بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

==========

CS432 Handouts

Network Modeling and Simulation

MIDTERM

Week (1 to 8)
Topic (1 to 112)

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

==========

Week 1

TOPIC 01

Network Modeling and Simulation

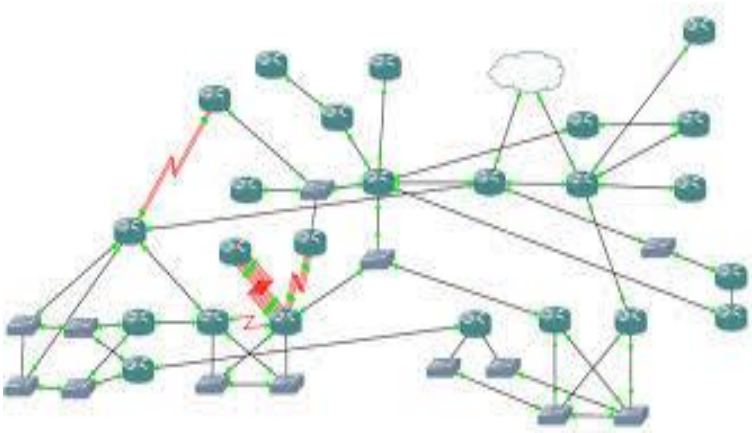
Course code CS432
Credits 3+0
Instructor
Dr. Ali Hammad Akbar
Lecturing style
Video lectures of short duration
(5-7 minutes)

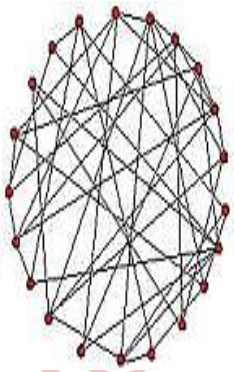
Evaluation

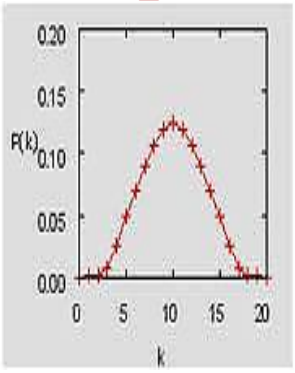
- Quizzes
- Assignment
- Simulation modules
- Mid-term and final

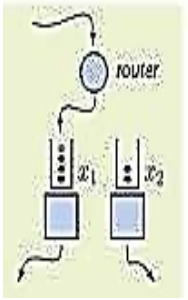
Enter the complex world of networks

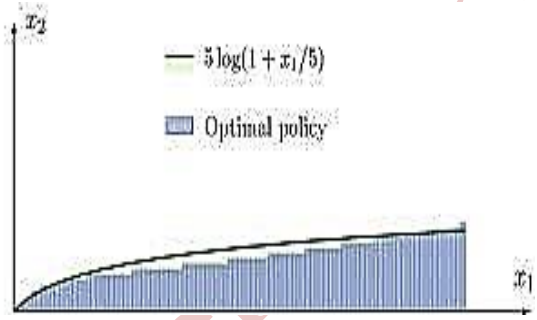
- Can we simplify it?
- Or at least the discussion of it?





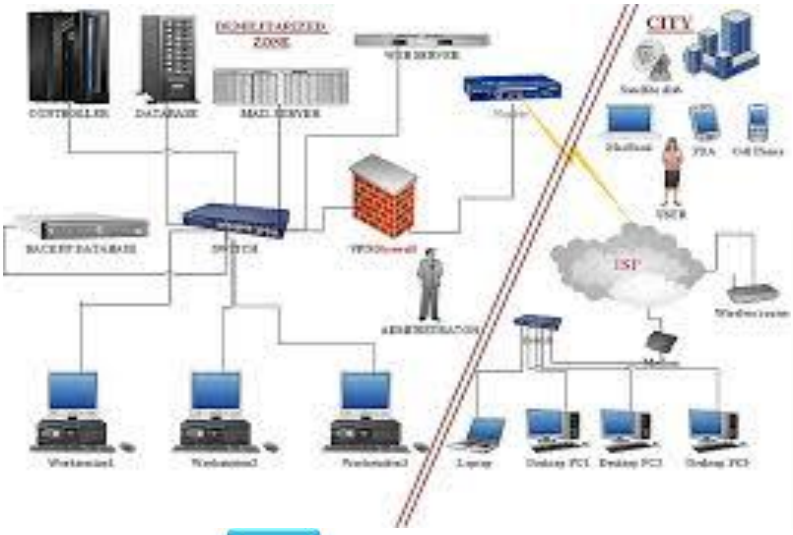




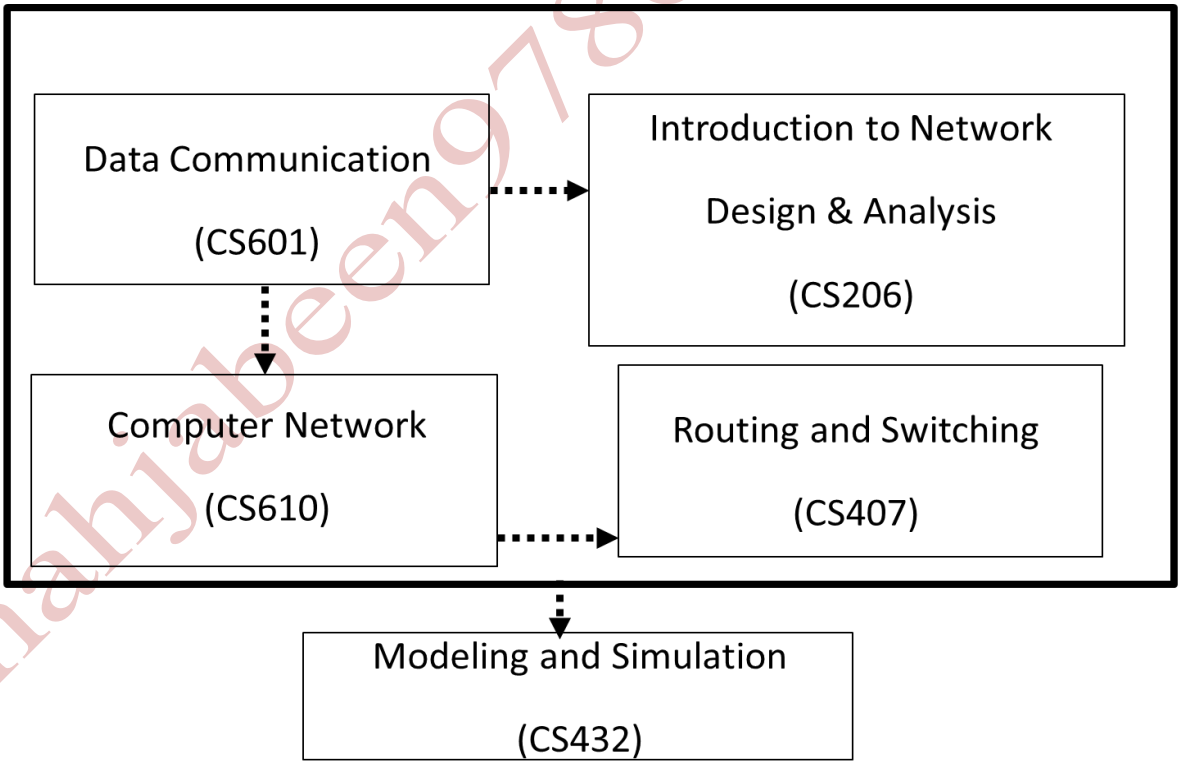


Knowing me

- 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

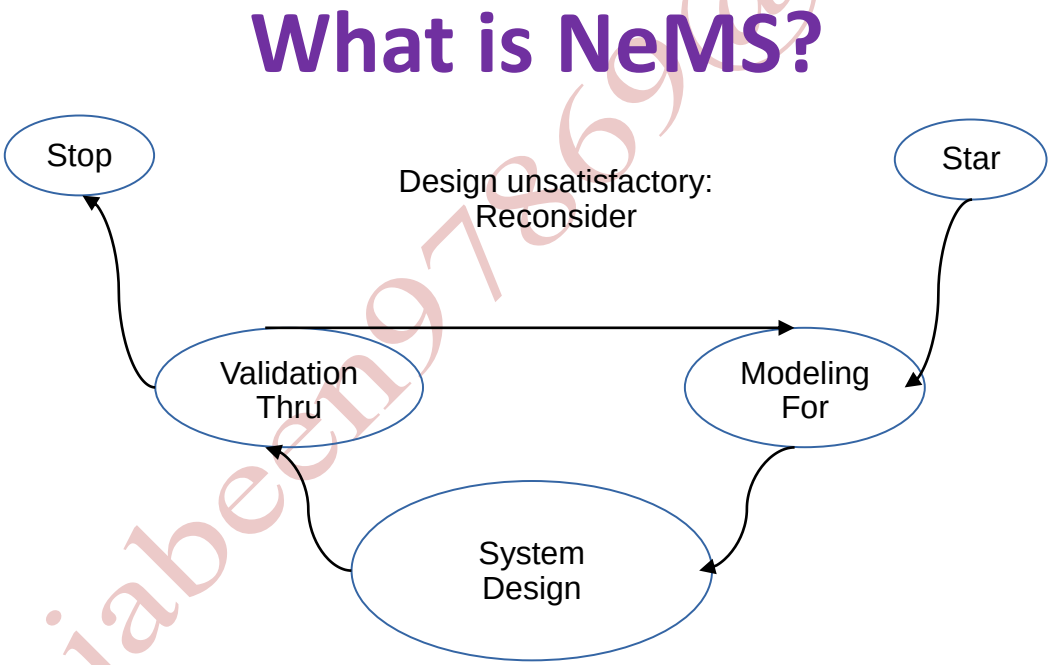
What is NeMS?

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.

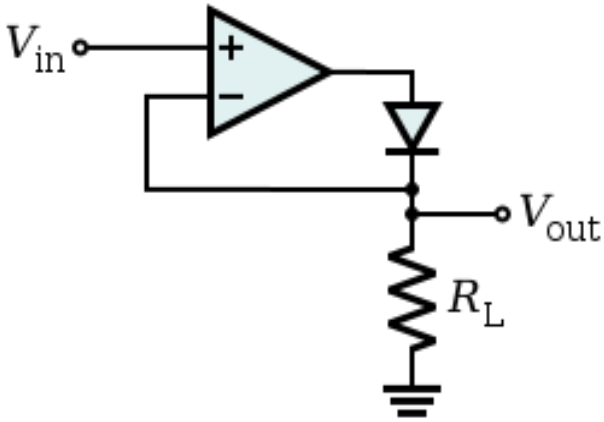
Simulation is

- 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



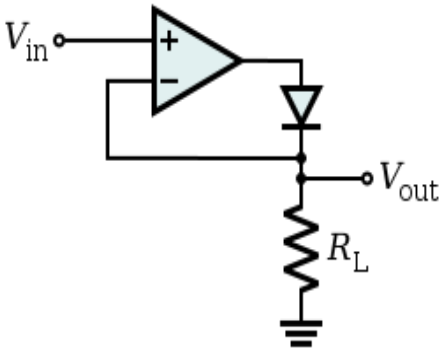
- Consider a simple signal detection circuit

of users with DBMS

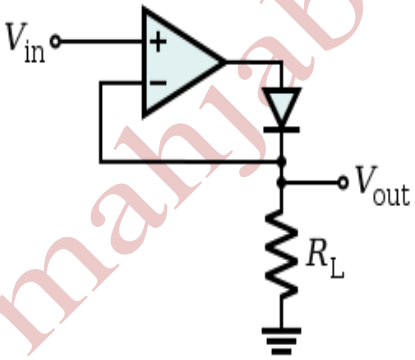


- Its **simulation** could give out
  - Correct result
  - Incorrect result
  - No result

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

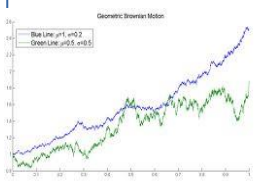
Open or Closed

- Open model define and require external inputs

- A model that does not take external inputs is closed
  - Automated systems

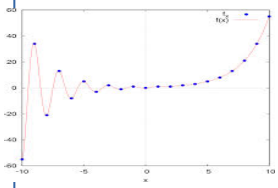
What is Model?

Stochastic
Deterministi



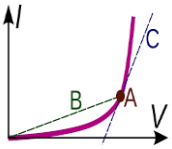
A

Continuous
Discrete



B

Linear
Nonlinear



C

Supra-modeling

A+B'+C
A'+C'

End

TOPIC 05

Modeling Perspective & Intra-Model Relation

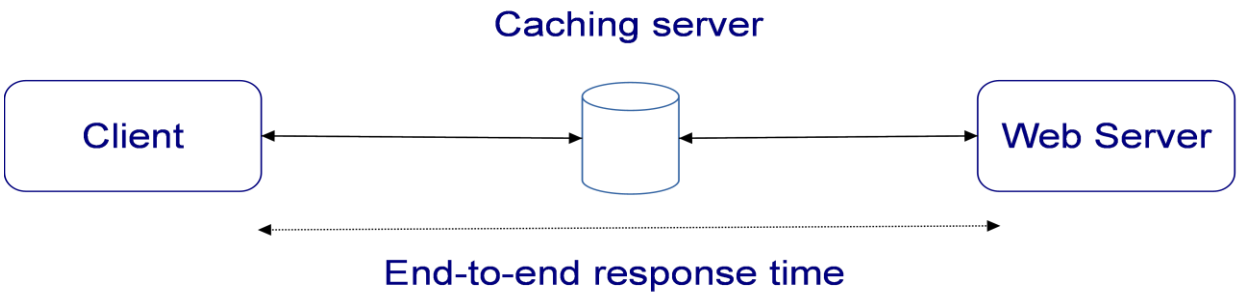
In this module
We shall learn
- What needs to be modeled?
- Which is the most suitable model?
- How to do most appropriate modeling?

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

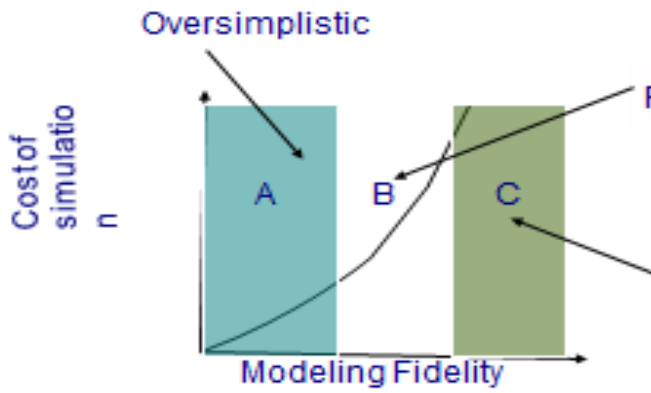
Model what you need & no more

- 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

Modeling Perspective & Intra-Model Relation



End

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

TOPIC 06

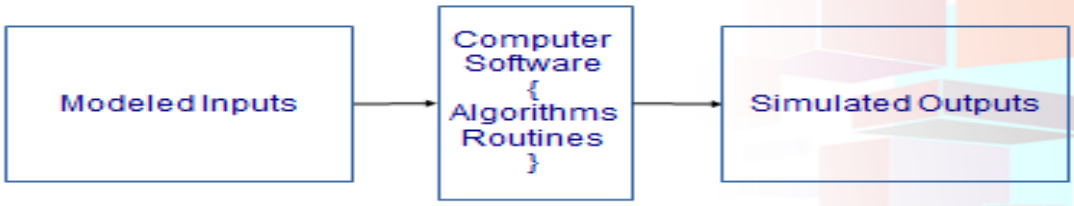
What is Simulation?

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

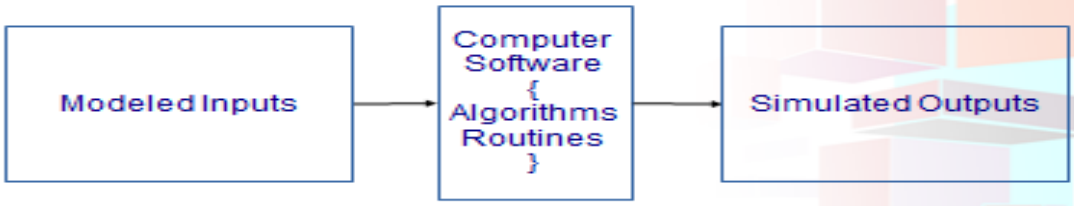
- A simple use-case
- What is its simulation?
- What are modeling and simulation terminologies

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

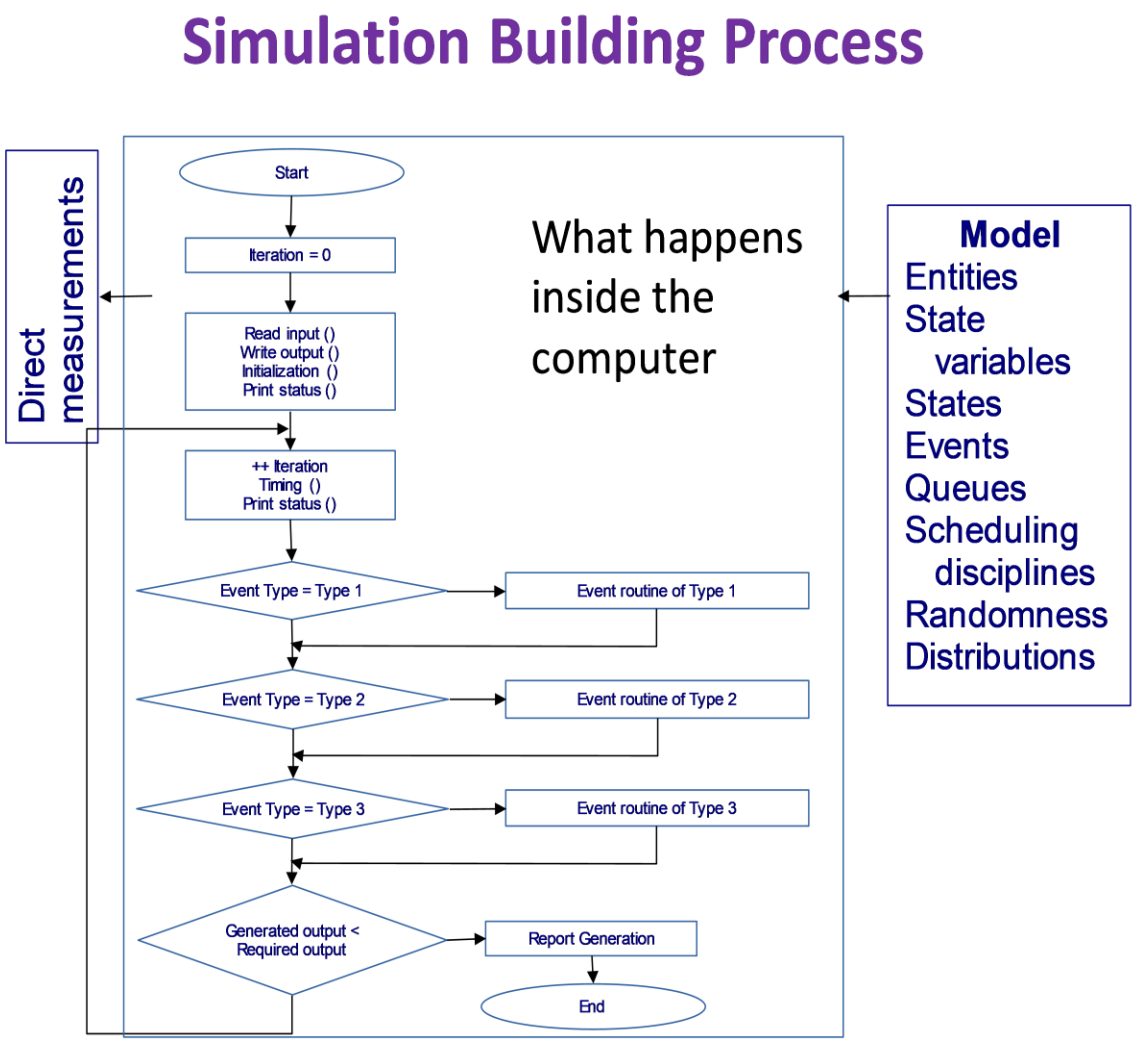
computers

- 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 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

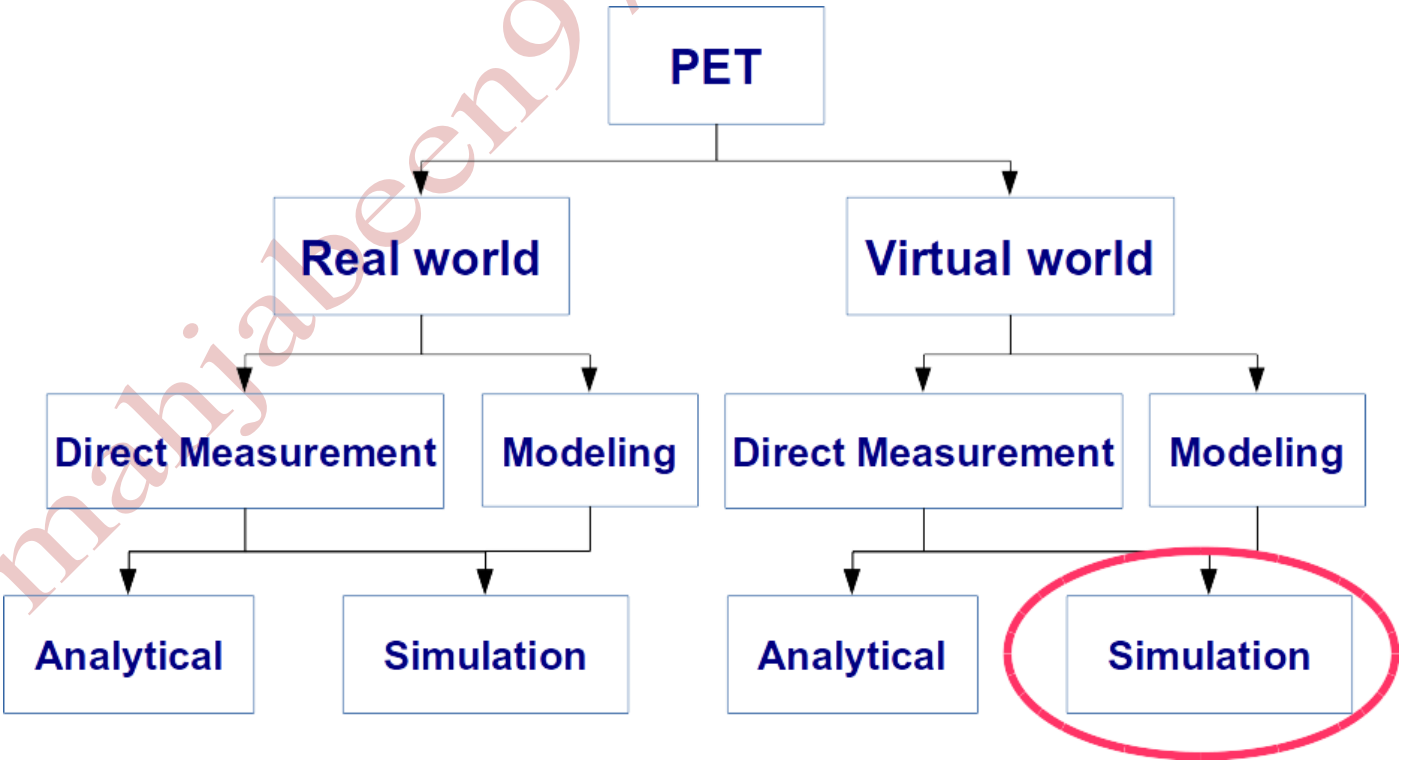
- Efficient and confident

PETs

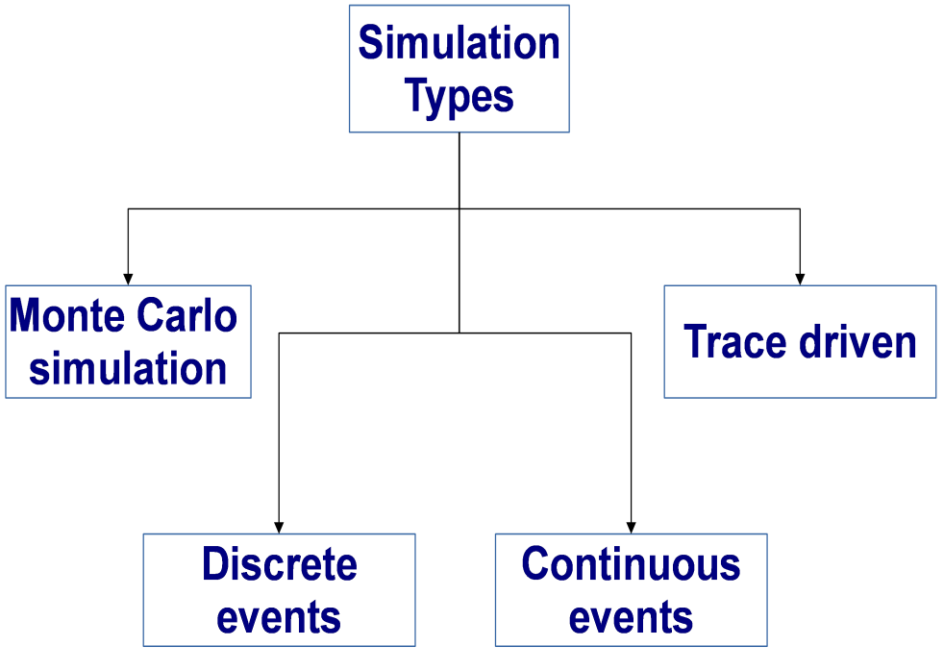
Direct measurement

- 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

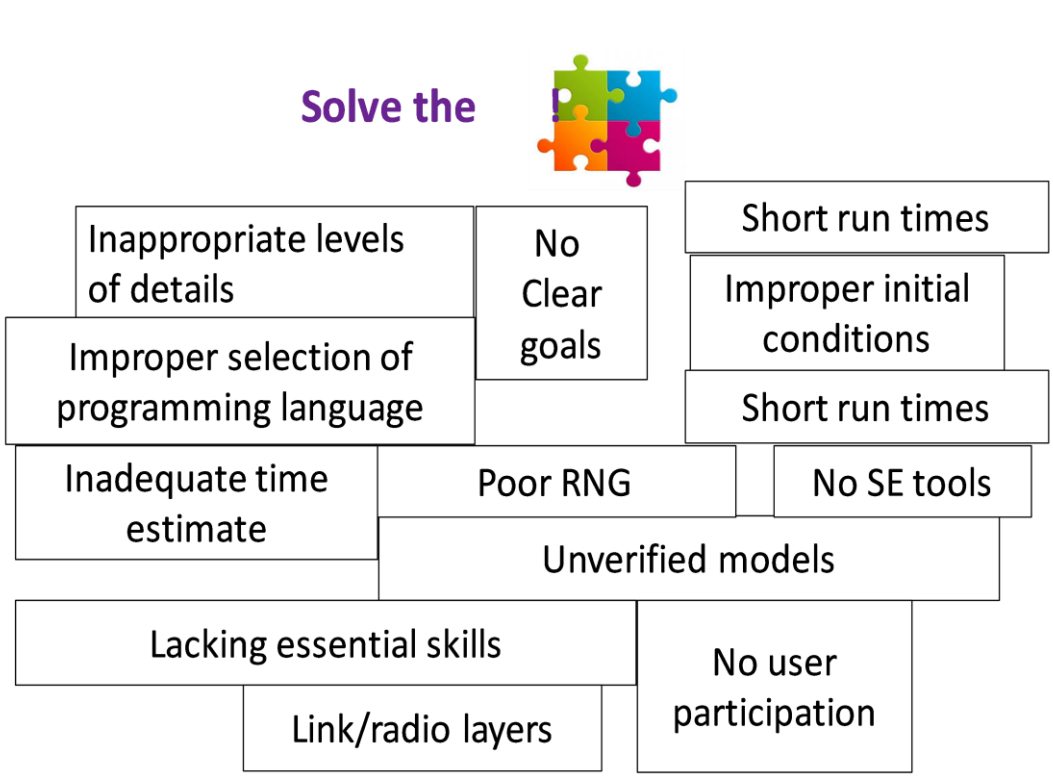
Over reliance on link budget methods for abstraction

- 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

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

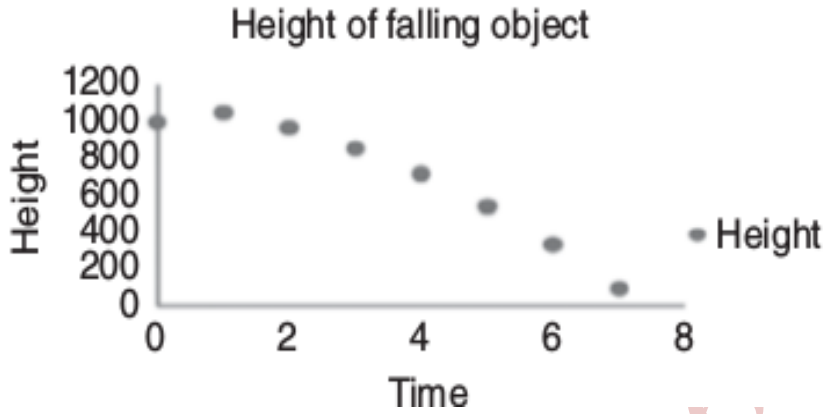
Available
h = height (feet)
t = time in motion (seconds)
v = initial velocity (feet per second, + is up)
s = initial height (feet)
a = acceleration (feet per second per second

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

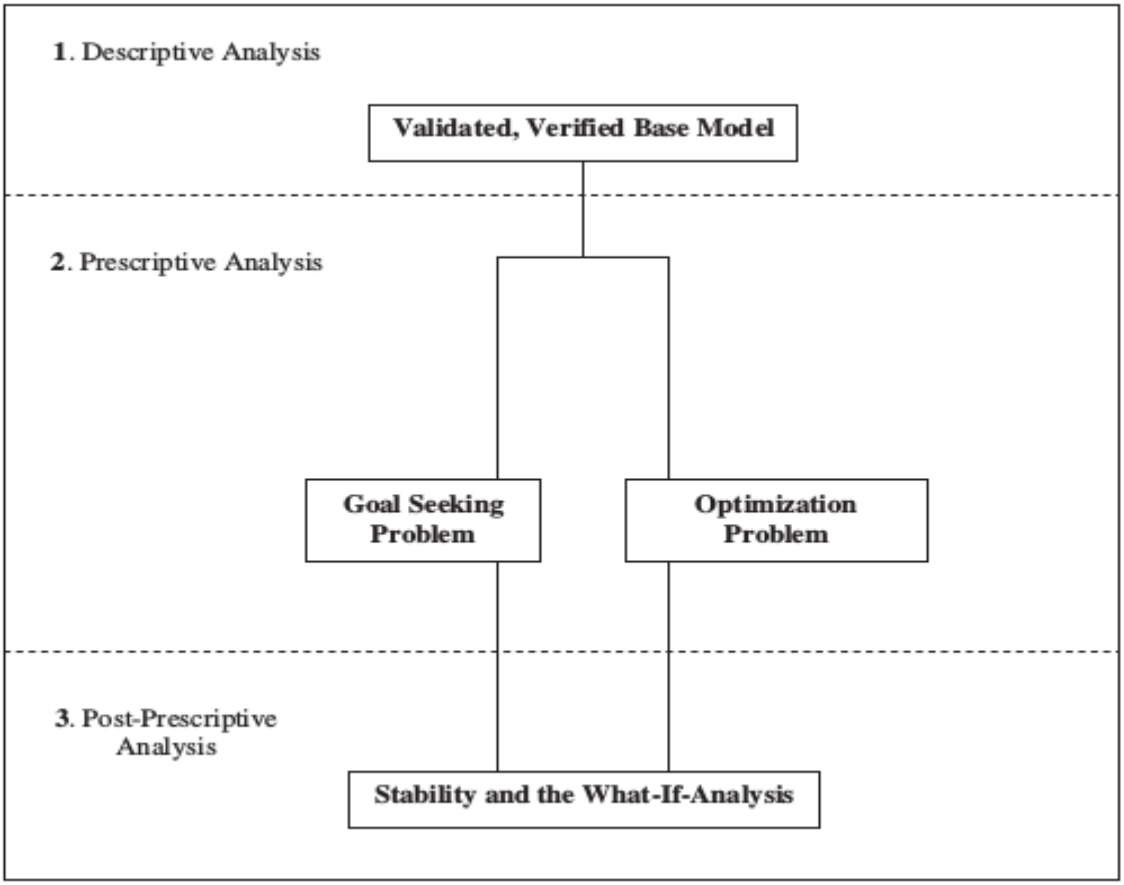
"What-if" analysis

- 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

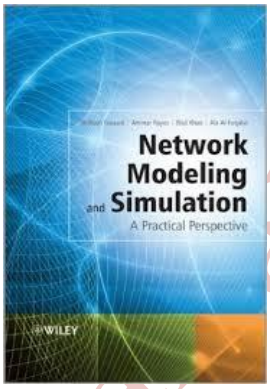
Recommended Text and References

In this module
We shall assess
- NeMS coverage of contents
- Most suitable resources
  - Text books
  - References Books
  - Online Resources

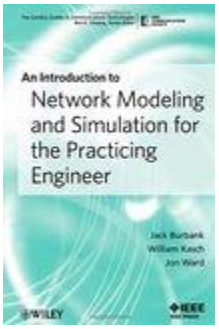
NeMS contents cover
- Well known mathematical models, equations and forms
- Widely used simulation tools and code reusability
- Their inter-relationship

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.

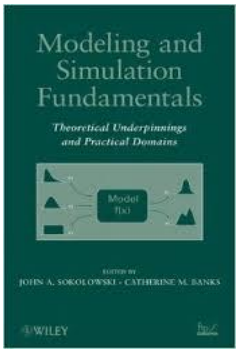


Uptill now
Basics of NeMS
Jack Burbank et al, "An Introduction to Network Modeling & Simulation for the Practicing Engineer" John Wiley , 2011



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

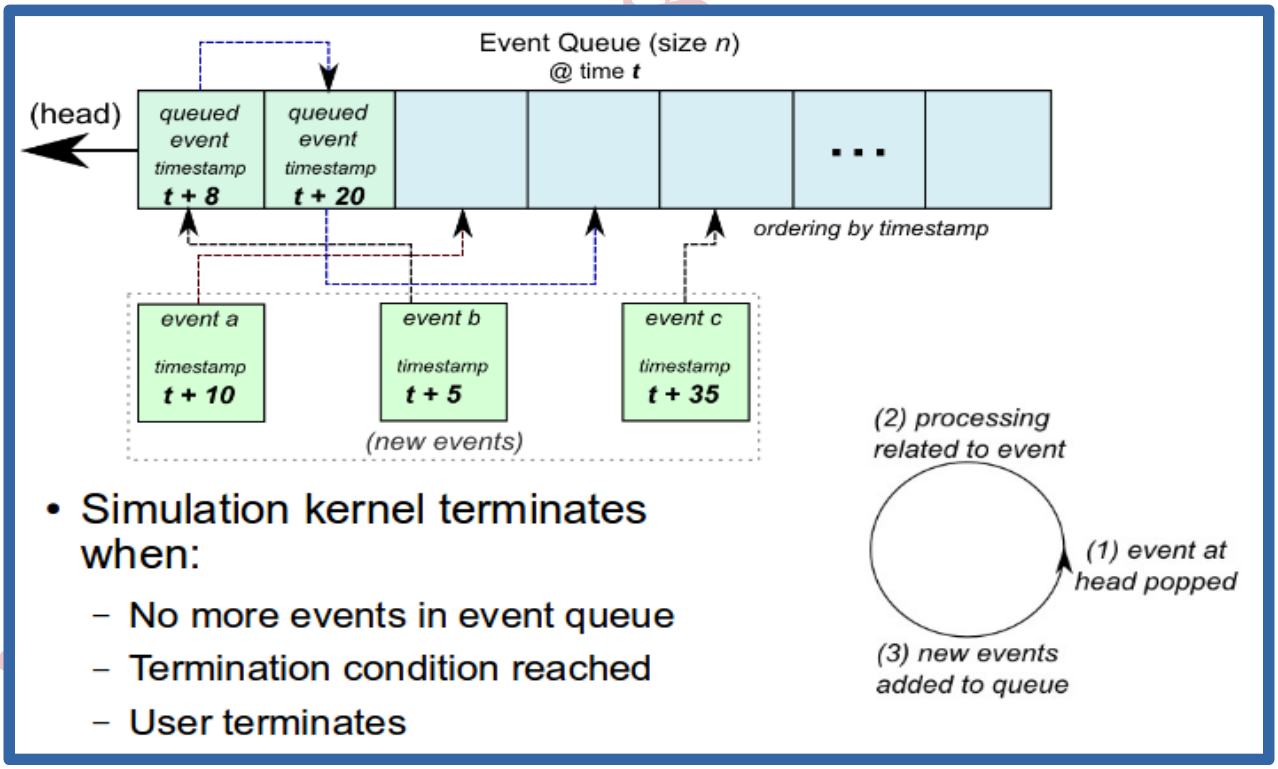
We shall cover

- 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

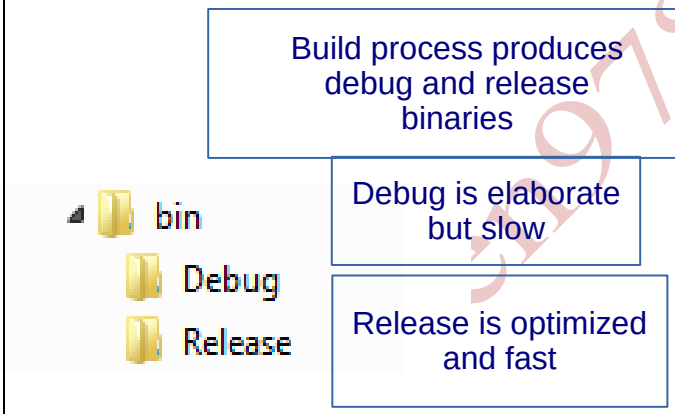
- Compiling and installing on Windows self-contained
- Enter OMNeT++ folder that you unzipped
- Run the file called Mingwenv.cmd

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

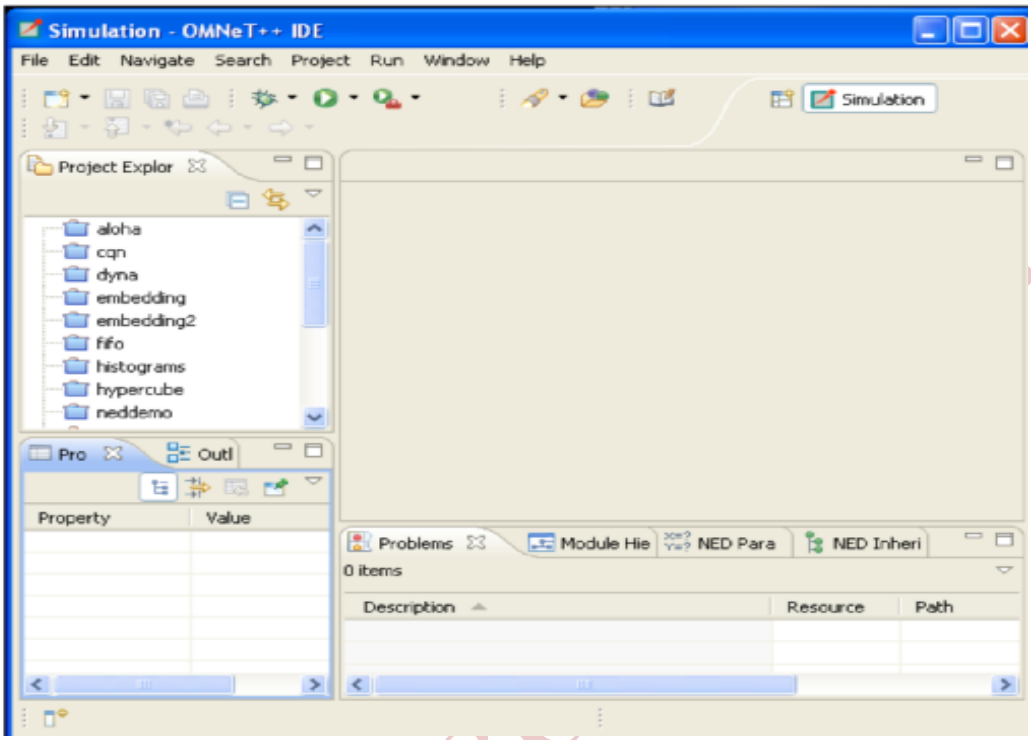
- Does not optimize the binary it produces
- Source code and generated instructions relationship is complex
- Allows accurate breakpoints setting
- Allows code step-through one line at a time
- Compiled with full symbolic debug information

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

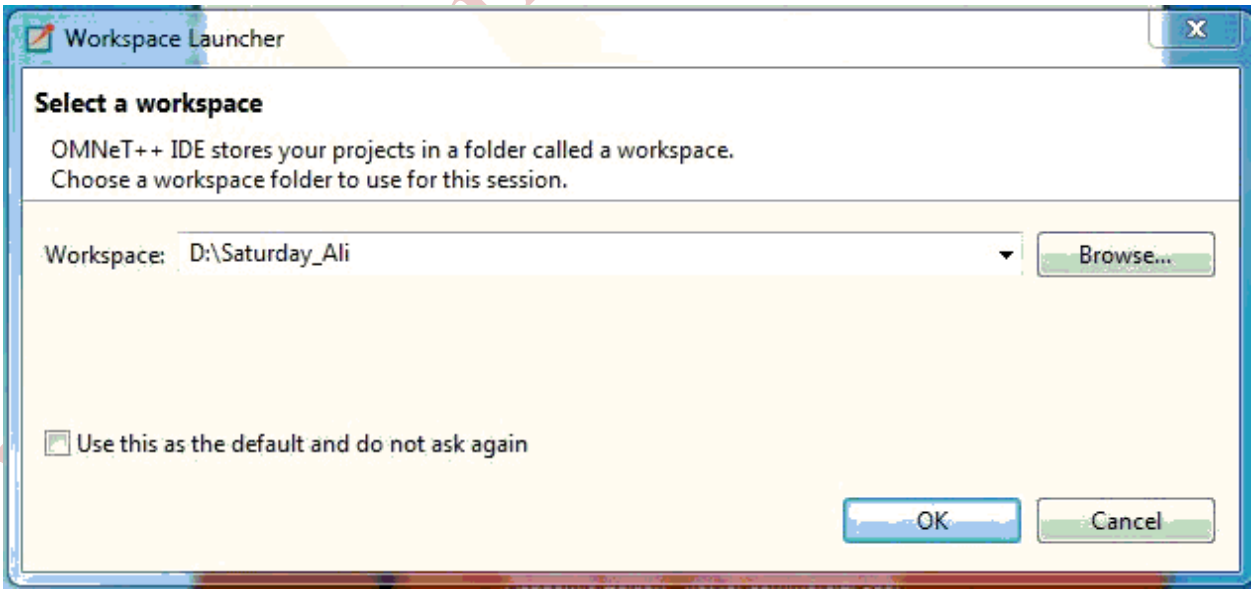
Running first time

- 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



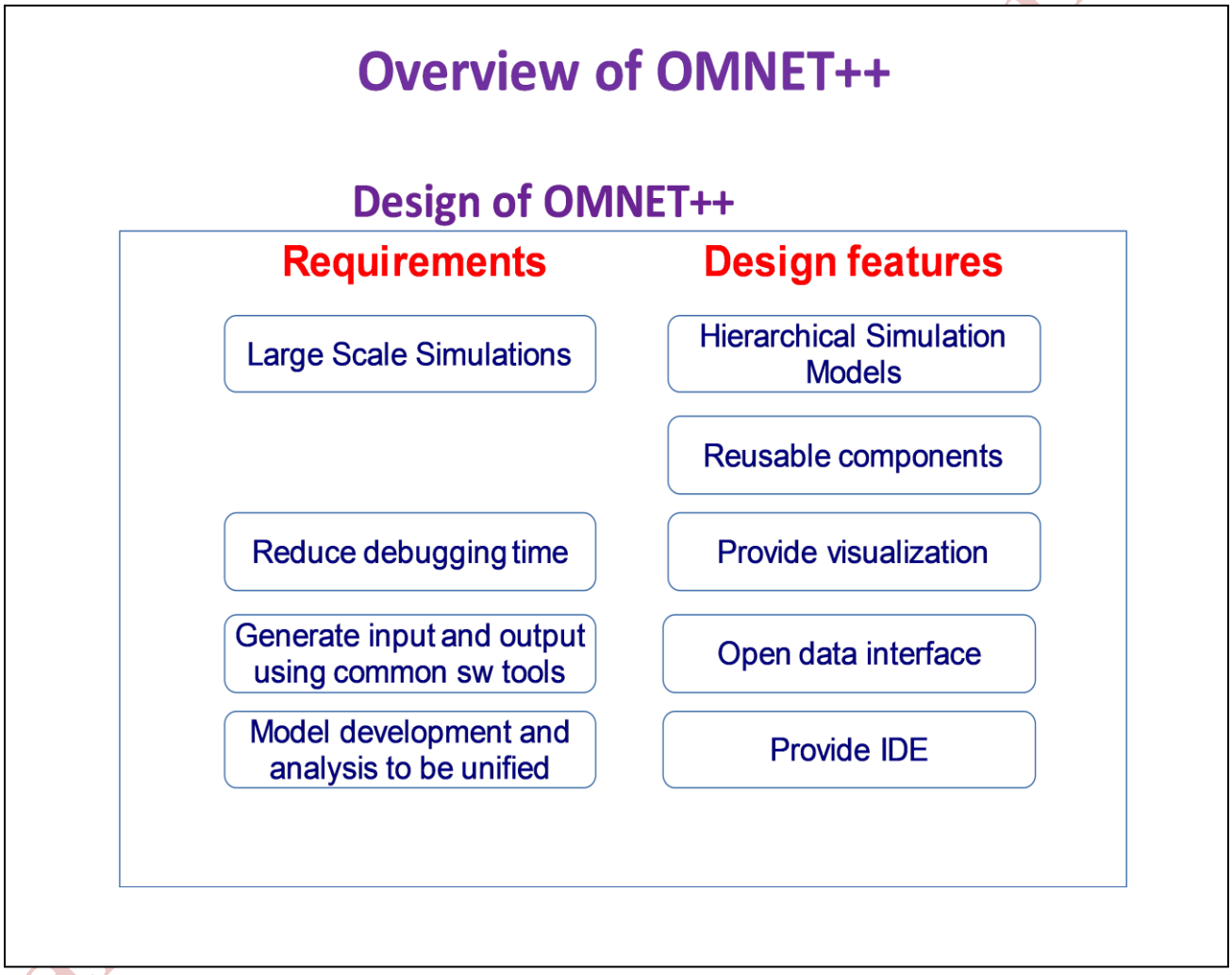
END

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

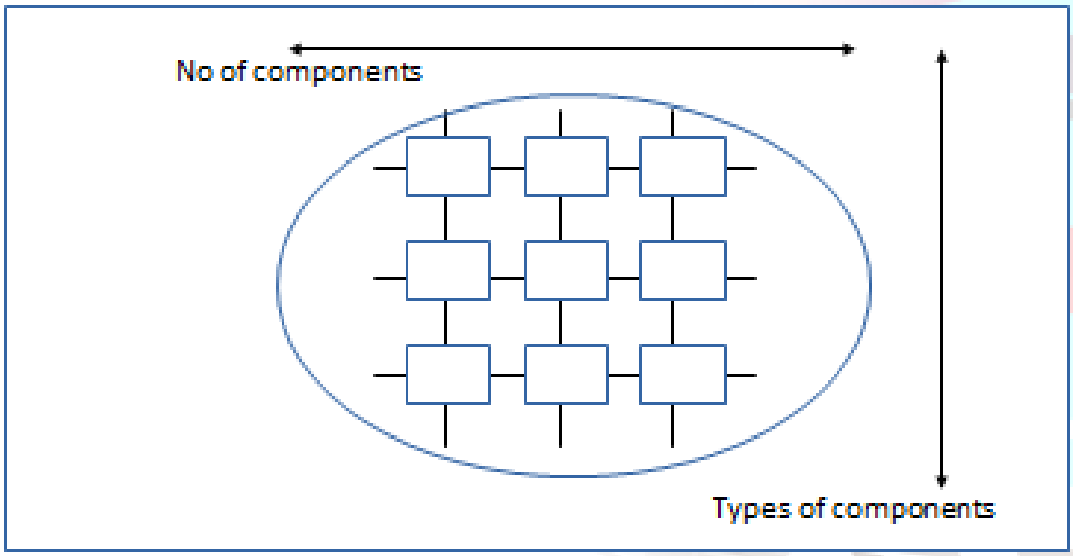
Model Structure

- 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++

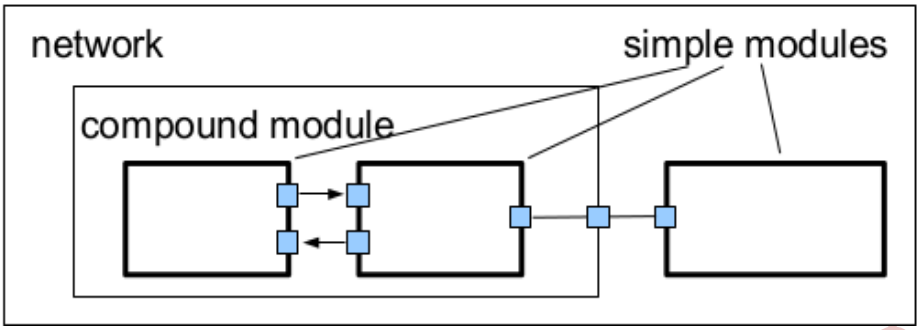
Model Structure

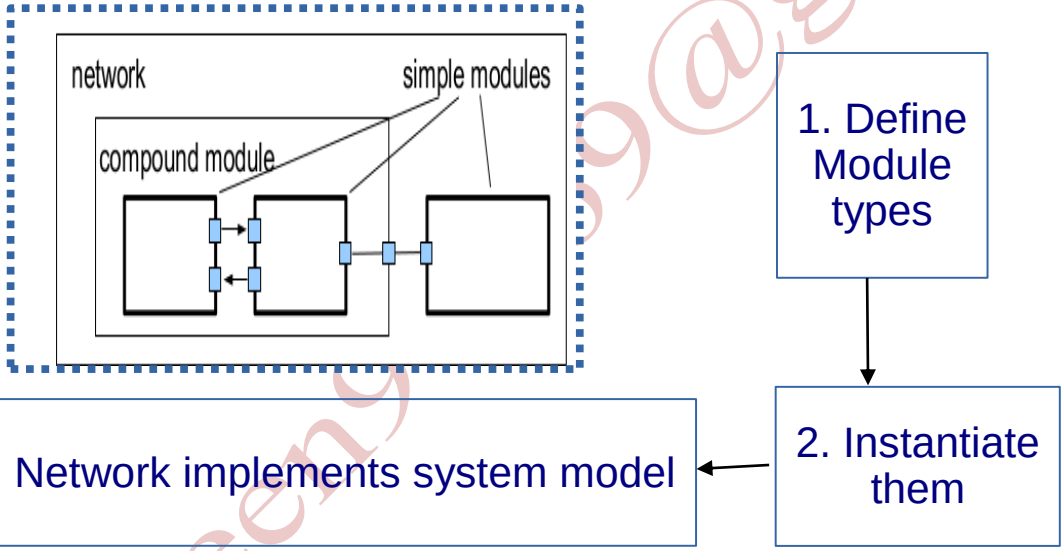
- No. of hierarchy levels not limited



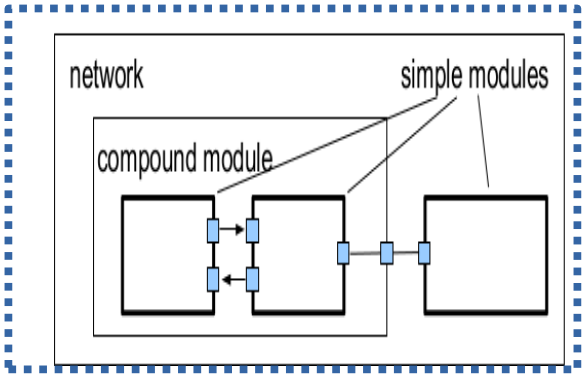
Model Structure

- **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

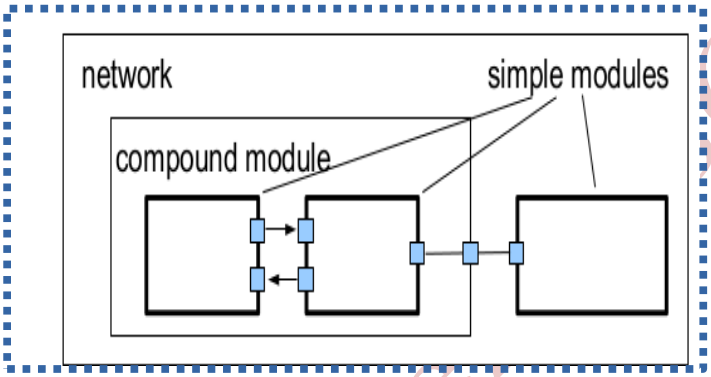


1. Define Module types

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

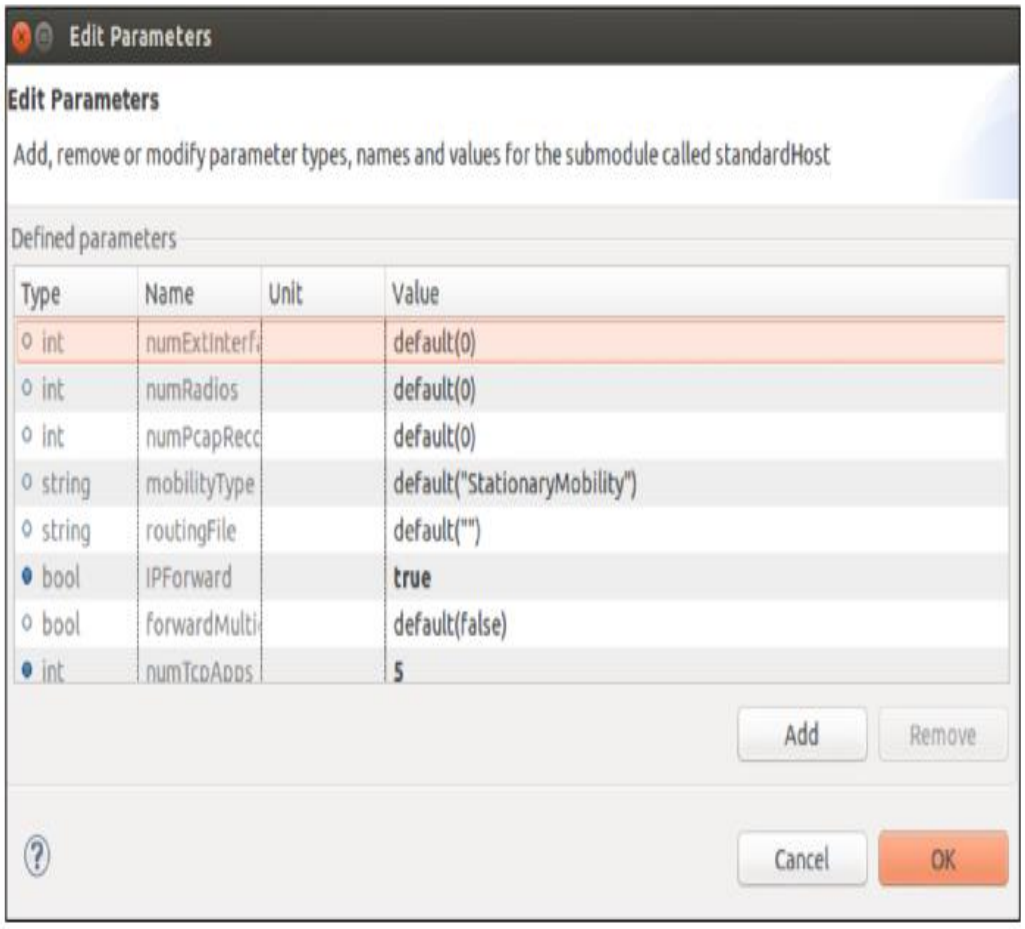
2. Instantiate them

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

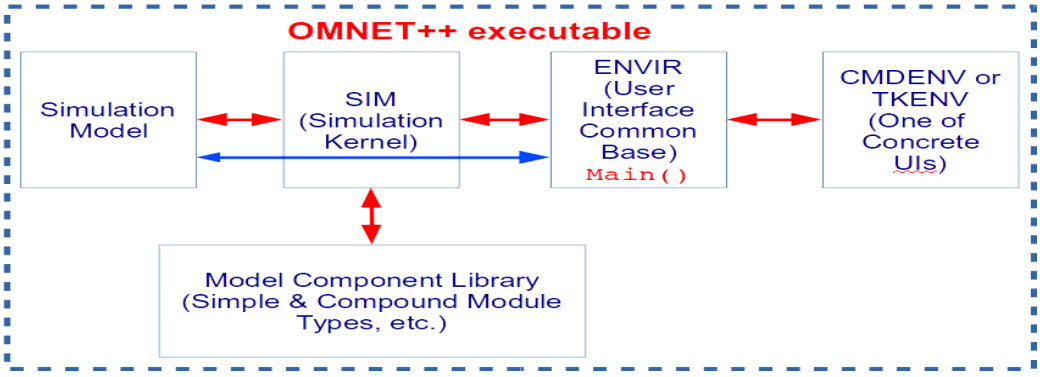
We shall cover

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

Logical Architecture of OMNET++ Simulation Program

Internal Architecture

- OMNeT++ simulation programs possess a modular structure



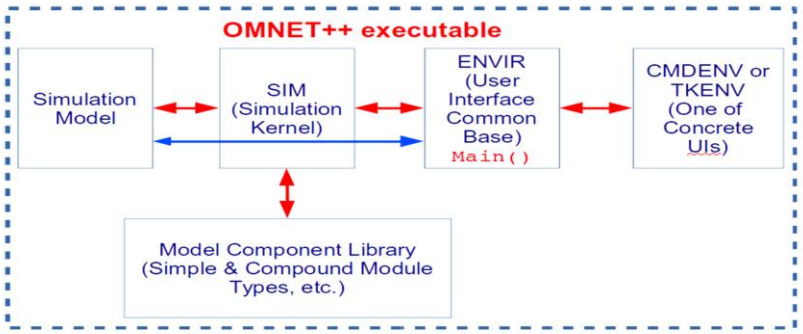
Model Component Library

- Consists of the code of compiled simple and compound modules

Logical Architecture of OMNET++ Simulation Program

Internal Architecture

- OMNeT++ simulation programs possess a modular structure



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

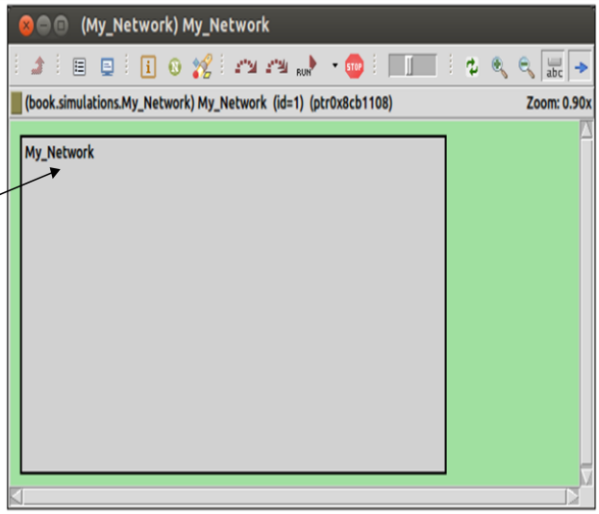
Compound module definitions

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

Overview of OMNET++

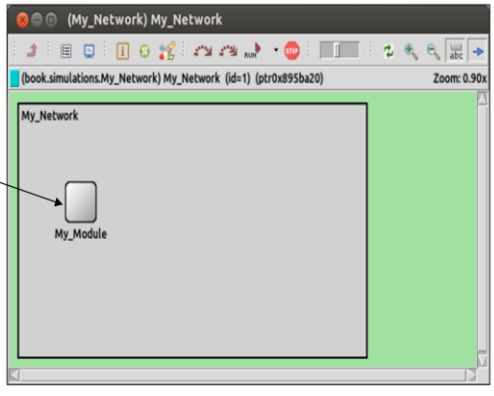
Let us create a topology called My_Network using Graphical Editor

