Network Simulator v.3
ns-3

Dmitry Petrov, PhD

dmitry.petrov@magister.fi

MAGISTER
SOLUTIONS

Why ns-3?

Available system-level simulators

- Most suitable:
  - **Network Simulator-2 (NS-2)** http://isi.edu/nsnam/ns/
  - **Network Simulator-3 (NS-3)** http://www.nsnam.org/
  - **OMNeT ++** http://www.omnetpp.org/
  - openWNS https://launchpad.net/openwns
  - LTE-Sim http://telematics.poliba.it/index.php/en/lte-sim
  - **The Vienna LTE simulators** http://www.nt.tuwien.ac.at/about-us/staff/josep-colom-ikuno/lte-simulators/
- Other open
  - JiST http://jist.ece.cornell.edu/
  - GoMoSim http://pcl.cs.ucla.edu/projects/glomosim/
- Proprietary
  - QualNet http://www.scalable-networks.com/products/qualnet/
  - NetSim http://tetcos.com/software.html

Simulators comparison

	ns-2	ns-3	OMNeT++	openWNS	Vienna LTE Simulator
License	GNU GPLv2	GNU GPLv2	Free for academy and education	LGPL	Academic usage only
Technologies	+	+	+	-	LTE link + system
Support/development	-	+	+	-	+
Real time integration	+-	+	+	-	-
Platform	C++ and OTcl	C++ and Python	C++ and NED	Python with C++ libraries	Matlab and C
Performance	-	+	+-	?	?

Advantages of Open Source tools

- All models are open, possible to check and validate
- Free of charge
- Big community for development, discussion, support
- Easy to develop/modify models on top of existing code

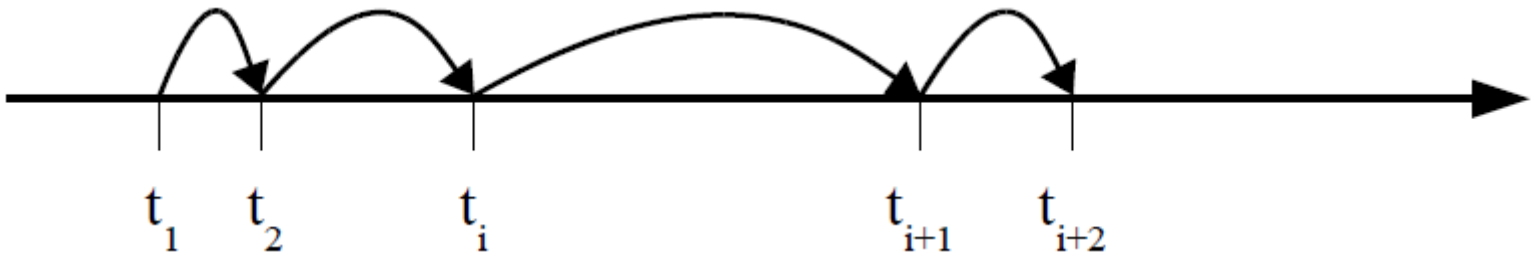
- Usual disadvantages:
  - Adjustments are needed to adopt to specific needs
  - No official support
  - How well code is managed and maintained (many separate repos)?

NS-3 in brief

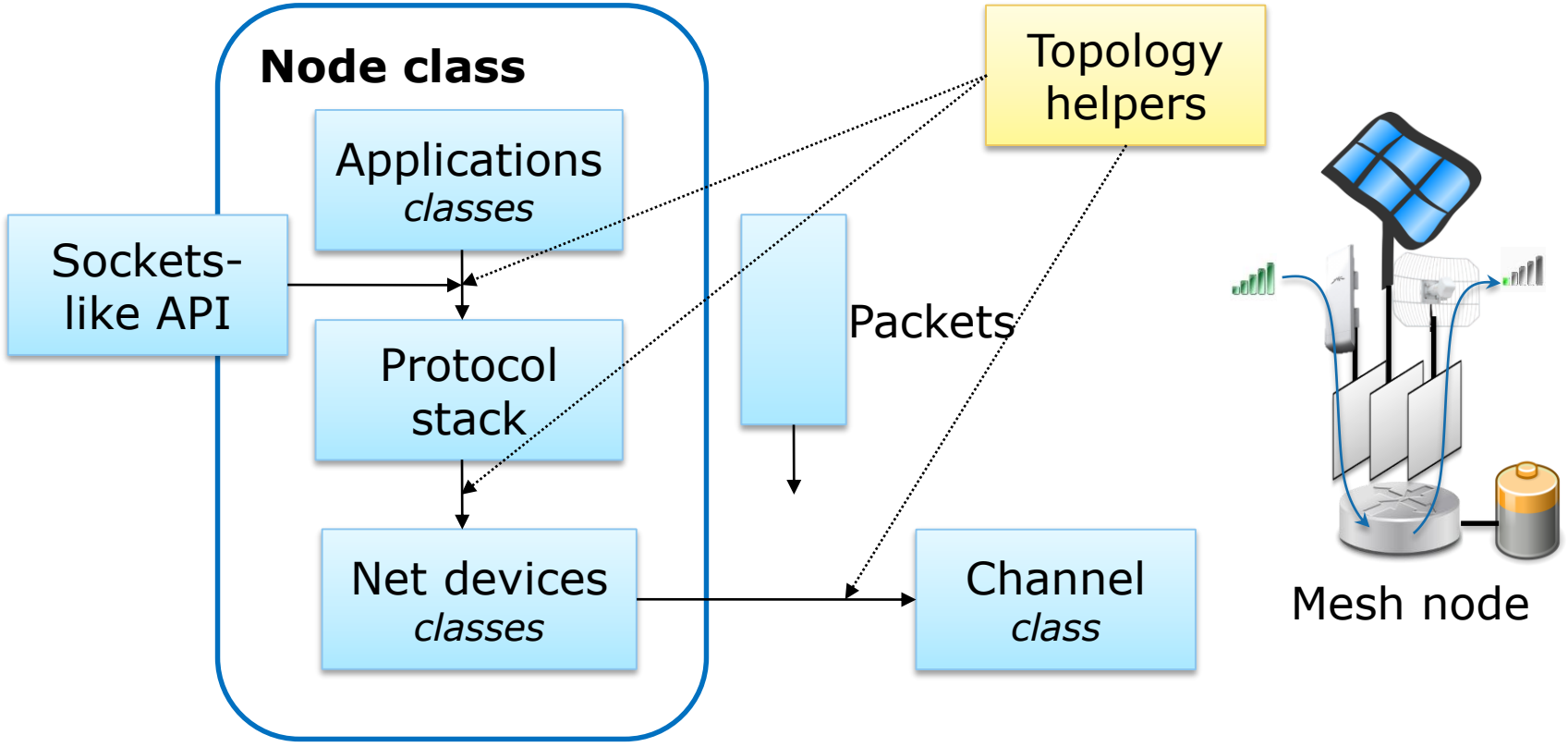
- Started in July 2006, the first release on June 30, 2008.
- Open-sours project (GPLv2)
- Simulation core and models are implemented in C++; Bindings in Python for python simulations
- NS-3 is a discrete-event network simulator
- Elaborated API, solid simulation core and module structure
- Logging and Tracing mechanisms
- Already big variety of existing models
- Targeted to be used as a realtime network emulator
- NS-2 experience was taken into account.
- Strict process of contribution, code review and maintenance

Discrete-event simulation

- The idea is to jump from one event to another
- Events are recoded in a future event list (FEL)
- Each event notice is composed at least to data: time, type
- Event routine or handler to process each event
- Each event may change the system state and generate new event notices

