Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 2, Issue. 2, February 2013, pg.64 – 73

RESEARCH ARTICLE

CYCLOPHENIX: Inexpensive Membership Management Protocol Supporting Resilient Low-Diameter P2P Topologies

[1]*Pallavi Totlani*
Assistant Professor, Department of Computer Science & Engineering, Sarvottam Institute of Technology & Management, Greater Noida, India
pallavi.totlani@gmail.com

[2]*Balaji Kottana*
Scientific Officer, National Informatics Centre, New Delhi, India
bala1205@gmail.com

[3]*Prabhakar Sharma*
Software Engineer, Open Solutions Software Service Pvt. Ltd., Gurgaon, India
prabhakar.sharma87@gmail.com

Abstract— Peer-to-Peer networks are usually unstructured network. An unstructured network is simply a network which does not have a specific layout. Anyone who joins it becomes an integral part of the network, and changes its topology. Anyone can walk off at any time; the network will eventually change, but without any fall out (ideally). Ideally, such networks should have a low diameter; offer a high degree of resilience against the dynamisms of nodes, nodes failure and malicious actions with a purpose of distracting the network. Whereas, a structured network are predestined networks, having predefined connectivity among the nodes, in order to put forward an assurance between the source node and the destination node. Neither the structured nor the unstructured networks can simultaneously give both good performance and resilience in the network. A gossip based membership management protocol is described here. Our protocol constructs a graph that have low diameter, low clustering, highly symmetric node degrees, that are highly resilient to massive node failures and robust to network dynamics such as joining/leaving of nodes, node failures and large scale network attacks.

Indexed Terms: - Peer-to-Peer network, Membership Management, Gossiping Protocol, Unstructured Network, Random Graphs, Resilient Networks.

I. INTRODUCTION

The popularity of the Internet and the accessibility of powerful computers and speedy networks as inexpensive service mechanism are changing the way we make use of computers today. These technical prospects have directed to the opportunity of using geographically distributed and multi-owner resources to work out with large-scale problems in science, engineering, and commerce. This has become possible with Grid Computing. Grid Computing basically refers to combining of many computers from various administrative domains, to achieve a common task and may just fade away as swiftly.

A vital building block for grid computing systems is job scheduling. Scheduling can be defined as the process of selecting N machines in the infrastructure such that they will be concurrently accessible to carry out a job in close proximity to future for a given period of time T. In a Grid computing environment, depending on a single

centralized scheduler is impossible due to the problems of scalability, single point failure and political reasons. As an alternative, the majority of Grid platforms rely on meta-scheduling, where each group is allotted with its own local scheduler, and global job scheduling is figured out by handing over each job to one cautiously chosen group capable of executing it [1]. Although, in an environment made up of huge numbers of small groups or individual machines, scheduling jobs onto resources across several organizations became crucial as no single group may have adequate resources or the readiness to execute a large job. This has directed a number of authors to put forward peer-to-peer solutions to the Grid scheduling problem [2, 3, 4].

Peer-to-Peer networks are usually unstructured network. An unstructured network is simply a network which does not have a specific layout. Anyone who joins it becomes an integral part of the network, and changes its topology. Anyone can walk off at any time; the network will eventually change, but without any fall out (ideally). Ideally, such networks should have a low diameter; offer a high degree of resilience against the dynamisms of nodes, nodes failure and malicious actions with a purpose of distracting the network.

Over the past several years, a rapid expansion in peer-to-peer application has been observed. Many gossip based membership management protocol that met the above specified requirements were developed, but they were not resilient against the malicious behaviour of the intruders. CYCLON- an inexpensive gossip based membership management protocol that constructs a graph with low diameter, low clustering, highly symmetric nodes degree, and resilient to massive node failure, was developed. This protocol is not resilient against various types of attacks, discussed further in the text. To make it more resilient against the attacks we would like to make some amendments in the procedure of adding a node into the network.

We propose CYCLOPHENIX algorithm that constructs a P2P low diameter resilient topologies. It supports low diameter operations by creating a topology of nodes whose degree distribution follows a power law. It is robust to network dynamics such as joining/leaving, node failure and large scale network attacks.

