NS3 Basics

CSE 322 - Computer Networking Sessional

Prepared By : Mashiat Mustaq

NS3 installation steps

- Version to be used - **3.39**
- *https://www.nsnam.org/docs/release/3.39/installation/html/quick-start.html*
- Follow the exact steps mentioned in the link. Install the **prerequisites** first.
- After following the steps, run the following command :

```
$ ./ns3 run hello-simulator
```

- If it outputs “Hello Simulator”, then it was installed correctly.
- *Some modules may not build due to missing dependency,* which won’t be a problem. You may solve this error by installing the missing dependencies if you wish.
- Python bindings are not required for this course.

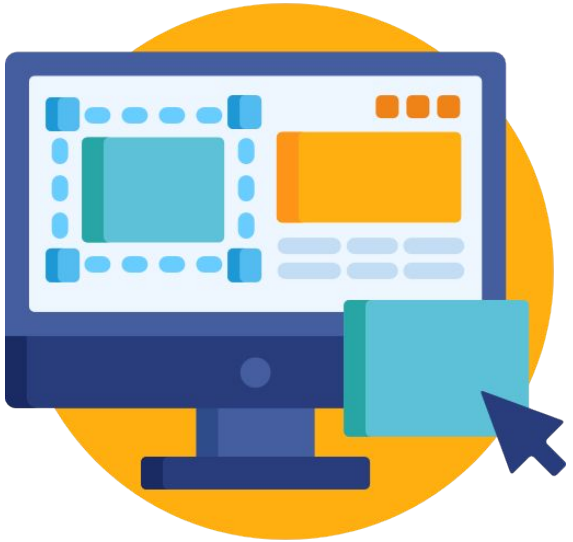
Prepared By : Mashiat Mustaq

NS3 - A Network Simulator

- **Resources:**
  - **Official Website :** https://www.nsnam.org/
  - **Tutorial** : https://www.nsnam.org/docs/release/3.39/tutorial/ns-3-tutorial.pdf
    - Useful chapters : 5-9
  - **Models:** https://www.nsnam.org/docs/release/3.39/models/html/index.html
    - Description of models are provided here. Helpful for understanding the concepts.
  - **Doxygen API documentation:**
    - https://www.nsnam.org/docs/release/3.39/doxygen/index.html
  - **IDEs :** VSCode, Jetbrains CLion etc (Install the necessary plugins)
  - **Google group :** https://groups.google.com/g/ns-3-users

Conceptual Overview - Node

- A basic computing device abstraction, e.g : Computer
- A class defined in C++
- Purpose :
  - Adding functionality such as applications, protocol stacks, peripheral cards etc



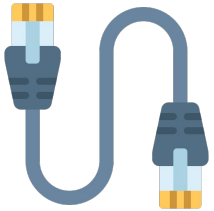
Conceptual Overview - Application

- Representation of user-level software applications
- A class defined in C++
- Purpose :
  - Runs on nodes to to run different types of simulations
- Examples:
  - *UDPEchoServer/ClientApplication, BulkSendApplication, OnOffApplication, PacketSinkApplication etc*
  - *built- in applications directory : src/application/models*

Prepared By : Mashiat Mustaq

Conceptual Overview - Channel

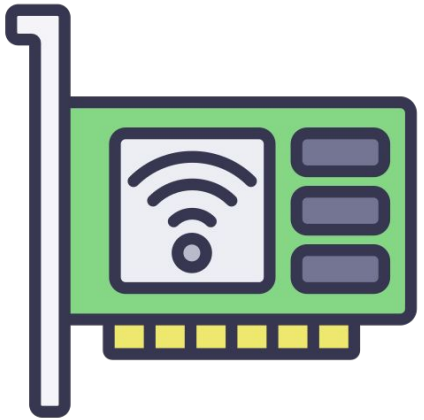
- Abstraction of the media through which data flows in the network
- Connect *Node* to a channel
- A class defined in C++
- Types :
  - CsmaChannel (Ethernet)
  - PointToPointChannel
  - WifiChannel



