

Information and Communication Technologies (ICT) Programme

Project No: FP7-ICT- 248036

## COAST



---

### D4.1: DPI Specification and High Level Complexity Evaluation of Algorithms

---

**Author(s):** S. Niccolini (NEC), M. Ahmed (NEC), I. Kelly (Yahoo!), M. Perdikeas (Synelixis), I. Koufoudakis (Synelixis), A. Platskos (Synelixis), S. Zezza (POLITO), G. Masera (POLITO)

**Status -Version:** Final

**Delivery Date (DOW):** 31 December 2010

**Actual Delivery Date:** 4 February 2011

**Distribution - Confidentiality:** Public

**Code:** COAST\_D4\_1\_POLITO\_FF\_20110204

#### Abstract:

In the COAST project, fast and efficient content-aware retrieval, delivery and streaming is going to be achieved by means of innovative techniques for network caching, searching and context awareness. A key element that is required to enable this kind of content aware functionalities is the capability of analyzing the content flowing through network routers and this can be done resorting to Deep Packet Inspection (DPI) technologies.

This deliverable analyses and specifies the methods and algorithms that will be utilized in COAST for in the network content detection & inspection. It considers various DPI algorithms including "stateful" and "stateless" techniques, and provides the specifications on DPI architecture and packet/traffic analysis.



## Disclaimer

This document contains material, which is the copyright of certain COAST contractors, and may not be reproduced or copied without permission. All COAST consortium partners have agreed to the full publication of this document. The commercial use of any information contained in this document may require a license from the proprietor of that information.

The COAST Consortium consists of the following companies:

No	Participant name	Participant short name	Country	Country
1	ST Microelectronics	STM	Co-ordinator	Italy
2	Synelaxis Solutions Ltd	Synelaxis	Contractor	Greece
3	Yahoo Iberia SL	Yahoo	Contractor	Spain
4	NEC Europe Ltd	NEC	Contractor	UK
5	TelefonicaInvestigacion Y Desarrollo SA	TID	Contractor	Spain
6	Fraunhofer HHI	HHI	Contractor	Germany
7	Politecnico di Torino	Polito	Contractor	Italy
8	Technische Universitaet Berlin	TUB	Contractor	Germany
9	Fundacio Barcelona Media	BM	Contractor	Spain
10	University of California, Los Angeles	UCLA	Contractor	USA
11	Seoul National University	SNU	Contractor	S. Korea

The information in this document is provided “as is” and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.



*This page has been intentionally left blank*



## Table of contents

<b>Acronyms .....</b>	<b>5</b>
<b>Executive Summary .....</b>	<b>6</b>
<b>1 Introduction .....</b>	<b>7</b>
1.1 <i>Passive Crawling</i> .....	8
1.2 <i>Content Popularity</i> .....	8
1.3 <i>Content discovery</i> .....	9
1.4 <i>Network topology discovery</i> .....	9
<b>2 Network Architecture and specifications of the DPI process .....</b>	<b>10</b>
2.1 <i>High-level DPI architecture and components</i> .....	10
2.1.1 DPI network architecture .....	11
2.1.2 Mapping to COAST FCCN architecture .....	12
2.1.3 Communication Requirements.....	13
2.1.4 DPI Information Maintenance .....	13
2.2 <i>System Requirements</i> .....	14
2.2.1 Traffic statistics .....	14
2.2.2 Protocol distribution.....	16
<b>3 Review of state of the art DPI algorithms.....</b>	<b>21</b>
3.1 <i>Stateless algorithm</i> .....	21
3.2 <i>Stateful algorithm</i> .....	26
3.2.1 Packet classification.....	27
3.2.2 Flow state management .....	34
<b>4 High-level Design of the DPI node.....</b>	<b>37</b>
4.1 <i>The DPI node architecture</i> .....	37
4.2 <i>PF RING and its recent extensions</i> .....	39
4.3 <i>Popularity plug-in</i> .....	41
4.3.1 TCP protocol .....	41
4.3.2 HTTP protocol .....	42
4.3.3 RTSP protocol.....	43
4.4 <i>Signalling and Named COAST Object Identifiers (COI)</i> .....	44
<b>5 Conclusions .....</b>	<b>48</b>
<b>6 References .....</b>	<b>49</b>
<b>7 Appendix A – EU Legal framework for data protection, privacy and the COAST project.....</b>	<b>54</b>
7.1 <i>National legislations</i> .....	56
7.1.1 The specific case of the UK .....	57
7.1.2 The specific case of Germany .....	58
7.1.3 The specific case of the USA .....	58
7.1.4 The specific case of the Republic of Korea .....	59
7.2 <i>Information and Communication Technologies under the aspect of European Data Protection and Privacy Framework</i> .....	59
7.3 <i>Privacy issues related to the use of DPI in COAST</i> .....	63
7.4 <i>Conclusion</i> .....	65
<b>8 Appendix B – Traffic statistics .....</b>	<b>66</b>



## Acronyms

3DV	3D Video
ALTO	Application Level Traffic Optimizer
BGP	Border Gateway Protocol
CCI	COAST Content Identifier
CCL	Content Cache Locator
CCO	Content Cache Optimizer
CEP	Content overlay Entry Point
CHR	Content Hybrid Routing
CMI	COAST Metadata Identifier
CN	Content-aware Node
COI	COAST Object Identifier
CURL	COAST-aware URL
DHT	Distributed Hash Table
DSHT	Distributed Sloppy Hash Table
DNS	Domain Name Service
DPI	Deep Packet Inspection
FCCN	Future Content-Centric Network
ISP	Internet Service Provider
MPD	Media Presentation Description
MVD	Multi-view Video Plus Depth
PQoS	Perceived Quality of Service
P2P	Peer to Peer
RG	Residential Gateway
RTSP	Real Time Streaming Protocol
RTP	Real Time Protocol
SLA	Service Level Agreement
UA	User Agent
URL	Unified Resource Locator



## Executive Summary

The COAST project aims to realise fast and efficient content-aware retrieval, delivery and streaming through the development of innovative techniques for network caching, search and context awareness. Key to releasing our goal of systems with well developed content aware functionality is the capacity to capture, and analyse content as it traverses the internet - in a transparent manner.

Deep Packet Inspection (DPI) refers to the process and methodology of filtering packets on the network - typically operating at Layer 2 through to Layer 7 of the OSI model; as implied by the name, the packet header as well as its data is inspected. In practice, most DPI applications are specialised to combat protocol non-compliance, viruses, spam, or intrusions. DPI has also been proposed as a technique to drive packet routing; based on statistical information about the packets' content. Since DPI can be used to identify and classify traffic based on a signature database that includes information extracted from the data part of a packet, it allows finer control than classification based only on header information.

Within the COAST project, we plan to use DPI for difference purpose, namely for the discovery and identification of interesting URLs and COAST objects. The information retrieved by the DPI may be employed in a number of contexts, for example, DPI collected information may be supplied to search engines in order to support the development of more proactive content rating and ranking algorithms.

It is planned to use DPI for different purposes, that are essentially related to the discovery of those URLs and COAST objects that are not easily found by existing search engines and to the classification of URLs or COAST objects based on their popularity. In order to achieve these objectives, inspection of packet content is required; moreover efficient data structures must be conceived to collect and maintain huge amounts of data. This kind of information is periodically sent to search engines and exploited to enhance search activities.

In this document, we present our study of the requirements of a DPI in the context discussed, we first introduce passive crawling as a method to collect information on accessed contents in the network and define the main objectives of this activity: discovery of new content and analysis of content popularity. We then propose a network architecture that enables low effort inclusion of DPI functionalities in the network.

Several DPI algorithms are reviewed and discussed in terms of complexity, achievable speed and memory requirements. Comparisons among these methods are used to drive choices in the subsequent COAST activities.

Finally high level organization and implementation of the DPI node are discussed, starting from an already available platform for wire speed packet capture and monitoring.



# 1 Introduction

The COAST project aims to build a Future Content-Centric Network (FCCN) capable of intelligently and efficiently linking billions of content sources to billions of content consumers. This includes offering fast content-aware retrieval, delivery and streaming, while meeting network-wide Service Level Agreements (SLAs) in content and services consumption. These goals will be achieved by integrating intelligent network caching, searching and network, terminal and user context awareness.

The requirements of COAST functionality identified so far are as follows:

- a) Supporting search over a massive numbers of documents and services using a scalable distributed approaches
- b) Identifying where the “best” content resides
- c) Identifying/analysing what content and traffic is flowing through the network routers
- d) Replicating and caching the content efficiently at the “best” place in the network
- e) Dynamically identifying the “best” host/cache and the end-to-end path (in terms of both efficiency and network-friendliness) for content delivery and streaming to a user
- f) Providing the “best” Perceived Quality of Service (PQoS) to the user by interactively adapting the content based on the user and the terminal capabilities, requirements and context.

The goal of this document is to provide a high-level specification of how to identify/analyze the content and traffic that flows through network routers using Deep Packet Inspection (DPI) technologies. As such, we identify the following set of functional requirements for a DPI:

- To discover URLs (or COAST objects) that are hard to crawl by search engines while these URLs are transferred over the network
- To identify COAST objects or streams that are transferred over the network
- To inform search engines about the popularity (access counts) of URLs seen in the network.

Such functionalities are specified in the COAST [1], as core functionalities CF1.1 “Content discovery”, CF1.2 “Service discovery” and CF1.5 “Content and services popularity discovery”:

- **CF1.1: Content discovery.** Includes the discovery of named content at different locations within the network and its reporting to search engines.
- **CF1.2: Service discovery.** Includes the discovery of named web services (WSDL) at different locations within the network.
- **CF1.5: Content and services popularity discovery.** Includes the identification and evaluation of the popularity of specific content and services through the combination of traditional popularity discovery methods at the search engine site (by counting the clicks at each hyperlink) and innovative DPI based methods at content-aware nodes in the network.

With the implementation of the functionality described so far, COAST can provide significant information to the search engine for its indexing in a timely manner. A major advantage of this “passive crawling” is the reduction of the time between having new content available and having this content indexed.



## 1.1 Passive Crawling

The wide majority of search engines discover content following a technique called **active crawling**. When performing active crawling, the search engine robots follows all known URLs and references, trying to find and then index new content and services. This function is by definition post-hoc and may often take hours or even days to identify new content. Moreover DPIs may be used to tackled the problems included by the phenomenon referred to the “**hidden web**”; whereby certain portions of the web are hard to crawl because of their lack of linkage to popular web sites, or because they contain dynamically generated content.

We hypothesise by utilizing Deep Packet Inspection (DPI) on the content that flows via a content aware node, the node is able to initially discover new URLs and index new content (**passive crawling**) much earlier than **active crawling**. As a result, DPI may significantly reduce the discovery time and provide search engines with a major competitive advantage, as it makes possible to find and index new popular content. Of course, regular active crawling at scheduled time intervals has still to be performed as a follow-up in order to ensure that the content remains published and to reduce the number of broken or unavailable links returned by the search engine, thereby increasing its effectiveness and reliability.

Apart from content, the same advantages apply also for web services. There are already a number of service search engines (e.g. seekda) that perform active crawling of services (mainly based on WSDL headers). DPI inspection may significantly help also Search as a Service functionality.

## 1.2 Content Popularity

Conventional search engines estimate the importance (or relevance) of a particular matching web page typically based on two broad aspects: the content of the web page, and the hypertext (or citation) structure of the surrounding web.

- First, a conventional search engine analyzes the contents of a particular web page and examines criteria such as the frequency of occurrence of the search terms, the location of the search terms (e.g. the title is more important than the appendix), the font size of the search terms relative to the font size of the surrounding text, the document format (e.g. certain file formats such as word processing files are usually more important than other file formats such as simple web pages), the web location of the document (e.g. documents on major web portals are more important than those on an individual’s web page). Each of these factors plays a role in determining the importance of a web page.
- Second a conventional search engine exploits the hypertext link structure of the World Wide Web by viewing it as a citation index. Pages that are referred to (or linked to) by more pages are likely to be more important than pages that are linked to by fewer pages.

However, conventional search engines are not capable of monitoring how many times particular web pages URLs where actually visited (i.e., the popularity of a web pages) for use in determining the importance of those web pages, although the actual number of visits to a web page would strongly indicate the importance of the web page. Conventional search engine merely estimate the importance of a particular matching web page based upon the content of the page and hypertext (or citation structure) of the surrounding web. The conventional search engines do not take into consideration the frequency of visits to the web page in estimating the importance of the web page. Furthermore, when propagating scores along the hypertext structure of the web, the score of a page is typically divided equally amongst the destination pages, rather than taking into consideration the relative popularity of the outgoing links from the page. Therefore there is a need for a method and system for monitoring and analyzing the current popularity of pages on a network



and also there is a need for monitoring and analyzing the popularity of links between pages in hyperlink network. This deliverables provides algorithms and methods for monitoring, using the DPI, the popularity of web pages (HTTP) and using such popularity information to rank the web pages retrieved in response to a search.

### ***1.3 Content discovery***

In addition to the two above mentioned functionalities COAST partners will also investigate the possibility of using the DPI for capturing the different chunks of the content and reconstructing/caching it in the local node in order to make it available to the COAST overlay network (e.g., as transparent cached content).

This last functionality can also be described as “publishing” albeit in a more demanding sense; content is not just discovered and made public, it is also “reassembled on-the-fly” and used to populate the COAST overlay network and the COAST caches. The main challenges in this kind of “publishing” are privacy, availability (in terms of when the content is to be considered expired) and of course performance - since DPI will need to incorporate storage and capabilities as well in order to recreate the entire media object over a succession of multiple packets.

### ***1.4 Network topology discovery***

Last but not least, DPI may further help in understanding the network topology by supporting the application of passive network tomography. However, without control plane traffic such as BGP traffic, it is not possible to directly observe topology information. DPI boxes could on the other hand perform a hash on a packet (including its contents) to create a unique identifier for that packet. Such an identifier can then be sent to the COAST Network Monitoring server at the Information Overlay in order to consistently trace the path that the packet is taking through the network as it travels from source to destination. Initially, when only a few DPI boxes are deployed, only parts of the network will be visible: packets flowing through paths with no DPI boxes are invisible to the system. As more DPI boxes are added and more packets are analyzed in this way, the topology of the network will become clearer. In the following the methodology followed by COAST DPI functionality in order to meet the above objectives are analyzed.



## 2 Network Architecture and specifications of the DPI process

Several works has already been done on DPI and several DPI algorithms are available in the literature for application in different domains and with different purposes. Implementation solutions can also be found for DPI algorithms both in software and hardware forms. Key known approaches will be reviewed and compared in Section 3. However, in order to evaluate the potential advantages offered by each analyzed algorithm, the basic requirements of DPI in the specific context of the COAST project have to be considered. In particular, two fundamental requirements must be addressed by a DPI engine in order to provide useful function at network level: the performance and the scalability.

On one hand, **performance** is very important as data rates at the Internet nodes are ever increasing and envisioned DPI functions must be able to support realistic traffic conditions. Pure software based approaches are preferable due to their higher flexibility and lower development time and cost. However most of DPI applications are today made possible by means of dedicated hardware accelerators, with sufficient parallelism to cope with typical wire speed. The actual performance needs for DPI applications in the COAST project must be carefully estimated and used to drive choices of algorithms, implementation platforms and even network architecture.

The second important DPI requirement is the **scalability**. Most of DPI techniques make use of huge amounts of memory, that strongly depend on the structure of the network, on the location of the DPI function and on the supported data rates. Although several methods are known to compress required data structures, so increasing the processing speed and decreasing the amount of memory to be managed, this issue is still critical. The selected DPI solution must be able to adapt to different network topologies and to different flows of data. The envisioned solution is based on a hierarchical structure of DPI engines that aggregate DPI results, before providing them to the Search Engines.

Besides the mentioned performance and scalability requirements, one should not overlook the **privacy** issues and implications that DPI may have. The legal framework and COAST approach is further analyzed in Annex A.

Addressing these requirements directly influences the high level specification of the DPI functionalities and the design of the DPI node presented in the following sections.

### 2.1 High-level DPI architecture and components

The primary aim of the DPI process is to intercept the data traversing through the network and identify it subject to some particular purpose. In our case, we are interested in processing data for content popularity analysis and in particular at the highest level, we would like to be able answer semantically rich questions such as:

- I. Which named contents/services are new?
- II. What is the popularity counter for a specific named content?
- III. Has the named content changed since the last time it was seen?
- IV. Which are the new nodes that have been identified in the network?

Clearly the task of answering such questions, while maintaining the requirements of high performance and scalability is challenging. Given that we are interested in being able to capture packets in a distributed manner across within and across ISPs, the system that we develop will need

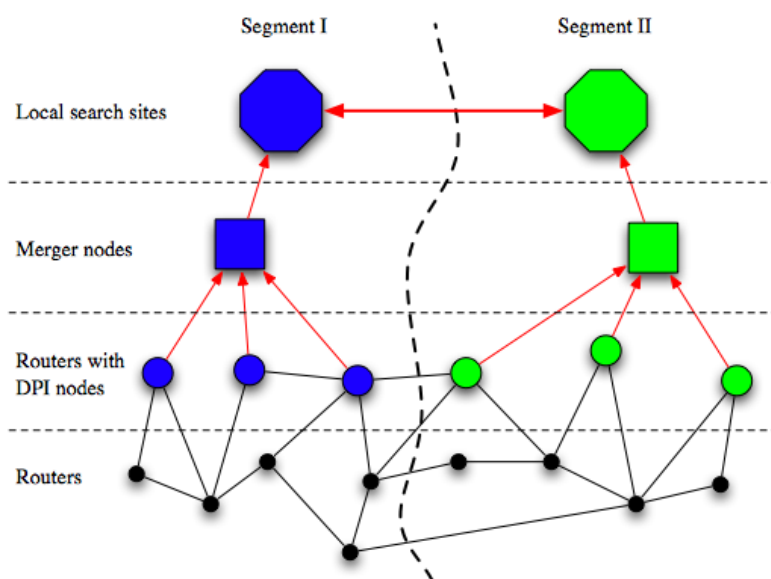


to be scalable. In the context of COAST, we define scalability as the capacity for the system to grow and shrink gracefully with the traffic volume and application demands.

### 2.1.1 DPI network architecture

To address this requirement, we first look at how COAST nodes are placed in the network. The choice of a placement for a COAST node may be made subject to a numerous constraints, such as the latency, inter-ISP policies and so on. As such, we define the logical construct of a network segment to refer to a network partition under the control of one *local search site*. Therefore a segment may refer to a geographical region or a logical region defined an under an arbitrary constraint, i.e. a subnet or a latency requirement. The definition of a network segment also means that we can group the set of nodes involved in generating data for a local search site as shown in Figure 1.

Further, to support scalability, we simplify the COAST DPI architecture and define three types of actors involved in a COAST based systems; these are: *DPI nodes*, *Merger nodes* and the *Local search site*. Together these actors cooperate to realize a COAST based system, such that for example; ISPs place *DPI nodes* on different segments of the network, whereby each segment contains a *Local search site*, a regional *URL Merger*, and a number of regional *DPI nodes*. The communication across the segments is only through the *Local search sites*. Finally, within a segment, the *DPI nodes* communicate only with the regional *Merger*, which communicates only with the *Local search site* of the segment.



**Figure 1: Communication between DPI segments**

In more detail, the roles of the COAST actors are as follows:

- DPI node:** Each segment is associated a set of *DPI nodes* and the number in the set is determined based on the traffic volume to be inspected and the capacity of each *DPI node* in the set. A given *DPI node* is associated with a router and has a static range of IPs assigned to it for monitoring. To prevent duplication of work, IP ranges are assumed to be non-overlapping between any two *DPI nodes*. While the basic duty of a *DPI node* is to inspect the packets passing through its router.

Given a packet for processing, DPI process extracts a source URL and a destination IP listed in the packet. If the destination IP is within the IP range of the DPI node, the URL is stored in a hash table by the node for further processing, otherwise, the URL is ignored. URL hash



tables are defined to live for some epoch of time, whereby during the epoch, the counters of the entries in the hash table are updated based on their occurrence. At the end of the epoch, the hash table is locked, compressed and transferred to the *Merger node*. The updates received in the mean time are performed on a new hash table, which eventually replaces the first table. For complex processing, *DPI nodes* are assumed to provide the means to monitor protocol state, but not application state. For example, though the DPI process may be able to identify whether a given packet is miss-formed according to the protocol, it will be able to maintain enough state to notify the user that the content of a URL has changed.

- **Merger node:** *Merger nodes* provide two functionalities; first they are used to mediate the interaction with *DPI nodes* as act to funnel the traffic to the *Local search site*. Second, *Merger nodes* apply data filtering and aggregation to the inputs from the *DPI nodes*, for merging eliminating duplicate entries. *Merger nodes* maintain a hash table, similar in structure to those in the *DPI nodes* for the aggregation of URL occurrence counts is performed over this hash table. The hash table is periodically compressed and transferred to the *Local search engine* that shares the same segment with the *Merger node*. The *Merger node* is the also the best candidate node for carrying out a stateful analysis of contents (e.g., detecting if the contents monitored are new or not).
- **Local search site:** Every *Local search site* accepts only the user queries coming from a fixed IP range – corresponding with the IP range of the *DPI nodes* that are assigned to its local segment. Once a local engine receives an update from a *Merger node*, it decompresses the incoming data and iterates over the URLs found. If the *Local search engine* did not previously store the content of the URL, the URL is crawled and its content is stored locally. If the content was previously stored on non-local search sites, their copies are updated with the newly downloaded content for consistency in content replication. If the content was already stored in the *Local search engine*, then simply its occurrence count is increased. For replicated documents, this information should be propagated to non-local search sites that store a copy of the document and popularity updates are also performed remotely. These updates can be done in batch mode as well.

### 2.1.2 Mapping to COAST FCCN architecture

This architecture may be further mapped to the COAST Future Content Centric Network (FCCN) architecture as described in Figure 3 of [34]. As it is shown here in Figure 2, we highlight the nodes that may offer DPI functionality. These are the Content overlay Entry Points (CEP) at the end-user edge nodes and the Content aware Nodes in the Content/Service Distributed Overlay. In Figure 2 three types of Content aware Nodes are shown: nodes that offer only Network Caches (NC), nodes that offer Deep Packet Inspection (DPI) and nodes that offer both NC and DPI.

The CEP nodes will have very limited (if any) DPI functionality, as they are intended to be very low cost nodes (e.g. embedded in a Residential Gateway or a low cost home-to-access network edge router). Yet, they may offer valuable information as information flows through them; moreover, the network load at that level is comparably small and the traffic speed in the level of Mbps.

At the Service/network Infrastructure overlay, no DPI is expected as this is the legacy infrastructure and the wide majority of today's routers do not offer DPI functionality. These nodes may directly communicate with COAST Network Monitor server for exchanging topology information.

The main DPI functionality will be located at the Content aware nodes in the Content/Service Distributed Overlay. Also at this level it is expected that only a percentage of the nodes will have full DPI functionality. Yet, based on the network topology some of them will operate as simple DPI nodes and some as DPI network merger nodes.

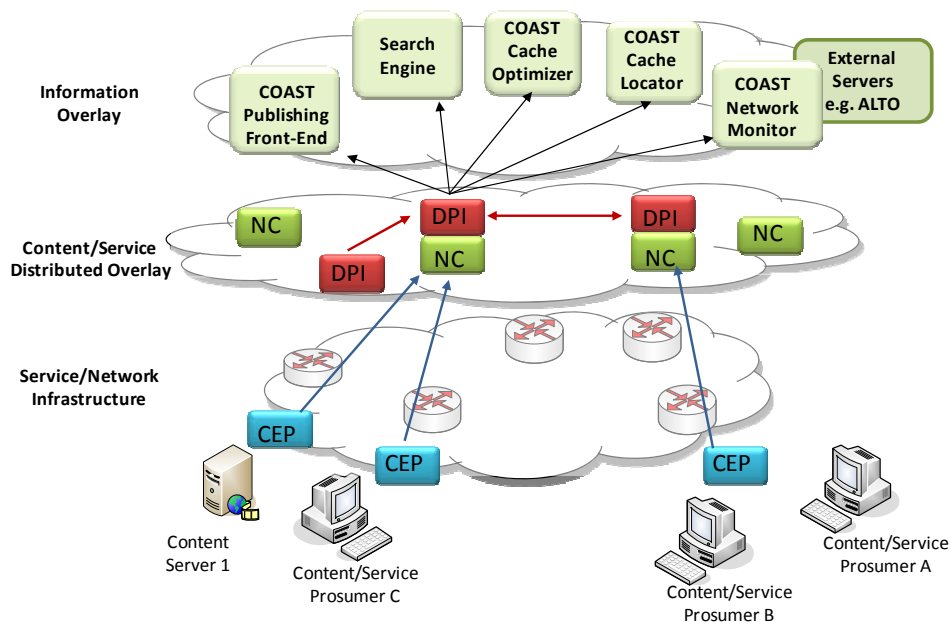


Figure 2: COAST Network architecture highlighting the DPI functionality

### 2.1.3 Communication Requirements

In general, we foresee just a few DPI merger nodes per region in order to lower the communication overhead with the Search Engines. Yet, standardized communication interfaces are required both between the COAST DPI nodes (either CEP or CNs) and between the DPI nodes and the search engines.

### 2.1.4 DPI Information Maintenance

As already detailed the collaborating DPI and Merger nodes should provide the information to help answer questions related to the discovery of new named content and services, such as: the popularity for specific already known named content, if the named content has changed since the last time it was seen and which are the new nodes that have been identified in the network.