Network name
=
My_Network



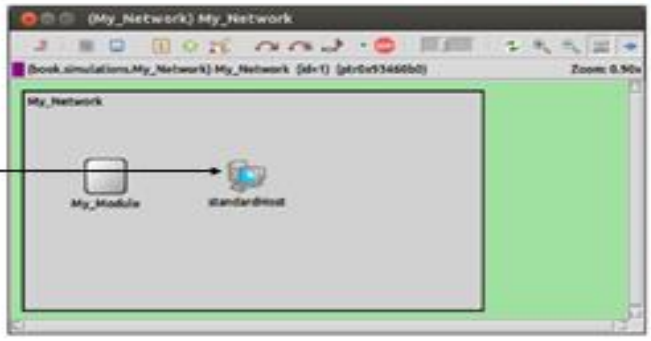
Overview of OMNET++

Module name
=
My_Module



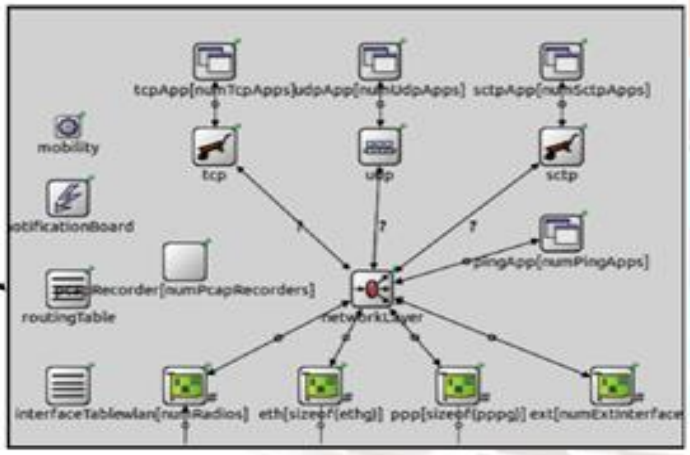
Overview of OMNET++

Compound module name= standardHost



Overview of OMNET++

Inside standardHost



TOPIC 21

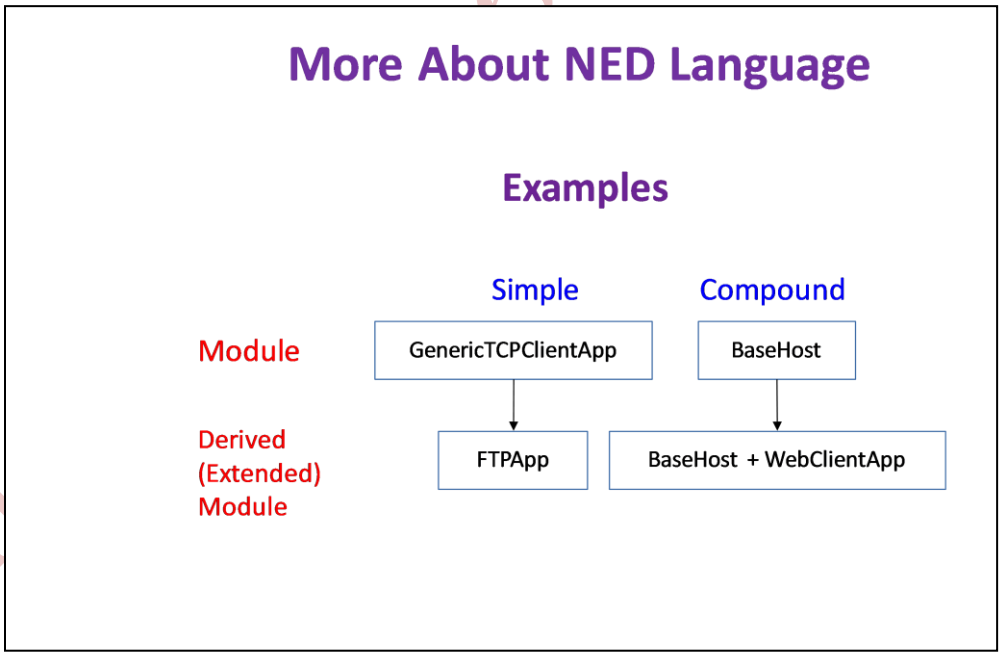
More About NED Language

In this module
We shall cover more
Details about NED

- Inheritance
- Interfaces
- Packages
- Inner types

Inheritance

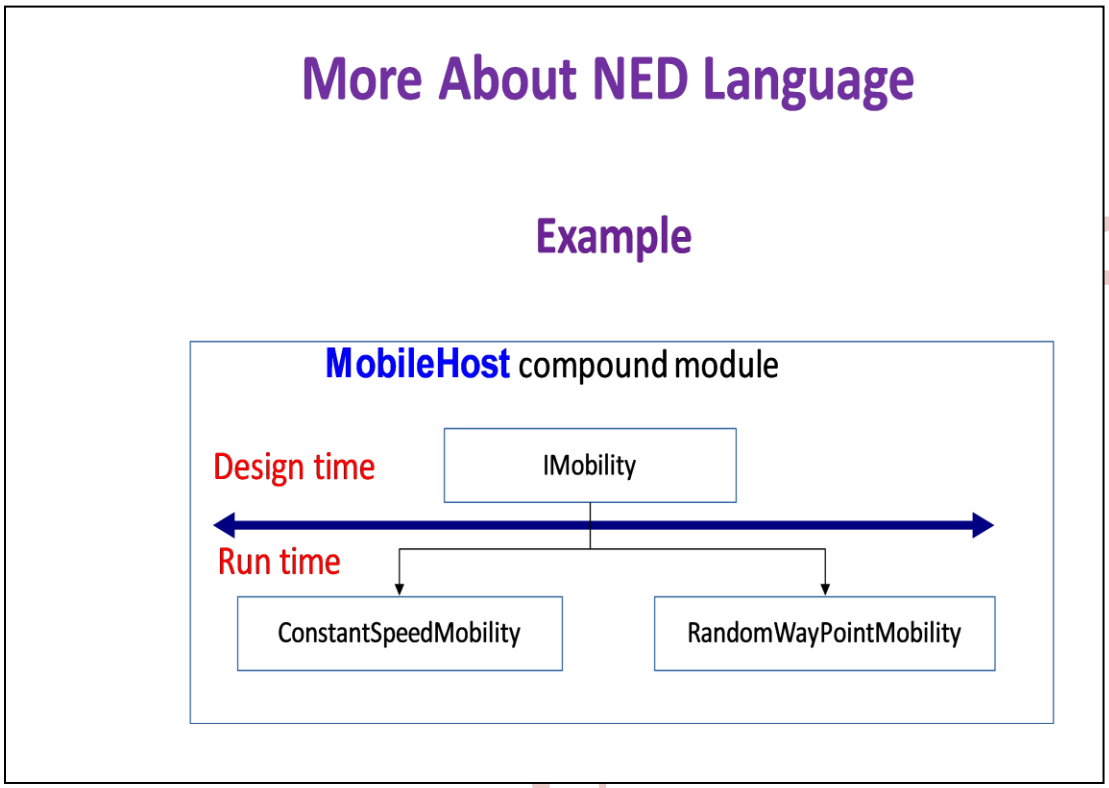
- Modules and channels can be subclassed
- Derived modules and channels may add
  - New parameters
  - Gates
- Similarly compound modules may add
  - New submodules
  - Connections



Interface instantiation

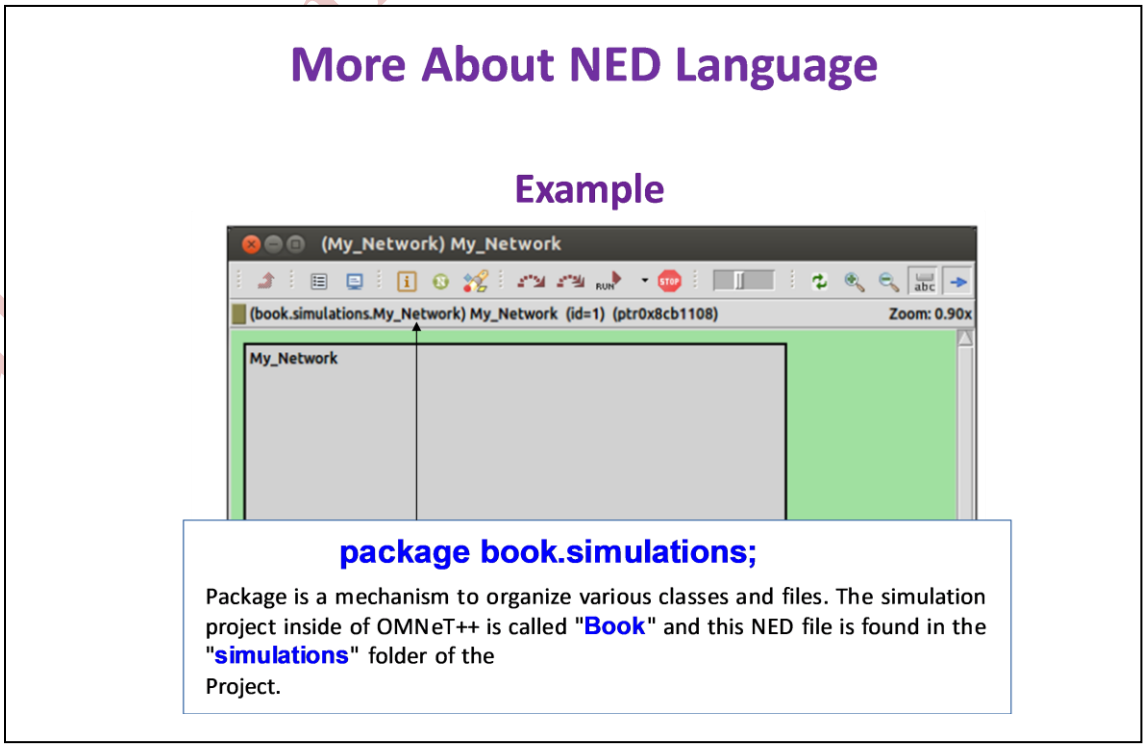
- Module and channel interfaces can be used as a placeholder
  - where normally a module or channel type would be used

- Concrete module or channel type determined
  - At network setup time by a parameter



Packages

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



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

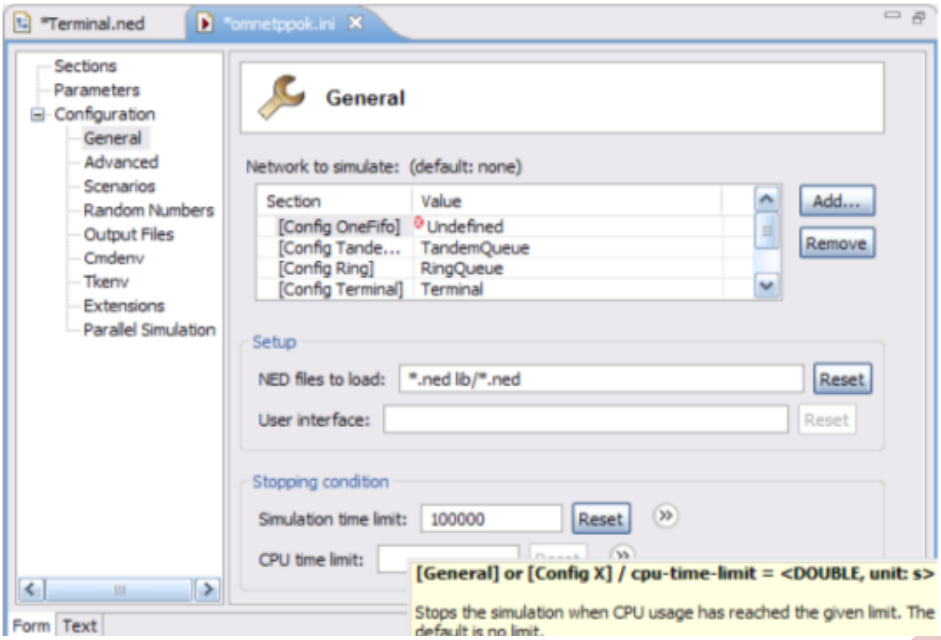
Separation of Model and Experiments
- Always good practice to try to separate different aspects of simulation
- **Model topology**
  - NED file
  - MSG file
- **Model behavior**
  - C++ code
- Provides cleaner model

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

My_Network
wildcarded as **

Example omnet.ini

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

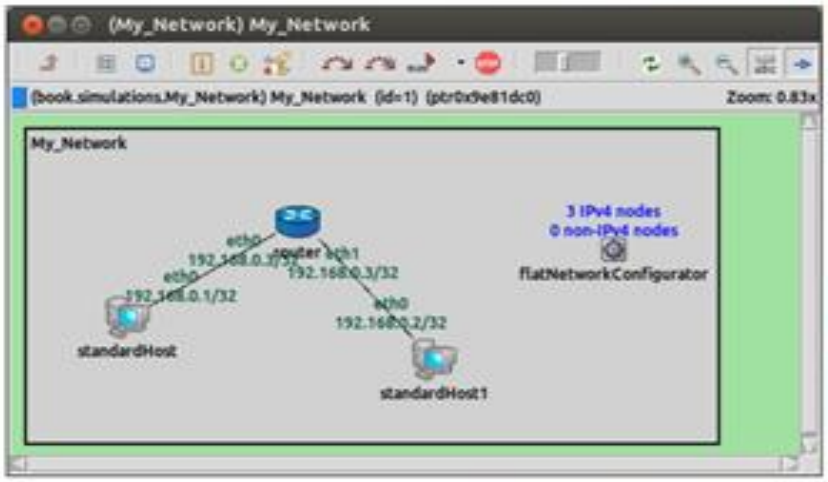
Example

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:

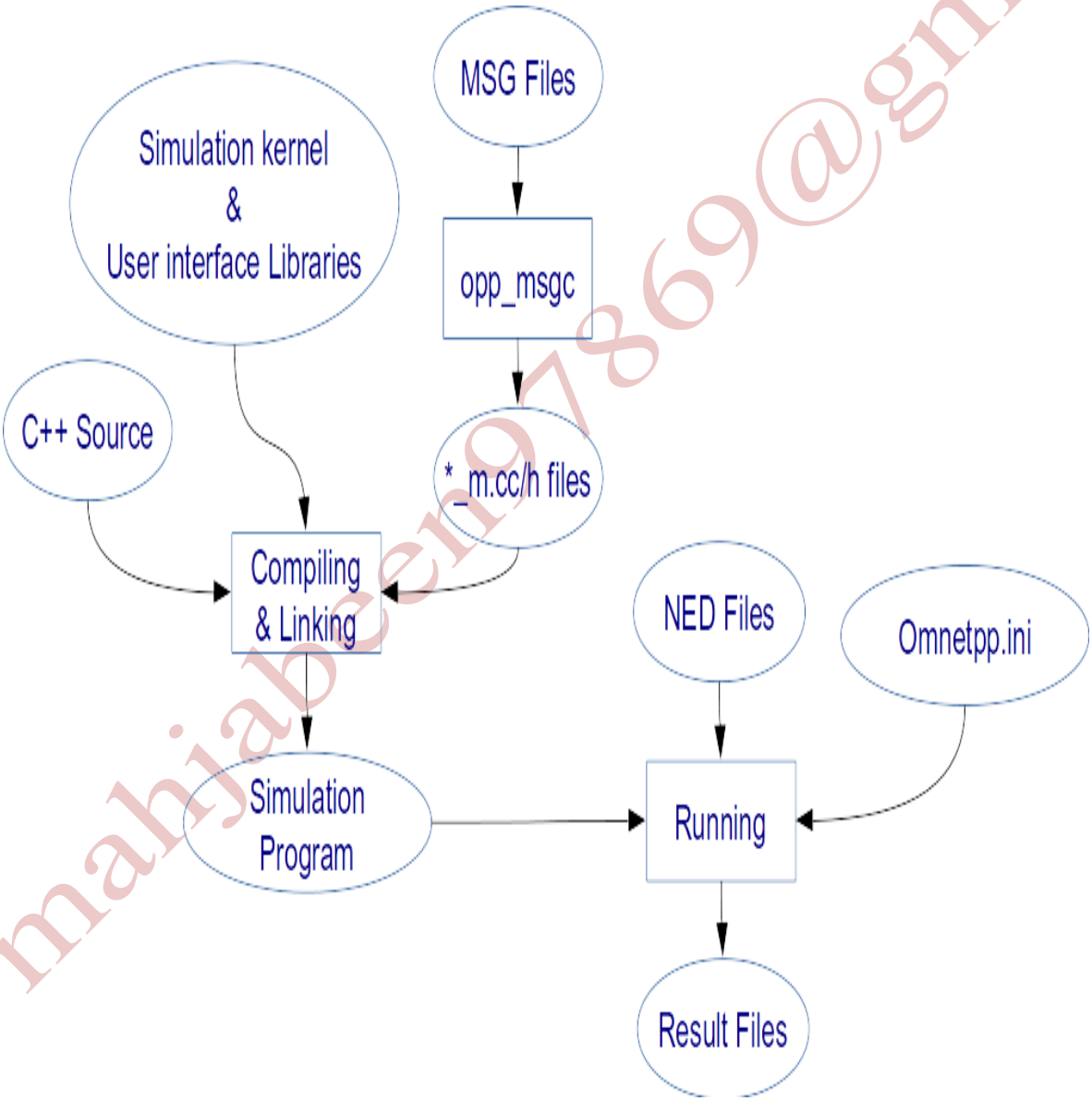
In this module

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

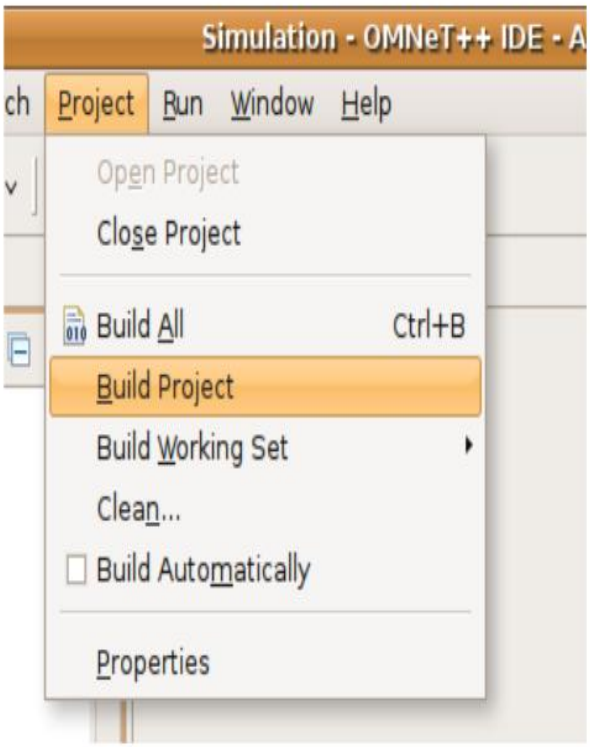
any C/C++ program from source

  - 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

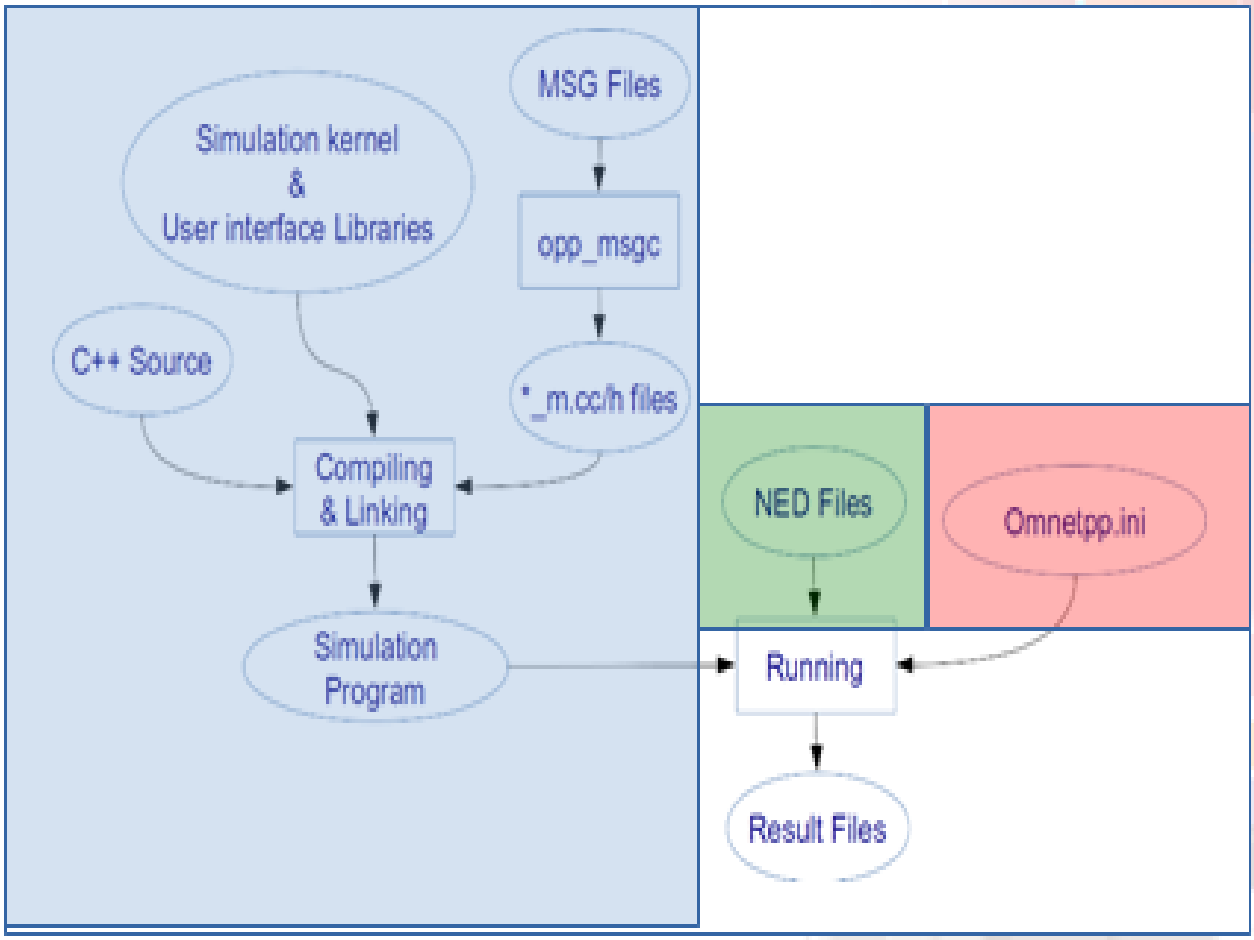
$ make

- 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

Where to next!



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

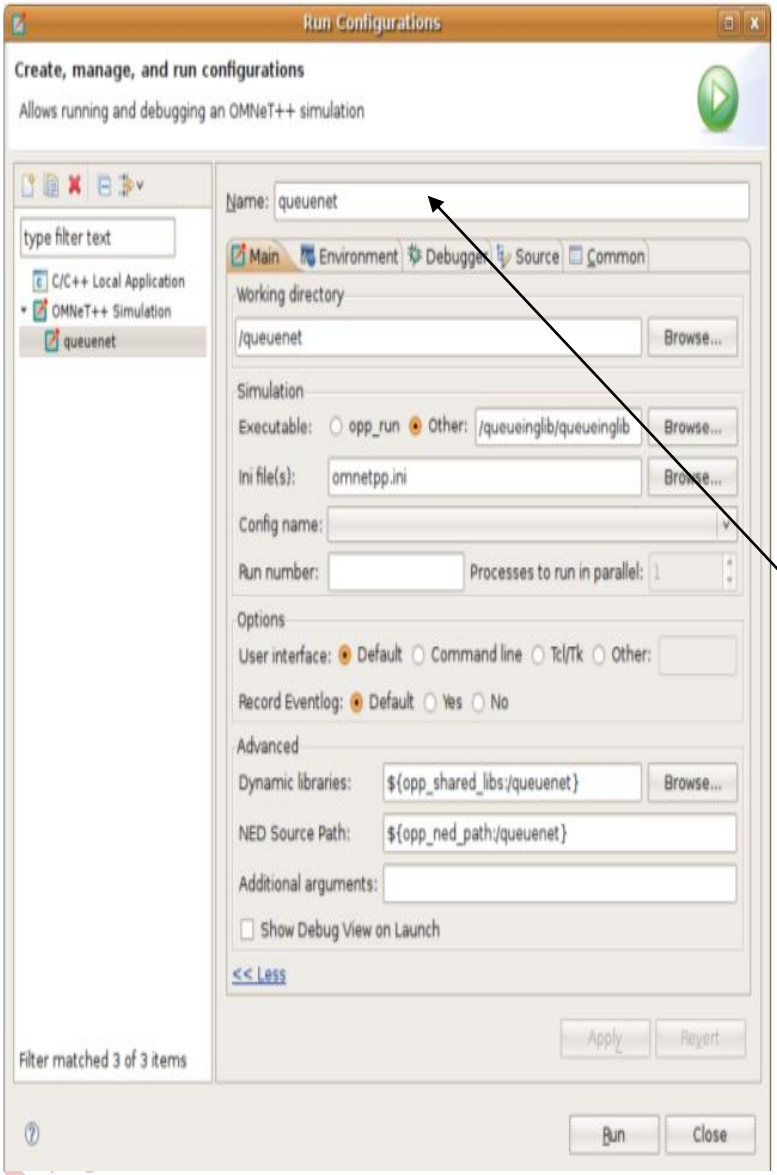
- Single runs
- Batch runs
- Run numbers
- Graphical mode (Tkenv)
- Command mode (Cmdenv)
- Simulation configuration
- Recording event logs
- Debug support

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

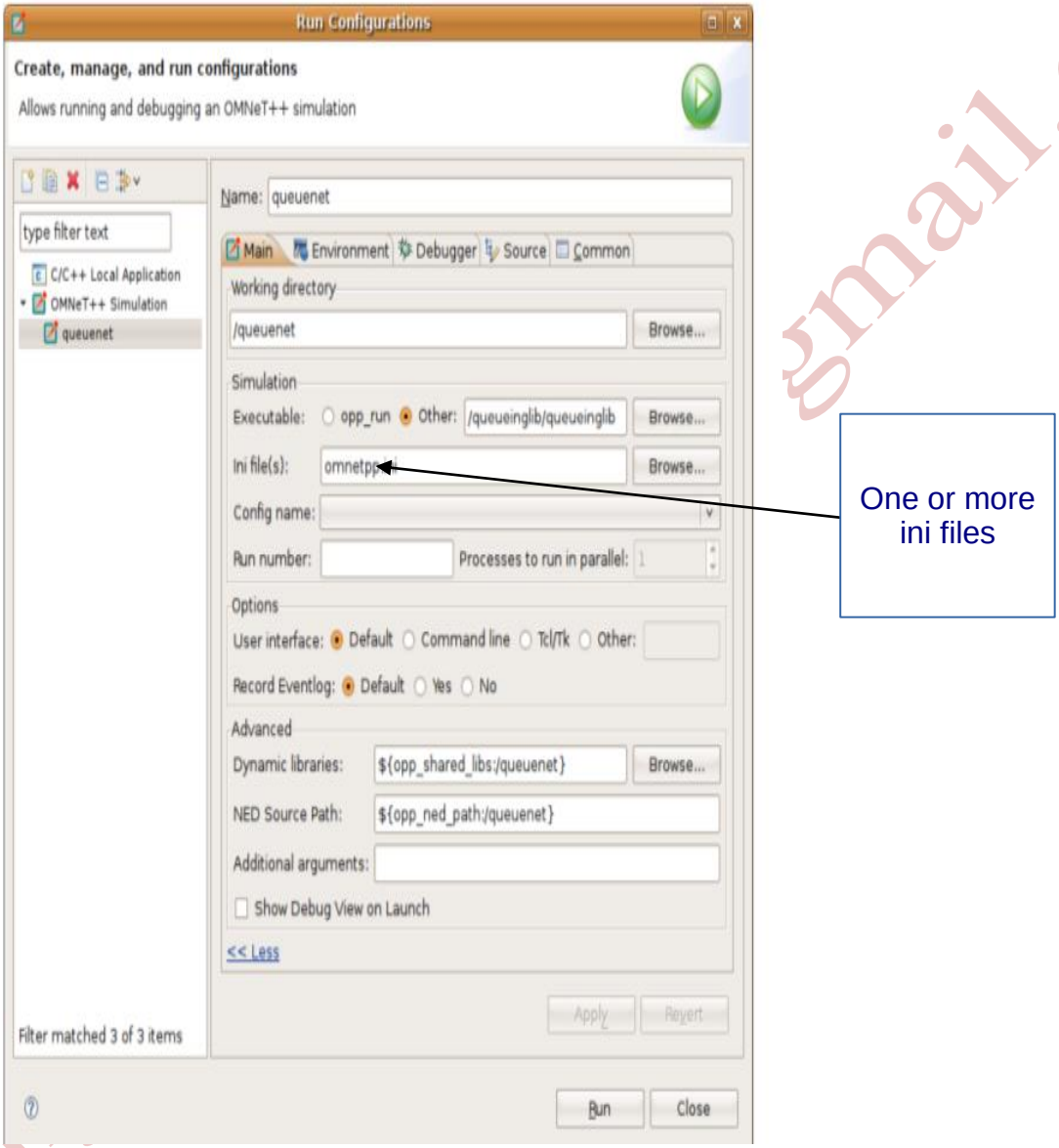


Run omnet.ini From /queuenet

queuenet Launch Configuration

Running Simulations

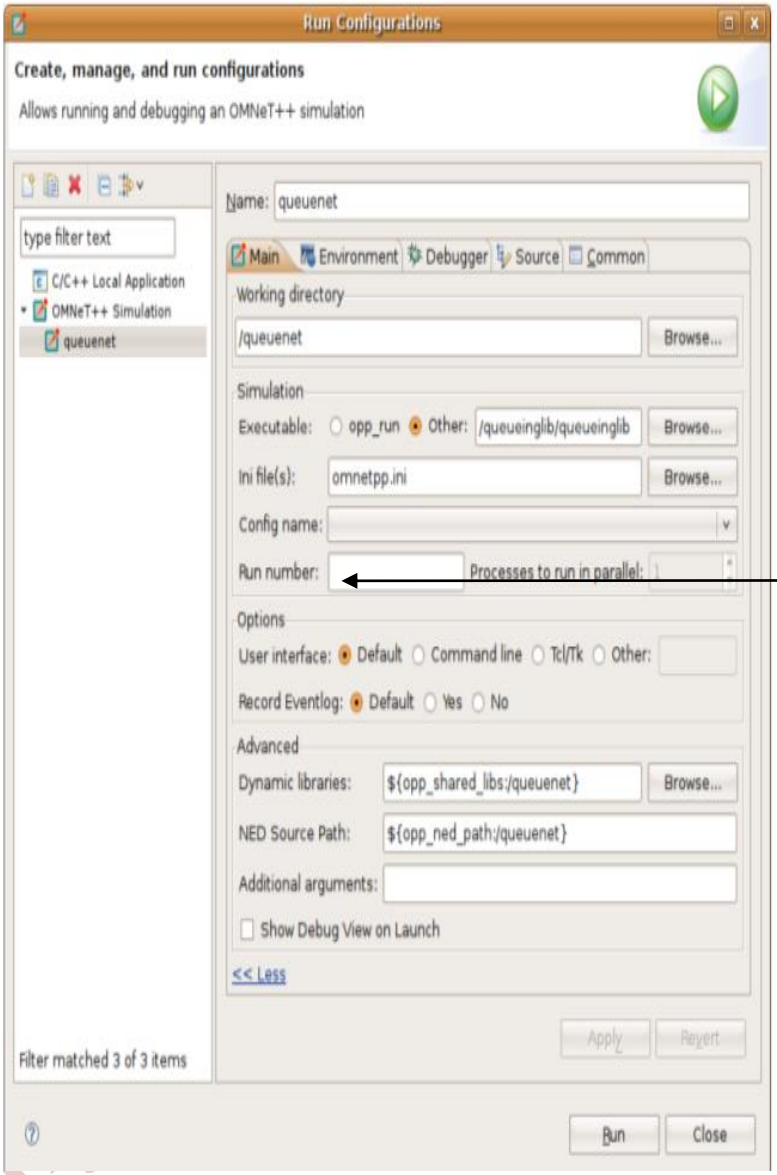
Launch Configuration



One or more ini files

Running Simulations

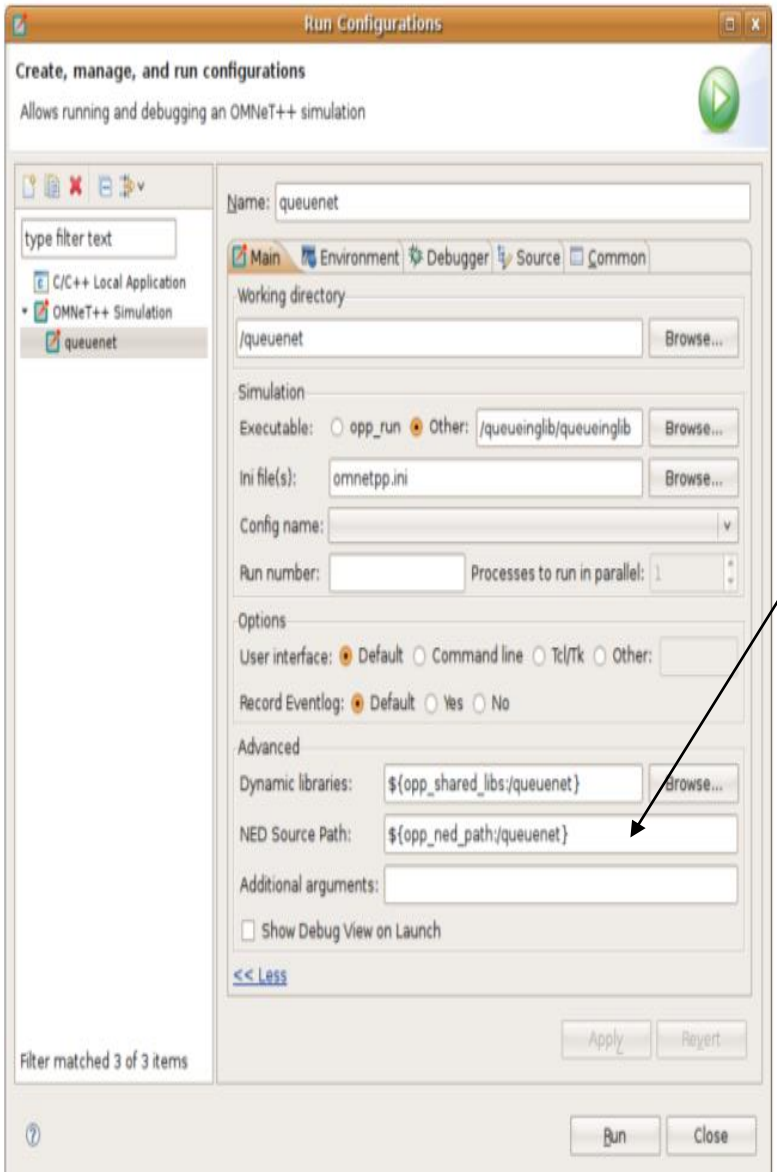
Launch Configuration



Run number
R = 0
One omnet.ini
one exe

Running Simulations

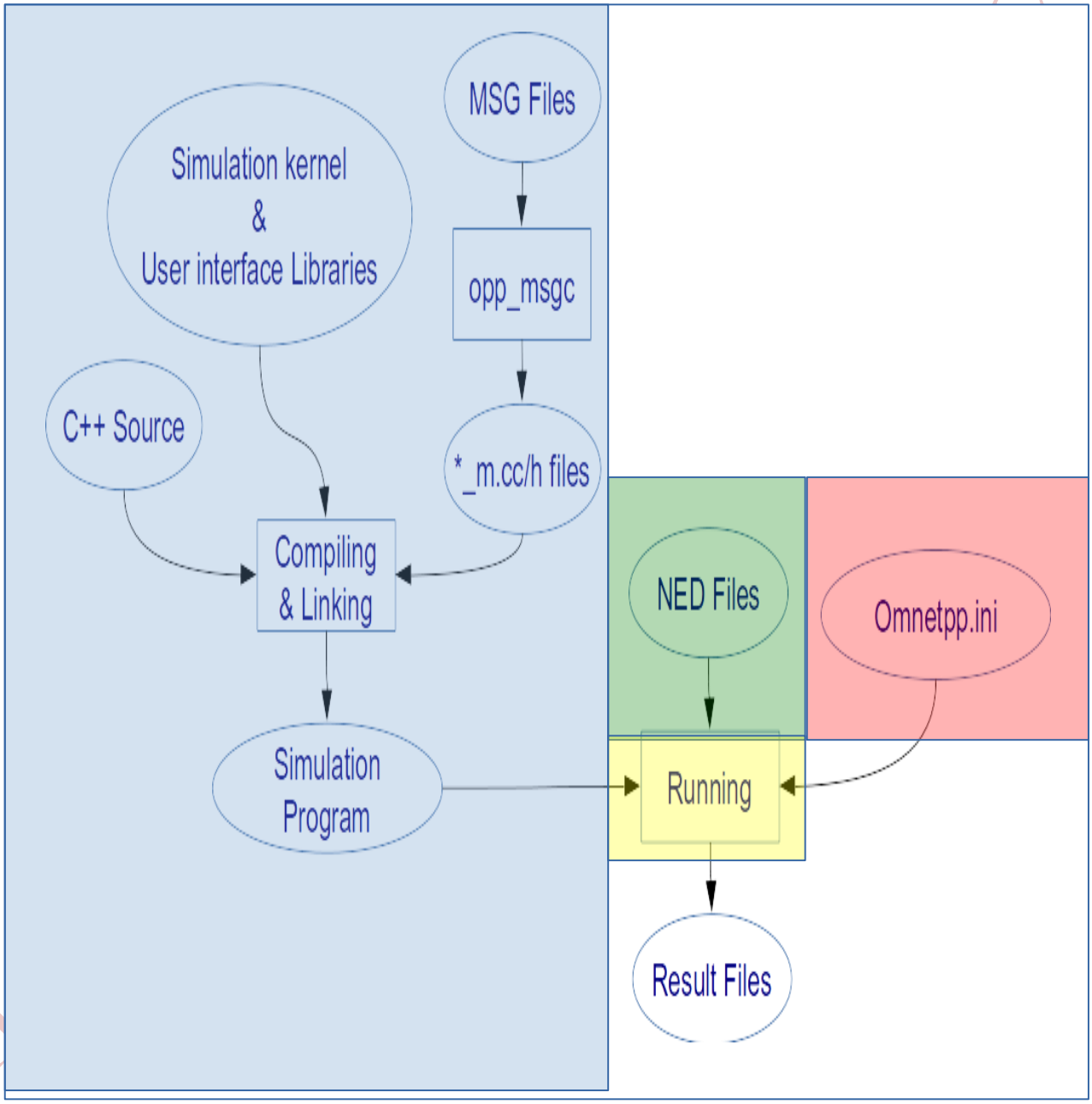
Launch Configuration



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

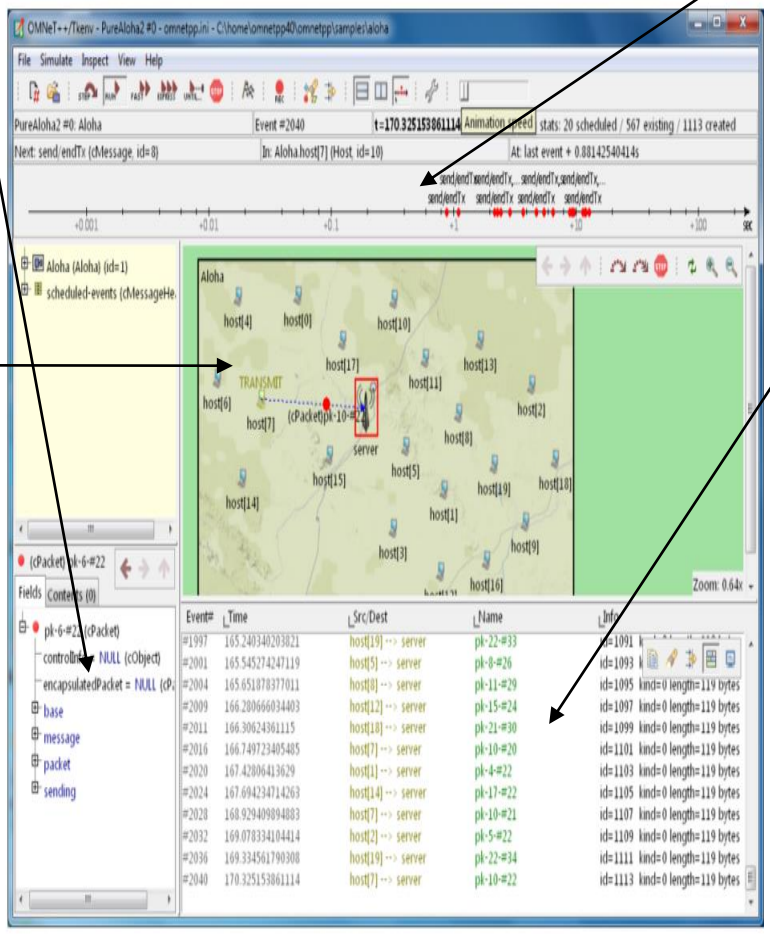
Tkenv in action

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

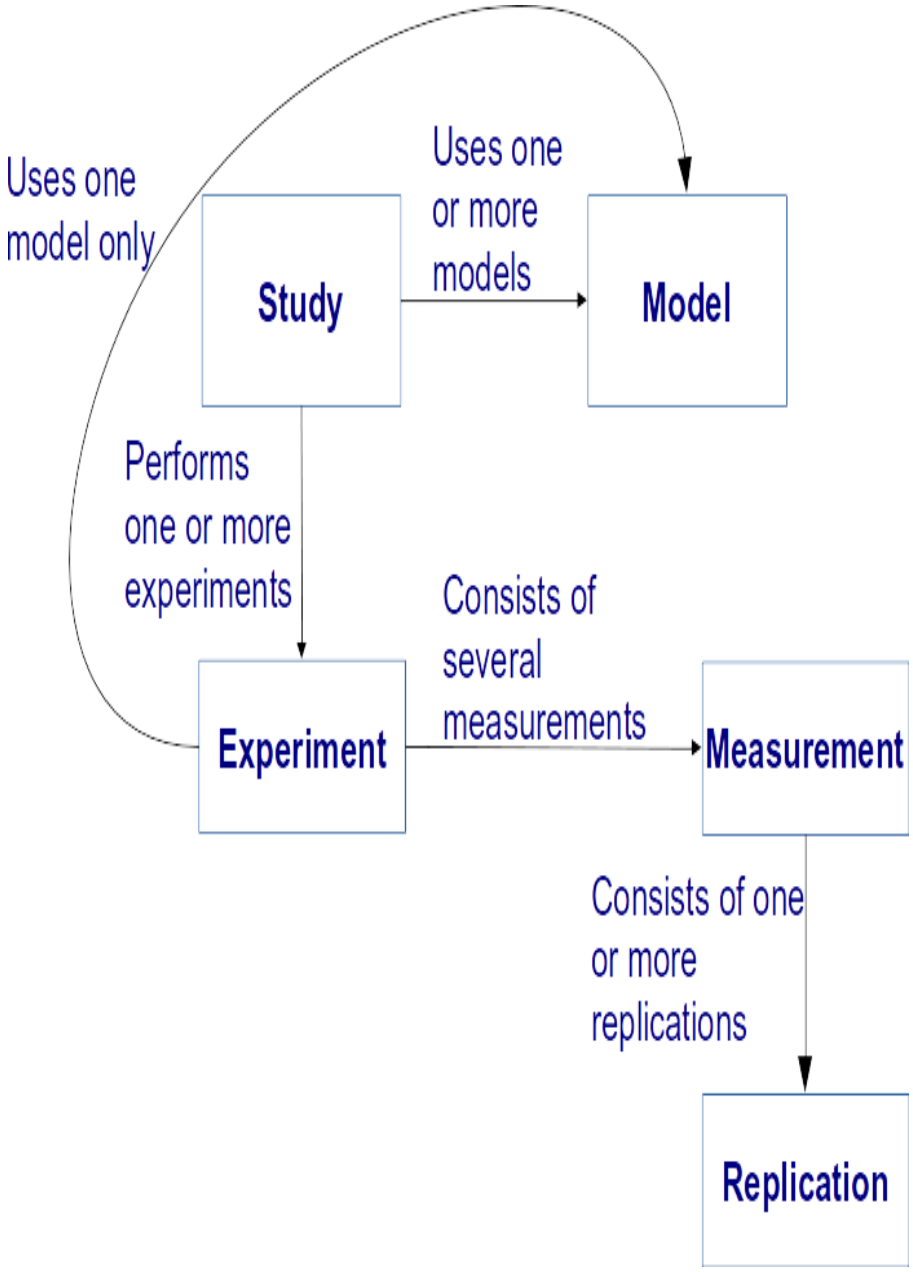
98 of 112 (87.5%) did not include confidence intervals

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

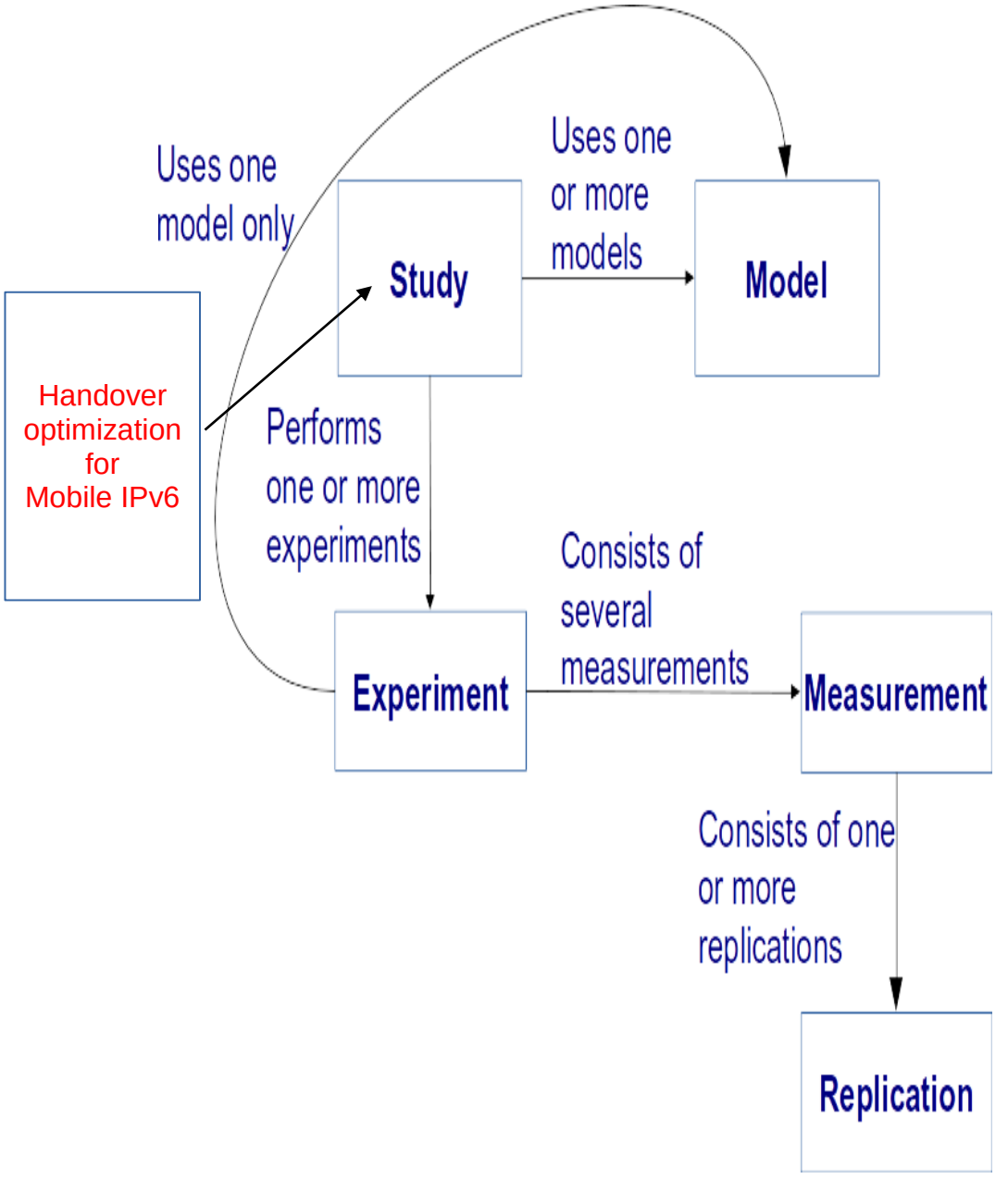
- One repetition of a measurement
- Replication can be characterized by the seed values it uses

How to organize experiments
(6 of 6)
Run

- One instance ofrunning the simulation
- Characterized by
  - exact time/date
  - computer (host name)

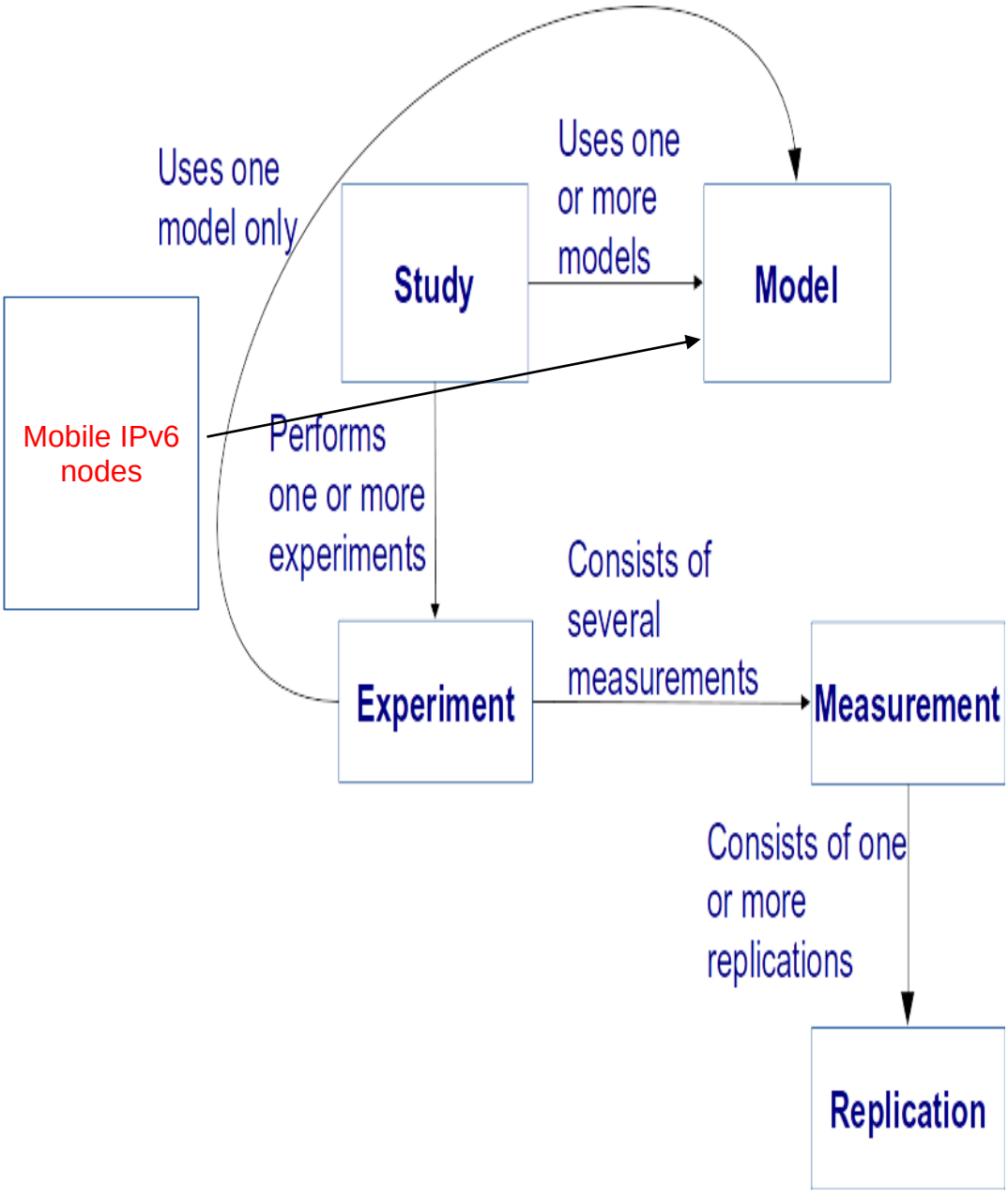
Organizing and Performing Experiments

Example



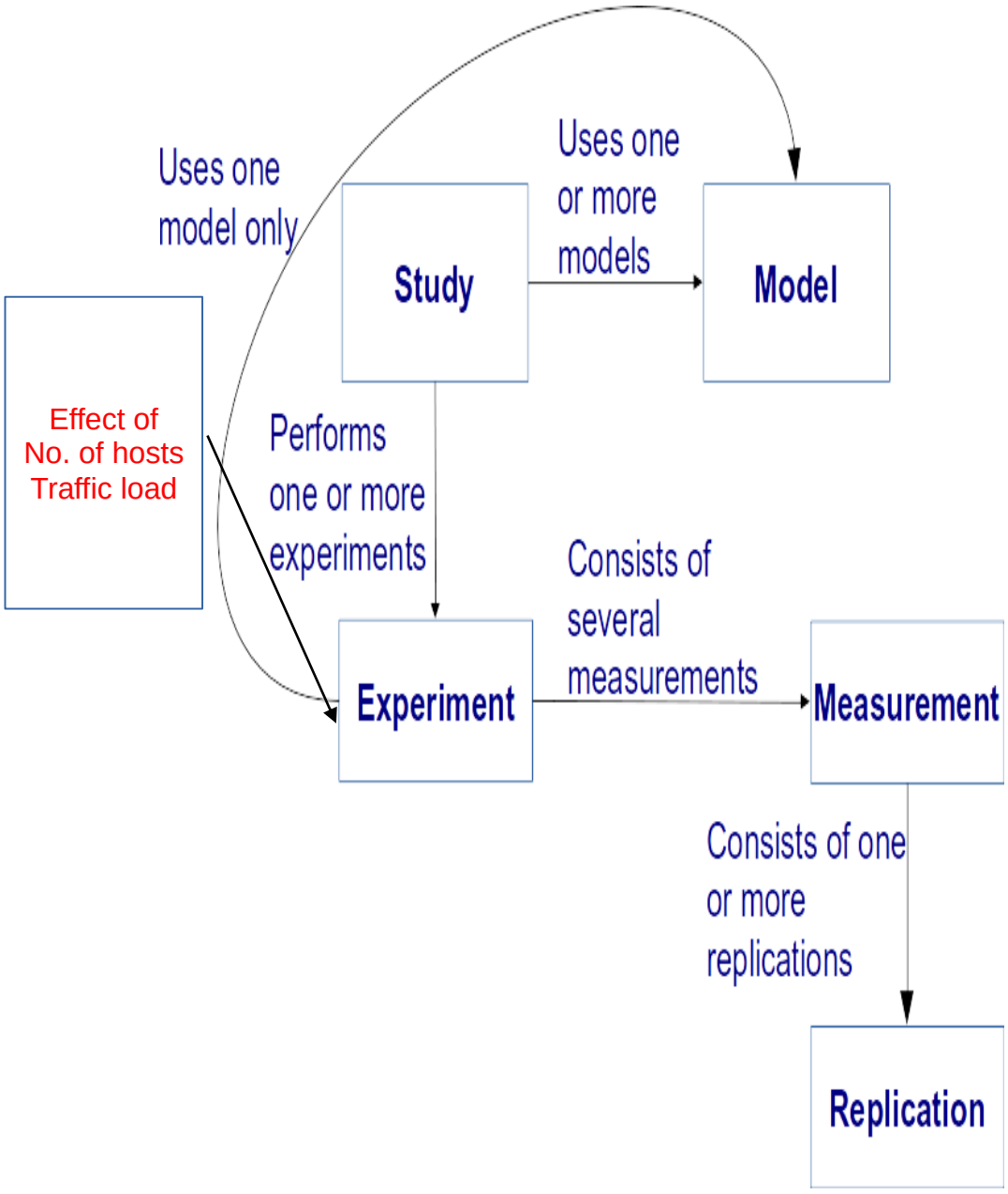
Organizing and Performing Experiments

Example



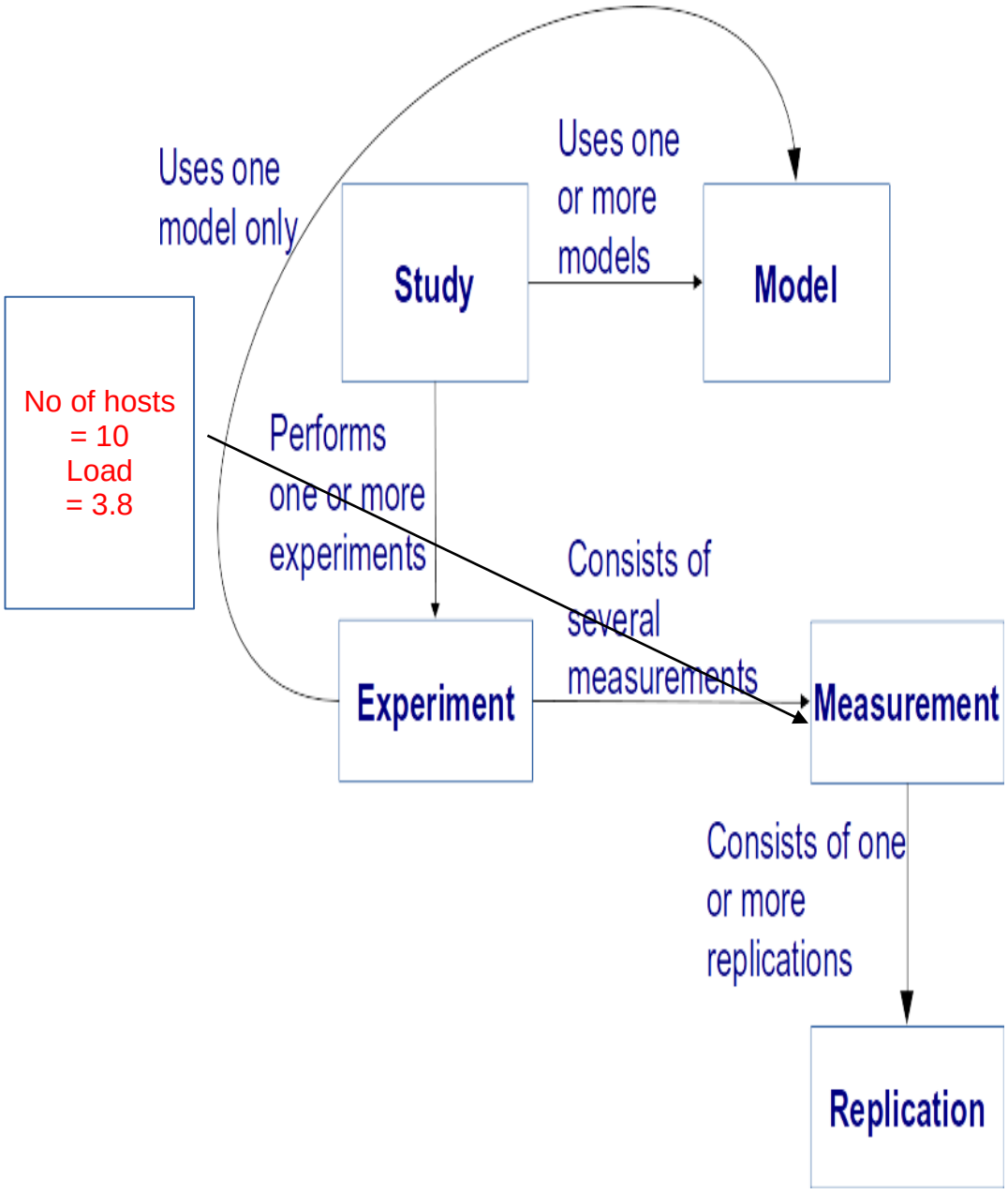
Organizing and Performing Experiments

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)

