INTERNATIONAL JOURNAL OF INTELLIGENT CONTROL AND SYSTEMS
VOL. 12, NO. 3, SEPTEMBER 2007, 263-271

PHAC: A P2P-based Environment for Distributed Collaborative Applications

Adnane CABANI, Srini RAMASWAMY , Mhamed ITMI and J.-P. PECUCHET

***Abstract*— Peer-to-Peer (P2P) networks can be used in an efficient way to distribute large scale scientific problems if the issues of heterogeneity, dynamism and volatility of peers can be effectively addressed. We introduce, and illustrate the use of, a P2P-based Highly Available Computing (PHAC) framework to distributed collaborative applications while improving their dependability and performance characteristics. The objective is to offer a P2P network based solution based upon high availability to improve an application's dependability and performance. To illustrate, we use the proposed approach to compute the results of . to a certain precision.**

***Index Terms*— P2P Networks, Highly Available Computing, Distributed Computing**

1. INTRODUCTION

As contrasted with a Client/Server model, a P2P network is formed by peer nodes which function as both clients and servers. Peer nodes may differ in processing speed, main memory (RAM), network bandwidth, and storage quantity / availability. One of the important goals of P2P networks is that all clients provide some resources including bandwidth, computer power and storage space, such as every peer that arrives in the network increases the systems' capacity. It also increases the robustness in case of failures by replicating data over multiple peers, although it is to be noted that one of the drawbacks of the P2P networks is the presence of non trusted peers.

A P2P model consists of several topologies; mainly they can be categorized as either hybrid or pure [1]. Hybrid systems are characterized by presence of some servers. The server corresponds to a single group which provides an index and data of available groups. Such systems overcome some drawbacks, such as the communication bottleneck occurred in the systems with the Client/Server topology. Decentralized systems are called P2P pure. In this kind, all peers communicate symmetrically and have equal roles. They are no bottleneck because there is no special centralized server. However, the inconvenience is when searching for other peers. This topology can be further classified by the manner in which decentralization is realized: structured or unstructured.

The use of Peer-to-Peer networks (P2P) is fast-growing due to the popularity of file sharing. Hence P2P networks have sparked a great deal of interdisciplinary excitement and research in recent years [1]. The majority of these works have concentrated on topology and search algorithms [2, 3]. We can distinguish two different types of P2P networks: structured (CAN [4], Chord [5], Viceroy [6]) and unstructured networks (PRU [7], Hypergrid [8]). Normally, P2P networks are formed without any particular attention given to a peers' capability. This means that while all the peers in the network are highly heterogeneous in their characteristics (CPU, memory, bandwidth, etc.), little attention is paid to these differences amongst the peers. For their effective use in collaborative applications within the scientific community, a framework for understanding, and reasoning with, such capabilities of peers is vitally important. However, only a few research works have been reported on such applications. Hence it is important that the capability of each peer be used as a determining factor so that a user can strive to increase their application's performance.

The remainder of the paper is organized as follows: In Section 2, we briefly review structured and unstructured networks. Then in Section 3, we present the description of our problem and introduce our framework to improve the dependability of applications. In Section 4, we present the PHAC framework. Section 5 illustrates the use of our approach for computing the results of pi (π). Section 6 concludes the paper and presents future research directions.

2. BACKGROUND LITERATURE

Structured systems [4, 5] adapt efficiently as nodes join and leave the system and can respond to queries even while the system is continuously changing. Unstructured systems are characterized by a complete lack of constraints on resources. Those systems are constructed by a new peer joining the network simply by copying an existing link of another node and forming its own links. In such networks, queries of desired data are routed through the network to find as many peers as possible that have the specific data. The disadvantage of such networks is that there is no guarantee that query flooding will find a peer that has the desired data, and the high amount of signaling traffic that will exist in the network. Most of the popular P2P networks such as Gnutella [9] and KaZaA [10] are unstructured.

In the literature, one may find different structured networks, such as CAN [4], Chord [5] and Viceroy [6], which emerged in an attempt to address scalability issues of P2P networks. In

Manuscript received June 1, 2007; Revised September 1, 2007.
Mr. Cabani, Drs. Itmi and Pecuchet are with the LITIS Laboratory at INSA de Rouen in France. (E-mail : {adnane.cabani, mhamed.itmi, jean-pierre.pecuchet}@insa-rouen.fr).
Dr. Ramaswamy is with the University of Arkansas at Little Rock. (e-mail: srini@ieee.org / srini@acm.org).

structured P2P systems, the overlay network topology is tightly controlled and objects are placed at precisely specified locations. These systems use a distributed routing or hash table (DHT) [11] to provide the mapping between the identified object and its location. Queries can be efficiently routed to the node where the desired data object is located. With the advent of this class of structured systems, different research groups have explored a wide variety of applications, services, and infrastructures built on top of a DHT abstraction. Examples include systems for wide-area distributed storage, indexing, search, querying [12]. However, all of the aforementioned structured networks implicitly assume uniformity among all peers vis-à-vis their resources and capabilities; and hence are deemed equally desirable to store prospective objects. We propose to improve the performance of associated algorithms by taking into account the heterogeneous nature of nodes in P2P systems.