To aid in achieving this, the DPI and the Merger nodes together maintain four types of information:

- **Content/services discovery** information simply contains the named content monitored by the DPI node. The named content information should be transferred to the search engine (after aggregation) only if the named content is discovered for the first time. The information is transferred immediately after a discovery (in the case of multiple content discoveries in a short amount of time, near-real-time transfer in batched mode will be also considered to reduce the impact on the network).
- **Popularity information** is similar but it contains a hit counter in addition to the named content itself. The DPI node periodically reports to the search engine the number of times a named content is observed. The count of a named content in the DPI node is increased by one every time the named content is encountered. A sliding window technique can be used to report only the most recent count information.
- **Content summary** is a summarized view of the content used for discovering if the content has changed or not since the last time it was discovered
- **Network Information** containing a list of IP addresses that have been discovered and the network segment that they belonging. This information may be very static in today's network topologies, as we assume that changes in the core and access network is not dynamic; yet in



the near future mobility is expected to significantly increase requesting for dynamic reconfigurability of the network and resources reassignment.

Specifications of the content discovery, popularity information and content summary data structures are reported in Section 4.1.

## 2.2 System Requirements

The role of DPI techniques in the COAST project is to inspect traffic flows with the purpose of extracting information related to visited contents, which have to be categorized based on popularity and relevance criteria. Some key requirements must be met by a DPI node with such purposes:

1. **High speed:** a DPI node should support real-time traffic classification on high speed links, introducing low overhead. Realistic data rates on high speed links are in the range 1 to 10 Gb/s: for example, in OC48c links line rate is 2.5Gb/s while in OC192c links the rate grows to 10Gb/s. In Section 2.2.1 we reported some statistical data about traffic volumes monitored daily, weekly, monthly and yearly on different routers, data centers and Internet Exchange Points located in different geographical area in the World. These data, together with indications on the required monitoring epochs, must be used to select proper DPI algorithms and implementation solutions for both processing and data structures.
2. **High accuracy:** we define accuracy as the percentage of correctly classified flows among the total number of inspected flows. Obviously the system should try to have high accuracy that means low misclassification error rates. To this purpose in Section 2.2.2, we analyze the application level protocol distribution and then summarize the characteristic of the distribution according to the collected results.
3. **Flexibility:** it is better for a DPI node to response in few seconds or minutes without interruption if pattern set change, e.g. a new pattern is added to the DPI node.

### 2.2.1 Traffic statistics

Through sites that make traffic statistics publicly available may not be a representative sample of all those on the Internet, for example their data is usually limited to certain classes of traffic, such as Internet exchanges or academic and research networks, with only a smattering of private companies or commercial service providers. Yet, they provide a sample that could be used for experimentation and prototyping purposes. In this Section we report detailed traffic statistics collected on the CAIDA Web site. Moreover, in Appendix B traffic statistics available at the Minnesota Internet Traffic Study and GARR-G are reported.

Among the passive data sources available to CAIDA [38] in the United States, a number of monitoring locations in several large Internet Service Providers (ISPs) can be found. These taps collect packet headers at large peering points and the resulting traces have been used for a wide variety of research projects, ranging from general attempts to characterize the global state of Internet traffic, specific studies of the prevalence of peer-to-peer file sharing traffic, to testing prototype software designed to stop the spread of Internet worms. There are essentially two data collection and monitoring sites.

The first one is located at an Equinix data center in Chicago, IL, and is connected to an OC192 backbone link (9953 Mbps) of a Tier1 ISP between Chicago, IL and Seattle, WA. Real time traffic for the equinix-chicago-dirA and equinix-chicago-dirB is reported in Table 1 and Table 2. The second one is the equinix-sanjose Internet data collection monitor located at an Equinix data center in San Jose, CA connected to an OC192 backbone link of a Tier1 ISP between San Jose, CA and Los Angeles, CA.



In this case the real time traffic for the two data collectors *equinix-sanjose-dirA* and *equinix-sanjose-dirB* is reported in Table 3 and Table 4 .

Application	1 Day (Max)[bit/s]	1 Week (Max)[bit/s]	1 Month (Max) [bit/s]	2 Years (Max)[bit/s]
HTTP	5,61E+08	4,87E+08	6,63E+08	2,30E+09
UNKNOWN_UDP	7,48E+07	1,02E+08	1,54E+08	3,27E+08
UNKNOWN_TCP	7,19E+07	7,27E+07	8,84E+07	1,21E+08
RTMP	4,18E+07	8,58E+07	2,26E+08	4,13E+07
ABACAST	5,00E+07	3,41E+07	2,05E+07	4,25E+07
HTTPS	1,64E+07	1,60E+07	1,48E+07	2,27E+07
QUAKE	7,05E+06	8,15E+06	8,06E+06	8,85E+07
WOW	3,77E+06	2,93E+06	2,04E+06	3,47E+08
DNS	1,12E+07	2,13E+06	3,07E+06	2,42E+07
FTP_DATA	3,26E+06	3,02E+06	1,29E+07	2,84E+07
SSH	5,44E+06	4,61E+06	1,93E+06	8,96E+06
BITTORRENT	2,62E+06	3,01E+06	4,07E+06	6,26E+06
MS_LIVE	1,93E+06	1,93E+06	2,55E+06	8,03E+06
SMTP	1,10E+06	4,12E+06	1,13E+06	5,05E+06
NOPORTS_UDP	9,46E+06	2,77E+06	2,13E+06	2,22E+07
OTHER	1,33E+07	1,88E+07	1,77E+07	8,30E+07
<b>Total</b>	<b>8,75E+08</b>	<b>8,49E+08</b>	<b>1,22E+09</b>	<b>3,48E+09</b>

Table 1: Network traffic monitored at the equinix-chicago-dirA data center

Application	1 day (Max)[bit/s]	1 week (Max)[bit/s]	1 Month (Max)[bit/s]	2 Years (Max)[bit/s]
HTTP	5,82E+09	6,95E+09	6,92E+09	6,73E+09
UNKNOWN_UDP	6,62E+08	7,22E+08	7,82E+08	1,70E+09
UNKNOWN_TCP	5,30E+08	6,15E+08	8,37E+08	6,29E+08
RTMP	1,73E+08	2,29E+08	4,01E+08	2,29E+08
IPSEC	1,53E+02	9,76E+07	8,64E+07	9,05E+07
HTTPS	6,01E+07	7,47E+07	7,80E+07	2,53E+08
ABACAST	1,10E+08	1,03E+08	8,79E+07	5,59E+07
QUAKE	3,15E+07	3,69E+07	3,45E+07	4,08E+07
REMOCONCHUBO	4,31E+07	3,66E+07	2,61E+07	7,54E+07
WOW	2,04E+07	2,23E+07	2,80E+07	5,23E+07
GNUTELLA	1,75E+07	3,09E+07	2,34E+07	2,91E+07
RTSP	1,65E+07	1,37E+07	1,23E+07	9,76E+07
SSH	3,57E+07	1,84E+07	1,72E+07	5,00E+07
OTHERS	1,39E+08	2,23E+08	5,63E+08	2,77E+08
<b>Total</b>	<b>7,66E+09</b>	<b>9,17E+09</b>	<b>9,90E+09</b>	<b>1,03E+10</b>

Table 2: Network traffic monitored at the equinix-chicago-dirB data center



Application	1 day (Max)[bit/s]	1 week (Max)[bit/s]	1 Month (Max)[bit/s]	2 Years (Max)[bit/s]
HTTP	1,12E+10	4,75E+09	3,76E+09	3,71E+09
UNKNOWN_UDP	8,36E+08	4,11E+08	4,45E+08	6,48E+08
UNKNOWN_TCP	6,42E+08	3,23E+08	2,52E+08	2,97E+08
RTMP	2,40E+08	4,77E+08	1,84E+08	3,03E+08
HTTPS	2,54E+08	1,42E+08	1,01E+08	1,05E+08
SMTP	9,39E+07	4,78E+07	4,58E+07	3,29E+07
SQUID	9,69E+07	7,87E+07	9,63E+07	3,35E+07
QUAKE	5,66E+07	2,74E+07	2,16E+07	1,85E+07
BATTELENET	9,99E+07	2,35E+07	2,40E+07	1,77E+07
YAHOO_MES	3,11E+07	1,55E+07	1,19E+07	3,69E+07
NOPORTS_UDP	5,44E+06	4,61E+06	1,93E+06	8,96E+06
RTSP	2,16E+07	9,68E+06	1,06E+07	9,80E+07
IPSEC	3,28E+07	4,23E+07	2,55E+06	6,89E+07
ABACAST	1,65E+07	1,94E+07	3,22E+07	6,12E+07
OTHER	1,72E+08	1,91E+08	8,30E+07	2,19E+09
<b>Total</b>	<b>1,38E+10</b>	<b>6,56E+09</b>	<b>5,07E+09</b>	<b>7,63E+09</b>

Table 3: Network traffic monitored at the equinix-sanjose-dirA data center

Application	1 day (Max)[bit/s]	1 week (Max)[bit/s]	1 Month (Max)[bit/s]	2 Years (Max)[bit/s]
HTTP	3,22E+09	4,02E+09	4,22E+09	4,62E+09
UNKNOWN_UDP	1,67E+08	3,64E+08	3,45E+08	2,97E+08
UNKNOWN_TCP	1,78E+08	3,39E+08	3,68E+08	9,69E+08
RTMP	4,80E+07	8,60E+07	1,85E+08	3,71E+08
HTTPS	1,75E+08	1,85E+08	1,81E+08	1,91E+08
SMTP	3,60E+07	1,90E+07	1,30E+07	1,60E+07
NOPORTS_UDP	1,80E+07	7,30E+07	7,40E+07	1,57E+08
QUAKE	1,20E+07	1,50E+07	2,10E+07	1,85E+07
NOPORTS_UDP	1,80E+07	7,30E+07	7,40E+07	1,57E+08
POP	1,20E+07	1,30E+07	1,19E+07	1,90E+07
IPSEC	6,66E+06	1,60E+07	1,17E+07	7,90E+07
ABACAST	1,40E+07	1,30E+07	1,31E+07	1,90E+07
OTHER	1,67E+08	1,65E+08	1,47E+08	1,73E+09
<b>Total</b>	<b>4,07E+09</b>	<b>5,38E+09</b>	<b>5,66E+09</b>	<b>8,64E+09</b>

Table 4: Network traffic monitored at the equinix-sanjose-dirB data center

## 2.2.2 Protocol distribution

The nature of Internet traffic is difficult to decipher, not least because the entities that have the best vantage points to gauge such knowledge tend to be and must often be secretive about what the data they collect manage on the behalf of service providers. In this following section, we will discuss some observation that have been gleamed by researchers studying and classifying the nature of traffic on the Internet.

For example the Internet Study 2008/2009 IPOQUE's ISP [39] and university customers agreed to provide anonymous traffic statistics collected by PRX Traffic Managers [40] installed in their networks. Protocols and applications are detected with a combination of **layer-7 signatures using DPI and behavioural traffic analysis**. The regional coverage includes eight regions of the world: Australia, Eastern Europe, Germany, the Middle East and Southern Europe, Northern and Southern



Africa, South America and Southern Europe and in this study 8 ISPs and three universities are involved. At its peak the study monitored about **1.3 PB of user traffic**, **1.1 million of users** and the period measurements was **2 weeks**. Figure 3 shows the proportion of the overall network traffic generated for each class of considered protocols.

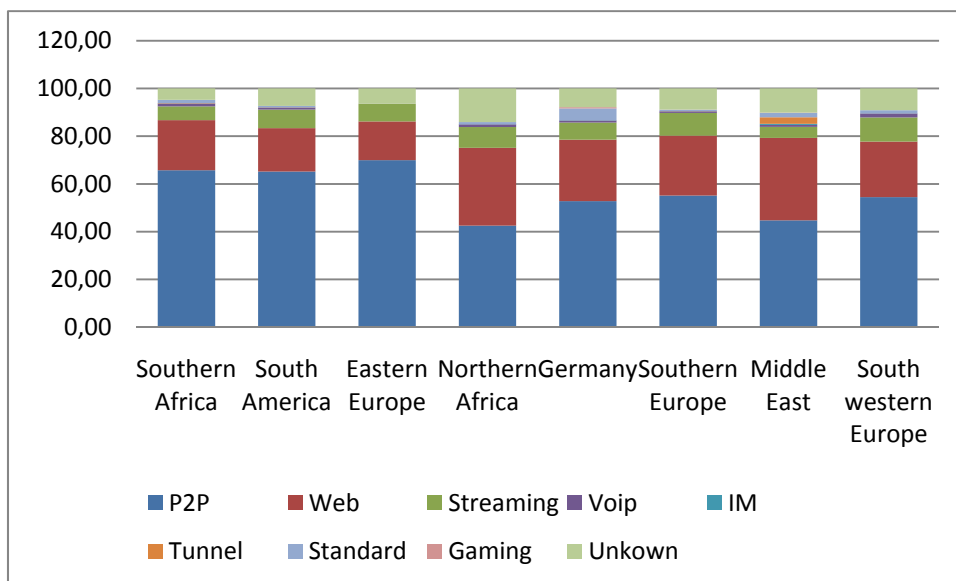


Figure 3: Distribution of protocol classes

During the term of the study, Peer to Peer file sharing (Ares, BitTorrent, eDonkey, Gnutella) generated far the largest portion of traffic in all monitored regions, ranging from the 43% in the Northern Africa to 70% in Eastern Europe. In particular in all the regions apart from South Africa BitTorrent is the dominating protocol.

Web traffic is the second largest traffic contributor after P2P worldwide ranging from 55% of South America to 86% in Southern Africa. These figures do not include embedded audio or video streaming contents (such as YouTube or Flash movies) that are covered in the streaming traffic. The share of Web traffic is higher if compared with the IPOQUE’s internet study conducted in 2007. This indicates a higher growth in popularity than other services such as P2P have experienced. One reason certainly is the surging popularity of social networking sites such as MySpace, Facebook and LinkedIn.

While independent of popularity, the average size of Web pages is constantly growing. For example, a typical news portal page such as CNN.com or NYTimes.com has a size of 1-2 Mbytes.

HTTPS, the encrypted version of the HTTP protocol used for online banking and software-as-a-service (SaaS) sites such as Salesforce.com, only generates little traffic, between 1 and 3.2%. While file hosting sites, such as Rapid Share and Mega upload, generate a substantial amount of Web traffic between 12% in Southern Africa and 44% in South America, contributing up to 10% to the overall Internet traffic. File hosting services are mainly used to share and download large files.

Media streaming is the third traffic contributor in the world. The growing popularity of audio and video streaming is undeniable. Adobe’s Flash platform, that is used to embed multimedia content into Web pages, is the uncontested number one. It makes up between 60% and 83% of all streaming traffic in all regions as depicted in Figure 4. Video portals such as YouTube, currently ranked as the third most popular Web site by Alexa, use Flash to distribute commercial and user generated video content.

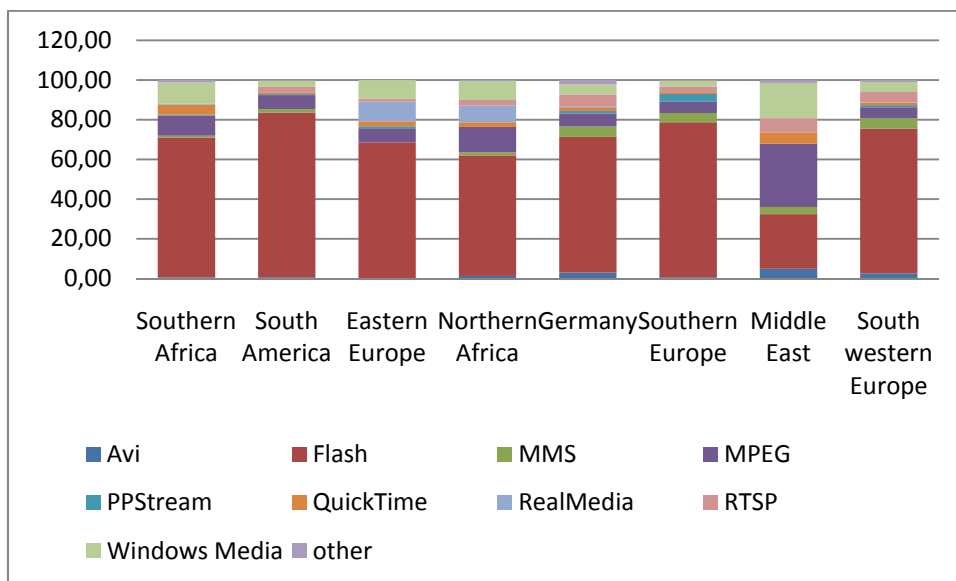


Figure 4: traffic distribution of media streaming

In one of the first large scale longitudinal studies of Internet inter-domain traffic [44], direct instrumentation of peering routers across multiple providers has been proposed. The study addresses significant experimental data collection and commercial privacy challenges to instrument 3,095 peering routers across 18 global carriers, 38 regional Tier-2, and 42 consumer and content providers in the Americas, Asia, and Europe. At its peak, the study monitored more than **12 terabits per second** of offered load and a total of more **than 200 exabytes of Internet traffic** over the two year life of the study (July 2007 to July 2009). Given the limitations of port-based application classification, the study dataset has been augmented with a smaller set of application statistics based on payload classification (DPI). From Table 5a, we see the majority of inter-domain traffic in 2009 consists of web (52%). SSL and other ports besides TCP port 80 account for less than 5% of this number. Video category represents the second largest application group at 2.64% and VPN protocols rank third at 1.41% followed by email at 1.38%. All other protocols including games, ftp, and news account for fractions of one percent of inter-domain traffic. In general ports and protocols alone provide a limited view of Internet application usage and Table 5a includes a sizable 46% and 37% percentage of unclassified traffic in 2007 and 2009, respectively. Since port-based classification only discovers the control traffic for many file transfer and multimedia protocols, Table 5a significantly underestimates traffic for video, P2P and file transfers.

We next look at payload based traffic breakdowns from the five consumer deployments in Table 5b. Shown values represent the average percentage of subscriber traffic attributed to each application group. We note that the configured application classifications used by the inline commercial appliances differ from the categories in Table 5a, including the lack of an explicit matching category for SSH and FTP. The “Other” category in Table 5b includes dozens of less common enterprise, database and consumer applications. Overall, the application breakdowns correspond closely between the two tables with the notable exception of P2P.

Both Table 5a and Table 5b show Web contributing 52% of Internet traffic and similarly close percentages for games and email. VPN and News shows a slightly larger variance between the two tables likely due to the consumer bias of the five inline deployments.

Data from the inline deployments also suggest that HTTP video may account for 25-40% of all HTTP traffic. In particular, one of the largest video sites, YouTube, uses progressive HTTP download. Payload analysis also suggests encrypted P2P/other ports represent another 10-15% of uncategorized traffic in Table 5a and other video/audio streaming protocols make up 3-5% of uncategorized traffic. Finally, the payload statistics show the remaining traffic consists of a heavy-



tailed distribution across hundreds of less common applications. Table 5a shows that well-known Web ports (i.e., TCP 80,443 and 8080) gained 10 percentage points between July 2007 and 2009.

Applications	2007	2009
Web	41,68	52
Video	1,58	2,64
Email	1,04	1,41
VPN	1,41	1,38
News	1,75	0,97
P2P	2,96	0,85
Games	0,38	0,49
SSH	0,19	0,28
DNS	0,2	0,17
FTP	0,21	0,14
Other	2,56	2,67
Unclassified	46,03	37

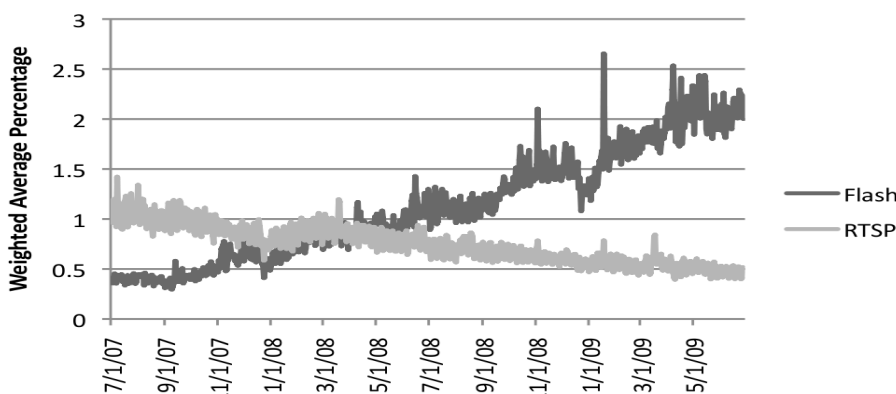
(a) Protocol/Port

Application	Av. [%]
Web	52,12
Video	0,98
Email	1,54
VPN	0,24
News	0,07
P2P	18,32
Games	0,52
SSH	N/A
DNS	N/A
FTP	0.16
Other	20,54
Unclassified	5,51

(b) Payload matching

**Table 5:** (a) top applications by weighted average percent of inter-domain traffic in July 2007 and 2009 based on port / protocol classification; (b) average application breakdowns in July 2009 across five consumer providers using proprietary payload and application behavioural classification heuristics.

Discussions with providers and analysis of the data from payload based classification of applications suggests that much of the Web (and particularly HTTP) growth is due to video. As a category, video represents both the second largest and second fastest growing application class. Table 5a shows a 1.05% growth in video protocol (i.e., Flash, RTSP, RTP, and RTCP) percentage points between July 2007 and July 2009. At the end of the study period, these video protocols represented a 2.64% weighted average of all inter-domain traffic. This growth in video corresponds to a widely documented increase in the popularity of Internet-based movie and television-based applications, including Hulu, YouTube, Veoh, and the BBC’s iPlayer. Further, data from payload based classification suggests up to 10% of HTTP traffic in Table 5a may be due to progressive HTTP download (e.g., YouTube). A graph showing the growth in video protocols is given in Figure 5.



**Figure 5:** Change in weighted average percent of video protocols inter-domain traffic contribution from July 2007 to July 2009

The graph shows the weighted average percentage of inter-domain traffic contributed by Flash and RTSP over the two year period of the study. Flash grew from 0.5% to 3.5% in two years, representing growth of more than 60%. Conversely, RTSP declined by .05% during the same period. Discussions with network operators suggests most of the RTSP traffic migrated to Flash and HTTP. These two protocols offer both more widely supported and simpler alternatives to RTSP. It is reported that many Internet IPTV offerings still use RTSP internally.



In conclusion the data reported above highlight that:

- HTTP is as expected by consensus the most popular protocol, the figures range wildly, on one account between 10% and close to 100% of the traffic observed, and in reported literature anywhere between 52% to 80% of the observed data-sets [39][41][42][43][44]. The interesting aspect of these measurements is that their variance seems to relate to the vantage point of the observer. Monitors near the user [41][43] tend to relate higher figures for HTTP, while those closer to the operator [42][44].
- HTTP is used to deliver more than 80% of multimedia data, which itself makes up the biggest category component in the traffic matrix.
- Understandably, the growth in HTTP traffic – as show in the data - driven by the changes in the user consumption of media on the Internet (see Table 5a and Table 5b ); traffic figures consistently reveal and a move away from P2P technologies which had in the past decade driven the major growth in Internet traffic, in fact according to recent results reported above, P2P represents the fastest decreasing component.



## 3 Review of state of the art DPI algorithms

Generally there are three methods for traffic monitoring and recognition. The traditional and straightforward is mainly port-based. In this approach, TCP or UDP port number fields are extracted from the packet header and analyzed, with the purpose of associating identified port numbers to higher layer applications. This can be done for example resorting to the Internet Assigned Number Authority (IANA) list of registered or well-known ports [47]. For example, if the TCP port number is 80, we can identify the flow as HTTP according to IANA. However, with the explosive rise of Internet traffic and the emergence of new network applications, port-based strategy becomes ineffective because many applications tend to hide themselves by using random ports instead of well-known ports.

One kind of alternative approach is the use of *machine learning (ML)* [88] [89] [90] [91] methods that operate on the transport-level information to identify network applications. This classification technique relies on the hypothesis that different classes of applications have flow behaviour characteristics that can be used to identify them on the network. Relevant examples of features that can be used to characterize applications are the packet size distribution per flow and the inter-arrival time between packets.

Another line of research in this domain focuses on packet payloads to identify the application. In this kind of approach, not only the packet headers, but also the content of packet payloads are inspected to determine the application-level protocol. DPI techniques can be categorized in two broad families. In the first family of DPI methods, the packets are considered as individual events, and we will refer to this family as *Stateless DPI*. *Stateless DPI* techniques match packet payloads with patterns of network applications, which are represented by means of exact strings or regular expressions. The purpose is to check whether a pattern appears in the packet payload or not. Obviously, *Stateless DPI* provides high identification accuracy; it can also map a packet flow to a specific application but not to a class of applications.

The second family of DPI methods is known as *Stateful DPI*. In this case, instead of processing packets as individual events, the DPI node fully reconstructs single traffic flows and the Layer 7 state of each specific application session. By maintaining state information, the *Stateful DPI* mode easily identifies applications that employ dynamically assigned port numbers, and tracks applications that involve multiple inter-related or spawned flows commonly found in voice-over-IP (VoIP) or multimedia streaming protocols.

In the following Sections, some of the relevant *Stateless* and *Stateful DPI* methods are reviewed.

### 3.1 *Stateless algorithm*

Originally the patterns to be searched in packets were simply represented by means of exact strings. However, regular expressions are replacing exact strings as the choice of pattern matching languages in packet payload scanning applications, increasingly new and emerging systems such as security appliances are using regular expressions. For example, all the protocol patterns adopted in L7-filter [45], which is an application layer packet classifier for Linux, and the rule set used by Snort [46], which is an open source network intrusion detection systems, are described as regular expressions. The widespread use of regular expressions is due to their expressive power, simplicity and flexibility for describing useful patterns.

The most popular method to implement regular expression matching is to use Finite Automata (FA). In this method, a finite automaton is built based on given regular expressions, and is run with packet payload as input. The finite automaton is either deterministic or non-deterministic, depending on underlying technologies and available resources.



