A Template Based Static Coalition Protocol – A^{3P}viGrid

Avinash Shankaranarayanan[1], Frank Dehne[2], Andrew Lewis[3]

[1][3]School of ICT, Griffith University, Brisbane, Australia.
[2]School of Computer Science, Carleton University, Ottawa, Canada

```
Avinash.Shankaranarayanan@student.griffith.edu.au,frank@dehne.net,
                  A.Lewis@griffith.edu.au
```

Abstract

Problems such as resource and service discovery models, load balancing and scheduling, brokering are eminent in grid systems due to bottlenecks such as bandwidth and network traffic in the underlying communication infrastructures and their associated costs in fabricating a scalable and cost effect grid services infrastructure. The primary goals of this paper is to apply grid based coalition formation concepts to agents: add efficient load balancing and scheduling (A^{3P}viLoad Scheduler) schemes; provide a replacement solution to resource discovery models by applying application oriented directory services; minimize message passing of state information updates and economic brokering services to the agent aware adhoc p2p virtual interconnect grid computing system or A^{3P}vigrid system.

Keywords: Grid, scheduling, coalition formation.

1   Introduction

A primary problem that effects the use of efficient grid computing systems is the discovery and usage of high performance grid computing applications. Most of the time, researchers involved in using high performance compute nodes tend to use custom developed or commercial software for efficient distributed computing such as Clustermatic or ROCKS. Here we give a brief introduction to the various techniques and technologies that will be used in designing the A^{3P}vigrid System. As defined by IBM, agents are suitably noted for following an agenda or schedule that is efficiently executed based on an environmental experience gained by the agent(s). It is this ability to intelligently execute user oriented or goal oriented objectives that makes agents far superior to normal programs or non-agents.

As denoted by Stan Franklin (1996), agents are classified based on their properties such as reactive, autonomous, temporally, goal oriented, communicative, socially able, learning, adaptive, mobile, flexible as characteristic properties that they adhere to. Similar properties can be observed from that of a web server to be recognized as a reactive, autonomous, temporally, communicative and goal oriented agent. Such agents are categorized in the form of multi agents adhering to different properties that are utilized to achieve different goals based on their respective environments.

2   Coalitions in Agent Based Grids

A coalition, in the context of agent-based systems, is defined to be a group of agents that come together to solve a common task or achieve a common objective. The coalition concept has its roots from game theory where players {agents} form groups and plot a strategy for winning a game. In general, with respect to agent based systems and game theory, coalition formation occurs on the fly where agents tend to form groups to achieve a common goal such as job processing or maximizing their utility value. Here, with respect to grid computing, we define two new concepts called static coalition and dynamic coalition in agents based grid systems.

Static coalitions are typically formed on the basis of more persistent common goals and tasks, and are less likely to change from problem to problem.

Dynamic coalitions, on the other hand, are groupings that are formed to address the needs of a specific task or common objective. Once these tasks are completed, or the common objectives met, dynamic coalitions tend to disband, and re-form in different ways. In A^{3P}vigrid we use static coalition formation techniques for effective job processing and aggregation of resources available. Our fundamental premise is that coalition mechanisms add value in the context of agent-based grids for the following reasons:

• Coalitions of peers can reduce the communication overhead involved in executing complex tasks and services which require the use of multiple peers.

• Coalitions of peers can result in better matching between the requirements of tasks/services and the infrastructure that is made available to execute these. For instance, an appropriate coalition formation mechanism can put together a collection of peers with similar platforms and QoS characteristics that are best suited for a given task.

• Coalition formation mechanisms can be used to optimize complex trade-offs between the objectives of maximizing the utility of the service requester(s) and the service providers. For example a service requestor could be maximizing its payoff for the given task by being an intermediary service provider that outsources its job to

third party agents thus maximizing his individual utility value.

• Coalition formation mechanisms can economically increase system throughput as a whole. After some negotiation among agents, tasks will be allocated to appropriate coalitions who can execute them with minimal costs and time. Thus agents seem to be better off. A good example of this would be the formation of coalition among agent in a local Linux cluster where the maximum payoff is achievable with minimal communication costs.

3 The A^{3p}viGrid System Overview

The A^{3p}vigrid system is a generic architectural schematic that tries to incorporate existing technologies such as peer to peer computing, multi-agents systems, efficient load balancing and scheduling, optimal network routing, web services and decision making based on intelligent and scalable solutions to the discovery of web based grid resources and application services, based on coalitions of P2P agents. For the sake of readability and understanding of the architectural scheme, the system is subdivided into sections where each section is a key functional part of the A^{3p}viGrid System.

