Optimal Model for the Controller Placement Problem in Software Defined Networks

Afrim Sallahi and Marc St-Hilaire

***Abstract*—In this letter, we propose a mathematical model for the controller placement problem in Software Defined Networks (SDN). More precisely, given a set of switches that must be managed by the controller(s), the model simultaneously determines the optimal number, location, and type of controller(s) as well as the interconnections between all the network elements. The goal of the model is to minimize the cost of the network while considering different constraints. The simulation results show that the model can be used to plan small scale SDN. When trying to solve larger instances of the problem, the solver is taking too much time and also running out of memory. The proposed model could be used by various enterprises and cloud-based networks to start integrating SDN or plan a new SDN.**

***Index Terms*—Controller placement, mathematical model, planning, software defined network (SDN).**

I. Introduction

ADVANCEMENT in personal communication devices and the demand for more data have challenged the computer networking implementation that is used today. Software Defined Network (SDN) brings abstractions to the software layer of networking by providing methods of accessing equipment through open interfaces. This makes it easier for innovation in the networking layers. Generally, the two most basic functions that networks perform are control and switching. The former makes decisions where information goes and the latter is responsible for moving information hop-by-hop. In SDN, the switching plane is kept the same but the control plane is abstracted and moved to a central location within the network referred to as "controller".

Having a single controller within a network is advantageous because it provides a single view of the whole network and decisions (routing, security, etc.) are made based on that view. However, having a single controller also has some disadvantages related to performance and scalability. A solution to these problems is to use multiple controllers. Although there is not a lot of research on the placement of multiple controllers, we believe this is an interesting avenue.

Whether the SDN contains a single or multiple controllers, the placement of the controller(s) will have an impact on the performance and the cost of the network. This is exactly what was shown in [1] where different locations were providing different delays. To that end, it is important to have complete models that will find the optimal location(s) of the controller(s) and interconnect the network elements for optimal performance and minimum cost.

In this letter, we propose a mathematical model to plan SDN. Given a set of switches that must be managed by the controller(s), the model simultaneously determines the optimal number, the location, and the type of controller(s) as well as the interconnections between all the network elements. The goal of the model is to minimize the cost of the SDN while considering different constraints such as the capacity of the controller(s), the latency of path setup and so on.

The rest of this letter is organized as follows. Section II reviews the related work on the controller placement problem in SDN. Then, the proposed model is introduced in Section III followed by the simulation results in Section IV. Finally, Section V concludes the letter.

II. Related Work

The planning problem is not new in SDN. In fact, a few papers are already dealing with various planning issues. However, they all tackle different aspects compared to what is proposed in this letter. For example, in [1], the authors tackled the issues of where and how many controllers to deploy. The authors used the k-median and the k-center algorithms and unfortunately, decisions are strictly based on the latency (average-case latency and worst-case latency) between the network elements. It is also important to note that all the optimal placements were obtained by using a brute force approach where all potential locations were evaluated.

A framework that automatically adjusts the switch to controller connectivity as the network progresses through different stages is proposed in [2]. The basic idea here is to enable or disable links live on a network to a controller from a pool of switches. They do consider the flow setup latency involved in setting up paths from controllers to switches—something we consider in our solution as well. Using this framework, we feel that controllers and links may stay idle and become active or inactive as they are needed throughout the network. Our work differs as we plan the switches that are needed instead of staying idle.

Similarly, proposals to solve the problem using brute force, greedy algorithm and heuristic are proposed in [3]. The parameter used to solve the problem is the control path of the network. The idea is that the fewer control paths that fail, the less impact it has on the network. Therefore, controllers are placed based on the control path of the network.

In a related area to controller placement, the authors in [4] take a look at the lessons learned from the distributed computing and use local algorithms to propose a method for separating the network structure from the traffic patterns. The authors look at the control plane using the flat and hierarchical

Manuscript received September 24, 2014; accepted October 26, 2014. Date of publication November 14, 2014; date of current version January 7, 2015. The associate editor coordinating the review of this paper and approving it for publication was M. Naeini.