II. RELATED WORK

Grid Computing is up-and-coming as a leading technology for High-Performance Computing. Even though there has been sizeable recent improvement on building the software infrastructure to facilitate transparent use of worldwide distributed computers on the grid, the matter of job scheduling on grids has not acknowledged much concentration. A study of the resource handling pattern at a number of supercomputer centers shows an interesting “sine wave” prototype. During evenings, the resource demand reaches and sometimes surpasses the maximum competence of the system, while usage plunges to a least amount in the very early hours of the morning. By amalgamating centers in a computational grid, in adding together to providing more computation power than any single site can offer, the time reliant and bursty nature of resource desires can be better averaged by distribute the requests to various different centres. Effectual scheduling is imperative in optimizing resource handling, but the mission of scheduling is more intricate in meta-computing surroundings since many clusters with dissimilar local scheduling policies are concerned.

Here, we reflect on the issues of meta-scheduling of jobs amongst a number of geographically dispersed centers. We first analyse formerly proposed meta-scheduling plots, both centralized and distributed, and weigh up them using trace-based simulation. The centralized schemes (which however are not scalable) generate schedules with lesser slowdown and turn-around time than a more scalable distributed plots. New distributed scheme was proposed that challenges to improve on previously projected schemes. The primary idea we weigh up is that of redundantly distributing each task to numerous sites as an alternative of only transferring it to the most lightly loaded locations. The new schemes provide significant reductions in average job slowdown and turn-around time. We also evaluate the impact of user inaccuracy in estimation of job runtimes, and the effect of remote data transfer overhead on the efficacy of the proposed schemes.

III. VARIOUS APPROACHES

The popularity of the Internet and the accessibility of powerful computers and speedy networks as inexpensive service mechanism are changing the way we make use of computers today. These technical prospects have directed to the opportunity of using geographically distributed and multi-owner resources to work out with large-scale problems in science, engineering, and commerce. This has become possible with Grid Computing. Grid Computing basically refers to combining of many computers from various administrative domains, to achieve a common task and may just fade away as swiftly.

A vital building block for grid computing systems is job scheduling. Scheduling can be defined as the process of selecting N machines in the infrastructure such that they will be concurrently accessible to carry out a job in close proximity to future for a given period of time T. In a Grid computing environment, depending on a single centralized scheduler is impossible due to the problems of scalability, single point failure and political reasons. As an alternative, the majority of Grid platforms rely on meta-scheduling, where each group is allotted with its own local scheduler, and global job scheduling is figured out by handing over each job to one cautiously chosen

group capable of executing it [1]. Although, in an environment made up of huge numbers of small groups or individual machines, scheduling jobs onto resources across several organizations became crucial as no single group may have adequate resources or the readiness to execute a large job. This has directed a number of authors to put forward peer-to-peer solutions to the Grid scheduling problem.

Peer-to-Peer networks are usually unstructured network. An unstructured network is simply a network which does not have a specific layout. Anyone who joins it becomes an integral part of the network, and changes its topology. Anyone can walk off at any time; the network will eventually change, but without any fall out (ideally). Ideally, such networks should have a low diameter; offer a high degree of resilience against the dynamisms of nodes, nodes failure and malicious actions with a purpose of distracting the network.

Over the past several years, a rapid expansion in peer-to-peer application has been observed. Many gossip based membership management protocol that met the above specified requirements were developed, but they were not resilient against the malicious behaviour of the intruders. CYCLON- an inexpensive gossip based membership management protocol that constructs a graph with low diameter, low clustering, highly symmetric nodes degree, and resilient to massive node failure, was developed. This protocol is not resilient against various types of attacks, discussed further in the text. To make it more resilient against the attacks we would like to make some amendments in the procedure of adding a node into the network.

A. *Basic Shuffling Protocol*

The basic shuffling algorithm, introduced in, is a straightforward peer-to-peer communication model. It shapes an overlay and maintains connection by means of an epidemic algorithm. The protocol is exceptionally simple: each peer is familiar with a small, continuously changing set of other peers, called as its *neighbours*, and occasionally contacts a random one of them to swap over some of their neighbours. More properly, each peer retains a neighbour list in a small, fixed-sized cache of *c* entry (with typical value 20, 50, or 100). A cache entry contains the network address (i.e., IP address and port) of another peer in the overlay. Each peer *P* repeatedly initiates a neighbour exchange operation, known as *shuffle*, by executing the following six steps:

1. Select a random subset of ` neighbours ($1 \cdot ` \cdot c$) from *P*'s own cache, and a random peer, *Q*, within this subset, where ` is a system parameter, called *shuffle length*.
2. Replace *Q*'s address with *P*'s address.
3. Send the updated subset to *Q*.
4. Receive from *Q* a subset of no more than ` of *Q*'s neighbours.
5. Discard entries pointing to *P*, and entries that are already in *P*'s cache.
6. Update *P*'s cache to include *all* remaining entries, by *firstly* using empty cache slots (if any), and *secondly* replacing entries among the ones originally sent to *Q*.

Upon reception of a shuffling request, peer *Q* randomly selects a subset of its own neighbours, of size no more than `, sends it to the initiating node, and executes steps 5 and 6 to update its own cache accordingly.

Figure1 presents a schematic example of the basic shuffling operation.

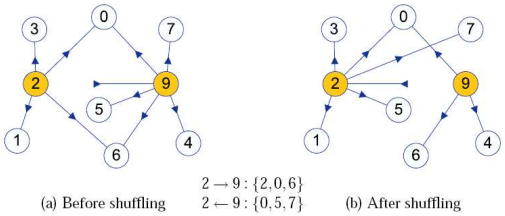


Figure1. An example of shuffling between nodes 2 and 9 and Links between 2 and 9 reverse directions. [1]

B. *Enhanced Shuffling*

CYCLON employs an enhanced edition of shuffling. It perks up the quality of the overlay in terms of arbitrariness. Enhanced shuffling pursues the identical model as basic shuffling does. The primary dissimilarity is that nodes do not *arbitrarily* select which neighbour to shuffle caches with. As a substitute, they select the neighbour whose information was the initial one to have been introduced in the network. There were 2 basic inspirations following enhanced shuffling, initially, to restrict the time a pointer can be wandered around until it is chosen by some node for shuffling. Secondly, to compel a predictable lifetime on each pointer, in order to manage the number of presented pointers to a provided node at any time.

The enhanced shuffling procedure is carried out by letting the initiating node *P* execute the following steps as described in [1]:

1. Increment by one the age of all neighbours.
2. Select neighbour *Q* with the highest ages among all neighbours and *l*-1 other random neighbours.

3. Swap *Q*'s entry with a new entry of age 0 and with *P*'s address.
4. Send the updated subset to node *Q*.
5. Obtain from *Q* a subset of no more that *l* of its own entries.
6. Discard entries pointing at *P* and entries already contained in *P*'s cache.
7. Update P's cache to include all remaining entries, by firstly using empty cache slots (if any), and secondly replacing entries among the ones sent to Q.

The receiver node Q act in response by conveying back a arbitrary subset of at most *l* of its neighbours, and fill in its own cache to lodge all received entries. It does not augment its age, because any entry's age is not incremented till it possesses turn comes to begin a shuffle.

C. ***Phenix***

Over the precedent numerous years, it has been observed that the speedy escalation of peer-to-peer applications and the appearance of overlay infrastructure for Internet, on the other hand, countless challenges linger as this fresh field grows-up. A resilient peer-to-peer networks and their competent performance in expressions of faster response time and low-diameter operations for client queries was projected in [4]. Low-diameter networks are frequently attractive because they provide a low average distance among nodes, often in the order of *O* (*logN*). The two groups of peer-to-peer networks either provide better flexibility to node dynamics such as joining/leaving, node malfunction or service attacks, as in the case of unstructured networks, or, they provide improved performance as in the case of structured networks. Because of the intrinsic trade-offs in the blueprints of these diverse classes of peer-to- peer networks, it is complicated to concurrently provide improved performance and flexibility without having to think again about some of the elementary blueprint selections made to widen these network systems. An alternative strategy was considered and a new peer-to-peer algorithm was projected that deliver both performance and flexibility. The projected algorithm construct a low-diameter flexible peer-to-peer network offering clients with a high likelihood of attaining a huge number of nodes in the system even under conditions such as node elimination, node malfunction, and malevolent system attacks. The algorithm does not compel arrangement on the network; rather, the established graph of network associations has the objective of building some order from the entire arbitrariness found in flexible unstructured networks. An unstructured peer-to-peer network, such as Gnutella, provides no assurance on the diameter as nodes be linked in an arbitrary style, resulting in whatever thing other than a proficient topology [4]. These unstructured schemes are often disapproved for their shortage of scalability, which can lead to separation in the network resultant in small islands of interconnected nodes that cannot get in touch with each other. However, these similar random associations provide the network an elevated degree of resiliency where the action of the resultant network as a complete is bearable to node elimination and malfunction. On the other hand, these systems force a comparatively inflexible structure on the overlay network, which is repeatedly the cause of ruined performance throughout node removals, necessitating non-trivial node upholding. This results in convinced vulnerabilities (e.g., feeble points) that assailants can aim and take advantage of. Due to the intend of DHTs, these planned topologies are also restricted in offering applications with the elasticity of standard keyword hunts because DHTs depend expansively on hashing the keys linked with objects. *Phenix* [4], a scale-free algorithm that builds low-diameter P2P topologies contributing fast reaction times to clients was projected. Another significant characteristic of Phenix is its built-in stoutness and flexibility to network dynamics, such as, equipped nodes adding to and leaving the overlays, node malfunctions, and significantly, malevolent large-scale assaults on overlay nodes. The most important design objective of Phenix can be recapitulated as follows:

- ***to construct*** low-diameter graphs that result in rapid response times for clients, where most nodes in the overlay network are inside a small number of jumps from each other;
- ***to maintain*** low-diameter topologies under regular operational circumstances where nodes at times add to and abscond the network, and under malevolent conditions where nodes are methodically attacked and removed from the network;
- ***to implement*** support for low-diameter topologies in an entirely dispersed way devoid of the need of any vital power that may become a solitary point of malfunction, which would unavoidably restrict the robustness and flexibility of peer-to-peer networks;
- ***to support*** connectivity among peer nodes in a broad and non-application precise way so a wide-variety of applications can exploit the network overlay infrastructure.

An imperative characteristic of Phenix is that it builds topologies based on power-law degree distributions with a built-in method that can accomplish a high degree of flexibility for the complete network. Even in the event of concentrated and besieged attacks, nodes in a Phenix network prolong to commune with a low diameter where they competently and punctually reorganize their connectivity with slight overhead and disturbance to the action of the network as an intact. Phenix corresponds to one of the former algorithms that construct resilient

low diameter peer-to-peer topologies purposely besieged toward, and consequent from, popular unstructured P2P network architectures [7].

IV. PROPOSED IDEA

A. The CYCLON:

CYCLON make use of an enhanced version of shuffling. It improves the quality of the overlay in terms of randomness. Enhanced shuffling follows the same model as basic shuffling. The key difference is that nodes do not *randomly* choose which neighbour to shuffle caches with. As an alternative, they choose the neighbour whose information was the earliest one to have been injected in the network. There were 2 basic motivations behind enhanced shuffling, firstly, to limit the time a pointer can be passed around until it is selected by some node for shuffling. Secondly, to impose an expected lifetime on each pointer, in order to control the number of existing pointers to a given node at any time.

The enhanced shuffling procedure is carried out by letting the initiating node *P* execute the following steps:

1. Increment by one the age of all neighbours.
2. Select neighbour *Q* with the highest ages among all neighbours and *l*-1 other random neighbours.
3. Swap Q's entry with a new entry of age 0 and with P's address.
4. Send the updated subset to node Q.
5. Obtain from Q a subset of no more that l of its own entries.
6. Discard entries pointing at P and entries already contained in P's cache.
7. Update P's cache to include all remaining entries, by firstly using empty cache slots (if any), and secondly replacing entries among the ones sent to Q.

The recipient node Q responds by sending back a random subset of at most *l* of its neighbours, and updates its own cache to accommodate all received entries. It does not increase its age, because any entry's age is not incremented till its own turn comes to initiate a shuffle.

Figure 2 shown below depicts the schematic example of CYCLON.

Here cache size is 4, *l* is 2.

