Redundant Group Peers Based Adaptive Load Distribution in Peer-to-Peer Grid

Yong-Hyuk Moon†, Jae-Hoon Nah†, and Chan-Hyun Youn††

† Electronics and Telecommunications Research Institute, Daejeon 305-700, Korea
††School of Engineering, Information and Communications University, Daejeon 305-714, Korea

Summary
We present our own research work that uses extents of Peer-to-Peer technology with a framework that allows reliable Grid computing (*P2P Grid*) over the Internet. We propose how to decide optimized redundancy level of group peers by using system cost function and grid local reliability. Moreover we discuss an effectiveness of *SLA-constrained load scheduling policy with multi-probing technology* in order to maintain group more stable. Especially SLA-constrained load scheduling policy is designed for handling divisible loads and indivisible loads simultaneously and guaranteeing the shortest time of completing task. Finally through the simulation, we provide that these two proposed schemes can be evaluated to the reasonable solution to overcome unexpected system fault or down regarding system dependability issues in redundant group peers based P2P Grid environment.

Key words:
P2P Network, Grid Computing, Load Distribution, Group Peer

1. Introduction

Recently, both approaches of Grid computing [1] and Peer-to-Peer (*P2P*) networks [2] have been rapidly evolved and widely deployed. These two technologies appear to have the same ultimate objectives: the pooling and coordinated use of large number of distributed resources, even though the current studies on these architecture tend to focus on different requirements and there are also important distinguishes such as method of resource management, motivation of target application and scalability level [3][4]. However, few noteworthy researches have made efforts currently how to integrated Grid to P2P, namely P2P Grid. The core issues in P2P Grid [5] are related to how to maintain the group permanently enduring dynamic nature of P2P Grid with high reliability; therefore we will discuss about optimum system dependability using reliability analysis as conventional method and a way of load imposition in terms of vital nodes as main scheme to make whole system stable and robust.

First, provision of reliable services in distributed computing system (*DCS*) can be considered as the main issue. Normally, using redundancy scheme (duplicate vital nodes) for reason of dependability, complete executing jobs or transferring tasks can be guaranteed. However, in the literature on redundancy, the reliability problem for a general *DCS* has been turn out highly complicated and reliability evaluation is also usually computationally expensive [6]. Thus, one thing to consider next is to define what primary factor is helpful to simply decide reliability (e.g. communication load, computation time, the extent of system instability) and how to generate and maintain duplicated nodes in the distributed computing environments (*DCEs*) (e.g. redundancy optimization [6][7]).

Due to the dynamics between the individual systems in practical, optimized hardware redundancy level is yet unknown so that it is worth to deserve some investigation. Secondly, the paradigm of load distributions is basically concerned with a single large load which originates or arrives at one of the nodes in the network. The load is massive and requires an enormous amount of time to process given the computing capability of the node. The processor partitions the load into many fractions, keeps one of the fractions for itself to process and sends the rest to its neighbors (or other nodes in the network) for processing. An important problem here is to decide to how to archive the balance in the load distribution between system resources so that the computation is completed in the possibly shortest time. This balancing can be done at the beginning or dynamically as the computation progresses and the computational requirements become more precise.

As a reason of that, many researchers focusing on the conventional divisible load theory (*DLT*) [8][9] based algorithms have attempted to achieve optimal partitioning of massive loads to be distributed among resources in *DCEs*. However, there is strong dependency upon prior knowledge of network parameters. Namely, existing algorithm based on the perfect information strategy [8] can handle the variations or lack of information about these parameters. Furthermore, system utilization is to represent indirectly the system availability, and it can be a criterion to decide how much intensive fraction of loads should be distributed to a particular peer in multi-level tree graph. In short, computation and communication capability of each

Manuscript received November 20, 2007
Manuscript revised November 27, 2007

peer can be thought of as parameters to decide load fraction assigned to individual system. Additionally, they are the primary performance indices how fast particular peer completes imposed load fraction. Multi installment [10] has better performance than single installment in the perfect information strategy, because of allowing a workstation to start computing without waiting for the whole load to be received. Therefore, we will use the probing technology to obtain the values.

In this paper, firstly we will discuss about a decision method of redundancy optimization in redundant group peers based P2P Grid architecture using the average system cost determined by *Grid Local Reliability* [5].

Furthermore, we will propose *SLA-constrained load scheduling policy* which support divisible and indivisible load with possibly minimum time of completing task as our main contribution, because Grid system try to decide required quality of services through the *SLA (Service Level Agreement)* scheme which can be thought of service requirement description or contract.

2. P2P Grid with Redundant Group Peers

We are here concerned with the key aspects of the distributed model of P2P Grid computing system: distributed peers provide cost-effective means for resource sharing and extensibility, and obtain potential increases in performance, reliability and fault tolerance. In order to provide functionalities oriented to P2P networks such as the efficiency of search, autonomy, load balancing, and robustness into Grid computing technology, there are very strong needs to employ the P2P conceptual model such as the super peer networks [11] since, its structure can be an acceptable method to make more decentralized Grid.

