Virtual Grid System Based on a Minimum Spanning Tree Algorithm

Hamed Saqer Al-Bdour, *Member, IACSIT*

***Abstract*—This paper describes the formation structure of a dynamic virtual GRID system that is built on the basis of the minimum spanning tree algorithm which increase the efficiency of GRID-system by presenting it in a dynamic virtual GRID system. Moreover, we develop a distributed routing algorithm that produces routing tables based on the routing agents located at the nodes of virtual GRID-systems. In addition to a comparative analysis of the proposed formation of the GRID system, we propose a new algorithm for constructing spanning tree for the DV GRID that adapts to the dynamics of the changeable environment and forms the tree based on the vertex with maximum degree. Our results clearly show the superiorities of the proposed algorithm over available algorithms in terms of performance and resource allocation. To create routing tables and virtual GRID-systems forming, we have used GridSim simulator.**

***Index Terms*—GRID system, dynamic GRID, CFG-computing Fabrics GRID, virtual private networks.**

I. INTRODUCTION

Grids are collections of heterogeneous computation and storage resources scattered along distinct network domains that provide tools that allow users to find, allocate and use available resources [1]. Grid computing joins computers that are distributed worldwide, allowing their computing power to be allocated and shared the world wide web permits access to information. Computer grids enable access to computing resources including data storage capacity, visualization tools, processing power, sensors etc. Therefore, grids can unify thousands of different computers' resources to establish a enormously powerful computing resource, accessible from the effortlessness by personal computers benefitting many applications, in science or business for example. Grids permit the analysis of huge investment portfolio in shorter periods, obviously speed up drug use progress and minimize time needed for design. Grid computing became useful for many scientific research applications including genetic engineering, mathematical modeling, biophysics, biochemistry, aircraft design, fluid mechanics, drug design, tomography, data mining, nuclear simulations, atmospheric studies, climate studies, neuroscience/brain activity analysis and astrophysics [2].

GRID-system includes hardware-and-software infrastructure that provide reliable, sustainable and inexpensive access to high-performance computer resources. As the computing environment, (CFG - Computing Fabrics GRID) uses a set of resources for various types of computer systems and computer networks, CFG provides the virtual environment with possibility of dynamic reconfiguration that provide maximum computational capability to solve the most complex tasks.

Typically, in Grid-systems, resources of different computers are combined together, which often are part of different computer networks managed by different administrators. The use of various protection techniques, such as a firewall, private IP address and safe shell (SSH) require special settings and maintenance by the administrator. For example, getting access to Dynamic Host Configuration Protocol (DHCP), the client must have database to store its address. The situation can be confusing when these addresses are local IP addresses, in this case, the overhead becomes a cumbersome process. With a small number of computers, the administrator can carry out the task manually. The user remembers the intermediate gateways to each host and keeps some specialized database to trace DHCP clients addresses. This substantially complicates the process of organizing the structure of CFG and its reconfiguration in the process of the functioning of GRID- system. To overcome this challenge we use the technology of virtual private networks (VPN) and on its base construct the virtual private Grid-systems (VPG) [3]-[10].

II. PROBLEM STATEMENT

At structural level the CFG can be represented in non-oriented loaded graph G (V, E, W), where V - set of nodes in the CFG; E - set of graph edges G (V, E, W), which correspond to communication channels in the CFG, W = {wi, j│i, j=1,2 … n} - set of graph edges weight. Edges Weight describes the parameters of information transmission through the communication channels. Here VPG represents subgraph Gs (Vs, Es, Ws) of graph G (V, E, W), where: Vs $\subseteq$ V; Es $\subseteq$E; Ws$\subseteq$W.

Generally Vs = V. As a rule, during organization of VPG the initial set E of graph edges G (V, E, W) is redundant. In this case, the task of forming a set of edges Es subgraph Gs (V, Es, Ws) leads to the problem of constructing a spanning tree, matching the criteria of optimality and have the required performance for fault tolerance by changing the parameters and structure of the CFG.

This is because the CFG consists of large number of nodes that dynamically change the structure of links. A large number of remote nods CFG defines the need for a decentralized spanning tree algorithm. In turn, dynamically changeable structure of links CFG predetermines the use of self-stabilized algorithms of spanning trees formation [11].

