OSTINATO

User Guide

Table of Contents

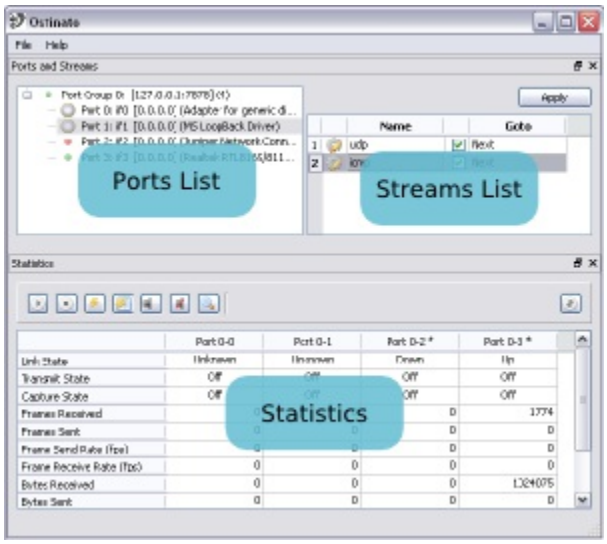
Ostinato is a packet crafter, network traffic generator and analyzer with a friendly GUI and powerful Python API for network test automation.

User Guide

Quickstart

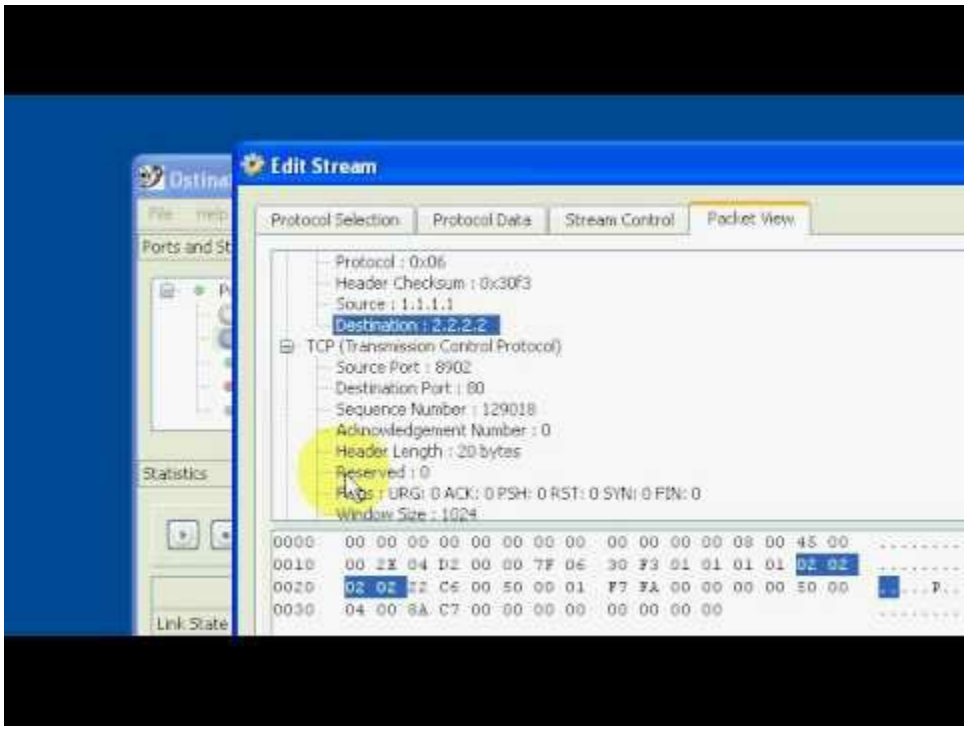
How to craft and send 10 UDP packets

1. At startup notice that the main workspace is divided into 3 main sections - the ports list, streams list and statistics window -



2. You should see a port group entry for "127.0.0.1" in the ports list with a green status
3. Double click the port group to expand it and you will see all the ports on your local system (if you don't see any ports, check FAQ - Port group has no interfaces)
4. Select the port in the ports list on which you want to send packets
5. In the Stream List pane, right click and create a new stream
6. Click the newly created stream to select it and then right click to edit it
7. This will open the Stream Configuration Dialog. This dialog is packed with a lot of options, but don't worry all you need to do for now is -
   - On the "Protocol Selection" tab, select the protocols Mac - Ethernet II - IPv4 - UDP
   - On the "Protocol Data" tab, go to the Internet Protocol ver 4 (IPv4) section and enter source IP as *1.1.1.1* and destination IP as *2.2.2.2*
   - On the "Stream Control" tab, configure no. of packets as 10
   - On the "Packet View" tab, you can preview how your packets will look
   - Click OK
8. Click the "Apply" Button in the Stream List pane **(Do not skip this step, otherwise no packets will be sent!)**
9. In the Statistics window, select the same port for which you configured the stream **(Click the port's column heading to select the full column, otherwise no packets will be sent)**
10. Click the *Start Transmit* button
11. Watch the *Frames Sent* stats increase for that port

See these steps in action -

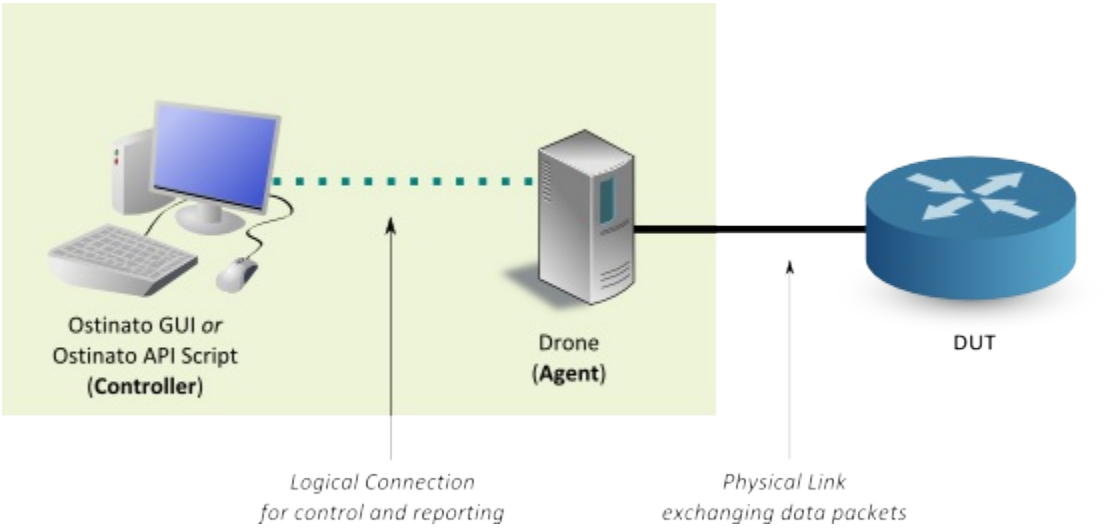


Video link

Controller-Agent Architecture

Ostinato has a controller-agent architecture. There are two corresponding binaries - `ostinato` is the controller GUI and `drone` is the agent. **Both the components are required**. The controller can also be a python script using the `python-ostinato` API.

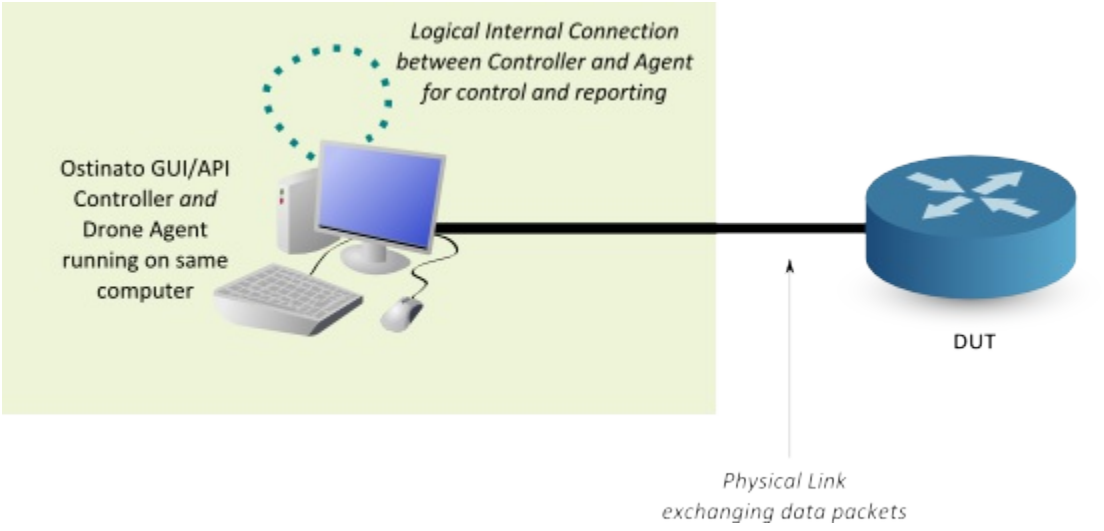
The agent does all the heavy lifting of traffic generation and capture. The controller instructs the agent and fetches reporting data like statistics etc. from the agent. Since the agent does the packet generation, hence the DUT (Device Under Test) is connected to the agent and not to the controller.



