ntop Users Group Meeting

nDPI: Open-Source High-Speed Deep Packet Inspection

Giuseppe Augiero <augiero@ntop.org>

SharkFest Europe

ntop meeting

Traffic Classification: an Overview

- In the network management it is very important understand what happens on the net.
- Uses, trends, problems, abuses and so on…

- Traffic classification is compulsory to understand the traffic flowing on a network and enhance user experience by tuning specific network parameters.

Classic method

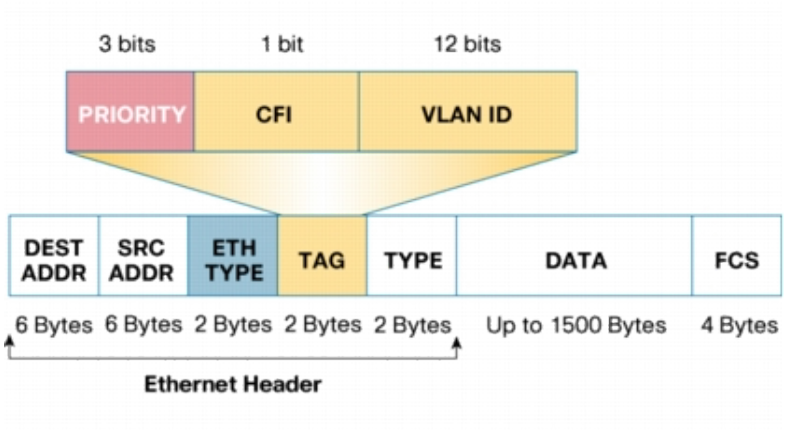
- Main classification methods (classic) include:
  - TCP/UDP port classification.
  - QoS based classification (DSCP).
  - Statistical Classification.
- The results are not complete or may be not correct.

Port-based Traffic Classification

- Port-based Classification
  - In the early day of the Internet, network traffic protocols were identified by protocol and port.
  - Can classify only application protocols operating on well known ports (no rpcbind or portmap).
  - **Easy to cheat and thus unreliable (TCP/80 != HTTP).**

DSCP-based Traffic Classification

- QoS Markers (DSCP)
  - Similar to port classification but based on QoS tags.
  - Usually ignored as it is easy to cheat and forge.



Statistical Traffic Classification

- Classification of IP packets (size, port, flags, IP addresses) and flows (duration, frequency, etc…).
- Based on rules written manually, or automatically using machine learning (ML) algorithms.
- ML requires a training set of very good quality, and it is generally computationally intensive.
- Detection rate can be as good as 95% for cases which were covered by the training set, and poor accuracy for all the other cases.

Deep Packet Inspection (DPI)

- Technique that inspects the packet payload.
- Computationally intensive with respect to simple packet header analysis.
- Concerns about privacy and confidentiality of inspected data.
- Encryption is becoming pervasive, thus challenging DPI techniques.
- No false positives unless statistical methods or IP range/flow analysis are used by DPI tools.

Using DPI in Traffic Monitoring

- Packet header analysis is no longer enough as it is unreliable and thus useless.
- Security and network administrators want to know what are the real protocols flowing on a network, this regardless of the port being used.
- Selective metadata extraction (e.g. HTTP URL or User-Agent) is necessary to perform accurate monitoring and thus this task should be performed by the DPI toolkit without replicating it on monitoring applications.

Why (n)DPI?

- There are many commercial DPI libraries: NDA-based, expensive (both in price and maintenance), closed source (you need to trust manufactures), non-extensible by end-users (vendor lock-in).
- Alternatives: Linux layer-7 filter (obsolete), Libprotoident (good but limited to 4 bytes analysis thus not extracting any metadata).

- In essence we need a opensource (n)DPI system.

