TRex Virtual Machine setup and basic usage

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Introduction

1.1 TRex traffic generator

TRex traffic generator is a tool designed to benchmark platforms using realistic traffic.

One of the tools through which TRex can be learned and tested is a virtual machine instance or Docker, fully simulating TRex without the need for any additional hardware.

Chapter 2

TRex inside Docker (from v2.37 and up)

With a few commands you can run TRex inside docker with low performance/low footprint. You will need to have docker installed on your system with privileges. We are using in this example RedHat 7.4

Docker hub is located docker hub

Pull the trex image

```
[bash]>docker pull trexcisco/trex                                    #❶
[bash]>docker run --rm -it --privileged --cap-add=ALL trexcisco/trex #❷
[bash]root@d8ec7a3a09d9 v2.36]./t-rex-64 -i                          #❸
```

❶ Pull the trex image could be with higher version

❷ Get into the docker shell

❸ Run TRex in Stateless mode

From **another** terminal

Run Stateless Console

```
[bash]>docker ps
CONTAINER ID        IMAGE                  COMMAND
*d4f956743f62*      trexcisco/trex:2.36    "/bin/bash"      #❶

[bash]>docker exec -it d4f956743f62 bash                     #❷
[bash]root@d4f956743f62 v2.36]./trex-console                 #❸
```

❶ Pull the trex image could be with higher version

❷ Insert to the docker container (take the CONTAINER ID from $1)

❸ Run stateless Console

From TRex Console

```
[root@d8ec7a3a09d9 v2.36]./trex-console

Using 'python' as Python interpeter


Connecting to RPC server on localhost:4501                   [SUCCESS]
```

```
Connecting to publisher server on localhost:4500             [SUCCESS]


Acquiring ports [0, 1]:                                      [SUCCESS]


Server Info:

Server version:   v2.36 @ STL
Server mode:      Stateless
Server CPU:       1 x Intel(R) Xeon(R) CPU E5-2667 v3 @ 3.20GHz
Ports count:      2 x 10Gbps @ Unknown

 -=TRex Console v2.0=-

Type 'help' or '?' for supported actions

trex>start -f stl/imix.py -m 10kpps --port 0          #❶

trex>tui                                              #❷

  Global Statistics

  connection   : localhost, Port 4501                 total_tx_L2  : 28.90 Mb/sec
  version      : v2.36                                total_tx_L1  : 30.50 Mb/sec
  cpu_util.    : 1.35% @ 1 cores (1 per port)         total_rx     : 28.90 Mb/sec
  rx_cpu_util. : 0.0% / 9.99 Kpkt/sec                 total_pps    : 9.99 Kpkt/sec
  async_util.  : 0.41% / 1.28 KB/sec                  drop_rate    : 0.00 b/sec
                                                      queue_full   : 0 pkts

  Port Statistics

   port    |         0         |         1         |       total
 ----------------------------------------------------------------------
 owner      |              root |              root |
 link       |                UP |                UP |
 state      |      TRANSMITTING |              IDLE |
 speed      |           10 Gb/s |           10 Gb/s |
 CPU util.  |             1.35% |              0.0% |
 --         |                   |                   |
 Tx bps L2  |        28.90 Mbps |          0.00 bps |        28.90 Mbps
 Tx bps L1  |        30.50 Mbps |             0 bps |        30.50 Mbps
 Tx pps     |         9.99 Kpps |          0.00 pps |         9.99 Kpps
 Line Util. |            0.31 % |            0.00 % |
 ---        |                   |                   |
 Rx bps     |          0.00 bps |        28.90 Mbps |        28.90 Mbps
 Rx pps     |          0.00 pps |         9.99 Kpps |         9.99 Kpps
 ----       |                   |                   |
 opackets   |            189282 |                 0 |            189282
 ipackets   |                 0 |            189282 |            189282
 obytes     |          68489264 |                 0 |          68489264
 ibytes     |                 0 |          68489264 |          68489264
 opackets   |      189.28 Kpkts |            0 pkts |      189.28 Kpkts
 ipackets   |            0 pkts |      189.28 Kpkts |      189.28 Kpkts
 obytes     |          68.49 MB |               0 B |          68.49 MB
 ibytes     |               0 B |          68.49 MB |          68.49 MB
 -----      |                   |                   |
 oerrors    |                 0 |                 0 |                 0
 ierrors    |                 0 |                 0 |                 0

 status:  \
```

```
Press 'ESC' for navigation
```