> You can also consider this as frontend-backend where the Ostinato controller is the frontend (GUI or API) and backend is the Drone agent. This guide however, will use the controller-agent terminology

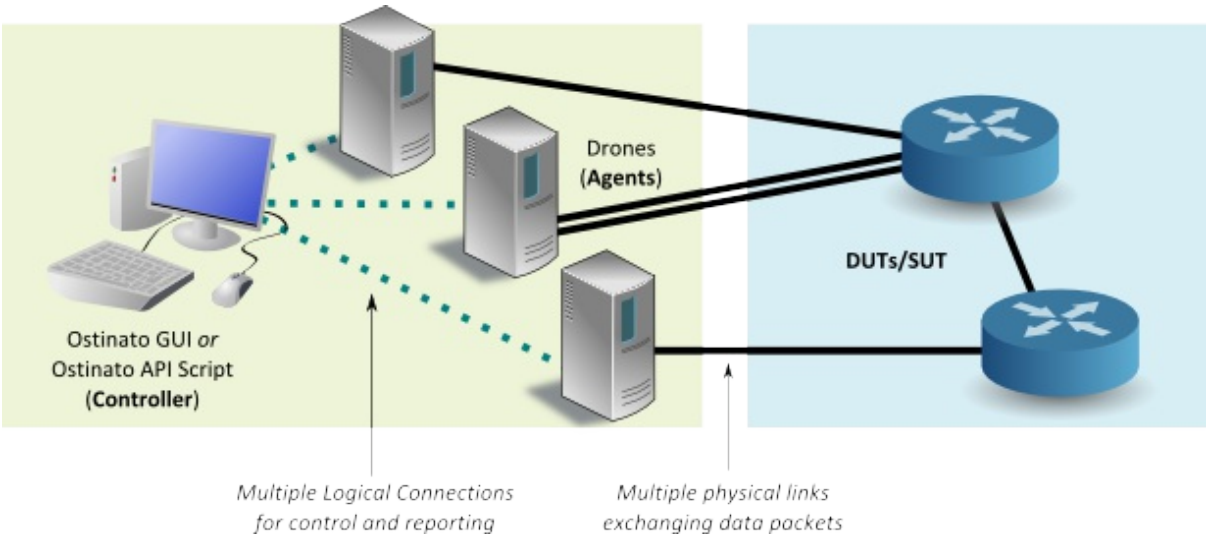
Default Mode

Although the controller and agent are two different binaries, they can also run on the same computer - this is the default mode. Whenever you run the `ostinato` GUI binary, it will internally spawn the `drone` agent binary also. The local agent is represented in the controller by a port group with IP address 127.0.0.1



One controller - many agents

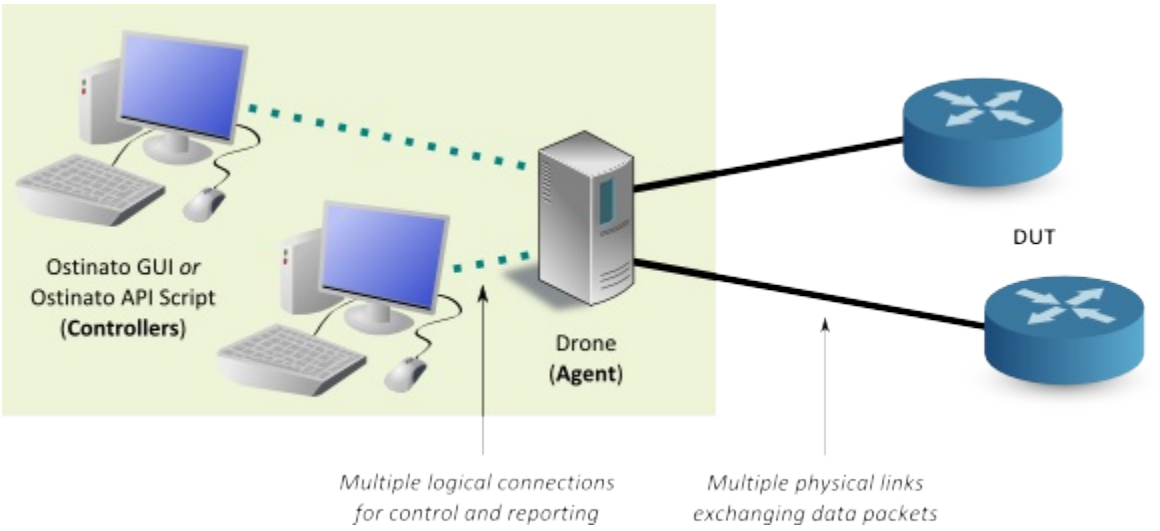
One controller can connect to many agents as shown below -



One example of this use case is when a single agent does not have enough ports to connect to the DUT or SUT (System Under Test)

Many controllers - one agent

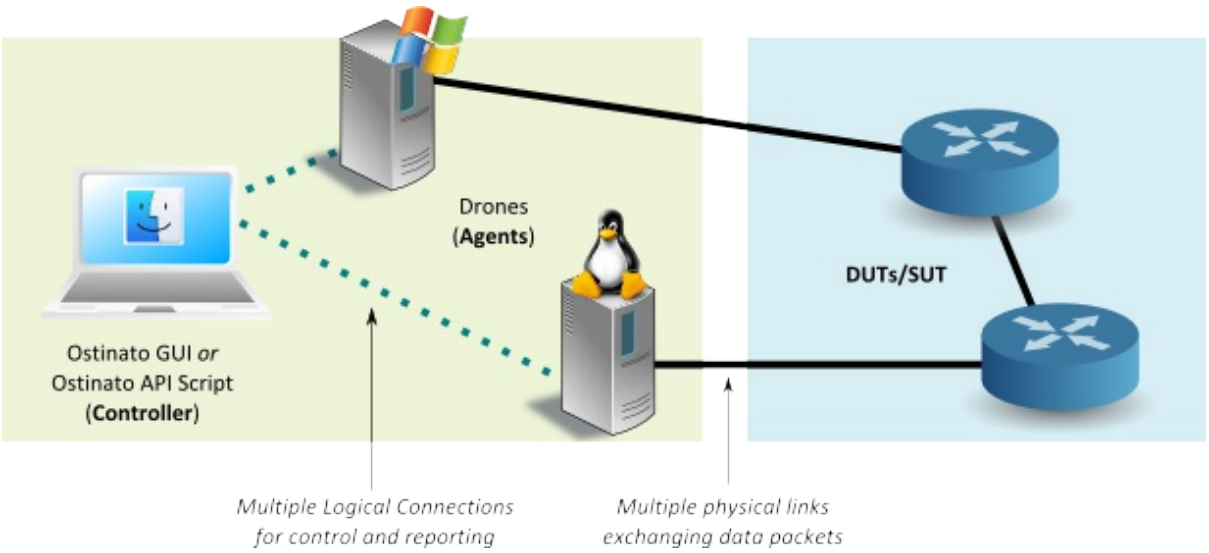
Multiple controllers can connect to the same agent as shown below -



One example of this use case is when an agent has a lot of ports and different controllers are using different ports of the agent connected to different DUT(s).

Cross Platform

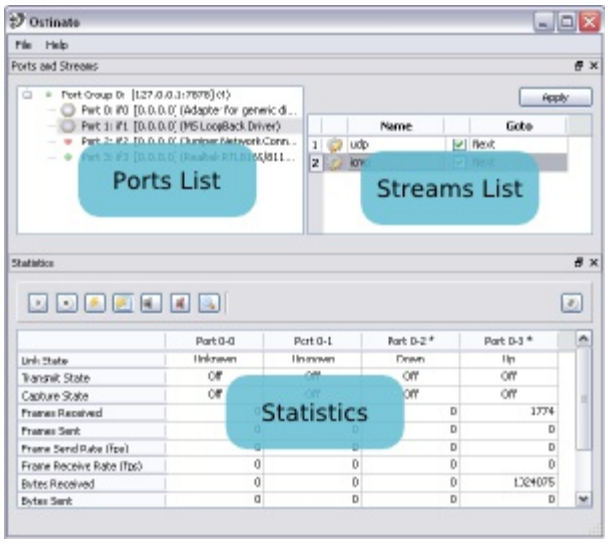
Since the controllers and agents can run on different computers, they can be running on same or different Operating Systems - e.g. you can have a Ostinato GUI controller running on Mac OSX laptop connected to an agent running on Windows Server and another agent running on a Linux Server.



Important Notes

1. Ostinato controller-agent architecture is **NOT** like the iperf client-server architecture. Ostinato and Drone do not exchange any data packets
2. The Ostinato GUI controller sends stream and device configuration information to the agent only when you click the `Apply` button. So if you add/delete/edit streams and/or devices but forget to click `Apply`, your changes won't be reflected when you start the transmit

GUI Layout



The Ostinato GUI main window layout is composed of 3 sections -

PortGroups and Ports List

This section displays all the port groups and all the ports within those port groups in a tree hierarchy

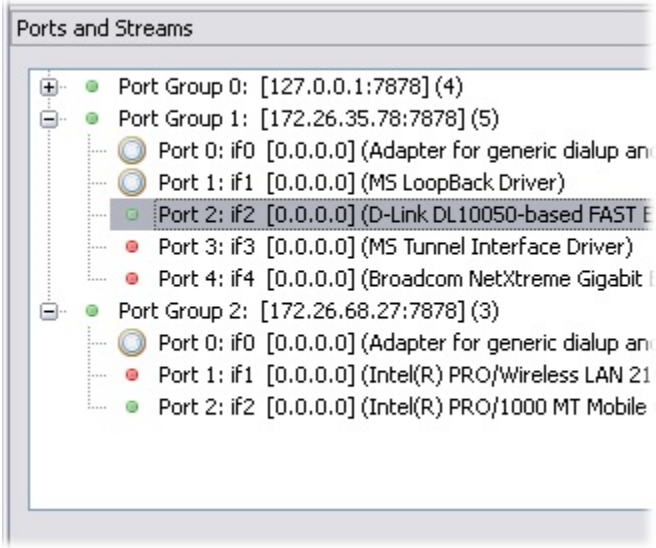
Streams and Devices

This section displays all the streams and devices for the *currently selected port*

Statistics

This section displays statistics for ALL the ports across all port groups

Ports List



The ports list shows all the ports that you can control. Ports are grouped into Port Groups. A Port Group is just a computer or device (local or remote) running the controller component (drone).