**Non-deterministic finite automaton (NFA)** requires more state transitions per character in the payload thus having a time complexity for lookup of  $O(m)$ , where  $m$  is the number of states in the NFA. On the other side, NFAs are very efficient in terms of space usage compared to a deterministic counterpart. These properties make NFAs more suitable for ASIC (application-specific integrated circuit) [92][93] or FPGA (field-programmable gate array) [94][95][96][97] implementations, which can provide wide bandwidth but small amounts of on-chip memory.

On the other hand, **Deterministic Finite Automaton (DFA)** requires only one state traversal per character, while it needs a much larger amount of memory; consequently DFAs are more suitable for software implementation on general purpose processor.

Many works have been recently presented with the goal of memory reduction of DFAs. In [51], Kumar et al. introduce the **Delayed Input DFA (D<sup>2</sup>FA)**, a new representation that allows reducing space requirements [52]; since many states have similar sets of outgoing transitions, redundant transitions can be replaced with a single default one. This approach has proved to be able to save more than 95% of the memory required by DFAs. The drawback of Delayed Input DFA is the traversal of multiple states when processing a single input character, which entails a memory bandwidth increase to evaluate regular expressions.

To address this issue, Becchi and Crowley [53] introduced an improved yet simplified algorithm (called **BEC-CRO**) which results in at most  $2N$  state traversals when processing a string of length  $N$ . This work is based on the observation that all regular expression evaluations begin at a single starting state, and the vast majority of transitions among states lead back either to the starting state or its near neighbours. From this consideration and by leveraging (during automaton construction) the concept of state distance from the starting state, the algorithm achieves comparable levels of compression with respect to D<sup>2</sup>FA, with lower provable bounds on memory bandwidth and lower complexity.

The work presented in [54] also focuses on the memory problems of DFAs, by proposing a technique that allows non equivalent states to be merged, thanks to a scheme where the input and output transitions in the DFA are labelled. In particular, the authors merge states with common destinations regardless of the characters which lead to those transitions (unlike D<sup>2</sup>FA). This creates additional opportunities for state merging, so achieving higher memory reduction. Moreover the authors regain the idea of bitmaps for compression purposes. Using bitmaps is a natural first step toward achieving memory compaction for DFAs. Bitmaps have been used before for packet classification [56] and [57] as well for string matching using Aho-Corasick state machine [58]. In [54] the bitmap data structure proposed for string matching has been extended to DFAs and a modification named pointer indirection has been introduced.

Data-set		Characteristics				Before Merging			After Merging		
IDS	source file	RegEx	Single wildcards	.	*	{x,y}	States	Transitions	Distinct Trans.	States	Distinct Trans.
Bro v0.8	ex-web	233	3	-	1		3,287	841,472	181,783	450	11,181
Bro v0.9	sig-addendum	32	15	-	1		3,187	815,872	79,561	2,251	41,754
SNORT 07/2006	policy	5	-	2	-		154	39,424	818	19	64
SNORT 07/2006	p2p	6	-	6	-		210	53,760	1,003	27	89
SNORT 07/2006	web-php	15	1	1	-		1,086	278,016	14,464	435	4,169
SNORT 07/2006	spyware/http	7	1	10	-		4,984	1,275,904	39,016	534	2,053
SNORT 07/2006	spyware/port25	20	1	21	-		7,000	1,792,000	64,844	322	1,686
SNORT 07/2006	backdoor	13	-	7	-		7,183	1,838,848	55,444	388	1,696

**Table 6: Summary of characteristics of the dataset used and of the corresponding DFAs**

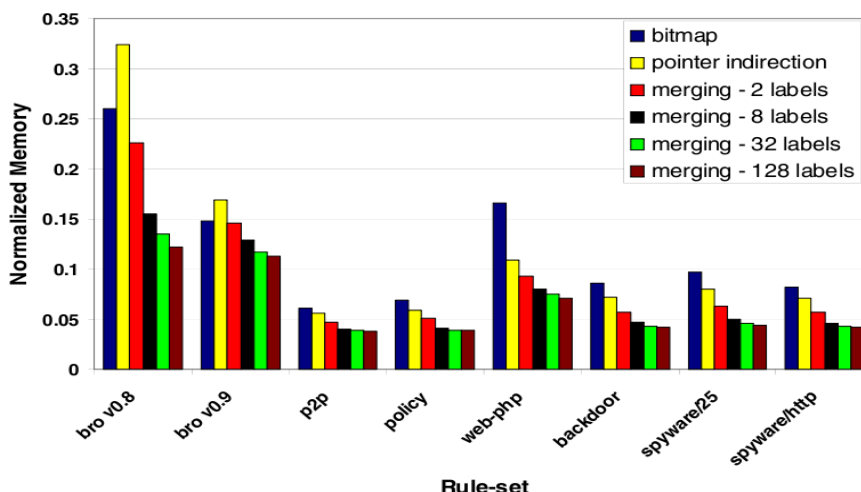


Figure 6: Memory reduction with state merging

In Table 6 columns one to six provide details on the sources of the regular expressions and on their general characteristics, whereas columns seven to eleven report the simulation results in terms of number of states and transitions achievable before and after the merging algorithm. Moreover Figure 6 compares the memory requirement of several solutions: the plain bitmap solution, bitmap with pointer indirection and different cases of state merging with 2, 8, 32 and 128 labels. The results reported both in Table 6 and Figure 6 show that the achievable memory compression factors are in the range 10 to 25 when comparison is done basic DFA data structures, and around 5 with respect to the bitmap based data structure.

In [59], Kumar et al. analyze three main limitations of the traditional DFAs. First, DFAs do not take advantage of the fact that normal data streams rarely match more than a few initial symbols of any signature; the authors propose to split signatures such that only one portion needs to remain active, while the remaining portions can be “put to sleep” (in an external memory) under normal conditions. Second, the DFAs are extremely inefficient in following multiple partially matching signatures and this yields the so-called state blow-up. A new improved Finite State Machine is proposed by the authors in order to solve these problems. The idea is to construct a machine which remembers more information, such as encountering a closure, by storing them in a small and fast cache, which represents a sort of history buffer. This class of machines is called History-based Finite Automaton (**H-FA**) and shows a space reduction close to 95%. The third limitation of DFAs is related to their incapability of keeping track of the occurrences of certain sub-expressions, thus resulting in a blow-up in the number of states. The authors introduce some extensions to address this issue in the History-based counting Finite Automata (**H-cFA**). The idea of adding some information to keep the transition history and, consequently, to reduce the number of states has been retrieved also in [60] and [61], where another scheme, named extended FA (**XFA**), is proposed. In more details, XFA augments traditional finite automata with a finite scratch memory used to remember various types of information relevant to the progress of signature matching (e.g., counters of characters and other instructions attached to edges and states). The experimental tests performed with a large class of NIDS signatures showed time complexity similar to DFAs and space complexity similar to or better than NFAs. Moreover both H-cFA and XFA schemes address, in a very similar way, the issue of state blow-up in DFA for multiple regular expressions. Therefore they are good candidates to be adopted in platforms equipped with a limited amount of memory, such as for example network processors, FPGAs or ASICs.

Ficara et al in [50] introduce a novel compact representation of **DFA states** (named **δFA**) which is based on the observation that, since most adjacent states share several common transitions, it is possible to delete most of them and take only transitions that have different source and destination states. This algorithm allows for iterative reduction of the number of states and for faster string



matching. Moreover a new state encoding scheme, Char State (**C-S**), enables compression based on input characters.

Different algorithms are compared in Figure 7 in terms of speed performance in memory access and space requirements. It is evident that  $\delta$ FA solution almost achieves the same compression rate as D2FA and BEC-CRO, while it guarantees higher speed (close to that of DFA). Moreover, by combining  $\delta$ FA with other methods, a general performance increase is obtained, as shown by the integration with XFA or H-cFA.

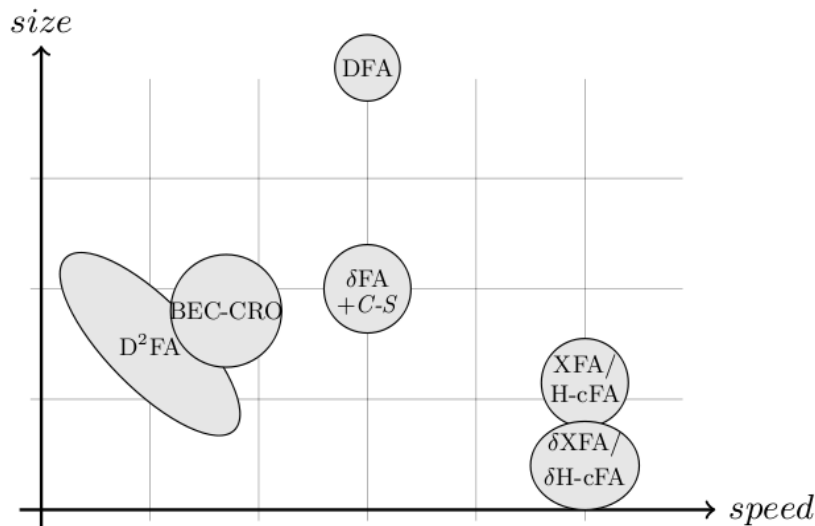


Figure 7: Comparison of speed performance and space requirements for the different algorithms

In [48] the impact of the complexity of regular expressions on the size of finite automata has been investigated. In particular the NFA based algorithm and DFA ( **$\delta$ FA**) based algorithm proposed by Ficara et al [50] have been implemented and compared using very long and complex regular expressions containing many wildcard repetitions. The number of NFA states and transitions are reported in Table 7 for several rule sets. Three sets of regular expressions from Snort rule sets, indicated as R1 , R2 , and R3 have been selected. They contain 98, 185, and 298 regular expressions, respectively. For comparison with simple regular expressions, the Authors of [48] also create three rule sets, E1 , E2 , and E3 , each of which has the same number and average length of exact-match regular expressions as the corresponding Ri. All the experimental results reported in [48] were obtained on an Intel 3.0 GHz Dual-Core machine with 4 GB main memory.

Rule set	# of states before/after compression	# of transitions before/after compression
E1	2746/1387	3596/1892
R1	2348/1206	15006/7220
E2	6044/3024	7124/3549
R2	5136/2631	28756/13398
E3	9554/4683	11624/5797
R3	8048/4176	41155/20014

Table 7: Number of NFA states and transitions

In particular each column of Table 7 contains the related numbers of states or transitions measured before applying the Ficara’s compression algorithm and after applying compression. From the reported results we can see that, in NFA, the number of states grows proportionally to the number



of regular expressions, and there is only a slight difference between the simple regular expressions and the complex regular expressions. On the other hand, the difference between the numbers of transitions is much larger. This result indicates that the complexity of regular expressions mainly affects the number of transitions, not the number of states.

The number of states in the NFA case is also compared with the DFA case in Table 8.

Rule set	# of states in NFA	#of states in DFA
E1	1381	1370
R1	1207	16673
E2	3024	3012
R2	2630	64297
E3	4781	4779
R3	4176	>200000
R3 with 3 DFAs	-	47235

**Table 8: State comparison between NFA and DFA**

Reported results show that the number of states grows exponentially in DFA and linearly in NFA, regardless of the complexity of regular expressions. The number of states of a DFA exceeds 200,000 for R3 rule set, which has fewer than 300 rules. The last row in Table 8 corresponds to a finite automaton implementing R3 with three DFAs. This is obtained by dividing R3 rule set into three components, building a DFA for each component, and then combining them using an NFA. The numbers of states of DFAs are 9304, 15782, and 22149, respectively. This means that we can use several small DFAs to replace a single huge DFA to reduce the number of states. Of course, such an approach tends to increase the memory bandwidth requirement, resulting in more memory accesses when there is not enough bandwidth.

In [48] an efficient DPI algorithm for regular expression matching is implemented on multi-core architecture. Regular expressions are split into NFA friendly components and DFA-friendly components, based on their complexity: for complex regular expressions NFA is more efficient, while DFA is preferable for simple regular expressions. The obtained components are then assigned to different cores. This hybrid method combines the merits of NFA and DFA implementations, and efficiently takes advantage of multi-core architecture. To evaluate the proposed hybrid approach, the authors build a single NFA on the main processor and a single DFA as coprocessor using the 298 rules in R3. Moreover a single DFA, a single NFA and the Hybrid proposed approaches are compared in terms memory consumption (state number) and processing speed (clock cycles/character):

- The single-DFA (running on a single core) achieves 29.72 cycles/character with cache hit rate 1%,
- the single-NFA (running on a single core) achieves 80.3 cycles/character with hit rate 47.89%,
- the hybrid approach achieves 23.98 (= 11.99 × 2) cycles/character with hit rate 64.31% for DFA, and almost 100% for NFA.

Figure 8, reports the maximum throughput achievable with the NFA, DFA and Hybrid solutions. From the evaluation results we can see that for the complex rule set, a DFA consumes too much memory while a NFA is slow due to multiple active states accessing memory in parallel. The algorithm outperforms these approaches not only in memory consumption but also in processing speed.

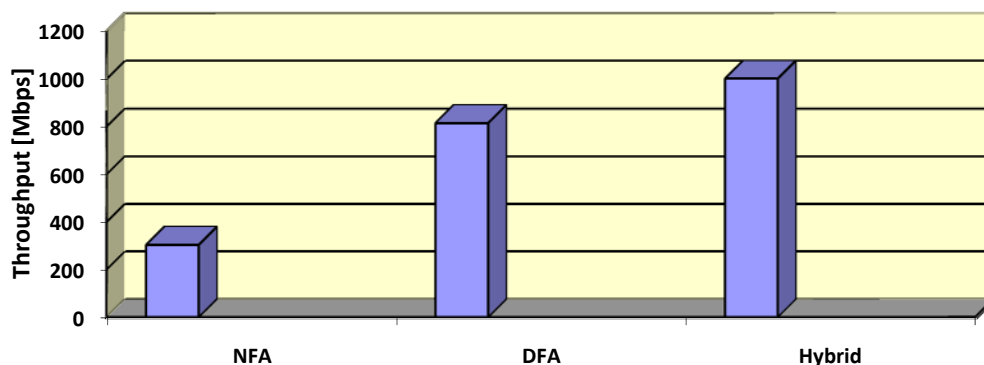


Figure 8: Performance comparison for NFA, DFA and the hybrid NFA/DFA achieved on 298 regular expressions from Snort. (Intel 3.0 GHz Dual-Core machine with 4 GB)

In [62] design alternatives in the implementation of regular expression matching architectures on network processors (NPs) and general purpose processors (GPPs) have been explored. Specifically, performance evaluations have been performed on an Intel IXP2800 NP, on an Intel Xeon GPP and on a multiprocessor system consisting of four AMD OPTERON 850 cores. The Intel IXP2800 NP is a highly-integrated multi-core processor designed specifically for networking tasks. One characteristic of the IXP memory hierarchy is the lack of caches.

In the comparison, the default transition compression to encode DFAs proposed in [53] (**BEC-CRO**) has been considered, together with the extension to NFAs [63] and the hybrid-FAs data structure proposed in [64] that brings together the strengths of DFAs and NFAs while avoiding their weaknesses.

The experiments are performed on a set of **534 complex regular expressions** drawn from the Snort. From the given results, the following elements can be highlighted. On a moderately provisioned NP based system, the sustainable performance varies from **20.2 Mbps to 90.5 Mbps** depending on the underlying automata representation and on the fraction of malicious content in the packet stream (between 35% and 95%). Similarly, the throughput varies from **2.4 Mbps to 192.7 Mbps** on the Intel Xeon, and from **15.6 Mbps to 534 Mbps** on a 4-way AMD OPTERON 850.

In [65] an high-speed and memory efficient system for traffic classification with deep packet inspection in Chip Multi- Processor (CMP) architecture has been proposed. In particular the application level protocol distribution has been analyzed and using these characteristics a reasonable architecture for traffic classification in multi-core servers has been proposed. Further a new universal DFA compression algorithm named **CSCA** has been proposed. The algorithm split the DFA into three parts and then compress them respectively. In order to show the compression effectiveness compatibility the Authors compare the CSCA algorithm with  $\delta$ FA proposed in [50]. The memory usage in most groups of regular expressions is less than that of  $\delta$ FA and value is relatively stable and around 5% whereas in the worst case the memory usage is less than 10% without significant impact on matching speed. Finally using real world traffic and all the L7-filter protocol patterns to make some experiments the system achieves a throughput variable from **876 Mbps to 1,5 GBps**.

### 3.2 Stateful algorithm

As specified above, a **Stateful Packet Inspection (SPI)** algorithms typically operate on packet flows or sets of flows. Basic operations inherent to such algorithm include: Packet classification, Flow state management. In the following Sections, some of the relevant Packet classification and Flow state managements algorithms are reviewed.



### 3.2.1 Packet classification

Packet classification performs searching the table of filters to assign a flow identifier for the highest priority filter that matches the packet in all the fields. The returning flow ID indicates the action that is next applied to the packet. An example of packet filter set is reported in Table 9. The standard five-tuple fields are shown in separate columns for the purpose of clarity. This table illustrates a small sample packet filter set with a very few bits for simplicity. Filters are usually sorted in the order of priority in the filter set. In this table x indicates wildcards inserted in any location of the fields. When a packet arrives, the first matching filter in the set is to be found in packet classifier. The matching filter should match the filter in all the five fields and the address (index) of the matching filter is used to point to the action that need to be applied to the packet.

Filter	Src. Addr.	Dest. Addr.	Src. Port.	Dsct. Port.	Prot.	Action
1	1001	01xx	2-4	7	TCP	Forwsrd 3
2	01xx	00xx	3-9	2-6	TCP	Forward 2
3	110x	10xx	1-7	4-6	UDP	Accept
4	1101	101x	x	x	ICMP	Queue
5	111x	01xx	x	x	x	Drop

Table 9: Example of packet filter set

Due to the complexity of the search, packet classification is often a performance bottleneck in network infrastructure; therefore, it has received much attention in the research community and several methods for packet classification have been proposed in the last years. All these algorithms can be grouped in four categories:

- Exhaustive search.** The most rudimentary solution to any searching problem is simply to search through all entries in the set. Let's assume that the set may be divided into a number of subsets to be searched independently. The two most common embodiments of the exhaustive search approach for packet classification are a linear search through a list of filters or a massively parallel search over the set of filters. Interestingly, these two solutions represent the extremes of the performance spectrum where the lowest performance option, linear search, does not divide the set into subsets and the highest performance option, **Ternary Content Addressable Memory (TCAM)**, completely divides the set such that each subset contains only one entry. Computational resource requirements for exhaustive search generally scale linearly with the degree of parallelism. Likewise, the realized throughput of the solution is proportional to the degree of parallelism. Linear search requires the minimum amount of computation resources while TCAMs require the maximum, thus linear search and TCAM provide the lowest and highest performance exhaustive search techniques, respectively. Performing a linear search through a list of filters has  $O(N)$  storage requirements where  $N$  is the number of filters, but it also requires  $O(N)$  memory accesses per lookup. It is possible to reduce the number of memory accesses per lookup by a small constant factor by partitioning the list into sub lists and pipelining the search where each stage searches a sub list. If  $p$  is the number of pipeline stages, then the number of memory accesses per lookup is reduced to  $O(N/p)$ , but the computational resource requirement increases by a factor of  $p$ . TCAM devices perform a parallel search over all filters in the filter set [66]. TCAMs were developed with the ability to store a "Don't Care" state in addition to a binary digit. Input keys are compared against every TCAM entry, thereby enabling them to retain single clock cycle lookups for arbitrary bit mask matches. TCAMs do suffer from four primary deficiencies: high cost per bit relative to other memory technologies, storage inefficiency, high power consumption, and limited scalability to long input keys.
- Decision Tree.** Another popular approach to packet classification on multiple fields is to construct a decision tree where the leaves of the tree contain filters or subsets of filters. In order to



perform a search using a decision tree, we construct a search key from the packet header fields. We traverse the decision tree by using individual bits or subsets of bits from the search key to make branching decisions at each node of the tree. The search continues until we reach a leaf node storing the best matching filter or subset of filters. In general all the decision tree based algorithms achieve a good balance between speed and space. Decision tree algorithms are: Grid-of-Tries [67], [68]; Hierarchical Intelligent Cuttings (HiCuts) [69]; HyperCuts [70]; Fat Inverted Segment (FIS) Trees [71].

- Decomposition.** Given the wealth of efficient single field search-techniques, decomposing a multiple-field search problem into several instances of a single-field search problem is a viable approach. Employing this high-level approach has several advantages. First, each single-field search engine operates independently, thus we have the opportunity to leverage the parallelism offered by modern hardware. Performing each search independently also offers more degrees of freedom in optimizing each type of search on the packet fields. In general, solutions using decomposition tend to provide the high throughput due to their amenability to parallel hardware implementations. The high level of lookup performance often comes at the cost of memory inefficiency and, hence, capacity constraints. Well known decomposition algorithms are: Parallel Bit-Vectors (BV) [72], Aggregated Bit-Vector (ABV) [73], Cross-producting [67], Recursive Flow Classification (RFC) [74], bitmap-RFC [78], Parallel Packet Classification ( $P^2C$ ) [76], Distributed Cross-producting of Field Labels (DCFL) [75].
- Tuple Space Search.** The basic tuple space search algorithm (Suri et al [77]) decomposes a classification query into a number of exact match queries. The algorithm first maps each dimensional rule into a tuple whose component stores the length of the prefix specified in the dimension of the rule (the scheme supports only prefix specifications). Hence, the set of rules mapped to the same tuple are of a fixed and known length, and can be stored in a hash table. Queries perform exact match operations on each of the hash tables corresponding to all possible tuples in the classifier.

Generally performance of different algorithms can be evaluated with two aspects: theoretical analysis of worst case complexity and experimental comparison of mostly average case.

**Error! Reference source not found.** is a list of worst case analysis for some of the aforementioned algorithms, where  $N$  is the number of filters,  $W$  is the number of bits used to specify the filter, and  $d$  is the number of fields contained in each filter.

Algorithm	Worst case time complexity	Worst case storage complexity
Linera Search	$N$	$N$
TCAM	$1$	$N$
Cross-producting	$dW$	$N^d$
HiCuts	$d$	$N^d$
RFC	$d$	$N^d$
Grid-of-Tries	$W^{d-1}$	$N^d$
Parallel BV	$\log N$	$N^2$
Tuple Space Search	$N$	$N$

Table 10: Worst case complexity comparison



Traditionally flow level packet processing devices rely on ASIC to perform IP forwarding at line-rate speed (10Gbps), however these device are only limited to be used at the backbone due to several issues:

- **Programmability.** Most of the ASIC architecture have special design for high performance which in turns lead to less general programming ability. This trends a trade-off between performance and programmability.
- **Special chips required.** Special chip like TCAM can accelerate the packet processing speed. However it requires too much power and board are to support large number of classification rules. Also special chips usually mean higher cost, longer time to market and more difficulties in product upgrade.

Today some researcher are seeking to combine the intelligence of software based solution and the performance of hardware based architectures, with the help of multi-core processors (NP), which can support both flexible software programmability and powerful hardware level packet processing. D. Srinivasan and W. Feng in [79] implemented the basic BV flow classification algorithm on Intel IXP1200 network processor [98]. They compared parallel mapping and pipelined mapping of the algorithm and showed that parallel mapping has better processing rate. The limitation of their work is that the bit-vector algorithm can only support 512 rules, which is too small compared to current real-life rule sets.

Recent work presented by D. Liu et al. in [80] proposed a modified RFC algorithm, named Bitmap-RFC, which reduces the memory requirements of RFC by applying a bitmap compression technique. Experimental results show that the Bitmap-RFC algorithm achieves **10 Gbps** speed on Intel IXP2800 NP [99]. Although Bitmap-RFC obtained high throughput on NP, it still requires large memory storage compared to other algorithms such as HiCuts and HyperCuts. Moreover, the total amount of memory words loaded from SRAM is 5 times larger than that of the original RFC algorithm. This affects the SRAM bandwidth utilization, and hence might reduce the overall performance of the whole application.

In [81] an effective flow classification algorithm is developed to support fast path parallel processing. The algorithm, named *Parallel Searching Cross-Producing* (PSCP), makes optimization on space mapping based on *the Cross-Producing* algorithm and its recent extension *HSM* [82] to improve the time performance on IXP2850 NP [99]. Worst-case memory accesses, as the most important performance metrics of flow classification algorithms, provide an evaluation of the worst-case processing speed. Figure 9 shows the worse-case memory accesses of the original *Cross-Producing*, *Binary Search Cross-Producing* (BSCP) and *Parallel Search Cross-Producing* (PSCP) with five real-life rule sets, from which it is possible to see that the worse-case memory accesses of PSCP is nearly 1/2 of that of BSCP and nearly 1/4 of that of Cross-Producing algorithm.

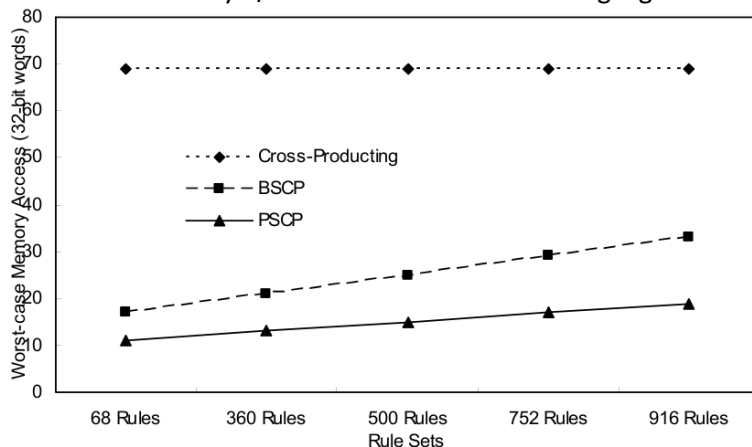


Figure 9: Worst case memory access



To evaluate the worst-case throughput on IXP2850, minimum 64Byte Ethernet packets are engaged here as the input traffic and all the packets are designed to match the longest prefix and reside in the leaf nodes of the tries. Figure 10 gives the worse-case throughput achieved by PSCP, BSCP and the original *Cross-Producing*. It is shown that PSCP reaches a throughput of **5-7 Gbps** with different sizes of rule sets, whereas BSCP obtains a throughput of **2.5-4.5Gbps** and *Cross-Producing* only reaches **1.7Gbps** throughput. Besides, the worse-case performance of PSCP and BSCP decreases slowly as the number of rules increases, because their single-field search is on the magnitude of  $\log(N)$ , where N is the number of rules.

