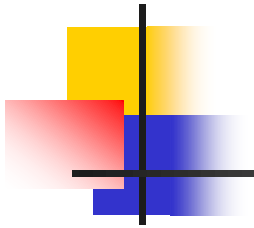
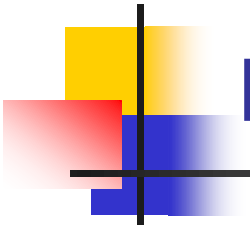


Final Project Presentation
Summer 2013

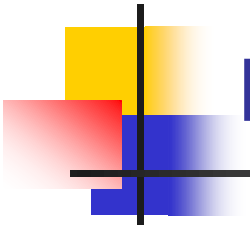
Ns-Modbus: Integration of Modbus with ns-3 network simulator

Mohammad Reza Sahraei
mrs16@sfu.ca



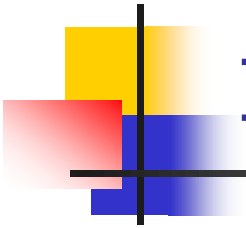
Road map

- Introduction
- Background knowledge
- Ns-Modbus design and implementation
- Verification and validation
- Model performance
- Future work
- Conclusion
- References



Road map

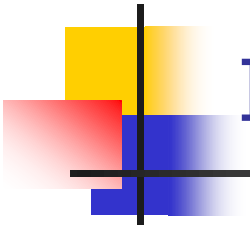
- Introduction
- Background knowledge
- Ns-Modbus design and implementation
- Verification and validation
- Model performance
- Future work
- Conclusion
- References



Introduction: project motivation

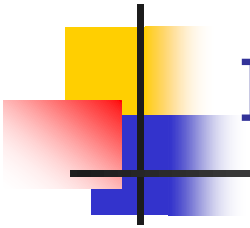
- EtherNet/IP then Modbus
- Learning ns-3 to assist CNL projects migration
- Graduation!

CNL: Communication Networks Laboratory



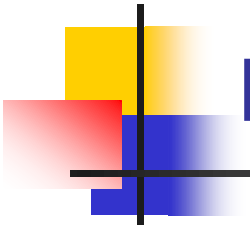
Introduction: project scope

- Understanding the Modbus protocol
- Gaining experience with ns-3
- Incorporating Modbus TCP in ns-3



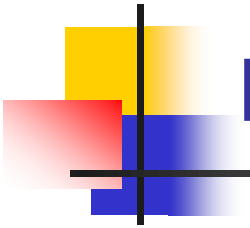
Introduction: project scope "creep"

- Incorporating Modbus UDP in ns-3
  - Modbus is based on client/server paradigm
  - Getting familiar with ns-3 using the simpler UDP implementation
  - Comparing TCP vs. UDP implementation



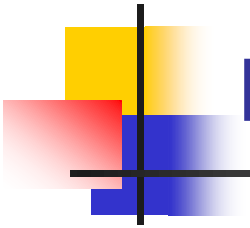
Road map

- Introduction
- Background knowledge
- Ns-Modbus design and implementation
- Verification and validation
- Model performance
- Future work
- Conclusion
- References



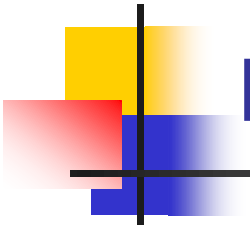
Background knowledge

- Modbus
- Network simulator



Background knowledge

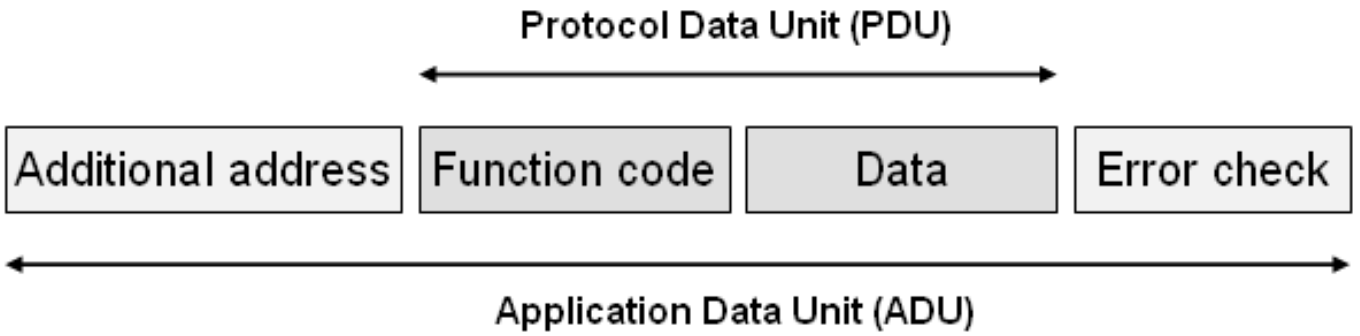
- Modbus
- Network simulator



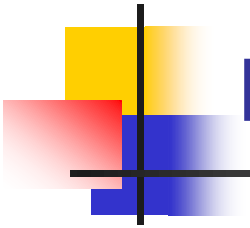
Modbus protocol overview

- Application layer protocol
- Request/response
- Exceptions (“status messages”)
- Timeout

Modbus protocol: frame



- Maximum ADU size: 256 bytes (RS 485)
- Maximum PDU size: 253 bytes

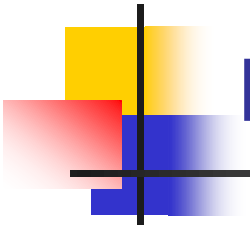


Modbus protocol: PDU type

- Modbus request
  - Function code (1 byte, 1:127)
  - Request data (variable length)
- Modbus response
  - Function code (echoed)
  - Response data (variable length)
- Modbus exception
  - Exception function code (129:255)
  - Exception code (1 byte)

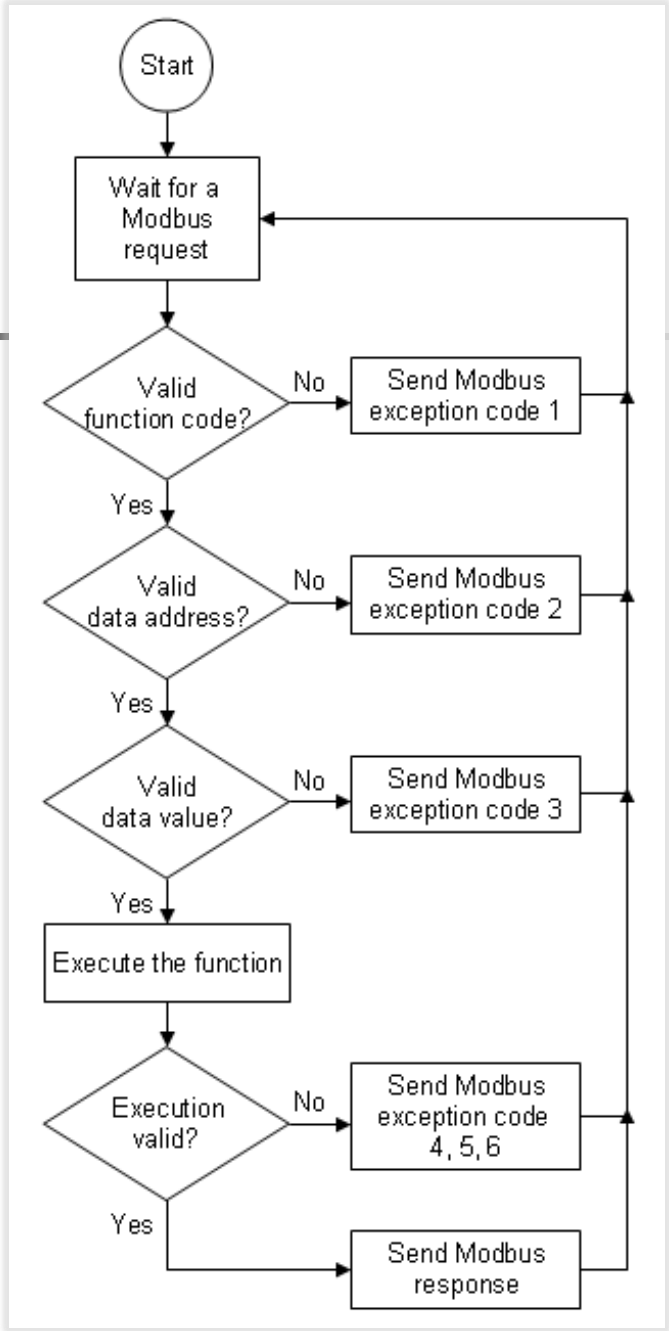
Modbus protocol: function code

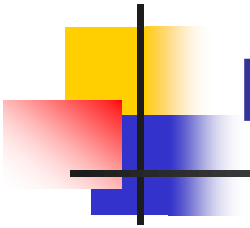
- Public
  - General purpose, well documented, conformance test
  - 1-64, 73-99, and 111-127
- User-defined
  - Vendor-specific, not in specification, functionality not unique
  - 65-72, and 100-110
- Reserved
  - Vendors' legacy applications, not public
  - 9, 10, 13, 14, 41, 42, 90, 91, 125, 126, and 127
  - sub-codes 0-12 (8) and 15-255 (43)



Modbus protocol: exception code

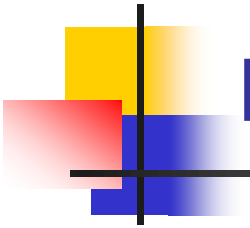
- 0x01: Illegal function
- 0x02: Illegal data address
- 0x03: Illegal data value
- 0x04: Device failure
- 0x05: Request in progress
- …


Start
Wait for a Modbus request
Valid function code?
No
Send Modbus exception code 1
Yes
Valid data address?
No
Send Modbus exception code 2
Yes
Valid data value?
No
Send Modbus exception code 3
Yes
Execute the function
Execution valid?
No
Send Modbus exception code 4, 5, 6
Yes
Send Modbus response



Modbus protocol: data model

- Data direction
  - Input
  - Output
- Data length
  - Bit addressable
  - Word addressable

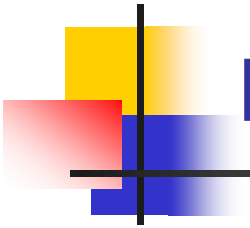


Modbus protocol: primary tables

- Discrete input: bit, read only, I/O system
- Coils: bit, read/write
- Input registers: 16-bit, read only, I/O system
- Holding registers: 16-bit, read/write

Background knowledge

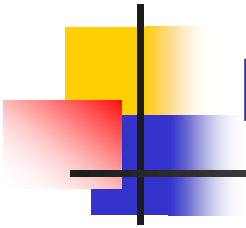
- Modbus
- Network simulator



Ns-3 overview

- Wired, wireless
- Routing algorithms
- Traffic generator: CBR, exponential, Pareto, trace file
- PCAP, trace file, log messages
- Visualization: NetAnim

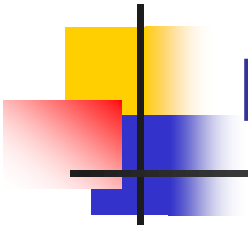
Ns: Network Simulator
CBR: Constant Bit Rate
PCAP: Packet CAPture
NetAnim: Network Animator



Ns-2 versus ns-3

- Ns-2
  - Code: C++ and OTcl
  - Script: OTcl
  - Nam
  - More models
- Ns-3
  - C++
  - C++ or Python
  - NetAnim
  - More support
  - Better design
  - Better performance

Nam: Network Animator

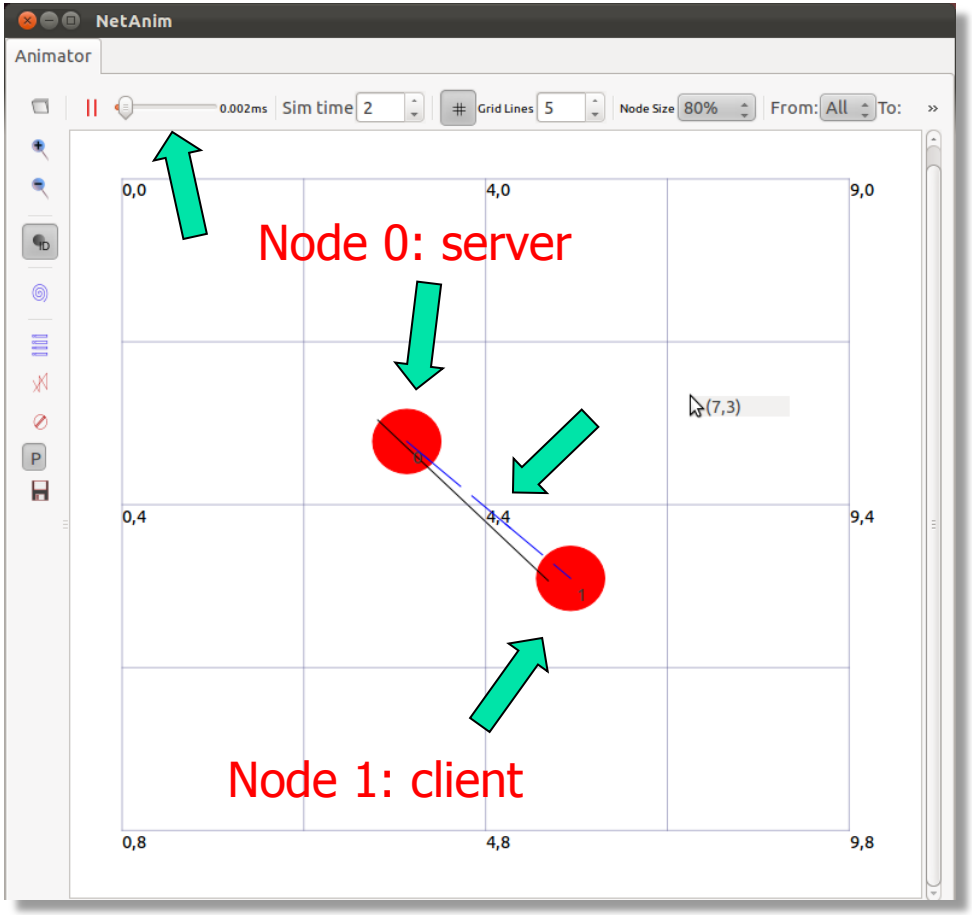


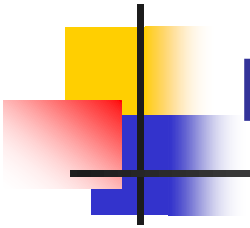
Ns-3 ecosystem

- Source control: Mercurial
  - Distributed
- Build system: Waf
  - Python-based
  - wscript
- Visualization: NetAnim
  - XML

Waf: The shortest and easily typed name, no particular meaning!