A. Redundant Group Peers

Now, we propose the Group Peer based P2P Grid computing architecture which has been introduced in [5]. Basically each peer can be classified into two types of peer such as a group peer (*GP*) and a client peer (*CP*). First, GP has many responsibilities to manage a group and peers and to communicate with other *GPs* as well. And it has a lot of functionalities as following: control message processing, resource discovery (e.g. Grid services, data, and computing element), and store the metadata of peers with index. In addition, as aspect of Grid middleware, it should take a role of the quality of services negotiator, job executor with monitoring, and aggregation of result sets. On the other hands, the latter can be considered as a resource consumer and a resource provider at the same time in the proposed system. It is obvious that *GP* will experience a problem in terms of system over-load, namely failure of single point. As the alternative way to overcome this, we propose the redundant *GPs* scheme for dependability and performance reasons. To utilize this efficiently a controller distributes the load between *GPs* in the best possible way. A variety of factors can take a system off-line, ranging from planned downtime for maintenance to catastrophic failure [7]. The *GP* based P2P Grid computing can be modeled by the *k-level* of a redundancy. Simply we consider that increasing *k* value makes system more reliable than system with low level of redundancy scheme. However, one more thing to consider next is that intuitively, unlimited duplication is improper due to infinite cost. Thus, redundancy optimization policy regarding performance issue (e.g. cost-wise strategy) is still remained to discuss.

B. Reliability Analysis with Redundant GPs

The system reliability of P2P Grid for a given job is the product of the component reliability that each processor on which a job is executed, is operational during the period of module execution, and the component reliability that each communication link on which the data communication takes place is operational.

- *Component Reliability*

A general model of component reliability in *DCEs* can be derived from a probability analysis with the concept of Poisson process and working time. We begin a derivation by examining component reliability with working time such as computation time on a resource and communication time at a job execution path as depicted in fig. 1. One of the main causes of failures in P2P Grid is the heavy data migration, which is the physical flow of data from one data source (*CP*) to next network nodes. Moreover, execution of a job inducing a critical software fault or system down due to the unexpected trials of an access to unauthorized memory area, lack of storage space, and internal exception of an operating system significantly should be dealt. Thus, with these points mentioned above we continue to set up component reliability model in detail.

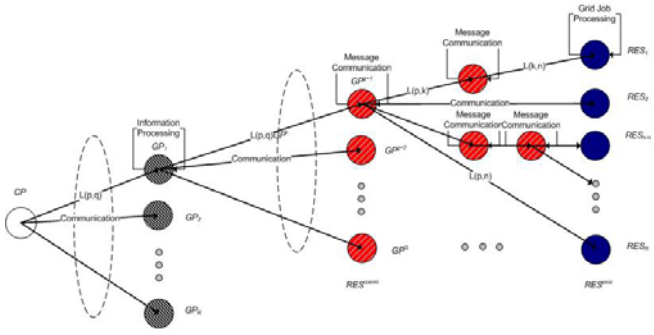


Fig. 1. Reliability model in P2P Grid job execution path

Fig. 1 shows the interaction (job execution path) between the P^{th} *CP* (CP_p) and level *k* redundant q^{th} *GPs* in the g^{th}

group, $GP_q^g(r)$. From this relation, the partial component reliability R_L could be derived as followings:

$$\prod_{p\in CP}[\prod_{q\in GP} b\inf(R_L(p,q\mid J_m))]= \prod_{p\in CP}\left[1-\prod_{q\in GP}[1-e^{-\lambda_{p,q}\{\sum_{i=1}^{k-1}\sum_{j=i+1}^{k}\tau(i,j,p,q)/w^*_{p,q}\}}]\right] \quad (1)$$

Where the $\lambda_{p,q}$ is the failure rate, satisfying Poisson process, for communication between CP_p and $GP_q^g(r)$ during mission. And J_m means an m^{th} instance of Grid job *J*. And $\tau(i,j,p,q)$ is the amount of data to be transmitted through path *L(p,q)*. Among the communication links from CP_p to $GP_q^g(r)$ if at least one connection is alive then *CP* will have an opportunity to transmit data sets or exchange information successfully during processing grid job. That is the reason why binary function $b\inf[R]$ should be used for the component reliability of group peer.

Furthermore, in order for finding the more accurate reliability of component, we derive new term $w^*_{p,q}$, "adjusted transmission rate" for the path $L(p,q)$ since it reflects the fact that the behavior of a path depends on both the transmission rates and the reliabilities of all links in the path. On that account, we mathematically define the term as below:

$$w^*_{p,q}=\frac{\mu_1+\mu_2+\mu_3+\ldots+\mu_p}{\frac{\omega_1}{\mu_1}+\frac{\omega_2}{\mu_2}+\frac{\omega_3}{\mu_3}+\ldots+\frac{\omega_p}{\mu_p}} \quad (2)$$

For the path $L(p,q)$ consisting of links $l_{1st}, l_{2nd}, l_{3rd}, \ldots, l_{pth}$, we let μ_i and ω_i be the failure rate and transmission rate of link $l_i\,(1\le i\le p)$, respectively.

Next, we will show the component reliability of *GPs* itself. In the perspective of *GPs*, formulating the component reliability of *GP* could be started from the consideration with two main functions such as computation of request and communication of data. Redundant GPs' reliability R_C is defined as follows:

$$\prod_{q=1}^{RES} b\inf[R_C(q)]=1-\prod_{q=1}^{RES}[1-e^{-\lambda_q\{\sum_{n=1}^{RES}x_{m,n}e_{m,n}+\sum_{i=1}^{k-1}\sum_{j=i+1}^{k}\tau(i,j,q,y)/w^*_{p,y}\}}] \quad (3)$$

Where $x_{m,n}$ is defined to an indicator whether the m_{th} Grid job instance J_m is allocated to n_{th} processing resource RES_n^{proc} or not. And $e_{m,n}$ means an Accumulative Execution Time (*AET*) for processing J_m on particular resources. Then, in order to guarantee the successful completion of a grid job, at a moment during the mission at least one constantly *GP* should keep taking requests sent by *CPs*.