Prepared By : Mashiat Mustaq

Conceptual Overview - Net Device

- Abstraction of both software driver and Network Interface Card used to connect a Node to a network
- A net device is “installed” in a *Node* to enable the *Node* to communicate with other Nodes via *Channels*.
- A *Node* may be connected to more than one *Channel* via multiple *NetDevices*.
- Types :
  - CsmaNetDevice (Ethernet)
  - PointToPointNetDevice
  - WifiNetDevice



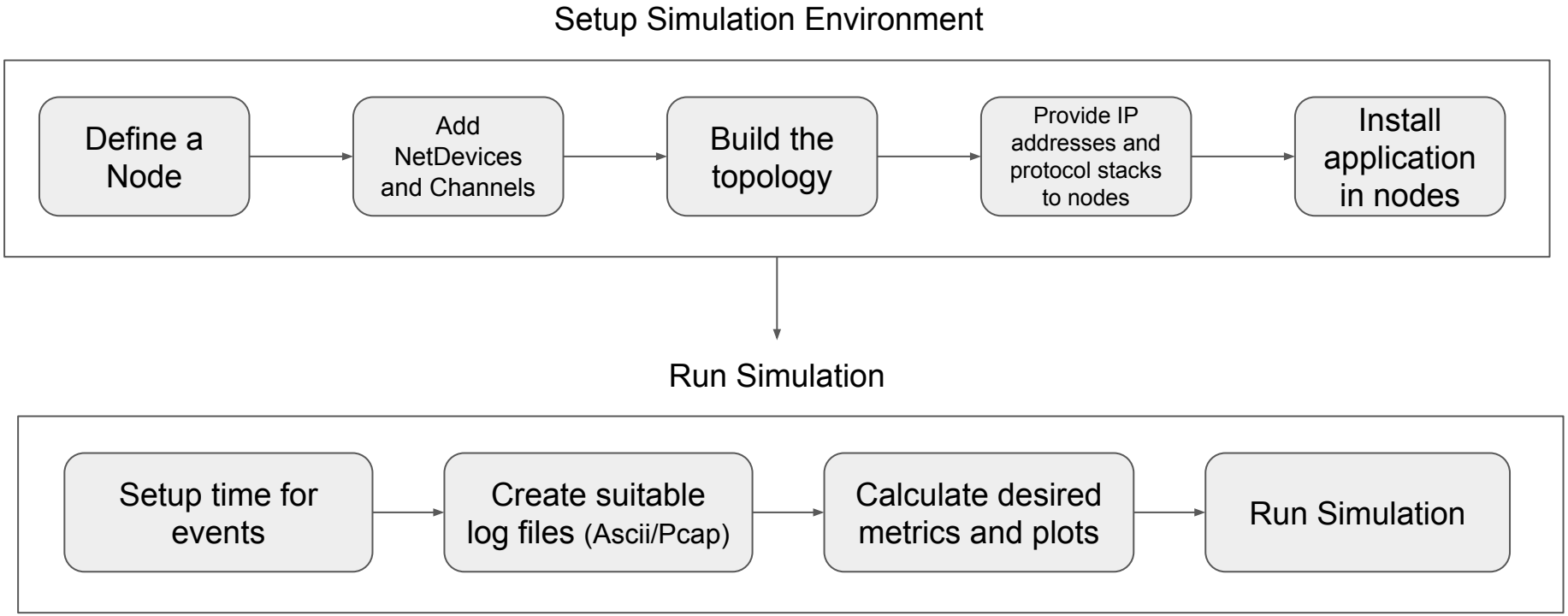
Conceptual Overview - Topology Helpers

In each simulation, simplify common tasks such as:

- Connecting Nodes to NetDevices
- Connecting NetDevices to Channels
- Assigning IP addresses etc.

Examples: PointToPointHelper, InternetStackHelper, UDPEchoServerHelper etc.

Simulation Overview



Prepared By : Mashiat Mustaq

Run first.cc

- cp examples/tutorial/first.cc scratch/first.cc
- ./ns3 build
- ./ns3 run scratch/first

When the simulator will stop?

