Available online at www.sciencedirect.com
ScienceDirect
ELSEVIER
October 2012, 19(Suppl. 2): 92–97
www.sciencedirect.com/science/journal/10058885
The Journal of China Universities of Posts and Telecommunications
http://jcupt.xsw.bupt.cn

On the placement of controllers in software-defined networks

HU Yan-nan (✉), WANG Wen-dong, GONG Xiang-yang, QUE Xi-rong, CHENG Shi-duan

State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China

Abstract

The software-defined network (SDN) approach assumes a logically centralized control plane, which is physically decoupled from the data plane. Since this approach simplifies network management and speeds up network innovations, these benefits have led not just to production-intent prototypes, but real SDN deployments. For wide-area SDN deployments, multiple controllers are often required, and the placement of these controllers influences every aspect of an SDN. Since optimizing every metric is generally an non-deterministic polynomial (NP)-hard problem, it is of great importance to find an efficient controller placement algorithm. In this paper, we develop several placement algorithms to make informed placement decisions, which can be used to maximize the reliability of SDN, since network failures could easily cause disconnections between the control and forwarding planes. These algorithms are evaluated using real topologies. Simulation results show that the controller placement strategies are crucial to SDN performance, and a greedy algorithm provides placement solutions that are close to optimal.

Keywords software-defined network, controller placement algorithm, reliability, OpenFlow

1 Introduction

Network architectures in which the control and data planes are decoupled have been growing in popularity. This approach, highlighted by a range of industry products and academic prototypes (such as 4D [1], routing control platform [2], Ethane [3], and OpenFlow-based technology [4–6]), is attractive since it simplifies control-plane design, allows for near optimal management of network traffic and yields a flexible, evolvable network. While the details vary, the common practice is to move the intelligence of the network off packet processing devices and 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. Nowadays, this network control paradigm is often referred to as SDN.

While it has been argued that SDN is desirable for various network environments, delegating control to a remote system raises many concerns about the performance characteristics of the control plane. Generally speaking, the SDN control plane consists of three parts [6]: control platform, which handles state distribution (e.g. reading information from and writing control instructions to forwarding elements using a well-defined application program interface (API), as well as coordinating the appropriate state among other control platform servers); 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 are a range of previous works that tackle the issues on control platform and control applications, such as scalability [6–9] and software bugs [10–11], so far, issues on control network, especially, the placement of controllers in a given wide-area network, have not been well investigated.

Fortunately, we are not the first to make this observation. As stated in Ref. [12], the controller placement strategies influence every aspect of an SDN, from node-to-controller latencies to network availability to performance metrics. Since optimizing every metric is generally an NP-hard

Received date: 25-08-2012
Corresponding author: HU Yan-nan, E-mail: huyannan@bupt.edu.cn
DOI: 10.1016/S1005-8885(11)60438-X

problem, it is of great importance to find an efficient controller placement algorithm that automate the placement decisions.

With these in mind, in this paper, we are interested in finding the answers to the following question: given a physical network and the number of controllers, how to place these controllers such that a pre-defined objective is optimized. To make the problem concrete, in this paper, we use the reliability of SDN control network as the placement metric. Improving reliability is important, since network failures could easily cause disconnections between the control and forwarding planes and further disable some of the switches. A reliable control network increases the network availability. To find the most reliable controller placements for SDNs, we first present a novel metric that reflects the reliability of the SDN control network, called expected percentage of valid control paths. After formulating the studied problem, several placement algorithms are then proposed. Through extensive simulations using real topologies, we show that placement performance depends on the specific algorithm used, and a greedy algorithm provides solutions that are close to optimal.

The remainder of this paper is organized as follows: Sect. 2 proposes our reliability metric and formulates the reliable controller placement problem. Sect. 3 presents several placement algorithms. The experimental results are given in Sect. 4. Finally, Sect. 5 concludes the paper.

2 Reliable controller placement problem

2.1 Placement metric

In SDNs, switches communicate with their controller via standard transport layer security (TLS) or transmission control protocol (TCP) connections. When multiple controllers are deployed, communications between these controllers are also required to achieve global consistency of network state (e.g. network topology and host address mapping). In this work, we define control paths as the routes (paths) set that is used by the network for communications between switches and their controllers, as well as between controllers. We assume that each of the control paths uses the existing connections between switches in the network, since this deployment model is more practical for wide-area networks. 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. Therefore, the main function of the SDN control network is to distribute network control traffic between different devices of control plane and data plane. In a healthy SDN, there requires valid control paths between switches and their controller, as well as between different controllers themselves. In case that a control path fails, disconnections between a forwarding device and its controllers or between controllers will partially invalidate the function of control networks. The more control paths fail, the severer the forwarding service disruption is. Therefore, our reliability placement metric is defined as the expected percentage of valid control paths when network failures happen.

