ns-3 Training

Emulation

1



Outline

- Main emulation devices
 - -Tap Bridge
 - -FdNetDevice
 - -NetmapNetDevice (coming soon)



Emulation support

- Support moving between simulation and testbeds or live systems
- A real-time scheduler, and support for two modes of emulation
- Linux is only operating system supported
- Must run simulator in real time
 - GlobalValue::Bind ("SimulatorImplementationType", StringValue ("ns3::RealTimeSimulatorImpl"));
- Must enable checksum calculations across models
 - GlobalValue::Bind ("ChecksumEnabled", BooleanValue
 (true));
- Must run as root



ns-3 emulation modes



ns-3 ns-3 real real machine machine **Testbed** 2) testbeds interconnect ns-3

NETWORK SIMULATOR

machines

stacks

Example use case: testbeds

• Support for use of Rutgers WINLAB ORBIT radio grid







Example use case: mininet

- Mininet is popular in the Software-Defined Networking (SDN) community
- Mininet uses "TapBridge" integration
- <u>https://github.com/mininet/mininet/wiki/Link-modeling-using-ns-3</u>

GitHub This repository Search or type a command Explore Features Enterp	rise Blog Sign up Sign in
PUBLIC imminimet / minimet	★ Star 468 ¥ Fork 204
Home Pages History	\$
Link modeling using ns 3	Page History Clone URL ①
Contents	Mininet Get Started
 Introduction o ns-3 emulation features 	Sample Workflow Walkthrough Overview
 Link simulation with ns-3 Details How to achieve communication of ns-3 process with TAP interfaces in distinct namespaces? 	Download Documentation Videos Source Code
 Architecture: single ns-3 thread or multiple processes? Code Minimet ns-3 patches 	Apps FAQ Wiki Teaching Papers



ns-3 training, June 2018

CORE emulator





Emulation Devices



Device models

- File Descriptor Net Device (FdNetDevice)
 - read and write traffic using a file descriptor provided by the user
 - this file descriptor can be associated to a TAP device, to a raw socket, to a user space process generating/consuming traffic, etc.
- Tap Bridge
 - Integrate Tun/Tap devices with ns-3 devices



"TapBridge": netns and ns-3 integration



Tap device pushed into namespaces; no bridging needed



TapBridge modes

- ConfigureLocal (default mode)
 - ns-3 configures the tap device
 - useful for host to ns-3 interaction
- UseLocal
 - user has responsibility for device creation
 - ns-3 informed of device using "DeviceName" attribute
- UseBridge
 - TapDevice connected to existing Linux bridge



ConfigureLocal





UseLocal





UseBridge





ns-3 training, June 2018

FdNetDevice

- Unified handling of reading/writing from file descriptor
- Three supported helper configurations:
 - EmuFdNetDeviceHelper (to associate the ns-3 device with a physical device in the host machine)
 - TapFdNetDeviceHelper (to associate the ns-3 device with the file descriptor from a tap device in the host machine) (not the same as TapBridge)
 - PlanetLabFdNetDeviceHelper (to automate the creation of tap devices in PlanetLab nodes, enabling ns-3 simulations that can send and receive traffic though the Internet using PlanetLab resource.



EmuFdNetDeviceHelper

• Device performs MAC spoofing to separate emulation from host traffic





ns-3 over host sockets

- Two publications about how to run ns-3 applications over real hosts and sockets
 - "Simulator-agnostic ns-3 Applications", Abraham and Riley, WNS3 2012
 - Gustavo Carneiro, Helder Fontes, Manuel Ricardo, "Fast prototyping of network protocols through ns-3 simulation model reuse", Simulation Modelling Practice and Theory (SIMPAT), vol. 19, pp. 2063– 2075, 2011.



Generic Emulation Issues

- Ease of use
 - Configuration management and coherence
 - Information coordination (two sets of state)
 - e.g. IP/MAC address coordination
 - Output data exists in two domains
 - Debugging can be more challenging
- Error-free operation (avoidance of misuse)
 - Synchronization, information sharing, exception handling
 - Checkpoints for execution bring-up
 - Inoperative commands within an execution domain
 - Deal with run-time errors
 - Soft performance degradation (CPU) and time discontinuities



Netmap NetDevice coming soon

- 2017 ESA Summer of Code project by Pasquale Imputato
- Netmap allows ns-3 to gain direct access to the network device, bypassing the host networking stack mapingthe device memory in user space area.
- Support for flow control and BQL (see discussion in traffic control presentation)
- Pasquale reports that the pps achievable with the NetmapNetDevice at lowest layer (i.e., the write method) is up to 1.38 Mpps on the e1000e adapter on the i7 cpu at full frequency of 3.8 GHZ.
- Latency is also reduced due to better flow control



ns-3 Training

Distributed simulation

(some slides/images credit due to Peter Barnes)



Distributed simulation

• By default, ns-3 runs a single threaded event loop on a single CPU core

- Multi-core machines or multiple machines not used

- Performance is limited (CPU cycles and memory)
 - ~10⁴ packet receives/wall clock second/core *
 - heavyweight applications may consume memory (DCE, routing tables in very large topologies)

* Peter Barnes, WNS3 training 2016



Distributed simulation

- Decompose model into Logical Processes
 - Separate objects and event queues
 - Execute independently
 - Events for other LPs become messages
 - ~ MPI Ranks





Virtual time evolution

- Sometimes ahead in virtual time, sometimes behind
- Time evolution constrained by slowest LP
- No causality violations
 - Hallmark of conservative execution





Using MPI in ns-3

- Need to install MPI on the host system (OpenMPI or MPICH libraries)
- Need to pass '--enable-mpi' to './waf configure'
 - Followed by usual build
- Only point-to-point links may be used to separate LPs
 - needs some level of propagation delay to support lookahead in each LP
 - splitting of wireless channels not presently supported
- The ns-3 scenario has to be constructed by assigning nodes to LPs manually



Running with 'mpirun'

- waf can't distinguish sequential and parallel
 - Need to specify mpirun and number of ranks explicitly

Running Parallel Scripts with waf and mpirun

\$./waf --run simple-distributed
Waf: Entering directory `build/debug'
Waf: Leaving directory `build/debug'
'build' finished successfully (2.118s)
This simulation requires 2 and only 2 logical processors.
Command ['build/debug/src/mpi/examples/ns3-dev-simple-distributed-debug'] exited with code 1

Multiple ranks on a single computer:

\$./waf --run simple-distributed --command-template="mpirun -np 2 %s"
Waf: Entering directory `build/debug'
Waf: Leaving directory `build/debug'
'build' finished successfully (2.104s)
At time 1.02264s packet sink received 512 bytes from 10.1.1.1 port 49153 total Rx 512 bytes
At time 1.0235s packet sink received 512 bytes from 10.1.2.1 port 49153 total Rx 512 bytes
At time 1.02437s packet sink received 512 bytes from 10.1.3.1 port 49153 total Rx 512 bytes
At time 1.02524s packet sink received 512 bytes from 10.1.4.1 port 49153 total Rx 512 bytes