- No further events are in the event queue.
- A special Stop event is found.
  - Simulator::Stop(stopTime)
  - Necessary when there are recurring events (WiFi)
  - *Important to call Simulator::Stop before calling Simulator::Run*

Logging Overview

Log Type	Purpose	Macro
LOG_ERROR	error messages	NS_LOG_ERROR
LOG_WARN	warning messages	NS_LOG_WARN
LOG_DEBUG	relatively rare, ad-hoc debugging messages	NS_LOG_DEBUG
LOG_INFO	informational messages about program progress	NS_LOG_INFO
LOG_FUNCTION	a message describing each function called	NS_LOG_FUNCTION - member func. NS_LOG_FUNCTION_NOARGS - static func.
LOG_LOGIC	messages describing logical flow within a function	NS_LOG_LOGIC
LOG_ALL	Log everything mentioned above	no associated macro

Logging Overview

- **LOG_LEVEL_TYPE:** Enables logging of all the levels above it.
  - **Ex : LOG_LEVEL_INFO :** Enable logging for ERROR, WARN, DEBUG, INFO types.
- **NS_LOG_UNCOND** – Log the associated message unconditionally (no associated log level).

Logging Overview

- Using the shell environment variable -> NS_LOG
  - increase the logging level without changing the script
    - export NS_LOG=UdpEchoClientApplication=level_all
  - Distinguish which method generates a log message - ORing
    - export 'NS_LOG=UdpEchoClientApplication=level_all|prefix_func'
  - Enable two logging components together - Colon separated
    - export 'NS_LOG=UdpEchoClientApplication=level_all|prefix_func:UdpEchoServerApplication=level_all|prefix_func'
  - See the simulation time
    - export 'NS_LOG=UdpEchoClientApplication=level_all|prefix_func|prefix_time:UdpEchoServerApplication=level_all|prefix_func|prefix_time'
  - Print all logging information
    - export 'NS_LOG=*=level_all|prefix_func|prefix_time'
    - ./ns3 run scratch/myfirst > log.out 2>&1

Prepared By : Mashiat Mustaq

Logging Overview

- Turn off logging previously enabled
  - export NS_LOG=""
- Enable logging in code
  - export NS_LOG=FirstScriptExample=info

Using Command Line Arguments

- Declare command line parser.
- Show general arguments for a program.
  - ./ns3 run "scratch/first --PrintHelp"
- Provide new command line argument
  - ./ns3 run "scratch/first --ns3::PointToPointNetDevice::DataRate=32Kbps"
- Provide multiple command line arguments
  - ./ns3 run "scratch/myfirst --ns3::PointToPointNetDevice::DataRate=32Kbps --ns3::PointToPointChannel::Delay=2ms"
- Add your own values with AddValue function

ASCII Tracing

+	An enqueue operation occurred on the device queue
-	A dequeue operation occurred on the device queue
d	A packet was dropped, typically because the queue was full
r	A packet was received by the netdevice

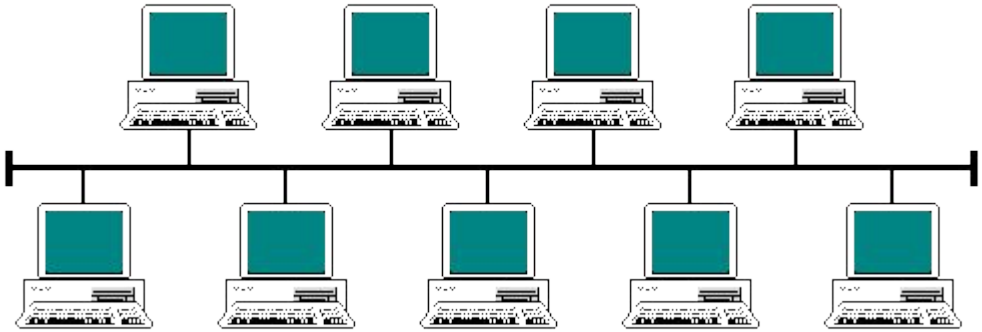
ASCII Tracing

+	Enqueue
2	Time (Seconds)
/NodeList/0/DeviceList/0/$ns3::PointToPointNetDevice/TxQueue/Enqueue	Trace source origin
ns3::PppHeader (Point-to-Point Protocol: IP (0x0021)) ns3::Ipv4Header (tos 0x0 DSCP Default ECN Not-ECT ttl 64 id 0 protocol 17 offset (bytes) 0 flags [none] length: 1052 10.1.1.1 > 10.1.1.2) ns3::UdpHeader (length: 1032 49153 > 9) Payload (size=1024)	Packet information

