May 8-9 2025, Zürich

Using nDPI to solve real life problems: from First Packet Classification to Obfuscated Traffic detection

Ivan Nardi: ivan@ai2m.eu, @i_nardi, @i_nardi@mastodon.social

AI2M srl

PacketFest'25

Agenda

- We are going to a look at some practical, real life problems that the latest nDPI versions can easily help you to solve. Let's try to avoid technical, low-level DPI details
  - Monitoring
  - Just the (sub-)classification
  - First Packet Classification
  - Obfuscated traffic detection
- nDPI: a new feature(?)

Monitoring

- nDPI usually processes only the first N packets of a flow (1-2 for UDP; 4-10 for TCP). After that (i.e. after having sub-classified the flow and after having extracted all the metadata), the packets of the flow are no more analyzed
- Can I have some detailed statistics about the entire flows (packet size, inter-arrival time, TCP retransmissions, RTT, MOS for RTP...)?
- Can I extract all the DNS transactions from a DNS flow?
- Can I extract ALL the username-IP pairs from an AAA flow? (Radius, Diameter, GTP-C)
- Bottom line: can nDPI process ALL the packets (of some) flow?

Monitoring: example

- UDP 192.168.12.67:45419 <-> 35.219.252.146:3478 [proto: 78.269/STUN.SignalVoip]...[DPI packets: 7][29 pkts/3570 bytes <-> 29 pkts/4210 bytes]...[Mapped IP/Port: 93.35.168.30:45250]...

- UDP 192.168.12.67:45419 <-> 35.219.252.146:3478 [proto: 78.269/STUN.SignalVoip]...[DPI packets: 58][29 pkts/3570 bytes <-> 29 pkts/4210 bytes]...[Mapped IP/Port: 93.35.168.30:45250, 35.219.226.11:54116, 35.219.252.146:22269, 35.219.226.11:12261]...

Just the classification

- nDPI gives you classification, metadata and flow risks.
- Can nDPI give me only flow (sub-)classification?
  - Typical user case: block some traffic

Just the classification: example

- TCP 192.168.1.6:60555 <-> 52.114.77.33:443 [proto: 91.219/TLS.Microsoft365]...[DPI packets: 11][Hostname/SNI: mobile.pipe.aria.microsoft.com]...[ServerNames: *.events.data.microsoft.com,events.data.microsoft.com,*.pipe.aria.microsoft.com,pipe.skype.com,*.pipe.skype.com,*.mobile.events.data.microsoft.com,mobile.events.data.microsoft.com,*.events.data.msn.com,events.data.msn.com][JA3S: 986571066668055ae9481cb84fda634a][Issuer: C=US, ST=Washington, L=Redmond, O=Microsoft Corporation, OU=Microsoft IT, CN=Microsoft IT TLS CA 4][Subject: CN=*.events.data.microsoft.com][Certificate SHA-1: 33:B3:B7:E9:DA:25:F5:A0:04:E9:63:87:B6:FB:54:77:DB:ED:27:EB][Firefox][Validity: 2019-10-10 21:55:38 - 2021-10-10 21:55:38][Cipher: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384]...

- TCP 192.168.1.6:60555 <-> 52.114.77.33:443 [proto: 91.219/TLS.Microsoft365]...[DPI packets: 4][Hostname/SNI: mobile.pipe.aria.microsoft.com]...

First Packet Classification

- Can I make routing decisions (on different interfaces) based on flow protocol?
- Can I implement some kind of VPN splitting?
- Can I apply different QoS profiles for different protocols?
- Can I send only some traffic (unknown? Windows updates?) to my IDS (for performance reasons)?
- Bottom line: can nDPI provide final flow (sub-)classification on the very FIRST packet of the flow?

First Packet Classification

- FPC is not perfect: you should expect same false positives and same false negatives
- Trade-off between classification accuracy and classification availability
- FPC is (usually!) not about performance!
  - It is about executing network policies fast and without latency

FPC: examples

- UDP 168.144.64.5:63736 -> 213.188.47.247:443 [proto: 188.124/QUIC.YouTube]...[FPC: 188.124/QUIC.YouTube]...
- TCP 192.168.1.7:53250 <-> 52.41.30.5:443 [proto: 91.133/TLS.NetFlix]...[FPC: 133/NetFlix]...
- TCP 192.168.0.103:60908 <-> 46.33.70.136:443 [proto: 91.211/TLS.Instagram]...[FPC: 211/Instagram]...
- UDP 192.168.12.156:49526 <-> 157.240.196.62:3478 [proto: 78.45/STUN.WhatsAppCall]... [FPC: 78.45/STUN.WhatsAppCall]
- TCP 192.168.2.100:51724 -> 179.60.195.49:5222 [proto: 142/WhatsApp]...[FPC: 142/WhatsApp]...
- TCP 192.168.1.117:50015 <-> 66.40.180.215:3724 [proto: 76/WorldOfWarcraft]...[FPC: 213/Blizzard]...

