2011 International Conference on Information and Electronics Engineering
IPCSIT vol.6 (2011) © (2011) IACSIT Press, Singapore

The Hardness of the Problems in the System Administration Model

Wiriya Techaploog[1, 2] and Sanpawat Kantabutra[1, +]

[1]The Theory of Computation Group, Department of Computer Engineering
Faculty of Engineering, Chiang Mai University, Chiang Mai, 50200, THAILAND
[2]Department of Computer Science, Faculty of Science, Chiang Mai University
Chiang Mai, 50200, THAILAND

Abstract. A system administration model is described in this article. Two problems called the MINIMUM REPLICATION PROBLEM and the MINIMUM REPLICATION PROBLEM WITH USER PREFERENCES are defined to illustrate the use of the model. The complexities of the two problems are analyzed and shown to be *NP*-complete. A polynomial-time algorithm to solve one of the problems is described and the experimental results are shown to support the feasibility of the algorithm. In our experiments, our algorithm produces solutions of values at most 1.38 times the optimal ones.

Keywords: System Administration Model, *NP*-complete, Intractability.

1. Introduction

System administration generally refers to the task of building and maintaining computer networks in order to achieve predetermined objectives. This field of research has only become more popular in the past few decades. In the past system administrators usually managed their systems with trials and errors. Only by experimenting with the systems would they know the best solutions. As recently as 15 years ago, many people still considered the task of network and system administration to comprise remembering and following complex recipes for building and maintaining systems and networks [2]. The complexity of many of these recipes made the task of system and network administrators more an art than a science. Not until a decade or so ago, a small group of scientists began to transform the task of system and network administration into a science. In this article we follow the same tradition set by these people. We begin by briefly reviewing some relevant and interesting research articles in the system and network administration.

Sandhu et al. proposed a role-based access control approach to security administration of large systems [13]. A role in this approach is a semantic construct forming the basis of access control policy. System administrators create roles according to the job functions performed in an organization, grant access authorization to those roles, and then assign users to the roles on the basis of their specific job responsibilities and qualifications. In [4] Burgess described a mean field approach to defining and implementing policy-based system administration and proposed two types of model to understand policy driven management of human-computer systems. It was shown how these models could be formulated. Verma looked at a general policy-based architecture that could be used to simplify new technologies emerging in the context of IP networks [14]. More specifically, the simplification of network administration was explained using two levels of policies and some algorithms for checking policy conflicts and unreachable policies were presented. Burgess presented an automated system administration with feedback regulation based on a notion of convergence towards a stable state [3]. In software engineering the capability maturity model (CMM) is used to improve the processes used to develop software products. Kubicki adapted CMM to describe the key processes required for a system and network administration [10]. Typically system administration involves a utilization of best practices that minimize a cost but yield a maximum value. However, the cost of system administration was not obvious. In [5] Couch et al. defined the problem of

[+] Corresponding author. Tel.: +66-53-892281; fax: +66-53-943433.
E-mail address: sanpawat@alumni.tufts.edu.

determining the cost of system administration. They illustrated simple models of cost that provide insight into why some practices cost more than expected, and why changing from one kind of practice to another is costly. Harlander described an automatic system administration tool to control the configuration and administration of UNIX workstations of different vendors from a central management host [7]. In large organizations database systems need to be available at all times. Oliveira et al. characterized typical administrator tasks, testing environments as well as mistakes and proposed system support to validate administrator actions [12]. Abdu et al. presented an adaptive model in which an initial optimal configuration of management agents is determined according to a set of use/system requirements [1]. These agents are then dynamically reconfigured to adapt to changes in resource availability and user/system constraints with little effect on the behavior of the system components. In UNIX systems administrators usually work with programs for interactive use such as *passwd* and *su*. These interactive programs cannot be kept in shell scripts. *expect* is a program which can talk to such interactive programs. In [11] Libes presented examples of using *expect* to automate system administration tasks. In system administration records of activities are kept in system logs. Authors in [9] used network traffic logs to detect and prevent potential damages caused by worms and in [6] authors studied a web log system of automatic backup. To the best of our knowledge, no one has proposed the same system administration model as ours. We are the first to study the system administration in the context of complexity theory.