3.1 The Peer to Peer (P2P) Component

Peer to Peer: "A set of applications or entities having a varied set of attributes (QOS factors) that takes care of discovery, utility and aggregation of resources such as storage, CPU cycles, content or information, human presence or user agents etc, that are available on a loosely connected, decentralized, scalable, heterogeneous and transparent network such as the Internet."

Figure 1: Functioning of a Peer-to-Peer System

As shown in the Fig 1, peers are standalone autonomous nodes very similar to the attributes of agents and agent based system. Usually there is no set topology fixed for a peer network as it is assumed that every peer in the local area network based on the local services tends to know every other peer in its domain. The success of the discovery of the peers needs some form of state information to be maintained to attain discovery and negotiation with the nodes. A directory service provides ample evidence of the existence of the peers by simple authentication and maintenance of peer location information that is very similar to the methods followed in the A^{3p}viGrid schematics where an agent based peer manager is utilized for agent based service discovery and negotiation which in turn forms coalition as seen in section 4.

3.2 Applying Peer to Peer concepts in A^{3p}viGrid System

The Fig 2 shows the indefinite role played by peer-to-peer computing in the A^{3p}viGrid System. Based on a decentralized scheme the peers use a partly centralized directory services infrastructure called the agent based peer manager or APM.

Agent Based Peer Manager [APM]: "A directory server that serves as an intelligent infrastructure that authenticates and stores state information of the peers and its agents using a location oriented and service based ordering, that helps in the negotiation and formulation of coalition in agents and agent based services."
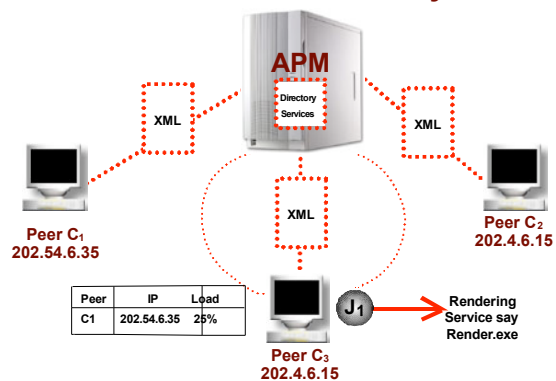

Figure 2: A Peer to Peer system

As shown in Fig 2 the Agent Based Peer Manager keeps track of peers that are pooled based on the services offered by the peers and its location. The location of the peers plays an important role in the turn around time of a job or task that is to be processed. That is it makes sense to have the peers as close as possible to have a minimalist turnaround time which adds to the time factor involved in distributed processing of the job and the results sent back to the originator of the job. A Job J1 is originated from a peer called C3 that is willing to distribute the job to other peers for processing due to overloading that has occurred in C3. The Job J1 needs a service called render.exe that is required for the rendering of an image file in C3. Hence the peer C3 authenticates and downloads a list in this case an XML file that is parsed using an XML parser and information of the peers in its local area network is used for the discovery of nodes having render.exe service at the closest location available. Based on this information obtained from the APM, Peer C3 offloads its job J1 to the closest peer that has better performance based on a load-balancing algorithm such as A^{3p}viload [8] [10] as researched previously and discussed in Section 6.

3.3   What is Coalition formation?

Coalition

{A,B,C,D,E,F,G,H} ······Nodes

{A,B}, {C,D}, {E,F}, {G,H}

Coalition based on Mutual
Interest Policy say Resources

Goal based   ······{A,B} == Achieve Goal 1
on
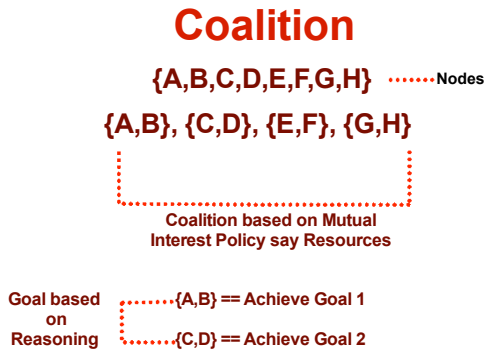Reasoning   ······{C,D} == Achieve Goal 2

**Figure 3: Represents an Example of Coalition in peers
A, B, C, D, E, F, G, H**

The term coalition is an acronym for grouping or clustering. Agents form groups or coalitions based on some form of common goal or interest. A good example of this would be politics where each party considers its position and then tries to form a coalition to achieve its goal namely political powers such as forming of a ministry or government body. Here the goal of each party or agent group was to come to power using coalition. The same example can be applied for sub-coalition or sub group formation.