Multiple computers: \$ mpirun -np 2 ./waf -run simple-distributed



Example

examples/tutorial/third.cc

	1	* _*_ Mode·C++· c_file_stvle·"anu"· indent_tabs_mode·nil· _*_ */	7	1	* _*_ Mode:(++: c_file_style:"anu": indent_tabs_mode:nil: _*_ */	
	2	* A ha also also as a shall a la				- 10
	3	1. Include mpi-module.n			rogram is free software; you can redistribute it and/or modify	
	4	* O Como ton close confit como o Deint to .		1 I:	er the terms of the GNU General Public License version 2 as	
	5	* Z. Same lopology, spill across Point-lo-	poin	ιIIN	K hed by the Free Software Foundation;	
	6	*	_	-		
	7	* This program is distributed in the hope that it will be useful,		7	* This program is distributed in the hope that it will be useful,	
	8	* but WITHOUT ANY WARRANTY; without even the implied warranty of		8	* but WITHOUT ANY WARRANTY; without even the implied warranty of	
	9	* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the		9	* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the	
	10	* GNU General Public License for more details.		10	* GNU General Public License for more details.	
	11	 * More should be a series of the CMU Conservation bit is a series 		11	* You should have a solved a source Caller Chill Convert Databased in the source of the Chill Convert Databased in the source of the source	
	12	* You should have received a copy of the GNU General Public License		12	* You should have received a copy of the GNU General Public License	
	15	* along with this program; if not, write to the Free Software		10	* along with this program; if not, write to the Free Software	
Πİ.	14	* Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA		14	* Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA	
	16	*/		16	*/	
	17	include "nc2/cone module h"		17	include "nc2/cone module h"	
	18	include "ns3/core-module.n		18	include "ns3/core-module.n	
	19	include "hs3/potne-co-potne-module.h"		19	include "ns3/point-to-point-module.n	
-	20	include "ns3/network-module.h"		20	include "ns3/network-module.n	
	21	include "ns3/upprications-module."		21	include "ns3/wifi_module h"	
_	22	include "ns3/wohility_module h"		22	include "ns3/mobility_module h"	
	23	include "hs3/module.h"		23	include "ns3/csma-module h"	
	24	include "hs3/internet-module h"	0.0.0	24	include "ns3/internet-module h"	
	25			25		
	26	/ Default Network Topology	1 1	26	include "ns3/mpi-module.h"	
	27	/		27		-
	28	/ Wifi 10.1.3.0		28	/ Default Network Topology	
	29	/ AP		29	/ (same as third.cc from tutorial)	
	30	/ * * * *		30	<pre>/ Distributed simulation, split along the p2p link</pre>	
	31	/ 10.1.1.0		31	/ Number of wifi or csma nodes can be increased up to 250	
	32	/ n5 n6 n7 n0 n1 n2 n3 n4		32	γ Ι	
	33	/ point-to-point		33	/ Rank Ø I Rank 1	
	34	/		34	/	
	35	/ LAN 10.1.2.0		35	/ Wifi 10.1.3.0	
	36		\bigcap	36	/ AP	
	37	sing namespace ns3;	(2)	37	/ * * * *	
	38		\sim	38		
	39	S_LOG_COMPONENT_DEFINE ("ThirdScriptExample");		39	/ n5 n6 n7 n0 n1 n2 n3 n4	
	40			40	/ point-to-point	
_	41	nt Extended to the test of the test of the test of the test of		41		
	42	ain (int argc, char *argv[])		42	/ LAN 10.1.2.0	
-	43			43		
-	44	pool verpose = true;		44	ising namespace ns3;	
	46	$u_{1}\pi_{2}\Sigma_{-}\pi_{1}u_{1}m_{1}m_{2}m_{2}m_{1}m_{1}m_{1}m_{1}m_{1}m_{1}m_{1}m_{1$		45	S LOC COMPONENT DEETNE ("Thind Example Distributed"):	
Ţ	47	$u(n(s_2 - n)) = s_i$		47	S_LOG_COMPONENT_DEFINE (INTRAExampleDistributed);	~

examples/tutorial/third.cc

Ε

.

25		/ / 33	// Rank 0 Rank 1
26	1 Different log component name		
27			.3.0
28	2 Command line argument to select Nul	ll messa	
30			
31	// 10.1.1.0	39	// n5 n6 n7 n0 n1 n2 n3 n4
32	// n5 n6 n7 n0 n1 n2 n3 n4	40	// point-to-point
33	// point-to-point	41	//
34	//	42	// LAN 10.1.2.0
35	// LAN 10.1.2.0	43	
36		44	using namespace ns3;
37	using namespace ns3;	45	
30	NS LOC CONDONIENT DEETNE ("ThirdCorineExample"):	40	NS_LOG_COMPONENT_DEFINE ("Inird ExampleDistributed");
40	NS_LOG_COMPONENT_DEFINE (Intrascriptexample);	48	int
41	int	49	main (int arac, char *arav[])
42	main (int argc, char *argv[])	50	{
43	{	51	<pre>bool verbose = true;</pre>
44	<pre>bool verbose = true;</pre>	52	uint32_t nCsma = 3;
45	uint32_t nCsma = 3;	53	uint32_t nWifi = 3;
46	uint32_t nWifi = 3;	54	<pre>bool tracing = false;</pre>
47	bool tracing = false;	55	bool nullmsg = false;
40	Command in a condi	57	Command in a condi
50	cond AddValue ("nCsma" "Number of \"extra\" (SMA nodes/devices" nCsma):	58	cmd AddValue ("nCsma" "Number of \"extra\" (SMA nodes/devices" nCsma);
51	cmd.AddValue ("nWifi", "Number of wifi STA devices", nWifi):	59	cmd.AddValue ("nWifi", "Number of wifi STA devices", nWifi):
52	<pre>cmd.AddValue ("verbose", "Tell echo applications to log if true", verbose);</pre>	60	<pre>cmd.AddValue ("verbose", "Tell echo applications to log if true", verbose);</pre>
53	<pre>cmd.AddValue ("tracing", "Enable pcap tracing", tracing);</pre>	61	<pre>cmd.AddValue ("tracing", "Enable pcap tracing", tracing);</pre>
54	La construction de la constructi	2)62	cmd.AddValue ("nullmsg", "Enable the use of null-message synchronization", 🜬
55	<pre>cmd.Parse (argc,argv);</pre>	63	
56		64	cmd.Parse (argc,argv);
57 0	// Check for valid number of csma or wifi nodes	65	// Charle Com welling workers of some on wilding day
59	// 250 should be enough, otherwise IP adaresses	67	// Check for Valia humber of csma or with hodes
60 /	if (nWifi > 250 n(sma > 250)	68	// soon become an issue
61		69	if (nWifi > 250 nCsma > 250)
62 🥖	std::cout << "Too many wifi or csma nodes, no more than 250 each." << std::	70	{
63 🥖	return 1;	71 🥖	<pre>std::cout << "Too many wifi or csma nodes, no more than 250 each." << std::</pre>
64	}	72	return 1;
65		73	}
66	if (verbose)	74	
67		75	1f (verbose)
69	LogComponentEnable ("UdpEchoClientApplication", LUG_LEVEL_INFU);	70	i
A 70	LogcomponentLindble (DupechoserverApplication , Log_LEVEL_INFO);	78	LogComponentEnable ("UdpEchoServerApplication" LOG LEVEL_INFO);
₹ 71	, 	79	



