Reliability-aware Controller Placement for Software-Defined Networks

Yannan Hu, Wang Wendong, Xiangyang Gong, Xirong Que, and Cheng Shiduan
State Key Laboratory of Networking and Switching Technology
Beijing University of Posts and Telecommunications
Beijing, China
{huyannan, wdwang, xygong, rongqx, chsd}@bupt.edu.cn

***Abstract*—The Software-Defined Network (SDN) approach decouples control and forwarding planes. Such separation introduces reliability design issues of the SDN control network, since disconnection between the control and forwarding planes may lead to severe packet loss and performance degradation. This paper addresses the problem of placing controllers in SDNs, so as to maximize the reliability of control networks. After presenting a metric to characterize the reliability of SDN control networks, several placement algorithms are developed. We evaluate these algorithms and further quantify the impact of controller number on the reliability of control networks using real topologies. Our approach can significantly improve the reliability of SDN control networks without introducing unacceptable latencies.**

I. Introduction

Unlike traditional networks, where both control and forwarding planes are highly integrated on the same boxes, the Software Defined Network (SDN) architecture decouples control and forwarding planes. Such separation is realized by moving the network intelligence onto one or more external servers, called controllers, which make up a network-wide logically centralized control plane that oversees a set of dumb, and simply forwarding elements [1].

Generally speaking, the SDN control plane consists of three parts [2]: control platform, which handles state distribution; control applications, which are developed upon a programmatic interface that the control platform provides; and control network (or connectivity infrastructure), which is used for propagating events to packet-forwarding devices or between multiple controllers. Although there have been a range of previous works that focus on control platform and control applications, so far, design issues of the control network, which is also another essential part of the SDN control plane, have not been well investigated.

Since network failures could cause disconnections between the control and data planes, and further disable some of the switches, it is of great importance to improve the reliability of SDN control networks. As a result, in this paper, we are interested in finding the answers to the following questions: given a physical network and the failure probability of each network component (e.g. links and switches), how many controllers are needed and how to place them such that a pre-defined reliability objective is optimized.

To find the most reliable controller placements for SDNs, we propose a novel metric that reflects the reliability of the SDN control network, called expected percentage of control path loss, where the control paths are defined as the set of routes that are used for communications between switches and their controllers, as well as between controllers themselves. The optimization target is then to minimize the expected percentage of control path loss. To maximize the reliability of SDN control networks, we develop several placement algorithms that automate the controller placement decision. These algorithms and the benefits of reliable-aware controller placement are evaluated through extensive simulations using real topologies. We also verify our proposal by quantifying the tradeoffs between metrics, e.g., reliability and latencies.

The remainder of this paper is organized as follows. We review related work in the next section. Section III proposes our reliability metric. Section IV presents four placement algorithms. Section V covers the simulations and our findings. Section VI concludes the paper.

II. Related Work

There have been extensive studies on resiliency of traditional networks, such as [3] and [4]. However, the models considered in these works can not be applied to the SDN architecture, since these models either assume that control and data packets are equally affected when a failure happens, or assume that each node in the network functions independently and consists of both control and forwarding planes.

There have been several research efforts on designing the control plane of SDNs. To assuage the scalability concerns raised when building networks with a logically centralized control plane, one of the practices is to work on distributed implementations of the controllers (also valuable for fault tolerance), which include Onix [2] and HyperFlow [5]. Although these implementations use multiple controller instances, how to design a control network such that a given objective can be satisfied is still an open question. We tackle this problem from the reliability perspective in this paper.

The most relevant work can be found in [6]. The authors motivated the controller placement problem and examined the impacts of placements on average and worst-case propagation latencies on real topologies. They find that most networks show diminishing returns for each added controller, and one controller location often suffices. While propagation latency is certainly a significant design metric, we argue that reliability design is also an essential part for operational SDNs. This paper optimizes the reliability of SDN control network, and further qualifies the tradeoffs between reliability and latencies.

