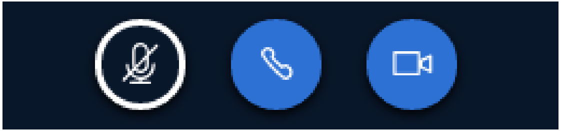
Webinars with BBB - Etiquette

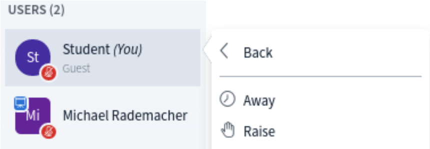
Audio and Video

- **Find a suitable space:**
  - W/o. interruptions
  - W/o. background noise
  - Have good lighting on your face
- Use a **headset** if available
- **Turn ON your camera** unless you are experiencing connection problems
- **Turn OFF your microphone** and only unmute it if you want to participate

Participation

- Click on your Name and "Set status" - "Raise"
- I will call your name
- Ask questions in the Shared Notes. Write "+1" if you have the same question.
- **Lectures (Audio, Video, all Chats) will be recorded**. If you do not like to appear in the recording, let me know





Do NOT hesitate to participate! You will get used to the system very quickly!

Short Personal Introduction

Personal Introduction - Michael Rademacher

Education

1987 – 2007	Grew up close to Olpe, Sauerland, NRW
2007 – 2011	Bachelor Industrial Engineering, FH Südwestfalen Information and Communication Technology
2011 – 2014	Master Computer Science, Hochschule Bonn-Rhein-Sieg Communication
2015 – 2019	Ph.D. (Dr.-Ing.), TU-Kaiserslautern WiFi Backhaul Networks with Directional Antennas

michael-rademacher.net

Hochschule Bonn-Rhein-Sieg University of Applied Sciences

Work experience

2009 – 2010	Research Assistant, FH Südwestfalen
2010 – 2011	(Student) Employee, DETECON Consulting
2011 – 2014	Research Assistant, Fraunhofer FOKUS
2014 – 2018	Founder, DeFuTech UG
2014 – 2019	Scientist, Hochschule Bonn-Rhein-Sieg
2019 – today	Scientist, Fraunhofer FKIE

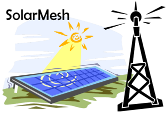
Fraunhofer FKIE

How we work in our research group @H-BRS

Research

- Research through supervised student work
- External funding through BMBF und BMWi
- Regular participation in CTFs (RedRocket)

SDN4 WIRELESS



HITB+CyberWeek

Abu Dhabi, UAE: 12-17 October 2019

Education and training

- Communication in distributed Systems
- Theses

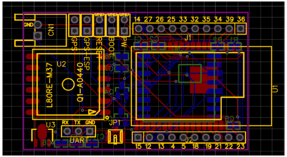
Lab for Cyber-Security:

- User-Security:
  - Basics
  - Cryptography
  - Network Security
- IoT-Security:
  - Networks
  - Protocols
- Blockchain-Tech

Fraunhofer ACADEMY

Industry Projects

- Cooperation with SMEs and cities
- Consulting and contract work
- Prototype production (Hardware + Software)
- Rhein-Sieg-Netz, Becker-Antriebe GmbH, ABUS KG, Stadt Bonn,...



Communication in Distributed Systems

1 Introduction and Methodology

Dr.-Ing. Michael Rademacher

2021-04-06

Introduction and Methodology of this term

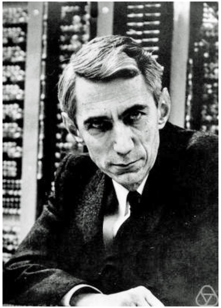
Goals and Methods

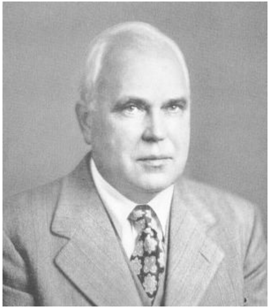
Goals of this course:

To teach theoretical and practical communication principles of distributed systems using real-world examples with currently hyped technologies, in order to enable students to perform scientific research in this field.

Methods:

- Lecture,
- readings,
- discussion,
- simulation,
- scientific writing.

