS
- Cable Modem

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

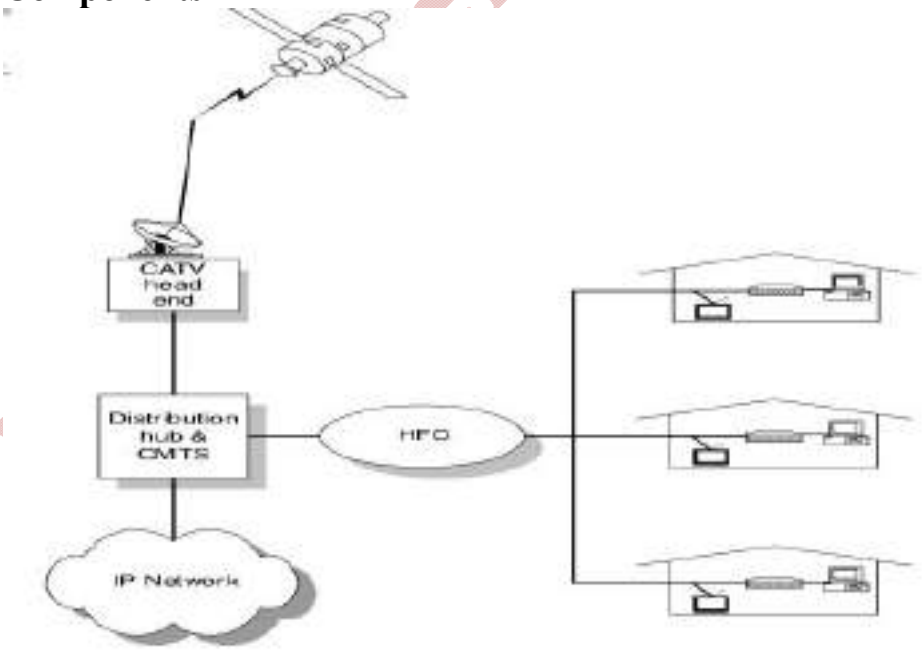
Architecture

- 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 (1 of 2)

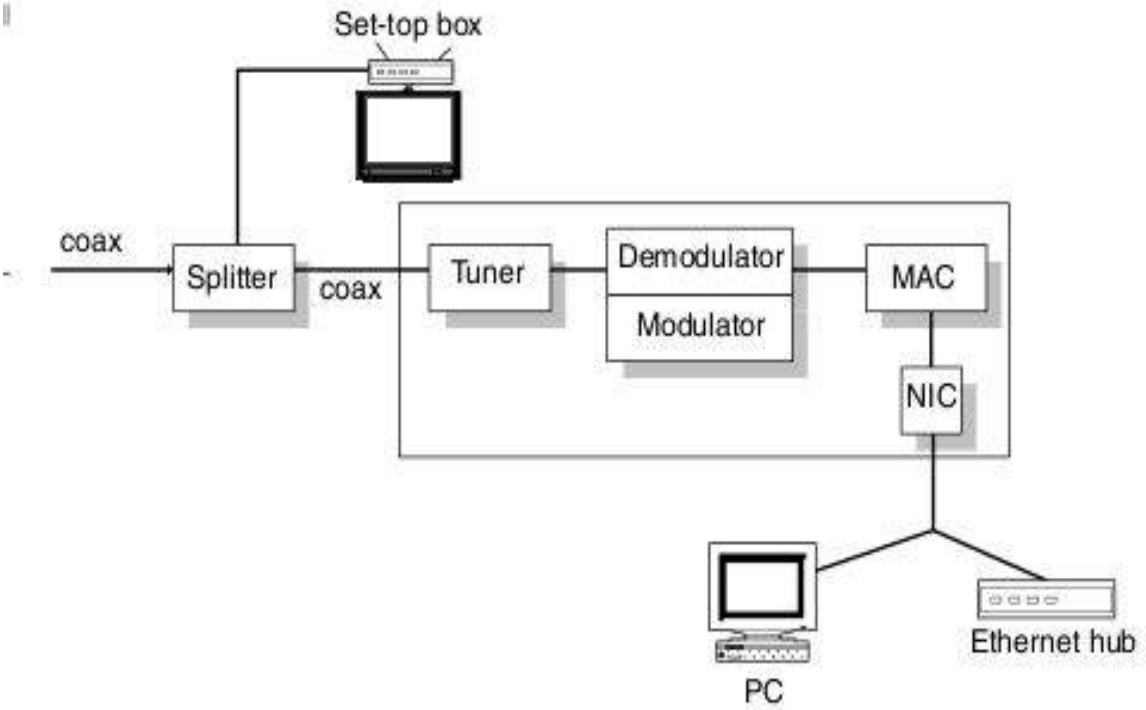
- Controlling bandwidth allocation for data traffic to each modem
- Enforcing bandwidth allocation policy
- Assigning a time slot to each cable modem for transmitting upstream messages
- Enforcing QoS policies such as traffic shaping and policing (packet classification based on QoS classes)