The port group status icons are as follows -

Icon	Description
	The client is not connected to the port group
	The agent is trying to connect to the port group
	The agent is connected to the port group
	The agent faced some error connecting to the port group

The port status icons are as follows -

Icon	Description
	The port current link status is unknown
	The port link status is down
	The port link status is up

NOTE: If port is administratively disabled, it may not be listed - this is a Pcap/WinPcap limitation. If no ports are listed - check that drone (the controller component) is running with administrative privileges. For more troubleshooting tips, check the FAQ

Actions

Icon	Action	Description

	New Port Group	Adds a new remote computer to the list and connects to it. You can specify a hostname or IP address and optionally the port number
	Delete Port Group	Deletes a remote computer from the list
	Connect Port Group	Attempts to reconnect to a disconnected remote computer
	Disconnect Port Group	Disconnects from a remote computer. The remote computer is not removed from the list. You can connect to it again
	Exclusive Port Control	See Exclusive Port Control
	Port Configuration	Configure Port properties

Controlling multiple computers

1. On the remote computer that you want to control, run drone (the controller component)
2. In the Ostinato GUI, goto File | New Port Group and enter the IP address
3. The remote computer should appear as a new Port Group in the Ports List

You can connect to any number of remote computers

Port Configuration

Currently the following port properties are available -

Transmit Mode

Sequential Streams: Streams are sent one after the other in a sequential fashion. All packets of one stream are sent before the next stream.

Example: There are 2 streams - TCP and UDP, the TCP stream configured to send 100 packets at the rate of 10 packets/sec and the UDP stream configured to send 500 packets at the rate of 5 packets/sec. In *sequential streams* transmit mode, 100 TCP packets will be sent first at the rate of 10 packets/sec, followed by 500 UDP packets at the rate of 5 packets/sec.

Interleaved Streams: Streams are interleaved based on their packet/burst rate. Packets of all streams are sent together in an interleaved fashion. This is a "continuous" mode - you cannot configure the number of packets/bursts to be sent, you can only configure the packet/burst rates.

Example: There are 2 streams - TCP and UDP, the TCP stream configured to send at the rate of 10 packets/sec and the UDP stream configured to send at the rate of 5 packets/sec. In *interleaved streams* transmit mode, 15 packets will be sent per second, of which 10 packets will be TCP and 5 will be UDP.

Reservation

Starting version 0.7, you can "reserve" ports that you are using - once reserved, other agents that connect to the same drone controller will see your name against the port. If a port is reserved by someone else, you can "force-reserve" it.

Note that reserving a port does NOT restrict others from changing any attributes of the port or even the streams configured on the port. Users are still expected to be cooperative and play nice with each other.

The name used for reservation is taken from the `USER` environment variable (`USERNAME` , for Windows users)

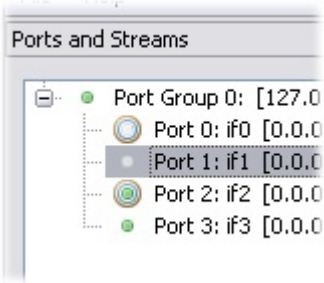
You can restrict the list of ports to your reserved ports using View | Show My Reserved Ports Only

Exclusive Port Control

Exclusive Port Control for Controlled Environment Testing is an "experimental" feature

To prevent the OS from sending packets on a port that you are using, you can take exclusive control of the port. To do so, select the port in the ports list, goto File | Exclusive Control (alternatively, right-click and use the same option from the context menu).

If exclusive control was granted the port's icon in the ports list is decorated with a ringed border as shown below for Port 0 and Port 2 -



NOTE: This feature is currently available only on Windows and not on other platforms. For Windows, this feature is implemented by using an external command line application - bindconfig

Stream Statistics

(*Available release 0.9 onwards*)

By default statistics are tracked at the port or interface granularity. By enabling *Stream Statistics*, you can track stats on the selected port at a per stream granularity

See Stream Statistics for more information on how to use this feature.

Average/Aggregate Port Rates

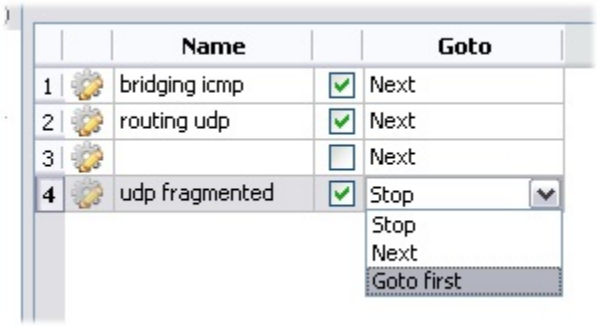
The average/aggregate port rate for all the active streams combined is displayed. Changing the average/aggregate port rate will change the rate for all active streams. The delta of the old rate and new rate is distributed amongst all the active streams such that the ratio of rates between the streams remain the same.

Creating, editing, duplicating and deleting streams

To create a stream, first select the appropriate port in the ports list and then goto File | New Stream (alternatively use the right-click context menu).

To insert a stream between two existing streams, select the bottom stream and then create a new stream - the new stream will be created before the selected stream.

To name the stream, double-click inside the 'name' cell and type a suitable name. The stream can be enabled or disabled using the checkbox. You can configure the flow of packets across all the streams using the 'Goto' column. By default after sending one stream, Ostinato proceeds to the next. You can alternatively configure it to stop or go back to the first stream.



To configure more details about a stream including protocols, packet lengths, rates etc, see Stream Configuration

Starting version 0.7, to duplicate one or more streams, select the stream(s) that you wish to duplicate and then goto File | Duplicate Stream (alternatively use the right-click context menu) - you will be prompted to enter the number of copies you wish to create.

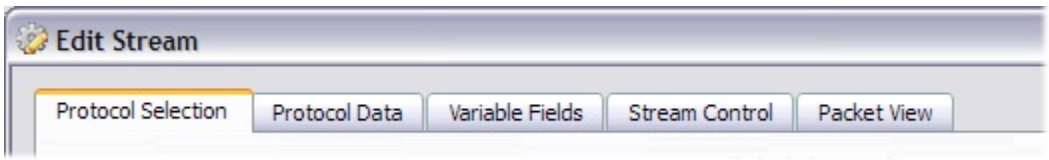
To delete a stream, select the stream(s) and then goto File | Delete Stream (alternatively use the right-click context menu)

IMPORTANT: The agent sends stream configuration information to the controller only when you click the "Apply" button. So if you add/delete/edit streams but forget to click "Apply", your changes won't be reflected when you start the transmit

Stream Configuration

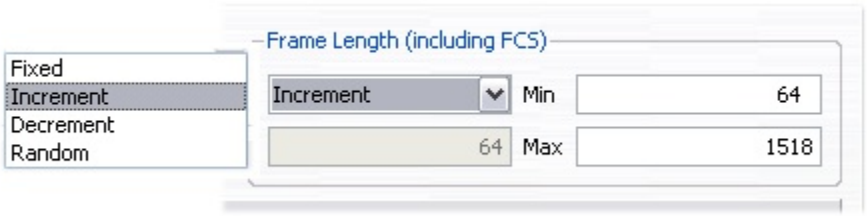
To configure a stream's properties select the stream and goto File | Edit Stream (or alternatively use the right-click context menu). You can even double click the stream icon in the stream list to edit the stream.

Whichever method you employ to edit the stream, you will now see the Stream Configuration Dialog Box. This dialog box has five tabbed pages -



Protocol Selection

Frame Length

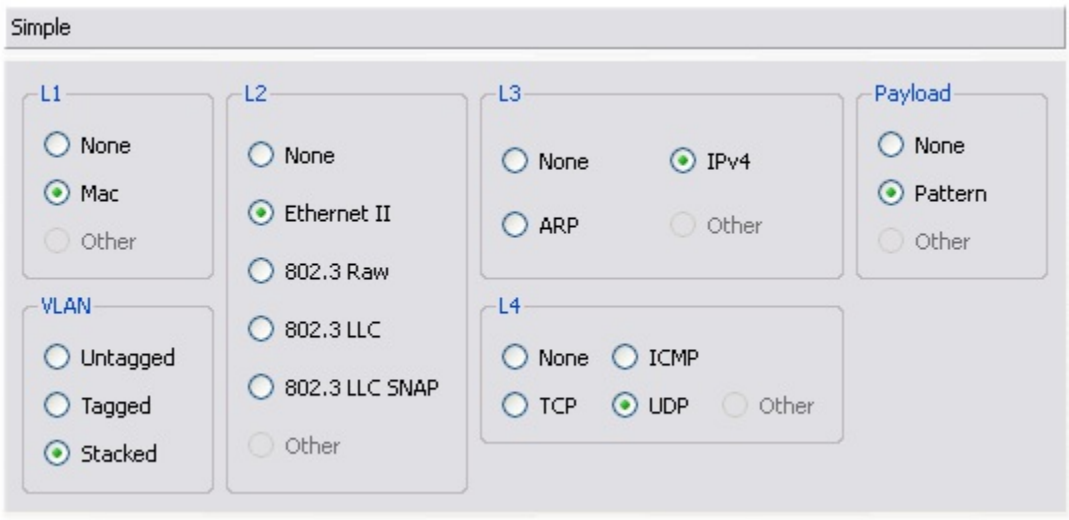


On this page, you can configure the frame length. You can set the frame length to be a fixed value or use any of the other modes viz. incrementing, decrementing or random. For the non-fixed modes, you can configure a minimum and maximum value between which the frame length will vary.

NOTE: For the non-fixed modes, you should configure the stream to send more than 1 packet otherwise you may not see any variation in the frame lengths.

NOTE: The frame length values specified includes the 4 byte FCS.

