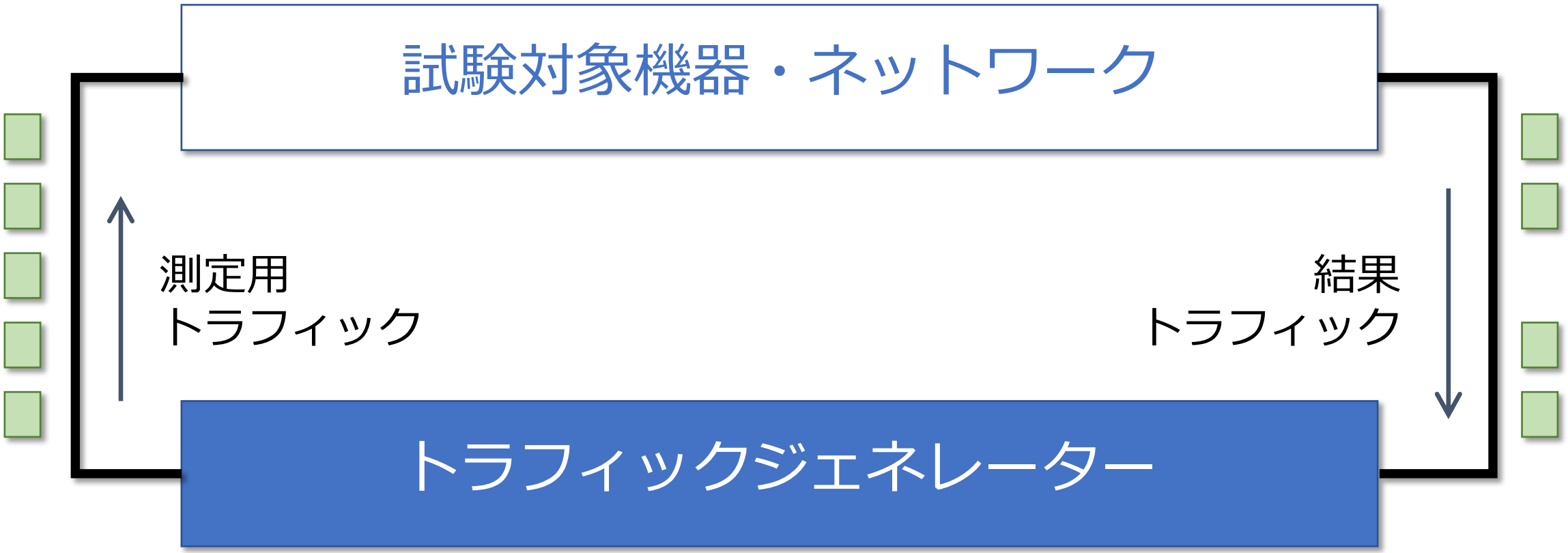
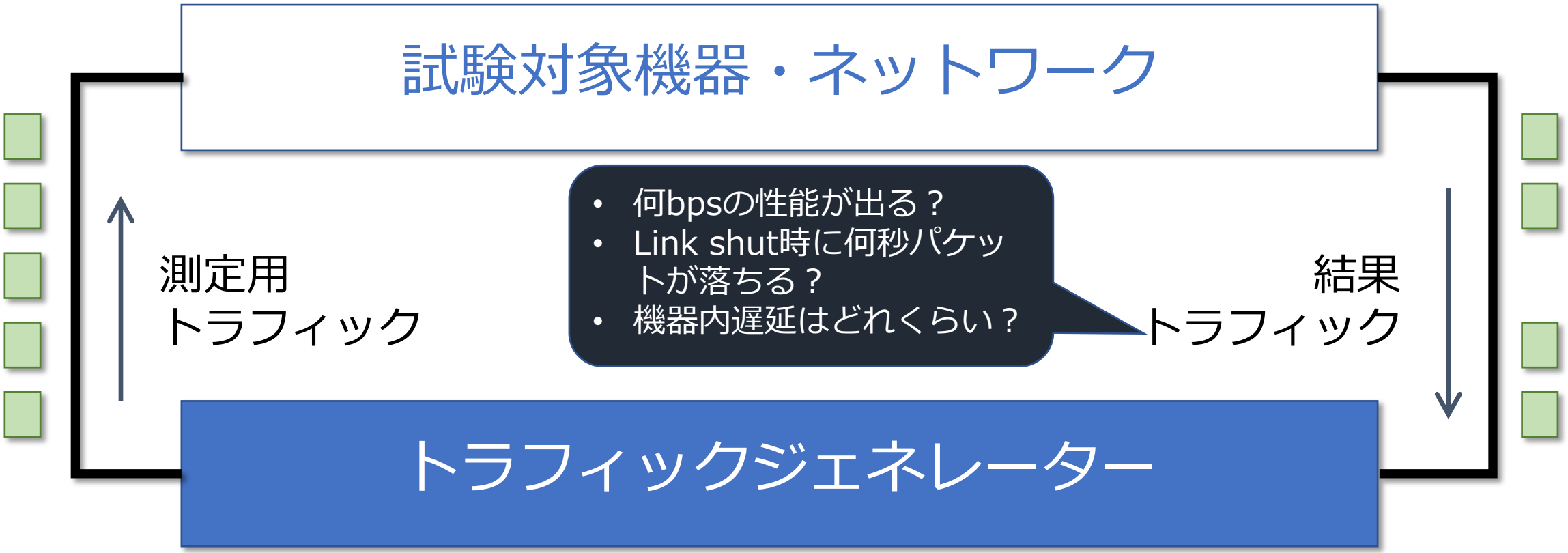
OSSなトラフィックジェネレーターなTRexを使ってみて