Each party is represented by a group of individuals who are autonomous. That is they join the party of their own free will based on a mutual interest policy. Although each person has his or her own goals, the primary goal that brings them together is a common goal namely achieving a political objective such as Member of Parliament, Minister, etc.

Hence every individual forms a group or coalition to become members of the party which here can be termed as a sub-group or Sub-Coalition. The party, based on turn of events, forms a coalition based on mutual interest with other parties or subgroups to form a Coalition. The example provides a good overview of how agents should form Coalition based on mutual interests and common goals.

3.4   Coalition Formation in Peers/Agents

As mentioned in previous sections, agents tend to be autonomous. It is the autonomicity attribute of the agent that makes it unique from normal programs or services. As shown in Fig 3 a set of nodes are working in a common domain space close to each other. Each of the nodes is represented by a set number of agents.

Each node is represented by an agent as say A, B, C, D, E, F, G, H as shown. Coalition is form between the agents based on mutual interests and commonality of goals. It is this commonality that tends to form the coalition. As shown the sets {A, B}, {C, D}, etc are formed based on a Goal based reasoning, where the coalition is formed based on the commonality of the Goals of the agents. The commonality goal factor could be anything from resources, services, brokerage, payoff, interests, etc.

3.5   Coalition formation Methodology

- Rules for formation of coalitions

- Utility or self interests that are goal based such as payoff, resources, services, etc.

- Commonality of resources and services such as operating systems, hardware and software resources etc, and self-interest based on a goals or utility values, that are directly proportional to payoffs.

- Coalition trust and beliefs

- Closest neighbours or shortest path of nodes based on turn around time, domain, or topology.

- Quality of service[QOS] and reliability factor

3.6   How is Coalition Formed in Agents?

Usually, coalitions are formed between agents based on the utility value. When applying agent based coalition concepts to grid computing the commonality factor is taken into consideration.
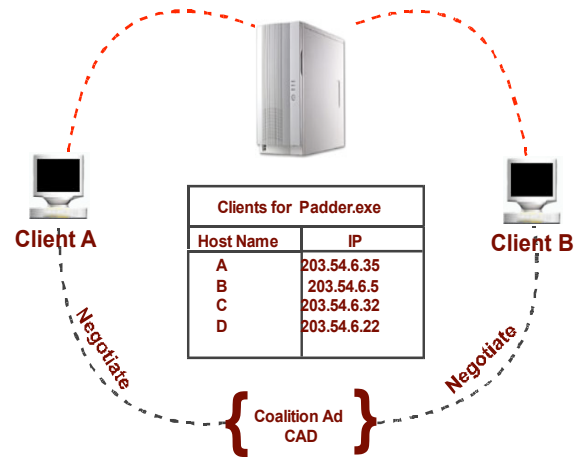
Clients for Padder.exe	
Host Name	IP
A	203.54.6.35
B	203.54.6.5
C	203.54.6.32
D	203.54.6.22

Client A   Client B

Negotiate   Negotiate

{ Coalition Ad
CAD }

Figure 4: A Peer to Peer based Grid System

As shown in Fig 4 it is assumed that Clients A and B are part of a common domain and they register their locations using the APM (Winjngaads, 2002). On registering they come to know about their existence and a Coalition Advertisement Token or CAD that is sent by either one of them based on commonality of services.

Here the commonality between A and B is the Padder.exe services offered by both of them. The APM tends to become a broker for A and B which brings together the formation of the coalition of the two nodes based on commonality and mutual interests.

Here the assumption is that some form of agent communication languages such KQML (Finn, 1994) is used for negotiation, communication and formation of the coalition.

3.7 Need for Coalition with a real world example

The primary need for coalitions is based upon the final individual utility value and the payoff gained by the agents. As defined before payoff can be a value such as money, brokerage, resources, learning, etc for an agent.

3.7.1 Sandal maker Example:

A good example can be expressed from two companies manufacturing quality sandals and shoes. Let us take from Fig 4 that A and B are two companies or agents that are making these sandals. Agent A represents the manufacturing of shoe/sandal uppers and Agent B represents the manufacturing of soles/bottom sandals. The utility value for both A and B is Zero unless a coalition is formed. Hence based on a mutual benefits namely the payoffs, {A, B} coalition formation takes place to form pairs of sandals for the market say X. Thus by utilizing the above sandal example applying coalitions in multi agent based grid computing has a different perspective. Here the utility value would be represented as more jobs for processing and more the jobs processes more the payoff such as money or resources gained.