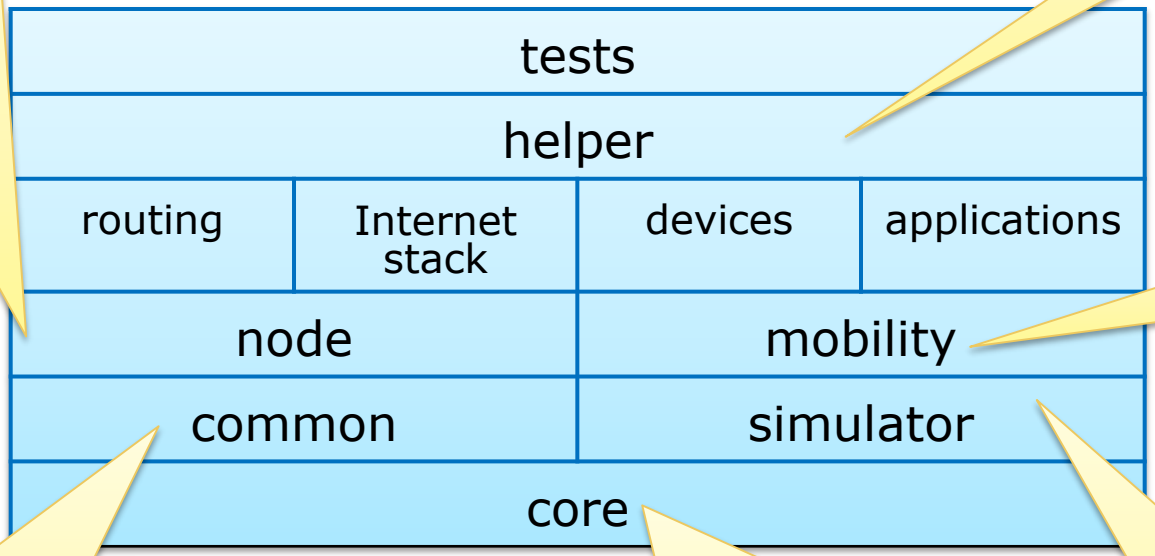


Main objects, attention to realism



- Model nodes like real computer
- Support key interfaces such as sockets API

NS 3 structure



Node class, NetDevice, Address types (Ipv4, MAC, …), Queues, Socket, Packet sockets

High-level wrapper
No smart pointers
Aimed at scripting

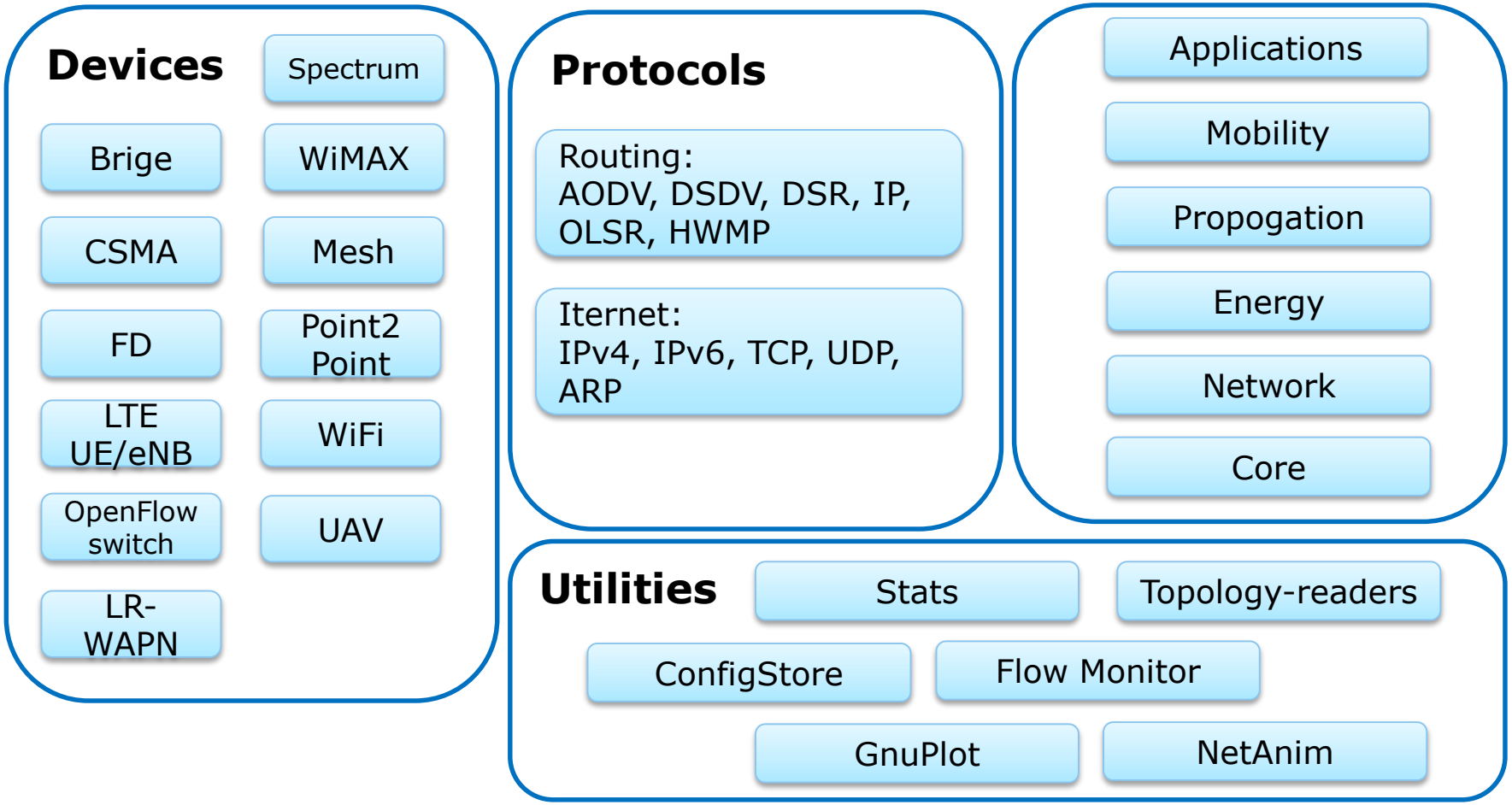
Mobility models (static, random walk, etc.)

