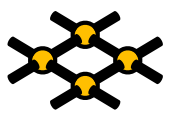
# Měření propustnosti sítě a analýza paketů

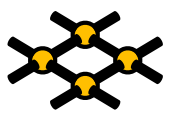
Petr Holub

PB156cv
2020–03–09

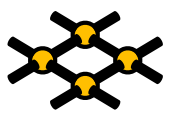


# Cíle cvičení

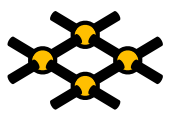
- Získat zpětnou vazbu na tvorbu protokolů na základě protokolů odevzdaných po minulém cvičení.
- Získat zkušenosti s analýzovou síťového provozu.
- Získat zkušenosti se základním testováním výkonu sítě.
- Vyzkoušet si základy aplikované statistiky pro vyhodnocování výsledků.



# Odevzdané protokoly



# Konfigurace počítačů



# Konfigurace počítačů

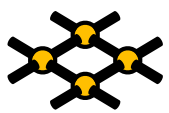
- Základní konfigurace sítě

```
ifconfig eth0 inet 10.1.1.2 netmask 255.255.255.0 up
netstat -rn
```

- Konfigurace sítě na Windows pomocí příkazové řádky

```
ipconfig /all
netsh interface ip show config
netsh interface ip set address name="Local Area Connection"
        static 192.168.1.2 255.255.255.0 192.168.1.1 1
netsh interface ip set dns "Local Area Connection" static 192.168.1.1

netsh interface ip set address "Local Area Connection" dhcp
netsh interface ip set dns "Local Area Connection" dhcp
```



# Konfigurace počítačů

- Informace o Ethernetových rozhraních:

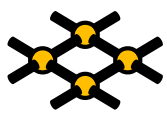
```
# ethtool eth2
# ethtool -S eth2
# netstat -s
```

- Nastavení MTU:

```
ifconfig eth0 mtu 9000
```

- Kontrola síťových bufferů `sysctl`:

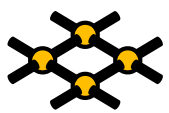
```
net.core.wmem_max
net.core.wmem_default
net.core.rmem_max
net.core.rmem_default
```



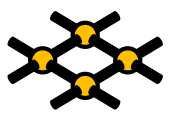
# Konfigurace počítačů

- Test průchodu paketů bez fragmentace:

```
ping -M do -s 8500 -c 5 1.2.3.4
From 1.2.3.4 icmp_seq=1 Frag needed and DF set (mtu = 1500)
```



# Analýza síťového provozu



# Nastavení počítače pro záchyt paketů

- Je vhodné minimalizovat vlastní provoz generovaný počítačem.
  - Vypnout demony typu Avahi

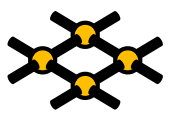
    ```
    service avahi-daemon stop
    ```
  - Nastavit statickou konfiguraci a vypnout automatické systémy (např. Network Manager na Linuxu)

    ```
    service network-manager stop
    ```
  - Vypnout IPv6

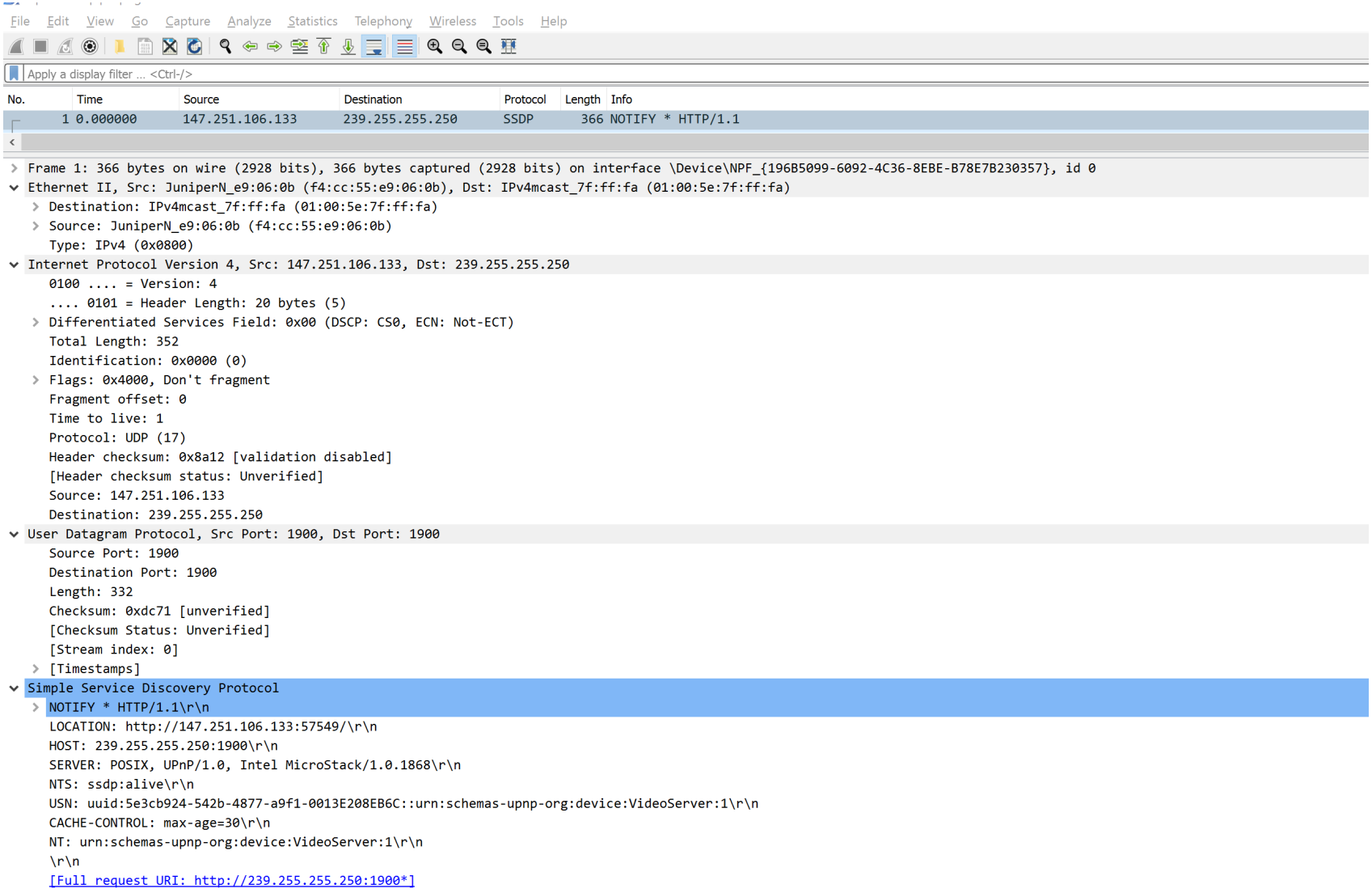
    ```
    echo 1 >/proc/sys/net/ipv6/conf/enp0s31f6/disable_ipv6
    ```
  - Odnastavit IPv4 adresy