978-3-901882-50-0

III. Reliable-aware Controller Placement

In SDNs, switches communicate with their controller via standard TLS or TCP connections. When multiple controllers are deployed, communications between these controllers are also required to achieve global consistency of network state [2]. In this paper, we assume that the above communication traffic uses existing connections between switches. We define control paths as the set of routes that are used for communications between switches and their controllers, as well as between controllers. If we view each control path as a logical link, communication packets are actually transmitted over an overlay network, named control network, above the physical network.

To analyze reliability of the SDN control network, we define a reliability metric similar to what has been proposed in [7]. Our reliability metric is defined as the *expected percentage of control path loss*, where the control path loss is the number of broken control paths due to network failures. The optimization target is then to minimize the expected percentage of control path loss.

Ideally, if failures of different control paths are independent, our metric can be calculated directly. However, two overlaid control paths may share a common physical link or switch, and thus control paths fail in a dependent fashion. Several dependent failure models have been proposed in the past for studying network reliability. In this paper, we adopt a commonly-used dependent failure model, the cause-based reliability analysis model [8], in SDN control networks. The basic idea is that failures of the control network components, e.g., control paths, have underlying physical causes that can be explicitly identified and are statistically independent.

We omit the analysis details due to space limitation. However, two things are important to mention: (1) In general, it is impractical to cover all failure causes that affect the states of control paths. However, in practice, we can get a satisfying statistical coverage by only analyzing the failure causes that could most likely happen. According to [9], [10], the possibility that multiple physical components fail simultaneously in one administrative domain is extremely small, and most of the failures only involve single network component in IP networks. Therefore, we only analyze the failure scenarios, in which we assume at most one physical component fails at any time in this work. (2) Because one goal of this paper is to understand how much reliability gains we can achieve by barely placing controllers carefully in the network, we ignore the benefit of any control path protection mechanisms in this work. In another word, when a physical component fails, we assume that the control paths traverse that physical component fail as well. These assumptions will be relaxed in future work.

For a network graph $G(V, E)$, where V is the set of nodes, E the set of links, let n be the number of nodes. Let k denote the number of controllers to be placed in the network. To reduce the propagation delay of control traffic, for any controller placement, we connect each switch to its nearest controller using the shortest path (in terms of propagation delay). Let m denote the total number of control paths in the control network. For each physical network component $l \in V \cup E$, define p_l to be the failure probability of component l. Without loss of generality, let us assume that the failure of l breaks d_l control paths. Then, the expected percentage of control path loss δ is

$$\delta = \frac{1}{m} \sum_{l \in V \cup E} d_l p_l \tag{1}$$

Note that m varies with the forms that SDN control networks take. In this paper, we assume hierarchical control network, meaning that multiple controllers are connected in a full mesh, which connect to forwarding nodes below. As a result, $m = n + k(k-1)/2$.

IV. Placement Algorithms

We have proved that the reliability-aware controller placement problem is NP-hard by reducing it from a known NP-hard problem – the minimum k-median problem [11]. Therefore, some approximation algorithms must be used for a sub-optimal solution. To solve the reliability-aware controller placement problem we present a number of algorithms to pick the placement of k controllers.

A. Random Placement

Under random placement, the algorithm randomly chooses k locations among $|V|$ potential sites.

B. l-w-greedy

Suppose we need to place k controllers among $|V|$ potential locations, the algorithm first generates a list of the potential locations, denoted as S_r, which is ranked increasingly based on on failure probabilities of the switches. Then, it chooses one location at a time, from the first $w|V|(0 < w \leq 1)$ elements of S_r, named candidate locations, for hosting controllers. For $l = 0$, in the first iteration, the algorithm computes the cost associated with each candidate location under the assumption that connections from all switches converge at that location, and pick the location that yields the lowest value. In the second iteration, the algorithm searches for a second controller location from the candidate locations which, in conjunction with the location already picked, yields the lowest cost. The algorithm iterate until k controllers have been chosen. For any other l value, after l controllers have been placed, the algorithm allows for l steps backtracking in each of the subsequent iterations: it checks all possible combinations of removing l of already placed controllers and replacing them with $l + 1$ new controllers. Similarly, the most reliable placement is selected as the starting point for the next iteration.