Packets, Packet tags, Packet headers, Pcap/Ascii file writing

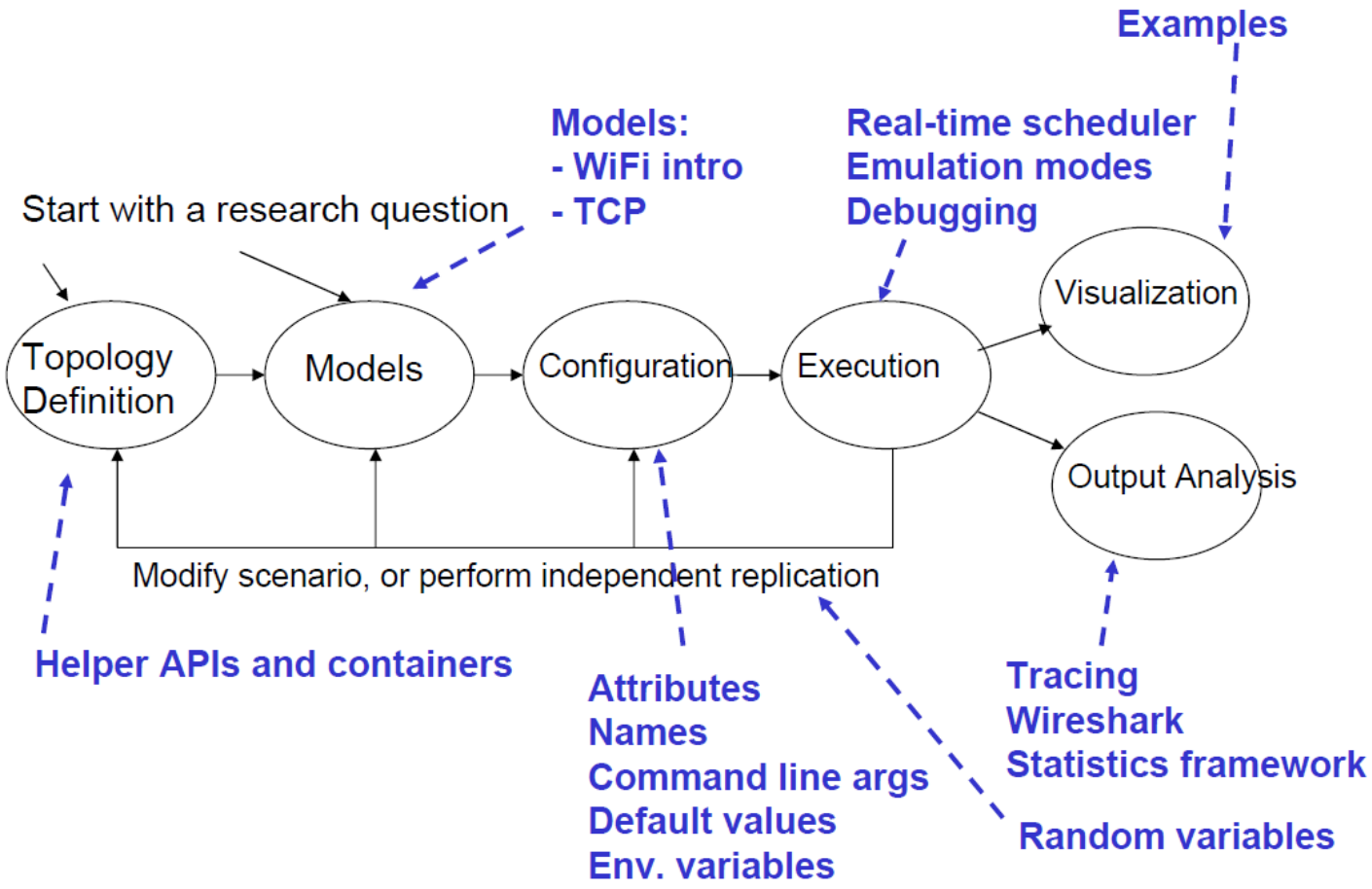
Smart pointers, Dynamic type systems, Attributes, Callbacks, Tracing, Logging, Random Variables

Events, Schedulers, Time arithmetic

ns-3 modules

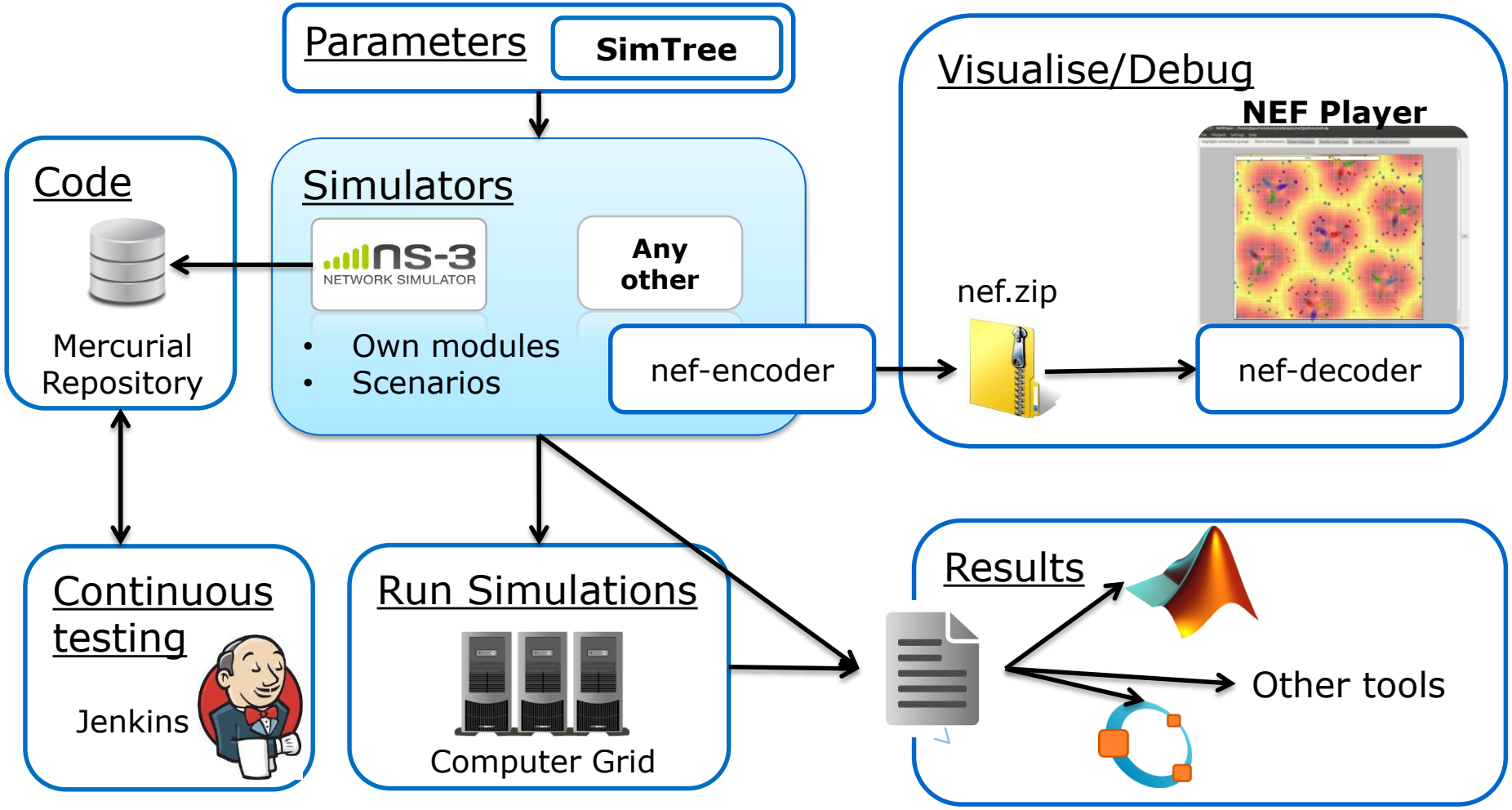


Workflow



* Tom Henderson and Mathieu Lacage, ns-3 tutorial, Workshop on ns-3, March 2009

Magister simulation environment



Documentation

- ns-3 tutorial: http://www.nsnam.org/docs/tutorial/singlehtml/index.html
- ns-3 manual: http://www.nsnam.org/docs/manual/singlehtml/index.html
- ns-3 model library: http://www.nsnam.org/docs/models/singlehtml/index.html
- Doxygen (automatically generated from the code, classes, attributes, etc.): http://www.nsnam.org/doxygen/index.html
- ns-3 LENA: http://iptechwiki.cttc.es/LTE-EPC_Network_Simulator_%28LENA%29
- ns-3 user group in Google: http://groups.google.com/group/ns-3-users

Ns-3 installation (UBUNTU)