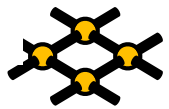
    ```
    ifconfig enp0s31f6 0
    ifconfig enp0s31f6 down
    ifconfig enp0s31f6 up
    ```
- tcpdump

  ```
  tcpdump -i eth0 -c 1000 -s 100 -w /tmp/file icmp
  ```



# Wireshark

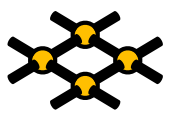

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000 147.251.106.133 239.255.255.250 SSDP 366 NOTIFY * HTTP/1.1
Frame 1: 366 bytes on wire (2928 bits), 366 bytes captured (2928 bits) on interface \Device\NPF_{196B5099-6092-4C36-8EBE-B78E7B230357}, id 0
Ethernet II, Src: JuniperN_e9:06:0b (f4:cc:55:e9:06:0b), Dst: IPv4mcast_7f:ff:fa (01:00:5e:7f:ff:fa)
Destination: IPv4mcast_7f:ff:fa (01:00:5e:7f:ff:fa)
Source: JuniperN_e9:06:0b (f4:cc:55:e9:06:0b)
Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 147.251.106.133, Dst: 239.255.255.250
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 352
Identification: 0x0000 (0)
Flags: 0x4000, Don't fragment
Fragment offset: 0
Time to live: 1
Protocol: UDP (17)
Header checksum: 0x8a12 [validation disabled]
[Header checksum status: Unverified]
Source: 147.251.106.133
Destination: 239.255.255.250
User Datagram Protocol, Src Port: 1900, Dst Port: 1900
Source Port: 1900
Destination Port: 1900
Length: 332
Checksum: 0xdc71 [unverified]
[Checksum Status: Unverified]
[Stream index: 0]
[Timestamps]
Simple Service Discovery Protocol
NOTIFY * HTTP/1.1\r\n
LOCATION: http://147.251.106.133:57549/\r\n
HOST: 239.255.255.250:1900\r\n
SERVER: POSIX, UPnP/1.0, Intel MicroStack/1.0.1868\r\n
NTS: ssdp:alive\r\n
USN: uuid:5e3cb924-542b-4877-a9f1-0013E208EB6C::urn:schemas-upnp-org:device:VideoServer:1\r\n
CACHE-CONTROL: max-age=30\r\n
NT: urn:schemas-upnp-org:device:VideoServer:1\r\n
\r\n
[Full request URI: http://239.255.255.250:1900*]



# Wireshark

```
> Frame 1: 366 bytes on wire (2928 bits), 366 bytes captured (2928 bits) on interface \Device\NPF_{196B5099-6092
v Ethernet II, Src: JuniperN_e9:06:0b (f4:cc:55:e9:06:0b), Dst: IPv4mcast_7f:ff:fa (01:00:5e:7f:ff:fa)
  > Destination: IPv4mcast_7f:ff:fa (01:00:5e:7f:ff:fa)
  > Source: JuniperN_e9:06:0b (f4:cc:55:e9:06:0b)
    Type: IPv4 (0x0800)
> Internet Protocol Version 4, Src: 147.251.106.133, Dst: 239.255.255.250
> User Datagram Protocol, Src Port: 1900, Dst Port: 1900
v Simple Service Discovery Protocol
  > NOTIFY * HTTP/1.1\r\n
    LOCATION: http://147.251.106.133:57549/\r\n
    HOST: 239.255.255.250:1900\r\n
    SERVER: POSIX, UPnP/1.0, Intel MicroStack/1.0.1868\r\n
    NTS: ssdp:alive\r\n
    USN: uuid:5e3cb924-542b-4877-a9f1-0013E208EB6C::urn:schemas-upnp-org:device:VideoServer:1\r\n
    CACHE-CONTROL: max-age=30\r\n
    NT: urn:schemas-upnp-org:device:VideoServer:1\r\n
    \r\n
    [Full request URI: http://239.255.255.250:1900*]
```

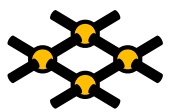
```
0000  01 00 5e 7f ff fa f4 cc  55 e9 06 0b 08 00 45 00   ..^..... U.....E.
0010  01 60 00 00 40 00 01 11  8a 12 93 fb 6a 85 ef ff   .`..@... ....j...
0020  ff fa 07 6c 07 6c 01 4c  dc 71 4e 4f 54 49 46 59   ...l.l.L .qNOTIFY
0030  20 2a 20 48 54 54 50 2f  31 2e 31 0d 0a 4c 4f 43    * HTTP/ 1.1..LOC
0040  41 54 49 4f 4e 3a 20 68  74 74 70 3a 2f 2f 31 34   ATION: h ttp://14
```