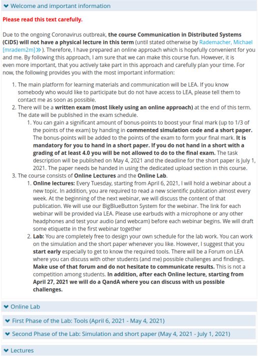




Students and LEA

Surname	Name	UserID
Abedin	Md-Sakhawat	mabedi2s
Afzal	Saad	safzal2s
Ahmed	Mohammad-Rasal	mahmed2s
Akala	Samson	sakala2s
Akuba	Godfrey Ojimah	gakuba2s
Anamah	Samuel	sanama2s
Appaiah	Dechamma	dappai2s
Arzensek	Josua	jarzen2s
Athuluru	Harshitha	hathul2s
Butt	Zubair	abutt2s
Chollapra	Febin	fcholl2s
Dadhich	Chhavi	cdadhi2s
Dharmalingam	Suganya	sdharm2s
Dietrich	Timon	tdietr2s
Gudala	Radha	rgudal2s
Hampe	Yannik	yhampe2s
Hussain	Syed	shussa2s
Janapati	Bhavya	bjanap2s
Keshavarzi	Ehsan	ekesha2s
Lakshmikanth	Varun	vlaksh2s
Lysek	Alexander	alysek2s
Madasu	Vijaya	vmadas2s
Meyerhoff	Johannes	jmeyer2s
Nava	Mildred	mnava2s
Obamwonyi	Nosa	nobamw2s
Oke	Oluwaseun	ooke2s
Panduru	Prashanthi	ppandu2s
Poluri	Sri	spolur2s
Pranti	Rubaiya	rprant2s
Roy	Subrata	sroy2s
Sajib	Ahmned	asajib2s
Sakhamuru	Bandhavi Sai Sri	bsakha2s
Shetty	Ranjan	rshett2s
Sifat	Mik	msifat2s
Stotz	Fabian	fstotz2s
Tabassum	Nuzat	ntabas2s

- According to LEA on: 2020-03-31



Welcome and important information

Please read this text carefully.

Due to the ongoing Coronavirus outbreak, **the course Communication in Distributed Systems (CiDS) will not have a physical lecture in this term** (until stated otherwise by Rademacher, Michael [mradem2m] »). Therefore, I have prepared an online approach which is hopefully convenient for you and me. By following this approach, I am sure that we can make this course fun. However, it is even more important, that you actively take part in this approach and carefully plan your time. For now, the following provides you with the most important information:

1. The main platform for learning materials and communication will be LEA. If you know somebody who would like to participate but do not have access to LEA, please tell them to contact me as soon as possible.
2. There will be a **written exam (most likely using an online approach)** at the end of this term. The date will be published in the exam schedule.
    1. You can gain a significant amount of bonus-points to boost your final mark (up to 1/3 of the points of the exam) by handing in **commented simulation code and a short paper.** The bonus-points will be added to the points of the exam to form your final mark. **It is mandatory for you to hand in a short paper. If you do not hand in a short with a grading of at least 4.0 you will be not allowed to do to the final exam.** The task description will be published on May 4, 2021 and the deadline for the short paper is July 1, 2021. The paper needs be handed in using the dedicated upload section in this course.
3. The course consists of **Online Lectures** and the **Online Lab**.
    1. **Online lectures:** Every Tuesday, starting from April 6, 2021, I will hold a webinar about a new topic. In addition, you are required to read a new scientific publication almost every week. At the beginning of the next webinar, we will discuss the content of that publication. We will use our BigBlueButton System for the webinar. The link for each webinar will be provided via LEA. Please use earbuds with a microphone or any other headphones and test your audio (and webcam) before each webinar begins. We will draft some etiquette in the first webinar together
    2. **Lab:** You are completely free to design your own schedule for the lab work. You can work on the simulation and the short paper whenever you like. However, I suggest that you **start early** especially to get to know the required tools. There will be a Forum on LEA where you can discuss with other students (and me) possible challenges and findings. **Make use of that forum and do not hesitate to communicate results.** This is not a competition among students. **In addition, after each Online lecture, starting from April 27, 2021 we will do a QandA where you can discuss with us possible challenges.**

Online Lab
First Phase of the Lab: Tools (April 6, 2021 - May 4, 2021)
Second Phase of the Lab: Simulation and short paper (May 4, 2021 - July 1, 2021)
Lectures

Methodology of this term - Online Lab

- We can not have a physical Lab this term :(
- But we can have an online Lab!