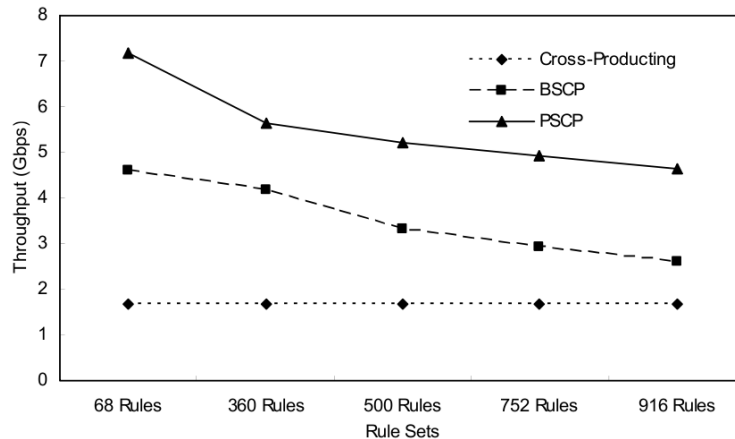


Figure 10: Throughput on IXP 2850

In [83] a new flow classification algorithm named *Aggregated Cuttings (AggreCuts)* has been proposed. It is based on the well-known *HiCuts* algorithm while optimized for multi-core network processors. Different from *HiCuts*, *AggreCuts* has explicit worst-case search time. To avoid burst of the memory usage, *AggreCuts* adopts a hierarchical space aggregation technique that significantly compresses the decision-tree data-structure. The Authors compare three main performance metrics of the *AggreCuts* to the *HiCuts* algorithm: the worst-case memory access, total memory usage and throughput on IXP2850. The testing rule sets, denoted as SET01-SET07, are all real-life 5-tuple ACLs obtained from large enterprises and from [84]. The size of these rule sets ranges from 68 (SET01) to 1530 (SET07). *AggreCuts-4* refers to  $w=4$ , i.e. at each internal node, the current search space is cut into  $2^4$  sub-spaces. Similarly, *AggreCuts-8* refers to  $w=8$ . Figure 11 shows that the worst-case memory accesses of *AggreCuts-8* and *AggreCuts-4* are less than 1/6 and 1/3 of that of *HiCuts* respectively. This is because the worst-case tree depth of *HiCuts* depends on the data structure of the rule set, while that of *AggreCuts* is set independent due to the explicit cutting scheme. Such a definite worst-case memory access is expected to guarantee stable performance of high-speed flow classification on the network processor.

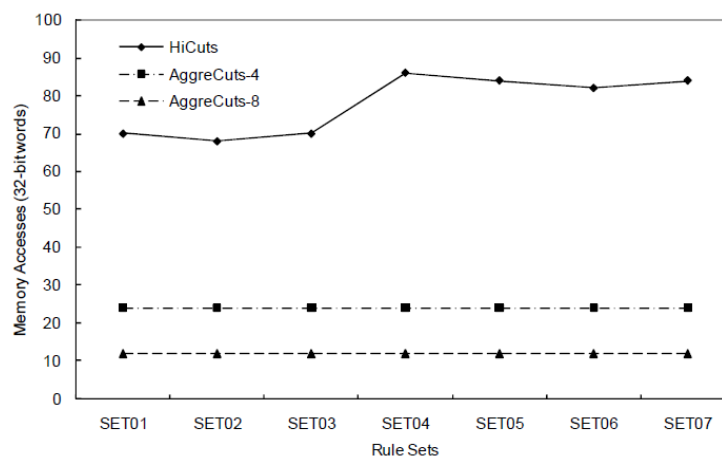


Figure 11: worst case memory access



To evaluate the throughput of the worst-case performance on IXP2850, we use minimum 64Byte Ethernet packets as the input traffic and set each packet to match the longest tree depth (i.e. each packet will incur the worst-case memory access). Figure 12 shows the throughput achieved by *AggreCuts-4*, *AggreCuts-8* and *HiCuts*. From this figure, we see that *AggreCuts-8* reaches nearly **9 Gbps** throughput and *AggreCuts-4* also has a stable **6Gbps** performance. In comparison, the throughput of *HiCuts* is less than **2Gbps**, and as the number of rules increases, its performance slowly decreases.

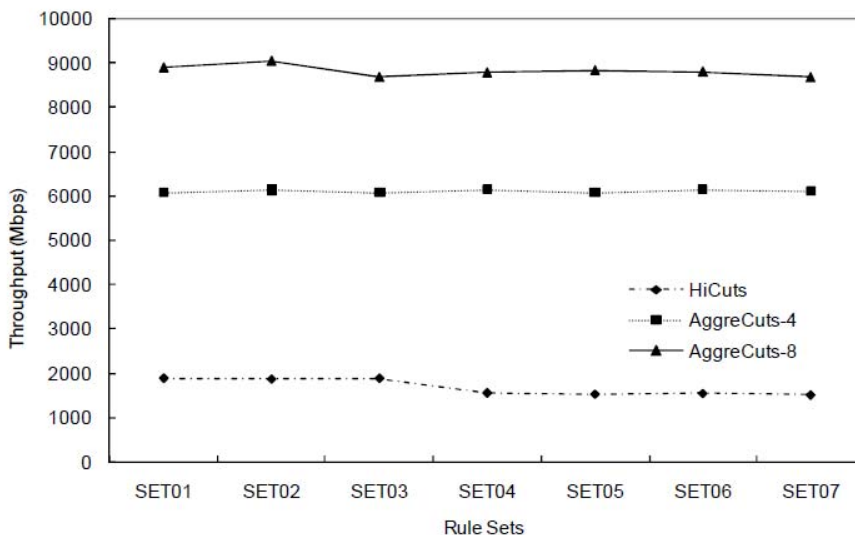


Figure 12: Throughput on IXP 2850

In [85] a novel packet classification algorithm named *Discrete Bit Selection (DBS)* has been proposed. This algorithm adopts a bit level heuristic to detect the inherent characteristics of the rule set. Due to the bit level heuristic, rule sets can be partitioned more efficiently, thus the storage required for data structures is significantly reduced. At the same time, to guarantee the speed of classification, two levels of flat structures are adopted, which require only two memory access times while searching.

To validate the performance of *DBS* on real NP platform, the Authors built the classification application on the Cavium OCTEON 5860 network processor [86]; for comparison, also the traditional well known algorithms *HiCuts* and *HSM* have been implemented on the same NP.

All the packet classification algorithms are evaluated on three different types of real-life rule sets, containing Access Control List (ACL), FireWall (FW) and IP Chain (IPC). The Authors use 12 rule sets with different sizes from several hundreds to about ten thousand, e.g. the ACL1-10K rule set contains about 10,000 rules. All rules in the sets are of 5 tuples with 32-bit source/destination IP addresses, 16-bit source/destination port numbers, and 8-bit transport layer protocol. More details about the rule sets can be found at [87].

Both average-case and worst-case memory access times are illustrated in Figure 13 and Figure 14. As Figure 13 shows that *DBS* has the least average-case access times, in most rule sets, the average case access times is only about 5%-10% of than *HiCuts* (both *HiCuts-1* and *HiCuts-8*), while 10% - 20% of that of *HSM*. From Figure 14, we can see that the worst-case access time of *DBS* is only about 30% of that of *HiCuts* (both *HiCuts-1* and *HiCuts-8*), while is nearly the same with that of *HSM*. We notice that *HiCuts-8* needs more memory access times than *HiCuts-1* in average case. This is because the linear searching inside leaf nodes increases the access times. Though *DBS* also adopts linear searching inside blocks, however, the size of blocks can be keep small because of the effective partition.

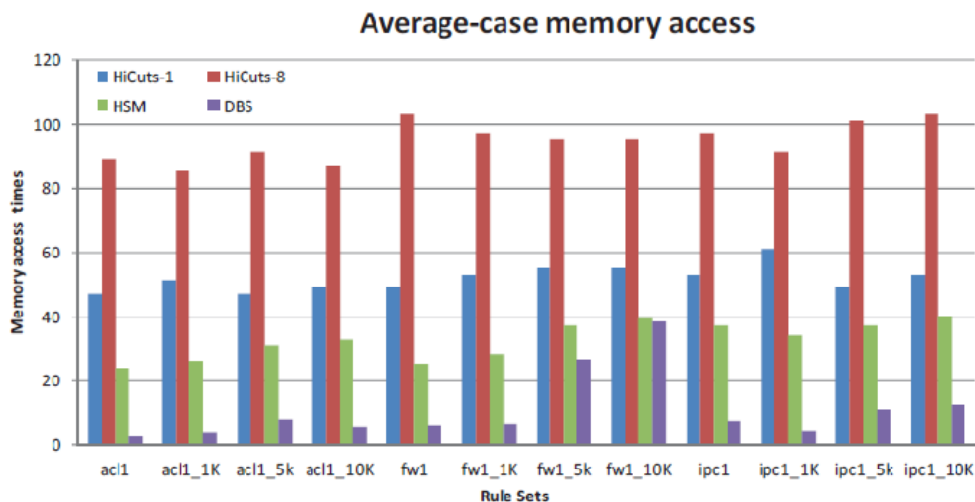


Figure 13: average case memory access times

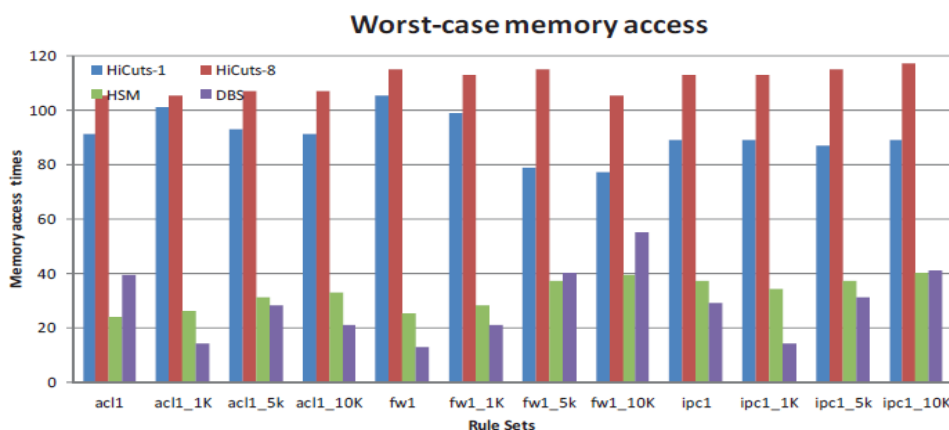


Figure 14: Worst case memory access times

Figure 15 - Figure 18 show the evaluation results of throughput performance on Cavium OCTEON 5860 multi-core platform. The rule set is ACL1-10K, which is the largest one that *HiCuts* and *HSM* can process successfully.

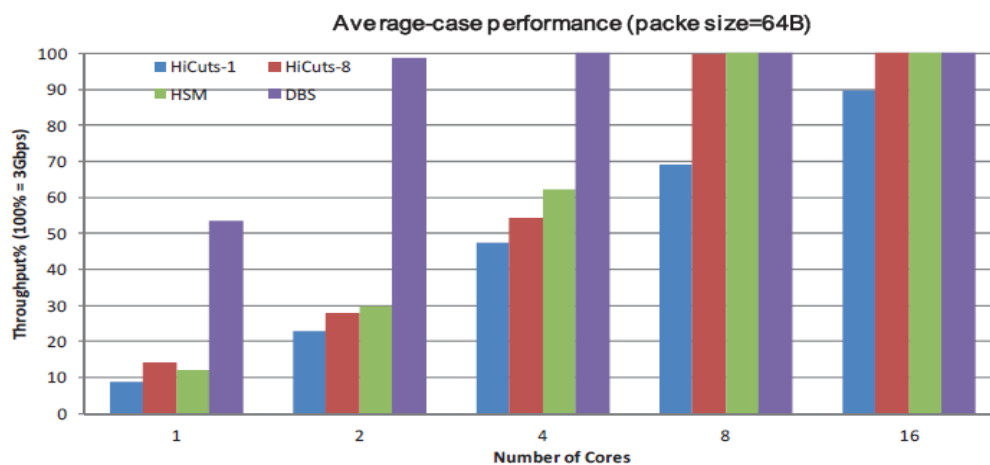


Figure 15: Average-case throughput performance (rules:ACL1-10K, #cores:1-16)

Figure 17 and Figure 18 show the average-case and worst-case throughput performance with different number of cores while the packet size is 64 Bytes. For the minimum 64 Bytes Ethernet packet, only *DBS* can achieve nearly 100% throughput (3Gbps) with 2 cores in average, while *HiCuts*-



8 and *HSM* need 8 cores, and *HiCuts-1* cannot achieve the full throughput even with all the 16 cores. Even in the worst-case, *DBS* is the only one which reach 100% throughput with 4 cores.

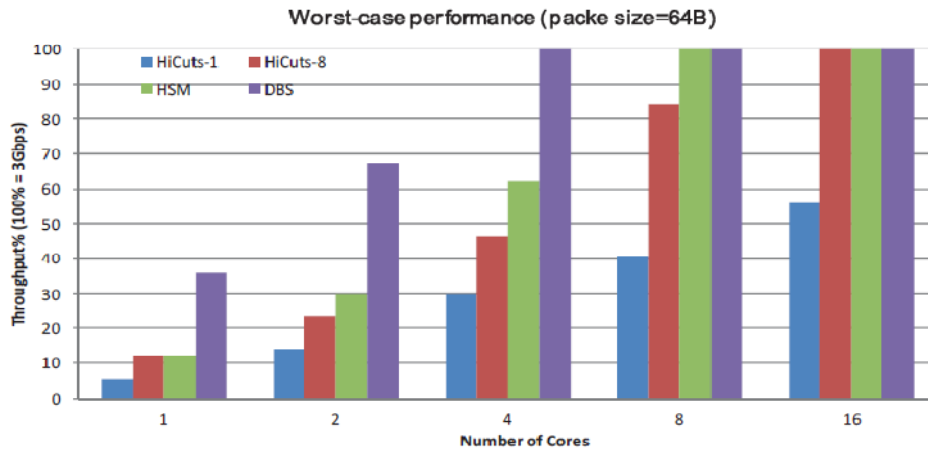


Figure 16: Worst-case throughput performance (rules:ACL1-10K, #cores:1-16)

Figure 17 and Figure 18 show the throughput results on only 1 MIPS core with different packet sizes (64 Bytes -1518Bytes). When input packet is 128 bytes, *DBS* can reach nearly 100% throughput in average-case and higher than 60% (about 2 Gbps) in worst-case. This result is about 300% higher than all other three algorithms.

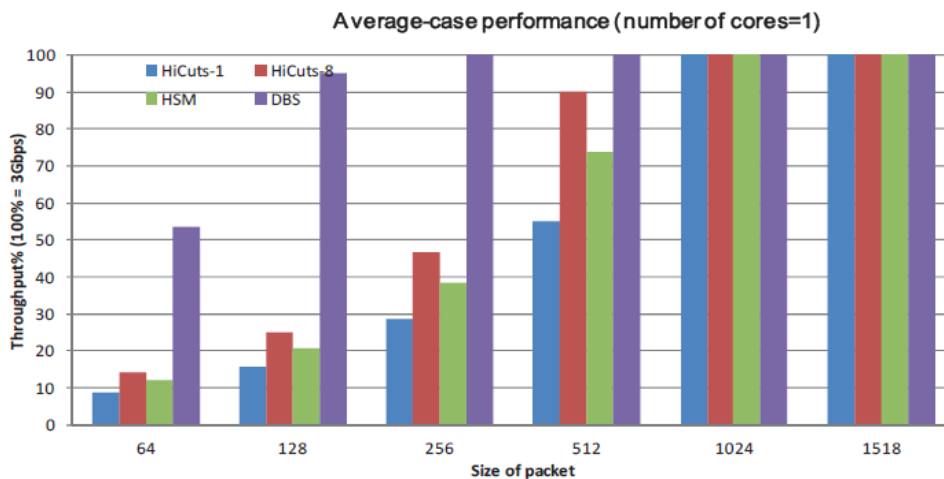


Figure 17: Average-case throughput performance (rules:ACL1-10K, packet: 64 -1518 B)

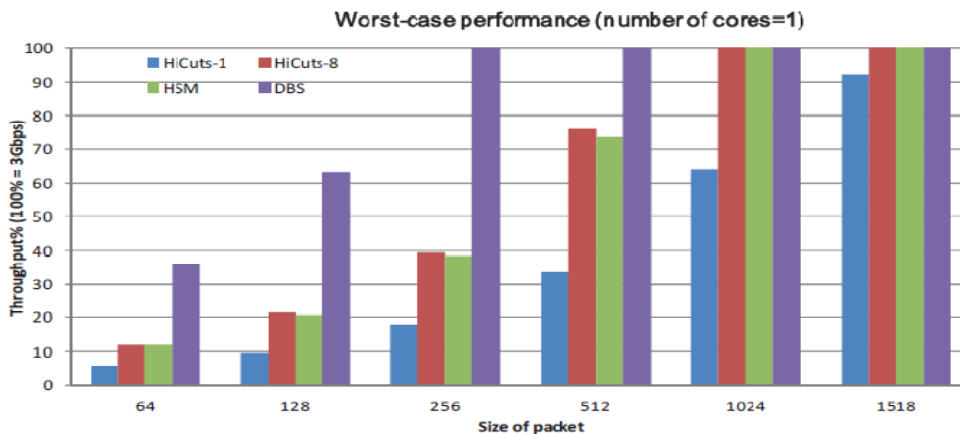


Figure 18: Worst-case throughput performance (rules:ACL1-10K, packet: 64 -1518 B)



### 3.2.2 Flow state management

Per-flow states are maintained in order to correctly perform packet processing at a semantic level higher than the network layer. Such *Stateful* analysis brings with it the core problem of state management: what hardware resources to allocate for holding state and how to efficiently access it. This is particularly the case for in-line devices, where flow state management can significantly affect the overall performance.

In general current session tables to store flow states can also be exceptionally large, on the order of 1 million entries. Hashing algorithms are well-suited for exact match problems and thus widely used for flow state management. The traditional hash scheme, which is named *DirectHash* in this document, stores flow states in a single hash table indexed by the hashing value and normally uses link lists to handle hash collisions. Each entry in the link list stores a full flow state consisting of the 5-tuple header, sequence and ACK sequence number, and other flow information according to different applications. When collision occurs, the link list entries will be checked one by one to find out the exact match. Although *DirectHash* is simple to implement on NP, this scheme suffers from: SRAM size limit, Excessive Memory Accesses.

To address these problems in [83] an efficient flow state management scheme named *Signature-based Hashing (SigHash)* has been proposed. Different from existed hashing table design, signatures for collision-resolving are stored in word-oriented SRAM chips in *SigHash*, while their corresponding flow states are stored in burst-oriented DRAM chips. Consequently, the flow state update speed at near line rate is guaranteed, and the large-storage requirement for millions of flows is met as well.

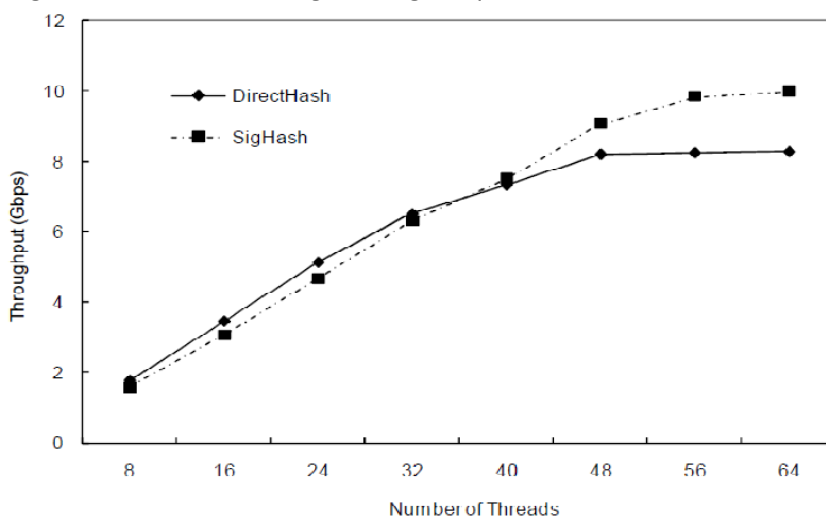


Figure 19: Direct Hash vs SighHash

Two hash schemes are implemented on the platform of Intel IXP2850 Network processor: the *DirectHash* scheme based on SRAM and the *SigHash* scheme based on SRAM as well as DRAM. From Figure 19, it is possible to see that the *DirectHash* scheme reaches **8.3Gbps** throughput with 64 threads, while the *SigHash* scheme reaches **10Gbps** line speed. The figure also shows that when the number of threads exceeds 40, the performance of *DirectHash* does not increase linearly. Moreover, with 64MB SRAM and 2GB DRAM on the IXP2850 NP, the *SigHash* scheme can support over **10M concurrent sessions** while the *DirectHash* scheme can only maintain less than **500K concurrent flow states**.

To address the memory problem of *Direct Hashing* in [81] an efficient TCP three-way handshake scheme designed for fast-path processing using a two-stage *Intelligent Hashing (IntelliHash)* has been proposed.

To evaluate the performance of *IntelliHash*, the Authors do experiments to compare its TCP three-way handshake processing speed with the *DirectHash* scheme.

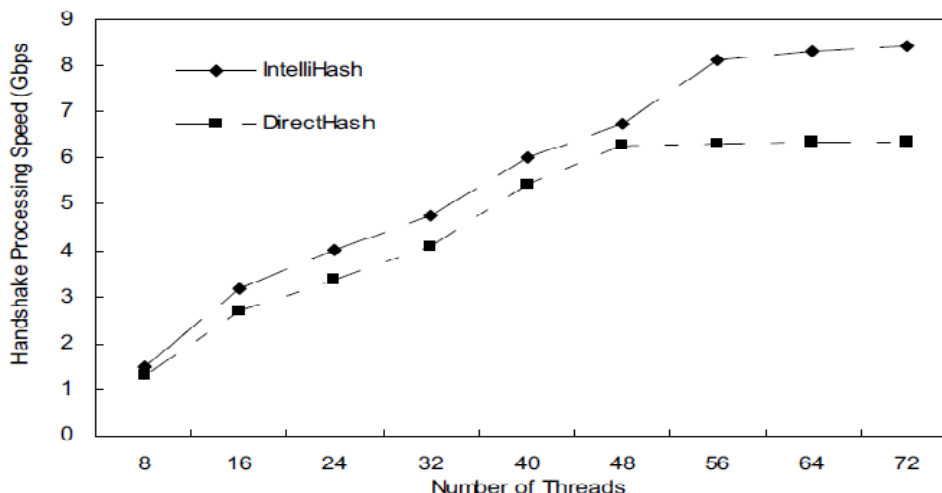


Figure 20: Direct Hash vs IntelliHash

From Figure 20, it is possible to see that the *DirectHash* scheme reaches **6.5 Gbps** handshake processing speed with 72 threads, while the *IntelliHash* scheme reaches **8.5 Gbps** speed. This is because the handshake processing of *IntelliHash* is implemented on SRAM while that of *DirectHash* can only be implemented on DRAM. Moreover the *IntelliHash* scheme achieves a session creation rate of **2M new sessions per second**.

In Section 4.2 the RTC\_MON tool has been widely described. In this section we will report only the performance achievable with the RTC-MON framework. To this purpose the testbed used for the experimental results consists of two computers. The first one has been used to generate both trash UDP and VoIP traffic; whereas the second one as monitoring system. The traffic generator consists of an AMD Athlon 2 GHz with 2 GB of memory running a Linux 2.6.24 kernel. The computer injects Voip traffic by replaying a packet trace with tcpreplay. The trace contained about 1000 calls each lasting 30s with 200 total number of concurrent calls in the trace. Moreover the test were run with a mixture of calls and other non-Voip traffic adding up to as many as 50,000 concurrent flows. The tcpreplay tool running on the same computer has been employed to generate trash UDP traffic with packet size of 250B.

The monitoring System has an Intel Centrino Duo running at 1.8 GHz, 2GB of memory, a Broadcom 1000 NIC and also run a linux 2.6.24 kernel. It receives the combination traffic from the UDP/VoIP generator. In the considered setup, RTC\_MON is able to process packets at about 865 Mbps keeping the CPU relatively IDLE about 80%. Moreover looking at the profiling data in Table 11, achieved running oprofile on the monitoring computer, we note that 99,99% of the CPU is allocate to the pf\_ring packet filtering.

Samples	%	Symbol name
70191	50.1	Add_skb_to_ring
47229	33.7	Skb_ring_hundler
11453	8.16	Parse_pkt
8202	5.84	Packet_rcv
3074	2.19	Ring_mmap
47	0.01	others

Table 11: RTC\_MON profiling results

To summarize in this section the most recent and sophisticated *stateless* and *stateful* algorithms available in literature have been briefly reported. For the *stateless* DPI algorithm, different memory size reduction and speed-up techniques in the context of regular expression matching



(NIDS system) have been compared both considering general purpose and network processor. Moreover techniques used in *stateful* DPI in order to both classify packets in different flows and maintain different flows have been compared in terms of memory and maximum throughput achievable. The COAST DPI node will be built on top of the existing RTC-Mon and will support the development of both stateless and *stateful* DPI algorithms.