All existing lookup algorithms in structured P2P systems assume that all of the peers are uniform in the availability of resources. Messages are routed on the overlay network without considering the capabilities differences among participating peers. However, measurement studies [13-15] have shown that the heterogeneity in deployed P2P systems is quite extreme. The bottleneck caused by very limited capabilities of some peers therefore could lead to inefficiency of existing lookup algorithms. Existent structure formations do not take into account the heterogeneity of peers and resources requested by the user. In following paragraph, we introduce our methodology for structuring which takes into account the needs of the users and their applications as well as heterogeneity of peers.

3. Problem Formulation & Overview

Until now, existing structured P2P network solutions ignored issues of heterogeneity and volatility of peers. These two characteristics are very important for the deployment of efficient collaborative applications within the scientific community. Heterogeneity can influence execution times and hence affect volatility. Besides if one peer leaves the network, the application may become suspended and may require restarting the application. For these reasons, we define the notion of a Job and Redundancy Peer Groups, called JobPeerGroup and RedundancyPeerGroup respectively.

Definition 1: The JobPeerGroup (JPG) is a group of peers that are tasked to accomplish a particular job. The role of the JPG is to make the necessary calculations through the appropriate choice of peers which can directly influence calculations times, the reliability and the robustness of the application system.

Definition 2: The RedundancyPeerGroup is another group of peers defined to increase the reliability of P2P systems by using redundancy.

3.1 Description

Overall a peer group is made up of entities that consist of all the available peers:

$$P = \bigcup_{1 \le i \le n} P_{ci} = \bigcup_{l \in \{1,2,3\}} P_l$$

where

P_{ci} is a peer group based upon some capability criterion

P_1: all peers at level 1, P_2: all peers at level 2, P_3: all peers at level 3. Hence the set of all peers is given by:

$$P_l = \bigcup_{\substack{cc=H,M,L \\ 1 \le i \le n}} P_{ci}^{cc}$$

Peers can be grouped by specific capability criteria and by their immediate locality (see figure 1). It is assumed that there are 3 principles localities of peers: trusted partners, friends & others.

For each criterion, peers may again be grouped into three sub-groups. These are related to the specific criterion capability (cc) which can be classified in to three groups: high (H), medium (M) or low (L).

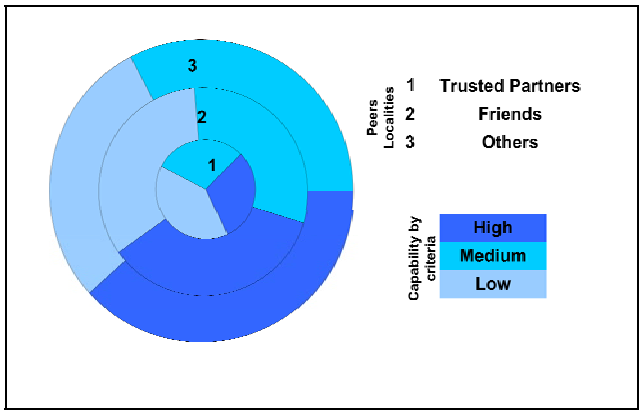


Figure 1. Organization of the P2P network

Property 1: Some peers may have access to specialized resources that may be required for the completion of a job.

$$\exists \underset{0 \le t \le m}{R_t} : R_t \in P$$

where, |P| is the number of peers in the system

We define P_{R_t} a set of all peers with access to R_t

Property 2: P_{ci} is made up of all peers that meet the specific capability criterion.

$$P_{ci} = \bigcup_{\substack{j=\{H,M,L\} \\ l=\{1,2,3\}}} P_{ci}^{j,l}$$

Property 3: A JobPeerGroup is made up of peers that meet specially stated criteria for a job. The problem is defined by the user requesting peers that meet some minimum value of chosen criteria called ExpectedJobPeerGroup (E_{JPG}).

$$E_{JPG} = \left\{ |E_p|, \bigcup_{\substack{1 \le i \le n \\ l=\{1,2,3\}}} E_{P_{ci}^{j,l}} / \; j = \{H, M, L, Undefined\} + \underset{t=0,1,\ldots,m}{E_{R_t}} \right\}$$

where

|Ep| is a number of peers requested and E_{R_t} is a set of resources required to complete the job.

3.2 Objective

The problem is to find ActualJobPeerGroup (A_{JPG}) such that cost is minimized and performance is maximized.

