Survivability-based Scheduling Algorithm for Bag-of-Tasks Applications with Deadline Constraints on Grids

Shupeng Wang[†], *Xiaochun Yun*[†], *and Xiangzhan Yu*[†]

[†]*School of Computer Science and Technology, Harbin Institute of Technology, Harbin, 150001 China*

Summary
In the dynamic, complex and unbounded Grid systems, failures of Grid resources caused by malicious attacks and hardware failures are inevitable and can have an adverse effect on the execution of applications. Therefore it becomes an important and difficult issue to guarantee that the applications execute normally in Grid environment. To alleviate this problem, the survivability-driven scheduling algorithms are proposed in our previous work. However these algorithms don't consider the deadline requirements of users`. In this paper, we propose a Survivability-Based scheduling algorithm for bag-of-tasks applications with Deadline Constraints that maximizes the survivability while meeting the deadline for delivering results, which is referred as *SBDC* algorithm. Compared with traditional scheduling algorithms, this algorithm is more adaptable to the complex Grid computing environment. Experimental results reveal that this algorithm can maximize the survivability of applications while meeting the application deadline.

Key words:
Grid scheduling, survivability, bag-of-tasks application, deadline

1. Introduction

Computational Grids[1, 2] are emerging as next generation parallel and distributed computing platforms for solving large-scale computational and data-intensive problems in science, engineering, and commerce. They enable the sharing, selection, and aggregation of a wide variety of geographically distributed resources including supercomputers, storage systems, databases, data sources, and specialized devices owned by different organizations. They also provide convenient mechanisms to access these resources.

While providing large computing capability, the large-scale and highly dynamic Grid systems also bring a troublesome problem. In the complex and unbounded Grid system, malicious attacks and hardware failures, which are uncontrollable and inevitable, can result in the failure of Grid resources which can affect the execution of applications in Grid systems. Tasks dispatched to remote sites may fail due to experiencing some hardware failures or malicious attacks there. Especially for the large-scale scientific applications, which need taking advantage of vast Grid resources and running for many days, failures of Grid resources may make them rarely finish. To alleviate this problem, we addressed the application survivability in Grids and proposed the survivability-driven scheduling algorithms for independent tasks and scientific workflow applications in our previous work [3]. However these algorithms ignore the deadline requirements of users` and didn't give an efficient method to trade off the application completion time and the survivability of applications according to users` requirement. Facing this problem we propose the *SBDC* scheduling algorithm for Bag-of-tasks applications. The objective of the proposed scheduling algorithm is to develop a schedule such that it maximizes the survivability and yet meets the deadline constraints imposed by users.

Task scheduling, as a critical problem in Grid computing, has been addressed by many researchers. Many scheduling algorithms for independent tasks and scientific workflow applications in Grid environment are proposed. However, most of these Grid scheduling algorithms only address the objective of minimizing the completion time of applications, and ignore the survivability of applications and deadline requirements of users`[4]. In traditional distributed systems, scheduling algorithms have been proposed to increase reliability of dependent tasks with precedence constrains [5]. However, these scheduling algorithms can not be applied to schedule tasks in dynamic, complex Grid systems. Some research efforts have been taken to increase reliability or security of tasks executing in Grids. Abawajy [6] suggested a new scheduling approach called Distributed Fault Tolerant Scheduling (*DFTS*) to provide fault tolerance to job execution in a Grid environment. Azzedin and Maheswaran [7] suggested integrating the "trust" concept into Grid resource management. Shanshan Song proposed security-aware job scheduling heuristics [8]. Lin *et al*. presented a trust enhanced security solution to increase Grid security [9].

Several works have been proposed to address scheduling problems based on users` deadline constraint. Nimrod-G[10] schedules independent tasks for parameter-sweep applications to meet users` deadline. Paper [11] proposes the cost-based scheduling algorithms for scientific workflow applications with deadline constraints. But these scheduling algorithms still do not consider the survivability of applications.

This paper was supported by the National Basic Research (973) Program of China (2005CB321806).
Manuscript revised April 2006.

In this paper, we consider the survivability of applications and the deadline requirements of users at the same time, and propose a Survivability-Based Scheduling algorithms with Deadline constrains (*SBSD*) for Bag-of-tasks applications. The objective of this algorithm is to maximize survivability of applications while meeting the deadline for delivering results.

