- About ns-3
- Installing ns-3
- Core concepts
  - Ns-3 Objects
  - Smart pointers
  - Object aggregation
  - Run-time type information
- Using Ns-3: step-by-step example
- Extending Ns-3
- Resources

ns-3

NETWORK SIMULATOR

- About ns-3
- Installing ns-3
- Core concepts
  - Ns-3 Objects
  - Smart pointers
  - Object aggregation
  - Run-time type information
- Using Ns-3: step-by-step example
- Extending Ns-3
- Resources

ns-3

NETWORK SIMULATOR

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.

www.nsnam.org

- Intended as the successor of Ns-2
  - Clean slate implementation: no re-use of Ns-2 code
  - Easier to use, more facilities, faster, more accurate, more flexible
- First version 3.1 June 2008
  - Current version 3.28
  - Available for Linux, OS X and Windows w/ Cygwin
- Currently 22,617 hits in ACM DL
  - In 2017: 2678
  - Opnet in 2017: 3
  - Omnet++ in 2017: 9
  - Ns-2 in 2017: 3495
- Written in C++
  - Simulation scripts in C++ (python optional)
  - Helper classes make “scripting” in C++ easy

- About ns-3
- Installing ns-3
- Core concepts
  - Ns-3 Objects
  - Smart pointers
  - Object aggregation
  - Run-time type information
- Using Ns-3: step-by-step example
- Extending Ns-3
- Resources

Installing Ns-3

- Simplest approach: download tar ball, extract and build

```
$ wget https://www.nsnam.org/release/ns-allinone-3.28.tar.bz2
$ tar jxfv ns-allinone-3.28.tar.bz2
$ cd ns-allinone-3.28/ns-3.28
$ ./waf configure --enable-examples
$ ./waf
```

  - Confirmed to work on Ubuntu 16.04.4 LTS

- For eclipse: see https://www.nsnam.org/wiki/HOWTO_configure_Eclipse_with_ns-3

- About ns-3
- Installing ns-3
- Core concepts
  - Ns-3 Objects
  - Smart pointers
  - Object aggregation
  - Run-time type information
- Using Ns-3: step-by-step example
- Extending Ns-3
- Resources

Ns-3 Objects

- Very important information to use ***and*** extend Ns-3
- Most objects inherit from ns3::Object
- Provides a range of useful properties for simulation
  - Smart pointers
  - Object aggregation
  - Run-time type information
  - Attributes
  - Trace sources

SimpleRefCount

ObjectBase

Object

- Ns-3 objects are created with CreateObject<Class> (constructor arguments)

Smart Pointers

- Provides a form of “garbage-collection”
- Enables object aggregation
- CreateObject returns smart pointer:

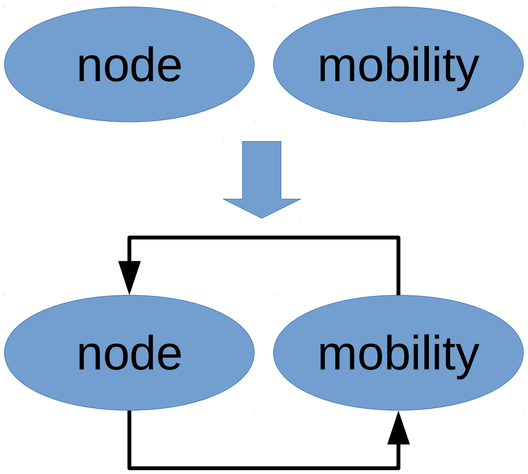
```
Ptr<PacketSocketFactory> factory =
    CreateObject<PacketSocketFactory> ();
```

- Always check return values and parameters: `Ptr<T>` or not?

Object Aggregation

- Objects can be dynamically aggregated to each other
- All objects in an aggregation can be access via any objects in said aggregation
  - Avoids huge classes that encompass all possible functionality

```
node->AggregateObject(mobility);

Ptr<MobilityModel> mob =
   node->GetObject<MobilityModel> ();
```



Run-Time Type Information

- All Ns-3 objects must implement **`TypeId GetTypeId(void)`**
- **`TypeId`** informs about attributes, runtime type information and trace sources

```
TypeId
RoutingProtocol::GetTypeId (void)
{
  static TypeId tid = TypeId ("ns3::olsr::RoutingProtocol")
    .SetParent<Ipv4RoutingProtocol> ()
    .SetGroupName ("Olsr")
    .AddConstructor<RoutingProtocol> ()
    .AddAttribute ("HelloInterval", "HELLO messages emission interval.",
                   TimeValue (Seconds (2)),
                   MakeTimeAccessor (&RoutingProtocol::m_helloInterval),
                   MakeTimeChecker ())
         ...
    .AddTraceSource ("RoutingTableChanged", "The OLSR routing table has changed.",
                     MakeTraceSourceAccessor (&RoutingProtocol::m_routingTableChanged),
                     "ns3::olsr::RoutingProtocol::TableChangeTracedCallback")
  ;
  return tid;
}
```

Run-Time Type Information

- Objects, attributes and trace sources can be located via textual paths via functions in the **`Config`** namespace (**objects**):

```
TypeId
RoutingProtocol::GetTypeId (void)
{
  static TypeId tid = TypeId ("ns3::olsr::RoutingProtocol")
    .SetParent<Ipv4RoutingProtocol> ()
    .SetGroupName ("Olsr")
    .AddConstructor<RoutingProtocol> ()

         ...
```

```
Config::MatchContainer m =
           Config::LookupMatches("NodeList/*/$ns3::olsr::RoutingProtocol”);

Ptr<Olsr::RoutingProtocol> olsr =
    m.Get(0)->GetObject<Olsr::RoutingProtocol> ();
```

Equivalent to:

```
nodes.Get(0)->GetObject<Olsr::RoutingProtocol> ();
```

Run-Time Type Information

- Objects, attributes an
  via textual paths via f
  namespace (**objects**

Example paths:

/NodeList/[3-5]|8|[0-1]
matches nodes index 0, 1, 3, 4, 5, 8

/NodeList/*
matches all nodes

/NodeList/3/$ns3::Ipv4
matches object of type ns3::Ipv4 aggregated to node number 3

/NodeList/3/DeviceList/*/$ns3::CsmaNetDevice
matches all devices of type ns3::CsmaNetDevice in node number 3

(See Doxygen for paths to particular objects)

```
TypeId
RoutingProtocol::GetTypeId (void)
{
  static TypeId tid = TypeId ("ns3::
    .SetParent<Ipv4RoutingProtocol>
    .SetGroupName ("Olsr")
    .AddConstructor<RoutingProtocol>

         ...
```

```
Config::MatchContainer m =
           Config::LookupMatches("NodeList/*/$ns3::olsr::RoutingProtocol”);

Ptr<Olsr::RoutingProtocol> olsr =
    m.Get(0)->GetObject<Olsr::RoutingProtocol> ();
```

Equivalent to:

```
nodes.Get(0)->GetObject<Olsr::RoutingProtocol> ();
```

Run-Time Type Information

- Objects, attributes and trace sources can be located via textual paths via functions in the **`Config`** namespace (**objects**):

```
TypeId
RoutingProtocol::GetTypeId (void)
{
  static TypeId tid = TypeId ("ns3::olsr::RoutingProtocol")
    .SetParent<Ipv4RoutingProtocol> ()
    .SetGroupName ("Olsr")
    .AddConstructor<RoutingProtocol> ()

        ...
```

Typecasting via object->GetObject<Class>

```
Config::MatchContainer m =
           Config::LookupMatches("NodeList/*/$ns3::olsr::RoutingProtocol”);

Ptr<Olsr::RoutingProtocol> olsr =
    m.Get(0)->GetObject<Olsr::RoutingProtocol> ();
```

Equivalent to:

```
nodes.Get(0)->GetObject<Olsr::RoutingProtocol> ();
```

Run-Time Type Information

- Objects, attributes and trace sources can be located via textual paths via functions in the **`Config`** namespace (**attributes**):

```
TypeId
RoutingProtocol::GetTypeId (void)
{
  static TypeId tid = TypeId ("ns3::olsr::RoutingProtocol")
    .SetParent<Ipv4RoutingProtocol> ()
    .SetGroupName ("Olsr")
    .AddConstructor<RoutingProtocol> ()
    .AddAttribute ("HelloInterval", "HELLO messages emission interval.",
                   TimeValue (Seconds (2)),
                   MakeTimeAccessor (&RoutingProtocol::m_helloInterval),
                   MakeTimeChecker ())
         ...
}
```

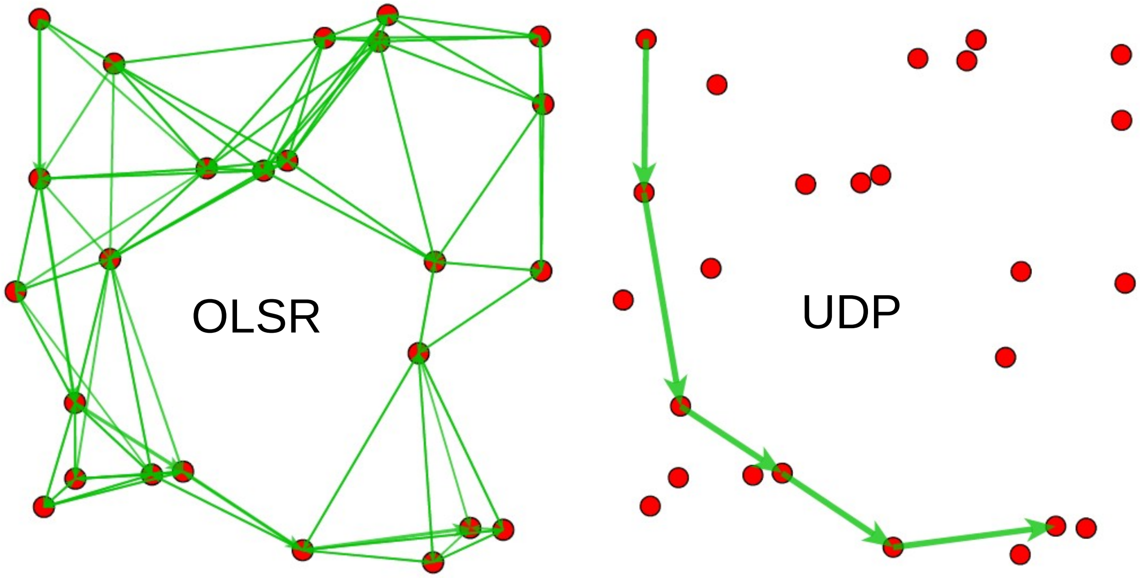
```
Config::SetDefault("ns3::olsr::RoutingProtocol::HelloInterval", TimeValue (Seconds(2)));
Config::Set("/NodeList/*/$ns3::olsr::RoutingProtocol/HelloInterval",
          TimeValue(Seconds(5)));
routingProtocolObject->SetAttribute("HelloInterval", TimeValue(Seconds(1)));
```

ns-3
NETWORK SIMULATOR

- About ns-3
- Installing ns-3
- Core concepts
  - Ns-3 Objects
  - Smart pointers
  - Object aggregation
  - Run-time type information
- Using Ns-3: step-by-step example
- Extending Ns-3
- Resources

Using Ns-3 – Via Example

- MANET with 25 nodes
  - Based on 802.11g wifi
- Routing protocol: OLSR
- Workload: uniform UDP traffic
  - 500-byte packets, 20 pps
  - No background traffic
- Mobility: random walk
  - Initial placement: 5 x ceil(5 / y) grid
  - y is the number of nodes
  - 100 meters between nodes
- Duration 10 minutes



9 Steps of an ns-3 Simulation Script

1) Handle command line arguments
2) Set default attribute values and random seed
3) Create nodes
4) Configure physical and MAC layers
5) Set up network stack, routing and addresses
6) Configure and install applications
7) Set up initial positions and mobility
8) Set up data collection
9) Schedule user-defined events and start simulation

Step 1: Command Line Arguments

- Enables parameterization of simulation from command line

```
int main (int argc, char *argv[])
{
  ...
  // Obtain command line arguments
  CommandLine cmd;
  cmd.AddValue ("cols", "Columns of nodes", cols);
  cmd.AddValue ("numnodes", "Number of nodes", numNodes);
  cmd.AddValue ("spacing", "Spacing between neighbouring nodes", nodeSpacing);
  cmd.AddValue ("duration", "Duration of simulation", duration);
  cmd.AddValue ("seed", "Random seed for simulation", seed);
  cmd.AddValue ("run", "Simulation run", run);
  cmd.AddValue ("packetrate", "Packets transmitted per second", packetRate);
  cmd.AddValue ("packetsize", "Packet size", packetSize);
  cmd.AddValue ("sourcenode", "Number of source node", sourceNode);
  cmd.AddValue ("destinationnode", "Number of destination node", destinationNode);
  cmd.AddValue ("showtime", "show ... time ... (default = true)", showSimTime);
  cmd.Parse (argc,argv);
  ...
```

For instance:

```
./waf –run "manet --nodespacing=50 --pktsize=100 --packetrate=500"
```

Step 2: Set Attribute Values and Random Seed

- Use `Config::-functions` to set default parameter values
- Remember to change run number between runs!