Algorithm 1: Formation of ActualJobPeerGroup

```
begin
REM   /* Initialization */
      A^R_JPG ← {}
      A^Pci_JPG ← {}
REM   /* Choice of peers that have specifics ressources */
      forall elements of E_Rt do
          P_i ∈ P_Rt : ∀i, j; i ≠ j; 0 ≤ i, j ≤ |P|; l, l' = 1, 2, 3 tel que P^l_i ≤ P^l'_j ;
          A^R_JPG ← A^R_JPG ∪ P_i;
      end
REM   /* Choice of peers that have required ressources */
      foreach E_Pci <> "undefined" do
          P_x ∈ P_ci : ∀x, y; x ≠ y; k = H, M, L; P^ci_x ≤ P^ci_y et P^ci_x = k;
          A^Pci_JPG ← A^Pci_JPG ∪ P_x;
      end
      return A^R_JPG + A^Pci_JPG
end
```

a. JobPeerGroup

Algorithm 2: Formation of unclustered RedundancyPeerGroup

```
begin
    return P − AJPG
end
```

b. RedundancyPeerGroup

Algorithm 3: Formation of clustered RedundancyPeerGroup

```
begin
REM   /* Initialization */
      UnRPG ← P − AJPG
      CRPG ← {}
      forall peer of UnRPG do
          score[peer] ← 0;
      end
REM   /* Score calculation */
      forall peer of UnRPG do
REM       /* Each criterion have one predefined priority p_i; ∑ p_i = 1 */
          foreach criterion do
              score[peer] ← score[peer] + p_i * P_ci;
          end
      end
REM   /* Clustering */
      CRPG ← QuickSorting(score)
      return CRPG
end
```

c. Clustered RedundancyPeerGroup

Figure 2 JobPeerGroup and RedundancyPeerGroup Formation

$$A_{JPG} \approx E_{JPG}$$

Subject to: Cost(AJPG) is minimum and Perf(AJPG) is maximum. So, the ActualJobPeerGroup is given by:

$$A_{JPG} = \left\{ E_p \middle| \leq |A_{JPG}|, A_{JPG}^{R} + A_{JPG}^{P_{ci}} \right\}$$

To find ActualJobPeerGroup (A_{JPG}), and the RedundancyPeerGroup we use the algorithms shown in Figure 2. Figure 2a depicts the algorithm to form the actual job peer group, while 2b and 2c illustrate the algorithm for an unclustered and an clustered formation for the redundancy peer group. In Figure 2c, the algorithm uses an optimizing function based upon the capability criterion of interest to the application that utilizes the P2P network.

4. THE PHAC FRAMEWORK

To offer users a P2P network based on high availability, we propose the P2P-based Highly Available Computing Framework (Figure 3). The user needs to define the criteria appropriate for application needs (for example: say the performance of some distributed calculation). The choice of peers participating in this calculation can directly influence calculations times, the reliability and the robustness of the system. Thus, the choice of peers forming the A_{JPG} is crucial. Our framework, implemented on JXTA [16, 17], searches the available list of peers and create two peer groups (JobPeerGroup and RedundancyPeerGroup) [18].

The role of a JobPeerGroup is to make the necessary computations for the application under consideration. The distinguishing characteristics of peers belonging to a Peer Group may include: *(i)* High bandwidth, *(ii)* Computer resources (Available CPU, free memory size, free storage size), *(iii)* Uptime and availability, *(iv)* Round-Trip Delay Time.

Reliability in P2P systems is a hard problem. The inherently distributed nature of peer networks makes it difficult to guarantee reliable behavior. The most common solution to addressing reliability across P2P systems is to leverage redundancy. Redundancy is introduced in the PHAC framework in the following manner. Every peer which does not belong to a JobPeerGroup is assigned to the RedundancyPeerGroup if it satisfies similar distinguishing characteristics conditions for peers (clustered mode), namely, high bandwidth, availability of computer resources (free disk space, etc.), uptime and availability.

Since response time between peers is a very important issue in any collaborative application process these characteristics influence the group formation by giving priority to both bandwidth and round-trip delay time. Uptime is an interesting parameter that can be a strong influencing factor. Let us consider an enterprise's machines as peers during the administrative schedule if these machines are turned off at closing time then it is more suitable to not attribute works to these ones before certain period of closing time. On the contrary if the machines are left available, it is more interesting to use them when they are idle. To implement this, every peer has a dashboard with its available characteristics. A weekly average of schedules and connection time are calculated. All these characteristics are saved in a XML file. The dashboard will concentrate the resources inactivity (CPU, memory and bandwidth) to be uploaded at the first consultations by a peer.