Cable Modem Network Configuration

- Cable Model Systems accommodates two way communication

DOCSIS (data over cable service interface specification)

CMTS cable modem termination system is a device that allow multiple cable modem to connect to the internet through a single connection



END

TOPIC 221

WiMax

In this module

We shall understand

- Background
- Introduction
- Architecture

- Components

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

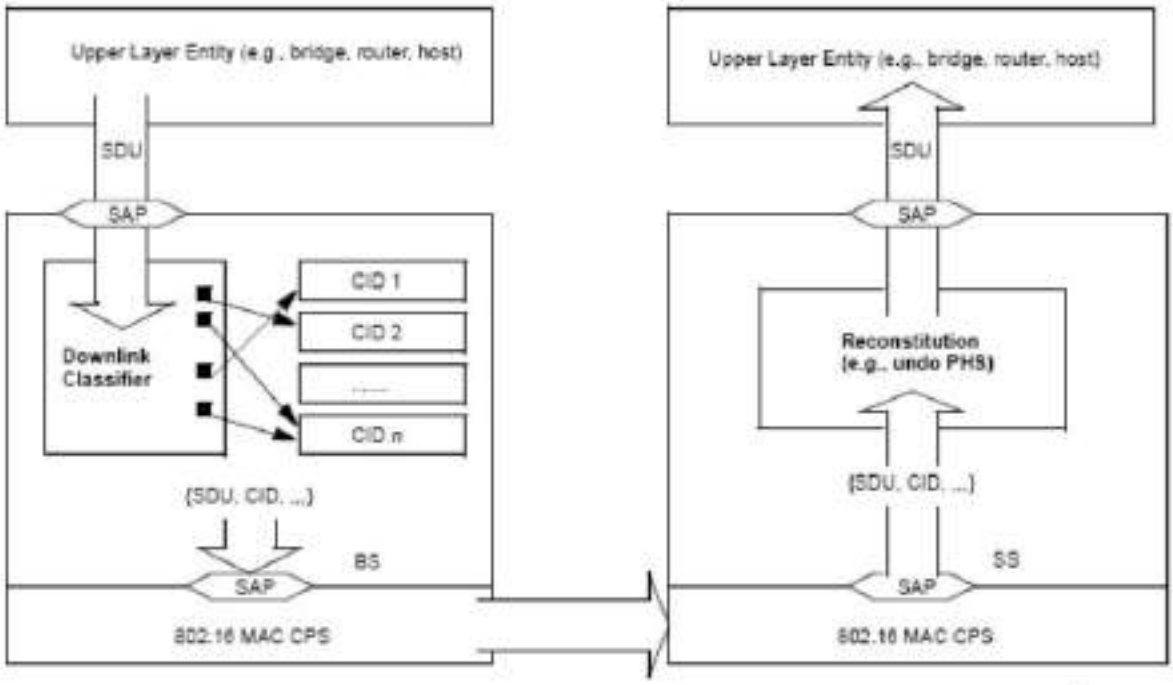
- Worldwide Interoperability for Microwave Access (WiMAX)
- Many basic ideas of 802.16 borrowed from DOCSIS/HFC applied to the wireless setting
- Good analogy : Wi-Fi : Ethernet :: WiMAX : DOCSIS/HFC