This article is organized as follows. Relevant definitions and notation are introduced in section 2. Section 3 illustrates a non-trivial instance of the system administration model. In section 4 the MINIMUM REPLICATION PROBLEM (MRP) is discussed and the proof of its complexity is presented. Section 5 introduces another variant of the Minimum Replication Problem called the Minimum Replication Problem with User PrefeRENCES (MRPUP) and its complexity is shown. A greedy algorithm for the MINIMUM REPLICATION PROBLEM WITH USER PREFERENCES (MRPUP) and some experiments to support the algorithm's feasibility are presented in section 6. Conclusions are then given in section 7.

2. Preliminary and Definition

We present our system administration model in this section. This model is designed to faithfully mimic the real computer system and its environment. However, no abstract model can represent the real system with an absolute degree of actuality. Therefore, in this article our model only represents some aspects of things in the real system that we want to investigate. We now present the formal definition of the model and its related definitions. Throughout this paper let $N = \{1, 2, 3, \ldots\}$ be the set of natural numbers.

Definition 2.1

Let $n, e, f, m, r, s, d \in N$. *A system administration model* M *is a seven-tuple* (C, L, U, P, S, O, A), *where*

1. $C = \{c_1, c_2, c_3, \ldots, c_n\}$ *is the finite set of n* ***computers,***
2. $L = \{l_1, l_2, l_3, \ldots, l_e\}$ *is the finite set of e* ***connection links*** *between pairs of computers and each* $l_k = \{c_i, c_j\}$ *such that* $c_i, c_j \in C$ *and* $i \neq j$ *for some* $i, j, k \in N$, and these links connect n computers together,$C = \{c_1, c_2, c_3, \ldots, c_n\}$ *is the finite set of n computers,*
3. $U = \{u_1, u_2, u_3, \ldots, u_m\}$ *is the finite set of m* ***users,***
4. $P = \{p_1, p_2, p_3, \ldots, p_r\}$ *is the finite set of r rules in the system. This set is called* ***policies,***
5. $S = \{s_1, s_2, s_3, \ldots, s_f\}$ *is the finite set of f* ***states*** *in consideration,*
6. $O = \{o_1, o_2, o_3, \ldots, o_s\}$ *is the finite set of s* ***system operators****. Each* o_k *is a function such that* $o_k: S \mapsto S$ *for* $1 \leq k \leq s$,
7. $A = \{a_1, a_2, a_3, \ldots, a_d\}$ *is the finite set of d* ***user activities*** *and* $a_k \in U \times O \times N$ *for* $1 \leq k \leq d$.

A few remarks are in order. First, we incorporate computers and communication links into the model and all computers are connected as will be defined shortly in definition 2.2. The two components make up a computer system. Second, a set of computer users is given. Users can use computers in the system according to their rights. Third, to govern the system, a set of rules is specified in policies. Fourth, a computer system is typically composed of several components. Each of these components can have its own state. Our model uses the set of states to represent the actual states in consideration. Fifth, in a computer system an administrator usually operates the system using commands. Our model incorporates system operators to reflect this reality. Finally, user activities are also defined in the model, where N indicates a discrete time step of an activity.

We next present the operational definition of communication. A pair of computers can communicate with each other if and only if there is connectivity between them. Connectivity in this context is defined as follows.

Definition 2.2

Computers $x, y \in C$ are said to be connected if and only if there exists a sequence $<c_1, c_2, c_3, \ldots, c_k>$, such that, for $1 \leq i \leq k-1$, $\{c_i, c_{i+1}\} \in L$ and $c_1 = x$, $c_k = y$.

3. Instance of the System Administration Model

To get a clearer picture of the system administration model, we illustrate an instance of the model in this section. We assume that we are interested in studying the storage aspect of the model. Therefore, the states in consideration here are those of hard disks.

1. We have six computers $C = \{c_1, c_2, c_3, c_4, c_5, c_6\}$ in the system and c_6 is a designated server.
2. In this system connection links are $L = \{\{c_1, c_6\}, \{c_2, c_6\}, \{c_3, c_6\}, \{c_4, c_6\}, \{c_5, c_6\}\}$. In this instance every computer connects to the server directly and communicates with other computers via the server.
3. Users are $U = \{u_1$=Alyssa, u_2=Benjamin, u_3=Chris, u_4=David, u_5=Emily$\}$.
4. Policies $P = \{p_1, p_2, p_3, p_4, p_5\}$ are specified as follows.

p_1	∃! [Alyssa] can *reset* ∀[USERS]'s password.
p_2	∃! [Benjamin] can *alter* [c_6].
p_3	[Chris, David, Emily] can *work* with only [c_1, c_2, c_3].
p_4	[Chris, David, Emily] can *execute* [INSERT, UPDATE, QUERY].
p_5	∃![Alyssa] can *execute* [DELETE].

