Presented by Luciano Jerez Chaves

Slides based on the followings presentations:

- ns-3 training (Tom Henderson, 2016)

- ns-3 introduction (Tom Henderson, 2014)

- Network simulation using ns-3 (Walid Younes, 2013)

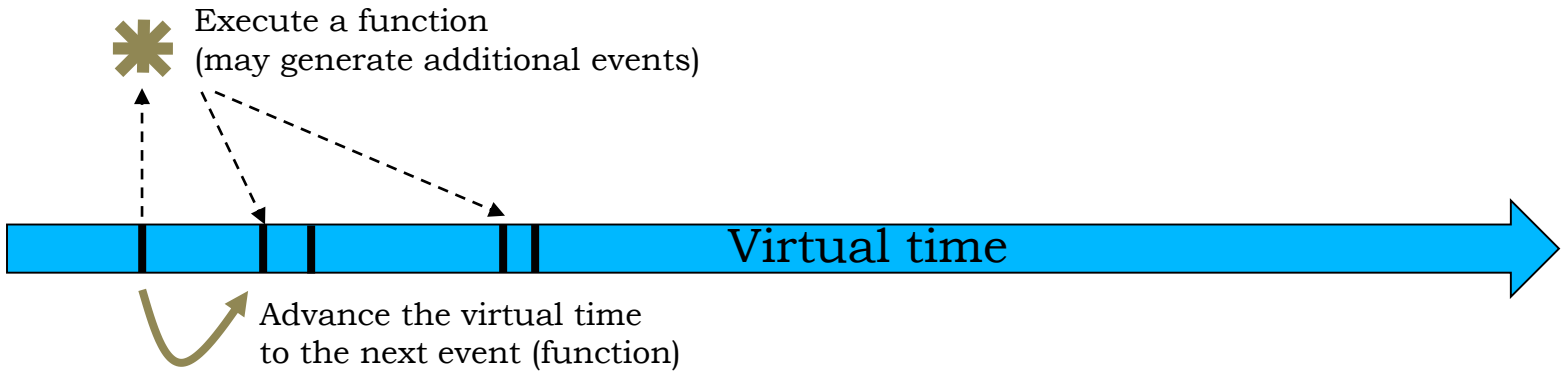
Introduction

History

- 1989: **REAL** *Srinivasan Keshav*
- 1995-1997: **ns-1** *Steve McCanne , Sally Floyd , Kevin Fall*
  - C++, Tcl-based scripting
- 1996-1997: **ns-2** *"Refactoring" by Steve McCanne*
  - Replace Tcl with Object Tcl (OTcl) of MIT
- 2006: **ns-3** *NSF team (Henderson, Riley, Floyd, Roy) and INRIA team (Dabbous, Lacage)*
  - C++ core with Python bindings
  - 26 releases and more than 175 open source contributors

Discrete-event simulation

- Simulation time moves in discrete jumps
- Functions schedule events and a scheduler orders the event execution
- Simulator executes a single-threaded event list until a specific time or when events end



Software overview

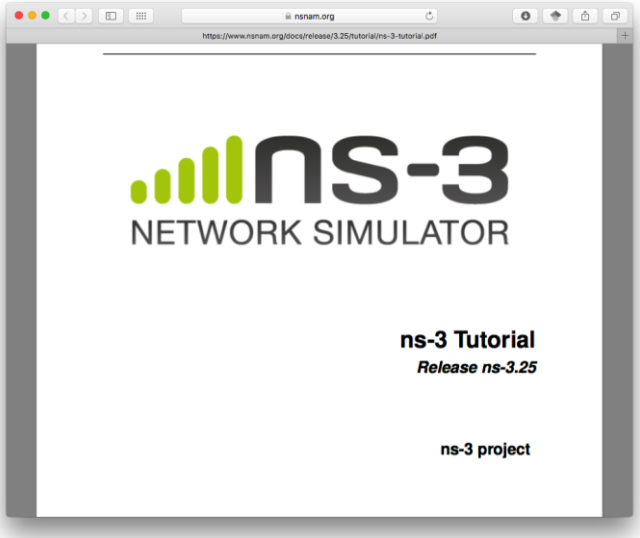
- ns-3 is a GNU GPLv2-licensed project
- ns-3 is written in C++, with bindings available for Python
- ns-3 is mainly supported for Linux, OS X, and FreeBSD
- ns-3 is a command-line, Unix-oriented software
- ns-3 is not backwards-compatible with ns-2

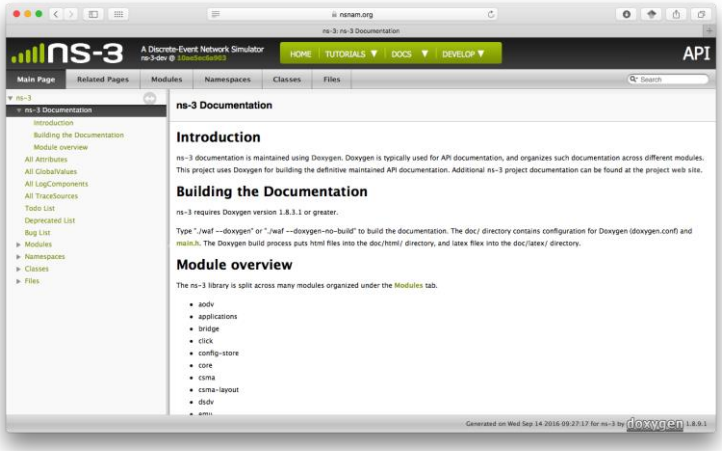
Web resources

- Web site:
  http://www.nsnam.org
- Mailing lists:
  https://groups.google.com/forum/#!forum/ns-3-users
  http://mailman.isi.edu/mailman/listinfo/ns-developers
- Wiki:
  http://www.nsnam.org/wiki/
- IRC:
  #ns-3 at http://freenode.net

Documentation

- Available at https://www.nsnam.org/documentation/
- **Written tutorial for beginners**
- Reference manual on the ns-3 core
- Model library documentation
- Source code API documentation





Software organization

- ns-3 is built as a system of independent software libraries (modules) that can be combined together
  - Remove dependences between modules
  - Source code heavily based on *callbacks*
- Each module is internally organized into:
  - Models
  - Examples
  - Tests
  - Documentation