Pcap Tracing

- Wireshark
- Tcpdump
  - tcpdump -nn -tt -r filename.pcap

Ethernet (Bus Network)

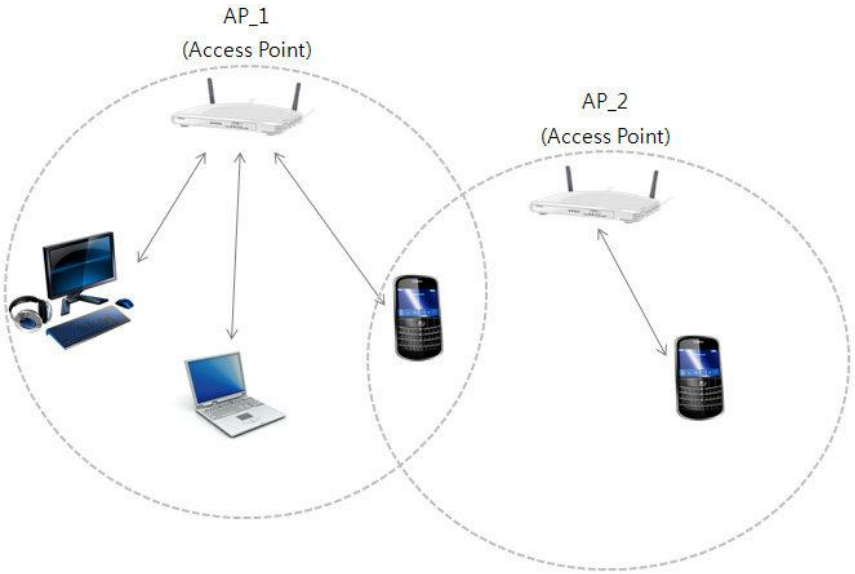


- CSMA NetDevice and channel
- Promiscuous mode - allows a network device to intercept and read each packet.
- ARP (Address Resolution Protocol) - retrieves the receiver’s MAC address.

Prepared By : Mashiat Mustaq

Wireless Network

- AP - Access Point
- AP generates beacons continuously
- Beacon - regular transmissions from access points (APs)
    - purpose to inform user devices (clients) about available Wi-Fi services and near-by access points



Prepared By : Mashiat Mustaq

Tracing

Trace Source	Trace Sink
signal events that happen in a simulation & provide access to underlying data	consume trace information
Signals when a state change happens in a model	Can be more than one sink for a trace source
Ex: the congestion window of a TCP model, when a packet is received by a netdevice and give access to the underlying packet contents	Ex: A function that outputs the new and old congestion window or increases the received packet count

Callbacks

- **Trace sinks** are **callbacks**
- Each **trace source** has a list for callbacks.
- When a trace sink expresses interest in receiving trace events, it **adds** itself as a Callback to the list of Callbacks internally held by the trace source.
- When an interesting event happens, the trace source makes an indirect call to all the **callbacks of its list.** (possible by overriding assignment operator)
- Resource : ns3 Manual -> Callbacks
  - https://www.nsnam.org/docs/manual/html/callbacks.html

Connecting Trace Source to sink

- **TraceConnect**
  - The object makes the call itself
  - Ex : fourth.cc : myObject->TraceConnectWithoutContext("MyInteger", MakeCallback(&IntTrace));

- **Config::Connect**
  - Parameters - path to the trace source and the trace sink
  - Ex: third.cc : Config::Connect(oss.str(), MakeCallback(&CourseChange));

- Context - passes an extra parameter that works as a context