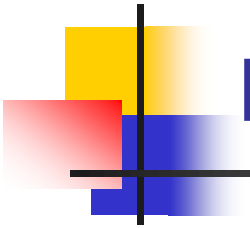
Ns-3: NetAnim


NetAnim
Animator
0.002ms
Sim time 2
Grid Lines 5
Node Size 80%
From: All
To:
0,0
4,0
9,0
0,4
4,4
9,4
0,8
4,8
9,8
(7,3)
Node 0: server
Node 1: client



Road map

- Introduction
- Background knowledge
- Ns-Modbus design and implementation
- Verification and validation
- Model performance
- Future work
- Conclusion
- References



Ns-Modbus classes

- Simulation scenario classes
  - Bus and star topologies
  - UDP and TCP
- Topology helper classes
  - Assist users with common tasks: assigning IP, defining nodes, arranging connections, setting application on each node
- Model classes
  - Core component, implements Modbus protocol specification

Ns-Modbus files

File Name	Relative Path	Action
myFirstModbusUdp.cc	~/ns-3-allinone/ns-3-dev/scratch/	Added
myFirstModbusTcp.cc	~/ns-3-allinone/ns-3-dev/scratch/	Added
busModbusUdp.cc	~/ns-3-allinone/ns-3-dev/scratch/	Added
busModbusTcp.cc	~/ns-3-allinone/ns-3-dev/scratch/	Added
busModbusTcpScenario1.cc	~/ns-3-allinone/ns-3-dev/scratch/	Added
busModbusTcpScenario2.cc	~/ns-3-allinone/ns-3-dev/scratch/	Added
busModbusTcpScenario3.cc	~/ns-3-allinone/ns-3-dev/scratch/	Added
starModbusUdp.cc	~/ns-3-allinone/ns-3-dev/scratch/	Added
starModbusTcp.cc	~/ns-3-allinone/ns-3-dev/scratch/	Added
wscript	~/ns-3-allinone/ns-3-dev/src/applications/	Modified
modbus-udp-helper.h	~/ns-3-allinone/ns-3-dev/src/applications/helper/	Added
modbus-udp-helper.cc	~/ns-3-allinone/ns-3-dev/src/applications/helper/	Added
modbus-tcp-helper.h	~/ns-3-allinone/ns-3-dev/src/applications/helper/	Added
modbus-tcp-helper.cc	~/ns-3-allinone/ns-3-dev/src/applications/helper/	Added
modbus-pdu.h	~/ns-3-allinone/ns-3-dev/src/applications/model/	Added
modbus-pdu.cc	~/ns-3-allinone/ns-3-dev/src/applications/model/	Added
modbus-mgr.h	~/ns-3-allinone/ns-3-dev/src/applications/model/	Added
modbus-mgr.cc	~/ns-3-allinone/ns-3-dev/src/applications/model/	Added
modbus-udp-client.h	~/ns-3-allinone/ns-3-dev/src/applications/model/	Added
modbus-udp-client.cc	~/ns-3-allinone/ns-3-dev/src/applications/model/	Added
modbus-udp-server.h	~/ns-3-allinone/ns-3-dev/src/applications/model/	Added
modbus-udp-server.cc	~/ns-3-allinone/ns-3-dev/src/applications/model/	Added
modbus-tcp-client.h	~/ns-3-allinone/ns-3-dev/src/applications/model/	Added
modbus-tcp-client.cc	~/ns-3-allinone/ns-3-dev/src/applications/model/	Added
modbus-tcp-server.h	~/ns-3-allinone/ns-3-dev/src/applications/model/	Added
modbus-tcp-server.cc	~/ns-3-allinone/ns-3-dev/src/applications/model/	Added

Ns-Modbus function codes

Function Code	Sub Code	Description	Comments
02		Read Discrete Inputs	Physical Discrete Inputs (bit access)
01		Read Coils	Internal bits or Physical Coils (bit access)
05		Write Single Coil	Internal bits or Physical Coils (bit access)
15		Write Multiple Coils	Internal bits or Physical Coils (bit access)
04		Read Input Register	Physical Input Registers (16-bit access)
03		Read Holding Registers	Internal Registers or Physical Output Registers (16-bit access)
06		Write Signle Register	Internal Registers or Physical Output Registers (16-bit access)
16		Write Multiple Registers	Internal Registers or Physical Output Registers (16-bit access)
23		Read/Write Multiple Registers	Internal Registers or Physical Output Registers (16-bit access)
22		Mask Write Register	Internal Registers or Physical Output Registers (16-bit access)
24		Read FIFO queue	Internal Registers or Physical Output Registers (16-bit access)
20		Read File Record	File Record access
21		Write File Record	File Record access
07		Read Exception Status	Diagnostics
08	0-18, 20	Diagnostic	Diagnostics
11		Get Com Event Counter	Diagnostics
12		Get Com Event Log	Diagnostics
17		Report Slave ID	Diagnostics
43	14	Read Device Identification	Diagnostics
43	13, 14	Encapsulated Interface Transport	Other
43	13	CANopen General Reference	Other

Ns-Modbus sample script

```
49  // MODBUS requests
50  uint8_t ENCAP_INTERFACE_TRANS_CANOPEN_GENERAL_REF_REQ_RSP_PDU[] = {0x2B, 0x0D};
51  uint8_t ENCAP_INTERFACE_TRANS_READ_DEVICE_ID__BASIC_DEVICE_ID_OBJ_ID_00[] = {0x2
52  uint8_t ENCAP_INTERFACE_TRANS_READ_DEVICE_ID__REGULAR_DEVICE_ID_OBJ_ID_00[] = {0
53  uint8_t ENCAP_INTERFACE_TRANS_READ_DEVICE_ID__EXTENDED_DEVICE_ID_OBJ_ID_00[] = {
54  uint8_t ENCAP_INTERFACE_TRANS_READ_DEVICE_ID__SPECIFIC_DEVICE_ID_OBJ_ID_00[] = {
55
56
57  using namespace ns3;
58
59  NS_LOG_COMPONENT_DEFINE ("BusModbusTcp");
60
61  int
62  main (int argc, char *argv[])
```

Predefined Modbus Requests

Ns-Modbus sample script

```
94  NS_LOG_INFO ("Build bus topology.");
95  NodeContainer csmaNodes;
96  csmaNodes.Create (nCsma);
97
98  CsmaHelper csma;
99  csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
100 csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));
101
102 NetDeviceContainer csmaDevices;
103 csmaDevices = csma.Install (csmaNodes);
104
105 NS_LOG_INFO ("Install internet stack on all nodes.");
106 InternetStackHelper internet;
107 internet.Install (csmaNodes);
108
109
110 NS_LOG_INFO ("Assign IP Addresses.");
111 Ipv4AddressHelper address;
112 address.SetBase ("10.1.1.0", "255.255.255.0");
113 Ipv4InterfaceContainer csmaInterfaces;
114 csmaInterfaces = address.Assign (csmaDevices);
115
116 NS_LOG_INFO ("Create applications.");
117 //
118 // Create a MODBUS TCP server on the star "hub"
119 //
120 ModbusTcpServerHelper cModbusTcpServerHelper;
121 ApplicationContainer serverApps = cModbusTcpServerHelper.Install (csmaNodes.Ge
122 serverApps.Start (Seconds (1.0));
123 serverApps.Stop (Seconds (10.0));
```

Server Node's Start/Stop Time

Ns-Modbus sample script

```
125   //
126   // Create MODBUS TCP client applications send UDP to the hub
127   //
128   for (uint32_t i = 1; i < nCsma; ++i)
129     {
130       ModbusTcpClientHelper cModbusTcpClientHelper (csmaInterfaces.GetAddress (0
131       cModbusTcpClientHelper.SetAttribute ("MaxPackets", UintegerValue (1));
132       cModbusTcpClientHelper.SetAttribute ("Interval", TimeValue (Seconds (1.0))
133
134       ApplicationContainer clientApps = cModbusTcpClientHelper.Install (csmaNode
135
136       // set MODBUS requests
137       cModbusTcpClientHelper.SetRequest (clientApps.Get(0), Seconds (0.01),
138         ENCAP_INTERFACE_TRANS_CANOPEN_GENERAL_REF_REQ_RSP_PDU);
139
140       cModbusTcpClientHelper.SetRequest (clientApps.Get(0), Seconds (0.2),
141         ENCAP_INTERFACE_TRANS_READ_DEVICE_ID__BASIC_DEVICE_ID_OBJ_ID_00);
142
143       cModbusTcpClientHelper.SetRequest (clientApps.Get(0), Seconds (0.1),
144         ENCAP_INTERFACE_TRANS_READ_DEVICE_ID__SPECIFIC_DEVICE_ID_OBJ_ID_00);
145
146       cModbusTcpClientHelper.SetCount (clientApps.Get(0), 2);
147
148       clientApps.Start (Seconds (2.0));
149       clientApps.Stop (Seconds (10.0));
150     }
```

Setting a Request with a Start Delay

Request Repetition

Client Node's Start/Stop Time

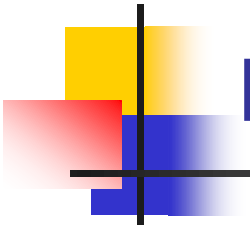
Ns-Modbus wscript

```
## -*- Mode: python; py-indent-offset: 4; indent-tabs-mode: nil; coding: utf-8; -*

def build(bld):
    module = bld.create_ns3_module('applications', ['internet', 'config-store', 't
    module.source = [
        'model/bulk-send-application.cc',
        'model/onoff-application.cc',
        'model/packet-sink.cc',
        'model/ping6.cc',
        'model/radvd.cc',
        'model/radvd-interface.cc',
        'model/radvd-prefix.cc',
        'model/udp-client.cc',
        'model/udp-server.cc',
        'model/seq-ts-header.cc',
        'model/udp-trace-client.cc',
        'model/packet-loss-counter.cc',
        'model/udp-echo-client.cc',
        'model/udp-echo-server.cc',
        'model/v4ping.cc',
        'model/modbus-udp-client.cc',
        'model/modbus-udp-server.cc',
        'model/modbus-tcp-client.cc',
        'model/modbus-tcp-server.cc',
        'model/modbus-pdu.cc',
        'model/modbus-mgr.cc',
        'model/bgp-router.cc',
        'model/bgp-pdu.cc',
        'helper/bulk-send-helper.cc',
        'helper/on-off-helper.cc',
        'helper/packet-sink-helper.cc',
        'helper/ping6-helper.cc',
        'helper/udp-client-server-helper.cc',
        'helper/udp-echo-helper.cc',
        'helper/v4ping-helper.cc',
        'helper/modbus-udp-helper.cc',
        'helper/modbus-tcp-helper.cc',
        'helper/bgp-router-helper.cc',
```

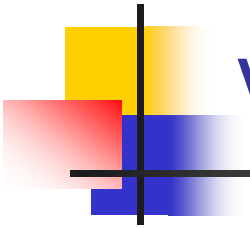
Ns-Modbus wscript

```
48      headers.source = [
49          'model/bulk-send-application.h',
50          'model/onoff-application.h',
51          'model/packet-sink.h',
52          'model/ping6.h',
53          'model/radvd.h',
54          'model/radvd-interface.h',
55          'model/radvd-prefix.h',
56          'model/udp-client.h',
57          'model/udp-server.h',
58          'model/seq-ts-header.h',
59          'model/udp-trace-client.h',
60          'model/packet-loss-counter.h',
61          'model/udp-echo-client.h',
62          'model/udp-echo-server.h',
63          'model/v4ping.h',
64          'model/modbus-udp-client.h',
65          'model/modbus-udp-server.h',
66          'model/modbus-tcp-client.h',
67          'model/modbus-tcp-server.h',
68          'model/modbus-pdu.h',
69          'model/modbus-mgr.h',
70          'model/bgp-router.h',
71          'model/bgp-pdu.h',
72          'helper/bulk-send-helper.h',
73          'helper/on-off-helper.h',
74          'helper/packet-sink-helper.h',
75          'helper/ping6-helper.h',
76          'helper/udp-client-server-helper.h',
77          'helper/udp-echo-helper.h',
78          'helper/v4ping-helper.h',
79          'helper/modbus-udp-helper.h',
80          'helper/modbus-tcp-helper.h',
81          'helper/bgp-router-helper.h',
82          ]
83
84      bld.ns3_python_bindings()
```



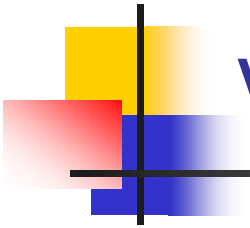
Road map

- Introduction
- Background knowledge
- Ns-Modbus design and implementation
- Verification and validation
- Model performance
- Future work
- Conclusion
- References