Getting started

Software download

- Download the latest release
  - Use a compressed tarball file
    - `wget http://www.nsnam.org/releases/ns-allinone-3.35.tar.bz2`
    - `tar xjf ns-allinone-3.25.tar.bz2`
  - Clone the source code from the mercurial repository
    - `hg clone http://code.nsnam.org/ns-3.25`

The 'waf' build system

- Waf is a Python-based framework for configuring, compiling and installing applications
  - Replacement for other tools such as Autotools, CMake or Ant
  - For those familiar with Autotools:
    - `./configure` → `./waf configure`
    - `make` → `./waf build`
- Different parameters for configuration
  - `--build-profile=optimized|debug`
  - `--enable-examples` and `--enable-tests`
  - Check other options with `./waf --help`

Testing ns-3

- You can run the unit tests of the ns-3 distribution by running the following command:
  - `./test.py -c core`
- This command is typically run by users to quickly verify that an ns-3 distribution has built correctly
  - Don't worry about 'passed' tests

Running scripts

- We typically run scripts under the control of Waf
  - It sets key environment variables for running programs
  - `./waf --run hello-simulator`

Congratulations! You are now an ns-3 user!

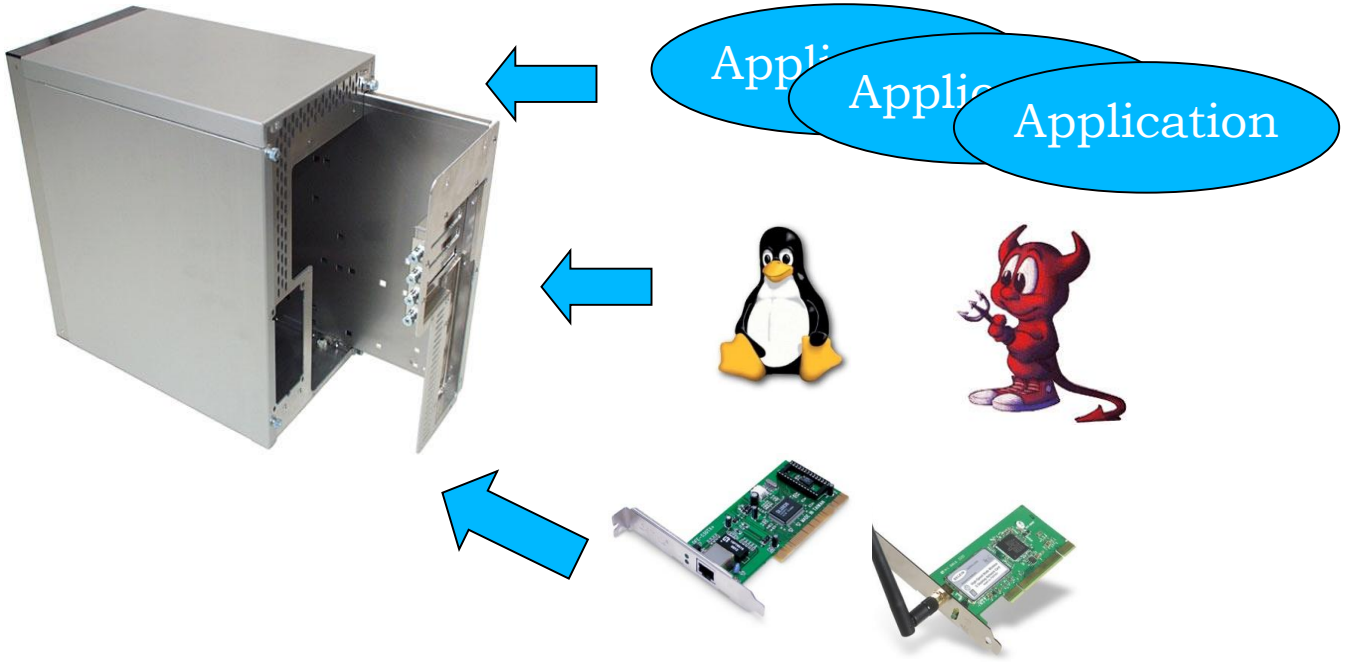
Conceptual overview

Basic components

- Nodes
- Applications
- Protocol stack
- Net devices
- Channels
- Helper API

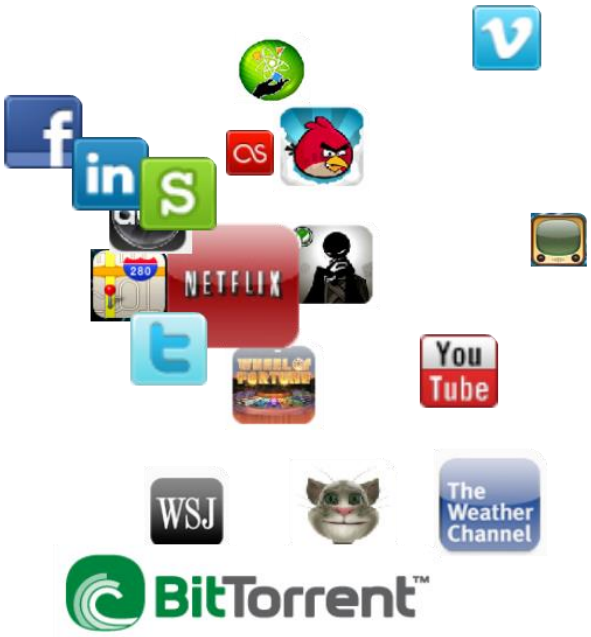
Node

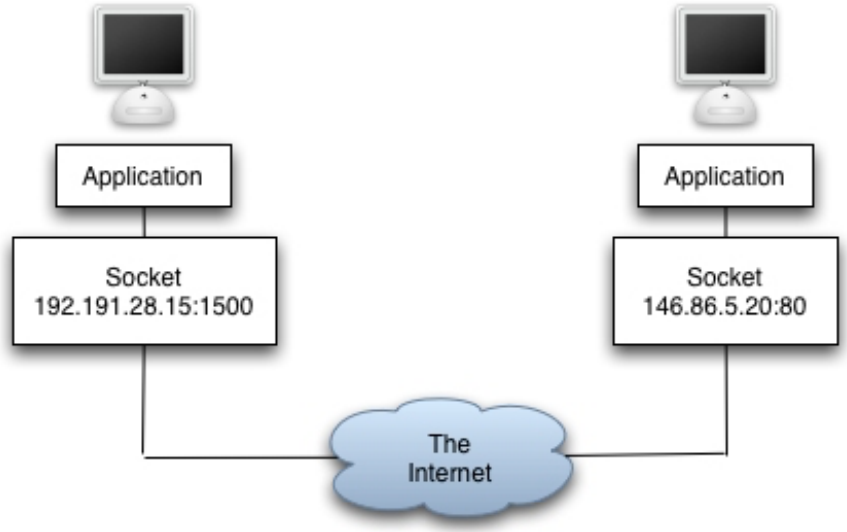
- A Node is a shell of a computer to which applications, protocol stacks, and net devices are added



Applications

- Abstraction for a user program that generates some network activity using an (asynchronous) socket API



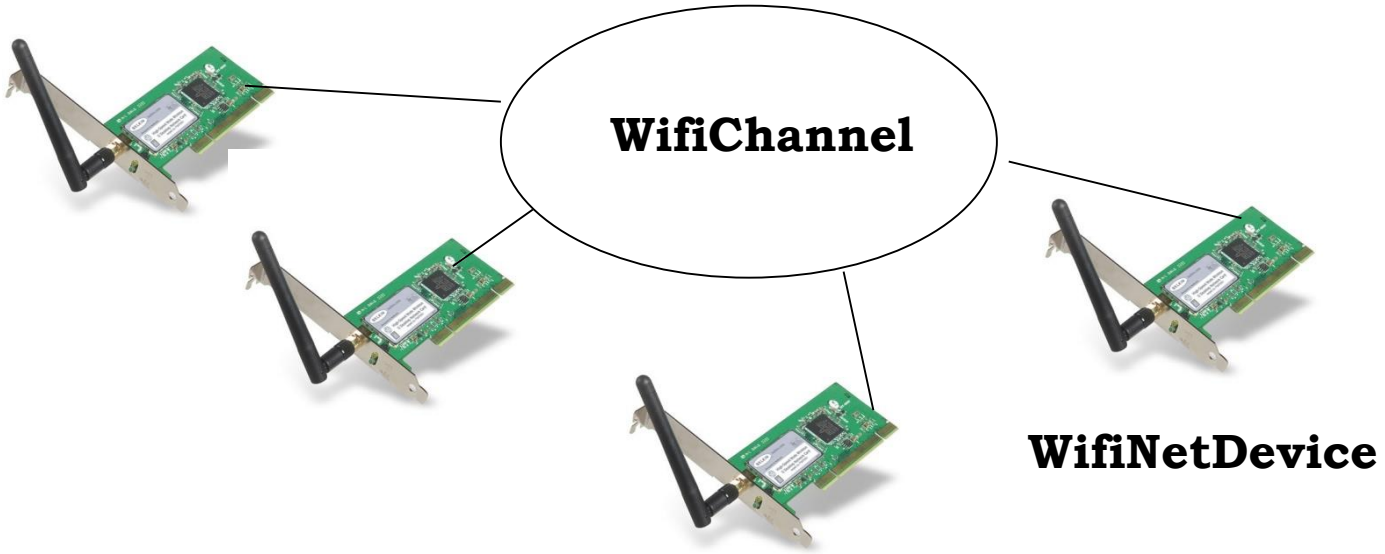


Protocol stack

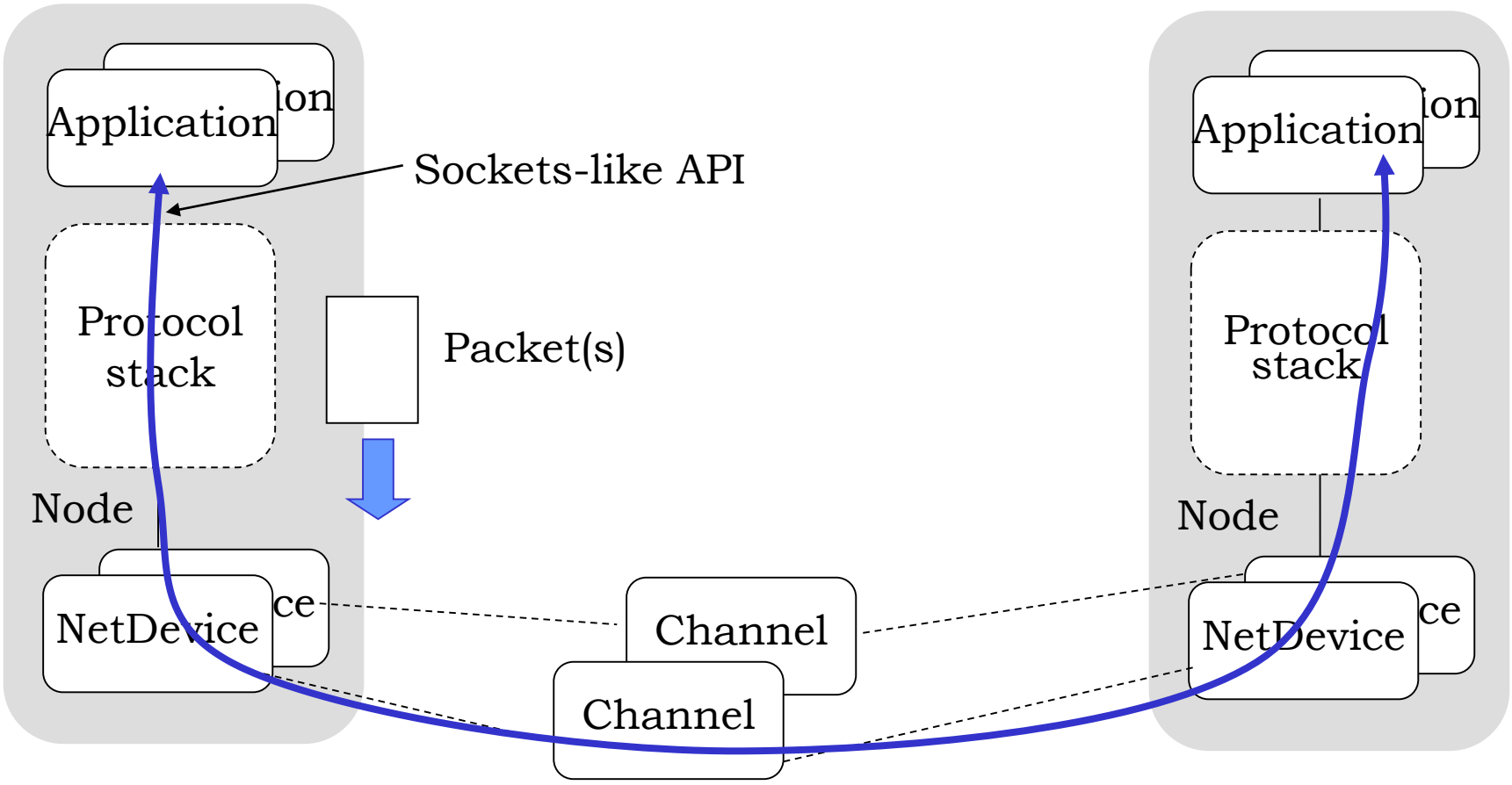
- Mainly TCP/IP stack
  - Layer 4 protocols (TCP, UDP, ICMP)
  - Layer 3 protocols (IPv4, IPv6, ARP)
- Also provides the socket API, routing capabilities and traffic control module