1. Start traffic on port 0 (imix profile)
2. Show the stats

TRex configuration file /etc/trex_cfg.yaml

```
- port_limit     : 2
  version        : 2
  low_end        : true                  #1
  interfaces     : ["veth0", "veth1"]    #2
  port_info      :  # set eh mac addr
                 - ip         : 1.1.1.1
                   default_gw : 2.2.2.2
                 - ip         : 2.2.2.2
                   default_gw : 1.1.1.1
```

1. low-footprint mode (require less resource), use one thread
2. use veth0/veth1 as DP ports conntect to an internal switch

---

Note

You will need to redirect the folowing ports 4500/4501/4507 to use the external GUI for that you can use this command `[bash]>docker run --rm -it --privileged --cap-add=ALL -p 4500:4500 -p 4501:4501 -p 4507:4507 trexcisco/trex`

Download the GUI from GitHub trex-gui-release

Chapter 3

TRex inside Virtual Box

The TRex Virtual Machine is based on Oracle's Virtual Box freeware. It is designed to enable TRex newbies to explore this tool without any special resources.

3.1 Setup and Usage

3.1.1 Setup

In order to use TRex VM, there are several easy steps to follow:

1. Download and install Oracle VM Virtual Box Manager (VB download link).
   During installation you will be asked to allow the installation of system devices component interactions. Allow it.
2. Download the latest TRex VM by clicking on this link. Notice that this is the latest VM image, not the latest TRex version. This can be used for demonstration purposes. After installation, you can upgrade to latest TRex image if needed (Instructions below).
3. Open Oracle VM Virtual Box application installed at step 1.
4. Under *File* tab, select *Import Appliance*. The following screen will apear:

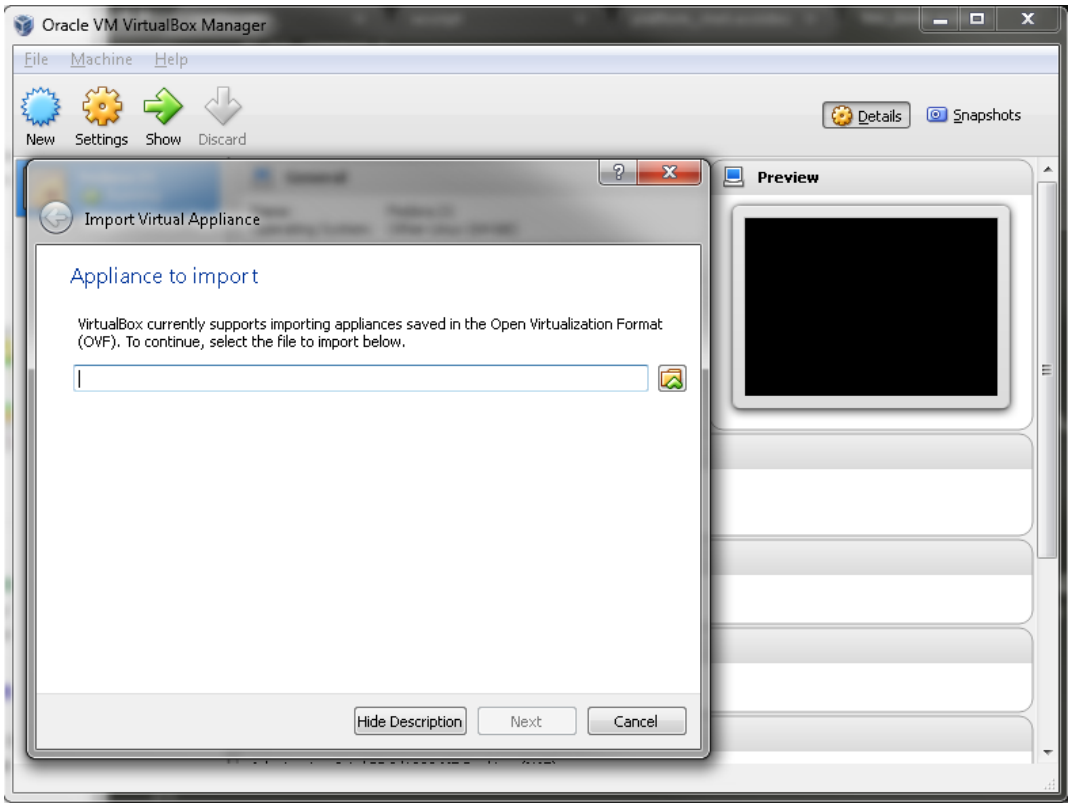

Figure 3.1: VM import screen

1. Browse and select the .ova file you have downloaded at step 2, and click *continue*.
2. Click *Next, and then make sure that the 'Reinitialize the MAC address of all network cards* checkbox is **not selected**.
3. Click *import* and wait for the import process to finish. **That's it! you're all good and set to go!**

3.1.2 Launching and logging into the machine

First, launch the virtual machine by selecting it in the VM's menu and hitting *Start* button.

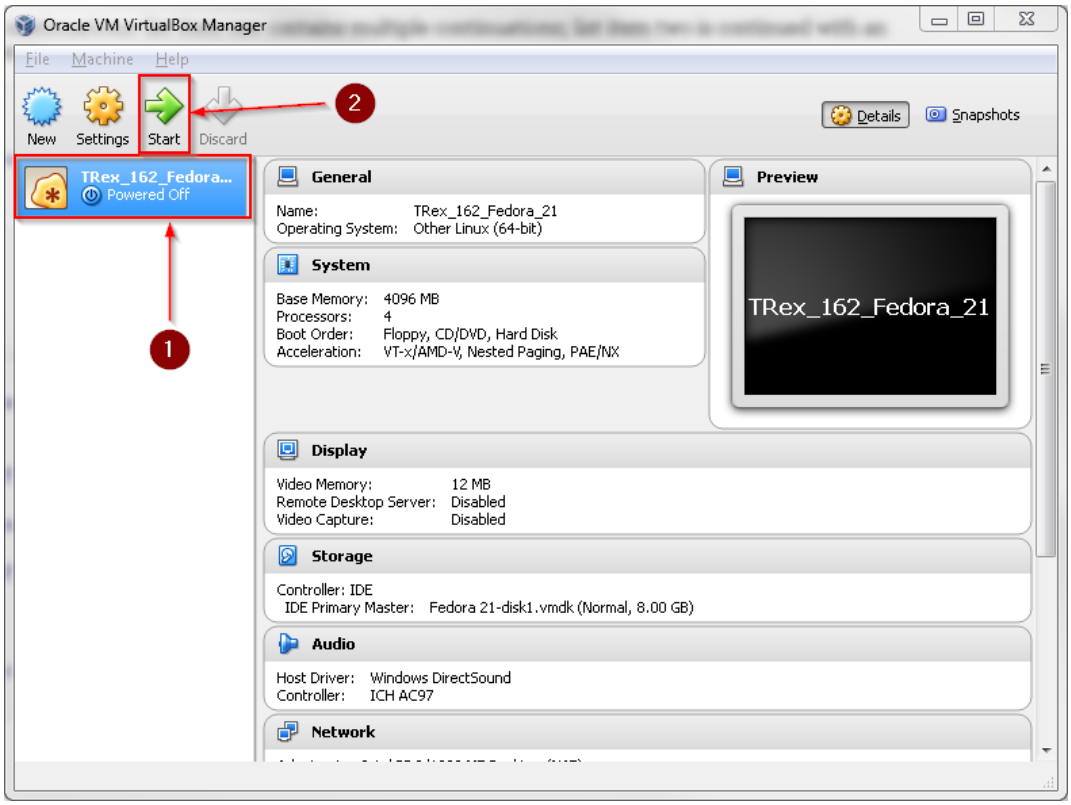

Figure 3.2: TRex VM launching screen

Important
You may encounter "VT-x is disabled" error, as shown in the image below.
In that case, please refer to this link and follow the provided steps to overcome this issue.