- https://www.nsnam.org/wiki/Installation
- Install necessary packages
  `apt-get install gcc g++ python … etc.`
- Clone simulator from the repository
  - Bake installation (new)
    `hg clone http://code.nsnam.org/bake`
  - Manual installation
    hg clone http://code.nsnam.org/ns-3-allinone
    ./download.py -n ns-3-d
- Configure and build with **waf** tool (from simulator's root directory)
  ```
  ./waf configure --enable-examples --enable-tests
  ./waf build
  ```
- Validating installation `./test.py`
- Run `./waf --run hello-simulator`
- Configure your favorite IDE (Eclipse: https://www.nsnam.org/wiki/HOWTO_configure_Eclipse_with_ns-3)

Simulator folder structure

- scratch – easiest place to run/test your code
- Examples
- utils
- src/model
  - docs
  - examples
    - Folder of a particular example
    - wsciprt: what examples to compile
  - helper
  - model
  - tests
  - wscript: which models and libraries to build together with the module

```
def build(bld):
            obj = bld.create_ns3_program('hello-simulator', ['core'])
            obj.source = 'hello-simulator.cc'
```

Some core features and objects

MAGISTER
SOLUTIONS

A typical simulation scenario

- Create necessary C++ objects
- Configure and interconnect them
- Each object creates events in the simulator's timeline
- Run events execution

- Scenario pseudo code:

```
Node *a = new Node();
Node *b = new Node();
Link *link = new Link (a,b);
Simulator::Schedule (Seconds (0.5),   //in 0.5 sec from now
                &Node::StartCbr, a,   //call StartCbr() on node "a"
                "100bytes", "0.2ms", b );     //pass these arguments
Simulator::Run ();
```

The basic model

```
# /*
#  * This program is free software; you can redistribute it and/or modify
#  * it under the terms of the GNU General Public License version 2 as
#  * published by the Free Software Foundation;
#  *
#  * This program is distributed in the hope that it will be useful,
#  * but WITHOUT ANY WARRANTY; without even the implied warranty of
#  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#  * GNU General Public License for more details.
#  *
#  * You should have received a copy of the GNU General Public License
#  * along with this program; if not, write to the Free Software
#  * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
#  */

import ns3

ns3.LogComponentEnable("UdpEchoClientApplication", ns3.LOG_LEVEL_INFO)
ns3.LogComponentEnable("UdpEchoServerApplication", ns3.LOG_LEVEL_INFO)

ns3.RandomVariable.UseGlobalSeed(1, 1, 2, 3, 5, 8)

nodes = ns3.NodeContainer()
nodes.Create(2)

pointToPoint = ns3.PointToPointHelper()
pointToPoint.SetDeviceAttribute("DataRate", ns3.StringValue("5Mbps"))
pointToPoint.SetChannelAttribute("Delay", ns3.StringValue("2ms"))

devices = pointToPoint.Install(nodes)

stack = ns3.InternetStackHelper()
stack.Install(nodes)

address = ns3.Ipv4AddressHelper()
address.SetBase(ns3.Ipv4Address("10.1.1.0"), ns3.Ipv4Mask("255.255.255.0"))

interfaces = address.Assign (devices);
```

```
// Define your topology
Node n0 = new Node;
Node n1 = new Node;
AddInternetStack (n0, n1);
Channel c0 = new Channel;.
Connect (n0, n1, c0);
Application a0 = new
        TrafficGenerator;

// Configure things
a0.SetDataRate (1Mb/s);
a0.Start (10.0 seconds);

// Define outputs
WriteTraceFile ("outfile");

// Run the simulator
Simulator::Run();
```

Attribute system

- Set a default value:

```
Config::SetDefaultValue
("ns3::WifiPhy::TxGain",
DoubleValue (10));
```

- Set a value on a specific object:

```
phy->SetAttribute ("TxGain",
DoubleValue (10));
```

- Each object has a set of attributes:
  - A name, help text
  - A type
  - An initial value

- /NodeList/[i]/DeviceList/[i]/$ns3::WifiNetDevice/Phy/$ns3::YansWifiPhy

Attributes

- **EnergyDetectionThreshold**: The energy of a received signal should be h signal.
  - Set with class: **ns3::DoubleValue**
  - Underlying type: **double** -1.79769e+308:1.79769e+308
  - Initial value: -96
  - Flags: construct write read
- **CcaMode1Threshold**: The energy of a received signal should be higher t state
  - Set with class: **ns3::DoubleValue**
  - Underlying type: **double** -1.79769e+308:1.79769e+308
  - Initial value: -99
  - Flags: construct write read
- **TxGain**: Transmission gain (dB).
  - Set with class: **ns3::DoubleValue**
  - Underlying type: **double** -1.79769e+308:1.79769e+308
  - Initial value: 1
  - Flags: construct write read
- **RxGain**: Reception gain (dB).
  - Set with class: **ns3::DoubleValue**
  - Underlying type: **double** -1.79769e+308:1.79769e+308
  - Initial value: 1
  - Flags: construct write read
- **TxPowerLevels**: Number of transmission power levels available betwee
  - Set with class: **ns3::UintegerValue**
  - Underlying type: **uint32_t** 0:4294967295

The Helper/Container high-level API

- Aim:
  - Make it easy to build topologies with repeating patterns
  - Make the topology generation more high-level and easier to read and understand
- The idea:
  - Sets of objects (nodes, devices, interfaces,…) are stored in Containers
  - Operations are encoded in a Helper object and applies on a Container
- Examples:
  - `NodeContainer, NetDeviceContainer, Ipv4Container`
  - `InternetStackHelper, WifiHelper, MobilityHelper` each model usually has a helper class

Helper example

```
NodeContainer wifiStaNodes;
wifiStaNodes.Create (nWifi);

MobilityHelper mobility;

mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
                               "MinX", DoubleValue (0.0),
                               "MinY", DoubleValue (0.0),
                               "DeltaX", DoubleValue (5.0),
                               "DeltaY", DoubleValue (10.0),
                               "GridWidth", UintegerValue (3),
                               "LayoutType", StringValue ("RowFirst"));

mobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
                           "Bounds", RectangleValue (Rectangle (-50, 50, -50, 50)));

mobility.Install (wifiStaNodes);
```

Parametrisation and tracing

MAGISTER SOLUTIONS

Command line parameters

```
using namespace ns3;

NS_LOG_COMPONENT_DEFINE (“MyExample");
int main (int argc, char *argv[])
{
        bool verbose = true;
        uint32_t simTime = 3;

        CommandLine cmd;
        cmd.AddValue ("nCsma", "Number of \"extra\" CSMA nodes/devices", nCsma);
        cmd.AddValue (" simTime ", “Total simulation time, sec", simTime );
        cmd.AddValue ("verbose", "Tell echo applications to log if true", verbose);
        cmd.Parse (argc,argv);

        if (verbose)
        {
                LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
                LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
        }

        Simulator::Stop (Seconds (11.0));
        Simulator::Run ();
        Simulator::Destroy ();
        return 0;
}
```

ConfigStore module

- ns-3 attribute system:
  - Default attributes (override default values of the model before simulations)
  - Define and check attributes of existing objects (even at runtime)
  - All attributes are documented in Doxygen

- The **ConfigStore** is a specialized database for attribute values and default values.
  - Modes: save and load
  - File format: text and xml
  - Save/load attributes or default values:

```
configDef.ConfigureDefaults ();        // in very beginning
configAttr.ConfigureAttributes ();     // just before Run()
```

ConfigStore file examples (text)