Ideally, if failures of different control paths are independent, our reliability metric can be obtained directly. However, in many scenarios, 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 [13], 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.

The physical network is represent as graph $G(V,E)$. We first identify all major failure scenarios in physical network. For example, a failure scenario can be the case in which a single switch or a single link fails. These failure scenarios are independent of each other. Each failure scenario may affect the states of several control paths in the control network and can be treated as a cause in the cause-based reliability analysis model. Let S be the set of all network scenarios we identified plus one special scenario where no failure exists. The probability that scenario s happens is r_s. If S includes all network states, $\sum_{s\in S} r_s = 1$. Let F_s denote the set of failed physical components in scenario s ($s \in S$), and F_s is a subset of $V \bigcup E$.

In scenario s, we assume that the control paths between the switches in F_s and their controller fail, and the ones that traverse F_s (i.e., control paths between other switches and their controllers as well as between different controllers) fail as well. In another word, we

assume that the control paths use fixed routes, without considering any protection mechanisms such as backup paths or control path rerouting in case of failure.

Finally, for each failure scenario, we calculate the number of valid control paths, and our reliability metric can be obtained. Without loss of generality, suppose m is the total number of control paths and the failure of F_s in scenario s breaks q_s control paths. Then, the expected percentage of valid control paths, δ, is

$$\delta = 1 - \frac{1}{m}\sum_{s\in S} q_s r_s \tag{1}$$

Generally, the size of S grows exponentially with the number of physical network components. However, in practice, we can get a satisfying statistical coverage (i.e., $\sum_{s\in S} r_s$ is very close to 1) by only analyzing the set of most probable failure scenarios. According to Ref. [14], 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. Therefore, we assume that at most one network component fails at any time in the network. Under this assumption, the number of network failure scenarios is reduced to $|V|+|E|$, and this also gives enough precision for measuring the reliability of SDN control network.

2.2 Problem formulation

For a network graph $G(V,E)$, where V is the set of nodes, $E \subseteq V \times V$ the set of links and link weights represent propagation latencies, let n be the number of nodes, i.e., $n = |V|$. We assume that the network nodes and links fail independently. For each physical network component $l \in V \bigcup E$, define p_l to be the failure probability of component l. Given any two nodes s and t, let path_{st} be the shortest path from s to t. We define $V_c \subseteq V$ to be the set of candidate locations for hosting controllers. Let $M \subseteq V_c$ denote the set of controllers to be placed in the network, $|M|$ and $P(M)$ denote the number and a possible topological placement of these controllers, respectively. 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); if multiple shortest paths exist, the most reliable one is picked. Let c denote the total number of controller-to-controller adjacencies in the control network, i.e., the total number of control paths between controllers. Obviously, c 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 [12]. Note that our approach is also applicable to other forms of control networks.

We now formally state the problem studied in this paper, which decides the placement of a given number of controllers to maximize the reliability of SDN control networks.

Definition 1 (Reliable controller placement (RCP) problem): Given 1) a graph $G(V,E)$; 2) a set of candidate locations for hosting controllers, V_c; 3) the failure probability of each component, p_l; 4) the shortest path between any node pairs, path_{st}; and 5) a positive integer k, the RCP problem is to find a controller placement, $P(M)'$, of size k such that the expected percentage of valid control paths, δ, is maximum.

