Cost-based placement of vDPI functions in NFV infrastructures

Mathieu Bouet*, Jérémie Leguay†, Vania Conan*
* Thales Communications & Security, {firstname.name}@thalesgroup.com
† France Research Center, Huawei Technologies Co. Ltd, jeremie.leguay@huawei.com

***Abstract*—Network Functions Virtualization (NFV) is transforming how networks are architected and network services delivered. The network is more flexible and adaptable, it can scale with traffic demands. To manage video traffic in the network, or get protection from cyber-attacks, Deep Packet Inspection is increasingly deployed at specific locations in the network. The virtual Deep Packet Inspection (vDPI) engines can be dynamically deployed as software on commodity servers within emerging NFV infrastructures. For a network operator, deploying a set of vDPIs over the network is a matter of finding the appropriate placement that meets the traffic management or cyber-security targets (such as the number of inspected flows) and operational cost constraints (license fees, network efficiency or power consumption). In this work, we formulate the vDPI placement problem as a cost minimization problem. The cost captures the different objectives the operator is pursuing. A placement of vDPIs on the network nodes realizes a trade-off between these possibly conflicting goals. We cast the problem as a multi-commodity flow problem and solve it as an Integer Linear Program (ILP). We then devise a centrality-based greedy algorithm and assess its validity by comparing it with the ILP optimal solution on a real data set (GEANT network with 22 nodes and real traffic matrix). We further analyze the scalability of the heuristic by applying it to larger random networks of up to 100 nodes. The results show the network structure and the costs strongly influence time performance. They also show that after a size limit (between 40 to 80 nodes in our case), the execution time increases exponentially due to combinatorial issues. Finally, they demonstrate that the heuristic well approximate the optimal on smaller problem instances.**

I. Introduction

Deep Packet Inspection (DPI) is a technique that allows fine-grained realtime monitoring of flows and users activity in networks and IT systems. It consists in filtering network packets to examine the data part (and possibly also the header) of a packet flow, identifying traffic types, searching for protocol non-compliance, viruses, spam, intrusions, or any defined criteria. DPI is a key enabler of traffic management, especially to deal with video traffic. Infonetics report that the DPI market is forecast to grow at 22% rate from 2013 to 2018 driven in particular by the video traffic growth in Asian markets and in LTE deployments [1].

Originally implemented as middle-boxes to be installed in the network infrastructure, DPI are evolving towards the network function virtualization (NFV) paradigm: they are virtualized and delivered as software bundles that can be deployed and used on demand on multicore commodity hardware platforms. This approach is strongly supported by Internet Services Providers (ISPs) [2] to build the so-called *NFV infrastructures* by ETSI where the different PoPs (Points-of-Presence) of the network embed dedicated cloud systems. NFV consists in delivering network functions as software which runs as virtualized instances at dedicated locations in the network (e.g., PoP), without the need to install specific equipment for each new service. It is applicable to all network functions, such as firewalling, caching, ciphering or load balancing, in both mobile and fixed networks. It is particularly relevant for DPI since this function is highly dependent on traffic pattern or dynamics. DPI market leaders such as Sandvine offer virtual series of their products with the same functionality and features as the physical ones. Virtualizing network functions enables to rapidly scale up (or down) services, that currently necessitate multiple dedicated hardware appliances, as it only requires the installation or activation of virtual functions on existing server equipment.

With virtualized Deep Packet Inspection (vDPI) network operators are facing a new set of challenges. The key question they have to address is the issue of where to instantiate the vDPI functions in their network. For the operator this decision is the result of a compromise between several possibly conflicting goals: in some case all flows must be monitored, and the obvious choice is to deploy one vDPI on every node. To reduce the cost (license fees, network efficiency or power consumption), it may be worth deploying less vDPI instances and reroute traffic towards these nodes. In other cases sampling the monitoring flows should be sufficient, but here again one should decide how many vDPI instances should be deployed and on which nodes.

In this work we provide a general formulation of the vDPI placement problem as viewed by the network operator. We define a cost function that evaluates the quality of any given placement of vDPI and corresponding routing of the flows. The cost captures the different objectives the operator is aiming at. A placement of vDPIs on the network nodes realizes a trade-off between the objectives. This includes cyber-security targets (such as the number of inspected flows) or operational cost limitations (license fees, network efficiency or power consumption). We cast the minimisation problem as a multi-commodity flow problem and solve it as an Integer Linear Program (ILP). The optimization problem also takes into account operational constraints such as the site opening cost and the used bandwidth cost.

We then devise a centrality-based greedy algorithm and assess its validity by comparing it with the ILP optimal solution on a real data set (GEANT network with 22 nodes and real traffic matrix). Importantly we show the scalability of the heuristic by applying it to larger random networks of up to 100 nodes. The results show the network structure and the costs strongly

978-1-4799-7899-1/15/$31.00

influence time performance. They also show that after a size limit (between 40 to 80 nodes in our case), the execution time increases exponentially due to combinatorial issues. Finally, they demonstrate that the heuristics well approximate the optimal on smaller graph instances.
Our cost-based method provides the number and the locations of the DPI engines to be deployed. It can be used at design time to lower costs and reduce capital expenditures by utilizing the appropriate number of software solutions rather than adding offload hardware. It may also be used at runtime to adapt dynamically DPI capabilities.
The paper is organized as follows. First Section II presents related work. Then Section III details the multi-commodity flow problem and corresponding Integer Linear Program (ILP), while Section IVpresents the centrality-based greedy heuristic. Evaluation results on the real GEANT graph and traffic, as well as on larger random networks are reported and analyzed in Section V. Finally, Section VI concludes the paper.