C. Simulated annealing

Simulated annealing (SA) is a generic probabilistic meta-algorithm. While SA is a known technique, our contribution lies in optimizing the configurations of the algorithm, which reduce the search space and ensure rapid convergence to a near optimal placement. (1) *Initial state*: The initial solution is obtained by placing k controllers at the k most reliable locations. (2) *Initial temperature*: The initial temperature T_0 should be a large value so that almost any neighbor solution is acceptable. We define P_0 as the acceptance probability in the first k iterations, and $\triangle_0$ as the cost difference between the best solution and the worst solution obtained in Y executions of the random placement. Then the initial temperature T_0 can

be computed by $-|\triangle_0|/\ln P_0$. In our implementation, we set P_0 to 0.999 and Y to 1000. (3) *Neighborhood structure*: Let $P(M)$ denote a possible placement of k controllers, x_c a controller location of $P(M)$ and x_k a location of $(V-P(M))$. Instead of choosing x_c and x_k randomly, for each x_c in $P(M)$, the best exchange x_k in $(V - P(M))$ is identified such that $\triangle_{ck} = min_{j\in(V-P(M))}\triangle_{cj}$, where $\triangle_{ij}$ is the reduction in the objection function that is obtained when $x_i \in P(M)$ is replaced by $x_j \in (V - P(M))$. The algorithm performs in the usual way once the pair of location is determined. A cycle of the algorithm is completed when all x_cs in $P(M)$ are examined. (4) *Temperature function*: The temperature decreases exponentially, i.e., $T_{new} = \alpha T_{old}$. We empirically determined that α = 0.75 gives the best results.

D. Brute Force

The brute force algorithm is implemented for evaluation purpose. It generates all the possible combinations of k controllers in every potential location. It measures the cost of each of the placements and returns the best one. Since this solution approach is exhaustive, it obtains the optimal result.

V. Evaluation

This section summarizes our simulation results based on real topologies, aiming to understand the benefits of reliability-aware controller placement. We first evaluate the performance of the various placement algorithms discussed in Section IV using the Internet2 OS3E topology [12]. To characterize reliability performance against the number of controllers, we then expand our analysis to Rocketfuel topologies [13]. Finally, we analyze the controller placement tradeoffs between reliability and latencies.

A. Algorithm Comparison

To compare the performance of the algorithms, we use the *relative performance* of the algorithms as a metric, which is defined as the ratio between δ of the feasible solution found by the algorithms to the one that is achieved by the brute force algorithm. A value of 1.0 implies the algorithm finds an optimal solution. The OS3E topology is used. In our experiments, we generate different settings of network component failure probabilities, which stand for different failure scenarios. For each of the settings, we run a set of simulations. In each set of the simulation, we first pick k, the number of controllers. For the given k, we run one simulation for the following algorithms: random, 0-1-greedy, 0-0.8-greedy, 1-1-greedy, 1-0.8-greedy, 2-1-greedy, 2-0.8-greedy, SA and brute force. Then, we repeat all simulations for the next k.

Due to lack of space, we only report the results under the setting where the switch failure probability is randomly generated from interval [0.015, 0.025] and the link failure probability is generated from [0.035, 0.045]. Actually, comparison results are insensitive to variations in the failure probability. Figure 1 shows the cumulative distribution (CDF) of the relative performance of the algorithms. As we can see, the simulated annealing, 2-1-greedy and 1-1-greedy are the three best algorithms. Among these three, the SA performs the best, and the 2-1-greedy always finds better placements than the 1-1-greedy. Regarding other algorithms, the 0-1-greedy, 0-0.8-greedy, 1-0.8-greedy and 2-0.8-greedy have the performances in between the three bests and the random placement, which has the worst performance.