Verification and validation: test cases

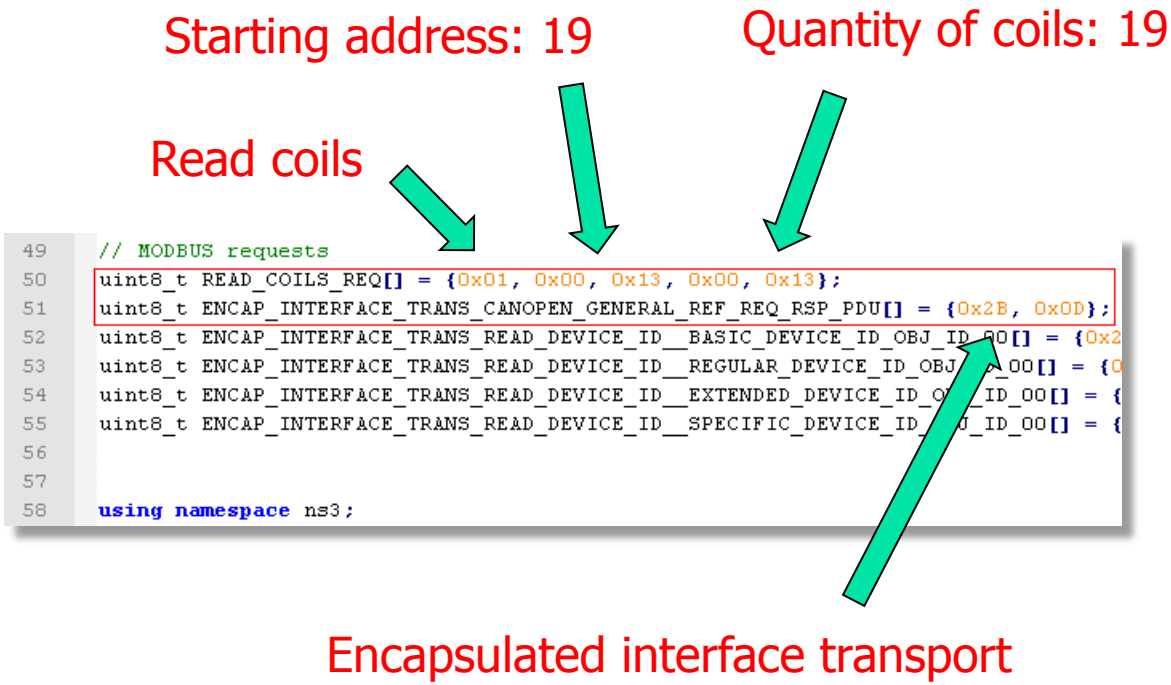
- Sending request and receiving response
  - Encapsulated interface transport
  - Read coils
- Writing to and reading from primary table
  - Write multiple coils
  - Read coils
- Sending Modbus request with illegal data
  - Undefined function code
  - Invalid data address
  - Invalid data value



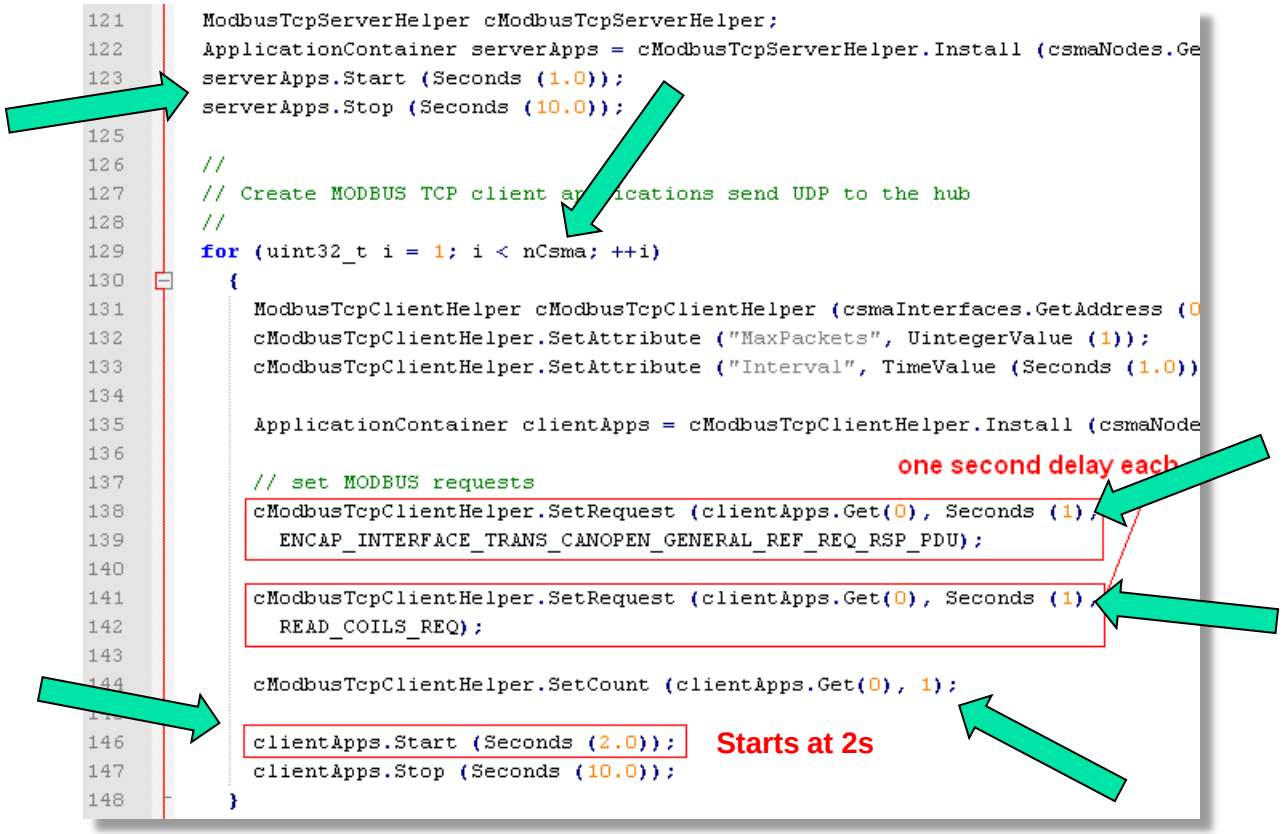
Verification and validation: test cases

- Sending request and receiving response
  - Encapsulated interface transport
  - Read coils
- Writing to and reading from primary table
  - Write multiple coils
  - Read coils
- Sending Modbus request with illegal data
  - Undefined function code
  - Invalid data address
  - Invalid data value

Sending request and receiving response test case: function codes



Sending request and receiving response test case: scheduled requests



Sending request and receiving response test case: verbose message logs

```
1  reza@Dena:~/temp/ns-3-allinone/ns-3-dev$ export NS_LOG=BusModbusTcpScenario1=level
2  reza@Dena:~/temp/ns-3-allinone/ns-3-dev$ ./waf --run "scratch/busModbusTcpScenario
3  Waf: Entering directory `/home/reza/temp/ns-3-allinone/ns-3-dev/build'
4  Waf: Leaving directory `/home/reza/temp/ns-3-allinone/ns-3-dev/build'
5  'build' finished successfully (1.694s)
6  Build bus topology.
7  Install internet stack on all nodes.
8  Assign IP Addresses.
9  Create applications.
10 ModbusTcpClient::SetRequest ()
11 ModbusTcpClient::SetRequest ()
12 ModbusTcpClient::SetCount ()
13 Enable pcap tracing.
14 Run Simulation.
15 ModbusTcpClient::Send ()
16 TraceDelay TX: 253 bytes to 10.1.1.1 Uid: 14 Time: 3
17 Parsing MODBUS TCP function code
18 Encapsulated Interface Transport
19 meiType: 0xd
20 CANopen General Reference Request and Response PDU
21 TraceDelay: RX 253 bytes from 10.1.1.2 Sequence Number: 0 Uid: 14 TXtime: +3000000
22 Sending MODBUS TCP response packet
23 ModbusTcpClient::HandleRead ()
24 Received 253 bytes from 10.1.1.1
25 Success - function code: 0x2b
26 pdu[0]: 0x2b
27 pdu[1]: 0xd
28 pdu[2]: 0x2b
29 pdu[3]: 0xe
30 pdu[4]: 0x1
31 pdu[5]: 0x0
32 pdu[6]: 0x2b
33 pdu[7]: 0xe
34 pdu[8]: 0x2
35 pdu[9]: 0x0
```

Client sends the request at 3s

Server parses the request

Client receives the response

Sending request and receiving response test case: verbose message logs

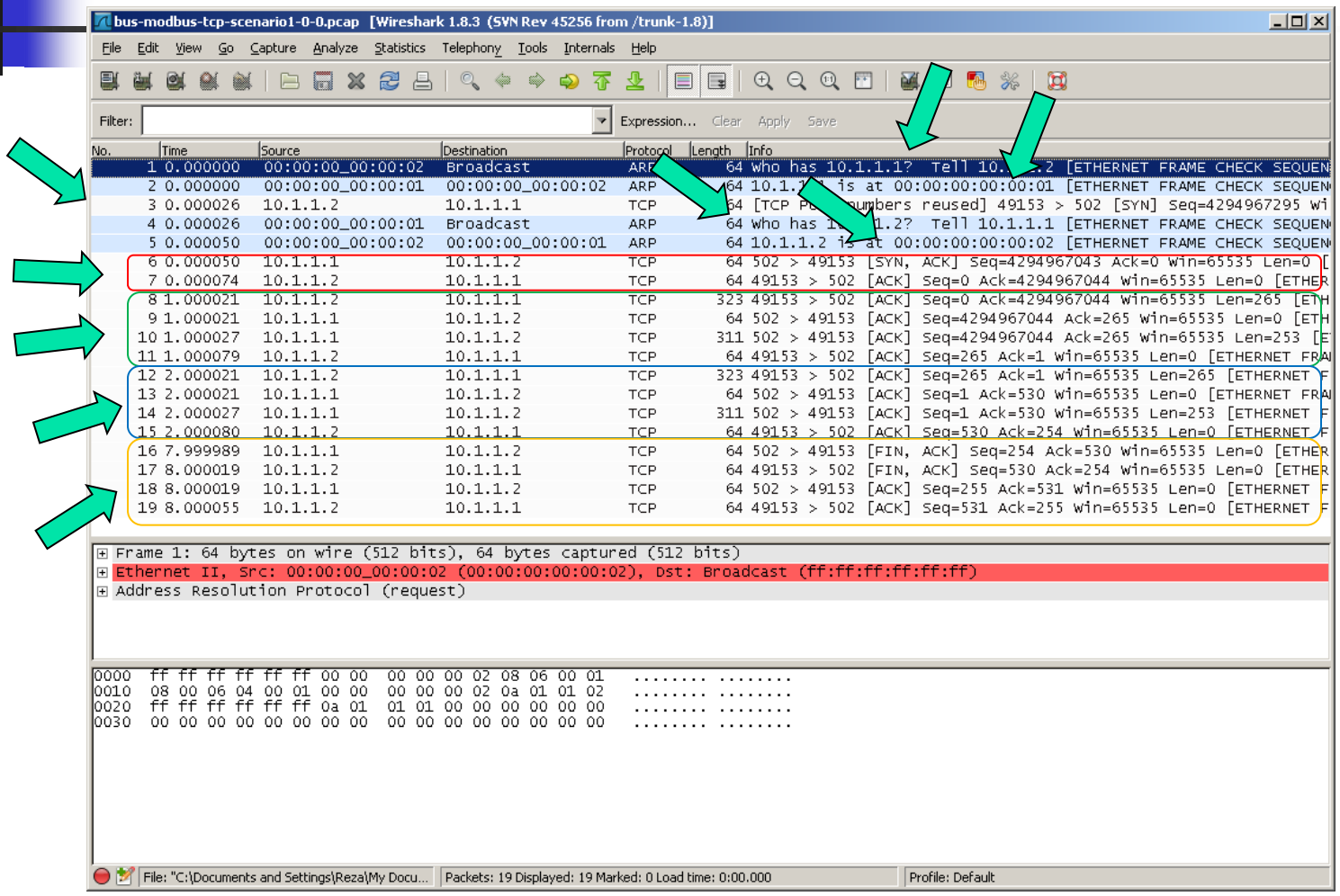
```
37  ModbusTcpClient::Send ()
38  TraceDelay TX: 253 bytes to 10.1.1.1 Uid: 22 Time: 4
39  Parsing MODBUS TCP function code
40  Read Coils
41  startAdr: 19
42  quantityOfCoils: 19
43  TraceDelay: RX 253 bytes from 10.1.1.2 Sequence Number: 1 Uid: 22 TXtime: +4000000
44  Sending MODBUS TCP response packet
45  ModbusTcpClient::HandleRead ()
46  Received 253 bytes from 10.1.1.1
47  Success - function code: 0x1
48  pdu[0]: 0x1
49  pdu[1]: 0x3
50  pdu[2]: 0xba
51  pdu[3]: 0x1
52  pdu[4]: 0x28
53  pdu[5]: 0x34
54  pdu[6]: 0x2
55  pdu[7]: 0x1
56  pdu[8]: 0x1
57  pdu[9]: 0x0
58
59  ModbusTcpServer, peerClose
60  ModbusTcpServer, peerClose
61  Done.
62  Total (ms): 111.01
63  Modbus Simulation time (ms): 97.7521
64  Node and Link creation (ms): 13.258
65  reza@Dena:~/temp/ns-3-allinone/ns-3-dev$
```