I want you to learn **three skills** from this lab to become **a good scientist** or at least **write a better Masterthesis**.

I. How to conduct a meaningful **simulation**.

II. How to analyze and **plot data**.

III. How to **write** a research paper.

Therefore, you will learn **mostly by yourself** three different **tools.**

I. **ns-3**

II. **matplotlib**

III. **Latex**

Your task for this semester is to conduct a ns-3 simulation about WiFi propagation, plot the data with matplotlib and write a short research paper in Latex about it.

Methodology of this term - Online Lab

- **It is mandatory for you to hand in a short paper. If you do not hand in a short with a grading of at least 4.0 you will be not allowed to do the final exam.**
- The final task description will be published on May 4, 2021 in the LEA course and the deadline for the short paper is **July 1, 2021**. The paper needs be handed in using the dedicated upload section in LEA.
- In addition to the paper, you need to hand in **the source code of your simulation.**
- Before May 4, 2021 **you have one month to get familiar with the tools**.
- **Warning: I do not like poorly written papers.** Please pay attention to grammar and spelling mistakes. If your paper is full of mistakes, it will be graded 5.0. **The paper can be in english (American or British) or german.**

In case you have problems with the task do **not** write me an E-Mail (I get a lot of these E-Mails.) You can use the Forum in LEA (**help each other!**) or asks questions after the lectures.

Methodology of this term - Lectures

ID	Date	Lecturer	Topic
1	2020-04-06	Rademacher	Introduction, task, simulations and ns-3
2	2020-04-13	Rademacher	Review of required background
3	2020-04-20	Rademacher	Modern Backhaul technologies
4	2020-04-27	Rademacher	WiLD — Physical Layer and Propagation
5	2020-05-04	Rademacher	WiLD — Hardware and MAC
6	2020-05-11	Rademacher	WiLD — Throughput Enhancement
7	2020-05-18	Jonas	Mesh Networks
8	2020-05-25	Jonas	IEEE802.11s
9	2020-06-01	Rademacher	Software Defined Networking
10	2020-06-08	Rademacher	LoRaWAN and MQTT
11	2020-06-15	Rademacher	Blockchain-Technology Introduction
12	2020-06-22	Rademacher	Blockchain-Technology Deep Dive
13	2020-06-29	Rademacher	5G Mobile Networks in a nutshell
14	2020-07-06	Rademacher	Summary and QandA

Methodology of this term - Readings

- After most of the Lectures, you should read a scientific publication about the topic of the last lecture.
- **There will be link or the pdf file available in LEA.**
- At the beginning of the next lecture, we will discuss this publication together.
- It is important that you read the publication carefully in order to participate in the discussion. The **main results** of the paper will be relevant for the exam!
- **“Why, this seems like a lot of work with all the lectures and the online lab!”**
- Being able to read and understand publications from other researchers in **YOUR field** is the most important skill for a scientist. If you are unable to do this, you will never work close to the state-of-the-art.

Some questions before we get started

1. Who has worked with ns-3 before?
2. Who has conducted network simulations?
3. Who has used C++ before?
4. Who has used Python before?
5. Who has used Matplotlib before?
6. Who has used Latex before?
7. Who has used an object-oriented programming language?
8. Who has used a Linux based operating system?

Network Analysis Techniques

Network Analysis Techniques

Modeling:

- Mathematical analysis

$$S = \frac{P_s E[P]}{E[Slot]} = \frac{P_s E[P]}{P_i\ \sigma + P_s\ T_s + P_c T_c}$$

$$D = \frac{\text{Number of stations}}{\text{Packets per Second}} = \frac{N}{S/E[P]}$$

Simulation:

- Model the system at abstract level via software
- **ns-3**, OPNET, OMNeT++

Emulation:

- HW components that behave like real system

Experimentation:

- Experiments using a testbed

Simulations in general

Reasons for a simulation:

- Scalability
- Repeatability
- Rapid Prototyping
- Education
- Predication/Retrodiction

Limits of a simulation:

- Hardware specific limits are difficult to simulate (CPU, memory)
- Energy consumption
- Reproduction of real user behaviour and its variations (e.g. Web)
- PHY, in particular radios
- Hardware variations (quarz/clock)

Characteristics of a good Simulation:

- Repeatability
- Unbiased
- Rigorous
- Statistically sound

Simulation results can lead towards good and useful implementations. They do not replace real tests!

