Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Institute of Computer Science
Chair of Communication Networks
Prof. Dr. Tobias Hoßfeld

Affordable Measurement Setups for Networking Device Latency with Sub-Microsecond Accuracy

KuVS Fachgespräch – WueWoWAS'22 – Würzburg

Alexej Grigorjew, Philip Diederich, Tobias Hoßfeld, Wolfgang Kellerer

alexej.grigorjew@uni-wuerzburg.de

Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Technische
Universität
München

Many “new“ use-cases for real-time networking

Audio Video Bridging
[802.1BA/Revision]

Fronthaul
[802.1CM/de]

Industrial Automation
[IEC/IEEE 60802]

Aerospace Onboard
[P802.1DP / SAE AS6675]

Service Provider
[P802.1DF]

Automotive In-Vehicle
[P802.1DG]

Many “new” use-cases for real-time networking

Audio Video Bridging
[802.1BA/Revision]

Fronthaul
[802.1CM/de]

Industrial Automation
[IEC/IEEE 60802]

Aerospace Onboard
[P802.1DP / SAE AS6675]

Service Provider
[P802.1DF]

Automotive In-Vehicle
[P802.1DG]

More Buzzwords

Brownfield deployments

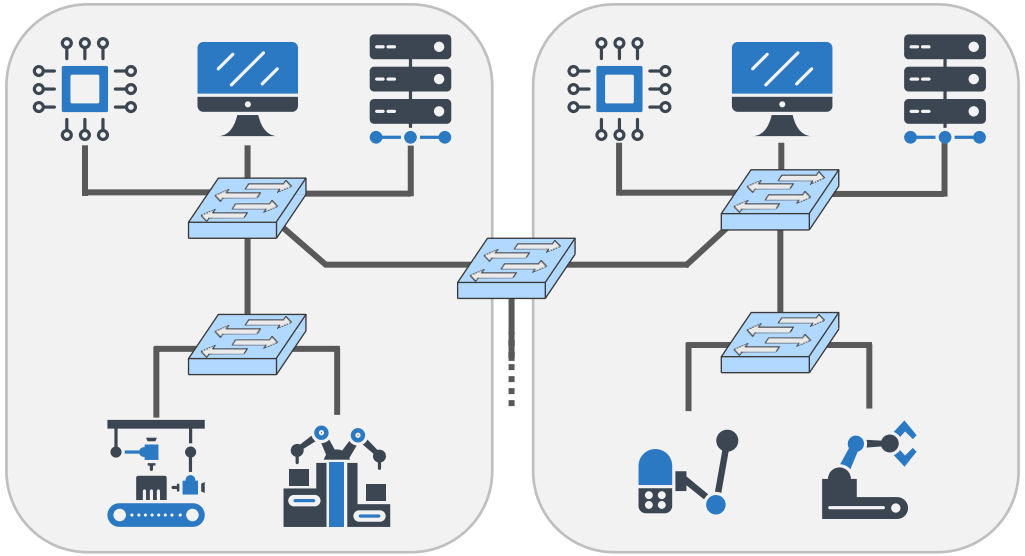
Resource reservation

Guarantees

Shaping, Filtering, Switching

Black box devices

Measurements!



Accurate measurement equipment can be expensive

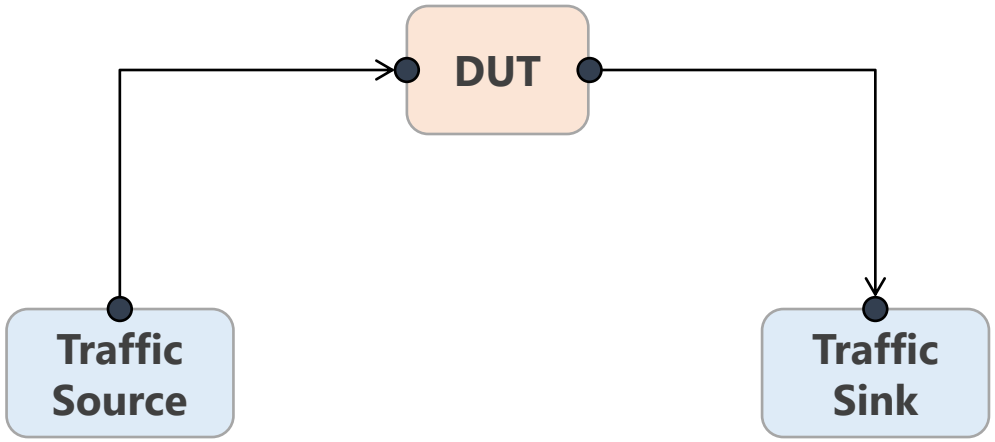
Dedicated Measurement-tailored Options

- Traffic generators and sinks
- Network taps
- Capture cards
- Licenses!

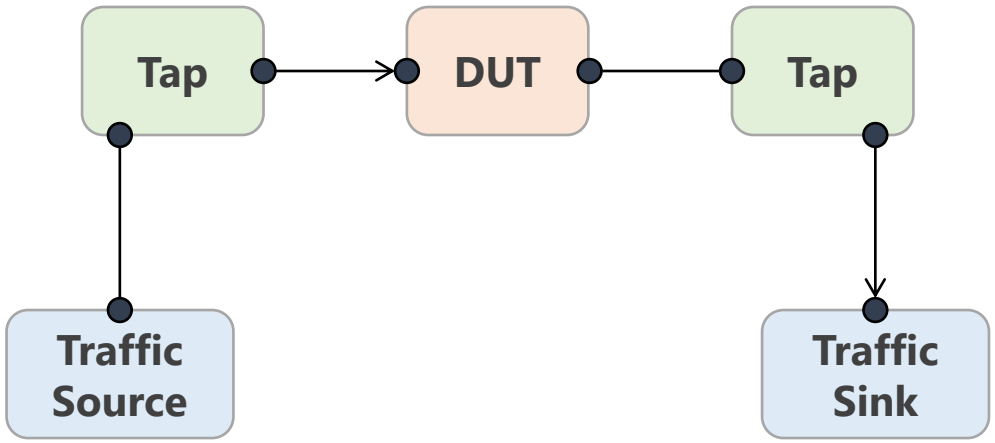
But what do we want?

- Accurate time stamps (by hardware)
- High performance (packets / second)

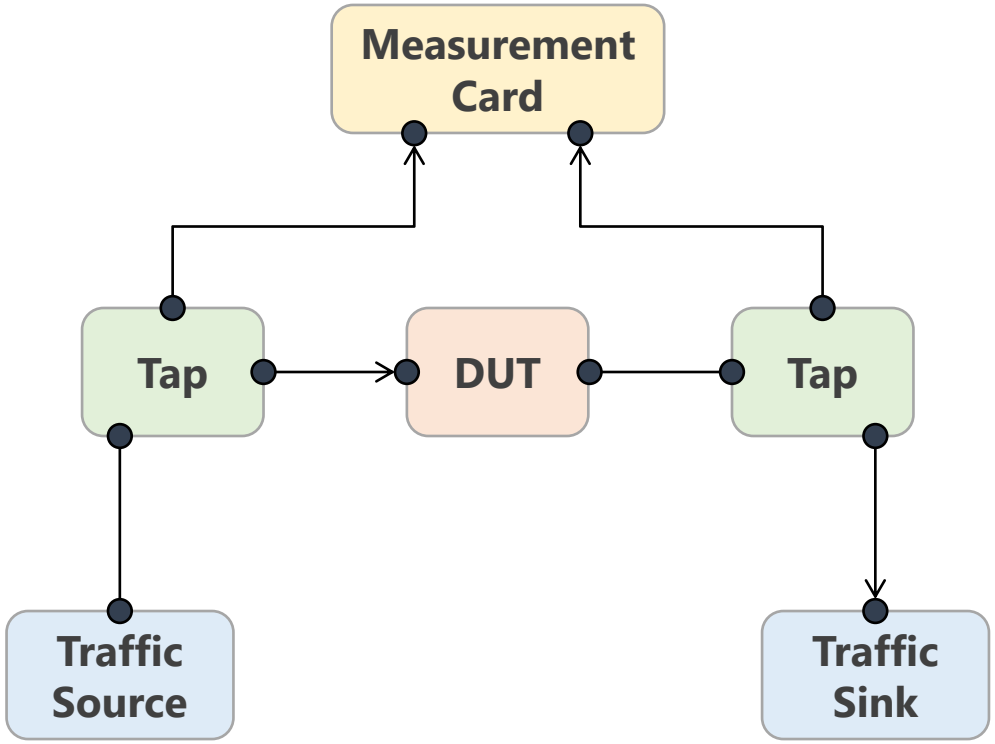
General measurement setup(s)



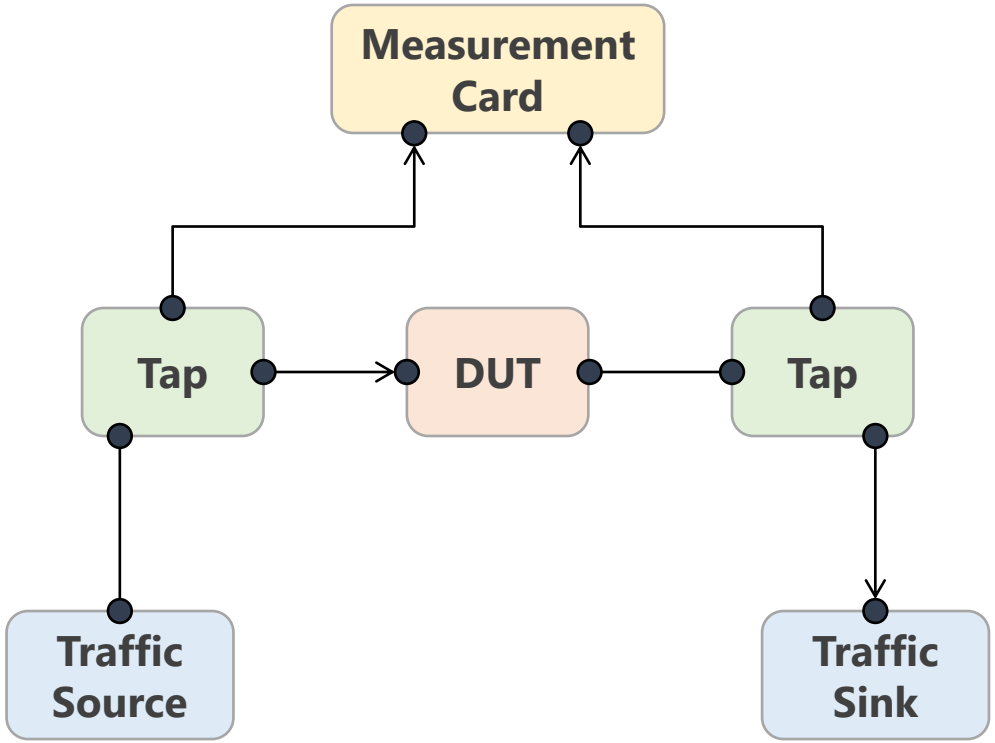
General measurement setup(s)