The rest of the paper is organized as follows. In the next section we describe the system model, assumptions and the definition and mathematical model for the survivability of Bag-of-tasks applications. Section 3 presents the *SBSD* algorithm for Bag-of-tasks applications. Section 4 shows the experiments and results. Section 5 concludes the paper with summary and future directions.

2. Model, Assumptions and Problem Formulation

2.1 System and Application Model

The Grid system is composed of a number of non-dedicated sites and each site is composed of computation machines and communication machines. Computation machines inside one site have different relative CPU speed. It is metascheduler that receives requirements of users in the Grid and maps users` tasks into geographically distributed resources. And the Grid metascheduler does not have control over the local sites. Let $M= \{m_1, m_2,\ldots,m_n\}$ be the set of computation machines in the Grid system.

Considering the states of the computation machines, the following assumptions are made. Assume resource have two states: {"*normal*", "*failure*"}. When a computation machine is in "*failure*" state, tasks can not execute on it. Hardware failures or malicious attacks all can result that computation machines turn into "*failure*" state, in which the machines can not provide appropriate execution environment for tasks. The computation machines have different capability to withstand the malicious attacks or hardware failures due to different hardware, software and security policies, so allocating different machines for tasks can affect the survivability of tasks execution. Assume the failures of computation machines follow a Poisson process and each machine $m_i \in M$ is accordingly associated with a constant failure rate λ_i. It should be noted that modeling the failure of a machine by a Poisson process may not always coincide with the actual failure dynamic of the machines. However, it is shown experimentally in [12] that such an assumption can still result in reasonably useful mathematical models. For mathematical tractability, failures of machines are assumed to be statistically independent.

Bag-of-Tasks (BoT) applications are those parallel applications whose tasks are independent of each other. Despite their simplicity, BoT applications are used in a variety of scenarios, including data mining [15], massive searches (such as key breaking), parameter sweeps, simulations, fractal calculations, computational biology[16], and computer imaging[17]. Moreover, due to the independence of their tasks, BoT applications can be successfully executed over widely distributed computational grids. According to the characteristics of Bag-of-Tasks applications, let $T= \{t_1, t_2,\ldots,t_m\}$ be the set of tasks involved in the BoT application A and tasks in T are independent with each other. For each task t_i, let $AST(t_i)$, $AET(t_i,m_j)$ and $ACT(t_i)$ denote the actual start time, the actual execution time and actual completion time respectively. These parameters are subject to constraints: $ACT(t_i) = AST(t_i) + AET(t_i,m_j)$. In addition, let $EAT(t_i,m_j)$, $EET(t_i,m_j)$ denote the expected available time of computation machine m_j to execute task t_i and the estimated execution time of task t_i on computation machine m_j respectively. These estimations can be got from prediction models [13] and historical information by facilities such as the Network Weather Service [14]. Then the expected completion time of task t_i on computation machine m_j can be defined as:

$$ECT(t_i, m_j) = EAT(t_i, m_j) + EET(t_i, m_j) \qquad (1)$$

According to the definition and assumption at above, the completion time of the application A can be computed as:

$$CT(A) = \max_{t_i \in T} ACT(t_i) \qquad (2)$$

In order to meet users` deadline constraints, the scheduling algorithm should produce appropriate schedules to make $CT(A) < D(A)$ , where $D(A)$ is the deadline requirements of users`.

2.2 Survivability Model

System survivability is a new security theory, which has become an important issue in the area of network security. There have been a number of different definitions on system survivability [19]. According to [20], survivability is the degree to which essential functions are still available even though some part of the system is down. And according to [21], survivability is the ability of a system to continue operating in the presence of accidental failures or malicious attacks. Paper [19] proposes that the definition of system survivability should be given according to the system, environment, users` requirement and other factors.

In order to describe the survivability of bag-of-tasks applications quantitatively, two definitions are given firstly as follows.

Definition 1. **Task survivability**: the survivability of a task is the probability that the task can execute on the Grid resources successfully.

Definition 2. Application survivability: the survivability of an application is the probability that all tasks involved in the application can execute on the Grid resources successfully.