For example, in policy p_1 only Alyssa can reset all other users' passwords and in policy p_5 only she can delete data from the server.

5. Because we are interested in the storage aspect of the model, the set of states $S = \{s_1, s_2, s_3, \ldots, s_f\}$ is all possible states of the data in the hard disks. For instance, s_1 can be a string of bits that shows the state of the hard disks in the system at a certain time.
6. System operators $O = \{o_1, o_2, o_3, o_4, o_5, o_6\}$ are given in the following table.

o_1	INSERT	o_4	ALTER TABLE
o_2	UPDATE	o_5	QUERY
o_3	DELETE	o_6	RESET PASS

These operators work as expected. For example, INSERT adds a new data item into the database. When an operator takes an invalid action such as deleting a record in an empty database, the model will automatically halt.

7. In this instance we have 6 user activities $A = \{a_1=(u_1, o_5, 1), a_2=(u_2, o_4, 1), a_3=(u_3, o_1, 2), a_4= (u_5, o_2, 2), a_5=(u_4, o_1, 2), a_6=(u_1, o_3, 3)\}$. From this, we know, for example, that at time 2 Chris and David insert new data records into the database while Emily updates some records.

Observe that our system administration model can be used to study all its seven components and their interactions. In this article we apply complexity theory to decision problems concerning some resources in the model.

4. The Minimum Replication Problem and Its Complexity

In a computer system each host computer typically has a group of users to service. These users share the same information located at their host computer. Occasionally we would like to make copies of this information available to other users at other host computers as well and we want to achieve it with a minimum number of host computers and a maximum number of users, assuming that one host computer carries a single copy of the information. This problem is called the MINIMUM REPLICATION PROBLEM (MRP).

Before we formally give the problem definition, let us define an accessibility function Φ: $C \rightarrow \mathcal{P}(U)$ where $\mathcal{P}$ is the power set. We can now give the following problem definition.

Definition 4.1 (THE MINIMUM REPLICATION PROBLEM)
GIVEN: *A system administration model* $M = (C, L, U, P, S, O, A)$, *accessibility function* Φ, *and two integers* k, u^*.
PROBLEM: *Is there a subset* $B \subseteq C$ *such that* $|B| \leq k$ *and* $|\bigcup_{b \in B} \Phi(b)| \geq u^*$?

To make the problem clearer, we give a simple illustration of this problem. Suppose in the system there are six computers in $C = \{c_1, c_2, c_3, c_4, c_5, c_6\}$, and 13 users in $U = \{u_1, u_2, u_3, \ldots, u_{13}\}$. These computers and users are shown in figure 1. Other tuple members are omitted for simplicity. The dotted lines define the accessibility function Φ while the double boldface lines define the connections.

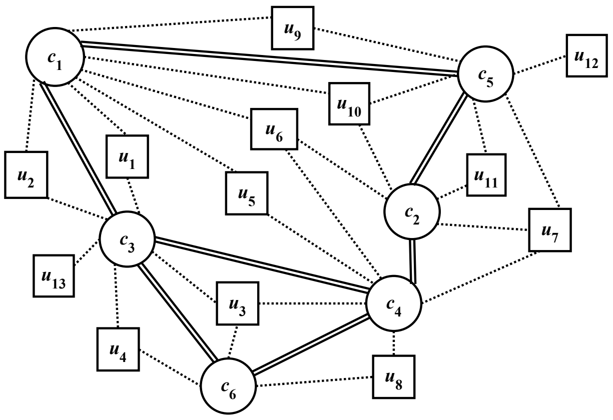

Figure 1: An instance of the Minimum Replication Problem

Suppose $u^* = 10$ and $k = 2$ in this problem instance. The answer to this decision problem is a YES. We can select computers c_3 and c_5 and both computers have exactly 10 users. Observe that this is just one possible solution.

At first glance the MINIMUM REPLICATION PROBLEM does not look at all hard to solve. However, as the numbers of computers and users grow, the only way to determine any solution seems to be an exhaustive search that grows exponentially with the size of the problem. Indeed, as we shall see in a moment, this problem is hard to solve unless *P* = *NP*. We will shortly prove that the MINIMUM REPLICATION PROBLEM is *NP*-complete. But now let us look at a problem called the SET COVER PROBLEM (SCP). This problem was one of Karp's famous 21 problems that were shown to be *NP*-complete in 1972 [8] and will be used in the *NP*-completeness reduction in the following theorem.