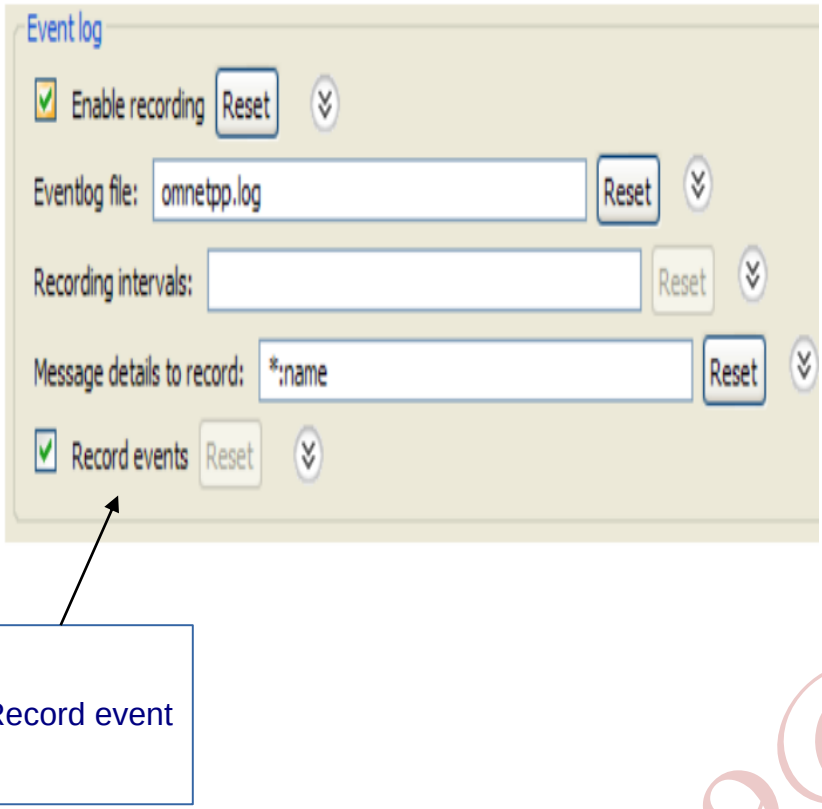
- Type

$ 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



Sequence Chart

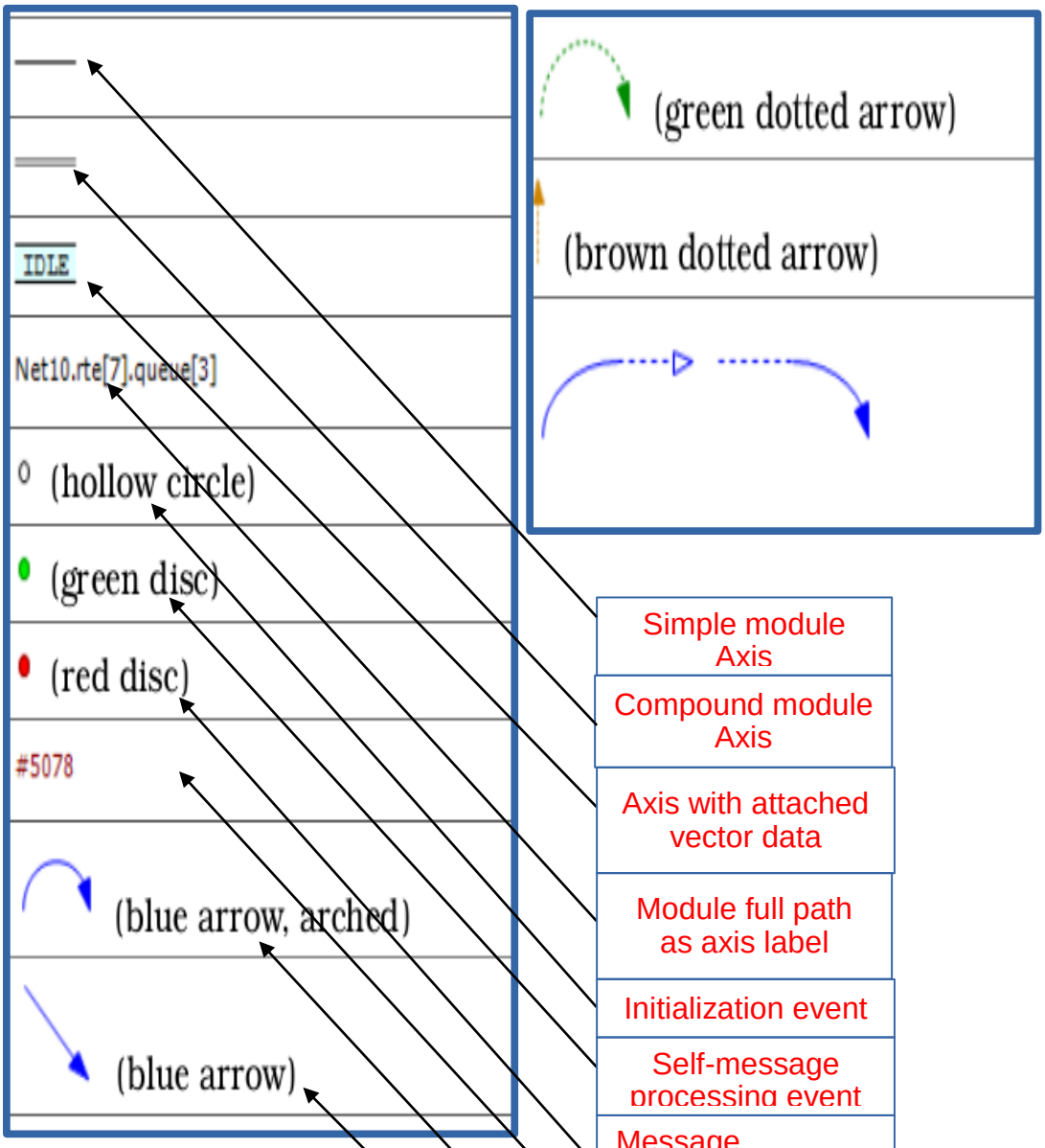
- Displays eventlog files

in a graphical form

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

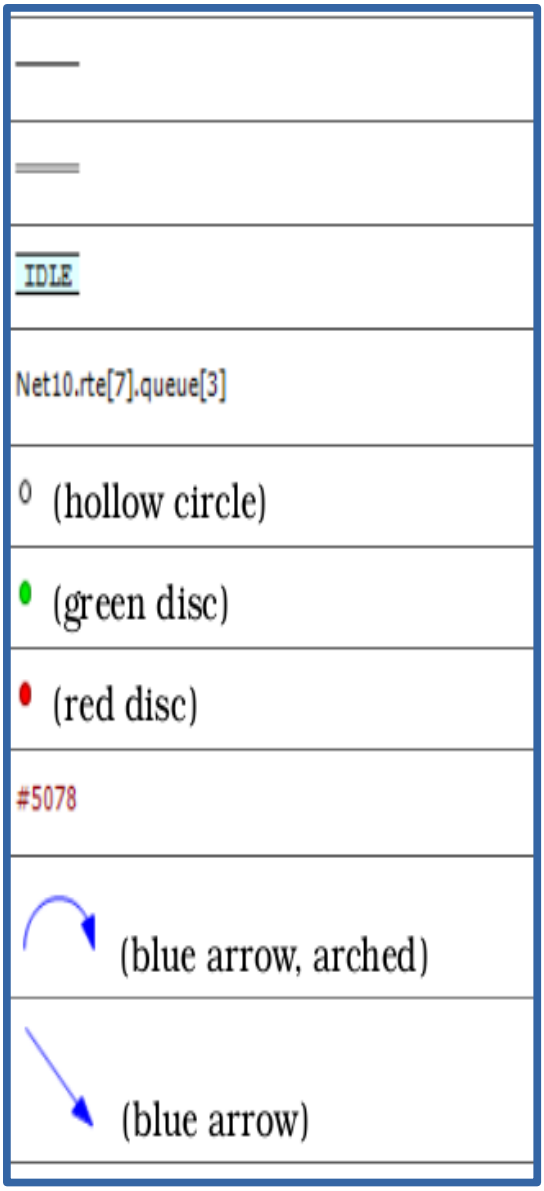
Sequence Charts

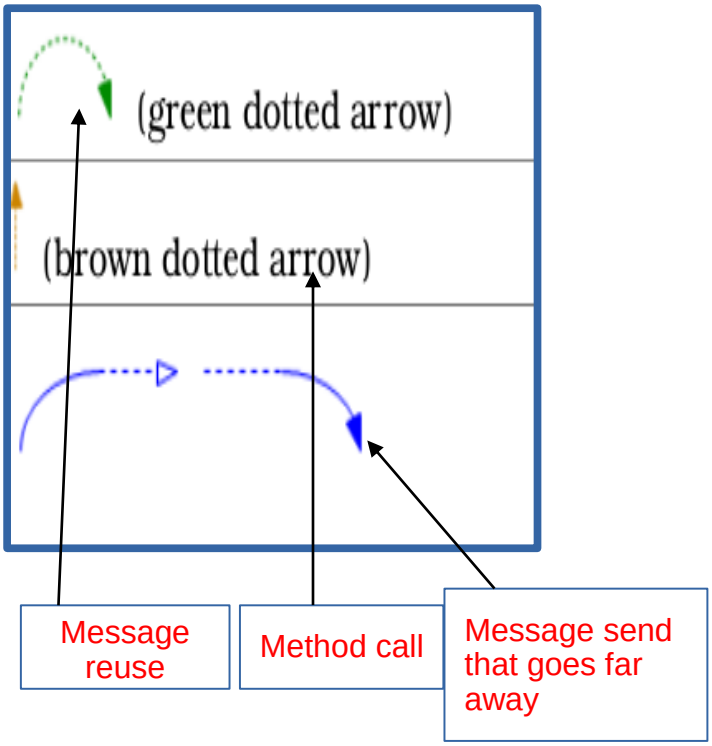
Understanding the Legend

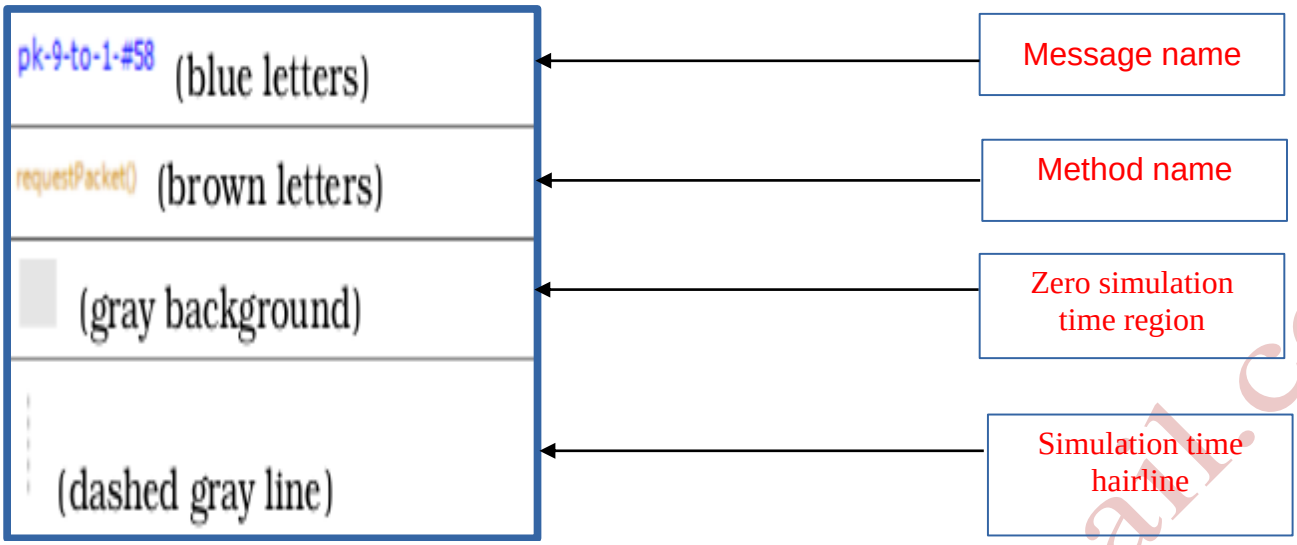


Sequence Charts

Understanding the Legend







END

TOPIC 28

Sequence Charts

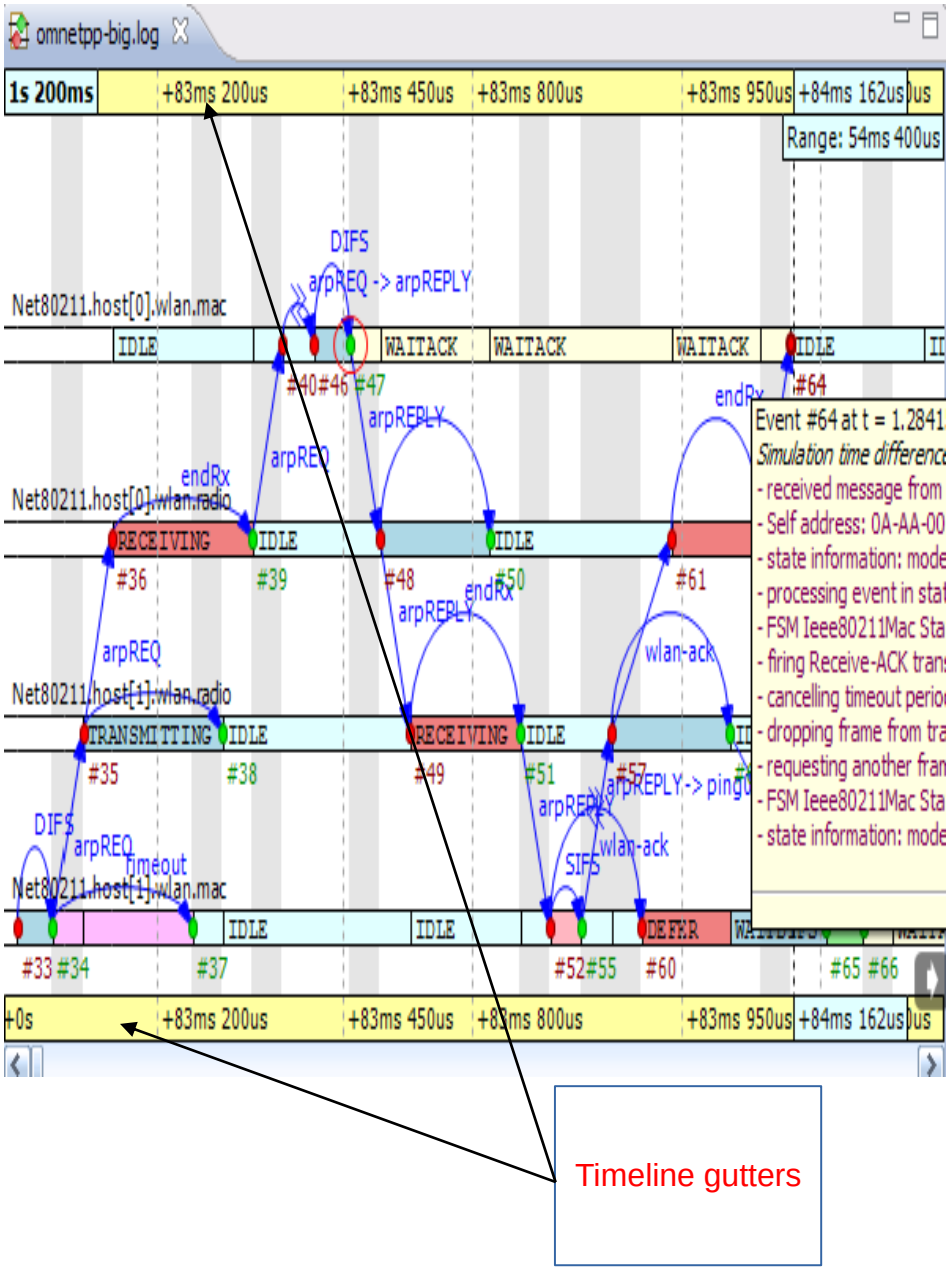
In this module

We shall understand

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

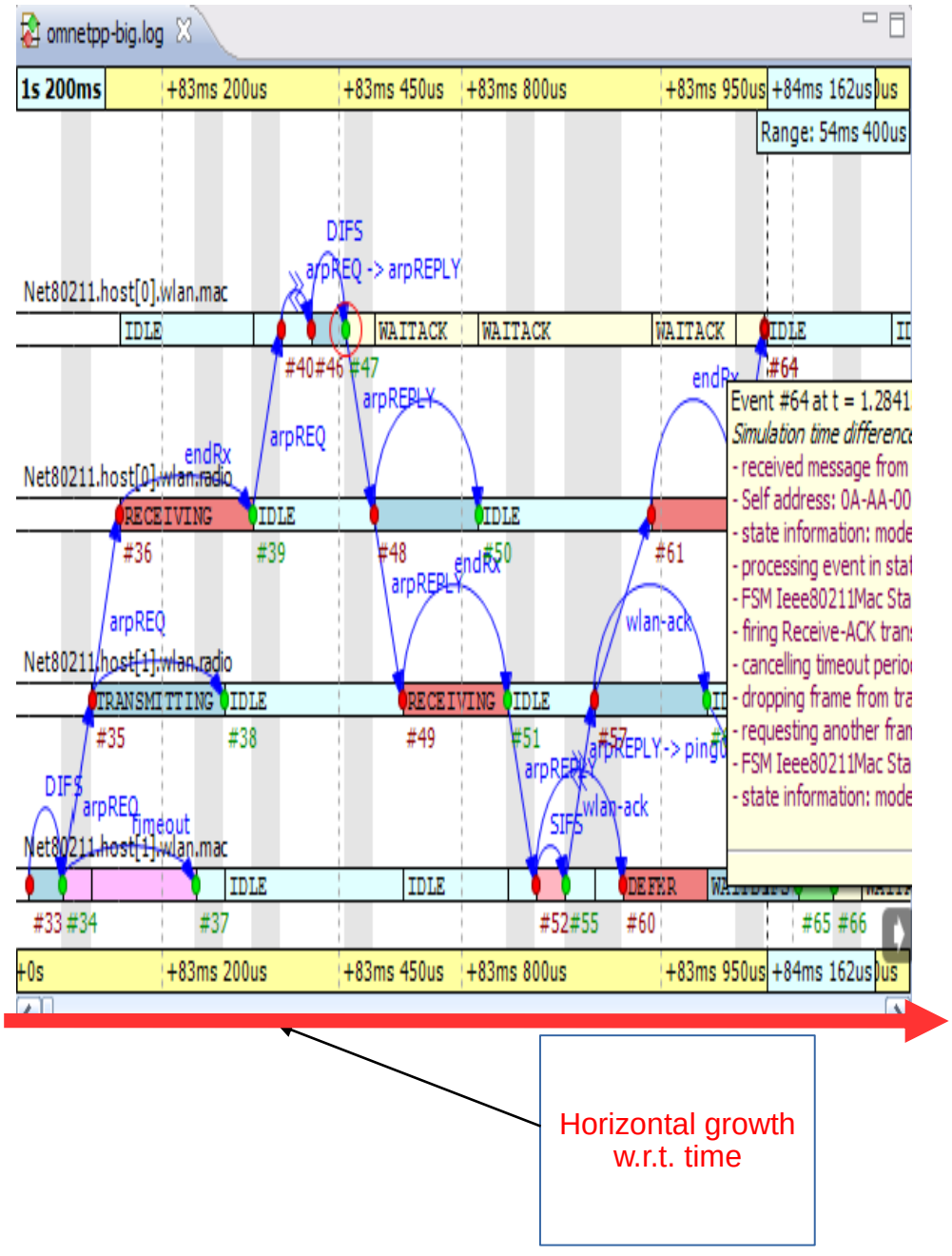
Sequence Charts

Parts of Sequence Charts



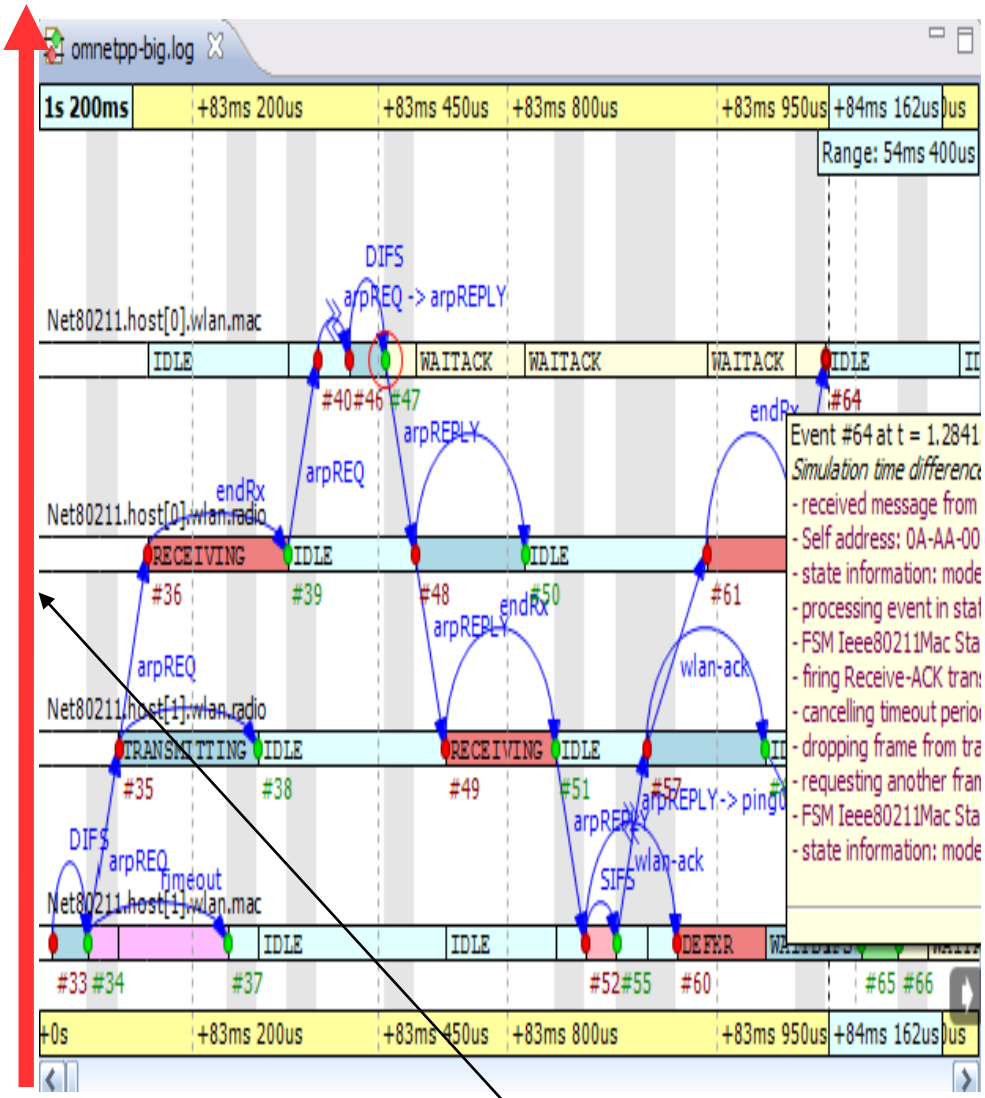
Sequence Charts

Parts of Sequence Charts



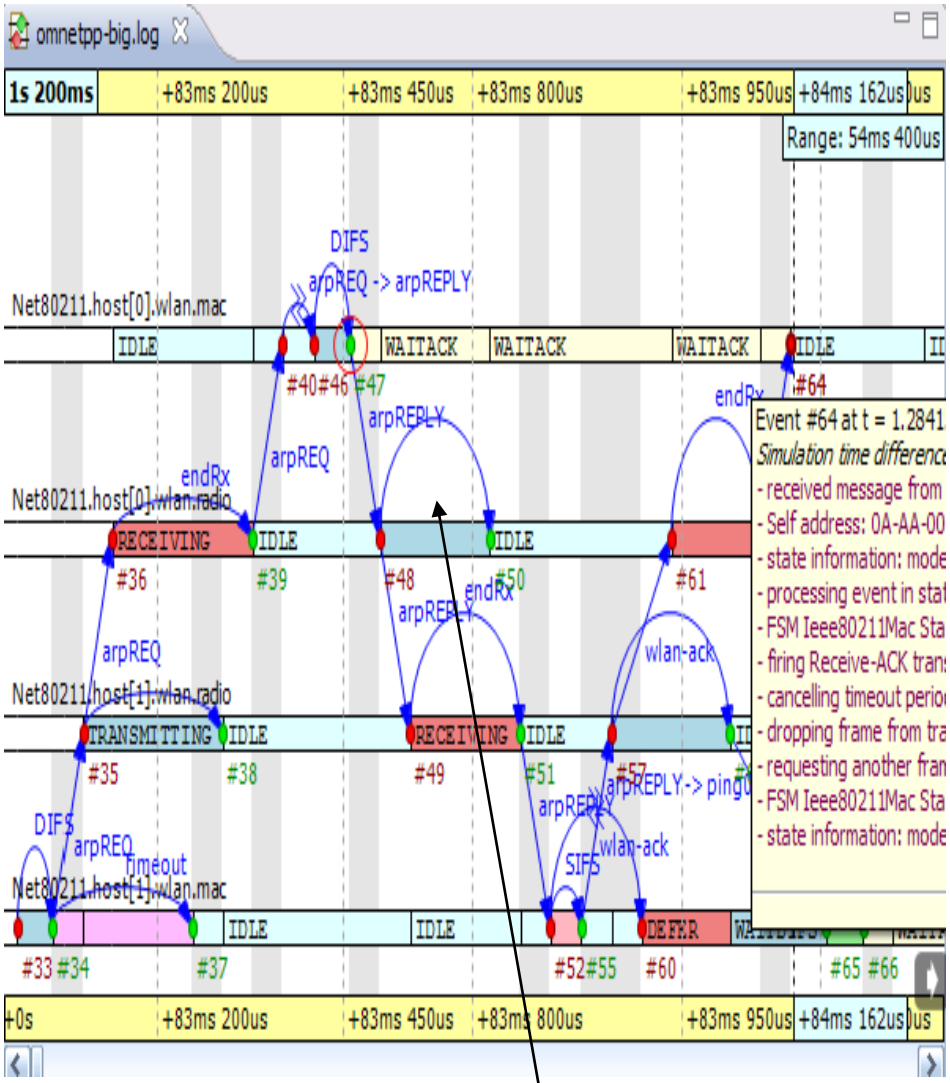
Sequence Charts

Parts of Sequence Charts



Sequence Charts

Parts of Sequence Charts



Modules, Events, Message Sends

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)

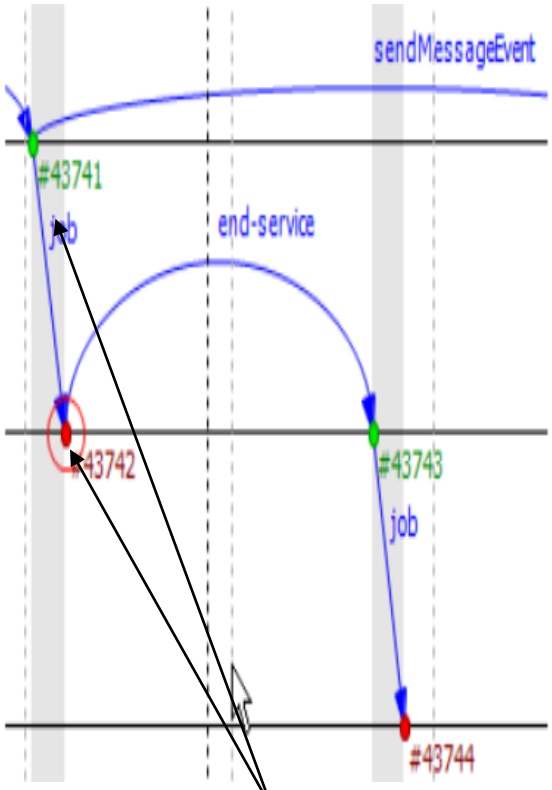
- **Step**: distance between subsequent events is same
- **Nonlinear**: distance between subsequent events is nonlinear function of simulation time between them

Interpreting Sequence Charts

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

Sequence Charts

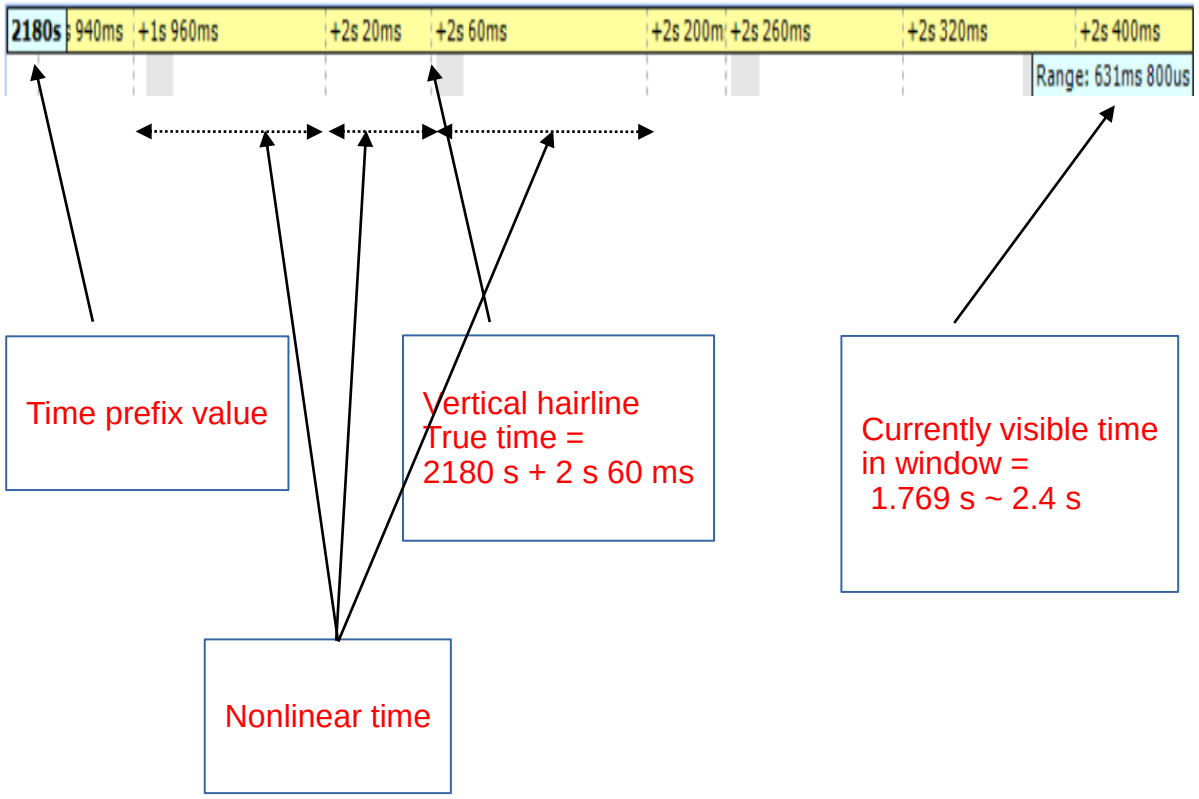
Zero Simulation Time Regions



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

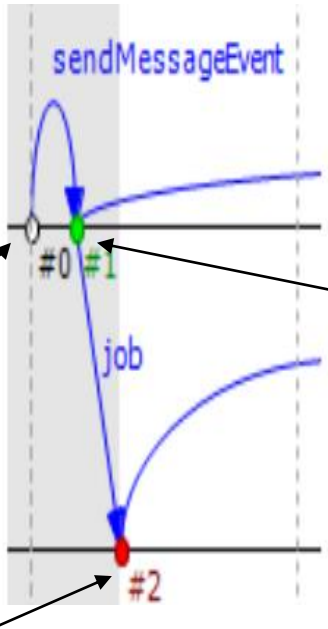
Sequence Charts

Gutter



Sequence Charts

Events Processing



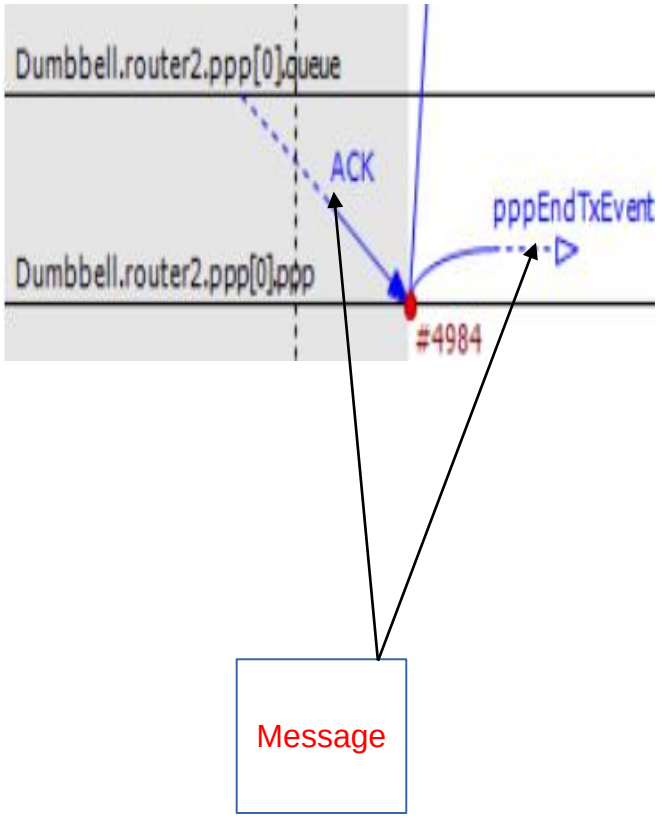
Initialization Message proc

Self message proc

Receiving Message proc

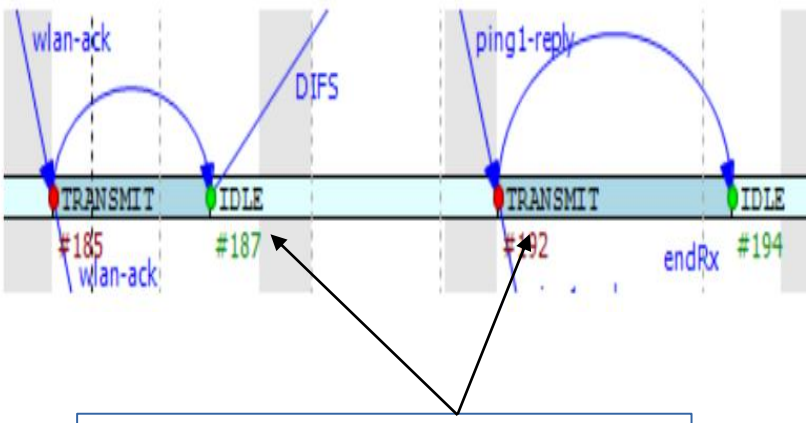
Sequence Charts

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

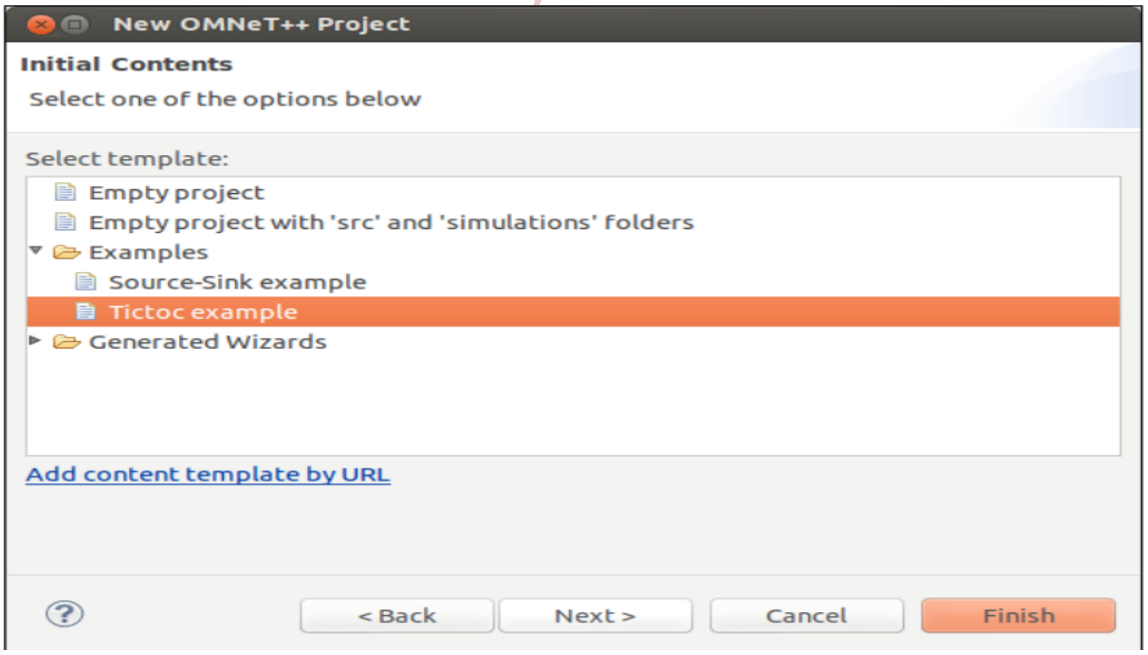
TicToc with 2-nodes

- 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

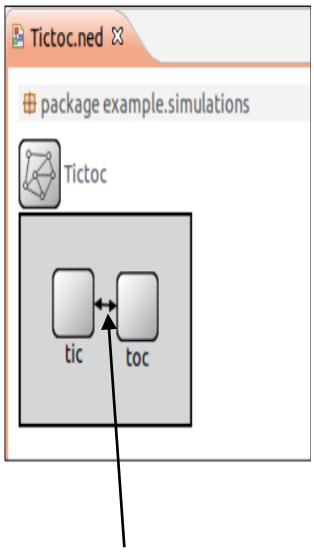
Tictoc.ned

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

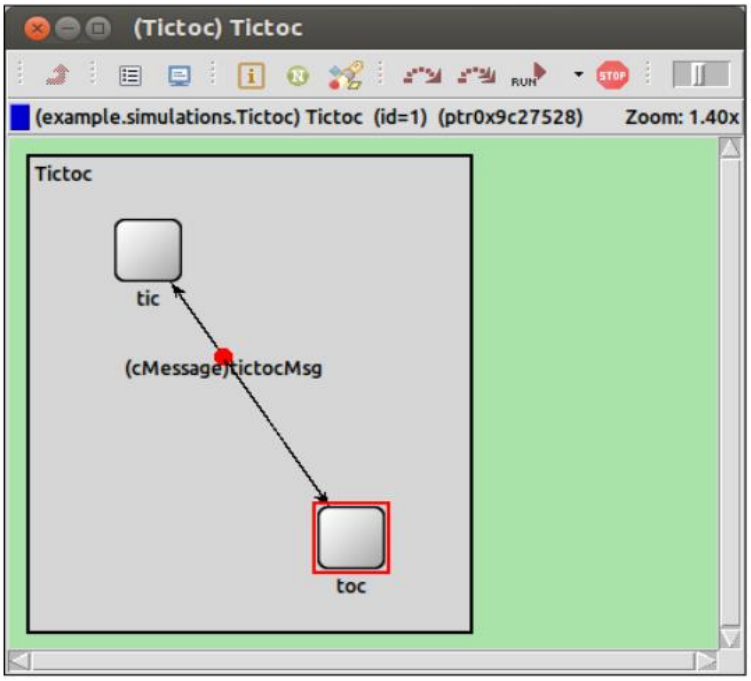
HandleMessage method

Echo back

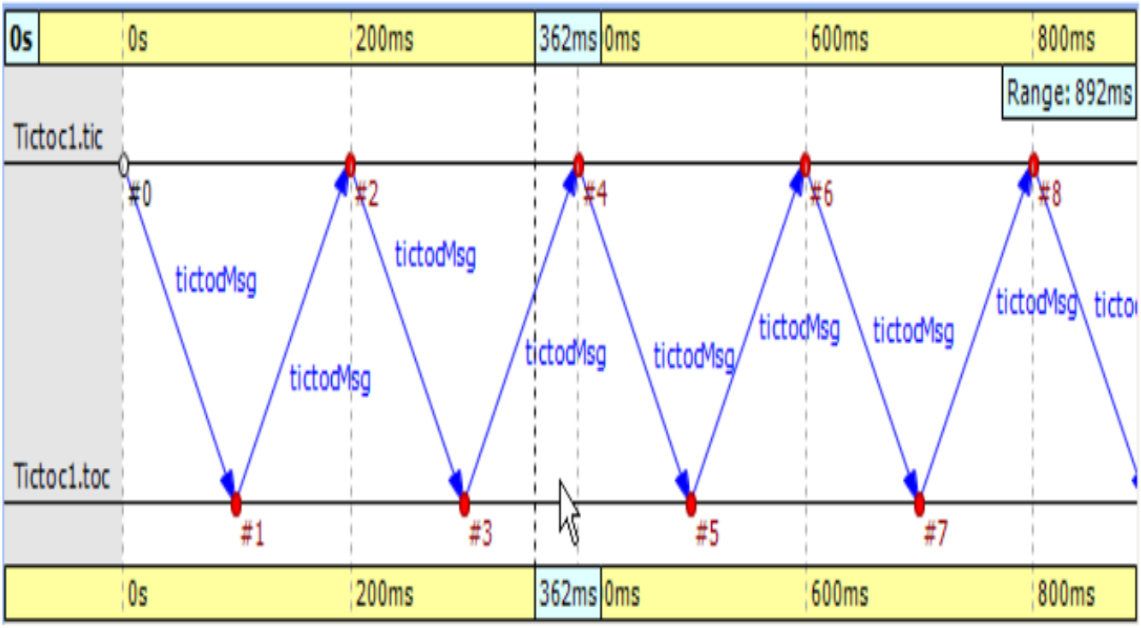
```
[General]
network = Tictoc
cpu-time-limit = 60s
#debug-on-errors = true
**.tic.sendInitialMessage = true
```

Starts the activity

Compiling & Running on Tkenv



Recap: TicToc Tutorial



TOPIC 30

Extending TicToc

In this module

We shall extend TicToc tutorial

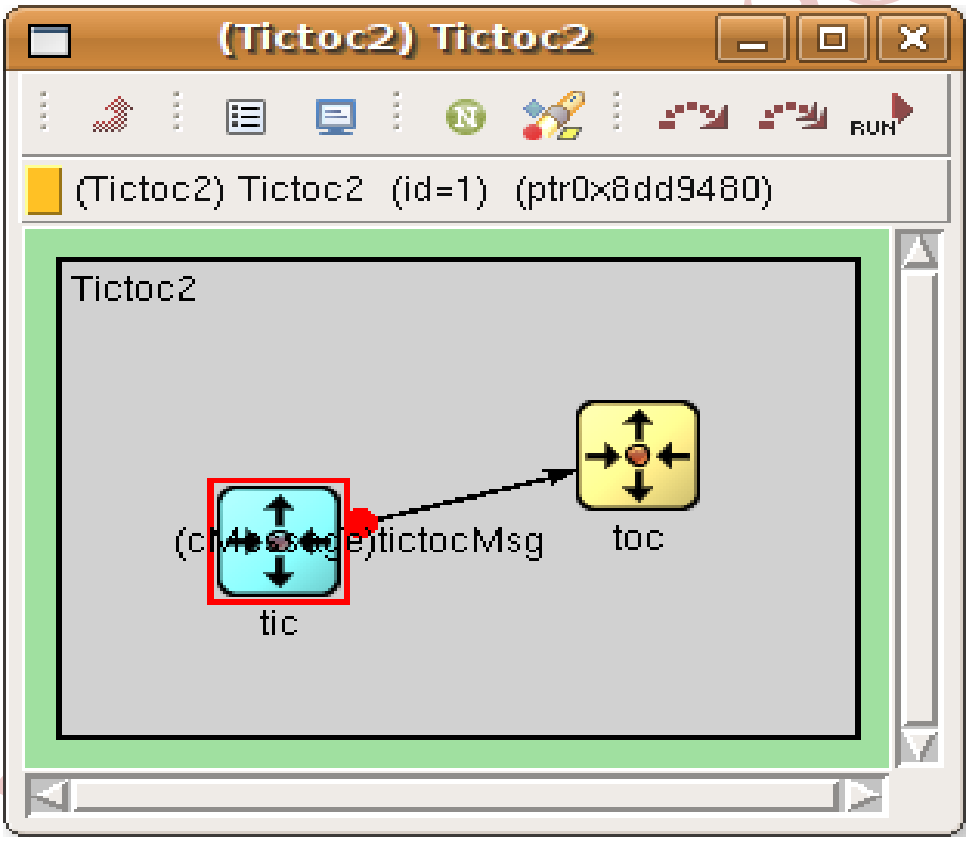
- Refine graphics, & add debugging output
- Add state variables
- Add parameters
- Model processing delay
- And more!

Refine graphics &

- Tictoc2.ned

Add debugging output

- Txc2.cc



Extending TicToc

2. Refine graphics & add debugging output

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

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)

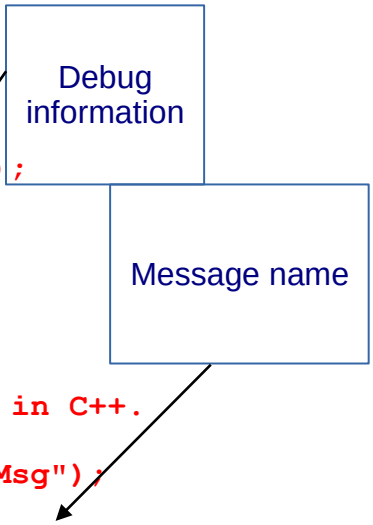
```
// Make the two module look a bit different with
colorization effect.
// Use cyan for `tic', and yellow for `toc'.
//
network Tictoc2
{
    submodules:
        tic: Txc2 {
            parameters:
                @display("i=,cyan"); // do not change the
icon (first arg of i=) just colorize it
  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

Change color

Extending TicToc

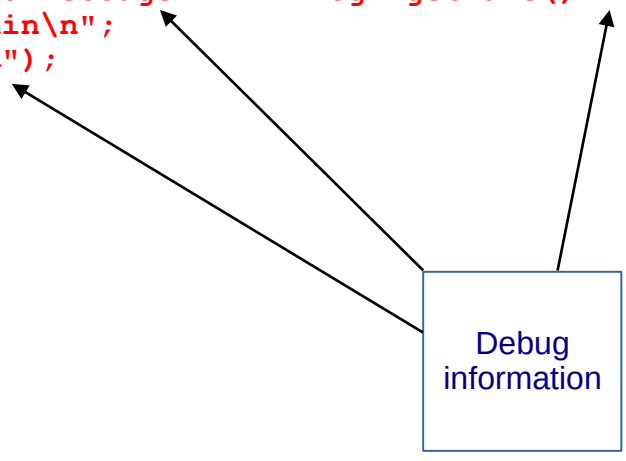
Txc2.cc (1 of 2)

```
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");
    }
}
```

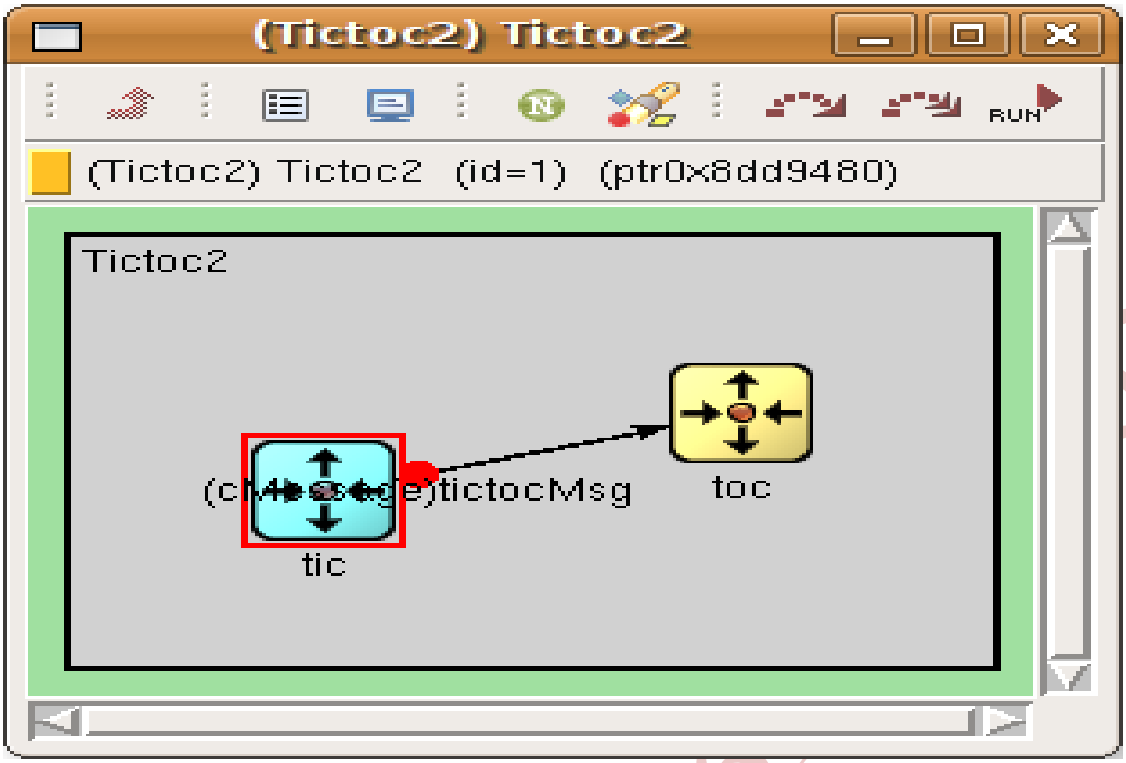


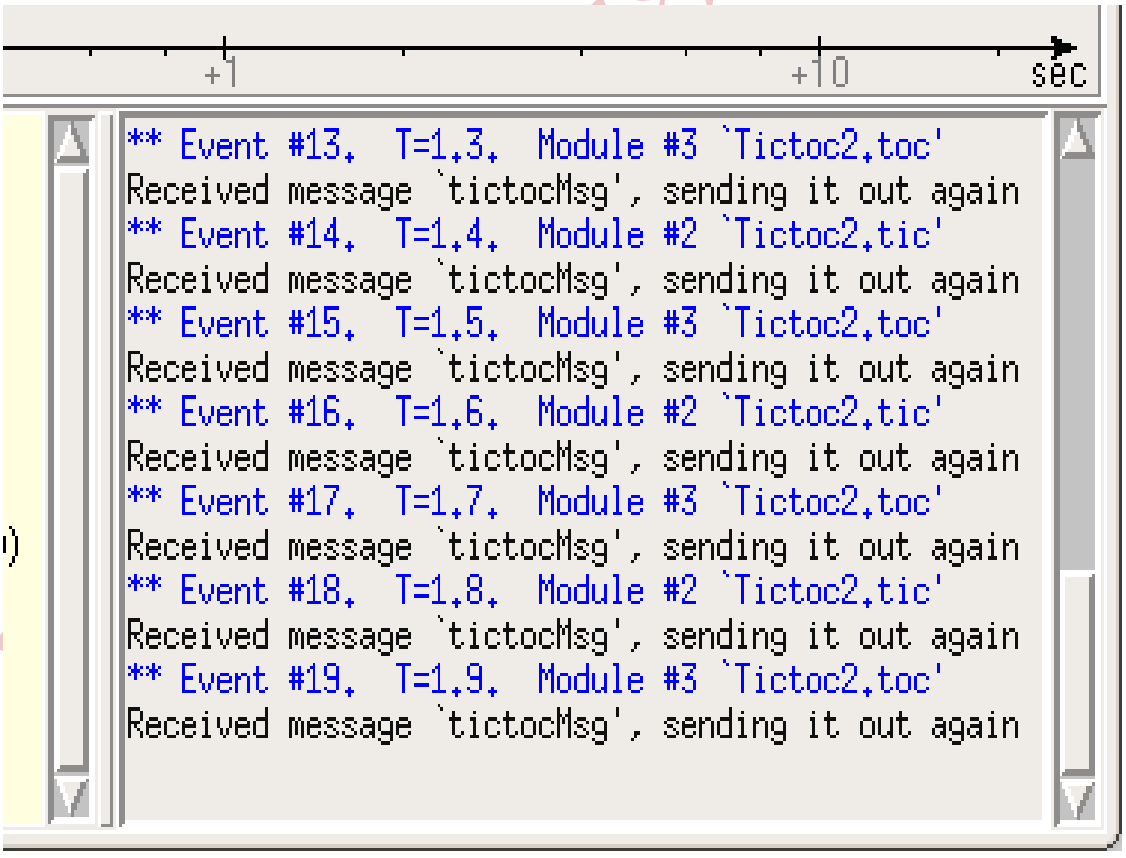
Txc2.cc (2 of 2)