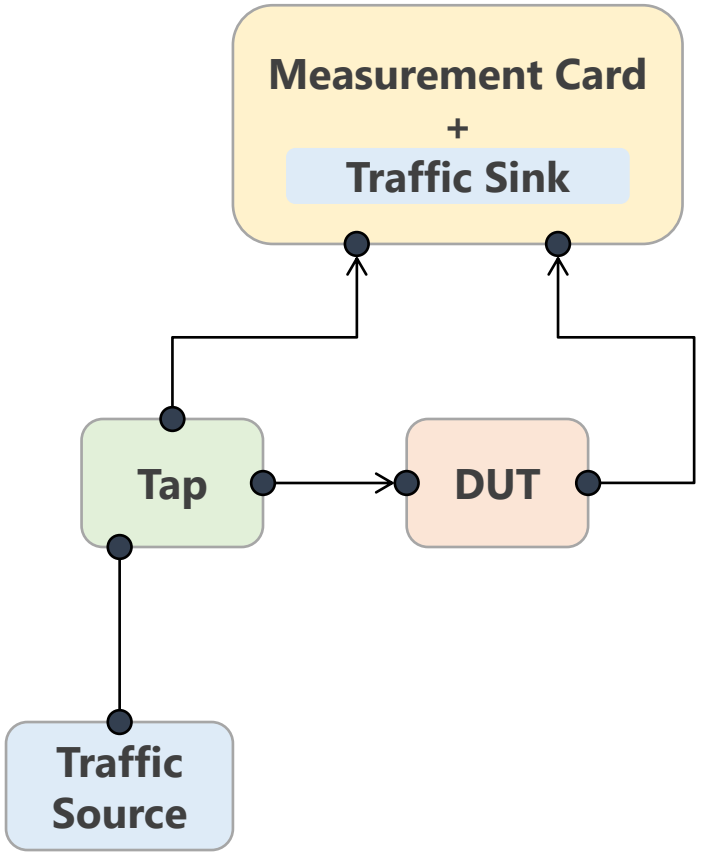


General measurement setup(s)

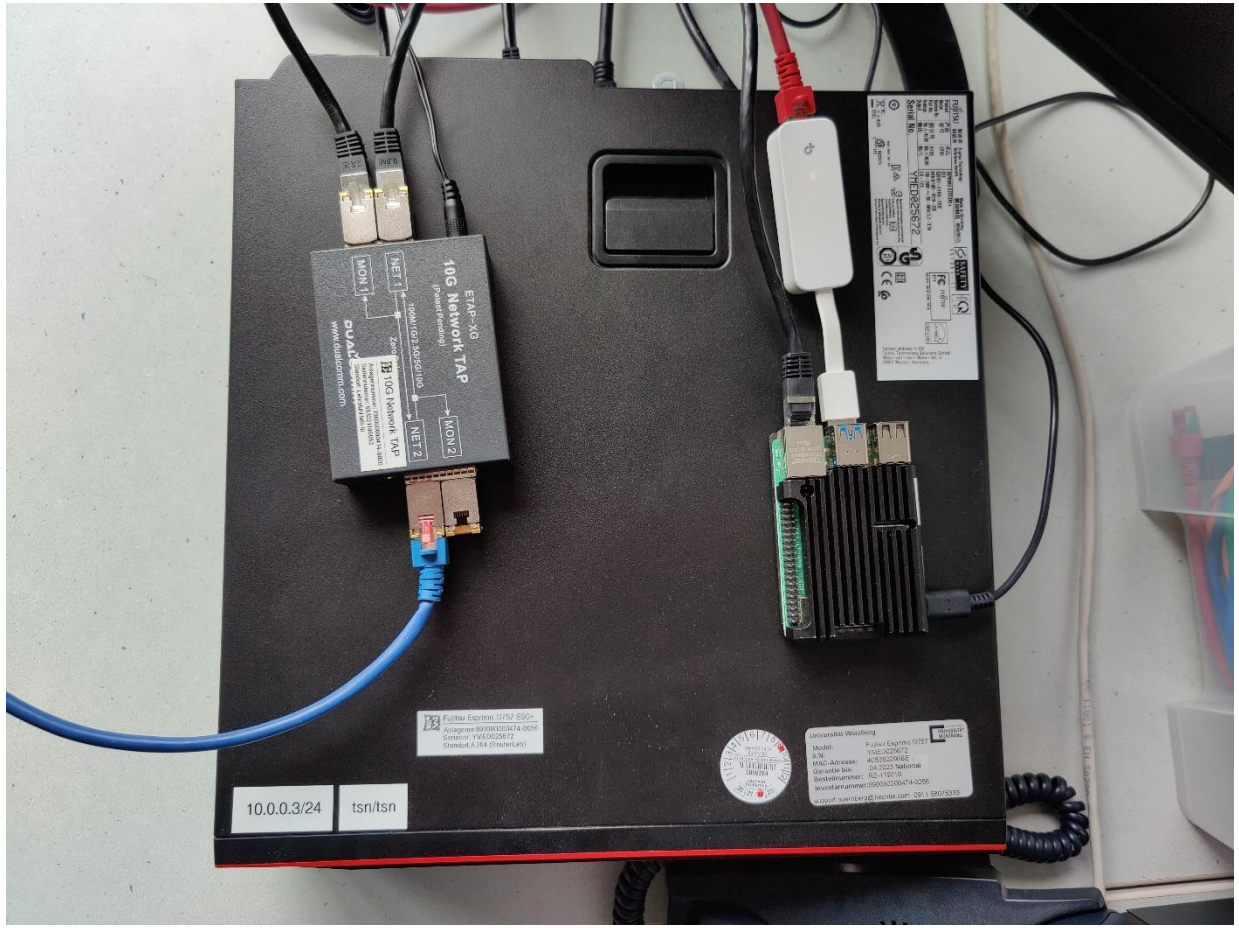


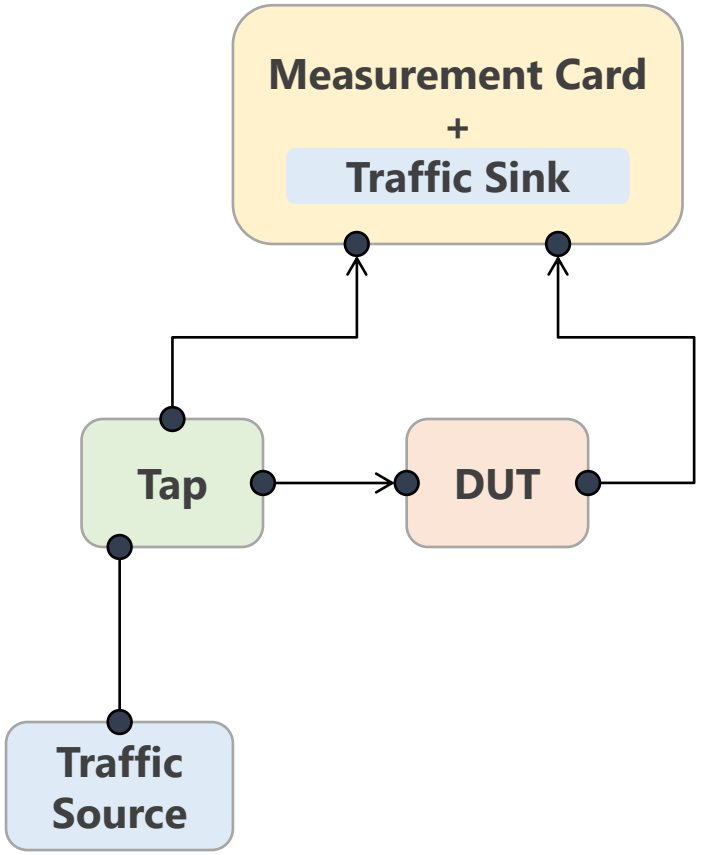
General measurement setup(s)