From the definition at above, the survivability of a BoT application A is the probability that all tasks in T execute on the Grid resources selected by the scheduling algorithm successfully. According to section 2.1, successful execution of a task requires that the resources used by it must be in "*normal*" state during the time that the task is executing. Let $P(m_j, t)$ denote the probability that machine m_j remains in "*normal*" state during time t. Because the failure of machines follows a Poisson process, $P(m_j, t)$ can be computed as: $P(m_j, t) = exp(-\lambda_j * t)$. Then the survivability of a task t_i can computed as follows:

$$\begin{aligned} S(t_i, m_j) &= P(m_j, AET(t_i, m_j)) \\ &= exp(-\lambda_j * AET(t_i, m_j)) \end{aligned} \quad (3)$$

where $S(t_i,m_j)$ denote the survivability of task t_i when it is scheduled onto the machine m_j. Finally, we can get the survivability $S(A, X)$ of the bag-of-task application A under the schedule X as follows:

$$S(A, X) = exp(-\sum_{j=1}^{n}\sum_{i=1}^{m}\lambda_j * x_{i,j} * AET(t_i, m_j)) \quad (4)$$

subject to

$$\sum_{m_j \in M} x_{i,j} = 1$$

where $x_{i,j}$ is a binary variable such that

$$x_{i,j} = \begin{cases} 1 & \text{if task } t_i \text{ is scheduled to machine } m_j, \\ 0 & \text{otherwise.} \end{cases}$$

and X is a two-dimension matrix which is composed by element $x_{i,j}$.

2.3 problem formulation

Based on the presentations at above, the problem of scheduling a bag-of-task application to Grid systems to maximize the survivability with the deadline constraint is formalized as to find a possible scheduling result X to maximize

$$S(A, X)$$

subject to

$$CT(A) < D(A)$$

where $D(A)$ is the deadline of the application A, and $CT(A)$ is the completion time of the application A.

3. Scheduling Algorithm

In most cases, the task scheduling problems are NP-complete. Consequently, many heuristics have been proposed to solve the NP-complete problem of efficiently scheduling tasks to computing resources. For example, the Min-min scheduling algorithm, the Sufferage scheduling algorithm, the Max-min scheduling algorithm, the GA scheduling algorithm and so on. However, these scheduling algorithms only focus on the objective of minimizing the completion time of the applications. In real Grid environment, these strategies may perform poorly because of the complex and dynamic nature of the Grid and the frequent failure of Grid resources. Facing this challenge, we propose the scheduling algorithm which can address the completion time and survivability of applications at the same time in our previous works [3]. However, these algorithms ignore the particular requirements of users`, such as the requirement for deadline. The *SBDC* algorithm, presented in this section, can maximize the survivability of applications at the time of meeting the deadline of applications.

The *SBDC* **Algorithm:**
Input: M, T, D/*the set of available computing machines, the set of tasks involved in the application A, the deadline of the application A*/
Output: scheduling decisions X, $CT(A)$ and $S(A)$ /*the scheduling result, the finish time of the application A, and the survivability of the application A*/

```
While T ≠ ∅ do
    TM = ∅ /*TM is the set of task-machine pairs*/
    for each task t_i in T do
        for each computing machine m_j in M do
            Compute the ECT(t_i, m_j) and S(t_i, m_j)
            if ECT(t_i, m_j) < D(A) then
                TM = TM + {( t_i, m_j)};
        end for
    end for
    if TM==∅ then
        This application can not be finished before deadline D(A)
        return NULL
    endif
    Select the task-machine pair (t_i, m_j) with the maximum S(t_i, m_j)
    from TM;
    Assign task t_i, to m_j;
    T←T – {t_i}; mark t_i as assigned;
    Update information related to computation machines and tasks;
end while
return (X, CF(A), S(A))
```

Figure 1: The *SBDC* Scheduling Algorithm

Figure 1 presents the *SBDC* scheduling algorithm for Bag-of-tasks applications. As can be seen from this figure, the *SBDC* scheduling algorithm includes two steps. The first step is to compute the expected finish time $ECT(t_i,m_j)$ and the expected survivability $S(t_i,m_j)$ of each task-machine pair (t_i,m_j), and put the task-machine pair in which the task can be finished before the deadline $D(A)$

into the set *TM*(line 3-9). The second step is to check whether the set *TM* is null. If the set *TM* is null, it indicates that the unscheduled tasks in *T* can not be completed before the deadline *D*(*A*), so we can consider that the application *A* can not be scheduled to meet the deadline(line 10-13). If the set *TM* is not null, we can select one task-machine pair (t_i,m_j) with the maximum survivability $S(t_i,m_j)$ from *TM*, then schedule the task t_i to computing resource m_j(line 14-15). In addition, delete this task from T_R, mark this task as assigned and update the information of corresponding computation machine and task (line 15-17). This is repeated until all the tasks in T_R have been scheduled.