II. Related work

Recently, NFV [2] has emerged as a new way to design, deploy and manage networking services. This initiative, triggered by the biggest service providers, now gathers more than 250 companies in the pre-standardization group of the ETSI and has been relayed at the IETF and IRTF in several groups such as Service Function Chaining.
NFV aims to shorten service deployment lifecycle by leveraging standard IT virtualization technology to consolidate many network equipment types (network address translation, ciphering, firewalling, intrusion detection, domain name service, caching etc.) onto industry standard high volume servers, switches and storage, which could be located in datacenters and Points of Presence (POP), network nodes and in the end user premises.
Network virtualization is also driven by the convergence of computation, storage and networks in cloud computing. A lot of recent work in the literature only concerns the placement of Virtual Machines (VM) without an integrated view of computation, storage and networks. Several techniques to optimize their placement with respect to server load balancing or energy saving have been proposed [3], [4]. However, the problem of optimizing the placement of VMs in a datacenter differs from the problem of optimizing the placement of VNF. Indeed, the first problem is node-centric, VMs being many and small endpoints, while the second problem is network-centric, VNF being few and large middlepoints.
Several recent works address the performances and the support of software network switching [5] and Deep Packet Inspection [6] on commodity hardware. However, very few works have addressed the optimization of the placement of virtualized network functions. [7] defines a language for specifying the chaining among virtualized network functions. It can be used for describing a network chain placement problem in a geographically distributed network. In our previous work, we addressed virtualized DPI placement with a first meta-heuristic based on a genetic algorithm [8]. This work was in the continuity of the monitoring placement problem in classic networks [9] and did not scale well to large networks. In this paper, we propose the first general formulation of the problem as a linear program and compare it with a new scalable heuristic based on a greedy algorithm that approximates the optimal.

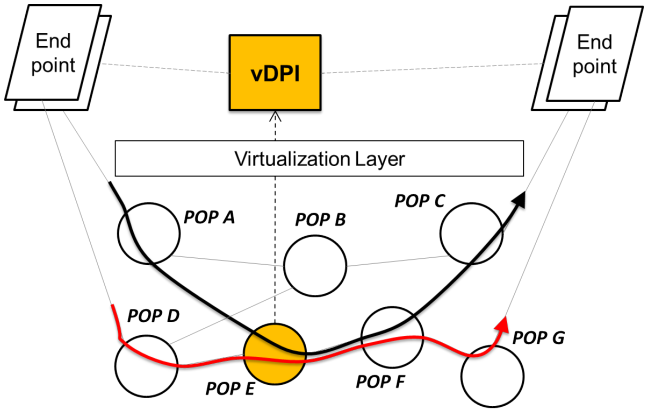


Fig. 1: The objective of minimizing the number of vDPI engines is orthogonal to the objective of minimizing the network load in NFV infrastructures.

III. The vDPI placement problem

This section defines the vDPI placement problem and proposes a linear programming formulation to solve it.

A. Model and Problem Definition

The problem we address in this paper can be formulated as follows: for a given NFV infrastructure and a given traffic demand, find a virtual DPI engine deployment that minimizes the overall cost. This cost optimization problem is the result of a joint minimization between i) the cost of DPI engines and ii) the cost of the overall network footprint induced by flow redirections through the DPI engines, while integrating iii) different operational constraints. These costs are financial costs associated with deployed DPI engines (e.g. license price, CPU utilization, energy consumption...) and the cost of network resources (e.g. total cost of network ownership, capacity of the network to absorb new traffic). Operational constraints may include management limits such as the maximum number of engines to be deployed, the maximum used bandwidth per link (to be able to absorb peaks) and the maximum unmonitored flows.
Different license cost models can be considered for NFV functions. A license cost can either be associated with the global volume of traffic processed or be related to the computing and network resources consumed. The choice of one model or another may depend on the type of business actors operating the service. An operator will offer and operate its NFV infrastructure with a resource-oriented model, while a service provider may adopt a service-oriented model.
In the rest of this work, we consider the case where an operator has to run a DPI function on its own NFV infrastructure, for accounting or cyber-security purpose. We assume that a third-party software editor provides the operator a vDPI software with a two-fold cost model: one cost for the deployment of a NFV POP (Point of Presence) or *site*, and one cost per vCPU reserved on each site. The deployment cost of NFV sites is justified by the fact that a dedicated management software has to be installed to manage the local vDPI instances. It controls

their scaling by activating or shutting down instances according to traffic load, performs software updates and coordinates with the routing system so that flows are redirected through the running instances. In addition to these per site and vCPU license costs, the operator has internal costs for network resources to minimize.

The two main objectives which are, on the one hand, minimizing the number of DPI engines (site and vCPU) and, on the other, minimizing the network load, are in fact orthogonal in this case. Indeed, all the flows have to go through at least one DPI engine to be analyzed. When the number of DPI engines is small the network paths tend to be elongated. Therefore, minimizing the number of engines increases the additional used bandwidth. On the contrary, minimizing the used bandwidth increases the number of DPI engines to be deployed. Fig. 1 illustrates the orthogonality of the objectives in NFV infrastructures. The minimal number of DPI engines, equal to 1 in the example, induces the redirection of the black flow and thus the increase of network usage. On the contrary, the minimal network load, which corresponds to the shortest path, requires the deployment of at least 2 DPI engines, one on each shortest path.

B. ILP formulation

We have extended the ILP formulation for the multi-commodity flow problem [10] to have a DPI probe monitoring each flow. We model the network with a connectivity graph G composed of V nodes and E edges, each of them having a capacity $C_{i,j}$.

Each node represents an NFV infrastructure in the operator's network, potentially hosting virtual DPI probes. Given a set F of flows and a set V of candidate infrastructure sites, we must decide on which site to instantiate DPI probes so as to minimize the global cost. All flow demands must be satisfied, and the DPI probes have a capacity limit of C_{dpi}. The network cost, the license cost per site and the one per vCPU are respectively noted ω_{bw}, ω_{dpi} and ω_{cpu}.