KDDI総合研究所　宮坂拓也

検証に欠かせないもの・・・



検証に欠かせないもの・・・


試験対象機器・ネットワーク
測定用
トラフィック
何bpsの性能が出る？
Link shut時に何秒パケットが落ちる？
機器内遅延はどれくらい？
結果
トラフィック
トラフィックジェネレーター

というわけで：本日の主役 (Agenda)

- **トラフィックジェネレーター**
  1. TRexの紹介
     - 概要紹介
     - Stateful and Stateless mode
     - サポート機種・NIC・OS
  2. TRex Statelessモードを使ってみて
     - Pythonによる試験自動化と結果の取得

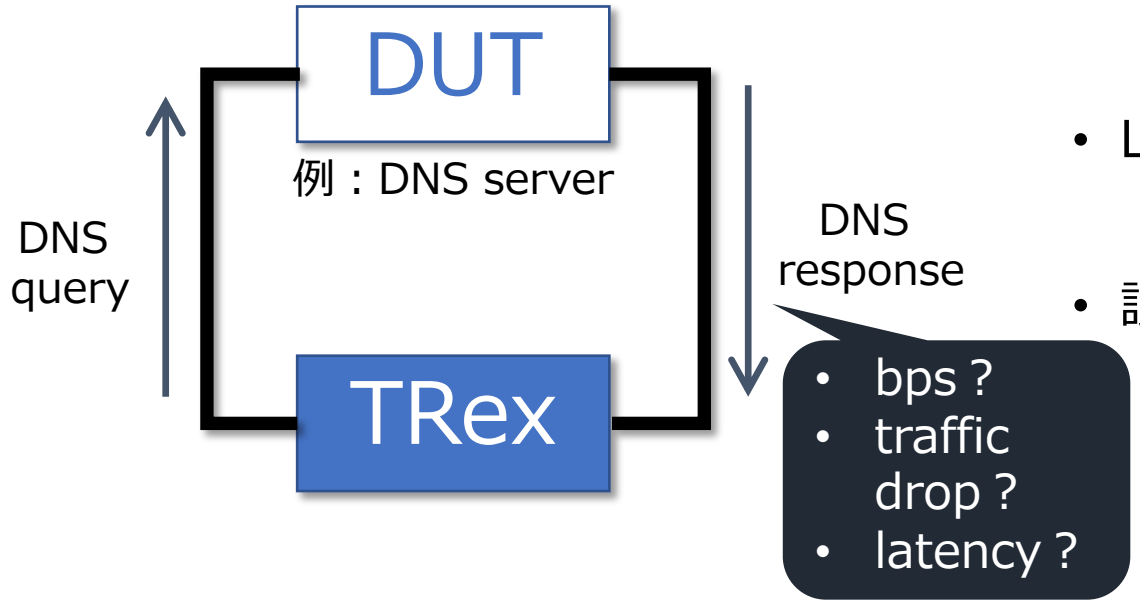
※詳細アーキテクチャ・インストール手順などはカバーしていません、公式ページなどを見てください