Definition 4.2 (THE SET COVER PROBLEM)
GIVEN: *A finite set X, a family F of subsets of X such that* $X = \bigcup_{Y \in F} Y$, *and an integer* k^*.
PROBLEM: *Is there a subset* $D \subseteq F$ *such that* $|D| \leq k^*$ *and* $X = \bigcup_{Y \in D} Y$?

Theorem 4.1 THE MINIMUM REPLICATION PROBLEM *is NP-complete.*
Proof: First, we will show that the MRP is in *NP*. Given a problem instance of the MRP and a certificate, a verification algorithm can check in a straight forward manner in polynomial time whether set $B \subseteq C$ exists such that $|B| \leq k$ and $|\bigcup_{b \in B} \Phi(b)| \geq u^*$. Next we show that the MRP is *NP*-hard. Given an instance of the SCP, we construct an instance of the MRP as follows. Let $n = |F|$, $C = \{c_1, c_2, c_3, \ldots, c_n\}$, and for each $f_i \in F$ let $\Phi(c_i) = f_i$ for each $c_i \in C$. Furthermore, let U be X. We next construct connection links. Let $e = n - 1$ and $L = \{\{c_i, c_{i+1}\} \mid 1 \leq i \leq n - 1\}$. Additionally, we let P contain a rule stating that all users in U can read their own databases and S be $\{s_1, s_2, \ldots, s_n\}$, where each s_i represents a string of bits that shows the state of the hard disk on computer c_i. We also let the set O contain a single read operator and A be the set whose user activities are that all users read their respective databases at time 1. Finally, let $u^* = |X|$ and $k = k^*$. This entire construction can definitely be done in polynomial time. It is clear that there is a subset $B \subseteq C$ such that $|B| \leq k$ and $|\bigcup_{b \in B} \Phi(b)| \geq u^*$ if and only if there is a subset $D \subseteq F$ such that $|D| \leq k^*$ and $X = \bigcup_{Y \in D} Y$. Hence, the theorem holds. ∎

The direct implication of theorem 4.1 is that, given $P \neq NP$, no efficient algorithm exists to solve the MINIMUM REPLICATION PROBLEM. In other words, it is highly unlikely that a polynomial time algorithm will be discovered for this problem.

5. The Minimum Replication Problem with User Preferences

In some networking environment it may be preferable to allow users to choose their neighbors who share the same machines. This may be for a security reason or simply for personal liking. In this section we extend the MINIMUM REPLICATION PROBLEM to reflect this characteristic and this new problem is called the MINIMUM REPLICATION PROBLEM WITH USER PREFERENCES (MRPUP). Let $T = \{t_1, t_2, t_3, \ldots, t_m\}$ be a set of m preference lists such that t_i is the preference list of user $u_i \in U$. Let $\pi^{(i)}$ be a permutation over $\{1, 2, 3, \ldots, m\}$ of user $u_i \in U$. Each t_i is defined to be a user sequence $<u'_1, u'_2, u'_3, \ldots, u'_m>$, where $u'_k = \pi^{(i)}(k)$. Let <> denote an empty sequence, meaning that a user has no preference or prefers everyone equally. In addition, we also assume that user u_i is always first in $\pi^{(i)}$. In other words, user u_i prefers himself the most. For simplicity, we assume that no pair of users is preferred equally.

Definition 5.1 THE USER PREFERENCE EVALUATION FUNCTION

Let $\rho(u, t, R)$ denote the user preference evaluation function of a preference list t of user u on user group R. The rank $r(u', t)$ of user u' on the preference list t is defined to be the position of u' on the list – 1 (i.e., $r(u_i, t_i) = 0$). When $t = <>$, $\rho(u, t, R) = 0$.

$$\rho(u, t, R) = \sum_{u' \in R} r(u', t)$$

Definition 5.2 THE OVERALL USER PREFERENCE EVALUATION FUNCTION

Let $e(U, T, H)$ denote the overall user preference evaluation function on a set U of users, a set T of preference lists, and a set H of user sets.

$$e(U, T, H) = \sum_{\substack{u_i \in U \\ t_i \in T \\ u_i \in R_i,\ R_i \in H}} \rho(u_i, t_i, R_i)$$

Let $V(C)$ denote $\bigcup_{c \in C} \{\Phi(c)\}$. We now give the following problem definition.