The total cost to minimize is the sum of the cost network resources used and the cost of DPI licenses activated. Let $dpi_i^f = 1$ represent choosing an infrastructure i to monitor the flow f, and 0 otherwise. The size of a flow f is denoted f_s and its source and destination are respectively denoted f_s and f_d. Also, let $x_{i,j}^f = 1$ and $y_{i,j}^f = 1$ represent the assignment of flow f to the network link (i,j), and 0 otherwise. For each flow, the link assignments $x_{i,j}^f$ are prior to the DPI probe and $y_{i,j}^f$ are posterior. To set these two variables, a demand and a conservation constraints are defined. Let also $dpi_i = 1$ mean that at least one DPI probe has been activated on site i, and 0 otherwise. cpu_i indicates the number of vCPU activated on site i.

The problem can then be formulated as the following integer linear program:

Minimize:

$$\sum_{(i,j)\in E, f\in F} f_{size} * (x_{i,j}^f + y_{j,i}^f) * \omega_{bw} + \sum_{i\in V} dpi_i * \omega_{dpi} + \sum_{i\in V} cpu_i * \omega_{cpu}$$

Subject to:

$$\sum_{i,j} x_{i,j}^f + dpi_i^f = 1 \quad \forall f \in F, i = f_s \quad \text{(demand)}$$

$$\sum_{i,j} x_{i,j}^f + dpi_i^f = \sum_{i,j} x_{j,i}^f \quad \forall f \in F, i \neq f_s \quad \text{(conserv.)}$$

$$\sum_{i,j} y_{j,i}^f + dpi_i^f = 1 \quad \forall f \in F, i = f_d \quad \text{(demand)}$$

$$\sum_{i,j} y_{j,i}^f + dpi_i^f = \sum_{i,j} y_{i,j}^f \quad \forall f \in F, i \neq f_d \quad \text{(conserv.)}$$

$$\sum_{f\in F} f_s * (x_{i,j}^f + y_{j,i}^f) \leq C_{i,j} \quad \forall (i,j) \in E \quad \text{(link cap.)}$$

$$\sum_{i\in V} dpi_i^f = 1 \quad \forall f \in F \quad \text{(probe unicity)}$$

$$\sum_{i\in V} f_s * dpi_i^f \leq C_{dpi} * cpu_i \quad \forall f \in F \quad \text{(cpu opening)}$$

$$dpi_i^f \leq dpi_i \quad \forall i \in V, \forall f \in F \quad \text{(site opening)}$$

The problem of producing a set of integer flows satisfying all demands is NP-complete. Our reference implementation uses the GNU linear programming kit[1]. This is a scalable open source linear solver written in C that we used on small instances of the problem to find optimal solutions.

C. Cost models evaluation

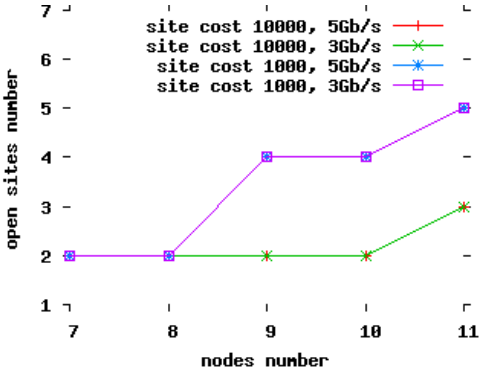
Considering a few random graphs comprising from 7 to 11 nodes, Fig. 2 presents the optimal solutions yielded by the ILP program for different costs. Fig. 2a plots the number of open sites and Fig. 2b plots the number of vCPUs for different DPI capacities (3Gb/s and 5Gb/s) and site opening costs ($ 1000 and $ 10000). An example of such vDPI can be found in [11] with a capacity of up to 8 Gb/s per vCPU. The vCPU opening cost is $ 1000 and the network cost per Mb/s is $ 10. The traffic matrix is a flat one considering a 100Mb/s flow between each pair of nodes. As expected, we can observe that the number of open sites decreases as their site opening cost increases, and that the number of vCPUs increases as the DPI capacity decreases.

In the rest of the paper, we consider the general case where the cost of launching each individual vCPU is significantly lower than that of opening a site for a vDPI. Indeed, when the vDPI probe has a capacity of several gigabits per vCPU, the cost of rerouting traffic becomes predominant over the cost of each vCPU. Under this model and our cost assumptions, the problem falls into the minimization of sites. The heuristic that we present in the next section has been designed for this reduced problem.

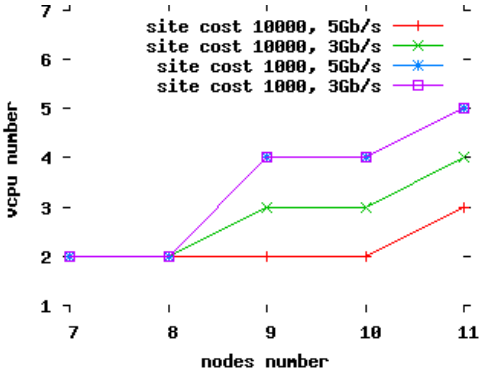
IV. A CENTRALITY-BASED GREEDY ALGORITHM

In this section, we propose a greedy algorithm to solve the virtualized DPI placement problem. Our main objective is to minimize the number of sites upon which DPI probes are activated. To identify key sites where vDPI should be placed, we base our approach on nodes' centrality. Several types of centrality have been proposed and studied in graph theory as