TRex：Realistic Traffic Generator

- CiscoがOSSとして公開しているトラフィックジェネレーター
  - https://trex-tgn.cisco.com/
- 技術概要
  - Linux application
  - DPDKによるパケット転送
  - 仮想化・クラウド環境での実行もサポート
    - 物理NIC＋仮想化Interfaceをサポート
- Stateful and Stateless Traffic Generation

TRex：Realistic Traffic Generator

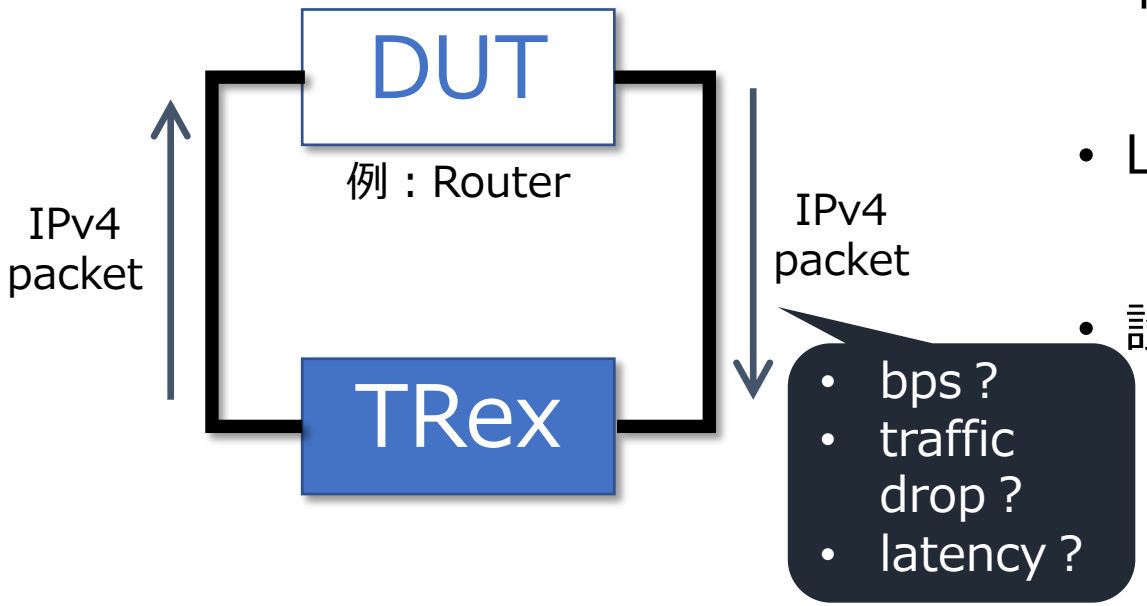
- CiscoがOSSとして公開しているトラフィックジェネレーター
  - https://trex-tgn.cisco.com/
- Stateful and Stateless Traffic Generation



(1) Stateful Mode
- トラフィックフローのステートを管理
  - 行きと返りでパケットが異なる場合などに利用する感じ？
  - https://trex-tgn.cisco.com/trex/doc/trex_manual.html
- L7の負荷試験などに用いられるのが主
  - DNS server
  - HTTP server
- 設定できるもの
  - トラフィックフローの「行き」と「帰り」のパケットのパターン(例：DNS query & response) (pcapで指定)
  - src/dst IP+src Port (yamlで指定)
  - Trexから流すトラフィック量(bps) (yamlで指定)

TRex：Realistic Traffic Generator

- CiscoがOSSとして公開しているトラフィックジェネレーター
  - https://trex-tgn.cisco.com/
- Stateful and Stateless Traffic Generation



(2) Stateless Mode

- トラフィックフローのステートを管理しない
  - ネットワークオペレーターはこちらの方が使いそう
  - https://trex-tgn.cisco.com/trex/doc/trex_stateless.html
- L2/L3の負荷試験などに用いられるのが主
  - スイッチ
  - ルーター
- 設定できるもの
  - トラフィックフローの「行き」のパケットのパターン(IP address, port) (python(Scapy)で指定)
  - Trexから流すトラフィック量(bps) (pythonで指定)