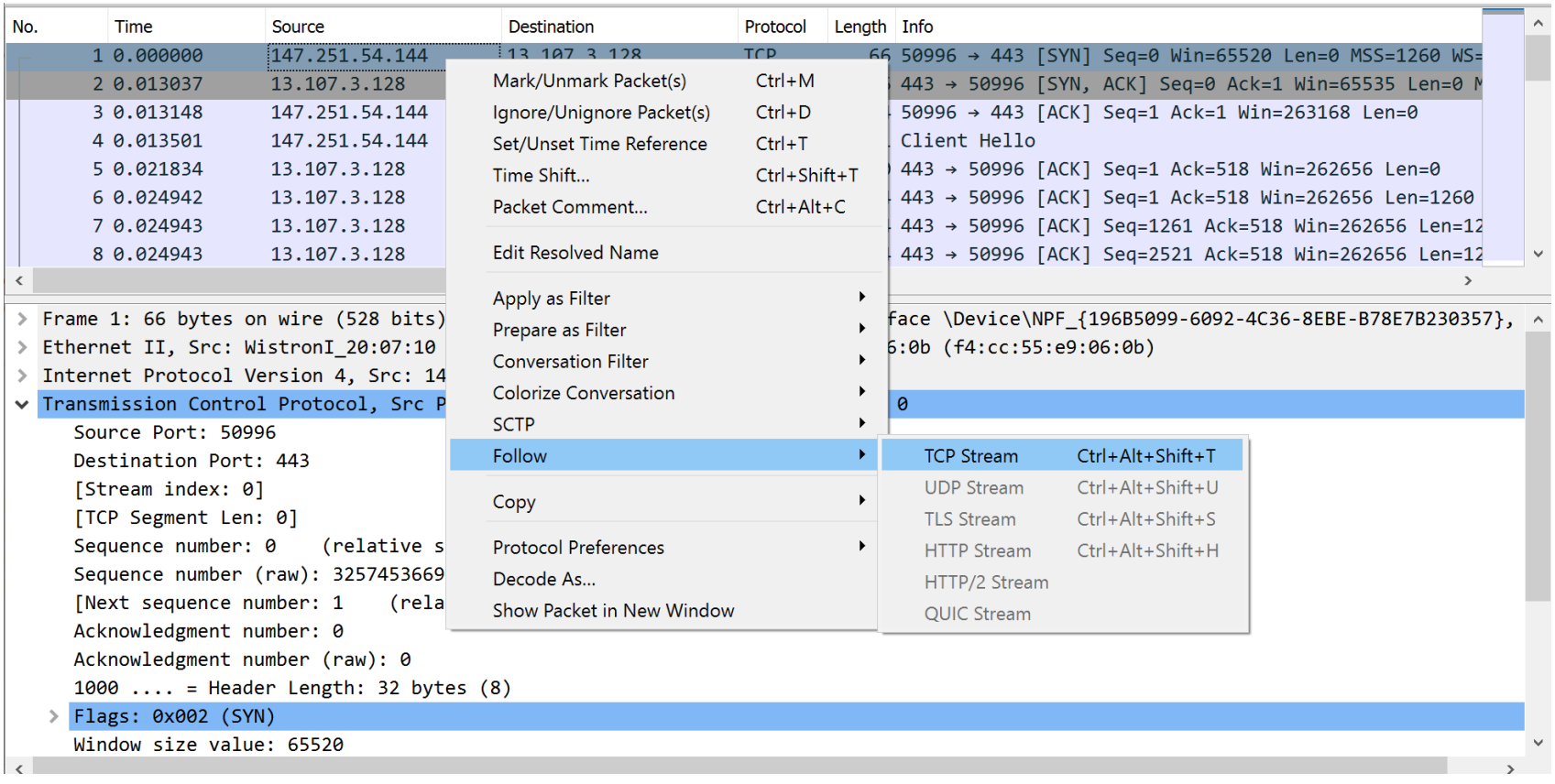
# Wireshark

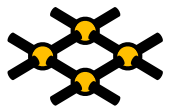
| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 1 | 0.000000 | 147.251.54.144 | 13.107.3.128 | TCP | 66 | 50996 → 443 [SYN] Seq=0 Win=65520 Len=0 MSS=1260 WS= |
| 2 | 0.013037 | 13.107.3.128 | 147.251.54.144 | TCP | 66 | 443 → 50996 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 M |
| 3 | 0.013148 | 147.251.54.144 | 13.107.3.128 | TCP | 54 | 50996 → 443 [ACK] Seq=1 Ack=1 Win=263168 Len=0 |
| 4 | 0.013501 | 147.251.54.144 | 13.107.3.128 | TLSv1.2 | 571 | Client Hello |
| 5 | 0.021834 | 13.107.3.128 | 147.251.54.144 | TCP | 60 | 443 → 50996 [ACK] Seq=1 Ack=518 Win=262656 Len=0 |
| 6 | 0.024942 | 13.107.3.128 | 147.251.54.144 | TCP | 1314 | 443 → 50996 [ACK] Seq=1 Ack=518 Win=262656 Len=1260 |
| 7 | 0.024943 | 13.107.3.128 | 147.251.54.144 | TCP | 1314 | 443 → 50996 [ACK] Seq=1261 Ack=518 Win=262656 Len=12 |
| 8 | 0.024943 | 13.107.3.128 | 147.251.54.144 | TCP | 1314 | 443 → 50996 [ACK] Seq=2521 Ack=518 Win=262656 Len=12 |

```
> Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NPF_{196B5099-6092-4C36-8EBE-B78E7B230357},
> Ethernet II, Src: WistronI_20:07:10 (48:2a:e3:20:07:10), Dst: JuniperN_e9:06:0b (f4:cc:55:e9:06:0b)
> Internet Protocol Version 4, Src: 147.251.54.144, Dst: 13.107.3.128
v Transmission Control Protocol, Src Port: 50996, Dst Port: 443, Seq: 0, Len: 0
    Source Port: 50996
    Destination Port: 443
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 0    (relative sequence number)
    Sequence number (raw): 3257453669
    [Next sequence number: 1    (relative sequence number)]
    Acknowledgment number: 0
    Acknowledgment number (raw): 0
    1000 .... = Header Length: 32 bytes (8)
  > Flags: 0x002 (SYN)
    Window size value: 65520
```