Manuscript received March 9, 2013; revised April 30, 2013.

H. S. Al-Bdour is with the Department of Information Technology, Mutah University, Karak, Jordan (e-mail: author@boulder.nist.gov).

The most known algorithms for constructing spanning tree are the various modifications of Prim's or Kruskal's algorithms [12].

The most common algorithm used to form the spanning tree is Prim's algorithm that consists of N - 1 iterations, each of which adds only one vertex, that does not violate the properties of the tree, that is, one end of an edge belongs to a formed tree, and another to added vertex that has the minimal weight. It should be noted that the Prim's algorithm differs its complexity, i.e. the lack of sensitivity to the degree of graph connectedness.

Distinctive feature of Kruskal's algorithm is the possibility to construct a tree simultaneously for several subtrees. In the process of the algorithm the separate subtrees are combined into a single spanning tree. The complete graph specified by the edges list. In the beginning of tree constructing the edges list is sorted according to weight in ascending order. In each step, the edges list is reviewed beginning from the edge, which has not yet processed.

The edge, which does not form a cycle with the edges already included into solution, joins the formed connectivity component. Before the start of algorithm operation, the number of connectivity components is equal to the number of the vertexes in the graph and contains, appropriately, one vertex for each component. After algorithm operation termination there remains only single connectivity component and the required spanning tree is determined by the edges which were used to join all connectivity components in single component.

The ability of Kruskal's algorithm to form a tree from several subtrees simultaneously is the precondition for developing the decentralized algorithm construction of spanning tree. However, in initial form the given algorithm is centralized and assumes a number of operations, such as viewing the list of edges.

In turn, the algorithm that forms the minimum spanning tree is oriented for dynamic environments, in particular to *CFG* and must have the following properties:

- Each network node should construct the spanning tree asynchronously, without knowing the topology of the entire network.
- The algorithm should be stable, to form spanning tree, even in frequently changing network topology.
- The algorithm must perform spanning tree reconfiguration with minimum time complexity.

In order to provide the maximum spanning tree stability and minimum time complexity of its reconfiguration, this paper proposes spanning tree formation relative to the vertexes with maximum degree.

III. The Construction Algorithm and Dynamic Reconfiguration of Spanning Tree CFG

By analogy with Kruskal's algorithm, initially, graph *CFG* is considered as spanning tree in which each vertex assumed as the root of the tree. During the operations of the algorithm, the vertexes are gradually combined in subtrees, which form a single spanning tree.

Each vertex v_i is characterized by its degree D_i determined by the number of adjacent edges of a given vertex. Its current priority P_i corresponds to priority of the root vertex of subtree $ST_j\,(V_j, E_j, W_j)$ which includes vertex v_i. The root vertex v^0_m of subtree $ST_m\,(V_m, E_m, W_m)$, assume it to be vertex v_k, that have the highest degree among all vertexes v_i of subtree $ST_m\,(V_m, E_m, W_m)$, i.e. $v^0_j = \{v_k \mid D_k \geq D_i \; \forall \; v_i \in V_m\}$.

At the beginning of spanning tree formation, the current priority P_i of each vertex $v_i \in V$ is equal to its degree D_i. On the first step of forming the minimum spanning tree with edges $e_{i,j} \in E$ the vertex v_i joins to adjacent vertex v_j with maximum degree D_j (Fig. 1). In this case, current priority P_i of vertex v_i becomes equal to D_j degree of vertex v_j.

This forms some set of subtrees $\{ST_m\,(V_m, E_m, W_m) \mid m=1, 2...l;\ l<n\}$. The first step of spanning tree formation is shown in Fig. 1. Dotted lines show the edges of the original graph, and solid lines show the edges of the formed subtrees.

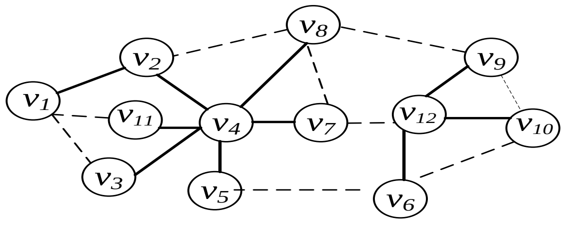

Fig. 1. First step of spanning tree formation.