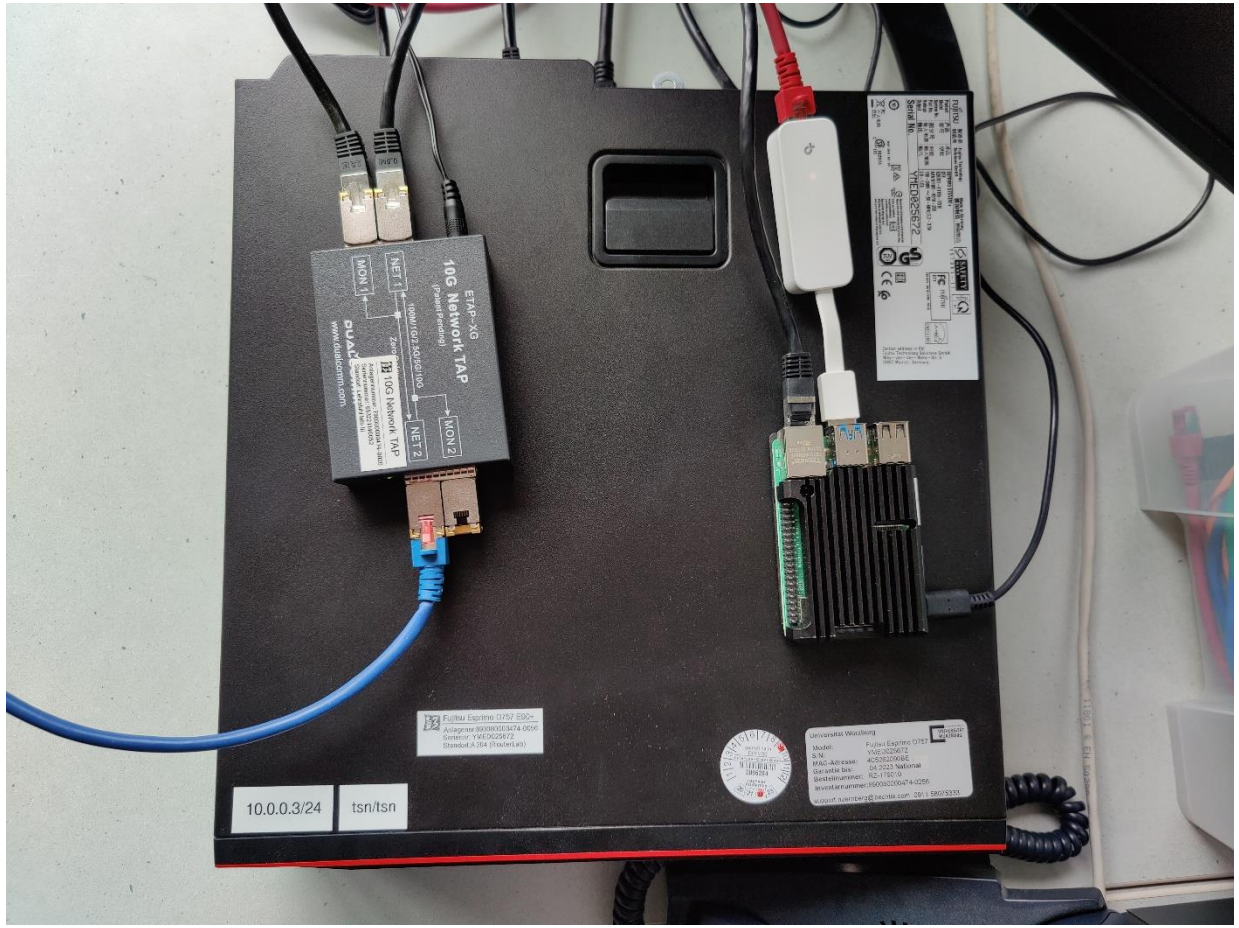


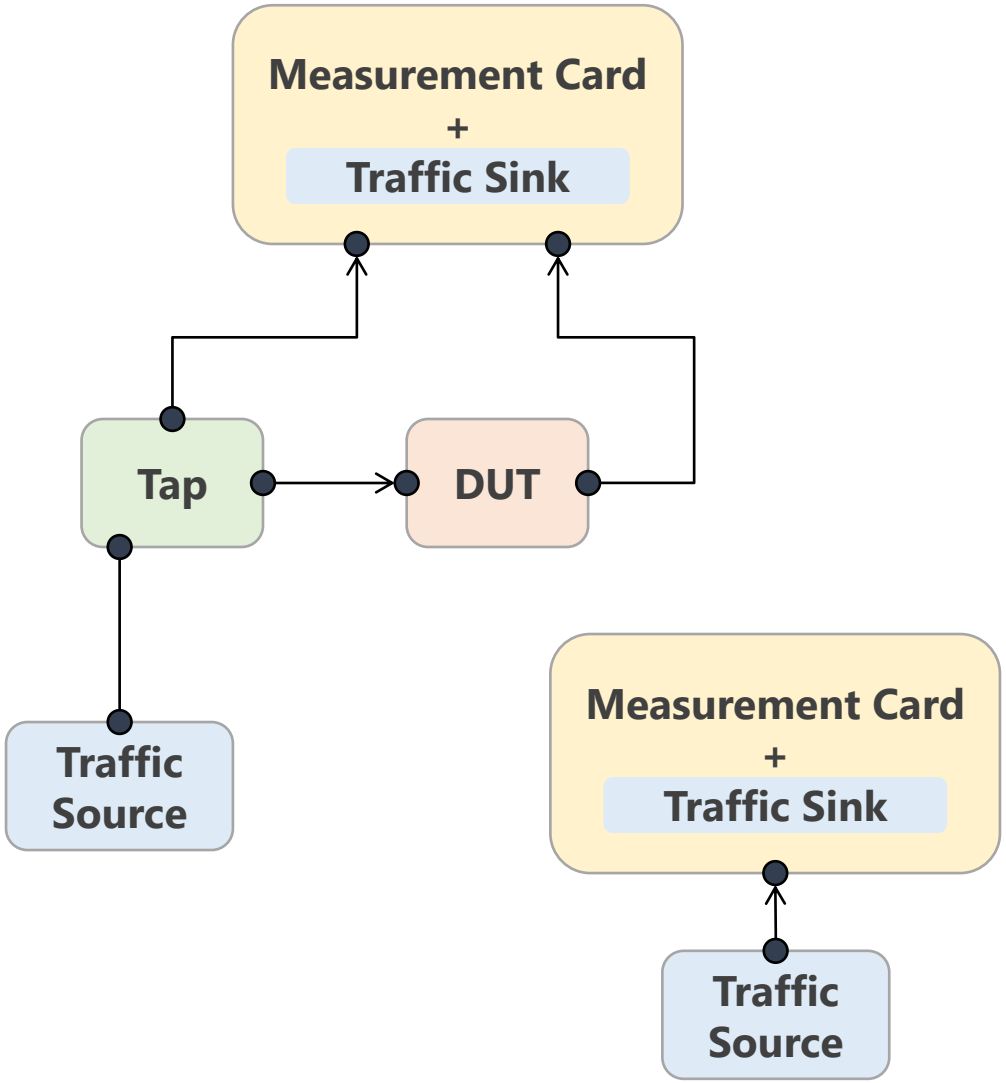
General measurement setup(s)





General measurement setup(s)

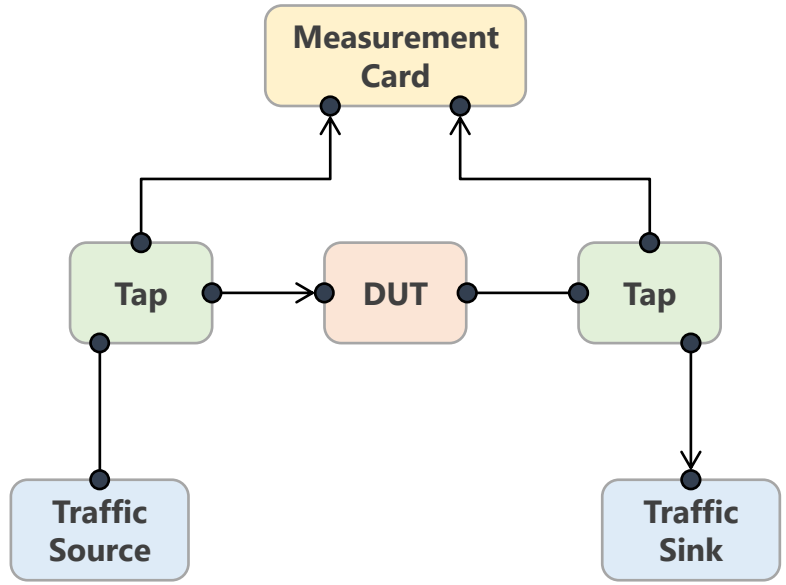




Hardware selection

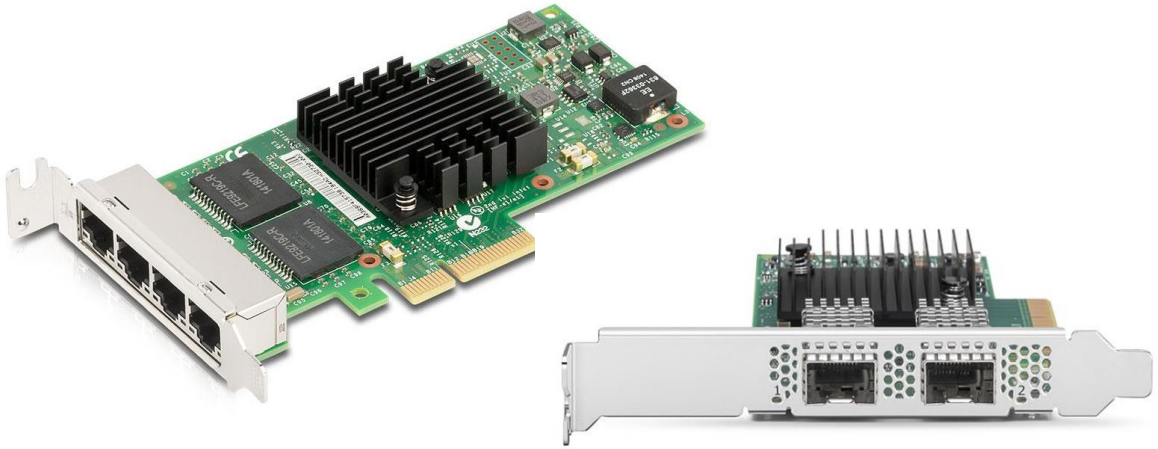
Tap

- Cheap
- No extra delay / jitter
- No packet drops
- No mirror port of a switch



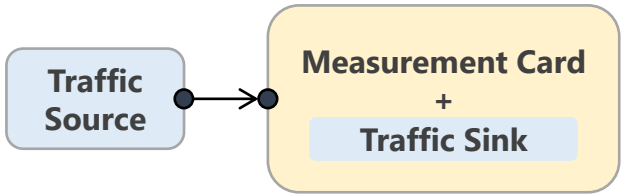
Measurement Card

- Cheap
- Hardware time stamping
- For **all** incoming packets
- Multiple ports with the same clock
- (or synchronized clocks)