4 Simulation Experiments

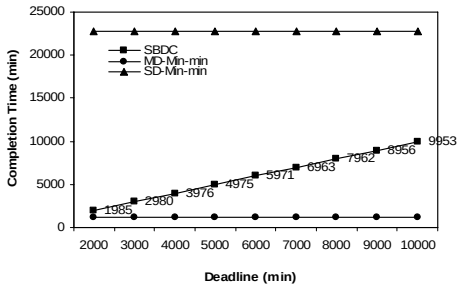
The experiments are taken based on a simulation platform implemented by extending the SimGrid simulator [18]. In the simulation experiments, the computing capability in MIPS of every computation machine is uniformly distributed in the range from 500 to 2000. The load on each computation machine is simulated by traces obtained from NWS [14] measurements on actual systems. These traces contain percentum of free computing capability as a function of time, and they are used in the simulations to make machines in the experiment to behave similar to real machines. The failure rate for each machine is chosen uniformly between 1×10^{-4}/hour and 1×10^{-3}/hour [12]. The length in MI of each task is uniformly chosen between 1×10^{5} and 5×10^{6}.

The simulation studies performed are grouped into three sets: 1) scheduling Bag-of-tasks applications with 300 tasks onto 30 computation machines in the case of different deadline constrains (from 2000 to 1000 minutes). 2) Scheduling Bag-of-tasks applications with different number of tasks (between 100 and 500) onto 30 computation machines in the case of deadline constraints 3000 and 6000 minutes. 3) Scheduling Bag-of-tasks applications with 200 tasks onto different number of machines (from 10 to 50) in the case of deadline constrains 3000 and 6000 minutes.

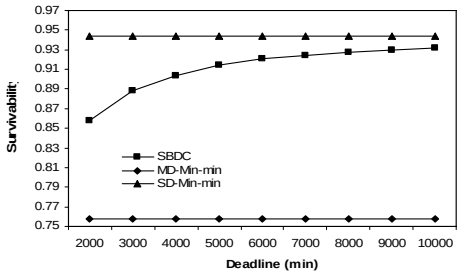
The two metrics used to evaluate the scheduling approaches are time constraint and survivability. The former indicates whether the schedule produced by the scheduling approaches meets the required deadline, while the latter indicates the probability that the applications execute normally on the Grid resources selected by the scheduling approaches. And all experiments data presented in this paper are the average of 100 experiment results.

We compare the performance of *SBDC* algorithm with the makespan-driven Min-min scheduling algorithm (*MD-Min-min*) and the survivability-driven scheduling Min-min algorithm (*SD-Min-min*). The objective of makespan-driven Min-min scheduling algorithm is to minimize the completion time of applications. And the objective of survivability-driven Min-min scheduling algorithm is to maximize the survivability of applications.

For the first set of simulation studies, the results are shown in figure 2. It can be seen that the *SD-Min-min* can get the optimal survivability, but obtain the worst completion time and can not guarantee that users` deadline is met. This is because that this algorithm only addresses the objective of maximizing the survivability of applications and ignores the completion time of applications and the deadline requirement of users`. The *MD-Min-min* algorithm obtains the optimal completion time and can meet the deadline requirement of users`, but the survivability is very low. This is because that the *MD-Min-min* algorithm only considers the objective of minimizing the completion time of application and ignores the survivability of applications. Although the *SBDC* algorithm incurs longer completion time that the *MD-Min-min* time, it still can meet the deadline requirement of users` and obtain the survivability of applications which is much better than that of the *MD-Min-min* algorithm. It also can be seen that with the increase of deadline, the completion time incurred by the *SBDC* algorithm increases but not larger than the deadline. As expected, the survivability incurred by this algorithm increases with the increase of deadline. This confirms that this algorithm can maximize the survivability of applications at the time of meeting deadlines.



(a) Completion Time

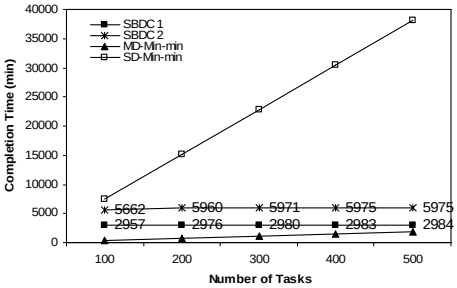


(b) Survivability

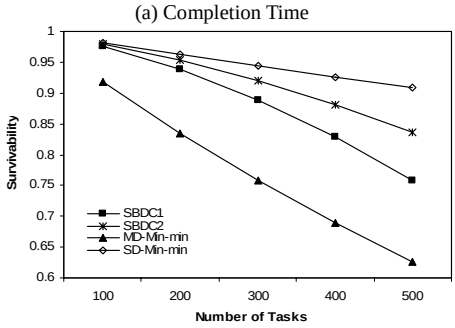
Figure 2: performance comparison in the case of different deadline (number of tasks 300, number of machines 30)