Client sends the request at 4s

Server parses the request

Client receives the response

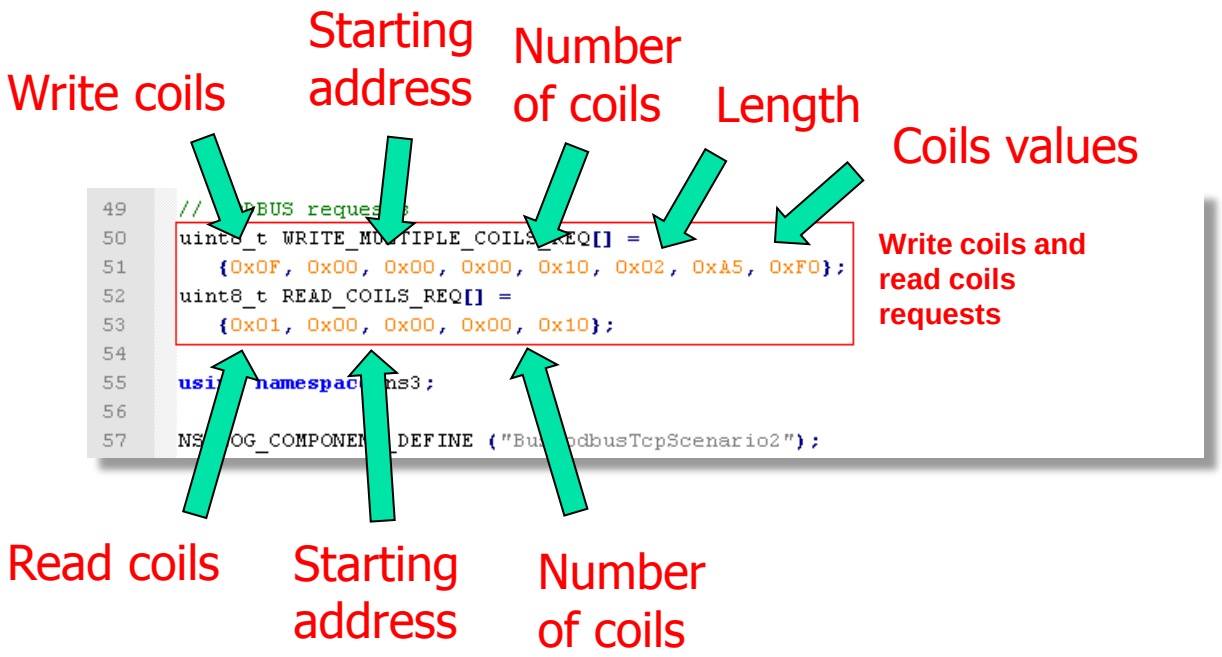
Sending request and receiving response test case: Wireshark capture



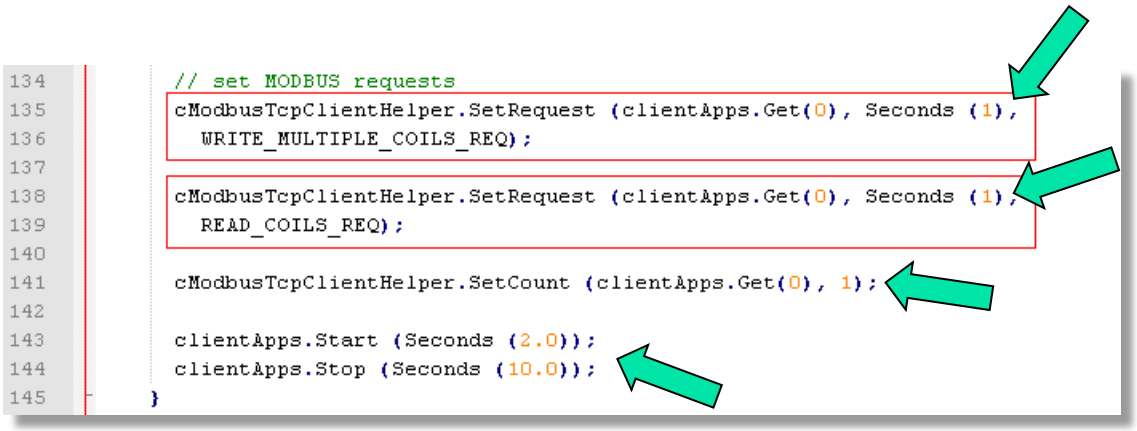
Verification and validation: test cases

- Sending request and receiving response
  - Encapsulated interface transport
  - Read coils
- Writing to and reading from primary table
  - Write multiple coils
  - Read coils
- Sending Modbus request with illegal data
  - Undefined function code
  - Invalid data address
  - Invalid data value

Writing to and reading from primary table test case: function codes



Writing to and reading from primary table test case: scheduled requests



```
134     // set MODBUS requests
135     cModbusTcpClientHelper.SetRequest (clientApps.Get(0), Seconds (1),
136       WRITE_MULTIPLE_COILS_REQ);
137
138     cModbusTcpClientHelper.SetRequest (clientApps.Get(0), Seconds (1),
139       READ_COILS_REQ);
140
141     cModbusTcpClientHelper.SetCount (clientApps.Get(0), 1);
142
143     clientApps.Start (Seconds (2.0));
144     clientApps.Stop (Seconds (10.0));
145   }
```

Writing to and reading from primary table test case: verbose message logs

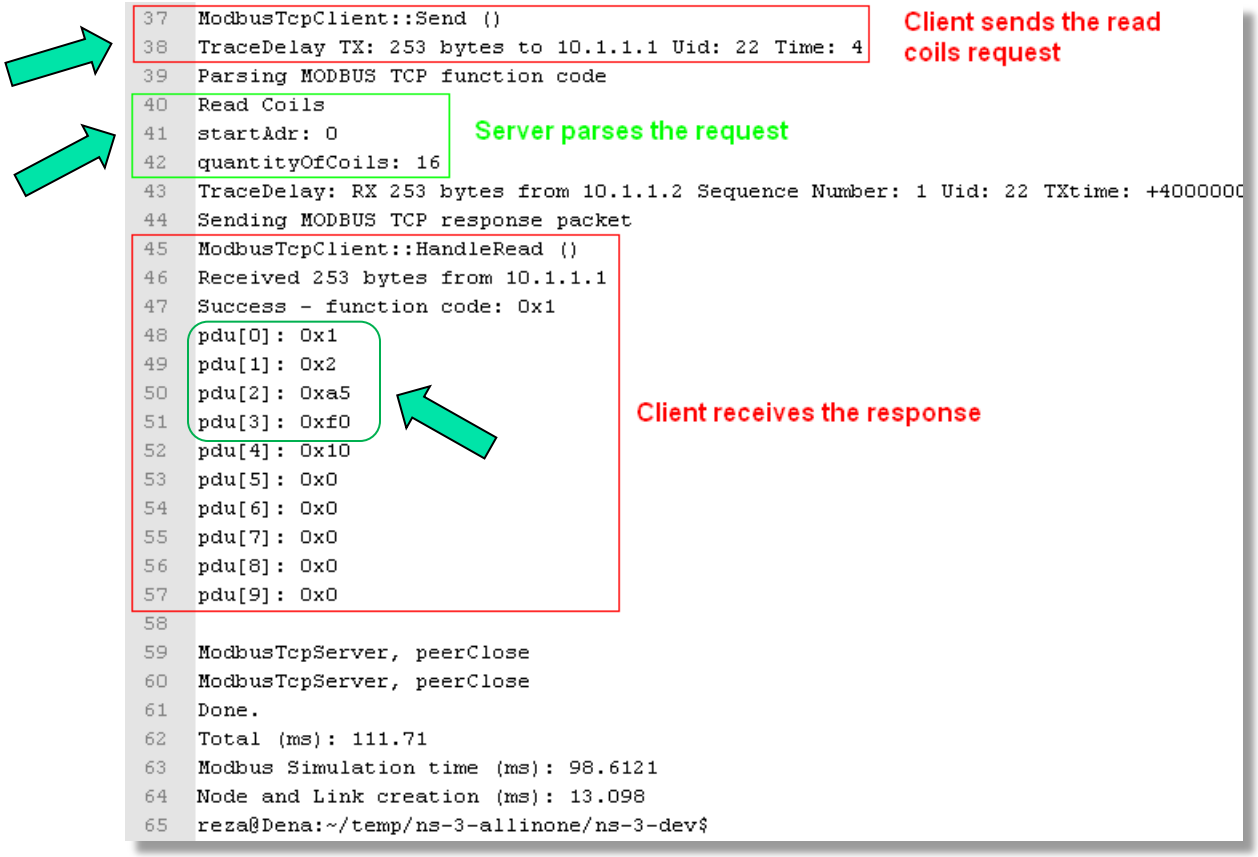
```
1  reza@Dena:~/temp/ns-3-allinone/ns-3-dev$ ./waf --run "scratch/busModbusTcpScenario
2  Waf: Entering directory `/home/reza/temp/ns-3-allinone/ns-3-dev/build'
3  Waf: Leaving directory `/home/reza/temp/ns-3-allinone/ns-3-dev/build'
4  'build' finished successfully (1.692s)
5  Build bus topology.
6  Install internet stack on all nodes.
7  Assign IP Addresses.
8  Create applications.
9  ModbusTcpClient::SetRequest ()
10 ModbusTcpClient::SetRequest ()
11 ModbusTcpClient::SetCount ()
12 Enable pcap tracing.
13 Run Simulation.
14 ModbusTcpClient::Send ()
15 TraceDelay TX: 253 bytes to 10.1.1.1 Uid: 14 Time: 3
16 Parsing MODBUS TCP function code
17 Write Multiple Coils
18 startAdr: 0
19 quantityOfOutputs: 16
20 byteCount: 2
21 TraceDelay: RX 253 bytes from 10.1.1.2 Sequence Number: 0 Uid: 14 TXtime: +3000000
22 Sending MODBUS TCP response packet
23 ModbusTcpClient::HandleRead ()
24 Received 253 bytes from 10.1.1.1
25 Success - function code: 0xf
26 pdu[0]: 0xf
27 pdu[1]: 0x0
28 pdu[2]: 0x0
29 pdu[3]: 0x0
30 pdu[4]: 0x10
31 pdu[5]: 0x0
32 pdu[6]: 0x0
33 pdu[7]: 0x0
34 pdu[8]: 0x0
35 pdu[9]: 0x0
```

Client sends the write multiple coils request

Server parses the request

Client receives the response

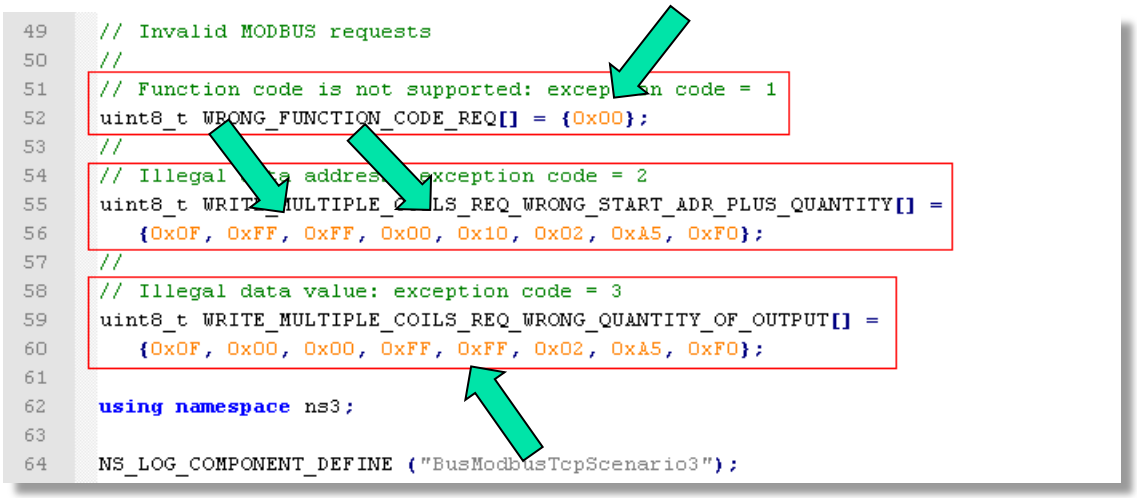
Writing to and reading from primary table test case: verbose message logs



Verification and validation: test cases

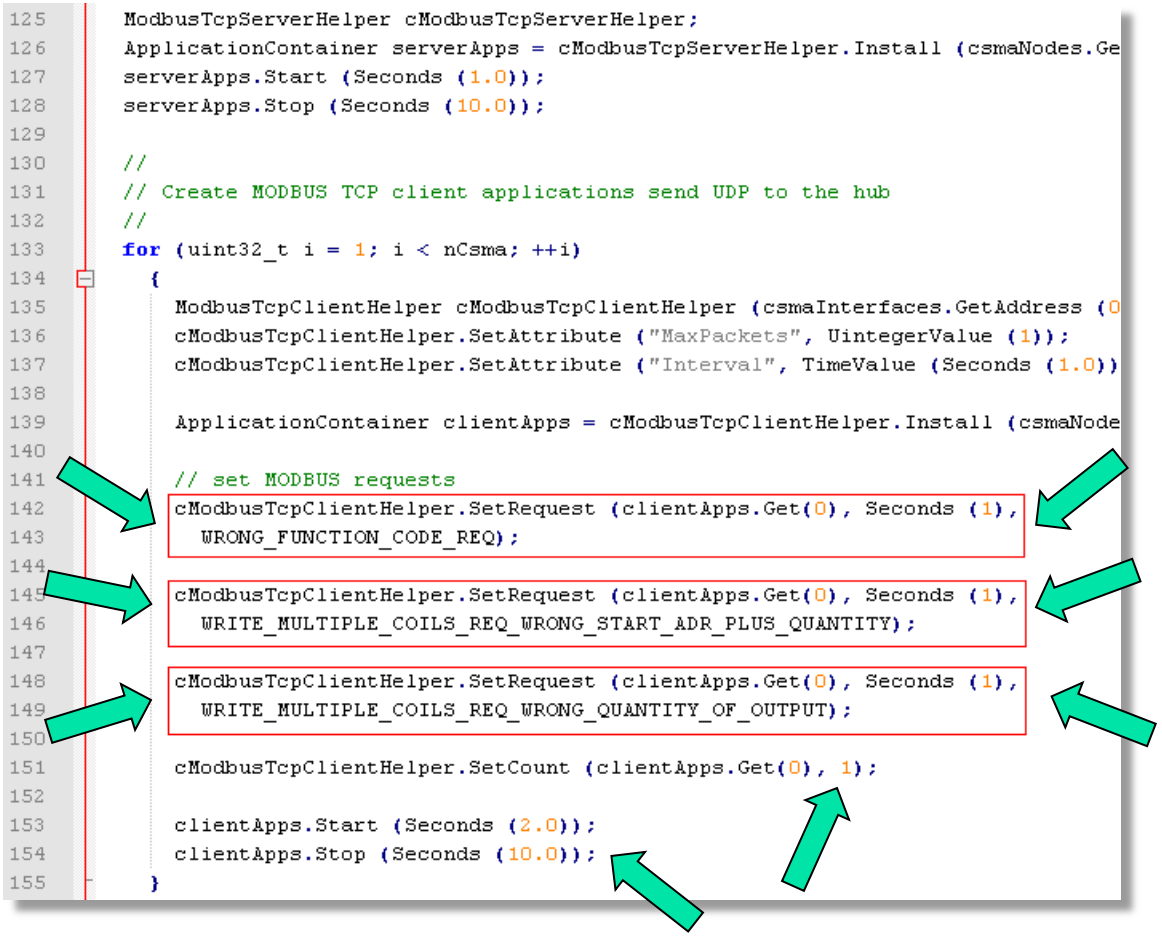
- Sending request and receiving response
  - Encapsulated interface transport
  - Read coils
- Writing to and reading from primary table
  - Write multiple coils
  - Read coils
- Sending Modbus request with illegal data
  - Undefined function code
  - Invalid data address
  - Invalid data value