On the second and subsequent steps of the spanning tree formation are formed individual subtrees of set $\{ST_m\,(V_m, E_m, W_m) \mid m=1, 2...l;\ l<n\}$. Each node asynchronously queries its neighbors to determine the network topology, if the nodes belong to different trees, then these trees are combined (Fig. 2).

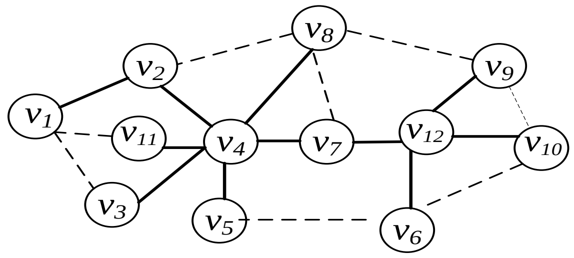

Fig. 2. Resulting spanning tree.

Any change to the CFG topology leads to change in the degree of individual vertexes subgraph $G_s\,(V, E_s, W_s)$, that results in the formation of new spanning tree. As an example, Fig. 3 shows the spanning tree with removed edge $e_{4,7}$.

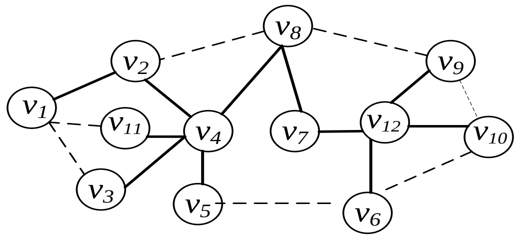

Fig. 3. Spanning tree after removing the edge ei, j.

IV. Analysis of the Spanning Trees Formation Algorithms

In order to analyze the temporal characteristics of the proposed algorithm and to compare it with well-known algorithms, we consider loosely-connected graphs with different number of nodes, which are the most typical for CFG environment structure. As shown in Fig. 4, the obtained dependences of time tree formation on the number of network nodes for the considered algorithms and the proposed algorithm compared with the well-known Prim and Kruskal algorithms has less time for the spanning tree formation, which suggests that is preferable to be used in a dynamically reconfigurable environment to which CFG is related.

In addition, when comparing the algorithms take into consideration their ability to parallel and decentralized execution. This possibility may be realized because each node asynchronously queries only its neighbors, while Prim and Kruskal algorithms realize a centralized construction of tree links, starting from the root of the tree.