どんな機器で使えるの？

- Server
  - CPU：3コア以上 (x86)
  - Memory：4GB以上
    - https://trex-tgn.cisco.com/trex/doc/trex_manual.html#_hardware_recommendations
    - https://trex-tgn.cisco.com/trex/doc/trex_faq.html#_can_trex_run_on_an_hypervisor_with_virtual_nics

Table 2. Preferred UCS hardware

UCS Type	Comments
UCS C220 Mx	**Preferred Low-End**. Supports up to 40Gb/sec with 540-D2. With newer Intel NIC (recommended), supports 80Gb/sec with 1RU. See table below describing components.
UCS C200	Early UCS model.
UCS C210 Mx	Supports up to 40Gb/sec PCIe3.0.
UCS C240 Mx	**Preferred, High-End** Supports up to 200Gb/sec. 6x XL710 NICS (PCIex8) or 2xFM10K (PCIex16). See table below describing components.
UCS C260M2	Supports up to 30Gb/sec (limited by V2 PCIe).

もちろんUCSじゃなくても動く！

- OS (Recommended)
  - CentOS/RHEL 7.6 (64bit)
    - https://trex-tgn.cisco.com/trex/doc/trex_manual.html#_installing_os

どんな機器で使えるの？

- NIC
  - DPDKをサポートするもの

Table 5. Supported NICs

Chipset	Bandwidth (Gb/sec)	LSO	LRO	Example
Any Kernel Linux interface	x	x	x	veth,tap,tun,eth0,wireless interface, up to ~1MPPS, one thread. see low footprint and Linux interfaces
Intel I350	1	+	-	Intel 4x1GE 350-T4 NIC
Intel 82599	0.1/1/2.5/5/10	+	+	Cisco part ID:N2XX-AIPCI01 Intel x520-D2, Intel X520 Dual Port 10Gb SFP+ Adapter. X550-T2,x540-T2 for tbase Cisco part ID:UCSC-PCIE-ID10GC
Intel 82599 VF	x	+	+	
Intel X710	10	+	-	Cisco part ID:UCSC-PCIE-IQ10GF SFP+, **Preferred** support per stream stats in hardware Silicom PE310G4i71L
Intel XL710	40	+	-	Cisco part ID:UCSC-PCIE-ID40GF, QSFP+ (copper/optical) **Preferred** support per stream stats in hardware
Intel XXV710	1/10/25	+	-	SFP28 Intel XXV710 **Preferred** support per stream stats in hardware
Intel XL710/X710 VF	x	+	-	
Napatech SmartNICs NT40E3-4	10	-	-	./b configure --with-ntacc to build the library
Napatech SmartNICs NT80E3-2	40	-	-	./b configure --with-ntacc to build the library
Napatech SmartNICs NT100E3-1	100	-	-	./b configure --with-ntacc to build the library. The only Napatech NIC in our regression. more info
Napatech SmartNICs NT200A01	100	-	-	./b configure --with-ntacc to build the library **Note:** This NIC require a BIOS with PCIe bifurcation support. PCIe bifurcation
Mellanox ConnectX-4/Lx	25/40/50/56/100	+	+	SFP28/QSFP28, ConnectX-4 ConnectX-4-brief (copper/optical) supported from v2.11 more details and issues TRex Support
Mellanox ConnectX-5	25/40/50/56/100	+	+	Supported, see issues TRex Support
Cisco 1300 series	40	+	-	QSFP+, VIC 1380, VIC 1385, VIC 1387 see more TRex Support
VMXNET3	VMware paravirtualized	+	-	Connect using VMware vSwitch
E1000	paravirtualized	+	-	VMware/KVM/VirtualBox
Virtio	paravirtualized	+	-	KVM
Amazon ENA	paravirtualized	+	-	Amazon Cloud
MS Failsafe	paravirtualized	+	-	Azure with DPDK mlx5 support