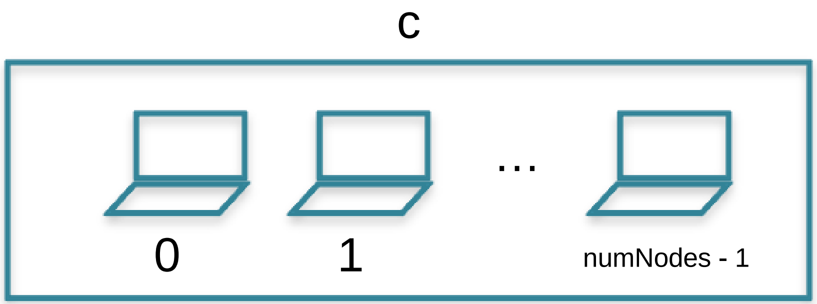
```
// Set default parameter values
Config::SetDefault("ns3::WifiRemoteStationManager::FragmentationThreshold",
  StringValue ("2200"));
Config::SetDefault("ns3::WifiRemoteStationManager::RtsCtsThreshold",
  StringValue ("2200"));

// Set random seed and run number
SeedManager::SetSeed (seed);
SeedManager::SetRun (run);
```

```
$ for run in "1 2 3"; do ./waf -run “manet --run=$run”; done
```

Step 3: Create Nodes

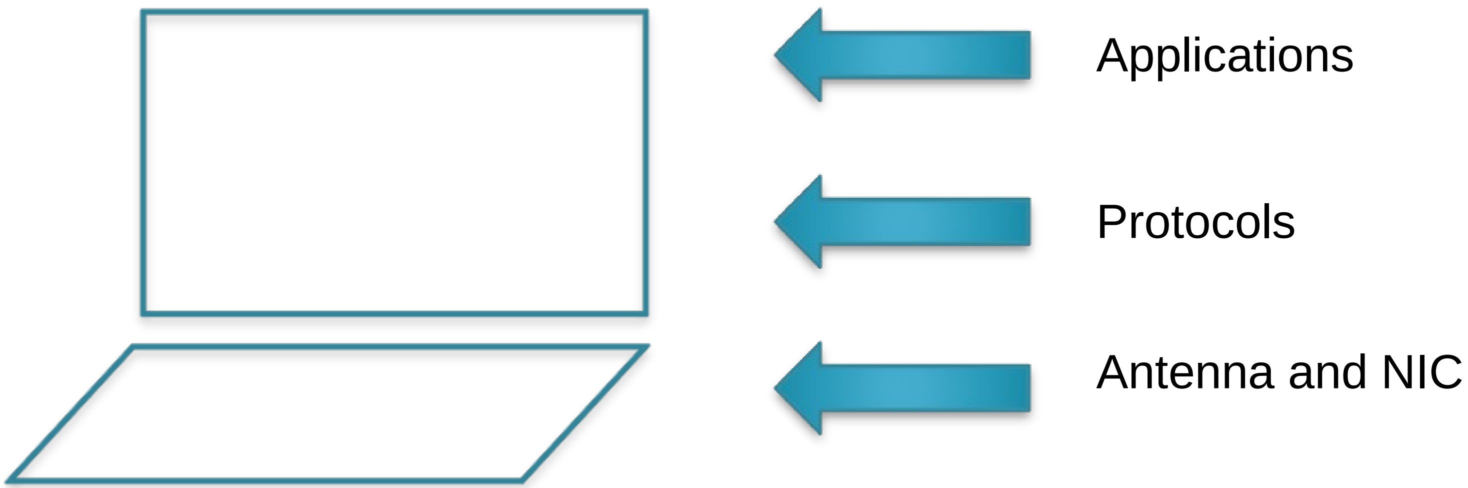
- Most objects in Ns-3 managed by ***containers***
  - Simulations consist of many objects of the same type
  - Later used by helper classes to install components
  - Entities in containers obtained with `container->get()`

```
// Create notes
NodeContainer c;
c.Create (numNodes);
...
apps = client.Install (c.Get (sourceNode));
```



Step 4-7: Configure Nodes

- Nodes are initially empty hulls



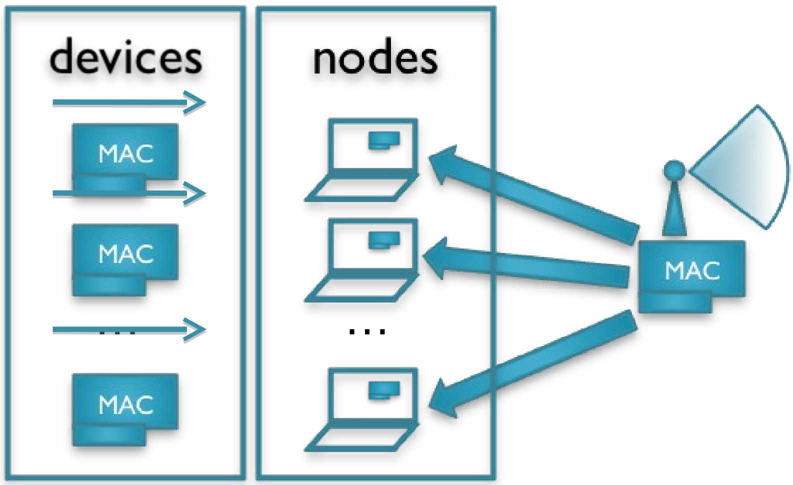
Step 4: Physical Layer

- ***Helpers*** enable script-like C++ programs
- Here:
  - 802.11g in ad-hoc mode
  - Automatic Rate Fallback (ARF)
    - Kamerman, Ad, and Leo Monteban. "WaveLAN®-II: a high-performance wireless LAN for the unlicensed band." Bell Labs technical journal 2.3 (1997): 118-133.
- Elsewise: default values
- Note that `wifi.install` uses node container `c`

```
// Set up physical and mac layers
WifiHelper wifi = WifiHelper::Default ();
wifi.SetStandard (WIFI_PHY_STANDARD_80211g);
wifi.SetRemoteStationManager ("ns3::ArfWifiManager");
NqosWifiMacHelper wifiMac = NqosWifiMacHelper::Default ();
wifiMac.SetType ("ns3::AdhocWifiMac");
YansWifiPhyHelper phy = YansWifiPhyHelper::Default ();
YansWifiChannelHelper wifiChannel = YansWifiChannelHelper::Default ();
phy.SetChannel (wifiChannel.Create ());
NetDeviceContainer devices = wifi.Install (phy, wifiMac, c);
```

Step 4: Physical Layer

- New container: `devices`

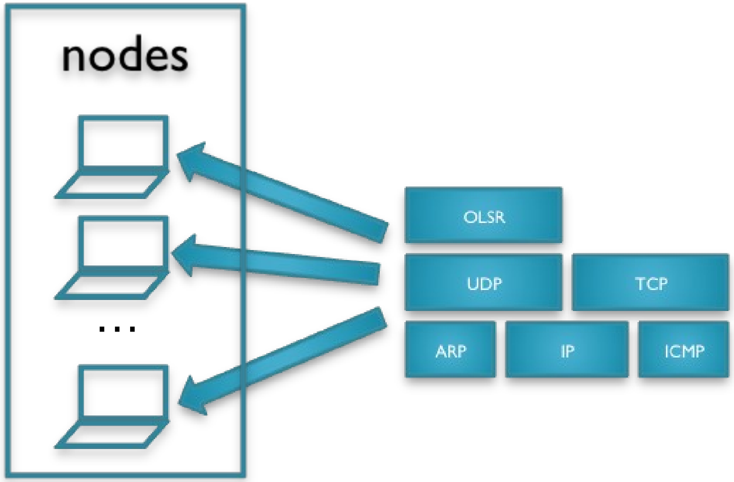



```
// Set up physical and mac layers
WifiHelper wifi = WifiHelper::Default ();
wifi.SetStandard (WIFI_PHY_STANDARD_80211g);
wifi.SetRemoteStationManager ("ns3::ArfWifiManager");
NqosWifiMacHelper wifiMac = NqosWifiMacHelper::Default ();
wifiMac.SetType ("ns3::AdhocWifiMac");
YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();
YansWifiChannelHelper wifiChannel = YansWifiChannelHelper::Default ();
YansWifiPhyHelper phy = wifiPhy;
phy.SetChannel (wifiChannel.Create ());
NetDeviceContainer devices = wifi.Install (phy, wifiMac, c);
```

Step 5: Install Internet Stack and Routing Protocol

- Select routing protocol
  - Ns-3 currently supports many routing protocols (e.g., OLSR, AODV, DSDV, ...)
  - Used in example: OLSR
- Internet stack: IP, TCP, UDP, ARP and ICMP

```
// Routing and Internet stack
ns3::OlsrHelper olsr;
InternetStackHelper internet;
internet.SetRoutingHelper(olsr);
internet.Install (c);
```



Step 5: Assign Addresses

- `c->Get(X)` gets
  IP address 10.0.0.(X+1)

  and
  MAC address 00:00:00:00:00:(X+1)

```
// Assign addresses
Ipv4AddressHelper address;
address.SetBase ("10.0.0.0", "255.255.255.0");
Ipv4InterfaceContainer interfaces = address.Assign (devices);
```

Step 6: Install Applications

- In example: simple UDP server and client
- Set attributes
- Specify when applications start and stop

```
// Server/Receiver
UdpServerHelper server (4000);
ApplicationContainer apps = server.Install (c.Get(destinationNode));
apps.Start (Seconds (1));
apps.Stop (Seconds (duration - 1));

// Client/Sender
UdpClientHelper client (interfaces.GetAddress (destinationNode), 4000);
client.SetAttribute ("MaxPackets", UintegerValue (100000000));
client.SetAttribute ("Interval", TimeValue (Seconds(1 / ((double) packetRate))));
client.SetAttribute ("PacketSize", UintegerValue (packetSize));
apps = client.Install (c.Get (sourceNode));
apps.Start (Seconds (1));
apps.Stop (Seconds (duration - 1));
```

The Ns-3 Node

- Node provides pointers to devices and applications

```
Ptr<Application> app = node->GetApplication(0);
Ptr<NetDevice> nic = node->GetDevice(0);
```

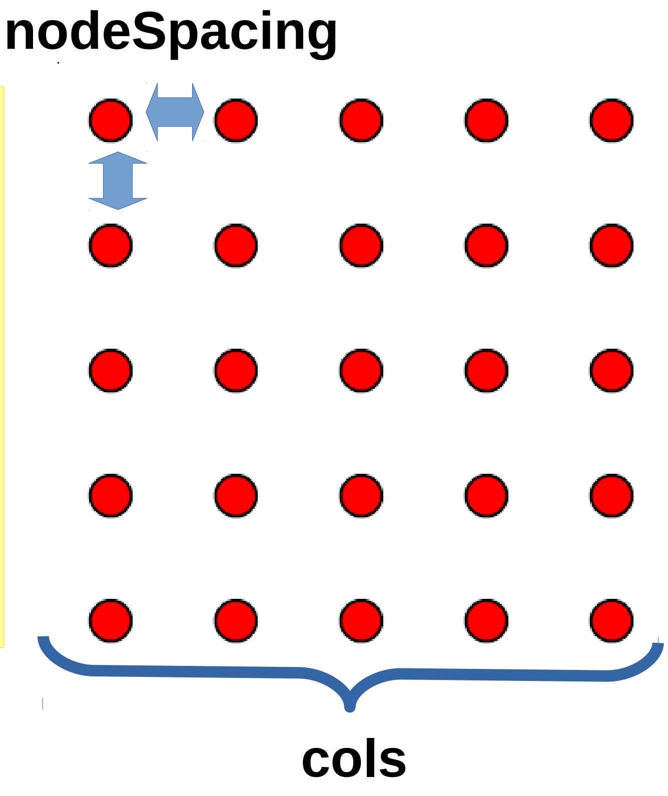
- Aggregated with stack, mobility model and energy model

```
Ptr<Ipv4> ip = nodes.Get(0)->GetObject<Ipv4>();
Ipv4Address addr = ip->GetAddress(1,0).GetLocal();
```

Step 7: Set up Initial Positions

- Several options available, including grid, disc, random placement and user-defined locations
  - Explained here: grid

```
// Set up mobility
MobilityHelper mobility;
mobility.SetPositionAllocator (
  "ns3::GridPositionAllocator",
  "MinX", DoubleValue (1.0),
  "MinY", DoubleValue (1.0),
  "DeltaX", DoubleValue (nodeSpacing),
  "DeltaY", DoubleValue (nodeSpacing),
  "GridWidth", UintegerValue (cols));
```

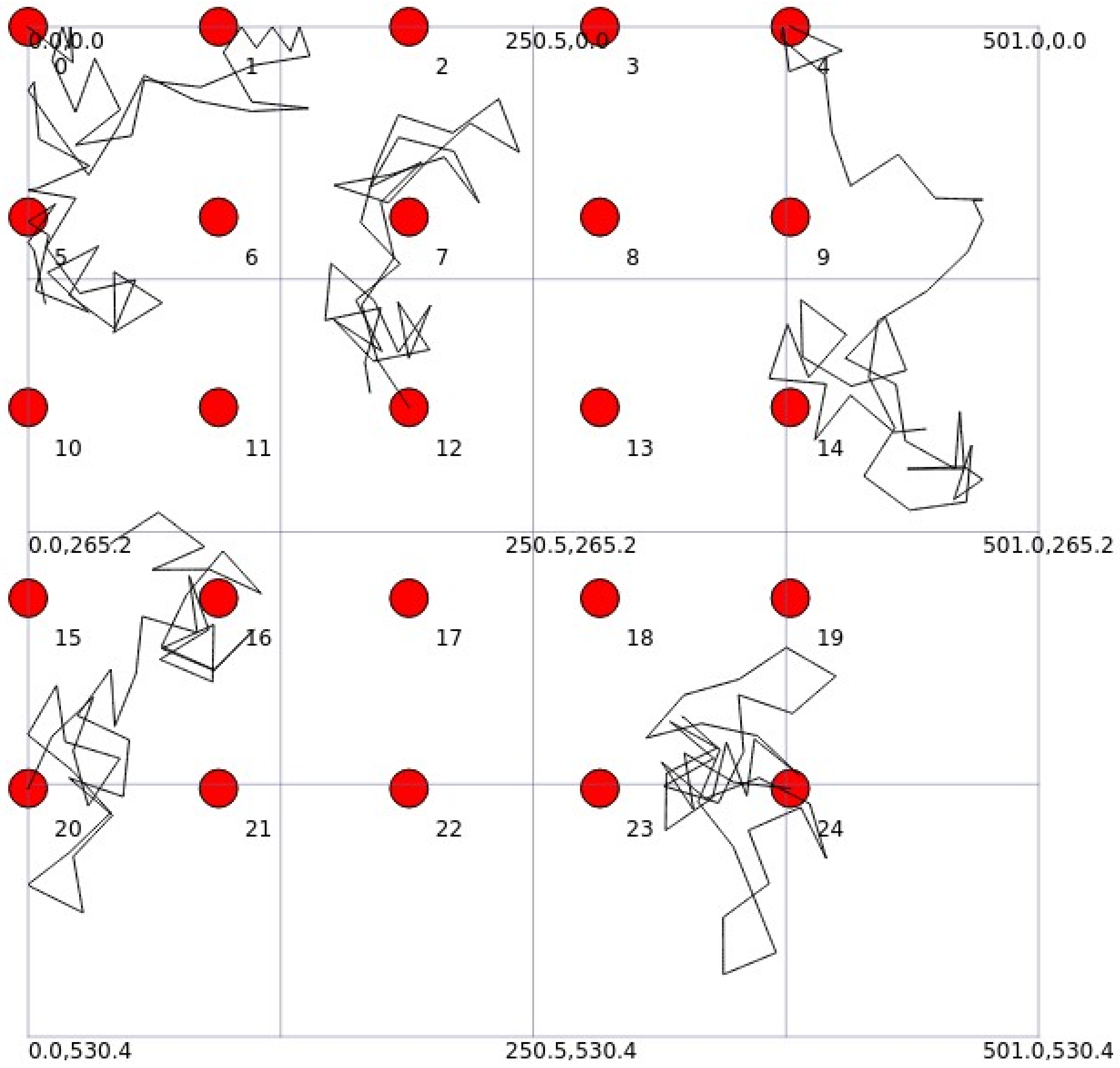


Step 7: Set Up Mobility Model

- Several alternatives
  - Random waypoint, random walk, user defined, …
- Used in Example: Random walk
  - Walk in random direction with random speed across fixed distance
    - Reflect upon hitting scenario boundaries
  - Speed defined with ***random variable***
  - Select new random direction

```
mobility.SetMobilityModel (
   "ns3::RandomWalk2dMobilityModel",
   "Bounds", RectangleValue
      (Rectangle (0,(cols * nodeSpacing) + 1,
          0,(rows * nodeSpacing) + 1)),
   "Speed",
      StringValue("ns3::UniformRandomVariable[Min=5.0,Max=10.0]"),
   "Distance", DoubleValue(30));

mobility.Install (c);
```