Obfuscated traffic: introduction

- Traffic content is almost always encrypted, but traffic type/classification (or some traffic characteristics) can be usually inferred anyway. This is basically the goal of any DPI/Network Visibility/Firewall system
- Techniques to avoid detection are often called "obfuscation": the general idea is to make the traffic "look like" something else, hiding its true nature. If it works, the obfuscated traffic will be detected as something else (different traffic type or classification): the DPI system has been fooled/bypassed
- There are two general strategies to obfuscate the traffic:
  - to mimic some content that it is allowed. Example: encapsulate the traffic in a TLS tunnel
  - to randomize the flow content, making it dissimilar to anything that it is specifically blocked. Example: (fully) encrypt (again) the traffic, removing any plain text info/magic word/common patterns

Obfuscated traffic: VPN

- VPN is not an obfuscation technique, because they are easily detectable
  - Using a VPN you "hide" your traffic content, but you don't "hide" the fact that you are using it
- Sometimes you might want to obfuscate the VPN traffic itself!
  - However all VPN apps have at least one option to enable some kind of obfuscation

Obfuscated traffic: detection

- Detecting obfuscated traffic doesn't mean to be able to tell the "real" (i.e. original) traffic type, but it is usually enough to know that some kind of obfuscation algorithm has been used
  - Obfuscated traffic is (very) suspicious per se
- Can nDPI detect (some kind of) obfuscated traffic?
  - OpenVPN obfuscated via XOR patch
  - Encapsulated TLS traffic (i.e. TLS-in-TLS)
  - Fully encrypted traffic

Obfuscated traffic: examples

- TCP 107.161.86.131:443 <-> 192.168.12.156:48072 [proto: 159/OpenVPN]...[Risk: Obfuscated Traffic]...[Risk Info: Obfuscated OpenVPN]
- TCP 192.168.12.156:37976 <-> 185.128.25.99:465 [proto: 91/TLS]...[Risk: Obfuscated Traffic]...[Risk Info: Obfuscated TLS traffic]
- TCP 127.0.0.1:44276 <-> 127.0.0.1:8388 [proto: 0/Unknown]...[Risk: Obfuscated Traffic]...[Risk Info: Fully Encrypted]
- TCP 127.0.0.1:41796 <-> 127.0.0.1:1234 [proto: 91/TLS]...[Risk: Obfuscated Traffic]...[Risk Info: Obfuscated TLS-in-TLS traffic]
- TCP 127.0.0.1:33702 <-> 127.0.0.1:1234 [proto: 7.251/HTTP.WebSocket]...[Risk: Obfuscated Traffic]...[Risk Info: Obfuscated TLS-in-HTTP-WebSocket traffic]
- TCP 172.18.82.242:41986 <-> 172.18.82.243:80 [proto: 7.251/HTTP.WebSocket]...[Risk: Obfuscated Traffic]...[Risk Info: Obfuscated SSH-in-HTTP-WebSocket traffic]
- TCP 172.30.84.193:40640 <-> 208.253.217.142:443 [proto: 91/TLS]...[Risk: Obfuscated Traffic]...[Risk Info: Abnormal Client Hello/Padding length]

A new feature: Multimedia Calls visibility

- Improve multimedia calls visibility:
  - from “flows” to “call” with its own metadata: duration, type, quality, direction, list of flows and multimedia streams…

- Very preliminary studies. Extremely complex:
  - Standard protocols but different non-compliant implementations
  - No documentation: we need to reverse-engineer the logic
  - Different applications have different (network) behaviors
  - Even the same application has different behaviors depending on OS, app vs browser, browser type, p2p vs group call, where you capture/tap the traffic…
    - The metadata set is not uniform!
  - Changes over time!