Net devices and channels

- A NetDevice is like a NIC, which is strongly bound to a channel of a matching type
  - Nodes are architected for multiple interfaces



The basic ns-3 architecture



Helper API

- The ns-3 “helper API” provides a set of classes and methods that make common operations easier
  - Container of objects
  - Helper classes

Helper API

- Containers group similar objects together, for convenience
  - They are often implemented using C++ std containers
- Helper classes provides simple “syntactical sugar” to make simulation scripts look nicer and easier to read
  - They are not generic
  - Each function applies a single operation on a "set of the same type objects”
  - A typical operation is `Install()`

Script example (first.cc)

first.cc (~/Documents/Unicamp/doutorado/codigos/ns-3.25/examples/tutorial) - VIM

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

#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");
                                                    1,1           Top
```

Script example (first.cc)

first.cc (~/Documents/Unicamp/doutorado/codigos/ns-3.25/examples/tutorial) - VIM

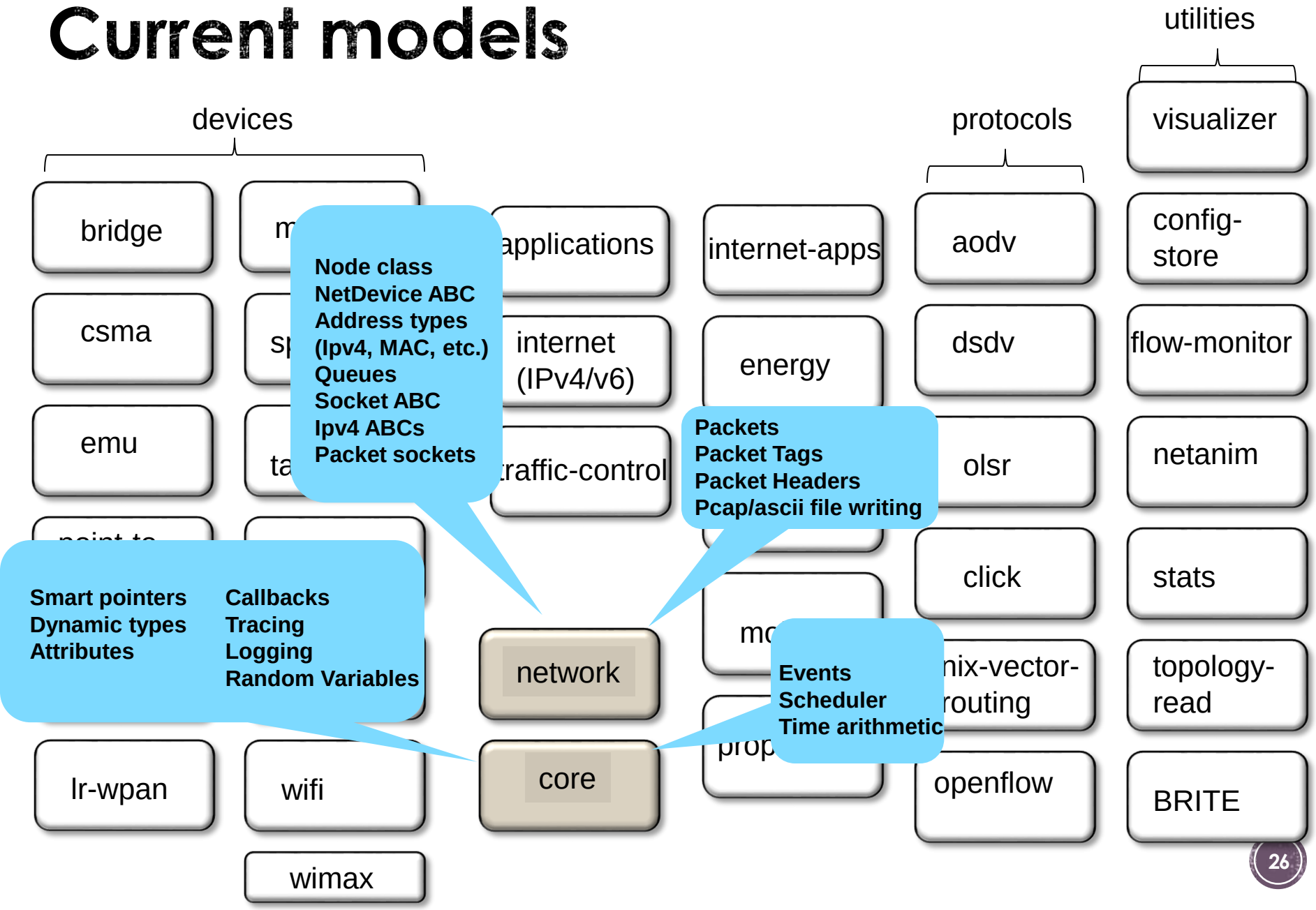
```
26
27 int
28 main (int argc, char *argv[])
29 {
30   Time::SetResolution (Time::NS);
31   LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
32   LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
33
34   NodeContainer nodes;
35   nodes.Create (2);
36
37   PointToPointHelper pointToPoint;
38   pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
39   pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
40
41   NetDeviceContainer devices;
42   devices = pointToPoint.Install (nodes);
43
44   InternetStackHelper stack;
45   stack.Install (nodes);
46
47   Ipv4AddressHelper address;
48   address.SetBase ("10.1.1.0", "255.255.255.0");