The authors are with the Department of Systems and Computer Engineering, Carleton University, Ottawa, ON K1S 5B6, Canada (e-mail: sallahi@sce.carleton.ca; marc_st_hilaire@carleton.ca).

Digital Object Identifier 10.1109/LCOMM.2014.2371014

layer. Both types of controller layout have their advantages and disadvantages and we will only consider the flat layout model because it reduces latency and keeps control traffic local [1]. Reducing latency is something we also consider in our model.

The model we propose in this letter is different from what has been proposed thus far. In fact, we propose a mathematical formulation for the controller placement problem in SDN. The model looks at the optimal planning for the whole SDN which simultaneously determines the optimal number, the location, and the type of controller(s) as well as the interconnections between the network elements. Information such as link cost, equipment cost, different controller capabilities, latency of path setup, and network traffic patterns are considered. The next section introduces the mathematical model.

III. Problem Formulation

To formulate the mathematical model, we assume the following information is known:

- The location of all the switches in the network and the amount of traffic that must go to the controller from each switch.
- The length and the bandwidth available for each link type to connect the switches and the controllers.
- The characteristics of the different types of controllers. Each type of controller has a cost ($), a number of physical ports available, a maximum number of requests it can handle per second and the number of available controllers of each type.
- The maximum link setup latency allowed for switch to controller communications.

Based on this information, we define the following notation.

A. Notation

1) Sets:

- $S = \{s_1, s_2, \ldots\}$, the set of switches that are present in the network.
  - — σ^s, the number of packets that do not match on the switch's $(s \in S)$ lookup table and that are sent to the future connected controller.
- $C = \{c_1, c_2, \ldots\}$, the set of controllers that can be installed.
  - — α^c, the number of ports available for the controller type $c \in C$.
  - — μ^c, the number of packets/second a controller of type $c \in C$ can process.
  - — κ^c, the price (in $) for a controller of type $c \in C$.
  - — φ^c, the number of available controllers of type $c \in C$.
- $L = \{l_1, l_2, \ldots\}$, the set of possible link types that can be used to connect controllers and switches.
  - — ω^l, the bandwidth (in Mbps) of a link of type $l \in L$.
  - — ϕ^l, the price (in $/meter) of a link of type $l \in L$.
- P, the set of possible locations to install the controllers.

2) Constants:

- β, the packet size (in bytes) for each packet that is sent to the controller.
- γ, the maximum delay (in milliseconds) allowed in the network for flow-setup.
- δ, the average time (in milliseconds) taken to process a packet by any controller.

3) Decision Variables:

- x_{cp}, a 0–1 variable such that $x_{cp} = 1$ if and only if a controller of type $c \in C$ is installed at location $p \in P$.
- v^l_{sp}, a 0–1 variable such that $v^l_{sp} = 1$ if and only if a link of type $l \in L$ is installed between switch $s \in S$ and controller installed at location $p \in P$.
- z^l_{pq}, a 0–1 variable such that $z^l_{pq} = 1$ if and only if location $p \in P$ is connected to location $q \in P$ with a link type $l \in L$.

B. Cost Function

The objective is to minimize the cost to plan the network. This includes the cost of installing controllers $C_c(x)$, the cost of linking the controllers to the switches $C_l(v)$, and the cost for linking the controllers together $C_t(z)$. The function dist(a, b) calculates the distance between point a and point b.

$$C_c(x) = \sum_{c \in C} \kappa^c \sum_{p \in P} x_{cp} \tag{1}$$

$$C_l(v) = \sum_{l \in L} \phi^l \sum_{s \in S} \sum_{p \in P} \texttt{dist}(s, p) v^l_{sp} \tag{2}$$

$$C_t(z) = \sum_{l \in L} \phi^l \sum_{\substack{q \in P \\ q < p}} \sum_{p \in P} \texttt{dist}(p, q) z^l_{pq} \tag{3}$$

C. The Model

The Controller Placement Problem, denoted (CPP), can be modelled as follows.