Example Mobility, 10 minutes



Step 9: Schedule Initial Events and Start Simulation

- Can schedule our own events before simulation
  - Example: print virtual time once every simulated second
- Simulation duration should be set

```
void PrintSeconds(void) {
    std::cerr << Simulator::Now() << std::endl;
    Simulator::Schedule(Seconds(1), &PrintSeconds);
}
```

```
    // Print simulated time
    if(showSimTime)
        Simulator::Schedule(Seconds(1), &PrintSeconds);

    Simulator::Stop(Seconds(duration));
    Simulator::Run ();
    Simulator::Destroy ();

    return 0;
}
```

Step 8: Data Collection

- Collect results = important step!
- Several alternatives
  1) std::cout << “Manual collection” << std::endl;
  2) Packet tracing
  3) Tracing subsystem
     - Low-Level: Trace sources and sinks
     - Medium/High-level: Data Collection Framework (DCF)
     - Highest-level: Statistics framework
  4) Logging via the logging facility *(see doc.)*
     - Intended for testing, debugging and verification

Step 8: Data Collection

- Collect results = important step!
- Several alternatives

  1) std::cout << "Manual collection" << std::endl;
  2) Packet tracing
  3) Tracing subsystem
     - Low-Level: Trace sources and sinks
     - Medium/High-level: Data Collection Framework (DCF)
     - Highest-level: Statistics framework
  4) Logging via the logging facility *(see doc.)*
     - Intended for testing, debugging and verification

Covered here

Packet Tracing

- Highly detailed packet models = enables real-world packet formats
- Popular packet capture format: PCAP
- One .pcap-file per node
- Pass device container from Step 4
- Set prefix (here "`MANET`")

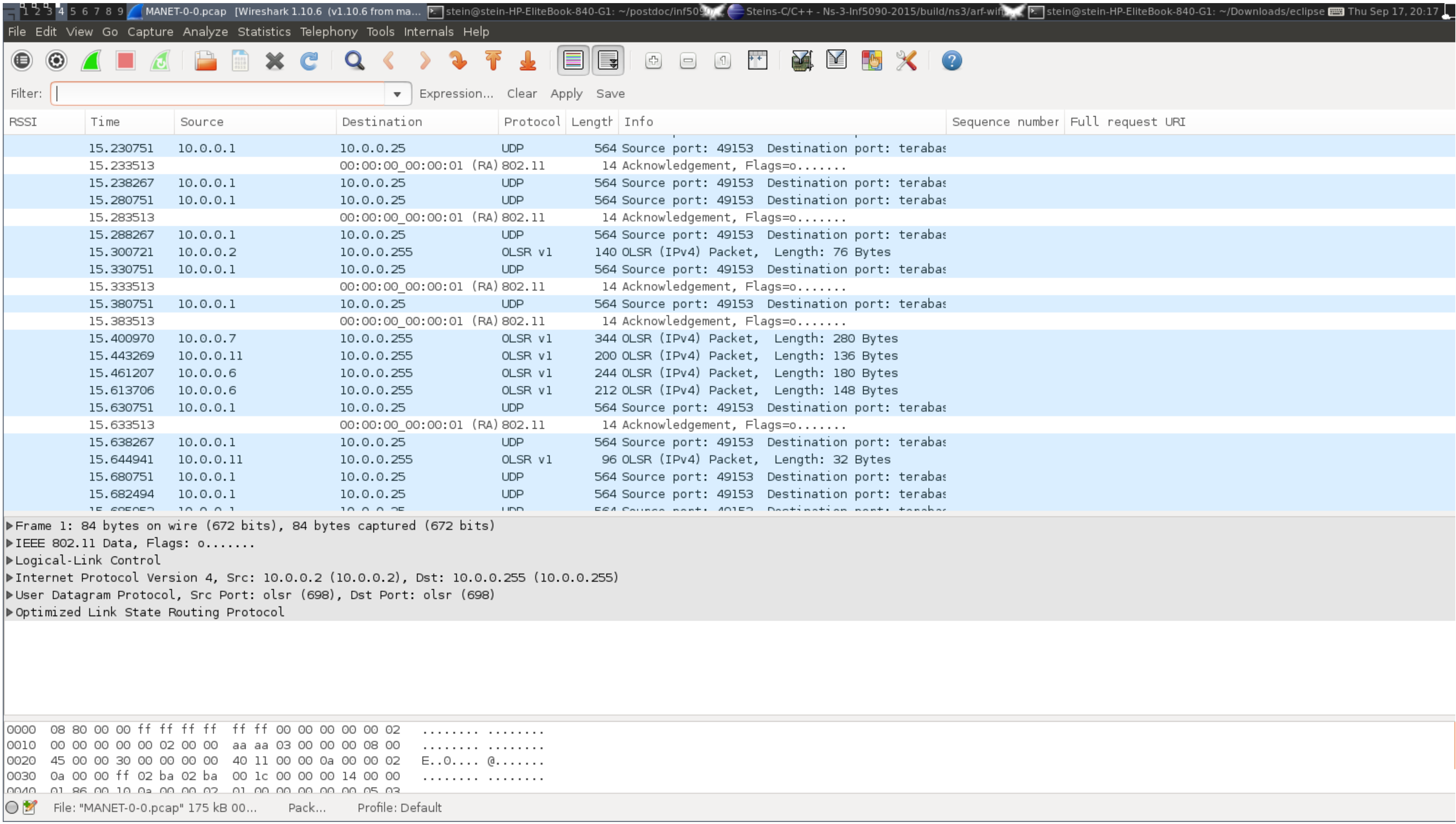
```
if(enablePcap)
   wifiPhy.EnablePcap ("MANET", devices);
```

Packet Tracing

- Resulting files: <prefix>-<node>-<device>.pcap

```
AUTHORS            MANET-17-0.pcap   routingtable-wireless.xml
bindings           MANET-18-0.pcap   scratch
build              MANET-19-0.pcap   src
CHANGES.html       MANET-20-0.pcap   test.py
doc                MANET-2-0.pcap    testpy.supp
dumbbell.xml       MANET-21-0.pcap   utils
examples           MANET-22-0.pcap   utils.py
LICENSE            MANET-23-0.pcap   utils.pyc
Makefile           MANET-24-0.pcap   VERSION
MANET-0-0.pcap     MANET-3-0.pcap    waf
MANET-10-0.pcap    MANET-4-0.pcap    waf.bat
MANET-1-0.pcap     MANET-5-0.pcap    waf-tools
MANET-11-0.pcap    MANET-6-0.pcap    wireless-animation.xml
MANET-12-0.pcap    MANET-7-0.pcap    wscript
MANET-13-0.pcap    MANET-8-0.pcap    wutils.py
MANET-14-0.pcap    MANET-9-0.pcap    wutils.pyc
MANET-15-0.pcap    README
MANET-16-0.pcap    RELEASE_NOTES
```

Can be opened in, e.g., Wireshark



Tracing Subsystem

- Based Ns-3 callbacks and attributes
- De-couples trace sources and sinks

Example trace source (from Ns-3 manual)

```
class MyObject : public Object
{
public:
  static TypeId GetTypeId (void)
  {
    static TypeId tid = TypeId ("MyObject")
      .SetParent (Object::GetTypeId ())
      .AddConstructor<MyObject> ()
      .AddTraceSource ("MyInteger",
                       "An integer value to trace.",
                       MakeTraceSourceAccessor (&MyObject::m_myInt))
      ;
    return tid;
  }

  MyObject () {}
  TracedValue<uint32_t> m_myInt;
};
```

Tracing Subsystem

- Based Ns-3 callbacks and attributes
- De-couples trace sources and sinks

Example trace sink (from Ns-3 manual)

```
void
IntTrace (Int oldValue, Int newValue)
{
  std::cout << "Traced " << oldValue << " to " << newValue << std::endl;
}

int
main (int argc, char *argv[])
{
  Ptr<MyObject> myObject = CreateObject<MyObject> ();

  myObject->TraceConnectWithoutContext ("MyInteger", MakeCallback(&IntTrace));

  myObject->m_myInt = 1234;
}
```

Tracing Subsystem

- Based Ns-3 callbacks and attributes
- De-couples trace sources and sinks

Example trace sink (from Ns-3 manual)

```
void
IntTrace (Int oldValue, Int newValue)
{
  std::cout << "Traced " << oldValue << " to " << newValue << std::endl;
}

int

  Ptr<MyObject> myObject = CreateObject<MyObject> ();

  myObject->TraceConnectWithoutContext ("MyInteger", MakeCallback(&IntTrace));

  myObject->m_myInt = 1234;
}
```

```
Traced 0 to 1234
```

Tracing Subsystem

- Objects, attributes and trace sources can be located via textual paths using functions in the **`Config`** namespace (**trace sources**):

```
TypeId
RoutingProtocol::GetTypeId (void)
{
        ...
    .AddTraceSource ("RoutingTableChanged", "The OLSR routing table has changed.",
                     MakeTraceSourceAccessor (&RoutingProtocol::m_routingTableChanged),
                     "ns3::olsr::RoutingProtocol::TableChangeTracedCallback")
        ...
```

```
void RouteChange(std::string source, uint32_t size) {
    std::cout << "Routechange at " << source << ", new size: " << size << std::endl;
}
```

```
Config::Connect("/NodeList/*/$ns3::olsr::RoutingProtocol/RoutingTableChanged",
    MakeCallback (&RouteChange));

routingProtocolObject->TraceConnect(“RoutingTableChanged”, “Context info”,
    MakeCallback(&RouteChange));
```

Tracing Subsystem

- Objects, attributes and trace sources can be located via textual paths via functions in the **`Config`** namespace (**trace sources**):

```
TypeId
```

```
Routechange at /NodeList/2/$ns3::olsr::RoutingProtocol/RoutingTableChanged, new size: 8
Routechange at /NodeList/11/$ns3::olsr::RoutingProtocol/RoutingTableChanged, new size: 13
Routechange at /NodeList/15/$ns3::olsr::RoutingProtocol/RoutingTableChanged, new size: 9
...
```

```
                    "ns3::olsr::RoutingProtocol::TableChangeTracedCallback")
        ...
```

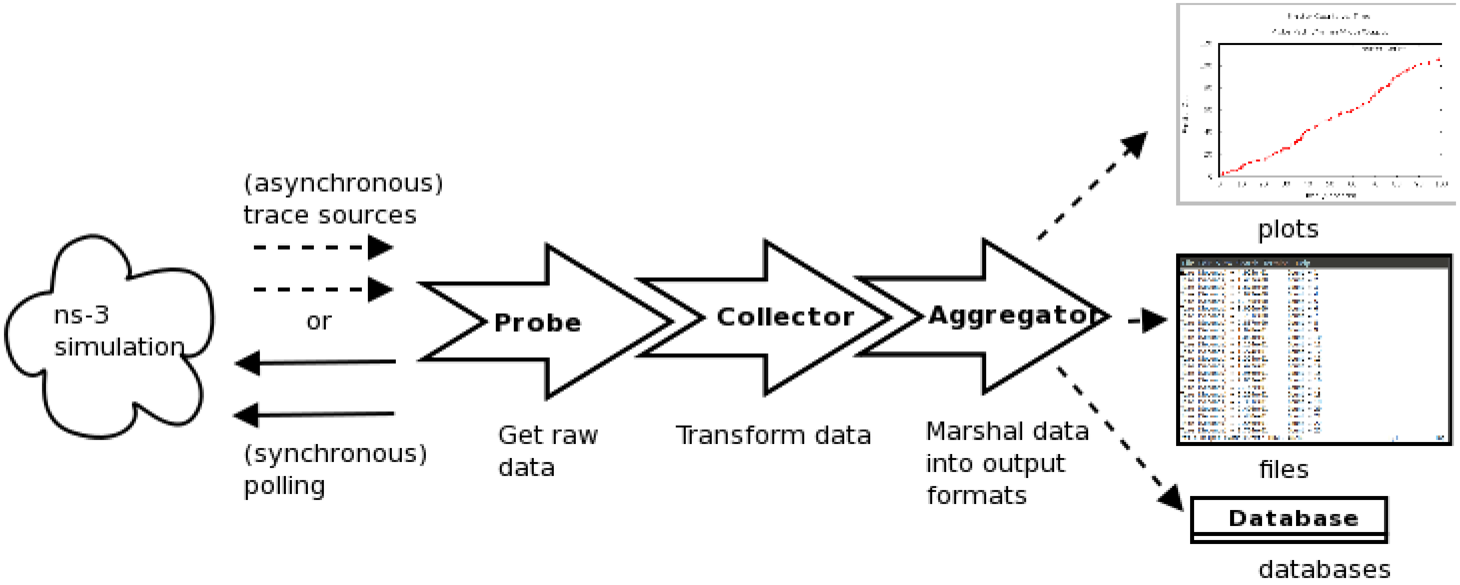
```
void RouteChan  std::string source, uint32_t
    std::cout   "Routechange at " << source
}
```

```
Routechange at Source node, new size: 5
...
```

```
Config::Connect("/NodeList/*/$ns3::olsr::RoutingProtocol/Routing  leChanged",
    MakeCallback (&RouteChange));

routingProtocolObject->TraceConnect("RoutingTableChanged", "Source node",
    MakeCallback(&RouteChange));
```