49
50   Ipv4InterfaceContainer interfaces = address.Assign (devices);
                                                    50,1          55%
```

Script example (first.cc)

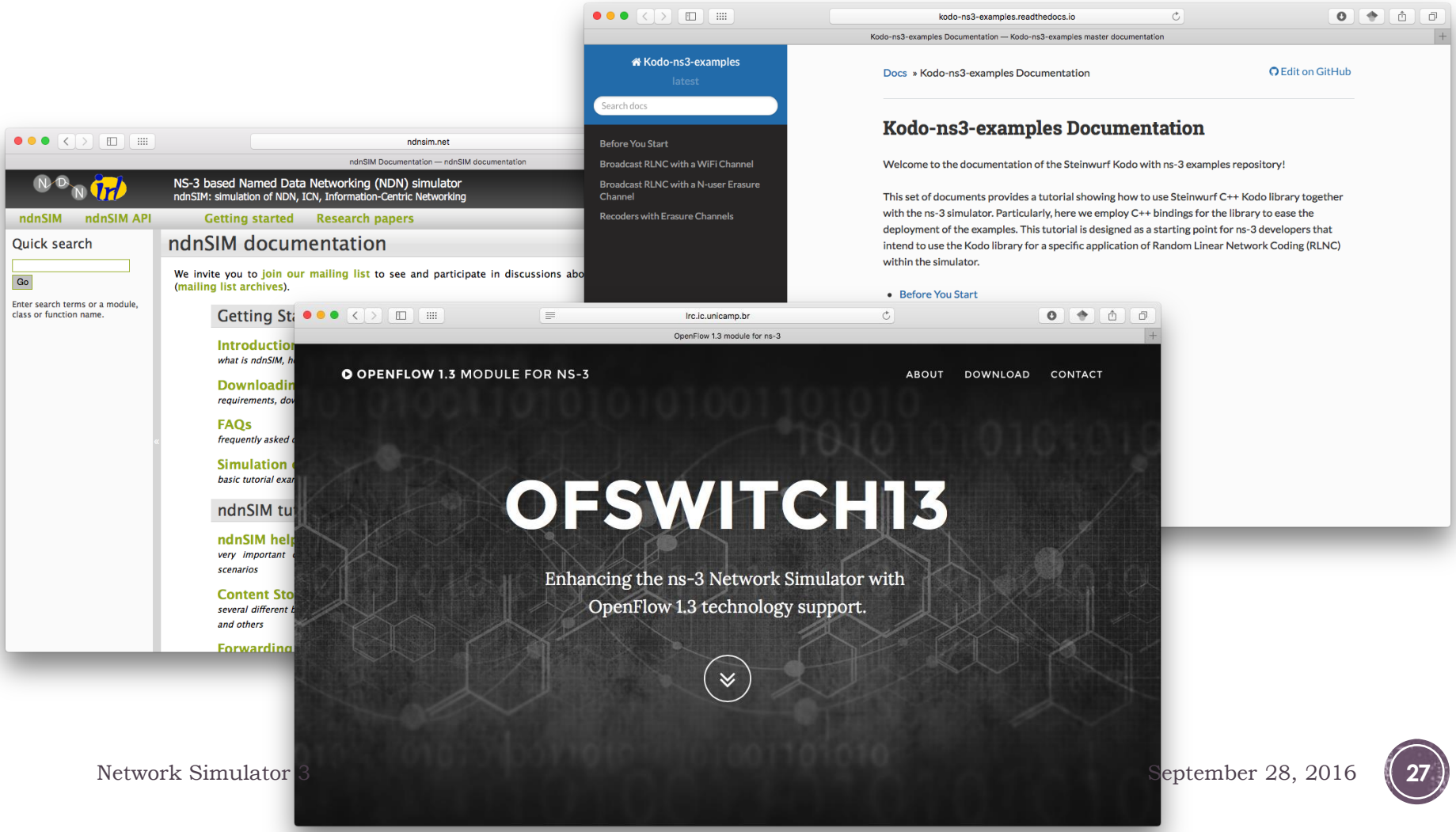
first.cc (~/Documents/Unicamp/doutorado/codigos/ns-3.25/examples/tutorial) - VIM

```
46
47   Ipv4AddressHelper address;
48   address.SetBase ("10.1.1.0", "255.255.255.0");
49
50   Ipv4InterfaceContainer interfaces = address.Assign (devices);
51
52   UdpEchoServerHelper echoServer (9);
53
54   ApplicationContainer serverApps = echoServer.Install (nodes.Get (1));
55   serverApps.Start (Seconds (1.0));
56   serverApps.Stop (Seconds (10.0));
57
58   UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);
59   echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
60   echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
61   echoClient.SetAttribute ("PacketSize", UintegerValue (1024));
62
63   ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));
64   clientApps.Start (Seconds (2.0));
65   clientApps.Stop (Seconds (10.0));
66
67   Simulator::Run ();
68   Simulator::Destroy ();
69   return 0;
70 }
                                                    70,1          Bot
```

Current models



Contributed code



More features…

Attribute system

- Each ns-3 object has a set of attributes that makes it easy to verify the parameters of a simulation
  - Name, help text, type, and initial value
- Dump and read them all in configuration files and visualize them in a GUI

Command line arguments

- Mechanism to parse command line arguments and automatically set local and global variables and attributes

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

- Passing `--PrintHelp` to programs will display command line options, if CommandLine is enabled
  - `./waf --run "sample-simulator --PrintHelp"`