[1] http://www.gnu.org/software/glpk

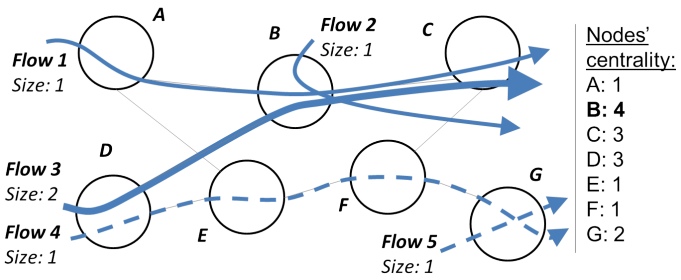


(a) Nb DPI site

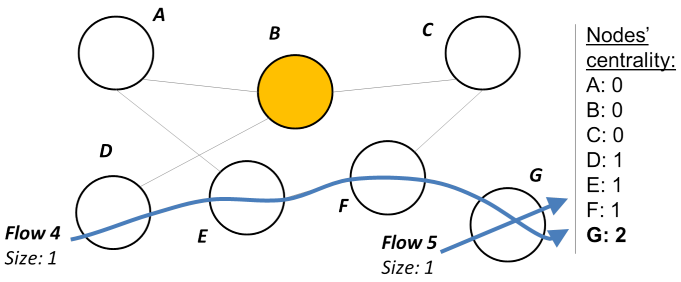


(b) Nb vCPU site

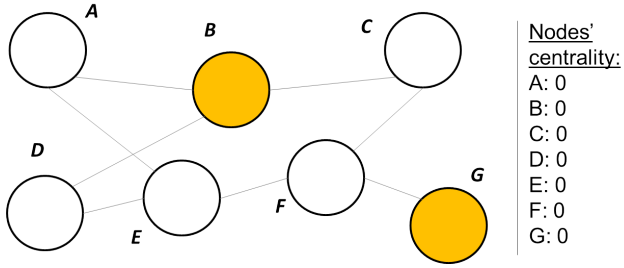
Fig. 2: Optimal placements from the ILP program on random graphs varying the DPI capacity (3Gb/s and 5Gb/s) and the site opening cost ($ 1000 and $ 10000).



(a) First step. Node B has the highest centrality.



(b) Second step. Node G has the highest centrality.



(c) Final stage. All flows have been allocated.

Fig. 3: Example steps for our Greedy heuristic based on centrality.

indicators which identify the most important vertices within a graph in terms of connectivity. One the most used centrality metric is the betweenness centrality, which is equal to the number of shortest paths that pass through a node. A node with high betweenness centrality has a large influence on the transfer of items through the network, under the assumption that item transfers follow shortest paths. We propose a new centrality metric, which is derived from the betweenness centrality. It combines two graphs, the first one being in our case the network topology $G_n(V, E_n)$ and the second one the traffic matrix $G_t(V, E_t)$. Our centrality for node i, namely $centrality_i$, is equal to total size of flows in G_t which have their shortest path going through i in G_n. A node with such a high centrality carries a large amount of traffic and is a good candidate to deploy a vDPI.

Fig. 3 illustrates how our Greedy heuristic works and how nodes' centrality is calculated at each step. Fig. 3a presents the initial network state where the node with the highest centrality is node B. This node is along the shortest paths of three flows: flow 1, flow 2 and flow 3, whose sizes are 1, 1 and 2 units of bandwidth respectively. Hence, the centrality of node B is equal to 4. If a DPI is placed in node B (Fig. 3c), two flows remain unallocated: flow 4 and flow 5, whose size is 1 unit of bandwidth. At this stage, if we forget about the flows that are already monitored at B, the node with the highest centrality is node G, as it is on the shortest paths of two flows. If a DPI is placed in node G, all flows are allocated to a DPI without deviating from their shortest path and therefore without using additional network resources. In more general cases, the greedy algorithm has to trade-off the deployment of vDPI probes and the use of additional network resources.

The objective function to balance the two costs remains the same as the one formulated for the integer linear program. It corresponds to the sum of the cost of used network resources and the cost of DPI licenses activated. The heuristic we propose consists in a greedy algorithm that, at each step, considers placing a new virtualized DPI in the node that has the highest centrality until the global cost stops decreasing. It is described in Algorithm 1. Once a new location has been chosen, the fitness value to evaluate the global cost of the DPI placement is given by Equation 1.

$$fitness(G(V,E), F, DPI) = \omega_{dpi} * \sum_{i \in DPI} i + \omega_{bw} * (netFootprint(G, F, DPI) - netFootprint(G, F, V)) \quad (1)$$

Algorithm 1 Pseudocode for the greedy placement algorithm.

function GREEDY PLACEMENT (Topology graph: $G(V, E)$, List of traffic flows: F)
  initialization of the variables
  $fit_{min} = \infty$ ▷ best fitness value
  $F_u = F$ ▷ list of unallocated flows
  $DPI = \{\}$ ▷ list of selected nodes

  $dpi \leftarrow$ the node with the highest centrality in (G, F_u)
  while $centrality_{dpi} > 0$ and $dpi \notin DPI$ **do**
    Add dpi to DPI
    if $fitness(G, F_u, DPI) < fit_{min}$ **then**
      $fit_{min} = fitness(G, F_u, DPI)$
    else
      break
    end if
    Remove from G the resources used by the flows in F_u whose the shortest path goes through dpi
    Remove from F_u the flows whose the shortest path goes through dpi
    $dpi \leftarrow$ the node with the highest centrality in (G, F_u)
  end while
end function
return DPI

Algorithm 2 Pseudocode for evaluating the network footprint.

function NET FOOTPRINT (Topology graph: $G(V, E)$, List of traffic flows: F, List of DPI nodes: $DPI \subset V$)
  initialization of the variable
  $networkFootprint = 0$ ▷ final result

  for flow f in F **do**
    $shortestPath = \{\}$
    for dpi in DPI **do**
      $shortestPath_{dpi}$ =
      $shortestPath(G, f_{src}, dpi, f_{size})$
      $\cup shortestPath(G, dpi, f_{dest}, f_{size})$
      if $length(shortestPath_{dpi}) < length(shortestPath)$ **then**
        $shortestPath \leftarrow shortestPath_{dpi}$
      end if
    end for
    Remove from G the resources used in $shortestPath$
    Add $f_{size} * length(shortestPath)$ to $networkFootprint$
  end for
end function
return $networkFootprint$