Data Collection Framework (DCF)

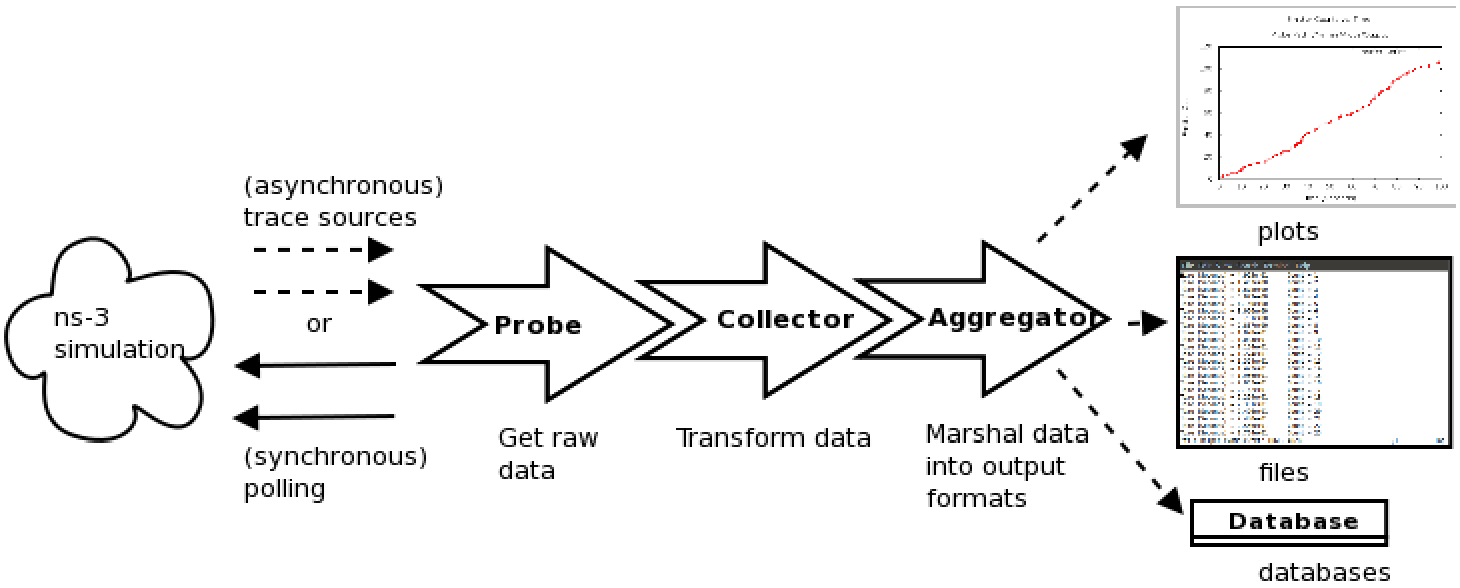
- Based on tracing subsystem
- On-line data reduction and processing
- Output format marshaling



https://www.nsnam.org/docs/release/3.28/manual/singlehtml/index.html#document-data-collection

Data Collection Framework (DCF)

- Two helpers currently implemented:
  - FileHelper
  - GnuplotHelper
- Additional supported: SQLList and OMNet++



https://www.nsnam.org/docs/release/3.28/manual/singlehtml/index.html#document-data-collection

DCF Example: FileHelper

ns-allinone-3.28/ns-3.28/examples/tutorial/seventh.cc

```
FileHelper fileHelper;

fileHelper.ConfigureFile ("seventh-packet-byte-count",
                          FileAggregator::FORMATTED);

fileHelper.Set2dFormat ("Time (Seconds) = %.3e\tPacket Byte Count = %.0f");

fileHelper.WriteProbe ("ns3::Ipv4PacketProbe",
                       "/NodeList/*/$ns3::Ipv4L3Protocol/Tx",
                       "OutputBytes");
```

- Output:

```
seventh-packet-byte-count-0.txt
seventh-packet-byte-count-1.txt
```

```
Time (Seconds) = 1.000e+00    Packet Byte Count = 40
Time (Seconds) = 1.004e+00    Packet Byte Count = 40
Time (Seconds) = 1.004e+00    Packet Byte Count = 576
Time (Seconds) = 1.009e+00    Packet Byte Count = 576
Time (Seconds) = 1.009e+00    Packet Byte Count = 576
Time (Seconds) = 1.015e+00    Packet Byte Count = 512
Time (Seconds) = 1.017e+00    Packet Byte Count = 576
Time (Seconds) = 1.017e+00    Packet Byte Count = 544
Time (Seconds) = 1.025e+00    Packet Byte Count = 576
Time (Seconds) = 1.025e+00    Packet Byte Count = 544

...
```

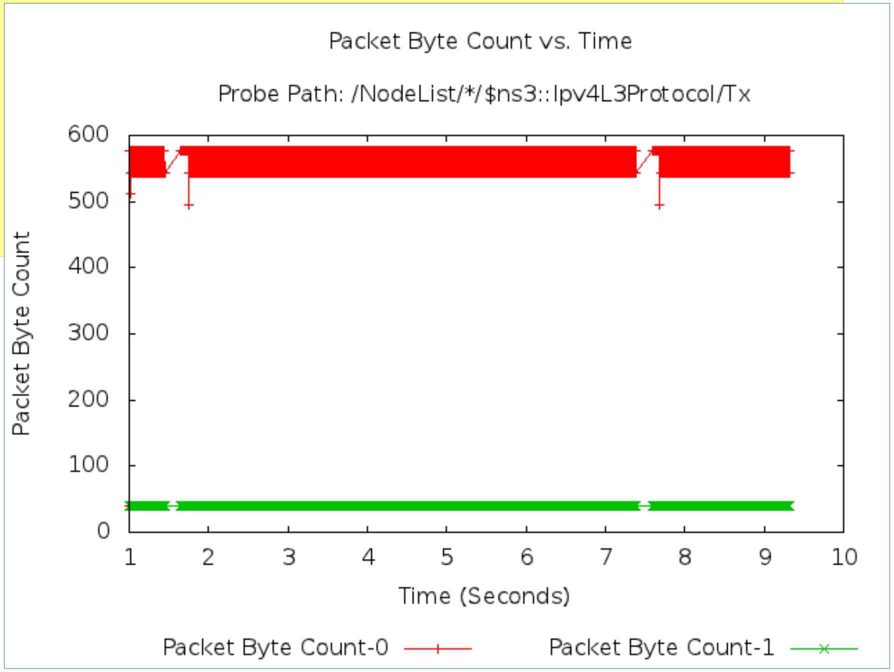
DCF Example: GnuplotHelper

```
GnuplotHelper plotHelper;

plotHelper.ConfigurePlot ("seventh-packet-byte-count",
                          "Packet Byte Count vs. Time",
                          "Time (Seconds)",
                          "Packet Byte Count");

plotHelper.PlotProbe (“ns3::Ipv4PacketProbe”,
                    “/NodeList/*/$ns3::Ipv4L3Protocol/Tx”,
                     "OutputBytes",
                     "Packet Byte Count",
                     GnuplotAggregator::KEY_BELOW);
```

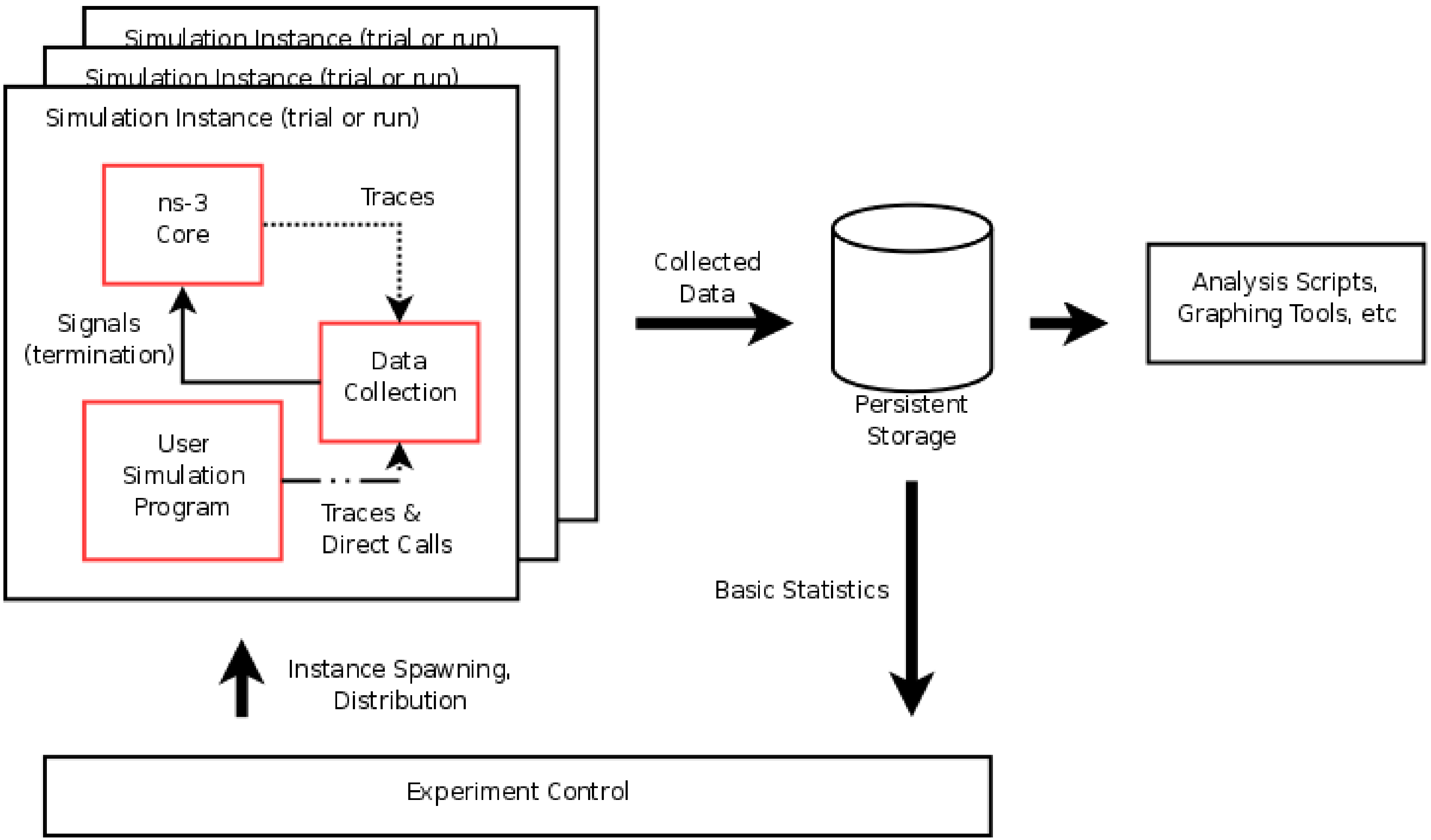
ns-allinone-3.28/ns-3.28/examples/tutorial/seventh.cc

- Output:

```
seventh-packet-byte-count.dat (data file)
seventh-packet-byte-count.plt (gnuplot script)
seventh-packet-byte-count.sh (runs .plt)
```



Statistics Framework



https://www.nsnam.org/docs/release/3.28/manual/singlehtml/index.html#document-data-collection

- About ns-3
- Installing ns-3
- Core concepts
  - Ns-3 Objects
  - Smart pointers
  - Object aggregation
  - Run-time type information
- Using Ns-3: step-by-step example
- Extending Ns-3
- Resources

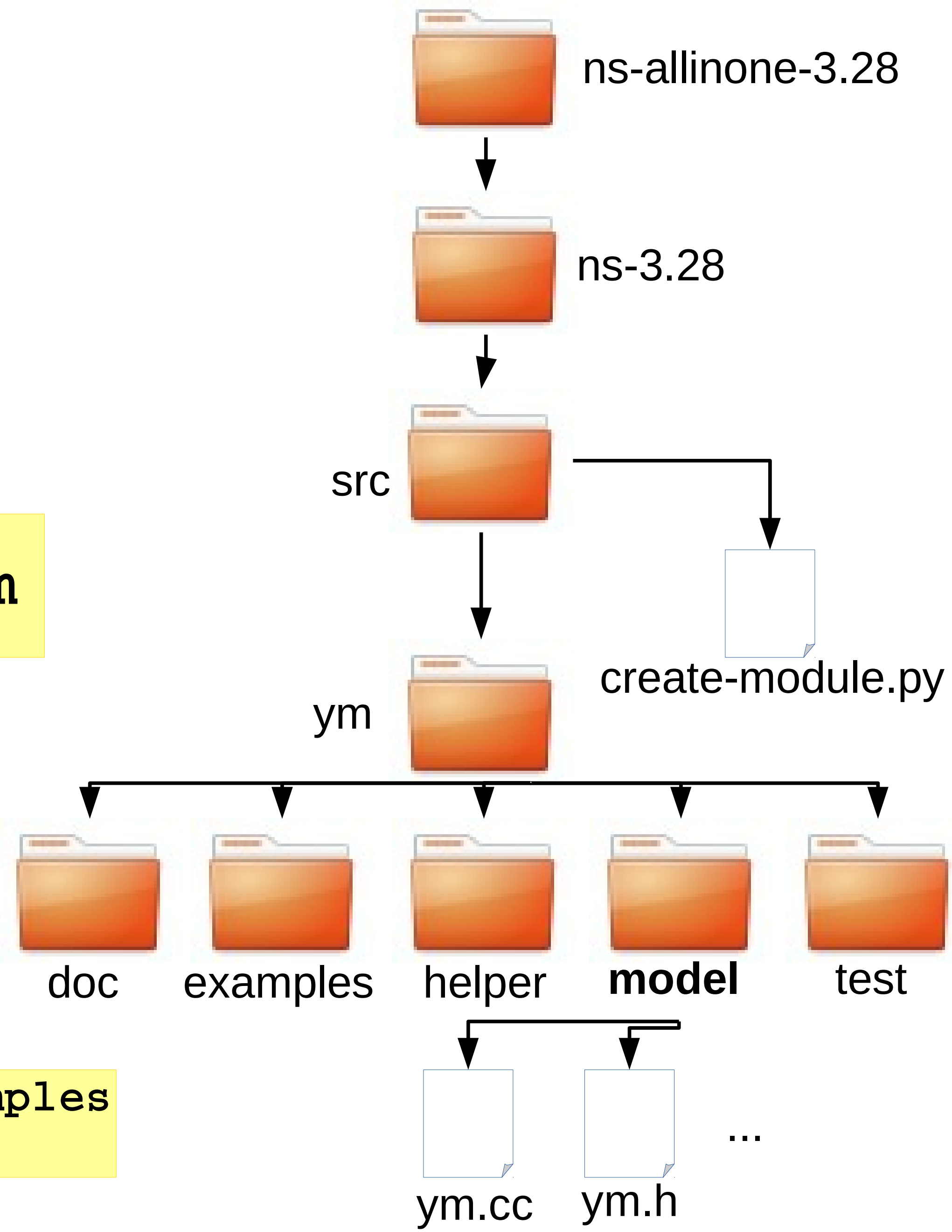
Extending Ns-3

- Prerequisite: C++ knowledge
- Module-based
- Create template with create-module.py

```
$ create-module.py ym
```