```
--PrintHelp: Print this help message.
--PrintGroups: Print the list of groups.
--PrintTypeIds: Print all TypeIds.
--PrintGroup=[group]: Print all TypeIds of group.
--PrintAttributes=[typeid]: Print all attributes of typeid.
--PrintGlobals: Print the list of globals.
```

Packets

- Packet uses an advanced data structure
  - Supports real or virtual application data
  - Serializable (for emulation)
  - Supports pretty-printing
  - Efficient (copy-on-write semantics)

- Packet tag objects allow packets to carry around simulator-specific metadata

Smart pointers

- Smart pointers in ns-3 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
- No need to use `malloc` and `free` ☺

Mobility and position

- ALL nodes have to be created before simulation starts
- Position Allocators setup initial position of nodes
  - List, Grid, Random position…
- Mobility models specify how nodes will move
  - Constant position, constant velocity/acceleration, waypoint…
  - Trace-file based from mobility tools such as SUMO, etc.
  - Routes Mobility using Google API

Logging

- Assertions that aborts the program for false expressions
  - NS_ASSERT (expression);
- Debug Logging (not to be confused with tracing!)
  - Used to trace code execution logic, not to extract results!
    - NS_LOG_ERROR (…): serious error messages only
    - NS_LOG_WARN (…): warning messages
    - NS_LOG_DEBUG (…): rare ad-hoc debug messages
    - NS_LOG_INFO (…): informational messages (eg. banners)
    - NS_LOG_FUNCTION (…): function tracing
    - NS_LOG_PARAM (…): parameters to functions
    - NS_LOG_LOGIC (…): control flow tracing within functions

Debugging

- The gdb debugger can be used directly on binaries in the build directory
  - `./waf --command-template="gdb %s" --run <program-name>`
- The valgrind memcheck can be used directly on binaries in the build directory
  - `./waf --command-template="valgrind %s" --run <program-name>`

FlowMonitor

- Network monitoring framework
  - Detect all flows passing through the network
  - Stores metrics for analysis such as bitrates, duration, delays, packet sizes, packet loss ratios

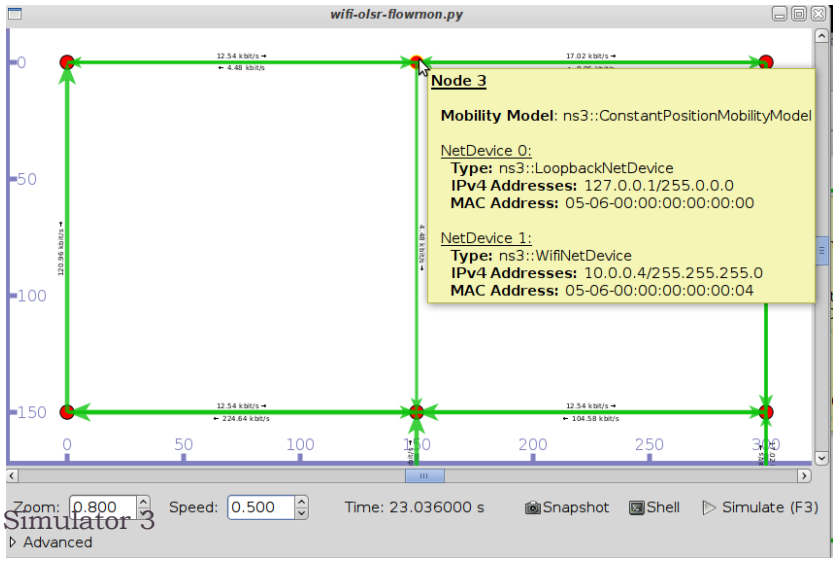
```
Hidden station experiment with RTS/CTS disabled:
Flow 1 (10.0.0.1 -> 10.0.0.2)
  Tx Bytes:   3847500
  Rx Bytes:   316464
  Throughput: 0.241443 Mbps
Flow 2 (10.0.0.3 -> 10.0.0.2)
  Tx Bytes:   3848412
  Rx Bytes:   336756
  Throughput: 0.256924 Mbps
------------------------------------------------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

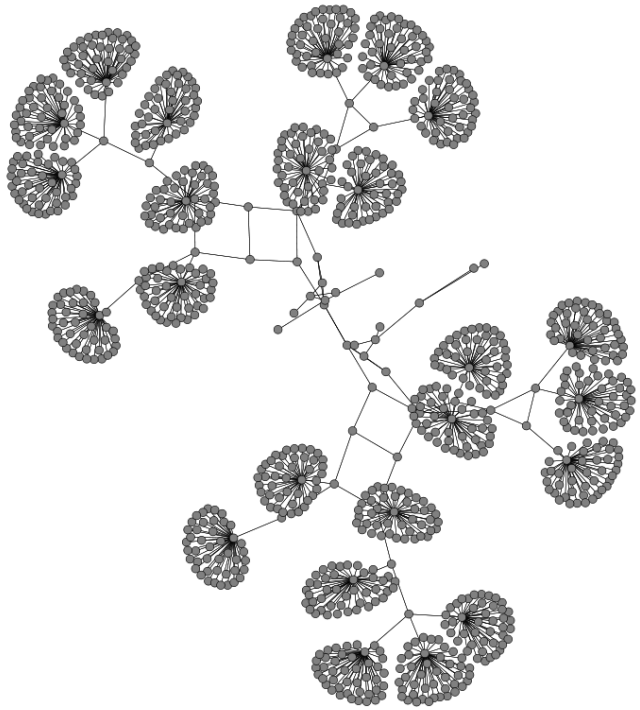
Visualization

- No preferred visualizer for ns-3
- Several tools have been developed over the years, with some scope limitations
  - Pyviz
  - NetAnim (George Riley and John Abraham)

PyViz

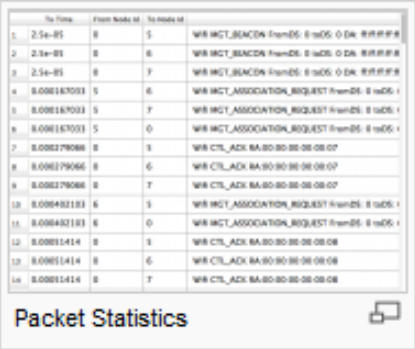
- Live simulation visualizer (no trace files)
- Useful for debugging
  - mobility model behavior
  - packets being dropped