## 4 High-level Design of the DPI node

Given the requirements listed for the DPI, it is clear that it needs to be capable of supporting a number of different tasks while still maintaining high performance. In this case, performance refers to the aggregate filtered packet throughput that DPI is able to achieve. Ideally, we would like a DPI to be capable of:

- i) capturing all the data that passes through the probe,
- ii) filtering their content while
- iii) minimizing the latency cost induced by this operation.

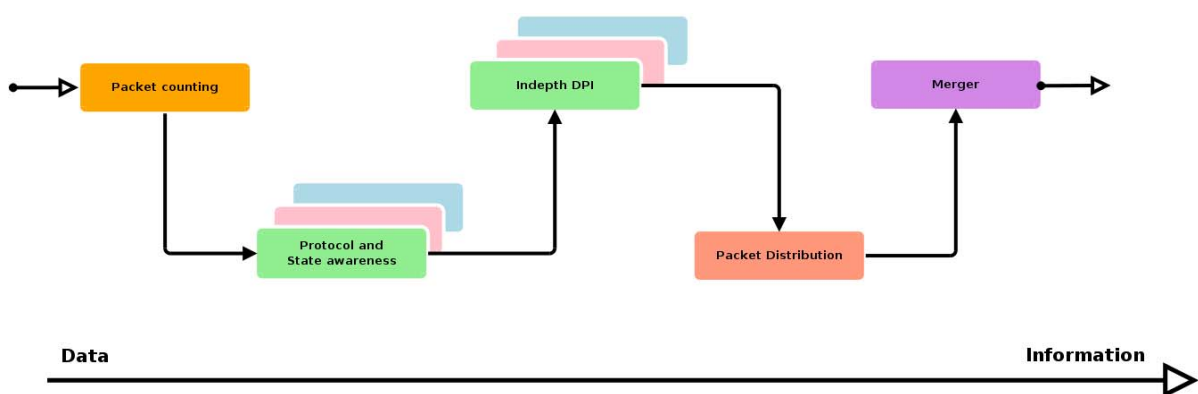
As is clear, these requirements raise contradictions; in capturing and filtering each packet that a probe encounters, we are explicitly increasing the computational cost of the work required at the probe, which in turn introduces latency in the time taken to do the work, and may potentially affect the likelihood of capturing/missing packets in the future.

Further, given the distributed set up envisioned for COAST nodes, we must keep in mind that probes/DPI stations may need to cooperate in order to fulfil the task of creating complete information about the state of the network – again a requirement that increases the workload at a given DPI terminal.

Therefore, to make such a mechanism work in practice, we must decompose its functionality in order to modularize it. In the following discussion, we first look at the functionality a single given node must provide, and then turn to how this information is distributed to other nodes.

### 4.1 The DPI node architecture

The functionality that the DPI offers can be divided in terms of data versus information; the lowest layers will handle basic data processing while higher layers will try to carry out more intelligent and in depth analysis, which may include the use of prior knowledge or protocol and application awareness as shown in Figure 21.



**Figure 21: High-level DPI packet processing architecture; DPI functionality is divided between functional layers which each handle a part of the over all process. Layer 1 performs basic packet counting, layer 2 provides protocol awareness, layer 3 implements deep packet inspection and layer 4 is responsible to distributing the data for merger.**

From Table 12, at the lowest layer of operation a given DPI node must perform; (i) content discovery, ii) popularity assignment and iii) content summarization. At this layer, we are chiefly



concerned with packet capture and processing throughput, nodes operating here aim not to minimize the number of packets they miss out on capturing – ideally all packets are captured.

Further, the operations carried at this layer depend on i) the expected traffic load at the node’s location in the network, ii) the complexity of the filtering operation required, i.e. packet header filtering as opposed to content analysis or protocol awareness and iii) the physical capabilities of the node relative to these constraints.

Operational layer	Function	Data in	Operation	Data out
<b>Layer 1</b>	Raw packet counting	<b>In:</b> Raw packet <b>Filter:</b> basic header based	Match in coming packets against the simplest filter to initiate the initial grouping and counting	<b>List:</b> <type, counter, pkt, pkt-hash>
<b>Layer 2</b>	Protocol awareness	<b>In:</b> <type, counter, pkt, pkt-hash> <b>Filter:</b> protocol filter	Apply protocol state awareness to in coming packets and schedule for action	<b>Set 1:</b> packet wait queue <b>Set 2:</b> packet control queue <b>Set 3:</b> packet filter queue
<b>Layer 3</b>	In depth packet filtering and analysis	<b>In:</b> packet filter queue <b>Filter:</b> application specific	Perform application specific filtering to the content of the <i>filter</i> queue	<b>List:</b> <type, counter, pkt, pkt-hash>
<b>Layer 4</b>	Packet distribution for merger	<b>In:</b> <b>List:</b> <type, counter, pkt, pkt-hash>	Package and distribute the filtered data	:<type, counter>

**Table 12: A Functional decomposition of the packet processing task performed at the DPI node**

For the sake of simplicity, we define the function of nodes at the lowest layer of operation as packet type counting, whereby type is defined as the filter constraint. For example, a very simple filter may define packets to be identified by their destination address; therefore, each type counter lists the number of matching packets. The functionality at this layer is deliberately limited, the aim here is not to perform deep packet inspection but instead to capture and group packets.

The Layer 2 functionality aims to contextualize the packet processing task to the protocol of interest. The result of processing at this functional layer is the creation of a set of packet queues for further processing. Clearly the number of queues and their content is arbitrarily defined, but we envision that at least three queues are necessary. These are;

- a protocol state queue - to define something equivalent to a signalling plane, indicating the current state of the protocol’s state machine. For example for an http *get* message, we may need to process the return header to identify the state of the interaction and if needed queue the packet for further processing later. Clearly this assumes that the probe is present on the return path.
- a protocol control queue - to handle how packets that are fall foul of the protocol state machine are handled. For example, packets to drop or pass on to a higher level controller for inspection
- a protocol filter queue – to handle packets that are ready for processing and DPI

The Layer 3 functionality performs the actual packet inspection. We envision this layer to contain the application specific logic for the filtering task and may be extended to process some n-layer



deep processing, for example, DPI of packet header, followed by the packet payload. Further, complex state that may require extra information such the past history may be processed here.

The Layer 4 functionality is responsible for packet distribution for merger. Data that arrives at this layer is assumed to have completed its DPI processing run and ready for consistency checking and merger. For the moment, the input to this layer is assumed to be lists of the data to distribute and its output a hash table listing the type and count, i.e. in the case of the popularity estimation, the key here is the URL and the count the number of hits associated with it.

As can be seen in the high-level description of our architecture, the DPI does not maintain state beyond the protocol state machine; therefore it does not have the database to be able to answer questions regarding the application or be capable of drawing inferences based on past history. This is deliberately the case. The aim question such as answering whether content has changed, a URL is new and so on are high-level and require semantically rich data sets and therefore incompatible with the scalability and raw performance requirements of the DPI node. Instead, these questions are answered by the results of the DPI process, i.e. using the data collated by the merger nodes.

## 4.2 PF RING and its recent extensions

In order to meet the requirements highlighted in Section 2.2, the COAST DPI node will be built on top of an existing framework called RTC-Mon (Real-Time Communications Monitoring framework). NEC Laboratories Europe designed RTC-Mon with the aim of helping software architects to develop monitoring applications scalable to ISP volume. The framework grants high performance and usability, hiding the complexity of traffic capture and protocol analysis and thus reducing the time to market of complex applications. RTC-Mon consists of an extensible, kernel-level plug-in architecture that performs high rate parsing and filtering. RTC-Mon is implemented on Linux OS for a variety of reasons, including its open source paradigm, the availability of kernel packages capable of processing packets at high rates, and the fact that Linux is used in a large range of platforms, from powerful servers to less capable embedded devices. Figure 22 depicts the RTC-Mon framework overview.

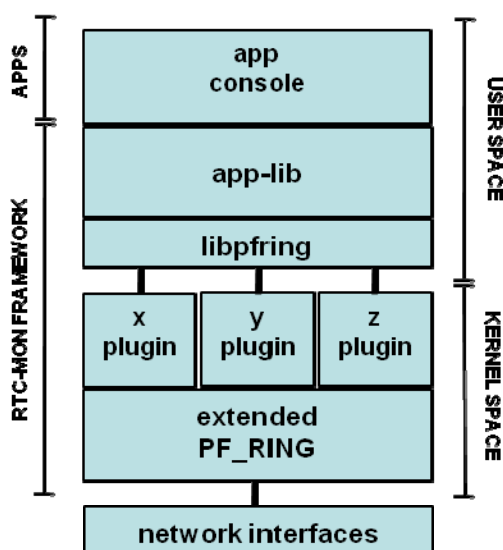


Figure 22: RTC-Mon framework overview

To achieve high performance, the framework relies on PF RING [2], a Linux kernel-level network socket that dramatically improves packet capture speed. PF RING was expanded by adding a plug-in architecture to it, thus allowing developers to extend it by creating high performance, custom plugins (see Figure 22) by interfacing to PF\_RING library (libpfring).



PF RING is a Linux kernel module providing a new type of socket that, coupled with device polling, allows packets to be captured at high rates. One of the advantages of the module is that it is independent of device drivers, and so can be used with any network card. In addition, PF RING comes with libpfring, a user-level library that allows applications transparent access to the kernel-level sockets. Finally, it was chosen because it is a mature project that has a wide and active community of users.

While PF RING has the basic mechanisms needed for packet capture, recently it was extended by NEC Laboratories Europe and other researchers [3] in order to meet the above stated requirements (e.g., IP defragmentation was added in order to process higher layers in the kernel, thus achieving higher performance). PF RING comes now with a plug-in architecture that enables easy implementation of higher-layer functions in the kernel.

The plug-in architecture allows the performance of a variety of crucial monitoring functions in the kernel, including packet payload parsing, packet content filtering and traffic statistics computation. Plug-in are essentially kernel modules, providing a simple way for developers to add support for functions and protocols to the framework.

The architecture allows for packets to be handled by one or many plug-in before being discarded, thus enabling the development of applications that rely on several protocols or functions. The process begins by creating a PF RING socket and assigning it to an interface<sup>1</sup>. The socket has a set of rules associated with it that decide which plug-in to send packets to. Each of these rules has three components: a filter, an ID identifying the plug-in to send the packet to in case the filter matches, and an action ID that decides what happens to the packet in case of a match once the plug-in has processed it.

Figure 23 illustrates the basic architecture. First, PF RING receives a packet on a device and parses its headers up to the transport layer, performing any IP defragmentation where needed. It then goes through the socket's set of rules one by one, applying a rule's associated plug-in to a packet only if the rule's filter matches (for instance, a rule for an HTTP plug-in could have a filter for TCP packets with port 80).

The filtering mechanism requires a closer look. As shown in the figure, filters can be of two types: hash or wildcard. Hash filters are used when it is necessary to track a six-tuple connection with the fields {van id, protocol, source IP, source port, destination IP, destination port} without incurring the linear evaluation costs of a rule list. The hash is managed by the plug-in, thus giving it the power to decide what connections to track and what state to keep; as we will show later, this is used by the RTP plug-in to track different calls. Wildcard filters, on the other hand, are more flexible, allowing matching, for example, all UDP packets going to a specific port. These filters can also specify a higher-layer, plug-in-specific filter. In this way, a user could instrument the system to process only packet containing specific L7 payload strings.

If a packet matches a rule's filter, the action determines what happens to a packet after it has gone through the rule and its plug-in (if the packet does not match the rule it is evaluated against the next rule). In our architecture, there are three options for the action:

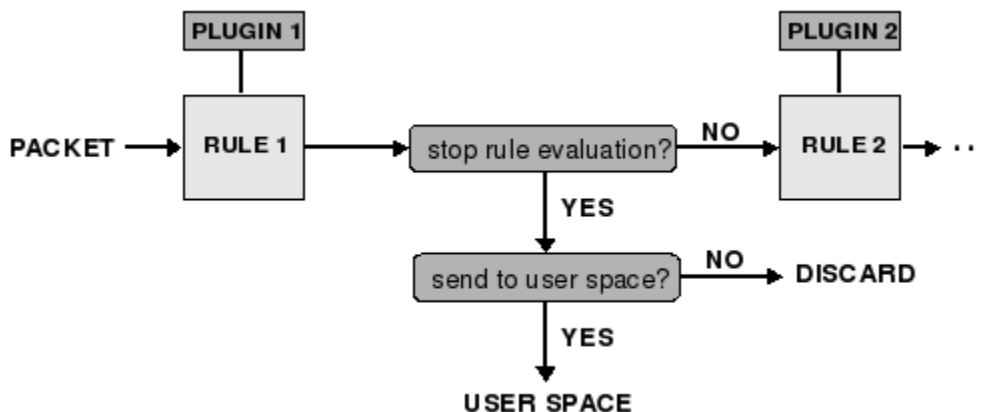
1. Continue rule evaluation.
2. Stop rule evaluation, send packet to user space.
3. Stop rule evaluation, do not send packet to user space.

---

<sup>1</sup> PF RING supports the creation of several sockets per interface, thus allowing several independent applications to run on the same interface.



The first option is straight-forward, allowing subsequent rules and plug-ins in a socket's set to also process packets. The other two stop the rule evaluation: if a packet has already been handled by the appropriate plug-in, a developer can use one of these two options to prevent any further and perhaps wasteful processing. Finally, a developer might need to pass some of the information gathered up to user space using option 2. Copying data to user space can be costly, however, and so option 3 is there to allow a developer to accumulate data in the plug-in that an application can poll from time to time.



**Figure 23: Extended PF RING overview with plug-in architecture.**

Creating a plug-in for the architecture is simple and consists of essentially implementing functions for parsing and filtering traffic as well as for polling packet statistics from user space.

In building up on the plug-in architecture of RTC-Mon the DPI node is componentized, in effect a COAST DPI node is a logical construct of several cooperating components. As such, the architecture address the scalability and performance issues raised since the components of a given DPI node may be distributed amongst several physical nodes each addressing a different requirement on the workload. Though timing and synchronization has been explicitly discussed here, we assume that the components that make up a given DPI node are synchronized and operate on an epoch related to their position in the workflow chain.

The definition of a filter based packet parsing strategy means that the entry point for the system is well defined, user may provide multiple chained filters to correspond with their interest and they system is flexible enough to accommodate a wide range of applications.

Therefore, to realize a COAST DPI node, users must implement; i) a PF Ring filter (rule) to identify the initial packets they interested in processing, ii) a protocol state machine to define when a packet should be processed, iii) a detailed PF Ring filter for the application specific analysis they wish to apply to packets and last iv) a distribution mechanism to relate their data to the merger nodes.

## 4.3 Popularity plug-in

### 4.3.1 TCP protocol

In general COAST is considering TCP traffic for DPI. Thus the initial step is to get the data out of the TCP packet. Figure 24 shows the general structure of the TCP packet. Initially the DPI check the packet, extracts the Source and Destination address of the IP header, and checks deeper in the packet (e.g. the port number) to decide if that packet has to be inspected or not.

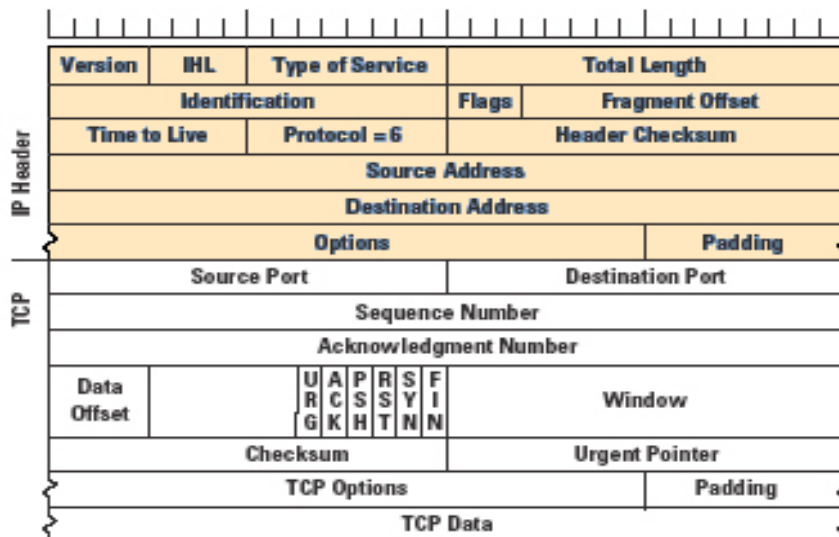


Figure 24: Structure of a typical TCP packet

Within COAST two protocols are considered: HTTP for normal web traffic and video over HTTP and RTSP for video streaming.

### 4.3.2 HTTP protocol

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. HTTP has been in use by the World-Wide Web global information initiative since 1990. The first version of HTTP, referred to as HTTP/0.9, was a simple protocol for raw data transfer across the Internet. HTTP/1.0 improved the protocol by allowing messages to be in the format of MIME-like messages, containing meta information about the data transferred and modifiers on the request/response semantics. However, HTTP/1.0 does not sufficiently take into consideration the effects of hierarchical proxies, caching, the need for persistent connections, or virtual hosts. In addition, the proliferation of incompletely-implemented applications calling themselves "HTTP/1.0" has necessitated a protocol version change in order for two communicating applications to determine each other's true capabilities. The HTTP/1.0 was revised in HTTP/1.1 [1]. HTTP /1.0 uses a separate connection to the same server for every request response transaction, while HTTP/1.1 can reuse the same connection multiple time, to download for instance images for just a delivered page. Figure 25 is a diagram illustrating the sequence of network activity of a typical web session.

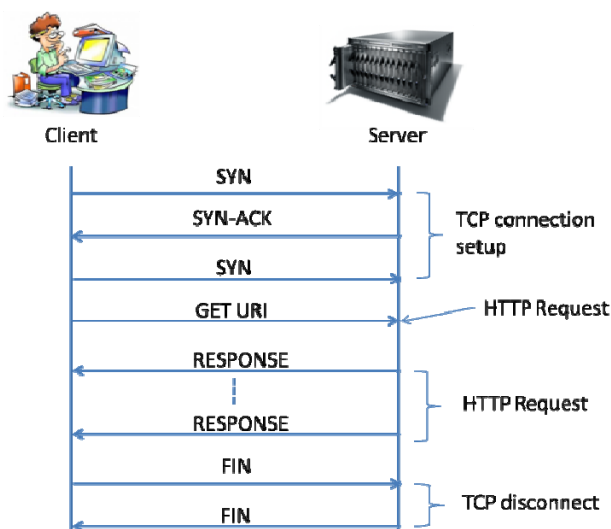


Figure 25: Network activity of a typical Web session



When a user visits a web site, the client device establishes a TCP connection with the server (TCP connection SETUP). TCP connection SETUP involves the exchange of three packet TCP packets labeled SYN, SYN ACK, and SYNC. Once the TCP connection has been established a request packet usually (GET URI) containing the GET request along with the desired URI is send from the client to the server. The GET URI packet is called HTTP request. An example of HTTP request header, when a client write on your web browser “http://www.bbc.com” is reported in Figure 26.

```
GET / HTTP/1.1[CRLF]
Host: www.bbc.com[CRLF]
Connection: close[CRLF]
User-Agent: Web-sniffer/1.0.37 (+http://web-sniffer.net/)[CRLF]
Accept-Encoding: gzip[CRLF]
Accept-Charset: ISO-8859-1,UTF-8;q=0.7,*;q=0.7[CRLF]
Cache-Control: no-cache[CRLF]
Accept-Language: de,en;q=0.7,en-us;q=0.3[CRLF]
Referer: http://web-sniffer.net/[CRLF]
```

Figure 26: HTTP request header

The server then responds with the requested web page (HTTP response). The HTTP response may be delivered to the client over a plurality of response packets, if the content of the Web page exceed the capacity of a single packet. An example of HTTP response header is reported in Figure 27.

```
Status: HTTP/1.1 200 OK
Server: Apache
Vary: Accept-Encoding
Cache-Control: max-age=60, private
Content-Type: text/html
Content-Encoding: gzip
Date: Thu, 03 Feb 2011
12:13:22 GMT
Keep-Alive: timeout=5, max=100
Transfer-Encoding: chunked
Etag: "1296735202"
Connection: close
```

Figure 27: HTTP response header

Finally the TCP session is terminated by the exchange of the FIN packets (TCP disconnected).

### 4.3.3 RTSP protocol

The Real Time Streaming Protocol (RTSP) is becoming the dominant control protocol for streaming content on the Internet. **Error! Reference source not found.** depicts a sample interaction between an RTSP client and server. RTSP is used to set up and control (pause, forward, etc.) the playback of streaming content across the Internet. RTSP is a classic request-response protocol, which also support for pipelining of messages in order to reduce latency. Protocol interaction starts with an OPTIONS request/ response whereby the client and server establish mutual capabilities.

The client then issues a DESCRIBE request for the media stream it is interested in. The response from the server contains media specific information about the stream, e.g. the encoding used, the clip length and the average bit rate. Depending on the particular session, more than one media stream might be described in a single DESCRIBE response message. After DESCRIBE, the client issues a SETUP request which contains the set of protocols and port numbers (or range of port numbers) on which the client is willing to receive the media stream. For RTSP this is normally UDP and a dynamically chosen port number, although it is also possible to use RTSP in “interleaved mode” where the data stream is interleaved on the original TCP control connection. This interleaving is typically only used to allow streaming through a firewall. The server selects one of these options and a port number and sends it back to the client in the response message. Following these



exchanges the client can issue a PLAY request to start the streaming and can issue PAUSE and other control requests for the stream. The session normally ends with a TEARDOWN request at which time the TCP connection is also terminated.

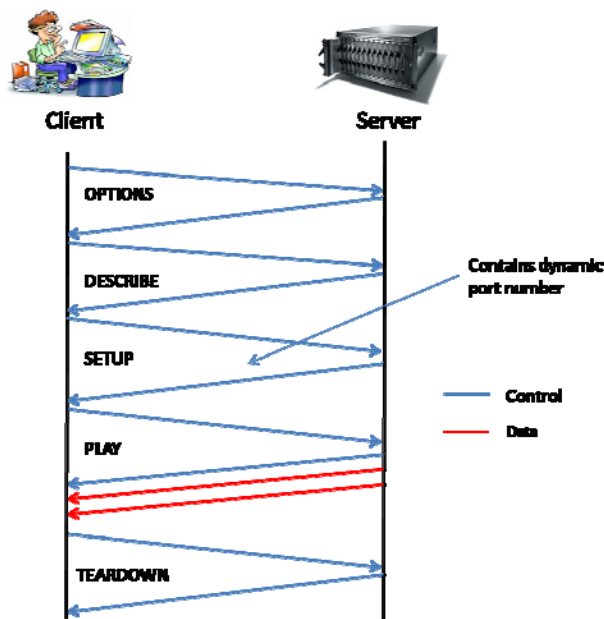


Figure 28: Dynamic port number assignment with RTSP

#### 4.4 Signalling and Named COAST Object Identifiers (COI)

COAST Deliverable D2.2 [34] has examined various possibilities of how COAST can identify its content and implement the routing functionality necessary to retrieve the content thus identified. In contrast to the approach used in today’s Internet which focuses on the endpoints, COAST focuses on the content itself. To be more concrete, in today’s Web content are identified by a URL / URI in the form of:

```

scheme://domain:port/path?query_string#fragment_id
    
```

The string of characters above organized to a valid URL can be used to identify any kind of resource in today’s Web ranging from concrete objects like web pages, pictures, videos or video segments up to answers to queries executed against a database or any kind of program output.

Typically the scheme is almost always “http” or “ftp” and the domain part then points to the host where the resource resides (or most precisely, where the server responsible for fetching the resource is listening). The remaining syntactical parts are then evaluated by that server and would typically point to a path in the file system or to a query to be executed against a back-stage database. But the important thing is that the identifier of a resource in the current web also identifies a physical location where the server providing that resource is hosted. Although the same domain can be used to map to different physical locations (e.g. by taking advantage of the layer of indirection inherent in the DNS resolution process – which is what Content Delivery Networks typically do), the fact remains that the physical location is a central component in the identification of every resource.

For reasons of practicality, backwards-compatibility and the sheer amount of effort and global coordination, between not just technical but also standardization and governmental bodies and industry players that would otherwise be required, it is simply not possible to implement the vision of a content-addressable or content-aware protocol in a clean-slate way without throwing away



almost the whole edifice of current Internet protocols, tools and user applications. The URL scheme identified above presupposes the domain name system mechanism to translate the domain component of the URL into an IP addresses<sup>2</sup> and then assumes that the User Agent application (i.e. the user's browser) will engage with the server thus identified (i.e. with the use of its IP address) using the HTTP protocol. Information flows between the user agent and the server will ultimately be carried out as the payload of IP packets which are of course routed according to their destination IP address. The web model is therefore inherently location-oriented mirroring and perhaps also owing to the location-based routing concept that's built in the IP protocol. The IP protocol is the thin waist of the Internet suite of protocols and any effort to implement a genuinely content aware network would necessitate substituting it or enhancing it with content functionality. Most likely it would also be necessary to replace the TCP layer and possibly additional layers as well as these protocols evolved or where defined with IP in mind and it might prove difficult to disentangle them (encapsulation notwithstanding). Such a clean-slate approach would require tremendous resources and would be doomed to failure and obsolescence unless the widest possible consensus was secured and powerful sponsor(s) were determined to provide the needed traction (these conditions being necessary but by no means adequate to ensure success). This clearly is not feasible for a STREP and, accordingly, COAST has taken the only reasonable approach which is to utilize an overlay over the existing Internet for the implementation of its content-addressable functionality.