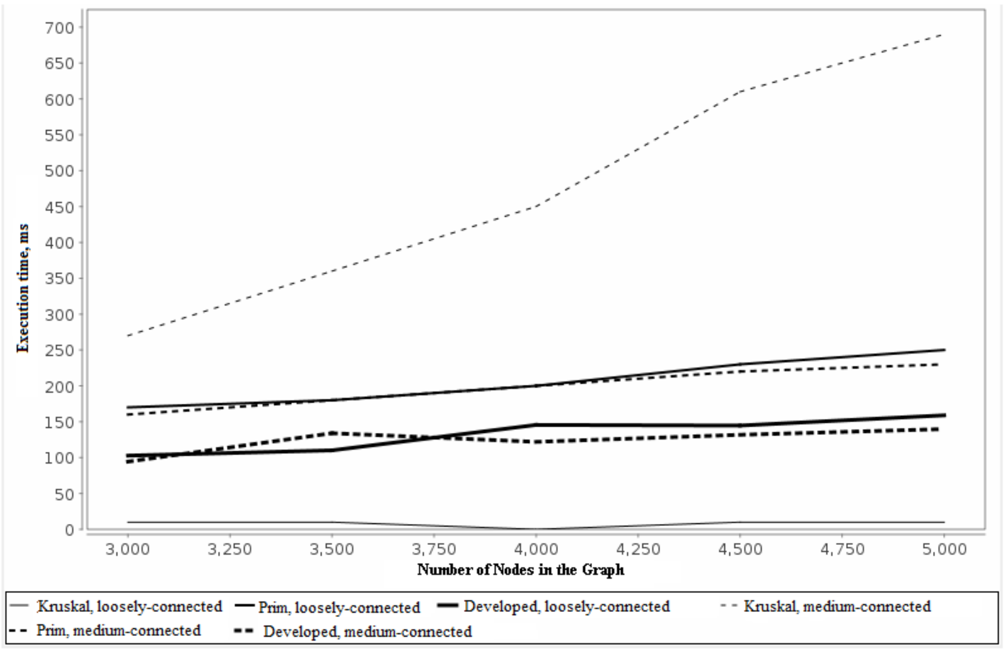

Fig. 4. Results of modeling the algorithms of a tree links construction.

The ability of a given algorithm for dynamic reconfiguration is the most important feature in dynamically changeable environment. This is because it functions completely decentralized, adapting to the changing *CFG* configuration, and does not require information about the network topology, and provides high performance and reliability.

V. Routing Algorithm

Now we present a distributed dynamic routing algorithm in terms of channels state. This algorithm is based on the interaction process routing agents (demons) that reside in each node of virtual GRID-system. During the execution, the neighboring agents are exchange routing information.

The process of routing algorithm can be described as follows:

1) Domains of Virtual Private Grid-network run on all hosts that the user wants to use. These domains can be loaded in any order.
2) Domains create and retain bidirectional (TCP) connections. They create the connections necessary to form a single graph through the exchange of information between neighbors. In this scheme, hosts with dynamic and/or private IP addresses become reachable, because they outside initiate bidirectional connections.
3) Hosts of spanning tree construction and domains are terminating connection creation.
4) The source node (home host) tracks the topology of entire network, determines the route to any participating node. Hosts and connections may fail or new hosts can become accessible. Whenever a network topology changes, domains construct new spanning tree by removing / adding connections and making all hosts accessible.

Thus, when the task is specified, source node should calculate the shortest path to the destination node, and tracks the topology of entire network by obtaining information from domains in form of topology fragments and then connects it. Every time when network topology changes, domains send new data to shell that updates information about the topology.

It should be noted that, in the beginning the domains do not know where the home host (source node), then each domain computes path to home host, using an additional variable *ToHome*. Essentially $ToHome_u$ equal to v, where v is the neighbor of u in the tree and is by one hop closer to home host than u, only when u - home host $ToHome_u$, equals u. If the host u has not computed path to home host yet, then $ToHome_u$ equals nil (ToHome is initialized as nil).

Node *u* computes the value *ToHome*$_u$, by repeating the algorithm with a certain interval note that the home host is considered as spanning tree root. Therefore, *ToHome*$_u$ computation, leads to define the host u parent.

- If host *u* is the home host, then *ToHome*$_u$ value becomes equal to *u*. Node *u* does not need to calculate path to itself.
- If *u* is tree leaf, and has only one neighbor. Therefore, *u* should send messages to home host via neighbor node and consider it as parent.
- If *v* is the neighbor of the node u and satisfies the conditions of *ToHome*$_u$ =*u*, then *v* is a child of *u*. If all nodes neighboring to *u* except *v*, are children to *u*, node *v* should be assumed as parent for u. host u sends messages to home host via *v*. If there are more than two neighbors who are not children to *u*, then *u* cannot determine exactly which node will *ToHome*$_u$ be its parent, in this case does not change.
- If node *v* neighboring but it's not a child to node *u* and *v* satisfies *ToHome*$_u$≠ *nil*, *v* has already determined the route to home host which does not pass via the node *u*, Therefore, *u* can send messages to home host via the node *v* and *u* accepts v as parent node.
- If *u* does not correspond to any of conditions above, *u* cannot determine the route to home host, and *ToHome*$_u$ becomes equal to nil.

VI. Simulation

In the research, there was implemented complex for simulation the considered routing algorithm and construction the minimum spanning tree as shown in Fig. 5.