```
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);
```

Counter

Let you examine the variable under Tkenv.

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

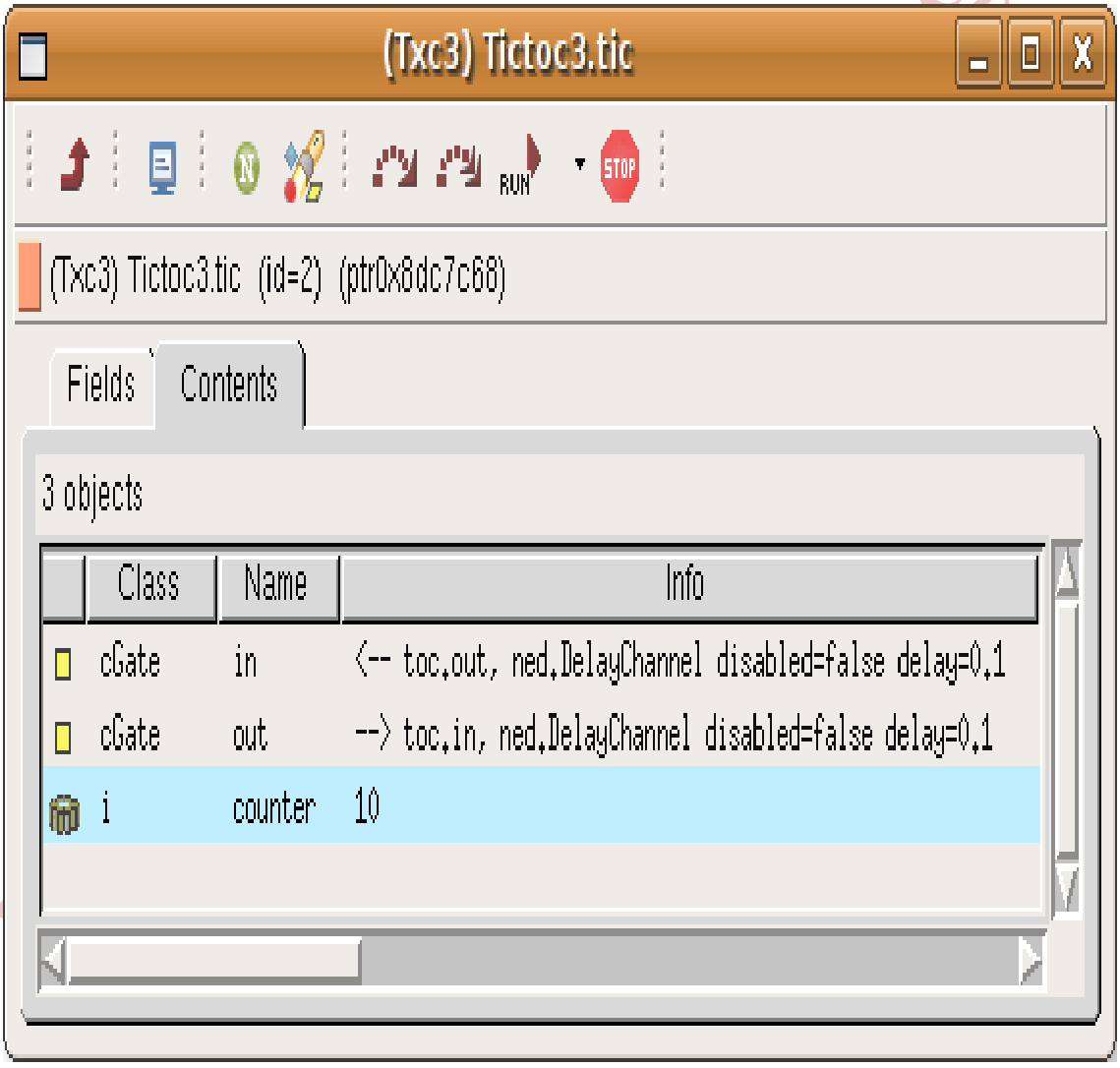
If counter is zero, delete message

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



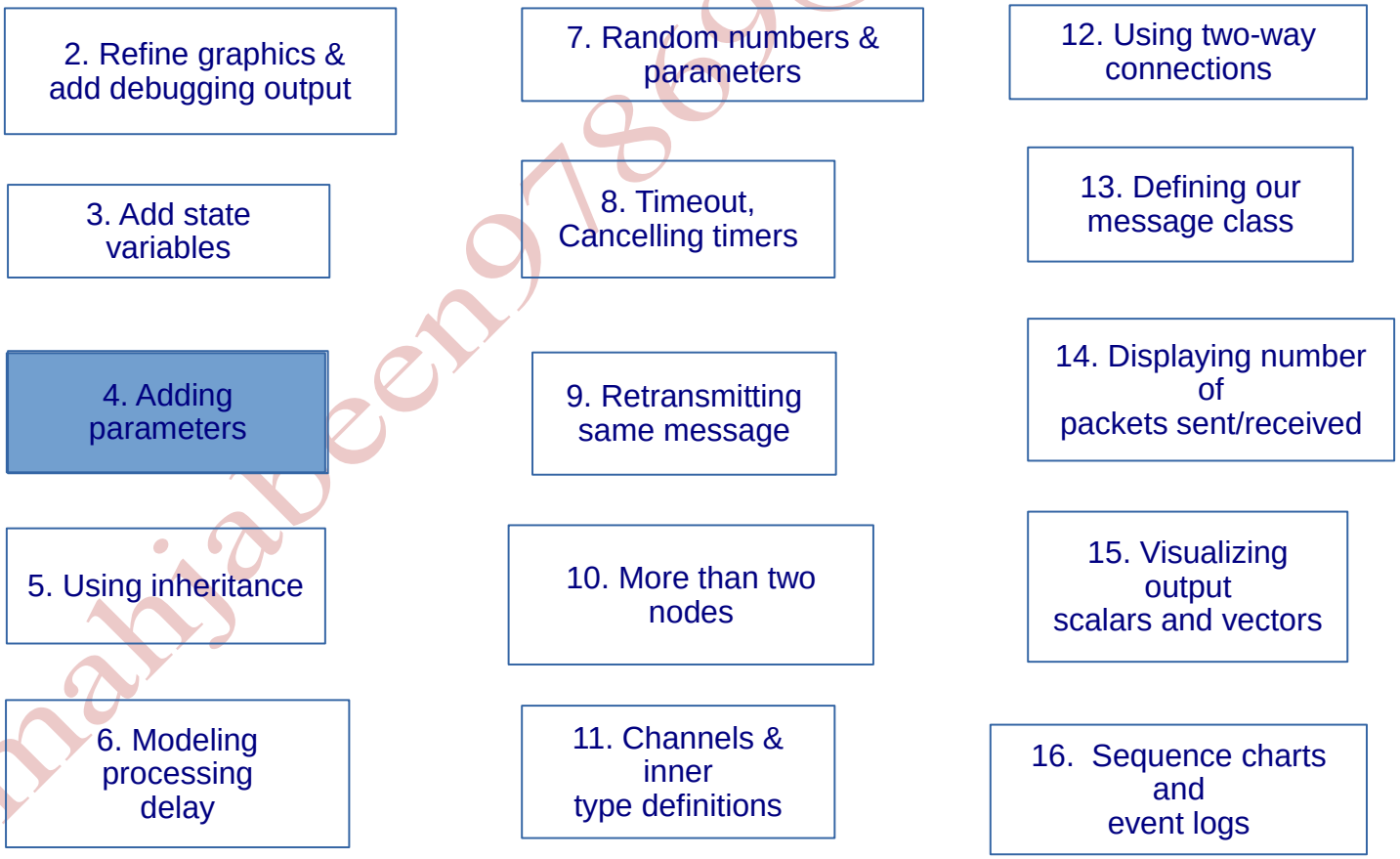
TOPIC 32

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)

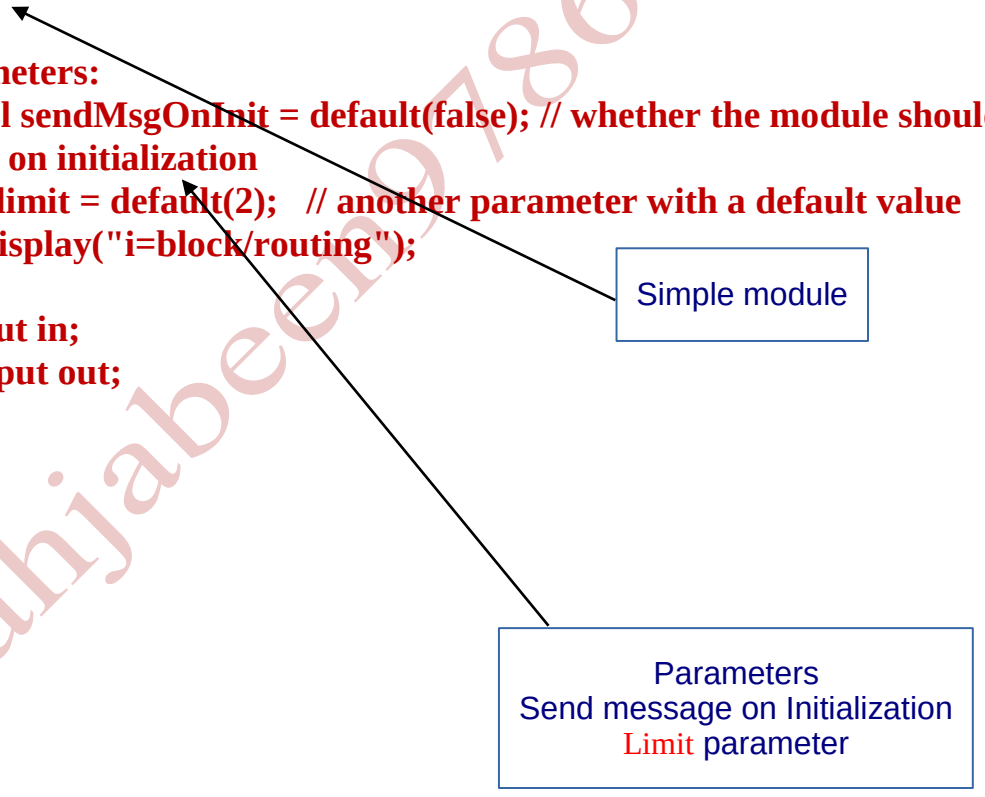
- **Add add input parameters to the simulations**
  - **Count = 10 now into a parameter that the user can define**
- **tictoc4.ned**
- **Txc4.cc**
- **Omnet.ini**

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

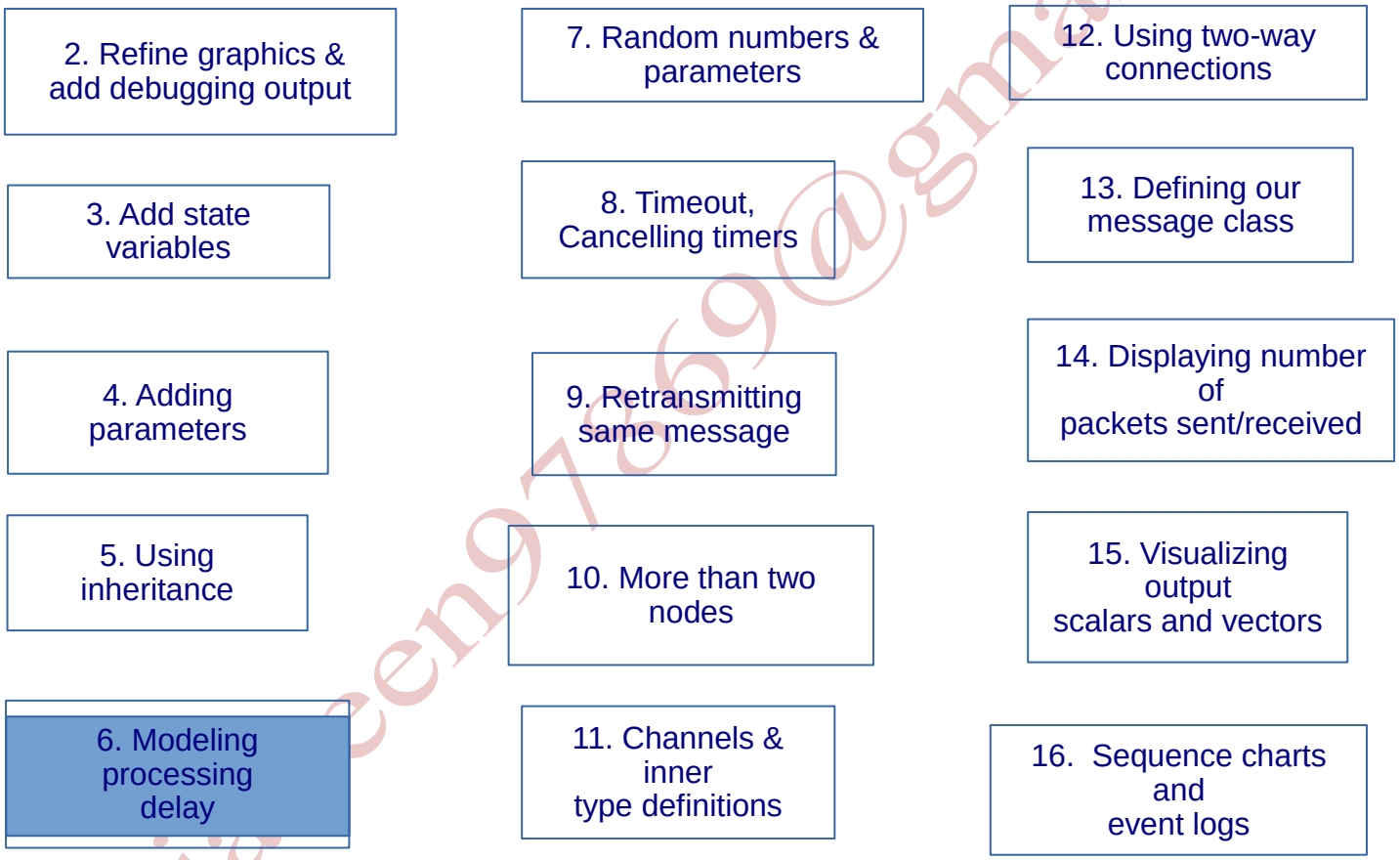
TOPIC 34

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!



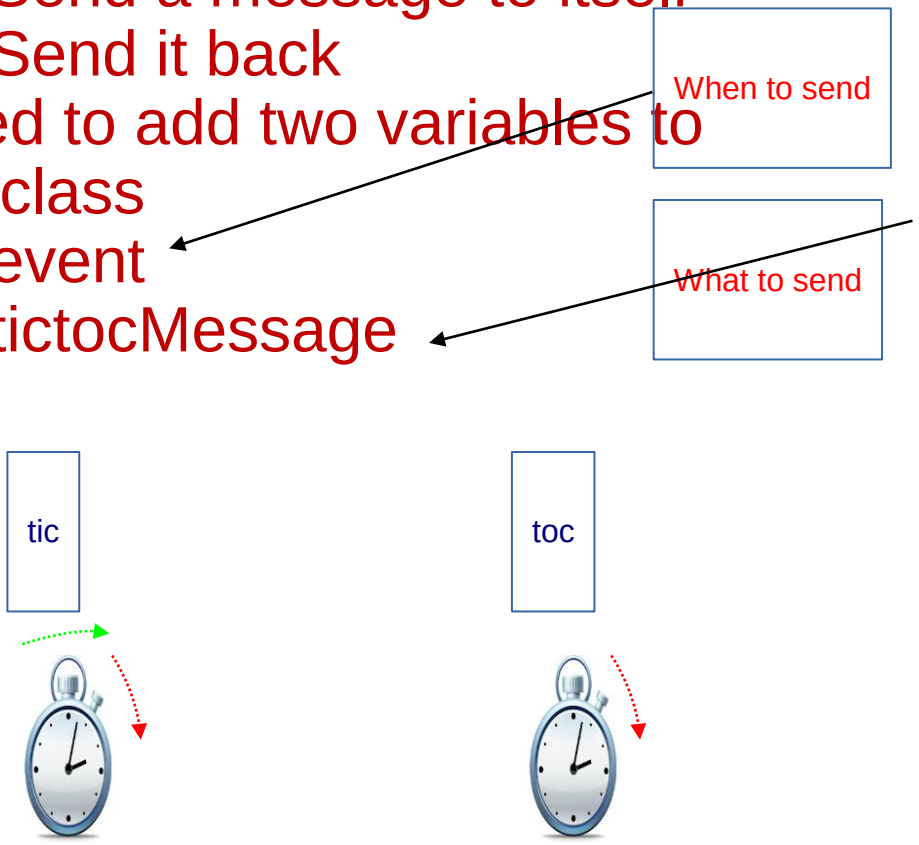
Modeling processing delay

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

Extending TicToc

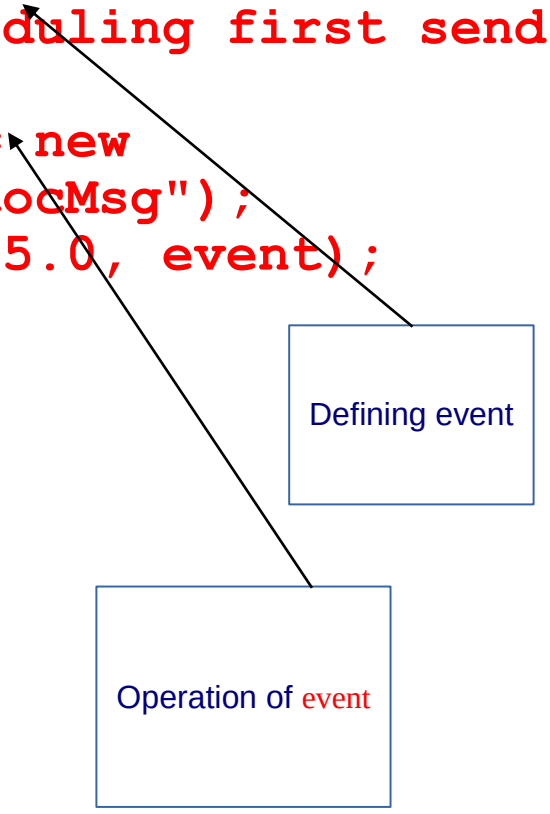
Strategy

- 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



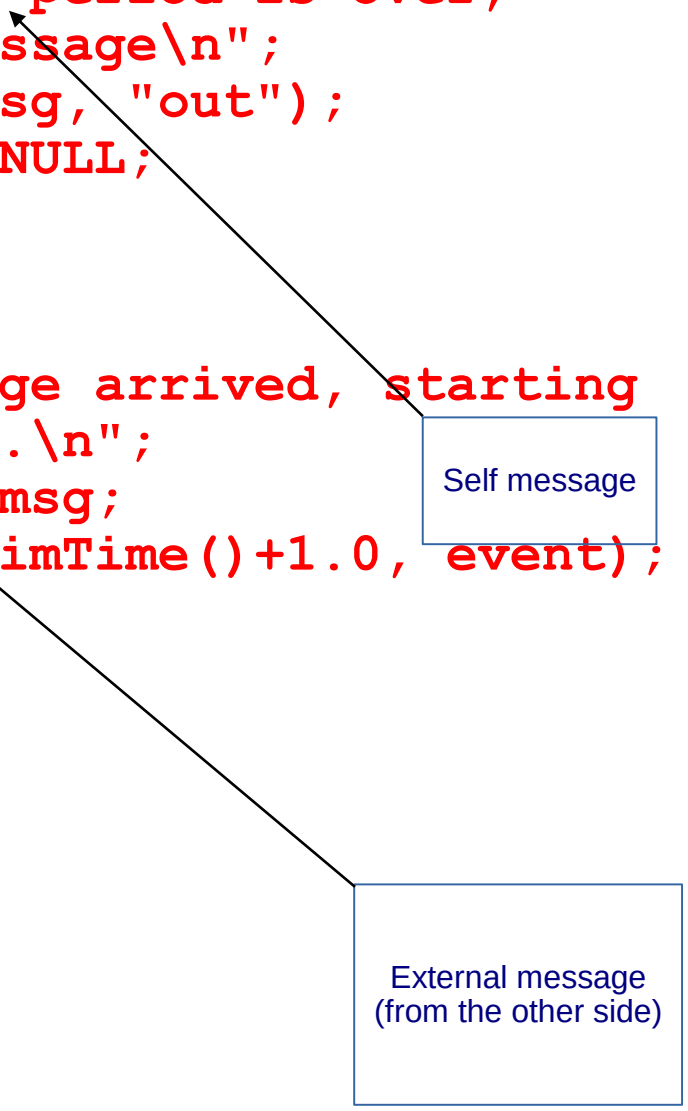
tx6.cc (1 of 2)

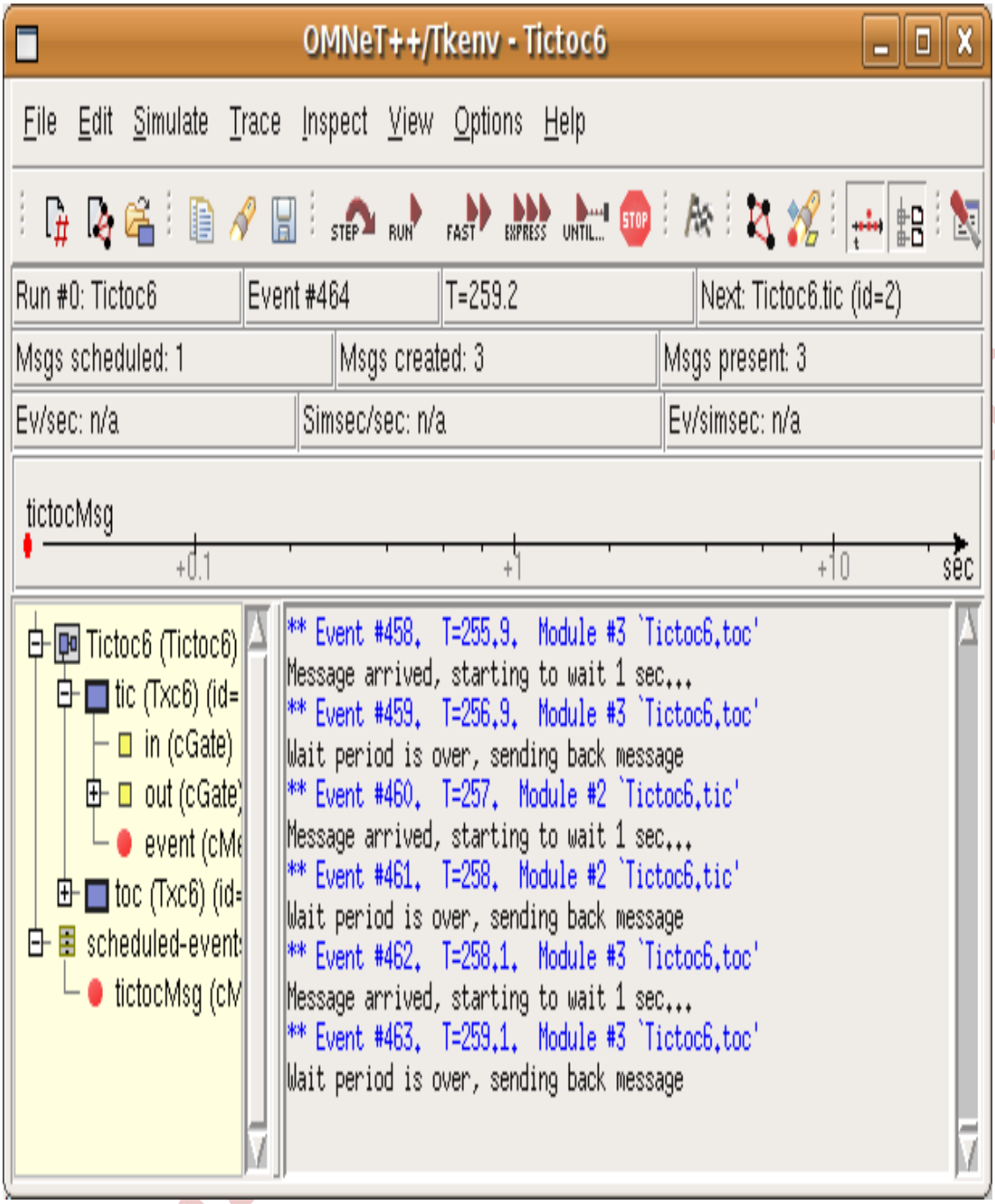
```
void Txc6::initialize()
{
   // Create the event object
(ordinary message) for  //timing
   event = new cMessage("event");
   tictocMsg = NULL;
   if (strcmp("tic", getName()) == 0)
   EV << "Scheduling first send to
t=5.0s\n";
   tictocMsg = new
cMessage("tictocMsg");
   scheduleAt(5.0, event);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



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

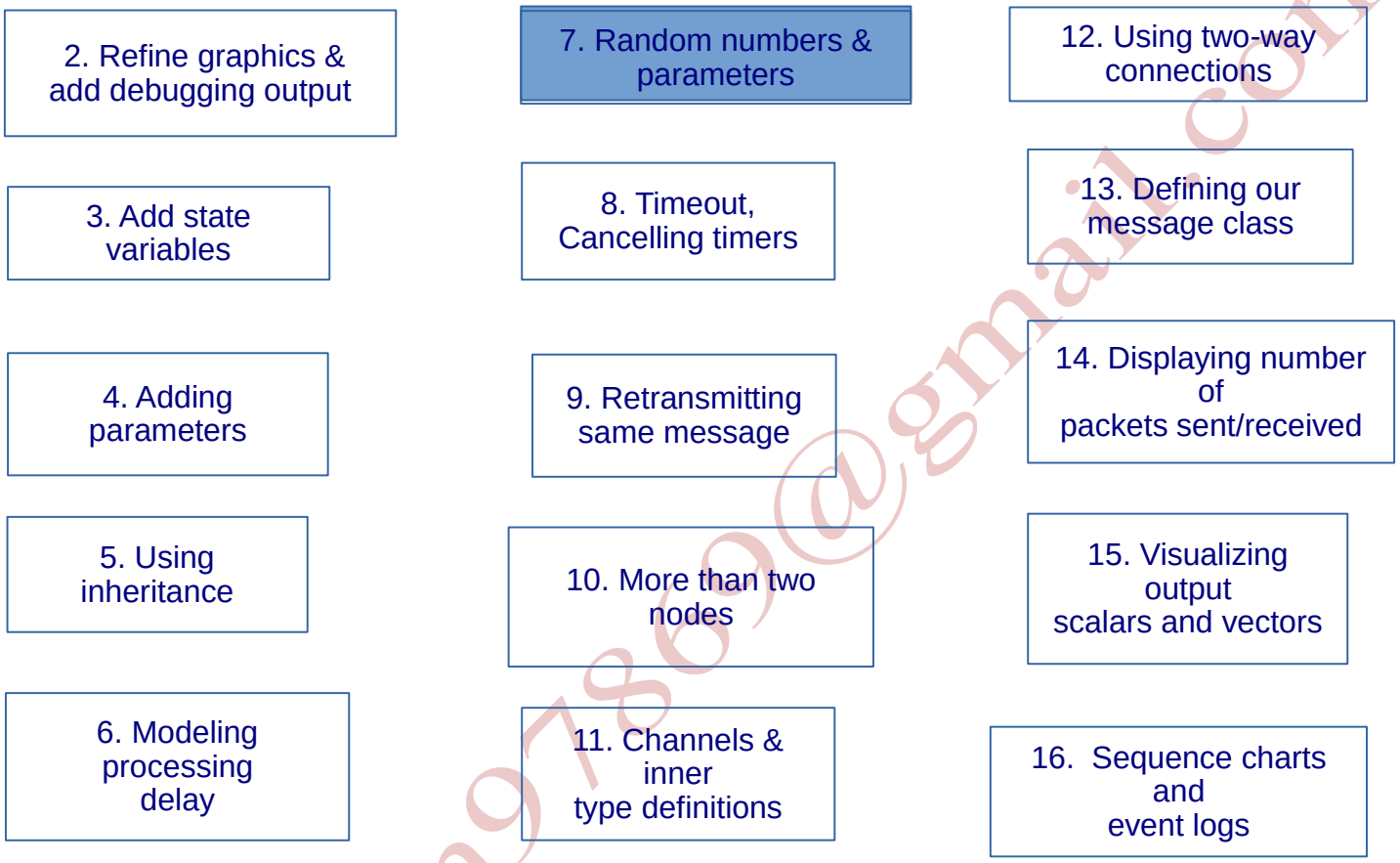
Extending TicToc

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)
```

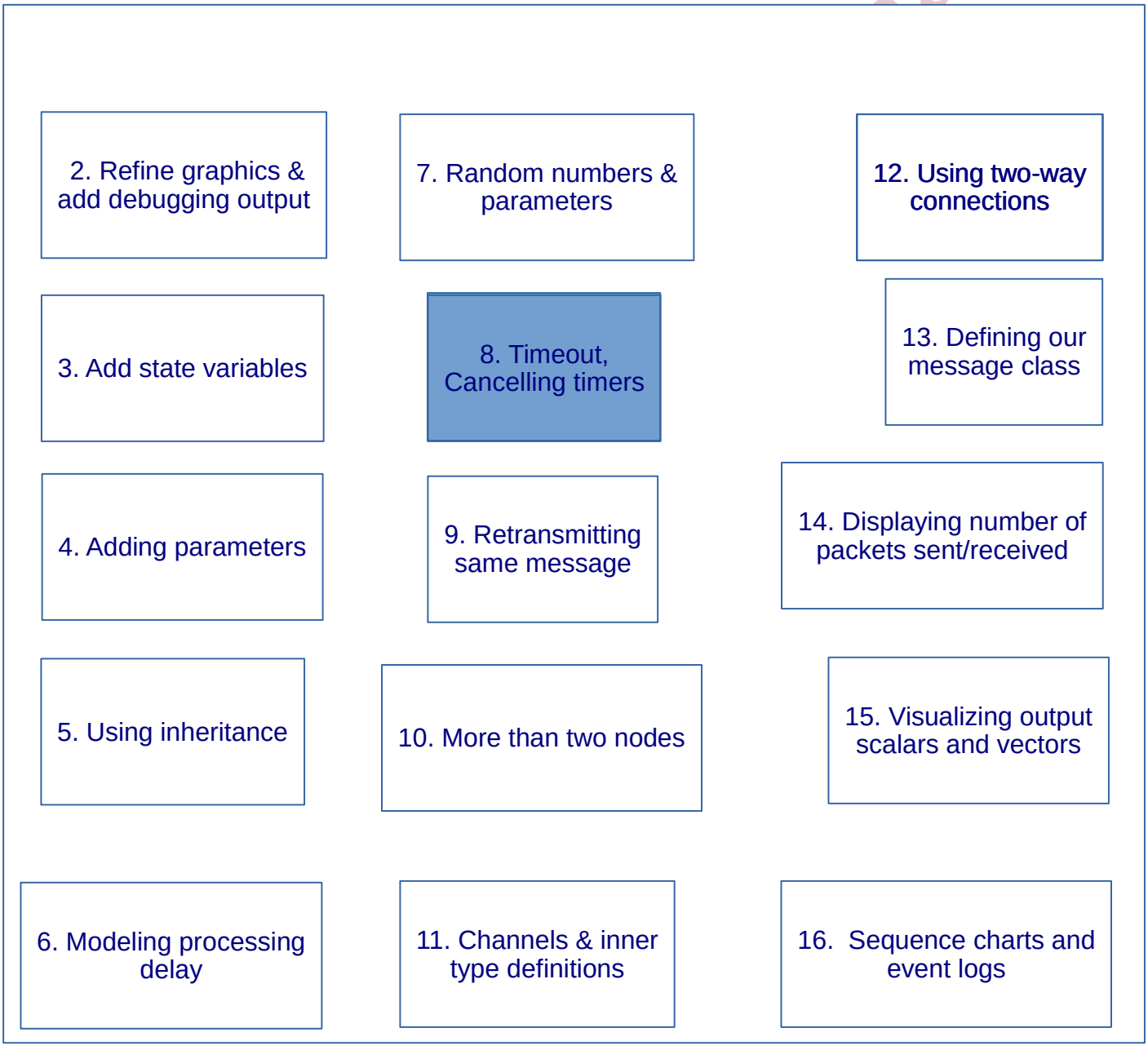
mean

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!

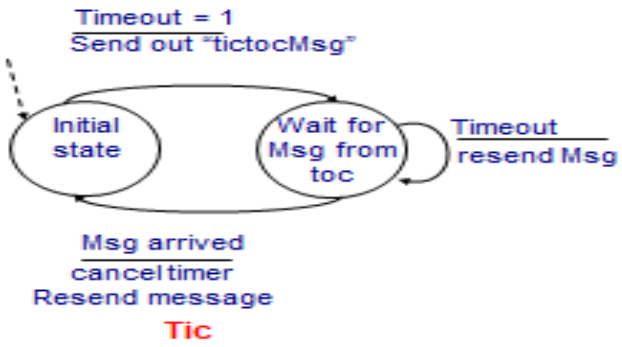


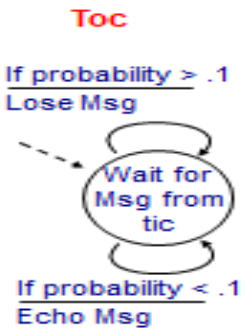
Timeout, cancelling timers

- **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

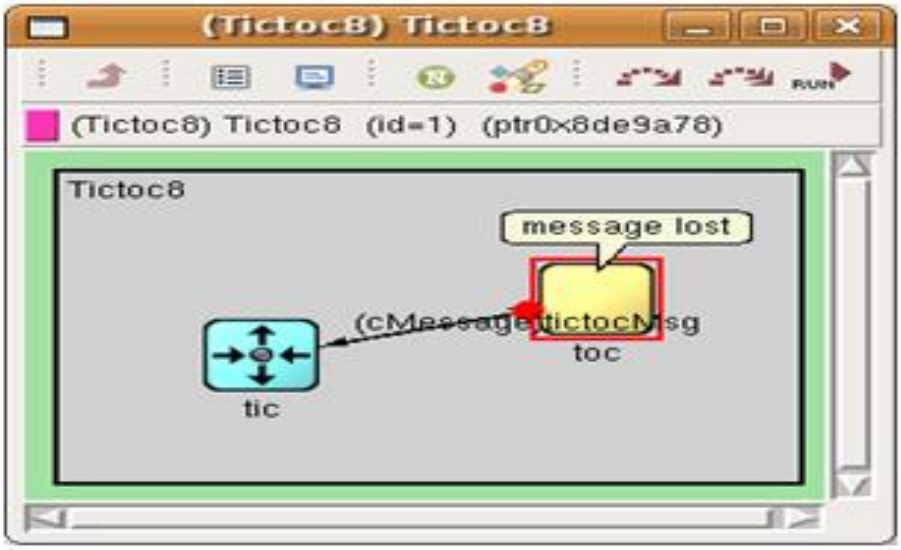
Extending TicToc

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

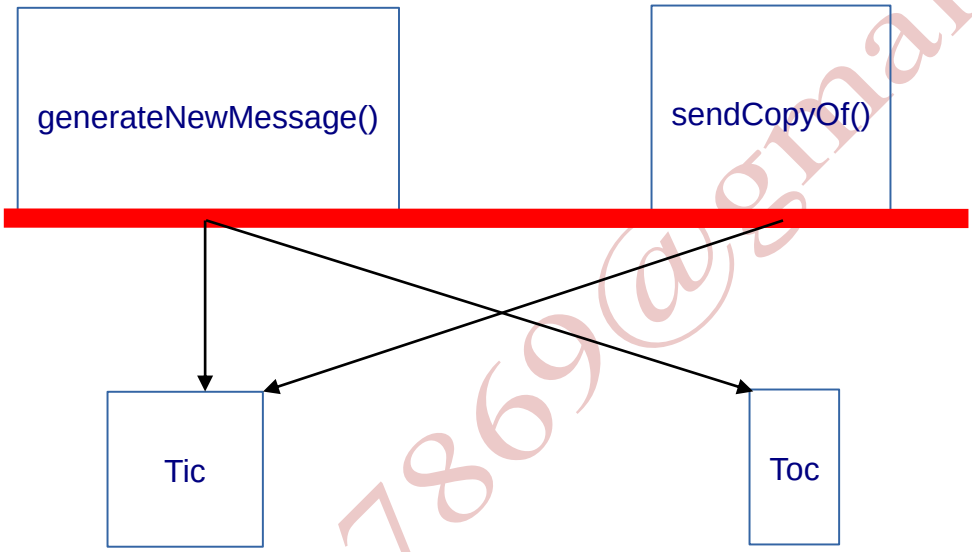
Retransmitting same message (2 of 2)

- **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

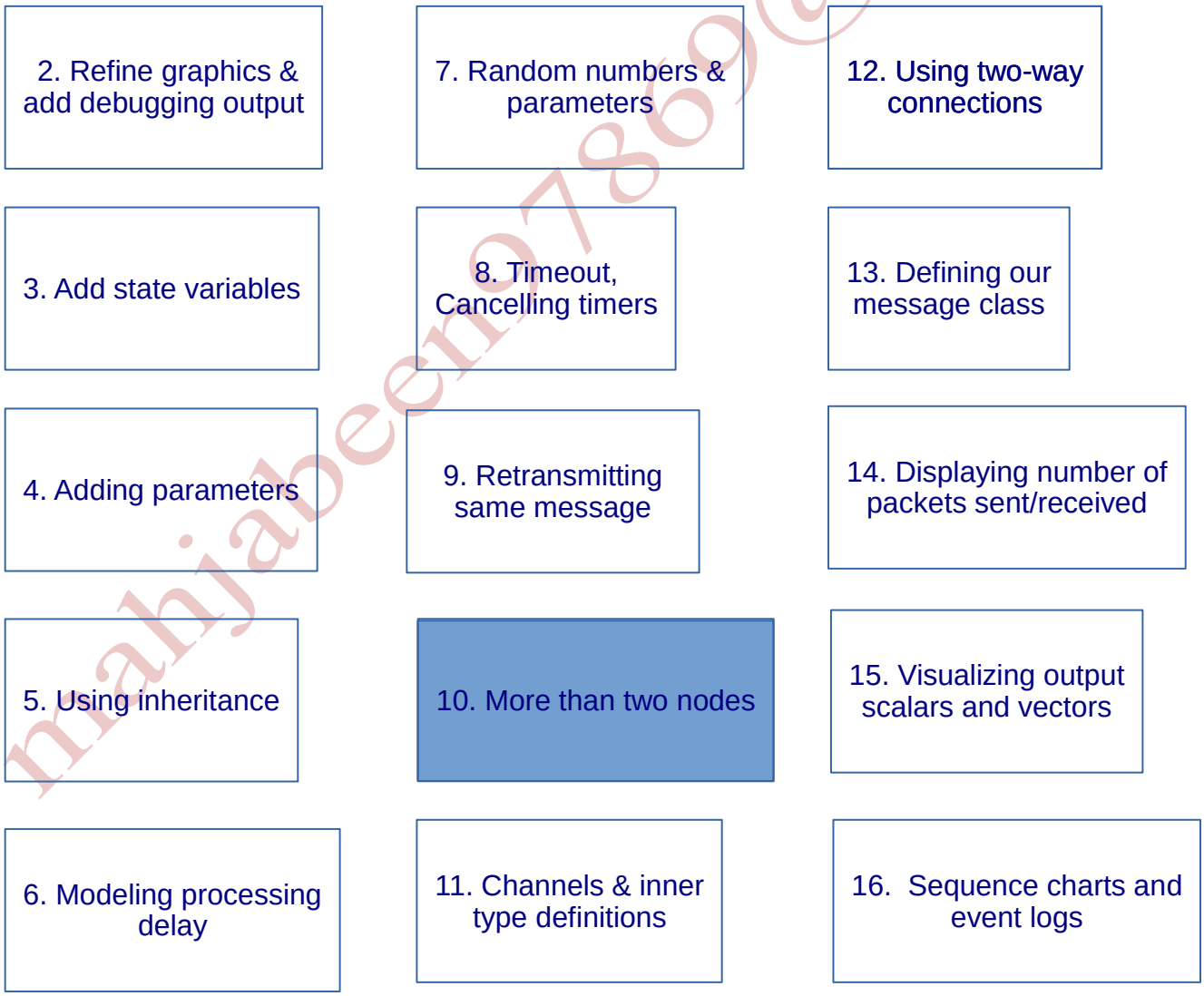
TOPIC 38

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



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!)

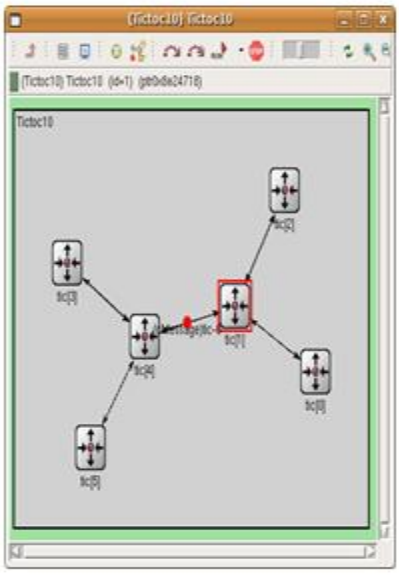
Extending TicToc

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

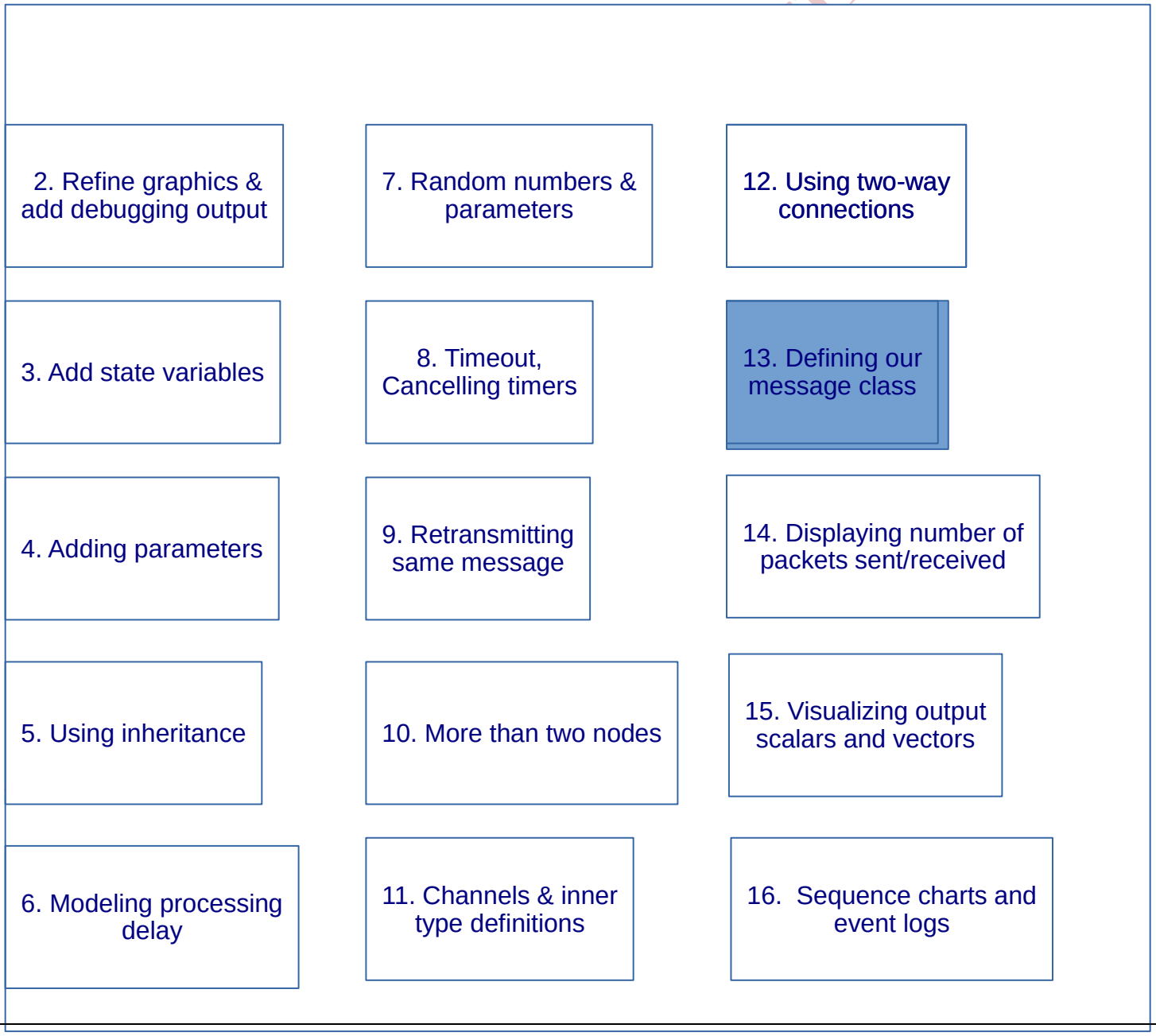
TOPIC 41

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

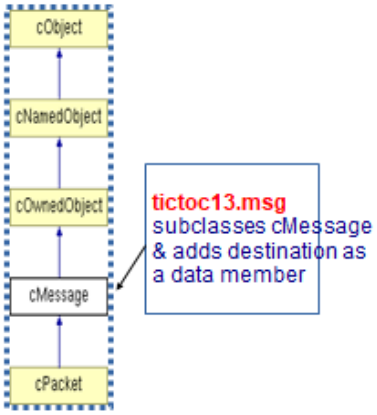


Defining our message class

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

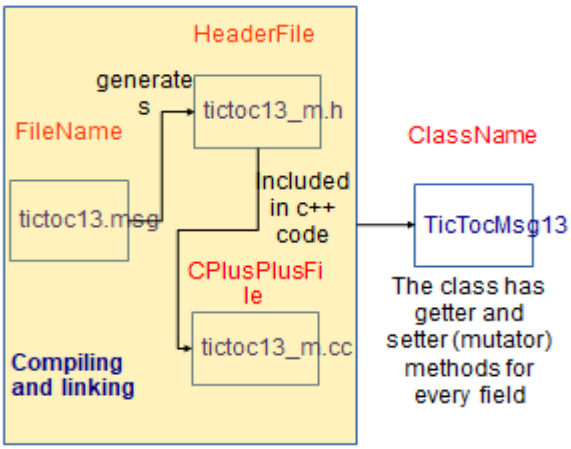
Extending TicToc

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

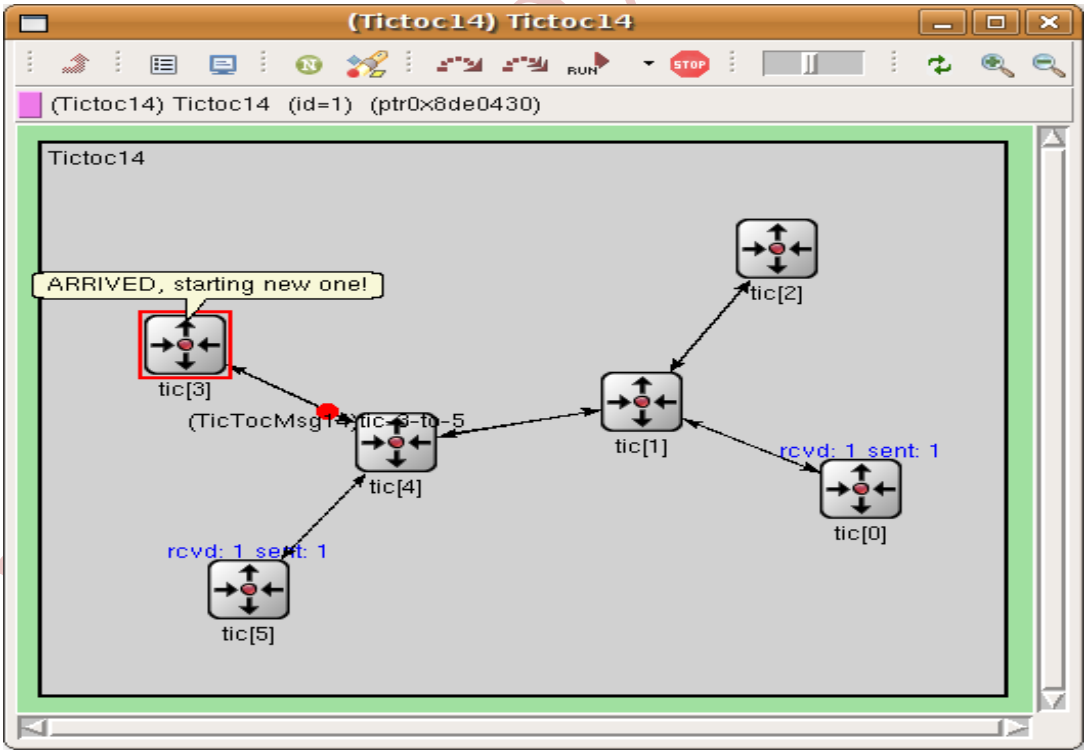
Destination becomes source now

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

Extending TicToc

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

Extending TicToc

Txc14.cc (2 of 3)

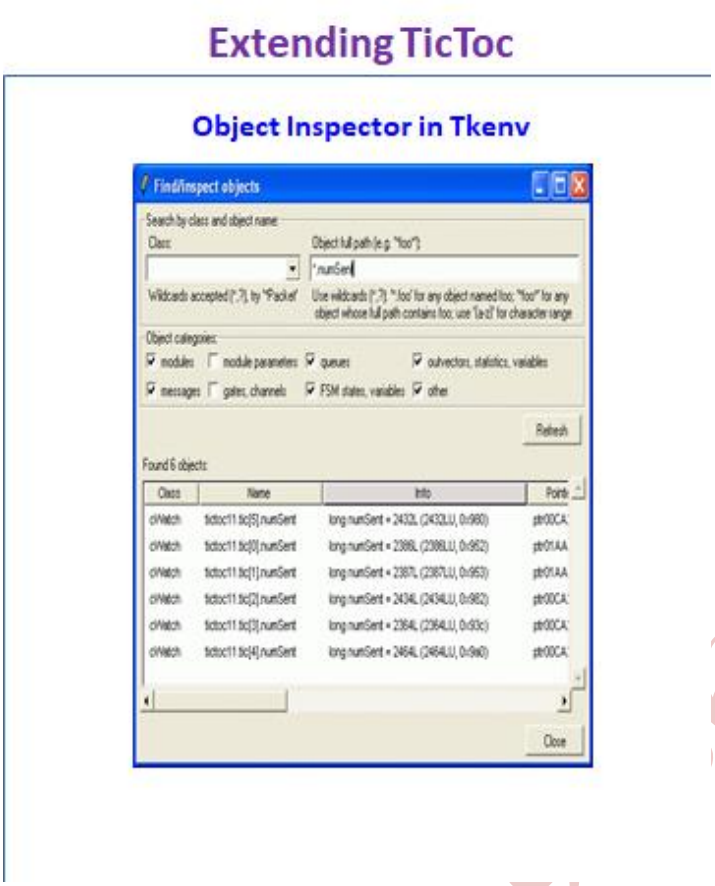
```
void Txc14::handleMessage(cMessage *msg)
{
 if (ttmsg->getDestination()==getIndex())
  {
   if (ev.isGUI())
    {
  updateDisplay();
    }
  }
}
```

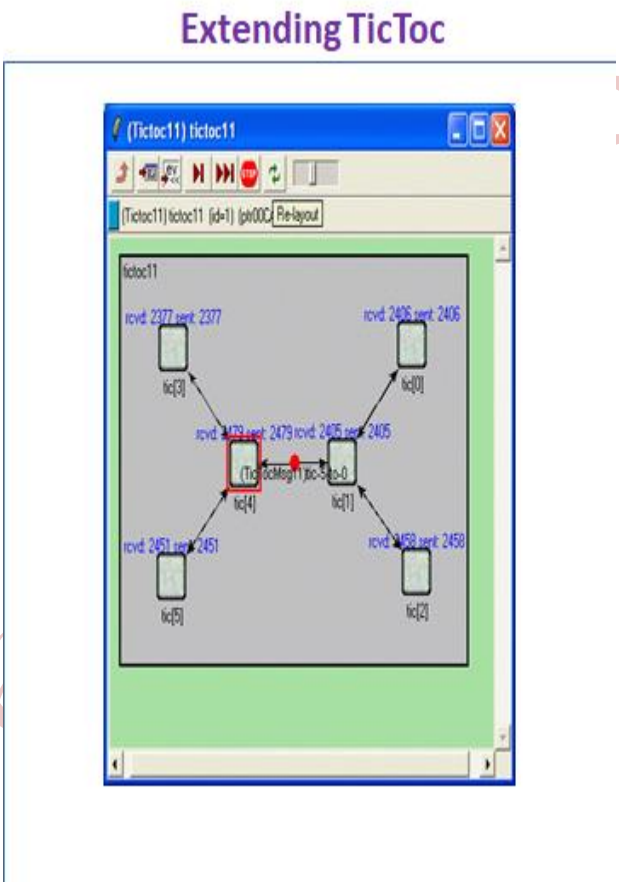
info appears above module icons

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

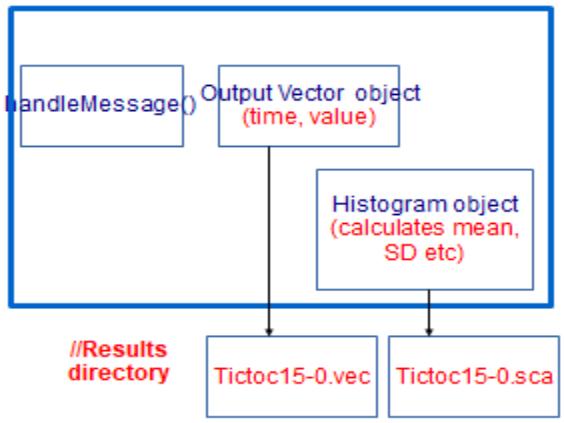
- txc15.cc
- tictoc15.msg
- omnetpp.ini

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

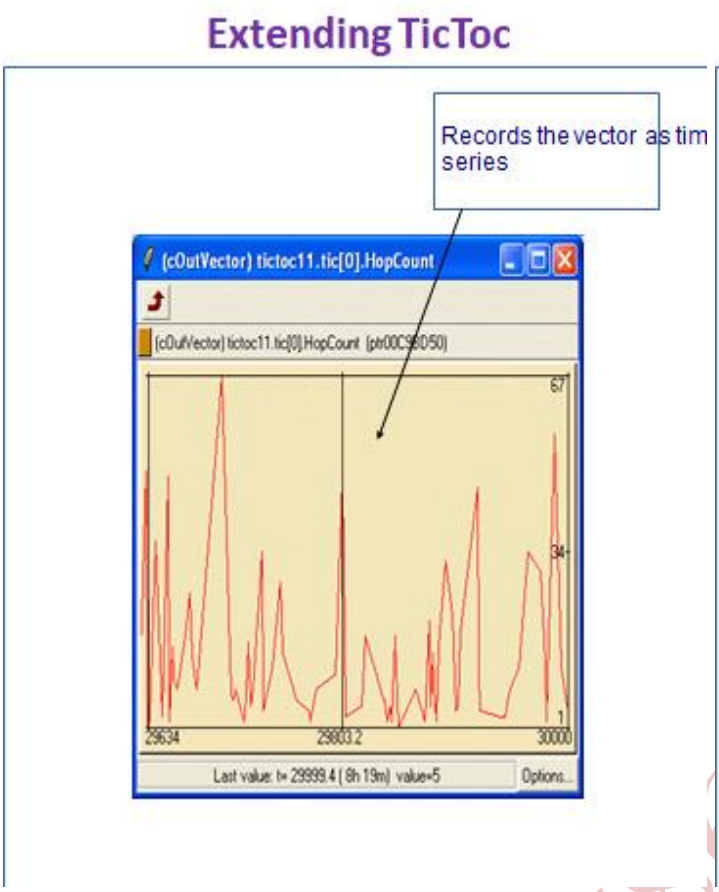
Extending TicToc

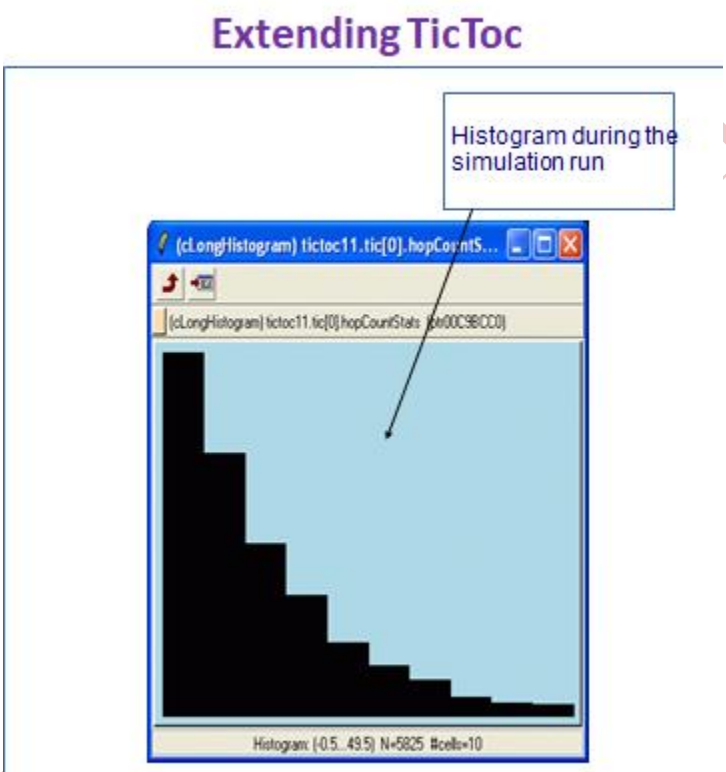
Txc15.cc (3 of 3)

```
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 cour





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

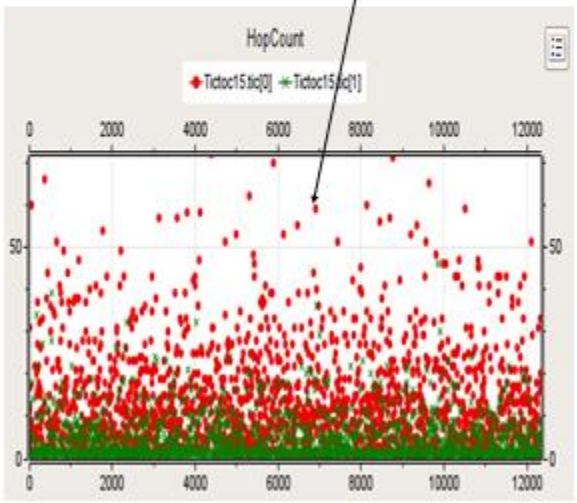
14. Displaying number of packets sent/received

15. Visualizing output scalars and vectors

16. Sequence charts and event logs

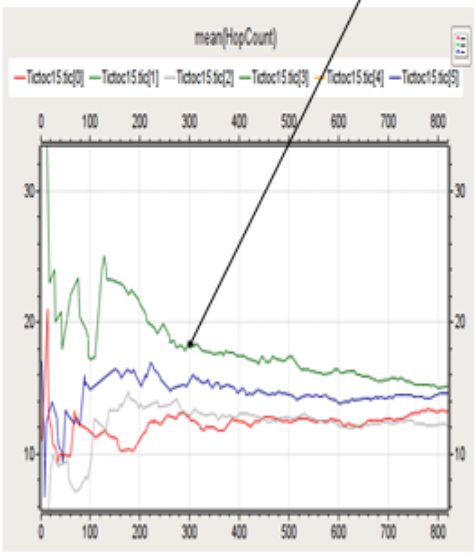
Extending TicToc

Vector data for node 0 and node 1 as time series



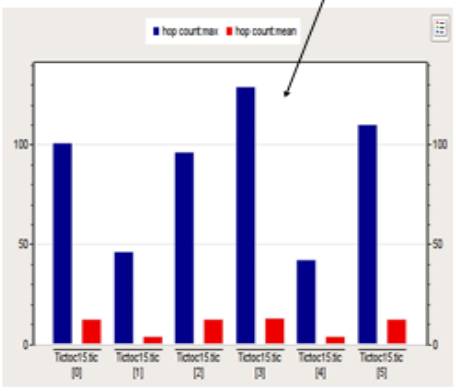
Extending TicToc

Applying mean operation allows us to see how hop count converges to average



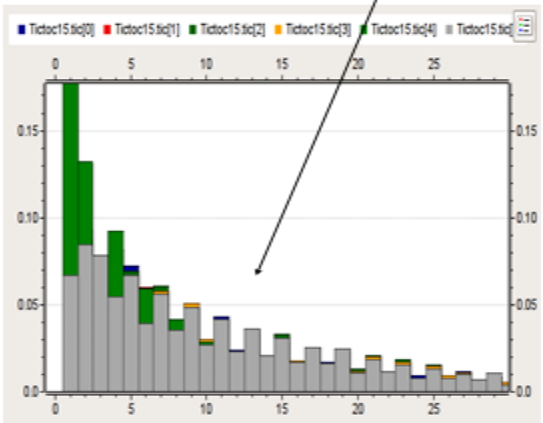
Extending TicToc

scalar data recorded at the end of the simulation shows mean and maximum



Extending TicToc

Histogram of hop count distribution



TOPIC 45

In this module

We shall cover

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

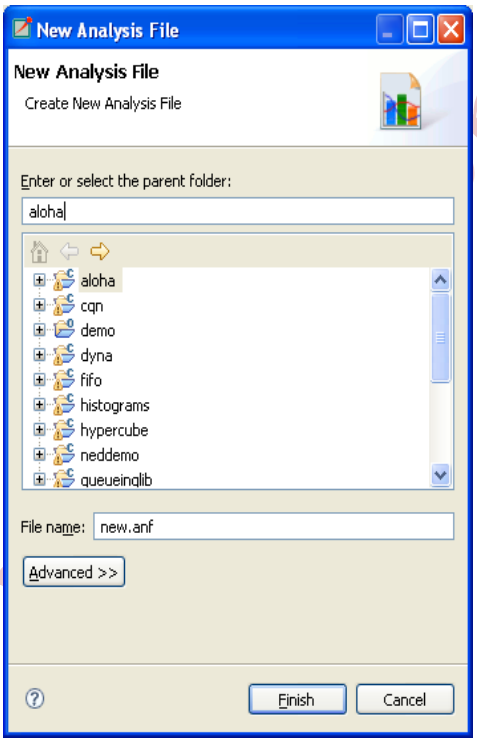
What is Simulation Analysis?