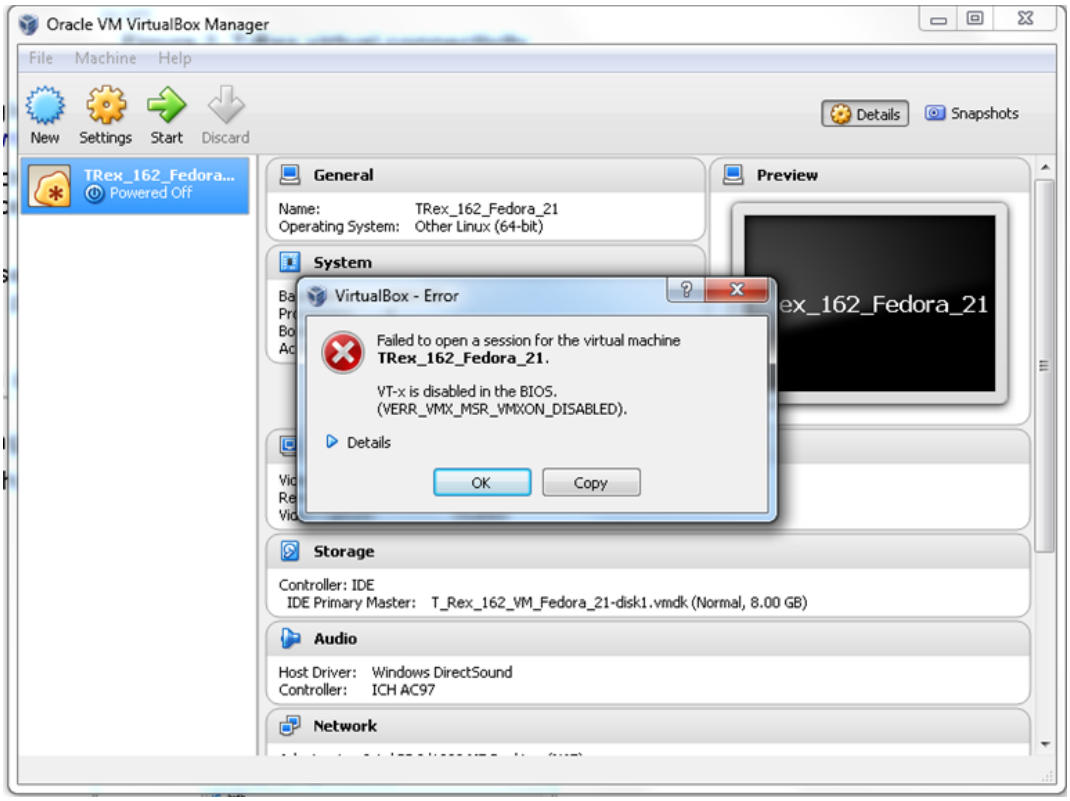

Figure 3.3: VT-x disabled possible error message

Once the machine finished booting, login to it using the following credentials:

- Username: `trex`
- Password: `trex`

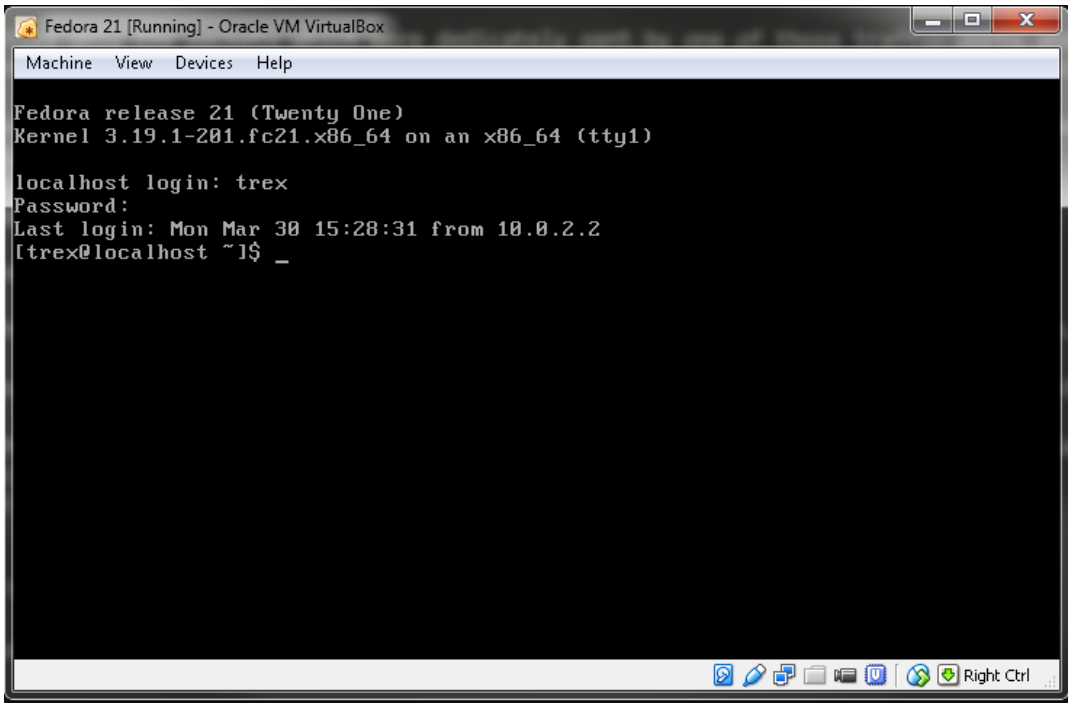

Figure 3.4: TRex VM login

Tip
Remote connection to the machine from anywhere in the hosting machine can be set up using the following command:
`ssh -p 3022 trex@127.0.0.1`

3.1.3 Running TRex traffic generator

1. **upgrade** to the latest trex package see how to upgrade
2. Change dir to latest version updated for example `cd /home/trex/v2.20/`. (don't use the old version in the OVA)
3. Run your desired TRex command.

Important
When launching a TRex command, pay attention to make sure that you are using `sudo` prefix at the beggining of the command line. + and that you updated to latest v1.62 is very old version

For example, let's run TRex with DNS traffic. The command is:

```
[trex@localhost v1.62]$ sudo ./t-rex-64 -f cap2/dns.yaml -d 100 -m 1 --nc
Starting  TRex 1.62 please wait  ...
found configuration file at /etc/trex_cfg.yaml
```

```
zmq publisher at: tcp://*:4500

...
❶
...

-Per port stats table
      ports |               0 |               1
 -----------------------------------------------------------------------------------------
   opackets |              17 |              17
     obytes |            1241 |            1513
   ipackets |              17 |              17
     ibytes |            1513 |            1241
    ierrors |               0 |               0
    oerrors |               0 |               0
      Tx Bw |     582.35  bps |     709.99  bps

-Global stats enabled
 Cpu Utilization : 0.8  %  0.0 Gb/core
 Platform_factor : 1.0
 Total-Tx        :       1.29 Kbps
 Total-Rx        :       1.29 Kbps
 Total-PPS       :       1.99  pps
 Total-CPS       :       1.00  cps

 Expected-PPS    :       2.00  pps
 Expected-CPS    :       1.00  cps
 Expected-BPS    :       1.30 Kbps

 Active-flows    :        0  Clients :      511   Socket-util : 0.0001 %
 Open-flows      :       17  Servers :      255   Socket :       17 Socket/Clients :  0.0
 drop-rate       :       0.00  bps
 current time    : 18.7 sec
 test duration   : 81.3 sec
```

❶ Output trimmed.