Protocols (Simple Mode)



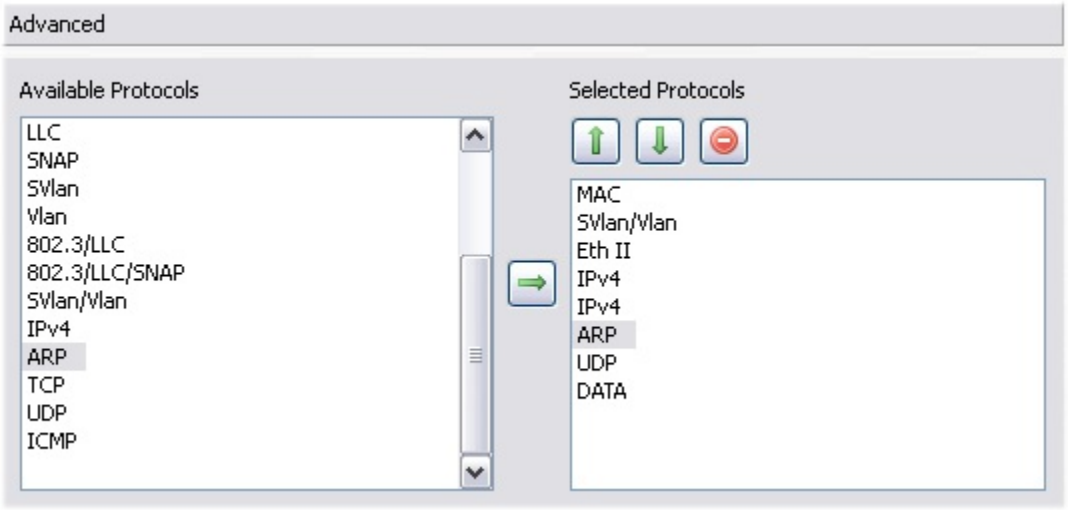
For most usual cases, you can configure the protocols in your packet by clicking on the appropriate radio buttons as desired. Only standard combinations of the various protocols are allowed here (e.g. you can't have have TCP after ARP).

If you need a non standard combination, click on advanced mode. Other cases when you may need advanced mode are -

- More than 2 VLAN tags
- If you need to use the "UserScript" protocol

NOTE: Each protocol level group here has a "Other" choice - this is not available for user selection but is used when a non-standard combination is selected using the advanced mode

Protocols (Advanced Mode)



In this mode, you can stack the protocols that you need in any order that you want (unlike simple mode no checks are performed and you can stack TCP following ARP). You can also stack multiple protocols of the same type (e.g. IP over IP)

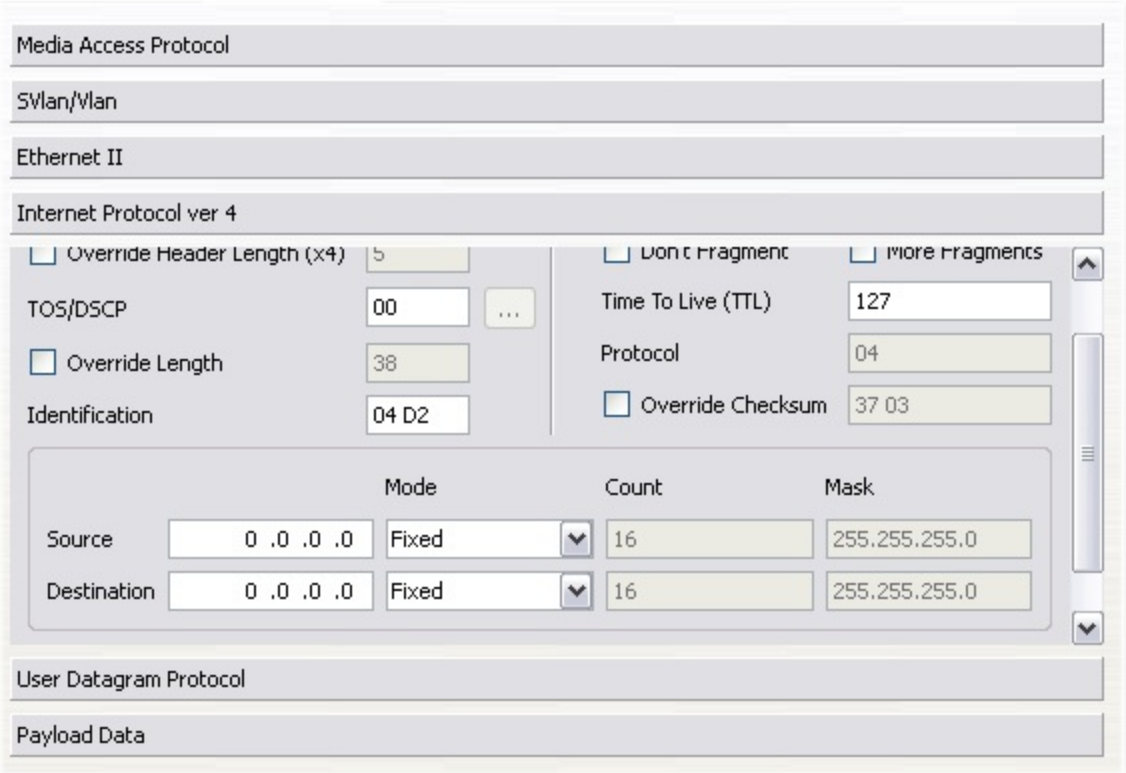
The left pane shows all the available protocols. The right pane shows the protocols that you select for your packet in the order that you want

Use the following toolbar buttons to configure the protocols required and their order -

Icon	Description
	Adds the protocol(s) selected in the left pane to the right pane
	Moves the protocol selected in the right pane up in order
	Moves the protocol selected in the right pane down in order
	Removes the protocol selected in the right pane from the selected list of protocols

Protocol Data

On this page you can configure the fields for each protocol that you selected in the Protocol Selection page

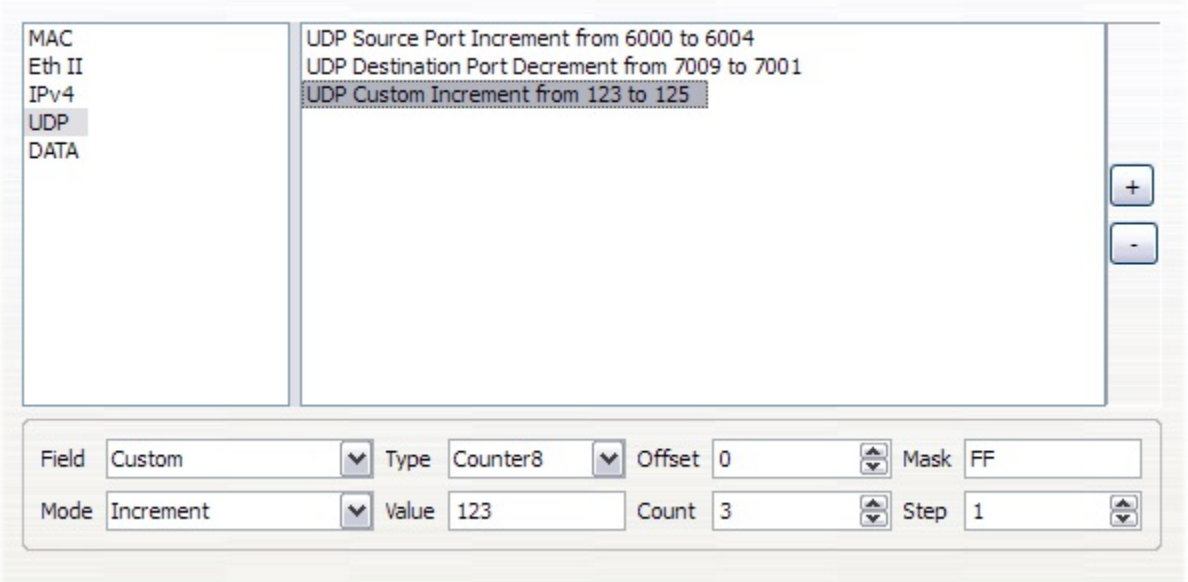


Click on the protocol name button to open up the configuration widget for the particular protocol and fill in values as desired.

NOTE: Currently "Protocol Id" fields for the implemented protocols cannot be modified (e.g. if you have Ethernet followed by IP, the Ethertype will be set to 0x0800 - you cannot change it)

Variable Fields

Some protocols allow configuring variable fields on the protocol data page for some fields (e.g. Mac/IP addresses). This page (available starting version 0.7) allows you to vary any field of any protocol, irrespective of whether you can do the same on the protocol data page or not.



Select the protocol, click the `'+'` button to add a variable field, select the variable field in the list and then specify which protocol field to vary and how. You can specify 'custom' fields also, if required.

The computation used for varying fields is as follows -

```
new = (old AND ~mask) OR ((value op n*step) AND mask)
```

where

- *old* is 8/16/32-bit value (depending on counter *type*) starting at *offset* bytes from start of protocol
- *op* is based on the mode - *increment/decrement/random*
- *n* varies from *0* to *(count-1)*
- *new* is the value that will replace *old*

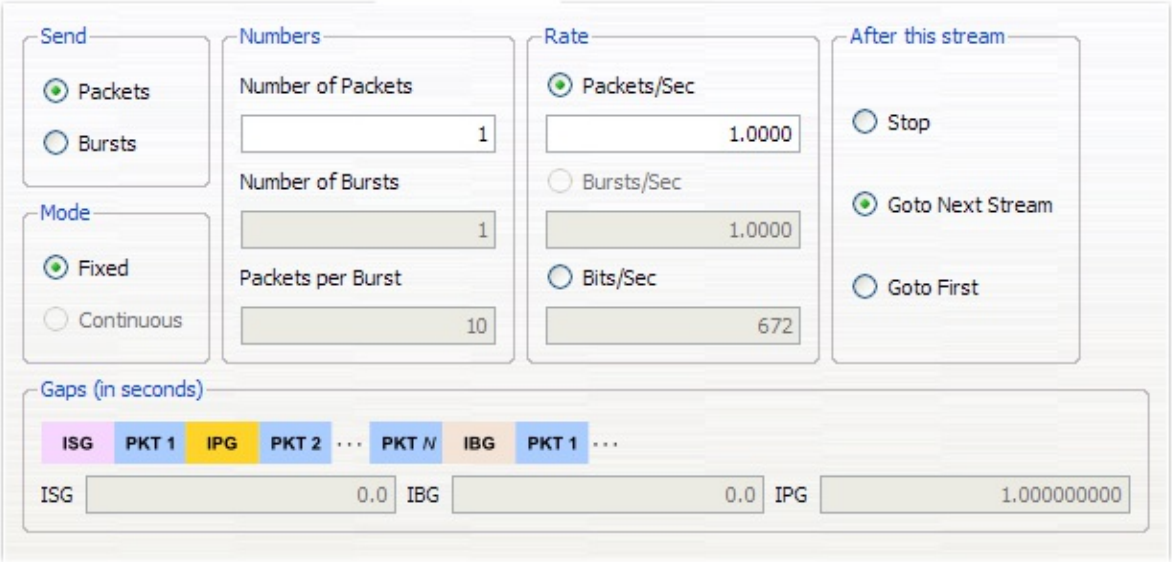
NOTE: The above computation has an implication for how value/step needs to be provided for fields which aren't aligned to the least significant bit of the counter type - in other words, fields which have trailing zeroes in the mask.