examples/tutorial/third.cc

H

•

		20		
49	Command ine and	80	// Sequential fallback values	
50	1 Condition on NS3 MPT	82	1/3 sequencial failback values	
51		83	uint32_t systemCount = 1;	
53	2. Null message selector	84		
54		1)85	#ifdef NS3_MPI	
55	3. Initialize MPI	86		
56	1 Cot rook # number of rooks	~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~	// Distributed simulation setup; by default use granted time window algorithm.	1
57	4. Get fank #, number of fanks	2)89	{	
59	5 Check number of ranks	90	GlobalValue::Bind ("SimulatorImplementationType",	
60 💋		91	<pre>StringValue ("ns3::NullMessageSimulatorImpl"));</pre>	
61	6. Use symbolic names for each rank	92	}	
62		94	else {	
64	\ 7. Create point-to-point nodes /	95	GlobalValue::Bind ("SimulatorImplementationType",	
65		96	<pre>StringValue ("ns3::DistributedSimulatorImpl"));</pre>	
66	if (verbose)	97	}	
67	{ {	2 98	Noi Intonfaco: Enghla (Ranac, Ranav):	
68	LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);	J ₁₀₀	mprincerraceindore (aurge, aurge),	Jli
70	LogComponentEnable ("UapEchoServerApplication", LUG_LEVEL_INFU);	101	<pre>systemId = MpiInterface::GetSystemId ();</pre>	
71		A) ¹⁰²	<pre>systemCount = MpiInterface::GetSize ();</pre>	1
72	NodeContainer p2pNodes;	103	// Charle for wellight distributed researchers	
73	p2pNodes.Create (2);		// Check for Valia distributed parameters.	
74	DointToDointUolman naintToDoint:	5 106	if (systemCount != 2)	
76	pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps")):	107	{	
77	<pre>pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));</pre>	108	std::cout << "This simulation requires 2 and only 2 logical processors.▶ <<	
78		109	return 1;	
79	NetDeviceContainer p2pDevices;	110	3	
80	p2pDevices = pointloPoint.Install (p2pNodes);	112	<pre>#endif // NS3_MPI</pre>	
82	NodeContainer csmaNodes:	113		
83	csmaNodes.Add (p2pNodes.Get (1));	114	// System id of Wifi side	
84	csmaNodes.Create (nCsma);	D)115	uint32_t systemWifi = 0;	li
85		117	// Svstem id of CSMA side	
87	CsmaHelper Csma;	118	<pre>uint32_t systemCsma = systemCount - 1;</pre>	
88	csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));	119		
89		120	NodeContainer p2pNodes;	
90	NetDeviceContainer csmaDevices;	$(7)_{122}^{121}$	<pre>rtr<node> p2pNode1 = CreateObject<node> (systemWif1); // Create node with r Ptr<node> n2nNode2 = CreateObject<node> (systemCsma): // (reate node with r ptr</node></node></node></node></pre>	
91	<pre>csmaDevices = csma.Install (csmaNodes);</pre>	123	p2pNodes.Add (p2pNode1);	
▲ 93	NodeContainer wifiStaNodes:	124	p2pNodes.Add (p2pNode2);	
94	wifiStaNodes.Create (nWifi);	125		
-		126	DointToDointHelner pointToDoint.	"

Example

examples/tutorial/third.cc

₿

-

•

src/mpi/examples/third-distributed.cc

66	<pre>LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);</pre>	119	
67		120	NodeContainer p2pNodes;
68	1 Create CSMA nodes on one rank	121	Ptr <node> p2pNode1 = CreateObject<node> (systemWifi); // Create node with rak</node></node>
69		122	Ptr <node> p2pNode2 = CreateObject<node> (systemCsma); // Create node with rakk 1</node></node>
70	2 Croate Wifi nodes on another rank	123	p2pNodes.Add (p2pNode1);
71		124	p2pNodes.Add (p2pNode2);
72		125	
73	PointToPointHelper pointToPoint;	126	PointToPointHelper pointToPoint;
74	<pre>pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));</pre>	127	<pre>pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));</pre>
75	pointToPoint.SetChannelAttribute ("Delay", StringValue ("Zms"));	128	pointToPoint.SetChannelAttribute ("Delay", StringValue ("Zms"));
76		129	
77	NetDeviceContainer p2pDevices;	130	NetDeviceContainer p2pDevices;
70	p2puevices = point/oPoint.install (p2pNodes);	131	p2puevices = pointioPoint.Install (p2pNodes);
79	Node Control ways a second data of	122	No de Combolina en completa de co
81	NodeContainer Csmanodes;	124	NodeContainer Csmanodes;
82	csmanodes.Add (p2pNodes.Get (1));	125	CSMANOdes.Add (p2pNodes.Get (1));
83	csmanodes.treate (ntsma);	135	CSmanodes.Create (nCsma, systemCsma); // Create csma hodes with rank I
84		137	ComeHalpan come
85	Commandelper Comma;	138	Commelper Coma;
86	csma.SetChannelAttribute ("Dalau", TimeValue (NaneSeconds (CEGO)));	130	csma. SetChannel Attribute ("Dalaw", TimeValue (ManeSeconda (6560)));
87	Csma. SetChannetActribule (beray , Timevalue (Nanoseconas (0500))),	140	CSING. SetCharmerActribute (betay , Trinevalue (Nanoseconds (6500))),
88	NatDavi caContainan comaDavi cas:	141	Not-Dovi coContainon comaDovi cos:
89	csmaDevices - csma Install (csmaNodes):	142	csmaDevices - csma Install (csmaNodes):
90		143	Canabevices - Cana. Install (Canaroues),
91	NodeContainer wifiStaNodes:	144	NodeContainer wifiStaNodes:
92	wifiStaNodes (reate (nWifi):	145	wifiStaNodes (reate (nWifi systemWifi): // (reate wifi nodes with rank 0
93	NodeContainer wifiAnNode = $p2nNodes.Get(0)$:	146	NodeContainer wifiAnNode = $p2nNodes$. Get (0):
94		147	
95	YansWifiChannelHelper channel = YansWifiChannelHelper::Default ():	148	YansWifiChannelHelper channel = YansWifiChannelHelper::Default ():
96	YansWifiPhyHelper phy = YansWifiPhyHelper::Default ():	149	YansWifiPhyHelper phy = YansWifiPhyHelper::Default ():
97	phy.SetChannel (channel.Create ());	150	<pre>phy.SetChannel (channel.Create ());</pre>
98		151	
99	WifiHelper wifi = WifiHelper::Default ();	152	WifiHelper wifi = WifiHelper::Default ();
100	<pre>wifi.SetRemoteStationManager ("ns3::AarfWifiManager");</pre>	153	<pre>wifi.SetRemoteStationManager ("ns3::AarfWifiManager");</pre>
101		154	
102	NqosWifiMacHelper mac = NqosWifiMacHelper::Default ();	155	NqosWifiMacHelper mac = NqosWifiMacHelper::Default ();
103		156	
104	Ssid ssid = Ssid ("ns-3-ssid");	157	Ssid ssid = Ssid ("ns-3-ssid");
105	<pre>mac.SetType ("ns3::StaWifiMac",</pre>	158	<pre>mac.SetType ("ns3::StaWifiMac",</pre>
106	"Ssid", SsidValue (ssid),	159	"Ssid", SsidValue (ssid),
107	"ActiveProbing", BooleanValue (false));	160	"ActiveProbing", BooleanValue (false));
108		161	
109	NetDeviceContainer staDevices;	162	NetDeviceContainer staDevices;
110	<pre>staDevices = wifi.Install (phy, mac, wifiStaNodes);</pre>	163	<pre>staDevices = wifi.Install (phy, mac, wifiStaNodes);</pre>
111		164	
₹ 112	mac SetType ("ns3::AnWifiMac"	165	mac SetType ("ns3++AnWifiMac"