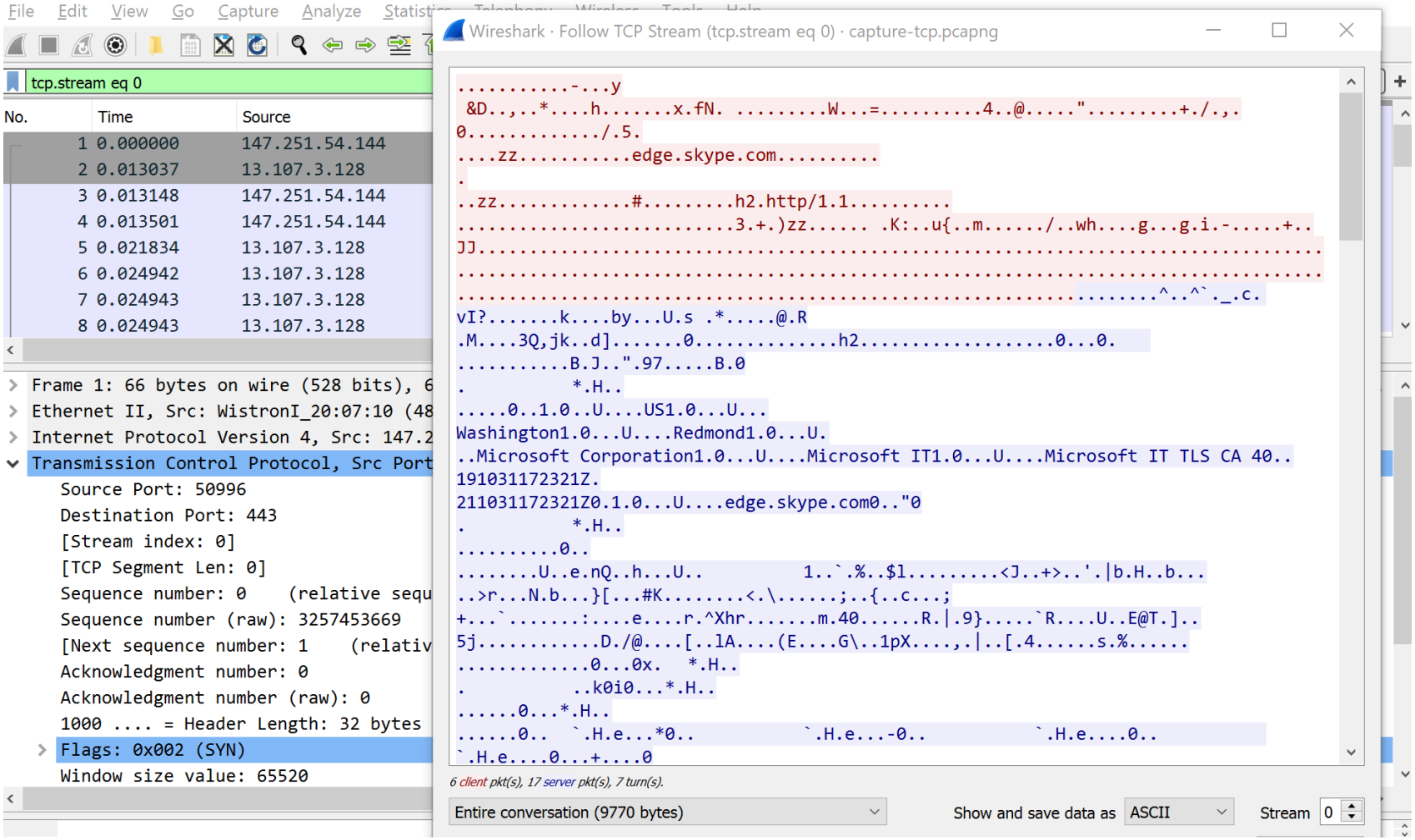


# Wireshark


No. Time Source Destination Protocol Length Info
1 0.000000 147.251.54.144 13.107.3.128 TCP 66 50996 → 443 [SYN] Seq=0 Win=65520 Len=0 MSS=1260 WS=
2 0.013037 13.107.3.128 443 → 50996 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 M
3 0.013148 147.251.54.144 50996 → 443 [ACK] Seq=1 Ack=1 Win=263168 Len=0
4 0.013501 147.251.54.144 Client Hello
5 0.021834 13.107.3.128 443 → 50996 [ACK] Seq=1 Ack=518 Win=262656 Len=0
6 0.024942 13.107.3.128 443 → 50996 [ACK] Seq=1 Ack=518 Win=262656 Len=1260
7 0.024943 13.107.3.128 443 → 50996 [ACK] Seq=1261 Ack=518 Win=262656 Len=12
8 0.024943 13.107.3.128 443 → 50996 [ACK] Seq=2521 Ack=518 Win=262656 Len=12
Frame 1: 66 bytes on wire (528 bits)
Ethernet II, Src: WistronI_20:07:10
Internet Protocol Version 4, Src: 14
Transmission Control Protocol, Src P
Source Port: 50996
Destination Port: 443
[Stream index: 0]
[TCP Segment Len: 0]
Sequence number: 0 (relative s
Sequence number (raw): 3257453669
[Next sequence number: 1 (rela
Acknowledgment number: 0
Acknowledgment number (raw): 0
1000 .... = Header Length: 32 bytes (8)
Flags: 0x002 (SYN)
Window size value: 65520
face \Device\NPF_{196B5099-6092-4C36-8EBE-B78E7B230357},
6:0b (f4:cc:55:e9:06:0b)
0
Mark/Unmark Packet(s) Ctrl+M
Ignore/Unignore Packet(s) Ctrl+D
Set/Unset Time Reference Ctrl+T
Time Shift... Ctrl+Shift+T
Packet Comment... Ctrl+Alt+C
Edit Resolved Name
Apply as Filter
Prepare as Filter
Conversation Filter
Colorize Conversation
SCTP
Follow
Copy
Protocol Preferences
Decode As...
Show Packet in New Window
TCP Stream Ctrl+Alt+Shift+T
UDP Stream Ctrl+Alt+Shift+U
TLS Stream Ctrl+Alt+Shift+S
HTTP Stream Ctrl+Alt+Shift+H
HTTP/2 Stream
QUIC Stream