Which simulator should I use?

Comparability

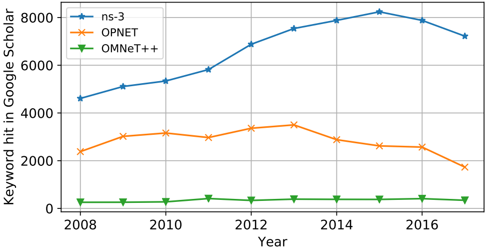
- Adapted in the community
- Active developments

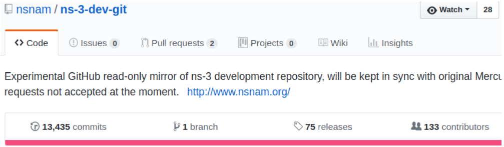
Extendability

- Free and Open Source Software
- Modular architecture

Reusability

- Various (tested) modules available



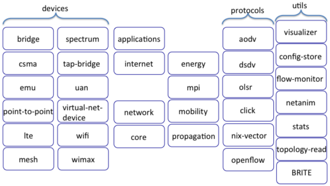

nsnam / ns-3-dev-git

Watch 28

Code | Issues 0 | Pull requests 2 | Projects 0 | Wiki | Insights

Experimental GitHub read-only mirror of ns-3 development repository, will be kept in sync with original Mercu
requests not accepted at the moment. http://www.nsnam.org/

13,435 commits | 1 branch | 75 releases | 133 contributors


devices

bridge, spectrum, csma, tap-bridge, emu, uan, point-to-point, virtual-net-device, lte, wifi, mesh, wimax

applications, internet, network, core

energy, mpi, mobility, propagation

protocols

aodv, dsdv, olsr, click, nix-vector, openflow

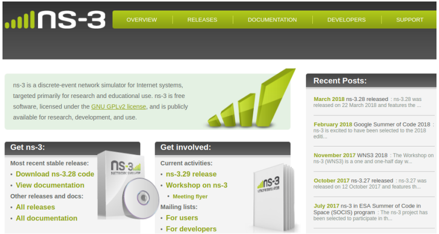
utils

visualizer, config-store, flow-monitor, netanim, stats, topology-read, BRITE

ns-3 in a nutshell

ns-3 Basics

- ns-3 is written in C++, bindings for Python
  - "A simulation": Executable with shared libraries
- ns-3 is supported for **Linux**, OS X, and FreeBSD
- ns-3 is **not** backwards-compatible with ns-2
- Key differences from other network simulators:
  1. Command-line/unix orientation
  2. There is no GUI/IDE
  3. Simulations in C++
  4. There is no domain-specific language



ns-3 is a discrete-event network simulator for Internet systems, targeted primarily for research and educational use. ns-3 is free software, licensed under the GNU GPLv2 license, and is publicly available for research, development, and use.

Get ns-3:

Most recent stable release:

- Download ns-3.28 code
- View documentation

Other releases and docs:

- All releases
- All documentation

Get involved:

Current activities:

- ns-3.29 release
- Workshop on ns-3
  - Meeting flyer

Mailing lists:

- For users
- For developers

Recent Posts:

March 2018 ns-3.28 released : ns-3.28 was released on 22 March 2018 and features the ...

February 2018 Google Summer of Code 2018 : ns-3 is excited to have been selected to the 2018 editi...