Multimedia Calls visibility: example 1

```
./example/ndpiReader -t -A -d -i ~/signal_mia3_android11_app_7.26.1_2_videocalls_1_to_lte_1_from_lte_both_answered.pcapng
[...]
SignalVoip          packets: 4329          bytes: 2502807       flows: 25
[...]
Num async events (calls): 2
Call [269 - SignalVoip]:
 Served: 192.168.1.181:45735
 Duration: 9 [s]
 Multiparty: 2 [Peer2Peer]
 Direction: 2 [Terminated (served called)]
 T init:   10/Dec/2024 17:23:46
 T start:  10/Dec/2024 17:23:55
 T end:    10/Dec/2024 17:24:04
 T deinit: 10/Dec/2024 17:24:04
 Flows:
  0) UDP 192.168.1.181:45735 <-> 35.219.252.146:3478
  1) UDP [2001:b07:XXXX:XXXX:6f8a:a4d7:XXXX]:38057 <-> [2600:1901:8110:215:0:27::]:3478
  2) UDP 192.168.1.181:45735 <-> 35.216.234.234:3478
  3) TCP 192.168.1.181:39723 <-> 35.219.252.146:80
  4) UDP [2001:b07:XXXX:XXXX:6f8a:a4d7:XXXX]:38057
  5) TCP [2001:b07:XXXX:XXXX:6f8a:a4d7:XXXX]:47269
  6) TCP 192.168.1.181:43077 <-> 35.216.234.234:80
  7) TCP [2001:b07:XXXX:XXXX:6f8a:a4d7:XXXX]:49149 <-> [2600:1900:4160:5999:0:36::]:80
  8) UDP 192.168.1.181:45735 <-> 35.219.226.11:3478
  9) UDP 192.168.1.181:45735 <-> XXX.163.65.69:11545
  10) UDP 192.168.1.181:45735 <-> 35.216.234.234:14309
  11) UDP 192.168.1.181:45735 -> 35.216.234.234:42596
  12) UDP 192.168.1.181:45735 -> 10.46.25.13:45770
[...]
```

Specific call information/metadata

List of all flows (IPv4,6 UDP/TCP) belonging to the same call

```
Call [269 - SignalVoip]:
 Served: 192.168.1.181:37249
 Duration: 13 [s]
 Multiparty: 2 [Peer2Peer]
 Direction: 1 [Originated (served calling)]
 T init:   10/Dec/2024 17:22:52
 T start:  10/Dec/2024 17:23:09
 T end:    10/Dec/2024 17:23:22
 T deinit: 10/Dec/2024 17:23:22
 Endpoints:
 Flows:
  0) UDP 192.168.1.181:37249 <-> 35.219.226.11:3478
  1) UDP [2001:b07:XXXX:XXXX:XXXX:6f8a:a4d7:XXXX]:45308 <-> [2600:1901:8110:215:0:28::]:3478
  2) UDP 192.168.1.181:37249 <-> 35.216.234.234:3478
  3) UDP [2001:b07:XXXX:XXXX:6f8a:a4d7:XXXX]:45308 <-> [2600:1900:4160:5999:0:35::]:3478
  4) UDP 192.168.1.181:37249 <-> 35.219.252.146:3478
  5) TCP 192.168.1.181:46821 <-> 35.219.226.11:80
  6) TCP [2001:b07:XXXX:XXXX:6f8a:a4d7:XXXX]:42435 <-> [2600:1901:8110:215:0:28::]:80
  7) TCP 192.168.1.181:40359 <-> 35.216.234.234:80
  8) TCP [2001:b07:XXXX:XXXX:6f8a:a4d7:XXXX]:42849 <-> [2600:1900:4160:5999:0:35::]:80
  9) UDP 35.219.226.11:63360 <-> 192.168.1.181:37249
  10) UDP 35.219.226.11:9401 -> 192.168.1.181:37249
```