The fitness value is composed of two parts. The first part is equal to the cost of the DPI licenses. The second part corresponds to the cost of the additional used network resources, which is the difference between the network footprint of the DPI placement minus the network footprint of the multi-commodity flow problem (equivalent to all nodes having an activated DPI engine). The network footprint is evaluated using Algorithm 2. For each flow, its shortest path through the nearest DPI is calculated taking into account the available resources. When this shortest path is found, the used resources are removed from the available resources and the next flow is considered. The total network footprint is equal to the sum of the used network resources. This flow allocation problem that we need to solve at each step could have been formulated in a similar way to the linear program presented in Sec. III and solved with a solver. However, we intentionally adopted an iterative shortest path allocation of all the flows for computational efficiency reason. This greedy algorithm is a fair approximation for this NP-hard problem when the traffic is largely lower than the global network capacity [12]. For each flow, it considers the different deployed DPI engines one by one. For each of them, it evaluates the shortest path between the source and the destination that goes through the considered DPI engine using the Constrained Shortest Path First (CSPF) algorithm [13] to take into account the available capacity on the links. We will see in the evaluation section that it has a very limited impact on the calculation of the cost function.

At all steps, if the fitness value, that is its total cost of a vDPI deployment, is higher than the one of the previous DPI placement, the greedy algorithm terminates and returns the previous DPI placement. Otherwise, the unallocated flows that traverse the new DPI are considered as allocated. Their resources are removed from the available resources. The next iteration starts with an updated topology and a reduced quantity of unallocated flows (see Fig. 3).
In addition to an increasing fitness value, the algorithm has two other termination conditions. It terminates when the highest centrality is equal to zero, which means that all the flows have been allocated to a vDPI, or when the new DPI has already been chosen, which means that all nodes have a vDPI.

V. EXPERIMENTAL EVALUATION

To evaluate the performance and the behavior of our placement algorithm, we have tested the heuristic and solved the linear program over different network instances. This section presents the simulation set-up and the experimental results.

A. Methodology

We have implemented the linear program with GLPK and the heuristic using the Java Universal Network/Graph Framework (JUNG)[2].

[2]http://jung.sourceforge.net/

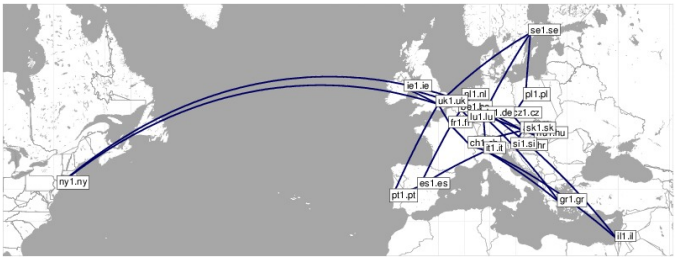
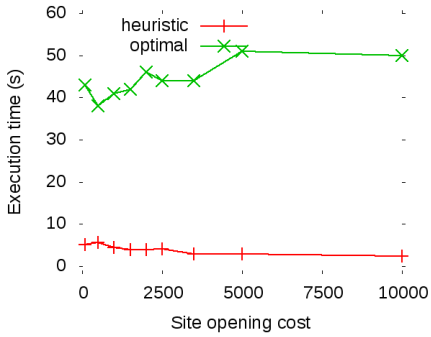
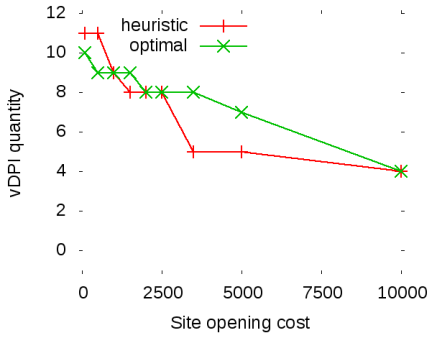


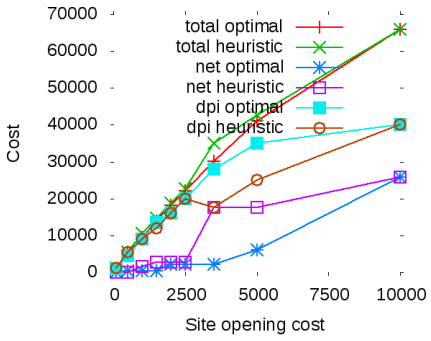
Fig. 4: 22 nodes topology from GEANT in 2006 (courtesy http://sndlib.zib.de).



(a) Execution time.



(b) Quantity of vDPI deployed.



(c) Total cost and its decomposition into used bandwidth and DPI costs.

Fig. 5: Comparison of the linear program and the heuristic on the GEANT dataset.

To evaluate our heuristic under realistic conditions, we used a real world dataset captured in 2006 by Uhlig et al. [14] on GEANT, the high bandwidth pan-European research and education backbone. The data set contains the network topology and traffic matrices measured at different times. However, in order to evaluate our heuristic on larger networks, we also generated random topologies with two different random graph models.

Note that we removed from the integer linear formulation the licence cost for vCPUs to have a fair comparison with the heuristic which does not optimize it. However, from the solutions found, one could derive the necessary number of vCPUs and the associated cost. We used a fix cost of $ 10 per Mb/s for the additional used bandwidth.

Finally, the simulation have been performed on an Intel Core i7-2620M CPU @ 2.70GHz x 4 machine with 7.8 GiB of RAM and a Ubuntu 14.04 64-bit but with mono thread implementations.