- 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

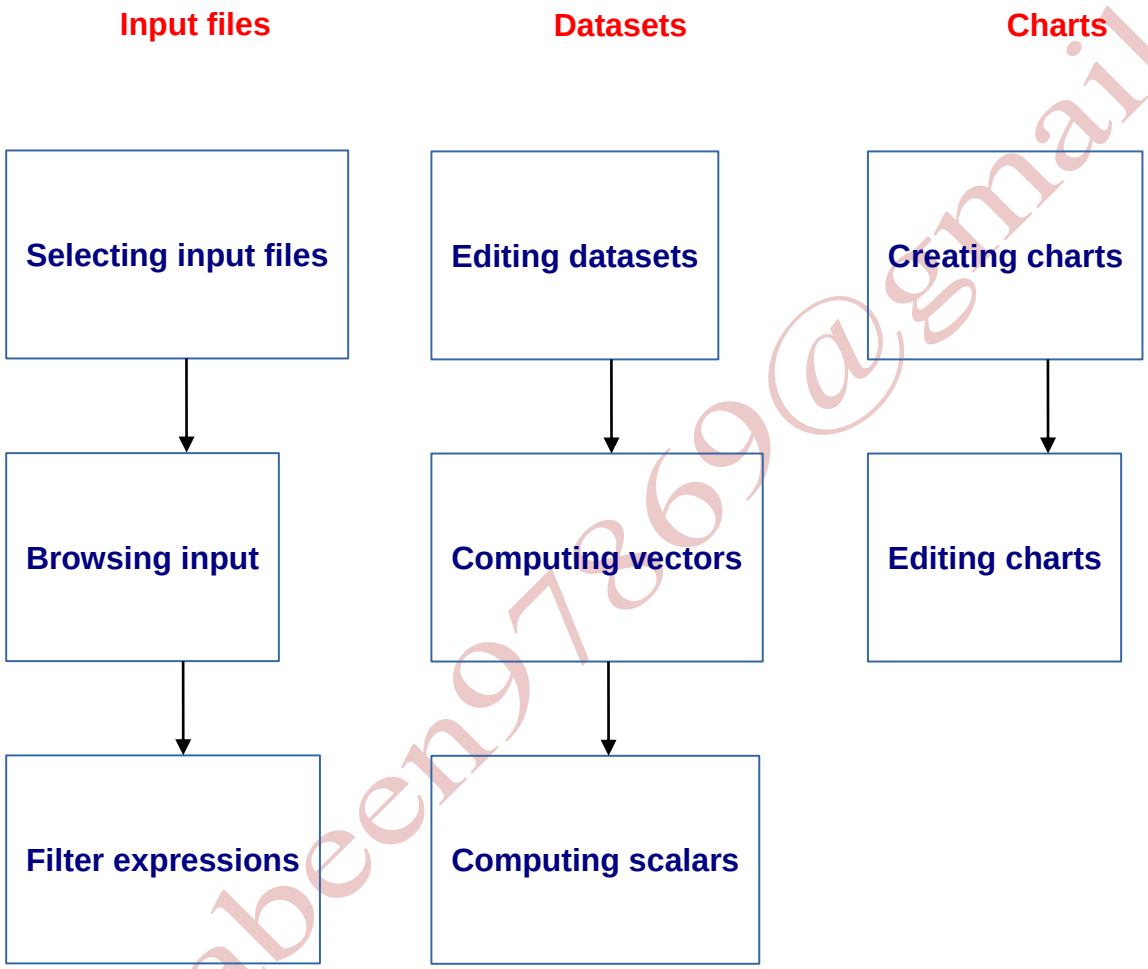


Quick way

- 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

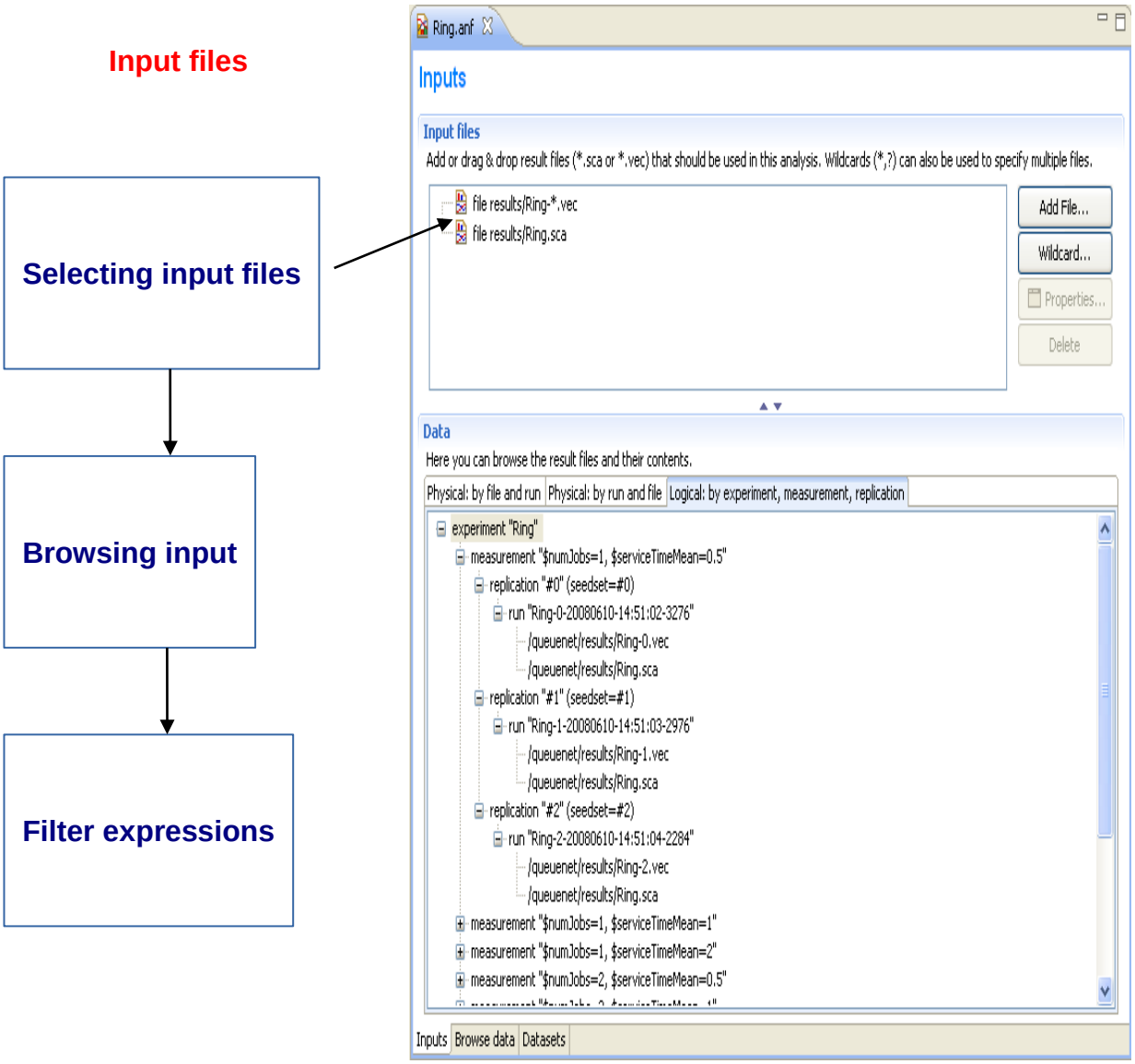
Using the Analysis Editor



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

Analyzing Results

Using the Analysis Editor



Analyzing Results

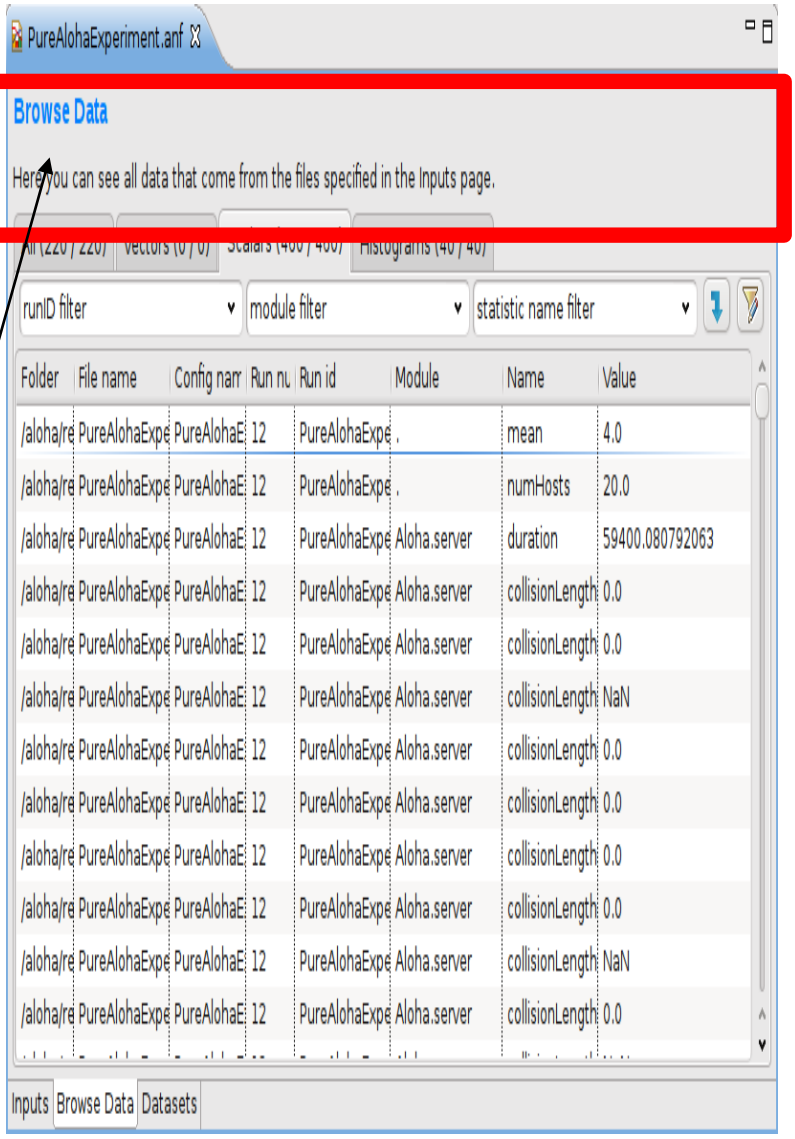
Using the Analysis Editor

Input files

Selecting input files

Browsing input

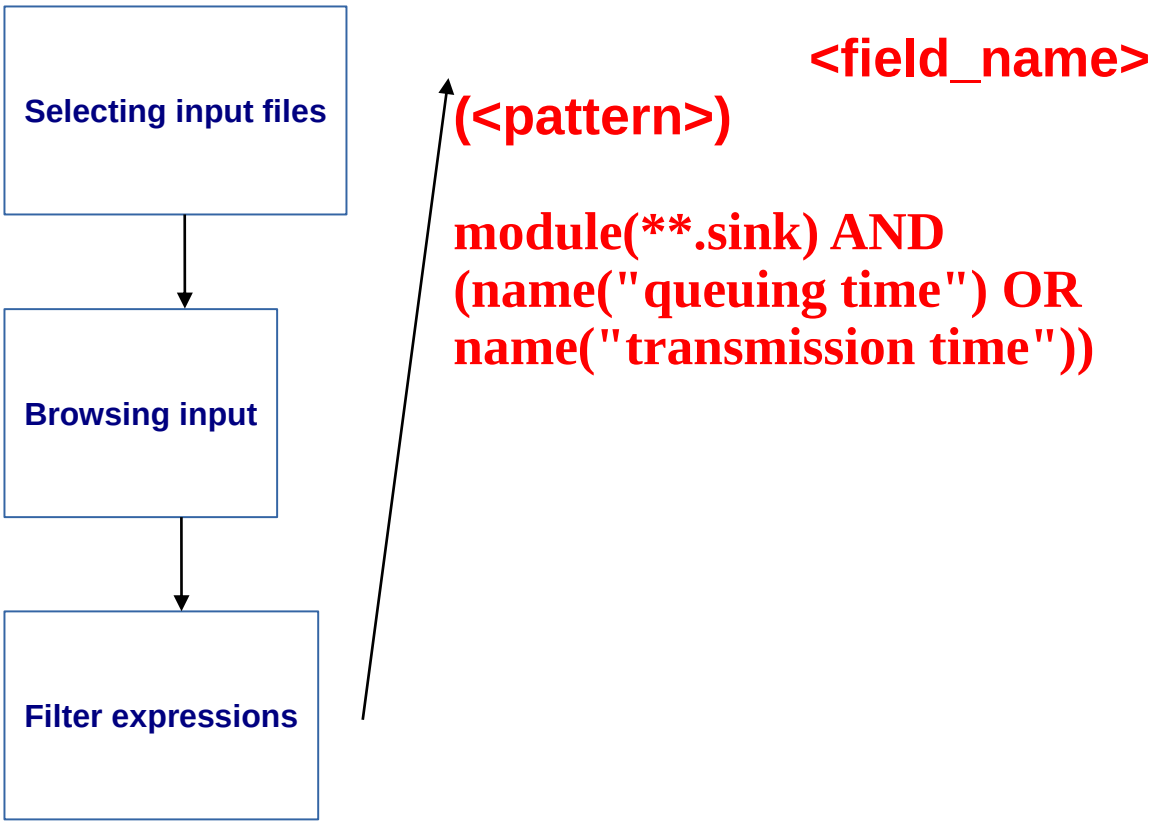
Filter expressions



Analyzing Results

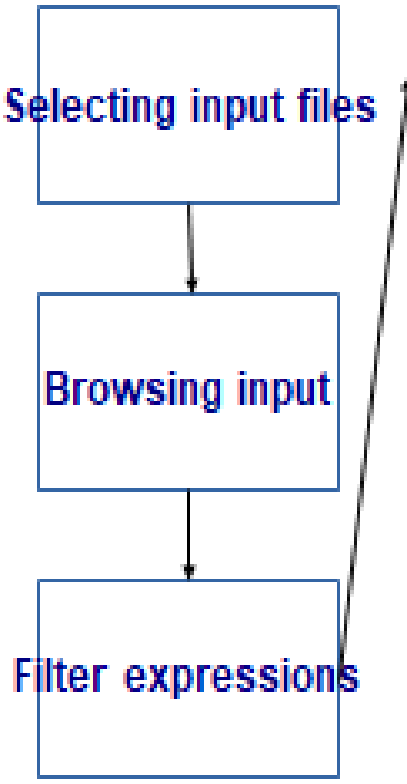
Using the Analysis Editor

Input files



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

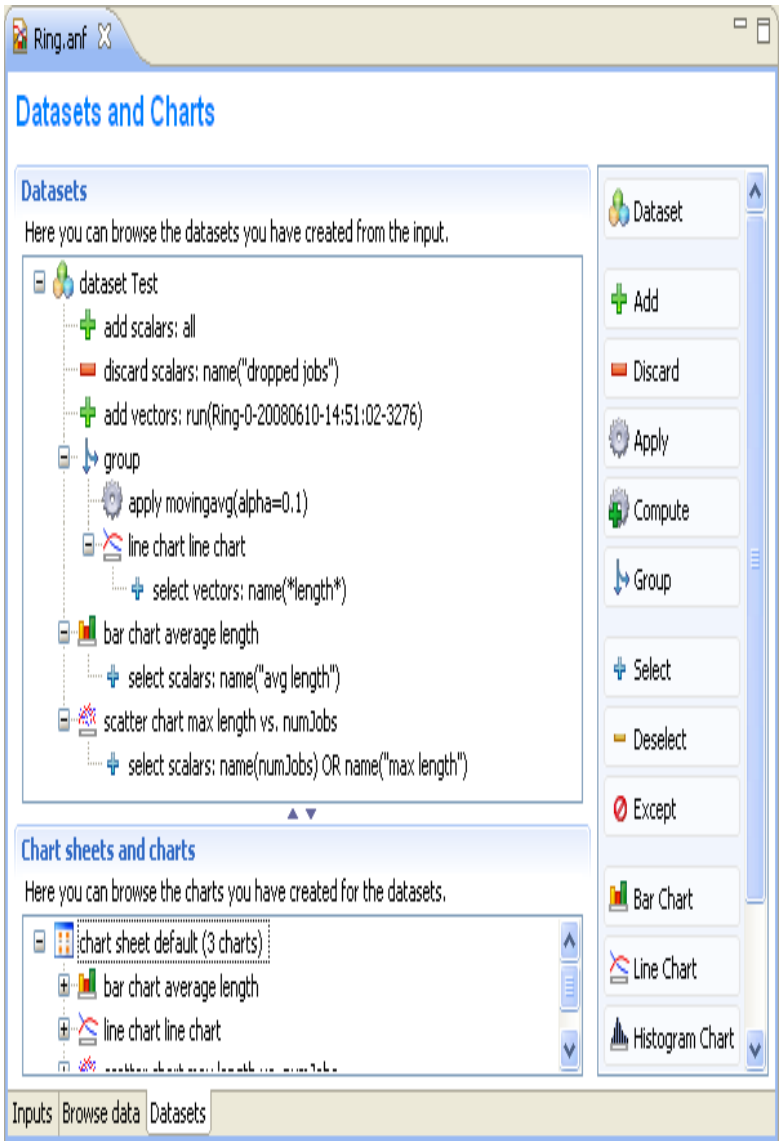
Using the Analysis Editor

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

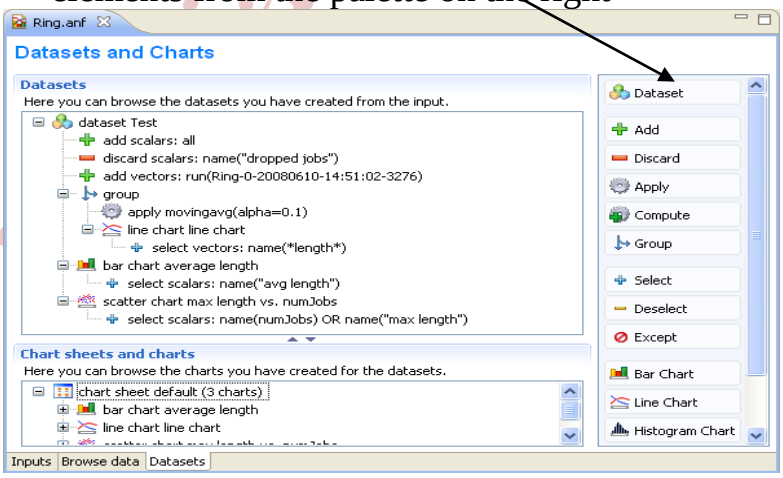
We shall understand

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

Datasets

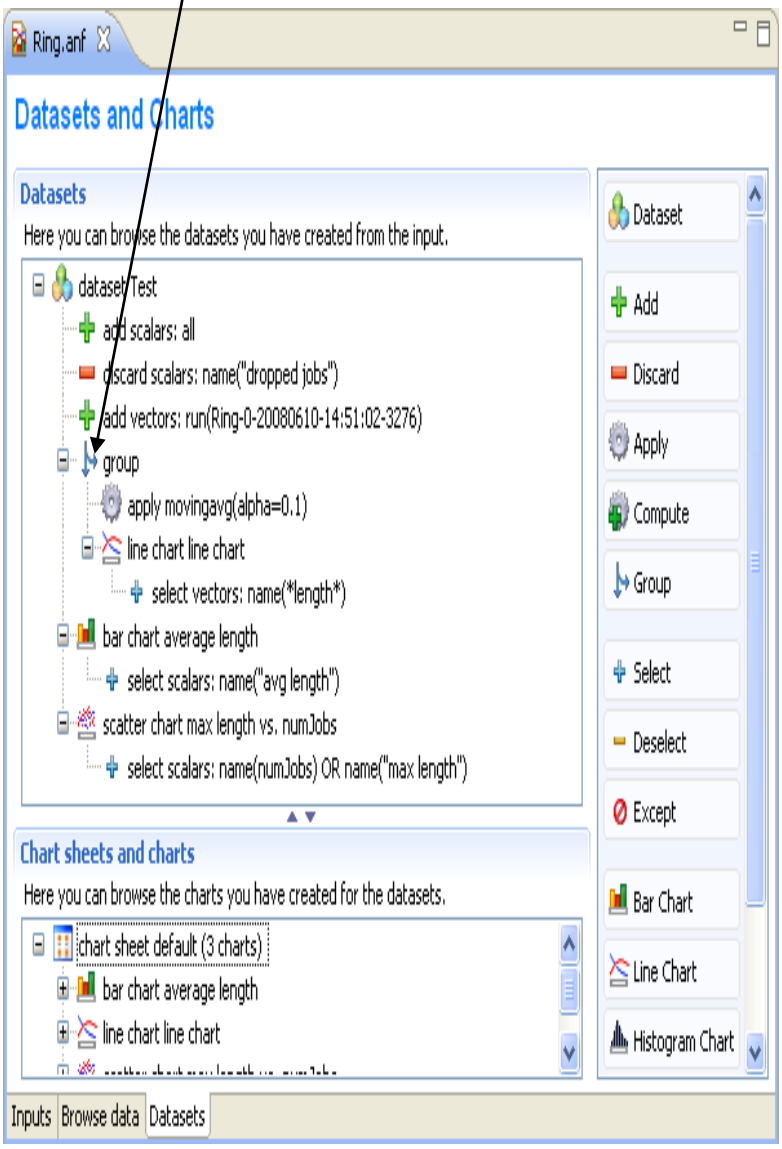
- Describe a set of input data, the processing applied to them and the charts
- Displayed as a tree of processing steps and charts
- Nodes are used for
  - — Adding and discarding data
  - — Applying processing to vectors and scalars
  - — Selecting the operands of the operations
  - — Content of charts,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

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



Editing Datasets

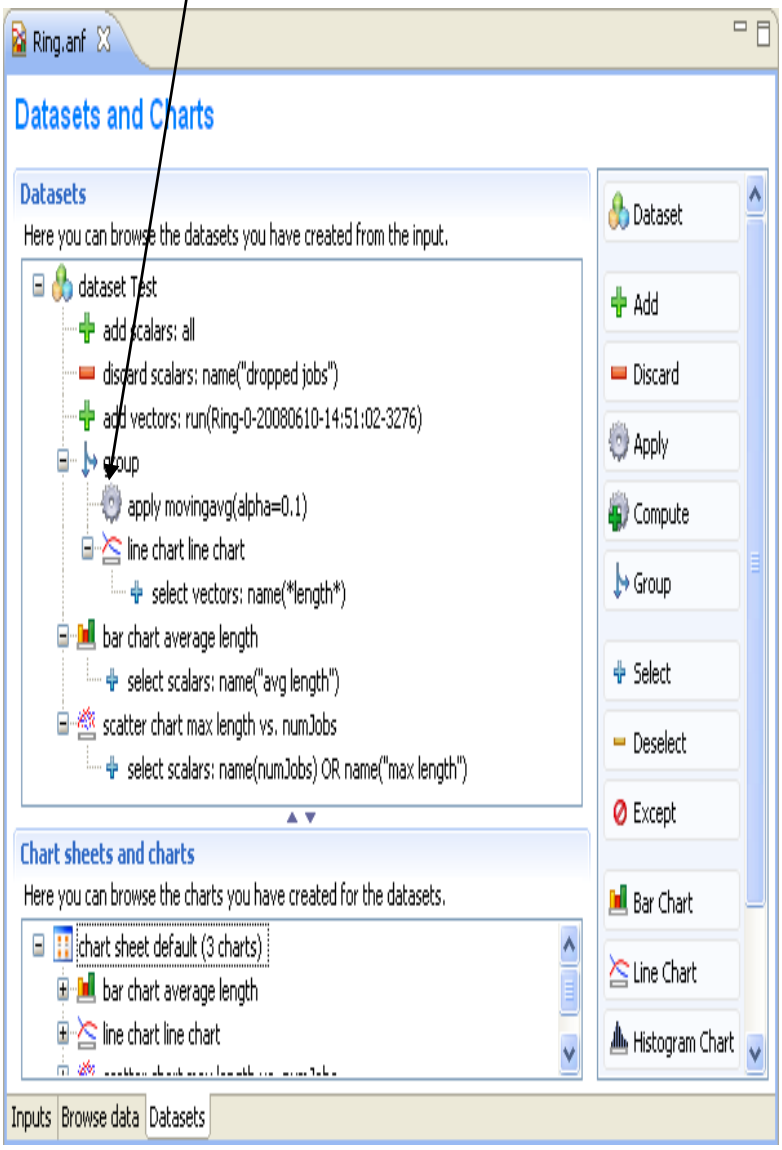
- Processing steps within a Group node only affect the group
- Allows branches to be created in the dataset
- A range of siblings can be grouped together by choosing “Group”



What is Compute Vectors?

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

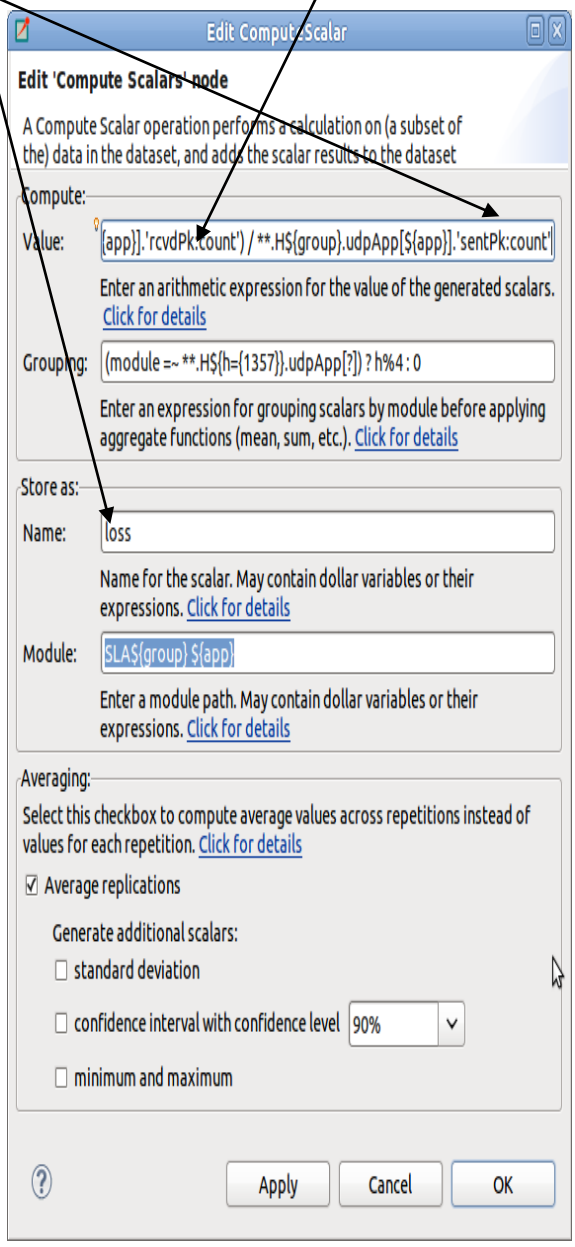
- Computations can be applied to the data by adding Apply to Vectors /Compute Vectors/Compute Scalars nodes to the dataset



What is Compute Scalars?

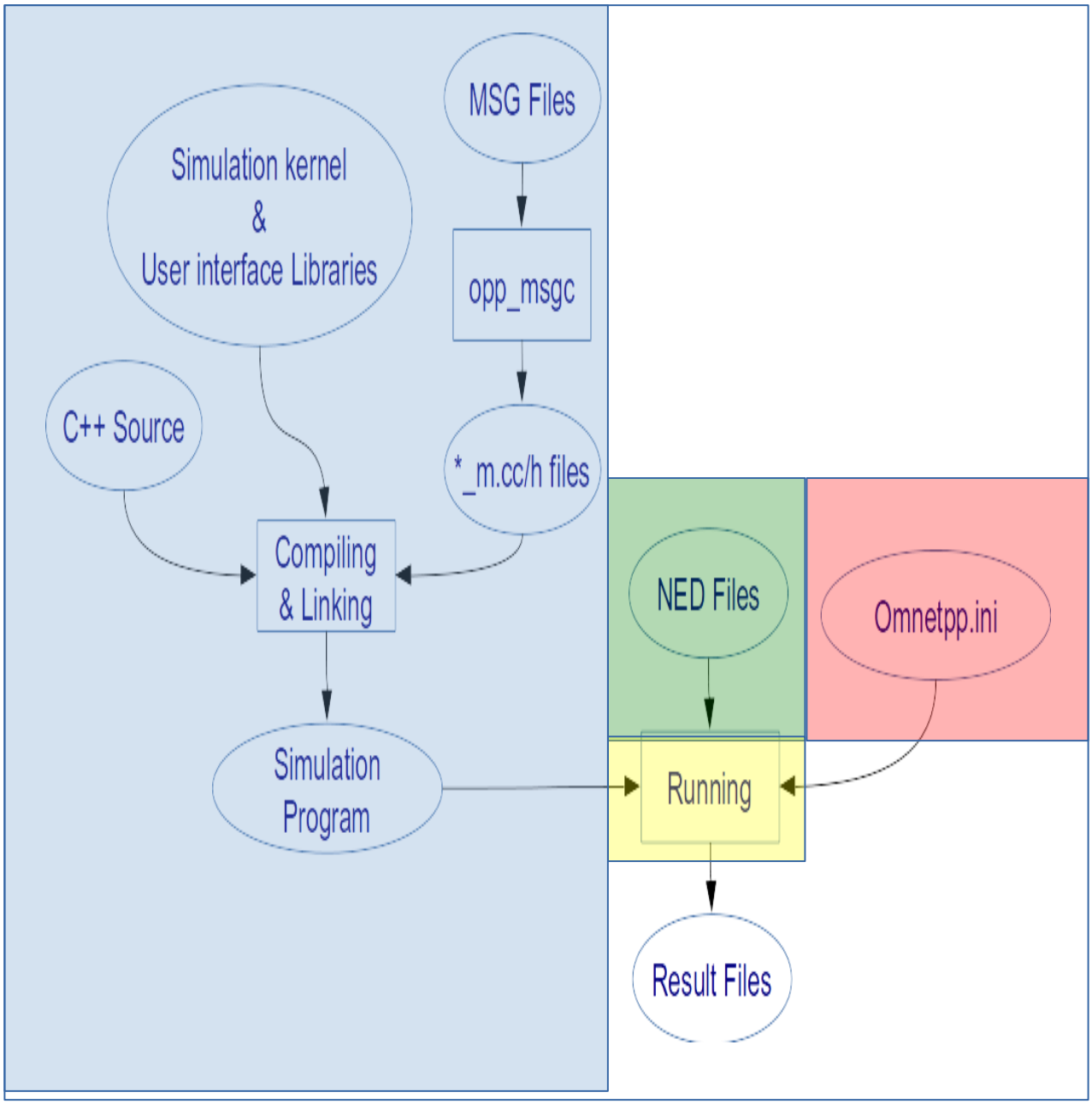
- The Compute Scalars dataset node adds new scalars to the dataset whose values are computed from other statistics in the dataset

- 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

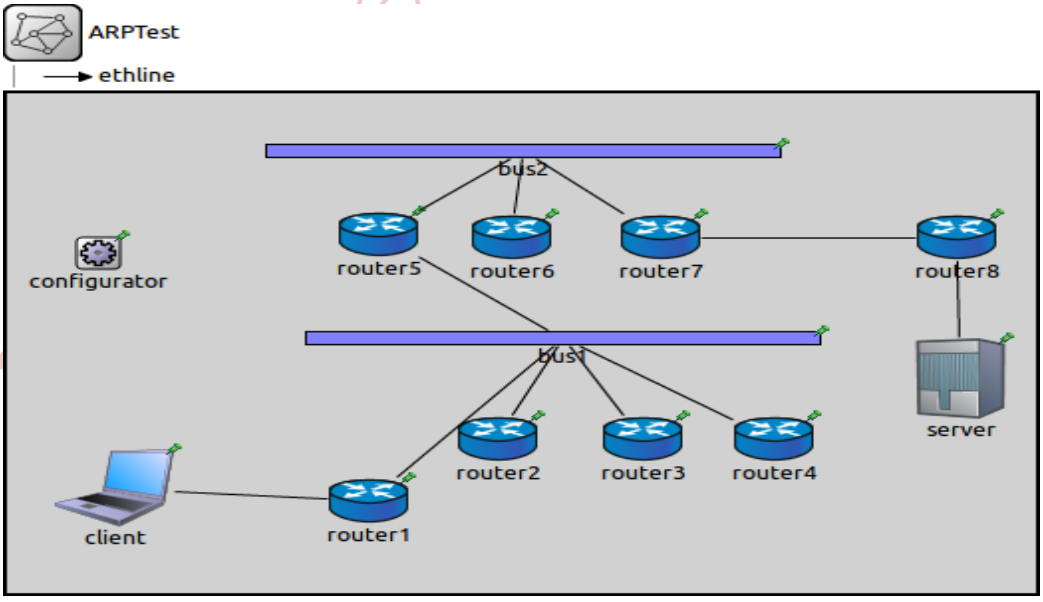
Why ARP scenario?

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

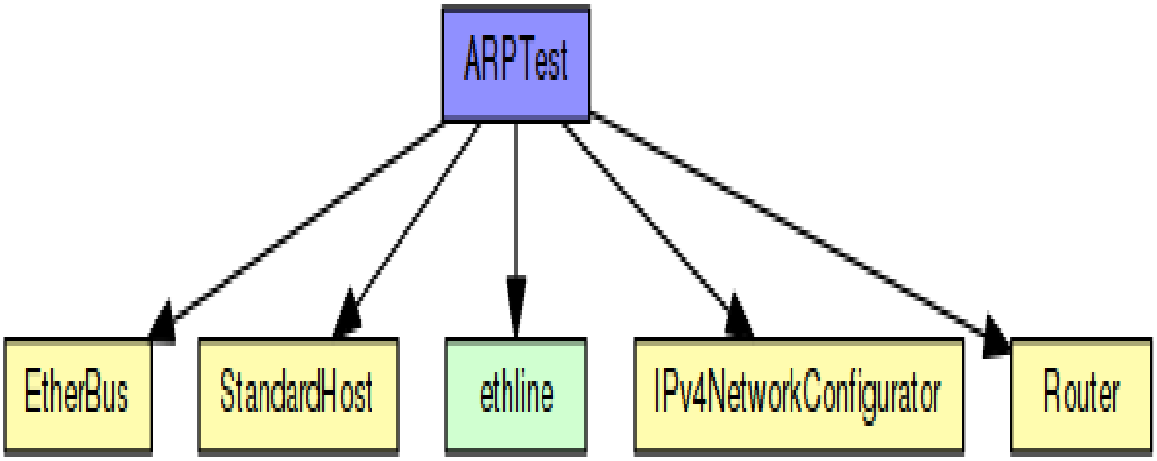
https://omnetpp.org/doc/inet/api-current/neddoc/index.html

Scenario

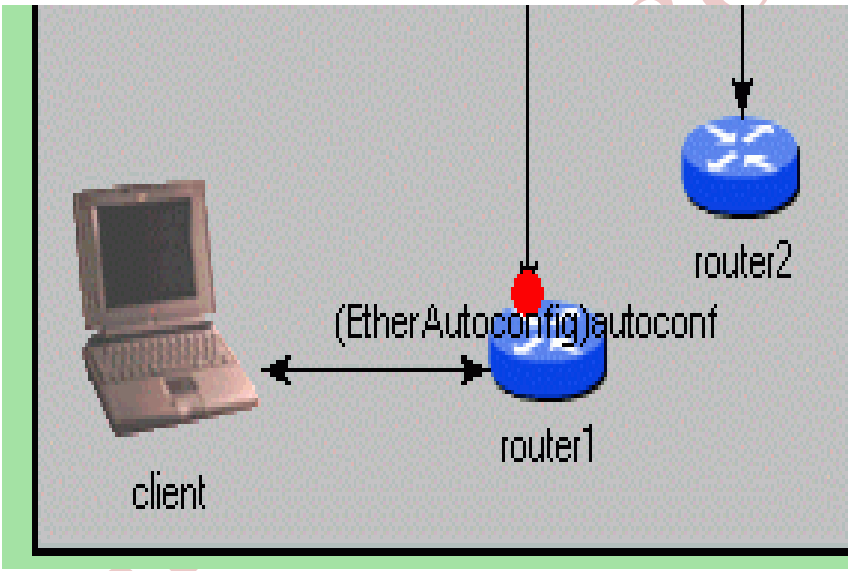
- Client computer opens TCP session with server
- Rest of operations (including ARP) follow
  - ARP has to learn the MAC address for the default router



Usage Diagram for ARP



On simulation start
Ethernet autoconfiguration precedes ARP



TOPIC 51

Design Tour of INET 2

In this module

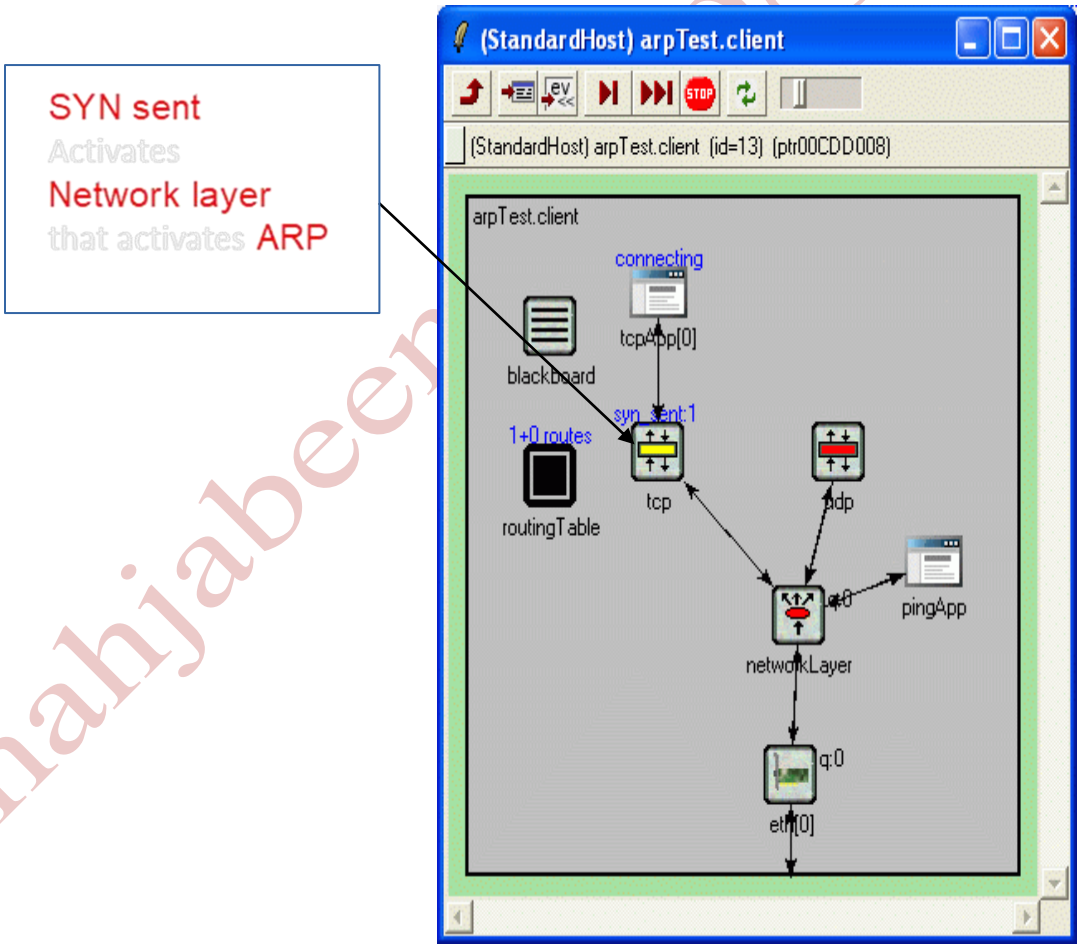
We shall take a guided
Tour of INET to

- 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?

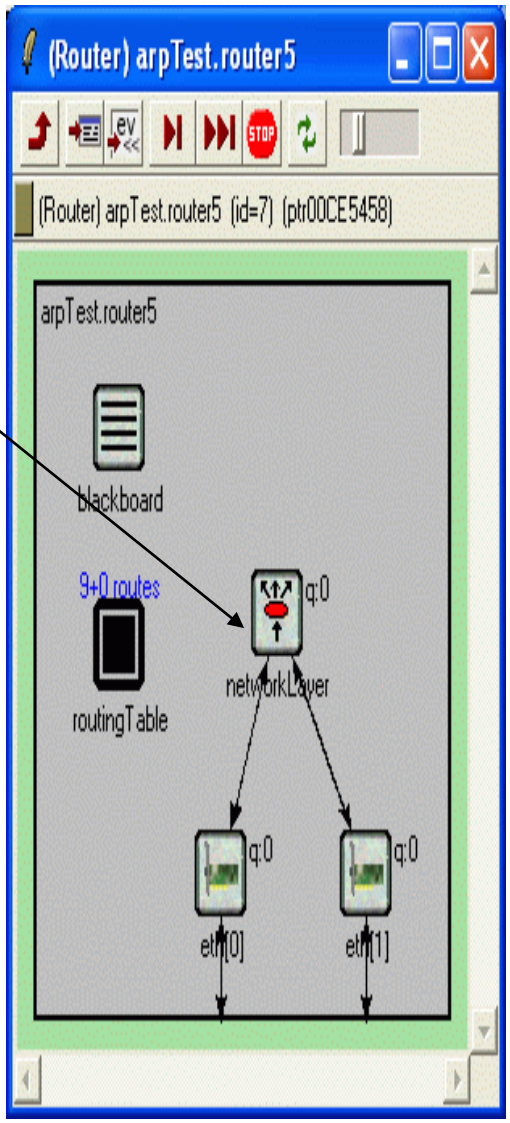
TCP Client



Design Tour of INET 2

Router

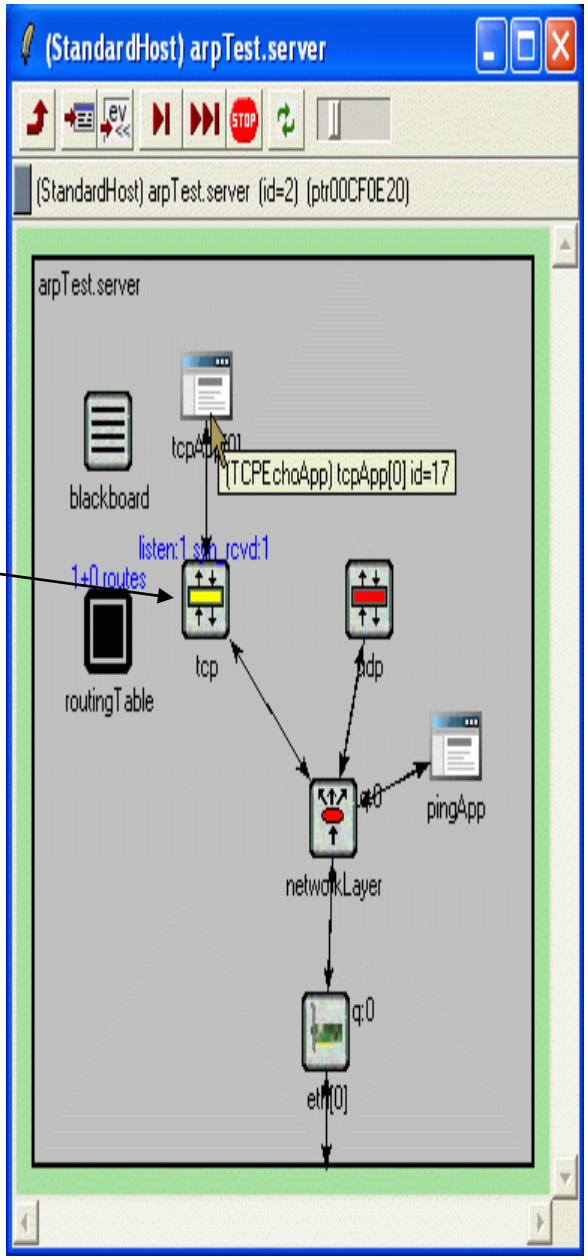
ARP request
Activates
Router to send
ARP reply



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

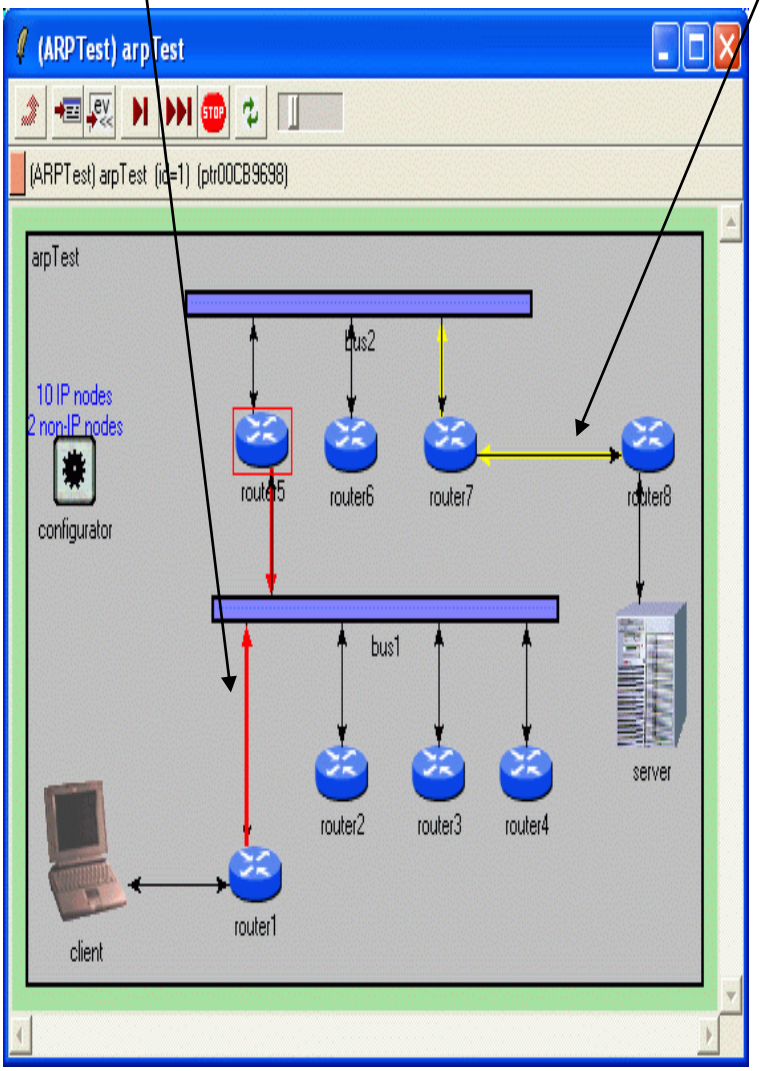


Design Tour of INET 2

End-to-end transmission

Red: Collided and backing off

Yellow: Node transmitting on link



TOPIC 52

Design Tour of INET 3

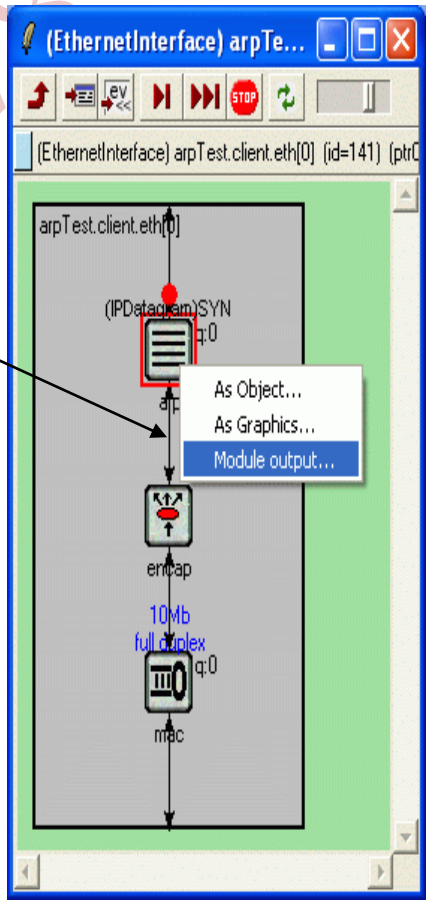
In this module
We shall take a guided
Tour of INET to