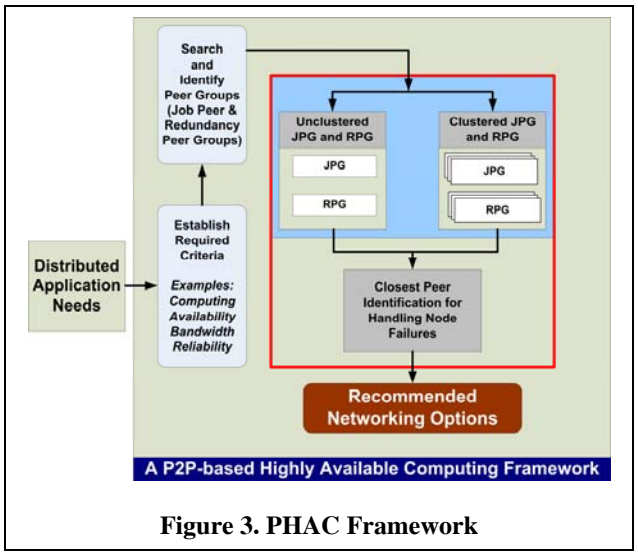


Figure 3. PHAC Framework

Every peer belonging to a JobPeerGroup is connected to one or more peers belonging to the RedundancyPeerGroup. It depends of the strategy adopted by developer. A JobPeer sends a copy of its state to RedundancyPeer at predefined steps in the process. When a JobPeer does not respond, its duplicate will either send the data to a new JobPeer or move over to the main JobPeerGroup. This allows the uninterrupted progress of the calculation.

In order to be able to detect the faults, various methods are presented in the literature. We chose to adopt the gossip-style protocol [19] due to its good scalability properties. With this model, the detection of the faults is distributed and resides at each peer on the network. Thus, the choice of this protocol is in agreement with the distributed topology adopted by our platform. Each peer maintains a table for each known member its addresses and an integer which will be used to detect the failure detection. This integer is one counter commonly called *heartbeat counter*. Every step, each peer increments its own heartbeat counter and selects another peer at random to send its list to. As soon as the message containing the list is respected, the peer merges the list which is received in the message with its own list, and adopts the maximum value of the heartbeat counter. Each peer also maintains, in its list for each other peer, the last time that its heartbeat counter has incremented. If the heartbeat counter of the messages, for the same entry, has not increased after a certain period, this peer is considered failed.

To build and structure the network appropriately, it's possible to choose between two methods: unclustered or clustered JobPeerGroup and RedundancyPeerGroup. The clustered approach increases the systems' dependability. Once the group of peers are identified, the closest peer is identified for handling node failures. Finally, it proposes a recommended network for the proposed job to the user. The user can modify

the structure of network before using it to distribute their application.

4.1 Distributing an Application Using PHAC

The following useful implementation terms are used in this section:

- MasterPeer, it is the peer to which a user connects and solicits the use of the P2P network.
- RequestPeer solicits calculation.
- ExecutantPeer is free until it is requested by a RequestPeer to carry out a calculation. It becomes free once it sends back the result.
- JobPeerGroup is the group formed by the peers which are tasked to perform the calculation.
- RedundancyPeerGroup is the group formed by the peers which stores the calculations according to their evolution with the goal to increase the systems' dependability.

Our approach to distribute an application on P2P network is represented in figure 4. The steps include:

1. Construct a P2P network with both groups - JobPeerGroup and RedundancyPeerGroup.
2. Divide application into sub-tasks.
3. Every task is sent in ExecutantPeer belonging to JobPeerGroup.
4. In every step of predefined time, results are recorded on peers belonging to RedundancyPeerGroup.
5. Once task was performed by ExecutantPeer, result is sent to MasterPeer.
6. As soon as all results of distributed tasks are recovered by MasterPeer, display the result to the user.

5. A Performance Analysis Model

5.1 Notation

We defined some parameters as follows:

Three different types of processors:

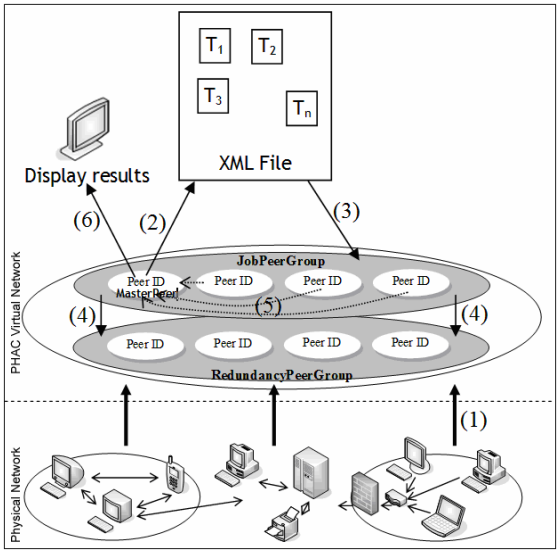


Figure 4. Distributing an application in PHAC

- H : very fast processor element.
- M : fast processor element.
- L : slow processor element.

Let P_l^j denote number of peer belonging l^{th} zone with processor of type j ($l \in \{1,2,3\}$; $j \in \{H, M, L\}$).

We define T_H , T_M and T_L the execution time of one instruction on P^H, P^M and P^L, respectively. T_H , T_M and T_L have to satisfy the following condition: $T_H < T_M < T_L$

Let N represent the total number of instructions that will be executed when running a program.

