Automated Test Case Generation from P4 Programs

Chris Sommers (presenter)
Kinshuk Mandal
Rudrarup Naskar
Prasenjit Adhikary

June 2018

ixia
A Keysight Business
KEYSIGHT
TECHNOLOGIES

The Need: Test any arbitrary protocol, conveniently, at line rates

Programmable Data plane (exemplified by P4)

THE NEXT STEP IN SDN

P4 Enables:

- ✓ Protocol Independence
- ✓ Handle existing and future protocols
- ✓ Target Independence
- ✓ Line-rate processing

PROBLEM: How do you test a new protocol with existing line-rate testers?

☹ new protocols not standardized yet, are experimental or proprietary
☹ tools don't generally anticipate unknown protocols, or else handle them inadequately

---

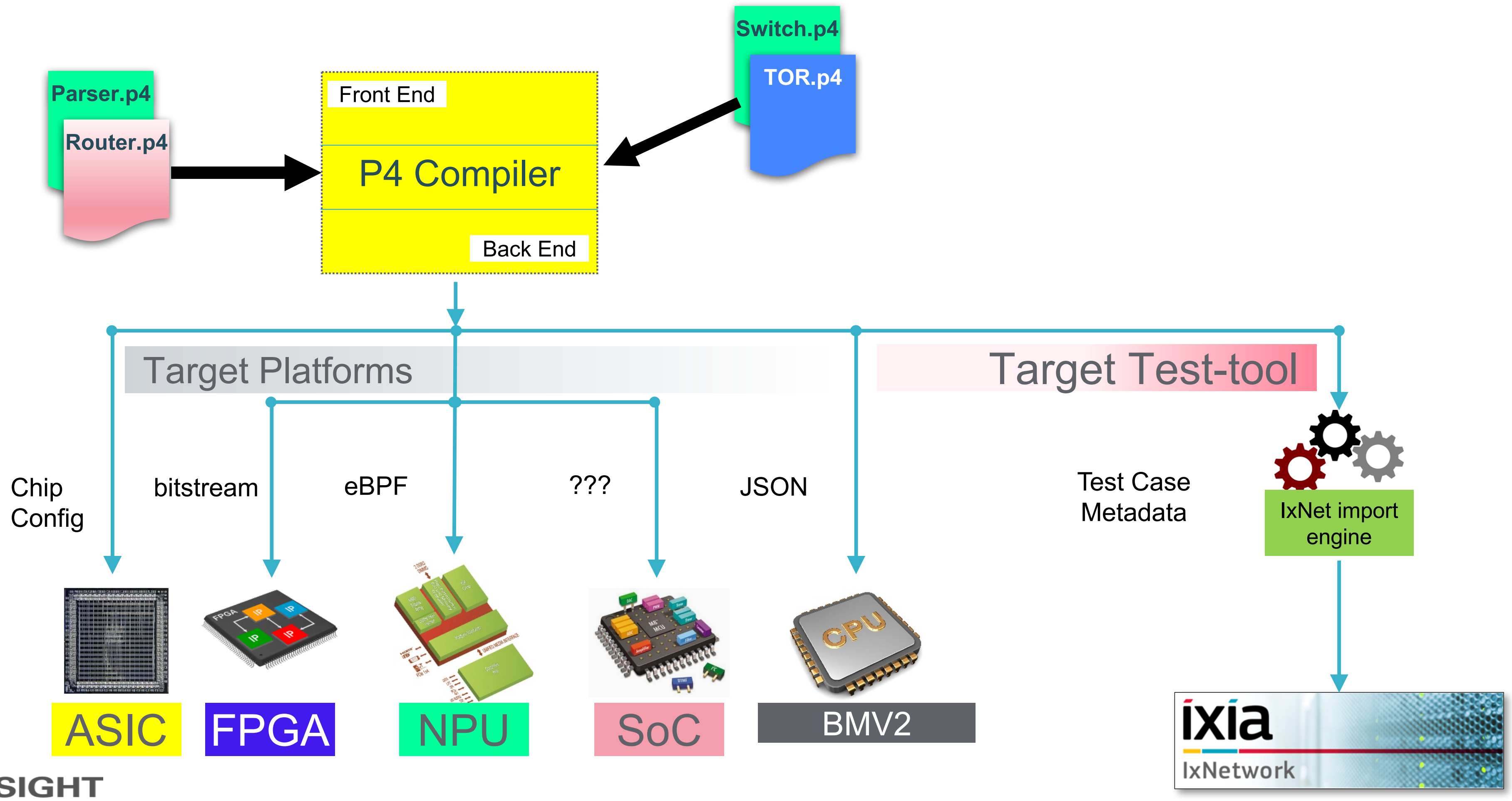
SOLUTION:
The P4 code which defines the function of a device, can also act as the **specification for the test-tool**.