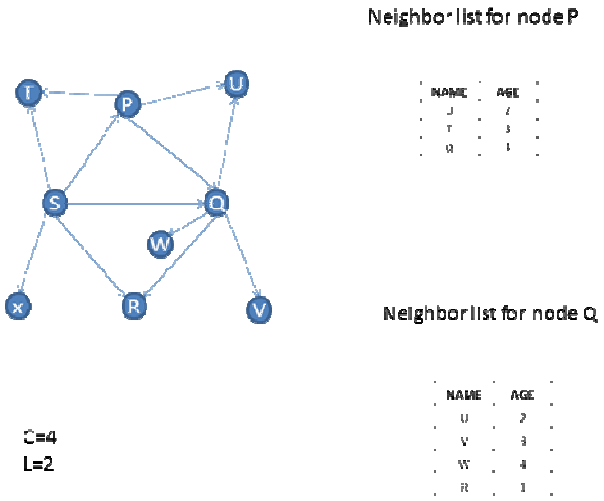


Figure2. A Network with cache size=4 and List size=2. Neighbour list of nodes P and Q.

Node P initiates shuffling. Then P increments age of all the entries in its neighbour list. The incremented list of P is shown below.

NAME	AGE
U	3
T	4
Q	5

Figure3. Incremented neighbour list of node P.

After that P finds that age of Q is maximum in its list, P selects Q for shuffling, and then it replaces Q's entry with its own entry with an age 0. Following list is passed to Q.

NAME	AGE
U	3
P	0

Figure4. List passed on to the node with whom shuffling is to be done.

On receipt of the above list node Q selects *l* random entries from its neighbours list and pass it to node P.

NAME	AGE
U	2
W	4

Figure5. List selected at another end by a node to transmitting to the initiating node.

On receiving the list node P updates its neighbour list by discarding the entries already present in P, and using empty slots in the list first and then replacing the entries sent to Q. After updating the list of P would looks like as shown below.

NAME	AGE
U	2
T	3
W	4

Figure6. Updated list at the initiating node.

Q also updates its neighbour list with the list received from P. In our case, Q's list after updating is as follows:

NAME	AGE
P	0
V	3
W	4
R	1

Figure7. Updated list of the node at another end.

After both the nodes have updated their respective list, the topology of the network changes.

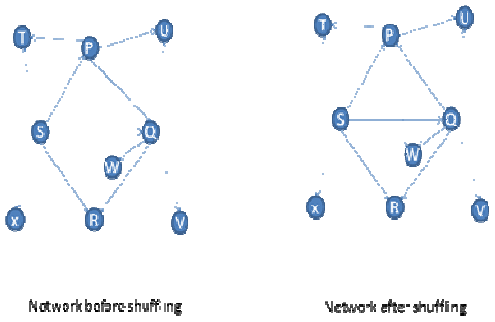


Figure8. Network before and after shuffling.

A. I. Adding a node to CYLON

To join, a node simply needs to know any single node, which is already in the overlay; it is called as the Introducer.

- If node P wants to join the network.
  - P's introducer initiates c (cache size) random walks, setting their TTL to a small value close to the expected average path length, such as 4 or 5.
  - A node Q a random walk ends, replaces one of its cache entries with a new entry of age 0 and address of P.
  - Q then forwards the replaced cache entry to P, who in turn, has to include the entry in one of its cache's empty slots.(this is equivalent to P initiating a shuffle of length 1.)
  - The join operation finishes when all the c random walks have ended and the corresponding neighbour exchanges have been accomplished.
- P's cache is filled up with c randomly chosen neighbours, which renders the values of P's average path length to reach any other node as well as its clustering coefficients, indistinguishable from the respective value of other, older nodes.
- Since there are c random nodes in the overlay that know P, P's in degree is equal to the cache size.
- No other nodes in-degree has been modified.

A. II. Removing Nodes

- Motive- once a node disconnects, other nodes should detect it and remove any pointer to it in timely manner.
- Disconnected nodes to consist a sort of cache pollution, as they take up slots that could be otherwise holding valid, useful links.
- Cyclon uses transparent dead-link detection mechanism, based on default shuffling message exchange.
- When a node tries to initiate a shuffle with a neighbour and get no reply within a predefined timeout, it simply assumes that neighbour to be disconnected and removes the corresponding entry from its cache.

A. III. Hitch in CYCLON

One hitch of this protocol can be that during the addition process security can be breached easily or in other words the attacker nodes can be appended to the network easily, as if any node knows only one node which is connected to the network than it can easily get connected to the network.