N_i is the number of instruction executed by P_i.

5.2 Execution time

The execution times of a program on one peer and P peers are:

Case 1: one peer with processor of type H

$$ET_1 = N.T_H \qquad (1)$$

Case 2: with P peers belonging locality 1, locality 2 and locality 3

$$P = \sum_{i=1}^{3} \left(P_i^H + P_i^M + P_i^L\right)$$

$$N = \sum_{i=1}^{P} N_i$$

$$ET_2 = \max_{i=1,\cdots,P} (N_i T_i + t_i) \qquad (2)$$

where t_i is the time to do one transfer between master peer and the others peers. It is influenced by bandwidth capacity (high, medium or low). From equations 1 and 2, the speed-up of case 2 with respect to case 1 is:

$$S_1 = \frac{ET_1}{ET_2} = \frac{N.T_H}{\max_{i=1,\cdots,P} (N_i T_i)}$$

In the best case, if $N_i = \frac{N}{P}$ and $P = P_1^H$ then the speed-up is P times. In the case where one peer exit the P2P network, $ET_2 \to \infty$ and $S_1 \to 0$. For this reason, we propose data redundancy. This allows taking back the calculation at the point at which is stopped.

6 Case Study

In this section, we present some results using our methodology. We chose to calculate π by using the expression BBP (Bailey-Borwein-Plouffe) which allows calculating up to the umpteenth decimal of π.

$$\pi = \sum_{k=0}^{\infty} \frac{1}{16^k} \left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6} \right)$$

Earlier we have implemented this algorithm with collaborative approach on a multiagent system [20]. Each agent has a role to calculate iteratively the decimal value of π between two given indices. The choice of this example for a case study is due to the fact that it is since easier to interpret the result of our framework due to its deterministic nature. The implementation was done on 10 computers interconnected by a Fast Ethernet 100 Mbps network where eight computers with a 2 Ghz processor and 512 MB RAM. Two other computers were equipped with 600 Mhz processor and 256MB RAM.

The first step is the building of network. The JobPeerGroup is formed out of the eight computers of 2Ghz and the RedundancyPeerGroup is composed of the two other computers. The calculation is divided into sub-tasks defined in a XML file. This decomposition takes into account appropriate load balancing. Figure 5 represents the precision of π computing. A precision of 3000 means the calculation of the 3000th decimal of π. The Y axis represents the calculation time. We tested three different configurations: one peer, four peers and eight peers. For every precision (500 to 5500) we performed calculation on each of configurations. We point out that the time of calculation increasing according to precision. The distribution calculation grows almost linearly in comparison with the number of peers used due to appropriate load balancing between the different peers. The speed-up and efficiency for the 4 and 8 peer cases are illustrated in Figure 6 and Figure 7 respectively.

Speed-up is an important indicator of the degree of improvement in the calculation time using a parallelized approach in comparison with adopting a sequential approach. A speedup of less than 1 would mean that an iterative sequential solution would give better results than a parallelized solution. For a precision of 5500, the speed-up is equal at 7.67 times in the case of distribution on 8 peers. This verifies the analysis model presented at the section 5.2. In fact, theoretically, the speed-up can be up to 8 and the difference between the possible and actual speedup is due exchange of messages between peers.

Efficiency is another performance indicator that links speed-up to the number of peers participating in calculation. In our case, speed-up is divided by four or eight according to the number of peers used. So to achieve a precision of 5500 for π, we obtained a 97% (4 peers) and 96% (for 8 peers) efficiency. Hence the observed results show that the parallelized implementation performing better.

To fully complete the testing of our approach, we study the scenario of a fault on participating peers during the calculation process. This mimics the real-world case wherein peers are volatile and it is necessary to consider issues such as fault tolerance. In the following, we present the results obtained for a precision of 5500 with eight peers used for calculation. In the figure 8, we observe three curves: the first one represents the ideal case where we have no faulty peers until the calculation is