3.7.2 Coalition Subscription

Just like subscription to a magazine, an agent or group of agents registering with the APM needs to form or join a coalition based on subscription. Rules of Subscription could defer from Coalition to Coalition based on mutual agreement and leadership proclaimed by the agents.

In the following sections we talk about how agents join or leave a coalition and what kinds of state information need to be updated during and after Subscription or removal of these agents.

3.8 Agent Joining a Coalition

As shown in Fig 5, agents depending on the language (Finn, 1994) tends to communication and negotiate based on a rule set that is followed in the collaboration of these agents.

Here it is assumed that each node is represented by one or more agents and a common coalition formation construct is used based on set rules. The rules could be different for each agent based on its environment and self-interests. The commonality factor plays a vital in the collaboration and formation of coalition in agents.

Let us assume that R is a node represented by an agent. The commonality namely padder.exe services between the agent R and the Coalition C1 that constitutes a previously formed coalition in nodes A, B, C, D leading to negotiation and joining of agent R in Coalition C1 based on utility factor.
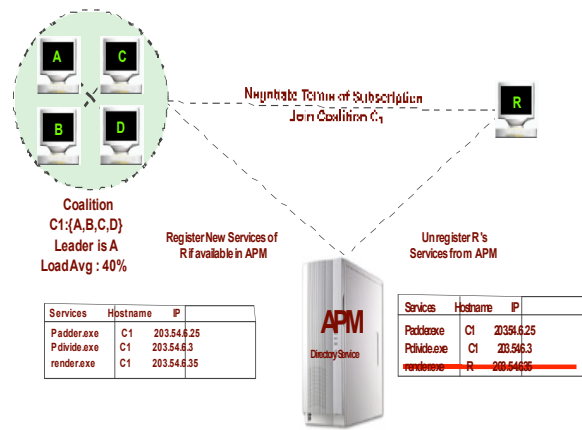

Figure 5: An Agent Joining the Coalition

Here the APM forms a directory services agent that negotiates and stores information of individual peers and Coalitions available in the immediate domain space. Here R finds interests in joining the coalition group C1 where a negotiation takes place based on utility. The agents in the coalition form a decision on the acceptance or rejection of R to joining the Coalition C1 based on coalition interests. Let U be the utility value for remote node R and Coalition C1. U{R} is the utility value for R. U{C1 : (A,B,C,D)} is the utility value of the Coalition C1 where agents A, B, C, D each have individual utility values such as U{A}.

If U{R} < U {C1: (A, B, C, D) then U {C1 : (A, B, C, D, R)}and a coalition is formed

If U{R} > U {C1: (A, B, C, D) then U {C1: (A, B, C, D)} then R does not form a coalition as its utility value is better by being a autonomous agent.

Here the utility factor is defined as the more jobs obtained the more utility value namely money is obtainable for the agent. Hence a coalition is formed to increase the utility value of an agent or coalition of agents. Factors such as the increase in resources, payoffs and additional services play a vital role in the decision factor of accepting or rejecting an agent system. With reference to Fig 6 after negotiations R tends to join the coalition group C1. A "un-register" message is sent to the APM which updates the required tables for the removal of the services of the agent R. Similarly upon joining a register service becomes a possibility if new services are rendered based on the entry of the new agent node R. C1 registers new services that become possible due to the Agent R in the APM which updates its tables accordingly.

3.9 Agent Leaving a Coalition

Similar to the previous section on agent joining a coalition, when an agent tends to leave a coalition due to un-availability, resource problems etc, a set of leaving rules are defined based on the subscription rules of the coalition group. Rules like group losses, individual losses, renegotiation and higher payoff for the stay of the agent, will have to be taken into account for the agent leaving the group. See Fig 6.
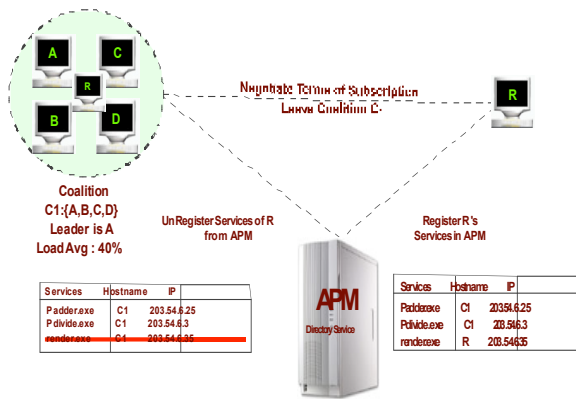
Figure 6: An Agent leaving a Coalition

Agent R is leaving the group C1 after negotiations. A un-register message service is called upon if specific services of R are affecting the Coalition C1 as a whole and the APM is updated. Similarly the autonomous node or agent registers its services back to the APM which does an update to its directory.