While the nodes are being added to the network various types of attacks are possible. We assume that attackers are powerful enough to drop down the nodes in the network. The network can be attacked in 3 ways. Firstly, a user that acquires host cache information like a genuine node might. The attacker associates with these acquired nodes with a message, getting the respective lists of their neighbours, and make his own list, as a result. Although, once the attacker has this information it will then attack the nodes present in this list more than once, removing them from the network. Secondly, in this attack, an attacker requires to add a number of nodes to the network that only point to each other, thus, raising the possibility that they will come forward as preferred nodes in the overlay network. Thirdly, this attack consists of adding a number of nodes to the network that would act like ordinary nodes do.

These last two types of attacks try to create variance in the network by introducing 'wrong' nodes that stay linked for a prolonged period of time. Such an establishment would make sure that other 'correct' nodes come to rely on these wrong malicious nodes due to the length of time that the wrong nodes are existing in the network. Under such attack circumstances, these wrong nodes abruptly disconnect from the overlay network all at the same time with the objective of disconnecting and partitioning the network into small islands of nodes. Some of the malicious nodes may use both second and third types of attacks.

A little more complex procedure can be implemented in the addition process of nodes to defend the network from the above mentioned attacks [4].

Procedure for adding a node

- Suppose a node i wants to join the network.
- It obtains the list of addresses using rendezvous mechanism by either contacting a host cache server or consulting its own cache from previous session in a fashion similar to an initial connection.
- The joining nodes divide the receiving list into 2 subsets viz: $G_{h,i} = [G_{random,\ i}\ ,\ G_{friends,i}\ ]$
- i initiates a request M_0 as "ping message" to the nodes in the list $G_{friends,i}$

  M_0 = (source=i, type=ping, TTL=1, hops=0)

- Each recipient nodes constructs a "pong message" as a reply containing the list of its own neighbours, increments the hop counter, decrements the TTL, and forwards a new ping message M_1 to its own neighbours

  M_1 = (source=i, type= ping, TTL=0, hops=1)
- Each j node receiving such message (M_1) will send no pong message in reply, but instead add the node i to a special list called Γ_j for a period of time denoted as τ.
- Following this procedure, node i obtains a new list of all neighbours of nodes contained in $G_{friends,i}$ and constructs a new list denoted by $G_{candidates,i}$.
- Then i sorts the new set nodes using the frequency appearance in descending order, and uses the topmost nodes to create a new set $G_{preferred,i}$.

  Where $G_{preferred,i} \subseteq G_{candidates,i}$
- Thus resulting set of neighbours to which i creates connections is given by

  $G_i = [G_{random,i}, G_{preferred,i}]$
- Node i opens a "servent"(server client), connection to a node m (m is a node from $G_{preferred,I}$) where the word servent is a term denoting a peer-to-peer node, which is typically a server and the client at the same time as it accepts connections as well as initiates them. The node m checks whether i is in its Γ_m list, and if it is there then increment an internal counter C_m and compare it against a constant r.
- If $C_m >= r$, then $C_m = C_m - r$, a connection is created with node i, which is called as Backward Connection, and a set of neighbours added as backward edges is updated as

  $G_{backward,m} = G_{backward,m} \cup \{i\}$
- The backward connection creates an undirected edge between the two nodes i and m (i< >m) from the initial direct edge i->m.
- In addition r ensures that a node does not make more connections than $d_{in,m}$ /r, where $d_{in,m}$ is the in degree of node m.
- When a node i receives a backward connection from node m it will consider its choice of node m as a good one, and accordingly update its neighbours list.

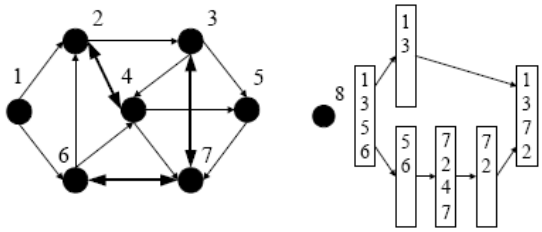
  $G_{preferred,i} = G_{preferred,m} - \{m\}$

  $G_{highly\_preferred,i} = G_{highly\_referred,m} + \{m\}$

  Therefore the final list of neighbors of node i is given by:

  $G_i=[G_{random,i}, G_{preferred,i}, G_{highly\_preferred,i}, G_{backward,i}]$

Figure 9 is the schematic example of the above mentioned procedure.