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



END

TOPIC 222

Digital Subscriber Line

In this module

We shall understand

- Background
- Introduction
- Architecture
- DSL Family

Background

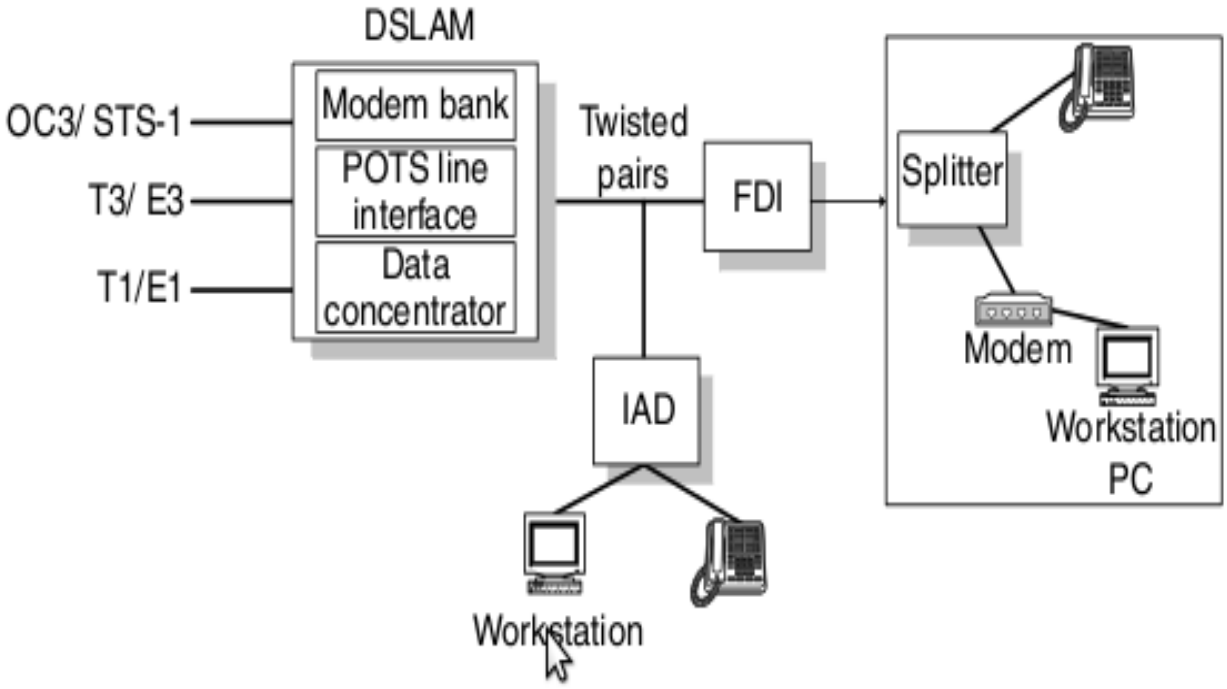
- A family of technologies for broadband last-mile solution using existing copper wires

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 (1 of 2)

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
GLite	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; 204 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); 204 Mbps (Europe) on a single duplex line	12,000 ft on 24-gauge wire	T1/E1 service replacement

DSL Family (2 of 2)

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	129 to 528 Mbps downstream; 1.6 Mbps to 2.3 Mbps upstream	4500 ft at 1296 Mbps	Connections to fiber-based networks

END

TOPIC 223

Wireless Personal Area Networks
In this module
We shall understand

- Introduction
- Comparison
- ZigBee vs Bluetooth

Introduction to LR-WPANs (1 of 2)

- Low-rate low-power wireless personal area networks
  - Types of wireless sensor networks

- Applications
  - Industrial control & monitoring
  - Environmental & health monitoring

Introduction to LR-WPANs (2 of 2)