CPP :

$$\texttt{Minimize}\,(C_c(x) + C_l(v) + C_t(z)) \tag{4}$$

Subject to the following constrains:

- Uniqueness constraint for controllers.

$$\sum_{c \in C} x_{cp} \leq 1 \quad (p \in P) \tag{5}$$

- Make sure that the number of connected switches and controllers to one specific controller is smaller than the number of ports on the controller.

$$\sum_{q \in P} \sum_{l \in L} \left(z^l_{pq} + z^l_{qp}\right) + \sum_{s \in S} \sum_{l \in L} v^l_{sp} \leq \sum_{c \in C} \alpha^c x_{cp} (p \in P) \tag{6}$$

- Make sure that exactly one link (of any type $l \in L$) is installed between a given switch and the controllers.

$$\sum_{l \in L} \sum_{p \in P} v^l_{sp} = 1 \quad (s \in S) \tag{7}$$

TABLE I
DIFFERENT CONTROLLER FEATURES

	Type 1	Type 2	Type 3	Type 4
Cost (κ^c)	\$1,200	\$2,500	\$6,500	\$12,000
Ports (α^c)	8	32	64	128
Processing (μ^c)	2,500	4,000	8,000	15,000
# Controllers (φ^c)	20	15	10	6

- Make sure that the number of packets sent by the switches can be processed by the controller.

$$\sum_{l\in L}\sum_{s\in S}\sigma^s v^l_{sp} \le \sum_{c\in C}\mu^c x_{cp} \quad (p \in P) \tag{8}$$

- Make sure that the inventory of each controller is not exceeded.

$$\sum_{p\in P} x_{cp} \le \varphi^c \quad (c \in C) \tag{9}$$

- Make sure that the link that is chosen between the controller and the switch can handle the bandwidth needed by the switch. $g(a,b)$ is a function that takes the number of packets per second (a) and the packet length in bytes (b) and converts it to bytes per second. $f(a)$ is a function that converts a value in Mbps or Gbps (a) to bytes per second.

$$g(\sigma^s,\beta) \le \sum_{l\in L} f(\omega^l)v^l_{sp} \quad (s \in S, p \in P) \tag{10}$$

- Keep the round trip flow-setup latencies for unmatched flows in each of the switches below or equal to the maximum allowable delay. $\mathrm{Tran}(v)$, $\mathrm{Prop}(x,v)$, and $\mathrm{Proc}(x)$ return the transmission delay, the propagation delay and the processing delay respectively.

$$2\mathtt{Tran}(v) + 2\mathtt{Prop}(x,v) + \mathtt{Proc}(x) \le \gamma \tag{11}$$

- Inter-controller topology connectivity. In this paper, we consider a full mesh topology although other topologies could be used [5].

$$\sum_{c\in C} x_{cq} + \sum_{c\in C} x_{cp} \le \sum_{l\in L} z^l_{pq} + 1 \quad (q < p, q \in P, p \in P) \tag{12}$$

- Ensure valid assignment values.

$$x_{cp} \in \{0,1\} \quad (c \in C, p \in P) \tag{13}$$

$$v^l_{sp} \in \{0,1\} \quad (l \in L, s \in S, p \in P) \tag{14}$$

$$z^l_{pq} \in \{0,1\} \quad (l \in L, p \in P, q \in P) \tag{15}$$

IV. SIMULATION

In this section, we study the performance of the model proposed in Section III. Tables I, II, and III show the input information that was used for the simulation. Using this information, the linear programming model (i.e., the lp file) was first generated with the help of the Python programming language and then sent to the optimizer.