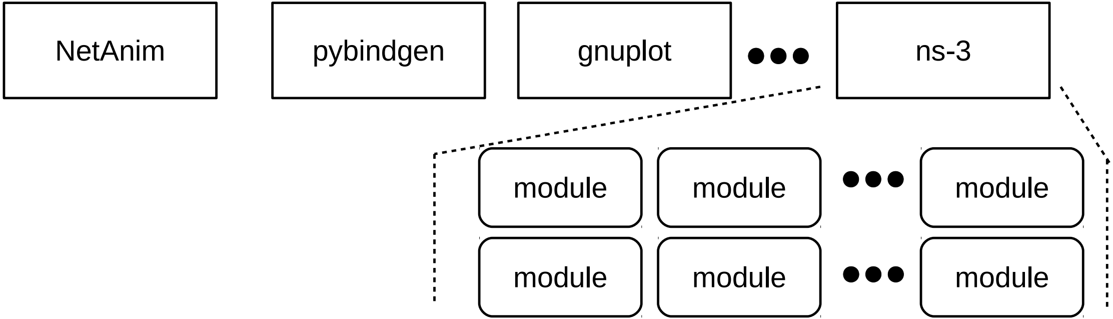
November 2017 WNS3 2018 : The Workshop on ns-3 (WNS3) is a one and one-half day w...

October 2017 ns-3.27 released : ns-3.27 was released on 12 October 2017 and features th...

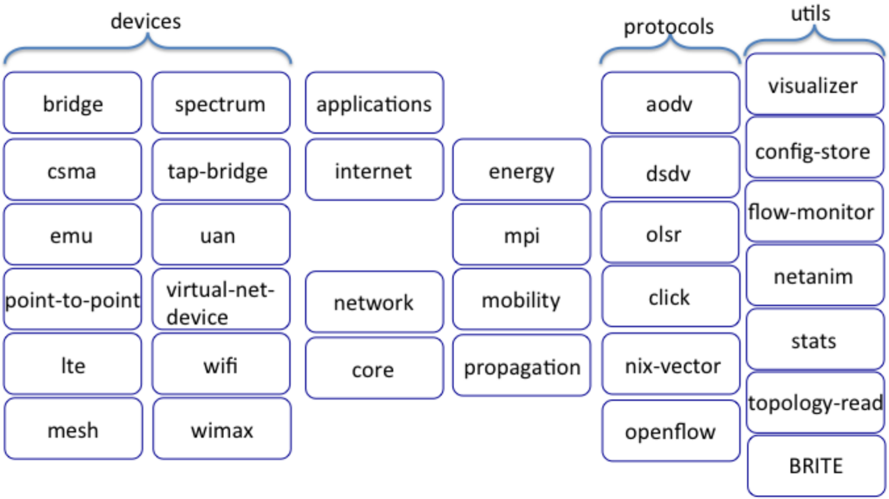
July 2017 ns-3 in ESA Summer of Code in Space (SOCIS) program : The ns-3 project has been selected to participate in th...

ns-3 software organization

- Two levels of ns-3 software and libraries
  - Supporting libraries, not system-installed, in parallel to ns-3
  - ns-3 modules exist within the ns-3 directory



Example for ns-3 modules



devices

bridge
spectrum
csma
tap-bridge
emu
uan
point-to-point
virtual-net-device
lte
wifi
mesh
wimax

applications
internet
network
core

energy
mpi
mobility
propagation

protocols

aodv
dsdv
olsr
click
nix-vector
openflow

utils

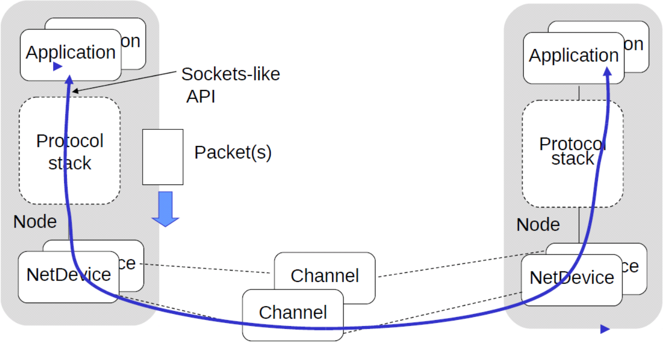
visualizer
config-store
flow-monitor
netanim
stats
topology-read
BRITE

Core Modules: Smart Pointer, Callbacks, Events, Scheduler, Logging, Tracing ,...
Essential Modules: Node, Sockets, Queues, Packets ,...

Fundamental parts of a ns-3 Simulation

- Nodes ("is a shell of a computer")
- Applications (e.g. echo servers, traffic generator)
- Stacks (e.g. IPv4, Routing)
- NetDevices (e.g. Wi-Fi, WiMAX, CSMA, Point-to-Point, Bridge)
- Channels (e.g. Wi-Fi channel)

Key objects of each ns-3 simulations!