Suppose $i \in V$, $j \in V_c$. Let $y_j = 1$ indicates that a controller is placed at location j, 0 otherwise. $x_{ij} = 1$ indicates there is a control path between i and j, which means, $x_{ij} = 1$ exactly when either $y_j = 1$ and switch i is assigned to j, or $y_i = y_j = 1$ and there is an adjacency between controller i and j, 0 otherwise. Let $h_{ijl} = 1$ denotes the control path between i and j passes l, 0 otherwise. Given the number of switches $n = |V|$, the total number of control paths in the network, m, is

$$m = n + c = n + \sum_{i,j\in V_c, i<j} x_{ij} \tag{2}$$

According to Eq. (1), the expected percentage of valid control paths when network failures happen is

$$\delta = 1 - \frac{1}{m}\sum_{l\in V\bigcup E}\sum_{\substack{i\in V, j\in V_c \\ V_c\subseteq V}} h_{ijl} x_{ij} p_l = 1 - \frac{1}{m}\sum_{\substack{i\in V, j\in V_c \\ V_c\subseteq V}} x_{ij} c_{ij} \tag{3}$$

where $c_{ij} = \sum_{l\in V\bigcup E} h_{ijl} p_l$, and can be viewed as the failure probability of the control path between i and j.

We now present the integer programming formulation for the RCP problem as follows:

$$\left.\begin{aligned} &\max \quad \delta \\ &\text{s.t.} \\ &\sum_{j\in V_c} y_j = k \end{aligned}\right\} \tag{4}$$

$$\sum_{j\in V_c} x_{ij} = 1; \quad \forall i \mid y_i = 0, i \in V \tag{5}$$

$$\sum_{\substack{i,j\in V_c\\ i<j}} x_{ij} = k\frac{k-1}{v} \tag{6}$$

$$\sum_{\substack{j\in V_c\\ j>i}} y_{ij} - \frac{(m-n-1)y_i}{2} \leqslant 0; \quad \forall i \in V_c \tag{7}$$

$$y_j, x_{ij}, h_{ijl} \in \{0,1\}; \quad \forall i \in V, j \in V_c, l \in V \bigcup E \tag{8}$$

The object function is to maximize the expected percentage of valid control paths. Eq. (4) in the formulation is the constraint that the total number of controllers should be equal to *k*. Eq. (5) ensures that a switch is assigned to exactly one controller. Eq. (6) is the constraint that guarantees the correct number of controller-to-controller adjacencies in the object function. Eq. (7) means that adjacencies for locations where a controller is not present are forbidden. Eq. (8) is the integrality constraints.

The above problem can be viewed as a generalized minimum k-median problem, which is a well-known NP-hard problem [15]. However, there are two fundamental differences between these two problems. 1) While the connection costs in classic k-median problem are typically defined as the distances or travel time of the shortest paths between node pairs, the ones in the RCP problem reflect the reliabilities of control paths. 2) Besides considering connections between nodes and medians as in the minimum k-median problem, connections between different medians (controllers) are also considered in our RCP problem.

3 Placement algorithms

As is stated above, we cast the RCP problem as a generalized minimum k-median problem, which is one of the most widely studied problems in location theory. If the cost metric satisfies the triangle inequality, a number of algorithms with constant approximation ratios have been proposed [16]. However, none of these algorithms can be directly applied to solve the RCP problem, since the cost of different control paths does not necessarily meet the requirement. In this section, we present three algorithms for solving the RCP problem.

3.1 Random placement

Under random placement, each candidate location has a uniform probability of hosting a controller. The algorithm randomly chooses *k* locations among $|V_c|$ potential sites.

3.2 *l*-*w*-greedy

This algorithm places controllers on the network iteratively in a greedy fashion. Suppose we need to placement *k* controllers among $|V_c|$ potential locations, the algorithm first generates a list of the potential locations, denoted as L_c, which is ranked decreasingly based on the switch failure rates. Then, it chooses one location at a time, from the first $w|V_c|(0<w\leqslant 1)$ elements of L_c, for hosting controllers. For simplicity in writing notations, we call the set of the first $w|V_c|$ elements of L_c as candidate locations. 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 highest 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 highest cost. The algorithm iterate until *k* controllers have been chosen. For $l>0$, the algorithm performs slightly differently. 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. Generally, the overall time complexity for placing *k* controllers among $|V_c|$ potential sites is bounded by $O\left(k\left(w|V_c|\right)^{l+1}\right)$. Algorithm 1 summarizes the algorithm.

Algorithm 1: *l*-*w*-greedy controller placement algorithm

Sort potential locations V_c in descending order of node failure probabilities, the first $w|V_c|$ elements of which is denoted as array L_c.

if ($k\leqslant 1$)

Choose among all sets M' from L_c with $|M'|=k$ the set M'' with maximum δ

return set M''

end

Set M' to be the most reliable placement of size *l*

While ($|M'|\leqslant k$)

Among all set *X* of l elements in M' and among all set *Y* of *l*+1 elements in $L_c - M' + X$, choose the sets *X*, *Y* with maximum δ

$M' = M' + Y - X$

end

return set M'

3.3 Brute force

The brute force algorithm is implemented for evaluation purpose. This approach 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 at the cost of extremely long execution time.