- 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

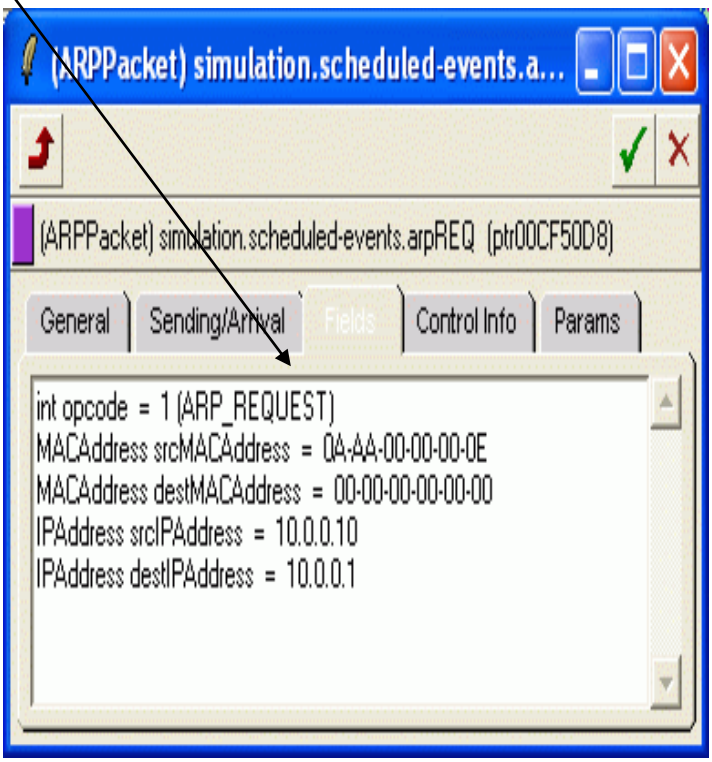
arpTest.client.eth[0].arp

module output

Design Tour of INET 3

Inside ARP Packet

ARP Broadcast Message

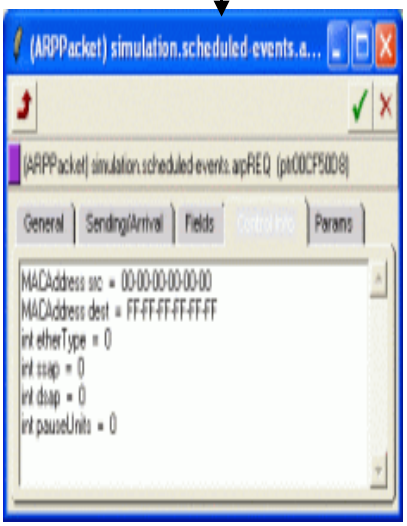


Design Tour of INET 3

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;
};
```

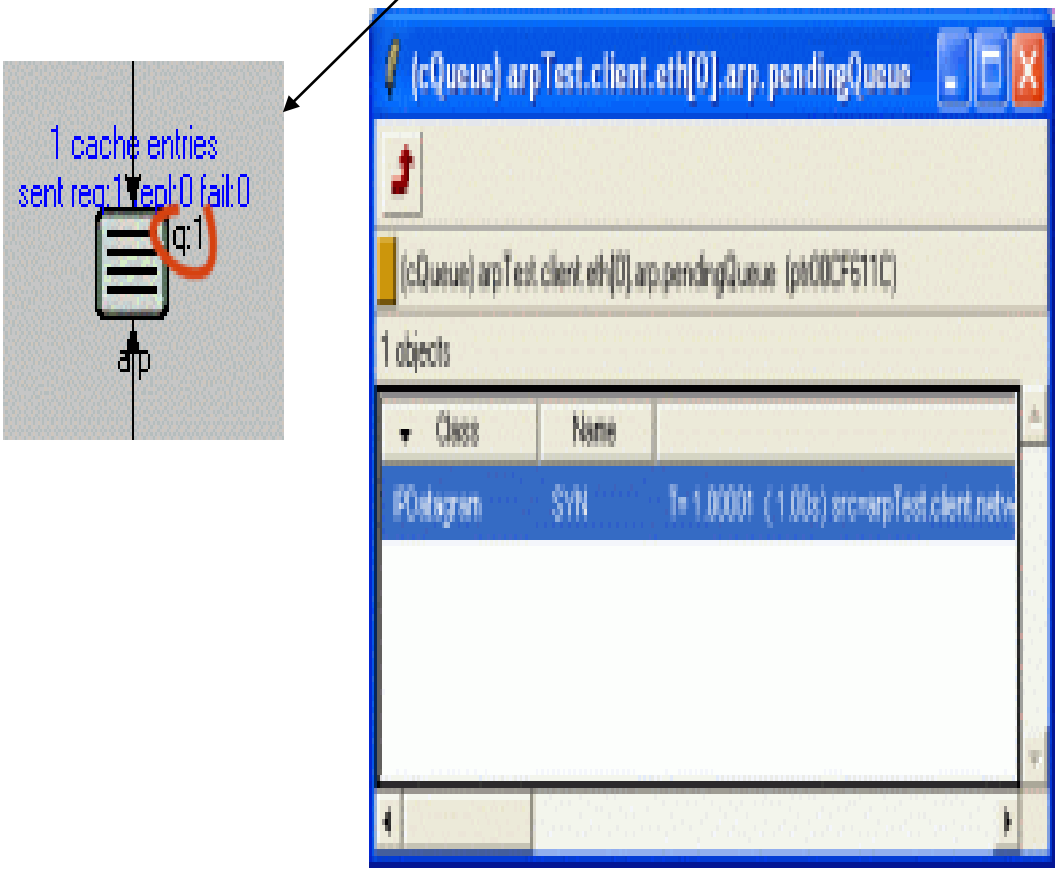
This packet is appended with broadcast address in control info (a small data structure)



Design Tour of INET 3

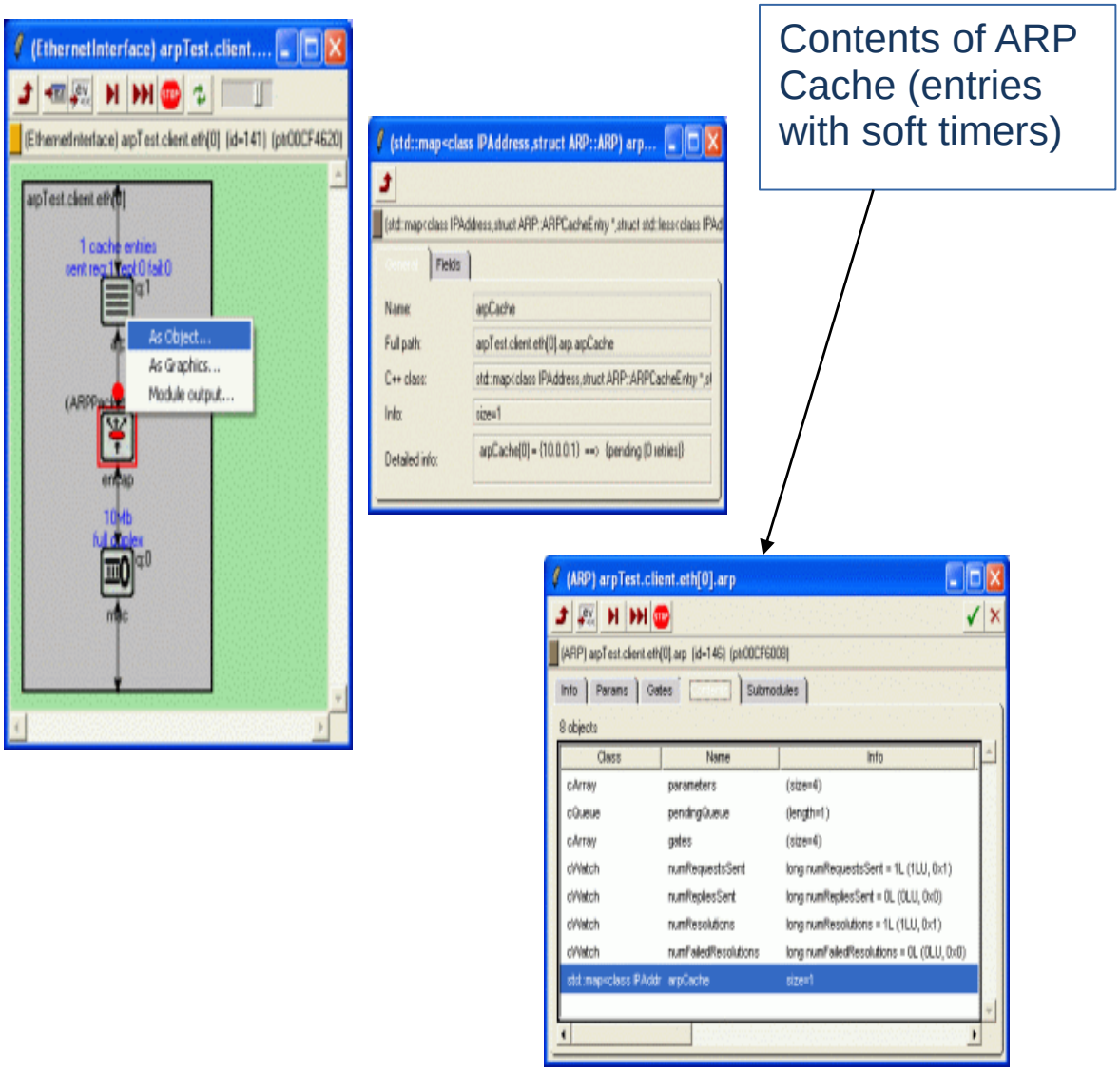
Packet Queue (Contains IP Packet)

This packet is enqueued till ARP resolution



Design Tour of INET 3

ARP Cache Build-up



TOPIC 53

Introduction to top-down approach to modelling and simulation

In this module

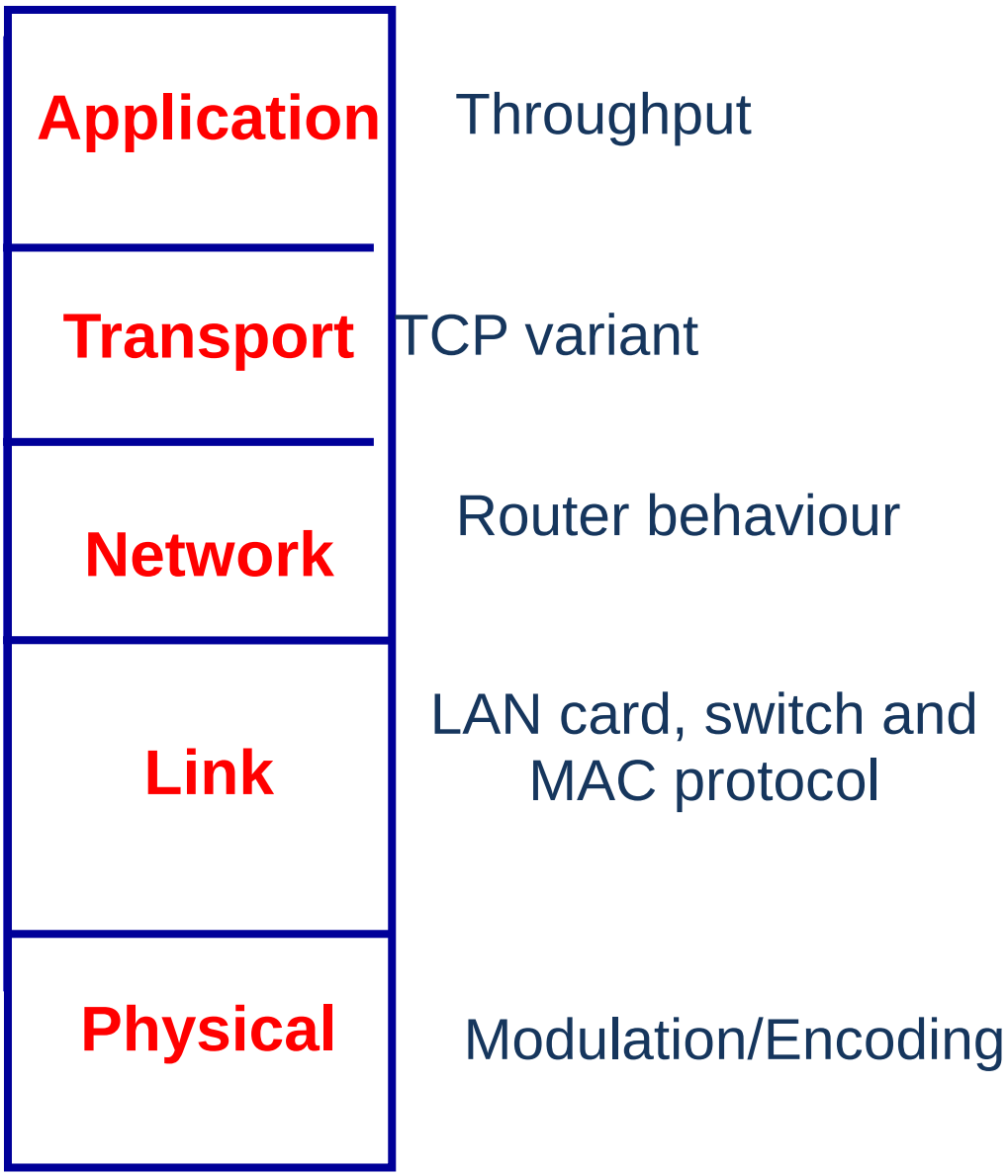
We shall understand

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

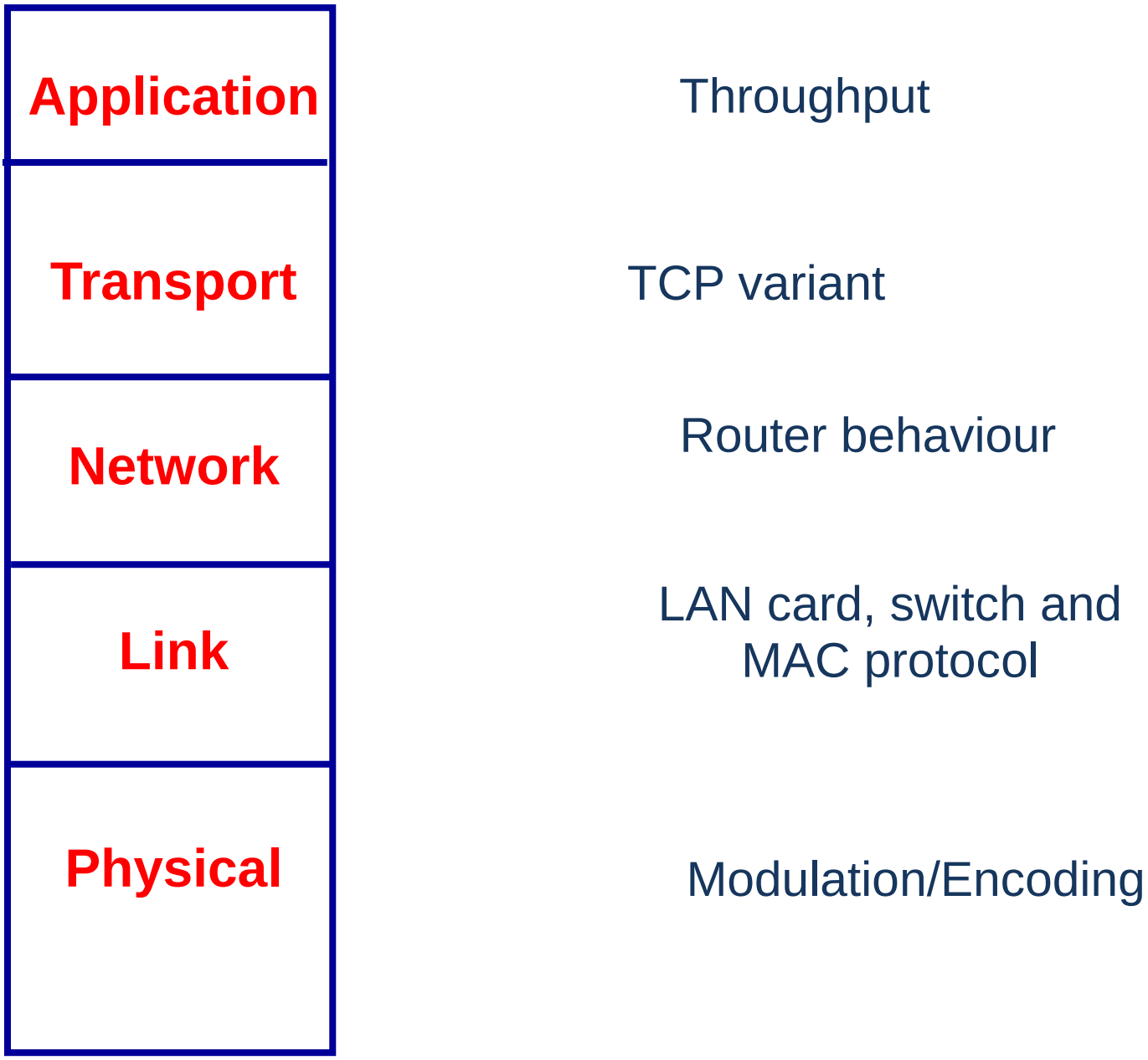
- **Top-down approach to NeMS**
- Networks are complex to design
- One-time design of simulation is cumbersome
- **Top-down**: Phased roll-out of model-simulate cycle
  - Iterative

Introduction to top-down approach to modelling and simulation

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

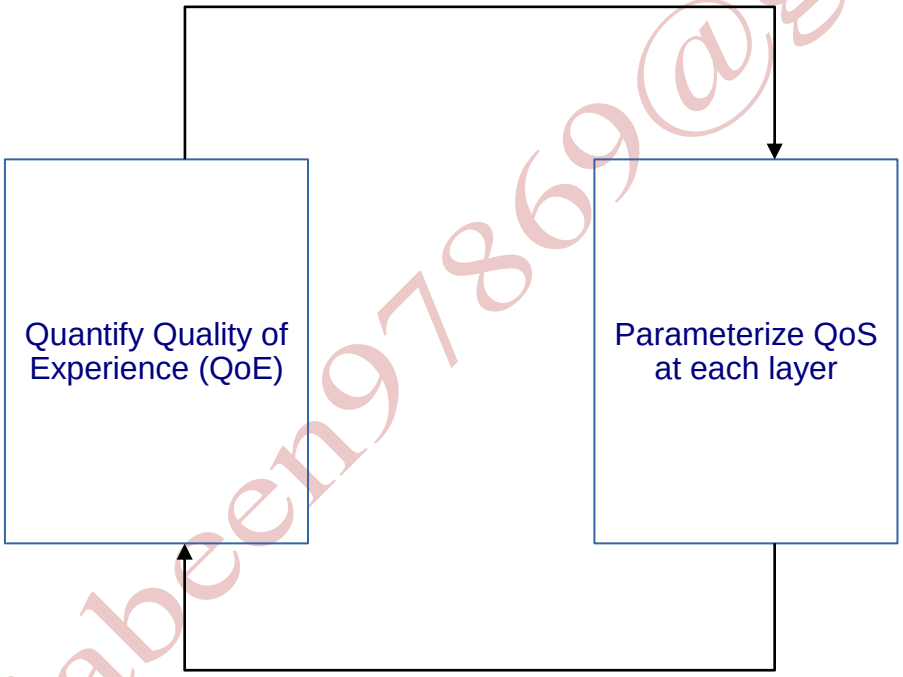


Scalability	Availability	Performance	Security
Manageability	Usability	Adaptability	Affordability



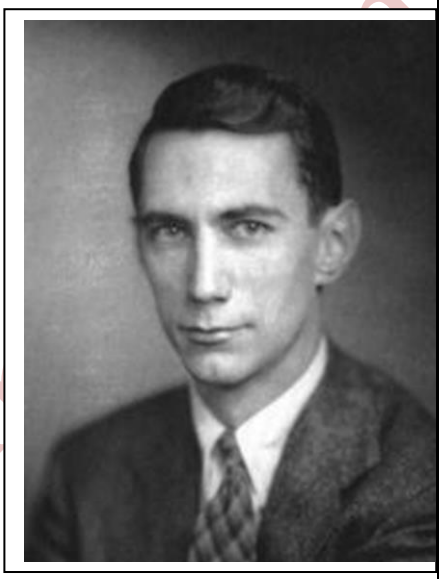
Introduction to top-down approach to modelling and simulation

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!

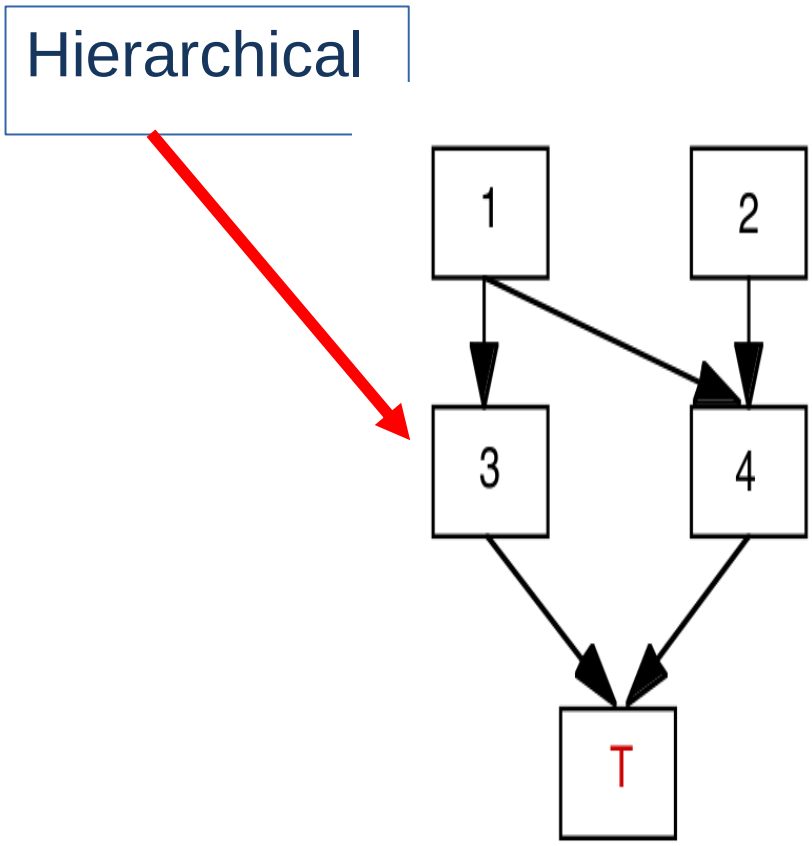
Constituents of an equation?
- Expressions consist of one or more of these arguments
  - Numerical constants
  - Symbolic names
  - Mathematical operators
  - Functions
  - Conditional expressions

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)

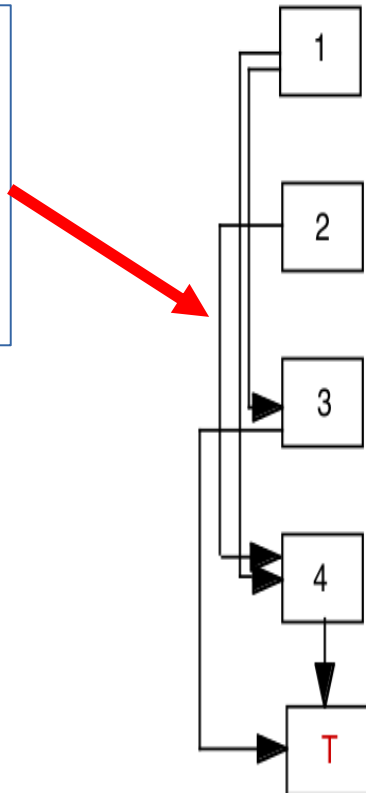


CS432 Handouts Made by Mahjabeen
mahjabeen97869@gmail.com
contact # 0321 2711298

Rules for Mathematical Writing

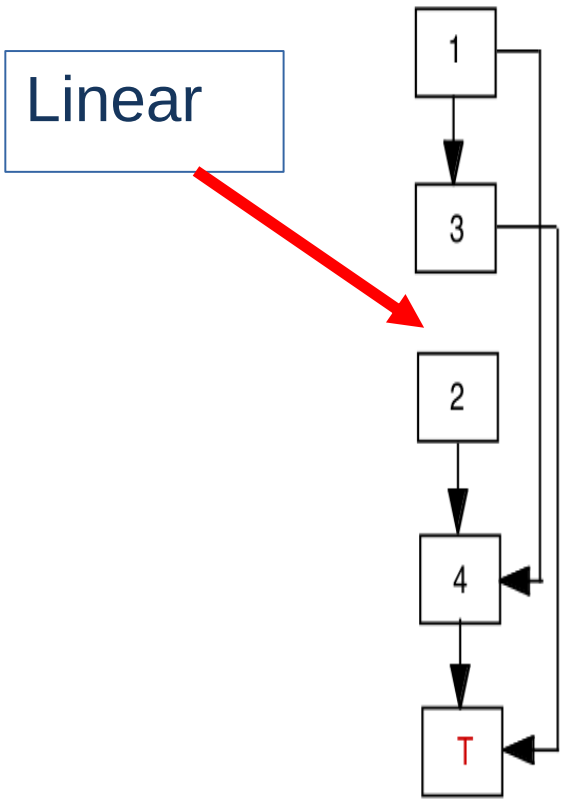
Relationship between arguments of a segment (2 of 3)

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

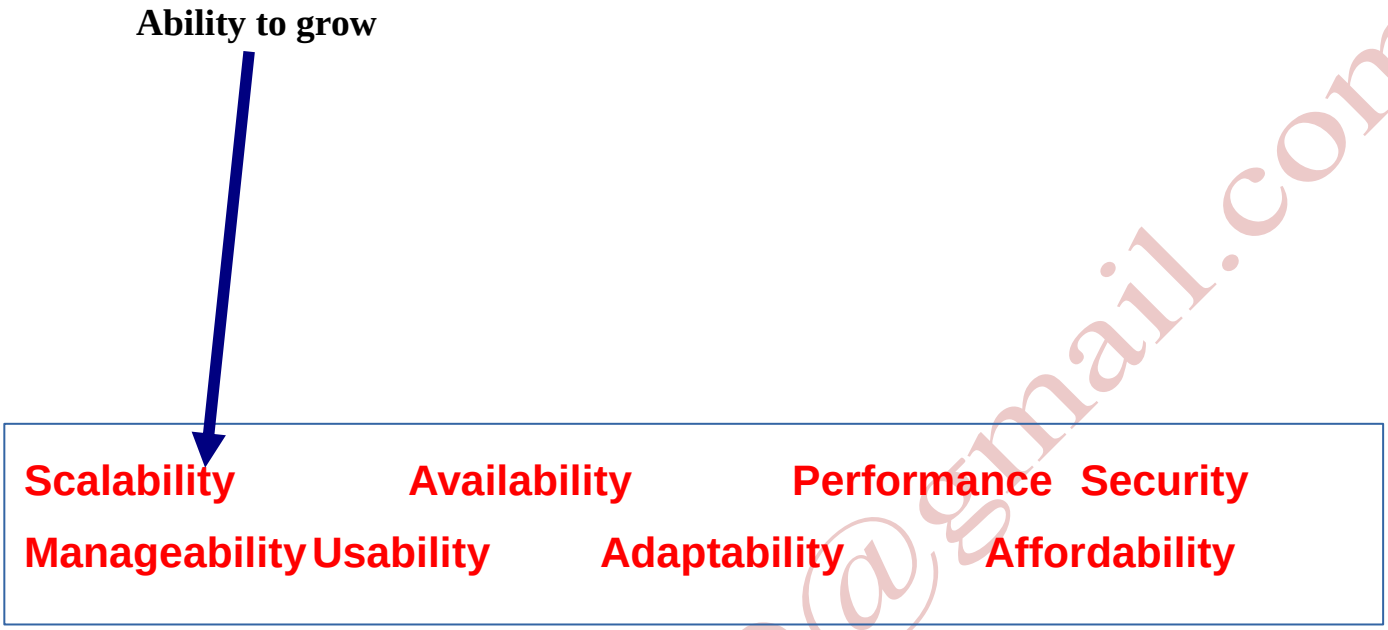
END

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

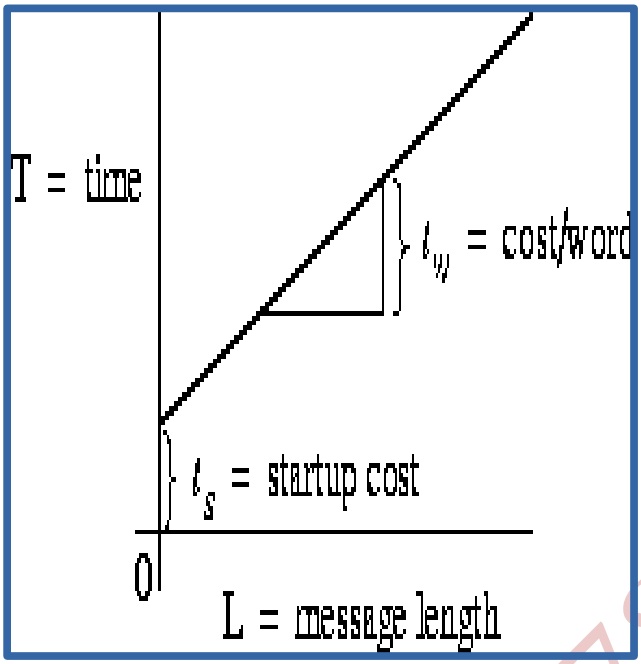
$E_{Relative} = T_1$ ✿ (No. of hosts ◇ $T_{No\ of\ hosts}$)

$S_{Relative}$ = No. of hosts ◇ 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

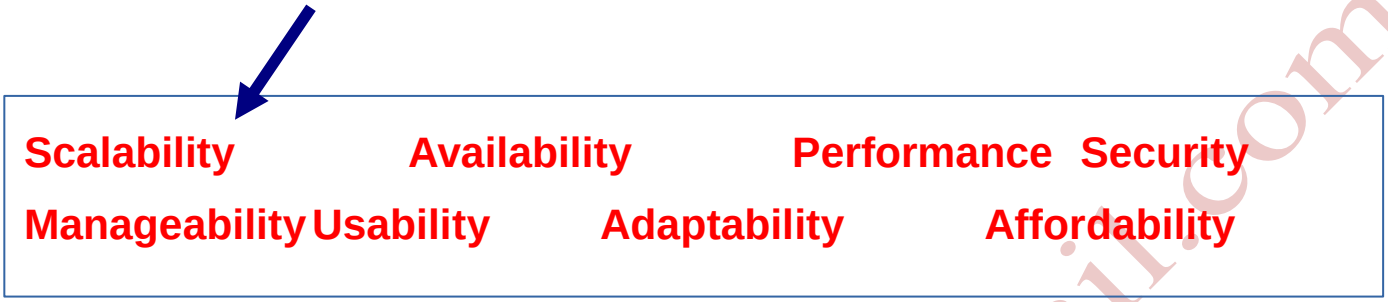
In this module

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

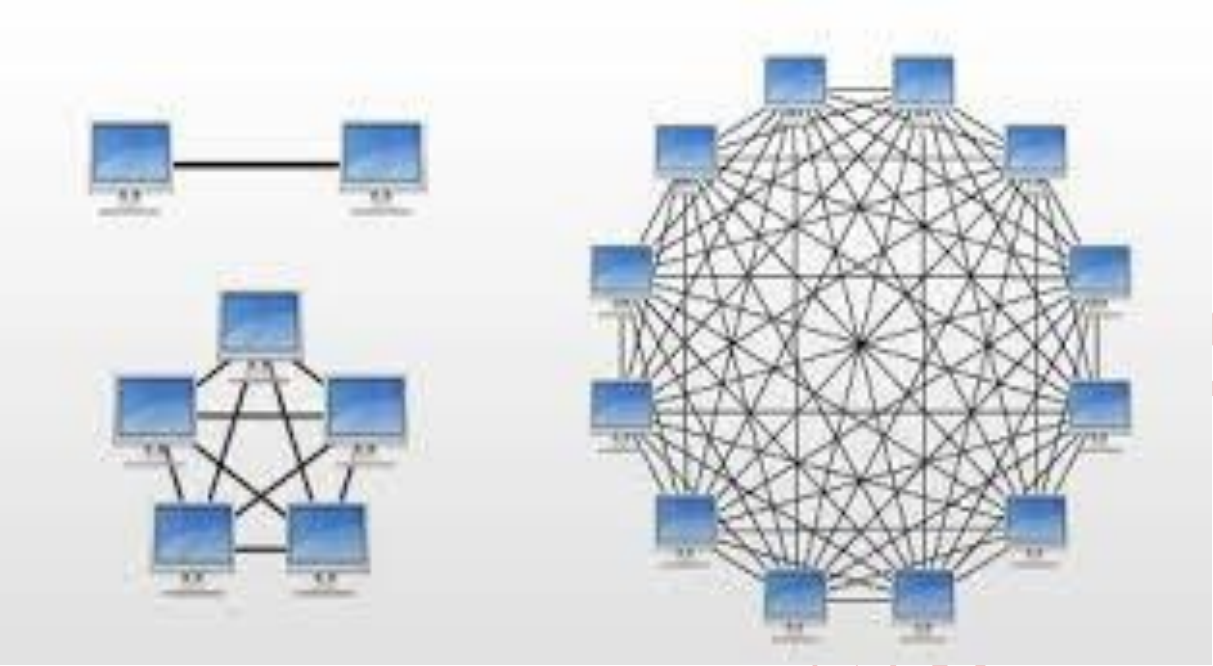
- Social networking has emerged
- Extranets need topology definitions & dedicated bandwidth allocation
  - Classic 80-20 rule <**?**>
- Increased access
  - Data available to more departments
  - Increased utilization of network services

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

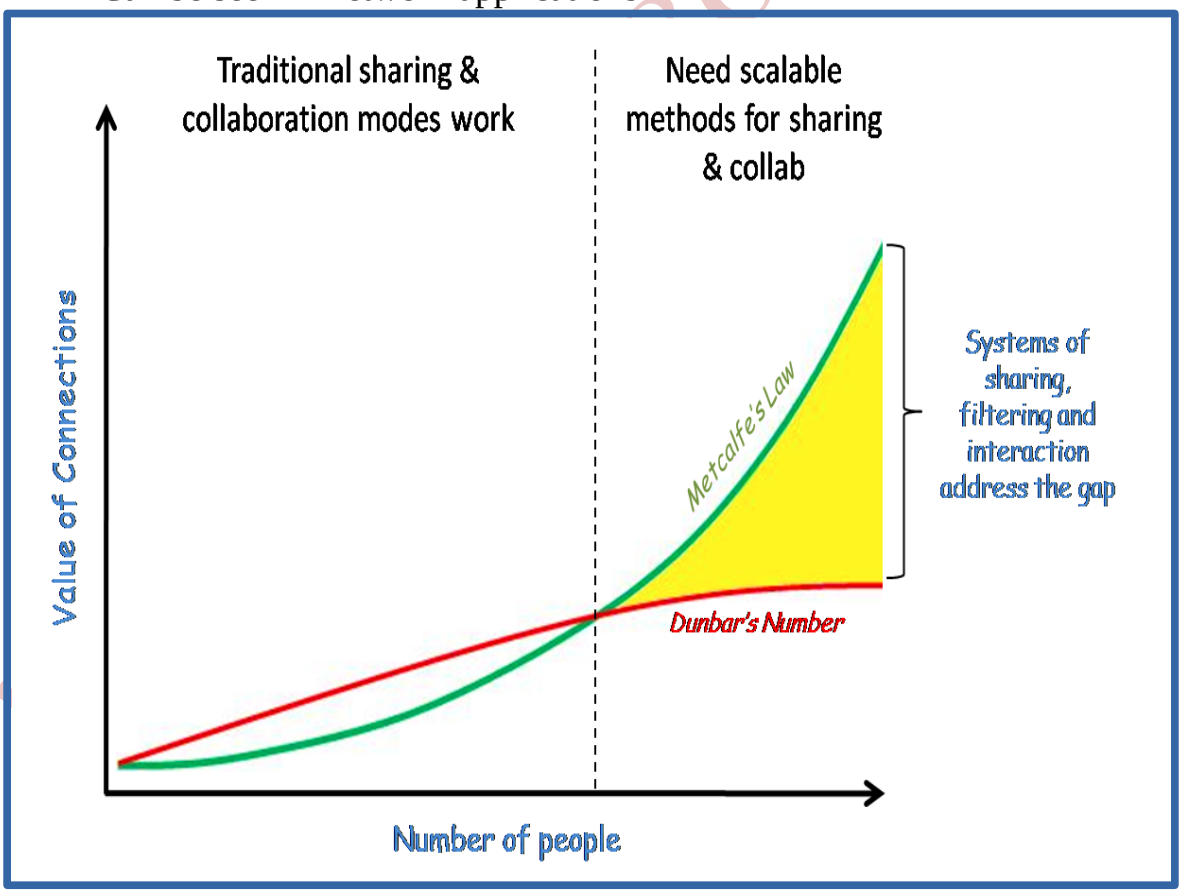
Expression for Metcalfe's Law

- $n(n-1)/2$ 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

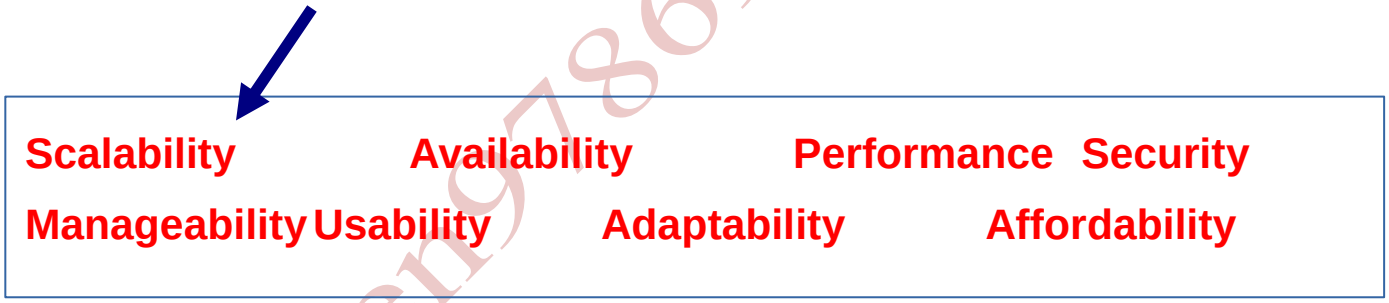
QoE—Constraints on Scalability

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

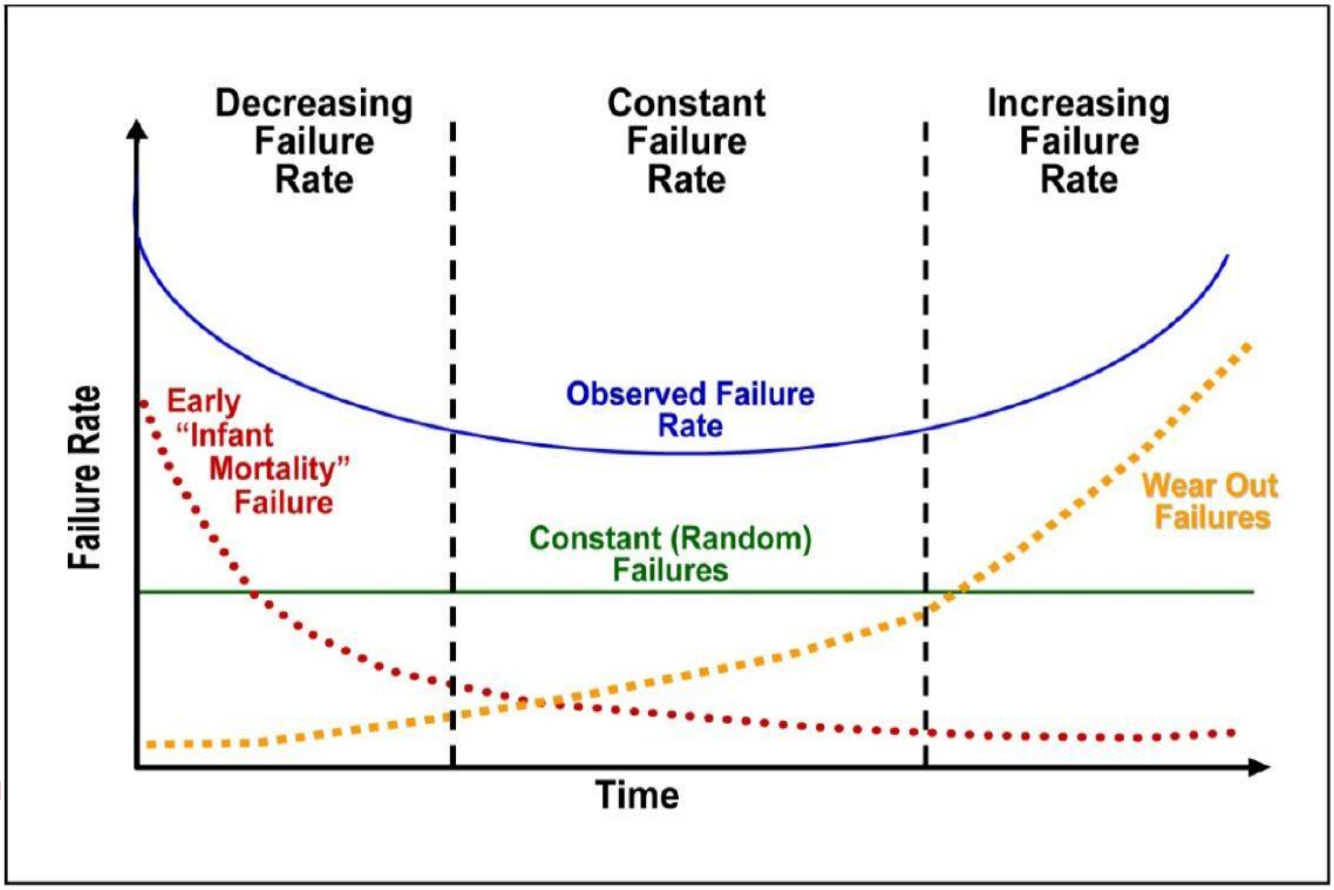
The degree to which a system, subsystem or equipment is in a specified operable and committable state

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

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

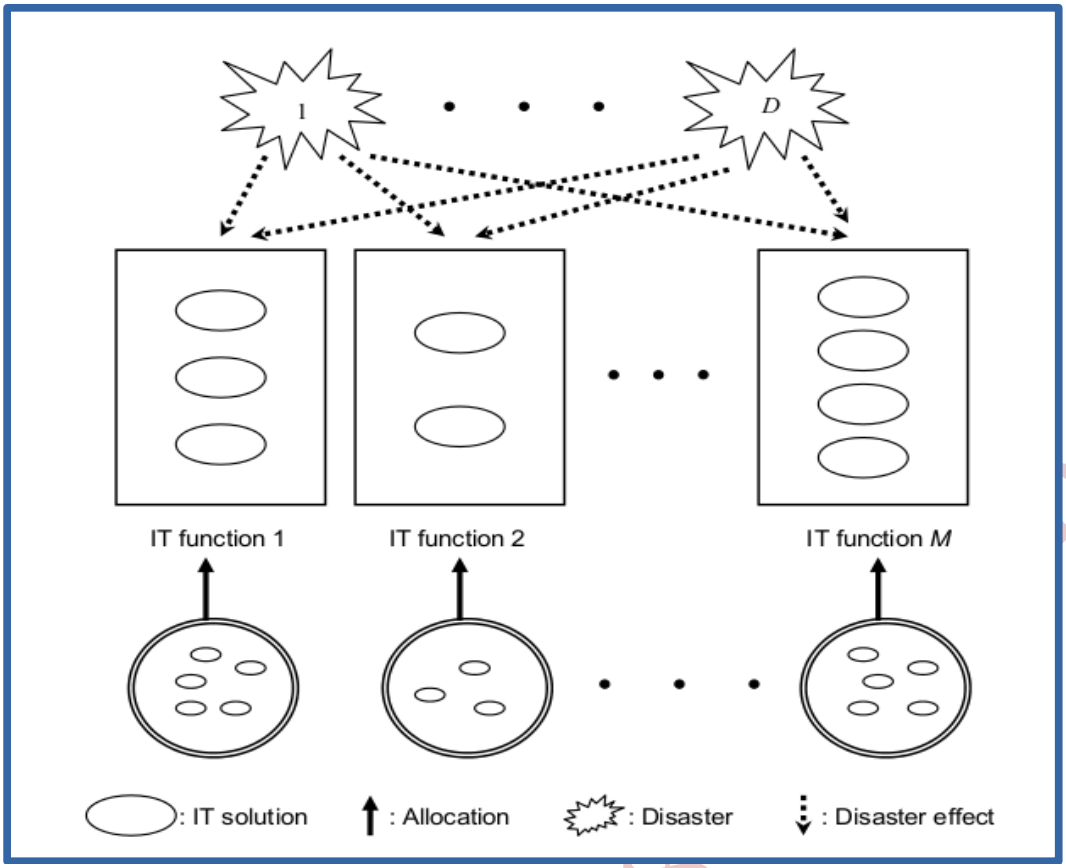
Benjamin B. M. Shao, "Allocating Redundancy to Critical Information Technology Functions for Disaster Recovery," *Proc. 10Th Americas Conference on Information Systems*, Aug. 2004

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

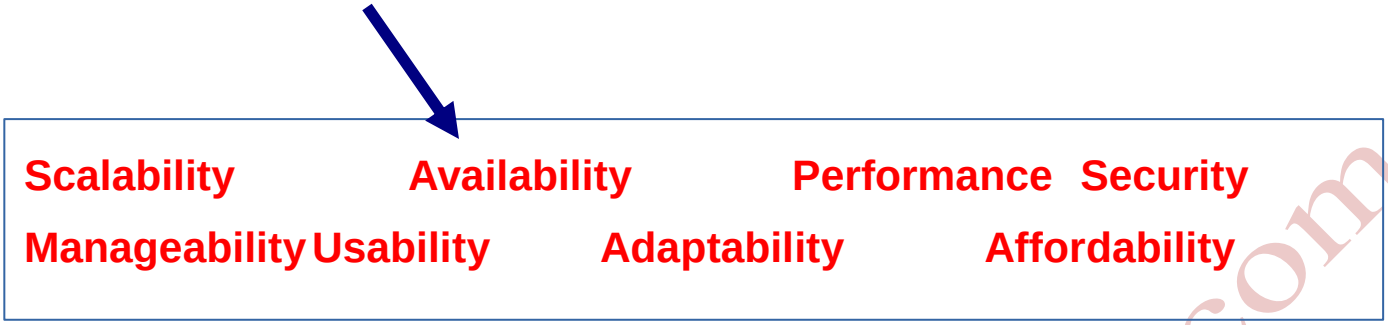
QoE—Specifying Requirements

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

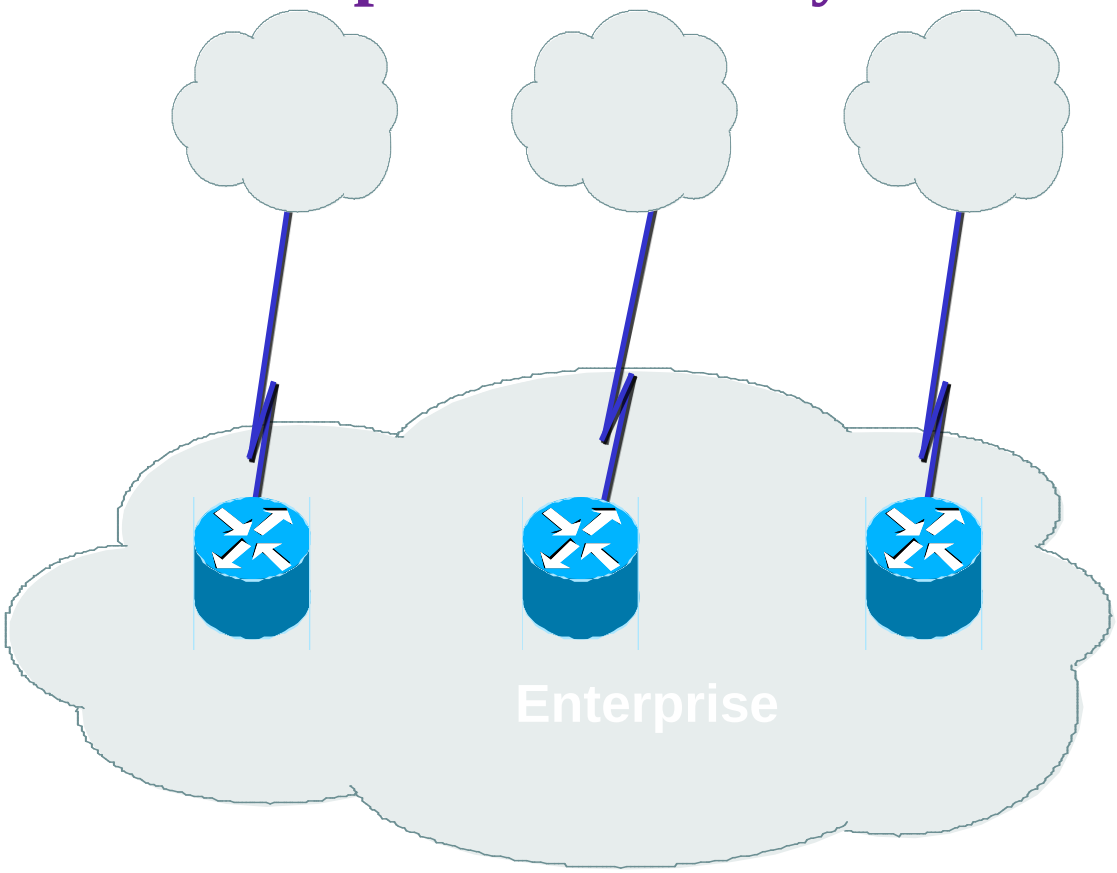
QoE—Five Nines Availability

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

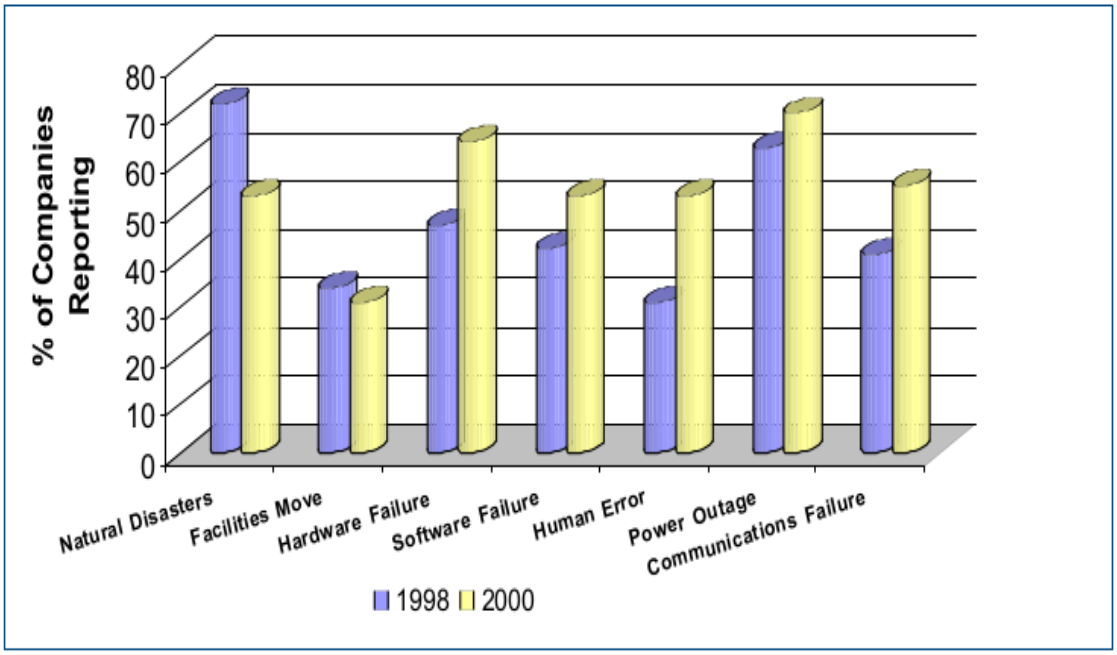
QoE—Cost of downtime

In this module
We shall understand

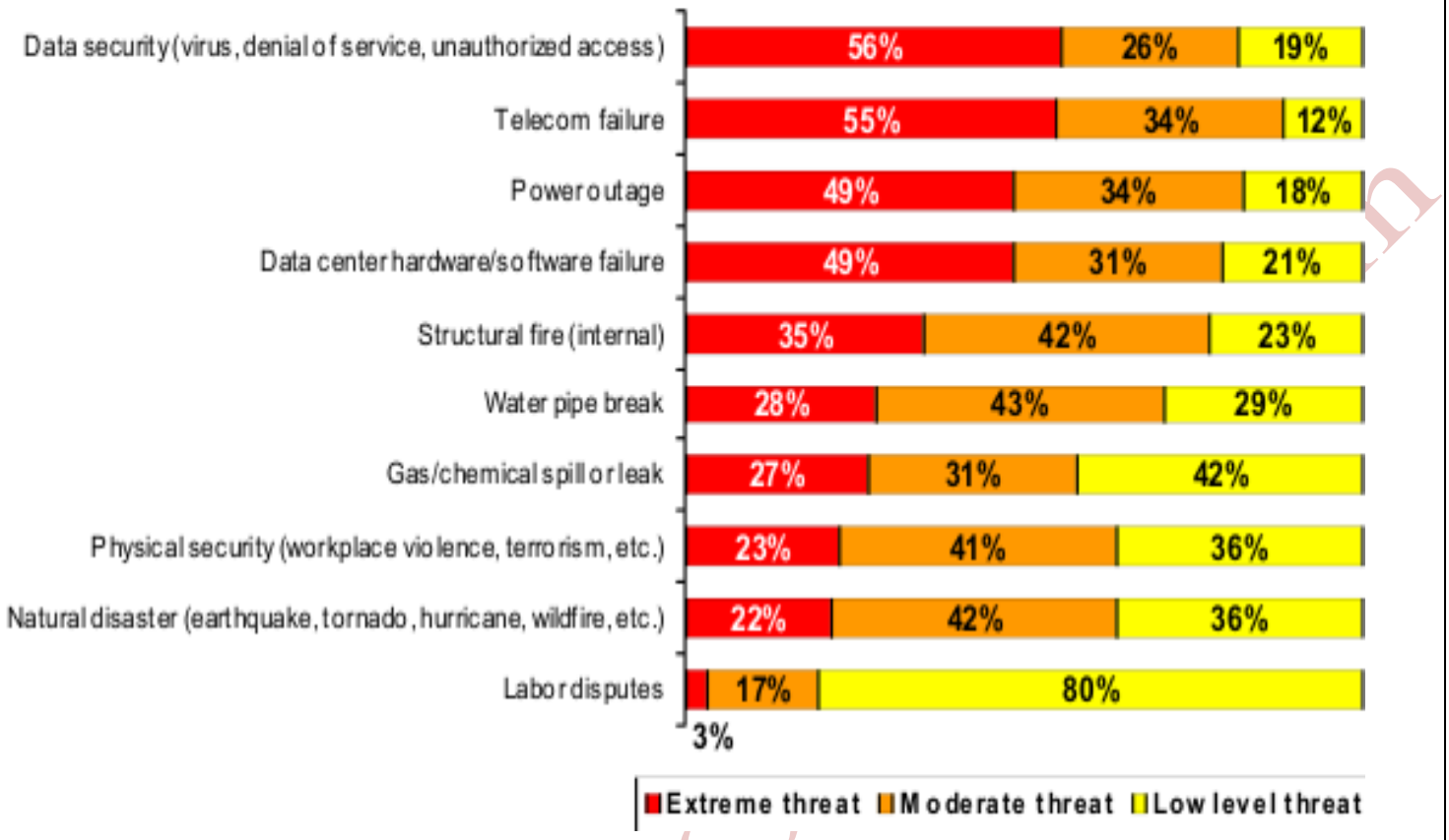
- What is the impact of downtime?
- Step-wise approach

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)

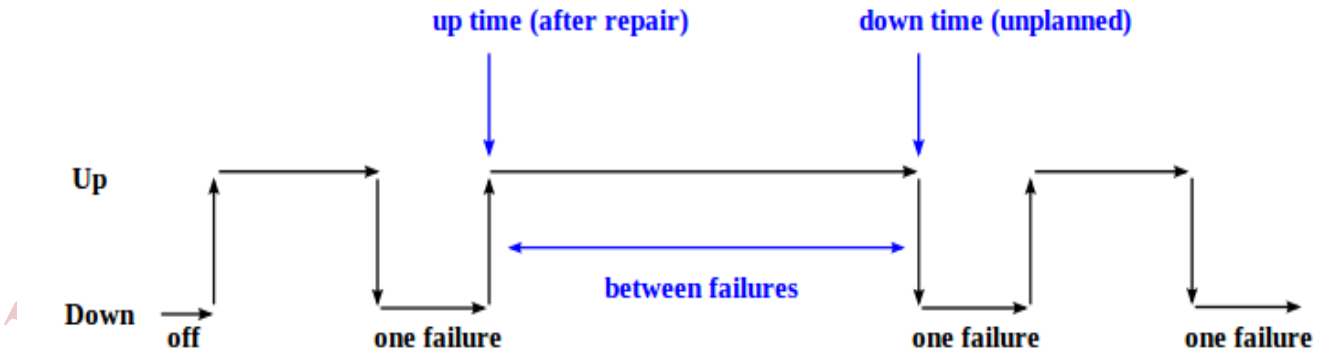
4,000/4,001 = 99.98% availability

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

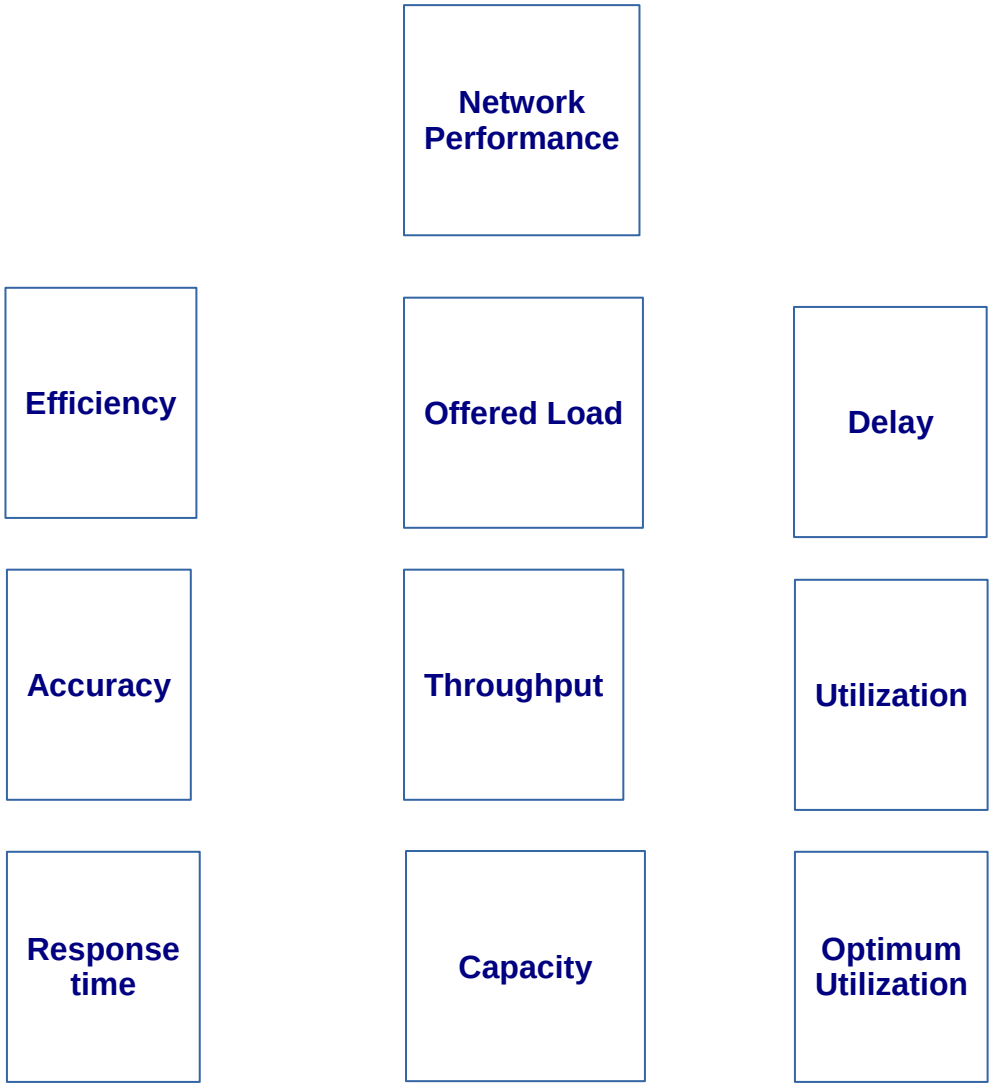
- How to define network performance?
- Performance factors

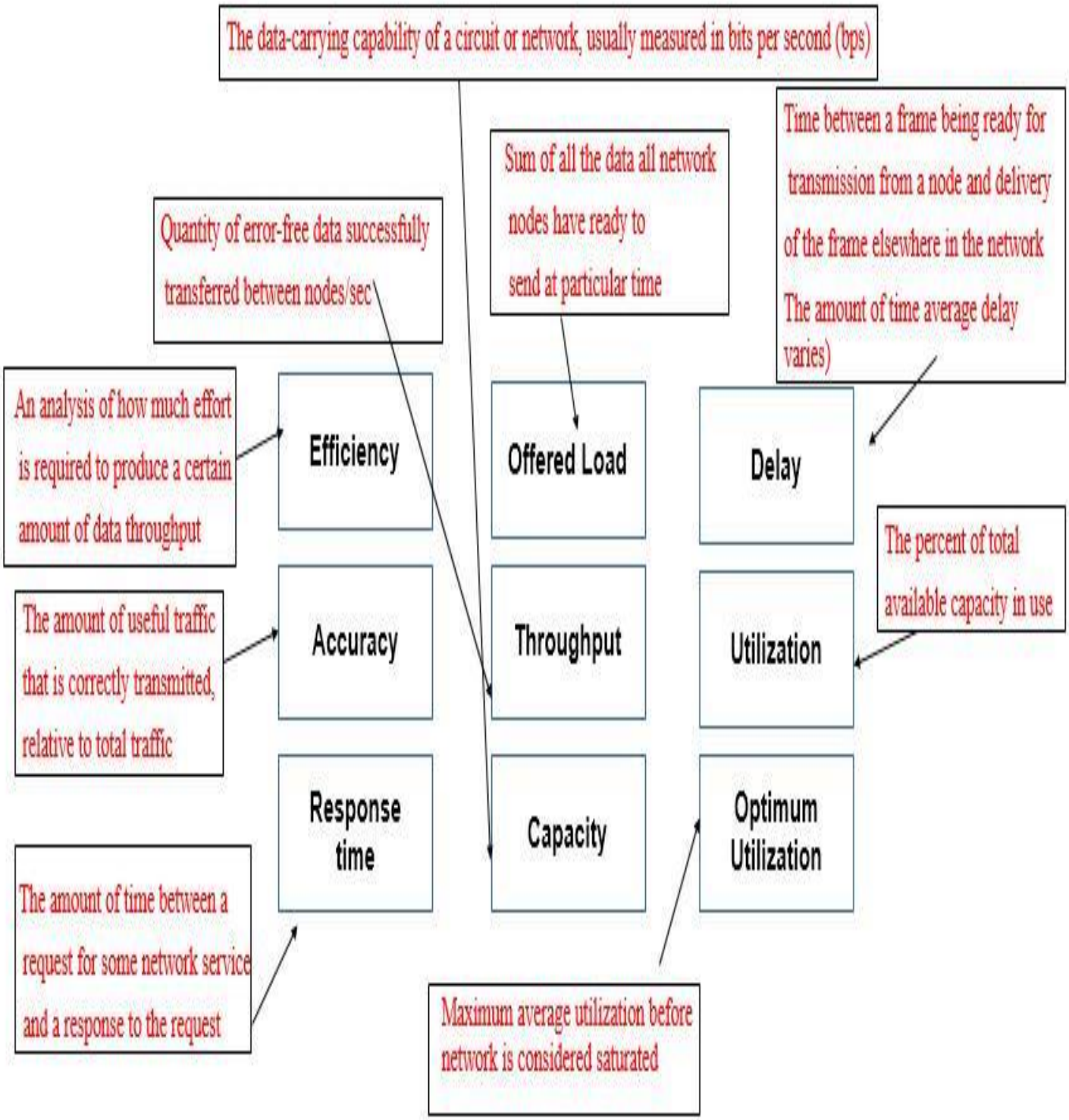
Composite metric that is end-to-end

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

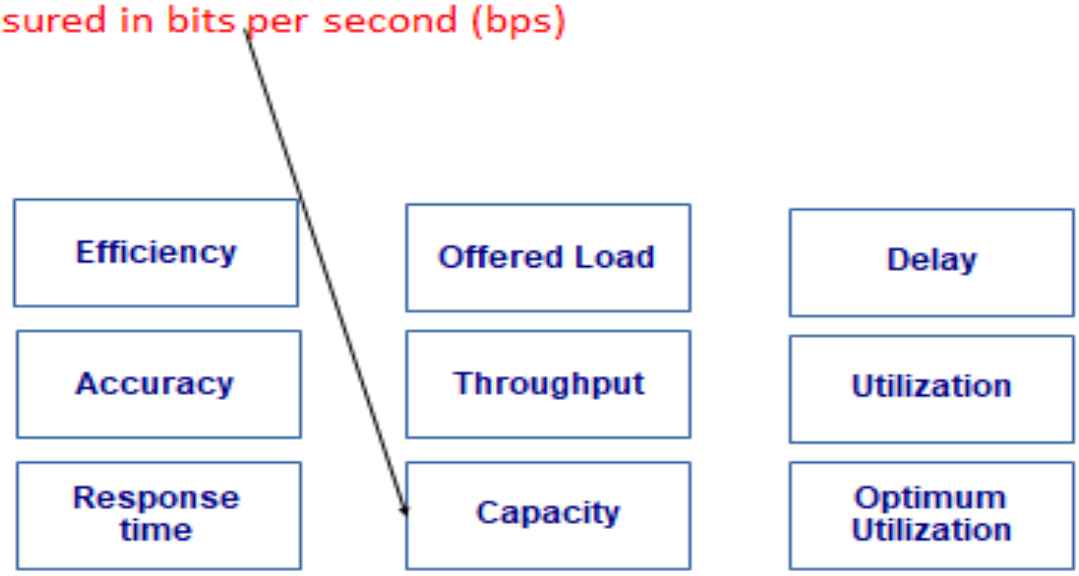
QoE—Network Performance




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

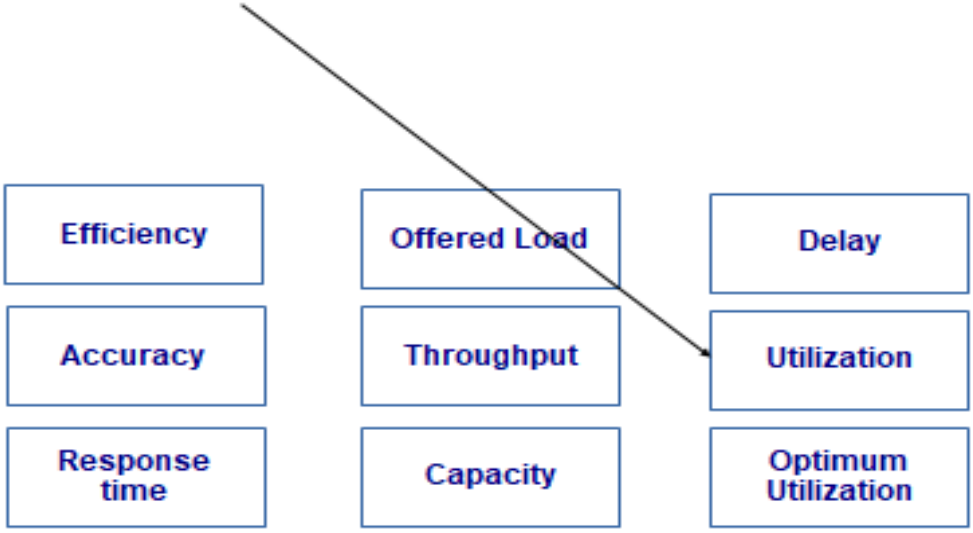
QoE—Network Performance

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



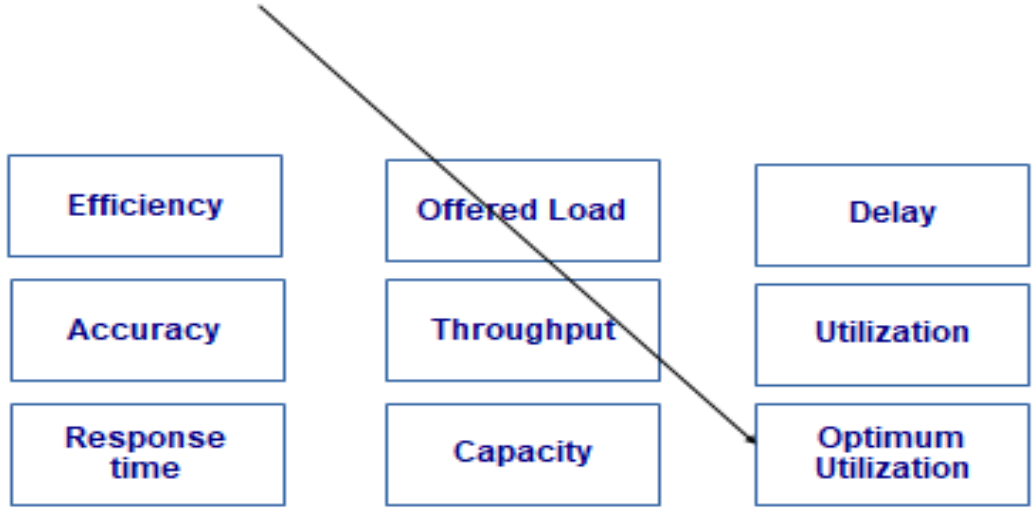
QoE—Network Performance

The percent of total available capacity in use



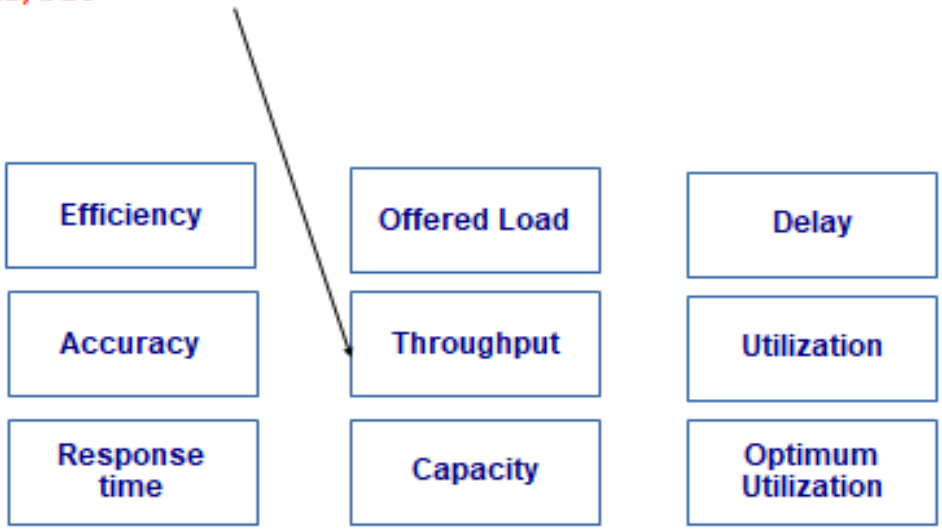
QoE—Network Performance

Maximum average utilization before network is considered saturated



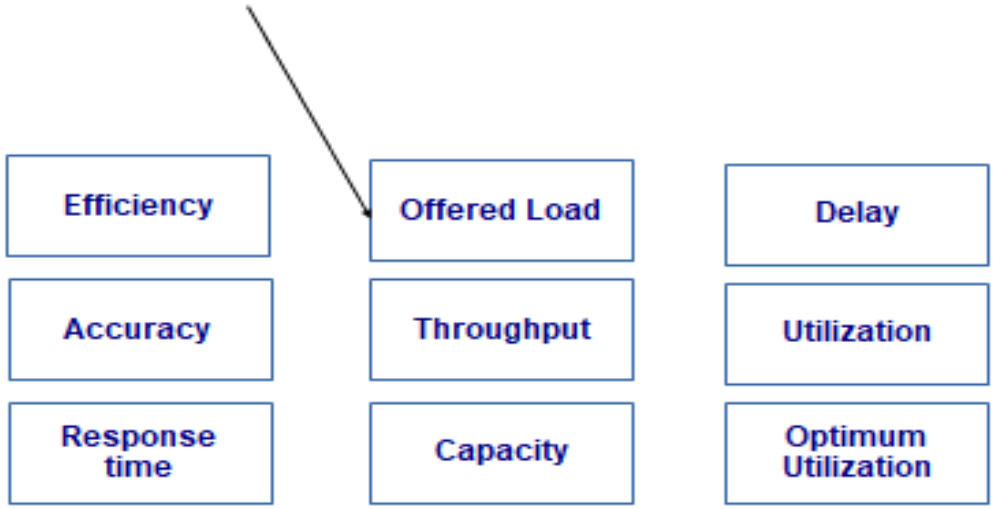
QoE—Network Performance

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



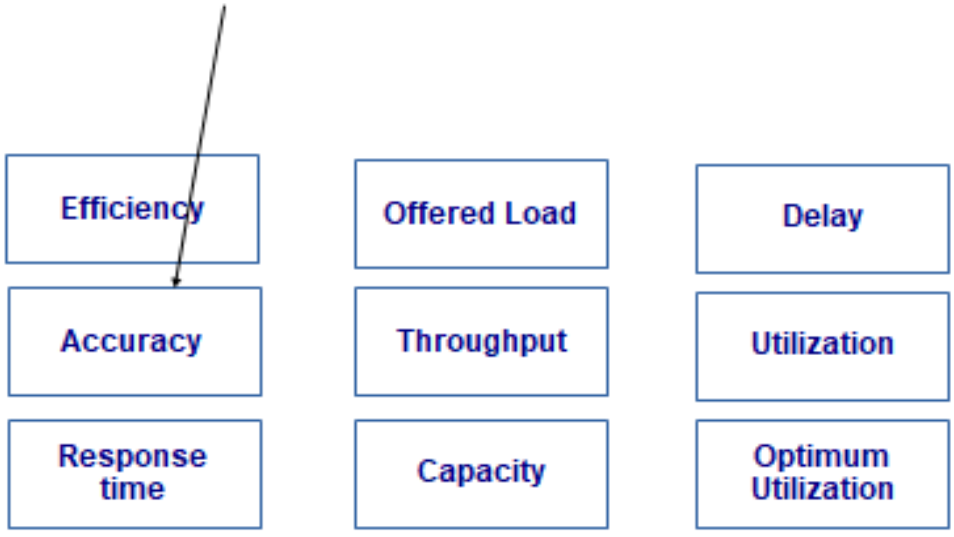
QoE—Network Performance

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



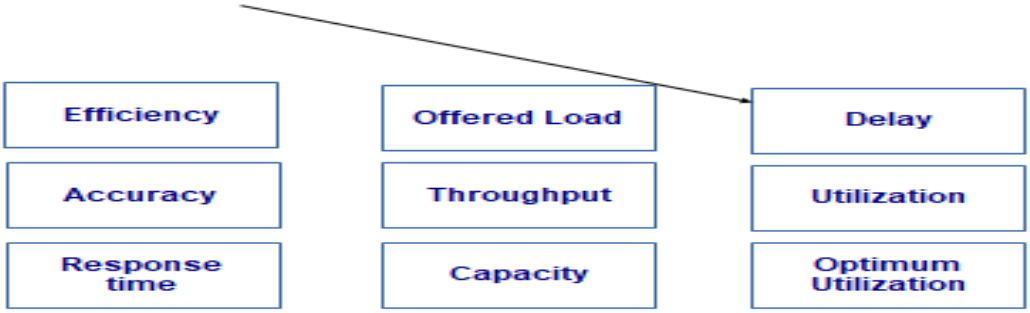
QoE—Network Performance

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

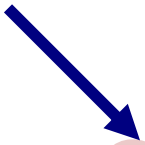
TOPIC 68

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

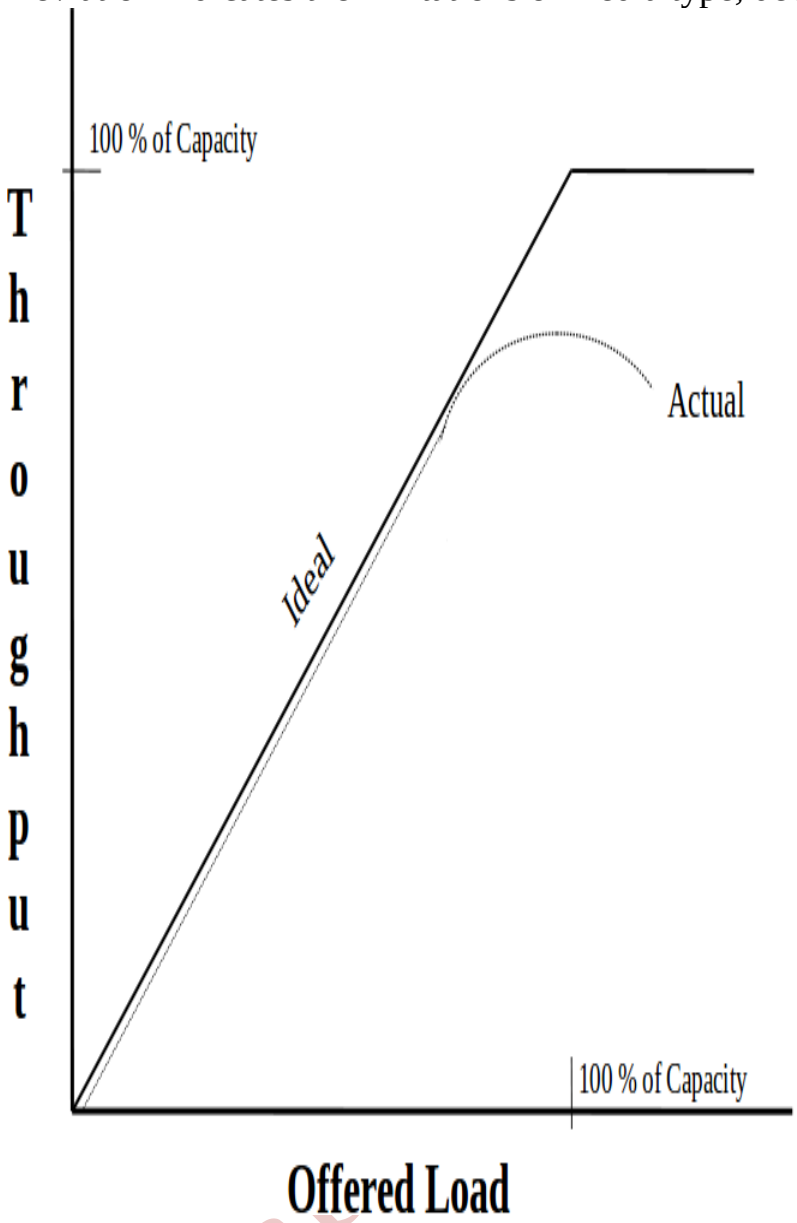
Scalability	**Availability**	**Performance**	**Security**
Manageability	**Usability**	**Adaptability**	**Affordability**

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

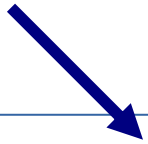
QoE—Throughput of devices

In this module

We shall understand

- What is throughput?
- Throughput vs offered load

Simulation of devices and specifications is vendor specific



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

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

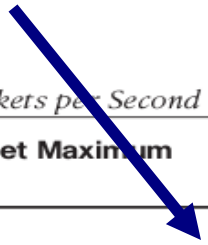