Definition 5.3 THE MINIMUM REPLICATION PROBLEM WITH USER PREFERENCES (MRPUP)

GIVEN: *A system administration model $M = (C, L, U, P, S, O, A)$, set T of preference lists, accessibility function Φ, overall user preference evaluation function e, and three integers k, u^*, p^*.*

PROBLEM: *Is there a subset $B \subseteq C$ such that $|B| \leq k$, $|\bigcup_{b \in B} \Phi(b)| \geq u^*$, and $e(U, T, V(B)) \leq p^*$?*

Theorem 5.1 THE MINIMUM REPLICATION PROBLEM WITH USER PREFERENCES *is NP-complete.*

Proof: We use proof by restriction to show that the MRPUP is *NP*-hard. We restrict the problem instances of the MRPUP to the instances of the MRP by letting $t = <>$ for all $t \in T$ and $p^* = 0$. We next show that the MRPUP is in *NP*. Given the problem instance and a certificate, a nondeterministic Turing machine can check the certificate in polynomial time bound to yield the correct answer. Hence, the theorem holds. ■

6. Algorithm for the MRPUP

In section 5 we have shown that the MINIMUM REPLICATION PROBLEM WITH USER PREFERENCES is at least as hard as the SET COVER PROBLEM. This fact implies the intractability of the MINIMUM REPLICATION PROBLEM WITH USER PREFERENCES. Therefore, no exact algorithm exists to solve this problem in reasonable time. We need to turn our attention to some "feasible" algorithm that yields an acceptable outcome instead. We show one such algorithm next. In designing this algorithm we have an assumption that $u^* = m$ and the condition $e(U, T, V(B)) \leq p^*$ is relaxed. Observe that this assumption does not conflict with the problem definition. That is, if the algorithm works for the case that $u^* = m$, it also works for the cases that $u^* < m$. This algorithm is called the MRPUP algorithm and is defined below.

```
MRPUP ALGORITHM ($M$, $T$)
1 {Input: model $M$ and set $T$ of preference lists}
```

```
2 {Output: set $B$ of computers}
3 begin
4   repeat
5     pick $u_i \in U$ for some $i$
6     find $c_j \in C$ that minimizes $\rho(u_i, t_i, \Phi(c_j))$ over $C$
7     $B = B \cup \{c_j\}$
8     $U = U - \Phi(c_j)$
9   until $U = \varnothing$
10 end
```

The MRPUP ALGORITHM is based on a simple greedy strategy and has a polynomial time complexity. We next show that this algorithm produces a correct and feasible solution to the MINIMUM REPLICATION PROBLEM WITH USER PREFERENCES. Observe that we abuse the notation $\Phi(B)$ to mean the union of $\Phi(c)$ for all $c \in B$.

Theorem 6.1 The MRPUP ALGORITHM produces a set B of computers such that $|\Phi(B)| = m$.

Proof: We use proof by induction on m. When $m = 1$ and $U=\{u_1\}$, the algorithm executes lines 4 to 9 to find c_j whose $\Phi(c_j) = \{u_1\}$ and includes c_j in B. Hence, the base case is true. Let a be a positive constant. Suppose the algorithm produces a set B of computers such that $|\Phi(B)| = m - a$ when line 9 has been executed. We show that this statement is also true for the case of m. The algorithm executes lines 5 to 8 to find c_j such that $|\Phi(c_j)| = a$ and adds c_j to B. When set U becomes empty, the algorithm terminates with $|\Phi(B)| = m$. Thus, the theorem holds. ∎

The MRPUP ALGORITHM gives a feasible solution in a sense that each user is connected to one of the computers in B while the overall user preference needs not be minimized. With the absence of a good approximation algorithm for this problem, our algorithm is the best known solution. We support the feasibility of our algorithm with some experimental results. In our experiments sets C, U, Φ, and T are generated randomly and the sizes of C are at most 25. Instances are generated with group sizes 20, 30, 40, 50, and 60 users. Each group size has 20 different cases to experiment. In generating accessibility function Φ we choose a user and the number of computers and the computers he or she can access are randomized. This process is repeated for every user. We generate each user preference list using a random permutation. We let sets P, S, O and A be the same as in the proof of theorem 4.1. The averages of the values of the overall user preference evaluation function $e(U, T, H)$ are then computed for each user size and compared with those of the brute-force algorithm as follows.