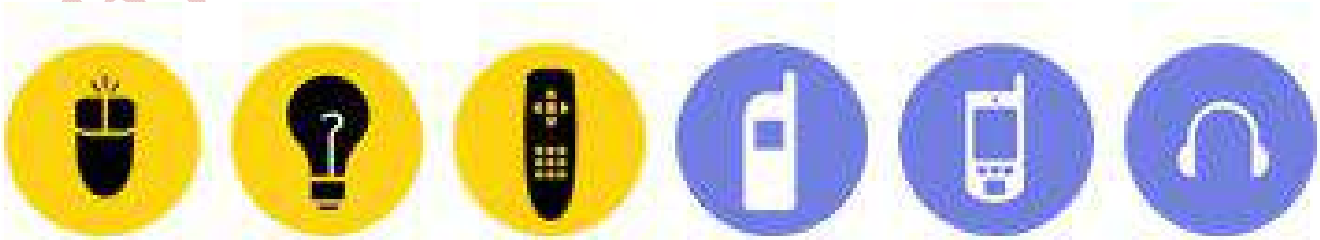
- Home automation, entertainment & toys
- Security, location and asset tracking
- Emergency and disaster response

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 (1 of 2)

- Smaller packets over large network
- Mostly Static networks with many, infrequently used devices
- Larger packets over small network
- Ad-hoc networks



ZigBee vs Bluetooth (2 of 2)

	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 meters

END

TOPIC 224

IEEE802.15.4

In this module

We shall understand

- Features
- Topology Models

Features (1 of 2)

- 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

Features (2 of 2)

- Allocation of guaranteed time slots (GTSs)
- CSMA-CA channel access
- Fully acknowledged data transfer

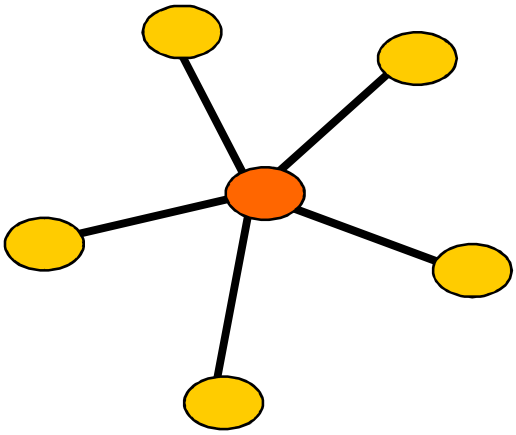
- Low power consumption
- Energy detection (ED)
- Link quality indication (LQI)

Topology Models (1 of 3)

ZigBee coordinator

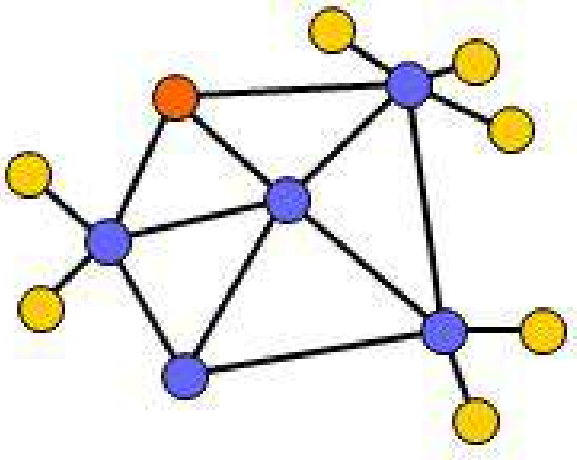
ZigBee Routers

ZigBee End Devices



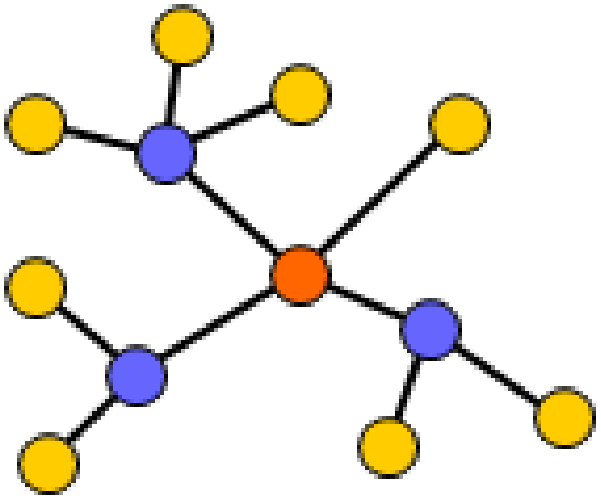
Star

Topology Models (2 of 3)