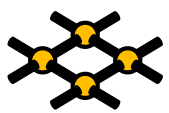


# Wireshark


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp.stream eq 0
No. Time Source
1 0.000000 147.251.54.144
2 0.013037 13.107.3.128
3 0.013148 147.251.54.144
4 0.013501 147.251.54.144
5 0.021834 13.107.3.128
6 0.024942 13.107.3.128
7 0.024943 13.107.3.128
8 0.024943 13.107.3.128
Frame 1: 66 bytes on wire (528 bits), 6
Ethernet II, Src: WistronI_20:07:10 (48
Internet Protocol Version 4, Src: 147.2
Transmission Control Protocol, Src Port
Source Port: 50996
Destination Port: 443
[Stream index: 0]
[TCP Segment Len: 0]
Sequence number: 0 (relative sequ
Sequence number (raw): 3257453669
[Next sequence number: 1 (relativ
Acknowledgment number: 0
Acknowledgment number (raw): 0
1000 .... = Header Length: 32 bytes
Flags: 0x002 (SYN)
Window size value: 65520
Wireshark · Follow TCP Stream (tcp.stream eq 0) · capture-tcp.pcapng
.....-...y
&D..,..*....h.....x.fN. .....W...=.....4..@....."....+./.,.
0...../.5.
....zz.....edge.skype.com.....
.
..zz.....#.....h2.http/1.1.....
.....3.+.)zz..... .K:..u{..m...../..wh....g...g.i.-.....+..
JJ.....
.....^..^`._.c.
vI?.....k....by...U.s .*.....@.R
.M....3Q,jk..d].....0.....h2.....0...0.
.....B.J..".97.....B.0
. *.H..
.....0..1.0..U....US1.0...U...
Washington1.0...U....Redmond1.0...U.
..Microsoft Corporation1.0...U....Microsoft IT1.0...U....Microsoft IT TLS CA 40..
1910311723212.
2110311723212Z0.1.0...U....edge.skype.com0.."0
. *.H..
.....0..
.....U..e.nQ..h...U.. 1..`.%..$1.....<J..+>..'.|b.H..b...
..>r...N.b...}[...#K.....<.\.....;..{..c...;
+...`.....:....e....r.^Xhr.....m.40.....R.|.9}.....`R....U..E@T.]..
5j.....D./@....[..lA....(E....G\..1pX....,.|..[.4.....s.%.....
.....0...0x. *.H..
. ..k0i0...*.H..
.....0...*.H..
.....0.. `.H.e...*0.. `.H.e...-0.. `.H.e....0..
`.H.e....0...+....0
6 client pkt(s), 17 server pkt(s), 7 turn(s).
Entire conversation (9770 bytes)
Show and save data as ASCII
Stream 0

# Zadání

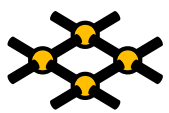
- V síti bude připojen generátor rámců. Odchytněte minimálně 100 rámců, ideálně z více různých síťových toků. Charakterizujte toky, které vidíte a vyberte z nich 2 pakety (z různých toků) a proveďte co nejpodrobnější analýzu jejich hlaviček.
  - Při odchytávání provozu dbejte na to, aby vaše počítače neposílaly do sítě zbytečné rámce.
  - Vyzkoušejte si zachycení provozu na počítači bez GUI (`tcpdump`) a následnou analýzu na vlastním počítači (Wireshark).



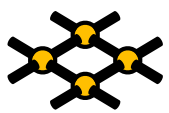
# Protokol

Každý samostatně zpracuje a odevzdá protokol. Protokol musí obsahovat minimálně následující části:

- charakterizaci typů provozu, které se podařilo zachytit a k čemu se používají,
- co nejpodrobnější analýzu hlaviček 2 paketů (z různých toků)
- do odevzdávárny nahrajte také výsledny PCAP

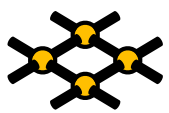


# Měření end-to-end propustnosti sítě



# Trocha “teorie”

- Omezení propustnosti spojů
- Omezení propustnosti síťových prvků
- Omezení propustnosti koncových zařízení
- Závislost na velikosti paketů



# Provádění měření

- iperf UDP

```
iperf -s -u -i 1 -l 8500
iperf -u -c hostname -i 1 -l 8500 -b 10M
```

- iperf TCP

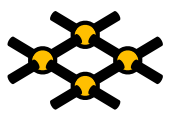
```
iperf -s -i 1 -w 8M
iperf -c hostname -i 1 -w 8M
```

- netperf UDP

```
netserver -n 4
netperf -H 10.0.10.1 -n 4 -t UDP_STREAM -- -s 8M -S 8M -m nnnn -M nnnn
```

- netperf TCP

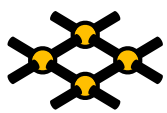
```
netserver -n 4
netperf -H 10.0.10.1 -n 4 -t TCP_STREAM -- -s 8M -S 8M -m nnnn -M nnnn
```



# Provádění měření

- nuttcp – trocha zábavy:

```
for h in 1.2.3.4 2.3.4.5; do for j in r t;
        do echo "";
        if [ "$j" = "r" ]; then echo "From $h to server";
        else echo "From server to $h"; fi;
                (for i in 200 400 600 800;
                        do ./nuttcp -i5 -T10 -u -R${i}M -v -v \
                                -${j} ${h};
                done ) | fgrep loss ;
        done;
done
```



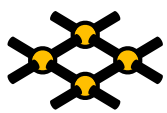
# Základy zpracování v R I

- Použijte buď samotné R[1] nebo RStudio[2]
- Jak instalovat balíky:

```
install.packages("plyr")
install.packages("tidyverse")
install.packages("Rmisc")
install.packages("kSamples")
install.packages("anomalize")
```

- Nahrání dat např.

```
nut <- scan("run1/nuttcp-forth-128.values", what=numeric())
```



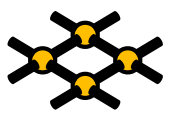
# Základy zpracování v R II

- Řetězení zpracování pomocí tidyverse

```
s <- c("32", "64", "128", "256", "512", "1024")
file_names <- paste0("run1/nuttcp-forth-", s, ".values")
nuts <- t(do.call(rbind,lapply(file_names,scan,what=as.numeric())))
colnames(nuts) <- s
nuts <- data.frame(nuts)
nuts %>% slice(2:n()) %>% lapply(CI)
nuts %>% lapply(shapiro.test)
```

[1] https://www.r-project.org/

[2] https://rstudio.com/

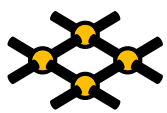


# Testy normality rozdělení

- test normality Shapiro-Wilk

```
> shapiro.test(nut)
W = 0.63028, p-value = 4.978e-11
> shapiro.test(nut[2:60])
W = 0.98598, p-value = 0.7302
```

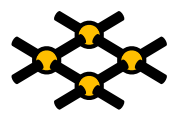
- nulová hypotéza: rozdělení je normální
- pokud je $p$-hodnota $< .05$, hypotézu zamítneme
- pokud je $p$-hodnota $> .05$, hypotézu nemůžeme zamítnout
- proč je hranice $p$-hodnoty 0.05?