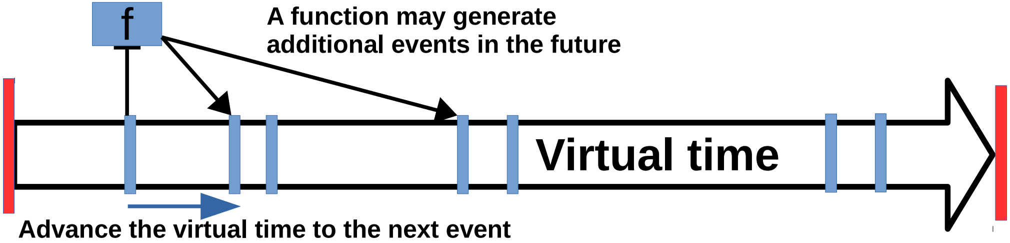
Typical structure of a ns-3 program

```
int main (int argc, char *argv[]) {
// Set default attribute values
// Parse command-line arguments
// Configure the topology;
//      nodes, channels, devices, mobility
// Add (Internet) stack to nodes
// Configure IP addressing and routing
// Add and configure applications
// Configure tracing
// Run simulation
}
```

- **ns-3 is, in the core parts, a C++ object system**
- ns-3 objects (inherit class ns3::Object) get several additional features:
  - an attribute system
  - smart-pointer memory management (Class Ptr)
  - ...

Discrete-event simulations in ns3

- Simulation time moves in discrete jumps from event to event
    - Time is stored as a large integer in ns-3
- C++ functions schedule events to occur at specific virtual times
    - i.e. callback functions
    - Events have IDs to allow them to be canceled or to test their status
- A **simulation scheduler** orders the event execution
- Simulator::Run() gets it all started
- Simulation stops:
    - At specific time
    - There are no more events


f
A function may generate additional events in the future
Virtual time
Advance the virtual time to the next event

Attribute system

Problem: Identify all the values affecting results of simulations and configure them easily

- **repeatability**

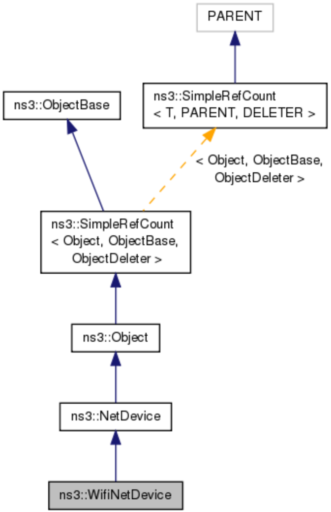
Approach: Each object has a set of attributes: name, type, initial value

- Attributes are exported into a string-based namespace (filesystem-like paths) supporting regular expressions

```
// For all Queues of the type DropTail
Config::SetDefault ("ns3::DropTailQueue::MaxPackets", 25);

// For all TxQueues
Config::Set ("/NodeList/*/DeviceList/*/TxQueue/MaxPackets", 25);

// Only for the TxQueue on Node 0 of the first device
Config::Set ("/NodeList/0/DeviceList/0/TxQueue/MaxPackets", 25);
```

Smart pointers

- Smart pointers use reference counting to improve memory management.
- The class ns3::Ptr is semantically similar to a traditional pointer, but **the object pointed to will be deleted when all references to the pointer are gone**



```
// Create a new WiFiNetDevice and return a Smart Pointer
Ptr<WifiNetDevice> device = CreateObject<WifiNetDevice> ();
```

Container and Helper API

- The ns-3 “helper API” provides a set of classes and methods that make common operations easier (“syntactical sugar”).
  - There exist various helper classes
  - Each function applies a single operation on a container
- Containers group similar objects (for convenience)
  - NodeContainer: vector of Ptr<Node>
  - NetDeviceContainer: vector of Ptr<NetDevice>

```
// Init a NodeContainer
NodeContainer nodes;
// Use the helper function instead of CreateObject<Node> ()
nodes.Create (2);
// Return the smart-pointer to the first node of the container
Ptr<Node> node = nodes.Get (0);
```

Tracing System (generate Output)