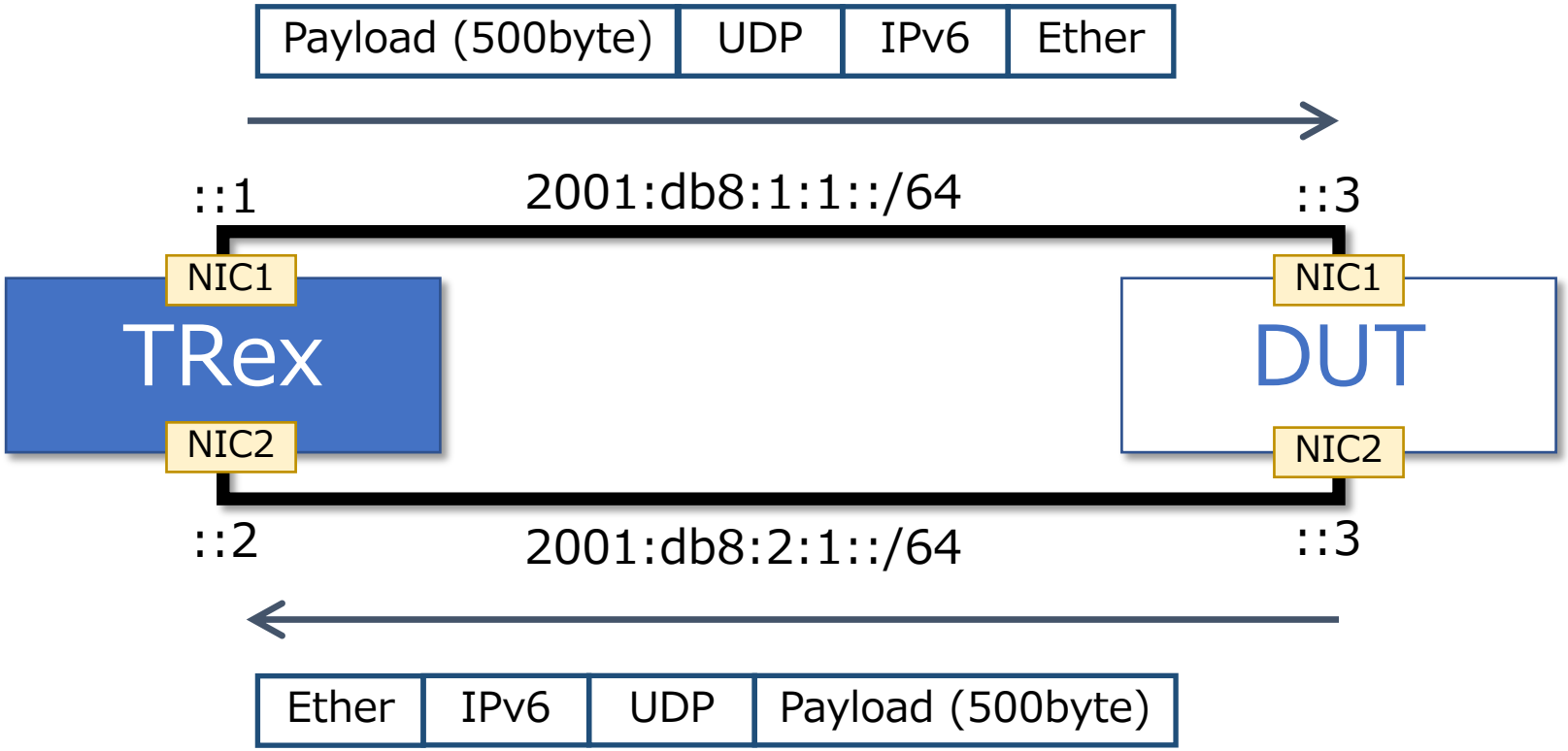
仮想化・クラウド環境でも動作する模様

Statelessで使ってみた

- 実行環境

	TRex	DUT
CPU	Intel Xeon CPU E5-2620 v4 (2.10GHz 8-core 16-thread)	Intel Xeon CPU E5-2680 v4 (2.40GHz 14-core 28-thread)
RAM	256GB	128GB
NIC chipset	Intel X710	Intel X710
OS	CentOS 7.6 (Linux kernel 3.10.0)	Ubuntu 18.04 (Linux kernel 5.0.0)