Sending Modbus request with illegal data test case: function codes



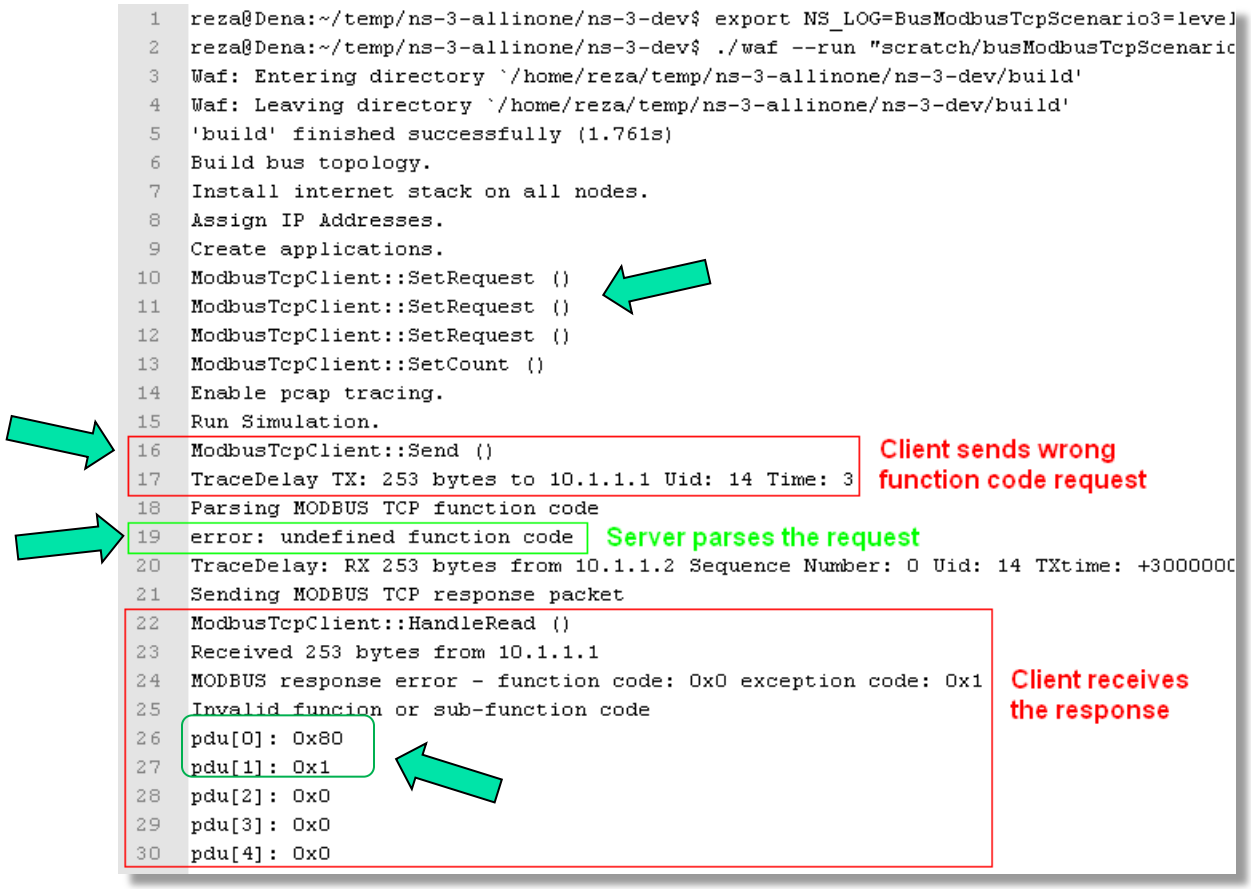
```
49  // Invalid MODBUS requests
50  //
51  // Function code is not supported: exception code = 1
52  uint8_t WRONG_FUNCTION_CODE_REQ[] = {0x00};
53  //
54  // Illegal data address: exception code = 2
55  uint8_t WRITE_MULTIPLE_COILS_REQ_WRONG_START_ADR_PLUS_QUANTITY[] =
56     {0x0F, 0xFF, 0xFF, 0x00, 0x10, 0x02, 0xA5, 0xF0};
57  //
58  // Illegal data value: exception code = 3
59  uint8_t WRITE_MULTIPLE_COILS_REQ_WRONG_QUANTITY_OF_OUTPUT[] =
60     {0x0F, 0x00, 0x00, 0xFF, 0xFF, 0x02, 0xA5, 0xF0};
61
62  using namespace ns3;
63
64  NS_LOG_COMPONENT_DEFINE ("BusModbusTcpScenario3");
```

Sending Modbus request with illegal data test case: scheduled requests

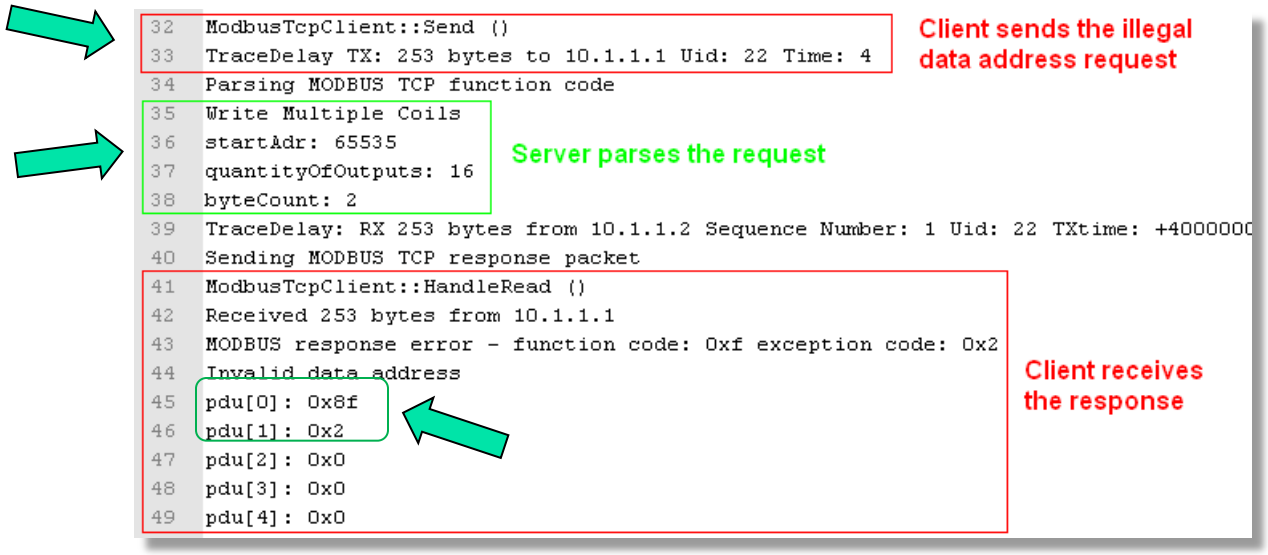


```
125   ModbusTcpServerHelper cModbusTcpServerHelper;
126   ApplicationContainer serverApps = cModbusTcpServerHelper.Install (csmaNodes.Ge
127   serverApps.Start (Seconds (1.0));
128   serverApps.Stop (Seconds (10.0));
129
130   //
131   // Create MODBUS TCP client applications send UDP to the hub
132   //
133   for (uint32_t i = 1; i < nCsma; ++i)
134     {
135       ModbusTcpClientHelper cModbusTcpClientHelper (csmaInterfaces.GetAddress (0
136       cModbusTcpClientHelper.SetAttribute ("MaxPackets", UintegerValue (1));
137       cModbusTcpClientHelper.SetAttribute ("Interval", TimeValue (Seconds (1.0))
138
139       ApplicationContainer clientApps = cModbusTcpClientHelper.Install (csmaNode
140
141       // set MODBUS requests
142       cModbusTcpClientHelper.SetRequest (clientApps.Get(0), Seconds (1),
143         WRONG_FUNCTION_CODE_REQ);
144
145       cModbusTcpClientHelper.SetRequest (clientApps.Get(0), Seconds (1),
146         WRITE_MULTIPLE_COILS_REQ_WRONG_START_ADR_PLUS_QUANTITY);
147
148       cModbusTcpClientHelper.SetRequest (clientApps.Get(0), Seconds (1),
149         WRITE_MULTIPLE_COILS_REQ_WRONG_QUANTITY_OF_OUTPUT);
150
151       cModbusTcpClientHelper.SetCount (clientApps.Get(0), 1);
152
153       clientApps.Start (Seconds (2.0));
154       clientApps.Stop (Seconds (10.0));
155     }
```

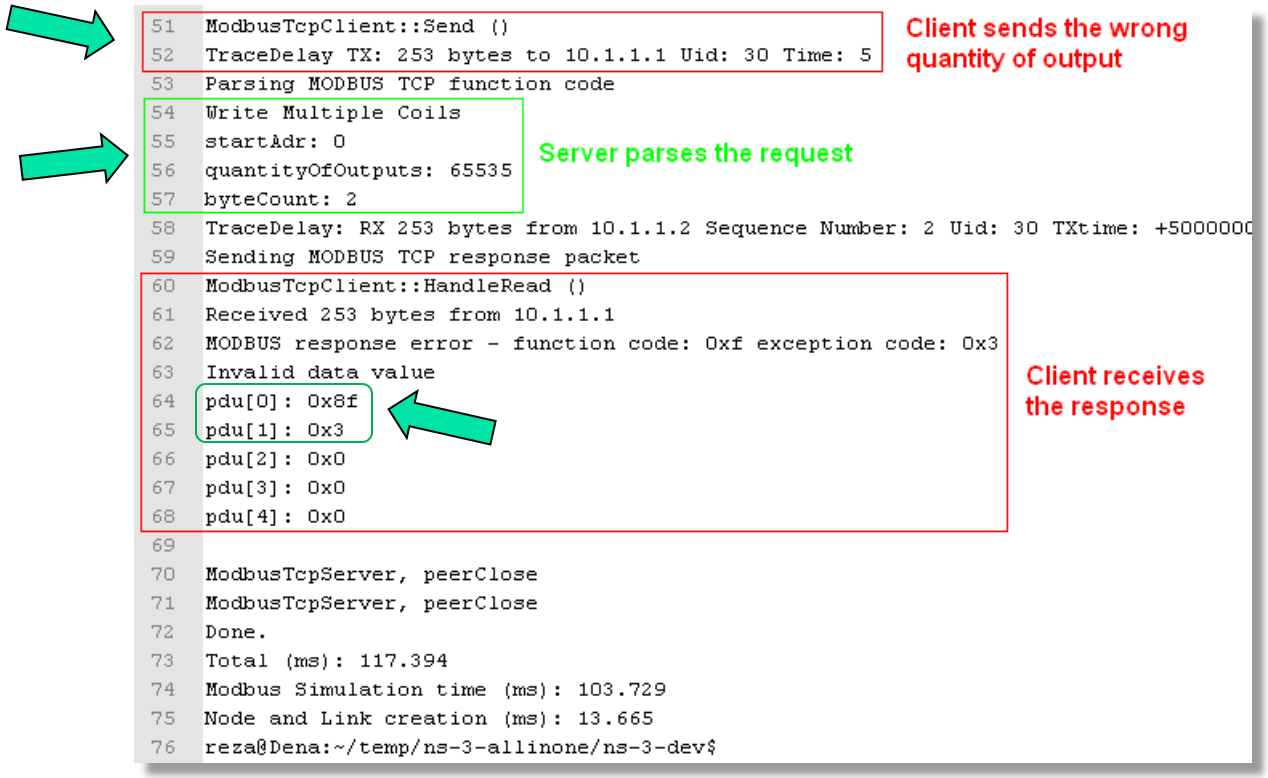
Sending Modbus request with illegal data test case: verbose message logs



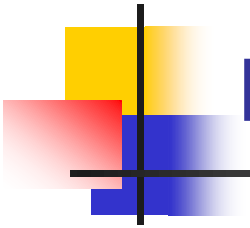
Sending Modbus request with illegal data test case: verbose message logs