As depicted in fig. 1, the link failure probability between GP^g and GP^{g+i} for all i, $1\le i\le G$ should be considered as the third component reliability. Incidentally, a successful completion of grid job is meant to all connections between GP^g and RES^{CP} should be stable to meet quality of services demanded by grid job. And the last part of component reliability to think is the failure on processing resources during execution of a grid job. Simply, we define the reliability of these cases in a similar manner.

- *System Reliability*

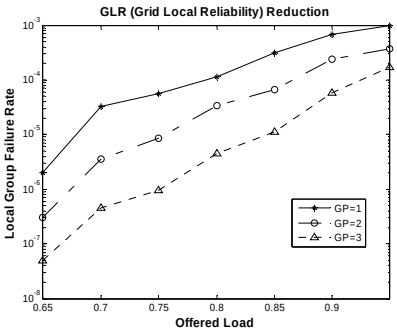
The system reliability could be classified into two parts: *GLR* (*Grid Local Reliability*) and *GSR* (*Grid System Reliability*). The former is focusing on how *GP* will perform to treat *CPs'* requests and data. Thus, *GLR* should consist of two component reliabilities previously mentioned in Eqs. (1) ~ (3), because the principal factors in *GLR* are strongly related to working times on *GP* which might suffer from unstable link status connected to *CPs*. With considerations to that at a pair of node-link, we set the *GSR* to as follows:

$$R(GLR)=\prod_{q=1}^{k} b\inf[R_L(p,q\mid J_m)]\cdot\prod_{q=1}^{k} b\inf[R_C(q)] \quad (4)$$

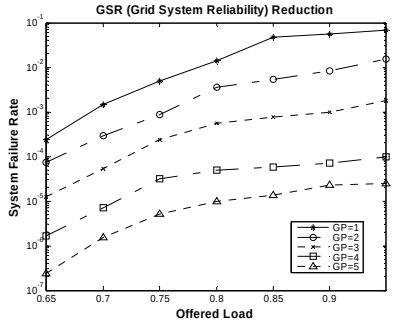
What is more, the latter (*GSR*) is defined to the degree of stability on the overall P2P Grid which performs a lot of grid sub-jobs within in a set of period. That means how stable service can provide to *CPs* or how P2P Grid system is reliable to endure dynamically changed state of processing nodes and communication links. Therefore we define the *GSR* concisely from the Eq. (4).

$$R(GSR)=R(GLR)\cdot\prod_{q\in GP,\,n\in RES}\prod R_L(q,n\mid J_m)\cdot\prod_{n\in RES}R_C(m,n) \quad (5)$$

Now we examine the result regarding two reliability concepts as depicted in fig. 2.



(a) GLR vs. Offered Load



(b) GSR vs. Offered Load

Fig. 2. A cost-optimized reliability in P2P Grid

When it comes to system dependability of *GPs*, we focus how long time or how large of load the system can tolerate. Thus indirectly *GLR* means the system availability. Moreover we need to consider overall system reliability during execution jobs since *GLR* does not tell us whether the mission completes successfully or not. For that reason, we derived *GSR* in considerations of component reliabilities with regard to communication links, *GP* systems, and resource nodes. In fig. 2 (a), there are three results with different analysis conditions: we set *CP* to 100 and *GP* to 1, 2, and 3 respectively. Each line has a similar increasing pattern, however a gradient of each line is quite different: in case of *GP*=1, *GLR* suddenly goes up in the early state, however after 0.8 normalized load approximately we obtain the line slowly moving up, namely saturated. Moreover, for evaluating the effective of the k-redundancy scheme in proposed P2P Grid, we compare the system reliability, *GSR*, with different level of redundancy of *GPs* from 1 to 5. Especially *GPs* equal to 1 or 2, system might not be operational or provide reliable service owing to rapid increment during mission as shown in fig. 2 (b).

Next we consider a relation the system cost duplicating *GPs* to system reliability.

C. Redundancy Optimization with System Cost

With additional endowment of *GPs* redundancy, the P2P Grid becomes more reliable, hence reducing average execution cost more significantly in the long run. However, such endowment increases system cost such as hardware deployment cost, maintenance cost, etc. This trade-off between system cost and the redundancy level is accounted for after examining the sources of system cost due to redundant group peers. Suppose that system cost is the sum of communication cost and computation cost due to generation of additional *GPs* and these costs are linear function of time. The two costs at a particular *GP* respectively can be defined as: $C_p(k \mid m) = k \cdot c_p \cdot t_n(m)$ and $C_c(k \mid m) = k \cdot c_c \cdot T_c(p,q)$. Therefore, finding k that minimizes the average system cost can be a solution for redundancy optimization in P2P Grid.

$$Avgf[C_{system}] = \sum_{m \in J} \left[\sum_{k=1}^{n(GP)} \left\{ C_p(k \mid m) + C_c(k \mid m) \right\} \Big/ R(GLR) \right] \qquad (6)$$

Where *Avgf[C_{system}]* is meant to average system cost. Usually, it is hard to find optimal redundancy level from Eq. (2) directly. Thus we need to examine the pattern of the graph by using the mathematical approach introduced in paper [6]. First let define *1/R(GLR)=$f_{opt}(k|m)$* then function *$f_{opt}(k|m)$* is decreasing convex with respect to the level of redundancy *k*, due to d[*$f_{opt}(k|m)$*]/dk<0 for all k≥ 1. Next total expected cost can be expressed by *Avgf[C_{system}] =(a+b)k*$f_{opt}(k|m)$* because *Avgf[C_{system}]* is the function of *k* so that the other terms are treated as constant. Especially, we set *F(k|m)= k*$f_{opt}(k|m)$* as the decision function.