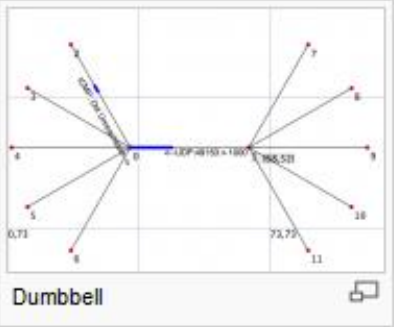


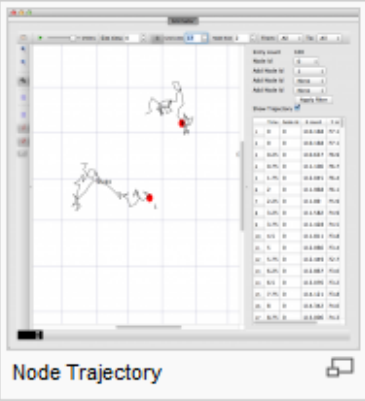


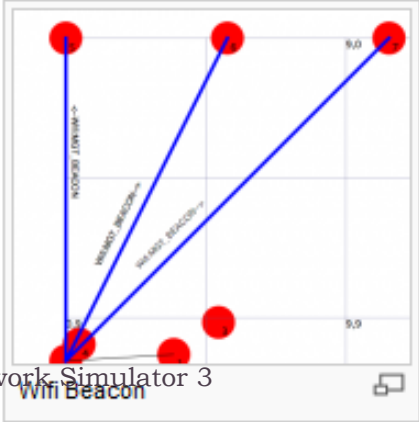
NetAnim

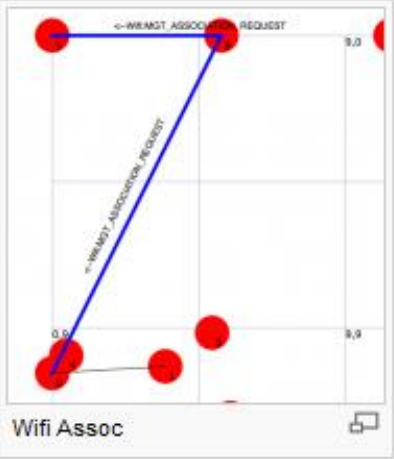
- Animate packets over wired-links and wireless-links based on trace files

