

Load Balancing in Wireless Ad Hoc Networks

Anand Mohanrangan

anandmr@stanford.edu

With the reduction in the cost of computing power, it has become easier and more economically feasible to deploy to grids of many nodes to act as sensors in such situations a disaster area to gather information. In addition, with the increasing popularity of hand-held devices and laptops, on-demand networks of computers or other electronic devices is also becoming more plausible. Such networks fall into the general category of ad hoc networks. In ad hoc networks, it can happen that some nodes are assigned greater load than others. Often, nodes on “popular” routes may have to forward more packets than nodes on the boundary. A special class of routing protocols for ad hoc networks deals with load-balancing as an energy-constraint that is integral to some ad hoc networks. Load balancing is also very important in heterogeneous decentralized systems – systems having nodes with varying battery lives or processing capabilities.

In this summary, I have looked at two papers that deal with the problem of load balancing in ad hoc networks. Each of these papers attacks the problem of load balancing in different ways. They also involve simulations which compare the performance of each of the schemes described. In addition to summarizing and analyzing various aspects of each paper, I will also attempt to compare and contrast the techniques in each paper with the other. These papers do not exactly solve the same problem, in that [2] looks specifically looks at load balancing in clusters while [1] considers a more generic case. However, in my conclusions I hope to bring out some advantages and disadvantages of both schemes as well try to obtain insights from one case that could be applied to the other.

The first paper, [1] was titled **Balancing Loads in mobile ad hoc networks**. This paper tries to develop a load-balancing algorithm that would try to find the best node to share the load with while minimizing the communication overhead involved in the load-balancing. In addition, the authors intend this algorithm to be invoked only when the node feels the need to spread out its load to other nodes based on certain thresholds for factors such as execution time, battery power and the balance factor.

The balance factor (BF) is the inverse of the variance of the load of the nodes in the system and is used to measure how well balanced the system is. A higher BF indicates a better balanced system (with $BF \rightarrow \infty$ for a perfect system). In order to reduce the cost of calculating the balancing factor, one of the following could be done. The frequency of calculating the BF factor has to be reduced or it could only be calculated when load falls below a certain level. Or only overloaded nodes can be allowed to distribute their load to other systems.

In order to explain the working of their algorithm, the authors give an example of a system of laptops, PDAs and other distributed nodes self-organized into a network. Periodically, the average system load is calculated. This is done by each node distributedly sending out query packets, essentially flooding the network. Eventually, each node has enough information to calculate the average system load. This metric is then used to compare against its own load. A nodes whose load is within a certain range (which is calculated depending on the type of node it is and its processing capability and battery power), is considered to be already balanced. If the node’s own load is above this range, it is considered “overloaded” and it then sends out requests for other “underloaded” nodes 1-hop away to share in its data processing. If none of them respond, then the request is propagated to nodes 2-hops away. This process repeats for nodes further away until a node is found to share the load. These operations are described in a 5-step process described in the paper.

The effect of this algorithm was observed through simulation. A heterogeneous collection of randomly distributed nodes was used. Each node generated loads at a different rate, the average of which for the system was λ . The number of nodes was varied from 100 to 300, while the average system load was between 100 million and 1000 million instructions. Nodes within 20% of the average system load were considered balanced. An overhead of 0.01% of instructions sent to the other nodes was included per hop that the data was sent and returned. The performance metrics used were the execution time improvement, the standard deviation in the number of instructions processed and the balance factor, BF.

The authors observed a drop in the execution time for both large numbers of nodes as well as larger job sizes. The authors also note a 3-fold decrease in the standard deviation in the number of instructions processed at each node. Finally, a higher BF was observed in the load-balancing case. Based on these results, the authors conclude that the load-balancing system helps achieve equilibrium in terms of per node battery utilization.

It must be observed that this paper gives a very high-level approach to load balancing. None of the results derived are unexpected. The authors did demonstrate one possible, though simplistic, framework for load balancing across different. However, a number of aspects are not covered in depth. For example, the overhead due to the extra communications is included by increasing the data to be processed. This is not necessarily accurate. In addition, considerations such as delay are not taken into account. Also, while the overall system energy consumption is balanced, it is not discussed whether the overhead associated with load-balancing actually leads to a much greater overall energy consumption.

Based on these observations, any number of extensions are possible to this paper. Every aspect of this system, from the range within which a node is considered balanced, to the energy overhead for balancing and delay constraint can be examined in greater detail. This paper provides a good starting point from which more detailed analysis should be conducted.

The second paper was titled **Load-Balancing Clusters in Ad Hoc Networks**. This paper deals with ad hoc networks where mobile nodes have been loosely classified into clusters based on their current location. One of the nodes in each cluster is selected as a clusterhead. Routing in such networks takes place with the clusterheads acting as virtual backbones. This can directly result in greater energy depletion in the clusterhead nodes as compared to the rest of the nodes in the network. The authors propose a load-balancing heuristic to extend the battery life of a clusterhead before allowing it to retire and allow another node to become the clusterhead.