In fig. 3 shows two graphs: one is $f_{opt}(k \mid m)$ decreasing strictly, the other is line straight denoting the gradient of $f_{opt}(k \mid m)$. In case of k_r, *x* can be determined by this simple Eq. as follows: $f'_{opt}(k_r \mid m) = \nabla f_{opt}(k \mid m)|_{k=k_r} = x/-k_r$, then $x = -k_r \cdot f'_{opt}(k_r \mid m)$. If $F(k \mid m)$ increasing with respect to *k*, then optimal *k* (k^*) is 1, otherwise we need to find the *k* making the derivative $F'(k \mid m) = 0$, since it means that $-k^* \cdot f'_{opt}(k^* \mid m) \approx f_{opt}(k^* \mid m)$ as depicted fig. 3. Namely, $F(k \mid m)$ is uni-modal and k^* is the optimal redundancy level such that $F(k^* \mid m)$ is minimized.

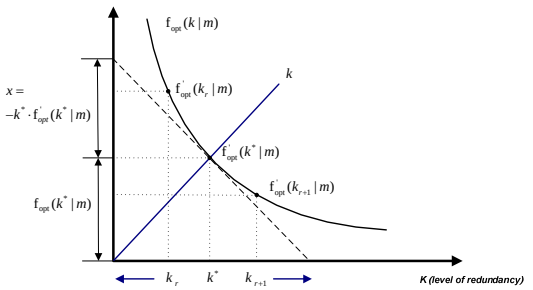


Fig. 3. Archiving the level of optimization

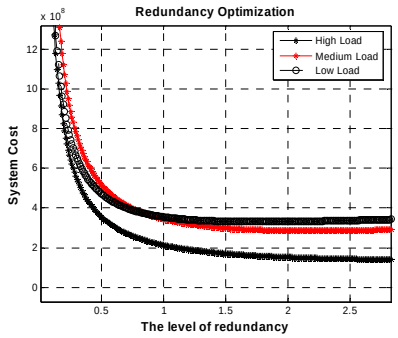


Fig. 4. Redundancy optimization with minimum system cost

Fig. 4 can be a good result how to decide the optimized redundancy level against system cost in P2P Grid. For showing the different optimal level of redundancy, we set unit load size to 500, 200, and 100 relatively. In the first case, we allocate heavy individual load to *GPs* so that we get 3.25 optima for supporting imposed load. The rest of both results show same appearance as first one, however as decreasing unit load size, we can save system cost to guarantee reliable service in P2P Grid.

3. Adaptive Load Distribution in P2P Grid

Although many studies on conventional reliability of *DCS* have been progressed, theoretical reliability analysis still has a weak point; it is not enough to prove whether decided optimum redundancy level might be useful or not in the practical *DCS*. However, our reliability model in P2P Grid is based on job execution path depicted in fig. 1, and it focuses only to check job completion. Thus, feasibly we can decide redundancy level under offered loads.

In order to improve this shortcoming, we suggest adaptive load distribution algorithm, because a heavy volume of loads imposed by *CPs* can be thought of a main cause to induce system fault in P2P Grid environment.

Finding a scheduling algorithm that minimizes the completion time for a distributed program consisting of a number of processes is one of the classical computer science problems, and has been shown to be *NP-complete* [9]. Therefore, a number of good heuristic methods have been suggested, and it is thus possible to come close to the optimal result for many important cases. However, all heuristics are highly diverse and are dependent on the structure of the parallel program and architecture.

As a reason of that, with the well-known *DLT* and multi-installment [10] based load distribution strategy, which might be appropriate to tree networks, we will devise heuristic algorithm in order for minimizing the finishing time in the proposed P2P Grid.

A. Analytical Model for P2P Grid Load Scheduling

We set the network graph which consists of peers and is based on star networks. Besides concurrent load distribution is used: suppose that divisible load (e.g. data set), and invisible load (e.g. Grid tasks or programs) [9] are thought of as main load type.

Entire loads are distributed to *local GPs* first and each of *GPs* sends sub-jobs or data sets to the *foreign GPs*. These loads finally arrive at each resource allocated to P2P Grid application by resource discovery [12]. We consider that incoming loads from *CP* at the k^{th} level are equivalent to entire load imposed to *GPs* at the $(k-1)^{th}$ level of star tree networks. Therefore, the aggregation load T can be expressed by the summation of individual load such as $T_1, T_2, T_3, .., T_i$.