Now, lets review the generated packets as they are observed by our promiscuous interface (interface #2 in the picture at the bottom).
Notice that (depending on your virtual box CPU performance), tcpdump output might be delayed.

```
[trex@localhost ~]$ sudo tcpdump -i enp0s8
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp0s8, link-type EN10MB (Ethernet), capture size 262144 bytes
09:38:53.953651 IP 16.0.0.2.1024 > 48.0.0.2.domain: 48 A? www.cisco.com. (31)
09:38:53.963969 IP 48.0.0.2.domain > 16.0.0.2.1024: 48* 1/0/0 A 100.100.100.100 (47)
09:38:54.960361 IP 16.0.0.3.1024 > 48.0.0.3.domain: 48 A? www.cisco.com. (31)
09:38:54.970358 IP 48.0.0.3.domain > 16.0.0.3.1024: 48* 1/0/0 A 100.100.100.100 (47)
09:38:55.967200 IP 16.0.0.4.1024 > 48.0.0.4.domain: 48 A? www.cisco.com. (31)
09:38:55.977222 IP 48.0.0.4.domain > 16.0.0.4.1024: 48* 1/0/0 A 100.100.100.100 (47)
09:38:56.975355 IP 16.0.0.5.1024 > 48.0.0.5.domain: 48 A? www.cisco.com. (31)
09:38:56.985379 IP 48.0.0.5.domain > 16.0.0.5.1024: 48* 1/0/0 A 100.100.100.100 (47)
09:38:57.981659 IP 16.0.0.6.1024 > 48.0.0.6.domain: 48 A? www.cisco.com. (31)
09:38:57.992358 IP 48.0.0.6.domain > 16.0.0.6.1024: 48* 1/0/0 A 100.100.100.100 (47)
09:38:58.990979 IP 16.0.0.7.1024 > 48.0.0.7.domain: 48 A? www.cisco.com. (31)
09:38:59.000952 IP 48.0.0.7.domain > 16.0.0.7.1024: 48* 1/0/0 A 100.100.100.100 (47)
09:39:00.009403 IP 16.0.0.8.1024 > 48.0.0.8.domain: 48 A? www.cisco.com. (31)
09:39:00.019456 IP 48.0.0.8.domain > 16.0.0.8.1024: 48* 1/0/0 A 100.100.100.100 (47)
09:39:01.015810 IP 16.0.0.9.1024 > 48.0.0.9.domain: 48 A? www.cisco.com. (31)
```

Let's have a look at another example.
We want to generate simple http traffic. The command will look like:

```
[trex@localhost v1.62]$ sudo ./t-rex-64 -f cap2/http_simple.yaml -d 100 -l 1000 -m 1 --nc
Starting  TRex 1.62 please wait  ...
found configuration file at /etc/trex_cfg.yaml
zmq publisher at: tcp://*:4500

...
(1)
...

-Per port stats table
      ports |               0 |               1
 -----------------------------------------------------------------------------------------
   opackets |           40983 |           41946
     obytes |         2563951 |         6015664
   ipackets |           41946 |           40983
     ibytes |         6015664 |         2563951
    ierrors |               0 |               0
    oerrors |               0 |               0
      Tx Bw |     520.83 Kbps |       1.27 Mbps

-Global stats enabled
 Cpu Utilization : 3.1  %  0.1 Gb/core
 Platform_factor : 1.0
 Total-Tx        :       1.79 Mbps
 Total-Rx        :       1.79 Mbps
 Total-PPS       :       2.11 Kpps
 Total-CPS       :       2.84  cps

 Expected-PPS    :     102.71  pps
 Expected-CPS    :       2.78  cps
 Expected-BPS    :     764.51 Kbps

 Active-flows    :        0  Clients :      255   Socket-util : 0.0000 %
 Open-flows      :      107  Servers :    65535   Socket :        0 Socket/Clients :  0.0
 drop-rate       :       0.00  bps
 current time    : 39.6 sec
 test duration   : 60.4 sec

-Latency stats enabled
 Cpu Utilization : 1.0 %
 if|   tx_ok , rx_ok  , rx   ,error,    average   ,   max         , Jitter ,  max window
   |         ,        , check,     , latency(usec),latency (usec) ,(usec)  ,
 ----------------------------------------------------------------------------------------
 0 |  39490,   39489,      0,   0,       1276  ,  106714,      91       |  1737  1880
 1 | 39490,   39490,      0,   0,        226  ,  107619,     203       |  1694  1041
```

(1) Output trimmed.

Note
See TRex full manual for a complete understading of the tool features and options.

3.1.4 Updating TRex

See Related manual section

3.1.5 TRex Live monitoring

Once we have TRex up and running, we can monitor its performance using TRexViewer application (Supported only on Windows OS).