Thus we can achieve a protocol-independent "protocol test tool". *

KEYSIGHT TECHNOLOGIES

* Keysight has HW based tools for data plane testing

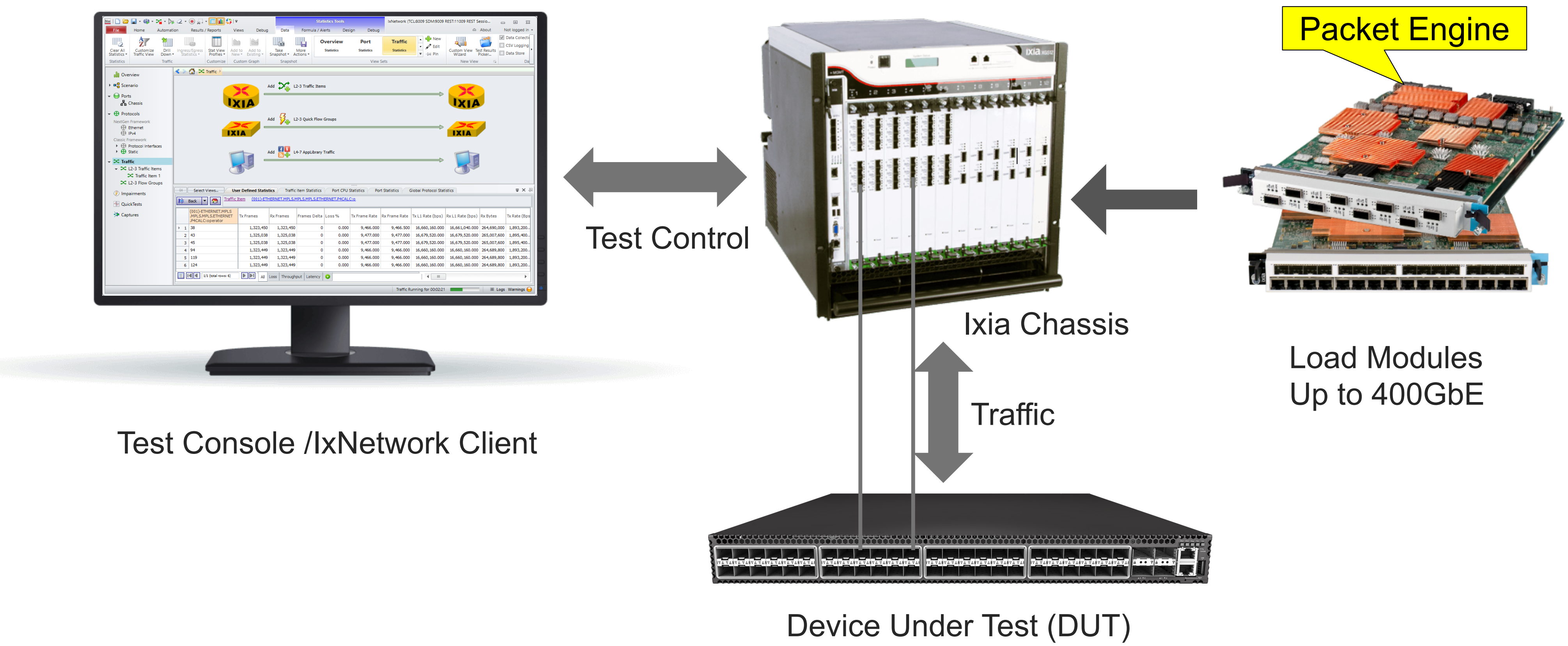
WORKFLOW



KEYSIGHT TECHNOLOGIES

Overview of IxNetwork

IxNetwork + Physical Test Chassis



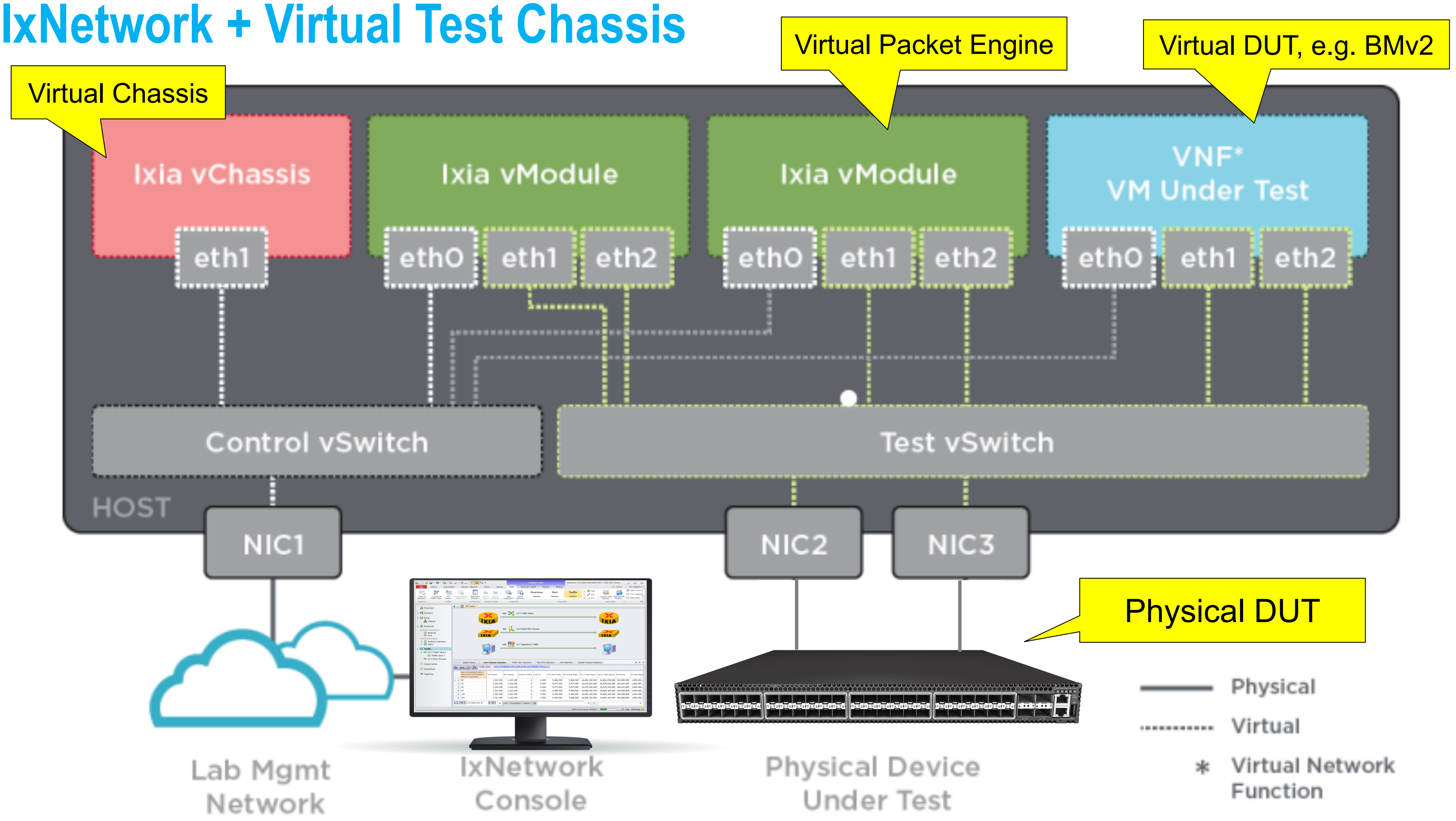
Traffic Item (001)-ETHERNET.MPLS.MPLS.MPLS.ETHERNET.P4CALC:o

	(001)-ETHERNET.MPLS.MPLS.MPLS.ETHERNET.P4CALC:operator	Tx Frames	Rx Frames	Frames Delta	Loss %	Tx Frame Rate	Rx Frame Rate	Tx L1 Rate (bps)	Rx L1 Rate (bps)	Rx Bytes	Tx Rate (Bps
1	38	1,323,450	1,323,450	0	0.000	9,466.000	9,466.500	16,660,160.000	16,661,040.000	264,690,000	1,893,200..
2	43	1,325,038	1,325,038	0	0.000	9,477.000	9,477.000	16,679,520.000	16,679,520.000	265,007,600	1,895,400..
3	45	1,325,038	1,325,038	0	0.000	9,477.000	9,477.000	16,679,520.000	16,679,520.000	265,007,600	1,895,400..
4	94	1,323,449	1,323,449	0	0.000	9,466.000	9,466.000	16,660,160.000	16,660,160.000	264,689,800	1,893,200..
5	119	1,323,449	1,323,449	0	0.000	9,466.000	9,466.000	16,660,160.000	16,660,160.000	264,689,800	1,893,200..
6	124	1,323,449	1,323,449	0	0.000	9,466.000	9,466.000	16,660,160.000	16,660,160.000	264,689,800	1,893,200..