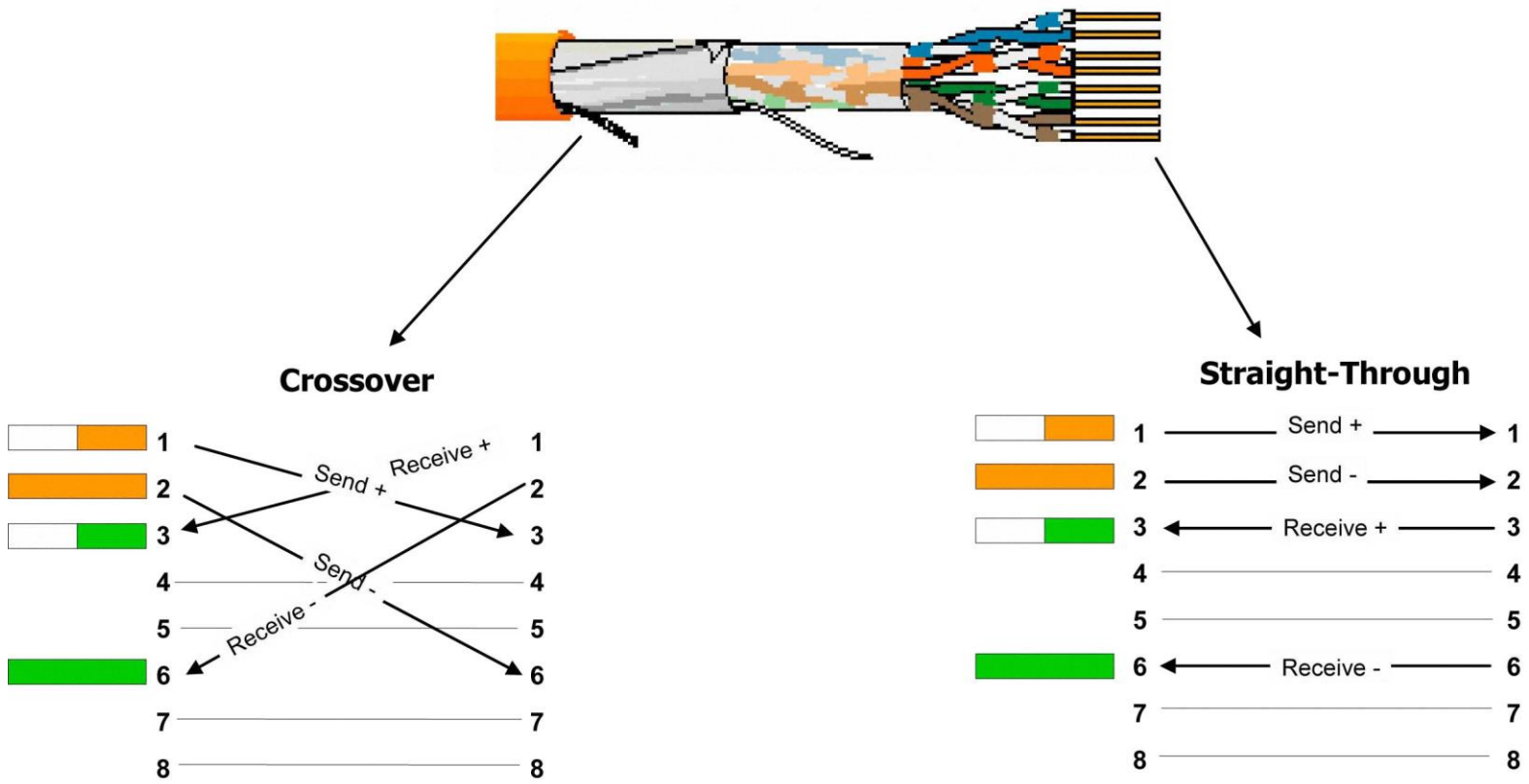


Measurement PC

- Free PCI-E slot & lanes
- Decent CPU
- 1 Gbit/s / 84 Bytes = 1.5 Million pkts/s
- (many Intel onboard ports support hardware TS)