//Default attributes:

```
default ns3::Settings::totalTime "+60.0s"
default ns3::Settings::interNbDist "500"
...
default ns3::LteEnbPhy::TxPower "30"
default ns3::LteEnbPhy::NoiseFigure "5"
default ns3::LteEnbNetDevice::DlEarfcn "100" default ns3::LteEnbNetDevice::UlEarfcn
"18100"
default ns3::LteHelper::Scheduler "ns3::PfFfMacScheduler"
default ns3::LteHelper::PathlossModel "ns3::FriisPropagationLossModel"
...
```

//Node attributes:

```
value
/$ns3::NodeListPriv/NodeList/3/$ns3::Node/DeviceList/0/$ns3::LteEnbNetDevice/LteEnbPhy/
$ns3::LteEnbPhy/TxPower "30"
value
/$ns3::NodeListPriv/NodeList/3/$ns3::Node/DeviceList/0/$ns3::LteEnbNetDevice/LteEnbPhy/
$ns3::LteEnbPhy/NoiseFigure "5"
value
/$ns3::NodeListPriv/NodeList/3/$ns3::Node/DeviceList/0/$ns3::LteEnbNetDevice/LteEnbPhy/
$ns3::LteEnbPhy/MacToChannelDelay "2"
```

ConfigStore file examples (xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<ns3>

 <default name="ns3::Settings::totalTime" value="+60.0s"/>
 <default name="ns3::Settings::sightsNumber" value="1"/>
 <default name="ns3::Settings::totalUeNumber" value="1"/>
...
 <default name="ns3::LteEnbPhy::TxPower" value="30"/>
 <default name="ns3::LteEnbPhy::NoiseFigure" value="5"/>
 <default name="ns3::LteEnbPhy::MacToChannelDelay" value="2"/>
 <default name="ns3::LteEnbPhy::UeSinrSamplePeriod" value="1"/>
 <default name="ns3::LteEnbPhy::InterferenceSamplePeriod" value="1"/>
...
<value
path="/$ns3::NodeListPriv/NodeList/4/$ns3::Node/DeviceList/0/$ns3::LteEnbNetDevice/LteE
nbMac/$ns3::LteEnbMac/RaResponseWindowSize" value="3"/>
 <value
path="/$ns3::NodeListPriv/NodeList/4/$ns3::Node/DeviceList/0/$ns3::LteEnbNetDevice/FfMa
cScheduler/$ns3::PfFfMacScheduler/CqiTimerThreshold" value="1000"/>

</ns3>
```

ConfigStore Gtk

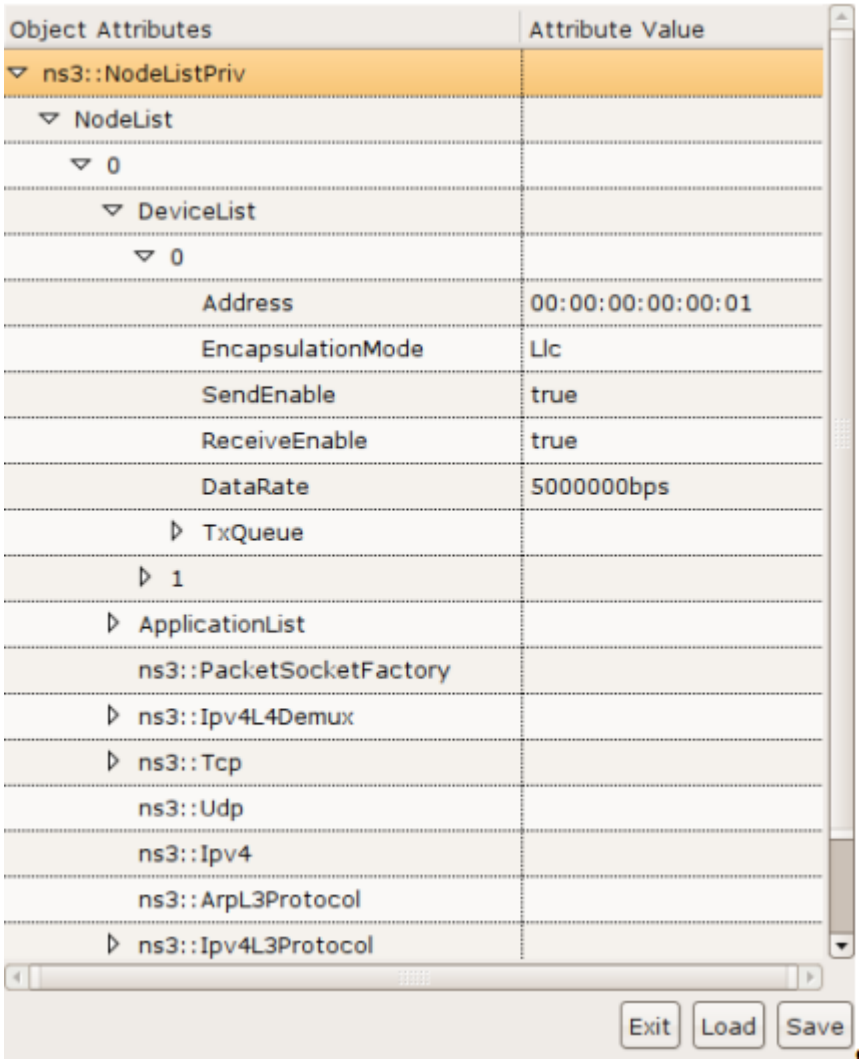
- Graphical configuration tool

```
int main (...)
{

  ... topology creation

  GtkConfigStore configstore;
  configstore.ConfigureAttributes();

  Simulator::Run ();
}
```



Logging in ns-3

- Usual C++ output (cout)
- Every ns-3 model has a defined logging component

```
NS_LOG_COMPONENT_DEFINE ("LteEnbPhy");
```

- Several levels of logging:
  - `NS_LOG_FUNCTION (this);`
  - `NS_LOG_ERROR ("UE already attached");`
  - `NS_LOG_INFO ("-----frame " << m_nrFrames << "-----");`
  - `NS_LOG_DEBUG (this << " eNB Expected TBs " << uldcilist.size ());`

  etc.
- Enabling logging

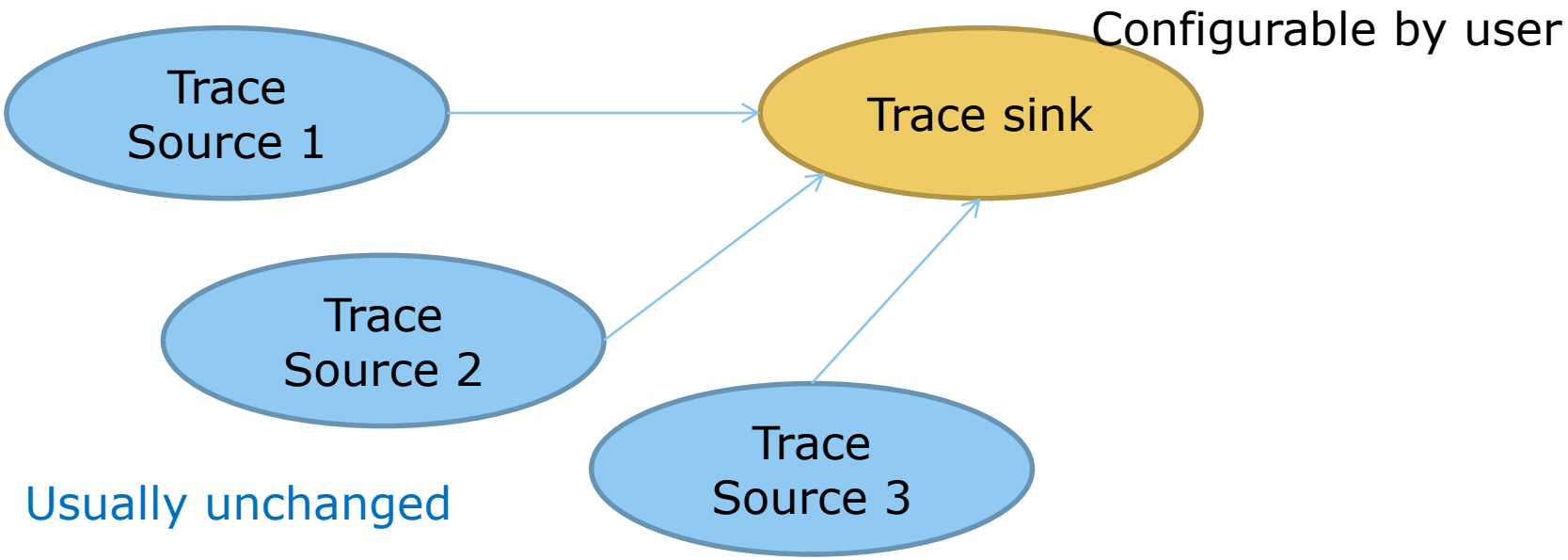
```
LogComponentEnable("UdpEchoClientApplication", LOG_LEVEL_INFO);
```

- Saving to a file