# Charakterizace rozdělení I

- Průměrování a konfidenční interval se typicky počítá s předpokladem normálního rozdělení. Pro charakterizaci nenormálních rozdělení je lépe používat medián a kvantily:

```
> summary(nut)
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
64.49   64.49   64.49   64.50   64.50   64.52
> CI(nut)
upper     mean    lower
64.49631 64.49523 64.49415 > quantile(nut, prob = seq(0, 1, length = 21))
0%       5%       10%      15%      20%      25%      30%      35%
64.48900 64.49099 64.49209 64.49265 64.49318 64.49340 64.49357 64.49367
40%      45%      50%      55%      60%      65%      70%      75%
64.49422 64.49446 64.49470 64.49495 64.49512 64.49580 64.49610 64.49638
80%      85%      90%      95%     100%
64.49672 64.49748 64.49842 64.49882 64.52190
```



# Porovnání rozdělení I

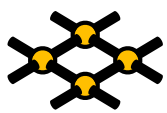
- Mějme 3 experimenty vzorkující data (nut, nut2, nut3)
- Nulová hypotéza: data pochází ze stejného rozdělení
- Neparametrické testy v případě, že máme/můžeme mít nenormální rozdělení
- Test Kolmogorov-Smirnov (K-S):

```
> ks.test(nut,nut2)

Two-sample Kolmogorov-Smirnov test

data:  nut and nut2
D = 0.083333, p-value = 0.9853
alternative hypothesis: two-sided

Warning message:
In ks.test(nut, nut2) : cannot compute exact p-value with ties
```



# Porovnání rozdělení II

- Test Anderson-Darling (A-D):
  - A-D více zohledňuje konce rozdělení

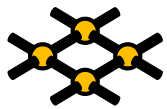
```
> library("kSamples")
> ad.test(nut,nut2,nut3)
Number of samples:  3
Sample sizes:  60, 60, 60
Number of ties: 100

Mean of  Anderson-Darling  Criterion: 2
Standard deviation of  Anderson-Darling  Criterion: 1.06249

T.AD = ( Anderson-Darling  Criterion - mean)/sigma

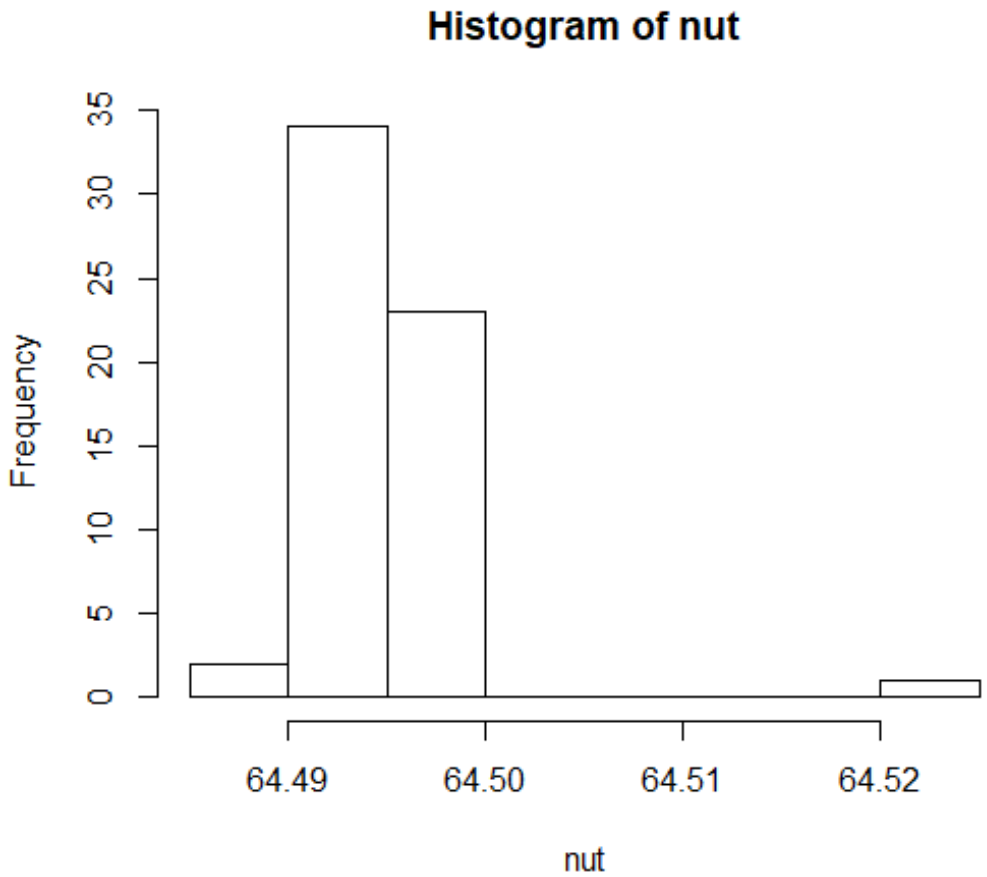
Null Hypothesis: All samples come from a common population.

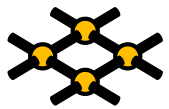
AD   T.AD  asympt. P-value
version 1: 0.6283 -1.291          0.9888
version 2: 0.5950 -1.322          0.9922
```