A number of possible options for the architecture as well as for the COAST identifiers and methods to locate and retrieve COAST objects are presented in D2.2 [34]. D2.2 was an early architecture deliverable and, to an extent, the architecture it motivated has a number of degrees of freedom as frequently options are discussed and evaluated in D2.2 but no conclusive decision is enforced for the implementation. We talk in this paragraph of "architecture", although the section this text falls under is about the COAST object identifiers, because in a content-centric network the design and the semantics of the content identifiers carry important architectural ramifications and inform available implementation choices. To put it simply, to build a functioning network we don't just need a way to identify content, we also need a mechanism to retrieve that content based on this identification, and that requires a supporting infrastructure whose architecture has been defined with that particular identification scheme in mind.

Despite the various options identified there are however certain core themes in the content identification naming schemes, architecture options and functionalities discussed in D2.2. The remainder of this section provides a recapitulation of the most essential concepts defined in D2.2 and re-iterates the common architecture themes with the purpose of demonstrating how the proposed COI scheme and the architecture to support it are compliant with D2.2:

- **COAST object:** A COAST object is according to D2.2: "the fundamental content items of COAST [...] stored or indexed in the COAST network". According to D2.2 COAST objects consist of the object itself and its metadata. Since in this implementation iteration of the COAST concept we are going to implement COAST as an overlay over the present Internet, COAST objects correspond to "resources" as defined and understood in the context of the Web architecture (i.e. any entity that can be identified, named, addressed or handled).
- **COAST object identifier (COI):** this is the equivalent of the URI in the Web architecture. Exactly as a URI is a string of characters used to identify a resource on the Internet, a COI is a string of characters used to identify a COAST object in the COAST network.

---

<sup>2</sup> It is also possible to directly enter an IP address in the place of the domain segment in which case approaches that rely on DNS would cease to work. This serves to further highlight the location-bound character of today's web.



- **COAST node:** a computer either in the core network or at its edges providing functionality necessary for the retrieval of COAST objects as well as additional services (e.g. caching, replication)
- **COAST network:** the COAST network comprises of the COAST nodes and is understood as an overlay over today Internet. The COAST network uses its own mechanisms for routing and retrieval of COAST objects. More specifically, COAST employs a DHT-based cache overlay where content is eventually retrieved following the DHT routing algorithm. At the moment of this writing we are using a prototype implementation based on the PASTRY DHT algorithm ([13], [31]). COAST also differs from existing distributed web cache systems which feature automatic caching of content as a side-effect of accesses; instead, content in COAST has to be explicitly / intentionally published by the users. This allows for a clear business model.
- **COAST entry point (CEP):** the entry point into the COAST network. This is the point where the COAST overlay content retrieval functionality becomes triggered. Retrieval on the COAST overlay takes place over a Distributed Hash Table and is content-centric rather than location-centric. The concept of a COI is indispensable to content-centric retrieval. In contrast to approaches like that of CORAL that uses URLs suffixed with the “nyud.net” suffix (e.g. “<http://example.com.8080.nyud.net>”) to trigger its overlay and cache the contents, a COI uniquely identifies the content itself as opposed to its location. For more information and justification on the use of COIs please refer to COAST deliverable D2.2 [34], section 4.1 “metadata and naming”.

According to the above, the COAST network can be understood as a cloud in which content is published and then, after an arbitrary amount of time, retrieved. The COAST network needs to have the following properties:

- **Durable:** users explicitly publish content in COAST so they need to be given assurances that their content will be maintained indefinitely (according to the business model used) and will not be flushed out. COAST is not a distributed cache that can afford to lose content or one that provides a best-effort service. To guarantee durability, content in COAST will have to be transparently replicated.
- **Optimized:** ideally we would like COAST to be able to dynamically re-arrange content based on observed patterns of access so that it is physically closer to the users that require it. Metadata might also be conceivable exploited for the same purpose (e.g. during the initial deployment).
- **Dynamic.** The population of COAST nodes should be allowed to dynamically expand / shrink and the network should be able to handle this churn without losing the durability property.
- **Self-certified names.** The identifiers used to reference COAST objects should belong to a flat namespace and be self-certifying, i.e. they should allow the network or the user to validate that the content fetched indeed corresponds to the content asked. This allows the network to be robust in the presence of malicious nodes and can accordingly relax security requirements and credentials checking when it comes to nodes joining the network or to secure communications among the nodes.

Deliverable D2.2 [34] discussed, in its Section 6, certain content fetching scenarios according to the above. Following the above, the COI in COAST is identified as the SHA1 checksum of the raw data of the COAST object. The COI will be directly available in the URL as the final (and perhaps the only) component of the path segment of the URL (with the possible exception of file extension). Since the system will rely on CEP to trigger COAST functionality, we need to unambiguously signify in the URL



whether a particular URL ought to be served by COAST. This “flagging” can be done either in the path component of the URL as suggested in D2.2 giving URLs of the form:

`http://my.domain.net/COAST248036_<sha1 hashvalue>.<possible file extension>`

(in this case the “magic word” or rather “magic string” being: COAST248036)

... or, in the domain component itself, giving URLs of the form:

`http://www.coast.org/<sha1 hashvalue>.<possible file extension>`

There are therefore two possible canonical forms of a CURL (Coast URL):

```
http://<any domain>:port/<path prefix><COAST magic word><sha 1 digest>.<possible ext>
```

or

```
http://<coast domain>:port/<sha 1 digest>.<possible ext>
```

The first CURL version relies on the HTTP proxy to filter the HTTP GET request based on the presence of the <COAST magic word>. The possibility that the <COAST magic word> will appear by accident in a URL, followed by a string of characters that has the right length to look like a SHA1 digest is negligible but even in these cases the HTTP proxy can revert to normal handling of the URL, once the COAST overlay lookup fails. This CURL version also has the advantage that the “original” domain where the content was published is present in the CURL and may be usable if the lookup fails.

The second CURL version is neater and uses a special <coast domain>. Filtering is done on that basis. This second version is more aligned with a publishing scenario where content is explicitly published in the COAST overlay alone and no other “primordial” legacy web site is used.

We are currently evaluating which approach to support (although both could be conceivably implemented).

Since the SHA1 algorithm generates a message digest of 160 bits, this can be represented by 40 ASCII characters in the range: ('0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'a', 'b', 'c', 'd', 'e', 'f') corresponding to the representation of the digest (each of the 20 bytes being represented as two hexadecimal digits).

According to the above discussion the below are all possible, valid URLs which embed a COI. Passing through a CEP they should trigger COAST functionality and eventual retrieval of the COAST object they identify from the COAST overlay network:

[http://my.domain.net/COAST248036\\_37a8d10c39e921da187c6929adcb0bfa59cf2192.mp3](http://my.domain.net/COAST248036_37a8d10c39e921da187c6929adcb0bfa59cf2192.mp3)

[http://my.domain.net/COAST948036\\_37a8d10c39e921ff187c6929adcb0bfa60cf2352.avi](http://my.domain.net/COAST948036_37a8d10c39e921ff187c6929adcb0bfa60cf2352.avi)

[http://my.domain.net/COAST248036\\_84d53e26a9976fc3250ba5cc462e7bb5f7862780.html](http://my.domain.net/COAST248036_84d53e26a9976fc3250ba5cc462e7bb5f7862780.html)

[http://my.domain.net/COAST248036\\_b8c5b5068e6ed3fe30ffe491f2890d42977246fa.jpg](http://my.domain.net/COAST248036_b8c5b5068e6ed3fe30ffe491f2890d42977246fa.jpg)

<http://www.coast.org/37a8d10c39e921da187c6929adcb0bfa59cf2192.mp3>

<http://www.coast.org/37a8d10c39e921ff187c6929adcb0bfa60cf2352.html>

<http://www.coast.org/84d53e26a9976fc3250ba5cc462e7bb5f7862780>

In all the above cases, the COAST content will be eventually reached by means of a modified web proxy which will act as the CEP. The modified proxy will detect that the above URLs are CURLs and will fetch the content from the COAST overlay network rather than from a specific physical location.



## 5 Conclusions

As stated in summary of, this document has presented a discussion of the function and requirements on DPI architecture aimed at supporting and aiding towards realising content aware networking.

Specifically, this document presents a discussion on the requirements and function on DPI with respect to the stated goal. Chapter 2 has presented a review of the current profile of traffic on the network from different vantage points. The dominance of HTTP as the main component of user traffic is shown and the results of recent publications which highlight the popularity of HTTP over other protocols as the main carrier for multimedia traffic are presented. We also the arguments that equate and correlate this growth with the rise of social networking and the popularity of media streaming, we postulate both legal such as YouTube, Hulu and the BBC iPlayer and the those in the grey area such as the services offered by tudou.com and megaupload.com. All in all, these numbers support our motivation to initially focus the DPI work on HTTP.

Chapter 3 has presented an overview of different DPI techniques highlighting their function, strengths and weaknesses, we as well as their representative application domain. More specifically related our problem, Chapter 4 presents the high-level design of our DPI node which aims to realise flexible and modular design. Moreover, the architecture we present supports the development of both stateless and state-full DPI (via the definition of the protocol and state awareness block.

This flexibility plus the integration of PF Ring means that we are able to support high performance DPI, further our modular decomposition means that we can look at scaling by distributing the cost of DPI.

With specific regard to the problem of content centric networking, we have presented the requirements and the high-level architecture providing the kind of information that will be required in order to identity content of interest, COAST Object Identifiers (COIs) may be built on the results of the DPI process.

In future work, we will realise the implementation of the architecture presented in this document, and provide the mechanism to support experimentation with various DPI algorithms subject to the problems that arise. For now, we for see the load/DPI throughput issue as the key component, since fundamentally, our task is to capture all the packets passing through a monitoring point.



## 6 References

- [1] COAST D2.1 “Service Requirement Specification,” COAST consortium, May 2010, [http://www.coast-fp7.eu/public/COAST\\_D2.1\\_TID\\_FF\\_20100514.pdf](http://www.coast-fp7.eu/public/COAST_D2.1_TID_FF_20100514.pdf)
- [2] L. Deri, “Improving Passive Packet Capture: Beyond Device Polling”, in System Administration and Network Engineering Conference (SANE), 2004.
- [3] F. Fusco, F. Huici, L. Deri, S. Niccolini, T. Ewald, “Enabling High-Speed and Extensible Real-Time Communications Monitoring”, in IEEE Integrated Management (IM), 2009
- [4] Packet dispersion technique tool CapProbe: <http://www.cs.ucla.edu/NRL/CapProbe/>
- [5] <http://www.ccnx.org/content/content-centric-networking-resources>
- [6] A Data-Oriented (and Beyond) Network Architecture, SIGCOMM’07, August 27–31, 2007, Kyoto, Japan
- [7] Van Jacobson et al., “Content-Centric Networking, Whitepaper Describing Future Assurable Global Networks”, In response to DARPA Assurable Global Networking, RFI SN07-12. 2007
- [8] “The CORAL Content Distribution Network,” <http://www.coralcdn.org/overview/>
- [9] Cambazoglu, B. B., Plachouras, V., Junqueira, F., and Telloli, L. On the feasibility of geographically distributed web crawling. In Proceedings of the 3rd International Conference on Scalable Information Systems, pp. 1-10, VicoEquense, Italy, 2008.
- [10] Baeza-Yates, R., Gionis, A., Junqueira, F., Plachouras, V., and Telloli, L. On the feasibility of multi-site web search engines. In Proceeding of the 18th ACM Conference on Information and Knowledge Management, pp. 425-434, Hong Kong, China, 2009.
- [11] Cambazoglu, B. B., Varol, E., Kayaaslan, E., Aykanat, C., and Baeza-Yates, R. Query forwarding in geographically distributed search engines. In Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 90-97, Geneva, Switzerland, 2010.
- [12] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, Scott Shenker, “A Scalable Content-Addressable Network,” SIGCOMM’01, San Diego, California, USA, August 27-31, 2001, pp. 161-172
- [13] Jihong Song, Shaopeng Wang “The Pastry Algorithm Based on DHT,” Computer and Information Science Journal, Vol. 2, No. 4, November 2009, pp.153-157
- [14] Yi Jiang, Jinyuan You, “Tapestry: A Low Latency Chord Routing Algorithm for DHT,” 1<sup>st</sup> International Symposium on Pervasive Computing and Applications, 3-5 Aug. 2006, pp.825 – 830
- [15] <http://en.wikipedia.org/wiki/REST>
- [16] Richardson, L., Ruby, S.: RESTful Web Services. O’ Reilly, Cambridge (2007)
- [17] JavaScript Object Notation: <http://www.json.org/>
- [18] D. Mazieres, M. Kaminsky, M. F. Kaashoek, and E. Witchel, “Separating Key Management from File System Security” In Proc. of SOSP ’99, pages 124–139, Charleston, SC, USA, Dec. 1999.
- [19] M. Gritter, D. R. Cheriton “TRIAD: A New Next-Generation Internet Architecture,” <http://www-dsg.stanford.edu/triad>, July 2000.
- [20] R. Moskowitz and P. Nikander, “Host Identity Protocol Architecture”, RFC 4423, IETF, May 2006.



- [21] B. Ford, J. Strauss, C. Lesniewski-Laas, S. Rhea, F. Kaashoek, and R. Morris, "Persistent Personal Names for Globally Connected Mobile Devices" In Proc. of OSDI 2006, pages 233–248, Seattle, WA, USA, Nov. 2006.
- [22] H. Balakrishnan, K. Lakshminarayanan, S. Ratnasamy, S. Shenker, I. Stoica, and M. Walfish. A Layered Naming Architecture for the Internet. In Proc. of ACM SIGCOMM '04, pages 343–352, Portland, OR, USA, Aug. 2004.
- [23] Adaptive streaming comparison: <https://sites.google.com/a/quavlive.com/quavlive/adaptive-streaming>
- [24] The MPEG home page: [http://mpeg.chiariglione.org/hot\\_news.htm](http://mpeg.chiariglione.org/hot_news.htm)
- [25] The 3GPP mobile broadband standard: [www.3gpp.org](http://www.3gpp.org)
- [26] 3GPP adaptive HTTP streaming: <https://labs.ericsson.com/apis/streaming-media/documentation/3gpp-adaptive-http-streaming-ahs>
- [27] IST-SEA project: <http://www.ist-sea.eu>
- [28] P2P-NEXT project: <http://www.p2pnext.org>, <http://trial.p2p-next.org>, <http://www.livinglab.eu>
- [29] Ratnasamy et al. (2001). A Scalable Content-Addressable Network (CAN). In Proceedings of ACM SIGCOMM 2001. Retrieved 2008-12-22.
- [30] Stoica, Ion et al. (2001). "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications". Proceedings of SIGCOMM'01 (ACM Press New York, NY, USA).
- [31] A. Rowstron and P. Druschel (Nov 2001). Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany: 329–350.
- [32] Zhao, B. Y. et al. Tapestry: A Resilient Global-Scale Overlay for Service Deployment, IEEE Journal On Selected Areas In Communications, VOL. 22, No. 1, January 2004, pg. 41.
- [33] VideoLAN Free Multimedia Solutions: [www.videolan.org](http://www.videolan.org)
- [34] gstreamer Open Source Multimedia Framework: [www.gstreamer.net](http://www.gstreamer.net)
- [35] COAST Deliverable D2.2 "End-to-End Future Content Network Specification"
- [36] Minnesota Internet Traffic Study (MINTS): <http://www.dtc.umn.edu/mints/igrowth.html>
- [37] GARR - The Italian Academic & Research Network: <http://www.garr.it/>
- [38] CAIDA: <http://www.caida.org/research/traffic-analysis/>
- [39] IPOQUE: [http://www.ipoque.com/resources/internet-studies/internet-study-2008\\_2009](http://www.ipoque.com/resources/internet-studies/internet-study-2008_2009)
- [40] PRX traffic manager: <http://www.ipoque.com/products/prx-traffic-manager>
- [41] Jeffrey Erman, Alexandre Gerber, Mohammad Hajiaghayi, Dan Pei, Subhabrata Sen, Oliver Spatscheck, To Cache or not to Cache: The 3G case, IEEE Internet Computing, 20 Dec. 2010. IEEE computer Society Digital Library. IEEE Computer Society.
- [42] Jeffrey Erman, Alexandre Gerber, Mohammad T. Hajiaghayi, Dan Pei, and Oliver Spatscheck. 2009. Network-aware forward caching. In *Proceedings of the 18th international conference on World wide web (WWW '09)*. ACM, New York, NY, USA, 291-300.
- [43] Dorado et.al What users do in the Internet, Jose Garcia/Dorado, Alessandro Finamore, Marco Mellia, Michela Meo, Maurizio Munafo. What do users do with the Internet? Technical Report Politecnico di Torino TR-012811-polito, No.TR-012811, 2011.
- [44] C.Labovitz, S. Iekel-Johnson, J.Oberheide and F. Jahanian, Internet inter-domain traffic, SIGCOMM'10, 2010.
- [45] Application layer packet classifier for linux: <http://l7-filter.sourceforge.net>



- [46] SNORT project: <http://www.snort.org>
- [47] IANA: TCP and UDP port numbers. <http://www.iana.org/assignments/port-numbers>
- [48] Yan Sun, Haiqin Liu, Valgenti, V.C. , Min Sik Kim, Hybrid regular expression matching for deep packet inspection on multi-core architecture, Proceedings of 19th International Conference on Computer Communications and Networks (ICCCN), 2-5 Aug. 2010.
- [49] M. Becchi and P. Crowley, A hybrid finite automaton for practical deep packet inspection, in Proceedings of ACM CoNEXT, Dec. 2007.
- [50] D. Ficara, S. Giordano, G. Procissi, F. Vitucci, G. Antichi, and A. Di Pietro, An improved DFA for fast regular expression matching, Computer Communication Review, vol. 38, no. 5, pp. 29-40, 2008.
- [51] S. Kumar, S. Dharmapurikar, F. Yu, P. Crowley, and J. Turner. Algorithms to accelerate multiple regular expressions matching for deep packet inspection. In Proc. of SIGCOMM '06, pages 339-350. ACM.
- [52] A. V. Aho, R. Sethi, and J. D. Ullman. Compilers, principles, techniques, and tools. Addison Wesley, 1985
- [53] M. Becchi and P. Crowley, An improved algorithm to accelerate regular expression evaluation, in Proceedings of the 3rd ACM/IEEE Symposium on Architecture for networking and communications systems, 2007, pp. 145-154.
- [54] M. Becchi and S. Cadambi, Memory-efficient regular expression search using state merging, in Proceedings of the 26th IEEE International Conference on Computer Communications, May 2007, pp. 29-40.
- [55] M. Becchi, C. Wiseman, and P. Crowley, Evaluating regular expression matching engines on network and general purpose processors, in Proceedings of the 2009 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), Oct. 2009.
- [56] F. Baboescu and G. Varghese, Scalable Packet Classification, *IEEE/ACM Transactions on Networking*, pp. 2.14, Feb 2005.
- [57] T. V. Lakshman and D. Stidialis, High Speed Policy-Based Packet Forwarding Using Efficient Multi-Dimensional Range Matching, In *ACM Sigcomm*, Sept 1998.
- [58] N. Tuck, T. Sherwood, B. Calder, and G. Varghese, Deterministic Memory Efficient String Matching Algorithms for Intrusion Detection, in *IEEE Infocom*, pp. 333.340, Mar 2004.
- [59] S. Kumar, B. Chandrasekaran, J. Turner, and G. Varghese. Curing regular expressions matching algorithms from insomnia, amnesia, and acalculia. In Proc. of ANCS '07, pages 155-164. ACM.
- [60] R. Smith, C. Estan, and S. Jha. Xfa: Faster signature matching with extended automata. In IEEE Symposium on Security and Privacy, May 2008.
- [61] R. Smith, C. Estan, and S. Jha. Xfas: Fast and compact signature matching. Technical report, University of Wisconsin, Madison, August 2007.
- [62] M. Becchi, C. Wiseman and P. Crowley, Evaluating regular expression matching engines on network and general purpose processor, Proceedings of the 5th ACM/IEEE Symposium on Architectures for Networking and Communications Systems, 2009.
- [63] M. Becchi and P. Crowley, Efficient Regular Expression Evaluation: Theory to Practice, in ACM/IEEE ANCS 2008.
- [64] M. Becchi and P. Crowley, A Hybrid Finite Automaton for Practical Deep Packet Inspection, in ACM CoNEXT 2007.
- [65] Tingwen Liu, Young sun, Li Guo, Fast and memory efficient traffic classification with deep packet inspection in CMP architecture, IEEE International conference on Networking, Architecture and Storage (NAS), 2010.



- [66] Apparatus for storing “Don’t Care” in a content addressable memory cell. United States Patent 5,319,590. HaL ComputerSystems, Inc.
- [67] Srinivasan, V., Suri, S., Varghese, G., and Waldvogel, Fast and scalable layer four switching. In *ACM Sigcomm*.
- [68] Baboescu, F., Singh, S., and Varghese, Packet classification for core routers: Is there an alternative to CAMs? In *IEEE Infocom*.
- [69] Gupta, P. and Mackown, Packet classification using hierarchical intelligent cuttings. In *Hot Interconnects VII*.
- [70] Singh, S., Baboescu, F., Varghese, G., and Wang, Packet classification using multidimensional cutting. In *Proceedings of ACM Sigcomm*. Karlsruhe, Germany.
- [71] Feldmann, A. and Muthukrishnan, Tradeoffs for Packet Classification. In *IEEE Infocom*.
- [72] Lakshman, T. V. and Stiliadis, High-speed policy-based packet forwarding using efficient multi-dimensional range matching. In *ACM Sigcomm*.
- [73] F. and Varghese, Scalable packet classification. In *ACM Sigcomm*.
- [74] Gupta, P. and Mckeown, Packet classification on multiple fields. In *ACM Sigcomm*.
- [75] Taylor, D. E. and Turner, Scalable packet classification using distributed crossproducting of field labels. In *Proceedings of IEEE Infocom*.
- [76] Van Lunteren, J. and Engbersen, Fast and scalable packet classification. *IEEE J. Select. Areas Comm.* 21, 4 (May) 560–571.
- [77] V. Srinivasan, S. Suri, and G. Varghese. “Packet Classification using Tuple Space Search”, *Proceedings of ACM Sigcomm*, pages 135-46, September 1999.
- [78] D. Liu, B. Hua, X. Hu and X. Tang, “Highperformance Packet Classification Algorithm for Many-core and Multithreaded Network Processor”, Proc. of the 6th IEEE International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES), 2006.
- [79] D. Srinivasan and W. Feng, “Performance Analysis of Multi-dimensional Packet Classification on Programmable Network Processors”, Proc. of the 29<sup>th</sup> Annual IEEE International Conference on LocalComputer Networks (LCN), 2004.
- [80] D. Liu, B. Hua, X. Hu and X. Tang, “High performance Packet Classification Algorithm for Many-core and Multi threaded Network Processor”, Proc. of the 6th IEEE International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES), 2006.
- [81] Bo Xu, Yaxuan Qi, Fei He, Zongwei Zhou, Yibo Xue, and Jun Li, Fast Path Session Creation on Network Processors, The 28th International Conference on Distributed Computing Systems, 2008.
- [82] B. Xu, D. Jiang and J. Li, HSM: A Fast Packet Classification Algorithm, Proc. of the 19th International Conference on Advanced Information Networking and Applications (AINA), 2005.
- [83] Yaxuan Qi, Bo Xu, Fei He, Baohua Yang, Jianming Yu, Jun Li, Towards High-performance Flow level Packet Processing on Multi-core Network Processors, Proceedings Of the 3<sup>rd</sup> ACM/IEEE Symposium on architecture for networking and communications system, 2007
- [84] <http://www.arl.wustl.edu/~hs1/PClassEval.html>
- [85] Baohua Yang, Xiang Wang, Yibo Xue and Jun Li, DBS: A Bit-level Heuristic Packet Classification Algorithm for High Speed Network, 15th International Conference on Parallel and Distributed Systems, 2009
- [86] “OCTEON™ Plus CN58XX Multi-Core MIPS64 Based SoC Processors,” 2009; [http://www.caviumnetworks.com/OCTEONPlus\\_CN58XX.html](http://www.caviumnetworks.com/OCTEONPlus_CN58XX.html)



- [87] D.E. Taylor and J.S. Turner, "ClassBench: A packet classification benchmark," *IEEE/ACM Transactions on Networking*, vol. 15, no. 3, 2007, pp. 499-511
- [88] A. McGregor, M. Hall, P. Lorier, and J. Brunskill, Flow clustering using machine learning techniques, *Lecture Notes in Computer Science*, vol. 3015, pp. 205-214, 2004
- [89] A. W. Moore and D. Zuev, Internet traffic classification using bayesian analysis techniques, *ACM SIGMETRICS Performance Evaluation Review*, vol. 33, pp. 50-60, 2005
- [90] T. Nguyen and G. Armitage, Synthetic sub-flow pairs for timely an stable IP traffic identification, 2006
- [91] Dehghani, F.; Movahhedinia, N.; Khayyambashi, M.R.; Kianian, S.; Real time traffic classification based on statistical and payload content features, Workshop on Intelligent Systems and Applications (ISA), 2010 2nd International
- [92] TippingPoint x505: <http://www.tippingpoint.com/products ips.html>
- [93] Cisco IOS IPS signature deployment guide: <http://www.cisco.com/>
- [94] I. Bonesana, M. Paolieri, and M. Santambrogio, An adaptable FPGAbased system for regular expression matching, in *Proceedings of the conference on Design, Automation and Test in Europe*, Mar. 2008, pp. 1262–1267
- [95] N. Yamagaki, R. Sidhu, and S. Kamiya, High-speed regular expression matching engine using multi-character NFA, in *International Conference on Field Programmable Logic and Applications*, Sep. 2008, pp. 131–136
- [96] C. R. Clark and D. E. Schimmel, Efficient reconfigurable logic circuit for matching complex network intrusion detection patterns, in *Conference on field-programmable logic and applications*, Sep. 2003, pp. 956–959
- [97] A. Mitra, W. Najjar, and L. Bhuyan, Compiling PCRE to FPGA for accelerating SNORT IDS, in *Proceedings of the 3rd ACM/IEEE Symposium on Architecture for networking and communications systems*, 2007, pp. 127–136.
- [98] Intel IXP1200 network processor: <http://int.xscale-freak.com/XSDoc/IXP12xx/27829810.pdf>
- [99] Intel IXP2800 network processor: <http://int.xscale-freak.com/XSDoc/IXP2xxx/27889504.pdf>



## 7 Appendix A – EU Legal framework for data protection, privacy and the COAST project

The individual's right of privacy was established with Article 12 of the Universal Declaration of Human Rights<sup>3</sup> in 1948 and with Article 8 of the European Convention for Protection of Human Rights and Fundamental Freedoms<sup>4</sup> in 1950. The obvious danger that this human right could be threatened by the emerging electronic data processing was mitigated by specific regulations and agreements for this sector. In 1960s and 70s there was a broad debate in several European states on the use of electronic data processing by public authorities that led to the initial national data-protection laws in the late 1970s. In 1981 the Convention No. 108 for the Protection of Individuals with regard to Automated Processing of Personal Data<sup>5</sup> was signed by the member States of the Council of Europe. The Organization for Economic Co-Operation and Development (OECD) developed a set of Guidelines on the Protection of Privacy and Transborder Flows of Personal Data<sup>6</sup> in 1980. Thus the principal elements for the protection of privacy have been created. These instruments set out specific rules covering the handling of information and have had a profound effect on the enactment of laws around the world. They describe personal information as data that is afforded protection at every step from collection to storage and dissemination.