TABLE II
DIFFERENT LINK FEATURES

	Type 1	Type 2	Type 3
Cost/meter (ϕ^l)	\$0.25	\$0.63	\$29
Bandwidth (ω^l)	100 Mbps	1 Gbps	10 Gbps

TABLE III
OTHER INPUT TO THE MODEL

	Value
Packet Size (β)	150 Bytes
Maximum Roundtrip Latency (γ)	250 ms
Average Controller Processing (δ)	0.001 ms
Unmatched Switch Lookups (σ^s)	Random from 100 to 999

TABLE IV
SIMULATION RESULTS OBTAINED FROM CPLEX FOR 27 DIFFERENT PROBLEM SIZES

	$\lvert S\rvert$	$\lvert P\rvert$	$\lvert P'\rvert$	$\lvert L'\rvert$	Packets	Cost (\$)	CPU (s)
1	10	10	3	13	5,783	4,243	<1
2	10	15	3	12	5,019	3,841	3
3	10	20	3	13	5,169	4,086	6
4	20	10	5	29	11,209	8,138	3
5	20	15	4	26	10,181	7,512	31
6	20	20	4	27	10,857	7,774	83
7	30	10	5	41	16,391	12,747	18
8	30	15	6	45	17,335	13,138	155
9	30	20	6	42	16,734	12,727	3,087
10	40	10	6	55	23,192	19,195	49
11	40	15	6	55	21,044	16,980	774
12	40	20	7	57	23,852	19,007	5,401
13	50	10	6	66	27,938	23,802	66
14	50	15	7	69	27,554	22,808	4,284
15	50	20	7	71	27,285	22,040	1,146
16	75	10	8	101	42,758	38,073	292
17	75	15	8	102	40,548	35,277	3,365
18	75	20	(8)	(98)	(40,994)	(35,484)	(58,657)
19	100	10	9	133	55,348	49,489	498
20	100	15	11	151	55,206	48,214	14,666
21	100	20	11	156	54,241	47,172	18,206
22	150	10	8	173	80,962	73,883	2,623
23	150	15	(12)	(211)	(81,141)	(73,102)	(46,219)
24	150	20	(12)	(216)	(82,105)	(73,981)	(108,000)
25	200	10	10	245	109,956	103,888	80
26	200	15	(12)	(262)	(109,908)	(100,128)	(38,475)
27	200	20	(11)	(253)	(105,541)	(96,483)	(108,000)

To evaluate the model, the number of switches in set S (i.e., $|S|$) is varied from 10, 20, 30, 40, 50, 75, 100, 150, and 200. For each value of $|S|$, the number of possible locations to install the controllers (i.e., $|P|$) is varied from 10, 15, and 20. These 27 combinations are generated in an area of 1 km $\times$ 1 km and will allow us to see how the time and the cost of the optimization model are affected.

The simulations are run on a PC that has 2 Intel Xeon X5675 processors running at 3.07 GHz with total memory of 96 GB. All the results are obtained from the CPLEX optimizer 12.5 [6]. A time limit of 30 hours was set and the optimizer was only allowed to use a single thread. All other parameters were set to their default values.

The results are shown in Table IV. It is important to note that each line in the table represents the average results obtained over 4 different instances of the problem. The table is formatted as follows. The first column represents the problem number. The second and third columns are the number of switches (i.e., $|S|$) and number of maximum possible placement locations for the controllers (i.e., $|P|$). Column labeled $|P'|$ is the average

number of controllers (rounded up) that have been installed by the solver. Similarly, column denoted $|L'|$ is the average number of links (rounded up) installed for the whole network. If more than one controller exists (i.e., if $|P'| > 1$), then there are a total of

$$|L'| = \frac{|P'|\,(|P'|-1)}{2} + |S| \tag{16}$$

links for each scenario. For example, in problem 1, three controllers are installed meaning that $3(3-1)/2$ links are required to connect the controllers using a full mesh topology. Then, 10 extra links are required to connect the switches to controllers for a total of 13 links as indicated in the table. The column labeled "Packets" represents the total number of packets for the network. Finally, the last two columns labeled "Cost" and "CPU" are the cost of the solution ($) and the time taken (in seconds) by the solver to find the optimal solution. It is important to note that when the values are in parenthesis (such as problems 18, 23, 24, 26, 27), it means that at least one of the 4 instances exceeded the time limit. As a result, these value may not represent the optimal solution but rather the best solution found when the time limit was reached.