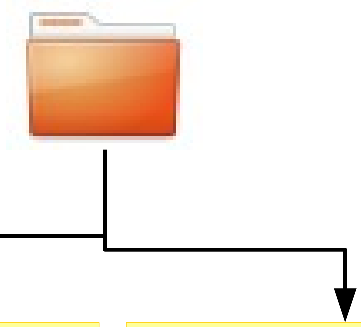
- Creates five folders
  - Your model in **“model”**
- **MUST reconfigure before re-compilation**

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



Resulting .cc and .h files in

model



```
...
#include "ym.h"

namespace ns3 {

/* ... */


}
```

ym.cc

```
...
#ifndef YM_H
#define YM_H

namespace ns3 {

/* ... */

}

#endif /* YM_H */
```

ym.h

Resulting .cc and .h files in

helper 

```
...
#include "ym-helper.h"

namespace ns3 {

/* ... */


}
```

ym-helper.cc

```
...
#ifndef INF5090_HELPER_H
#define INF5090_HELPER_H

#include "ns3/ym.h"

namespace ns3 {

/* ... */

}

#endif /* INF5090_HELPER_H */
```

ym-helper.h

Resulting .cc and .h files in 

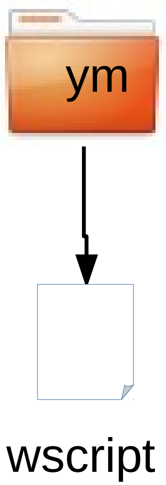
examples

example.cc:

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */

#include "ns3/core-module.h"
#include "ns3/ym-helper.h"

using namespace ns3;


int
main (int argc, char *argv[])
{
  bool verbose = true;

  CommandLine cmd;
  cmd.AddValue ("verbose", "Tell application to log if true", verbose);

  cmd.Parse (argc,argv);

  /* ... */

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

When Adding Files, Update wscript

```
...
def build(bld):
    module = bld.create_ns3_module('ym', ['core'])
    module.source = [
        'model/ym.cc',
        'helper/ym-helper.cc',
        ]

    module_test = bld.create_ns3_module_test_library('ym')
    module_test.source = [
        'test/ym-test-suite.cc',
        ]

    headers = bld(features='ns3header')
    headers.module = 'ym'
    headers.source = [
        'model/ym.h',
        'helper/ym-helper.h',
        ]

    if bld.env.ENABLE_EXAMPLES:
        bld.recurse('examples')

    # bld.ns3_python_bindings()
```



- About ns-3
- Installing ns-3
- Core concepts
  - Ns-3 Objects
  - Smart pointers
  - Object aggregation
  - Run-time type information
- Using Ns-3: step-by-step example
- Extending Ns-3
- Resources

- www.nsnam.org
- www.nsnam.org/wiki
- www.nsnam.org/documentation
  - Ns-3 manual
  - Ns-3 tutorial https://www.nsnam.org/docs/release/3.28/tutorial/html/index.html
  - Doxygen
  - Slides
  - Videos
  - ...
- Examples in the source code

Appendix

- Summary of simulation concepts
- Static routes
- User defined locations
- Constant positions
- The Ns-3 logging facility

Discrete-Event Simulation Concepts

Concept	Network Simulation Example
System	The Internet, MANET, WSN, ...
Model	C++ classes, math formulas, ...
Model state	C++ objects, packets, node positions, ...
Entity	Link, queue, packet, protocol, ...
Attributes	Link capacity, queue size, packet type, ...
List	Packets in a queue, nodes in a subnet, ...
Event	Transmission/arrival of packet, packet drop, ...
Event notice	Ns-3: Scheduler::Event (obj. w/ func. pointer)
Event list	Ns-3: DefaultSimulatorImpl::m_events
Activity	Transmission delay, part of movement, ...
Delay	Queuing delay, end-to-end delay, ...
Clock	Ns-3: DefaultSimulatorImpl::m_currentTs

Step 6: Static Routing

- Setting static routes
  - Use Ipv4StaticRoutingHelper
- We provide a function to manipulate table

```
Ipv4StaticRoutingHelper staticRouting;
InternetStackHelper internet;
internet.SetRoutingHelper(staticRouting);
internet.Install (nodes);
```

Step 6: Static Routing

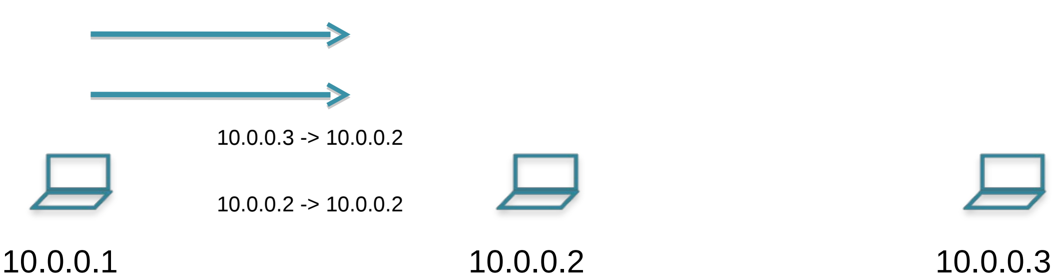
- Setting static routes
  - Use Ipv4StaticRoutingHelper
- We provide a function to manipulate table

```
void SetStaticRoute(Ptr<Node> n, const char* destination, const char* nextHop, uint32_t
interface) {
  Ipv4StaticRoutingHelper staticRouting;
  Ptr<Ipv4> ipv4 = n->GetObject<Ipv4> ();
  Ptr<Ipv4StaticRouting> a = staticRouting.GetStaticRouting (ipv4);
  a->AddHostRouteTo (Ipv4Address (destination), Ipv4Address (nextHop), interface);
}
```

Step 6: Configuring Static Routes

- Setting static routes:

```
// Set addresses

SetStaticRoute(nodes.Get(0), "10.0.0.3", "10.0.0.2", 1);
SetStaticRoute(nodes.Get(0), "10.0.0.2", "10.0.0.2 ,1);
SetStaticRoute(nodes.Get(1), "10.0.0.1", "10.0.0.1", 1);
SetStaticRoute(nodes.Get(1), "10.0.0.3", "10.0.0.3", 1);
SetStaticRoute(nodes.Get(2), "10.0.0.1", "10.0.0.2", 1);
SetStaticRoute(nodes.Get(2), "10.0.0.2", "10.0.0.2", 1);
```



Step 6: Configuring Static Routes

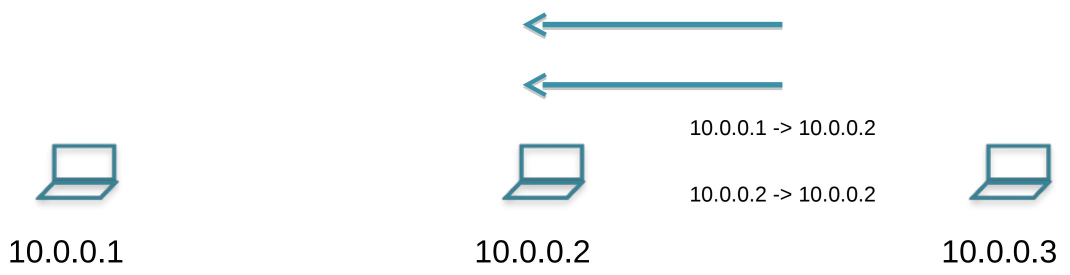
- Setting static routes:

```
// Set addresses

SetStaticRoute(nodes.Get(0), "10.0.0.3", "10.0.0.2", 1);
SetStaticRoute(nodes.Get(0), "10.0.0.2", "10.0.0.2", 1);
SetStaticRoute(nodes.Get(1), "10.0.0.1", "10.0.0.1", 1);
SetStaticRoute(nodes.Get(1), "10.0.0.3", "10.0.0.3", 1);
SetStaticRoute(nodes.Get(2), "10.0.0.1", "10.0.0.2", 1);
SetStaticRoute(nodes.Get(2), "10.0.0.2", "10.0.0.2", 1);
```

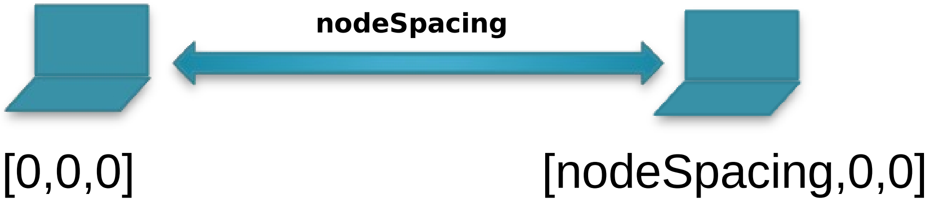
Step 6: Configuring Static Routes

- Setting static routes:

```
// Set addresses

SetStaticRoute(nodes.Get(0), "10.0.0.3", "10.0.0.2", 1);
SetStaticRoute(nodes.Get(0), "10.0.0.2", "10.0.0.2", 1);
SetStaticRoute(nodes.Get(1), "10.0.0.1", "10.0.0.1", 1);
SetStaticRoute(nodes.Get(1), "10.0.0.3", "10.0.0.3", 1);
SetStaticRoute(nodes.Get(2), "10.0.0.1", "10.0.0.2", 1);
SetStaticRoute(nodes.Get(2), "10.0.0.2", "10.0.0.2", 1);
```



Step 8: Explicit Locations and Constant Positions




```
MobilityHelper mobility;

Ptr<ListPositionAllocator> positionAlloc = CreateObject<ListPositionAllocator>();
positionAlloc->Add(Vector(0.0, 0.0, 0.0));
positionAlloc->Add(Vector(0.0, nodeSpacing, 0.0));
mobility.SetPositionAllocator(positionAlloc);
```

```
MobilityHelper mobility;

// Set positions

mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");

mobility.Install(nodes);
```

The logging facility

- Ns-3 has an extensive logging facility
- Seven levels: error, warn, debug, info, function, logic, all

```
NS_LOG_COMPONENT_DEFINE ("MANET");

...

NS_LOG_INFO("Area width: " << (rows - 1) * nodeSpacing);
NS_LOG_INFO("Area height: " << (cols - 1) * nodeSpacing);
```

- Can activate component from script or from shell
  - LogComponentEnable (”MANET", LOG_LEVEL_INFO);
  - $ export NS_LOG=”MANET=level_info”

- About ns-3
- Installing ns-3
- Core concepts
  - Ns-3 Objects
  - Smart pointers
  - Object aggregation
  - Run-time type information
- Using Ns-3: step-by-step example
- Extending Ns-3
- Resources

- About ns-3
- Installing ns-3
- Core concepts
  - Ns-3 Objects
  - Smart pointers
  - Object aggregation
  - Run-time type information
- Using Ns-3: step-by-step example
- Extending Ns-3
- Resources

ns-3

NETWORK SIMULATOR

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.

www.nsnam.org

- Intended as the successor of Ns-2
  - Clean slate implementation: no re-use of Ns-2 code
  - Easier to use, more facilities, faster, more accurate, more flexible
- First version 3.1 June 2008
  - Current version 3.28
  - Available for Linux, OS X and Windows w/ Cygwin
- Currently 22,617 hits in ACM DL
  - In 2017: 2678
  - Opnet in 2017: 3
  - Omnet++ in 2017: 9
  - Ns-2 in 2017: 3495
- Written in C++
  - Simulation scripts in C++ (python optional)
  - Helper classes make “scripting” in C++ easy

ns-3
NETWORK SIMULATOR

- About ns-3
- Installing ns-3
- Core concepts
  - Ns-3 Objects
  - Smart pointers
  - Object aggregation
  - Run-time type information
- Using Ns-3: step-by-step example
- Extending Ns-3
- Resources

Installing Ns-3

- Simplest approach: download tar ball, extract and build

```
$ wget https://www.nsnam.org/release/ns-allinone-3.28.tar.bz2
$ tar jxfv ns-allinone-3.28.tar.bz2
$ cd ns-allinone-3.28/ns-3.28
$ ./waf configure --enable-examples
$ ./waf
```

  - Confirmed to work on Ubuntu 16.04.4 LTS

- For eclipse: see https://www.nsnam.org/wiki/HOWTO_configure_Eclipse_with_ns-3

- About ns-3
- Installing ns-3
- Core concepts
  - Ns-3 Objects
  - Smart pointers
  - Object aggregation
  - Run-time type information
- Using Ns-3: step-by-step example
- Extending Ns-3
- Resources

Ns-3 Objects

- Very important information to use ***and*** extend Ns-3
- Most objects inherit from ns3::Object
- Provides a range of useful properties for simulation
  - Smart pointers
  - Object aggregation
  - Run-time type information
  - Attributes
  - Trace sources

Smart pointers	SimpleRefCount	Object
Object aggregation	ObjectBase	Object
Run-time type information	ObjectBase	Object
Attributes	ObjectBase	Object
Trace sources	ObjectBase	Object

- Ns-3 objects are created with
  CreateObject<Class> (constructor arguments)

Smart Pointers

- Provides a form of “garbage-collection”
- Enables object aggregation
- CreateObject returns smart pointer:

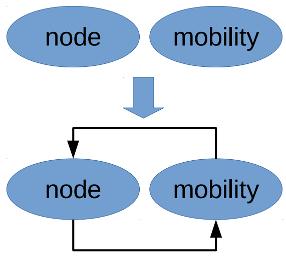
```
Ptr<PacketSocketFactory> factory =
    CreateObject<PacketSocketFactory> ();
```

- Always check return values and parameters: **`Ptr<T>`** or not?