Consider the vlan header format (numbers in braces indicate field size in bits) -

```
+-----------------------+------+-----+-----------+
|         TPID          | Prio | CFI |  VLAN ID  |
|         (16)          |  (3) | (1) |    (12)   |
+-----------------------+------+-----+-----------+
```

Using a Counter8, the 3-bit priority field (possible values 0 to 7) is at byte offset 2 with a mask of 0xE0 - to vary this field between values 3 and 6, you will have to configure (initial) value as 96 (hex 0x60), count as 4 and step as 32 (hex 0x20).

Stream Control



On this page you can configure whether you send packets or bursts.

If you send packets, you can configure the number of packets to send and the pkts/sec rate at which to send the packets.

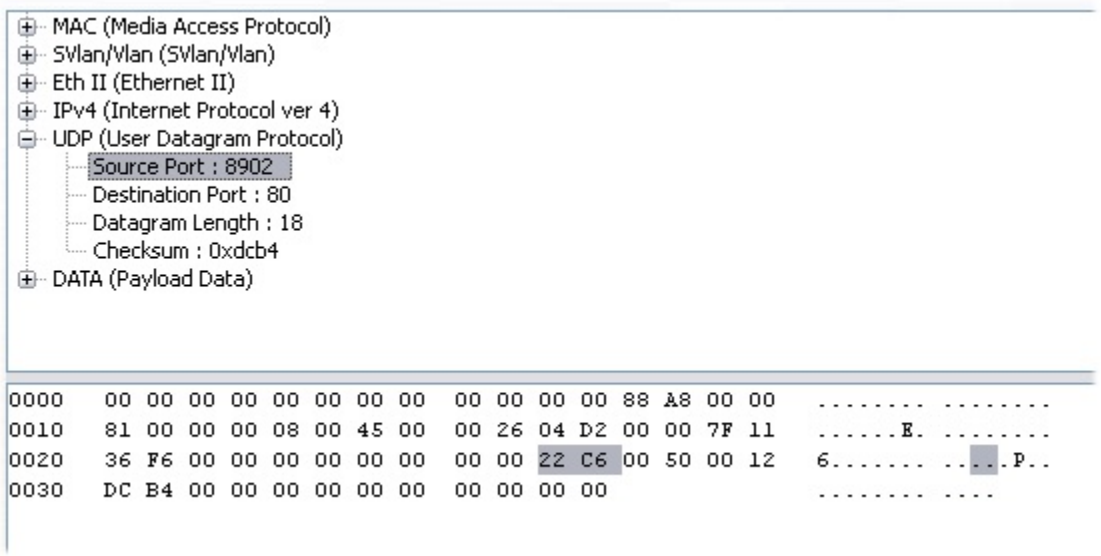
If you send bursts, you can configure the number of bursts to send and the bursts/sec rate at which to send the bursts; additionally you can configure the burst size in terms of packets per burst (all packets in a burst are sent back to back without any delay between them)

You can also configure the stream transmission order here. After this stream, you can either goto the next stream in the stream list order, or goto stream 0 (i.e. go back to the first stream in the stream list), or stop transmission (even if there are subsequent streams in the stream list)

NOTE: Depending upon the Transmit Mode, some fields may not be available

IMPORTANT: Ostinato tries its best to send the stream at the rate you have requested - but cannot guarantee it. The rates and accuracy achievable for a port depends on the computer running the controller component (drone). The computer's inherent processing capability and the current load on the computer are big factors that may affect the rates. See the FAQ "top-speed" question for achieving maximum transmit rate.

Packet View



On this page you can view the packet that you configured in the traditional fashion - tree view and hex dump.

Unimplemented

- If you have configured multiple packets and a varying packet field, you can only view the first of those packets
- You can click on any item in the tree view to highlight the corresponding bytes in the hexdump pane, but not vice-versa

Saving and Opening Stream Files

To open a stream file, first select the appropriate port in the ports list and then goto File | Open Streams (alternatively use the right-click context menu).

To save the streams configured on a port to a file, first select the appropriate port in the ports list and then goto File | Save Streams (alternatively use the right-click context menu).

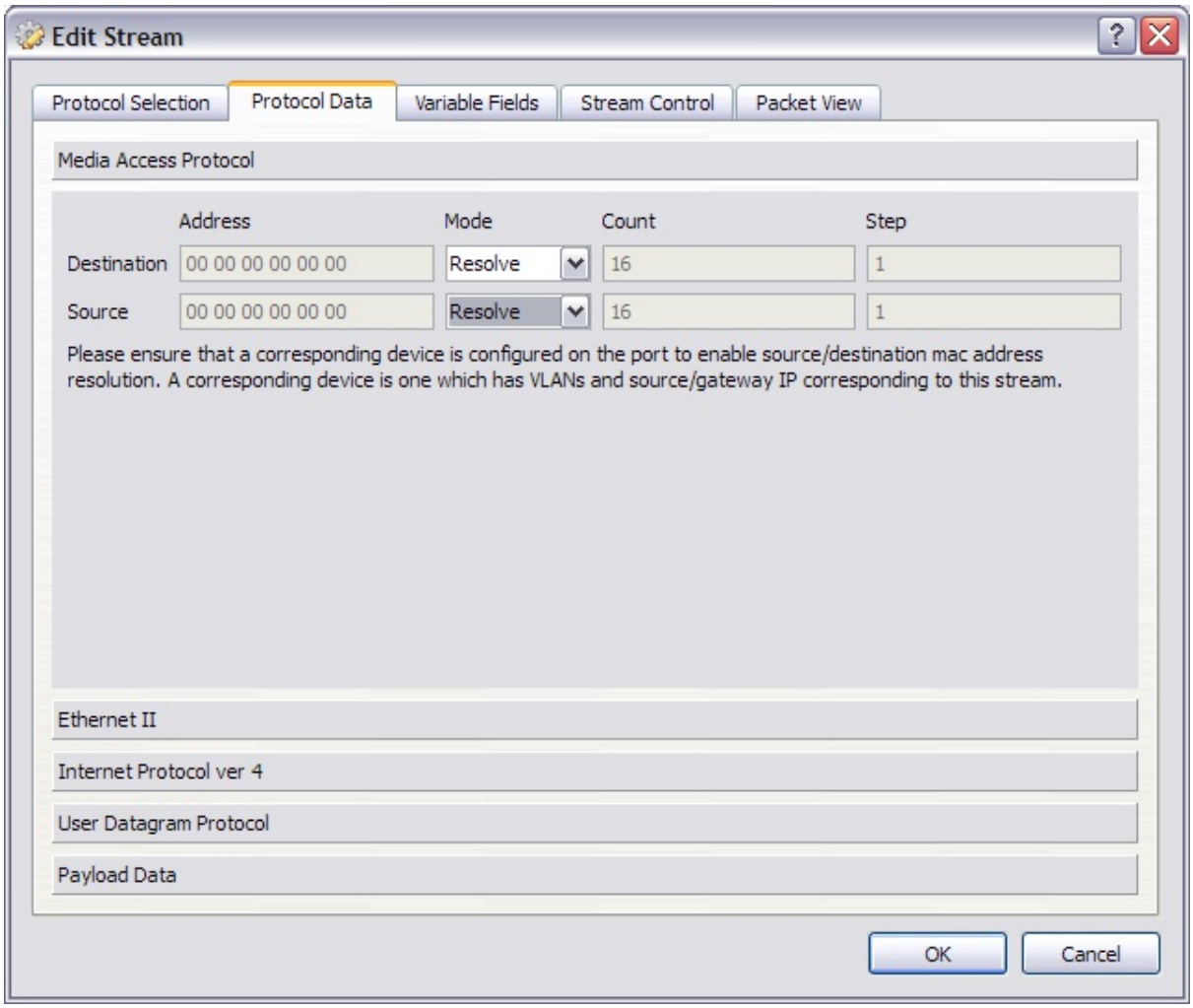
Ostinato has its own native file format.

Starting version 0.4, you can import/export PCAP and PDML files.

Starting version 0.7, you can export the configured streams as a python script for use with the Ostinato Python API (note: you cannot import a python script into the GUI, however). See Python API Guide for more details.

Device Emulation

Starting version 0.8, you can emulate multiple devices/hosts with ARP/NDP and ping support. Streams can be configured to resolve their MAC addresses using ARP/NDP if corresponding device(s) are configured.