1/1 (total rows: 6)

Traffic Running for 00:02:21

IxNetwork + Virtual Test Chassis

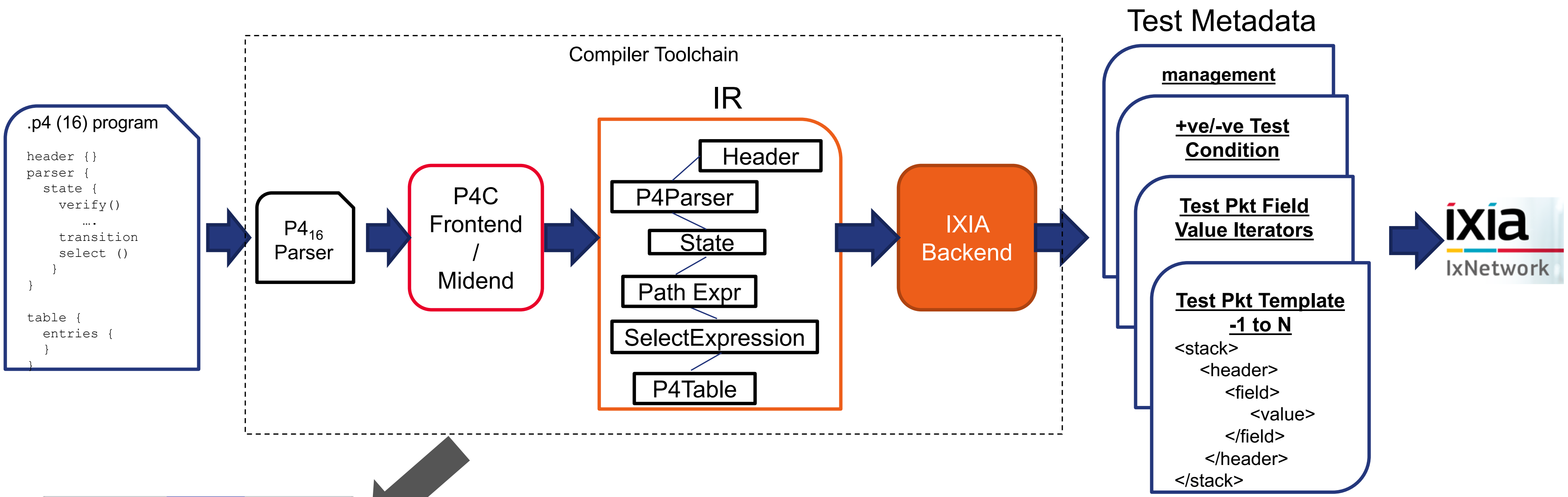


What did we do?

Enhancing IxNetwork to be P4-Aware

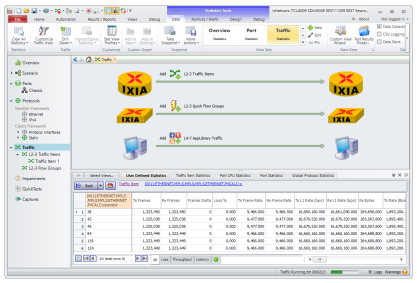
- ✓ Create a new backend for the p4c compiler: p4c-ixia. Output is test-case metadata
- ✓ Enhance IxNetwork to embed and launch p4c-ixia and to import the test-case metadata
- ✓ Enhance IxNetwork to translate test-case metadata into test data streams utilizing our packet engines
- ✓ Existing load modules (physical and virtual) are already highly programmable and largely protocol-agnostic. *No modifications were required on the packet engines.*
- ✓ This also allows both our physical and virtual packet testers to support p4 testing.

Architecture



```
header {}
parser {
   state {
     verify()
       ….
     transition
     select ()
   }
}

table {
   entries {
   }
}
```

```
<stack>
   <header>
      <field>
         <value>
      </field>
   </header>
</stack>
```



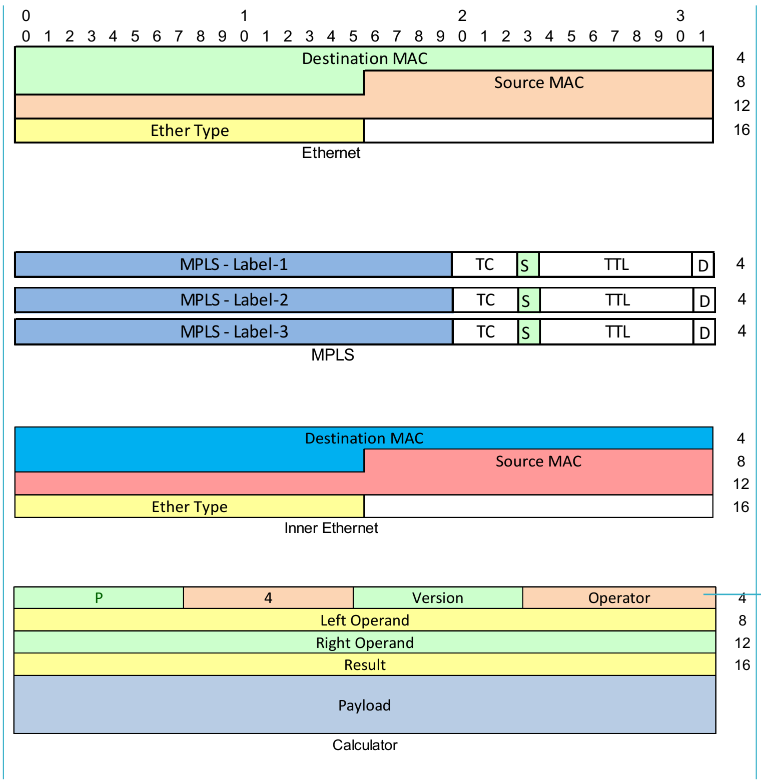
p4c-ixia embedded within IxNetwork

Metadata Packet Structures & Field Values:

- All possible valid sequence of protocol headers (packet templates).
- “Header Stacks” information (eg: multiple MPLS labels)
- Erroneous Packets (eg: hitting the reject rules, exceeding the boundary conditions)
- Packet fields values to test the limit of “Verify” condition
- Packet structure and field values needed to execute “Key Set” for “Select”
- Dealing with constant entries in “P4 Table”.