```
./waf --run scratch/myfirst > log.out 2>&1
```

ns-3 tracing model

- advantage: decoupling trace sources from trace sinks
- benefit: customizable trace sinks



Tracing in ns-3

- ns-3 configures multiple 'TraceSource' objects (TracedValues and Traced Callbacks)
- Multiple types (user-defined) 'TaceSink' objects can be hooked to these sources
- Namespace helps to manage access to trace sources

TracedValue

```
Config::Connect ("/path/to/traced/value", callback1);
```

TraceSource

```
Config::Connect ("/path/to/trace/source", callback2);
```

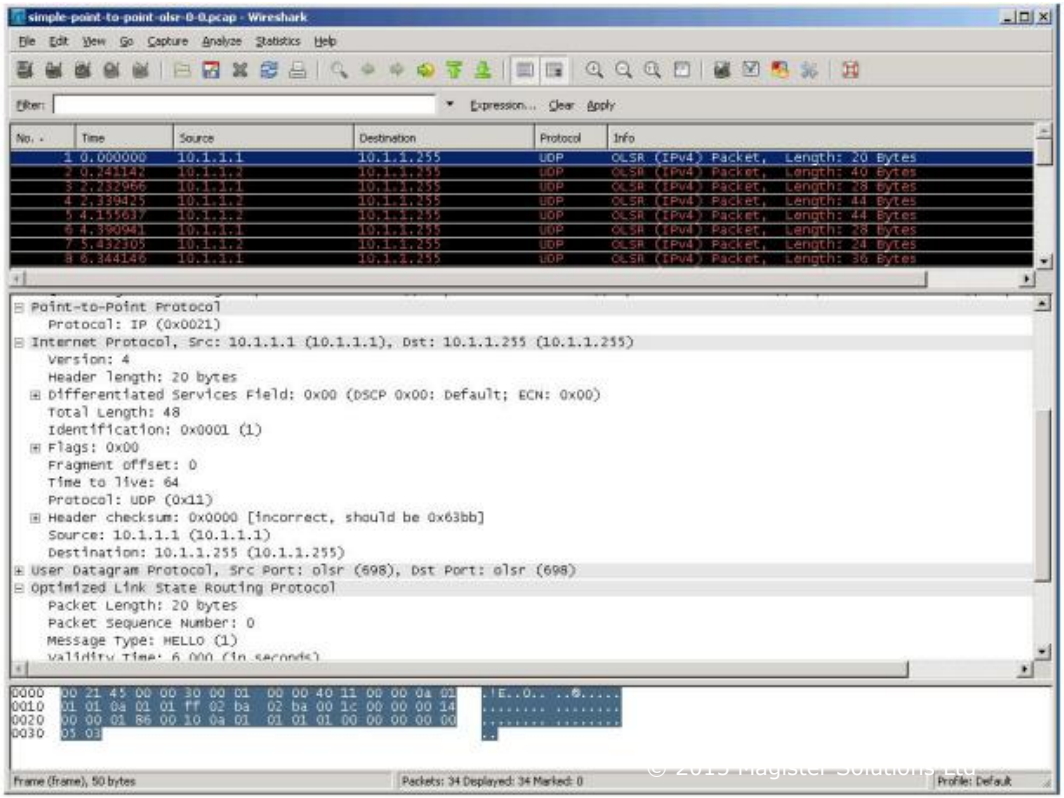
TraceSource

unattached

External PCAP tools

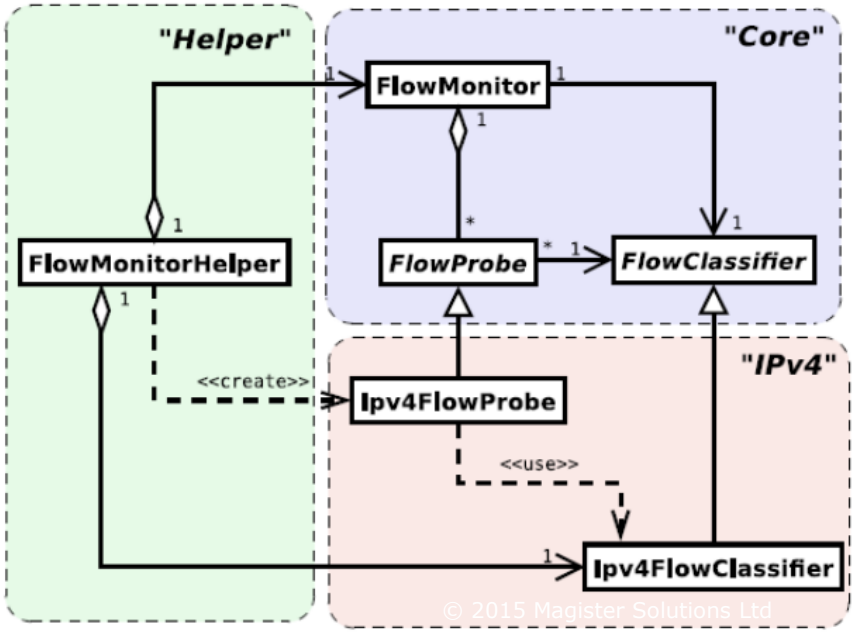
- ns-3 conforms to standard input/output formats so that other tools can be reused
  - Pcap trace output, ns-3 mobility scripts
  - `wifiPhy.EnablePcapAll (std::string ("mp-"));`

- Ns3 trace view with Wireshark

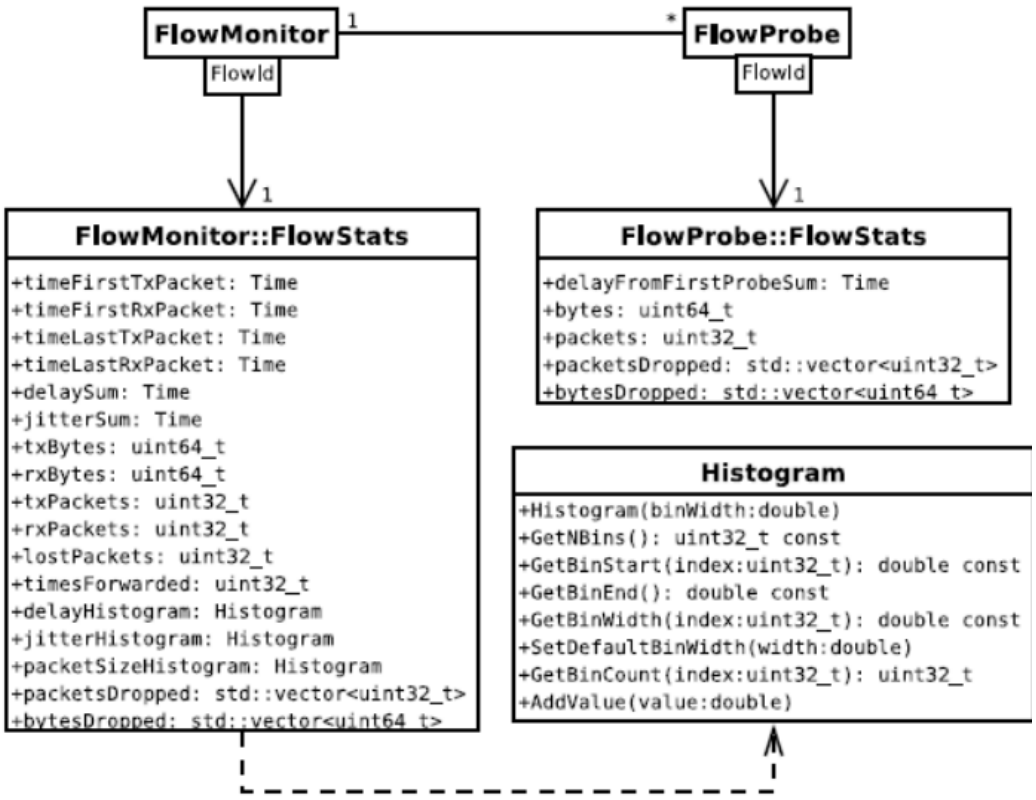


FlowMonitor architecture

- FlowMonitor is high-level (Ipv4 only) network monitoring framework
- Goals:
  - Detect all data flows passing though the network
  - Stores metrics for analysis such as bitrates, duration, delays, packet sizes, packet loss ratios
- Basic classes
  - FlowMoitor
  - FlowProbe
  - FlowClassifire
  - FlowMonitorHelper



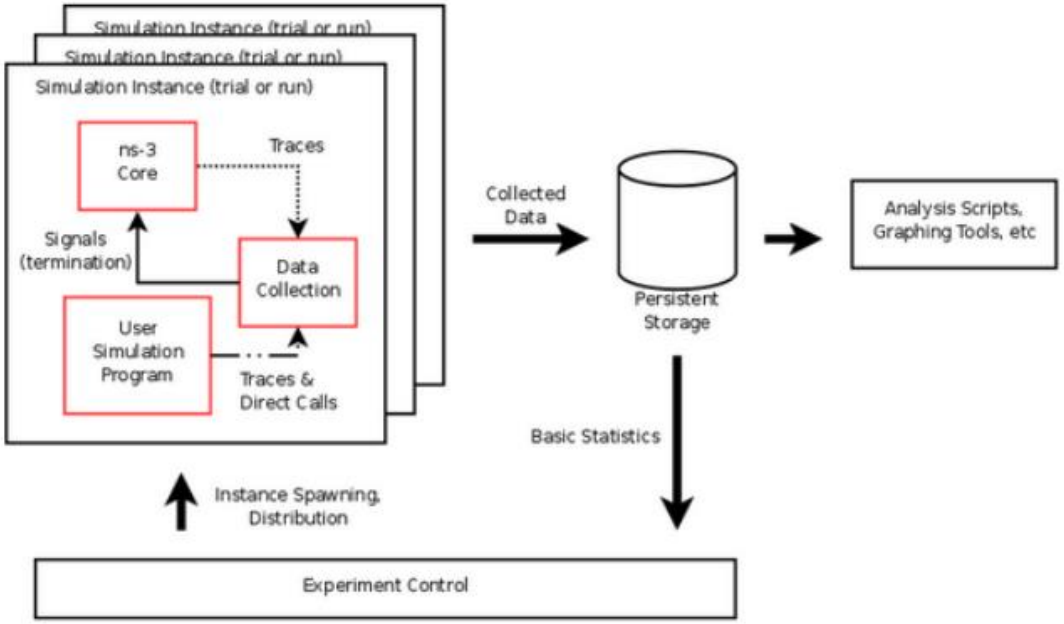
FlowMonitor output



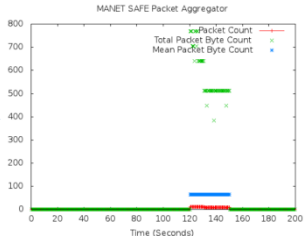