Sending Modbus request with illegal data test case: verbose message logs

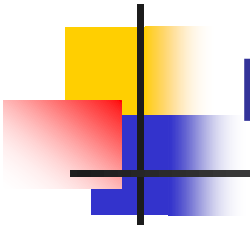



```
51  ModbusTcpClient::Send ()
52  TraceDelay TX: 253 bytes to 10.1.1.1 Uid: 30 Time: 5
53  Parsing MODBUS TCP function code
54  Write Multiple Coils
55  startAdr: 0
56  quantityOfOutputs: 65535
57  byteCount: 2
58  TraceDelay: RX 253 bytes from 10.1.1.2 Sequence Number: 2 Uid: 30 TXtime: +5000000
59  Sending MODBUS TCP response packet
60  ModbusTcpClient::HandleRead ()
61  Received 253 bytes from 10.1.1.1
62  MODBUS response error - function code: 0xf exception code: 0x3
63  Invalid data value
64  pdu[0]: 0x8f
65  pdu[1]: 0x3
66  pdu[2]: 0x0
67  pdu[3]: 0x0
68  pdu[4]: 0x0
69
70  ModbusTcpServer, peerClose
71  ModbusTcpServer, peerClose
72  Done.
73  Total (ms): 117.394
74  Modbus Simulation time (ms): 103.729
75  Node and Link creation (ms): 13.665
76  reza@Dena:~/temp/ns-3-allinone/ns-3-dev$
```



Road map

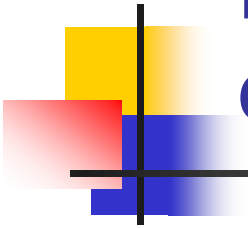
- Introduction
- Background knowledge
- Ns-Modbus design and implementation
- Verification and validation
- Model performance
- Future work
- Conclusion
- References



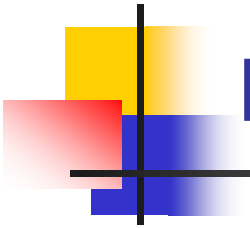
Model performance

- Application profiling
  - Model scalability: Bus topology and star topology
  - Runtime overhead
- Simulation robustness

Model performance: simulation environment



- Dell Inspiron
  - Intel Pentium® Dual-Core CPU T4200 @ 2.00 GHz
  - 3 GB DIMM 800 MHz DDR, 4.5 GB Swap
  - Hitachi 250 GB, Dual Port Cache, 5400 RPM HDD
- Ubuntu 11.10, with Linux 3.0.0-12-generic
- GCC (Ubuntu/Linaro 4.6.1-9ubuntu3) 4.6.1
- Ns-3.13 (released Dec 2011)



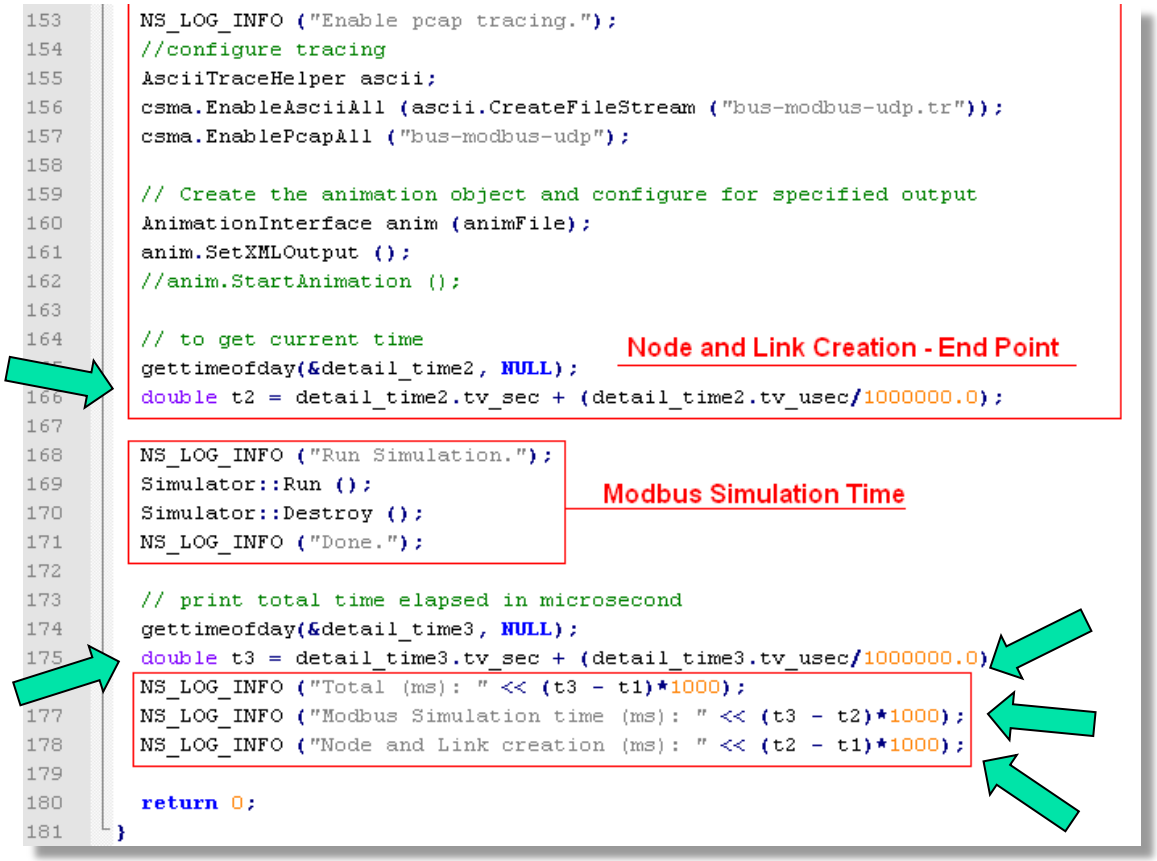
Model performance: application profiling

- Node and link creation time
- Modbus simulation time
- Round-trip Modbus request/response time
- Total time

Model performance: application profiling probe locations

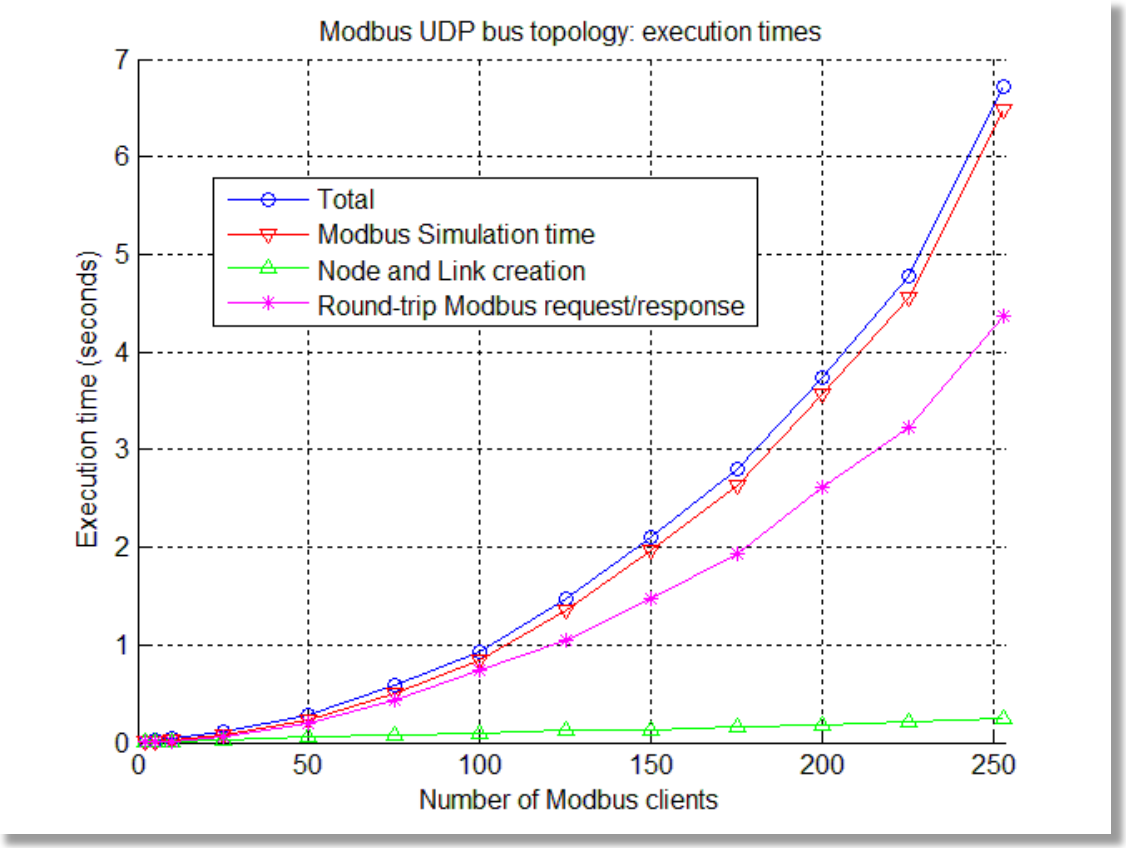
```
60  NS_LOG_COMPONENT_DEFINE ("BusModbusUdp");
61
62  int
63  main (int argc, char *argv[])
64  {
65    struct timeval detail_time1, detail_time2, detail_time3;
66
67    // to get current time
68    gettimeofday(&detail_time1, NULL);
69    double t1 = detail_time1.tv_sec + (detail_time1.tv_usec/1000000.0);
                                        Node and Link Creation - Start Point
71    bool verbose = false;
72    std::string animFile = "bus-modbus-udp-animfile.xml";
73
74    //
75    // Default number of nodes.  Overridable by command line argument.
76    //
77    uint32_t nCsma = 4;
78
79    // under ns-3-dev type ./waf --run 'scratch/myfirst --PrintHelp'
80    // to see a list of parameters can be passed to the parser
81    CommandLine cmd;
82    cmd.AddValue ("nCsma", "Number of CSMA nodes/devices", nCsma);
83    cmd.AddValue ("verbose", "Tell MODBUS applications to log if true", verbose);
84    cmd.AddValue ("animFile",  "File Name for Animation Output", animFile);
85    cmd.Parse (argc, argv);
```

Model performance: application profiling probe locations

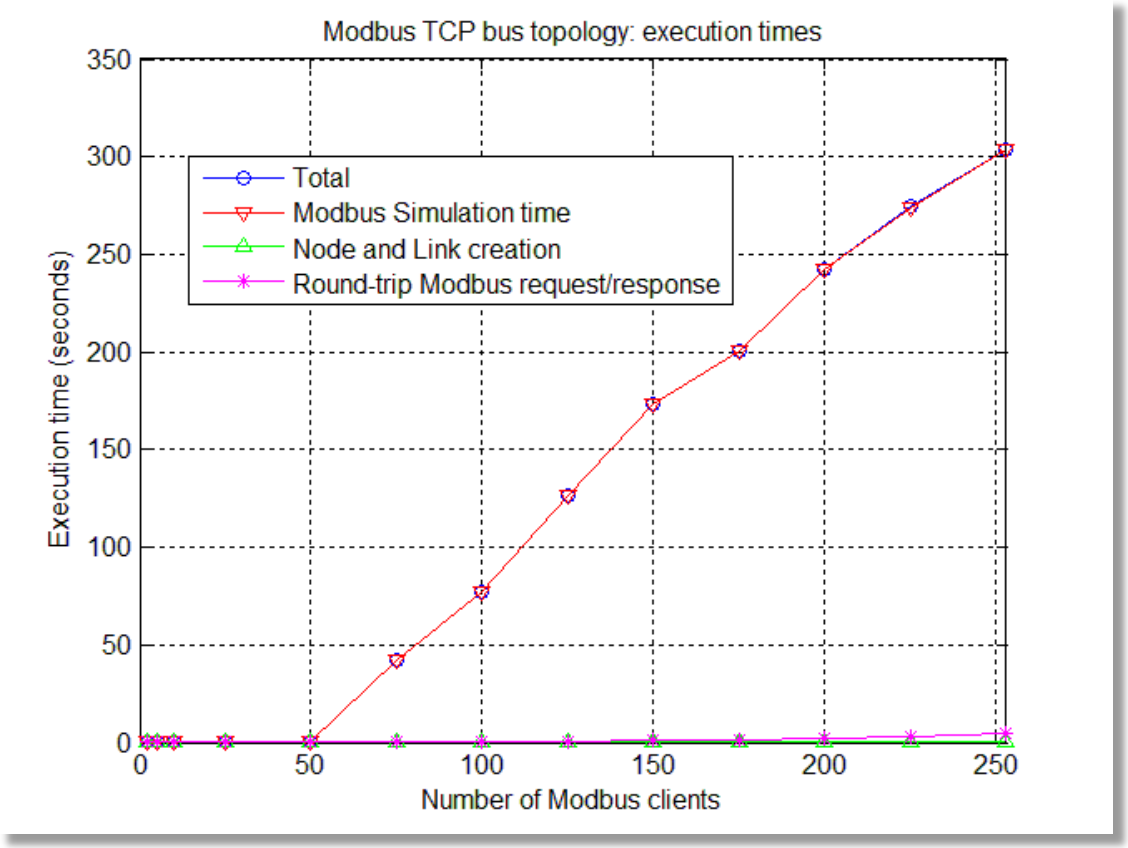



```
153   NS_LOG_INFO ("Enable pcap tracing.");
154   //configure tracing
155   AsciiTraceHelper ascii;
156   csma.EnableAsciiAll (ascii.CreateFileStream ("bus-modbus-udp.tr"));
157   csma.EnablePcapAll ("bus-modbus-udp");
158
159   // Create the animation object and configure for specified output
160   AnimationInterface anim (animFile);
161   anim.SetXMLOutput ();
162   //anim.StartAnimation ();
163
164   // to get current time
165   gettimeofday(&detail_time2, NULL);
166   double t2 = detail_time2.tv_sec + (detail_time2.tv_usec/1000000.0);
167
168   NS_LOG_INFO ("Run Simulation.");
169   Simulator::Run ();
170   Simulator::Destroy ();
171   NS_LOG_INFO ("Done.");
172
173   // print total time elapsed in microsecond
174   gettimeofday(&detail_time3, NULL);
175   double t3 = detail_time3.tv_sec + (detail_time3.tv_usec/1000000.0);
176   NS_LOG_INFO ("Total (ms): " << (t3 - t1)*1000);
177   NS_LOG_INFO ("Modbus Simulation time (ms): " << (t3 - t2)*1000);
178   NS_LOG_INFO ("Node and Link creation (ms): " << (t2 - t1)*1000);
179
180   return 0;
181 }
```