- Packets can be saved to .pcap files (Wireshark)
- **Simulator provides** a set of pre-configured trace **sources**
- **User provides** trace **sinks** and attach to these trace source

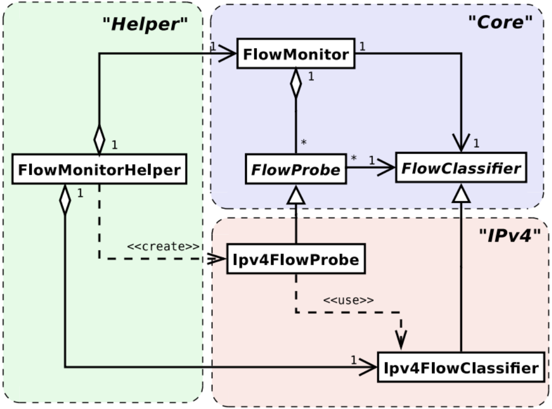
```
//  User writes a Contention Window Trace sink
void CwndTracer (uint32_t oldval, uint32_t newval) {
// Do Something with oldval and newval
}
// Attach CongestionWindow Trace Source to own callback function
Config::ConnectWithoutContext (
"/NodeList/0/$ns3::TcpL4Protocol/SocketList/0/CongestionWindow",
MakeCallback (&CwndTracer));
```

Logging

- ns-3 has a build-in logging component
  - Avoid using ”std::cout « myValue « std::endl”
- Useful to debug the progress of simulations
- Logging is only compiled in the debug build (default)
- Nearly each module has an associated logging component

```
int main (int argc, char *argv[])
{
  ...
  NS_LOG_LOGIC ("The␣simulation␣has␣started");
  // Enable Udp Client Logging
  LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
  LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
  ...
```

Flow Monitor (Network monitoring framework)



- Detect all flows passing through the network (5-tuple)
- Stores metrics (throughput, delay, PER,...)
- Exports statistics to stdout, additional processing and plotting is needed

First functional example I

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("FirstScriptExample");
int main (int argc, char *argv[]){
   CommandLine cmd;
   cmd.Parse (argc, argv);

   LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
   LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);

   NodeContainer nodes;
   nodes.Create (2);
```

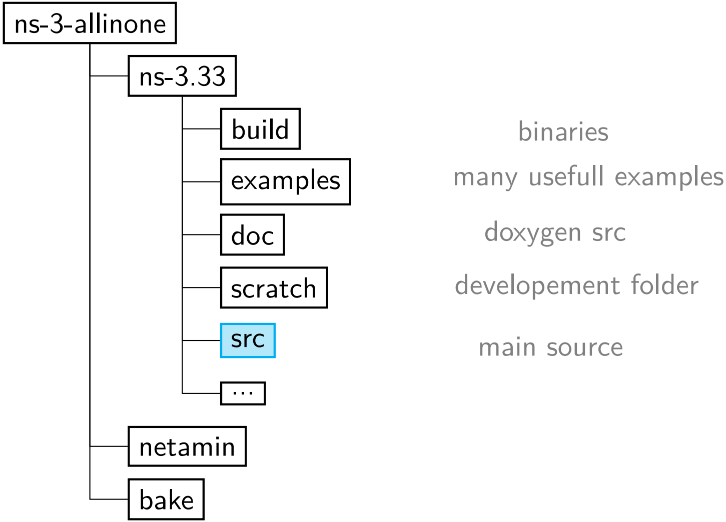
First functional example II

```
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute (”DataRate”, StringValue (”5Mbps”));
pointToPoint.SetChannelAttribute (”Delay”, StringValue (”2ms”));

NetDeviceContainer devices;
devices = pointToPoint.Install (nodes);

InternetStackHelper stack;
stack.Install (nodes);

Ipv4AddressHelper address;
address.SetBase (”10.1.1.0”, ”255.255.255.0”);

Ipv4InterfaceContainer interfaces = address.Assign (devices);

UdpEchoServerHelper echoServer (9);