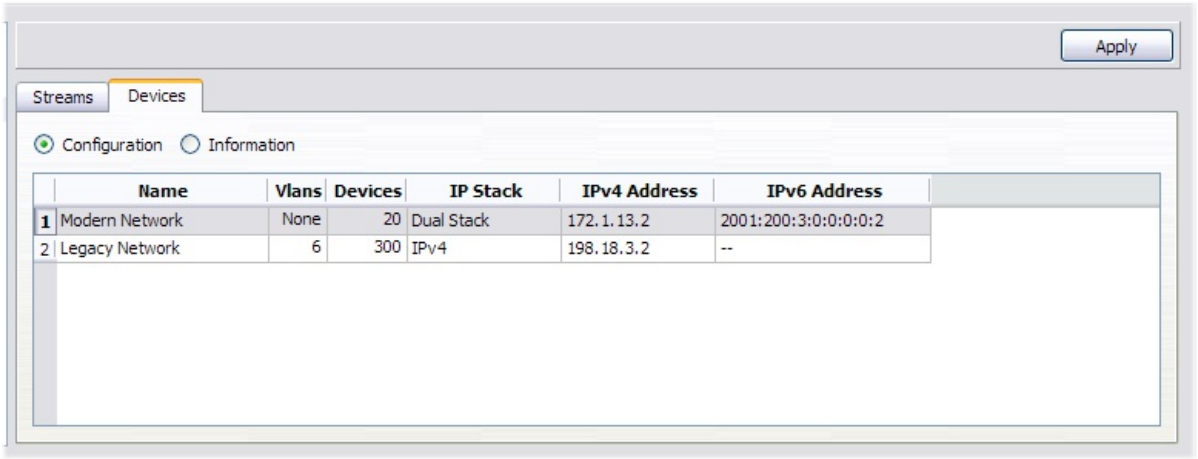
Before triggering stream transmit, invoke ARP/NDP resolution using the *Resolve Neighbors* button in the Statistics Window (see Statistic Window Actions). The controller will resolve all the Gateway IP addresses and all the destination IP addresses configured in each stream by sending a ARP/NDP request to the DUT. If ARP/NDP is not resolved (or failed), `00:00:00:00:00:00` will be used as the MAC address. You can check if all ARP/NDP are resolved in the Device Information pane.

In the reverse direction, if the DUT sends ARP/NDP request for any of the emulated device's IP address, the Controller will reply back with a suitable ARP/NDP reply. Similarly, if the DUT sends a ping (IPv4 or IPv6) request, it would reply with a ping reply.

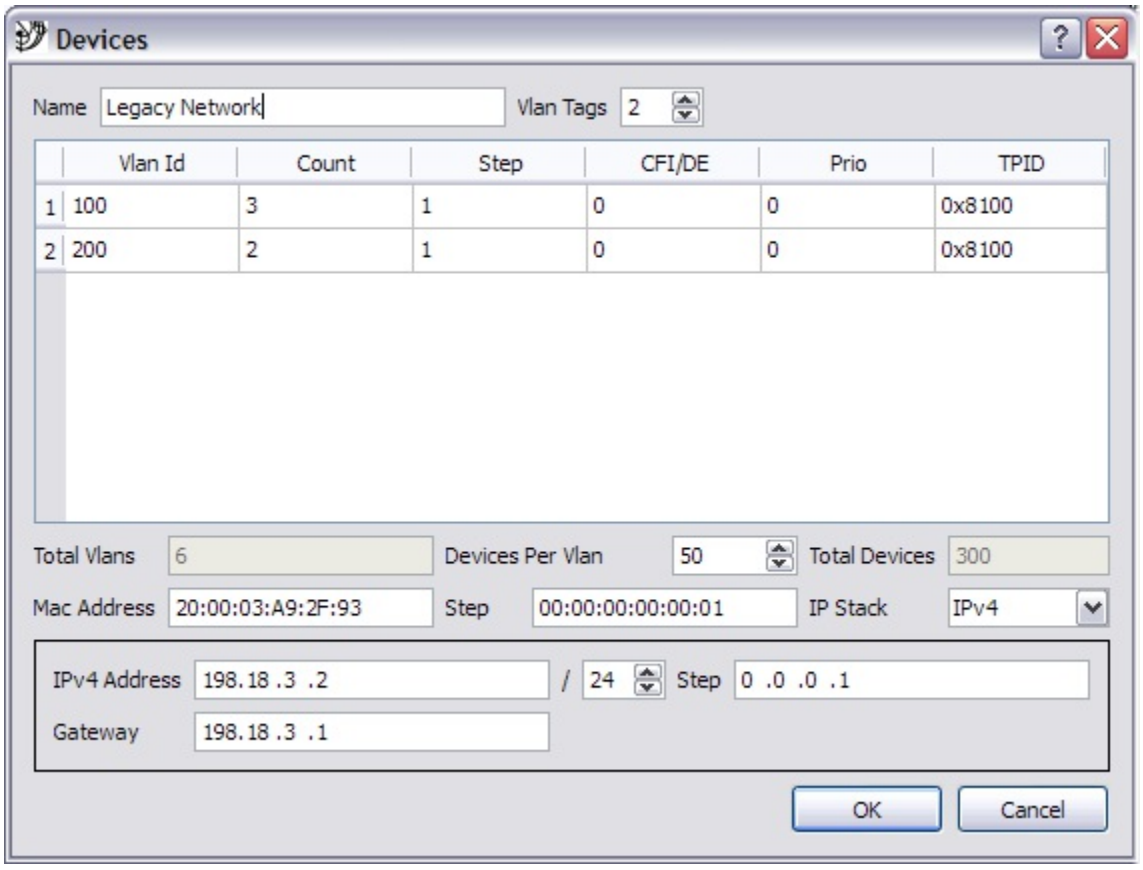
Device Configuration

Instead of configuring each device individually, you configure a device group and specify the number of devices in the group. This makes it easy to configure multiple devices by configuring only a single device group.

To create a device group, first select the appropriate port in the ports list and then goto File | New Device Group (alternatively, click the Devices Tab and use the right-click context menu).



To configure a device group's properties such as vlan, IP address etc, double-click the device group (or alternatively use the right-click context menu). You will be presented with the Device Configuration Dialog Box where you can configure the various attributes of the device group.



To delete a device group, select the device group(s) and then goto File | Delete Device Group (alternatively use the right-click context menu)

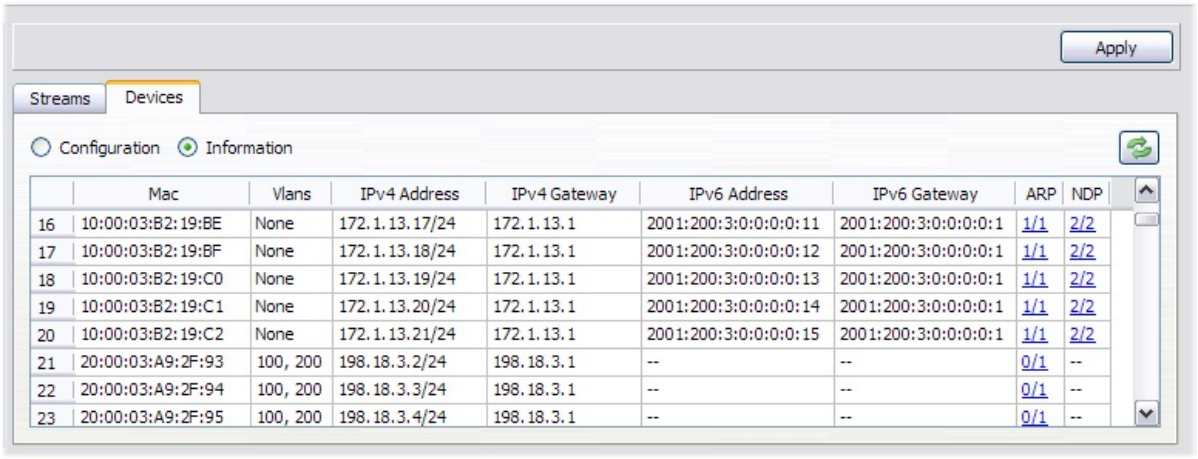
NOTE: Unlike streams, the order of device groups does not matter.

IMPORTANT: The agent sends device group configuration information to the controller only when you click the "Apply" button. So if you add/delete/edit device groups but forget to click "Apply", your changes won't be reflected.

To see how the configured device groups are expanded to multiple devices, see Device Information

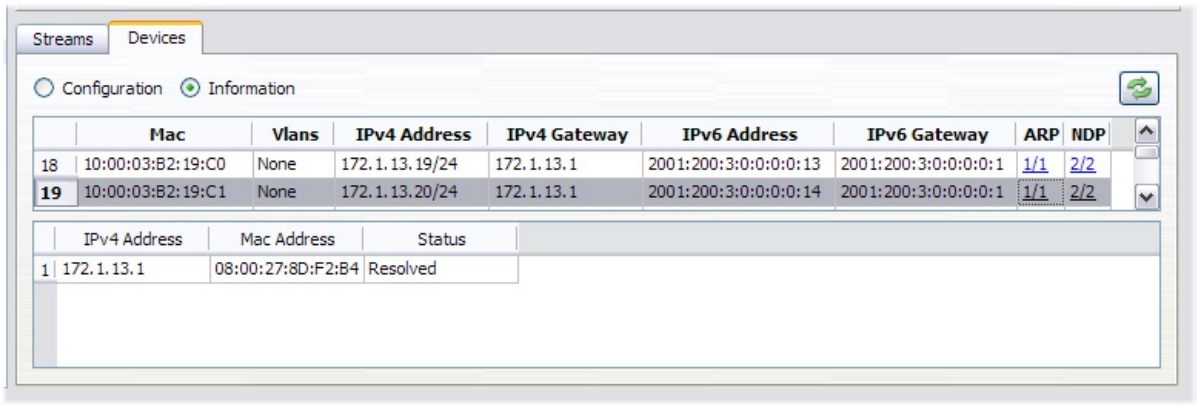
Device Information

Click on the *Information* radio button to see a list of expanded device groups with one device per row. This information is fetched from the controller, so make sure you click "Apply" if you don't see any information here.