```
Hidden station experiment with RTS/CTS disabled:
Flow 1 (10.0.0.1 -> 10.0.0.2)
  Tx Bytes:   3847500
  Rx Bytes:   316464
  Throughput: 0.241443 Mbps
Flow 2 (10.0.0.3 -> 10.0.0.2)
  Tx Bytes:   3848412
  Rx Bytes:   336756
  Throughput: 0.256924 Mbps
------------------------------------------------
Hidden station experiment with RTS/CTS enabled:
Flow 1 (10.0.0.1 -> 10.0.0.2)
  Tx Bytes:   3847500
  Rx Bytes:   306660
  Throughput: 0.233963 Mbps
Flow 2 (10.0.0.3 -> 10.0.0.2)
  Tx Bytes:   3848412
  Rx Bytes:   274740
  Throughput: 0.20961 Mbps
```

Stats module for multiple experiments

- Experiment metadata:
  - Name of the experiment
  - Strategy/description
  - runID
- Data output in SQLLite format
- Basic statistical data calculator



New data collection framework



Data

Filter trace source data within time window

Compute statistics on Packet and byte counts

Gnuplot
Mathplotlib
other

ns-3 Lower level API

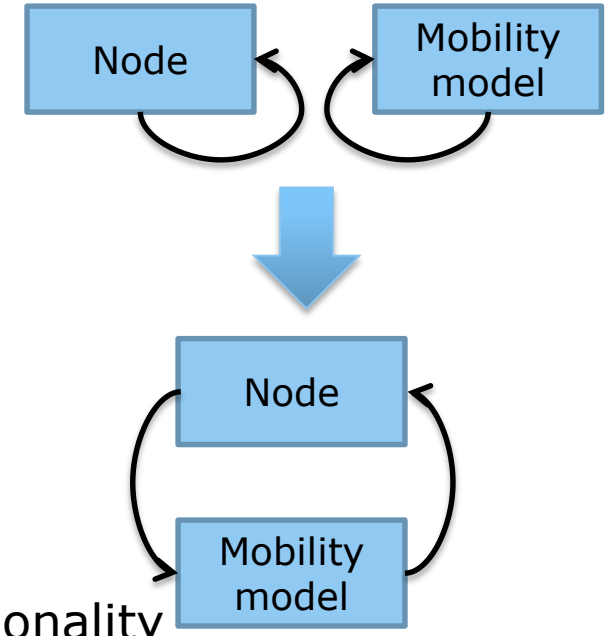
MAGISTER SOLUTIONS

Object aggregation

- Adding mobility model:

```
Ptr<Node> node = CreateObject<Node> ();
Ptr<MobilityModel> mobility = CreateObject<...> ();
node->AggregateObject (mobility);
```

- Usual problems:
  - Different nodes need different models
  - Add everything to the base class?
    - Uncontrollably grouth of the class
    - Everyone will need to patch this class
    - High code inter-dependence
- Solution:
  - Separate functionality to separate classes
  - Aggregation at runtime to add extra functionality



Getting objects