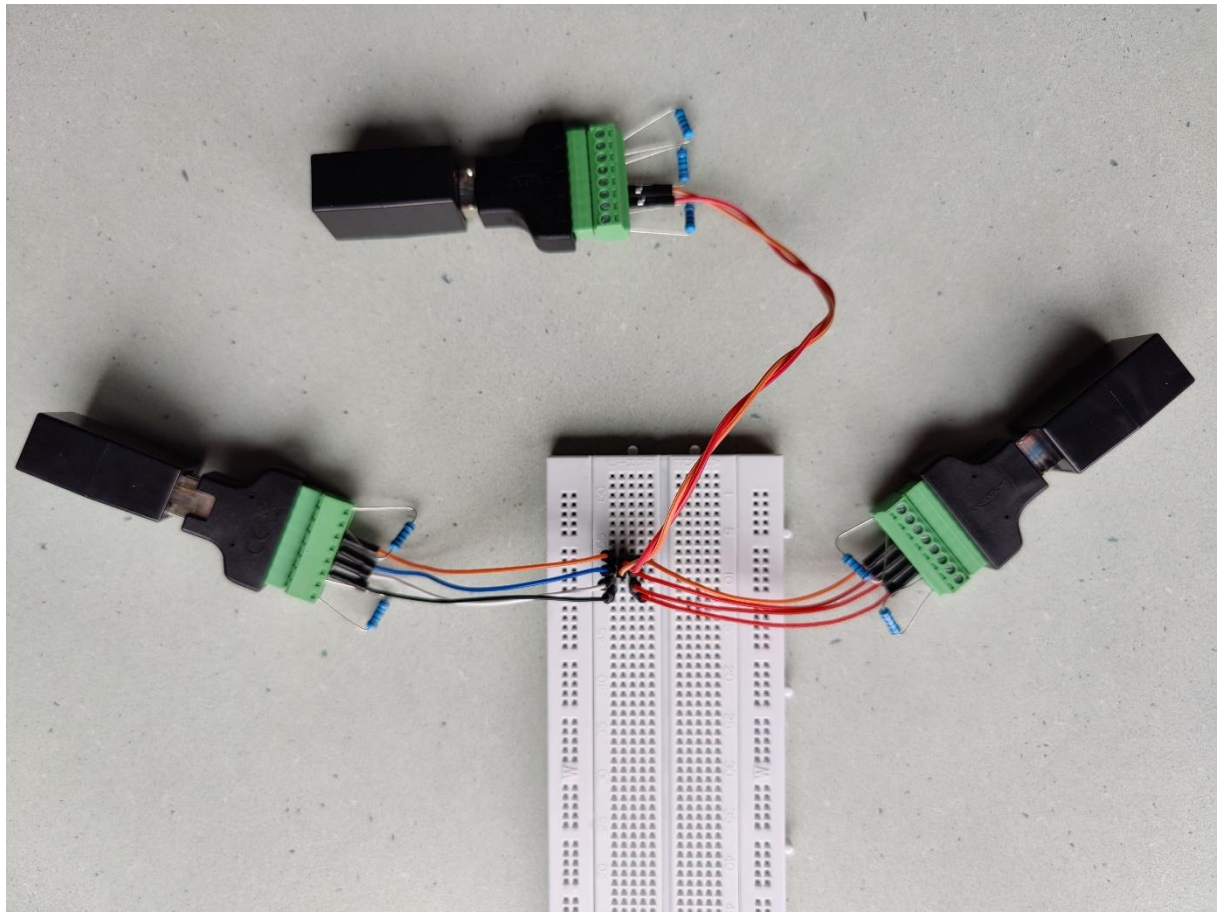


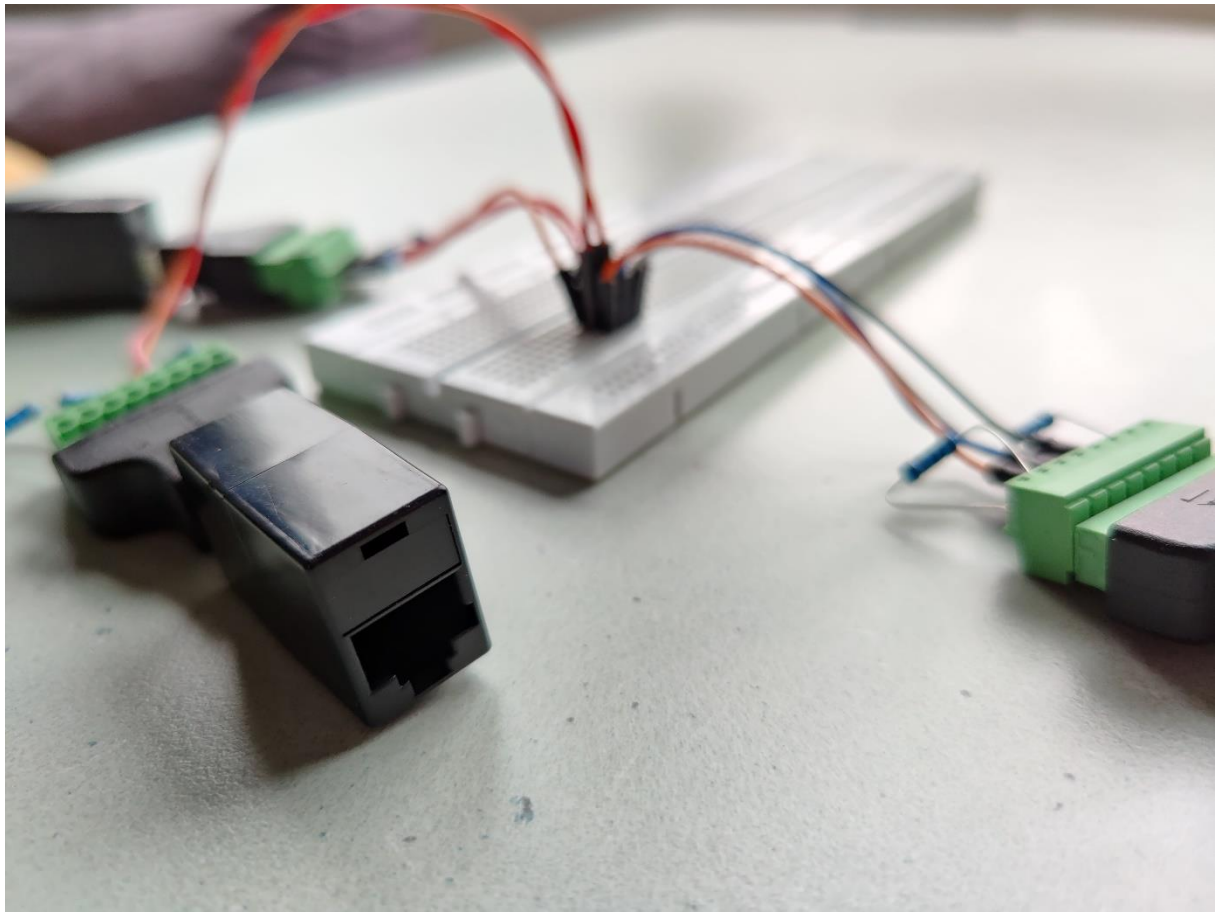
Building your own tap



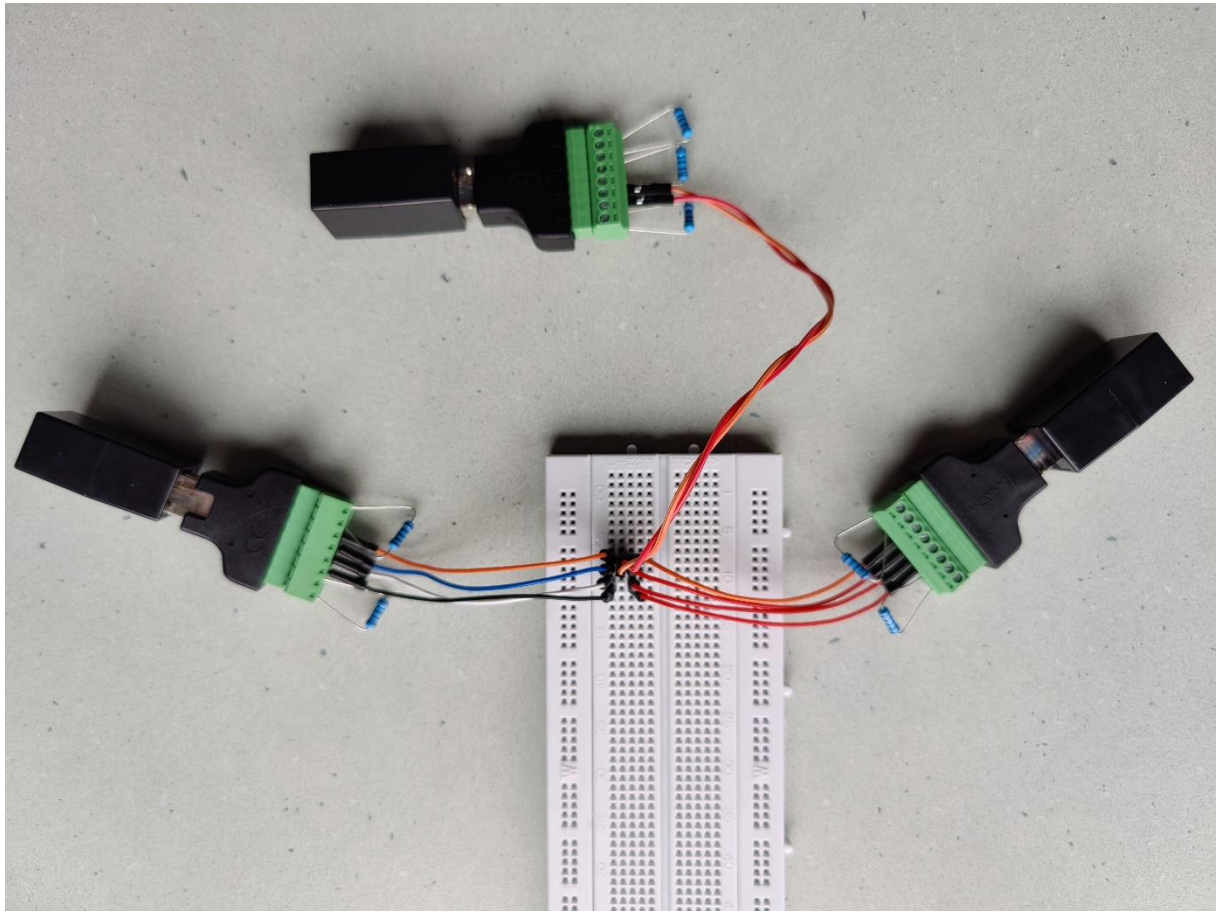
https://www.airnet.de/basic_internetworking1/de/html/MedStecktech_learningObject17.xml

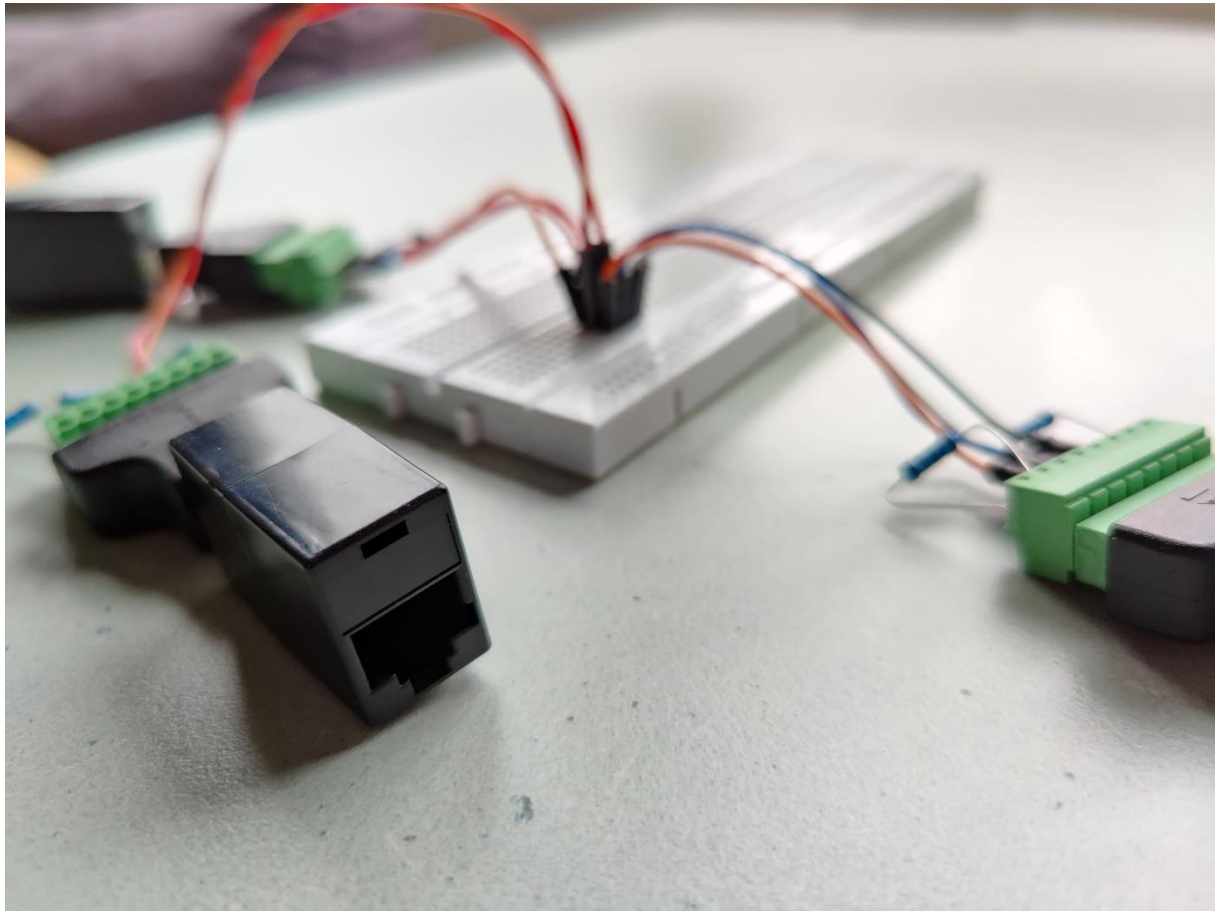
Building your own tap





Building your own tap





A closer look: ethtool -T

```
tsn@tsn-host ~ % sudo ethtool -T enp0s31f6
Time stamping parameters for enp0s31f6:
Capabilities:
        hardware-transmit
        software-transmit
        hardware-receive
        software-receive
        software-system-clock
        hardware-raw-clock
PTP Hardware Clock: 4
Hardware Transmit Timestamp Modes:
        off
        on
Hardware Receive Filter Modes:
        none
        all
        ptpv1-l4-sync
        ptpv1-l4-delay-req
        ptpv2-l4-sync
        ptpv2-l4-delay-req
        ptpv2-l2-sync
        ptpv2-l2-delay-req
        ptpv2-event
        ptpv2-sync
        ptpv2-delay-req
tsn@tsn-host ~ %
```

```
tsn@tsn-host ~ % sudo ethtool -T enp5s0f0
Time stamping parameters for enp5s0f0:
Capabilities:
        hardware-transmit
        software-transmit
        hardware-receive
        software-receive
        software-system-clock
        hardware-raw-clock
PTP Hardware Clock: 2
Hardware Transmit Timestamp Modes:
        off
        on
Hardware Receive Filter Modes:
        none
        all
tsn@tsn-host ~ %
```

```
alex@iris ~ % sudo ethtool -T enp37s0
Time stamping parameters for enp37s0:
Capabilities:
        software-transmit
        software-receive
        software-system-clock
PTP Hardware Clock: none
Hardware Transmit Timestamp Modes: none
Hardware Receive Filter Modes: none
alex@iris ~ %
```

```
tsn@tsn-host ~ % sudo ethtool -T enp7s0f4
Time stamping parameters for enp7s0f4:
Capabilities:
        hardware-transmit
        software-transmit
        hardware-receive
        software-receive
        software-system-clock
        hardware-raw-clock
PTP Hardware Clock: 5
Hardware Transmit Timestamp Modes:
        off
        on
Hardware Receive Filter Modes:
        none
        ptpv1-l4-sync
        ptpv1-l4-delay-req
        ptpv2-l4-event
        ptpv2-l4-sync
        ptpv2-l4-delay-req
tsn@tsn-host ~ %
```