Number of Users	AVG($\frac{e(U,T,H)_{\text{Greedy}}}{e(U,T,H)_{\text{Brute Force}}}$)
20	1.2220414
30	1.3193747
40	1.3764368
50	1.3709160
60	1.3411097

Figure 2: Table shows the averages of the values of $e(U, T, H)$

Note that we want the average values to be as close to one as possible because one implies the optimality of our algorithm. Our experiments show that our algorithm produces solutions of the values at most 1.38 times the optimal ones.

7. Conclusion

In this article a system administration model was initially defined. An instance of the model was then illustrated. The MINIMUM REPLICATION PROBLEM was given and its complexity was analyzed. We show that this problem is at least as hard as the famous SET COVER PROBLEM. The implication is that it is highly unlikely that a reasonably fast algorithm exists to solve the MINIMUM REPLICATION PROBLEM. Additionally, we also show another variant of the first problem called the MINIMUM REPLICATION PROBLEM WITH USER PREFERENCES. This problem models a situation in which a user has his wish list of his neighbors who share the same computers. As it turns out, this problem is also at least as hard as the famous SET COVER PROBLEM. Therefore, no reasonably fast algorithm exists for it. A feasible algorithm based on a greedy approach was instead presented. We use experiments to support the feasibility of our algorithm. From our experiments, we conclude that our algorithm produces solutions at most 1.38 times the optimal ones. We conclude this article with an observation that our system administration model can be easily manipulated to study other aspects of the system administration by posing the right questions and problems as we have shown in this article.

8. Acknowledgement

The first author would like to thank the Supit Kantabutra scholarship for academic excellence in supporting his graduate study. Both authors would like to thank Professor Raymond Greenlaw for his advice and the Theory of Computation Group members who provided lively discussion during the work of this paper.

9. References

[1] H. Abdu, H. Lutfiyya, and M. Bauer. *A Model for Adaptive Monitoring Configurations*, Proceedings of the 6th IFIP/IEEE International Symposium on Integrated Network Management, pp. 371-384, 1999.

[2] M. Burgess. Analytical *Network and System Administration: Managing Human-Computer System*. Wiley and Sons, 2004.

[3] M. Burgess. *Automated System Administration with Feedback Regulation*, Software: Practice and Experience, 28(14):1519-1530, 1998.

[4] M. Burgess. *On the Theory of System Administration*, Science of Computer Programming, 49:1-46, 2003.

[5] A. L. Couch, N. Wu, and H. Susanto. *Toward a Cost Model for System Administration*, Proceedings of the 19th Conference on Large Installation System Administration, pp. 125-141, 2005.

[6] Zhou Hangxia, Zheng Peng, and Yan Yong. *Web Log System of Automatic Backup and Remote Analysis.* Proceedings of 2010 International Conference on Computer Application and System Modeling (ICCASM 2010), Taiyuan, Chaina, 2010.

[7] M. Harlander. *Central System Administration in a Heterogeneous Unix Environment: GeNUAdmin*, Proceedings of the 8th USENIX Conference on System Administration, pp. 1-8, 1994.

[8] R. M. Karp. *Reducibility Among Combinatorial Problems*, Complexity of Computer Computations, New York: Plenum, pp. 85-103, 1972.

[9] T. Katoh, H. Kuzuno, T. Kawahara, and A. Watanebe. *A wide Area Log Analyszing System Based on Mobile Agents.* Proceeding of International Conference on Computer Intelligence for Modelling Control and Automation ,and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06), 2006.

[10] C. Kubicki. *The System Administration Maturity Model*, Proceedings of the 7th System Administration Conference, pp. 213-226, 1993.

[11] D. Libes. *Using expect to Automate System Administration Tasks*, Proceedings of the 4th USENIX LISA Large Installation Systems Administration Conference, pp. 1-14, 1990.

[12] F. Oliveira, K. Nagaraja, R. Bachwani, R. Bianchini, R. P. Martin, and T. D. Nguyen. *Understanding and Validating Database System Administration*, Proceedings of the USENIX Annual Technical Conference, pp. 213-228, 2006.

[13] R. S. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman. *Role-Based Access Control Models*, Computer, 29(2):38-47, 2002.

[14] D. Verma. *Simplifying Network Administration Using Policy-Based Management*, IEEE Network, 16(2):20-26, 2002.