4   Templates and QOS Factors

As discussed before, jobs are based on a set of quality of service factors such as a specific storage requirement, load of a node, etc. A template is a predetermined QOS factoring that has calculated the requirements of the job in order to match the jobs existing in the job queue. A template is formulated based on the QOS factors that are represented by say Q1, Q2, Q3… Qn.

4.1   Template formation

Let us now assume that a job J1 has a specific service requirement say padder.exe with the following QOS factors. Load <=40, storage >= 10 and CPU % >=50 for the job J1. Based on job history, a template is formulated that becomes a standard template for all the peers for referencing an incoming job. Let all the template for a common service be considered as T1, T2…..Tn. In order to define a template, we need to have some form of job history where all jobs having similar QOS properties are put into a template based job queue.

4.2   Matching templates with jobs

Each time a new job arrives along with the details of its template that denotes the QOS requirements for that job. Based on the preformed templates a distance-matching algorithm was developed as discussed in section 3.1.5. Once the template complete or partial match is obtained, the job is offloaded to the closest matching template based coalition and negotiations take place among the agents.

4.3   Job history

Let us take the same case of job J1 having Load <=40, storage >= 10 and CPU % >=50 as its QOS factors. Each time a new job arrives; the jobs history is tracked to see how many jobs have similar QOS constraints. Once this is determined, templates are formulated and shared as a standard job processing reference among the agents.

5   Calculating Job Processing Requirements and Template Matching

Let us now see how templates can be computing using a set of QOS factors. Let us take the same example and compute the requirements of the job say J1. Let us take the same case of job J1 having Load <=40, storage >= 10 and CPU % >=50 as its QOS factors. Let us take the QOS factors to be Q1, Q2, Q3 and the distance needs to be computed for the job J1 using a template T1. It is assumed here that T1 satisfies the QOS factors Q1..Q3.

The distance is calculated by:

$$Distance = \sum (T_i - J_i) + w(P_i, Q_i)$$

Distance calculation formula:

	P1	P2	P3	Q1	Q2	Q3
Template (Ti)	T	T	T	50	30	20
Job Req. (Ji)	T	F	T	<=40	>=35	>=10

Table 1: Sample data for computing the distance between the Job and Template.

Here P1 and P2 are the predicate properties of the Job J1 and Q1 Q2 and Q3 are the real values R1, R2…Rn of the job. So let us take that the storage requirements and load requirements of the job J1 are satisfied that is denoted by T (True) and CPU % is not satisfied which is denoted by F (False). The predicate value for Template T1 is Load =50, storage = 20 and CPU % = 45 and the real value requirements of the job J1 is denoted by Load <=40, storage >= 10 and CPU % >=50. Let W1, W2…Wn be the weights associated with the job J1 and O is computed using the real values of job J1. If the predicate value matches, it is denoted by a 0 else by a 1. By applying these values to the Distance formula we can compute the distance for calculating the best coalition or agent available for the job J1. The summation and end distance value is computing using Ti and Ji the real values and Pi and Qi, which is again the predicate values, required by the job J1.

The predicate values are denoted by a true or false condition based on the matching of the template Qos factors to that of the Job J1. Therefore if the outcome is a true and true it's represented as a 0 and computed, else no computation takes place. Similarly a simple subtraction of the real value gives the minimum distance between that of the template and job properties. Let us now take the example given in the table above and try to find the distance manually for J1 and the template T1.

Let us compute w to be the weight, which is equal to 1. Now for Q1 factor we compute the values in the distance formula obtaining a distance of 10. Similarly for Q2 we get −5 and Q3 we get −9. The sum of Q1, Q2 and Q3 gives us the distance between the job J1 properties and the predetermined template T1, which is obtained as −4. Similarly the number of templates is created based on the history of the Jobs and a definitive distance calculation

helps to confirm which of the coalitions are eligible for job processing. By calculating the distance between each template and the Qos factors of the job we can find out which are the best set of optimal coalitions or peers that are available to R for offloading job J1.