(a) The Existing Network (b) Node 8 Arrives

Figure9. Example of Phenix Overlay construction. [4]

$G_{random} = [1,3]$
$G_{friend} = [5,6]$
$G_{candidate} = [7,2,4,7]$
$G_{preferred} = [7,2]$
$G_{random} \cup G_{preferred} = [1,3,7,2]$

To make resilient connection, a set of guiding principle were to be followed, which can be reviewed, as follows. First, we try to hide the identity of highly connected nodes as much as possible, making the job of obtaining a complete list that contains these nodes practically impossible. Second prevent deals with neighbor updates, or 'node maintenance' (discussed below), where a network under attack can get well when existing nodes reorganize their connections and maintain connectivity.

Algorithm:

Begin $Obtain\ G_{host}\ from\ web\ cache$

$Divide\ G_{host}\ into\ G_{random}\ and\ G_{friend}$

$$let\ s\ the\ size\ of\ G_{friend}$$
$$G_{candidate} = \emptyset$$
$$for\ (x = 0; x < s; x++)$$
$$Send\ M_0\ ; where\ M_0 = ping\ (i, G_{friend}\ [x], 1, 0);$$
$$G_{candidate} = G_{candidate} \cup G_{G_{candidate}\ [x]}$$
$$G_{referred}\ is\ a\ subset\ of\ G_{candidate}$$
$$Connect\ to\ all\ the\ nodes\ in\ G = G_{random} \cup G_{preferred}$$

Hiding Node Identity- Three important methods to limit the possibility of a malicious user get hold of a global view of the whole overlay graph of the network are:

- When a node receives a ping message *M*0 will reply with a pong message, and forwards a ping message *M*1 to its neighbours. All nodes which receiving *M*1 will append the initiator to a list denoted by Γj. This list supports the concept of either 'temporary blocking' or 'black listing', where if the same initiator node sends a ping message with the intention of "crawling" the network to capture global or partial graph state information, such a message will be mutely dropped with no response sent back to the initiating node. Black lists can be shared with higher layer protocols to separate such malicious practices and can serve to detach such nodes. These mechanisms are outside the initial scope of this report. With a mechanism that detects a node crawling the network and silently discards the message and does not add it to the neighbour list, will not stop a malicious user, but rather, slow down its progress because the malicious node needs to get a new node ID to carry on the crawl of the overlay, or hang around for enough time for nodes to wash out their black lists Γ*j* . Peer-to-peer networks such as Guntella have proposed including the MAC address as part of the node ID, making it even more difficult for an attacker to obtain a new and distinctly different node ID at a rate fast enough to continue the crawl. It is worth noting that if joins/leaves of an overlay network are dynamic enough then crawling at slower time scales will not yield an accurate view of the network state and topology. Even though such a scheme helps limit the impact that malicious nodes can have, it still does not fully eradicate potential attacks on the network.
- Any ping message, whose *TTL (Time to live)* value is greater than 1 can be silently dropped. A non-confirming node with malicious intention might create such a message. Nodes drop these messages without replying to the initiator or forwarding such a message to neighbours. This has the effect of removing crawling even if the initiating node is not on the list Γ*j* of the receiving node.
- A node that creates backward connections to other nodes in the network will not return these connections when it gets a ping in any of its pong reply messages. This policy is not destined to defend the node's $G_{backward}$ sub-list of neighbours. Rather, it protects the identity of the node itself and any probable preferential status that the node may have, from an attacking node. If an attacker were to obtain a long neighbours list from a node, it can conclude that such a node is a highly connected node from the size of its neighbours' list. Thus, a node will only return the subset $G_{outside\_world}$ in a pong message. In this case, this node does not need to forward *M*1 to all of its neighbours. Rather, it only forwards *M*1 to nodes in its $G_{outside\_world}$ subset since these are the nodes that might risk exposure to an attacker, where,

$$G_{outside\_world} = [G_{random}, G_{prferred}, G_{highly\_preferred}]$$

Node Maintenance - In the event of an attack, the network needs to be responsive and able to reorganize connectivity in order to sustain strong connections between its nodes. A state probing mechanism that makes Phenix responsive to failed nodes or nodes that drops out of the overlay because of attacks.