Statelessで使ってみた



Pythonによる測定 (1/2)

```
def trex_test(tx_port,rx_port):

    # create client
    c = STLClient()

    # connect to server and acquire ports
    c.connect()
    c.reset(ports=[tx_port, rx_port])

    # set ipv6 address on TRex
    c.set_service_mode(ports=[tx_port, rx_port], enabled=True)
    c.conf_ipv6(port=tx_port, enabled=True, src_ipv6="2001:db8:1:1::1")
    c.conf_ipv6(port=rx_port ,enabled=True, src_ipv6="2000:db8:2:1::2")
    c.set_service_mode(ports=[tx_port, rx_port], enabled=False)

    # create packet format with Scapy and traffic stream
    Eth_frame = Ether()
    IPv6_pkt = IPv6(src="2001:db8:1:1::1", dst="2001:db8:2:1::2")
    UDP_dgm = UDP(dport=50000, sport=50001)
    Payload = 500 * 'x'
    base_pkt = Eth_frame / IPv6_pkt / UDP_dgm / Payload
    pkt = STLPktBuilder(pkt=base_pkt)
    s1 = STLStream(name='test',
                   packet=pkt,
                   mode=STLTXCont(bps_L1=1000000000))
```

TRexを操作するために、
Clientインスタンスを生成

Clientインスタンスを通じて、
TrexのIPv6アドレスを設定

Scapyを用いて、送信するパケットフォーマットを決定
(Ether+IPv6+UDP+Payload)

生成したパケットを元にトラフィックストリームを作成
(1Gbpsで上記パケットを送信)

Pythonによる測定 (2/2)

```
    # add the traffic stream to TRex
    c.add_streams(s1, ports=[tx_port])

    # start the traffic stream
    c.clear_stats()
    c.start(ports = [tx_port])
    time.sleep(10)

    # get a results
    global_flow_stats = c.get_stats()
    print(global_flow_stats)
    print("Sending pps is {}".format(global_flow_stats[tx_port]['tx_pps']))
    print("Receiving pps is {}".format(global_flow_stats[rx_port]['rx_pps']))
    print("Dropped bps is {}".format(global_flow_stats['global']['rx_drop_bps']))

    # stop the traffic stream and disconnect from TRex
    c.stop(ports = [tx_port])
    c.disconnect()

    return
```