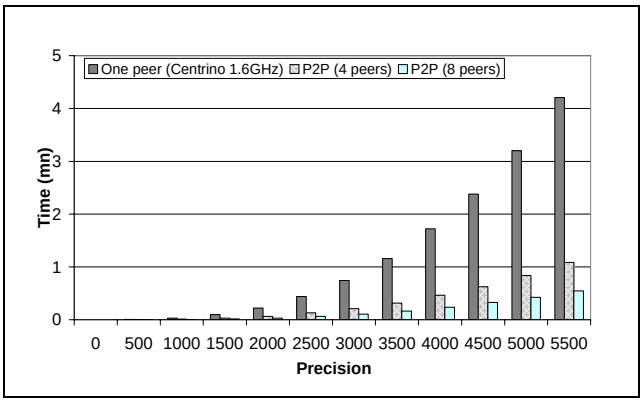


Figure 5. Calculation time of π computing

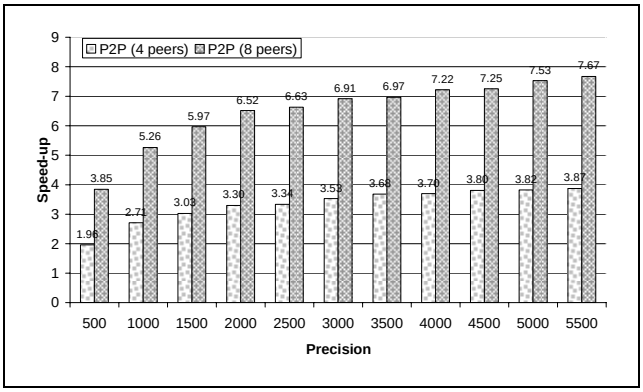


Figure 6. Speed-up

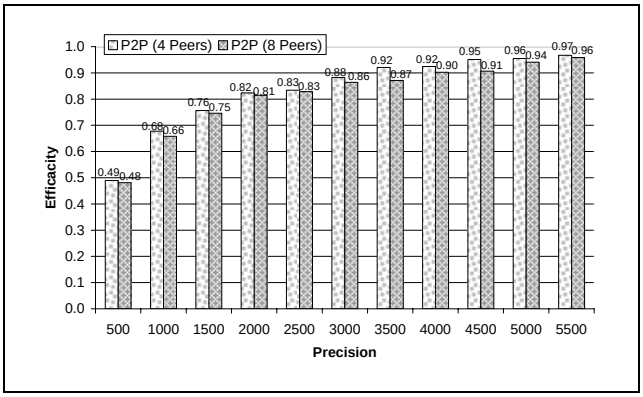


Figure 7. Efficiency

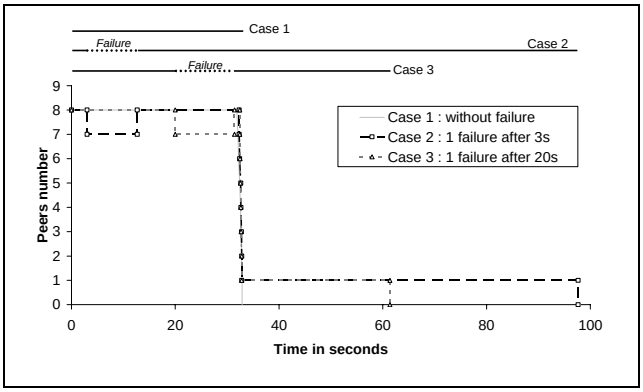
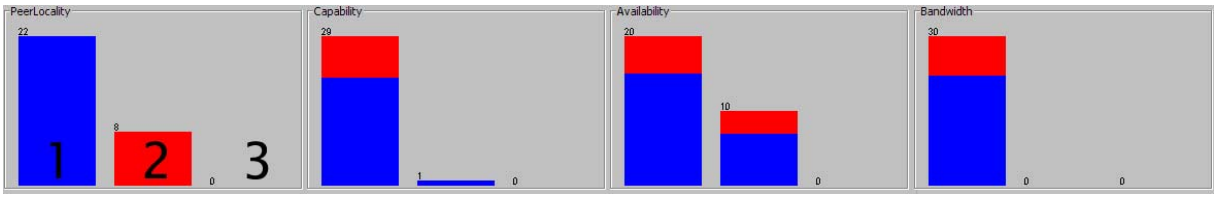
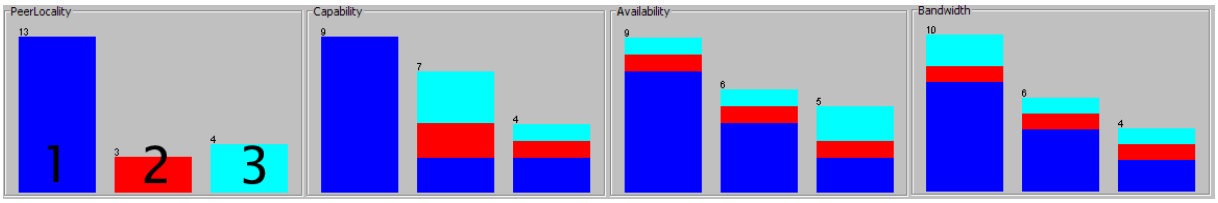


Figure 8. Calculation time of π with fault tolerance



a. JobPeerGroup



b. RedundancyPeerGroup

Figure 10. Peers' capabilities build the Network

completed, the second and the third represents the calculations at the time of a fault. These are simulated to occur after three seconds and twenty seconds of the starting the calculations.

One can observe that when the fault occurred after three seconds the total calculation time is 100s. In the case where the fault occurred after twenty seconds, the calculation finishes in 61s. This is due to the use of a RedundancyPeerGroup; which can substitute adequately for the lost peer. Also the observed results are better than in the case of a sequential calculation. And, it's also better than entirely repeating the calculation with parallel approach. To note that in case of the availability of RequestPeers, i.e. free peers belonging to JobPeerGroup, the restart of calculation would have been made by these peers after they recover the data replicated by the peer belonging to the RedundancyPeerGroup. The calculation time would then be better than the case wherein one RedundancyPeer becomes the ExecutantPeer.