The network needs to be responsive and able to rearrange connectivity in order to maintain strong connections between the nodes. h_i = no. of neighbours of node i.

$$h_i = h_i^r + h_i^p + h_i^q$$

Nodes examine their neighbours' table in order to make sure that they are not disconnected from the network due to node departures, failures, or denial of service attacks.

If the condition in the following equation is satisfied, signalling a drop, then node i run a node maintenance procedure.

$$h_i^r + h_i^q < threshold$$

If a node on the neighbour's list of node i leaves the network gracefully, then if informs all the nodes connecting to it by closing the connection. If a node is forcefully removed or it fails then node i will be informed for the fact only through probing a message sent to its neighbours as

$$M_2 = (\text{source=i, type= ping, TTL=0, hops=0})$$

If no reply is received after the timeout then the neighbouring node is declared as dead.

V. RESULTS

Shuffling- The procedure implemented in shuffling in both the algorithm is same. During shuffling process, when the age of the members of the list of the shuffling initiator node is incremented by one this step computes in O (1) time. Selection of neighbour with highest age and selecting l-1 random elements take O (l) time. Replacing the entry of the neighbour with highest age with its own address and age 0 takes O (1) time. Sending that list to the selected neighbour takes O (1) time. Receiving a list form the neighbour takes O (1) time. The entries pointing to itself are discarded from the list which is obtained from the neighbour and this step takes O (l) time. Now both the nodes are updated which requires O (2l) =O (l) time. So, the overall complexity of the shuffling algorithm is O (l).

Adding nodes- When any node wants to join a network, its introducer initiates 'c' random walks. When this random walk ends on a node that node replaces one of its cache's entries with a new entry of age 0, and the address of the originating node, this takes O (1) time. Then this replaced entry is passed to the originating node and the originating node updates its cache with the received entry, this step takes O (1) time. These two steps are to be repeated for 'c' times, so the total complexity of this algorithm is O(c).

In the new procedure of adding the nodes, getting a list from web cache requires O (1) time, then dividing this list again take O (1) time. Then pong messages are only sent to the one part of the received list (let size of this sub list be 's') so the time required in computing this is of O(s). So the total complexity of this algorithm is O(s).

In Cyclon and in CycloPhenix the shuffling process is same, so the execution time will also be the same. When a node was added in Cyclon then the time required for adding a node was of O(c). When we add the node with the new procedure defined here than the time required to add the node will be of O(s) where $0<s<c$. With this the execution time of adding a node is decreased and the network becomes resilient against the malicious behaviour of the attacking nodes.

Complexity Table1. Complexities of various approaches.

	CYCLON	**CYCLOPHENIX**
Shuffling	O(C)	O(C)
Adding node	O(C)	O(s) s<=C

VI. CONCLUSION

Peer-to-Peer networks are usually unstructured network. An unstructured network is simply a network which does not have a specific layout. Anyone who joins it becomes an integral part of the network, and changes its topology. Anyone can walk off at any time; the network will eventually change, but without any fall out (ideally). Ideally, such networks should have a low diameter; offer a high degree of resilience against the dynamisms of nodes, nodes failure and malicious actions with a purpose of distracting the network.

We have proposed an algorithm that constructs a P2P low diameter resilient topologies. It supports low diameter operations by creating a topology of nodes whose degree distribution follows a power law. It is robust to network dynamics such as joining/leaving, node failure and large scale network attacks.

REFERENCES

[1] Spyros Voulgaris, Daniela Gavidia, Marteen Van Steen. "*CYCLON:Inexpensive membership Management for Unstructured P2P Overlays*."

[2] Viktors Berstis "*Fundamentals of Grid Computing.*" IBM Corporation 2002.

[3] Jeffrey Dowskin, Sujiy Basu, Vanish Talwar, Raj Kumar, Fred Kitson and Ruby Lee. "*Scoping Security Issues for Interactive Grids.*" IEEE: 37th Asilomar Conference on Signals, Systems and Computers, November 9-12, 2003. Pacific Grove, CA USA.

[4] Rita H. Wouhaybai and Andrew D. Campbell. "*Phenix:Supporting Resilient Low-Diameter Peer-to-Peer Topologies.*" IEEE INFOCOM, 2004.