B. Probing Strategy for Unknown Parameters

In order to estimate network parameters, some papers [8][10] have proposed the *feedback strategy* as probing technology with multi-installment such as below:

- PDD (Probing and Delayed Distribution) is the simplest strategy, but time delay to receive feedback such as *CTC* (Communication Task Completion) and *PTC* (Processing Task Completion) messages will be very high. Also it does not have fault tolerance capability; when one workstation gets isolated due to some faults, the *PTC* message will not arrive and the processing of the remaining load will never commence.

- PCD (Probing and Continuous Distribution) strategy shortens idle time with fast workstation-link pairs. But still workstation should wait for the last feedback (*PTC*) message computing network parameter in order to estimate workstation-link's performance. Especially, with a consideration of slow workstation-link, *PCD* might be slower than *PDD*.

- PSD (Probing and Selective Distribution) is the strategy improving *PCD*; a very slow workstation-link pair will get only a small fraction of the entire load, and the one which has a fault will not get any load.

With the *PSD* as probing technique, we estimate the capability of peers and put the dropping condition [10] that finds slow peers for optimal load distribution in the completion time of task. The problem which we have to consider next is the loads with no precedence relations. Since in case of indivisible loads the scheduling algorithm has been done in such a way that an entire load is assigned to only one processor [10], we simply integrate indivisible loads as a part of our dynamic load distribution.

Before starting the algorithm, we define the following parameters used in load distribution analysis:

Table 1: Type Sizes for Camera-Ready Papers

Symbol	Description
L	Set of load consisting of indivisible and divisible tasks e.g. $L = \{ T_{d1}, T_{i2}, T_{i3}, T_{d4}, T_{i5},\ldots, T_{dn}\}$
M	The total number of resources allocated
T_i	Set of indivisible tasks
$T_{i\text{-}rs}$	Set of indivisible tasks after executing step 2
T_d	Set of divisible tasks
$n(T_i)$	The number of indivisible tasks
$n(T_d)$	The number of divisible tasks
$S_{i\text{-}k}$	The finish time needed to execute all the individual tasks on k dedicated resources
t_i	The finish time of the i^{th} task
ft_i	The finish time of resources i
k	The number of resources dedicated to serve the indivisible tasks from M available resources

C. SLA-Constrained Load Scheduling Policy

Our proposed load distribution algorithm is aimed to arrange the entire loads to resource-limited computing nodes in order to guarantee the most minimum completion time of tasks. We assume that offering load composes with indivisible load and divisible load according to a given constraint of *SLA* (Service Level Agreement); the former is the set of sub-processes (e.g. MPICH-G2 application [12]) which have strong relation among sub-processes such as very small delay time for exchanging messages, while the latter can be considered as data sets with no precedence relations. The *adaptive SCL (SLA-Constrained Load) scheduling policy* can be split to two sub-algorithms. The first one (step 1 ~2, see fig. 5) is to check the pre-defined conditions to schedule each load (task): dividing entire load to indivisible and divisible one under a SLA condition and with *PSD* feedback strategy, attempting to probe network capabilities in terms of the time taken by a reference link to communicate one unit of load and by a reference processing node to process or compute one unit of load. And the second sub-algorithm (step 3 ~ 5) is designed to distribute loads optimally under meaningful steps depicted in fig. 6. Next we will discuss more about sub-algorithms.

- The 1st Sub-Algorithm

The first sub-algorithm describe in fig. 5, divides a set of load into the set of indivisible load and divisible load respectively then we estimate t_i for each resource through the probe phase in step 1. And checking the number of indivisible tasks is larger than the given available resources M; if it is so, just distribute all tasks to idle resources, shortly it proceeds to schedule indivisible tasks as large as only the number of available resources, otherwise from the remaining set of indivisible tasks, it tries to find any task such that minimize the gap between the average finish time and practical one on each resource iteratively. Then this selected task first will be allocated to the most available (idle) resource with respect to finish time. Finally if we do not have any divisible tasks, the *SCL* scheduling policy ends here, otherwise it goes to the second sub-algorithm in order to treat the divisible tasks.

BEGIN:
Step 1
Divide L into T_d and T_i
Estimate network capabilities by using *PSD* strategy

Step 2
Let $L = \{ T_{i1}, T_{i2}, T_{i3},\ldots, T_{in}\}$, assume that M available resources
If $n(T_i) > M$ continue
Schedule the $n(T_i)=M$ to the M available resources
Else *l*et $T_{i\text{-}rs} = T_{i,}$
From Remaining Set, $RS=\{1, 2,\ldots, n(T_i)\text{-}M\text{-}1, n(T_i)\text{-}M\}$ of unscheduled indivisible task, Select any tasks and distribute them to any resources satisfying as followings:

$$Min\left[D = \left| \sum_i^{n(T_i)} t_i / M - ft_x \right| \right] \quad for\ x = 1, 2, ..., M$$

repeat until all indivisible tasks are scheduled to any resources
If $n(T_d)=0$
ft_i = *Final time stopping to execute last task*
exit
Else go to *Step 3*
END

Fig. 5. The 1st sub-algorithm of SCL scheduling policy

- The 2nd Sub-Algorithm

The question of what would be the best way to schedule the divisible tasks remains unsettled. As a reason of that, we will devote some space to the discussion of the 2nd sub-algorithm. Note that term scheduling has a different meaning from distribution: scheduling is pre-decision prior to distribute task on a particular resource.