Example

examples/tutorial/third.cc

₿

-

H

►

<pre>1. Install devices, addresses and internet stack everywhere 2. Install applications only on rank- local nodes implications onthy only only only only only only only onl</pre>	136	stack Install (csmallodes);	\frown	195	address SetBase ("10.1.1.0", "255.255.255.0");
<pre>133 14. Install devices, addresses and 157 158 159 151 151 152 152 154 155 155 155 155 155 155 155 155 155</pre>	137	7 Y	1	196	Ipv4InterfaceContainer p2pInterfaces;
<pre>internet stack everywhere internet everywhere every</pre>	138	1. Install devices, addresses and		197	<pre>p2pInterfaces = address.Assign (p2pDevices);</pre>
11 Internet stack everywhere 2. Install applications only on rank- local nodes 14 Orders.SetBose (10.1.3.0", "25.25.25.0"); address.Assign (charder); cmainterfaces = address.Assign (charder); cmainterfaces = address.Assign (charder); cmainterfaces = address.Assign (charder); cmainterfaces = address.Assign (charder); address.Assign (charder); cmainterfaces = address.Assign (charder); cmainterfaces = address.Assign (charder); address.Assign (charder); cmainterfaces = address.Assign (charder); cmainter = charder = true); cmainterfaces = address.Assign (charder); cmainte	139			198	
2. Install applications only on rank- local nodes address.sign (stalevices); address.sign (s	140	Internet stack everywhere		199	address.SetBase ("10.1.2.0", "255.255.255.0");
<pre>2. Install applications only on rank- local nodes Invdinterfaces: address.astig (subartices); address.stig (subartices); address.stig (subartices); address.stig (subartices); address.astig (subartices); addre</pre>	141	O Install applications only on rank		200	ipv4interfaces - address Assian (csmaDevices);
<pre>144 145 146 147 148 148 148 148 148 149 149 149 149 149 149 149 149 149 149</pre>	143	2. Install applications only on rank-		202	
115 204 address.Assign (staDevices); 146 inderess.Assign (csmolevices); address.Assign (apdevices); 147 inderess.SetBase ("10.1.3.0", "255.255.25.0"); address.Assign (csmolevices); 148 csmolnterfaces - address.Assign (staDevices); 206 149 address.Assign (staDevices); 207 144 ddress.Assign (staDevices); 208 151 address.Assign (staDevices); 208 152 address.Assign (staDevices); 201 153 address.Assign (staDevices); 201 154 udpEchoServerHelper echoServer (9); 212 155 applicationContainer serverApps = echoServer.Install (csmaNodes.Get (nCsmo)); serverApps.Start (Seconds (1.0)); 155 serverApps.Start (Seconds (1.0)); 217 156 echoClient.SetAttribute ("Macketes", UnitegerValue (302); 218 159 // If this rank is systemWifi 219 160 udpEchoClientHelper echoClient (csmolnterfaces.GetAdress (nCsmo), 9); // If (staDuld contain the client application, 210, 211, 212 161 echoClient.Isstall (wifiStaNodes.Get (nWifi - 1)); (If (systemId = systemWifi) 218 161 echoClient	144	local nodes		203	address.SetBase ("10.1.3.0", "255.255.255.0");
146 Apv4Interface(ontainer csmaInterfaces; 147 Ipv4Interface(ontainer csmaInterfaces; 148 csmaInterfaces = address.Assign (csmaDevices); 149 address.Assign (staDevices); 151 address.Assign (polve); 153 address.Assign (polve); 154 UdpEchoServerHelper echoServer (9); 154 UdpEchoServerHelper echoServer (9); 155 ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get (nCsma)); 157 serverApps.Start (Seconds (1.0)); 158 serverApps.Start (Seconds (1.0)); 159 udpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma)); 158 serverApps.Start (Seconds (1.0)); 159 udpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma)); 159 v/ If this rank is systemWifil 160 UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma)); 170 IpvEchoLient.Install (wifiStaNodes.Get (nWifi - 1)); 166 echoClient.SetAttribute ("MaxPackets", UintegerValue (102+); 167 clientApps.Start (Seconds (1.0.0); 171 j 172 simulation:.istattribute ("MaxPackets", UintegerValue (102+);	145)	204	address.Assign (staDevices);
<pre>1/4 [p4]nterfaces address.setBase ("10.1.3.e", "255.255.25.0"); 1/4 ddress.SetBase ("10.1.3.e", "255.255.25.0"); 1/4 ddress.SetBase ("10.1.3.e", "255.255.25.0"); 1/4 ddress.Assign (staDevices); 1/4 ddress.Assign (staDevices); 1/4 udpEchoServerHelper echoServer (9); 1/5 dupEchoServerHelper echoServer (9); 1/6 udpEchoServerHelper echoServer (9); 1/6 udpEchoServerHelper echoServer.Install (csmaNodes.Get (nCsma)); 1/6 serverApps.Start (Seconds (1.0.0)); 1/6 serverApps.Start (Seconds (1.0.0)); 1/6 serverApps.Start (Seconds (1.0.0)); 1/6 echoClient.SetAttribute ("MaxRackets", UintegerValue (1)); 1/6 echoClient.SetAttribute ("MaxRackets", UintegerValue (1)); 1/6 echoClient.SetAttribute ("MaxRackets", UintegerValue (1024)); 1/6 echoClient.Install (wifiStaNodes.Get (nWifi - 1)); 1/6 clientApps.Start (Seconds (10.0)); 1/6 echoClient.Install (wifiStaNodes.Get (nWifi - 1)); 1/6 echoClient.SetAttribute ("MaxRackets", UintegerValue (1024)); 1/6 lientApps.Start (Seconds (10.0)); 1/7 serverApps.Start (Seconds (10.0)); 1/8 serverApps.Start (Seconds (1.0.0)); 1/8 echoClient.Install (wifiStaNodes.Get (nWifi - 1)); 1/9 // I this rank is systemWifi 1/0 ludpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9); 1/7 echoClient.SetAttribute ("MaxRackets", UintegerValue (1024)); 1/8 echoClient.Install (wifiStaNodes.Get (nWifi - 1)); 1/8 echoClient.Install (wifiStaNodes.Get (nWifi - 1)); 1/8 echoClient.SetAttribute ("MaxRackets", UintegerValue (1024)); 1/9 // I this rank is systemWifi 1/0 ludpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9); 1/9 echoClient.SetAttribute ("MaxRackets", UintegerValue (1024)); 1/9 echoClient.SetAttribute ("MaxRackets", UintegerValue (1024)); 1/9 // I this rank is systemWifi 1/0 ludpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9); 1/9 echoClient.SetAttribute ("MaxRackets", UintegerValue (1024)); 1/9 // I this rank is systemWifi 1/0 ludpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9); 1/9 echoClient.SetAttribute ("MaxRackets", UintegerV</pre>	146	a	· · ·	205	address.Assign (apDevices);
<pre> 149 149 149 149 149 149 149 149 149 149</pre>	147	Ipv4InterfaceContainer csmaInterfaces;	[206	// If this mapk is sustanGama