This can be done by following these steps:

1. Download the latest version of TrexViewer application and install it using this link.
2. Start the application and fill in the following:
   - Trex ip: `127.0.0.1:4500`
3. Click the play button.

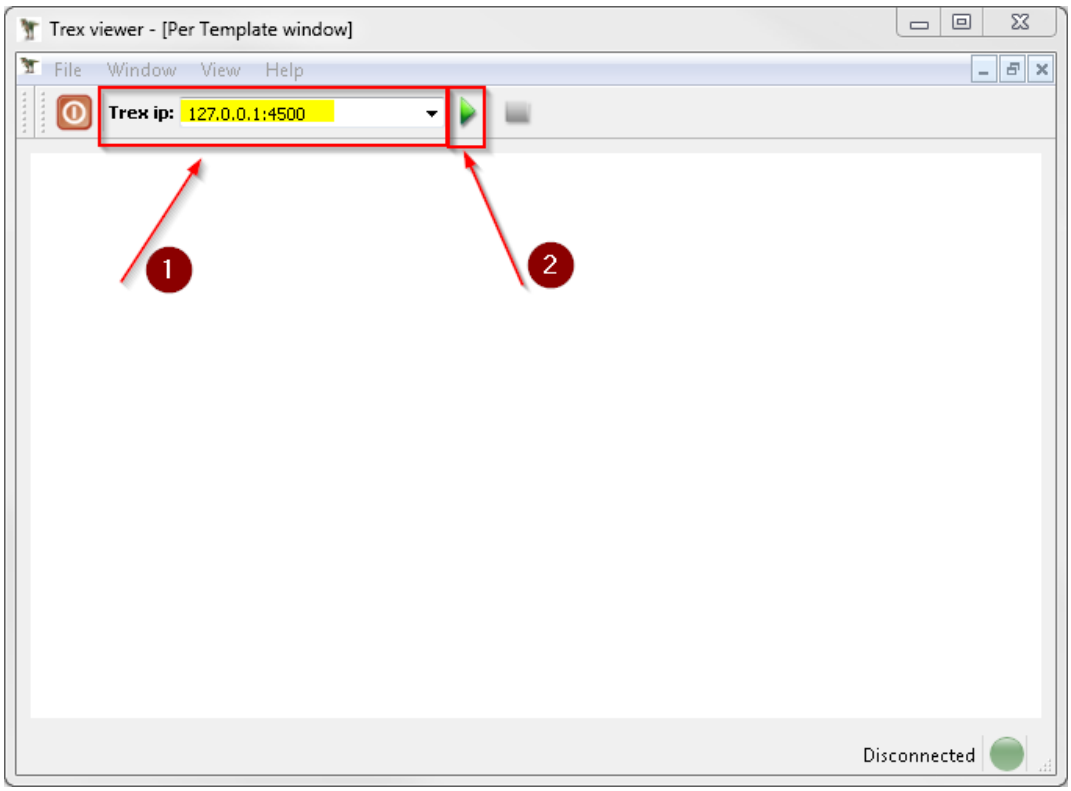

Figure 3.5: TRex viewer start screen

That's it!
Now the live data from TRex will be displayed on the screen.