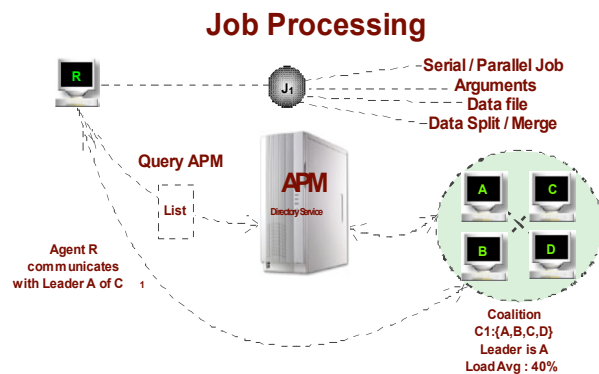## 5.1 Agent based job processing and negotiations

Figure 7: Remote Job Processing in Coalition C1 Based on Job Properties

As shown in Fig 7 R is a Remote peer that has a need to process job J1. As R is heavily loaded it tends to search for new peers having similar job processing capabilities like R for Job processing. Here R finds a Coalition represented by a leader [refer section 4.2] A that has the capabilities to do the job J1. Negotiations take place between the agents R and A and accordingly remote job processing takes place. The load balancing scheme $A^{3p}viload$ [10] is utilized for local as well as remote load balancing in both R as well as leader A. The coalition could have services that do parallel or serial processing of data files and its respective programs.

5.2 Serial Vs Parallel Jobs

Coalitions could be formed based on the properties of the job namely serial or parallel processing of the job at hand. Based on the property of the job a coalition could increase its utility value. Each coalition could have a number of gains or losses due to the addition or removal of self-interested agents in its coalition. Serial job pose as a big problem to coalitions as the fairness of the payoff become negligible to the agent or peer doing a singular or serial task.

That is if the job J1 as shown in Fig 7 represent a serial task then one of the peer in the coalition formed tends to get to do the job based on the load balancing scheme AviLoad. This leads to the payoff split between A, B, C and D instead of a single peer getting a full payout for the job at hand. One solution to this could be the usage of fairness in payoff using the shapely algorithm.

5.3 Electing a leader for the representation of the Coalition

Every coalition formed need to elect a mutual leader based on the qualities of the system such as speed of processing, reliability, history etc. The leader represents the Coalition formed and the responsibility of the leader is to

achieve negotiation with remote peer and agents for obtaining new jobs that lead to the increase of the overall utility value of the coalition as a whole and thus ensuring an increase in payoff to individual agents. The primary advantage of the formation of the leader representing the group is to minimize the messages passed in between remote peers available in the immediate domain space.

6 Coalition in $A^{3p}viGrid$ System Architecture

$A^{3p}viGrid$ Components: The following are all the components that play a vital role in the function of the $A^{3p}viGrid$ system.

(a) Agent Based Peer Manager [APM]: The Agent based Peer manager is an intelligent agent that handles negotiating and registering if services based on directory services. Although it poses a threat as a centralized scheme, it's primarily used for discovery and communications between agents and their respective peers based on a lightweight directory services model such as LDAP [6]. The primary role of the APM is to register services of agents based on commonality and self-interests factors and also help in discovery and formation of Coalitions in agents based on the commonality of their services rendered. The APM also tends to act as a broker or middleware for all agents associated with it. An economic brokering system could be incorporated based usage of the APM which renders as service to self-interested agents and agent based system in the locality.

(b) Load Average of a Coalition: The load average of a coalition is calculated by taking individual like nodes based on their services and resources offered and average load is calculated. Load can be sub-categorized as local load where local jobs are being run and global load where remote jobs are executed based on the services offered as a singular autonomous system or in coalition. The load of a system is calculated based on the CPU cycles in percentage say 100 % divided by the number of processes running in the system.

$$Loadavg = CPUtime / Nproc$$

Where Loadavg is the load average calculated based on the CPUtime divided by Nproc which is the number of processes running on the system to obtain the average load value.

(c) Service Table: The Service table is a location based commonality of services table that stores common services and its respected agents or coalition represented by agents in the form of a table using a light weight directory service such as LDAP [6]. The importance of the state information stored is that it is used only for negotiation and discovery of peers and their respective services offered to the end user or user agents.