For the second set of simulation studies, the bag-of-tasks applications with different number of tasks are assumed to execute on 30 computation machines. The results of this simulation study are shown in Fig 3. In this figure, *SBDC*1 and *SBDC*2 represent the *SBDC* algorithm with deadline 3000 and 6000 respectively. According to

this figure, with the increase of the number of tasks the application completion time incurred by *SBDC* algorithm has few changes, but the survivability decreases gradually. This is because with the increase of the number of tasks, in order to meet the deadlines the *SBDC* algorithm has to select faster computing machines that may be less reliable. Because the *MD-Min-min* and *SD-Min-min* algorithm do not consider the deadline, the completion time of the application incurred by these two algorithms increase with the increase the number of tasks greatly. And although the *SD-Min-min* algorithm obtains the optimal survivability of applications, the completion time incurred by it is much longer than the deadline. The *MD-Min-min* algorithm makes the applications complete the earliest, but the survivability of the applications obtained by it is too worse.



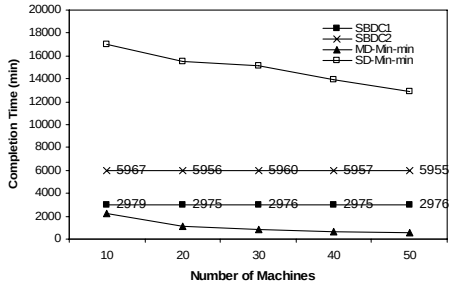
(a) Completion Time



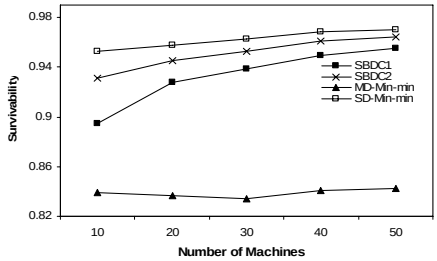
(b) Survivability

Figure 3: performance comparison in the case of different task (number of machines 30, Deadline 3000, 6000)

Figure 4 presents the experimental results of the third set of simulation studies. It can be seen that the *SBDC* can meet the deadline with the variance of the number of computation machines. With the increase of the number of computation machines the survivability incurred by the *SBDC* algorithm increases gradually. This is because that this algorithm attempts to select more reliable computing resources to execute applications on the premise of not exceeding the constraint of the deadline, and with the increase of the number of computing machines this algorithm has more chances to select computing machines with higher reliability. And the simulation results of *SM-Min-min* and *SD-Min-min* algorithm is the same as that presented at above.



(a) Completion Time



(b) Survivability

Figure 4: performance comparison in the case of different machine (number of tasks 200, Deadline 3000, 6000)

5 Conclusions

In this paper, we have presented the survivability-based scheduling algorithms with deadline constraints for bag-of-tasks applications. The objective of this algorithm is to maximize the survivability of applications while meeting the deadline. The experiment results shown in section 5 have indicated the effectiveness of the proposed algorithm. Also, we compared the algorithm with the *SD-Min-min* algorithm and *MD-Min-min* algorithm under different configuration, which further shows the usability of this algorithm.

In this paper, we only address the Bag-of-tasks applications which are composed of independent tasks. In the near future, we will consider the scientific workflow applications composed of interdependent tasks and devise the survivability-based scheduling algorithm with deadline constraints for that. Also, we will address the data-intensive applications and research the survivability-based data replication and job scheduling algorithm with deadline constraints.

References

[1] Foster and C. Kesselman, the Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, 1999.

[2] Foster, C. Kesselman et al., The Anatomy of the Grid: Enabling Scalable Virtual Organizations, International Journal of High Performance Computing Applications, Vol. 15, pp. 200–222, 2001.