Application

The call has been answered

Standard call; no group call

Direction

Multimedia Calls visibility: example 2

```
./example/ndpiReader -t -A -d -i ~/wa_mia3_android11_app_2.25.11.75_multiparty_videocall_answered_from_lte.pcapng
[...]
WhatsAppCall         packets: 3230         bytes: 2309299       flows: 7
[...]
Num async events (calls): 1
Call [45 - WhatsAppCall]:
 Served: 192.168.12.67:41805
 Duration: 20 [s]
 Multiparty: 1 [Multiparty]
 Direction: 0 [Unknown]
 Multimedia: Audio, Video
 T init:   23/Apr/2025 13:53:19
 T start:  23/Apr/2025 13:53:40
 T end:    23/Apr/2025 13:54:00
 T deinit: 23/Apr/2025 13:54:00
[...]
 Streams:
  SSRC 0x87a77970 [To served], Pkts 825
   PT 97, Pkts 824 [First: seq 1 timestamp 1994310 [23/Apr/2025 13:53:40]][Last: seq 824 timestamp 3792420 [23/Apr/2025 13:54:00]]
   PT 105, Pkts 1 [First: seq 825 timestamp 3798990 [23/Apr/2025 13:54:00]][Last: seq 0 timestamp 0 [01/Jan/1970 00:00:00]]
   1) UDP 157.240.203.62:3478 -> 192.168.12.67:41805 [Dir: To served]
   2) UDP 157.240.17.62:3478 -> 192.168.12.67:41805 [Dir: To served]
  SSRC 0xfabac6c6 [To served], Pkts 310
   PT 120, Pkts 310 [First: seq 1 timestamp 355200 [23/Apr/2025 13:53:40]][Last: seq 277 timestamp 682560 [23/Apr/2025 13:54:00]]
   2) UDP 157.240.17.62:3478 -> 192.168.12.67:41805 [Dir: To served]
   1) UDP 157.240.203.62:3478 -> 192.168.12.67:41805 [Dir: To served]
  SSRC 0x7f92142 [From served], Pkts 790
   PT 97, Pkts 790 [First: seq 1 timestamp 1940490 [23/Apr/2025 13:53:40]][Last: seq 790 timestamp 3665790 [23/Apr/2025 13:53:59]]
   3) UDP 192.168.12.67:41805 -> 193.135.148.227:3478 [Dir: From served]
   2) UDP 192.168.12.67:41805 -> 157.240.17.62:3478 [Dir: From served]
   1) UDP 192.168.12.67:41805 -> 157.240.203.62:3478 [Dir: From served]
[...]
```

Group call

Sometime some meladata is not available

Videocall

The same RTP stream might use different flows over time!

List of all RTP streams belonging to the same call

MC visibility: example 3 (behind NAT)

```
./example/ndpiReader -t -A -d  -i ~/wa_2_simultenously_calls_behind_nat.pcapng
[...]
WhatsAppCall        packets: 12958        bytes: 3355573      flows: 19
[...]
Num async events (calls): 2
Call [45 - WhatsAppCall]:
 Served: 192.168.1.183:45463
 Duration: 11 [s]
 Multiparty: 2 [Peer2Peer]
 Direction: 0 [Unknown]
 Multimedia: Audio, Video
 T init:   27/Apr/2025 10:57:04
 T start:  27/Apr/2025 10:57:09
 T end:    27/Apr/2025 10:57:20
 T deinit: 27/Apr/2025 10:57:20
 Flows:
  0) UDP 192.168.1.183:45463 <-> 157.240.203.62:3478
  1) UDP 192.168.1.183:45463 <-> 157.240.231.62:3478
  2) UDP 192.168.1.183:45463 <-> 157.240.17.62:3478
  3) UDP 192.168.1.183:45463 <-> 157.240.196.62:3478
  4) UDP 192.168.1.183:45463 -> 10.223.178.109:60790
  5) UDP XXX.40.53.67:63569 -> 192.168.1.183:45463
  6) UDP 192.168.1.183:56916 <-> XXX.40.53.67:63569
[...]
```

:(

2 independent but overlapping calls made by 2 different devices behind a NAT (i.e. from nDPI POV they share the same ip)