4 Evaluation

The performances of the placement algorithms are evaluated using real topologies, the Internet2 OS3E network (http://www.internet2.edu/network/ose/, Internet2 open science, scholarship and services exchange) and two Internet service provider (ISP) topologies obtained from Rocketfuel project [17]. The key characteristics of these topologies are summarized in Table 1.

Table 1 Main characteristics of experiment networks

Networks	OS3E	AS 1221	AS 6461
Node number	34	104	183
Link number	42	302	744

In our experiments, we assume all the nodes in the networks are capable of hosting controllers, i.e., $V_c = V$. In order to qualify the performance of the algorithms, we use the relative performance of the algorithms as a metric, which is defined as the ratio between δ, the expected percentage of valid control paths, of the feasible solution found by the algorithms to the one that is achieved by the brute force algorithm. The relative performance is an appropriate metric, since it reflects how far away we are from the optimal. A value of 1.0 implies the algorithm finds an optimal solution. Recall from Sect. 2 that we assume shortest path route for each control path, for OS3E topology, we adopt the distances between nodes as link weights, whereas for Rocketfuel topologies, we use the reported latency values between vertices pairs as link weights. The failure probabilities of each switch and each link in all the topologies are randomly generated from interval [0, 0.02] and [0, 0.04], respectively.

For OS3E topologies, we run a set of simulations. In each set of the simulation, we first pick k, the number of controllers to be placed. For the given k, we run one simulation for the following algorithms: 0-1-greedy, 0-0.8-greedy, 1-1-greedy, 1-0.8-greedy, 2-1-greedy, 2-0.8-greedy, and brute force. Since random algorithm returns different result based on the locations selected, we execute random placement over 1 000 times for a given number of controllers, and select the placement that yields the best performance. Then, we repeat all simulations for the next k.

Fig. 1 shows the relative performance of algorithms on OS3E topology. The x axis is the number of controllers to be placed in the network, k. The y axis is the relative performance. As we can see, the performances of algorithms vary with the controller number, and each of the algorithms is able to find the optimal placement in some cases. However, we observe that the 2-1-greedy and 1-1-greedy perform the best regardless of the exact controller number, with the 2-1-greedy slightly better. The performance of random placement is the worst among all algorithms. Regarding other settings of l and w, although choosing placement locations from the most reliability of nodes reduces the algorithm complexity, under the same settings of l, it gives worse solutions than the algorithms where w is set to 1; On the other hand, regardless of value of w, performance improvements with l suggest that spending the cycles to backtracking is worthwhile, and one or two steps backtracking is sufficient to provide near-optimal placement solutions.

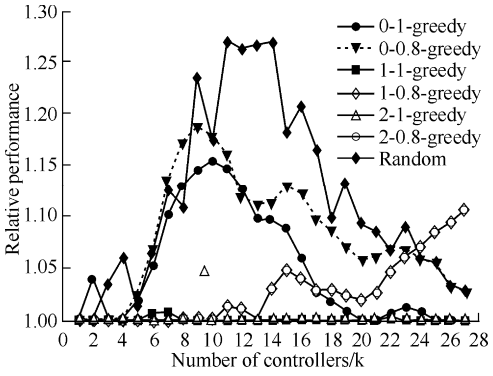
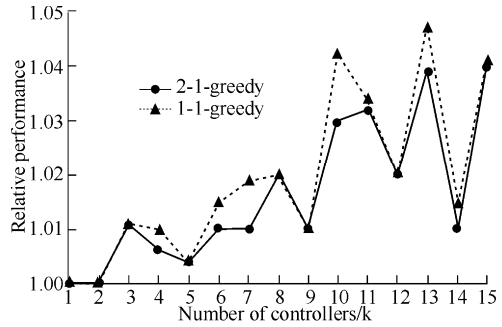