Mesh

Topology Models (3 of 3)



Cluster Tree

END

TOPIC 225

Radio Frequency Identification

In this module

We shall understand

- Introduction
- Application Areas
- Architecture
- Traffic flow

Introduction

- Presence known if within a certain radius
  — Object identified
- Do not know exactly the position

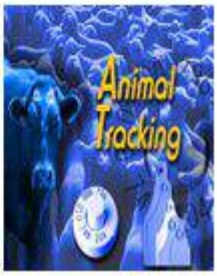
Application areas (1 of 2)

Airline baggage ID

Ticketing

Sports Timing

Document Tracking



Application areas (2 of 2)

Product Authentication

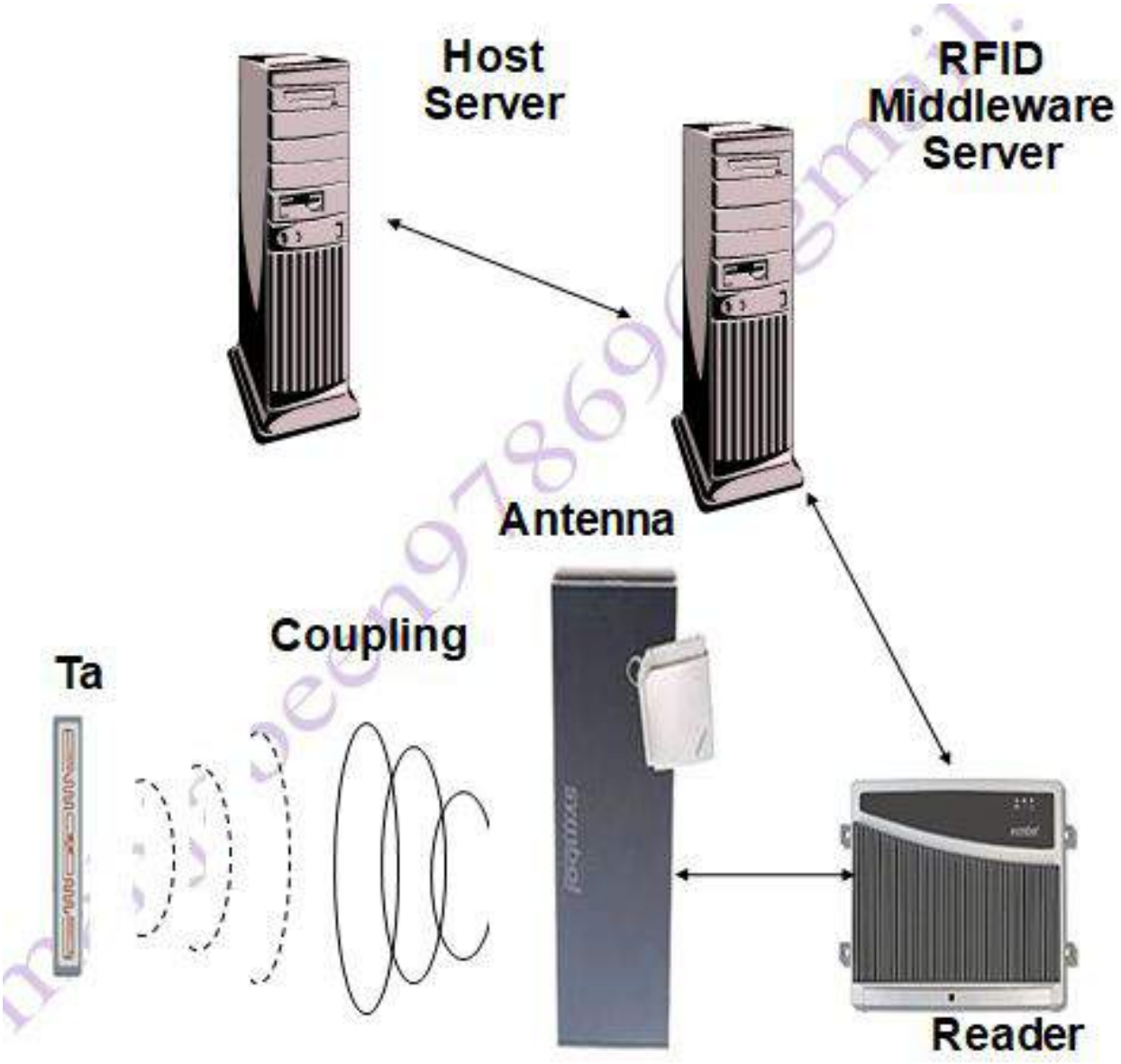
Highway Toll Collection

Keyless Remote

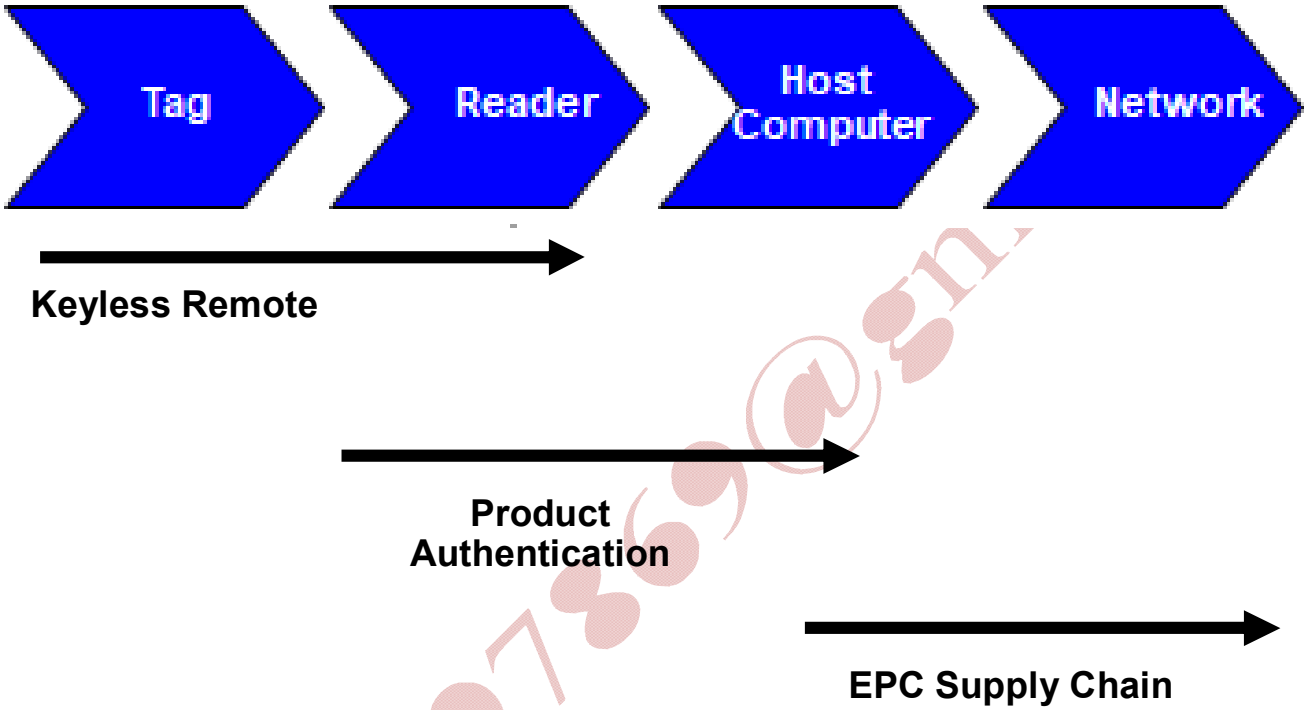
Supply

Smart Cards

Architecture



Traffic Flow



END

THE END

Remember me and my family in your prayers .
CS432 Handouts Made by Mahjabeen
mahjabeen97869@gmail.com
contact # 0321 2711298