The authors describe previous work in the area of clusterhead election. These include the LCA (Linked Cluster Algorithm), LCA2 and Min-Max heuristics. Most of these algorithms tend to be stable. Min-Max has faster convergence than LCA. However, both these algorithms favour nodes with higher node IDs in selection of clusterheads. Another class of algorithms base their selection on the node which has the highest degree of connectivity to nodes with a fixed number of hops in a certain group. However, in this case, even a low degree of mobility can cause excessive control overhead for changing clusterheads. The authors propose to combine the un-biased approach of the connectivity algorithms with the stability of the Min-Max and LCA2 heuristics.

The authors state that the main goals of the new heuristic are reducing the control overhead, improving stability, allow all nodes equal opportunity to be the clusterhead (irrespective of node ID) and extend the clusterhead duration based on some input parameter. In order to reduce the control overhead, only 2 parameters are used to identify nodes and their electability as

clusterhead (thus reducing the need to query for statistics like battery life when trying to determine the next node to be elected as clusterhead). The two parameters are the Physical ID (PID) to uniquely identify a node and a Virtual ID (VID) which is initially the same as PID but changes with time to reflect electability.

Load-balancing in this paper is treated as an extension to clusterhead election. A circular queue is maintained to determine the next node to be elected clusterhead. An elected clusterhead is assigned a VID larger than any other in the queue. Each node relinquishes its role as clusterhead after it has exhausted its clusterhead budget and goes to the bottom of the queue and sets its VID to 0. The clusterhead budget is determined by each node independently depending on factors such as battery power and so on. The node with the highest VID in the queue is elected the next clusterhead. As a node moves through the queue, it increments its VID by 1. If two clusterheads move into range of each other, the one with higher VID is selected as clusterhead. The other clusterhead adds to its VID prior to becoming clusterhead, the number of times the clusterhead election was run on it.

The authors propose a second heuristic based on the degree-of-connectivity. In this case, only one parameter is used. This is the elected degree, which defines the number of nodes the clusterhead was connected to when it was elected clusterhead. Everytime the clusterhead selection is run, if the number of nodes currently connected to is less than the elected degree – MAX_DELTA (which is a user-defined parameter), then a new node is selected as the cluster.

Simulations were performed to see the performance improvement obtained by this new heuristic on the LCA, LCA2, Min-Max and Highest-Connectivity Degree algorithms. The number of nodes was varied with 100, 200, 400 and 600 nodes. The distance between nodes was varied (though the path loss model is not specified). Nodes at 2 and 3 hops were included in a cluster and the performance was observed in each case. The duration that a node remained as the clusterhead was the performance metric, since a longer duration implied greater stability. It was observed that for all algorithms, the addition of the new heuristic lead to better results. However, the authors do not specify which heuristic specifically leads to the improvement. It could be assumed that the first heuristic was used for the Min-Max and LCA algorithms while the second was used for the Degree algorithms. Based on these observations, the authors conclude that their algorithms perform well.

It can be observed that the improvements suggested by the authors certainly lead to improvements over the traditional algorithms. However, a number of avenues have been left unexplored and choices not clearly explained. For instance, while the algorithms allow for a fairer selection of the clusterhead, the overhead imposed by the clusterhead process is not discussed. For both heuristics, considerable overhead may be incurred in exchanging the parameters needed to make a decision on the clusterhead. For the first heuristic, it is not clear which node maintains the current state of the various nodes to determine the next clusterhead. Also, in a mobile situation with nodes changing clusters frequently, the information exchange may be prohibitive.

Again, based on the above, this paper only gives a high-level overview and does not dwell into specifics. Many of these specifics such as the parameters used for clusterhead election need to be better defined to get optimal performance.

Both the papers discussed algorithms that can be used to more uniformly balance the load on different nodes within a network. While the first paper considered a heterogeneous collection of nodes and tried to balance the processing involved, the second paper looked more at the routing overhead. However, concepts discussed in each paper could be applied to the other. For example,

the first paper talks about using a Balancing Factor to determine the load on a node. This factor could be made to incorporate the routing load as well. Then, it could be used as an additional metric or included in the current parameters used by the two heuristics in the second paper. Consequently, some nodes might be involved in intensive processing, while others could be more involved in the routing aspects. At present, the Virtual ID in the first heuristic is left as an open measure of the load of a node. The balancing factor could ideally be incorporated here.

Both the papers did not really focus on using load-balancing in conjunction with energy conservation. For instance, in order to conserve energy, nodes could be put to sleep after clusterhead election. They could be woken up for clusterhead election (and any data transfers to those nodes could be processed then).

In conclusion, both papers provide interesting starting points from where future research can be conducted. These papers provide an insight into some of the techniques that can be used to achieve load balancing in ad hoc networks. They have demonstrated that without using very complex techniques, considerable savings can be obtained. This is especially of importance in energy constrained environments. However, these algorithms need to be further refined to extract much greater savings from them.

References:

[1] Balancing loads in mobile ad hoc networks

Turgut, D.; Turgut, B.; Das, S.K.; Elmasri, R.;

Telecommunications, 2003. ICT 2003. 10th International Conference on , Volume: 1 , 23 Feb.-1 March 2003

[2] Load-balancing clusters in wireless ad hoc networks

Amis, A.D.; Prakash, R.;

Application-Specific Systems and Software Engineering Technology, 2000. Proceedings. 3rd IEEE Symposium on , 24-25 March 2000