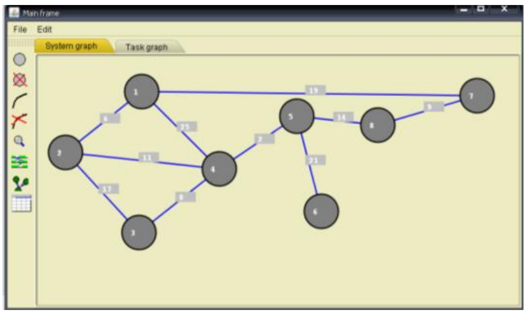

Fig. 5. Complex for simulation the virtual private grid network.

This complexity allows to specify the nodes and links between them as a weighted graph, as well as to edit a given topology. To perform simulation it is necessary to specify tasks to the source and destination nodes, (Fig. 6).

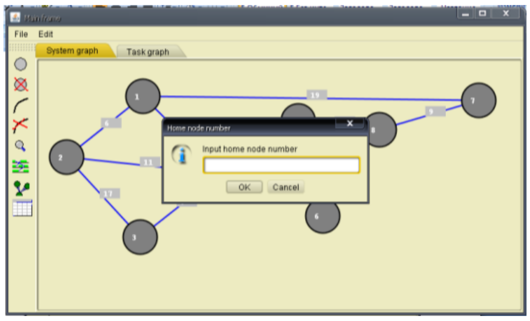

Fig. 6. Specifying the source node

As a result, on the screen there are displayed the selected (dark color) arcs of the minimum spanning tree according to specified topology (Fig. 7).

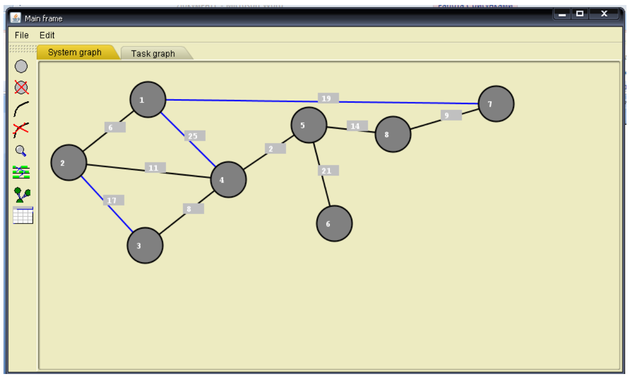

Fig. 7. Minimum spanning tree.

As a result, the algorithm generated the routing tables (Fig. 8), which contain information about the structure of the minimum spanning tree. In the table of each node indicates the number (code) of the nearest node to the vertex of the tree and the distance to this node. At presented in (Fig. 7), vertex 2 is the minimum spanning tree for root vertex. In this case, for vertex 5, the nearest vertex in the direction to the root vertex is vertex 4 and the distance to it is equal 2.

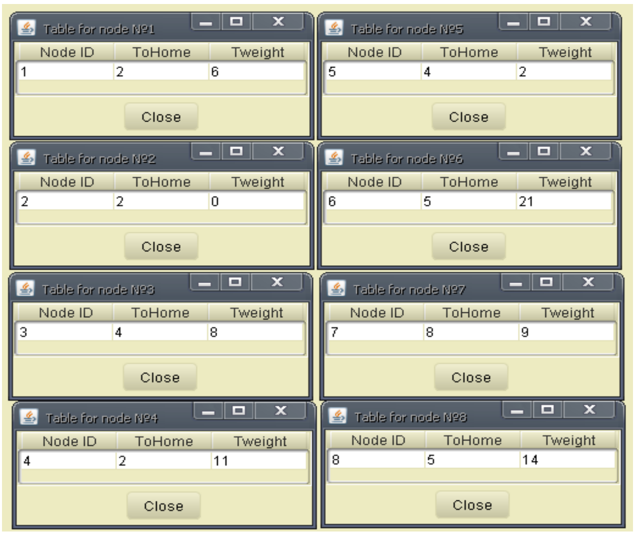

Table for node №1

Node ID	ToHome	Tweight
1	2	6

Close

Table for node №5

Node ID	ToHome	Tweight
5	4	2

Close

Table for node №2

Node ID	ToHome	Tweight
2	2	0

Close

Table for node №6

Node ID	ToHome	Tweight
6	5	21

Close

Table for node №3

Node ID	ToHome	Tweight
3	4	8

Close

Table for node №7

Node ID	ToHome	Tweight
7	8	9

Close

Table for node №4

Node ID	ToHome	Tweight
4	2	11

Close

Table for node №8

Node ID	ToHome	Tweight
8	5	14

Close

Fig. 8. Routing tables.