B. Evaluation on GEANT

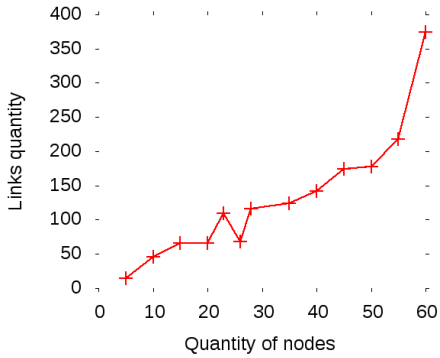
The GEANT dataset contains a topology of 22 nodes, 36 high capacity 40G links, and traffic matrices with 462 demands. The topology is presented on Fig. 4.

Fig. 5 presents the performance evaluation results. We vary the site opening cost, also noted ω_{DPI}, to compare the linear program and the heuristic. On Fig. 5a, we can observe that the heuristic is 9 to 20 times faster than the linear program. The difference increases with the cost ω_{DPI}. Indeed, the execution time of the linear program tends to increase while the execution time of the heuristic tends to decrease. The reason is that the quantity of vDPI diminishes when the site opening cost increases (Fig. 5b), hence the number of iterations realized by the heuristic diminishes. For instance, for a site opening cost of $ 10000, 4 sites have a vDPI deployed. It means that the heuristic performed only 5 iterations while the linear program kept its complexity.

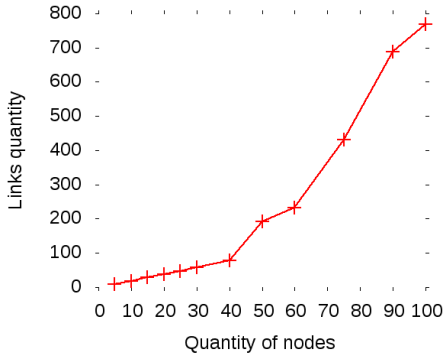
Fig. 5c presents the total cost and its decomposition in network and vDPI license costs both for the linear program and the heuristic. We can observe that the total cost of the heuristic is very close to the optimal total cost, at most 10% higher. The two programs find a tradeoff between the vDPI cost and the used bandwidth cost. For small ω_{DPI} values, the total cost is almost composed of the vDPI cost. When ω_{DPI} increases, the used bandwidth cost tends to take a larger part in the total cost. The solutions found require more and more flows to be derouted from the shortest path to go through the costly vDPI.

C. Evaluation on larger random graphs

To study larger network instances, we generated random graphs with two types of degree distribution: power-law and

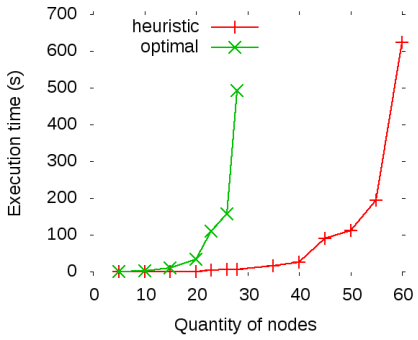


(a) Graph generated with the Flat model.

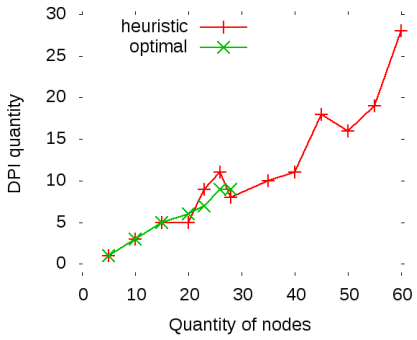


(b) Graph generated with the Barabasi model.

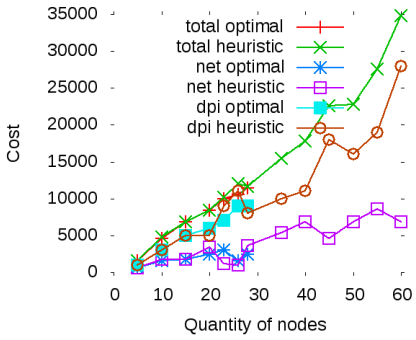
Fig. 6: Number of edges in Flat and Barabasi graphs.



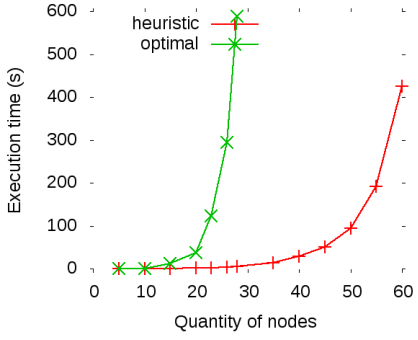
(a) Execution time (Flat, site cost $ 1000).



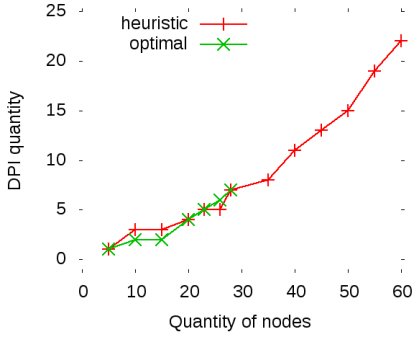
(b) DPI number (Flat, site cost $ 1000).



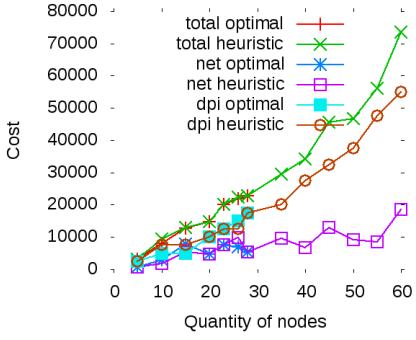
(c) Total cost (Flat, site cost $ 1000).