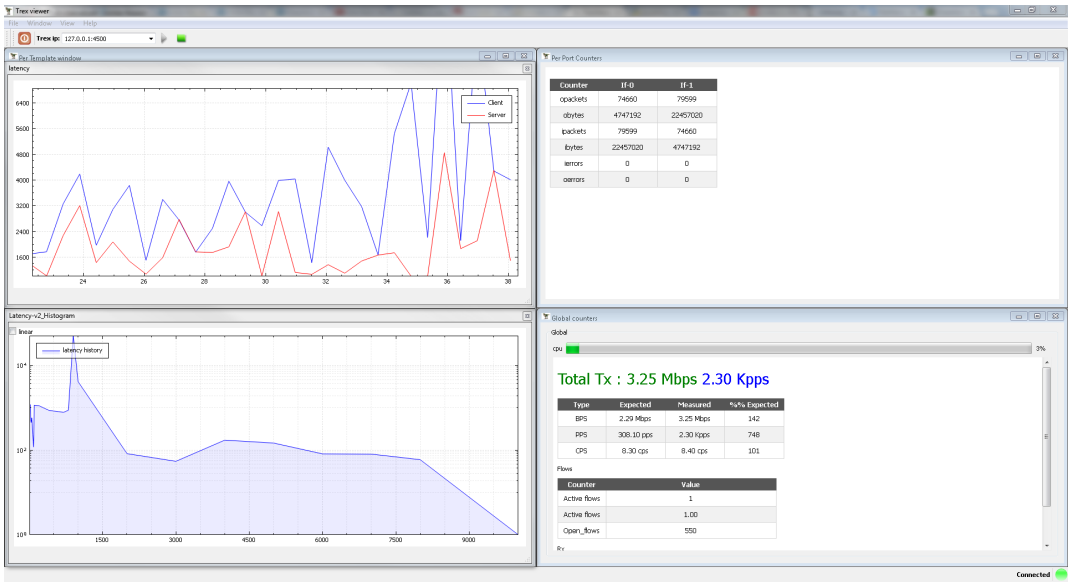

Figure 3.6: TRex viewer monitor screen

Note
Make sure TRex is running, otherwise data will not be available at TRexViewer.

3.1.6 Architecture and network design

Since no hardware is used, TRex simulates traffic using a virtual internal network, named *trex_intnet*.

The following figure describes the virtual "wiring" of the virtual machine to support TRex traffic simulation.

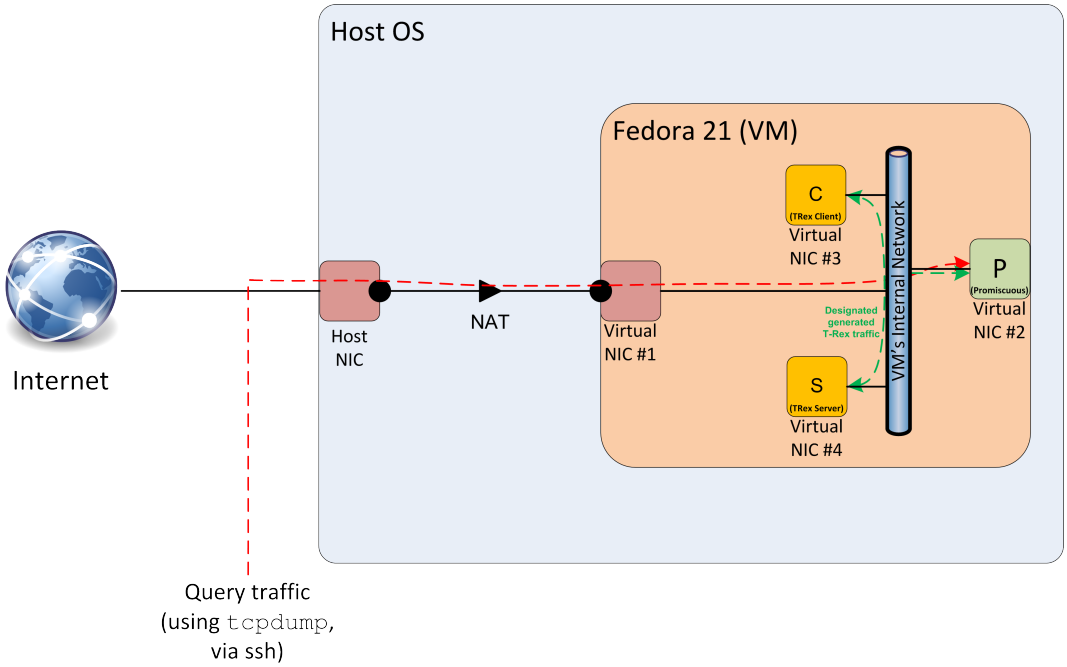

Figure 3.7: TRex virtual connectivity

The VM runs TRex with single client and single server port. The traffic generated by each of those ports is switched over the *trex_intnet* virtual network and received by the other side.

TRex identifies only the packets which were dedicately sent by one of those traffic ports and receives them on the other port. Hence, packets generated by client port will be received by the server port and vice versa.

Network adapter #4 can be used for capturing all traffic generated by both of TRex's ports.