# Vizuální kontrola I

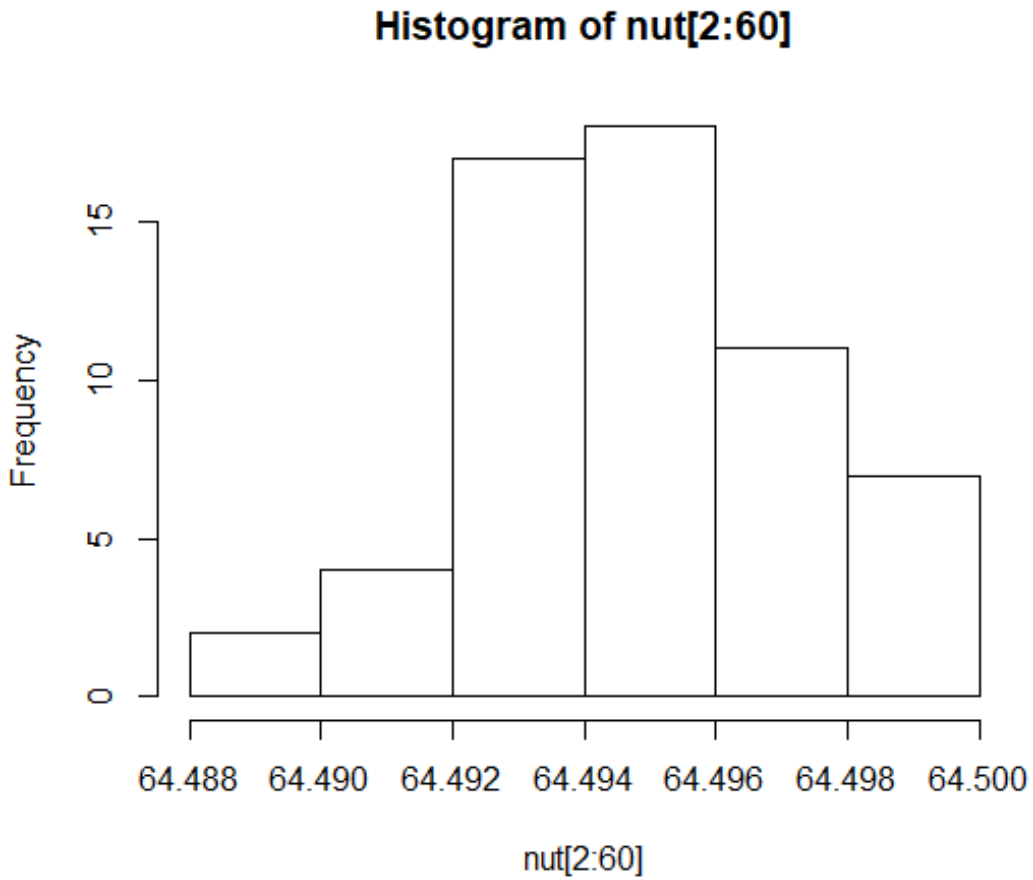
- Histogram

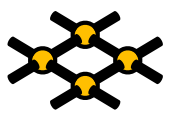


# Vizuální kontrola II

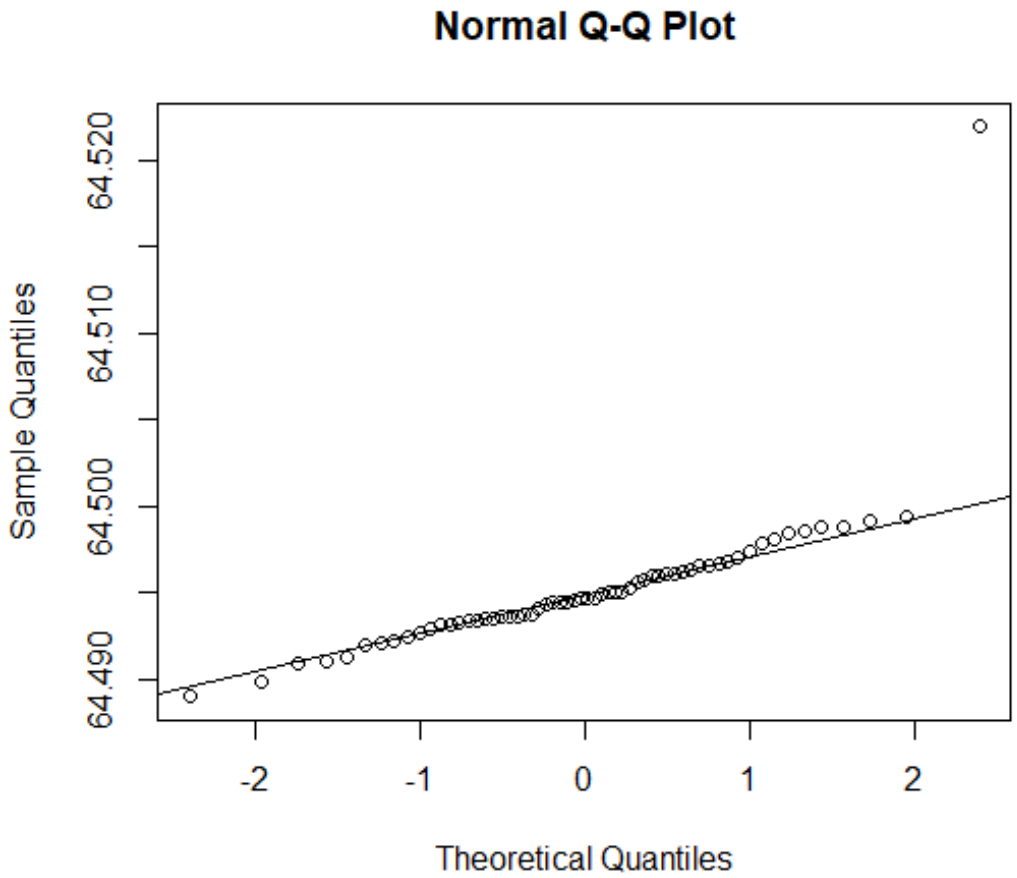
- Histogram po odstranění hrubé chyby

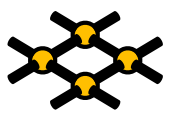


# Vizuální kontrola III

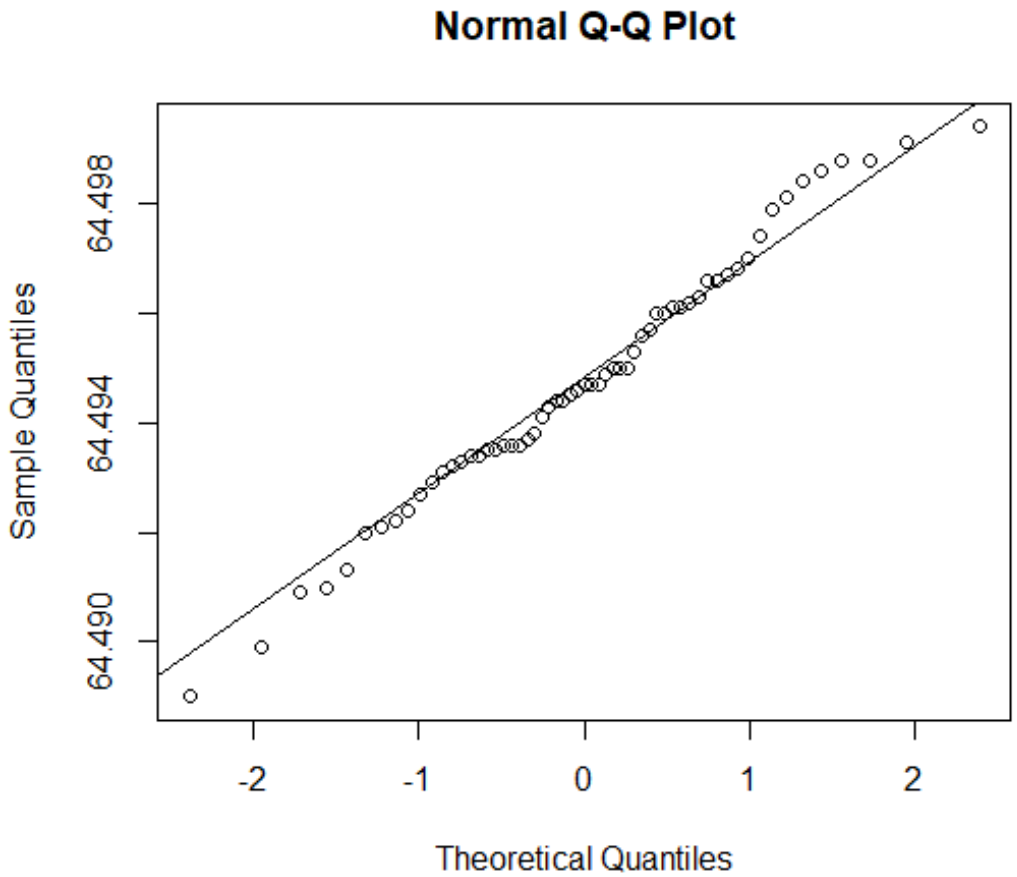
- Q-Q



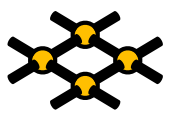
# Vizuální kontrola IV

- Q-Q po odstranění hrubé chyby



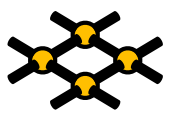

# Zadání I

- Experiment budete provádět ve dvojicích, zpracování výsledků a protokol samostatně.
- Připojte dva notebooky k 100Mbps přepínači
- Nastavte pevně danou IPv4 adresu na ethernetovém rozhraní
  - `service network-manager stop`
  - `ifconfig`
    zjistěte si jméno rozhraní
  - `ifconfig rozhrani adresa/24`
    přičemž použijete privátní adresy z rozsahu 192.168.0.0/16 a domluvíte se mezi sebou na jednom přepínači, abyste si nepřidělili duplicitní adresy
- Na jednom z počítačů nastartujte nuttcp server
  - `nuttcp -S`
- Na druhém spusťte měření propustnosti UDP v závislosti na velikosti paketů pro velikosti bufferu 16, 32, 64, 128, 256, 512, 1024, 1500 v obou směrech (meření v opačném směru zajišťuje přepínač `-r`):
- Všimněte si, co se stalo pro velikosti $< 32$ a $> 1448$ a v protokolu vysvětlete.



# Zadání II

- Zvažte použití automatizace shellovým skriptováním, např.:
  - ```
    for delka in 16 32 64 128 256 512 1024 1500; do \
    nuttcp -u -l ${delka} -R100M -i -T20 cil_ip >nuttcp-${delka}.log; done
    ```
- Měření celkem 3× zopakujte pro ověření stability výsledků. Zamyslete se, co byste měli správně provést, abyste provedli nezávislá měření (m.j. systém uvedli do výchozího stavu).
- Stáhněte si dosažené výsledky.


    ```

# Protokol I

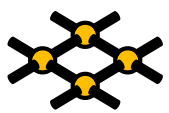
Každý samostatně zpracuje a odevzdá protokol.

- Výsledkem měření propustností jsou časové řady: 60 s běhy jsou rozděleny na 1 s intervaly. Data z logu můžete extrahovat např. takto:

```