Object Aggregation

- Objects can be dynamically aggregated to each other
- All objects in an aggregation can be access via any objects in said aggregation
  - Avoids huge classes that encompass all possible functionality

```
node->AggregateObject(mobility);

Ptr<MobilityModel> mob =
   node->GetObject<MobilityModel> ();
```



Run-Time Type Information

- All Ns-3 objects must implement **`TypeId GetTypeId(void)`**
- **`TypeId`** informs about attributes, runtime type information and trace sources

```
TypeId
RoutingProtocol::GetTypeId (void)
{
  static TypeId tid = TypeId ("ns3::olsr::RoutingProtocol")
    .SetParent<Ipv4RoutingProtocol> ()
    .SetGroupName ("Olsr")
    .AddConstructor<RoutingProtocol> ()
    .AddAttribute ("HelloInterval", "HELLO messages emission interval.",
                   TimeValue (Seconds (2)),
                   MakeTimeAccessor (&RoutingProtocol::m_helloInterval),
                   MakeTimeChecker ())
         ...
    .AddTraceSource ("RoutingTableChanged", "The OLSR routing table has changed.",
                     MakeTraceSourceAccessor (&RoutingProtocol::m_routingTableChanged),
                     "ns3::olsr::RoutingProtocol::TableChangeTracedCallback")
  ;
  return tid;
}
```

Run-Time Type Information

- Objects, attributes and trace sources can be located via textual paths via functions in the **`Config`** namespace (**objects**):

```
TypeId
RoutingProtocol::GetTypeId (void)
{
  static TypeId tid = TypeId ("ns3::olsr::RoutingProtocol")
    .SetParent<Ipv4RoutingProtocol> ()
    .SetGroupName ("Olsr")
    .AddConstructor<RoutingProtocol> ()

        ...
```

```
Config::MatchContainer m =
           Config::LookupMatches("NodeList/*/$ns3::olsr::RoutingProtocol");

Ptr<Olsr::RoutingProtocol> olsr =
    m.Get(0)->GetObject<Olsr::RoutingProtocol> ();
```

Equivalent to:

```
nodes.Get(0)->GetObject<Olsr::RoutingProtocol> ();
```

Run-Time Type Information

- Objects, attributes an
  via textual paths via f
  namespace (**objects**

```
TypeId
RoutingProtocol::GetTypeId (void)
{
  static TypeId tid = TypeId ("ns3::
    .SetParent<Ipv4RoutingProtocol>
    .SetGroupName ("Olsr")
    .AddConstructor<RoutingProtocol>

         ...
```

Example paths:

/NodeList/[3-5]|8|[0-1]
matches nodes index 0, 1, 3, 4, 5, 8

/NodeList/*
matches all nodes

/NodeList/3/$ns3::Ipv4
matches object of type ns3::Ipv4 aggregated to node number 3

/NodeList/3/DeviceList/*/$ns3::CsmaNetDevice
matches all devices of type ns3::CsmaNetDevice in node number 3

(See Doxygen for paths to particular objects)

```
Config::MatchContainer m =
           Config::LookupMatches("NodeList/*/$ns3::olsr::RoutingProtocol");

Ptr<Olsr::RoutingProtocol> olsr =
    m.Get(0)->GetObject<Olsr::RoutingProtocol> ();
```

Equivalent to:

```
nodes.Get(0)->GetObject<Olsr::RoutingProtocol> ();
```

Run-Time Type Information

- Objects, attributes and trace sources can be located via textual paths via functions in the `Config` namespace (**objects**):

```
TypeId
RoutingProtocol::GetTypeId (void)
{
  static TypeId tid = TypeId ("ns3::olsr::RoutingProtocol")
    .SetParent<Ipv4RoutingProtocol> ()
    .SetGroupName ("Olsr")
    .AddConstructor<RoutingProtocol> ()

        ...
```

Typecasting via object->GetObject<Class>

```
Config::MatchContainer m =
           Config::LookupMatches("NodeList/*/$ns3::olsr::RoutingProtocol");

Ptr<Olsr::RoutingProtocol> olsr =
    m.Get(0)->GetObject<Olsr::RoutingProtocol> ();
```

Equivalent to:

```
nodes.Get(0)->GetObject<Olsr::RoutingProtocol> ();
```

Run-Time Type Information

- Objects, attributes and trace sources can be located via textual paths via functions in the **`Config`** namespace (**attributes**):

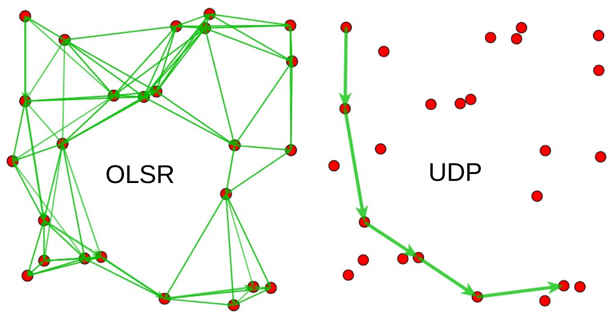
```
TypeId
RoutingProtocol::GetTypeId (void)
{
  static TypeId tid = TypeId ("ns3::olsr::RoutingProtocol")
    .SetParent<Ipv4RoutingProtocol> ()
    .SetGroupName ("Olsr")
    .AddConstructor<RoutingProtocol> ()
    .AddAttribute ("HelloInterval", "HELLO messages emission interval.",
                   TimeValue (Seconds (2)),
                   MakeTimeAccessor (&RoutingProtocol::m_helloInterval),
                   MakeTimeChecker ())
         ...
}
```

```
Config::SetDefault("ns3::olsr::RoutingProtocol::HelloInterval", TimeValue (Seconds(2)));
Config::Set("/NodeList/*/$ns3::olsr::RoutingProtocol/HelloInterval",
           TimeValue(Seconds(5)));
routingProtocolObject->SetAttribute("HelloInterval", TimeValue(Seconds(1)));
```

- About ns-3
- Installing ns-3
- Core concepts
  - Ns-3 Objects
  - Smart pointers
  - Object aggregation
  - Run-time type information
- Using Ns-3: step-by-step example
- Extending Ns-3
- Resources

Using Ns-3 – Via Example

- MANET with 25 nodes
  - Based on 802.11g wifi
- Routing protocol: OLSR
- Workload: uniform UDP traffic
  - 500-byte packets, 20 pps
  - No background traffic
- Mobility: random walk
  - Initial placement: 5 x ceil(5 / y) grid
  - y is the number of nodes
  - 100 meters between nodes
- Duration 10 minutes



9 Steps of an ns-3 Simulation Script

1) Handle command line arguments
2) Set default attribute values and random seed
3) Create nodes
4) Configure physical and MAC layers
5) Set up network stack, routing and addresses
6) Configure and install applications
7) Set up initial positions and mobility
8) Set up data collection
9) Schedule user-defined events and start simulation

Step 1: Command Line Arguments

- Enables parameterization of simulation from command line

```
int main (int argc, char *argv[])
{
  ...
  // Obtain command line arguments
  CommandLine cmd;
  cmd.AddValue ("cols", "Columns of nodes", cols);
  cmd.AddValue ("numnodes", "Number of nodes", numNodes);
  cmd.AddValue ("spacing", "Spacing between neighbouring nodes", nodeSpacing);
  cmd.AddValue ("duration", "Duration of simulation", duration);
  cmd.AddValue ("seed", "Random seed for simulation", seed);
  cmd.AddValue ("run", "Simulation run", run);
  cmd.AddValue ("packetrate", "Packets transmitted per second", packetRate);
  cmd.AddValue ("packetsize", "Packet size", packetSize);
  cmd.AddValue ("sourcenode", "Number of source node", sourceNode);
  cmd.AddValue ("destinationnode", "Number of destination node", destinationNode);
  cmd.AddValue ("showtime", "show ... time ... (default = true)", showSimTime);
  cmd.Parse (argc,argv);
  ...
```

For instance:

```
./waf -run “manet --nodespacing=50 --pktsize=100 --packetrate=500”
```

Step 2: Set Attribute Values and Random Seed

- Use `Config::-functions` to set default parameter values
- Remember to change run number between runs!

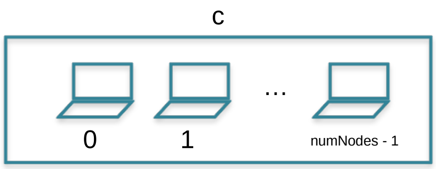
```
// Set default parameter values
Config::SetDefault("ns3::WifiRemoteStationManager::FragmentationThreshold",
  StringValue ("2200"));
Config::SetDefault("ns3::WifiRemoteStationManager::RtsCtsThreshold",
  StringValue ("2200"));

// Set random seed and run number
SeedManager::SetSeed (seed);
SeedManager::SetRun (run);
```

```
$ for run in "1 2 3"; do ./waf -run “manet --run=$run”; done
```

Step 3: Create Nodes

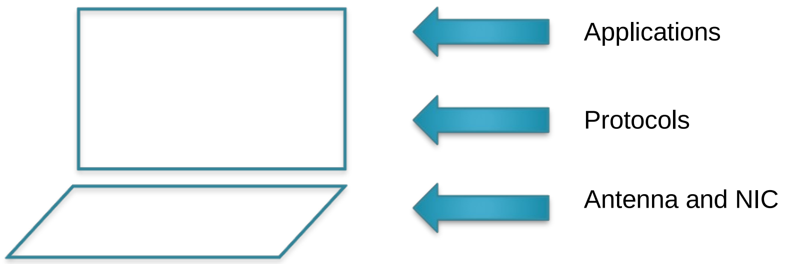
- Most objects in Ns-3 managed by ***containers***
  - Simulations consist of many objects of the same type
  - Later used by helper classes to install components
  - Entities in containers obtained with `container->get()`

```
// Create notes
NodeContainer c;
c.Create (numNodes);
...
apps = client.Install (c.Get (sourceNode));
```



Step 4-7: Configure Nodes

- Nodes are initially empty hulls



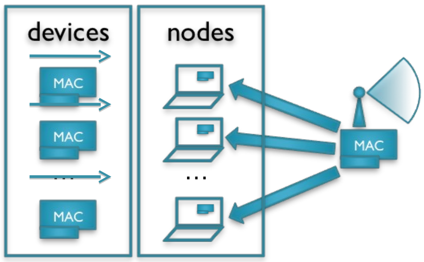
Step 4: Physical Layer

- ***Helpers*** enable script-like C++ programs
- Here:
  - 802.11g in ad-hoc mode
  - Automatic Rate Fallback (ARF)
    - Kamerman, Ad, and Leo Monteban. "WaveLAN®-II: a high-performance wireless LAN for the unlicensed band." Bell Labs technical journal 2.3 (1997): 118-133.
- Elsewise: default values
- Note that `wifi.install` uses node container `c`

```
// Set up physical and mac layers
WifiHelper wifi = WifiHelper::Default ();
wifi.SetStandard (WIFI_PHY_STANDARD_80211g);
wifi.SetRemoteStationManager ("ns3::ArfWifiManager");
NqosWifiMacHelper wifiMac = NqosWifiMacHelper::Default ();
wifiMac.SetType ("ns3::AdhocWifiMac");
YansWifiPhyHelper phy = YansWifiPhyHelper::Default ();
YansWifiChannelHelper wifiChannel = YansWifiChannelHelper::Default ();
phy.SetChannel (wifiChannel.Create ());
NetDeviceContainer devices = wifi.Install (phy, wifiMac, c);
```

Step 4: Physical Layer

- New container: `devices`

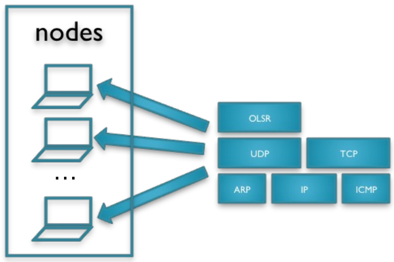



```
// Set up physical and mac layers
WifiHelper wifi = WifiHelper::Default ();
wifi.SetStandard (WIFI_PHY_STANDARD_80211g);
wifi.SetRemoteStationManager ("ns3::ArfWifiManager");
NqosWifiMacHelper wifiMac = NqosWifiMacHelper::Default ();
wifiMac.SetType ("ns3::AdhocWifiMac");
YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();
YansWifiChannelHelper wifiChannel = YansWifiChannelHelper::Default ();
YansWifiPhyHelper phy = wifiPhy;
phy.SetChannel (wifiChannel.Create ());
NetDeviceContainer devices = wifi.Install (phy, wifiMac, c);
```

Step 5: Install Internet Stack and Routing Protocol

- Select routing protocol
  - Ns-3 currently supports many routing protocols (e.g., OLSR, AODV, DSDV, ...)
  - Used in example: OLSR
- Internet stack: IP, TCP, UDP, ARP and ICMP

```
// Routing and Internet stack
ns3::OlsrHelper olsr;
InternetStackHelper internet;
internet.SetRoutingHelper(olsr);
internet.Install (c);
```



Step 5: Assign Addresses

- `c->Get(X)` gets
  IP address 10.0.0.(X+1)

  and
  MAC address 00:00:00:00:00:(X+1)

```
// Assign addresses
Ipv4AddressHelper address;
address.SetBase ("10.0.0.0", "255.255.255.0");
Ipv4InterfaceContainer interfaces = address.Assign (devices);
```

Step 6: Install Applications

- In example: simple UDP server and client
- Set attributes
- Specify when applications start and stop

```
// Server/Receiver
UdpServerHelper server (4000);
ApplicationContainer apps = server.Install (c.Get(destinationNode));
apps.Start (Seconds (1));
apps.Stop (Seconds (duration - 1));

// Client/Sender
UdpClientHelper client (interfaces.GetAddress (destinationNode), 4000);
client.SetAttribute ("MaxPackets", UintegerValue (100000000));
client.SetAttribute ("Interval", TimeValue (Seconds(1 / ((double) packetRate))));
client.SetAttribute ("PacketSize", UintegerValue (packetSize));
apps = client.Install (c.Get (sourceNode));
apps.Start (Seconds (1));
apps.Stop (Seconds (duration - 1));
```

The Ns-3 Node

- Node provides pointers to devices and applications

```
Ptr<Application> app = node->GetApplication(0);
Ptr<NetDevice> nic = node->GetDevice(0);
```