As given in below, it iterates to find the j^{th} minimum time of indivisible task being less than maximum time of indivisible task. And if j is found then continue to step 4, otherwise go to step 5 then distribute all indivisible tasks to execute on idle resources and check the final time as completion time of task.

In step 4, resources are divided into two parts; k resources are dedicated to the indivisible tasks and M-k among them devoted to serve the divisible tasks respectively. And in the final step 5, we have to find k such that makes the finish time minimum. There are two cases; if given condition is satisfied then k is equal to 0, else we continue find k with an additional condition such that equation:

$Max\left(\sum_{d}^{n(T_d)} t_d / (M-k),\ S_{i-k}\right)$. Especially, k equal to 0 means that the finish time is minimum if we use all resources to execute the divisible (indivisible) tasks first and the indivisible (divisible) tasks next.

BEGIN:

Step 3

Do *find max j satisfying the following equation*:

$t_j of\ T_{i-rs} + \sum_{d}^{n(T_d)} t_d \le Max\, t_j$

While *($j \le n(T_{i\text{-}rs})$)*

If *j* is found continue Else *Step 5*

Step 4

Distribute all T_i *to idle M* and *w*ait until finish to finish j^{th} indivisible task then divide divisible tasks to all idle resources

ft_i = *Final time stopping to execute last task*

exit

Step 5

Let *k* resources dedicated to the indivisible tasks then *M-k* resource to serve the divisible tasks, then finding *k* makes finish time minimum

If $\left(\sum_{d}^{n(T_d)} t_d / M + Max\ t_i < \sum_{d}^{n(T_d)} t_d / (M-1)\right)$

$k = 0$

Else Find *k* that provides the minimum of

$Max\left(\sum_{d}^{n(T_d)} t_d / (M-k),\ S_{i-k}\right)$

ft_i = *Final time stopping to execute last task*

END

Fig. 6. The 2nd sub-algorithm of SCL scheduling policy

Therefore, we can predict the minimum finish time of entire tasks (loads) through the two sub-algorithms. In short, *SCL* scheduling policy gives an answer of adaptive and dynamic load balancing policy with respect to the finish time in the P2P Grid computing system.

4. Simulation and Discussion

In this section we discuss about the performance evaluation by using an analytical model as stated in the section 2 ~ 3. Through the analysis, we will show the effectiveness of usage in proposed schemes.

A. Evaluation Preliminaries

In order to consider characteristics of proposed P2P Grid computing system (e.g. scalability, duplication of core system) we design the evaluation model which meets the requirements related to two performance indices such as cost-optimized system redundancy, and *SCL* scheduling policy. The number of total peers in network graph is defined to 10,000 ~ 20,000 and each group has the same number of *GPs*. The policies with respect to task assignment depend on the *PQRM* [12] which is the Grid resource brokering system. We summarize those assumptions of into table 2.

Table 2: Configuration of Parameters

Parameters	Default Range
$N = n(Peers)$	$10,000 \sim 20,000$
$Group_{size}$	$50 \sim 100$
$k = n(GP)$	$1 \sim 5$ $(no-redundant, if\ k=1)$
$Unit(Load)$	$100, 200, 300, ...,1000$ $(e.g. MBytes)$
$Failure\ Rate$	$10^{-1} \sim 10^{-8}$

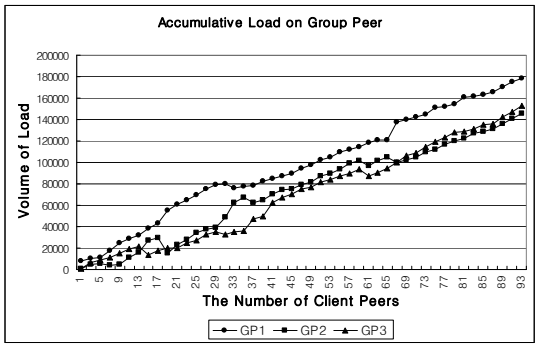
The several meaningful simulation results generated by numerical analysis will be discussed in detail. Simulation procedures are composed with three phases. First we estimate system capability for each resource then classify loads to divisible and indivisible one according to *SLA* constraints of application. Second we schedule each load to resources with individual strategy of *RLD*, *SCL*, and *PIS* scheduling policy respectively. At last *load controller (LC)* check the time of final task completion then distribute loads optimally according to the pre-decided load distribution policy.

For showing obvious effectiveness of *SCL* scheduling policy and extending scalability of P2P Grid system, we set the maximum number of *CPs* and *GPs* to about 100 and 5, respectively. Approximately 100 ~ 200 resources in a single group and 100 unit load size are applied to this analysis.

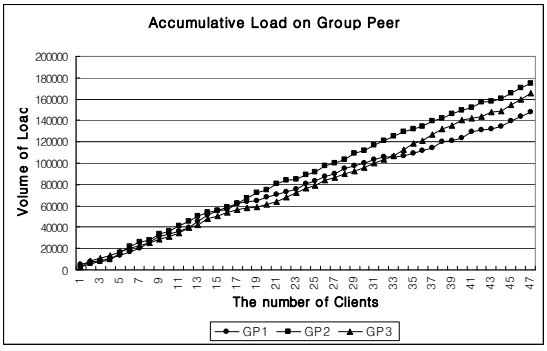
B. Simulation Results

The fig. 7 shows that difference of the degree of load balancing among *GPs* is getting larger as the number of *CPs* increases. Since we suppose that all *GPs*' system capability (e.g. communication and computation) is not predictable (almost unknown), in the fig. 7 (a) this result shows a disorder pattern of load distribution. In this paper, we name this policy *Random Load Distribution* (*RLD*) which might be a lower bound of performance. However fig. 7 (b) gives us more stable feature when we use the *SCL* scheduling algorithm as our main policy for load distribution. Moreover a simulation result in fig. 7 (c) represents an ideal appearance of balanced loads among *GPs*. The last case comes from the *Perfect Information Strategy* (*PIS*) [8].