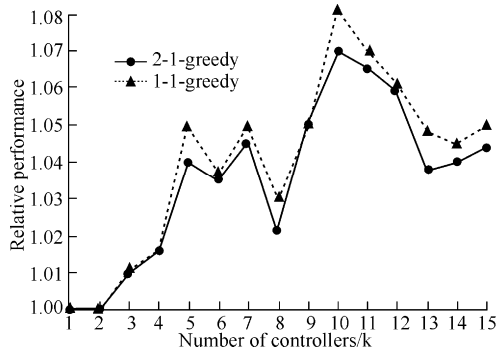
Fig. 1 Relative performance of the placement algorithms on OS3E topology

For Rocketfuel topologies, since it is infeasible to run brute force to completion, for each input number k, we stopped its executions after 2 weeks on a 4-core 2.4 GHz machine with 8 GB of memory – enough time to explore all possible placements of up to 15 controllers for the largest topologies in our experiments. Besides, since the

2-1-greedy and 1-1-greedy algorithms tend to provide better solutions, other algorithms are excluded from this part of the experiments. Fig. 2 depicts the relative performance of the 2-1-greedy and 1-1-greedy algorithms on two Rocketfuel topologies. As expected, the performances descent with the topology scales. However, these two algorithms still provide satisfactory placement solutions, which are averagely within 5% worse than the optimal for the 104-nodes AS 1221 topologies and within 8% worse than the optimal for the 183-nodes AS 6441 topologies, respectively. Although the 2-1-greedy performs slightly better than the 1-1-greedy algorithm, it takes longer time to run. As the complexity increases with topology scales, we conclude that the 2-1-greedy could provide the controller placements that are closer to optimal; whereas the 1-1-greedy is more suitable when urgent placement decisions are needed.



(a) AS 1221 (104 nodes)



(b) AS 6441 (183 nodes)

Fig. 2 Relative performance of the placement algorithms on Rocketfuel topologies

5 Conclusions

In this paper, we investigate one of the most important practical issues in wide-area SDN deployments, the placement of controllers. To make the problem concrete, we focus on maximizing the reliability of SDN control network. After presenting a novel reliability metric (expected percentage of valid control paths) and formulating the reliable controller placement problem, a number of placement algorithms are proposed. Simulation results using real topologies show that the placement of controllers should be carefully chosen and greedy algorithms with only a few steps backtracking provides SDNs with reliability performances that are close to optimal. As for future work, we are interested in exploring other placement algorithms and expanding our analysis to more topologies. The findings will appear in future papers.

Acknowledgements

This work was supported by the Hi-Tech Research and Development Program of China (2011AA01A-101), the National Basic Research Program of China (2009CB320504), the National Natural Science Foundation of China (61271041) and the Fundamental Research Funds for the Central Universities.

References

1. Greenberg A, Hjalmtysson G, Maltz D, et al. A clean slate 4D approach to network control and management. ACM SIGCOMM Computer Communication Review, 2005, 35(5): 41–54
2. Caesar M, Caldwell D, Feamster N, et al. Design and implementation of a routing control platform. 2nd USENIX Symposium on Networked Systems Design & Implementation (NSDI'2005), May 2–4, 2005, Boston, USA: USENIX. 2005, 2: 15–28
3. Casado M, Freedman M, Pettit J, et al. Ethane: taking control of the enterprise. ACM SIGCOMM Computer Communication Review, 2007, 37(4): 1–12
4. McKeown N, Anderson T, Balakrishnan H, et al. OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69–74
5. Gude N, Koponen T, Pettit J, et al. Nox: towards an operating system for networks. ACM SIGCOMM Computer Communication Review, 2008, 38(3): 105–110
6. Koponen T, Casado M, Gude N, et al. Onix: a distributed control platform for large-scale production networks. 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI' 2010), Oct 4–6, 2010, Vancouver, Canada: USENIX. 2010: 14p
7. Tootoonchian A, Gorbunov S, Ganjali Y, et al. On controller performance in software-defined networks. 2nd USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-Ice'2012), Apr 24, 2012, San Jose, USA: USENIX. 2012: 6p
8. Tootoonchian A, Ganjali Y. HyperFlow: a distributed control plane for OpenFlow. 2010 USENIX Internet Network Management Conference on Research on Enterprise Networking (INM/WREN'2010). Apr 27, 2010, San Jose, USA: USENIX. 2010: 6p
9. Cai Z, Cox L A, Ng E S T. Maestro: a system for scalable OpenFlow control. Tech. Rep. TR10-11, Department of Computer Science: Rice University. Dec 2010
10. Canini M, Venzano D, Peresini P, et al. A nice way to test OpenFlow applications. 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI'2009). Apr 25–27, San Jose, USA: USENIX. 2012: 12p

To p. 171

Soft-IDF to guide BoW feature optimization. We performed experiments on different sizes of image databases and different category methods to indicate the benefits of the proposed method. Experimental results showed that the proposed method obtained higher classification accuracy than comparing methods.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (61005004, 61175011), Chinese 111 Program Advanced Intelligence and Network Service (B08004), and a Key Project of the Ministry of Science and Technology of China (2011ZX03002-005-01).