A closer look: ethtool -T

```
tsn@tsn-host ~ % sudo ethtool -T enp0s31f6
Time stamping parameters for enp0s31f6:
Capabilities:
        hardware-transmit
        software-transmit
        hardware-receive
        software-receive
        software-system-clock
        hardware-raw-clock
PTP Hardware Clock: 4
Hardware Transmit Timestamp Mod
        off
        on
Hardware Receive Filter Modes:
        none
        all
        ptpv1-l4-sync
        ptpv1-l4-delay-req
        ptpv2-l4-sync
        ptpv2-l4-delay-req
        ptpv2-l2-sync
        ptpv2-l2-delay-req
        ptpv2-event
        ptpv2-sync
        ptpv2-delay-req
tsn@tsn-host ~ %
```

```
tsn@tsn-host ~ % sudo ethtool -T enp5s0f0
Time stamping parameters for enp5s0f0:
Capabilities:
        hardware-transmit
        software-transmit
        hardware-receive
        software-receive
```

TABLE I
EXAMPLE NICS AND THEIR TIME STAMPING CAPABILITIES.

Hardware	BW	Driver	HWTSTAMP_FILTER_ALL
Intel i210	1 Gbit/s	igb	yes
Intel I350-T2	1 Gbit/s	igb	yes
Intel X520-DA2	10 Gbit/s	ixgbe	no (only PTP)
Intel X710	10 Gbit/s	i40e	no (only PTP)
Chelsio BT-520	10 Gbit/s	cxgb4	no (only PTP)
Mellanox ConnectX-4 Lx	10 Gbit/s	mlx5	yes

```
Time stamping parameters for enp37s0:
Capabilities:
        software-transmit
        software-receive
        software-system-clock
PTP Hardware Clock: none
Hardware Transmit Timestamp Modes: none
Hardware Receive Filter Modes: none
alex@iris ~ %
```

```
tsn@tsn-host ~ % sudo ethtool -T enp7s0f4
Time stamping parameters for enp7s0f4:
Capabilities:
        hardware-transmit
        software-transmit
        hardware-receive
        software-receive
        ftware-system-clock
        dware-raw-clock
        e Clock: 5
        ansmit Timestamp Modes:

        ceive Filter Modes:
        e
        v1-l4-sync
        v1-l4-delay-req
        v2-l4-event
        v2-l4-sync
        v2-l4-delay-req
        t ~ %
```

What if I do not have the hardware yet? → Check the driver source

```
3140  static int ixgbe_get_ts_info(struct net_device *dev,
3141                               struct ethtool_ts_info *info)
3142  {
3143          struct ixgbe_adapter *adapter = netdev_priv(dev);
3144
3145          /* we always support timestamping disabled */
3146          info->rx_filters = BIT(HWTSTAMP_FILTER_NONE);
3147
3148          switch (adapter->hw.mac.type) {
3149          case ixgbe_mac_X550:
3150          case ixgbe_mac_X550EM_x:
3151          case ixgbe_mac_x550em_a:
3152                  info->rx_filters |= BIT(HWTSTAMP_FILTER_ALL);
3153                  break;
3154          case ixgbe_mac_X540:
3155          case ixgbe_mac_82599EB:
3156                  info->rx_filters |=
3157                          BIT(HWTSTAMP_FILTER_PTP_V1_L4_SYNC) |
3158                          BIT(HWTSTAMP_FILTER_PTP_V1_L4_DELAY_REQ) |
3159                          BIT(HWTSTAMP_FILTER_PTP_V2_EVENT);
3160                  break;
3161          default:
3162                  return ethtool_op_get_ts_info(dev, info);
3163          }
3164
3165          info->so_timestamping =
3166                  SOF_TIMESTAMPING_TX_SOFTWARE |
3167                  SOF_TIMESTAMPING_RX_SOFTWARE |
3168                  SOF_TIMESTAMPING_SOFTWARE |
3169                  SOF_TIMESTAMPING_TX_HARDWARE |
3170                  SOF_TIMESTAMPING_RX_HARDWARE |
3171                  SOF_TIMESTAMPING_RAW_HARDWARE;
3172
3173          if (adapter->ptp_clock)
3174                  info->phc_index = ptp_clock_index(adapter->ptp_clock);
3175          else
3176                  info->phc_index = -1;
3177
3178          info->tx_types =
3179                  BIT(HWTSTAMP_TX_OFF) |
3180                  BIT(HWTSTAMP_TX_ON);
3181
3182          return 0;
3183  }
```

ixgbe/ixgbe_ethtool.c

```
1018          case HWTSTAMP_FILTER_PTP_V1_L4_DELAY_REQ:
1019                  tsync_rx_ctl |= IXGBE_TSYNCRXCTL_TYPE_L4_V1;
1020                  tsync_rx_mtrl |= IXGBE_RXMTRL_V1_DELAY_REQ_MSG;
1021                  adapter->flags |= (IXGBE_FLAG_RX_HWTSTAMP_ENABLED |
1022                                     IXGBE_FLAG_RX_HWTSTAMP_IN_REGISTER);
1023                  break;
1024          case HWTSTAMP_FILTER_PTP_V2_EVENT:
1025          case HWTSTAMP_FILTER_PTP_V2_L2_EVENT:
1026          case HWTSTAMP_FILTER_PTP_V2_L4_EVENT:
1027          case HWTSTAMP_FILTER_PTP_V2_SYNC:
1028          case HWTSTAMP_FILTER_PTP_V2_L2_SYNC:
1029          case HWTSTAMP_FILTER_PTP_V2_L4_SYNC:
1030          case HWTSTAMP_FILTER_PTP_V2_DELAY_REQ:
1031          case HWTSTAMP_FILTER_PTP_V2_L2_DELAY_REQ:
1032          case HWTSTAMP_FILTER_PTP_V2_L4_DELAY_REQ:
1033                  tsync_rx_ctl |= IXGBE_TSYNCRXCTL_TYPE_EVENT_V2;
1034                  is_l2 = true;
1035                  config->rx_filter = HWTSTAMP_FILTER_PTP_V2_EVENT;
1036                  adapter->flags |= (IXGBE_FLAG_RX_HWTSTAMP_ENABLED |
1037                                     IXGBE_FLAG_RX_HWTSTAMP_IN_REGISTER);
1038                  break;
1039          case HWTSTAMP_FILTER_PTP_V1_L4_EVENT:
1040          case HWTSTAMP_FILTER_NTP_ALL:
1041          case HWTSTAMP_FILTER_ALL:
1042                  /* The X550 controller is capable of timestamping all packets,
1043                   * which allows it to accept any filter.
1044                   */
1045                  if (hw->mac.type >= ixgbe_mac_X550) {
1046                          tsync_rx_ctl |= IXGBE_TSYNCRXCTL_TYPE_ALL;
1047                          config->rx_filter = HWTSTAMP_FILTER_ALL;
1048                          adapter->flags |= IXGBE_FLAG_RX_HWTSTAMP_ENABLED;
1049                          break;
1050                  }
1051                  fallthrough;
```

ixgbe/ixgbe_ptp.c

What if I do not have the hardware yet? → Check the driver source

```
3148        switch (adapter->hw.mac.type) {
3149        case ixgbe_mac_X550:
3150        case ixgbe_mac_X550EM_x:
3151        case ixgbe_mac_x550em_a:
3152                info->rx_filters |= BIT(HWTSTAMP_FILTER_ALL);
3153                break;
3154        case ixgbe_mac_X540:
3155        case ixgbe_mac_82599EB:
3156                info->rx_filters |=
3157                        BIT(HWTSTAMP_FILTER_PTP_V1_L4_SYNC) |
3158                        BIT(HWTSTAMP_FILTER_PTP_V1_L4_DELAY_REQ) |
3159                        BIT(HWTSTAMP_FILTER_PTP_V2_EVENT);
3160                break;
```

```
1039    case HWTSTAMP_FILTER_PTP_V1_L4_EVENT:
1040    case HWTSTAMP_FILTER_NTP_ALL:
1041    case HWTSTAMP_FILTER_ALL:
1042            /* The X550 controller is capable of timestamping all packets,
1043             * which allows it to accept any filter.
1044             */
1045            if (hw->mac.type >= ixgbe_mac_X550) {
1046                    tsync_rx_ctl |= IXGBE_TSYNCRXCTL_TYPE_ALL;
1047                    config->rx_filter = HWTSTAMP_FILTER_ALL;
1048                    adapter->flags |= IXGBE_FLAG_RX_HWTSTAMP_ENABLED;
1049                    break;
1050            }
1051            fallthrough;
```