(d) Load Balancing: A variety of Load balancing and scheduling schemes are available for distributed systems. The most common method used both in distributed systems as well as web server technologies that are used for serving WebPages are based on the Round robin scheduling scheme. Scheduling and scheduling policies depends

on the usage of a distributed system. Round Robin scheduling is probably the simplest of scheduling algorithms used where in the load is distributed in a round about fashion. Jobs / Processes are migrated to the next available node that is put in the job queue thus ensuring a reliable job processing. A study conducted by Eager et al. was based on the performance of three main algorithms and their complexities. The A^{3p}viLoad (Shankar 2005) load balancing scheme segregates nodes based on common resource thus leading to a very promising scalable load balancing system.
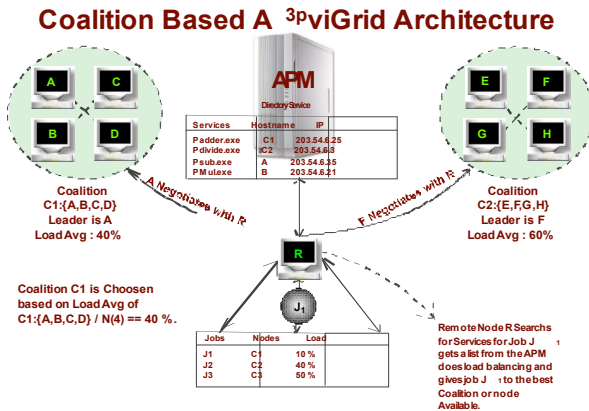

Figure 8: A Blue print of the A^{3p}viGrid Architectural Scheme

Nodes are aggregated and listed based on the commonality of resources such as Operating systems, architectures such as intel or sparc, etc that leads to a uniform scalable solution as compared and discussed in a previous paper. The Load sharing scheme is used here to effectively Load balance local nodes in Clusters. The scheme also provides a De-centralized and minimal state information based process migration that is transparent over the Cluster. The message passing and over all process communication is altogether minimized leading to a very responsive Clustering / GRID internetworking system. The machine status, is define as,

Machine status = CPU utilization in percentage %...5

A list of statuses for every machine is kept in the form of state information in a table called the My table. Now each machine creates a table for itself that rates the machines according to their machine status and includes the IP addresses of the machines that rank lower than itself on the status table in ascending order of the load (CPU utilization) on each machine at that particular point in time. Thus the load balancing scheme offers a very scalable and efficient load balancing scheme.

(e) Job Table: The job table is a local job queue that is created based on the commonality of the jobs which are then offloaded to better systems for remote job processing. A job is differentiated only by its previous history and the resources needed for the job to run. New jobs are offloaded based on the load threshold of the system.

$$Loadproc < Tload$$

Here the Load of the process is denoted by Loadproc that is less than that of the threshold value Tload of the system and hence it is possible for the system to do local processing.

$$Loadproc > Tload$$

Load of the process is increasingly higher than the local threshold value of the system and hence has to be off-loaded. Thus the job is put into its respective job queue for offloading and remote job processing

(f) Load balancing in Local Coalition: The group leader tends to organize local job processing based on the load of the local coalition. In case of serial jobs a singular node or agent is to be selected and the job needs to be aggregated to that node in the coalition. The load balancing scheme discussed in (4) is utilized for load balancing in local nodes of the Coalition.

(g) Load Balancing in Originator: The originator is where a job evolves. The originator tends to get a list of peers from the APM as requested based on the commonality of services available in remote peer or Coalitions. The Originator agent then takes the decision of offloading its job(s) based on the Loadavg and negotiations with the remote Coalitions or autonomous Agents.

(h) Economic Brokering System: An economic brokering system can be achieved here by incorporating the concept of brokerage in the APM, which acts as the negotiator and service discovery of the peers, and also a mutual payment scheme based on the negotiation between agents in the various remote peers having common services. The utility value here would be money and the agents in the coalition negotiate with the leader based on the job to have fairness in payoffs. The fairness value could be achieved by using methods such as shapely value [7] based on fairness in payoff.

7   Functioning of the A^{3p}viGrid System

As shown in Fig 8 the A^{3p}viGrid system runs on a commonality of services and negotiations based on mutual interests. Coalitions are formed and registered in the APM. In Fig 8 there are two coalition formations namely C1 (A, B, C, D) and C2 (E, F, G, H) based on commonality of services and self-interests of the agents. Let us assume that a service called parallel addition or padder.exe is available in nodes A, B, C, D, E, F, G and H.