Calculator Protocol – An arbitrary data-plane as test case

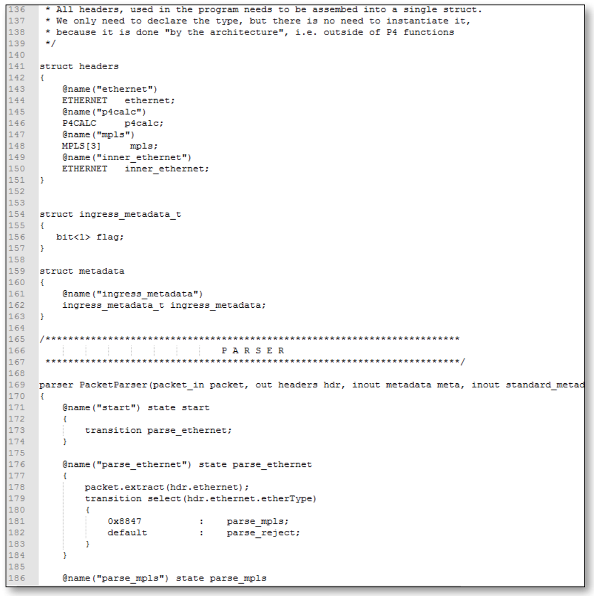
The Data Plane



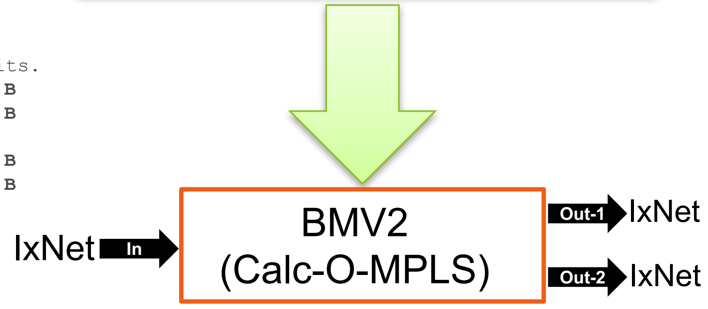
- Header stack
- Validation : pkts with 1~3 labels

OPERATOR is an operation to Perform. It is of 8 bits.

- '+' (0x2b) **Result=Left Operand + Right Operand B**
- '-' (0x2d) **Result=Left Operand - Right Operand B**
- '&' (0x26) **Result=Left Operand & Operand B.**
- '|' (0x7c) **Result=Left Operand | Right Operand B**
- '^' (0x5e) **Result=Left Operand ^ Right Operand B**



```
 * All headers, used in the program needs to be assembed into a single struct.
 * We only need to declare the type, but there is no need to instantiate it,
 * because it is done "by the architecture", i.e. outside of P4 functions
 */

struct headers
{
    @name("ethernet")
    ETHERNET   ethernet;
    @name("p4calc")
    P4CALC     p4calc;
    @name("mpls")
    MPLS[3]     mpls;
    @name("inner_ethernet")
    ETHERNET   inner_ethernet;
}


struct ingress_metadata_t
{
   bit<1> flag;
}

struct metadata
{
    @name("ingress_metadata")
    ingress_metadata_t ingress_metadata;
}

/*************************************************************************
                                   P A R S E R
 *************************************************************************/

parser PacketParser(packet_in packet, out headers hdr, inout metadata meta, inout standard_metad
{
    @name("start") state start
    {
        transition parse_ethernet;
    }

    @name("parse_ethernet") state parse_ethernet
    {
        packet.extract(hdr.ethernet);
        transition select(hdr.ethernet.etherType)
        {
            0x8847          :    parse_mpls;
            default         :    parse_reject;
        }
    }

    @name("parse_mpls") state parse_mpls
```

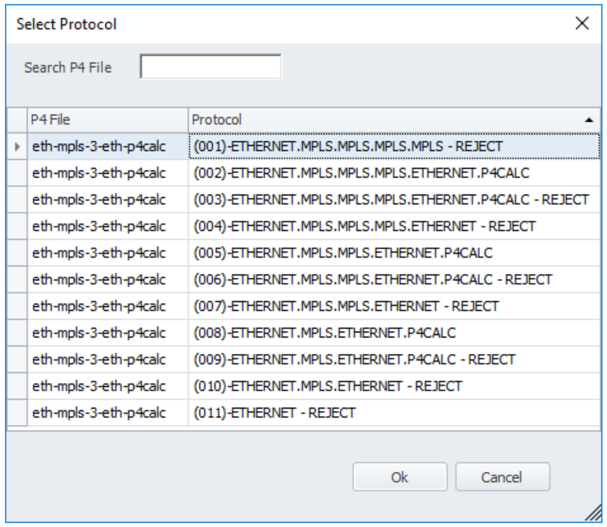


https://github.com/p4lang/tutorials/blob/master/P4D2_2017_Spring/exercises/calc/solution/calc.p4

Video Demonstration

Results : Templates (Positives & Negatives)