150 address.SetBose ("10.1.3.0", "255.255.25.0"); 151 address.Assign (staDevices); 152 address.Assign (staDevices); 153 address.Assign (staDevices); 154 UdpEchoServerHelper echoServer (9); 155 ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get (nCsma)); 156 ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get (nCsma)); 157 serverApps.Start (Seconds (10.0)); 158 serverApps.Stop (Seconds (10.0)); 159 serverApps.Stop (Seconds (10.0)); 159 echoClient.SetAttribute ("MaxPackets", UintegerValue (1)); 161 UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9); 162 echoClient.SetAttribute ("MaxPackets", UintegerValue (1024)); 164 ApplicationContainer clientApps = 165 ApplicationContainer (Seconds (10.0)); 166 choClient.Istattribute ("MaxPackets", UintegerValue (1024)); 167 choClient.SetAttribute ("MaxPackets", UintegerValue (1024)); 168 ApplicationContainer clientApps 169 ldpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9); 165 echoClient.Istattribute ("MaxPackets", UintegerValue (1024));	149	csindificer aces = address.Assign (csindlevices),		208	// it should contain the server application.
151 address.Assign (staDevices); 152 address.Assign (apDevices); 153 UdpEchoServerHelper echoServer (9); 154 UdpEchoServerHelper echoServer.Install (csmaNodes.Get (nCsma)); 155 ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get (nCsma)); 156 ApplicationContainer serverApps.Stor (Seconds (1.0)); 157 serverApps.Stor (Seconds (1.0)); 158 serverApps.Stop (Seconds (1.0)); 159 218 160 UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9); 151 echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)); 157 echoClient.SetAttribute ("MaxPackets", UintegerValue (10); 158 echoClient.SetAttribute ("MaxPackets", UintegerValue (1024)); 159 // If this rank is systemWifi 160 UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9); 161 echoClient.SetAttribute ("PacketSize", UintegerValue (1024)); 162 echoClient.SetAttribute ("PacketSize", UintegerValue (1024)); 164 echoClient.SetAttribute ("MaxPackets", UintegerValue (1024)); 165 echoClient.SetAttribute ("MaxPackets", UintegerValue (1024)); 166 echoClient.SetAttribute (150	address.SetBase ("10.1.3.0", "255.255.255.0");	\frown	209	// since it is on one of the csma nodes
152 address.Assign (apDevices); 211 { 153 UdpEchoServerHelper echoServer (9); 211 { 154 UdpEchoServerHelper echoServer (9); 213 ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get (nCsmo)); 155 ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get (nCsmo)); 35 serverApps.Stop (Seconds (1.0.0)); 158 serverApps.Stop (Seconds (10.0)); 101 serverApps.Stop (Seconds (10.0)); 102 159 UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9); echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0))); 101 echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0))); 102 114 160 echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0))); 102 114 115 161 echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0))); 115 115 116	151	address.Assign (staDevices);		210	if (systemId == systemCsma)
153 UdpEchoServerHelper echoServer (9); 154 UdpEchoServerHelper echoServer (9); 155 ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get (nCsma)); 156 ApplicationContainer serverApps.Start (Seconds (1.0)); 158 serverApps.Start (Seconds (10.0)); 159 217 160 UdpEchoServer.Install (csmaNodes.Get (nCsma)); 159 217 160 UdpEchoClient.fectAttribute ("MaxPackets", UintegerValue (1)); 161 echoClient.SetAttribute ("MaxPackets", UintegerValue (1024)); 162 echoClient.SetAttribute ("MaxPackets", UintegerValue (1024)); 164 ApplicationContainer clientApps = 165 ApplicationContainer clientApps = 166 echoClient.Install (wifiStaNodes.Get (nWifi - 1)); 167 clientApps.Start (Seconds (2.0)); 168 clientApps.Start (Seconds (10.0)); 171 Ipv4GlobalRoutingHelper::PopulateRoutingTables (); 172 Simulator::Stop (Seconds (10.0)); 173 f 174 if (tracing == true) 175 { 176 maintInPoint EnchleParentIL ("thind-dittributedif");	152	address.Assign (apDevices);	Ζ,	211	{
<pre>134 UdpEchoServerHelper echoServer(3); 135 ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get (nCsma)); 136 serverApps.Start (Seconds (1.0)); 137 serverApps.Start (Seconds (1.0)); 138 serverApps.Start (Seconds (1.0)); 139 echoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9); 140 UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9); 151 echoClient.SetAttribute ("MaxPackets", UintegerValue (1)); 152 echoClient.SetAttribute ("NaxPackets", UintegerValue (1024)); 154 echoClient.Install (wifiStaNodes.Get (nWifi - 1)); 155 clientApps.Stop (Seconds (10.0)); 156 clientApps.Stop (Seconds (10.0)); 157 clientApps.Stop (Seconds (10.0)); 158 serverApps.Start (Seconds (10.0)); 159 echoClient.Install (wifiStaNodes.Get (nWifi - 1)); 159 clientApps.Stop (Seconds (10.0)); 150 intLops.Start (Seconds (10.0)); 151 clientApps.Stop (Seconds (10.0)); 152 clientApps.Stop (Seconds (10.0)); 153 intLops.Start (Seconds (10.0)); 154 clientApps.Stop (Seconds (10.0)); 155 intLops.Start (Seconds (10.0)); 156 clientApps.Stop (Seconds (10.0)); 157 f 158 intLops.Start (Seconds (10.0)); 159 intLops.Start (Seconds (10.0)); 159 intLops.Start (Seconds (10.0)); 150 intLops.Start (Seconds (10.0)); 151 if (tracing = true) 152 intLops.it ExphleRemain[a = true] 155 intLops.it ExphleRemain[a = true] 156 intLops.Start (Seconds (10.0)); 157 intLops.it ExphleRemain[a = true] 158 intLops.it ExphleRemain[a = true] 159 intLops.it ExphleRemain[a = true] 150 intLops.it ExphleRemain[a = true] 151 intLops.it ExphleRemain[a = true] 155 intLops.it ExphleRemain[a = true] 156 intLops.it ExphleRemain[a = true] 157 intLops.it ExphleRemain[a = true] 158 intLops.it ExphleRemain[a = true] 159 intLops.it ExphleRemain[a = true] 150 intLops.it ExphleRemain[a = true] 150 intLops.it ExphleRemain[a = true] 151 intLops.it ExphleRemain[a = true] 152 intLops.it ExphleRemain[a = true] 153 intLops.it ExphleRemain[a = true] 154 intLops.it ExphleRemain[a = true] 155 intLops.it ExphleRemain</pre>	153		\sim	212	UdpEchoServerHelper echoServer (9);
ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get (nCsma)); isf ApplicationContainer serverApps.Stort (Seconds (1.0)); isf serverApps.Stort (Seconds (10.0)); isf cholient.SetAttribute ("MaxPackets", UintegerValue (10); ie echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0))); ie echoClient.Install (wifiStaNodes.Get (nWifi - 1)); ie cholient.SetAttribute ("MaxPackets", UintegerValue (1024)); ie echoClient.Install (wifiStaNodes.Get (nWifi - 1)); ie cholient.SetAttribute ("MaxPackets", UintegerValue (10); ie echoClient.SetAttribute ("MaxPackets", UintegerValue (1024)); ie echoClient.Install (wifiStaNodes.Get (nWifi - 1)); ie cholient.SetAttribute ("MaxPackets", UintegerValue (1024)); ie cholient.SetAttribute ("MaxPackets", UintegerValue (1024)); ie cholient.SetAttribute ("MaxPackets", UintegerValue (1024));	154	UdpEchoServerHelper echoServer (9);		213	Auglierting Carleinen annundung ander Genung Tratelli (annuhleta Cat. (a Carrol)