VII. Conclusions

This paper presented a model of virtual dynamic Grid that can be effectively used in specialized mobile Grid, which operates on networks based on Ad Hoc class. These mobile Grids are characterized by its similar opportunity nodes, not require the administrator's control, and allow fully dynamic distributed control mechanisms.

It also presented a self-stabilize spanning tree algorithm, intended to construct communication tree nodes and fairly distributes tasks over Virtual Dynamic Grid nodes.

The proposed algorithm meets the necessary requirements of a dynamic Grid environment, and has the properties of decentralized execution, that increases its speed and reliability and the possibility of operation in a dynamically changing environment.

REFERENCES

[1] S. Jarvis, N. Thomas, and A. van. Moorsel, "Open issues in grid performability," *I. J. of Simulation*, vol. 5, pp. 3–12, 2004.

[2] K. A. Manjula and P. Karthikeyan, "Distributed computing Approaches for Scalability and High Performance," *International Journal of Engineering Science and Technology*, vol. 2, no. 6, pp. 2328-2336, 2010.

[3] M. Tsugawa and J. A. B. Fortes, "A Virtual Network (ViNe) Architecture for Grid Computing," Presented at 20th Intl Parallel and Distributed Processing Symposium (IPDPS-2006), 04/06.

[4] E. Harney, S. Goasguen, J. Martin, M. Murphy, and M. Westall, "The efficacy of live virtual machine migrations over the Internet," Presented at the Second International Workshop on Virtualization Technology in Distributed Computing, Reno, NV, November 2007.

[5] B. Sotomayor, K. Keahey, and I. Foster, "Combining batch execution and leasing using virtual machines," in *Proc. the 17th international symposium on High performance distributed computing*, pp. 87–96, New York, NY, USA, 2008. ACM.

[6] M. McNett, D. Gupta, A. Vahdat, and G. M. Voelker. Usher, "An Extensible Framework for Managing Clusters of Virtual Machines," in *Proc. the 21st conference on Large Installation System Administration Conference*, pp. 1–15, Berkeley, CA, USA, 2007. USENIX Association.

[7] S. Adabala, V. Chadha, P. Chawla, R. Figueiredo, J. Fortes, I. Krsul, A. Matsunaga, M. Tsugawa, J. Zhang, M. Zhao, L. Zhu, and X. Zhu, "From Virtualized Resources to Virtual Computing Grids: The In-VIGO System," *Future Generation Computer Systems*, vol. 21, no. 6, pp. 896-909, Jun. 2005.

[8] Foster. What is the Grid? A Three Point Checklist. [Online]. Available: http://wwwfp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf, 2002

[9] S. Puri and Q. Abbas, "Grid Operating System: Making Dynamic Virtual Services in Organizations," *International Journal of Computer Theory and Engineering*, vol. 2, no. 1, pp. 2328-2336, February, 2010.

[10] F. Palmieri, "Introducing Virtual Private Overlay Network services in large scale Grid infrastructures," *Journal of Computers*, vol. 2, no. 2, pp. 61-72, April 2007

[11] F. C. G ̈artner and H. Pagnia, "Time-efficient self-stabilizing algorithms through hierarchical structures," in *Proc. the 6th Symposium on Self- Stabilizing Systems,* Lecture Notes in Computer Science, San Francisco, June 2003.

[12] C. G ́enolini and S. Tixeuil, "A lower bound on dynamic k-stabilization in asynchronous systems," in *Proc. 21st Symposium on Reliable Distributed Systems*, IEEE Computer Society Press, pp. 211–221, 2002.

Hamed Saqer Bdour is a professor at the Department of Information Technology /Faculty of Science at Mutah University / Jordan. He received his Ph.D. in Computer Engineering from National Technical University of Ukraine (Kiev Polytechnic University) in 1997. Dr. Bdour published several papers in the areas of Networks, Mobile networks and Security. His research interests include Parallel Computing, and Computer Network.