[3] Shupeng Wang, Xiaochun Yun, Xiangzhan Yu. Research on Multi-objective Grid Task scheduling algorithms based on

Survivability and Makespan. Journal of China Institute of Communications. Vol.27 No.2, Feb. 2006,
[4] P.-J. Chuang, C.-H. Wei, On Task Matching and Scheduling in Heterogeneous Computing Systems, Proc. 2001 Int'l Conf. on Parallel and Distributed Processing Techniques and Applications, 2001, 901-907.
[5] Atakan Dogan, Füsun Özgüner. Reliable Scheduling of Precedece-Constrained Tasks Using a Genetic Algorithm, Proc. 2000 Int`l Conf. Parallel and Distributed Processing Techniques and Application, 2000:549-555
[6] J. H. Abawajy, Fault-Tolerant Scheduling Policy for Grid Computing Systems, Proc. IPDPS 2004.2004
[7] Azzedin, M. Maheswaran, Integrating Trust into Grid Resource Management Systems, Proc. ICPP 2002. 2002
[8] Shannshan Song, Yu-Kwong Kwok, Kai Hwang: Security-Driven Heuristics and A Fast Genetic Algorithm for Trusted Grid Job Scheduling. IPDPS 2005,2005
[9] C. Lin, V. Varadharajan, Y. Wang, V. Pruthi, Enhancing Grid Security with Trust Management, in Proceedings of Service Computing 2004 (SCC2004).
[10] R. Buyya, J. Giddy, and D. Abramson, An Evaluation of Economy-based Resource Trading and Scheduling on Computational Power Grids for Parameter Sweep Applications. In 2nd workshop on Active Middleware Services (AMS 2000), Kluwer Academic Press, August 1, 2000, Pittsburgh, USA.
[11] Jia Yu, Rajkumar Buyya and Chen Khong Tham. Cost-based Scheduling of Scientific Workflow Applications on Utility Grids. First International Conference on e-Science and Grid Computing (e-Science'05) pp. 140-147
[12] Plank J. S., Elwasif W. R. Experimental assessment of workstation failures and their impact on checkpointing systems. In Proc. Int. Symp. On Fault-Tolerant Computing, Munich, Germany, 1998: 48-57
[13] Gong L, Sun X H, Waston E. Performance modeling and prediction of non-dedicated network computing. IEEE Trans. Computer, 2002, 51(9): 1041-1055
[14] R. Wolski, N.T.Spring, J. Hayes, The Network weather service: a distributed resource performance forecasting service for meta-computing, Future Generation computer System, 1999, 15(5-6):757-768
[15] D. Abramson, J. Giddy and L. Kotler. High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid, IPDPS 2000, pp. 520-528, Cancun Mexico, IEEE CS Press, USA, 2000.
[16] J.R.Stiles, I.M.Bartol. E.E. Salpeter, and M.M. Salpeter. Monte Carlo Simulation of Neuromuscular Transmitter Release Using MCell, a General Simulator of Cellular Physiological Processes. Computational Neuroscience, pages 279-284, 1998
[17] S. Smallen, W. Cirne and J.Frey et al. Combining Workstations and Supercomputers to support Grid Applications: The Parallel Tomography Experience. Proceedings of the HCW 2000 – Heterogeneous Computing Workshop 2000.
[18] The SimGrid Project. http://grail.sdsc.edu/projects/simgrids
[19] Vickie R.Westmark. A Definition for Information System Survivability. Proceeding of the 37th Hawii Internal Conference on System Sciences (HICSS`04), Track 9.
[20] Deutsch, M.S. & Willis, R.R., Software Quality Engineering: A Total Technical and Management Approach, Englewood Cliffs, NJ: Prentice-Hall, 1988.
[21] Jha, S., Wing, J.; Linger, R.; Longstaff, T., Survivability analysis of network specifications, International Conference on Dependable Systems and Networks, IEEE, 2000, pp. 53-58.

Shupeng Wang received the M.S. degrees in Computer Science from Harbin Institute of Technology in 2002 and 2004. From 2004-to date, he is working for hit doctoral degree stayed at Computer science department of Harbin Institute of Technology. His research interests include Network Security, System Survivability and Grid Computing.

Xiaochun Yun received the B.S. and Dr. Degree in Computer science from Harbin Institute of Technology (HIT) in 1993 and 1998 respectively. He has been working in HIT from 1998 and been a Professor at computer science of HIT since 2002. His research interests include Computer Network, Network Security and Network Simulation.

Xiangzhan Yu received the M.S. and Dr. Degree in Computer Science from Harbin Institute of Technology in 1997 and 2005 respectively. He has been working in HIT since 1997 and been an associate Professor in Computer science of HIT since 2005. His research interests include Disaster Tolerance, Network Security.