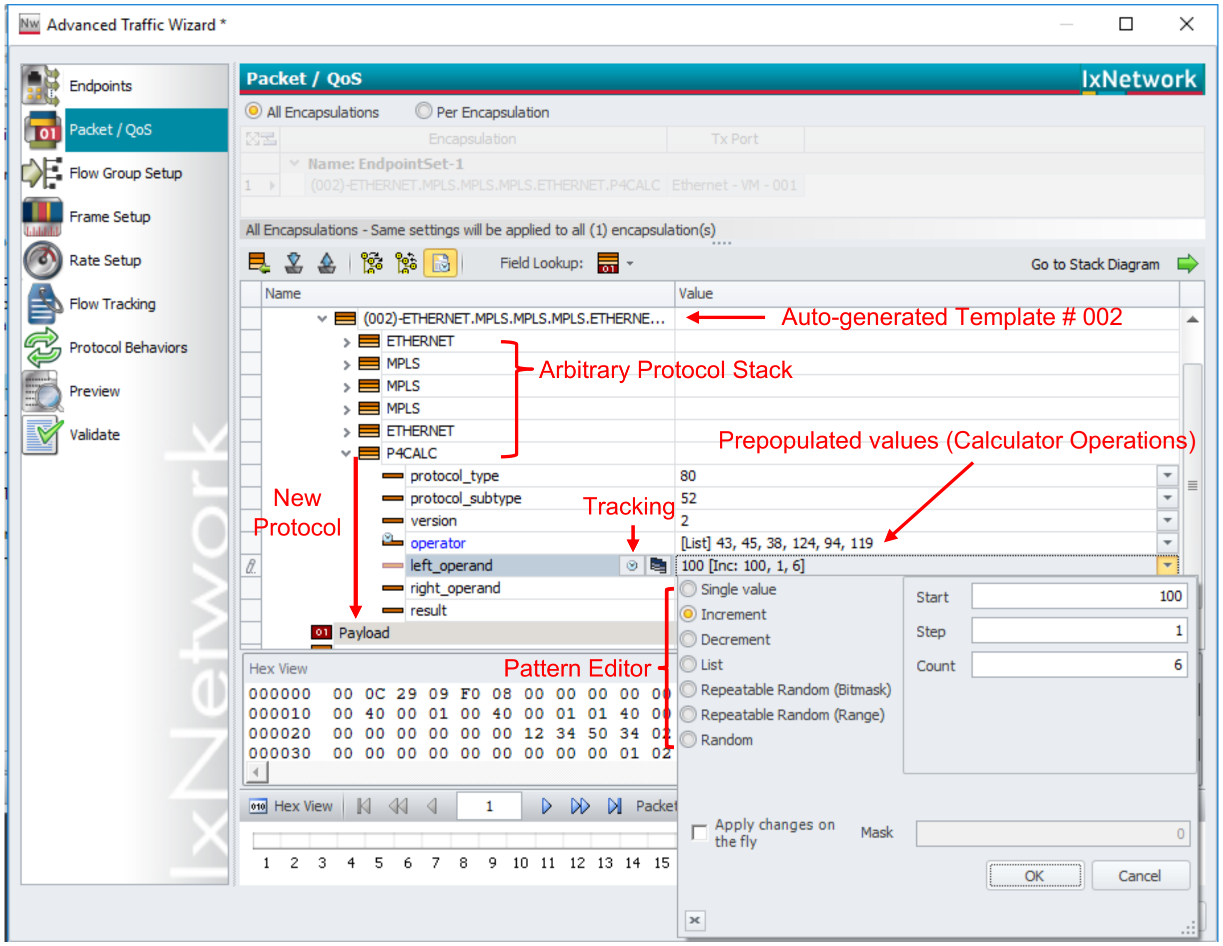
Stack name	Details
"(001)-ETHERNET.MPLS.MPLS.MPLS.MPLS - REJECT "	Rejected as 4th MPLS stack not supported.
"(002)-ETHERNET.MPLS.MPLS.MPLS.ETHERNET.P4CALC"	
"(003)-ETHERNET.MPLS.MPLS.MPLS.ETHERNET.P4CALC - REJECT"	Rejected by p4calc version (0x503402 accepted type)
"(004)-ETHERNET.MPLS.MPLS.MPLS.ETHERNET - REJECT"	Rejected by inner eitherType (0x1234 accepted type – calc protocol)
"(005)-ETHERNET.MPLS.MPLS.ETHERNET.P4CALC"	
"(006)-ETHERNET.MPLS.MPLS.ETHERNET.P4CALC - REJECT"	Rejected by p4calc version (0x503402 accepted type)
"(007)-ETHERNET.MPLS.MPLS.ETHERNET - REJECT"	Rejected by inner eitherType (0x1234 accepted type – calc protocol)
"(008)-ETHERNET.MPLS.ETHERNET.P4CALC"	
"(009)-ETHERNET.MPLS.ETHERNET.P4CALC - REJECT"	Rejected by p4calc version (0x503402 accepted type)
"(010)-ETHERNET.MPLS.ETHERNET - REJECT"	Rejected by inner eitherType (0x1234 accepted type – calc protocol)
"(011)-ETHERNET - REJECT"	Rejected by outer eitherType (0x8847 accepted type)



P4 File	Protocol
eth-mpls-3-eth-p4calc	(001)-ETHERNET.MPLS.MPLS.MPLS.MPLS - REJECT
eth-mpls-3-eth-p4calc	(002)-ETHERNET.MPLS.MPLS.MPLS.ETHERNET.P4CALC
eth-mpls-3-eth-p4calc	(003)-ETHERNET.MPLS.MPLS.MPLS.ETHERNET.P4CALC - REJECT
eth-mpls-3-eth-p4calc	(004)-ETHERNET.MPLS.MPLS.MPLS.ETHERNET - REJECT
eth-mpls-3-eth-p4calc	(005)-ETHERNET.MPLS.MPLS.ETHERNET.P4CALC
eth-mpls-3-eth-p4calc	(006)-ETHERNET.MPLS.MPLS.ETHERNET.P4CALC - REJECT
eth-mpls-3-eth-p4calc	(007)-ETHERNET.MPLS.MPLS.ETHERNET - REJECT
eth-mpls-3-eth-p4calc	(008)-ETHERNET.MPLS.ETHERNET.P4CALC
eth-mpls-3-eth-p4calc	(009)-ETHERNET.MPLS.ETHERNET.P4CALC - REJECT
eth-mpls-3-eth-p4calc	(010)-ETHERNET.MPLS.ETHERNET - REJECT
eth-mpls-3-eth-p4calc	(011)-ETHERNET - REJECT

Results: What does the Ixia Traffic Engine see & do ?