	Mac	Vlans	IPv4 Address	IPv4 Gateway	IPv6 Address	IPv6 Gateway	ARP	NDP
16	10:00:03:B2:19:BE	None	172.1.13.17/24	172.1.13.1	2001:200:3:0:0:0:0:11	2001:200:3:0:0:0:0:1	1/1	2/2
17	10:00:03:B2:19:BF	None	172.1.13.18/24	172.1.13.1	2001:200:3:0:0:0:0:12	2001:200:3:0:0:0:0:1	1/1	2/2
18	10:00:03:B2:19:C0	None	172.1.13.19/24	172.1.13.1	2001:200:3:0:0:0:0:13	2001:200:3:0:0:0:0:1	1/1	2/2
19	10:00:03:B2:19:C1	None	172.1.13.20/24	172.1.13.1	2001:200:3:0:0:0:0:14	2001:200:3:0:0:0:0:1	1/1	2/2
20	10:00:03:B2:19:C2	None	172.1.13.21/24	172.1.13.1	2001:200:3:0:0:0:0:15	2001:200:3:0:0:0:0:1	1/1	2/2
21	20:00:03:A9:2F:93	100, 200	198.18.3.2/24	198.18.3.1	--	--	0/1	--
22	20:00:03:A9:2F:94	100, 200	198.18.3.3/24	198.18.3.1	--	--	0/1	--
23	20:00:03:A9:2F:95	100, 200	198.18.3.4/24	198.18.3.1	--	--	0/1	--

To view the ARP/NDP cache corresponding to each device, click on the ARP/NDP column corresponding to the device



	Mac	Vlans	IPv4 Address	IPv4 Gateway	IPv6 Address	IPv6 Gateway	ARP	NDP
18	10:00:03:B2:19:C0	None	172.1.13.19/24	172.1.13.1	2001:200:3:0:0:0:0:13	2001:200:3:0:0:0:0:1	1/1	2/2
19	10:00:03:B2:19:C1	None	172.1.13.20/24	172.1.13.1	2001:200:3:0:0:0:0:14	2001:200:3:0:0:0:0:1	1/1	2/2

	IPv4 Address	Mac Address	Status
1	172.1.13.1	08:00:27:8D:F2:B4	Resolved

Click on the *Refresh* button to fetch updated information from the Controller. Note that this just fetches information and does not trigger ARP/NDP resolution.

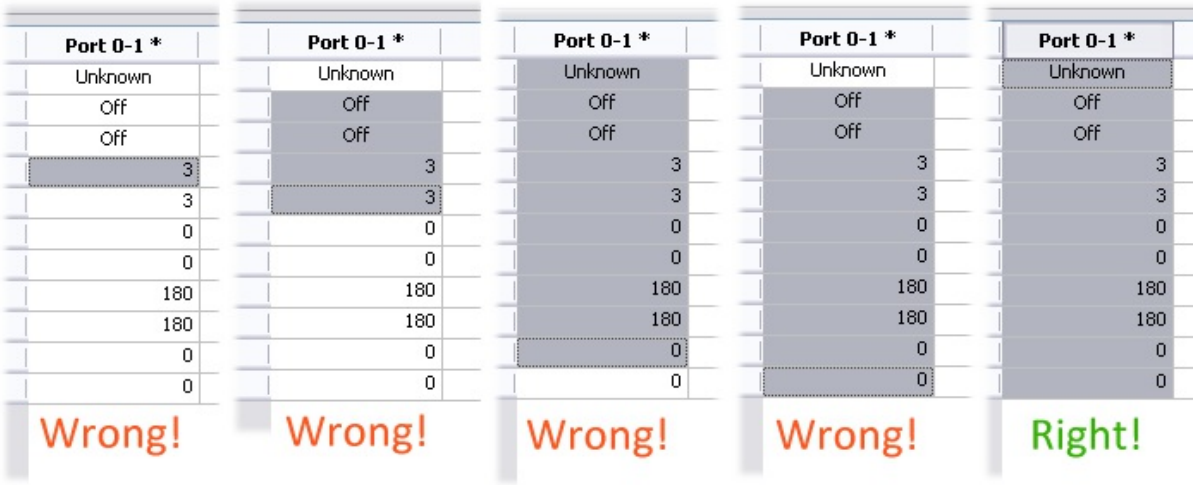
To trigger ARP/NDP resolution, use the *Resolve Neighbors* button in the Statistics Window (see Statistic Window Actions)

Statistics Window

The most important thing about using this window is that **ALL** the toolbar buttons (except `Clear All Stats`) operate on the port(s) selected in the statistics window itself (not the port selected in the ports list). If no ports are selected, no operation will be performed.

Release 0.9 onwards: To select a port, click anywhere in the port column.

Release 0.8 and older: A port is deemed to be "selected" only when the **full column** is selected - you can do that by clicking on the "heading row" of the port. See the image below for the right way to select a port -



Actions

Icon	Action	Description
	Start Transmit	Starts packet transmit on selected port(s)
	Stop Transmit	Stops packet transmit on selected port(s)
	Clear Stats	Clears statistics on selected port(s)
	Clear All Stats	Clears statistics on all displayed ports (*ver 0.6 and earlier*: all ports - displayed and hidden)
	Start Capture	Starts packet capture on selected port(s)
	Stop Capture	Stops packet capture on selected port(s)
	View Capture	View captured packets on selected port(s)
	Resolve Neighbors	Resolve Device Neighbors on selected port(s)
	Clear Neighbors	Clear Device Neighbors on selected port(s)
	Configure View	Select which port(s) to display in which order in the Statistics Window

Stream Statistics

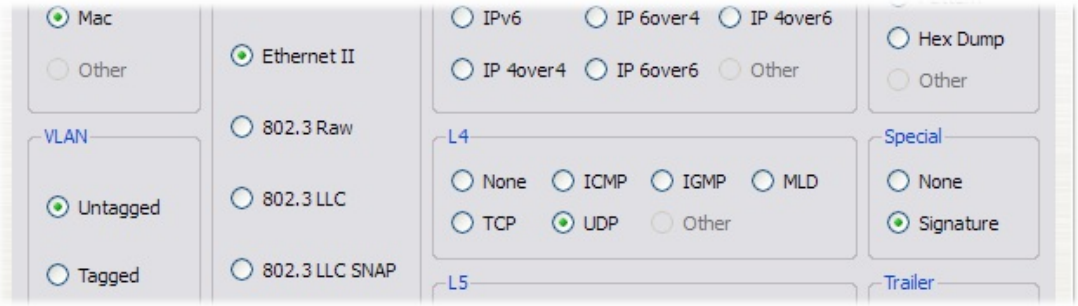
(*Available release 0.9 onwards*)

The Statistics Window displays interface or port level statistics and may include packets sent by other applications or the OS itself. This makes it difficult to compare whether received packet/byte count is same as sent count. Also there may be cases where you may be sending multiple streams and you would like to compare tx/rx counts for each stream.

For such cases, you can use Stream Statistics.

Configuration

- Enable Stream Statistics tracking on **both** Tx and Rx ports - note you can have multiple Tx/Rx ports
- For all streams that you wish to track statistics for -
  - Include the `Special|Signature` protocol on the *Protocol Selection* tab



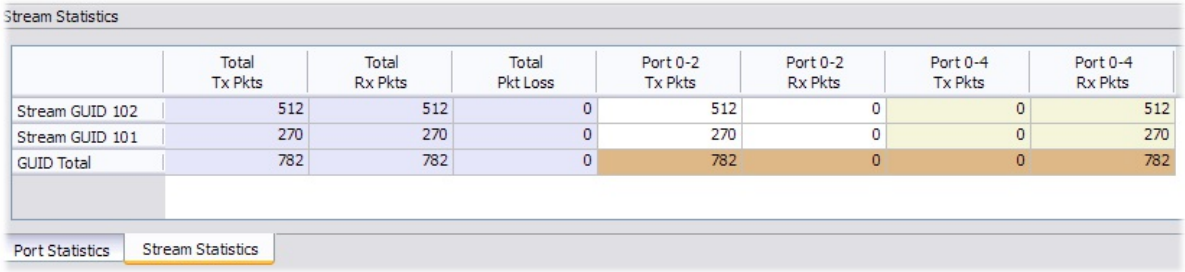
  - Configure a unique value as the Stream GUID in *Protocol Data* tab

NOTE: A port with *Stream Statistics Tracking* enabled can have streams with the *Special Signature* and streams without it - stream level statistics will be tracked only for the former.

Display

- Ensure Transmit is finished i.e. *Transmit State* is `Off`

- Select all the Tx/Rx ports for which you wish to fetch and view stream statistics
- Click on the  *Fetch selected port stream stats* button
- The stream statistics will open in a new tab in the Statistics Window



Limitations

- Tracking stream level statistics is CPU intensive and may have an impact on max transmit rate
- ICMP streams cannot be tracked
- Tx Stream Statistics is counted only after transmit is finished. Fetching stats before transmit is finished will return 0 values

How does it work

The signature protocol is added to the end of the Ethernet frame just before the FCS. The last 4 bytes are a magic value indicating that the frame includes a special signature containing the Stream GUID. The transmitted frames will contain this special signature.

For a port with *Stream Statistics* tracking enabled -

- All received packets are captured and parsed to check for the special signature and if present, further parsed for the Stream GUID to count the Rx stats
- Since all frames are pre-generated in Ostinato, at the end of transmission we can calculate how many packets of each stream was transmitted to get the Tx stats - this avoids the costly capture/parsing required for Rx stats, but introduces the limitation that Tx stats are not available until end of transmission

How to write a userscript protocol

Introduction

For unimplemented protocols (or implemented protocols not meeting your specific needs) you can write a script to simulate it.

Scripts are written in `QtScript` (which is very similar to `JavaScript` - both based on ECMA Script)

Documentation

Two objects are available to the script - `protocol` and `Protocol` (note the difference in case). The script uses these two objects to define the protocol.