Each node is functioning as autonomous peers where a bunch of Agents do routine job processing. Based on the commonality of services which here would be the padder.exe service and self interested agents a Coalition is formed based on the locality of the peers in a local domain space having very minimal network threshold or turnaround time. That is the systems are located in a local domain space where the nodes can provide Quality of services (QOS) to every peer available in its domain. The primary goal of the A^{3p}viGrid system is to minimize

overall communication between nodes; minimize state information updates and the removal of a resource discovery model based on an Agent based Peer manager directory services that tend to provide service and resources discovery between remote peer using the authentication of the peers. Here the coalitions C1 and C2 are registered in the APM directory service with a commonality of service namely padder.exe.

A remote node R has a padder.exe service requirement for a job called J1 based on the load in R. The remote node R then authenticates and searches for peers having the padder.exe services for offloading of the process J1. A list of Coalition and peers are then downloaded into R. The node then starts to negotiate with Coalitions C1 and C2 and mutual agreement is negotiated based on the payoff or utility value of the job to be processed. Before negotiations with the coalitions C1 and C2, R tends to do local load balancing based on the load averages of the Coalitions and also the turnaround time or closeness of the coalition domain is calculated and a decision is taken for the offloading of J1 based on the turnaround time and the payoff value offered during negotiations. Once the agents agree upon a template system, jobs are compared with the templates available to do job processing in the closest matching template.

Once a satisfactory match is obtained, based on the distance calculated the coalition sends an ACK and job processing takes place followed by the results sent back to remote node R the originator of the job. Thus the need for a resource discovery model is minimized as it is noted that services offered by remote nodes becomes a proof of concept for the satisfying resource requirements of the services offered by the remote peer. The Coalition formation in agents tends to minimize the number of messages passed between the nodes and self interested agents form coalitions based on the payoff value.

8   Limitations

Any Architecture that is designed poses some form of limitations which is no exception pertaining to the A^{3p}viGrid system. The following are some limitations posed by the system:

• Dynamic updating of State information

• Fault tolerance in Centralized Directory service APM

• Agent languages

• Dynamic Coalition formation based on the incoming Job properties

• Fault tolerance in Agent based Coalitions formations

• Trust issues and payoffs

• Agent gains and losses due joining and leaving of Coalitions

9   Conclusion and Future Directions

Applying coalition methodologies to multi agent based high performance grid computing systems has lead to a new perspective to the usage of intelligent agents in grid computing systems. The A^{3p}viGrid system provides solutions to resource discovery models made void by the usage of directory services such as the APM and the usage of effective coalition schemas used in agents with a very efficient and scalable load balancing scheme namely A^{3p}viLoad that provisions the use of local load balancing in individual peers as well as local coalitions formed by intelligent agents with a commonality of goals based on a service oriented schematic.

10   References

D.Eager, E.Lazowska, J.Zahorjan (1986): "Adaptive Load Sharing in Homogeneous Distributed Systems", IEEE Transactions On Software Engineering, vol. 12, pages 662 –675.

S. Franklin, A. Graesser (1996): "Is it an agent, or just a program?: A taxonomy for autonomous agents", in Proc. 3rd International Workshop on Agent Theories, Architectures, and Languages, Springer-Verlag LNCS 1193.

T.Finin, Y.Labrou, J.Mayfield (1994): "KQML as Agent Communication Languages", Proc. 3rd Int. Conf. on Information and Knowledge Management, pp. 456-463.

M.Mitzenmacher (2000): "How Useful is Old Information?", IEEE Transactions on Parallel and Distributed Systems, Vol. 11, No. 1, pg. 6-20.

"Open LDAP: a light weight open source directory protocol project", www.openldap.org, 2005

A. Shankar (2005):"Applying Coalition Concepts to Service Oriented Multi-Agent Load Balancing Systems – A^{3p}viLoad", to appear in Proc. PDPTA.

A. Shankar, C. Sombattheera, A. Krishna, A. Ghose, P. Ogunbona (2005): "Dynamic coalition in agent aware adhoc virtual p2p interconnect grid computing system - A^{3p}viGrid.", in Proc. 7th International Conference on Enterprise Information Systems - ICEI, 170-175

L. Somdeb (1993): "Generalized Shapely Value for Games with a Coalition Structure", IIMA Working Papers, Indian Institute of Management Ahmedabad, Research and Publication Department.

N.J.E.Winjngaards, B.J.Overeinder, M.Vansteen (2002): "Supporting Internet scale Multi Agent Systems", Data Knowledge Engineering, Vol. 41, No. 2-3, pp. 229-245.