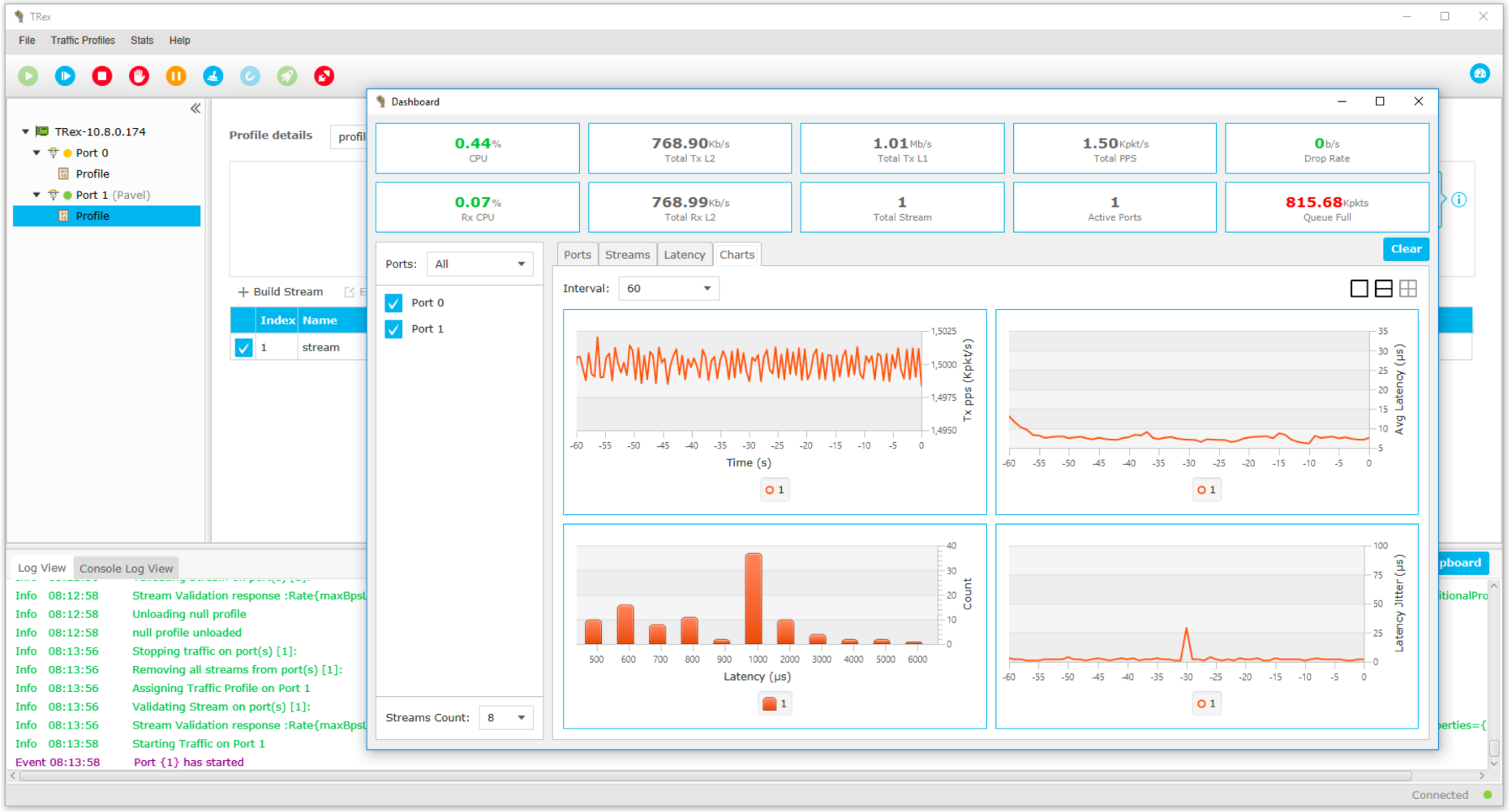
生成したストリームを送信ポートに紐付ける

統計情報をクリアし、トラフィックの送信開始

統計情報の取得&表示

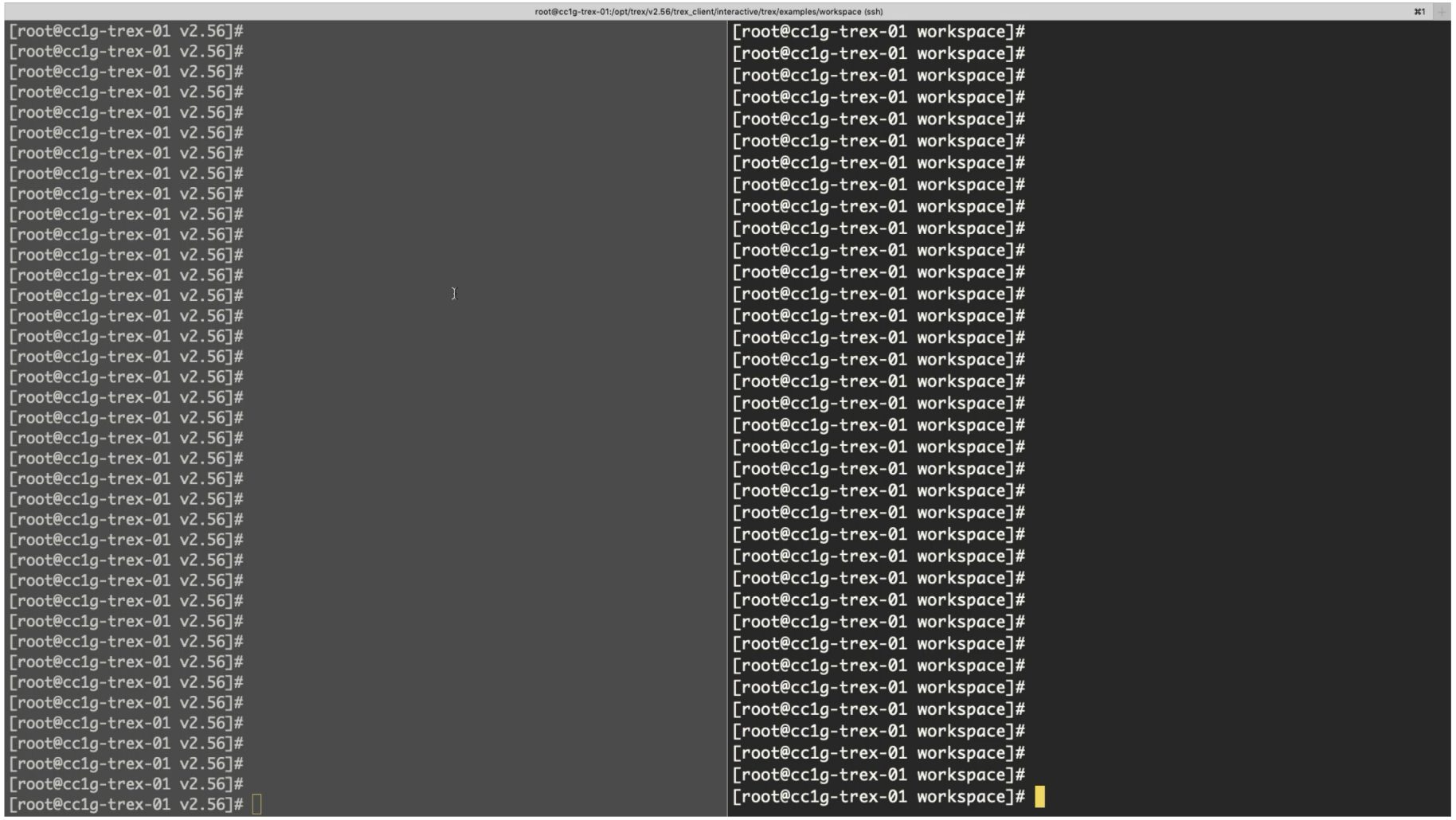
トラフィックの停止、ポートの開放を実施

(参考) Stateless用のGUI



https://github.com/cisco-system-traffic-generator/trex-stateless-gui

実行デモ


root@cc1g-trex-01:/opt/trex/v2.56/trex_client/interactive/trex/examples/workspace (ssh)
[root@cc1g-trex-01 v2.56]#
[root@cc1g-trex-01 workspace]#

パケットキャプチャ

```
> Frame 4: 562 bytes on wire (4496 bits), 562 bytes captured (4496 bits)
> Ethernet II, Src: HewlettP_6e:76:00 (14:02:ec:6e:76:00), Dst: HewlettP_6e:82:80 (14:02:ec:6e:82:80)
v Internet Protocol Version 6, Src: 2001:db8:1:1::1, Dst: 2001:db8:2:1::2
    0110 .... = Version: 6
  > .... 0000 0001 .... .... .... .... .... = Traffic Class: 0x01 (DSCP: CS0, ECN: ECT(1))
    .... .... .... 0000 1111 1100 0000 0000 = Flow Label: 0x0fc00
    Payload Length: 508
    Next Header: UDP (17)
    Hop Limit: 64
    Source: 2001:db8:1:1::1
    Destination: 2001:db8:2:1::2