References

1. Csurka G, Dance C, Fan L, et al. Visual categorization with bags of keypoints. ECCV'04, 2004: 1–22
2. Lazebnik S, Schmid C, Ponce J. Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. CVPR'06, 2006: 2169–2178
3. Yang J, Yu K, Gong Y, et al. Linear spatial pyramid matching using sparse coding for image classification. CVPR'09, 2009: 1794–1801
4. Yu K, Zhang T, Gong Y. Nonlinear learning using local coordinate coding. NIPS'09, 2009
5. Wang J, et al. Locality-constrained linear coding for image classification. CVPR'10, 2010: 3360–3367
6. Chum O, Philbin J, Zisserman A. Near duplicate image detection: min-hash and tf-idf weighting. BMVC, 2008
7. Battiato S, Farinella G M, Gallo G, et al. Exploiting texts on distributions on spatial hierarchy for scene classification. Eurasip Journal on Image and Video Processing, 2010: 1–13
8. Battiato S, Farinella G M, Guarnera G C, et al. Bags of phrases with codebooks alignment for near duplicate image detection. International ACM Workshop on Multimedia in Forensics, Security and Intelligence (MiFor 2010). Firenze, Italy, Oct 29, 2010: 65–70
9. Gemert J C, Geusebroek J M, Veenman C J, et al. Kernel codebooks for scene categorization. ECCV, 2008: 696–709
10. Mairal J, Bach F. Supervised dictionary learning. NIPS, 2008: 1033–1040
11. Lazebnik S, Raginsky M. Supervised learning of quantizer codebooks by information loss minimization. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2009, 31 (7): 1294–1309
12. Jiang Z L, Lin Z, Davis L S. Learning a discriminative dictionary for sparse coding via label consistent k-svd. CVPR'11, 2011: 1697–1704
13. Guo J, Guo H, Wang Z. An activation force-based affinity measure for analyzing complex networks. Sci. Rep. 1, 113, 2011
14. Wu H C, Luk R W P, Wong K F, et al. Interpreting TF-IDF term weights as making relevance decisions. ACM Trans. Inf. Syst., Jun 2008, 26(13): 1–37
15. Fei-Fei L, Perona P. A Bayesian hierarchical model for learning natural scene categories. CVPR'05, 2005: 524–531
16. Fei-Fei L, Fergus R, Perona P. Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. IEEE CVPR Workshop on Generative-Model Based Vision, 2004: 178
17. Oliva A, Torralba A. Modeling the shape of the scene: a holistic representation of the spatial envelope. IJCV, 2001, 42(3): 145–175

From p. 97

11. Foster N, Harrison R, Freedman M, et al. Frenetic: a network programming language. ACM SIGPLAN Notices, 2011, 46(9): 279–291
12. Heller B, Sherwood R, McKeown N. The controller placement problem. 2012 ACM SIGCOMM Workshop on Hot topics in Software Defined Networks (HotSDN'2012). Aug 13, 2012, Helsinki, Finland. 2012: 7–12
13. Le V K, Li O V. Modeling and analysis of systems with multimode components and dependent failures. IEEE Transactions on Reliability, 1989, 38(1): 68–75
14. Markopoulou A, Iannaccone G, Bhattacharyya S, et al. Characterization of failures in an IP backbone network. IEEE/ACM Transactions on Networking, 2008, 16(4): 749–762
15. Arya V, Garg N, Khandekar R, et al. Local search heuristics for k-median and facility location problems. SIAM Journal on Computing, 2004, 33(3): 544–562
16. Vazirani V V. Approximation algorithms. Hongkong: Springer, 2001
17. Spring N, Mahajan R, Wetherall D, et al. Mesuring ISP topologies with Rocketfuel. IEEE/ACM Transactions on Networking, 2004, 12(1): 2–16