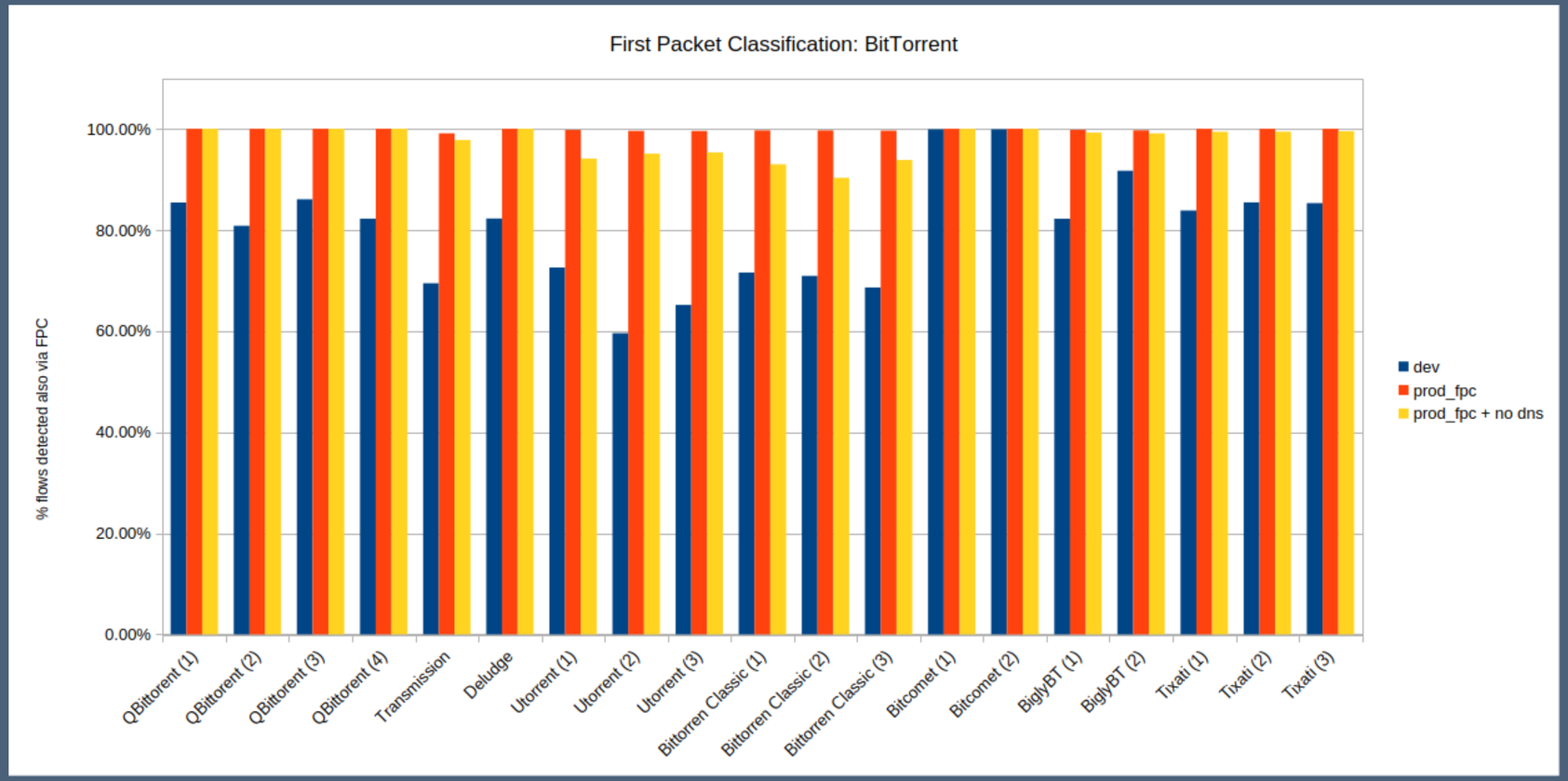
```
Call [45 - WhatsAppCall]:
 Served: 192.168.1.183:44111
 Duration: 422 [s]
 Multiparty: 2 [Peer2Peer]
 Direction: 1 [Originated (served calling)]
 Multimedia: Audio
 T init:   27/Apr/2025 10:55:51
 T start:  27/Apr/2025 10:56:03
 T end:    27/Apr/2025 11:03:05
 T deinit: 27/Apr/2025 11:03:05
 Flows:
  0) UDP 192.168.1.183:44111 <-> 157.240.0.62:3478
  1) UDP 192.168.1.183:44111 <-> 157.240.17.62:3478
  2) UDP 192.168.1.183:44111 <-> 31.13.86.48:3478
  3) UDP 192.168.1.183:44111 <-> 157.240.252.62:3478
  4) UDP 192.168.1.183:44111 -> 10.45.249.173:42515
  5) UDP 192.168.1.183:44111 <-> XXX.211.76.47:3201
  6) UDP 192.168.1.183:43942 <-> 157.240.0.62:3478
  7) UDP 192.168.1.183:43942 <-> 157.240.17.62:3478
  8) UDP 192.168.1.183:43942 <-> 31.13.86.48:3478
  9) UDP 192.168.1.183:43942 <-> 157.240.252.62:3478
  10) UDP 192.168.1.183:43942 -> 10.45.249.173:42515
  11) UDP 192.168.1.183:43942 <-> XXX.211.76.47:3201
[...]
```

That's all folks!