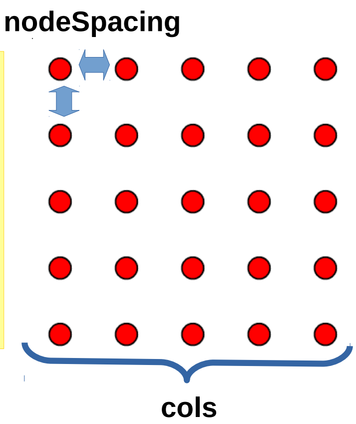
- Aggregated with stack, mobility model and energy model

```
Ptr<Ipv4> ip = nodes.Get(0)->GetObject<Ipv4>();
Ipv4Address addr = ip->GetAddress(1,0).GetLocal();
```

Step 7: Set up Initial Positions

- Several options available, including grid, disc, random placement and user-defined locations
  - Explained here: grid

```
// Set up mobility
MobilityHelper mobility;
mobility.SetPositionAllocator (
  "ns3::GridPositionAllocator",
  "MinX", DoubleValue (1.0),
  "MinY", DoubleValue (1.0),
  "DeltaX", DoubleValue (nodeSpacing),
  "DeltaY", DoubleValue (nodeSpacing),
  "GridWidth", UintegerValue (cols));
```

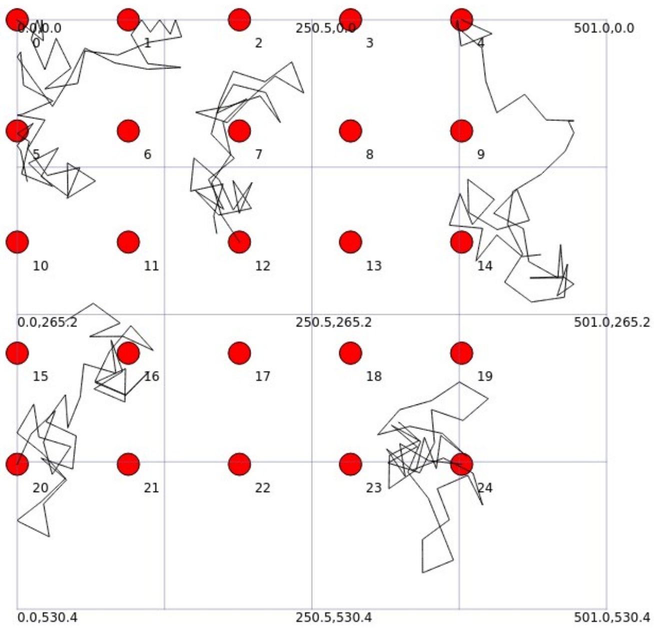


Step 7: Set Up Mobility Model

- Several alternatives
  - Random waypoint, random walk, user defined, …
- Used in Example: Random walk
  - Walk in random direction with random speed across fixed distance
    - Reflect upon hitting scenario boundaries
  - Speed defined with ***random variable***
  - Select new random direction

```
mobility.SetMobilityModel (
  "ns3::RandomWalk2dMobilityModel",
  "Bounds", RectangleValue
    (Rectangle (0,(cols * nodeSpacing) + 1,
       0,(rows * nodeSpacing) + 1)),
  "Speed",
    StringValue("ns3::UniformRandomVariable[Min=5.0,Max=10.0]"),
  "Distance", DoubleValue(30));

mobility.Install (c);
```

Example Mobility, 10 minutes



Step 9: Schedule Initial Events and Start Simulation

- Can schedule our own events before simulation
  - Example: print virtual time once every simulated second
- Simulation duration should be set

```
void PrintSeconds(void) {
    std::cerr << Simulator::Now() << std::endl;
    Simulator::Schedule(Seconds(1), &PrintSeconds);
}
```

```
    // Print simulated time
    if(showSimTime)
        Simulator::Schedule(Seconds(1), &PrintSeconds);

    Simulator::Stop(Seconds(duration));
    Simulator::Run ();
    Simulator::Destroy ();

    return 0;
}
```

Step 8: Data Collection

- Collect results = important step!
- Several alternatives
  1) std::cout << "Manual collection" << std::endl;
  2) Packet tracing
  3) Tracing subsystem
     - Low-Level: Trace sources and sinks
     - Medium/High-level: Data Collection Framework (DCF)
     - Highest-level: Statistics framework
  4) Logging via the logging facility *(see doc.)*
     - Intended for testing, debugging and verification

Step 8: Data Collection

- Collect results = important step!
- Several alternatives

1) std::cout << “Manual collection” << std::endl;
2) Packet tracing
3) Tracing subsystem
   - Low-Level: Trace sources and sinks
   - Medium/High-level: Data Collection Framework (DCF)
   - Highest-level: Statistics framework
4) Logging via the logging facility *(see doc.)*
   - Intended for testing, debugging and verification

Covered here

Packet Tracing

- Highly detailed packet models = enables real-world packet formats
- Popular packet capture format: PCAP
- One .pcap-file per node
- Pass device container from Step 4
- Set prefix (here "`MANET`")

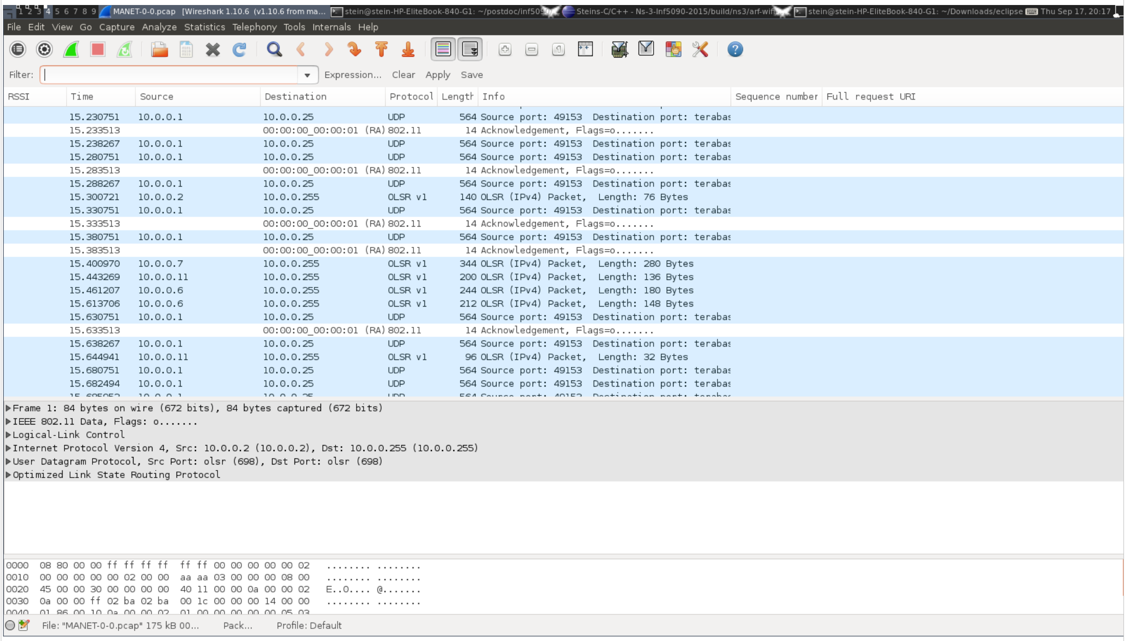
```
if(enablePcap)
   wifiPhy.EnablePcap ("MANET", devices);
```

Packet Tracing

- Resulting files: <prefix>-<node>-<device>.pcap

```
AUTHORS           MANET-17-0.pcap   routingtable-wireless.xml
bindings          MANET-18-0.pcap   scratch
build             MANET-19-0.pcap   src
CHANGES.html      MANET-20-0.pcap   test.py
doc               MANET-2-0.pcap    testpy.supp
dumbbell.xml      MANET-21-0.pcap   utils
examples          MANET-22-0.pcap   utils.py
LICENSE           MANET-23-0.pcap   utils.pyc
Makefile          MANET-24-0.pcap   VERSION
MANET-0-0.pcap    MANET-3-0.pcap    waf
MANET-10-0.pcap   MANET-4-0.pcap    waf.bat
MANET-1-0.pcap    MANET-5-0.pcap    waf-tools
MANET-11-0.pcap   MANET-6-0.pcap    wireless-animation.xml
MANET-12-0.pcap   MANET-7-0.pcap    wscript
MANET-13-0.pcap   MANET-8-0.pcap    wutils.py
MANET-14-0.pcap   MANET-9-0.pcap    wutils.pyc
MANET-15-0.pcap   README
MANET-16-0.pcap   RELEASE_NOTES
```

Can be opened in, e.g., Wireshark



Tracing Subsystem

- Based Ns-3 callbacks and attributes
- De-couples trace sources and sinks

Example trace source (from Ns-3 manual)

```
class MyObject : public Object
{
public:
  static TypeId GetTypeId (void)
  {
    static TypeId tid = TypeId ("MyObject")
      .SetParent (Object::GetTypeId ())
      .AddConstructor<MyObject> ()
      .AddTraceSource ("MyInteger",
                       "An integer value to trace.",
                       MakeTraceSourceAccessor (&MyObject::m_myInt))
      ;
    return tid;
  }

  MyObject () {}
  TracedValue<uint32_t> m_myInt;
};
```

Beginning user can easily control which objects are participating in tracing;

Intermediate users can extend the tracing system to modify the output format generated or use existing trace sources in different ways, without modifying the core of the simulator;

Advanced users can modify the simulator core to add new tracing sources and sinks.

Tracing Subsystem

- Based Ns-3 callbacks and attributes
- De-couples trace sources and sinks

Example trace sink (from Ns-3 manual)

```
void
IntTrace (Int oldValue, Int newValue)
{
  std::cout << "Traced " << oldValue << " to " << newValue << std::endl;
}

int
main (int argc, char *argv[])
{
  Ptr<MyObject> myObject = CreateObject<MyObject> ();

  myObject->TraceConnectWithoutContext ("MyInteger", MakeCallback(&IntTrace));

  myObject->m_myInt = 1234;
}
```

Tracing Subsystem

- Based Ns-3 callbacks and attributes
- De-couples trace sources and sinks

Example trace sink (from Ns-3 manual)

```
void
IntTrace (Int oldValue, Int newValue)
{
  std::cout << "Traced " << oldValue << " to " << newValue << std::endl;
}

int

  Ptr<MyObject> myObject = CreateObject<MyObject> ();

  myObject->TraceConnectWithoutContext ("MyInteger", MakeCallback(&IntTrace));

  myObject->m_myInt = 1234;
}
```

```
Traced 0 to 1234
```

Tracing Subsystem

- Objects, attributes and trace sources can be located via textual paths using functions in the `Config` namespace (**trace sources**):

```
TypeId
RoutingProtocol::GetTypeId (void)
{
        ...
    .AddTraceSource ("RoutingTableChanged", "The OLSR routing table has changed.",
                    MakeTraceSourceAccessor (&RoutingProtocol::m_routingTableChanged),
                    "ns3::olsr::RoutingProtocol::TableChangeTracedCallback")
        ...
```

```
void RouteChange(std::string source, uint32_t size) {
    std::cout << "Routechange at " << source << ", new size: " << size << std::endl;
}
```

```
Config::Connect("/NodeList/*/$ns3::olsr::RoutingProtocol/RoutingTableChanged",
    MakeCallback (&RouteChange));

routingProtocolObject->TraceConnect("RoutingTableChanged", "Context info",
    MakeCallback(&RouteChange));
```

Tracing Subsystem

- Objects, attributes and trace sources can be located via textual paths via functions in the `Config` namespace (**trace sources**):

```
TypeId
```

```
Routechange at /NodeList/2/$ns3::olsr::RoutingProtocol/RoutingTableChanged, new size: 8
Routechange at /NodeList/11/$ns3::olsr::RoutingProtocol/RoutingTableChanged, new size: 13
Routechange at /NodeList/15/$ns3::olsr::RoutingProtocol/RoutingTableChanged, new size: 9
...
```

```
                     "ns3::olsr::RoutingProtocol::TableChangeTracedCallback")
        ...
```

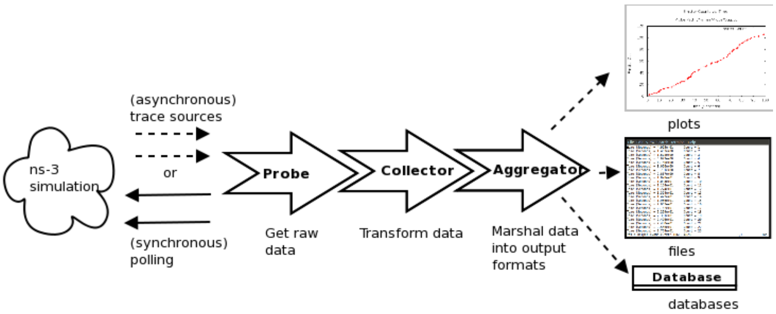
```
void RouteChan   std::string source, uint32_t
    std::cout     "Routechange at " << source
}
```

```
Routechange at Source node, new size: 5
...
```

```
Config::Connect("/NodeList/*/$ns3::olsr::RoutingProtocol/Routing   leChanged",
    MakeCallback (&RouteChange));

routingProtocolObject->TraceConnect("RoutingTableChanged", "Source node",
    MakeCallback(&RouteChange));
```

Data Collection Framework (DCF)

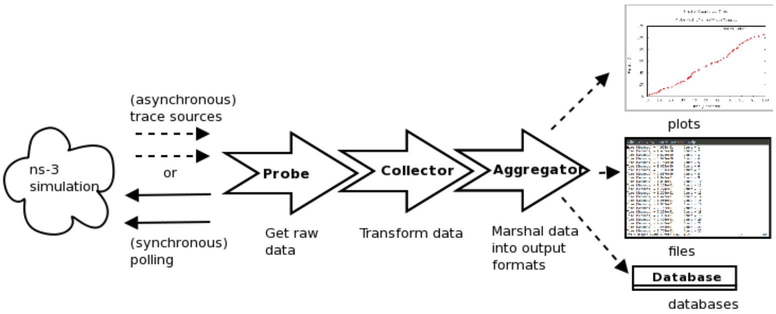
- Based on tracing subsystem
- On-line data reduction and processing
- Output format marshaling



https://www.nsnam.org/docs/release/3.28/manual/singlehtml/index.html#document-data-collection

Data Collection Framework (DCF)

- Two helpers currently implemented:
  - FileHelper
  - GnuplotHelper
- Additional supported: SQLList and OMNet++



https://www.nsnam.org/docs/release/3.28/manual/singlehtml/index.html#document-data-collection