<pre>hpp:fide:Understand output of serverApps.Start (Seconds (10.0)); serverApps.Start (Seconds (20.0)); serverApps.Start (Seconds (20.0)); serverApps.Start (Seconds (10.0)); serverApps.Start</pre>	155	ApplicationContainer serverApps - echoServer Install (csmaNodes Get (n(sma));		214	ApplicationContainer serverApps = echoserver.install (csmanodes.Get (ncsma)) serverApps Start (Seconds (1.0)):
158 serverApps.Stop (Seconds (10.0)); 217 } 159 217 } 210 160 UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9); echoClient.SetAttribute ("MaxPackets", UintegerValue (1)); 219 // If this rank is systemWifi 21 161 echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0))); 221 // is should contain the client application, 222 162 echoClient.IsetAttribute ("PacketSize", UintegerValue (1024)); 221 // since it is on one of the wifi nodes 222 if (systemId == systemWifi) 164 echoClient.Install (wifiStaNodes.Get (nWifi - 1)); 165 clientApps.Start (Seconds (2.0)); echoClient.SetAttribute ("MaxPackets", UintegerValue (1024)); 224 UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9); echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0))); 225 echoClient.SetAttribute ("MaxPackets", UintegerValue (1)); 226 167 clientApps.Stop (Seconds (10.0)); 128 227 echoClient.SetAttribute ("PacketSize", UintegerValue (1024)); 168 clientApps.Stop (Seconds (10.0)); 227 echoClient.Install (wifiStaNodes.Get (nWifi - 1)); 228 171 Simulator::Stop (Seconds (10.0)); 229 ApplicationContainer clie	157	serverApps. Start (Seconds (1.0)):		216	serverApps.Stop (Seconds (10.0));
159 218 160 UdpEchoClientHelper echoClient (smaInterfaces.GetAddress (nCsma), 9); echoClient.SetAttribute ("MaxPackets", UintegerValue (1)); echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0))); echoClient.SetAttribute ("Interval", TimeValue (1024)); 220 // it should contain the client application, // since it is on one of the wifi nodes 221 // it should contain the client application, // since it is on one of the wifi nodes 163 echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0))); 164 224 UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9); echoClient.Install (wifiStaNodes.Get (nWifi - 1)); 165 224 UdpEchoClient.SetAttribute ("MaxPackets", UintegerValue (1)); echoClient.SetAttribute ("MaxPackets", UintegerValue (1)); 166 226 echoClient.SetAttribute ("MaxPackets", UintegerValue (1)); echoClient.SetAttribute ("NaxPackets", UintegerValue (1024)); 164 224 UdpEchoClient.SetAttribute ("MaxPackets", UintegerValue (1024)); echoClient.SetAttribute ("PacketSize", UintegerValue (1024)); 165 clientApps.Stop (Seconds (10.0)); 173 10 10 226 174 if (tracing -= true) 175 233 234 234 175 q 234 234 234 176 romitToPoint Employeenall ("thick-distributed=wifi"); 235 ImutGlobalRoutingHelper::PopulateRoutingTopLes (); 176 romitToPoint Emplo	158	serverApps.Stop (Seconds (10.0));		217	}
160 UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9); 161 echoClient.SetAttribute ("MaxPackets", UintegerValue (1)); 162 echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0))); 163 echoClient.SetAttribute ("PacketSize", UintegerValue (1024)); 164 achoClient.Install (wifiStaNodes.Get (nWifi - 1)); 165 ApplicationContainer clientApps = echoClient.Install (wifiStaNodes.Get (nWifi - 1)); echoClient.SetAttribute ("MaxPackets", UintegerValue (1024)); 166 echoClient.Install (wifiStaNodes.Get (nWifi - 1)); echoClient.SetAttribute ("MaxPackets", UintegerValue (1024)); 167 clientApps.Stor (Seconds (10.0)); echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0))); 168 clientApps.Stop (Seconds (10.0)); echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0))); 169 if (tracing = true) 227 echoClient.SetAttribute ("Interval", TimeValue (1024)); 173 if (tracing == true) 233 if (tracing == true) 174 if (tracing == true) 234 234 234 234 176 paintTappsint EnchlePcontl ("thind-distributed.wifi"): 234 234 234 234 234 234 234 234	159			218	
<pre>161 echoClient.SetAttribute ("MaxPackets", UintegerValue (1)); 162 echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0))); 163 echoClient.SetAttribute ("PacketSize", UintegerValue (1024)); 164 165 ApplicationContainer clientApps = 166 echoClient.Install (wifiStaNodes.Get (nWifi - 1)); 167 clientApps.Star (Seconds (2.0)); 168 clientApps.Stop (Seconds (10.0)); 169 170 Ipv4GlobalRoutingHelper::PopulateRoutingTables (); 173 Simulator::Stop (Seconds (10.0)); 174 if (tracing == true) 175 { 175 { 176 Ipv4GlobalRoutingHelpert::PopulateRoutingTables (); 177 Ipv4GlobalRoutingHelper::PopulateRoutingTables (); 178 If (tracing == true) 175 { 179 Ipv4GlobalRoutingHelper::PopulateRoutingTables (); 170 Ipv4GlobalRoutingHelper::PopulateRoutingTables (); 173 If (tracing == true) 175 { 176 Ipv4GlobalRoutingHelper::PopulateRoutingTables (); 177 Ipv4GlobalRoutingHelper::PopulateRoutingTables (); 178 If (tracing == true) 175 { 179 Ipv4GlobalRoutingHelper::PopulateRoutingTables (); 170 Ipv4GlobalRoutingHelper::PopulateRoutingTables (); 171 Ipv4GlobalRoutingHelper::PopulateRoutingTables (); 172 Simulator::Stop (Seconds (10.0)); 173 If (tracing == true) 175 { 174 If (tracing == true) 175 { 175 [176 Ipv4GlobalRoutingHelper::PopulateRoutingTables (); 177 Ipv4GlobalRoutingHelper::PopulateRoutingTables (); 178 Ipv4GlobalRoutingHelper::PopulateRoutingTables (); 179 Ipv4GlobalRoutingHelper::PopulateRoutingTables (); 170 Ipv4GlobalRoutingHelper::PopulateRoutingTables (); 171 Ipv4GlobalRoutingHelper::PopulateRoutingTables (); 172 Ipv4GlobalRoutingHelper::PopulateRoutingTables (); 173 Ipv4GlobalRoutingHelper::PopulateRoutingTables (); 174 If (tracing == true) 175 { 175 } 176 Ipv4GlobalRoutingHelper::PopulateRoutingTables (); 177 Ipv4GlobalRoutingHelper::PopulateRoutingTables (); 178 Ipv4GlobalRoutingHelper::PopulateRoutingTables (); 179 Ipv4GlobalRoutingHelper::PopulateRoutingTables (); 170 Ipv4GlobalRoutingHelper::PopulateRoutingTables (); 171 Ipv4GlobalRoutingHelper::PopulateRo</pre>	160	UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9);	\backslash	219	// If this rank is systemWifi