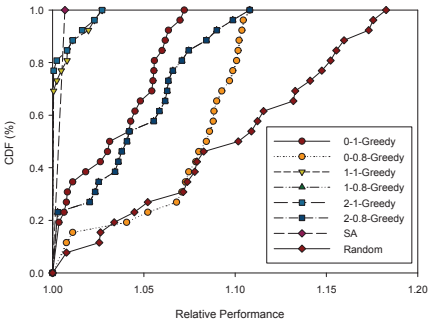


Fig. 1. The CDF of relative performance of the placement algorithms on OS3E topology.

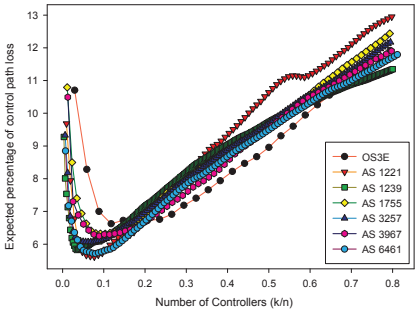
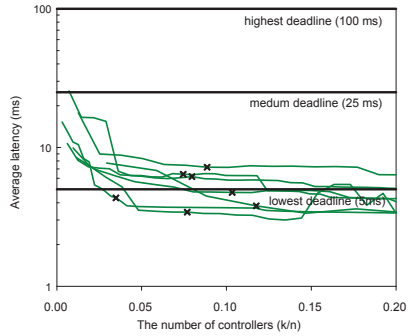


Fig. 2. Reliability metric vs. the number of controllers on different topologies. Smaller is better.

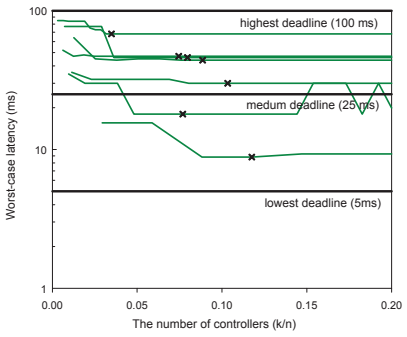
B. Impact of controller number on reliability

We now present the results of our simulations on the OS3E and Rocketfuel topologies, aiming to find the answer to the following question: if the controllers are carefully placed, how many controllers should we use in order to maximize reliability. In this set of simulations, we consider only the SA algorithm. For all the topologies, we set the failure probabilities of each switch and each link to the same values, 0.01 and 0.02, respectively.

Figure 2 depicts the impact of the number of controllers on the control network reliability. The x-axis is the ratio between k and the total number of network nodes, n. The y-axis shows the corresponding δ. We are surprised to find that reliability optimizations on different topologies provide similar results. As expected, using too few controllers reduces reliability. However, for all topologies, past a certain amount of controllers, adding more controllers has an adverse effect. The reason is that: when k is a large number, the control network is similar to a full mesh, and too many control paths between controllers make the graph reliability low. In our experiments, the best controller number, k', is in between $[0.035n, 0.117n]$. Looking at data not shown due to space constraints, we observe that although larger networks generally require more controllers to maximize reliability, the ratio between k' and n decreases with the topology scale. However, we see no n-dependent patterns.



(a) Average latencies



(b) Worst-case latencies

Fig. 3. One-way latencies when optimizing for reliability on different topologies (log-scaled). The X mark on each curve represents the corresponding latency of the placement that achieves the best reliability.

C. Tradeoffs between reliability and latencies

As propagation delay bounds the control reactions with a remote controller in SDN [6], we are also interested in finding the tradeoffs between reliability and latencies. We run a set of simulations on the OS3E and the Rocketfuel topologies, where the network component failure probabilities of each topology are set to the same values as in Section V-B.