DCF Example: FileHelper

ns-allinone-3.28/ns-3.28/examples/tutorial/seventh.cc

```
FileHelper fileHelper;

fileHelper.ConfigureFile ("seventh-packet-byte-count",
                          FileAggregator::FORMATTED);

fileHelper.Set2dFormat ("Time (Seconds) = %.3e\tPacket Byte Count = %.0f");

fileHelper.WriteProbe (“ns3::Ipv4PacketProbe”,
                       “/NodeList/*/$ns3::Ipv4L3Protocol/Tx”,
                       "OutputBytes");
```

- Output:

```
seventh-packet-byte-count-0.txt
seventh-packet-byte-count-1.txt
```

```
Time (Seconds) = 1.000e+00    Packet Byte Count = 40
Time (Seconds) = 1.004e+00    Packet Byte Count = 40
Time (Seconds) = 1.004e+00    Packet Byte Count = 576
Time (Seconds) = 1.009e+00    Packet Byte Count = 576
Time (Seconds) = 1.009e+00    Packet Byte Count = 576
Time (Seconds) = 1.015e+00    Packet Byte Count = 512
Time (Seconds) = 1.017e+00    Packet Byte Count = 576
Time (Seconds) = 1.017e+00    Packet Byte Count = 544
Time (Seconds) = 1.025e+00    Packet Byte Count = 576
Time (Seconds) = 1.025e+00    Packet Byte Count = 544

...
```

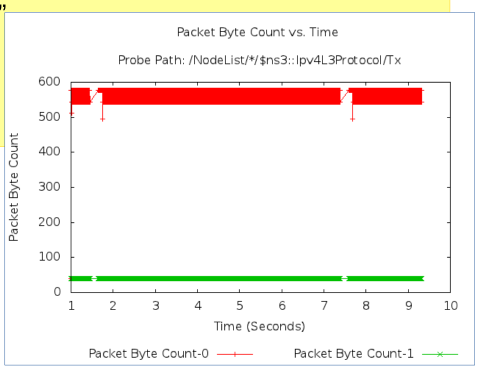
DCF Example: GnuplotHelper

```
GnuplotHelper plotHelper;

plotHelper.ConfigurePlot ("seventh-packet-byte-count",
                          "Packet Byte Count vs. Time",
                          "Time (Seconds)",
                          "Packet Byte Count");

plotHelper.PlotProbe ("ns3::Ipv4PacketProbe",
                      "/NodeList/*/$ns3::Ipv4L3Protocol/Tx",
                      "OutputBytes",
                      "Packet Byte Count",
                      GnuplotAggregator::KEY_BELOW);
```

ns-allinone-3.28/ns-3.28/examples/tutorial/seventh.cc

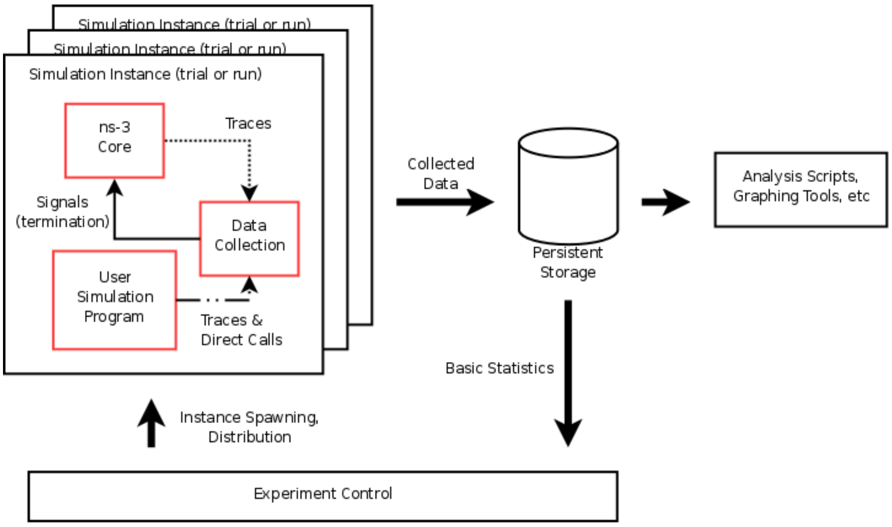
- Output:

```
seventh-packet-byte-count.dat (data file)
seventh-packet-byte-count.plt (gnuplot script)
seventh-packet-byte-count.sh (runs .plt)
```



Note that the trace source path specified may contain wildcards. In this case, multiple datasets are plotted on one plot; one for each matched path.

Statistics Framework



https://www.nsnam.org/docs/release/3.28/manual/singlehtml/index.html#document-data-collection

- About ns-3
- Installing ns-3
- Core concepts
  - Ns-3 Objects
  - Smart pointers
  - Object aggregation
  - Run-time type information
- Using Ns-3: step-by-step example
- Extending Ns-3
- Resources

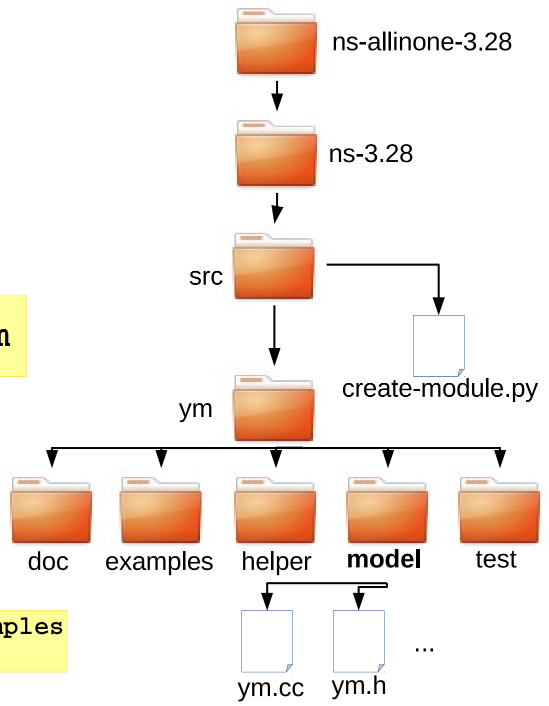
Extending Ns-3

- Prerequisite: C++ knowledge
- Module-based
- Create template with create-module.py

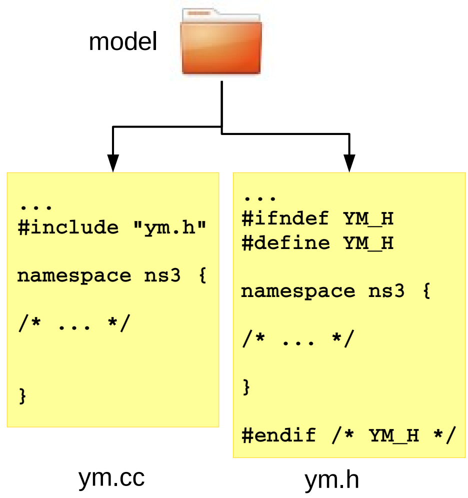
```
$ create-module.py ym
```

- Creates five folders
  - Your model in **“model”**
- **MUST reconfigure before re-compilation**

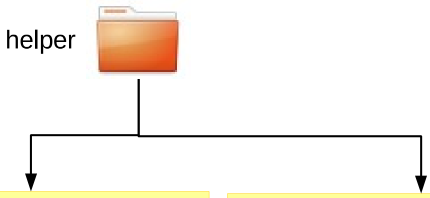
```
$ ./waf configure --enable-examples
$ ./waf
```



Resulting .cc and .h files in



Resulting .cc and .h files in




```
...
#include "ym-helper.h"

namespace ns3 {

/* ... */


}
```

ym-helper.cc

```
...
#ifndef INF5090_HELPER_H
#define INF5090_HELPER_H

#include "ns3/ym.h"

namespace ns3 {

/* ... */

}

#endif /* INF5090_HELPER_H */
```

ym-helper.h

Resulting .cc and .h files in

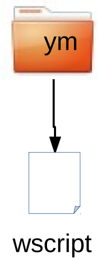

examples

example.cc:

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */

#include "ns3/core-module.h"
#include "ns3/ym-helper.h"

using namespace ns3;


int
main (int argc, char *argv[])
{
  bool verbose = true;

  CommandLine cmd;
  cmd.AddValue ("verbose", "Tell application to log if true", verbose);

  cmd.Parse (argc,argv);

  /* ... */

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

When Adding Files, Update wscript

```
...
def build(bld):
    module = bld.create_ns3_module('ym', ['core'])
    module.source = [
        'model/ym.cc',
        'helper/ym-helper.cc',
        ]

    module_test = bld.create_ns3_module_test_library('ym')
    module_test.source = [
        'test/ym-test-suite.cc',
        ]

    headers = bld(features='ns3header')
    headers.module = 'ym'
    headers.source = [
        'model/ym.h',
        'helper/ym-helper.h',
        ]

    if bld.env.ENABLE_EXAMPLES:
        bld.recurse('examples')

    # bld.ns3_python_bindings()
```



- About ns-3
- Installing ns-3
- Core concepts
  - Ns-3 Objects
  - Smart pointers
  - Object aggregation
  - Run-time type information
- Using Ns-3: step-by-step example
- Extending Ns-3
- Resources

- www.nsnam.org
- www.nsnam.org/wiki
- www.nsnam.org/documentation
  - Ns-3 manual
  - Ns-3 tutorial https://www.nsnam.org/docs/release/3.28/tutorial/html/index.html
  - Doxygen
  - Slides
  - Videos
  - ...
- Examples in the source code

Appendix

- Summary of simulation concepts
- Static routes
- User defined locations
- Constant positions
- The Ns-3 logging facility

Discrete-Event Simulation Concepts

Concept	Network Simulation Example
System	The Internet, MANET, WSN, ...
Model	C++ classes, math formulas, ...
Model state	C++ objects, packets, node positions, ...
Entity	Link, queue, packet, protocol, ...
Attributes	Link capacity, queue size, packet type, ...
List	Packets in a queue, nodes in a subnet, ...
Event	Transmission/arrival of packet, packet drop, ...
Event notice	Ns-3: Scheduler::Event (obj. w/ func. pointer)
Event list	Ns-3: DefaultSimulatorImpl::m_events
Activity	Transmission delay, part of movement, ...
Delay	Queuing delay, end-to-end delay, ...
Clock	Ns-3: DefaultSimulatorImpl::m_currentTs

Step 6: Static Routing

- Setting static routes
  - Use Ipv4StaticRoutingHelper
- We provide a function to manipulate table

```
Ipv4StaticRoutingHelper staticRouting;
InternetStackHelper internet;
internet.SetRoutingHelper(staticRouting);
internet.Install (nodes);
```

Step 6: Static Routing

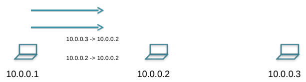
- Setting static routes
  - Use Ipv4StaticRoutingHelper
- We provide a function to manipulate table

```
void SetStaticRoute(Ptr<Node> n, const char* destination, const char* nextHop, uint32_t
interface) {
  Ipv4StaticRoutingHelper staticRouting;
  Ptr<Ipv4> ipv4 = n->GetObject<Ipv4> ();
  Ptr<Ipv4StaticRouting> a = staticRouting.GetStaticRouting (ipv4);
  a->AddHostRouteTo (Ipv4Address (destination), Ipv4Address (nextHop), interface);
}
```

Step 6: Configuring Static Routes

- Setting static routes:

```
// Set addresses

SetStaticRoute(nodes.Get(0), "10.0.0.3", "10.0.0.2", 1);
SetStaticRoute(nodes.Get(0), "10.0.0.2", "10.0.0.2 ,1);
SetStaticRoute(nodes.Get(1), "10.0.0.1", "10.0.0.1", 1);
SetStaticRoute(nodes.Get(1), "10.0.0.3", "10.0.0.3", 1);
SetStaticRoute(nodes.Get(2), "10.0.0.1", "10.0.0.2", 1);
SetStaticRoute(nodes.Get(2), "10.0.0.2", "10.0.0.2", 1);
```



Step 6: Configuring Static Routes

- Setting static routes:

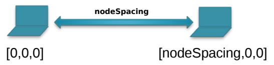
```
// Set addresses

SetStaticRoute(nodes.Get(0), "10.0.0.3", "10.0.0.2", 1);
SetStaticRoute(nodes.Get(0), "10.0.0.2", "10.0.0.2", 1);
SetStaticRoute(nodes.Get(1), "10.0.0.1", "10.0.0.1", 1);
SetStaticRoute(nodes.Get(1), "10.0.0.3", "10.0.0.3", 1);
SetStaticRoute(nodes.Get(2), "10.0.0.1", "10.0.0.2", 1);
SetStaticRoute(nodes.Get(2), "10.0.0.2", "10.0.0.2", 1);
```

Step 6: Configuring Static Routes

- Setting static routes:

```
// Set addresses

SetStaticRoute(nodes.Get(0), "10.0.0.3", "10.0.0.2", 1);
SetStaticRoute(nodes.Get(0), "10.0.0.2", "10.0.0.2", 1);
SetStaticRoute(nodes.Get(1), "10.0.0.1", "10.0.0.1", 1);
SetStaticRoute(nodes.Get(1), "10.0.0.3", "10.0.0.3", 1);
SetStaticRoute(nodes.Get(2), "10.0.0.1", "10.0.0.2", 1);
SetStaticRoute(nodes.Get(2), "10.0.0.2", "10.0.0.2", 1);
```

Step 8: Explicit Locations and Constant Positions




```
MobilityHelper mobility;

Ptr<ListPositionAllocator> positionAlloc = CreateObject<ListPositionAllocator>();
positionAlloc->Add(Vector(0.0, 0.0, 0.0));
positionAlloc->Add(Vector(0.0, nodeSpacing, 0.0));
mobility.SetPositionAllocator(positionAlloc);
```

```
MobilityHelper mobility;

// Set positions

mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");

mobility.Install(nodes);
```

The logging facility

- Ns-3 has an extensive logging facility
- Seven levels: error, warn, debug, info, function, logic, all

```
NS_LOG_COMPONENT_DEFINE ("MANET");

...

NS_LOG_INFO("Area width: " << (rows - 1) * nodeSpacing);
NS_LOG_INFO("Area height: " << (cols - 1) * nodeSpacing);
```

- Can activate component from script or from shell
  - LogComponentEnable (”MANET", LOG_LEVEL_INFO);
  - $ export NS_LOG=”MANET=level_info”