ixgbe/ixgbe_ethtool.c

Sometimes, this can be inconsistent…

```
1553  static int get_ts_info(struct net_device *dev, struct ethtool_ts_info *ts_info)
1554  {
1555          struct port_info *pi = netdev_priv(dev);
1556          struct  adapter *adapter = pi->adapter;
1557
1558          ts_info->so_timestamping = SOF_TIMESTAMPING_TX_SOFTWARE |
1559                                     SOF_TIMESTAMPING_RX_SOFTWARE |
1560                                     SOF_TIMESTAMPING_SOFTWARE;
1561
1562          ts_info->so_timestamping |= SOF_TIMESTAMPING_RX_HARDWARE |
1563                                      SOF_TIMESTAMPING_TX_HARDWARE |
1564                                      SOF_TIMESTAMPING_RAW_HARDWARE;
1565
1566          ts_info->tx_types = (1 << HWTSTAMP_TX_OFF) |
1567                              (1 << HWTSTAMP_TX_ON);
1568
1569          ts_info->rx_filters = (1 << HWTSTAMP_FILTER_NONE) |
1570                                (1 << HWTSTAMP_FILTER_PTP_V2_L4_EVENT) |
1571                                (1 << HWTSTAMP_FILTER_PTP_V1_L4_SYNC) |
1572                                (1 << HWTSTAMP_FILTER_PTP_V1_L4_DELAY_REQ) |
1573                                (1 << HWTSTAMP_FILTER_PTP_V2_L4_SYNC) |
1574                                (1 << HWTSTAMP_FILTER_PTP_V2_L4_DELAY_REQ);
1575
1576          if (adapter->ptp_clock)
1577                  ts_info->phc_index = ptp_clock_index(adapter->ptp_clock);
1578          else
1579                  ts_info->phc_index = -1;
1580
1581          return 0;
1582  }
```

cxgb4/cxgb4_ethtool.c

TABLE I
EXAMPLE NICS AND THEIR TIME STAMPING CAPABILITIES.

Hardware	BW	Driver	HWTSTAMP_FILTER_ALL
Intel i210	1 Gbit/s	igb	yes
Intel I350-T2	1 Gbit/s	igb	yes
Intel X520-DA2	10 Gbit/s	ixgbe	no (only PTP)
Intel X710	10 Gbit/s	i40e	no (only PTP)
Chelsio BT-520	10 Gbit/s	cxgb4	no (only PTP)
Mellanox ConnectX-4 Lx	10 Gbit/s	mlx5	yes

```
3120        case HWTSTAMP_FILTER_ALL:
3121        case HWTSTAMP_FILTER_PTP_V1_L4_SYNC:
3122        case HWTSTAMP_FILTER_PTP_V1_L4_DELAY_REQ:
3123        case HWTSTAMP_FILTER_PTP_V2_L4_SYNC:
3124        case HWTSTAMP_FILTER_PTP_V2_L4_DELAY_REQ:
3125                pi->rxtstamp = true;
3126                break;
3127        default:
3128                pi->tstamp_config.rx_filter =
3129                        HWTSTAMP_FILTER_NONE;
3130                return -ERANGE;
```

cxgb4/cxgb4_main.c

So how can we use it?

GitHub

https://github.com/lsinfo3/hwtstamp-snippets

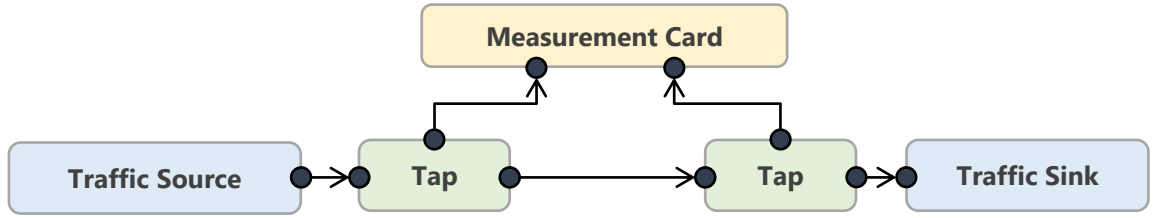
```
--time-stamp-type adapter_unsynced --time-stamp-precision nano
```

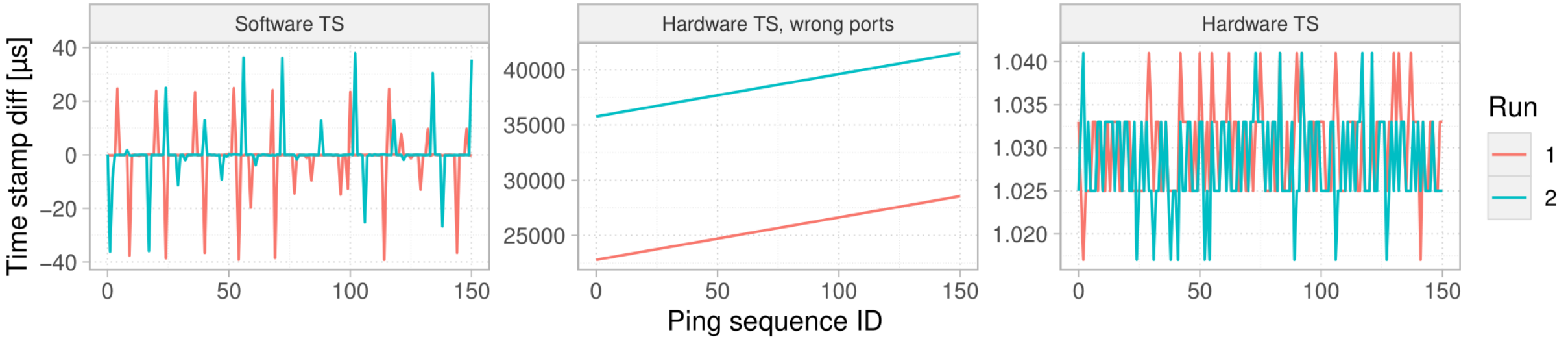
```
tsn@tsn-host ~ % sudo tcpdump -i enp0s31f6 -Q in --time-stamp-type adapter_unsynced --time-stamp-precision nano
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on enp0s31f6, link-type EN10MB (Ethernet), snapshot length 262144 bytes
07:18:32.183684837 IP _gateway.32812 > tsn-host.ssh: Flags [.], ack 1291875485, win 501, options [nop,nop,TS val 2935390078 ecr 1396821535], length 0
07:18:32.257323462 IP _gateway.32812 > tsn-host.ssh: Flags [.], ack 189, win 501, options [nop,nop,TS val 2935390152 ecr 1396821609], length 0
07:18:32.360020337 IP _gateway.32812 > tsn-host.ssh: Flags [.], ack 369, win 501, options [nop,nop,TS val 2935390254 ecr 1396821712], length 0
07:18:32.463440087 IP _gateway.32812 > tsn-host.ssh: Flags [.], ack 549, win 501, options [nop,nop,TS val 2935390358 ecr 1396821815], length 0
07:18:32.534335087 STP 802.1w, Rapid STP, Flags [Learn, Forward], bridge-id 8000.08:55:31:34:6f:1f.8003, length 36
07:18:32.566873212 IP _gateway.32812 > tsn-host.ssh: Flags [.], ack 849, win 501, options [nop,nop,TS val 2935390461 ecr 1396821919], length 0
07:18:32.670164587 IP _gateway.32812 > tsn-host.ssh: Flags [.], ack 1029, win 501, options [nop,nop,TS val 2935390565 ecr 1396822022], length 0
07:18:32.773561587 IP _gateway.32812 > tsn-host.ssh: Flags [.], ack 1209, win 501, options [nop,nop,TS val 2935390668 ecr 1396822125], length 0
07:18:32.876829087 IP _gateway.32812 > tsn-host.ssh: Flags [.], ack 1389, win 501, options [nop,nop,TS val 2935390771 ecr 1396822229], length 0
07:18:32.980157462 IP _gateway.32812 > tsn-host.ssh: Flags [.], ack 1569, win 501, options [nop,nop,TS val 2935390875 ecr 1396822332], length 0
07:18:33.083298587 IP _gateway.32812 > tsn-host.ssh: Flags [.], ack 1749, win 501, options [nop,nop,TS val 2935390978 ecr 1396822435], length 0
07:18:33.186833337 IP _gateway.32812 > tsn-host.ssh: Flags [.], ack 1929, win 501, options [nop,nop,TS val 2935391081 ecr 1396822539], length 0
07:18:33.290140962 IP _gateway.32812 > tsn-host.ssh: Flags [.], ack 2109, win 501, options [nop,nop,TS val 2935391185 ecr 1396822642], length 0
07:18:33.393027087 IP _gateway.32812 > tsn-host.ssh: Flags [.], ack 2289, win 501, options [nop,nop,TS val 2935391287 ecr 1396822745], length 0
07:18:33.496832712 IP _gateway.32812 > tsn-host.ssh: Flags [.], ack 2469, win 501, options [nop,nop,TS val 2935391391 ecr 1396822849], length 0
07:18:33.600216462 IP _gateway.32812 > tsn-host.ssh: Flags [.], ack 2649, win 501, options [nop,nop,TS val 2935391495 ecr 1396822952], length 0
```

If we can't use tcpdump? → Just use the right system calls

```
# create socket for ioctl call
s = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.htons(ETH_P_ALL))
s.bind((args.interface, 0))

# get the current device flags; 16sh = char[16] + short
ifr = struct.pack("16sh", args.int_b, 0)
req = fcntl.ioctl(s.fileno(), SIOCGIFFLAGS, ifr)
ifr_flags = struct.unpack("16sh", req)[1]

# add PROMISC flag and set flags back on the interface
ifr_flags |= IFF_PROMISC
ifr = struct.pack("16sh", args.int_b, ifr_flags)
if not fcntl.ioctl(s.fileno(), SIOCSIFFLAGS, ifr):
    raise ValueError(f"fcntl.ioctl(SIOCSIFFLAGS) returned False")

# request hardware timestamps and nanosecond resolution
#s.setsockopt(socket.SOL_SOCKET, SO_TIMESTAMPNS, 1)
s.setsockopt(socket.SOL_SOCKET, SO_TIMESTAMPING, SOF_TIMESTAMPING_RX_HARDWARE | SOF_TIMESTAMPING_RAW_HARDWARE)
conf = HWTSTAMP_CONFIG(0, HWTSTAMP_TX_OFF, HWTSTAMP_FILTER_ALL)
ifr = HWTSTAMP_IFREQ(args.int_b, pointer(conf))
if x := fcntl.ioctl(s.fileno(), SIOCSHWTSTAMP, ifr) != 0:
    raise ValueError(f"fcntl.ioctl(SIOCSHWTSTAMP) returned {x}")
```