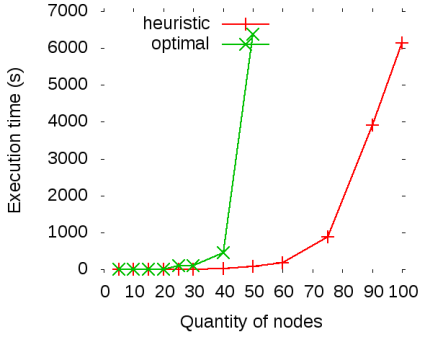
(d) Execution time (Flat, site cost $ 2500).



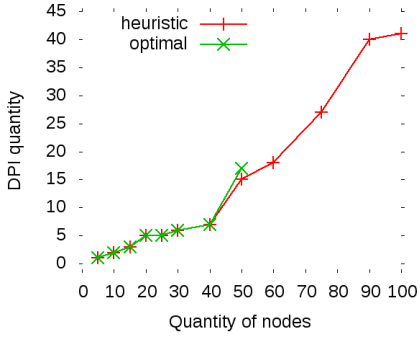
(e) DPI number (Flat, site cost $ 2500).



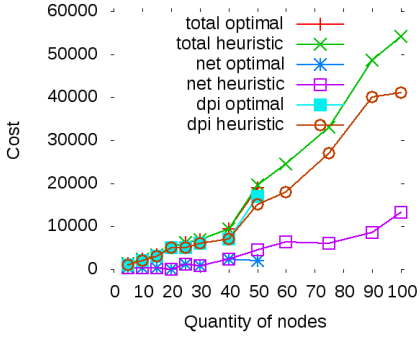
(f) Total cost (Flat, site cost $ 2500).



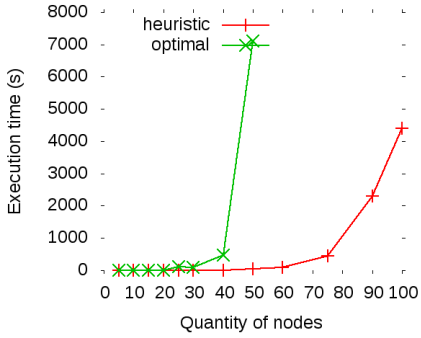
(g) Execution time (Barabasi, site cost $ 1000).



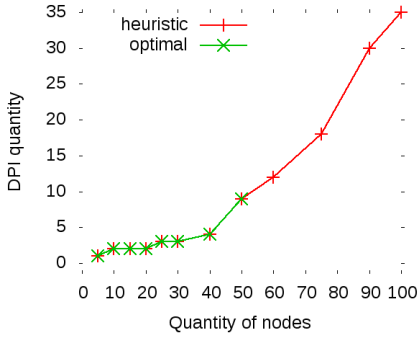
(h) DPI number (Barabasi, site cost $ 1000).



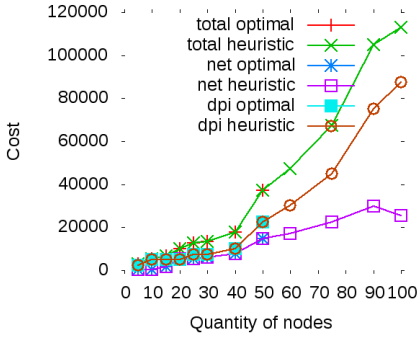
(i) Total cost (Barabasi, site cost $ 1000).



(j) Execution time (Barabasi, site cost $ 2500).



(k) DPI number (Barabasi, site cost $ 2500).



(l) Total cost (Barabasi, site cost $ 2500).

Fig. 7: Evaluation on random graphs with nearly flat (i.e., Flat) and power-law (i.e., Barabasi) node degree distributions. Figures represent for two different site opening costs ($ 1000 and $ 2500) the execution time (right), the number of DPI deployed (middle) and the costs (left).

nearly flat. As there is no widely accepted model for operator networks, these two models are used as case studies to generate graphs with very different structures.

In the power-law case, we choose the well-known Barabasi-Albert [15] model that uses preferential attachment to iteratively build the network. In the case of nearly-flat degree distribution, we randomly add edges between random pairs of nodes until the graph becomes fully connected. In the rest of this section, we refer to the two models as *Barabasi* and *Flat* respectively. We generated sequences of graphs with an increasing number of nodes, but also with a constant graph density (e.g., the percentage of possible edges in the graph) for Barabasi, so that the complexity stays proportional to the number of nodes. We considered a flat traffic matrix where a 10 Mbit/s demand is created in both directions for each pair of nodes. We, of course, acknowledge that it does not represent reality, but aim at capturing an average case.

As presented in Fig. 6a and Fig. 6b, the number of edges increases with the number of nodes. The constraint on the density allows deriving the number of edges that need to be created when adding a new node for Barabasi. For Flat, the function is not monotonous as the graph connection does not happen after a predictable number of iterations.

Fig. 7 presents the results for two different site opening costs, respectively \$ 1000 and \$ 2500. Figures represent the execution time (right), the number of DPI deployed (middle) and the different costs (left). Comparing the optimal and the heuristic, we can observe a good fit in terms of cost and number of DPI deployed. While slight deviations can be observed with Flat, the match is almost perfect for Barabasi. This is due to the fact that the structure of a Barabasi graph does not offer as much route diversity as Flat. Most of the flows pass through high degree nodes, making the placement of DPI probed more constrained.

In terms of execution times, we can observe that the linear program stops solving the problem in reasonable time earlier for Flat than for Barabasi, at around 30 nodes compared to 50 nodes. The size and the complexity of the problem instance are tightly linked with the number of edges and the number of local optimums which are both higher in the Flat case. The same phenomenon can be observed for the heuristic where the execution time is twice smaller for Barabasi on 60 nodes as compared to Flat. The heuristic uses several times Dijkstra which has a complexity of $O(m + n \log n)$ where m is the number of edges and n the number of nodes. Increasing the number of edges increases the execution time. Note also that the problem instances grow as the traffic demands are proportional to the number of node pairs.