To define your protocol, you need to define the following properties of the `protocol` object -

NOTE: Some of the functions take an `index` param. `index` is incremented by 1 for every packet at runtime - if you would like to vary your protocol size/contents at runtime you can use it, otherwise just ignore it

Property	Type	Required?	Default Value	Description
`protocolFrameSize(index)`	Function	Mandatory	--	function returning the size of the protocol header in bytes
`protocolFrameValue(index)`	Function	Mandatory	--	function returning an array containing the protocol header bytes; the size of the array should be same as what `protocolFrameSize()` returns;
`name`	String	Optional	"" (Empty String)	String identifying the protocol
`protocolId(type)`	Function	Optional	0	function returning the protocol id of your protocol - type is one of `ProtocolIdLlc` , `ProtocolIdEth` , `ProtocolIdIp` or `ProtocolIdTcpUdp` ; if your protocol immediately follows a LLC/Eth/IP/TCP/UDP Header, they will use the value that you return to fill-in their `ProtocolId` fields, if you don't define this function or don't provide a value, 0 will be used
`protocolFrameCksum(index, type)`	Function	Optional	Calculated checksum as per the CksumType	function returning the cksum of type for the protocol header; type is one of `CksumIp` , `CksumIpPseudo` , `CksumTcpUdp`
`protocolFrameValueVariable`	Boolean	Optional	False	(*Deprecated since version 0.7*) boolean indicating whether the protocol contents vary at run time with

				every packet; default value is false - set to true if required
`protocolFrameSizeVariable`	Boolean	Optional	False	boolean indicating whether the protocol header size varies at run time with every packet; default value is false - set to true if required
`protocolFrameVariableCount`	Integer	Optional	1	integer specifying the minimum number of frames required to vary its fields/size before the values are repeated again; default value is 1 indicating the protocol does not vary its fields with every packet

The `protocol` object provides some predefined functions that you can use in your functions -

Function	Description
`payloadProtocolId(type)`	returns the protocol id of the subsequent protocol - use when your protocol has a "Protocol Id" field that you need to fill-in based on what protocol follows yours
`protocolFrameOffset(index)`	returns the byte offset within the frame from which your protocol will start
`protocolFramePayloadSize(index)`	returns the cumulative size of all subsequent protocols and the payload - use if your protocol has a "length" or "payloadLength" field that you need to fill-in
`isProtocolFramePayloadValueVariable`	returns a boolean indicating if the protocols that appear as payload to this protocol have varying fields; if your protocol has some fields dependent on the payload, you can use this function to determine if your protocol may need to vary its fields accordingly
`isProtocolFramePayloadSizeVariable`	returns a boolean indicating if the protocols that appear as payload to this protocol have a varying size; if your protocol has some fields dependent on the payload size, you can use this function to determine if your protocol may need to vary its fields accordingly
`protocolFramePayloadVariableCount`	returns the minimum number of frames required by the payload protocols to vary their fields; if your protocol has some fields dependent on the payload, you can use this function to determine your protocol's `protocolFrameVariableCount`
`protocolFrameHeaderCksum(index, cksumType)`	returns the cumulative checksum of all protocol headers that appear before your protocol - useful for protocols like TCP/UDP which need a Pseudo-Ip Checksum of the preceding IP protocol; if `cksumType` is not specified, it defaults to `CksumIp`
`protocolFramePayloadCksum(index, cksumType)`	returns the cumulative checksum of all protocol headers and payloads that appear after your protocol- use if your protocol has a "checksum" field that needs the checksum of the payload also; if `cksumType` is not specified, it defaults to `CksumIp`

The `Protocol` object is a Read-Only object providing some convenient enums as properties -

```
ProtocolIdLlc
ProtocolIdEth
ProtocolIdIp
ProtocolIdTcpUdp
```

```
CksumIp
CksumIpPseudo
CksumTcpUdp
```

Examples

Here are some example scripts. You can search in the mailing list archives for more.

ICMP

Here`s an example implementing ICMP -

```
protocol.protocolFrameSize = function() { return 8; }
protocol.protocolFrameValue = function(index)
{
    var type = 8; // Echo Request
    var code = 0;
    var id  = 0x9127; // example value
    var seq = 0x1234; // example value
    var sum = (type << 8 | code) + id + seq + (0xFFFF & ~protocol.protocolFramePayloadCksum());

    var pfv = new Array(8);

    pfv[0] = type;
    pfv[1] = code;

    while(sum>>16)
        sum = (sum & 0xFFFF) + (sum >> 16);

    sum = ~sum;
    pfv[2] = sum >> 8;
    pfv[3] = sum & 0xFF;

    pfv[4] = id >> 8;
    pfv[5] = id & 0xFF;

    pfv[6] = seq >> 8;
    pfv[7] = seq & 0xFF;

    return pfv;
}
protocol.protocolId = function() { return 0x1; }
```

IPv6

Here`s a slightly more involved example for IPv6 (a protocol that is not implemented as of this writing)

```
protocol.name = "IPv6"

protocol.protocolFrameSize = function() { return 40; }

protocol.protocolId = function(id_type)
{
    if (id_type == Protocol.ProtocolIdEth)
        return 0x86dd;
```

```
    if (id_type == Protocol.ProtocolIdIp)
        return 0x29;

    return 0;
}


protocol.protocolFrameValue = function(index)
{
    var len;
    var pfv = new Array(40);

    // ip version = 6
    pfv[0] = 0x60;

    // payload length
    len = protocol.protocolFramePayloadSize(index);
    pfv[4] = len >> 8;
    pfv[5] = len & 0xFF;

    // Fill-in other fields as required

    // NextHeader
    pfv[6] = protocol.payloadProtocolId(Protocol.ProtocolIdIp);
    pfv[7] = 64; // HopLimit

    return pfv;
}

protocol.protocolFrameCksum = function(index, type)
{
    var sum = 0;

    if (type == Protocol.CksumIpPseudo) {
        sum += protocol.protocolFramePayloadSize(index);
        sum += protocol.payloadProtocolId(Protocol.ProtocolIdIp);
    }
    return ~sum;
}
```

VxLAN

Inficone has a blog post about writing a userscript for VxLAN

Save and Restore a Session

Starting version 0.8, you can save the entire session from the GUI to a file. A session includes all the currently connected port groups, all the ports within each portgroup, all the streams and device groups within each port. However, if one or more ports (in any port group) is reserved, only ports reserved by the current user will be saved. To save a session, goto File | Save Session.

A saved session file can be opened to restore the session. The GUI will add all the port groups saved in the session file, connect to each port group, restore the saved ports and their streams and device groups from the file. This will overwrite any previous configuration on that port. However, if a port is currently reserved by another user, it will not be overwritten with the new configuration from the file. A port not saved in the file will also retain its configuration as before opening the session file. To restore a session, goto File | Open Session.

Ostinato

`TODO`

Drone

(*Drone settings are available release 0.7 onwards*)

Drone settings are specified in a `.ini` file. This file is read only once at startup and never written to by the application (i.e. it is expected to be user created and edited). It is located at one of the below locations (searched in given order) -

On Unix and Mac OS X -

1. `<path-to-drone-executable>/drone.ini`
2. `$HOME/.config/Ostinato/drone.ini`
3. `$HOME/.config/Ostinato.ini`
4. `/etc/xdg/Ostinato/drone.ini`
5. `/etc/xdg/Ostinato.ini`

On Windows -

1. `<path-to-drone-executable>/drone.ini`
2. `%APPDATA%\Ostinato\drone.ini`
3. `%APPDATA%\Ostinato.ini`
4. `%COMMON_APPDATA%\Ostinato\Ostinato.ini`
5. `%COMMON_APPDATA%\Ostinato.ini`

The `%APPDATA%` path is usually `C:\Documents and Settings\User Name\Application Data`; the `%COMMON_APPDATA%` path is usually `C:\Documents and Settings\All Users\Application Data`.

The .ini file contents are **case-sensitive** and the format is -

```
[General]
RateAccuracy=
...

[RpcServer]
Address=

[PortList]
Include=
Exclude=
```

General

RateAccuracy

(*Available release 0.8 onwards*)

To ensure that the actual transmit rate is as close as possible to the configured transmit rate, Drone runs a busy-wait loop. While this provides the maximum accuracy possible, the CPU utilization is 100% while the transmit is on. You can however, sacrifice the accuracy to reduce the CPU load.

Supported values currently are `High` (default), `Low`

RpcServer

Address

(*Available release 0.8 onwards*)

By default, the Drone RPC server will listen on all interfaces and local IPv4 adresses for incoming connections from clients. Specify a single IPv4 or IPv6 address if you want to restrict that. To listen on *any* IPv6 address, use `::`

PortList

Use the PortList Include/Exclude list to filter the list of ports that drone manages. Both Include and Exclude are comma-separated glob patterns which match port names (in case of Windows, port description instead of port name is matched). For a port to pass the filter and appear on the port list managed by drone, it should be allowed by the Include list and not disallowed by the Exclude List. An empty Include list matches all ports (i.e. all ports are allowed). An empty Exclude list matches no ports (i.e. no ports are disallowed)

e.g. to filter out usbmon ports -

```
[PortList]
Include=
Exclude=usbmon*
```

or e.g. to have drone work on only eth ports and loopback ports but not on eth0 -

```
[PortList]
Include=eth*, lo*
Exclude=eth0
```

Command Line Options

Ostinato

(*Available release 0.9 onwards*)

```
ostinato [-c] [<session-file>]
```

Options

Option	Required?	Description
-c	Optional	Controller only - don't start local Drone and the corresponding 127.0.0.1 portgroup

Arguments

Argument	Required?	Description
session-file	Optional	Open the session file at startup

Drone

TODO

Starting from version 0.6, Ostinato provides python bindings to enable scripting support.

See Python API Guide for download, installation, usage and reference information.