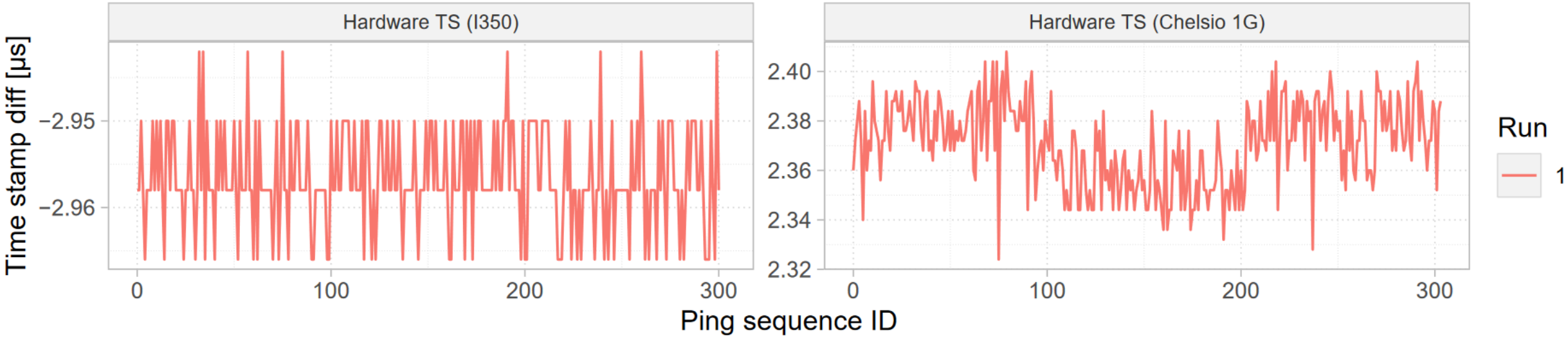
A few preliminary results



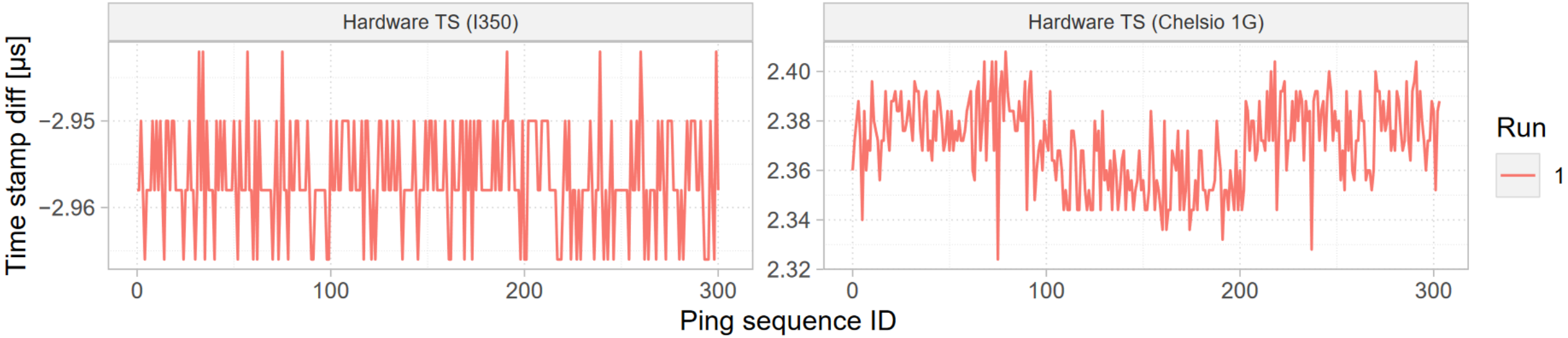


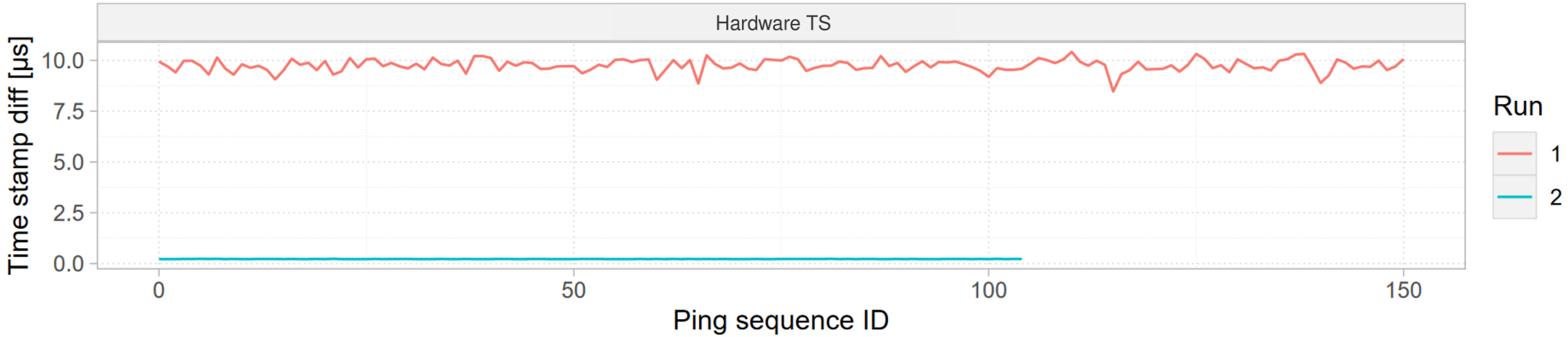
- Software TS shows jitter (±40 µs)
- Hardware TS with different clocks shows clock drift (±20 µs / second, or 20ppm, or 0.002%)
- Hardware TS with the same clock shows very stable results
  - Constant offset of ~ 1030 µs → **calibration measurements!**
  - Jitter of ±16 ns in steps of 8 ns

More results



More results





What if you need more performance?

```
for (;keep_running;) {
    recv_rc = recvmsg(fd_socket, &msg, MSG_DONTWAIT); // returns size or -1 in case of error
```

```
while True:
    # read 1 packet
    raw_data, ancdata, flags, address = s.recvmsg(65535, 1024)
```

- Usually endless loop with **recvmsg** calls
- Those are system calls!
- Improve performance by…
  - Using **recvmmsg** for multiple messages
  - Using **PACKET_RX_RING** socket option with **mmap** for zero-copy reception

```
ioctl(1, TCGETS, 0x7ffc73269a10)        = -1 ENOTTY (Inappropri
ioctl(3, SIOCGIFNAME, {ifr_ifindex=0})  = -1 ENODEV (No such de
ioctl(3, SIOCETHTOOL, 0x7ffc732699b0)   = 0
ioctl(4, SIOCGIFINDEX, {ifr_name="lo", ifr_ifindex=1}) = 0
ioctl(4, SIOCGIFHWADDR, {ifr_name="enp0s31f6", ifr_hwaddr={sa_f
ioctl(4, SIOCGIFINDEX, {ifr_name="enp0s31f6", ifr_ifindex=6}) =
setsockopt(4, SOL_PACKET, PACKET_ADD_MEMBERSHIP, {mr_ifindex=if
setsockopt(4, SOL_SOCKET, SO_TIMESTAMPNS_OLD, [1], 4) = 0
setsockopt(4, SOL_PACKET, PACKET_VERSION, [2], 4) = 0
setsockopt(4, SOL_PACKET, PACKET_RESERVE, [4], 4) = 0
ioctl(4, SIOCSHWTSTAMP, 0x7ffc73269970) = 0
setsockopt(4, SOL_PACKET, PACKET_TIMESTAMP, [64], 4) = 0
setsockopt(4, SOL_PACKET, PACKET_RX_RING, 0x7ffc73269950, 28) =
setsockopt(4, SOL_SOCKET, SO_ATTACH_FILTER, {len=1, filter=0x7f
setsockopt(4, SOL_SOCKET, SO_ATTACH_FILTER, {len=1, filter=0x55
```

Conclusion

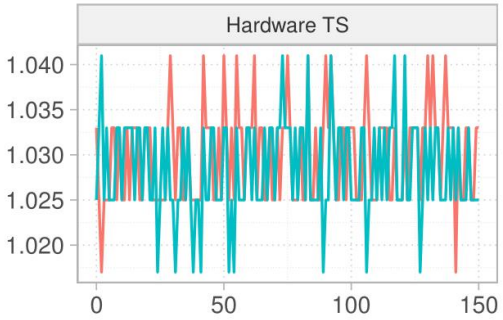
- Many new TSN and real-time use cases
- Accurate measurements required
- Dedicated measurement equipment can be really expensive

More Buzzwords

Brownfield deployments
Resource reservation
Guarantees
Shaping, Filtering, Switching
Black box devices
Measurements!

- In principle: just “good“ network taps and accurate time stamping
- Hardware time stamping NICs: what to look out for?
- Improve measurement accuracy by using the available features

- Evaluation shows sub-microsecond accuracy
- Using the same local clock avoids synchronization
- Calibration runs with direct cable are necessary



- Snippets are available on Github
- Throughput of measurements can be improved
- Zero-copy packet reception

GitHub

THANK YOU!

Questions, comments, suggestions?

Alexej Grigorjew

University of Wuerzburg
Chair of Communication Networks
Email: alexej.grigorjew@uni-wuerzburg.de