Similarly to the results on GEANT, we clearly see that increasing the site opening cost also reduces the execution time of the heuristic as less DPI probes are deployed. However, the execution time for the heuristic significantly increases with the number of nodes but with a different slope for Flat and Barabasi. The structure of Barabasi allows the heuristic to scale more smoothly.

These results clearly show that the network structure and the costs strongly influence time performance. They also show that after a size limit (between 40 to 80 nodes in our case), the execution time increases exponentially due to combinatorial issues. Finally, they demonstrate that the heuristic approximates well the optimal on the smaller graph instances. We conjecture that it remains the case for larger graphs.

VI. Conclusion

In today's IT systems and carrier networks, traffic analysis for traffic management or cyber security require situational awareness that has to be fine-grained, flexible, adaptable and cost optimized. The emergence of new networking technology trends, like NFV and SDN, provide new ways to build traffic monitoring and cyber security tools. DPI engines can be virtualized and deployed on commodity hardware as a piece of software. The deployment a set of vDPIs over a network requires finding the appropriate placement that meets the functional targets (such as the number of inspected flows) and operational cost constraints (license fees or power consumption). We formulated the vDPI placement problem as a cost minimization problem, capturing the different constraints the operator is facing. A placement of vDPIs on the network nodes realizes a trade-off between the constraints. We casted the problem as a multi-commodity flow problem and solved it as an Integer Linear Program (ILP). We also proposed a centrality-based greedy algorithm. Finally, we assessed its validity and it scalability comparing it to the linear program. On the GEANT topology and traffic dataset, we varied the site opening cost and observed that the heuristic is 9 to 20 times faster than the linear program. The two methods find a trade-off between the vDPI cost (e.g. licenses) and the induced network footprint. Furthermore, we applied both the ILP and the heuristic to larger random networks of up to 100 nodes and based on the flat and Barabasi models. The results showed the network structure and the costs strongly influence time performance. They also show that after a size limit (between 40 to 80 nodes in our case), the execution time increases exponentially due to combinatorial issues.

The current development of NFV infrastructures in carrier networks is a technology disruption creating a tremendous amount of flexibility for service providers. However, cost-efficient optimization and management procedures have to be developed to fully benefit from the power of such architecture. This paper is a first step in this direction. Future works along these lines include the use of optimization mathematical frameworks to increase the scalability and the robustness of the approximation algorithm. We also envision taking into consideration the delay constraints on the flows since redirections and flow analysis may increase it. We also plan to enrich our model to integrate a more accurate performance model of DPI probes with regards to the rules that they execute. Finally, we intend to study online system optimization where the deployment of additional vDPI engines, or their shutdown, is incremental based on an existing setup.

VII. Acknowledgments

This work is partly funded by the French ANR under the DISCO project (ANR-13-INFR-013) and the REFLEXION project (ANR-14-CE28-0019).

References

[1] I. Research, "Service Provider Deep Packet Inspection product report," 2014.

[2] M. Chiosi *et al.*, "Network Functions Virtualization - An Introduction, Benefits, Enablers, Challenges and Call for Action," ETSI NFV, Oct. 2012.

[3] O. Biran, A. Corradi, M. Fanelli, L. Foschini, A. Nus, D. Raz, and E. Silvera, "A stable network-aware VM placement for cloud systems," in *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, 2012.

[4] J. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang, "Joint VM placement and routing for data center traffic engineering," in *INFOCOM, 2012 Proceedings IEEE*, 2012.

[5] G. Lu, R. Miao, Y. Xiong, and C. Guo, "Using CPU as a traffic co-processing unit in commodity switches," in *Proceedings of the first workshop on Hot topics in software defined networks (HotSDN)*. ACM, 2012, pp. 31–36.

[6] F. Gringoli, A. Este, and L. Salgarelli, "MTCLASS: Traffic classification on high-speed links with commodity hardware," in *IEEE International Conference on Communications (ICC)*, 2012, pp. 1177–1182.

[7] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," *CoRR*, vol. abs/1406.1058, 2014. [Online]. Available: http://arxiv.org/abs/1406.1058

[8] M. Bouet, J. Leguay, and V. Conan, "Cost-based placement of virtualized deep packet inspection functions in sdn," in *Military Communications Conference, MILCOM 2013 - 2013 IEEE*, 2013.

[9] C. Chaudet, E. Fleury, I. G. Lassous, H. Rivano, and M.-E. Voge, "Optimal positioning of active and passive monitoring devices," in *Proceedings of the 2005 ACM Conference on Emerging Network Experiment and Technology (CoNEXT)*, 2005.

[10] S. Even, A. Itai, and A. Shamir, "On the complexity of timetable and multi-commodity flow problems," in *FOCS*. IEEE Computer Society, 1975, pp. 184–193.

[11] P. Networks, "Network Application Visibility Library (NAVL)." [Online]. Available: http://files.proceranetworks.com/resources/NAVL%201014.pdf

[12] W. Xiao, B.-H. Soong, C. Law, and Y.-L. Guan, "Evaluation of heuristic path selection algorithms for multi-constrained QoS routing," in *IEEE International Conference on Networking, Sensing and Control*, 2004.

[13] C. Gen-Huey and H. Yung-Chen, "Algorithms for the constrained quickest path problem and the enumeration of quickest paths," *Computers & operations research*, vol. 21, no. 2, pp. 113–118, 1994.

[14] J. L. S. B. Steve Uhlig, Bruno Quoitin, "Providing public intradomain traffic matrices to the research community," *ACM SIGCOMM Computer Communication Rev*, vol. 36, no. 1, 2006.

[15] A.-L. Barabasi and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.