```
Ptr<NetDevice> dev = NodeList::Get (5)->GetDevice (0);
Ptr<WifiNetDevice> wifi = dev->GetObject<WifiNetDevice> ();
Ptr<WifiPhy> phy = dev->GetPhy ();
uint16_t chN = phy->GetChannelNumber ();
```

- In C++ to call a method of an Object its pointer is needed
- Usually, to get a pointer we have problems:
  - Keep local copiers of pointers at every object
  - Walk pointer chains to get Object from other Objects

Smart pointers

- Object creation in ns-3:

```
Ptr<Node> node0 = CreateObject<Node> ();
```

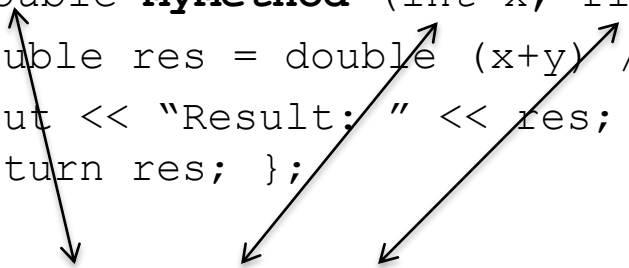
- Usual C++ problems:
  - No automatic garbarge collection
  - Easy to forget to delete an object
  - Pointer cycles

- So in ns-3:
  - Refernce counting: track number of pointers to and object
  - Smart pointers
    - Takes care of ref/unref counting work

Callback objects

- Ns-3 Callback class implement **function objects**
  - Type safe callbacks, manipulated by value
  - Used for modularization and tracing
- Example:

```
class MyClass {
public: double MyMethod (int x, float y) {
        double res = double (x+y) / 2;
        cout << "Result: " << res;
        return res; };
…
Callback<double, int, float> cb1;
MyClass myobj;
cb1 = MakeCallback (&MyClass::MyMethod, &myobj);
double result = cb1 (2,3);
```



ns-3 namespace

- Set attributes:

```
Config::SetAttribute (“NodeList/5/DeviceList/0/Phy/TxGain”,
StringValue (“10”));
```

- Connect trace sink to trace source:

```
Config::Connect (“NodeList/*/DeviceList/0/Phy/TxGain”,
MakeCallback(&LocalSink));
```

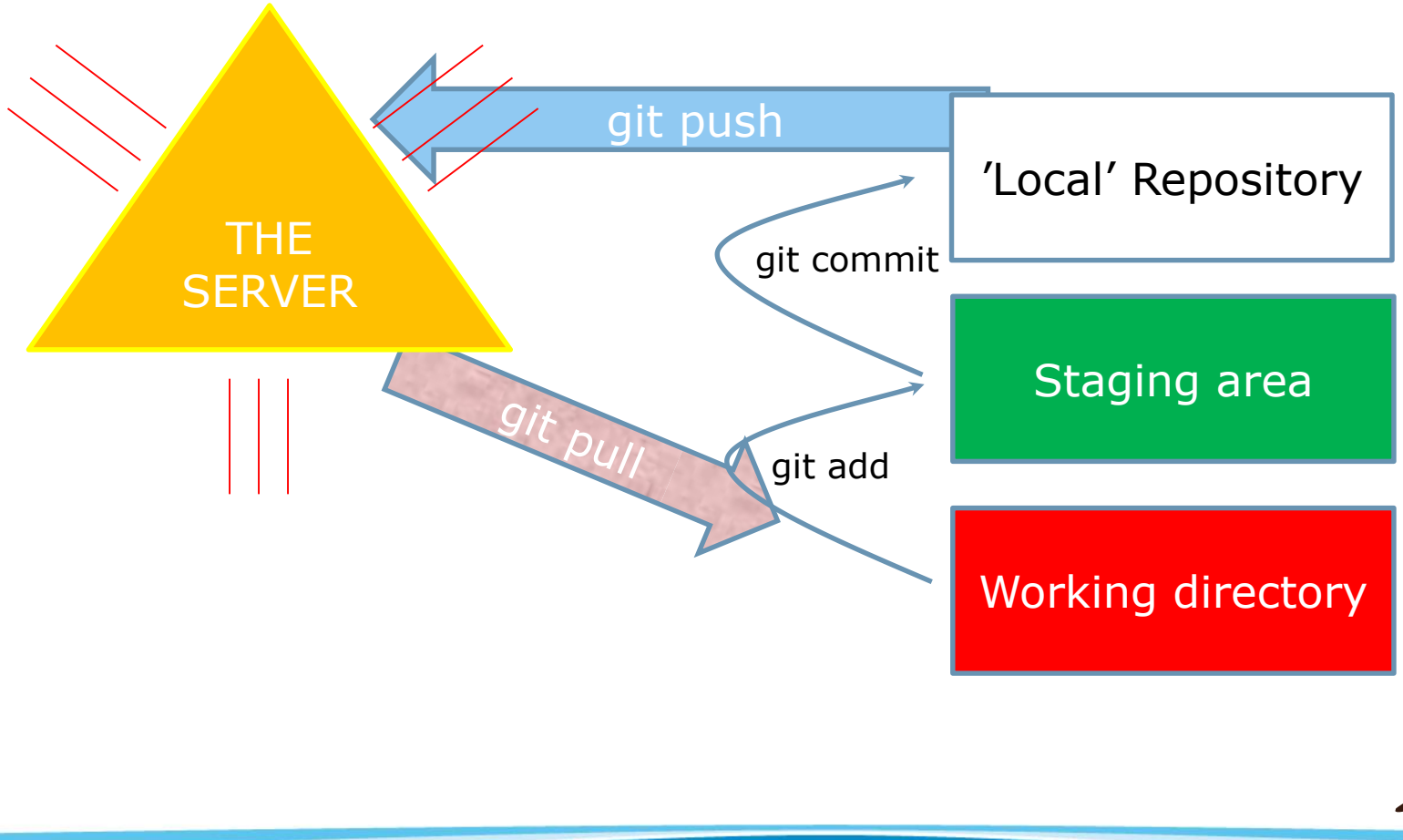
- Just get a pointer:

```
Config::ConnectContainer match;
match = Config::LookupMatches (“NodeList/5/DeviceList/0/Phy/”);
Ptr<WifiPhy> phy = match.Get(0)->GetObject<WifiPhy> ();
```

Practical part (seminar)

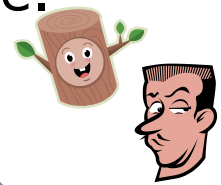
- Starting to work with the simulator, Eclipse
  - debugging
- First example from the scratch folder
  - Look though the code
  - Run simulations, output (NetAnim, Wireshark)
  - Adding trace sources
- Making new module
- Example with routing
  - Creating your own ns-3 object

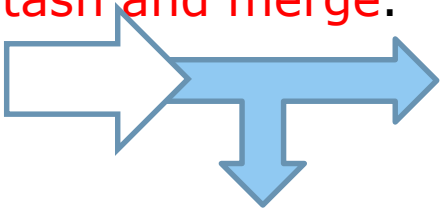
Contribute to the simulator!



Conflict management

- Git automatically merges when pulling, if there are diverging modifications.
- Conflicts arise when Git does not know how to merge.
  - Usually same line has been changed in different locations.
  - Manual merging needed.
- Strategies when parallel development is taking place.
  - You are happy with your code: Commit and merge.
  - You are still working on your code: Stash and merge.
  - You can do it all by yourself: Branch.





In case of conflict / problems

- Search the web for a solution.
- Ask your git expert friend.
- *Make an appointment with a psychologist.*

Thank you for attention!

Thanks to my colleagues!

Dmitry Petrov,
dmitry.petrov@magister.fi

MAGISTER SOLUTIONS