In 1995 the EU adopted Directive 95/46/EC on the Protection of Individuals with regard to the Processing of Personal Data and on the Free Movement of such Data<sup>7</sup> in order to harmonize Member States' laws in providing consistent levels of protections for citizens and ensuring the free flow of personal data within the European Union. It established the basic principles for the collection, storage, and use of personal data that should be respected by governments, businesses and any other organizations or individuals engaged in handling personal data. This Directive (Data Protection Directive) is the reference text, at European level, on the protection of personal data. It sets up a regulatory framework which seeks to strike a balance between a high level of protection for the privacy of individuals and the free movement of personal data within the EU. To do so, the Directive sets strict limits on the collection and use of personal data and demands that each Member State set up an independent national body responsible for the protection of these data.

Due to the rapid technical progress of information and communication technologies and the resulting new ways of generating and analyzing personal data the existing regulations were increasingly regarded as insufficient. In 1997 the European Union supplemented the 95/46/EC Directive by introducing the Telecommunications Privacy Directive 97/66/EC concerning the

---

<sup>3</sup> Universal Declaration of Human Rights, G.A. res. 217A (III), U.N. Doc A/810 at 71 (1948)

<sup>4</sup> European Convention for the Protection of Human Rights and Fundamental Freedoms, ETS 5; available from: <http://conventions.coe.int/Treaty/Commun/QueVoulezVous.asp?NT=005&CM=8&DF=4/25/2006&CL=ENG>

<sup>5</sup> European Convention for the Protection of Individuals with regard to Automated Processing of Personal Data, CETS.108; available from: <http://conventions.coe.int/Treaty/Commun/QueVoulezVous.asp?NT=108&CM=8&DF=22/12/2010&CL=ENG>

<sup>6</sup> OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data, available from: [http://www.oecd.org/document/18/0,3746,en\\_2649\\_34255\\_1815186\\_1\\_1\\_1\\_1,00.html](http://www.oecd.org/document/18/0,3746,en_2649_34255_1815186_1_1_1_1,00.html)

<sup>7</sup> Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data, OJ L 281, 23.11.1995, p. 31–50



Processing of Personal Data and the Protection of Privacy in the Telecommunications Sector<sup>8</sup>. This directive established specific protections covering telephone, digital television, mobile networks and other telecommunications systems. New advanced digital technologies have been recently introduced in public communications networks in the Community, which give rise to specific requirements concerning the protection of personal data and privacy of the user. Directive 97/66/EC has been repealed and replaced by the Directive 2002/58/EC concerning the Processing of Personal Data and the Protection of Privacy in the Electronic Communications Sector<sup>9</sup> (Privacy and Electronic Communications Directive).

This Directive was adopted in 2002 and forms part of a new legislative framework for electronic communications networks and services, the main aim of which is to make the electronic communications sector more competitive. This framework consists of one framework Directive and four specific Directives, namely the: “Framework Directive”<sup>10</sup>; “Authorization Directive”<sup>11</sup>; “Access Directive”<sup>12</sup>; “Universal Service Directive”<sup>13</sup>; “Privacy and Electronic Communications Directive”.

Directive 2002/58 on Privacy and Electronic Communications, otherwise known as E-Privacy Directive, is an EU directive on data protection and privacy in the digital age. It has been drafted specifically to address the requirements of new digital technologies and ease the advance of electronic communications services. It complements the Data Protection Directive 95/46/EC and applies to all matters which are not specifically covered by that Directive.

In particular the subject of the Directive is stated in Article 1: “This Directive harmonizes the provisions of the Member States required to ensure an equivalent level of protection of fundamental rights and freedoms, and in particular the right to privacy, with respect to the processing of personal data in the electronic communication sector and to ensure the free movement of such data and of electronic communication equipment and services in the Community.” Contrary to Data Protection Directive, which specifically addresses only individuals, Article 1(2) makes it clear that E-Privacy Directive also applies to legal persons. It includes provisions on security of networks and services, confidentiality of communications, access to information stored on terminal equipment, processing of traffic and location data, calling line identification, public subscriber directories and unsolicited commercial communications.

The Directive 2002/58/EC has been amended by the Directive 2009/136/EC amending Directive 2002/22/EC on universal service and users’ rights relating to electronic communications networks and services, Directive 2002/58/EC concerning the processing of personal data and the protection of

---

<sup>8</sup> Directive 97/66/EC of the European Parliament and of the Council of 15 December 1997 concerning the processing of personal data and the protection of privacy in the telecommunications sector, OJ L 24, 30.1.1998, p. 1–8

<sup>9</sup> Directive 2002/58/EC of the European Parliament and of the Council of 12 July 2002 concerning the processing of personal data and the protection of privacy in the electronic communications sector (Directive on privacy and electronic communications), OJ L 201, 31.7.2002, p. 37–47

<sup>10</sup> Directive 2002/21/EC of the European Parliament and of the Council of 7 March 2002 on a common regulatory framework for electronic communications networks and services, OJ L 108, 24.4.2002, p. 33–50

<sup>11</sup> Directive 2002/20/EC of the European Parliament and of the Council of 7 March 2002 on the authorization of electronic communications networks and services, OJ L 108, 24.4.2002, p. 21–32

<sup>12</sup> Directive 2002/19/EC of the European Parliament and of the Council of 7 March 2002 on access to, and interconnection of, electronic communications networks and associated facilities, OJ L 108, 24.4.2002, p. 7–20

<sup>13</sup> Directive 2002/22/EC of the European Parliament and of the Council of 7 March 2002 on universal service and user’s rights relating to electronic communications networks and services, OJ L 108, 24.4.2002, p. 51–77



privacy in the electronic communications sector<sup>14</sup>. This Directive forms part of the telecoms reform package that also includes Regulation (EC) No 1211/2009<sup>15</sup>, and Directive 2009/140/EC<sup>16</sup> and went into force with its publication in the EU's Official Journal (18 December 2009). Its transposition into national legislation in the 27 EU Member States must be ensured by June 2011. The above mentioned amendments are out of the scope of this document, as they have not yet been implemented into national law by Member States.

The European Commission will propose in 2011 a new general legal framework for the protection of personal data in the EU covering data processing operations in all sectors and policies of the EU<sup>17</sup>. COAST will be continuously monitoring any changes in this field, and will adapt accordingly its practices.

## 7.1 National legislations

The Data Protection Directive as well as the e- Privacy Directive has been transposed to the national legislation of the EU countries. The table below denotes the correspondence between those Directives and National Laws implementing them for the European countries represented in the COAST consortium.

	Data Protection Directive 95/46/EC	e- Privacy Directive 2002/58/EC
Italy	Data Protection Code- Legislative Decree No. 196 of 30 June 2003)	Data Protection Code- Legislative Decree No. 196 of 30 June 2003)
Greece	Data Protection Law 2472/1997 on the Protection of Individuals with regard to the Processing of Personal Data	Law 3471/2006, on the protection of personal data and privacy in the electronic telecommunications sector and amendment of law 2472/1997.  Law 3783/2009 amending law 3471/2006.
Spain	Organic Law 15/1999 relating to Personal Data Protection  Royal Decree 1720/2007 (“ <b>Data Protection Regulations</b> ”)	General Telecommunications Law 32/2003  Royal Decree 424/2005, as amended by Royal Decree 1768/2007.

<sup>14</sup> Directive 2009/136/EC of the European Parliament and of the Council of 25 November 2009 amending Directive 2002/22/EC on universal service and users’ rights relating to electronic communications networks and services, Directive 2002/58/EC concerning the processing of personal data and the protection of privacy in the electronic communications sector and Regulation (EC) No 2006/2004 on cooperation between national authorities responsible for the enforcement of consumer protection laws (Text with EEA relevance), OJ L 337, 18.12.2009, p. 11–36

<sup>15</sup> Regulation (EC) No 1211/2009 of the European Parliament and of the Council of 25 November 2009 establishing the Body of European Regulators for Electronic Communications (BEREC) and the Office (Text with EEA relevance), OJ L 337, 18.12.2009, p. 1–10

<sup>16</sup> Directive 2009/140/EC of the European Parliament and of the Council of 25 November 2009 amending Directives 2002/21/EC on a common regulatory framework for electronic communications networks and services, 2002/19/EC on access to, and interconnection of, electronic communications networks and associated facilities, and 2002/20/EC on the authorization of electronic communications networks and services (Text with EEA relevance), OJ L 337, 18.12.2009, p. 37–69

<sup>17</sup> <http://europa.eu/rapid/pressReleasesAction.do?reference=MEMO/10/542>



UK	Data Protection Act 1998	Privacy and Electronic Communications (EC Directive) Regulations 2003 (SI 2003 No. 2426)  Article 5.1 : the Regulation of Investigatory Powers Act 2000  Article 5.2 : the Telecommunications (Lawful Business Practice) (Interception of Communications) Regulations 2000 (SI 2000 No 2699)
Germany	Federal Data Protection Law ( <i>Bundesdatenschutzgesetz</i> ) 2001, as amended in 2009 by the Federal Data Protection Amendment Law ( <i>Novelle des Bundesdatenschutzgesetzes</i> )	Federal Telecommunications Law (TKG)  Federal Law against Unfair Competition (UWG) 2004

**Table 13: Implementation of the EU Directives into national legislation**

### 7.1.1 The specific case of the UK

It is worth mentioning that the data protection regime in the UK remains very weak overall in comparison to other European countries. It should also be noted that the 1998 Act still fails to fully implement the requirements of the main Data Protection Directive in many ways. The Commission launched legal action against the UK in April 2009 following citizens' complaints about how the UK authorities had dealt with their concerns about the use of behavioural advertising by internet service providers (targeted advertising based on prior analysis of users' internet traffic)<sup>18</sup>.

The Commission considers that existing UK law governing the confidentiality of electronic communications is in breach of the UK's obligations under the ePrivacy Directive 2002/58/EC and the Data Protection Directive 95/46/EC in three specific areas:

- there is no independent national authority to supervise the interception of some communications, although the establishment of such authority is required under the ePrivacy and Data Protection Directives, in particular to hear complaints regarding interception of communications
- UK law authorizes interception of communications not only where the persons concerned have consented to interception but also when the person intercepting the communications has 'reasonable grounds for believing' that consent to do so has been given. These UK provisions do not comply with EU rules defining consent as "freely given, specific and informed indication of a person's wishes"
- UK law prohibiting and providing sanctions in case of unlawful interception are limited to 'intentional' interception only, whereas EU law requires Members States to prohibit and to ensure sanctions against any unlawful interception regardless of whether committed intentionally or not.

<sup>18</sup> <http://europa.eu/rapid/pressReleasesAction.do?reference=IP/09/570>



### **7.1.2 The specific case of Germany**

Data protection in Germany is complicated by the fact that it is dealt with in both Federal and State laws: there are therefore, in Germany, no less than 16 general data protection laws (the Federal Law and one for each of the 15 States), all of which had to be brought in line with the Framework Directive. Supervision and enforcement is similarly spread over a range of different authorities. In addition, there are numerous other laws that either deal with data protection in a specific context, or that otherwise contain data protection (or related) rules. Overall, there are literally hundreds of different federal and State laws and regulations. The overall legal situation is therefore very complex. However, as far as both federal public authorities and the private sector are concerned, the Federal Law is the main instrument.

German data protection law is strong, and has a very solid base in the Constitution. At the same time, it is also extremely detailed and technical. The Law reflects the data protection principles, set out in Art. 6(1) of the Directive 95/46, but somewhat indirectly, or in different forms. In particular, it applies these principles differently to the public and private sectors, which are regulated in separate parts of the Federal Data Protection Law. The data protection criteria (i.e., the conditions on the basis of at least one of which processing of personal data must be based, if the processing is to be "legitimate"), set out in Art. 7 of the Directive are also reflected in the Law, but again in a rather particular and complex way, and with differences between the public and private sector. It is worthy to note that the Law is applied most strictly to the public sector, and in a markedly more relaxed way to the private sector. Moreover, the enforcement system is not always as strong as one would expect, especially in the private sector.

### **7.1.3 The specific case of the USA**

The situation in the USA is rather complex. The USA has no comprehensive data protection legislation. Although a signatory to the 1981 OECD Guidelines, the USA has not implemented them domestically. Instead, a sectoral approach, with a mix of legislation, regulation and self-regulation, is utilized. Privacy legislation in the United States tends to be adopted on an ad hoc basis, with legislation arising when certain sectors and circumstances require (e.g., the Video Privacy Protection Act of 1988, the Cable Television Protection and Competition Act of 1992, the Fair Credit Reporting Act, and the 2010 Massachusetts Data Privacy Regulations). Therefore, while certain sectors may already satisfy the EU Directive, at least in part, most do not. The introduction of Directive 95/46/EC could have therefore restricted the ability of US organizations to engage in transactions with their European counterparts, for it prohibited the transfer of personal data to non EU states that do not meet the "adequacy" standard for the protection of privacy. As a result of this, the US Department of Commerce developed the "safe harbour" system in consultation with the European Commission. This offers a method by which US organizations can comply with the Directive. The EU approved "Safe harbour"<sup>19</sup> in July 2000. Organizations that sign up to the scheme are certified as offering 'adequate' protection under the terms of the Directive, thus enabling transactions between those organizations and European organizations to proceed smoothly and within the law.

---

<sup>19</sup>2000/520/EC: Commission Decision of 26 July 2000 pursuant to Directive 95/46/EC of the European Parliament and of the Council on the adequacy of the protection provided by the safe harbour privacy principles and related frequently asked questions issued by the US Department of Commerce (notified under document number C(2000) 2441) (Text with EEA relevance.) , Official Journal L 215 , 25/08/2000 P. 0007 - 0047



### **7.1.4 The specific case of the Republic of Korea**

The Republic of Korea is a member of the Organization for Economic Cooperation and Development and has adopted the OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data. The Act on Promotion of Information and Communications Network Utilization and Data Protection<sup>20</sup> came into effect in 2000. The Act adopts "fair information principles" and rules similar to the European ones for the collection, use, and disclosure of personal data by "providers of information and communications services," such as common carriers, Internet service provider and other intermediaries, particularly content providers.

## **7.2 Information and Communication Technologies under the aspect of European Data Protection and Privacy Framework**

Information and communication technologies (ICT) are enabling tremendous capabilities in virtually every aspect of our lives. ICT have been compared to other important inventions of the past, such as electricity. While it may be too early to assess their real historical impact, the link between ICT and economic growth in developed countries is clear. ICT have created employment, economic benefits and contributed to overall welfare. The impact of ICT goes beyond the purely economic, since it has played an important role in boosting innovation and creativity. Furthermore, ICT have transformed the way people work, socialise and interact. For example, people increasingly rely on ICT for social and economic interactions. In the light of such benefits, the European Institutions have all expressed their commitment to support ICT as a necessary tool to improve the competitiveness of European industry and to accelerate Europe's economic recovery.

With the opportunities and benefits that accompany the development of ICT came new risks, particularly for the privacy and protection of personal data of individuals. ICT often lead to a proliferation (quite often in ways that are out of sight to individuals) in the amount of information that is collected, sorted, filtered, transferred or otherwise retained, and the risks to such data therefore multiply. The potential benefits of ICT could only be enjoyed in practice if they were able to generate trust, in other words, if they could secure user willingness to depend on ICT because of their characteristics and benefits.

Such trust would only be generated if ICT were reliable, secure, under individuals' control and if the protection of their personal data and privacy was guaranteed. Widespread risks and failures such as those illustrated above, particularly when they entail the misuse or breaches of personal data exposing the privacy of individuals, are likely to endanger user trust in the information society. This could seriously jeopardise the development of ICT and the benefits they could bring. Trust has been identified as a core issue in the emergence and successful deployment of information and communications technologies. If people do not trust ICT, these technologies are likely to fail.

The EU has a robust data protection and privacy framework enshrined in Directive 95/46/EC, Directive 2002/58/EC and the jurisprudence of the European Court of Human Rights and the Court of Justice. The Data Protection Directive applies to 'any operation or set of operations which is performed upon personal data' (collection, storage, disclosure, etc.). It imposes compliance with certain principles and obligations upon those who process personal data ('data controllers'). It sets forth individual rights, such as the right to access personal information. The ePrivacy Directive deals specifically with the protection of privacy in the electronic communication sector.

---

<sup>20</sup> Available from: <http://unpan1.un.org/intradoc/groups/public/documents/APCITY/UNPAN025694.pdf>



According to European Convention for the Protection of Human Rights and Fundamental freedoms: "Everyone has the right to respect for his private and family life, his home and his correspondence". On the other hand, information relating to individuals, called 'personal data', is collected and used in many aspects of everyday life. An individual gives personal data when he/she, for example, registers for a library card, signs up for gym membership, opens a bank account, etc. Personal data can be any data that identifies an individual, such as a name, a telephone number, or a photo.

Advancement in computer technology along with new telecommunications networks is allowing personal data to travel across borders with greater ease. As a result, data concerning the citizens of one Member State are sometimes processed in other Member States of the EU. Therefore, as personal data is collected and exchanged more frequently, regulation on data transfers becomes necessary.

In this context, the E.U. Directives regarding data protection demanded good data management practices on the part of the entities who process data, called 'data controllers'. These included the obligation to process data fairly and in a secure manner and to use personal data for explicit and legitimate purposes and also the right of individuals to be informed when personal data was processed and the reason for this processing, the right to access the data and if necessary, the right to have the data amended or deleted.

The data protection Directive applies to 'any operation or set of operations which is performed upon personal data,' called 'processing' of data. Such operations include the collection of personal data, its storage, disclosure, etc.

In addition, there is a separate Directive, Directive 97/66/EC, that deals specifically with the protection of privacy in telecommunications. This Directive states that Member States must guarantee the confidentiality of communication through national regulations. This means that any unauthorised listening, tapping, storage or other kinds of interception or surveillance of communications is illegal.

Therefore, European law regarding the protection of personal data and the confidentiality of communications by means of a public communications network is sufficient and covers up the lack of such a legislation in national standards, where constitutional law by general references to human value and protection of ones personality, along with the independent authority of personal data and the independent authority of the confidentiality of communications, seek to face the complexity of the issues involved in the electronic communication sector in terms of suppression of the problems emerged when such a rights' violation arises. So it is in motto of European law that this research will be centralized, and most specifically the following research is based on directives 95/46/EC, 97/66/EC, 2002/58/EC, 2002/22/EC, 2009/135/EC.

All the above directives ensure an equivalent level of protection of fundamental rights and freedoms, in particular the right to privacy and the right to confidentiality, with respect to the processing of personal data in the electronic communications sector, and the free movement of such data and of electronic communications equipment and services in the Community.

According to directive 95/46/EC:

- (a) **'personal data'** means any information relating to an identified or identifiable natural person ('data subject'); an identifiable person is one who can be identified, directly or indirectly, in particular by reference to an identification number or to one or more factors specific to his physical, physiological, mental, economic, cultural or social identity;
- (b) **'processing of personal data'** ('processing') means any operation or set of operations which is performed upon personal data, whether or not by automatic means, such as collection, recording, organization, storage, adaptation or alteration, retrieval, consultation, use,



disclosure by transmission, dissemination or otherwise making available, alignment or combination, blocking, erasure or destruction;

- (c) **“personal data filing system”** ('filing system') means any structured set of personal data which are accessible according to specific criteria, whether centralized, decentralized or dispersed on a functional or geographical basis;
- (d) **“controller”** means the natural or legal person, public authority, agency or any other body which alone or jointly with others determines the purposes and means of the processing of personal data; where the purposes and means of processing are determined by national or Community laws or regulations, the controller or the specific criteria for his nomination may be designated by national or Community law;
- (e) **“processor”** means a natural or legal person, public authority, agency or any other body which processes personal data on behalf of the controller;
- (f) **“third party”** means any natural or legal person, public authority, agency or any other body other than the data subject, the controller, the processor and the persons who, under the direct authority of the controller or the processor, are authorized to process the data;
- (g) **“recipient”** means a natural or legal person, public authority, agency or any other body to whom data are disclosed, whether a third party or not; however, authorities which may receive data in the framework of a particular inquiry shall not be regarded as recipients;
- (h) **“the data subject's consent”** means any freely given specific and informed indication of his wishes by which the data subject signifies his agreement to personal data relating to him being processed.

Where measures aiming to ensure that terminal equipment is constructed so as to safeguard the protection of personal data and privacy are adopted pursuant to Directive 1999/5/EC or Council Decision 87/95/EEC of 22 December 1986 on standardization in the field of information technology and telecommunications. According to all the above directives data controllers are subject to the following rules.

- Data must be processed fairly and lawfully.
- They must be collected for explicit and legitimate purposes and used accordingly.
- Data must be relevant and not excessive in relation to the purpose for which they are processed.
- Data must be accurate and where necessary, kept up to date.
- Data controllers are required to provide reasonable measures for data subjects to rectify, erase or block incorrect data about them.
- Data that identifies individuals must not be kept longer than necessary. Personal data can only be processed (e.g. collected and further used) if:
  - The data subject has unambiguously given his or her consent, if he or she has agreed freely and specifically after being adequately informed.
  - Data processing is necessary for the performance of a contract involving the data subject or in order to enter into a contract requested by the data subject
  - Processing is required by a legal obligation
  - Processing of data is necessary to protect an interest that is essential for the data subject's life.
  - Processing is necessary to perform tasks of public interests or tasks carried out by official authorities (such as the government, the tax authorities, the police etc.);
  - Finally data can be processed whenever the controller or a third party has a legitimate interest in doing so. However, this interest cannot override the interests or fundamental rights of the



data subject, particularly the right to privacy. This provision establishes the need to strike a reasonable balance, in practice, between the business interest of the data controllers and the privacy of data subjects.

According to that, personal data can be accessed only by authorised personnel for legally authorised purposes. Personal data stored or transmitted must be protected against accidental or unlawful destruction, accidental loss or alteration, and unauthorised or unlawful storage, processing, access or disclosure, by a security policy with respect to the processing of personal data.

In particular, listening, tapping, storage or other kinds of interception or surveillance of communications and the related traffic data by persons other than users, without the consent of the users concerned, is prohibited except technical storage which is necessary for the conveyance of a communication without prejudice to the principle of confidentiality.

It would be rather illogical and without legal justification to exempt such an important means of transfer as the Internet from the scope of the data protection Directive. On the contrary, the sheer volume and multiple nature of personal data transmitted through the Internet all over the world, including to countries with no adequate protection, require particular attention. The data protection Directive is therefore technologically neutral: its provisions apply, irrespective of the technological means used to process personal data. For example, the Directive applies to the invisible collection of personal data on the Internet (e.g.: the "cookies" which are used to track the individual surfing habits). On the other hand, if personal data are collected in a "visible" way, it might be argued that an individual transferring his own data has given his consent to such a transfer, provided that he is properly informed about the risks involved.

More thoroughly, the use of electronic communications networks to store information or to gain access to information stored in the terminal equipment of a subscriber or user is only allowed on condition that the subscriber or user concerned is provided with clear and comprehensive information in accordance with Directive 95/46/EC, inter alia about the purposes of the processing, and is offered the right to refuse such processing by the data controller. On the other hand, this shall not prevent any technical storage or access for the sole purpose of carrying out or facilitating the transmission of a communication over an electronic communications network, or as strictly necessary in order to provide an information society service explicitly requested by the subscriber or user.

At the same time, the liberalisation of electronic communications networks and services markets, also provided by the same directives, and rapid technological development, have combined to boost competition and economic growth and resulted in a rich diversity of end-user services accessible via public electronic communications networks. It is necessary to ensure that consumers and users are afforded the same level of protection of privacy and personal data, regardless of the technology used to deliver a particular service. Both efficient investment and competition should be encouraged in tandem, in order to increase economic growth, innovation and consumer choice.

Reliable and secure communication of information over electronic communications networks is increasingly central to the whole economy and society in general.