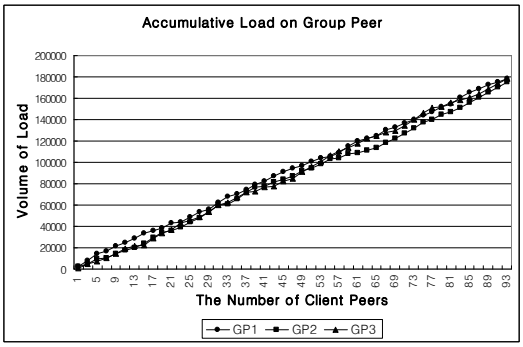
In summary, three lines go up with a large difference of loads intuitively as a volume of offered loads increases highly. Hence, we expect that heavy loads imposed to a particular *GP* causes a problem so called *failure of single point* (e.g. down of core system). As a reason of that, the degree of load balancing can be thought of as a significant factor to decide how stable a group lasts in proposed P2P Grid networks.



(a) Unknown (applying Random Load Distribution)



(b) PSD based estimation (applying SCL Scheduling Policy)



(c) Ideally balanced (applying Perfect Information Strategy)

Fig. 7. The effective of scalable load balancing policy

As we have shown in above simulation results, *PIS* is the ideal measure of load distribution. It means that *LC* has already known about network parameters so that *LC* can optimally distribute given loads to resources without any process for estimation of network dynamics. Moreover, we suppose that resource nodes have similar capability. These assumptions give us upper bound of performance.

Consequently, the difference of the degree of load balancing between *SCL* and *PIS* comes from a burden to compute optimal load fraction and expected processing time on resources. However, these differences are not critical, because simulation result of *PIS* accounts for diverse nature of peer systems in P2P Grid networks.

Now we turn to discuss about finish time in *SCL* scheduling policy comparing to *RLD* and *PIS* algorithm. Usually, the effective of load distribution strategy can be measured by a completion time of task in *DLT* even though we try to handle indivisible loads simultaneously. The finish time of given tasks such as data, jobs is defined to the period such that spent for last execution of job on a particular resource node. Here is a fig. 8 which shows the completion time of task.

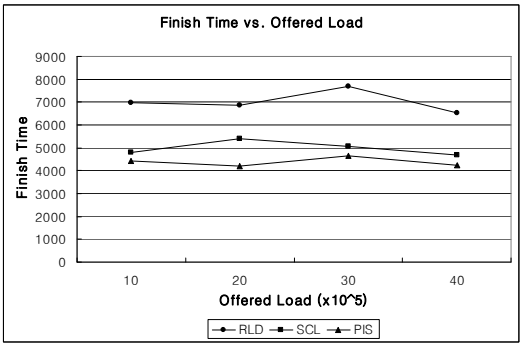


Fig. 8. Performance Comparison of Completion Time

In case of *SCL* scheduling policy, as offered load going up, the line representing final time of task processing is sharply increasing at particularly 20×10^5 in fig. 8. The occurrence of differences might be caused by the lack of available resources, wrong decision of *SCL* scheduling policy or network delay for probing; however, a primary reason that makes such an unreliable outcome is originated from the network delay obviously. Namely *LC* should wait the last computation completion message about probe (small task) in order to release load fraction to assigned resources, because *LC* decides capability of resource through the return time of completion message. This procedure makes delay. However this exception can be acceptable in practical networking system.

Moreover, after 30×10^5 point, all strategies show declining line where resources are added for taking load fractions according to cost-optimized redundancy scheme mentioned in the section 2. In other words, this feature means that more number of available resources is given to P2P Grid.

Since a large chunk of loads is assign to *LC*, the volume of load distribution must increase probabilistically. Especially, despite there are still lots of resources which have enough capacity to handle individual load, *RLD* algorithm is blind to about that up-to-date information. Hence we get relatively high completion time throughout

the simulation. While *PIS* offers us delicate results such that it tries to schedule individual load into more-capable machines in advance. Namely more-capable machine has a high priority to treat load since that machine can guarantee the more fast completion of task. As compared with *PIS*, we obtain a quite good result from the *SCL* scheduling policy. From what has been discussed above, we can conclude that the result base on *SCL* has a relatively small difference.

So far, we have pointed out two performance aspects regarding dynamic group peering mechanism in this section; one is load balancing in a single group where a lot of *CPs* transfers a heavy volume of loads to *redundant-GPs*, the other is how to *GPs* as *LC* can optimally distribute loads to resources in terms of minimum completion time of task. Additionally, we have shown that through the reliability analysis, *LC* can recognize the appropriate time. In other words, that offers critical point to system; when it should assign more available resources to P2P Grid system in order for protection from a performance degrades of load distribution.

5. Conclusion and Future Work

Our research topic belongs to an area finding a way to make more decentralized and reliable Grid computing system. Therefore, we have proposed P2P Grid as a prominent solution using Peer-to-Peer technology in this paper.