To simulate the clustered method, a lot of variations are possible. We chose a case wherein we have 50 total peers, where 30 peers belonging to JobPeerGroup and 20 belonging to RedundancyPeerGroup defined in accordance with the capability distribution as shown in figure 9, accounting for different capability characteristics and differing locality of peers.

To study the clustered approach, we distribute the calculation of π to reach a precision of 8000. We 'bring down' one peer every 10 sec in order to try the robustness of the system and especially the applicability of the clustering mechanism and the identification of the closest peer for handling a peer node failure. In the observed results, as shown in figure 10, we see three curves. The first one is for the case where we induced no faults. The second represents the calculation time in the case which have some failures. When one peer leaves the network, we choose the closest peer (cluster ranked using algorithm 3). The third curve represents the calculation time of the same (as case 2) but without using a clustering scheme i.e. when one failure is detected one available peer belonging RPG replaces it without any consideration of its capability characteristics.

We can readily observe that the results in the case of clustered approach are better than an unclustered scheme. This is because when we replace one faulty peer node, we chose the closest peer in terms of its capabilities from the RPG.

6. CONCLUSIONS

In this paper, we presented a simple and effective framework (PHAC) to improve the dependability of applications executing over a P2P network. We showed its need and feasibility using a simple application for demonstration purposes. We showed the results when we selected peers to form JobPeerGroup and RedundancyPeerGroup and the case when we improve the selection process with clustering the peers in the JobPeerGroup and RedundancyPeerGroup. This latter case allows us to improve the dependability of applications such as that envisaged for the π calculation. The clustered case of JobPeerGroup and RedundancyPeerGroup improves the dependability and performance of the users' application. These

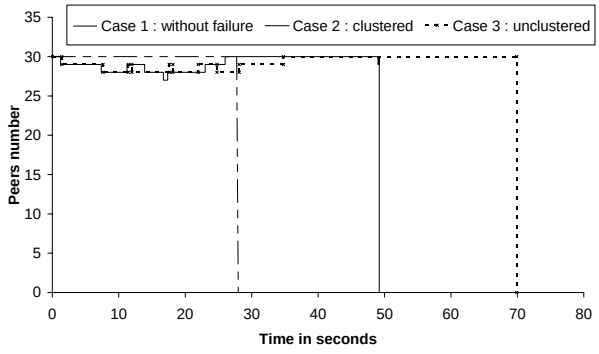


Figure 9. Calculation time, comparison with clustered and unclustered case

results are encouraging and are similar to that of distributed ants simulation with agent modeling [21]. We will continue our study of this framework and its applicative value in other types of applications. We also plan to extend this with structured peer groups and the case with increasing the number of peers in larger-scale peer clusters.

7. ACKNOWLEDGEMENTS

This material is based in part, upon work supported by the National Science Foundation, under Grant No. CNS-0619069. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu, "Peer-to-Peer Computing," HP, External HPL-2002-57, 2002.

[2] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the Internet topology," in *Applications, technologies, architectures, and protocols for computer communication (ACM SIGCOMM)*, Cambridge, Massachusetts, United States, 1999, pp. 251-262.

[3] M. Newman, "The structure and function of complex networks," *SIAM Review 45*, pp. 167-256, 2003.

[4] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Adressable Network," in *ACM SIGCOMM 2001*, San Diego, California, United States, 2001, pp. 161-172.

[5] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications," in *ACM SIGCOMM 2001*, San Diego, California, United States, 2001, pp. 149-160.

[6] D. Malkhi, M. Naor, and D. Ratajczak, "Viceroy: A Scalable and Dynamic Emulation of the Butterfly," in *21st ACM Symposium on Principles of Distributed Computing* Monterey, California, United States, 2002.

[7] G. Pandurangan, P. Raghavan, and E. Upfal, "Building Low-Diameter P2P Networks," *IEEE Journal on Selected Areas in Communications*, vol. 21, pp. 995-1002, 2003.

[8] F. Saffre and R. Ghanea-Hercock, "Beyond anarchy: self-organized topology for peer to peer networks," *Resilient and adaptive defense of computing networks*, vol. 9, pp. 1076-2787, 2003.

[9] "Gnutella," http://www.gnutella.com.

[10] Kazaa: http://www.kazaa.com/.

[11] C. Yoshikawa, B. Chun, and A. Vahdat, "Distributed Hash Queues: Architecture and Design," in *3rd International Workshop on Agents and Peer-to-Peer Computing*, New York City, USA, 2004.

[12] B. Hari, M. F. Kaashoek, K. David, M. Robert, and S. Ion, "Looking up data in P2P systems," *Communications of the ACM*, vol. 46, pp. 43-48, 2003.