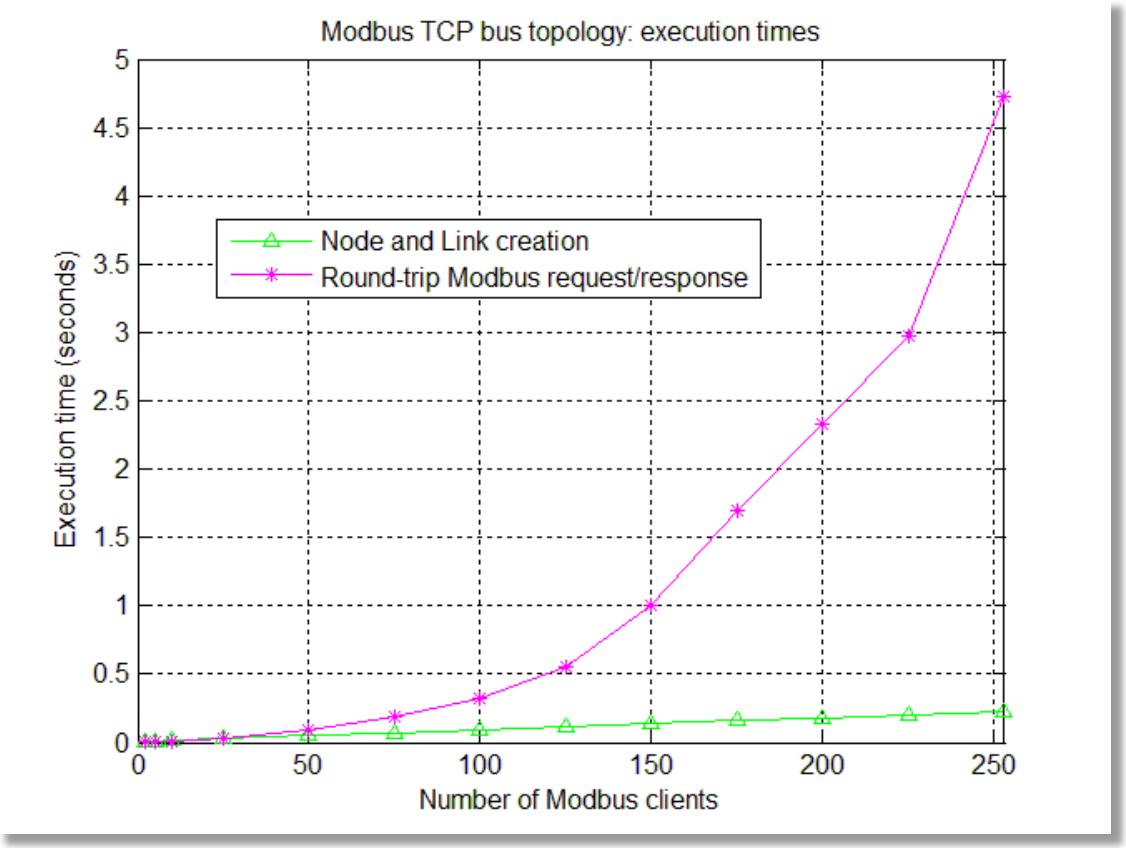
Model performance: scalability Modus UDP -> bus topology



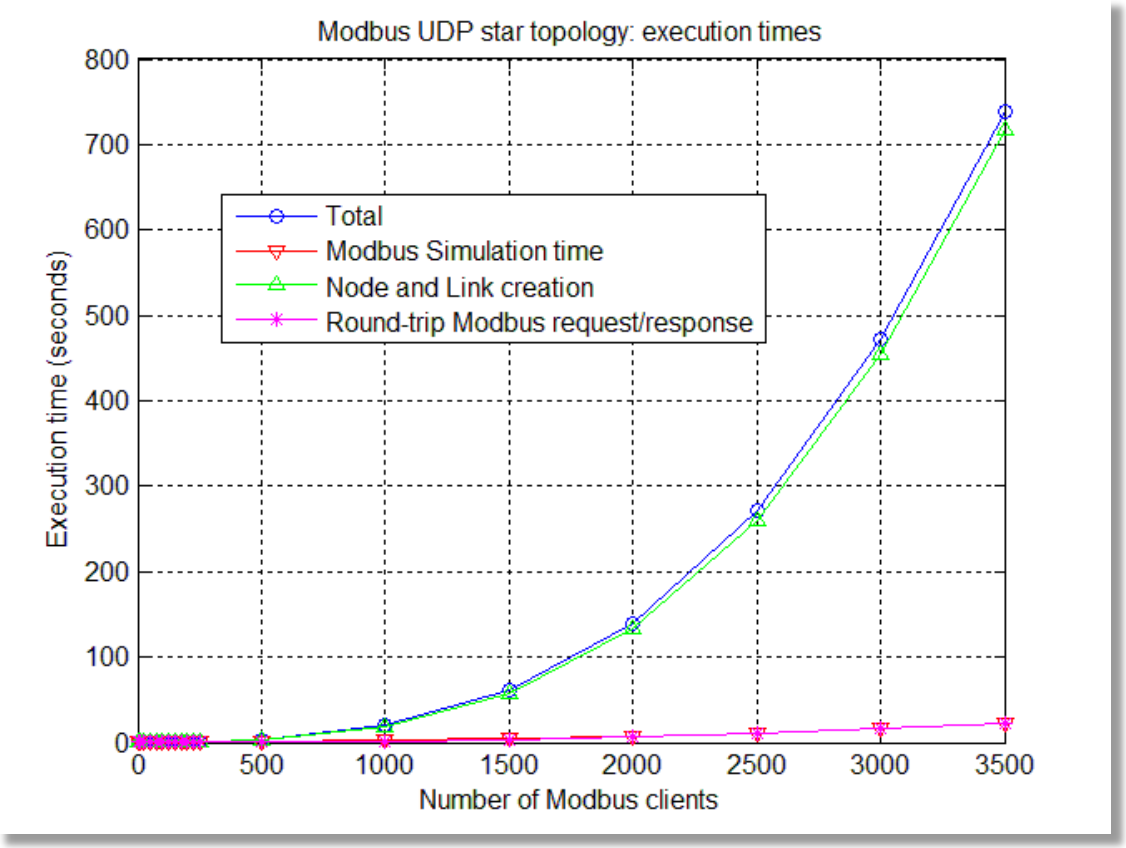
Model performance: scalability Modbus TCP -> bus topology



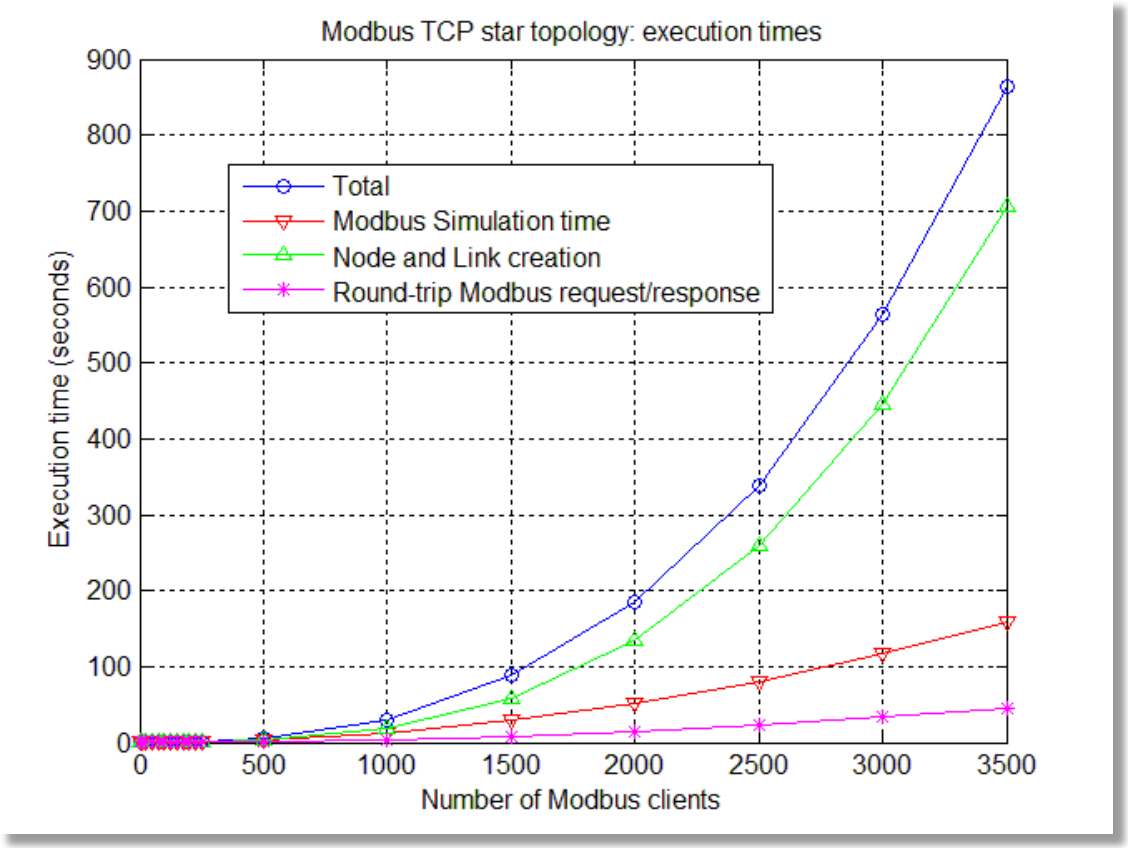
Model performance: scalability Modbus TCP -> bus topology



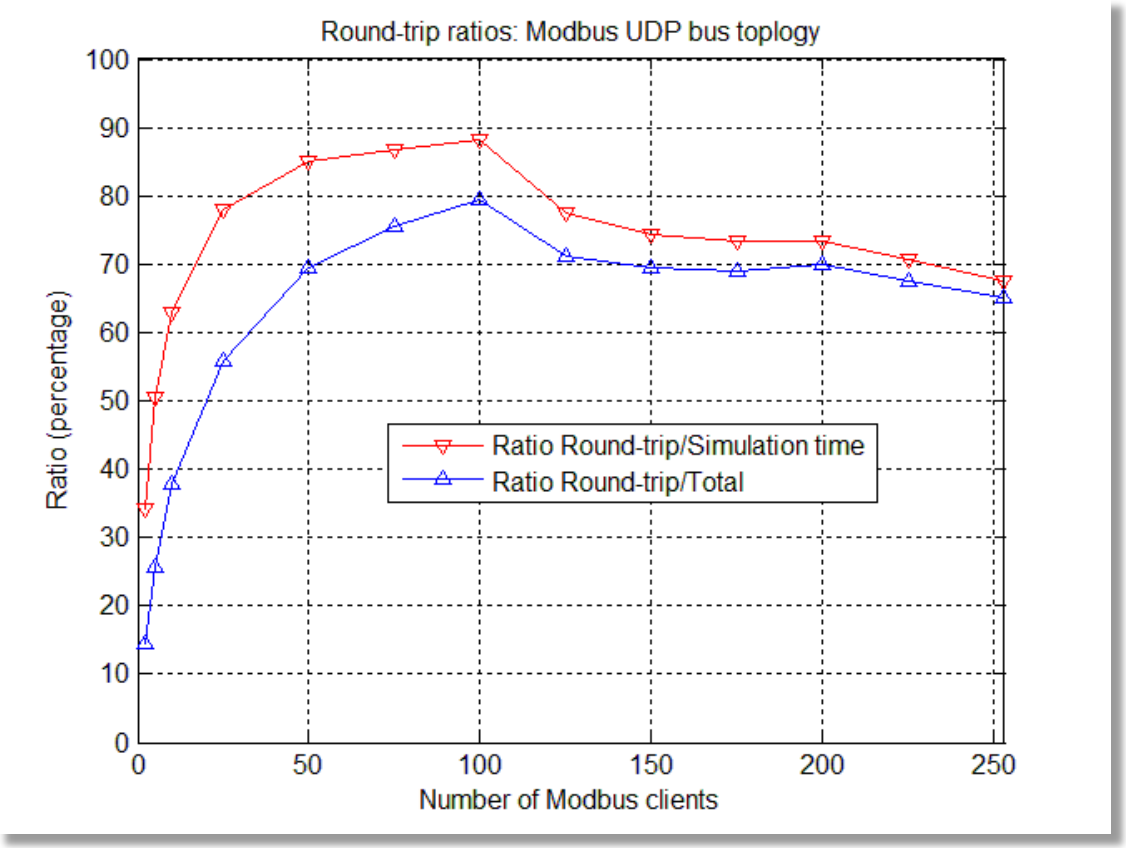
Model performance: scalability Modbus UDP -> star topology