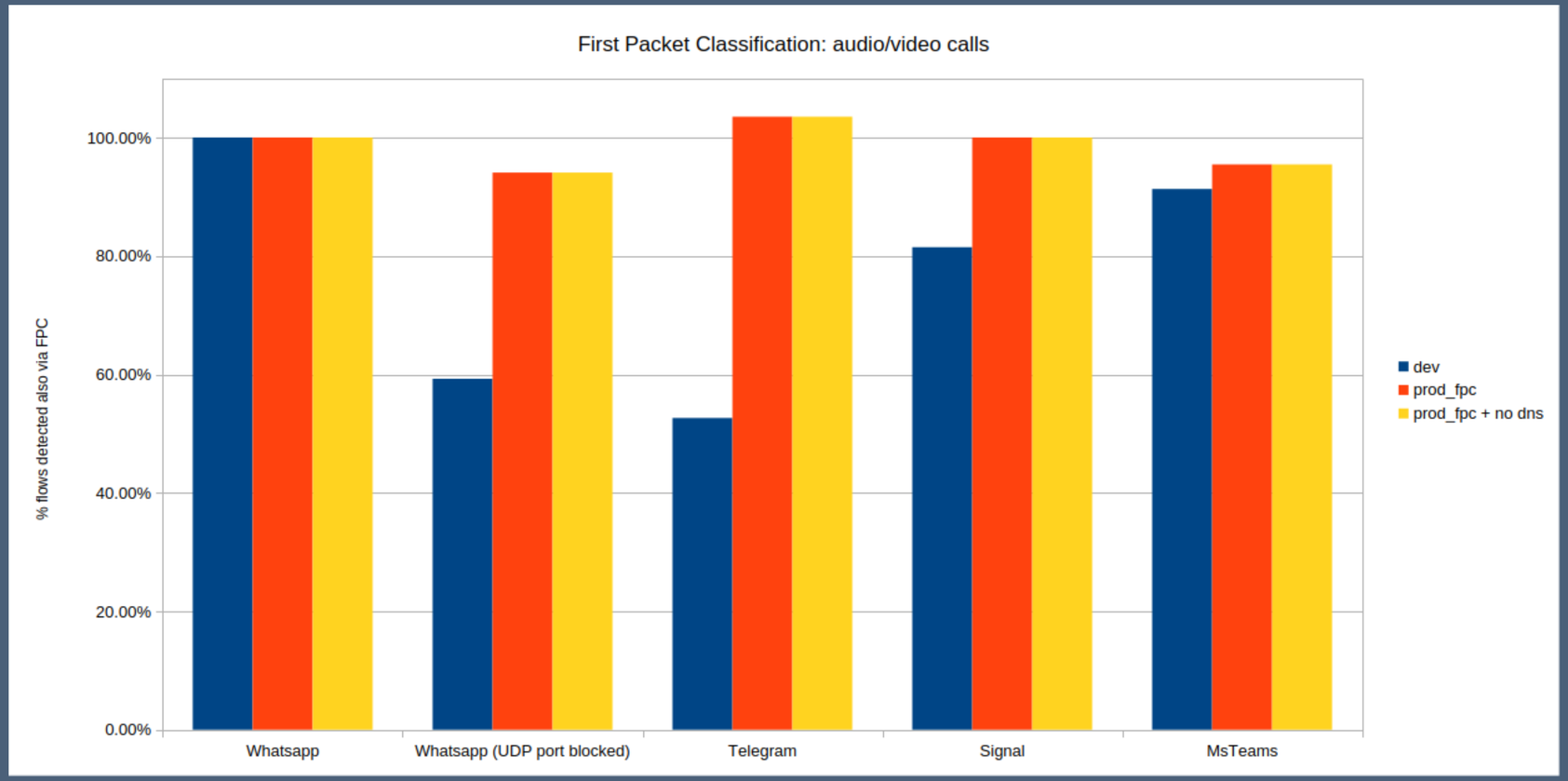
- You can use nDPI to solve practical problems, even if you don't understand/care about its internal details
- Thanks for your time!

- Further readings:
  - https://www.ntop.org/ntop/a-deep-dive-into-traffic-fingerprints/
  - https://fosdem.org/2025/schedule/event/fosdem-2025-5461-passive-network-traffic-fingerprinting/

FPC: improvements (BitTorrent)



FPC: improvements (STUN)



FPC: improvements (VPN)