<pre>162 echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0))); 163 echoClient.SetAttribute ("PacketSize", UintegerValue (1024)); 164 165 ApplicationContainer clientApps = 166 echoClient.Install (wifiStaNodes.Get (nWifi - 1)); 167 clientApps.Start (Seconds (2.0)); 168 clientApps.Stop (Seconds (10.0)); 169 170 Ipv4GlobalRoutingHelper::PopulateRoutingTables (); 171 172 Simulator::Stop (Seconds (10.0)); 173 if (tracing == true) 174 if (tracing == true) 175 clientApps.Stop (Seconds (10.0)); 175 int (tracing == true) 176 clientApps.Stop (Seconds (10.0)); 177 if (tracing == true) 176 clientApps.Stop (Seconds (10.0)); 177 if (tracing == true) 178 if (tracing == true) 179 Ipv4GlobalRoutingHelper::PopulateRoutingTables (); 170 Ipv4GlobalRoutingHelper::PopulateRoutingTables (); 171 if (tracing == true) 175 if (tracing == true) 176 if (tracing == true) 176 if (tracing == true) 177 if (tracing == true) 178 if (tracing == true) 179 Ipv4GlobalRoutingHelper::PopulateRoutingTables (); 170 Ipv4GlobalRoutingHelper::PopulateRoutingTables (); 173 if (tracing == true) 175 if (tracing == true) 176 clientApps.Stop (Seconds (10.0)); 177 if (tracing == true) 178 if (tracing == true) 179 if (tracing == true) 170 Ipv4GlobalRoutingHelper::PopulateRoutingTables (); 170 Ipv4GlobalRoutingHelper::PopulateRoutingTables (); 171 if (tracing == true) 175 if (tracing == true) 176 if (tracing == true) 177 if (tracing == true) 178 if (tracing == true) 179 if (tracing == true) 170 if (tracing == true) 170 if (tracing == true) 170 if (tracing == true) 171 if (tracing == true) 175 if (tracing == true) 175 if (tracing == true) 176 if (tracing == true) 177 if (tracing == true) 178 if (tracing == true) 179 if (tracing == true) 170 if (tracing == true) 170 if (tracing == true) 170 if (tracing == true) 170 if (tracing == true) 171 if (tracing == true) 175 if (tracing == true) 175 if (tracing == true) 176 if (tracing == true) 177 if (tracing == true) 178 if (tracing == true) 179 if (tr</pre>	161	echoClient.SetAttribute ("MaxPackets", UintegerValue (1));	\frown	220	// it should contain the client application,
<pre>164 echoCtrent.setAttribute (Packetsize , officegervatue (1024)), 165 166 echoCtient.Install (wifiStaNodes.Get (nWifi - 1)); 167 clientApps.Start (Seconds (2.0)); 168 clientApps.Stop (Seconds (10.0)); 169 170 Ipv4GlobalRoutingHelper::PopulateRoutingTables (); 171 172 Simulator::Stop (Seconds (10.0)); 173 174 if (tracing == true) 175 175 175 176</pre>	162	echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));	2)	221	// since it is on one of the wifi nodes
165 ApplicationContainer clientApps = 166 echoClient.Install (wifiStaNodes.Get (nWifi - 1)); 167 clientApps.Start (Seconds (2.0)); 168 clientApps.Stop (Seconds (10.0)); 169 224 170 Ipv4GlobalRoutingHelper::PopulateRoutingTables (); 171 229 172 Simulator::Stop (Seconds (10.0)); 173 if (tracing == true) 174 if (tracing == true) 175	164	echocitent. setActi ibute (Pucketsize , othreger value (102+)),	Z)	223	
166 echoClient.Install (wifiStaNodes.Get (nWifi - 1)); 167 clientApps.Start (Seconds (2.0)); 168 clientApps.Stop (Seconds (10.0)); 169 227 170 Ipv4GlobalRoutingHelper::PopulateRoutingTables (); 171 230 172 Simulator::Stop (Seconds (10.0)); 173 if (tracing == true) 175 174 175 175 176 nointToPoint EngbleProphile ("third-distributed.wifi"));	165	ApplicationContainer clientApps =		224	UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9);
167 clientApps.Start (Seconds (2.0)); 168 clientApps.Stop (Seconds (10.0)); 169 226 170 Ipv4GlobalRoutingHelper::PopulateRoutingTables (); 171 229 171 230 172 Simulator::Stop (Seconds (10.0)); 173 231 174 if (tracing == true) 175 233 176	166	<pre>echoClient.Install (wifiStaNodes.Get (nWifi - 1));</pre>		225	<pre>echoClient.SetAttribute ("MaxPackets", UintegerValue (1));</pre>
168 clientApps.Stop (Seconds (10.0)); 169 227 169 228 170 Ipv4GlobalRoutingHelper::PopulateRoutingTables (); 171 230 172 Simulator::Stop (Seconds (10.0)); 173 230 174 if (tracing == true) 175 { 176 233 177 1 178 234 179 1 170 1 171 235 172 1 173 231 174 if (tracing == true) 175 { 176	167	<pre>clientApps.Start (Seconds (2.0));</pre>		226	<pre>echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));</pre>
109 Ipv4GlobalRoutingHelper::PopulateRoutingTables (); 229 ApplicationContainer clientApps = 171 230 echoClient.Install (wifiStaNodes.Get (nWifi - 1)); 172 Simulator::Stop (Seconds (10.0)); 231 clientApps.Start (Seconds (2.0)); 173 if (tracing == true) 233 j 175 { 234 234 176	168	clientApps.Stop (Seconds (10.0));	٦.	227	<pre>echoClient.SetAttribute ("PacketSize", UintegerValue (1024));</pre>
171 171 171 230 172 Simulator::Stop (Seconds (10.0)); 173 231 174 if (tracing == true) 175 { 176	170	Try/(Clobal PourtingHolmon: · Ponul at a PourtingTables ():	\backslash	228	ApplicationContainan clientApps -
172 Simulator::Stop (Seconds (10.0)); 173	171	ipvistobutkouttignetperropututekouttigrubtes (),		230	echoClient.Install (wifiStaNodes.Get (nWifi - 1)):
173 174 if (tracing == true) 175 { 176	172	Simulator::Stop (Seconds (10.0));		231	clientApps.Start (Seconds (2.0));
174 if (tracing == true) 175 { 176	173			232	clientApps.Stop (Seconds (10.0));
175 { 176 point ToPoint EngblePcapAll ("third-distributed-wifi"): 234 235 Toy4GlobalRoutingHelper::DopulateRoutingTables ():	174	if (tracing == true)	ľ	233	}
$1/b$ $noint[oVoint Frab]oVoint[("third_dictributed_witi"); (3) 1/b$	175		٦	234	
177 pbtterofiletelikater (distributed wirft), and and a second se	176	pointioroint.EnablePcapAll ("third-distributed-wifi");		235	<pre>ipv4GlobalKoutingHelper::PopulateRoutingTables ();</pre>
178 csm EngblePcon ("third-distributed-wift", csm Devices det (0), true) 237 Simulator::Ston (Seconds (10.0))	178	csma EnablePcap ("third-distributed-wifi", appevices.Get (0), true);		237	Simulator::Stop (Seconds (10 0)):
179 238	179		1 \	238	
180 pointToPoint.EnablePcapAll ("third-distributed-csma"); 239 if (tracing == true)	180	<pre>pointToPoint.EnablePcapAll ("third-distributed-csma");</pre>	11	239	if (tracing == true)
181 phy.EnablePcap ("third-distributed-csma", apDevices.Get (0)); 240 {	181	<pre>phy.EnablePcap ("third-distributed-csma", apDevices.Get (0));</pre>		240	{