Table 2-1 *Theoretical Maximum Packets per Second (pps)*

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

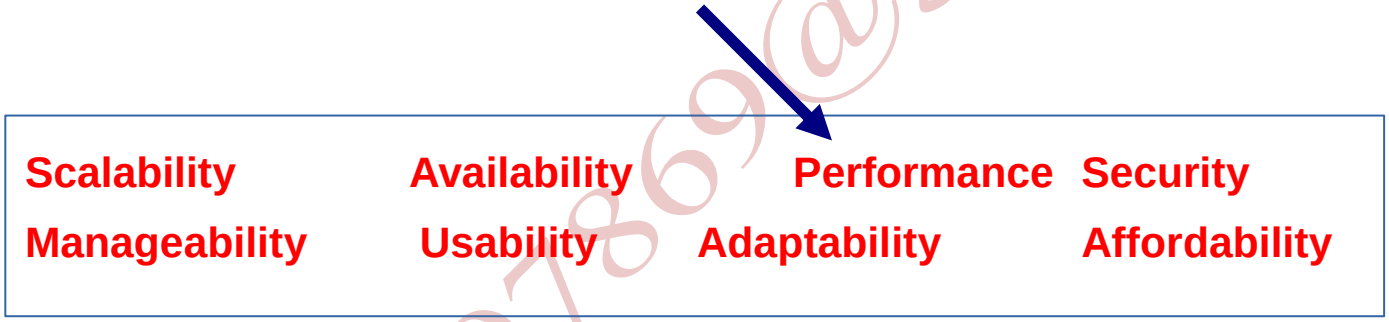
QoE—Application Layer Throughput

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:

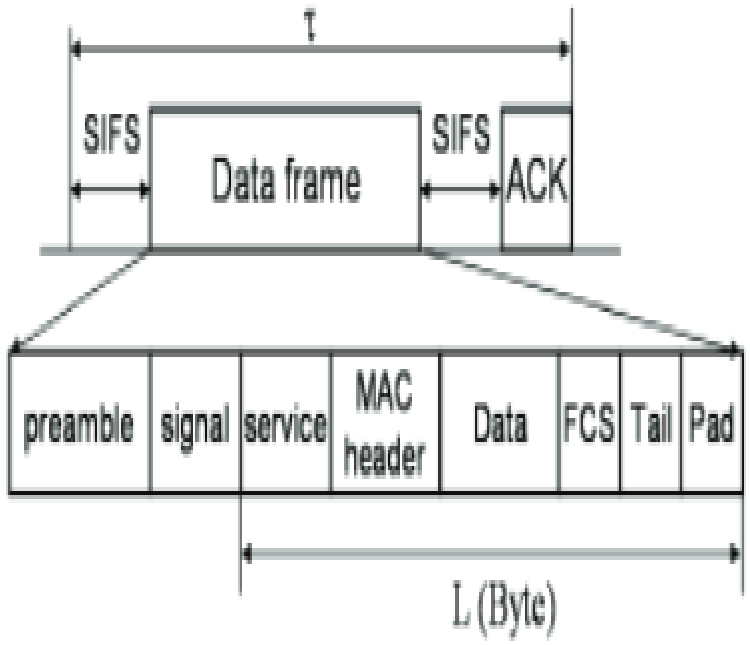
  - Disk-access speed
  - Disk-caching size
  - Device driver performance
- Computer bus performance (capacity/arbitration)

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

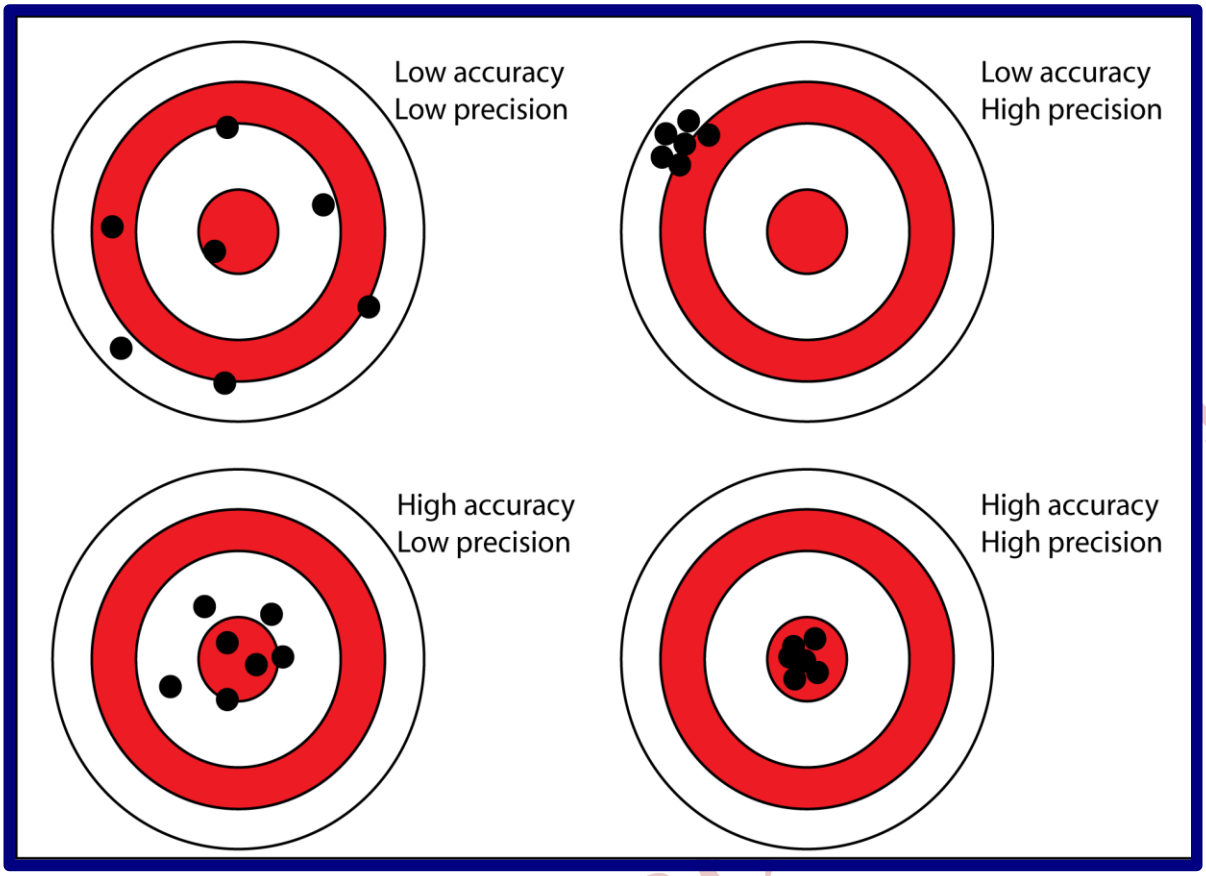
We shall understand

- 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

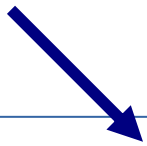
QoE—Efficiency

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

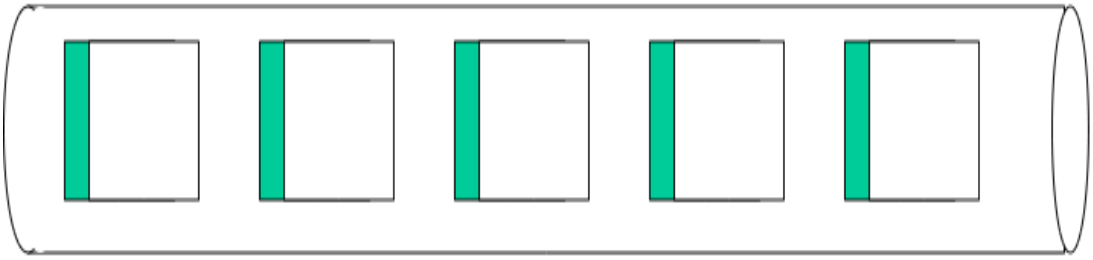
Factors affecting efficiency (1 of 3)

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

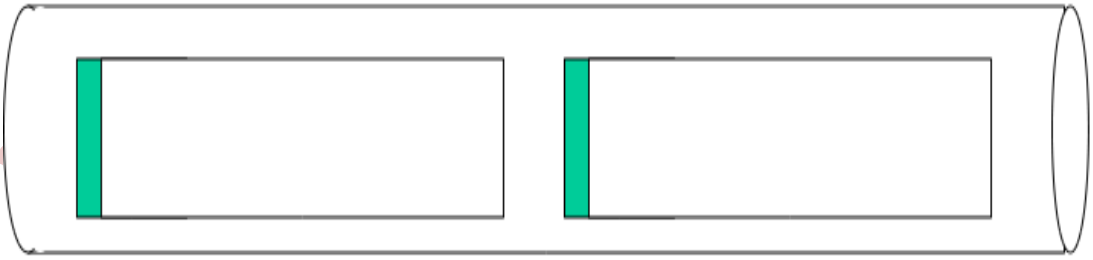
Factors affecting efficiency (2 of 3)

- 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)

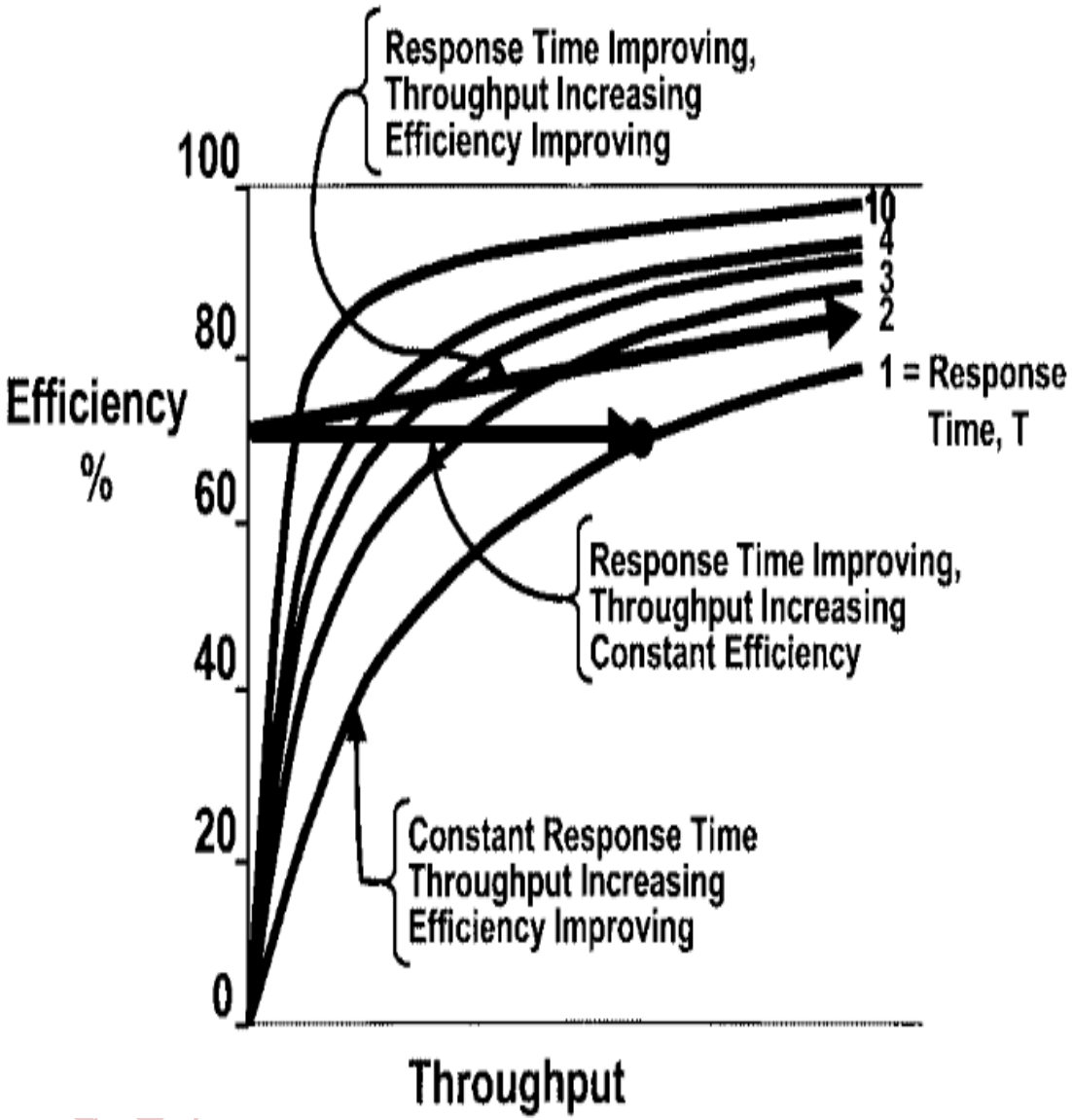


Large Frames (More Efficient)



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

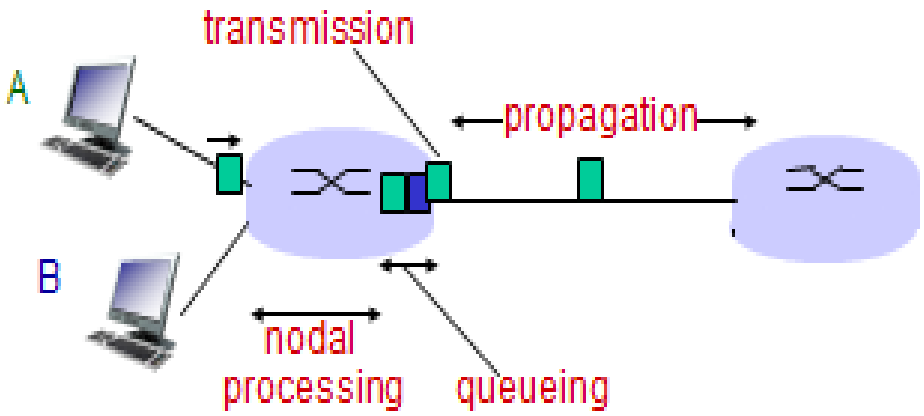
Scalability	**Availability**	**Performance**	**Security**
Manageability	**Usability**	**Adaptability**	**Affordability**

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

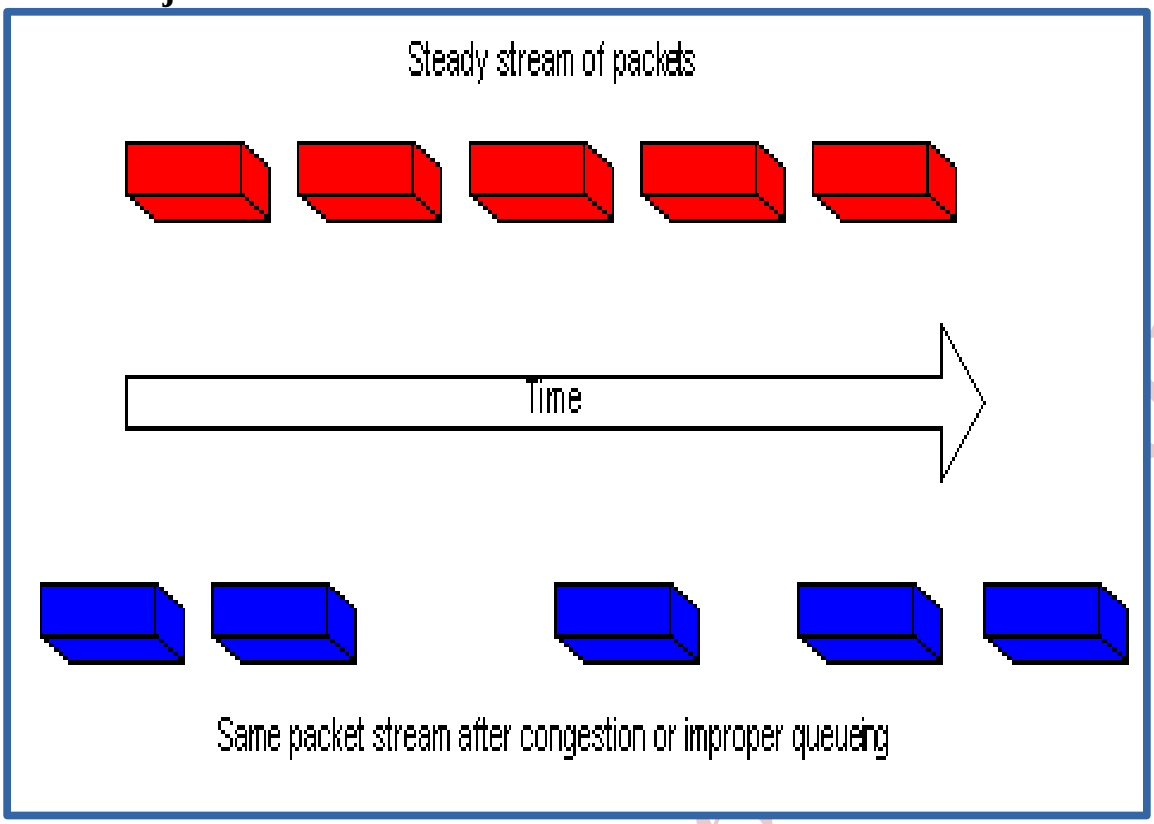


$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

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 (1 of 3)

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

Causes of Delay (2 of 3)

- Router delay
  - Look-up, router architecture, configuration
  - Software features that optimize the forwarding of packets
- NAT, IPSEC, QoS, ACL

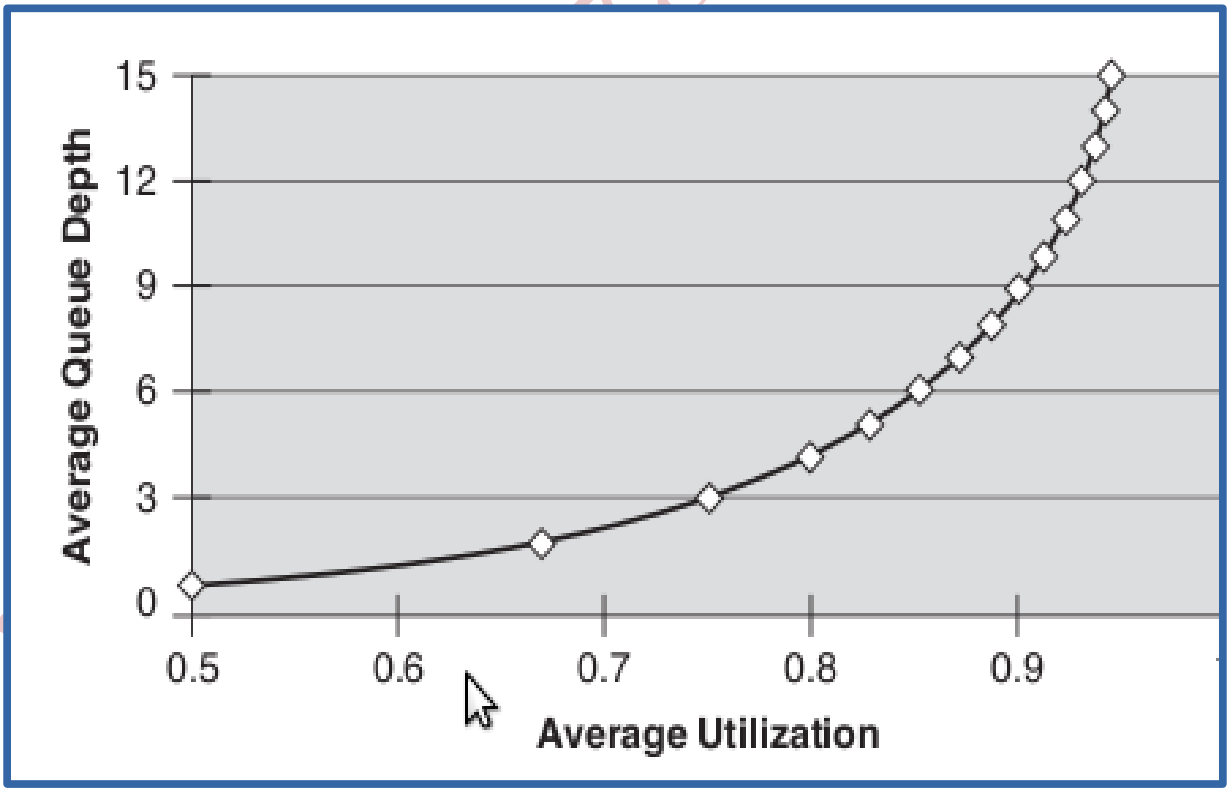
Causes of Delay (3 of 3)

- Queuing delay
  - Dependent upon utilization

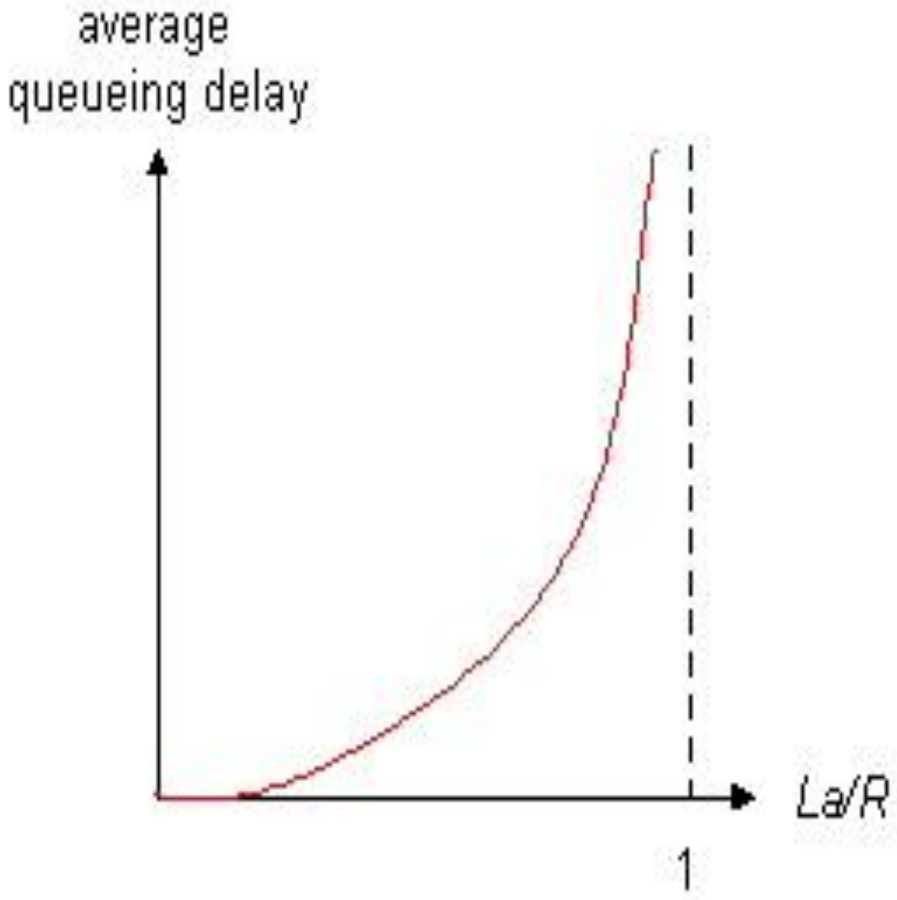
Formula

Queue depth = Utilization/(1 – Utilization)

Queue Depth vs Utilization



Implications of queuing delay



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 (1 of 2)

- 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 (2 of 2)

- 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 (1 of 4)

- Jitter is quantified in two ways
- Delay jitter
  - — bounds maximum difference in total delay of different packets
  - — Assumes source is perfectly periodic

Jitter types (2 of 4)

- Used for Interactive communication
  - voice and video teleconferencing
- Helps to translate to maximum buffer size needed at the destination

Jitter types (3 of 4)

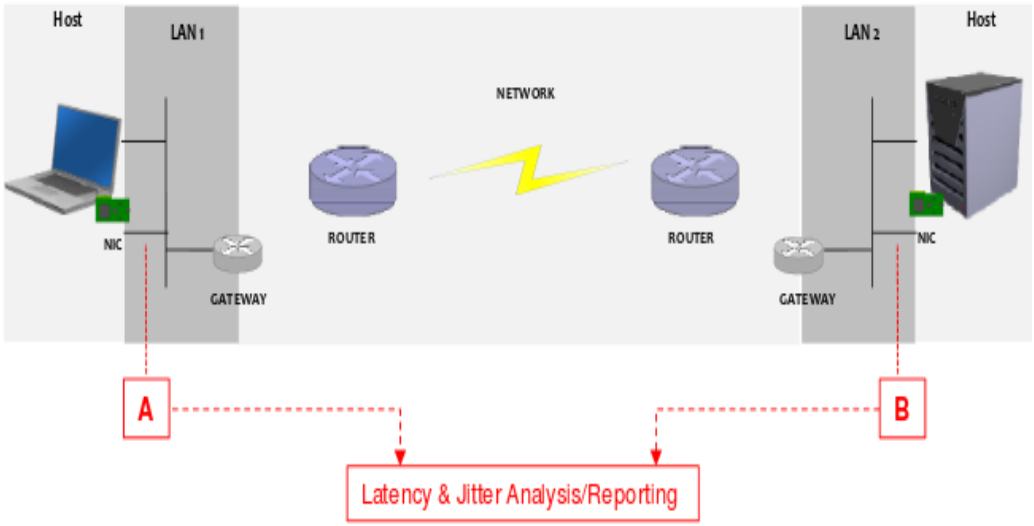
- Second measure is rate jitter
  - Bounds difference in packet delivery rates at various times
- Measures difference between minimal and maximal inter-arrival times (reciprocal of rate)

Jitter types (4 of 4)

- 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

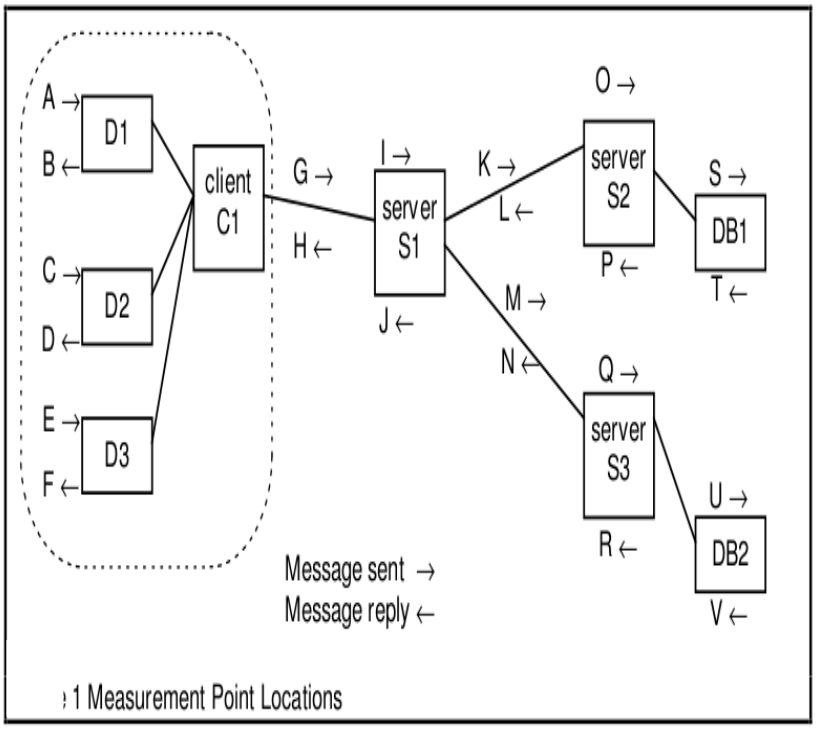
Scalability	**Availability**	**Performance**	**Security**
Manageability	**Usability**	**Adaptability**	**Affordability**

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.

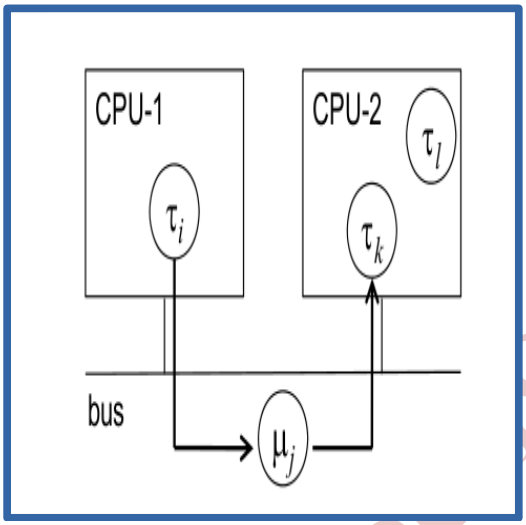


1 Measurement Point Locations

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

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

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

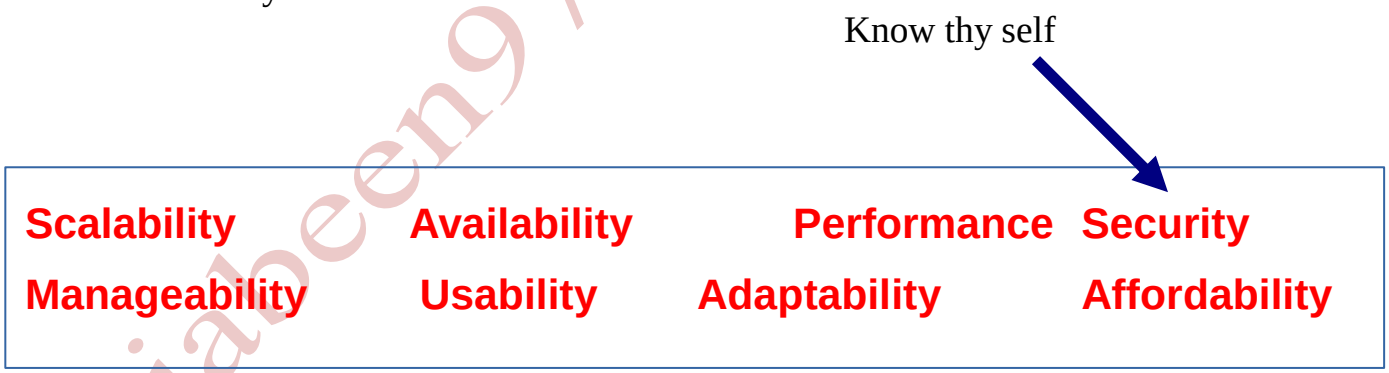
TOPIC 79

QoE—Identifying Network Assets

In this module

We shall understand

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



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

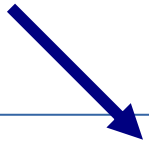
TOPIC 80

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

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

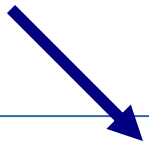
QoE—Security Requirements

In this module

We shall understand

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

Comprehensive planning is half success



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

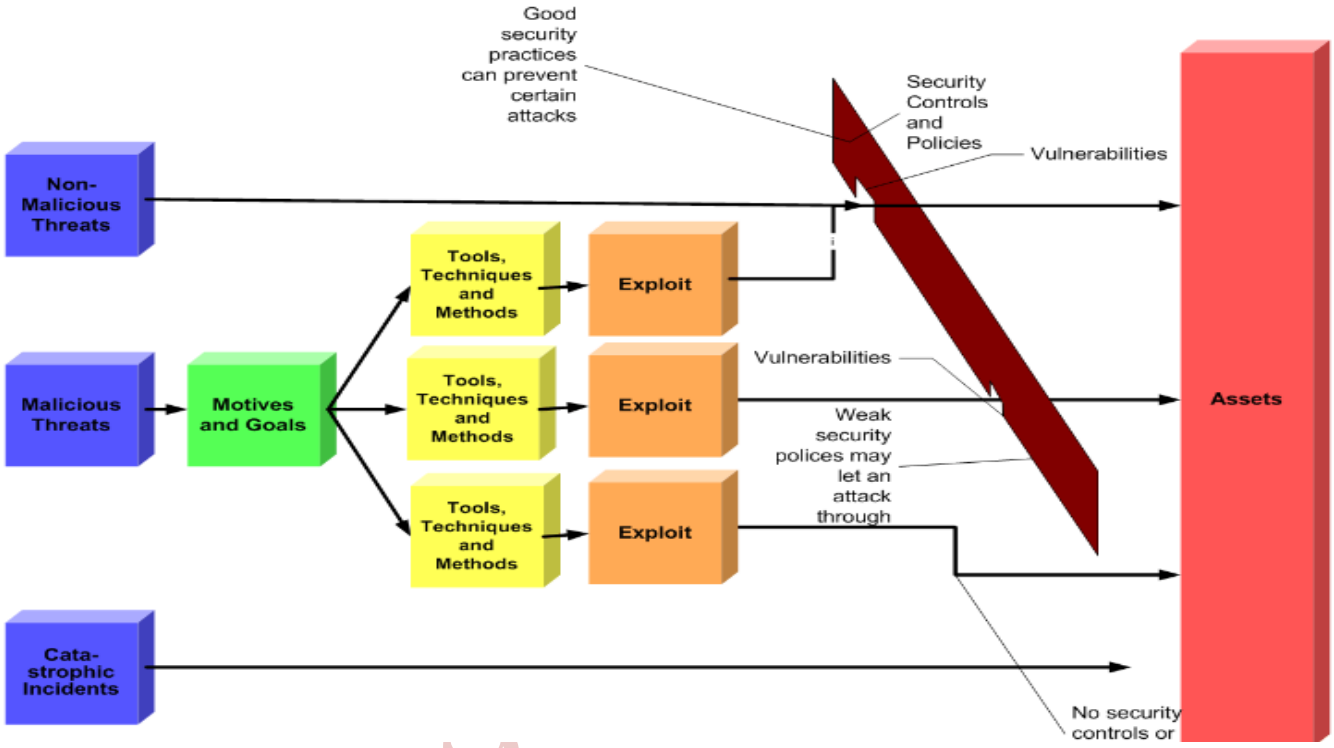
Definition

- 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

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

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

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

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

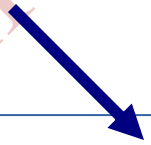
TOPIC 82

QoE—Manageability

In this module
We shall understand

- What is manageability?
- Manageability metric

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



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

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

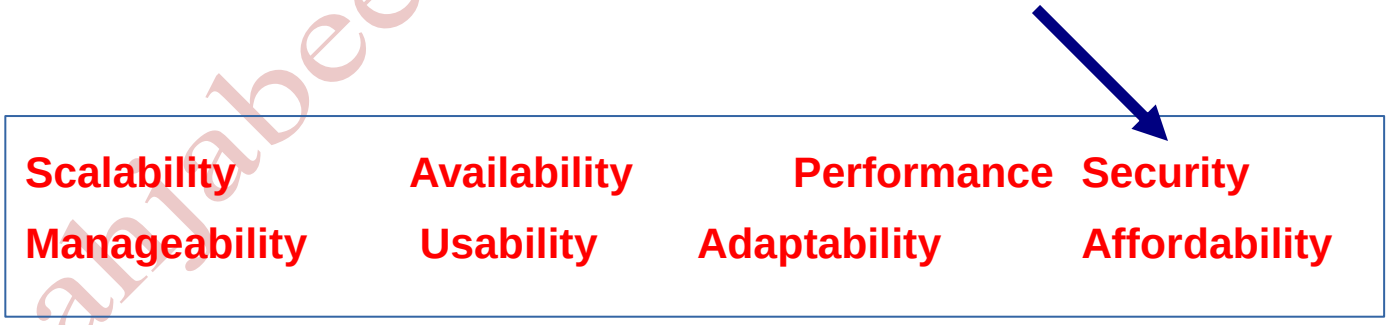
QoE—DoS Attack

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

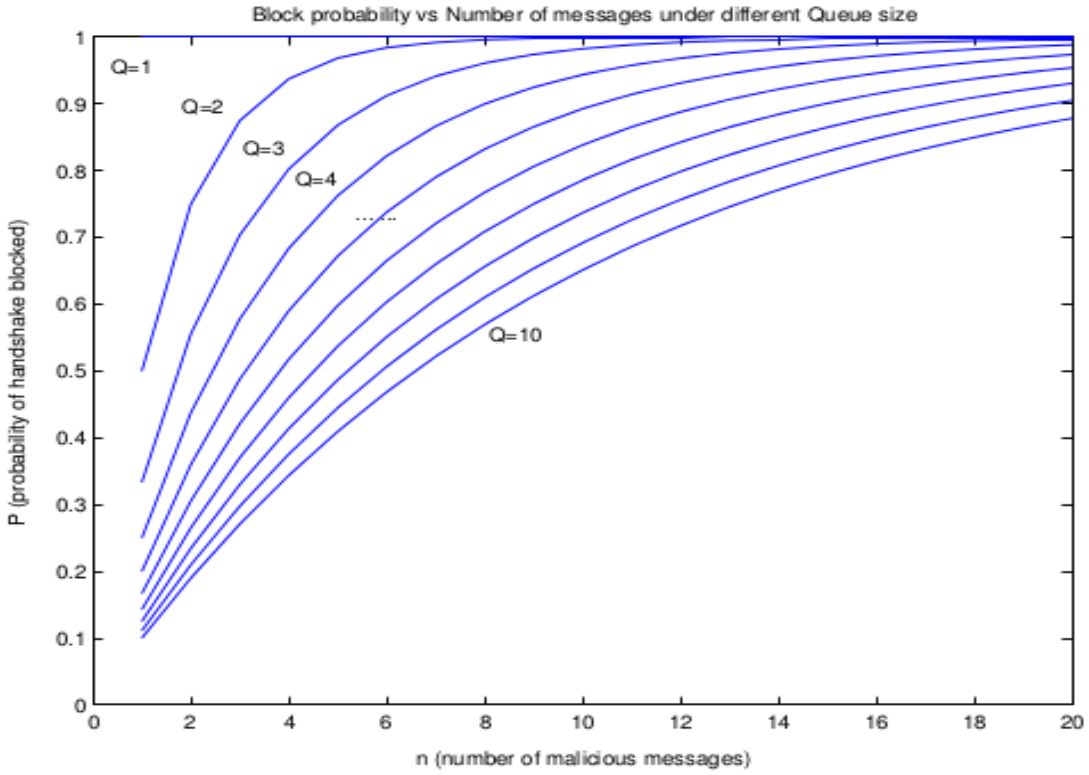
Attack success probability

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

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

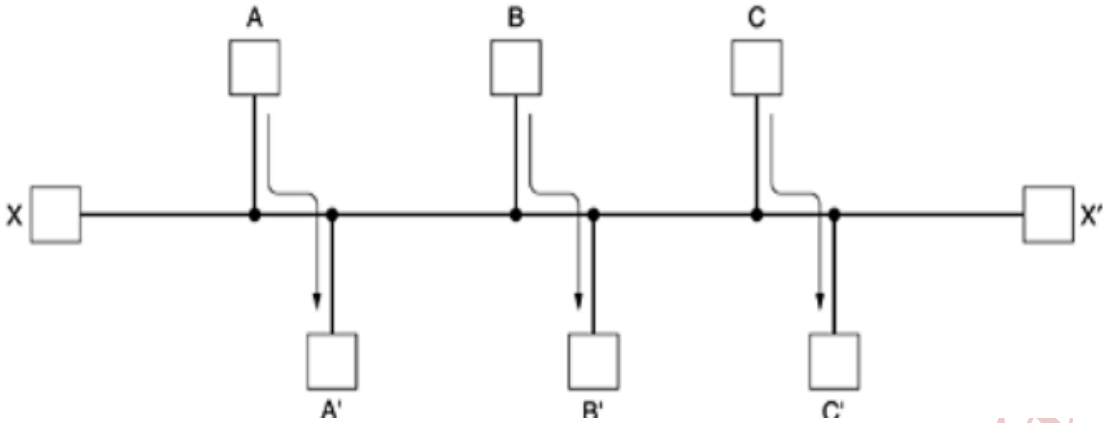
Scalability	**Availability**	**Performance**	**Security**
Manageability	**Usability**	**Adaptability**	**Affordability**

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

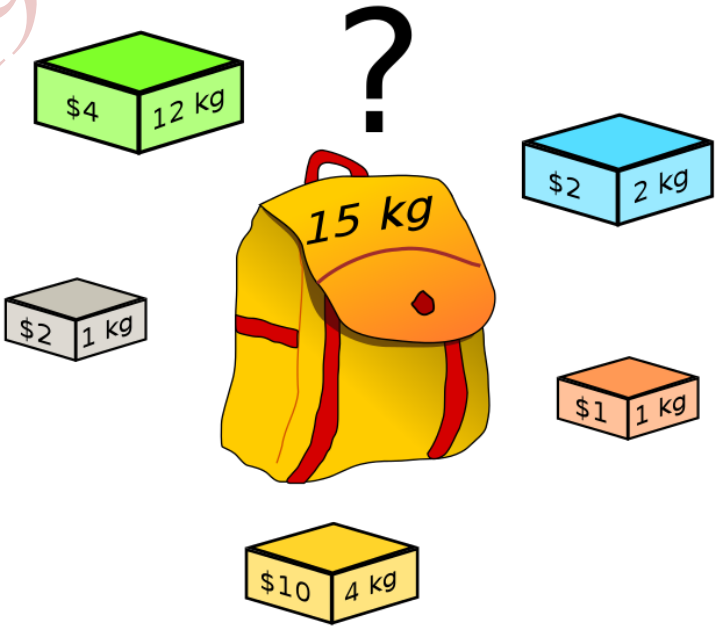


Tradeoff at Network Level

- 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!



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

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

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

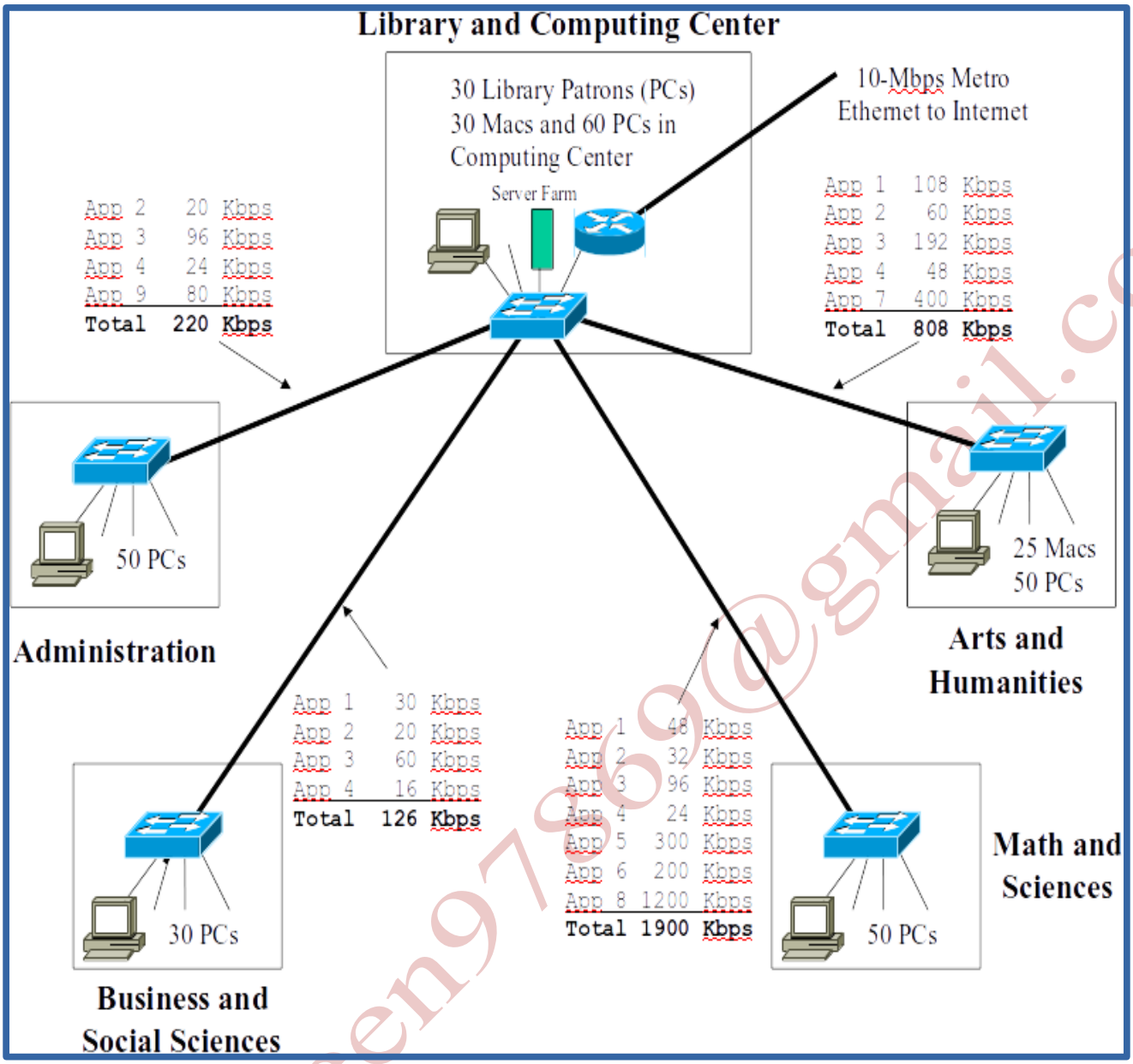
TOPIC 85

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?

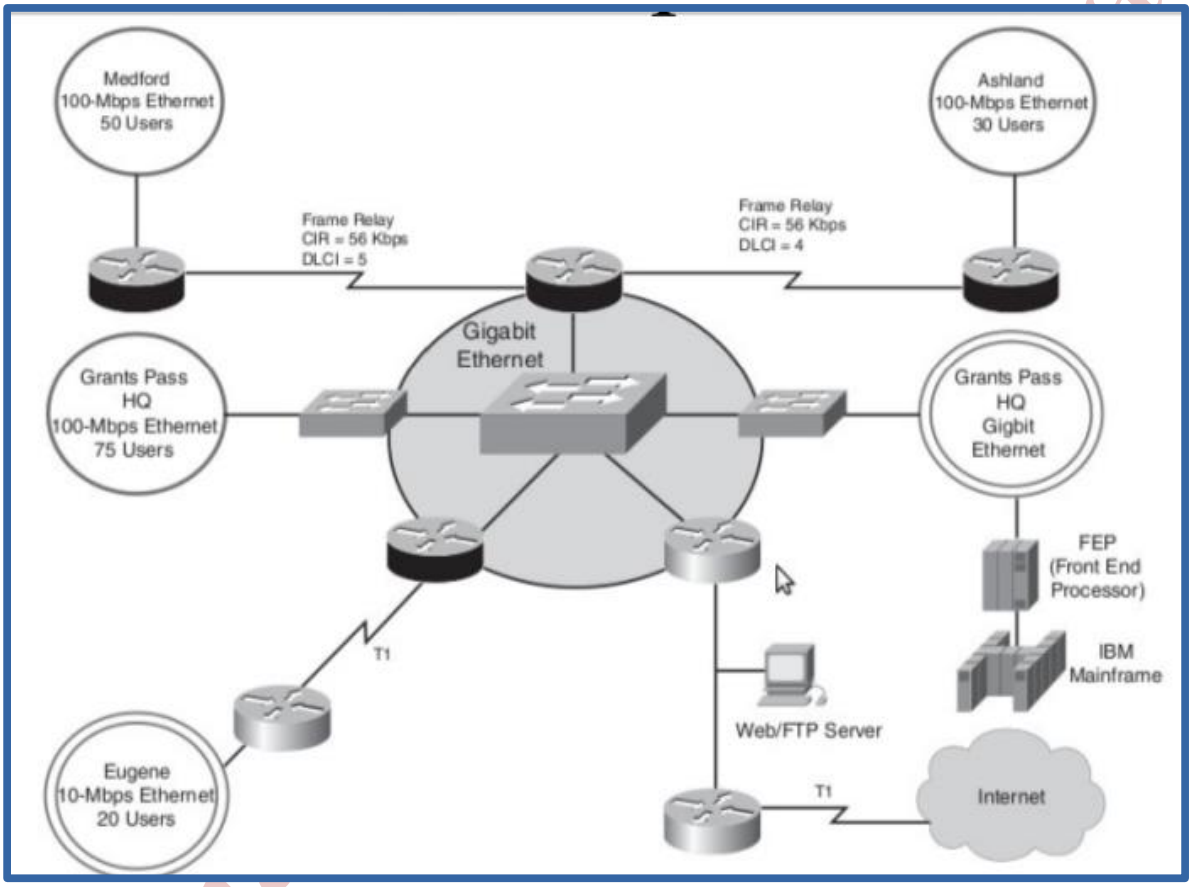
END

TOPIC 86

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?
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?

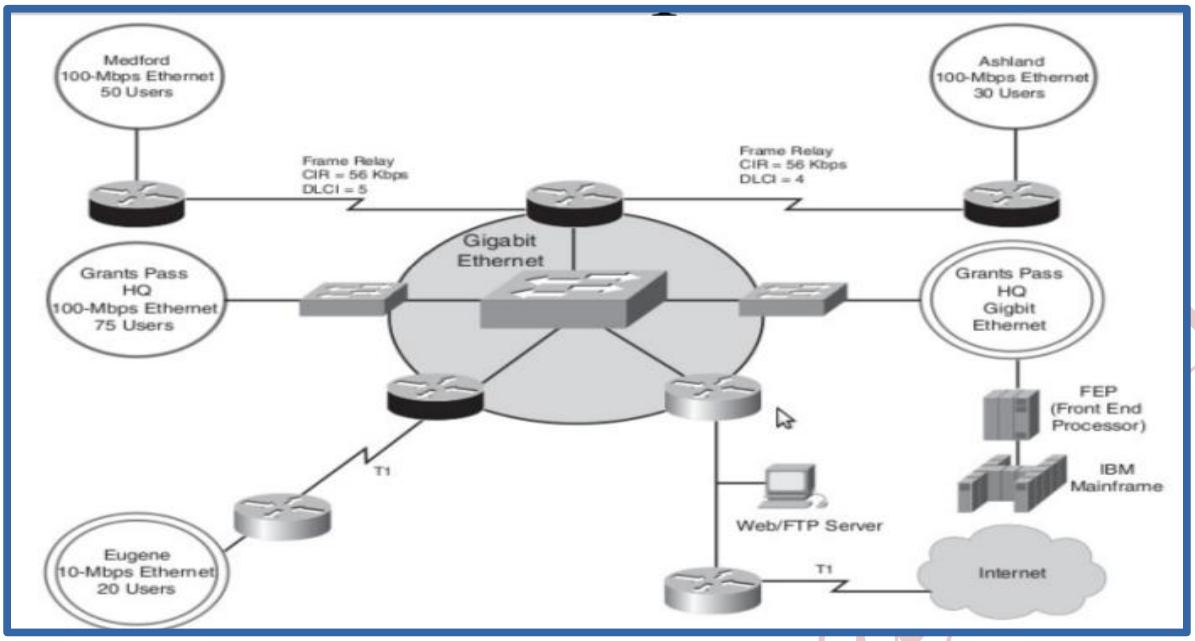
Factors affecting goodput (1 of 3)

- 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

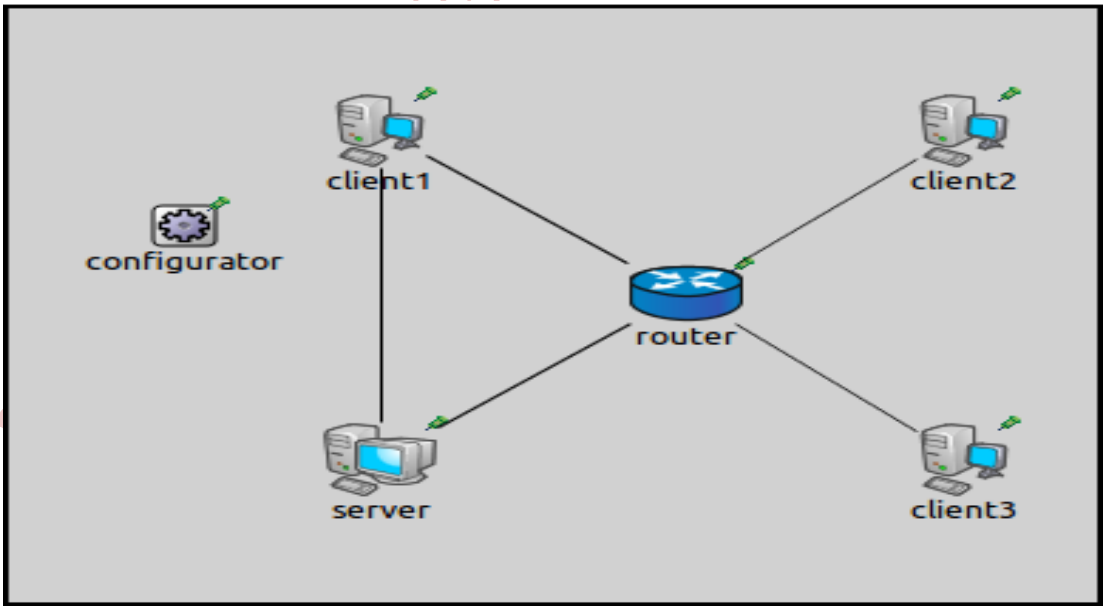


Factors affecting goodput (3 of 3)

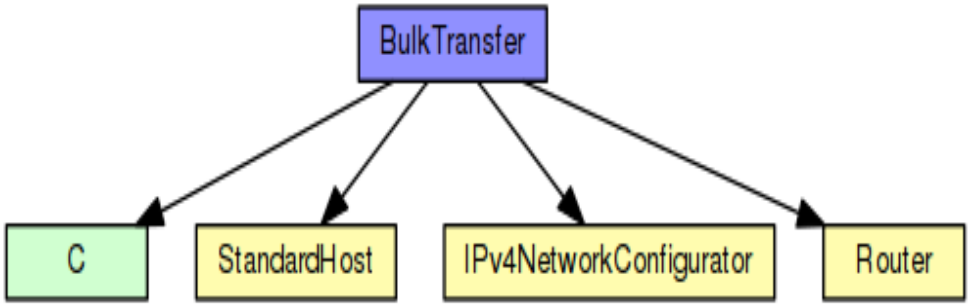
- Processor (CPU) performance
- Memory performance (access time for real and virtual memory)
- Operating system inefficiencies
- Application inefficiencies or bugs