If we start by analyzing the CPU time taken to find the solutions, we can see that for a fixed value of $|S|$, the time generally increases as the number of potential locations for the controllers is increased. However, this is not always the case. For example, problem 15 takes less time to solve than problem 14 even though it has more potential locations for the controllers. A reason why the optimizer may take less time for a larger input size is because the optimizer uses the branch and bound algorithm to try different combinations and heuristics are used to shorten the combinational space by finding the solution to the relaxation of the current node [7]. Similarly, we can also see that for a fixed value of $|P|$, the time generally increases as the number of switches is increased. Over the 108 different instances that were analyzed, 11 instances ($\sim$10%) were still not solved before the time limit was reached. This is an indication that only small size problems can be computed within a reasonable amount time. We also computed the 95% confidence interval and the largest interval occurs at problem 18 with an interval of $\pm$27,019 seconds. This wide interval occurs because some of the instances reached the time limit and some others were "quickly" solved.

In terms of solution cost, we can make two observations. First, we can see that for a fixed value of $|P|$, the cost increases linearly with an increase in the number of switches. This was somehow expected because the network has to accommodate more switches therefore requiring more controllers and more links. The second observation is that for a fixed value of $|S|$, when the number of potential placement locations increases, we can see a small decrease (up to a certain extent) in the cost of the solution. This can be explained by the fact that the solver has more options to select better locations for the controllers thus reducing the overall cost. We also computed the 95% confidence interval and the largest interval occurs at problem 26 with an interval of $\pm$\$8,789. Overall, the results show narrow confidence intervals meaning that the costs reported by CPLEX are reliable.

V. Conclusion

In this letter, we proposed a new mathematical model for the controller placement problem in software defined networks. Given a set of switches that must be managed by the controller(s), the model simultaneously determines the optimal number, location, and type of controller(s) as well as the interconnections between all the network elements to minimize the cost. The proposed model can be used in existing networks that plan to migrate to a SDN architecture or plan brand new SDN networks.

Since we try to optimize an NP-hard problem, our simulation results show that only small size problems can be optimized within a reasonable amount of time. In fact, approximately 10% of the problems cannot be solved within the time limit of 30 hours. To gain performance, one option could be to derive approximate algorithms. Therefore, future work involves the development of approximate algorithms and compare them to the optimal model proposed in this letter.

References

[1] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proc. 1st Workshop Hot Topics Softw. Defined Netw.*, 2012, pp. 7–12.

[2] M. F. Bari *et al.*, "Dynamic controller provisioning in software defined networks," in *Proc. 9th Int. CNSM*, 2013, pp. 18–25.

[3] Y.-N. Hu, W.-D. Wang, X.-Y. Gong, X.-R. Que, and S.-D. Cheng, "On the placement of controllers in software-defined networks," *J. China Univ. Posts Telecommun.*, vol. 19, no. S2, pp. 92–171, Oct. 2012.

[4] S. Schmid and J. Suomela, "Exploiting locality in distributed sdn control," in *Proc. 2nd ACM SIGCOMM Workshop HotSDN*, New York, NY, USA, 2013, pp. 121–126.

[5] S. Chamberland, M. St-Hilaire, and S. Pierre, "An analysis of different colocated router network topologies within a pop in ip networks," in *Proc. IEEE CCECE*, 2003, vol. 2, pp. 733–736.

[6] CPLEX: IBM's Linear Programming Solver. [Online]. Available: http://www.ilog.com/product/cplex/

[7] *IBM ILOG CPLEX V12.1—User's Manual for CPLEX*, IBM, 2009.