The tradeoffs vary with the number of controllers. For example, in the OS3E topology, when optimizing for average latency, the "best" placement for $k = 1$ also yields the optimal reliability metric. However, when placing 3-4 controllers, optimizing for average and worst-case latencies decreases reliability by 13.7% and 13.8%, respectively; on the other hand, optimizing for reliability increases the average and worst-case latencies by 17.3% and 13.4%, respectively.

Although it seems that one must choose between optimizing for reliability or latencies, we find that the additional latencies introduced by optimizing for reliability are acceptable. Figure 3 depicts the corresponding one-way latencies when placing controllers at the reliability-optimized locations on different topologies (represented by different curves). These latencies are compared to the three delay bounds proposed in [6], which are illustrated as horizontal lines in the figure. Obviously, the corresponding average latencies in all experiment topologies are sufficient to meet the ring protection targets when reliability is optimized. Furthermore, even in the worst case, optimizing for reliability still presents no fundamental limit to meeting existing delay goals.

VI. Conclusion

In this paper, we address the problem of placing controllers in Software-Defined Networks (SDNs) to maximize the reliability of control networks. After presenting a novel reliability metric (expected percentage of control path loss), we developed several placement algorithms. Their benefits are examined using real topologies. Our main results and conclusions are as follows. First of all, placement performance depends on the specific algorithm used. Among the algorithms proposed in this paper, simulated annealing algorithm provides solutions that are close to optimal. In addition, even when controllers are strategically placed, the number of them should be chosen properly. Placing too many or too few controllers reduces reliability. Finally, simulation results show tradeoffs between metrics. However, the corresponding latencies when optimizing for reliability is sufficient to meet existing response-time requirements.

Acknowledgment

This work was supported in part by the National Basic Research Program of China (973 Program) under Grant No. 2009CB320504 and the National High-tech Research and Development Program of China (863 Program) under Grant No. 2011AA01A101.

References

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[2] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker, "Onix: a distributed control platform for large-scale production networks," in *Proc. OSDI*, 2010.

[3] R. Albert, H. Jeong, and A.-L. Barabasi, "Error and attack tolerance of complex networks," *Nature*, 2000.

[4] G. Li, J. Yates, D. Wang, and C. Kalmanek, "Control plane design for reliable optical networks," *IEEE Communications Magazine*, vol. 40, no. 2, pp. 90–96, Feb. 2002.

[5] A. Tootoonchian and Y. Ganjali, "Hyperflow: a distributed control plane for openflow," in *Proc. INM/WREN*, 2010.

[6] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proc. HotSDN*, 2012, pp. 7–12.

[7] G. Liu and C. Ji, "Scalability of network-failure resilience: Analysis using multi-layer probabilistic graphical models," *IEEE/ACM Transactions on Networking*, vol. 17, no. 1, pp. 319 –331, Feb. 2009.

[8] K. Le and V. Li, "Modeling and analysis of systems with multimode components and dependent failures," *IEEE Transactions on Reliability*, vol. 38, no. 1, pp. 68 –75, Apr. 1989.

[9] D. Zhou and S. Subramaniam, "Survivability in optical networks," *IEEE Network*, vol. 14, no. 6, pp. 16 – 23, Nov. 2000.

[10] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, Y. Ganjali, and C. Diot, "Characterization of failures in an operational ip backbone network," *IEEE/ACM Transactions On Networking*, vol. 16, no. 4, pp. 749–762, Aug. 2008.

[11] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit, "Local search heuristics for k-median and facility location problems," *SIAM Journal on Computing*, vol. 33, no. 3, pp. 544–562, 2004.

[12] "Internet2 open science, scholarship and services exchange." [Online]. Available: http://www.internet2.edu/network/ose/

[13] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring isp topologies with rocketfuel," *IEEE/ACM Transactions on Networking*, vol. 12, no. 1, pp. 2 – 16, Feb. 2004.