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

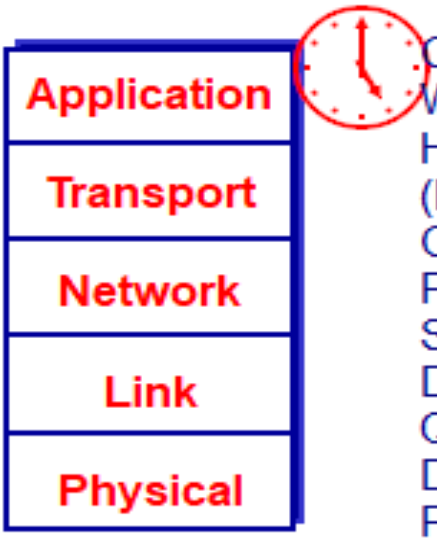
In this module

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

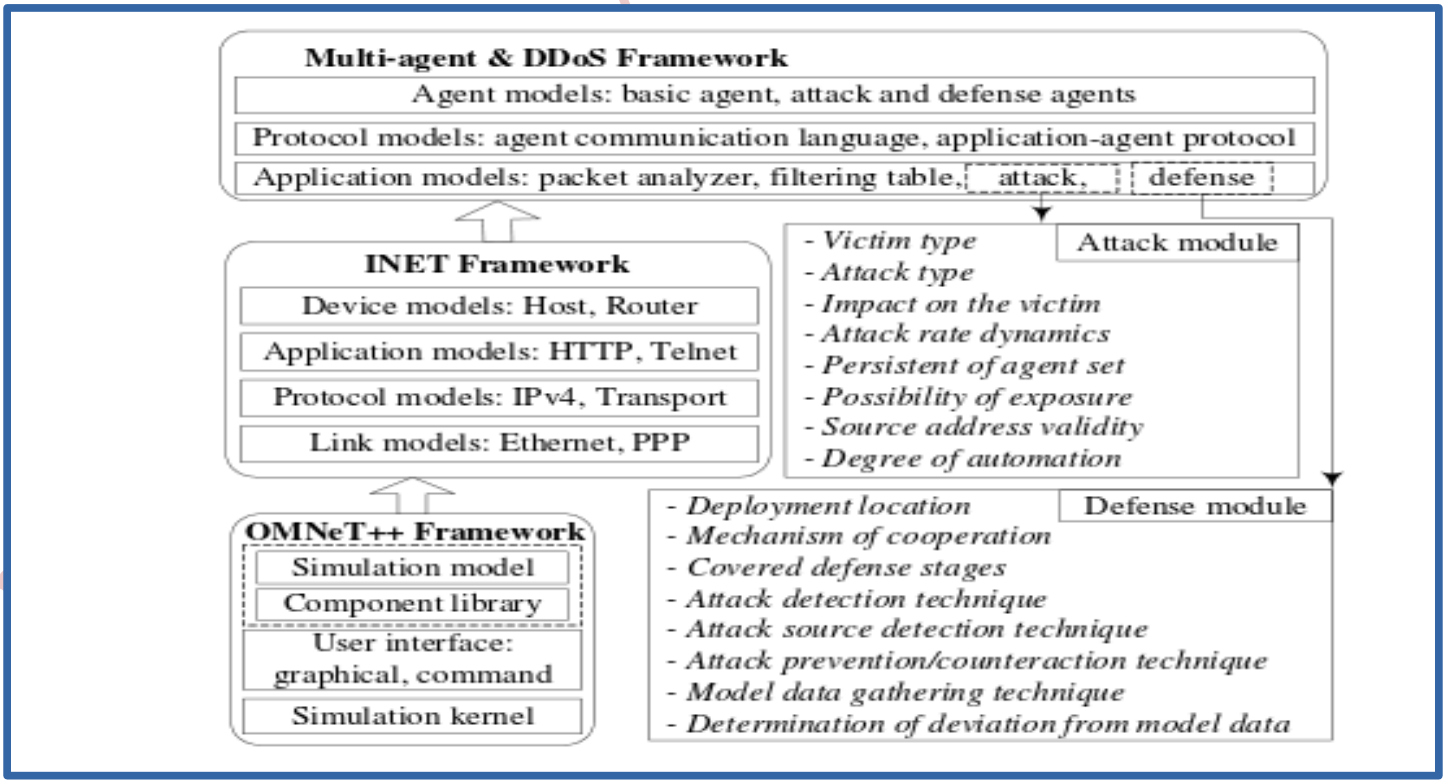
In this module
We shall understand

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

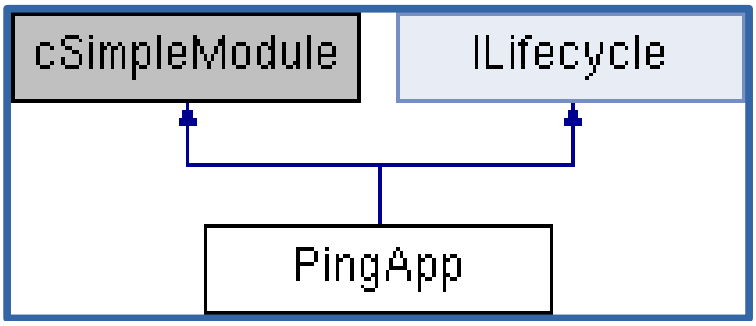
Igor Kotenko & Alexander Ulanov , “Simulation of Internet DDoS Attacks and Defense ,” ISC 2006, LNCS 4176, pp. 327–342, 2006.

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

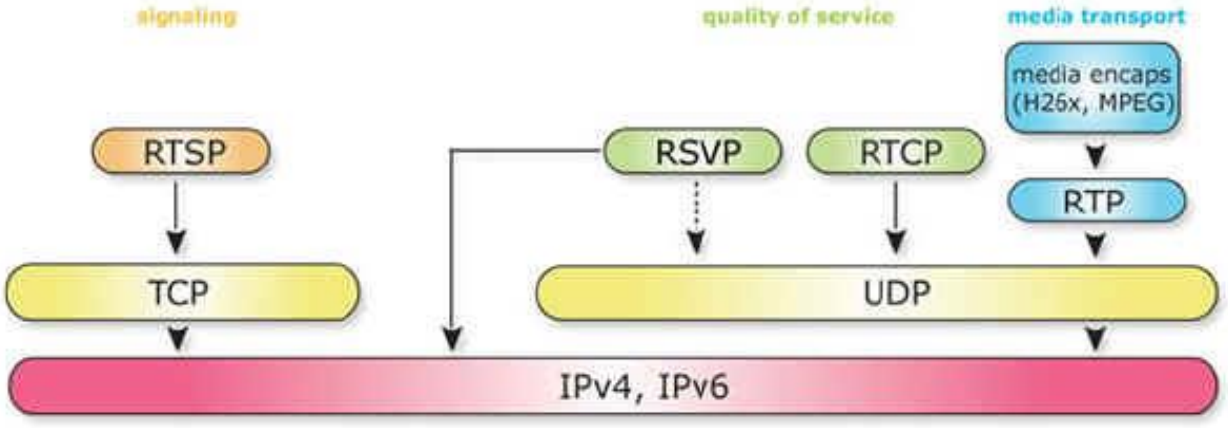
Simulate RTP with Packet Loss

In this module

We shall understand

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

Family of RTP



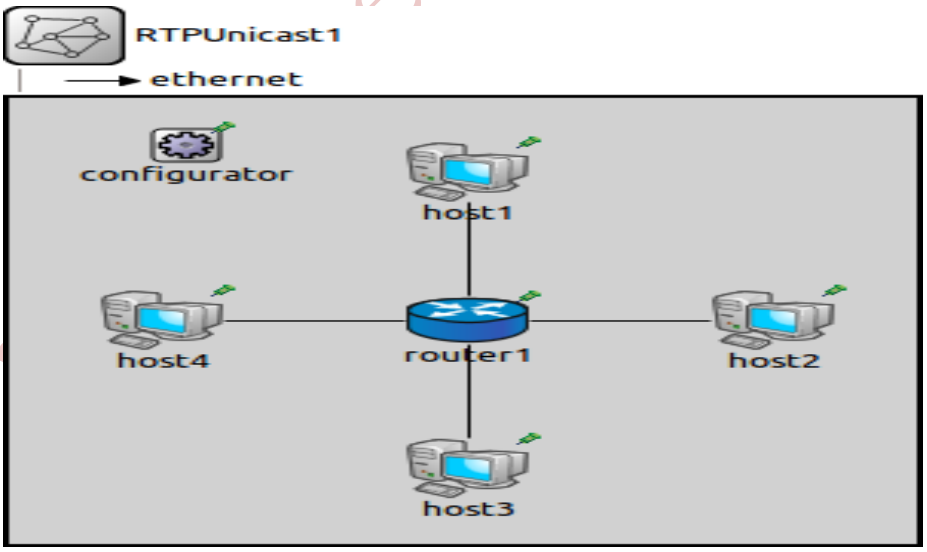
RTP

- 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

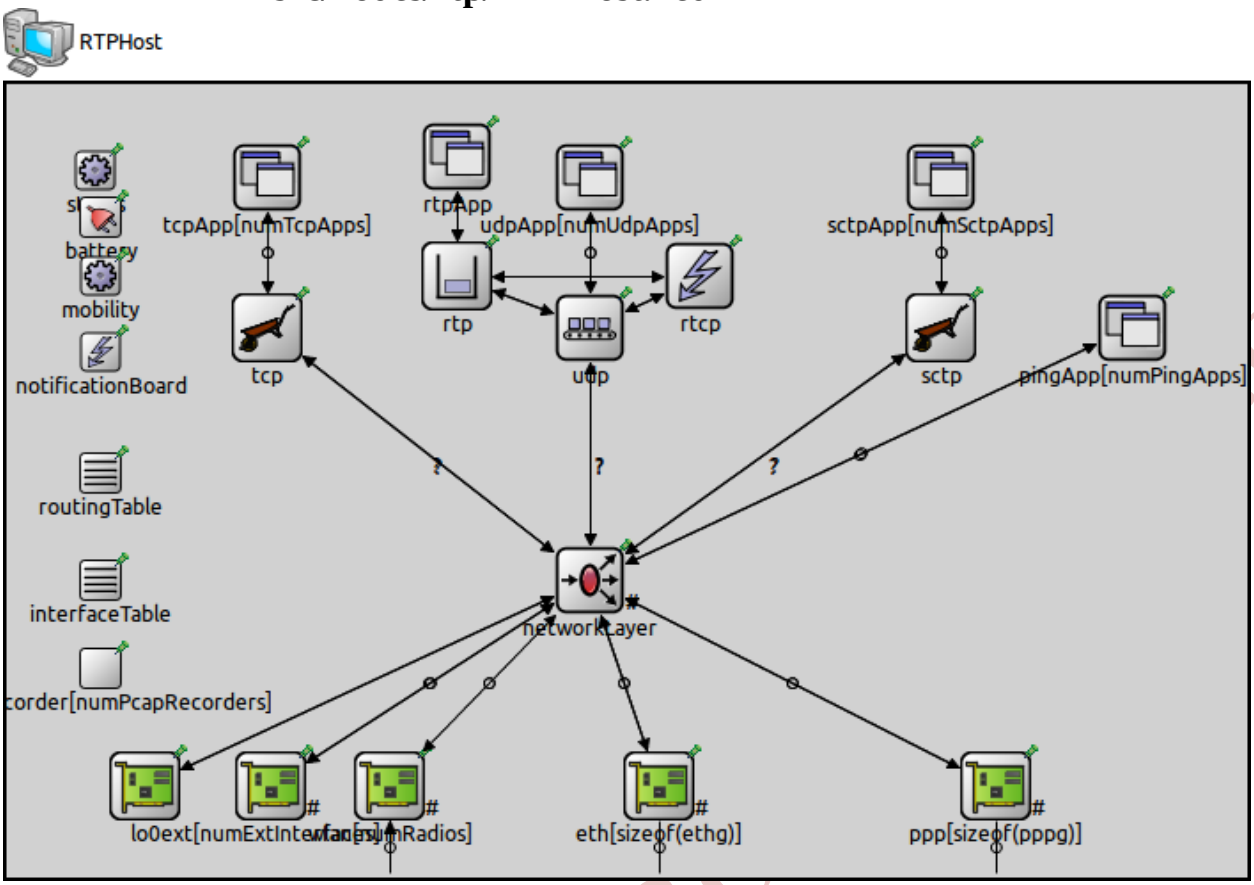
Simulate RTP with Packet Loss

Delay/Jitter Analysis Points

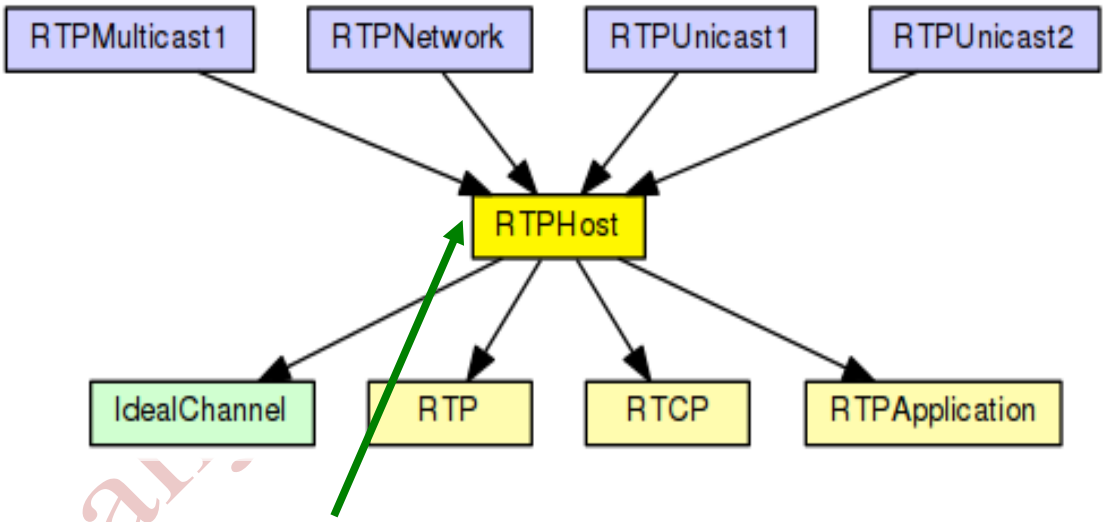
Inet for Simulating RTP
(examples/rtp/unicast1/unicast1.ned



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

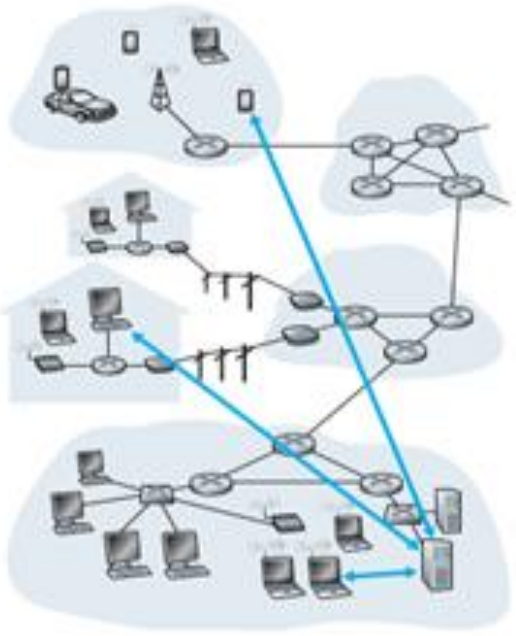
Client Server Architectures

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

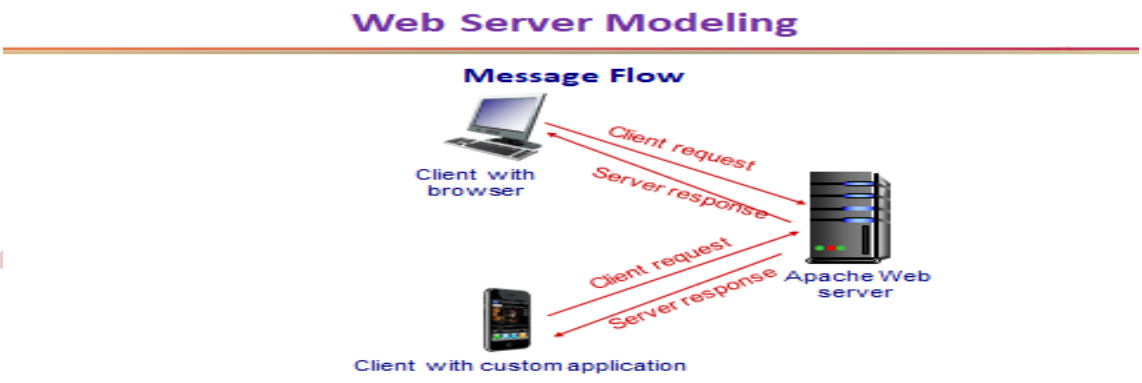
TOPIC 94

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

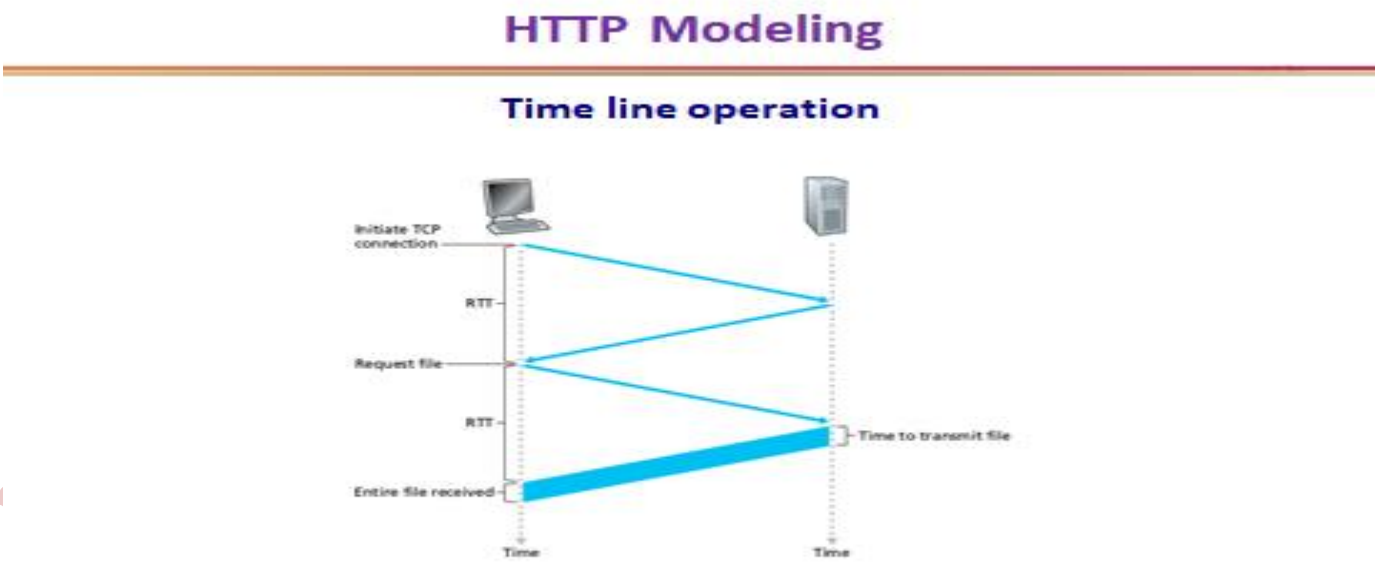
TOPIC 95

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

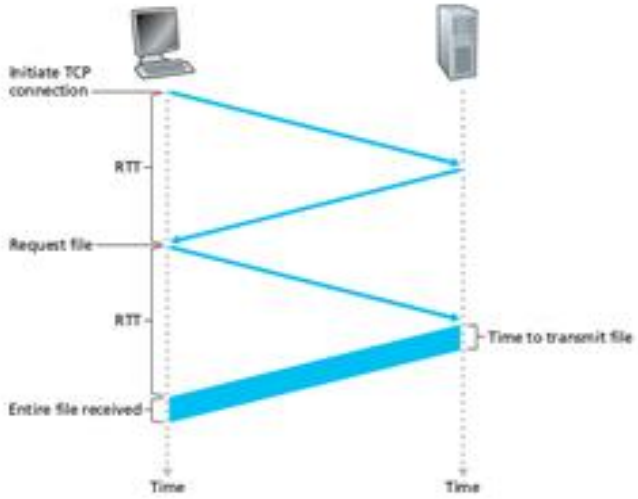
In this module

We shall understand

- What is non-persistence
- Modeling non-persistence

Non-Persistent Connections

TCP handshaking required for every object



Modeling Non-persistence

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

N*2RTT + N*Transmit time

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

END

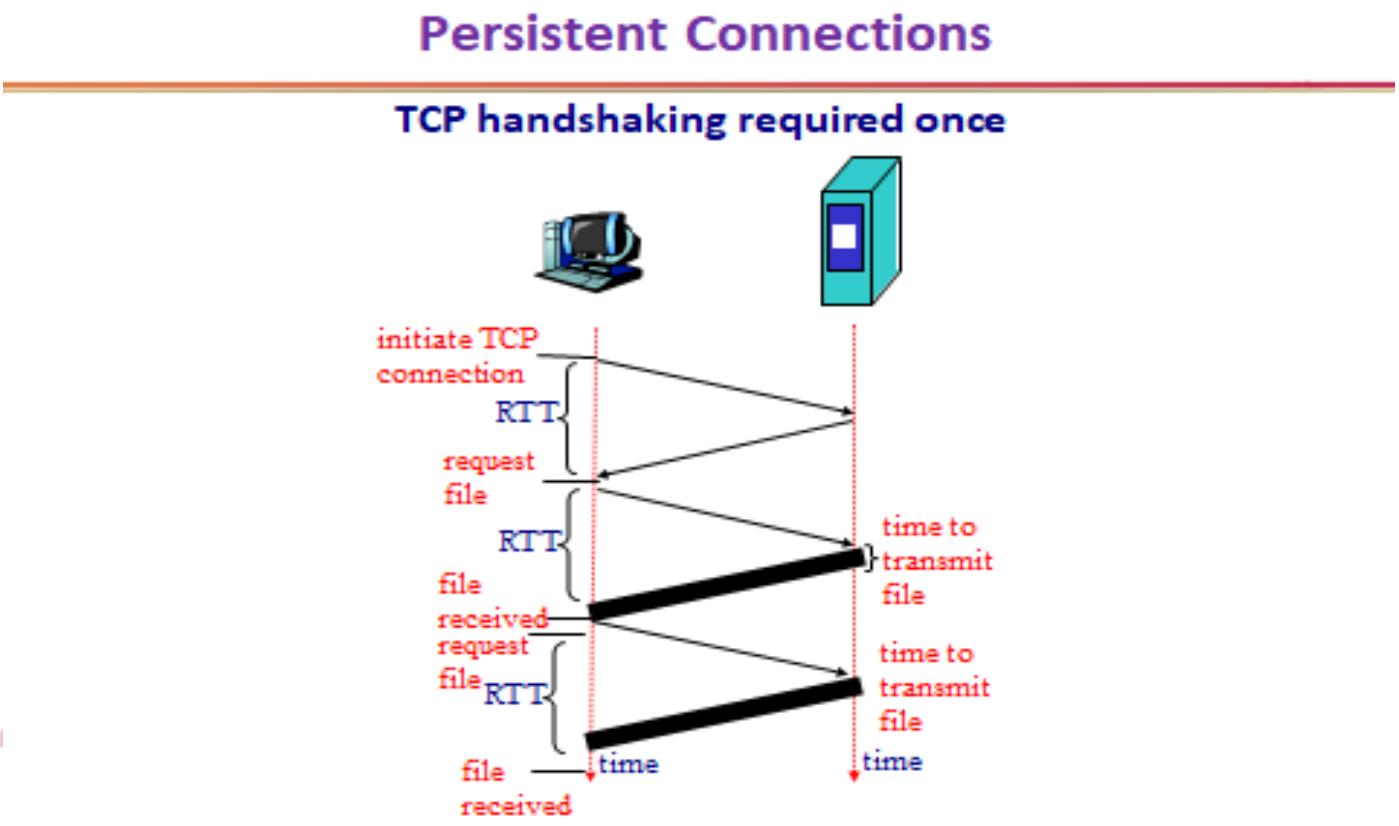
TOPIC 97

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

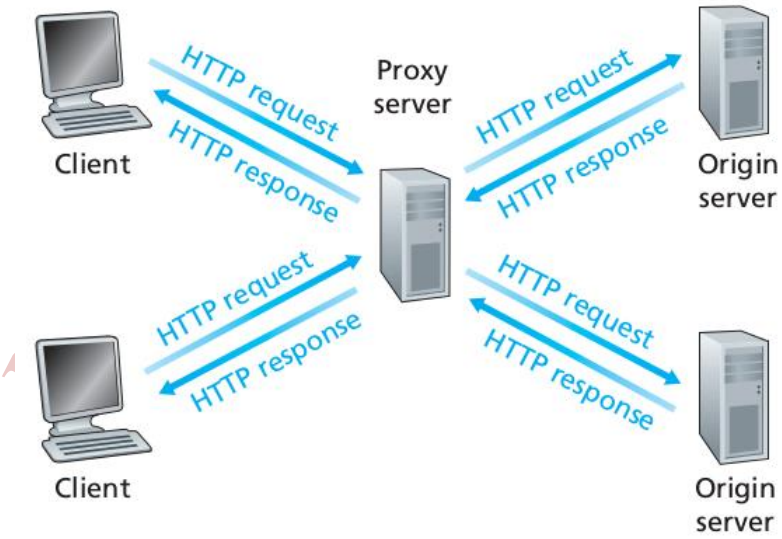
We shall understand

- 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

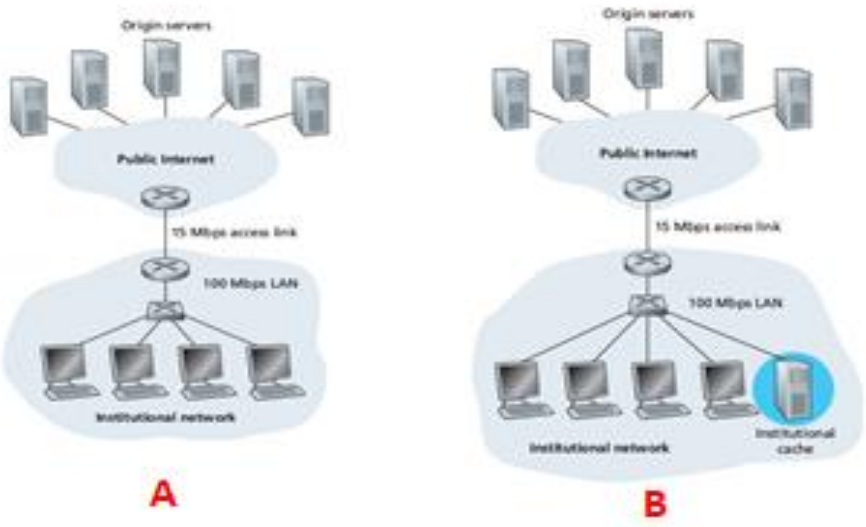


Advantages of caching

- 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

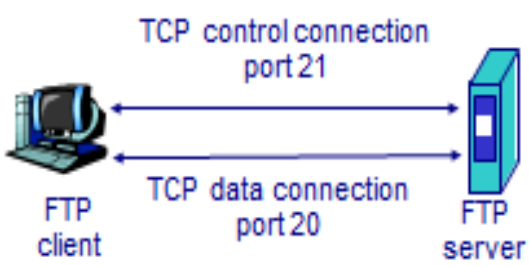
- Basic operation of FTP
- Evaluating file transfer

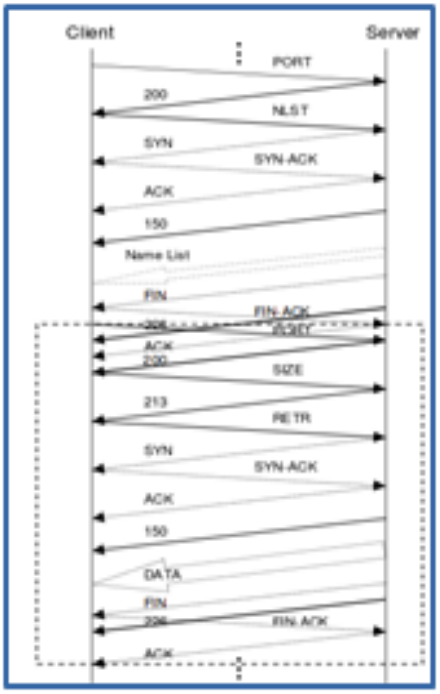
FTP operation

- 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

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

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

END

TOPIC 100

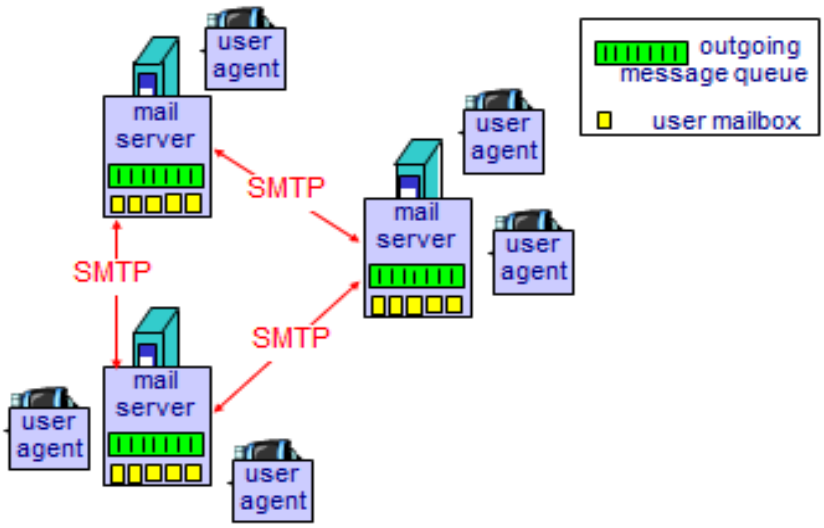
SMTP Scalability

In this module
We shall understand

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

SMTP Scalability

Entities of SMTP Architecture



Recall scalability

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

Efficiency & speed-up for SMTP

Mail delivery time tends to vary with scaling factors

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

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

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

END

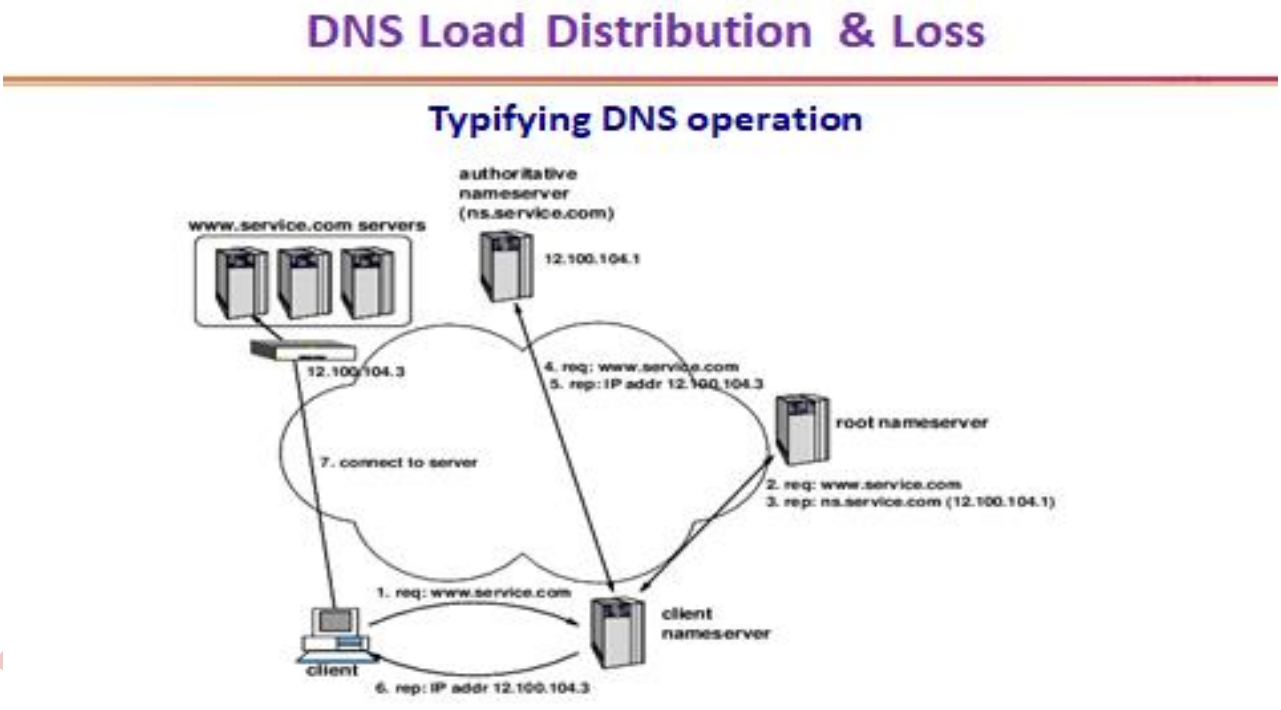
TOPIC 101

DNS Load Distribution & Loss

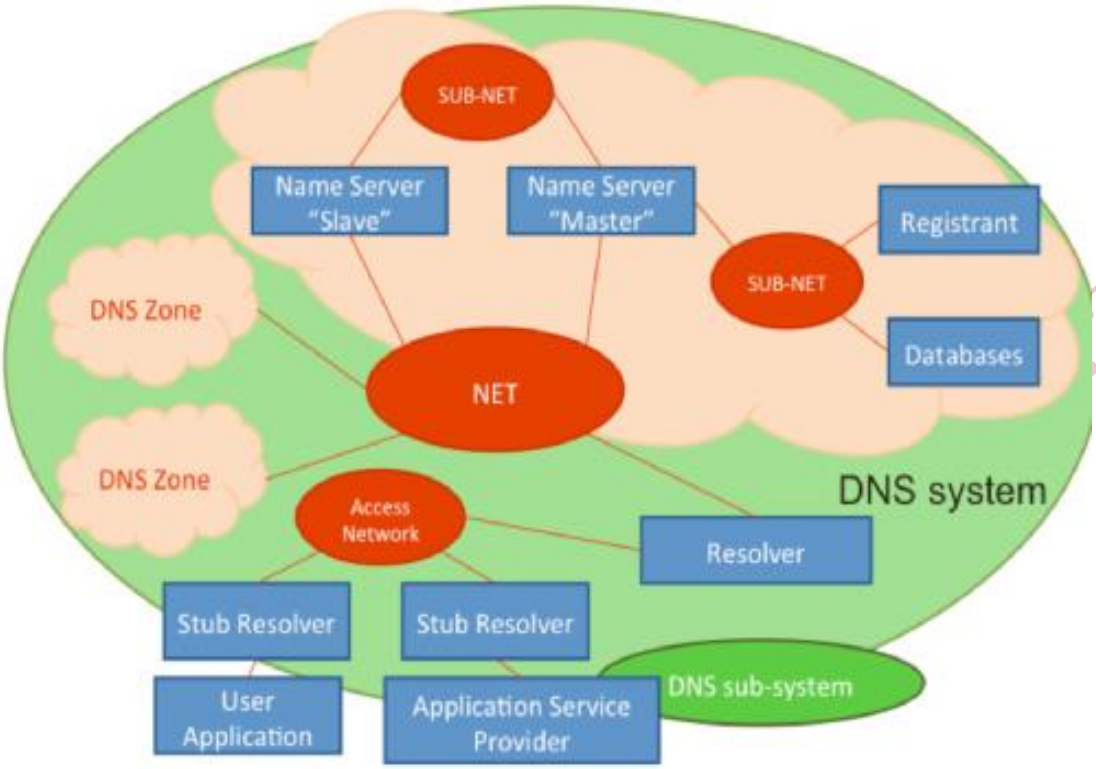
In this module

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

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

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

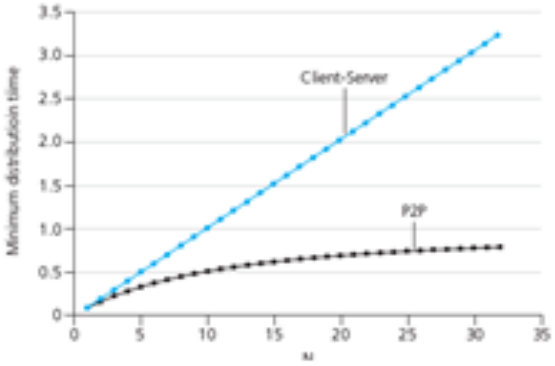
- P2P operation
- File distribution problem
- P2P performance

Operation

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

Peer to Peer Scalability

File Distribution Problem



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

Performance

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

- 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

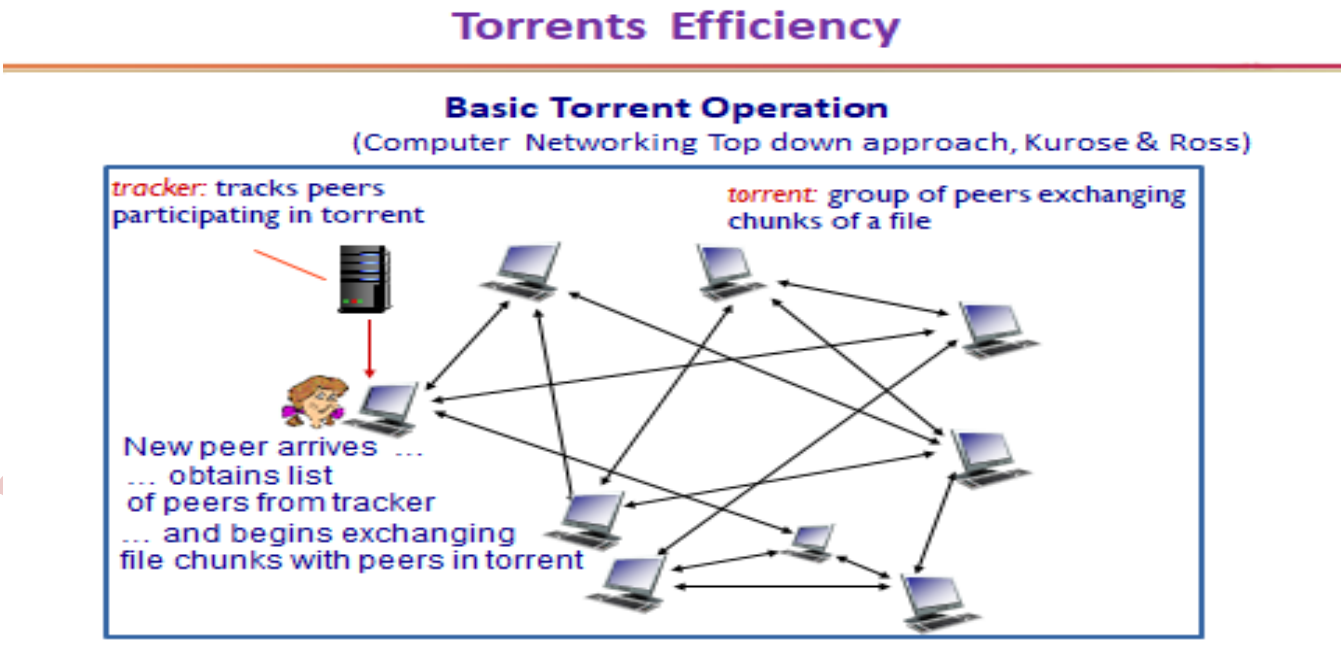
TOPIC 103

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

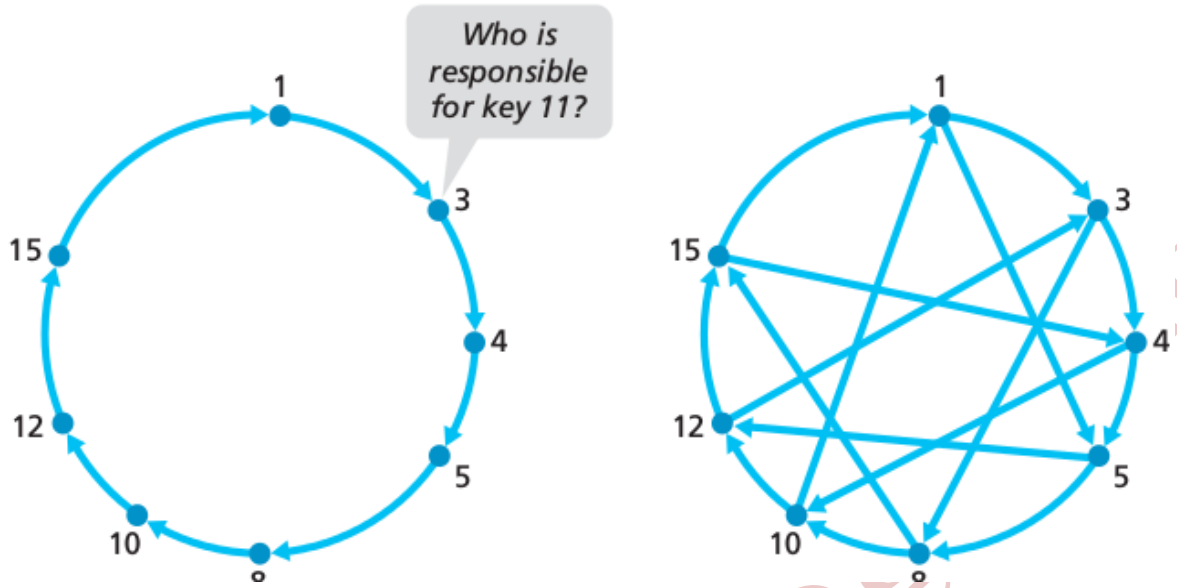
Reliability of Circular DHT

In this module

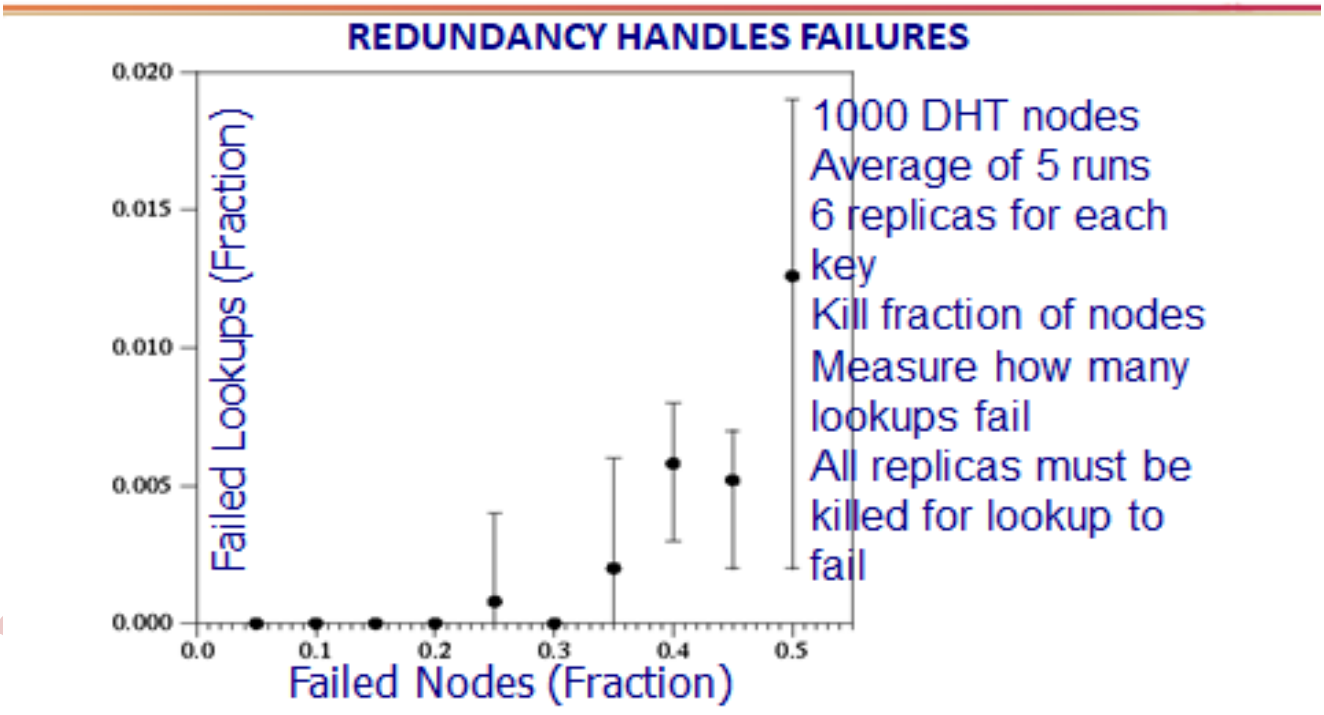
We shall understand

- 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

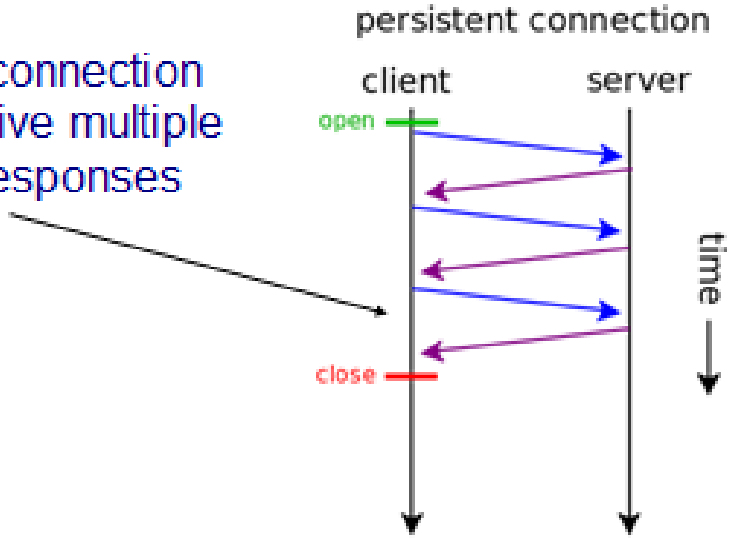
- Basic operation of HTTP Persistence
- Support in OMNET++
- Customization

Simulate HTTP Persistence

Basic Operation

(Source: Wikipedia)

Use single TCP connection to send and receive multiple HTTP requests/responses



HTTP Evolution

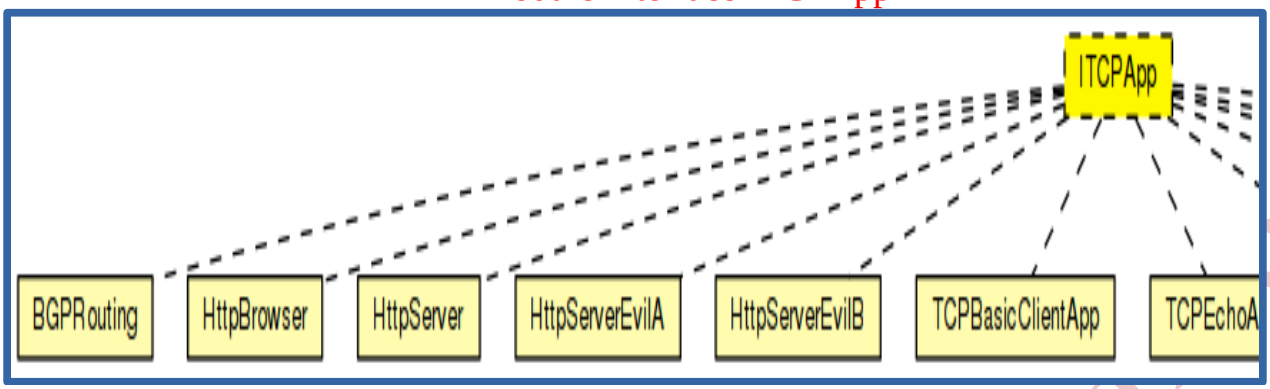
- RFC 793 does not support persistence
  - HTTP1.0
- Additional mechanism needed
  - Use keep-alive
- HTTP 1.1 is persistent by default

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

- 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

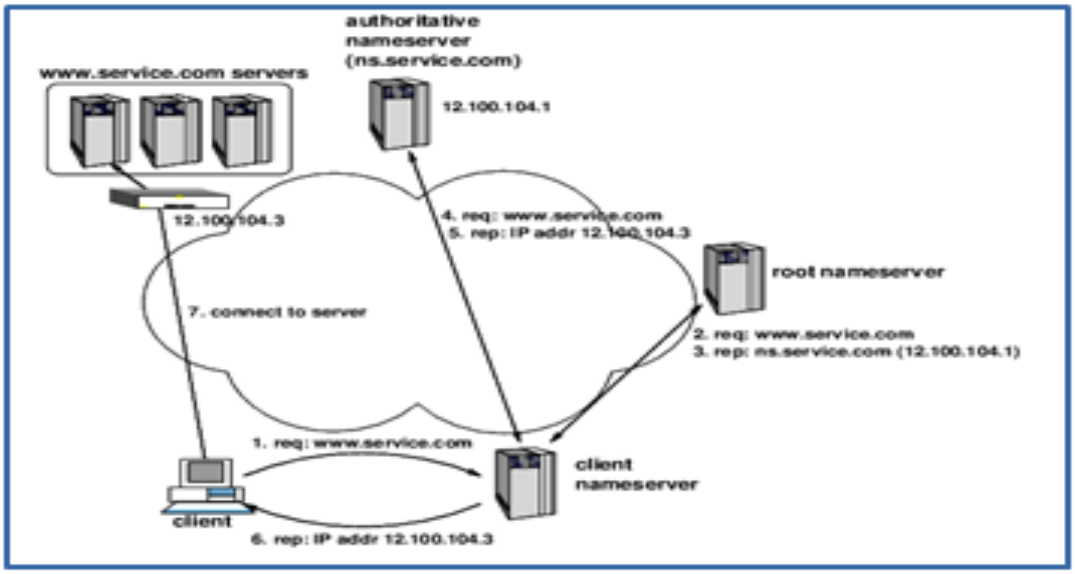
Simulate DNS Query Response

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

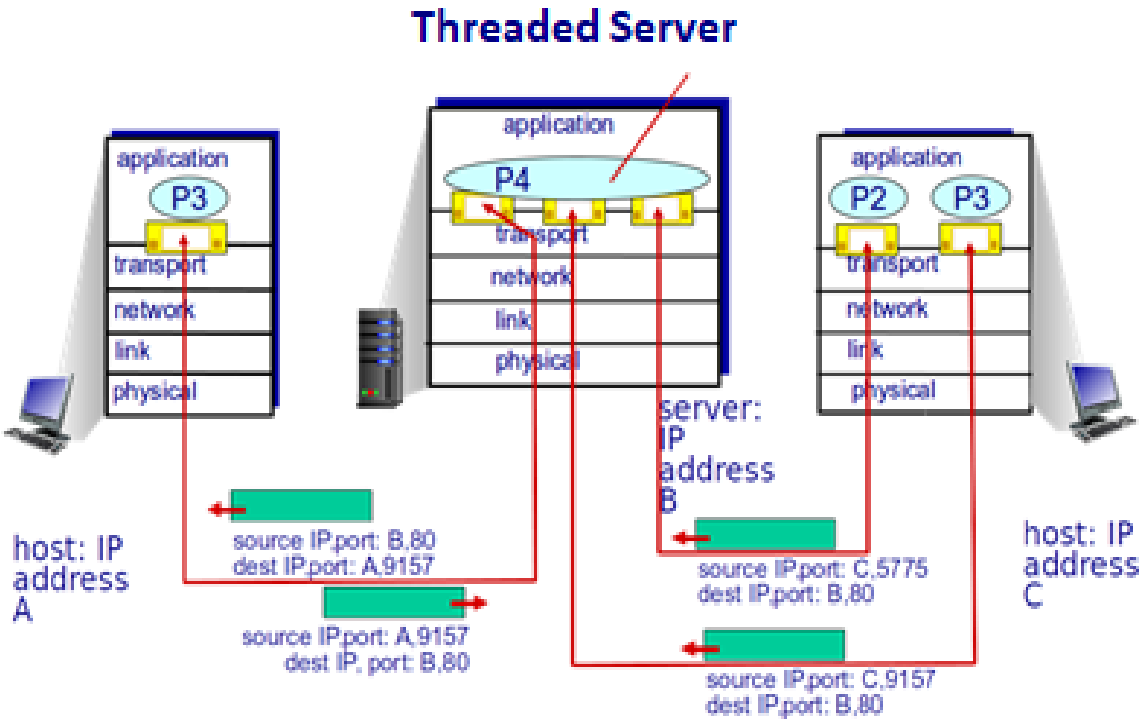
Simulate TCP Threading

In this module

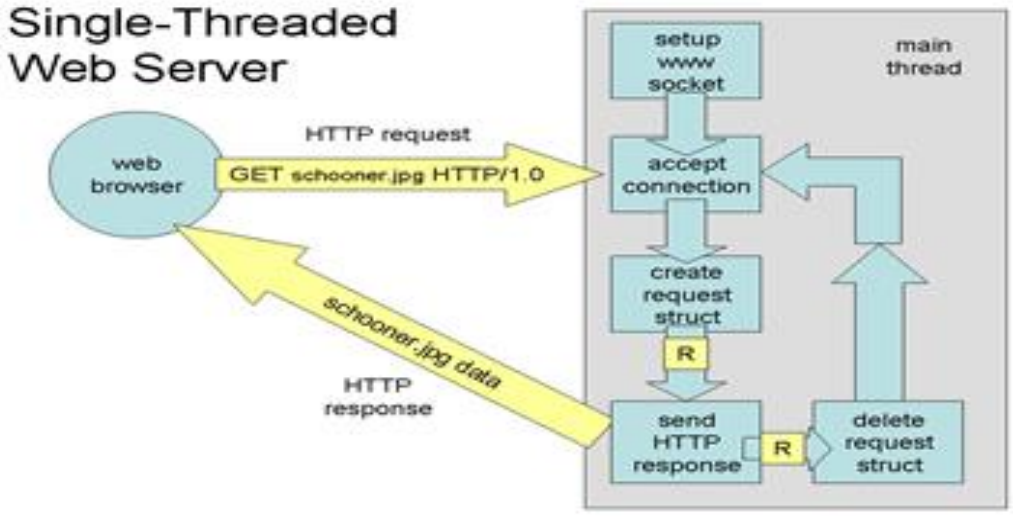
We shall understand

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

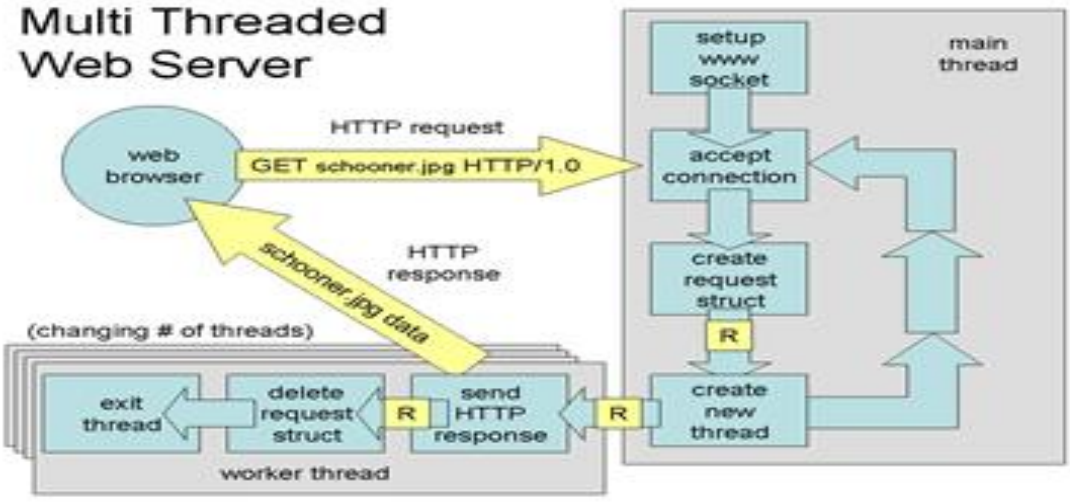
Simulate TCP Threading



Simulate TCP Threading



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:

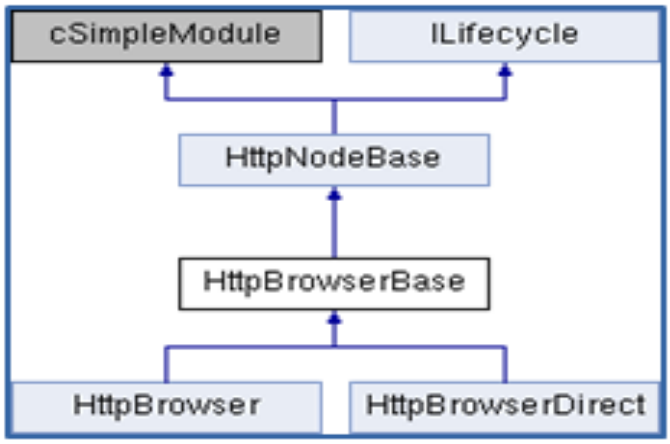
- GET, POST, HEAD
- PUT
- uploads file in entity body to path specified in URL field
- DELETE
- deletes file specified in the URL field

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)

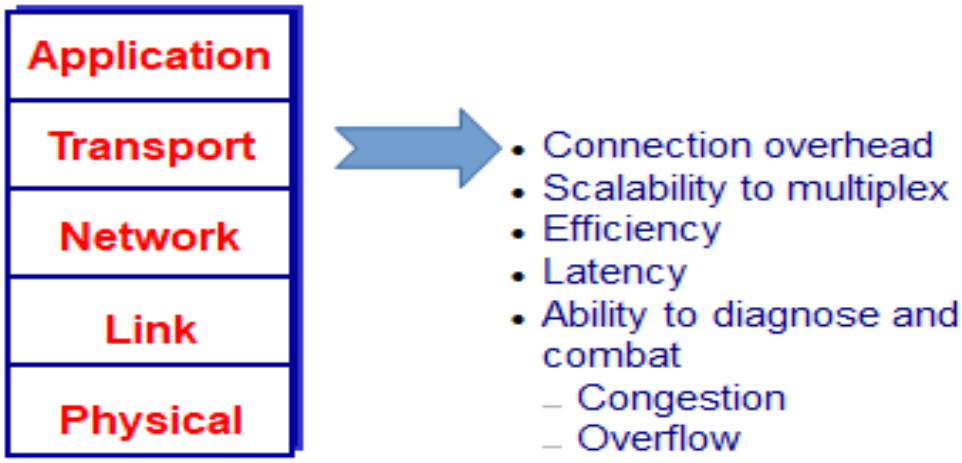
- Multiplexing and demultiplexing
- Reliable, in-order delivery (TCP)
- Congestion control
- Flow control
- Connection setup

Transport Services (2 of 2)

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

Intro & Transport Services

Modeling Approach



END

TOPIC 112

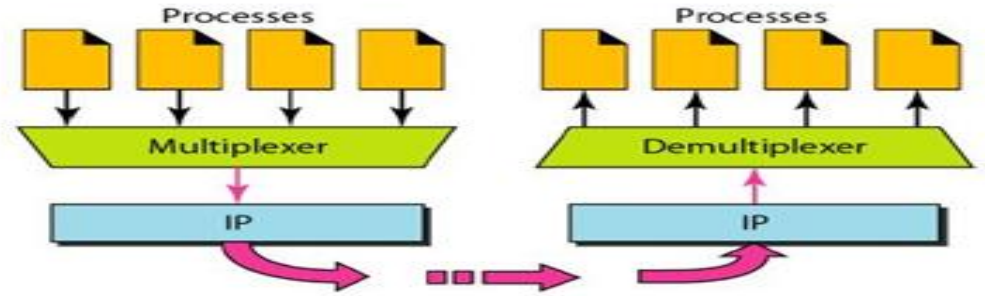
Multiplexing & Demultiplexing

In this module
We shall understand

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

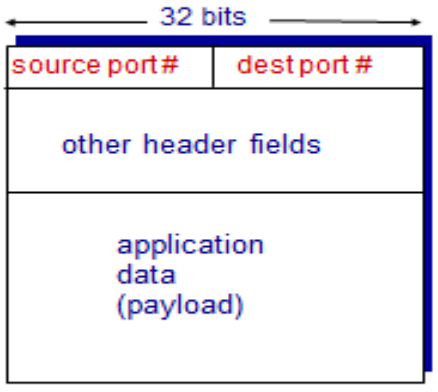
Multiplexing & Demultiplexing

Basics



Multiplexing & Demultiplexing

Capability of Port #

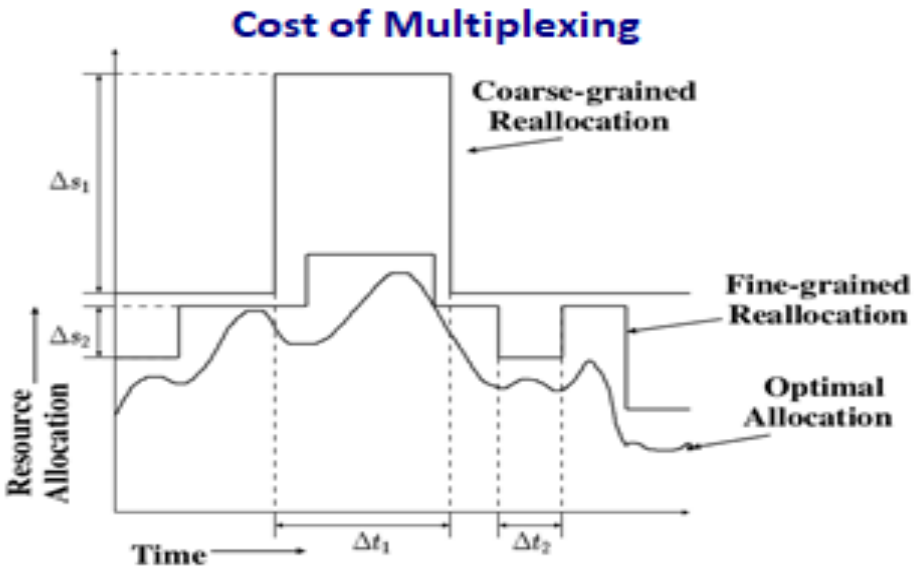


Unsigned 16-bit integer
$= 2^{16}-1$

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

MID TERM WEEK (1 TO 8)