examples/tutorial/third.cc

src/mpi/examples/third-distributed.cc

142	address.SetBase ("10.1.1.0", "255.255.255.0");		223	1
143			224	UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9);
144			225	echoClient.SetAttribute ("MaxPackets", UintegerValue (1)):
145			226	acholient SetAttribute ("Interval", TimeValue (Seconds (1.0)))
146	1 Enable PCAP tracing on local		227	echoClient SetAttribute ("PacketSize", HintegerValue (1024));
147	1. Enablet erti traeing en leea		228	echocitent. Secarci ibute (FucketSize , othreger vulue (1024)),
148	nodes?		220	AugliesticsContainer alientAuge
140			223	Application container clientApps =
149	2 Close MDI eleentry		230	echollient.Install (wifiStaNodes.Get (nWifi - 1));
150			231	clientApps.Start (Seconds (2.0));
151			232	clientApps.Stop (Seconds (10.0));
152			233	{
153			234	
154	UdpEchoServerHelper echoServer (9);		235	<pre>Ipv4GlobalRoutingHelper::PopulateRoutingTables ();</pre>
155			236	
156	<pre>ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get (nCsma));</pre>		237	Simulator::Stop (Seconds (10.0));
157	<pre>serverApps.Start (Seconds (1.0));</pre>		238	
158	serverApps.Stop (Seconds (10.0));		239	if (tracing == true)
159			240	{
160	UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9);		241	// Depending on the system Id (rank), the pcap information
161	<pre>echoClient.SetAttribute ("MaxPackets", UintegerValue (1));</pre>		242	<pre>// traced will be different. For example, the ethernet pcap</pre>
162	<pre>echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));</pre>		243	// will be empty for rank0, since these nodes are placed on
163	<pre>echoClient.SetAttribute ("PacketSize", UintegerValue (1024));</pre>		244	// on rank 1. All ethernet traffic will take place on rank 1.
164	······································		245	// Similar differences are seen in the p2p and wireless pcaps.
165	ApplicationContainer clientApps =	(4	246	if (systemId == systemWifi)
166	echo(lient Install (wifiStaNodes Cet (nWifi = 1));		247	s
167	clientAnne Start (Seconds (2.0)):		248	nointToPoint EnghlePcanAll ("third_distributed_wifi"):
168	clientAnns Ston (Seconds (10 0));		249	nby EnghleBcan ("third-distributed-wifi" anDevices (et (0)):
169			250	some EnghleDcon ("thind-distributed wifi", oppevices.det (0);
170	Inv/() ohal PoutingHalmon; Ponul at a PoutingTables ();		251	
171	ipv+droburkourrigherperFopururekourrightbres (),		252	
172	Simulatory Stor (Cocords (10 0))		252	etse // systemcsma
172	StimulatorStop (Seconds (10.0));		255	1 maintTaDaint EnghlaDaanAll ("thind distributed same"):
173	16 (Institute - Inst)		254	pointiopoint.EnablePcapAll (third-distributed-csmd);
174	if (tracing == true)		255	phy.EnablePcap ("third-distributed-csma", appevices.Get (0));
175			250	csma.EnablePcap ("thira-aistributea-csma", csmaDevices.Get (0), true);
170	pointioroint.EnablePcapAll ("thira-alstributea-with");		257	
177	pny.EnablePcap ("third-distributed-wifi", apDevices.Get (0));		258	1
1/8	csma.EnablePcap ("third-distributed-wifi", csmaDevices.Get (0), true);	_/ /	259	
1/9			260	Simulator::Run ();
180	<pre>pointToPoint.EnablePcapAll ("third-distributed-csma");</pre>		261	Simulator::Destroy ();
181	<pre>phy.EnablePcap ("third-distributed-csma", apDevices.Get (0));</pre>	\frown	262	
182	<pre>csma.EnablePcap ("third-distributed-csma", csmaDevices.Get (0), true);</pre>	2	263	#ifdef NS3_MPI
183	}		264	<pre>// Exit the MPI execution environment</pre>
184			265	MpiInterface::Disable ();
185	Simulator::Run ();		266	#endif
186	Simulator::Destroy ();		267	
187	return 0;		268	return 0;
¥ 188	}		269	}



Script Output–Identical

\$./waf -run third

Waf: Entering directory `build/debug' Waf: Leaving directory `build/debug' 'build' finished successfully (2.152s) At time 2s client sent 1024 bytes to 10.1.2.4 port 9 At time 2.01796s server received 1024 bytes from 10.1.3.3 port 49153 At time 2.01796s server sent 1024 bytes to 10.1.3.3 port 49153 At time 2.03364s client received 1024 bytes from 10.1.2.4 port 9

\$./waf --run third-distributed \ --command-template="mpirun -n 2 %s --tracing"

Waf: Entering directory `build/debug' Waf: Leaving directory `build/debug' 'build' finished successfully (2.050s) At time 2s client sent 1024 bytes to 10.1.2.4 port 9 At time 2.01796s server received 1024 bytes from 10.1.3.3 port 49153 At time 2.01796s server sent 1024 bytes to 10.1.3.3 port 49153 At time 2.03364s client received 1024 bytes from 10.1.2.4 port 9