First, we have discussed the reliability analysis with a consideration to cost-optimized redundancy scheme as one part of dynamic group peering mechanism. Second, *SCL* scheduling policy which utilizes allocated resources with the most efficient way regarding completion time of task mainly has been proposed. The *SCL* is a dynamic strategy because it uses probe based feedback technology such as *PSD* to estimate variations of resource capability. Moreover, Adaptively *SCL* can handle two different types of load: divisible and indivisible depending upon *SLA* constraints.

In conclusion through performance evaluations, we have shown that effectiveness of *SCL* scheduling strategy with optimized redundant *GPs* as acceptable solutions for provision of reliable and stable service under P2P Grid system.

Acknowledgments

This work was supported by the IT R&D program of MIC/IITA [2005-S-090-03, Development of P2P Network Security Technology based on Wired/Wireless IPv6 Network].

References

[1] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. “The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems”, http://www.globus.org/alliance/ publications/papers/ogsa.pdf.

[2] Schollmeier, R. (2001), “A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications”, 2001, Proceedings of the First International Conference on Peer-to-Peer Computing, Linköping, Sweden, pp. 101-102.

[3] Jon Crowcroft, Tim Moreton, Ian Pratt, Andrew Twigg, “Peer-to-Peer Systems and the Grid”, University of Cambridge Computer Laboratory, JJ Thomson Avenue, Cambridge, UK.

[4] Ian Foster, Adriana Iamnitchi, “On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing”, Department of Computer Science, University of Chicago.

[5] Yong-Hyuk Moon, et al., “Design of P2P Grid Networking Architecture Using k Redundancy Scheme Based Group Peer Concept”, WINE’05, LNCS 3828, pp. 748-757, 2005.

[6] Raghavendra CS, Hariri S., “Reliability optimization in the design of distributed systems”, IEEE Transactions on Reliability 1985.

[7] Chung-Chi Hsieh, Yi-Che Hsieh, “Reliability and cost optimization in distributed computing systems”, VOL. 30 Elsevier Science Ltd. Oxford, UK, Issue 8 (July 2003), pp 1103 – 1119, ISSN: 0305-0548.

[8] Debasish Ghose., et al., “Adaptive Divisible Load Scheduling Strategies for Workstation Clusters with Unknown Network Resources”, IEEE Transactions on parallel and distributed systems, vol. 16, No. 10, Oct. 2005.

[9] Sameer Bataineh, Bassam Al-Asir, “Efficient scheduling algorithm for divisible and indivisible tasks in loosely coupled multiprocessor systems”, IEE Software Engineering Journal, Jan. 1994.

[10] V. BHARADWAJ, et al., “Multi-installment Load Distribution in Tree Networks With Delays”, IEEE Transaction on Aerospace and Electronic Systems VOL. 31, NO. 2 April 1995.

[11] Beverly Yang, Hector Garcia-Molina, “Designing a Super-Peer Network”, 19th International Conference on Data Engineering (ICDE'03), p. 49, 2003.

[12] Chan-Hyun YOUN, Byungsang KIM, and Eun Bo SHIM, “Resource Reconfiguration Scheme Based on Temporal Quorum Status Estimation for Grid Management”, IEICE Transaction on Communication, VOL.E88-B, NO.11, Nov. 2005.

Yong-Hyuk Moon received BS degree in Computer Engineering from Dankook University, Seoul, Korea in 2003. And He received MS degree in Information and Communications University (ICU), Daejeon, Korea in 2006. Currently he is with Division of Information Security in Electronics and Telecommunications Research Institute (ETRI), Daejeon, Korea. His research topics are related to Grid Computing, P2P networking architecture, distributed computing security, IPTV security and various networked computing issues.

Jae-Hoon Nah received M.S. degree in Computer Engineering from Chung-Ang University in 1987. He received the Ph.D. degree in Electronic and Information Engineering from Hankuk University of Foreign Studies in 2005. He is a principal research engineer and a team leader in Division of Information Security in Electronics and Telecommunications Research Institute (ETRI), Daejeon, Korea. His research interest includes distributed network security, peer-to-peer network, overlay multicasting, and IPTV security.

Chan-Hyun Youn received BS and MS degrees in Electronics Engineering from Kyungpook National University, Daegu, Korea, in 1981 and 1985, respectively. He also received a Ph.D. in Electrical and Communications Engineering from Tohoku University, Japan, in 1994. He served at Korean Army as a communications officer, first Lieutenant, from 1981 to 1983. Before joining the University, from 1986 to 1997, he was a leader of high-speed networking team at Korea Telecom (KT) Telecommunications Network Research Laboratories where he had been involved in the research and developments of centralized switching maintenance system, MOVE, and HAN/B-ISDN network test-bed. Especially, he was a principal investigator of high-speed networking projects including ATM technical trial between KT and KDD, Japan, Asia-Pacific Information and Communications University (ICU), Daejeon, Korea. Prof. Youn also was a visiting scholar at MIT, Cambridge in US since 2004. He is a vice president of Grid Forum Korea. Currently, he is interested in the Grid middleware, high performance routing, multicasting, optical Internet, and network performance measurement. He was a recipient of IEICE PAACS friendship prize, Japan, in 1994. He is a member of IEEE, IEICE, KICS and KISS respectively.