- ✓ User friendly mechanism to vary any protocol fields.
- ✓ Use Ixia's powerful pattern editor to vary the fields. Underneath Ixia UDF(s) are used to support variation.
- ✓ Flexibility of any fields to track (including the new protocol) at Line rate.
- ✓ Ingress and Egress tracking support.
- ✓ Track on meta data (Frame size, Flow Group etc.).
- ✓ Facility to utilize the latency bin(s)
- ✓ Flow Grouping - lowest level of control on Frame rate / size / start & stop



Packet / QoS

[✓ All Encapsulations] Per Encapsulation

Encapsulation	Tx Port
Name: EndpointSet-1	
(002)-ETHERNET.MPLS.MPLS.MPLS.ETHERNET.P4CALC	Ethernet - VM - 001

All Encapsulations - Same settings will be applied to all (1) encapsulation(s)

Name	Value
(002)-ETHERNET.MPLS.MPLS.MPLS.ETHERNE...	
ETHERNET	
MPLS	
MPLS	
MPLS	
ETHERNET	
P4CALC	
protocol_type	80
protocol_subtype	52
version	2
operator	[List] 43, 45, 38, 124, 94, 119
left_operand	100 [Inc: 100, 1, 6]
right_operand	
result	
Payload	

Single value / [✓ Increment] / Decrement / List / Repeatable Random (Bitmask) / Repeatable Random (Range) / Random

Start	100
Step	1
Count	6
Mask	0

Hex View

```
000000  00 0C 29 09 F0 08 00 00 00 00 00
000010  00 40 00 01 00 40 00 01 01 40 00
000020  00 00 00 00 00 00 12 34 50 34 02
000030  00 00 00 00 00 00 00 00 00 01 02
```

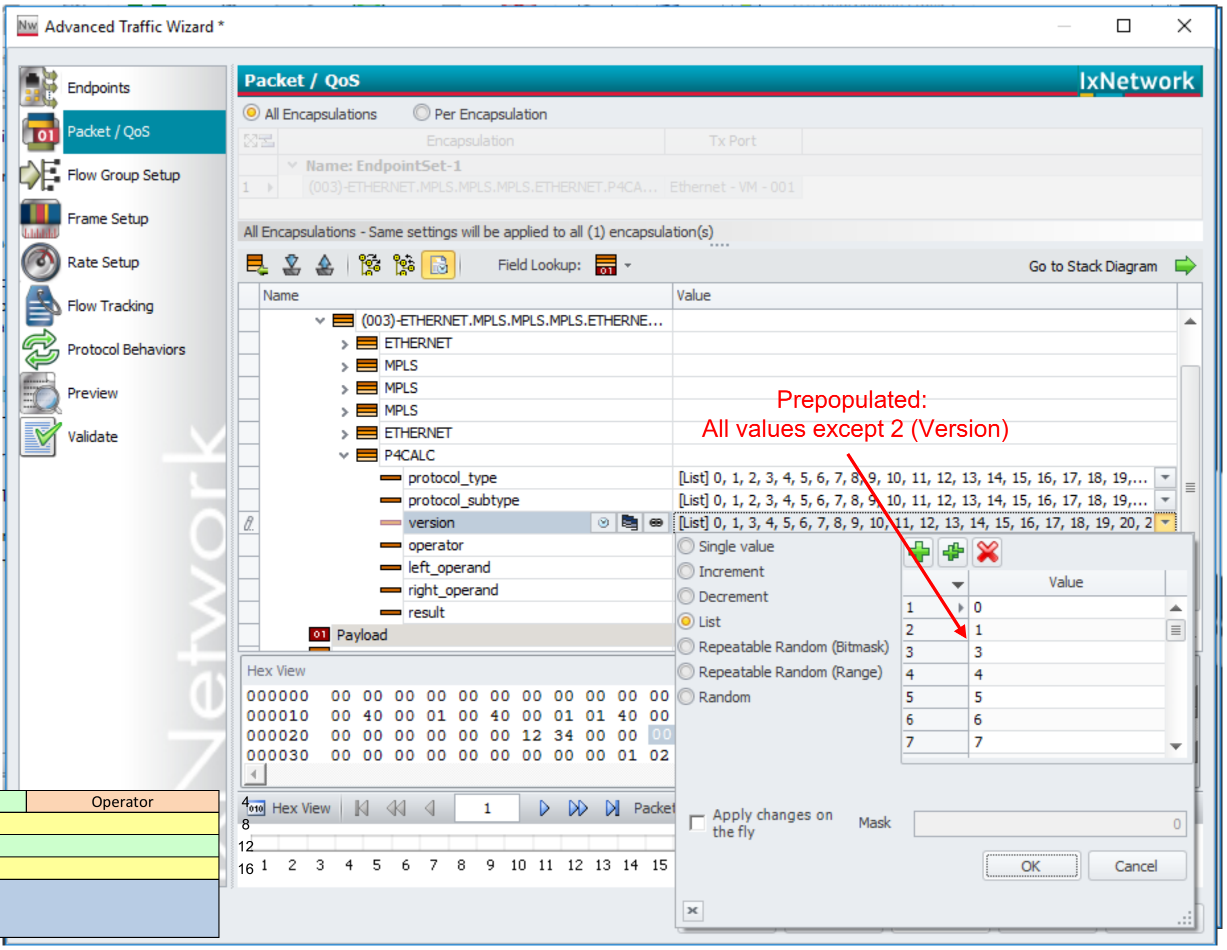
Hex View 1 Packet

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

In this way we achieve a protocol independent "protocol test tool"

Invalid field values (for negative test case 003)

- ✓ Fields are pre-populated with invalid values.
- ✓ For Calculator Protocol, type should be ASCII ‘P’ (decimal 80) and Subtype should be ASCII ‘4’ (decimal 52).
- ✓ Fields are pre-populated with all 8-bit values *except* the valid ones.
- ✓ Pattern Editor customization
- ✓ Similarly, for version field, field is pre-populated with all values other than 2, to verify target behavior for reject scenario.
- ✓ And so forth…



[● All Encapsulations] Per Encapsulation

	Encapsulation	Tx Port
	Name: EndpointSet-1	
1	(003)-ETHERNET.MPLS.MPLS.MPLS.ETHERNET.P4CA…	Ethernet - VM - 001

All Encapsulations - Same settings will be applied to all (1) encapsulation(s)

Name	Value
(003)-ETHERNET.MPLS.MPLS.MPLS.ETHERNE…	
ETHERNET	
MPLS	
MPLS	
MPLS	
ETHERNET	
P4CALC	
protocol_type	[List] 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,…
protocol_subtype	[List] 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,…
version	[List] 0, 1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 2
operator	
left_operand	
right_operand	
result	
Payload	

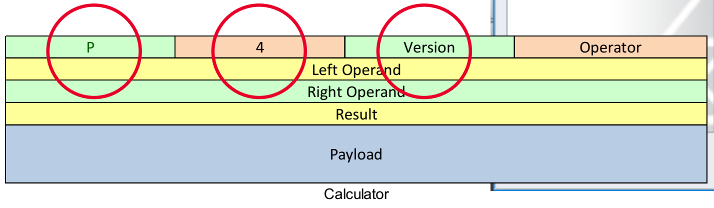
Single value / Increment / Decrement / [● List] / Repeatable Random (Bitmask) / Repeatable Random (Range) / Random