[13] S. Saroiu, K. Gummadi, and S. D. Gribble, "A measurement study of peer-to-peer file sharing systems," in *Multimedia Computing and Networking*, San Jose, California, United States, 2002.

[14] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient overlay networks," *ACM SIGOPS Operating Systems Review*, vol. 35, pp. 131-145, 2001.

[15] M. Srivatsa, B. Gedik, and L. Liu, "Large Scaling Unstructured Peer-to-Peer Networks with Heterogeneity-Aware Topology and Routing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, pp. 1277-1293, 2006.

[16] JXTA-v2.0, "JXTA v2.0 Protocols Specification," http://spec.jxta.org/nonav/v1.0/docbook/JXTAProtocols.html.

[17] JXTA-v2.3.x, "JXTA v2.3.x: Java™ Programmer's Guide," http://www.jxta.org/docs/JxtaProgGuide_v2.3.pdf.

[18] A. Cabani, M. Itmi, and J. P. Pécuchet, "Improving Collaborative Jobs in P2P Networks " in *12th IEEE International Conference on Electronics, Circuits and Systems (ICECS'05)*, Gammarth, Tunisia, 2005, pp. 135-138.

[19] R. V. Renesse, Y. Minsky, and M. Hayden, "A gossip-style failure detection service," Ithaca, NY, USA, Technical report 1998.

[20] A. Cabani, M. Itmi, and J.-P. Pécuchet, "Multi-agent Distributed Simulation : Discussions and prototyping a P2P architecture," in *Computer Simulation Conference (SCSC'05)*, Philadelphia, Pennsylvania, United States, 2005, pp. 281-286.

[21] A. Cabani, M. Itmi, and J. P. Pécuchet, "Distributed Multiagent Simulation on P2P Architecture," in *11th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications (IEEE/ACM DS-RT 2007)* Chania, Crete Island, Grece, 2007.

Adnane Cabani received the M.S. degree in computer science from the National Institute of Applied Sciences (INSA), Rouen, in 2003. Since 2003, he has been with the Computer Science, Information Processing, and Systems Laboratory (LITIS). His research interests include multiagent system, modeling and simulation, networks and distributed system. He was an invited visiting research exchange scholar at the Department of Computer Science at University of Arkansas at Little Rock, during 2007.

Dr. Srini Ramaswamy is currently the chairperson of department of Computer Science at University of Arkansas at Little Rock. Currently, he also serves as the program manager for the wireless nano sensors and systems center as part of the Arkansas ASSET program, an NSF-funded initiative. Earlier he was the Chairperson of the Computer Science Department at Tennessee Tech University. His research interests are on intelligent and flexible control systems, behavior modeling, analysis and simulation, software stability and scalability; particularly in the design and development of better software systems for real-time control issues in manufacturing and other distributed, real-time applications. He has actively consulted on the design, development and implementation of a knowledge capture, classification and mining project with eFutureKnowledge, Inc. and on a NIST sponsored ATP project with InRAD, LLC. He was an invited visiting professor at the National Institute of Applied Sciences (INSA) at Rouen, during 2006 and 2007. In the summers of 2003, 2004 and 2007, he has been a visiting research professor of Computer Science in the Institute of Software Integrated Systems (ISIS) at Vanderbilt University. In 1994-1995, and subsequently during the summer months of 1996 and 1998, he was a post-doctoral research fellow / visiting scientist in the Laboratory for Intelligent Processes and Systems (LIPS) at the University of Texas at Austin where he helped with research efforts on Sensible Agents. Dr. Ramaswamy earned his Ph.D. degree in Computer Science in 1994 from the Center for Advanced Computer Studies (CACS) at the University of Southwestern Louisiana, Lafayette, LA, USA (now University of Louisiana at Lafayette). He is member of the ACM, Society for Computer Simulation International, Computing Professionals for Social Responsibility and a senior member of the IEEE.

Dr Mhamed Itmi is an Assistant Professor at INSA-Rouen, France and Adjunct Research Associate Professor at CSU-Chico, CA, USA. He earned a Ph.D. degree in Probability theory and Statistics in 1980 and a second Ph.D. degree in Computer Science in 1989. His research interests are on distributed systems, modeling and simulation, performance evaluation of Discrete Event Systems. He managed different research projects in Logistics and Transportation domains particularly for container terminals. Dr Itmi is a member of the SCS (The Society for Modeling and Simulation International). He is the Director of the RIC (Rouen INSA Center), a French node of the SCS international network MISS (McLeod Institute of Simulation Sciences).

Dr Jean-Pierre Pécuchet is computer scientist, full Professor and Chief Information Officer of INSA Rouen (the Normandy's implantation of the biggest French public engineering school network). His research fields are in discrete systems Modeling and Simulation, Education & Training systems, Knowledge Engineering and Information Retrieval. He is involved in many industrial partnerships, and also contributes to many conferences scientific committees.