for i in *.log; do cat $i | perl -pe 's/^.*\s+(\d+\.\d+)\sMbps\s+.*$/\1/' |
head -n -2 >${i%%.log}.values; done
```

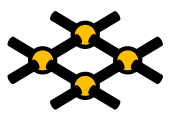
- Načtěte časové řady do R, např.

```
nut <- scan("run1/nuttcp-128.values", what=numeric())
```



# Protokol II

- *Hypotéza: časové řady mají normální rozdělení.* Lze nebo nelze tuto hypotézu vyvrátit pomocí Shapiro-Wilkova testu normality?
  `shapiro.test(nut)`
  Srovnejte vizuálně s histogramem a Q-Q diagramem:
  `hist(nut)`
  `qqnorm(nut); qqline(nut)`
- Lze-li tuto hypotézu vyvrátit, dokážete ji vyvrátit i v případě, pokud se do měření nezahrnou hodnoty, které byly naměřeny při současně registrovaných výpadcích – tedy hodnoty zatížené hrubou chybou (outliers)? Jde-li pouze o první hodnotu (častý případ – ale ověřte!), můžte jednoduše použít výběr `nut[2:60]`.



# Protokol III

- *Hypotéza: propustnost síťové karty a sítě nezávisí na velikosti paketů.* Lze tuto hypotézu vyvrátit nebo nelze? Zdůvodněte. Pokud jsou rozdělení přibližně normální, využijte průměrné hodnoty měření pro jednotlivé velikost paketů s odhadnutou chybou měření při zvolené hranici intervalu spolehlivosti (např. 95%). V opačném případě použijte medián s uvedením rozsahu 1.–3. kvartilu. Příklad v R pro normální rozdělení:

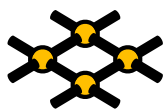
  `library("Rmisc")` – příp. `install.packages("Rmisc")`, pokud je třeba instalace

  `CI(nut, ci=.95)`

  a pro jiné než normální rozdělení

  `summary(nut)`

  `quantile(nut, prob = seq(0, 1, length = 21))`



# Protokol IV

- *Hypotéza: opakovaná měření poskytují výsledky vzorkované ze stejného rozdělení.* Lze tuto hypotézu vyvrátit nebo nelze? Zdůvodněte. Můžete využít např. Anderson-Darling k-vzorkového testu.

```
library("kSamples")
nut <- scan("run1/nuttcp-forth-128.values", what=numeric())
nut2 <- scan("run2/nuttcp-forth-128.values", what=numeric())
ad.test(nut,nut2)
```

- *Všechny presentované výsledky musí být řádně zaokrouhleny a musí být uvedeny jednotky měření.*