	Value
1	0
2	1
3	3
4	4
5	5
6	6
7	7

Apply changes on the fly [ ] Mask 0

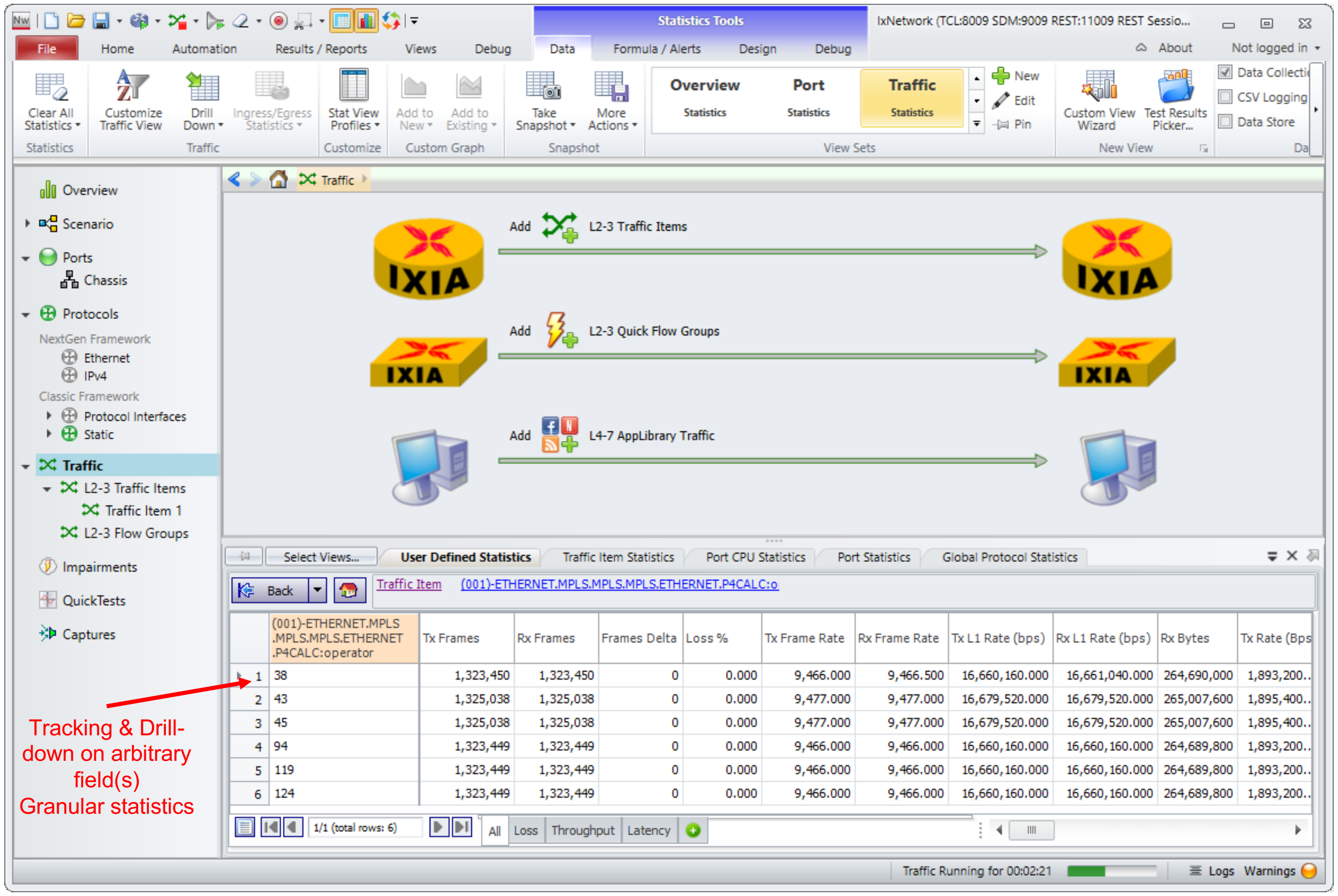
Prepopulated:
All values except 2 (Version)

```
000000   00 00 00 00 00 00 00 00 00 00 00
000010   00 40 00 01 00 40 00 01 01 40 00
000020   00 00 00 00 00 00 12 34 00 00 00
000030   00 00 00 00 00 00 00 00 00 01 02
```



Results : Tracking based on arbitrary fields

- ✓ Packet generation at Line rate.
- ✓ Drill down statistics based on the tacking fields (including arbitrary fields in arbitrary protocol)
- ✓ Simulates thousands of packets in specific order and verify correct order and latency



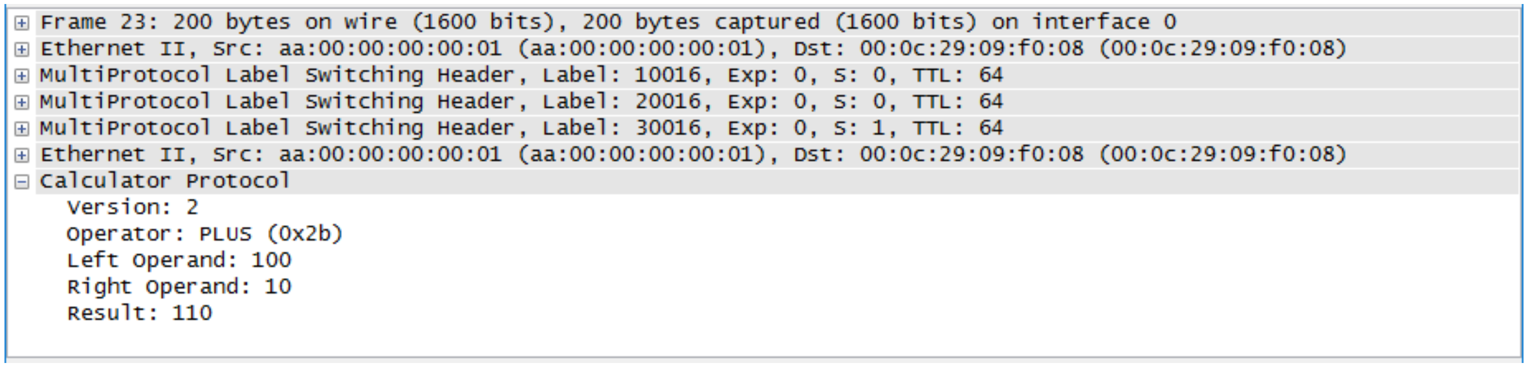
	(001)-ETHERNET.MPLS .MPLS.MPLS.ETHERNET .P4CALC:operator	Tx Frames	Rx Frames	Frames Delta	Loss %	Tx Frame Rate	Rx Frame Rate	Tx L1 Rate (bps)	Rx L1 Rate (bps)	Rx Bytes	Tx Rate (Bps
1	38	1,323,450	1,323,450	0	0.000	9,466.000	9,466.500	16,660,160.000	16,661,040.000	264,690,000	1,893,200..
2	43	1,325,038	1,325,038	0	0.000	9,477.000	9,477.000	16,679,520.000	16,679,520.000	265,007,600	1,895,400..
3	45	1,325,038	1,325,038	0	0.000	9,477.000	9,477.000	16,679,520.000	16,679,520.000	265,007,600	1,895,400..
4	94	1,323,449	1,323,449	0	0.000	9,466.000	9,466.000	16,660,160.000	16,660,160.000	264,689,800	1,893,200..
5	119	1,323,449	1,323,449	0	0.000	9,466.000	9,466.000	16,660,160.000	16,660,160.000	264,689,800	1,893,200..
6	124	1,323,449	1,323,449	0	0.000	9,466.000	9,466.000	16,660,160.000	16,660,160.000	264,689,800	1,893,200..