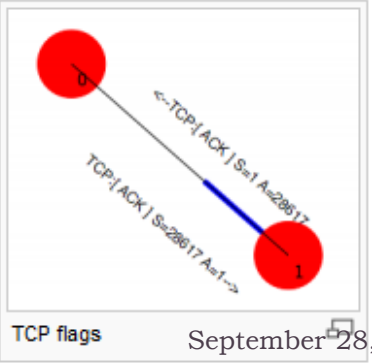

Packet Statistics


Dumbbell


Node Trajectory


Wifi Beacon


Wifi Assoc



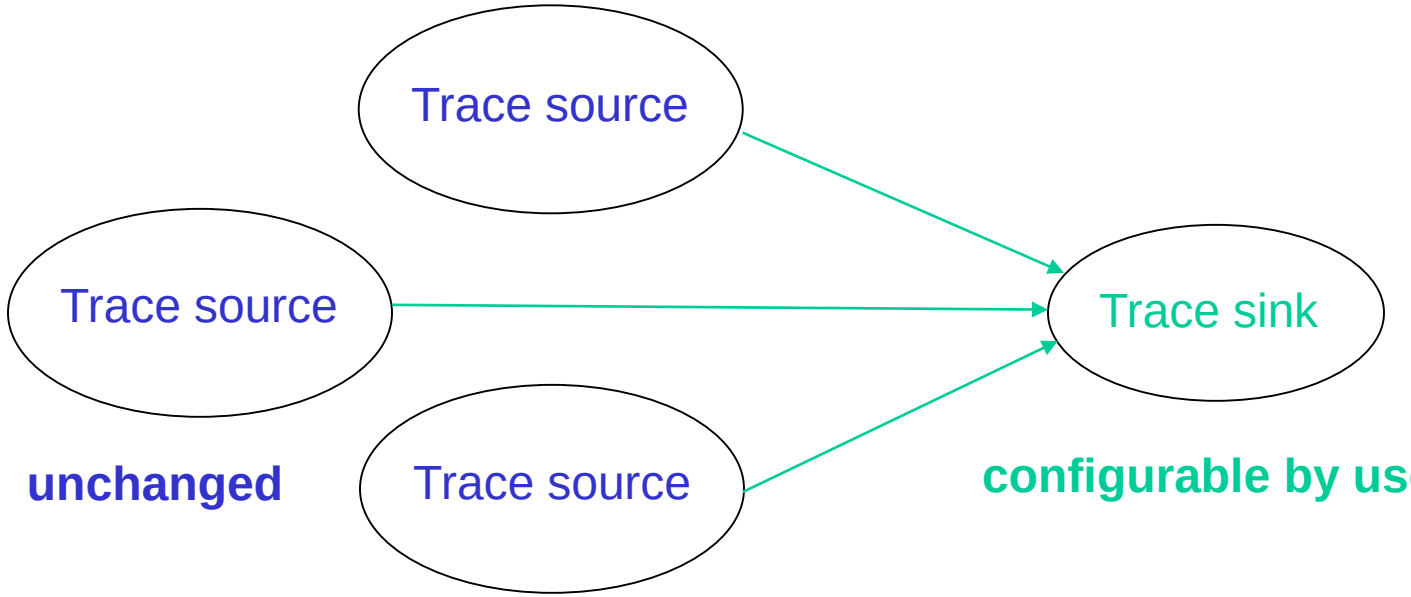
TCP flags

Emulation

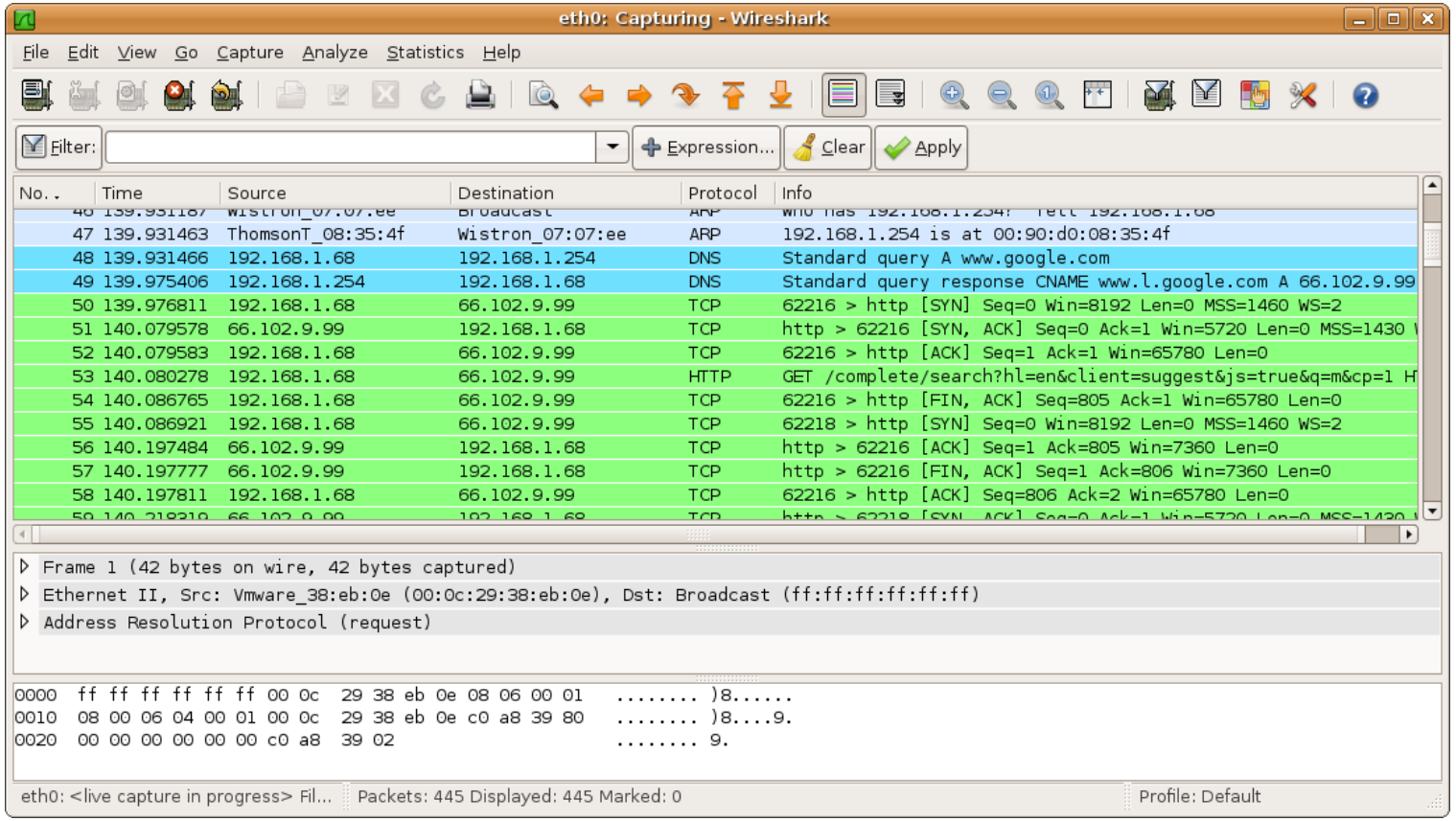
- Moving between simulation, testbeds, and live systems
  - Linux is only operating system supported

Tracing

- Decouple trace sources from trace sinks
- Users can implement specialized trace sinks and connect to existing trace sources using pointer to functions



Tracing


eth0: Capturing - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression... Clear Apply
No. Time Source Destination Protocol Info
47 139.931463 ThomsonT_08:35:4f Wistron_07:07:ee ARP 192.168.1.254 is at 00:90:d0:08:35:4f
48 139.931466 192.168.1.68 192.168.1.254 DNS Standard query A www.google.com
49 139.975406 192.168.1.254 192.168.1.68 DNS Standard query response CNAME www.l.google.com A 66.102.9.99
50 139.976811 192.168.1.68 66.102.9.99 TCP 62216 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=2
51 140.079578 66.102.9.99 192.168.1.68 TCP http > 62216 [SYN, ACK] Seq=0 Ack=1 Win=5720 Len=0 MSS=1430
52 140.079583 192.168.1.68 66.102.9.99 TCP 62216 > http [ACK] Seq=1 Ack=1 Win=65780 Len=0
53 140.080278 192.168.1.68 66.102.9.99 HTTP GET /complete/search?hl=en&client=suggest&js=true&q=m&cp=1
54 140.086765 192.168.1.68 66.102.9.99 TCP 62216 > http [FIN, ACK] Seq=805 Ack=1 Win=65780 Len=0
55 140.086921 192.168.1.68 66.102.9.99 TCP 62218 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=2
56 140.197484 66.102.9.99 192.168.1.68 TCP http > 62216 [ACK] Seq=1 Ack=805 Win=7360 Len=0
57 140.197777 66.102.9.99 192.168.1.68 TCP http > 62216 [FIN, ACK] Seq=1 Ack=806 Win=7360 Len=0
58 140.197811 192.168.1.68 66.102.9.99 TCP 62216 > http [ACK] Seq=806 Ack=2 Win=65780 Len=0
Frame 1 (42 bytes on wire, 42 bytes captured)
Ethernet II, Src: Vmware_38:eb:0e (00:0c:29:38:eb:0e), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Address Resolution Protocol (request)
0000 ff ff ff ff ff ff 00 0c 29 38 eb 0e 08 06 00 01 ....)8......
0010 08 00 06 04 00 01 00 0c 29 38 eb 0e c0 a8 39 80 ....)8....9.
0020 00 00 00 00 00 00 c0 a8 39 02 .... 9.
eth0: <live capture in progress> Fil... Packets: 445 Displayed: 445 Marked: 0 Profile: Default