ApplicationContainer serverApps = echoServer.Install (nodes.Get (1));
```

First functional example III

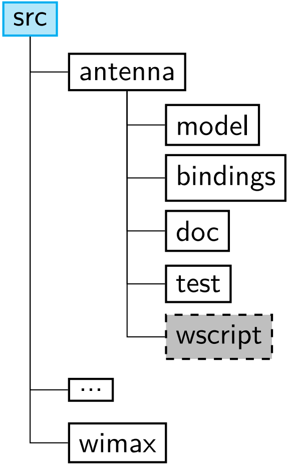
```
  serverApps.Start (Seconds (1.0));
  serverApps.Stop (Seconds (10.0));

  UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);
  echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient.SetAttribute ("PacketSize", UintegerValue (1024));

  ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));
  clientApps.Start (Seconds (2.0));
  clientApps.Stop (Seconds (10.0));

  Simulator::Run ();
  Simulator::Destroy ();
  return 0;
}
```

Module and program organization



- Your first C++ file should be placed in the scratch directory

Module and program organization



src

antenna

model — main source code of this module

bindings — python bindings

doc — doxygen src

test — unit test

wscript — "Makefile" equivalent

· · ·

wimax

waf configuration and build

- The ns-3 build tool: **“waf”**
  - Framework for configuring, compiling and running ns-3 programs
  - A replacement tool for Autotools or CMake
- Configure ns-3

```
./waf configure --enable-examples --enable-tests
```

  - Debug build (default): all asserts and debugging code vs optimized

```
./waf -d [debug|optimized] configure
```

- Building the simulation

```
./waf build
```

- Running the simulation
  - A special shell for running programs

```
./waf shell
```

  - Run the program ”mySimulation”

```
./waf --run "scratch/mySimulation␣--param1=100"
```

My personal lessons learned

- Use a **C++ IDE** (i.e. vcsode + ./waf shell)
- Work in the /scratch folder with subfolders
  - Develop your own model in /src only when necessary
- The **google user group** is a great source of information
- Try to **adapt an example** if possible
- Do not start with max. throughput tests (runtime)
- Tracing/Logging: grep in the code for examples
- Use the internal logging class instead of std::cout
- **Give the simulation (protocols) enough time to settle**
- Use your own plotting functions (flowmonitor + csv + matplotlib)
- Stick to the debug build until the final runs (asserts!)
- Mutli-core simulations with gnu-parallel

Link List

Be sure to use the current Version: ns-3.33

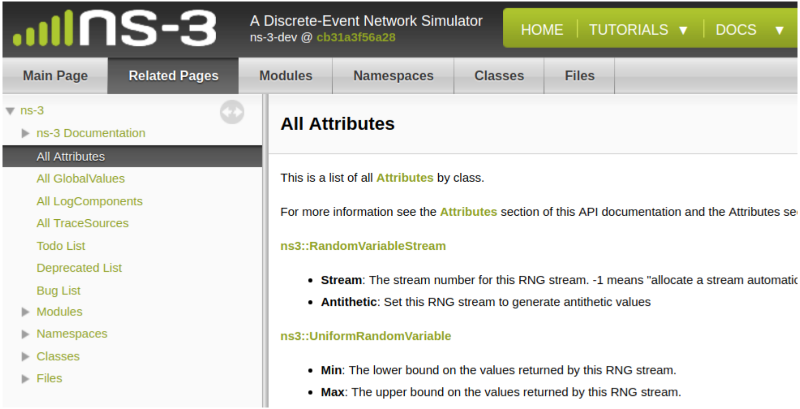
Home: `https://www.nsnam.org/`

Releases: `www.nsnam.org/releases/`

Tutorial: `https://www.nsnam.org/docs/tutorial/html/`

Documentation: `https://www.nsnam.org/doxygen/`

These slides are based on Tom Henderson's work at `https://www.nsnam.org/wiki/AnnualTraining2015`



All Attributes

This is a list of all **Attributes** by class.

For more information see the **Attributes** section of this API documentation and the Attributes se

ns3::RandomVariableStream

- **Stream**: The stream number for this RNG stream. -1 means "allocate a stream automatic
- **Antithetic**: Set this RNG stream to generate antithetic values

ns3::UniformRandomVariable

- **Min**: The lower bound on the values returned by this RNG stream.
- **Max**: The upper bound on the values returned by this RNG stream.

Thank you for your attention.
Are there any questions left?

Room K331
Rathausallee 10
Technopark
Sankt Augustin

michael.rademacher@h-brs.de
www.mc-lab.de
https://michael-rademacher.net

References