KEYSIGHT TECHNOLOGIES

What's Next ?

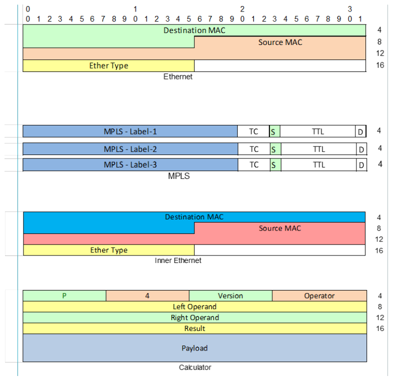
✓ Focus on testing, stability and react to community feedback.

✓ Automatic packet decoder



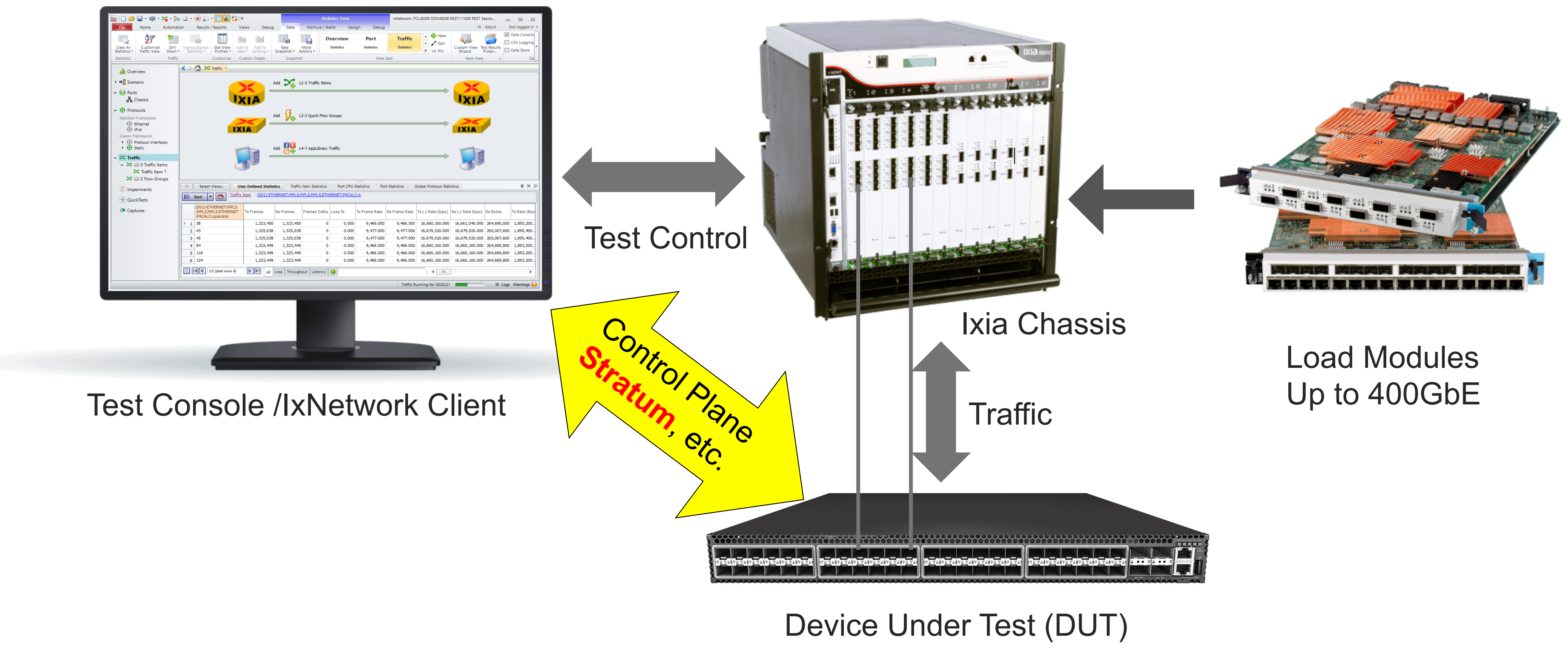
```
⊞ Frame 23: 200 bytes on wire (1600 bits), 200 bytes captured (1600 bits) on interface 0
⊞ Ethernet II, Src: aa:00:00:00:00:01 (aa:00:00:00:00:01), Dst: 00:0c:29:09:f0:08 (00:0c:29:09:f0:08)
⊞ MultiProtocol Label Switching Header, Label: 10016, Exp: 0, S: 0, TTL: 64
⊞ MultiProtocol Label Switching Header, Label: 20016, Exp: 0, S: 0, TTL: 64
⊞ MultiProtocol Label Switching Header, Label: 30016, Exp: 0, S: 1, TTL: 64
⊞ Ethernet II, Src: aa:00:00:00:00:01 (aa:00:00:00:00:01), Dst: 00:0c:29:09:f0:08 (00:0c:29:09:f0:08)
⊟ Calculator Protocol
    Version: 2
    Operator: PLUS (0x2b)
    Left Operand: 100
    Right Operand: 10
    Result: 110
```

✓ Stateful Fuzzing of arbitrary protocol



KEYSIGHT TECHNOLOGIES

Future possibility - Control Plane Integration



KEYSIGHT TECHNOLOGIES

Thank You

Questions?