v User Datagram Protocol, Src Port: 50001, Dst Port: 50000
    Source Port: 50001
    Destination Port: 50000
    Length: 508
    Checksum: 0x7434 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 0]
v Data (500 bytes)
    Data: 78787878787878787878787878787878787878787878787878...
    [Length: 500]
```

提供されるPython Module

- Python Modules for TRex
  - Client Module
    - TRexクライアント操作
  - Traffic Profile Module
    - トラフィックストリーム操作
  - Packet Builder Module
    - パケットフォーマット作成
  - Field Engine Module
    - パケット内ヘッダー値変更用
      - 例：送信元IPアドレスを1ずつインクリメントして送る
  - Namespace Module
    - Linux Namespace操作
- https://trex-tgn.cisco.com/trex/doc/cp_stl_docs/index.html

まとめ＆使ってみての感想

- TRexを紹介しました
- Statelessモードで測定を使ってみました
  - 感想
    - ☺ インストールはすごい楽
    - ☺ Pythonで全てかけるので操作が簡単。Scapyによるパケットフォーマット操作はとても楽。pandasやmatplotlibなどと組み合わせて結果の出力も容易
    - ☺ 試してはいないけど、仮想化/クラウド環境などでも使えるみたいなので、そのような環境での試験にも使えそう
    - ☺ 日本でも最近人気が出てきて、TRexに関する日本語情報がけっこう出てくる！(blog・同人誌)
    - ☹ 正確な測定が求められる場合などは、自分で測定器の校正的なものをやらないと流石にダメかなあ