So, it is most likely that third parties may wish to store information on the equipment of a user, or gain access to information already stored, for a number of purposes, ranging from the legitimate (such as certain types of cookies) to those involving unwarranted intrusion into the private sphere (such as spyware or viruses). It is therefore of paramount importance that users be provided with clear and comprehensive information when engaging in any activity which could result in such storage or gaining of access. The methods of providing information and offering the right to refuse should be as user-friendly as possible. Exceptions to the obligation to provide information and offer the right to refuse should be limited to those situations where **the technical storage or access is**



**strictly necessary for the legitimate purpose of enabling the use of a specific service explicitly requested by the subscriber or user.**

With all the above thoughts, we can draw as a conclusion, that protection of personal data and safeguarding of confidentiality of communications within the scope of community law is equally important as the facilitating and the evolution of network communications, in terms of promoting ones right to access to all information within one's means, which is regarded as the ultimate foundation of democracy. At that prospective, any initiative aiming at easier or chipper or faster spreading of information, that is aiming at more friendly access to information, should be encouraged, as long as both safety and confidentiality of communications and preservation of personal data is ensured.

Practically and as far as the present research is concerned, that means, that administration of information already spread or stored for the purpose of enabling the use of a specific service requested by a user or a subscriber, should be restrained within the limits of technical storage only, and should never penetrate into the content of the information stored. Otherwise, the user should be informed in advance and should be given the right to refuse such an invasion or provide permission to any access.

### ***7.3 Privacy issues related to the use of DPI in COAST***

The deployment of DPI in COAST is compatible with EU law given that COAST complies with all relevant requirements. More precisely, article 4 of the e-Privacy Directive as well as article 7 (b) of the Data Protection Directive legitimize the use of DPI, provided that certain conditions are met.

Directive 2002/58/EC of the European Parliament and of the Council concerning the processing of personal data and the protection of privacy in the electronic communications sector ("e-Privacy Directive") applies to the processing of personal data in connection with the provision of publicly available electronic communications networks in the Community. The provisions of this Directive particularize and complement the Data Protection Directive. The confidentiality of communication is protected, in particular, by Article 5 of the e-Privacy Directive which reads as follows:

"...Member States shall ensure the confidentiality of communications and the related traffic data by means of a public communications network and publicly available electronic communications services, through national legislation. In particular they shall prohibit listening, tapping, storage or other kinds of interception or surveillance of communications and the related traffic data by persons other than users, without the consent of the users concerned, except when legally authorized to do so..."

However, Article 4 of the e-Privacy Directive foresees that "The provider of a publicly available telecommunications service must take appropriate technical and organizational measures to safeguard security of its services, if necessary in conjunction with the provider of the public telecommunications network with respect to network security".

DPI can be justified by the obligation to take appropriate technical and organizational measures to safeguard security of the COAST services as foreseen in Article 4 of the e-Privacy Directive quoted above. Indeed the COAST architecture is designed to exploit the data gathered through DPI in order to prevent network congestion thus eliminating threats to its general performance and securing its services<sup>21</sup>. Using DPI filters for the purpose of Article 4 can thus be compatible with Article 5 of the e-Privacy Directive above.

---

<sup>21</sup> The Working Party 29 considers that Article 4 of the e-Privacy Directive, requiring providers to take appropriate technical and organisational measures to safeguard security of their services, is concerned about the security of the ESP and network services per se but also about the general performance of network



In addition, the processing of personal data within COAST may also be legitimized under Article 7 b of the Data Protection Directive 95/46 which foresees such processing when “necessary for the performance of a contract to which the data subject is a party”. Here “data subject” is the person to which the aforementioned data refers to. For example DPI information is used in COAST to allocate a higher priority to low latency services, such as VoIP, video conferencing or video streaming, thus ensuring the performance of a service contract with the COAST users.

Moreover, the processing of personal data within COAST complies with the specific requirements of the Data Protection Directive (95/46/EC) and e-Privacy Directive (2002/58/EC). In particular, under Article 10 « Information to be given to the data subject » of the Data Protection Directive, COAST informs its users of their network management policies, and in particular DPI filtering, in a clear and unambiguous way. COAST respects the requirements of Article 6 “Principles relating to data quality” of the Data Protection Directive. More particularly Article 6 paragraph 1, letter (a) of the Data Protection Directive establishes that data must be processed fairly and lawfully, which reinforces the obligation to be completely transparent regarding the conditions of the processing of individuals’ data. Article 6(1)(b) sets forth the purpose limitation principle. This principle prohibits the processing of personal data which is not compatible with the purposes that legitimized the initial collection. Article 6, paragraph 1, letter (c) establishes the data minimization principle, thus personal data must be “adequate, relevant and not excessive in relation to the purposes for which they are collected and/or further processed”. COAST architecture has been designed taking particular account of the objectives of minimizing the processing of personal data and of using anonymous or pseudonymous data where possible. Article 6, paragraph 1, letter (e) requires data to be deleted when it is no longer necessary for the purpose for which the data were collected (retention principle). Compliance with this principle requires limiting the storage of information. Accordingly, COAST specifies and respects express timeframes under which data will be retained. In addition to the above, COAST also complies with Article 4 of the e-Privacy Directive which requires providers of a publicly available electronic communication service to inform subscribers of particular risks of breaches of the security of the network.

The next table regroups all the aforementioned requirements as well as the measures taken by COAST to ensure its compliance with the relevant Directives.

Requirements	Measures
Directive 95/46/EC , Article 10 « Information to be given to the data subject »	COAST will ensure that its users are properly informed prior to using of the COAST network through the terms and conditions of use
Directive 95/46/EC, Article 6 paragraph 1, letter (a)	COAST will ensure that all data used by its servers or applications will be treated and processed in a transparent and lawful way.
Directive 95/46/EC, Article 6 paragraph 1, letter (b)	COAST is taking every measure possible to guarantee that the data collected by its servers is used only for the legitimate purpose of network management and for no other purpose. Appropriate security and authentication mechanisms are in place to prohibit any unauthorized access.

services. In other words, threats to the general performance of network services can justify ISPs and ESPs to engage in DPI. (Working Party 29, Opinion 2/2006, WP 118)



Directive 95/46/EC, paragraph 1, letter (c)	Article 6	6	COAST will use only the minimum data necessary for its purposes. Wherever possible anonymisation or pseudonimisation of data will be used.
Directive 95/46/EC, paragraph 1, letter (e)	Article 6	6	COAST will not attempt to store any data for a longer period than the limited timeframes during which they need to be processed and fed to the network management modules.
Directive 2002/58/EC, paragraph 2	Article 4	4	Even though COAST takes all necessary precautions and measures to avoid being in a situation of having to cope with a security risk, COAST will nevertheless inform its users in such unlikely circumstances.

## 7.4 Conclusion

The COAST project has taken into account the all applicable recommendations and guidelines mentioned above in the provision of its service. Therefore its network management practices including DPI comply with the European legal Framework.

The European Commission will propose in 2011 a new general legal framework for the protection of personal data in the EU covering data processing operations in all sectors and policies of the EU. COAST will be continuously monitoring any changes in this field, and will adapt accordingly its practices, should the need arise.



## 8 Appendix B – Traffic statistics

In this Section we report traffic statistics collected at the Minnesota Internet Traffic Studies, and on the GARR-G network web sites .

**Minnesota Internet Traffic Studies (MINTS):** reach information about Internet traffic from a variety of sources is collected on the web pages available at [36]. Most of these web sites, over 100 at this point, are public (their URLs are provided), and they continuously monitor traffic on a variety of networks. Monitored networks also include the largest public Internet exchanges in the world, and altogether they account for a substantial fraction of the world's Internet traffic. Sites being monitored are listed in Table 1 , where in the second column the average aggregate traffic volume monitored in the period reported in column 5 and 6 is given.

**GARR-G:** The GARR [37] is a Wide Area Network and its services are dedicated to the Italian Research, Academic and Education communities. Currently the GARR network connects approximately 450 end sites, including research centers, laboratories and other facilities, universities, Institutes for Research in Health Care (IRCCS), Music Conservatories and Academies of Art (AFAM), Libraries, Archives, Schools, Museums and other R&E institutions of national relevance, for overall 2.000.000 end users. The GARR-G network topology is reported in Figure 29.

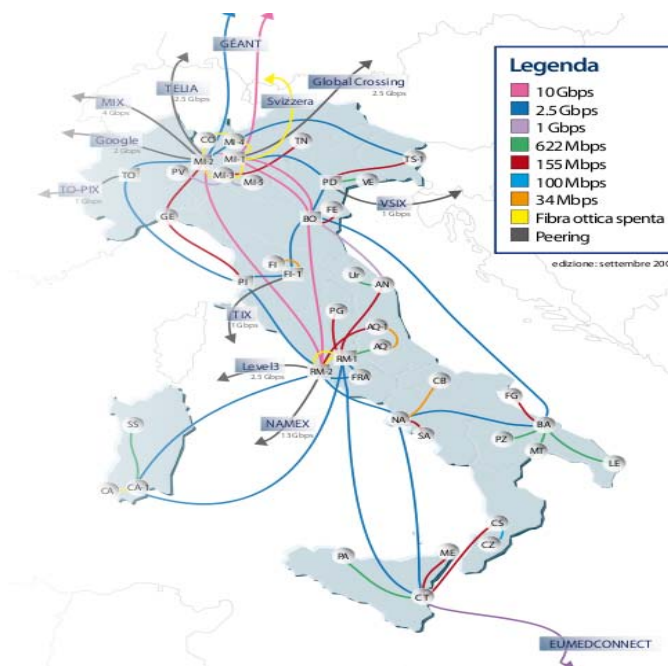


Figure 29: GARR-G network topology

Figure 30 depicts a small portion of the network weather map, where we have:

- routers (blue in Figure 30).
- interconnection with General Internet using three of the main international commercial upstream provider (Global Crossing, Level3 and Telia) (red in Figure 30), interconnection with all major Italian commercial ISPs (Telecom Italia, Wind, Fastweb, etc) at the main public Internet Exchange Points in the country, namely MIX in Milan (red in Figure 30).
- The GARR-G network is part of the worldwide system of Research and Education Networks (NERENs It connects other NERENs in Europe and Worldwide through a 10 Gbps link (plus 2.5 Gbps backup link) to the GEANT2 pan- European backbone (red in Figure 30).



Site	Traffic [bit/s]	[bit/s]	Growth Rate	DD-YYYYY)	YYYYY)
Amsterdam Internet Exchange (AMS-IX)	4.23E+011	4.34E+011	1.084	12/31/08	08/22/09
Athens Internet Exchange (AIX)	5.16E+009	2.57E+009	0.124	12/31/08	09/01/09
Australian National University (ANU)	9.35E+007	1.29E+008	2.606	12/31/08	09/01/09
Bangladesh Internet Exchange (BDIX)	6.98E+006	7.81E+006	1.4	12/31/08	09/01/09
Belgian National Internet Exchange (BNIX)	1.07E+010	1.23E+010	1.502	12/31/08	09/01/09
Boca Network (Boca)	2.36E+008	2.37E+008	1.021	01/05/09	08/31/09
Budapest Internet Exchange (BIX)	6.63E+010	6.38E+010	0.889	12/31/08	08/25/09
California Research and Education Network at University	3.45E+009	9.64E+008	0.019	12/31/08	08/22/09
Catalonia Neutral Internet Exchange (CATNIX)	3.43E+008	4.51E+008	2.264	12/31/08	09/01/09
Creighton University (CREIGHTON)	5.38E+007	5.32E+007	0.967	12/31/08	09/01/09
Deutsche Commercial Internet Exchange (DE-CIX)	6.50E+011	7.01E+011	1.314	12/31/08	07/20/09
Espananix Punto Neutro Espanol de Internet (ESPANIX)	8.40E+010	6.78E+010	0.527	12/31/08	09/01/09
Finnish Communication and Internet Exchange (FICIX)	1.37E+010	1.26E+010	0.762	12/31/08	08/22/09
Hong Kong Internet Exchange (HKIX)	5.71E+010	6.22E+010	1.299	12/31/08	08/24/09
Internet Neutral Exchange Point (INEX)	2.05E+009	2.52E+009	1.843	12/31/08	09/01/09
Israeli Internet Exchange (IIX)	9.35E+008	1.11E+009	1.669	12/31/08	09/01/09
Japan Internet Exchange (JPIX)	7.97E+010	8.48E+010	1.238	12/31/08	07/30/09
Kharkov Internet Exchange (KH-IX)	2.25E+008	2.39E+008	1.261	12/31/08	07/20/09
KleyRex Internet Exchange (KLEYREX)	1.52E+009	1.79E+009	1.624	12/31/08	09/02/09
Korea Internet Exchange (KINX)	3.78E+010	3.30E+010	0.665	12/31/08	09/01/09
London Internet Exchange (LINX)	2.93E+011	3.14E+011	1.248	12/31/08	08/21/09
London Network Access Point (LoNAP)	4.89E+009	3.72E+009	0.44	12/31/08	08/31/09
Los Angeles International Internet Exchange (LAIX)	4.82E+009	4.98E+009	1.104	12/31/08	09/01/09
Moroccan Academic and Research Wide Area Network (M	3.79E+007	3.22E+007	0.614	12/31/08	09/01/09
Nautilus Mediteranean Exchange Point (NAMEX)	8.61E+009	7.28E+009	0.597	12/31/08	08/24/09
Nepal Internet Exchange (NPIX)	8.06E+006	8.05E+006	0.994	12/31/08	09/01/09
Neutral Internet Exchange of India (NIXI)	3.20E+009	4.38E+009	2.571	12/31/08	09/01/09
New York Internet Exchange (NYIX)	5.21E+010	5.19E+010	0.991	12/31/08	09/01/09
Norwegian Internet Exchange (NIX)	1.09E+010	1.01E+010	0.787	12/31/08	09/01/09
Novosibirsk Internet Exchange (NSK-IX)	3.32E+008	3.60E+008	1.285	12/31/08	08/25/09
Nuremburg Internet Exchange (N-IX)	1.94E+009	2.28E+009	1.643	12/31/08	08/25/09
Ottawa Internet Exchange (OTTIX)	4.76E+007	4.75E+007	0.993	12/31/08	09/01/09
Pacific Wave (PACIFICWAVE)	3.78E+009	3.85E+009	1.058	12/31/08	09/01/09
Paris Network Access Point (PANAP)	3.90E+010	4.02E+010	1.102	12/31/08	08/20/09
Polish Internet Exchange (PL-IX)	6.07E+010	6.63E+010	1.314	12/31/08	08/24/09
Porto Federal de Interconexao de Redes (PTT-FIX)	7.86E+007	9.51E+007	1.901	12/31/08	08/05/09
Punto Neutro Vasco de Internet (EUSKONIX)	2.66E+007	2.14E+007	0.522	12/31/08	09/01/09
Reykjavik Internet Exchange (RIX)	2.34E+008	2.67E+008	1.478	12/31/08	09/01/09
Romanian Network for Internet Exchange (RoNIX)	3.86E+009	3.82E+009	0.967	12/31/08	09/01/09
Sitel Internet Exchange (SITEPIX)	2.51E+008	2.45E+008	0.931	12/31/08	09/01/09
St. Petersburg Internet Exchange (SPB-IX)	6.80E+009	8.09E+009	1.714	12/31/08	08/24/09
Swedish Internet Exchange Netnod (NETNOD-IX)	8.38E+010	6.85E+010	0.533	12/31/08	08/22/09
Swiss Internet Exchange (SWISSIX)	2.19E+009	2.39E+009	1.312	12/31/08	08/24/09
Taiwan Internet Exchange (TWIX)	2.80E+009	1.91E+009	0.317	12/31/08	09/01/09
Tallinn Internet Exchange (TLLIX)	6.72E+008	5.96E+008	0.699	12/31/08	09/01/09
Tallinn Internet Exchange (2) (TIX.EE)	2.40E+008	7.03E+007	0.022	12/31/08	08/23/09
TelX Internet Exchange (ATLANTAIX)	1.29E+010	1.75E+010	2.594	12/31/08	08/23/09
Torino Piedmonte Internet Exchange (TOP-IX)	1.34E+010	1.29E+010	0.901	12/31/08	08/24/09
Tuscany Internet Exchange (TIX.IT)	9.98E+007	9.77E+007	0.941	12/31/08	09/01/09
University of Memphis (UMEMPHIS)	1.07E+008	6.35E+007	0.212	12/31/08	09/01/09
Warsaw Internet Exchange (WIX)	2.68E+009	1.96E+009	0.376	12/31/08	08/24/09

Table 14: MINTS traffic volume

For each router/exchange point in the GARR-G backbone it is possible to download the aggregate traffic statistics available per day, week, month and year that we have reported in Table 15.

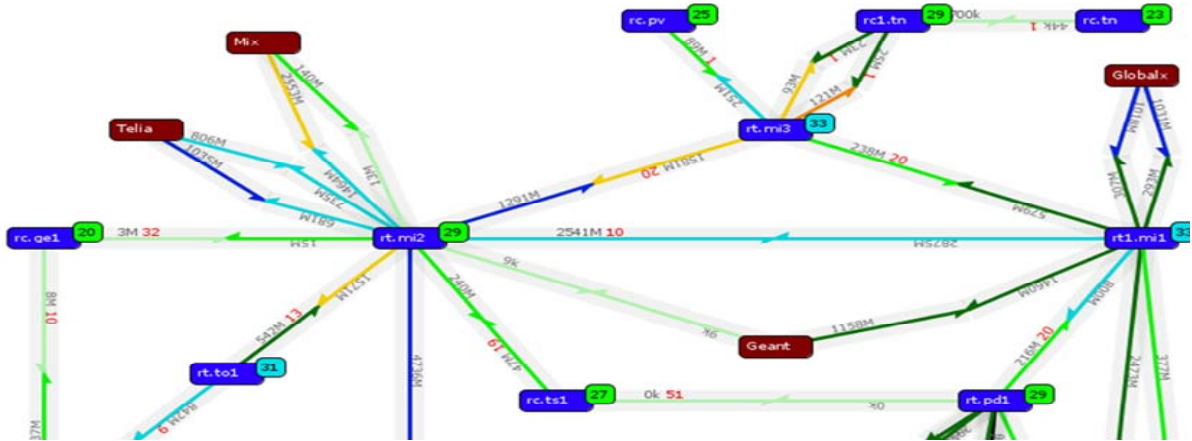


Figure 30: GARR-G connectivity topology

Cite	Daily		Weekly		Monthly		Yearly	
	Max in [bit/s]	Max out [bit/s]	Max in [bit/s]	Max out [bit/s]	Max in [bit/s]	Max out [bit/s]	Max in [bit/s]	Max out [bit/s]
RC.PA1	4,90E+08	5,15E+08	5,19E+08	5,18E+08	6,46E+08	6,36E+08	1,12E+09	8,91E+08
RC.ME	1,11E+08	1,13E+08	1,11E+08	1,13E+08	1,29E+08	1,36E+08	1,62E+08	1,66E+08
RC.CS	2,25E+08	2,33E+08	2,38E+08	2,35E+08	2,39E+08	2,43E+08	2,71E+08	2,99E+08
RC.CB	3,61E+07	4,90E+08	4,90E+08	4,90E+08	4,90E+08	4,90E+08	4,90E+08	4,90E+08
RC.SA	3,61E+07	3,77E+07	4,47E+07	4,46E+07	5,68E+07	5,51E+07	6,08E+07	6,04E+07
RC.PG	1,98E+08	1,99E+08	2,24E+08	2,39E+08	2,31E+08	2,64E+08	2,70E+08	2,96E+08
RC.CZ	3,89E+07	3,75E+07	7,72E+07	7,70E+07	8,96E+07	8,82E+07	1,11E+08	1,03E+08
RT.RM1	2,21E+09	2,27E+09	2,41E+09	2,51E+09	3,18E+09	3,38E+09	3,40E+09	3,58E+09
RT.CT1	1,57E+09	1,64E+09	1,81E+09	1,76E+09	2,90E+09	2,93E+09	3,38E+09	3,45E+09
RT.NA1	3,06E+09	2,98E+09	3,57E+09	3,56E+09	4,86E+09	4,79E+09	6,74E+09	6,50E+09
RC1.SA	5,49E+08	5,43E+08	5,65E+08	5,68E+08	5,77E+08	5,82E+08	6,17E+08	6,22E+08
RC.SS	9,71E+07	9,51E+07	1,55E+08	1,26E+08	1,62E+08	1,45E+08	1,85E+08	1,55E+08
RC.CA	9,83E+07	9,94E+07	1,26E+08	1,53E+08	1,37E+08	1,58E+08	4,46E+08	4,47E+08
RC.AN	3,51E+08	3,64E+08	4,28E+08	4,44E+08	4,51E+08	4,81E+08	5,12E+08	6,14E+08
RC.UR	1,35E+08	1,32E+08	2,34E+08	2,29E+08	2,38E+08	2,41E+08	2,89E+08	2,80E+08
RC.PZ	1,23E+08	1,22E+08	1,58E+08	1,67E+08	1,62E+08	1,82E+08	2,14E+08	2,42E+08
RC.CA1	3,40E+08	3,35E+08	4,86E+08	5,56E+08	1,05E+09	1,08E+09	1,88E+09	2,12E+09
RC.FG	5,57E+07	5,60E+07	6,59E+07	6,61E+07	7,85E+07	7,88E+07	1,05E+08	9,60E+07
RC.FRA	7,18E+08	7,34E+08	1,16E+09	1,17E+09	1,93E+09	1,94E+09	2,39E+09	2,38E+09
RT.BA1	1,80E+09	1,77E+09	3,33E+09	3,33E+09	4,29E+09	4,44E+09	5,98E+09	6,23E+09
<b>RT.RM2</b>	<b>1,07E+10</b>	<b>1,22E+10</b>	<b>1,24E+10</b>	<b>1,33E+10</b>	<b>1,63E+10</b>	<b>1,71E+10</b>	<b>1,82E+10</b>	<b>2,08E+10</b>
<b>RT1.BO1</b>	<b>1,62E+10</b>	<b>1,50E+10</b>	<b>2,33E+10</b>	<b>2,29E+10</b>	<b>2,86E+10</b>	<b>2,93E+10</b>	<b>3,95E+10</b>	<b>4,25E+10</b>
NAMEX	6,16E+08	3,02E+09	5,99E+08	2,97E+09	6,15E+08	2,97E+09	6,15E+08	3,06E+09
RC.FU	1,16E+06	1,15E+06	3,89E+06	3,95E+06	6,01E+06	6,19E+06	4,01E+07	4,03E+07
RT.PI1	4,07E+09	3,95E+09	4,34E+09	4,30E+09	4,79E+09	4,94E+09	6,88E+09	6,85E+09
RT.TO1	3,06E+09	3,08E+09	4,27E+09	4,27E+09	4,87E+09	4,66E+09	5,41E+09	5,36E+09
RT.TS1	1,02E+09	1,00E+09	1,47E+09	1,56E+09	1,61E+09	1,77E+09	2,10E+09	2,19E+09
RT.PD1	3,59E+09	3,29E+09	3,60E+09	3,35E+09	4,26E+09	4,13E+09	6,33E+09	6,21E+09
GEANT	2,16E+09	3,31E+09	2,86E+09	4,75E+09	4,72E+09	7,76E+09	8,77E+09	9,41E+09
RT1.MI1	4,18E+08	1,47E+09	7,62E+08	1,58E+09	1,06E+09	1,58E+09	1,27E+09	1,72E+09
<b>RT.MI2</b>	<b>1,50E+10</b>	<b>1,50E+10</b>	<b>1,53E+10</b>	<b>1,52E+10</b>	<b>1,90E+10</b>	<b>1,95E+10</b>	<b>2,09E+10</b>	<b>2,07E+10</b>
RT.MI3	3,91E+09	4,38E+09	3,95E+09	4,56E+09	5,01E+09	5,73E+09	2,80E+10	6,69E+09
TELIA	1,08E+09	7,57E+08	1,16E+09	8,36E+08	2,15E+09	1,07E+09	2,38E+09	1,07E+09
GLOABALX	1,31E+09	4,37E+08	1,31E+09	4,16E+08	1,26E+09	3,64E+08	7,04E+08	6,22E+06
MIX	2,74E+09	1,85E+09	2,74E+09	1,85E+09	2,94E+09	1,85E+09	3,06E+09	2,70E+09

Table 15: GARR-G traffic volume



For visual evaluation of the traffic statistics collected at the MINTS, GARR-G and CAIDA we plot the results in Figure 31.

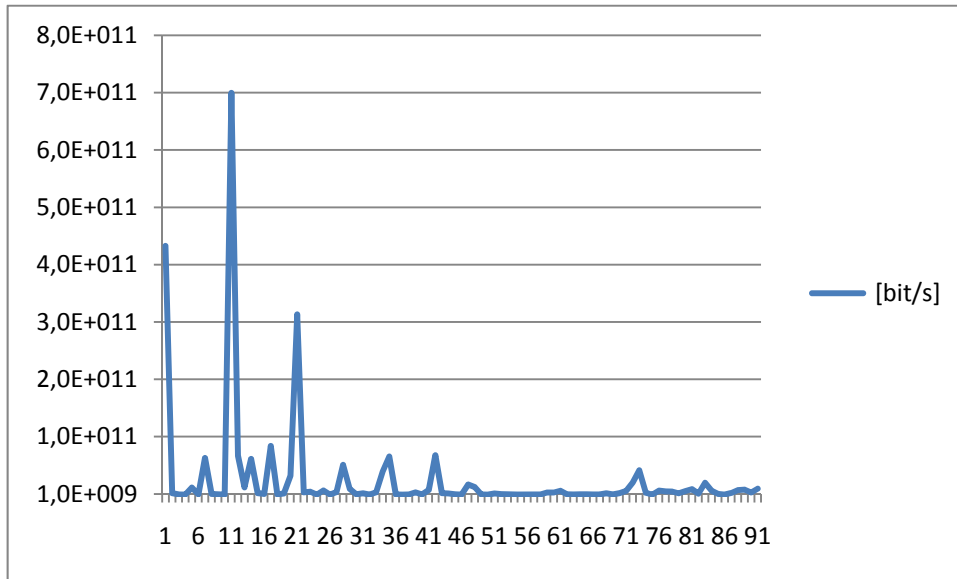


Figure 31: Traffic statistics