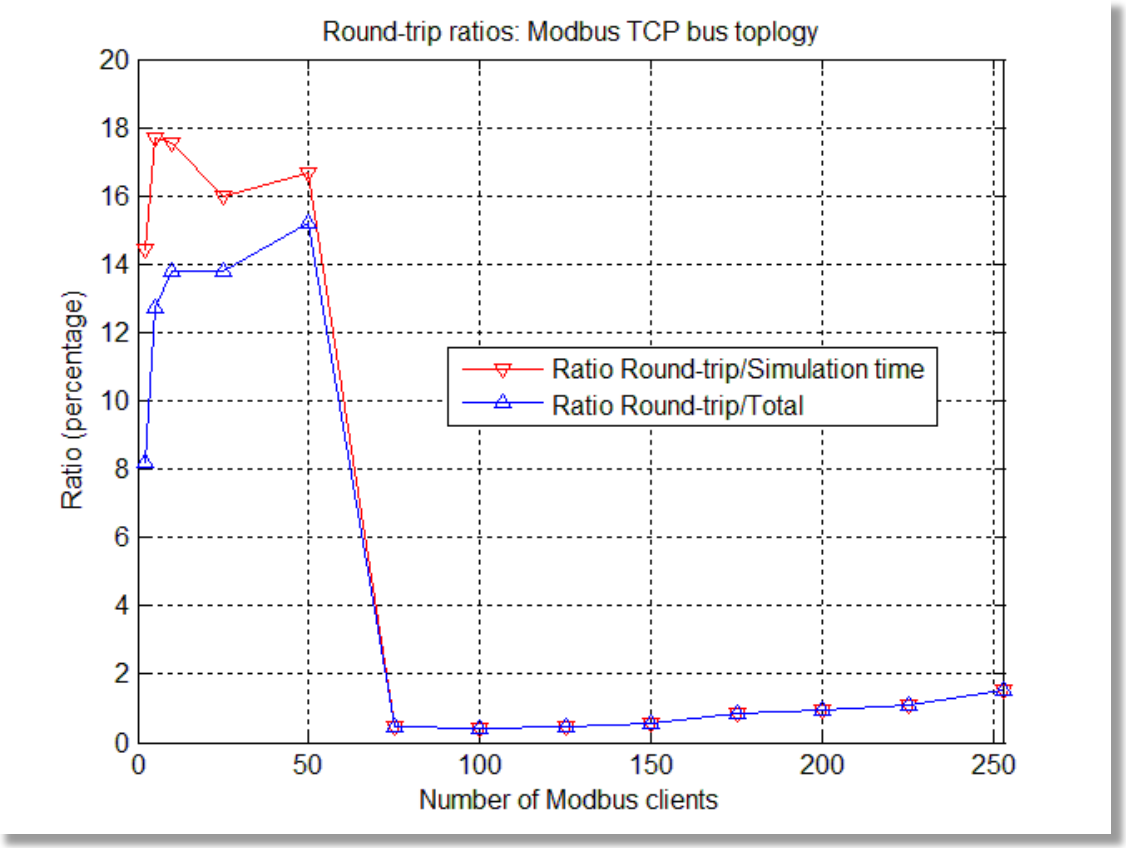
Model performance: scalability Modbus TCP -> star topology



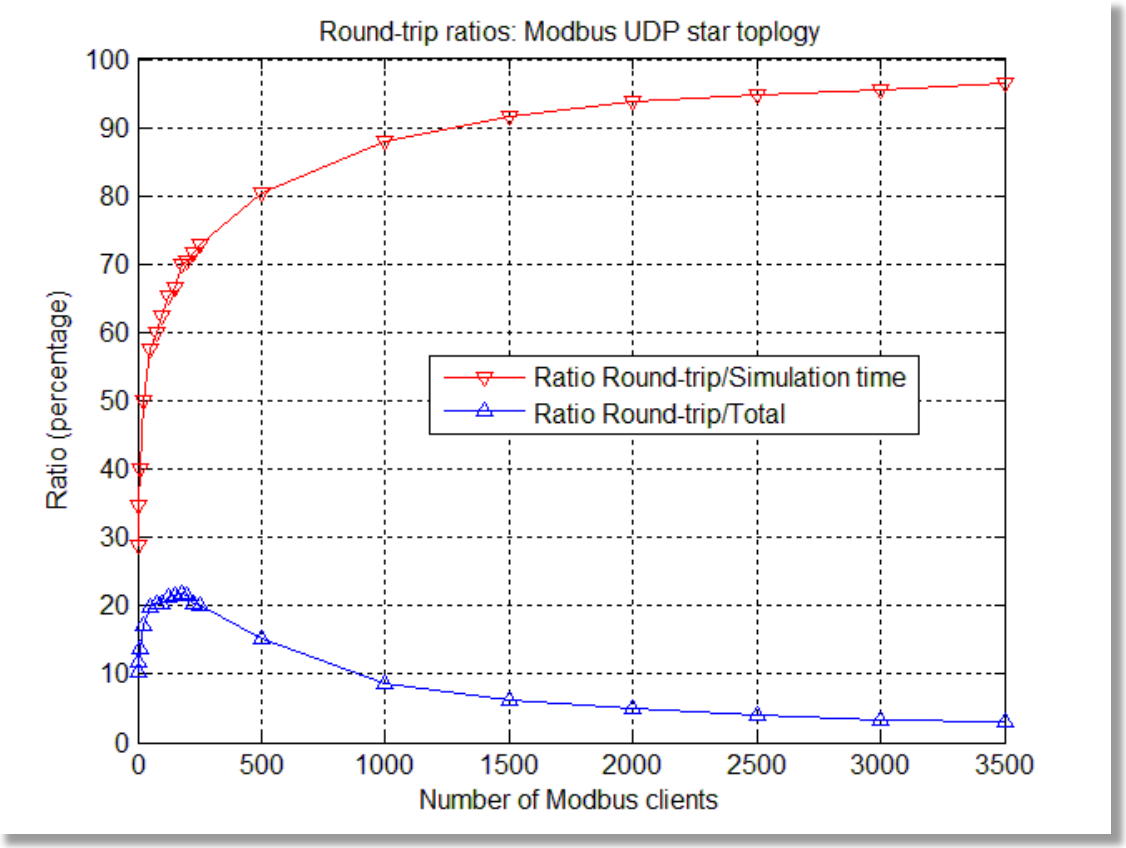
Model performance: runtime overhead Modbus UDP -> bus topology



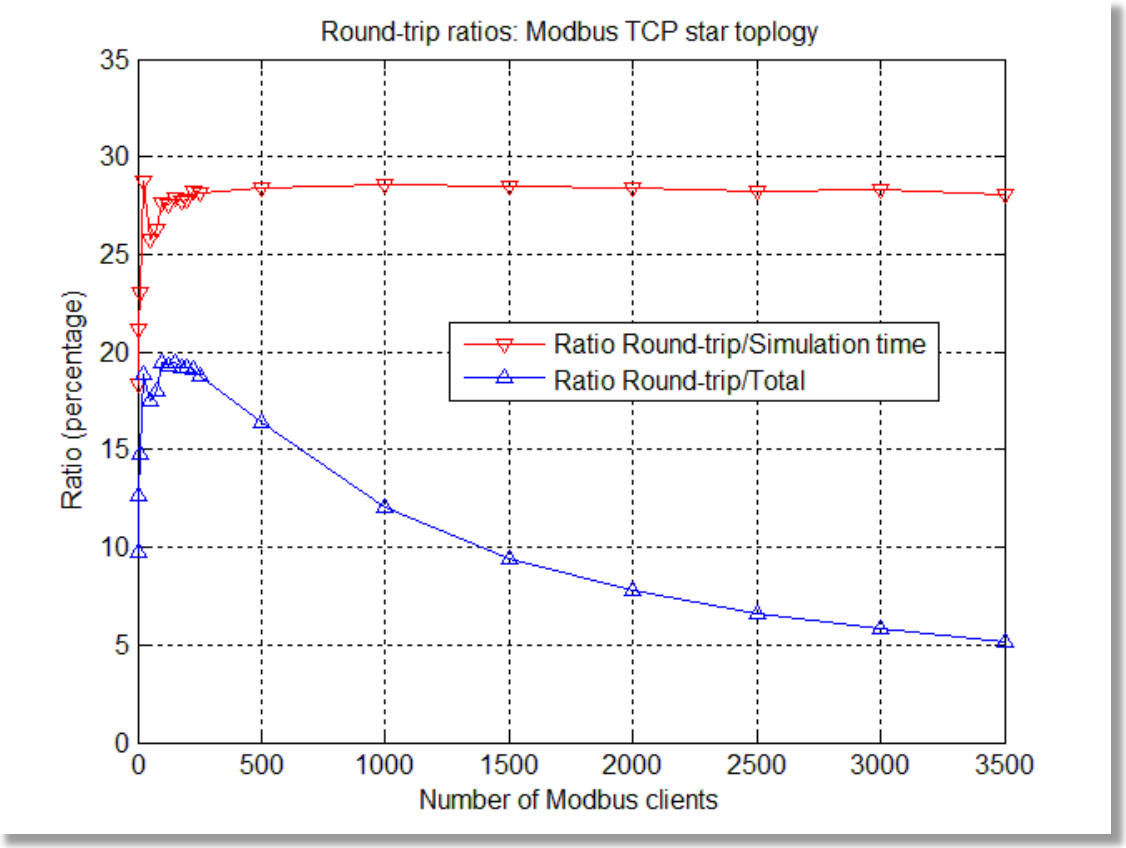
Model performance: runtime overhead Modbus TCP -> bus topology



Model performance: runtime overhead Modbus UDP -> star topology

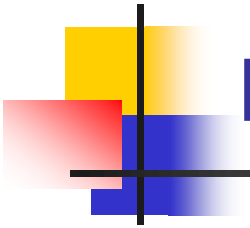


Model performance: runtime overhead Modbus TCP -> star topology



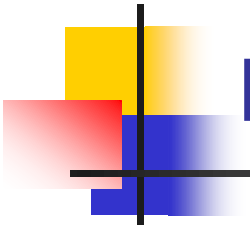
Model performance: simulation robustness

- The maximum number of nodes tested for star topology is 3,500
- The maximum number of nodes tested for bus topology is 253
- No application crashes have been observed



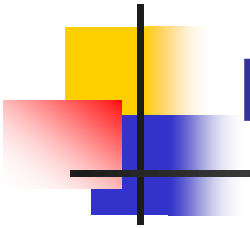
Road map

- Introduction
- Background knowledge
- Ns-Modbus design and implementation
- Verification and validation
- Model performance
- Future work
- Conclusion
- References



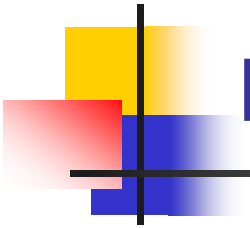
Future work

- Incorporate the ns-Modbus code into ns-3 distribution
- Upgrade ns-Modbus to the latest ns-3 build (ns-3.16)
- Add emulating feature
- Send the Modbus frames in variable length



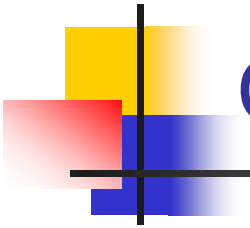
Future work

- Add Modbus client timeout
- Add bit manipulation memory blocks (*m_inputsStatus* and *m_coilsStatus)*
- Employ Modbus conformance test specifications for verification of the application



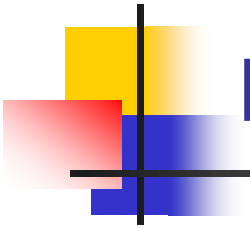
Road map

- Introduction
- Background knowledge
- Ns-Modbus design and implementation
- Verification and validation
- Model performance
- Future work
- Conclusion
- References



Conclusion

- Achieved the project objectives
  - Modbus TCP and UDP integrated with ns-3
- Validation result
  - Logging messages, PCAP captures, and output animation show ns-Modbus works properly and robustly
- Modbus core classes may be used in industrial devices
- Ns-Modbus over UDP has much better performance especially for the number of nodes higher than 50

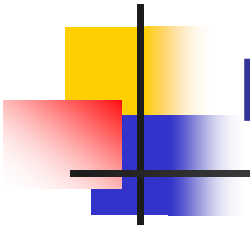


Road map

- Introduction
- Background knowledge
- Ns-Modbus design and implementation
- Verification and validation
- Model performance
- Future work
- Conclusion
- References

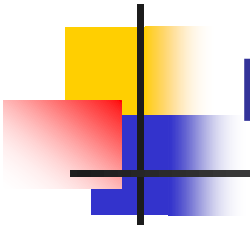
References

- [1] Modbus-IDA. (2006, December) Modbus application protocol specification V1.1b. [Online]. Available: http://www.modbus.com/docs/Modbus_Application_Protocol_V1_1b.pdf.
- [2] D. Peng, H. Zhang, L. Yang, and H. Li, "Design and realization of Modbus protocol based on embedded Linux system," in *Proc. ICESS'*08, July 2008, pp. 275–280.
- [3] ns-3. (2012, November) Network simulator release 3.15. [Online]. Available: http://www.nsnam.org.
- [4] GNU Operating System. (2012, November) GNU General Public License. [Online]. Available: http://www.gnu.org/copyleft/gpl.html.
- [5] ns-3. (2012, November) ns-3 Tutorial, Release ns-3.15 [Online]. Available: http://www.nsnam.org/docs/release/3.15/tutorial/ns-3-tutorial.pdf.
- [6] Mercurial. (2012, November) mercurial [Online]. Available: http://www.mercurial.selenic.com.
- [7] waf. (2012, November) waf [Online]. Available: http://code.google.com/p/waf.



References

- [8] ns-3. (2012, April) ns-3 Tutorial [Online]. Available: http://www.nsnam.org/docs/release/3.12/tutorial/singlehtml/index.html.
- [9] Modbus Organization (2009, December) Conformance test specification for Modbus TCP Version 3.0. [Online]. Available: http://www.modbus.org/docs/MBConformanceTestSpec_v3.0.pdf.
- [10] ns-3. (2012, November) Download [Online]. Available: http://www.nsnam.org/ns-3-dev/download.
- [11] ns-3 Network Simulator. (2012, December) ns-3 Manual, Release ns-3.15 [Online]. Available: http://www.nsnam.org/docs/release/3.15/manual/ns-3-manual.pdf.
- [12] ns-3 Network Simulator. (2012, December) ns-3 Model Library, Release ns-3.15 [Online]. Available: http://www.nsnam.org/docs/release/3.15/models/ns-3-model-library.pdf.
- [13] ns-3 A Discrete-Event Network Simulator. (2012, December) ns-3 API, Release ns-3.15 [Online]. Available: http://www.nsnam.org/docs/release/3.15/doxygen/index.html.



References

- [14] ns-3. (2012, December) Coding Style [Online]. Available: http://www.nsnam.org/developers/contributing-code/coding-style.
- [15] ns-3. (2012, December) wiki [Online]. Available: http://www.nsnam.org/wiki/index.php/Main_Page.

Thank You