Welcome to nDPI

- We decided to develop our own GNU GPL DPI toolkit (based on a unmaintained project named OpenDPI) in order to build an open DPI layer for ntop and third-party applications.
- Protocols supported exceed 200 and include:
  - P2P (Skype, BitTorrent)
  - Messaging (Viber, Whatsapp, MSN, The Facebook)
  - Multimedia (YouTube, Last.gm, iTunes)
  - Conferencing (Webex, CitrixOnLine)
  - Streaming (Zattoo, Icecast, Shoutcast, Netflix)
  - Business (VNC, RDP, Citrix, *SQL)

nDPI vs OpenDPI [1/2]

- Code has been changed to be really end-user extensible by coding a new protocol dissector.
- Various code sections have been rewritten to make them reentrant (multithread).
- Major performance improvements, and introduction of “hints” (e.g. for traffic on TCP/80 try the HTTP dissector first).
- Added support for SSL certificate decoding, used for detecting specific communications (e.g. classify encrypted Apple traffic: iTunes vs. FaceTime).

nDPI vs OpenDPI [2/2]

- Introduction of substring-matching for searching specific words on strings. For instance users can configure at runtime rule where for HTTP traffic matching host names *google.com should be considered as Google (protocol) traffic.
- Extraction of metadata such as HTTP URL, DNS queried hostnames to be used by user-space applications.
- Port to non-x86 platforms and embedded platforms.

nDPI Internals

- The library engine is responsible for maintaining flow state (no DPI is performed).
- Based on flow protocol/port all dissector that can potentially match the flow are applied sequentially starting from the one that most likely match.
- Each dissector is coded into a different .c file for the sake of modularity and extensibility.
- There is an extra .c file for IP matching (e.g. identify spotify traffic based on Spotify AS).

Traffic Classification Lifecycle

- nDPI divides the traffic in 5-tuple flows.
- Based on traffic type (e.g. UDP traffic) dissectors are applied sequentially starting with the one that will most likely match the flow.
- Each flow maintains the state for non-matching dissectors in order to skip them in future iterations.
- Analysis lasts until a match is found or after too many attempts (8 packets is the upper-bound in our experience).

Evaluating nDPI

- nDPI has been evaluated both in terms of accuracy and performance.
- "*The best accuracy we obtained from nDPI (91 points), PACE (82 points), UPC MLA (79 points), and Libprotoident (78 points)*"*
- Issues on nDPI are mostly due to dissectors that conservative and thus prefer report a flow as unknown rather than misclassify it.

* T. Bujlow, V. Carela-Español, P. Barlet-Ros, Comparison of Deep Packet Inspection (DPI) Tools for Traffic Classification, Technical Report, June 2013.

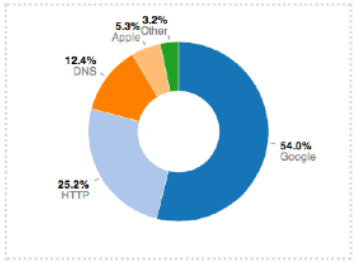
nDPI Performance

```
# taskset -c 1 ./pcapReader -i ~/test.pcap
Using nDPI (r7253)
pcap file contains
IP packets: 3000543 of 3295278 packets
IP bytes:1043493248(avg pkt size 316 bytes)
Unique flows: 500
nDPI throughput: 3.42 M pps / 8.85 Gb/sec
```

- With two cores it is possible to analyse a full 10 Gbit link on a Intel i7-860 both using traffic traces or capturing live on top of PF_RING (home-grown packet processing framework).

nDPI In Real Life

- nDPI is used by several projects on the Internet including:
  - Network Forensics (Xplico).
  - Linux-kernel packet filtering (ndpi-netfilter).
  - Ntopng.



Final Remarks

- We have presented nDPI an open source DPI toolkit able to detect many popular Internet protocols and scale at 10 Gbit on commodity hardware platforms.
- Its open design make it suitable for using it both in open-source and security applications where code inspection is compulsory.
- Code Availability (GNU LGPLv3)
  https://github.com/ntop/nDPI

ntop Users Group Meeting

nDPI: Open-Source High-Speed Deep Packet Inspection

Giuseppe Augiero <augiero@ntop.org>

SharkFest Europe

ntop meeting