AN ALGORITHM TO DISTRIBUTE AND LOAD BALANCE SHARED-FATE TASKS IN MOBILE AD HOC NETWORKS

BY

ALEXANDER THOMAS LOEB

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2011

Urbana, Illinois

Adviser:

Associate Professor Robin Kravets

Abstract

The pace of mobile networking is rapidly changing. Each year, waves of new smartphones, tablets, netbooks, and laptops are being released to consumers and subsequently being adopted in greater numbers than ever before. The effect is a much denser environment, rich with mobile devices able to keep people connected throughout their daily activities. With these advances come new problems. Devices are becoming so dense in social environments that there is increasing redundancy in computational efforts, wasting energy. Tasks including 802.11 neighbor discovery and GPS location among others have deterministic results for devices in close proximity, as is common in dense environments. It is the objective of this thesis to design an algorithm to allow varied devices in dense mobile ad hoc networks to distribute and load balance these tasks among each other, ultimately leading to energy savings through reduced redundancy.

To CKH

Acknowledgments

Thanks to Robin Kravets for general guidance and support, to Mehedi Bakht for countless conversations and emails, to Johnny Carlson for help handling the shortcomings of the Android platform, and to Simon Krueger for inspiration. Lastly, additional thanks to Robin, Mehedi, and Johnny as I am just a fraction of a greater scheme.

Table of Contents

Chapter 1: Introduction

Mobile devices ranging from laptops and netbooks to tablets and smartphones are becoming increasingly pervasive as their mobility and connectivity increase. From 2010 to 2011, adoption rates of smartphones alone increased roughly 10% in the US, UK, and Spain, putting the total adoption rate of each at approximately 30%. Nearly every new mobile device has at least one wireless interface, more commonly having two or more. The cost of this flourishing wireless connectivity is the energy consumption required to operate the wireless interfaces of these mobile devices. Further, it is widely recognized that these wireless interfaces are the most power intensive components of modern mobile devices. Specifically, wireless interfaces account for over 50% of the total energy consumption on mobile devices when active [8]. In accord with Amdahl's law there has been a lot of work done on reducing energy consumption of these wireless interfaces. Previous work has shown that redundant links in wireless ad hoc networks can be shut off intermittently by exploiting node density [2], effectively conserving energy. It has further been shown that leveraging multiple, complimentary wireless interfaces in a coordinated manner [8] [9] effectively conserves energy. Both techniques rely on using other nodes or radios to allow a node or nodes to temporarily turn off their high power wireless interfaces without compromising network connectivity.

However, existing work focuses on tasks either within a single node (e.g., device association) or on tasks that require a set of nodes to act in concert (e.g., routing). An artifact of the increased node density resulting from an increase in mobile devices is the formation of a third, new and unstudied, task. When nodes are in close proximity and acting independently, any sensory input is likely to be identical for those nodes, as if both nodes occupied the same physical space, and they would both observe the same environment around them. While it is impossible for multiple nodes to physically occupy the same space, many sensors are inaccurate enough that the distance between nodes in close proximity may cause them to observe the same or very similar environments. Examples of this kind of task are GPS location and long range neighbor discovery. These tasks can be said to be fate-shared and exhibit nearly deterministic results across nodes in close proximity.

Given that an 802.11 radio consumes roughly an order of magnitude more energy than a Bluetooth radio under nominal use of each radio respectively [8], in combination with the observation that nodes in close proximity achieve deterministic results for this new class of tasks, it would be energy conserving to use a low power radio such as Bluetooth to distribute the responsibility of performing these tasks across nodes in close proximity. The assumptions are that any nodes not performing a task of this nature has its radio turned off, and that the range of the low power radio is within the approximate tolerance for error of the sensory input. With respect to GPS and 802.11 neighbor discovery, this happens to be true.

While it is not the aim of this thesis to evaluate the energy savings of these new tasks, it is the aim of this thesis to design, deploy, and evaluate an implicit leader election algorithm that enables mobile devices in highly dynamic ad hoc networks to coordinate their efforts in accomplishing these shared-fate tasks. The following chapters will examine the existing work motivating the design of such an algorithm, the design and implementation of the algorithm, its evaluation, and finally conclude.

Chapter 2: Motivation

The motivation for this thesis begins with the observation that increased device availability and adoption rates of modern mobile devices results in greater network density in social environments. As such, it is appropriate that the exploration of existing work begins with work attempting to exploit similar densities. Span [2] exploits these node densities to eliminate redundant links in 802.11 packet forwarding. Specifically, in ad hoc networks, every node that has an 802.11 radio on can serve as a vertex along a path while routing a packet from source to destination. In a dense ad hoc network, there is an abundance of links, creating redundancy. Span identifies this redundancy and reduces it by forming one hop clusters with the one node acting as a clusterhead. The clusterhead keeps it 802.11 radio active to route packets through the network while the remaining nodes in the cluster put their 802.11 radios into power saving mode. This drastically reduces energy consumption within the network. Further, to attain fairness, each node use energy levels and neighbor counts, combined with a random back-off value, to decide whether or not it should elect itself as a clusterhead. This allows the nodes to rotate in and out of leadership, distributing the cost among one another, and ultimately increasing the longevity of the entire network. Finally, each node does all of this while making purely local decisions. The caveat to Span is their requirement that connectivity must be maintained at all times when possible, and so their definition of fairness is not absolutely fair. That is, if a node is a bridge between two connected components and it is the only such node, it will never be rotated out of leadership, causing it to expend more energy than other nodes in the network. The authors of Span argue that node mobility reduces the likelihood of a node remaining in such a position for prolonged durations. While many of the core features of the work in this thesis are similar to Span: randomized back-off, fairness, clustering, and leader election, the application and mechanisms vary considerably in detail. Routing is inherently a task that must be distributed in nature to operate correctly. The tasks in this thesis are inherently isolated operations, always performed by a single node even when many nodes in the network are performing it and obtaining the same results. Further, Span is unable to turn off a node's 802.11 radio because it disrupts the node's ability to be asynchronously woken up. This thesis leverages the commonality of multiple, heterogeneous wireless interfaces to allow each node to turn off its long range, typically 802.11, radio, resulting in greater energy savings.

Expanding further on the subject of multiple, heterogeneous wireless interfaces on modern devices, the exploration of existing work that follows is a summary of relevant works that similarly exploit multiple, heterogeneous wireless interfaces. CoolSpots [8] achieves energy conservation by designing several protocols to control which wireless interface is used to transmit and receive data based on the available bandwidth of the channel. Specifically, the authors recognize that Bluetooth consumes roughly an order of magnitude less energy when active compared to 802.11 at the cost of range and bandwidth. Upon further assuming that it is fairly common to be within Bluetooth range of 802.11 infrastructure, and thus the only practical trade off between Bluetooth and 802.11 becomes the energy cost for bandwidth. This allows them to develop several protocols that control the node's ability to detect when this assumption is true, and more importantly when to switch from using Bluetooth to 802.11

and vice versa. While they explore the extremes of always using 802.11 or Bluetooth, the more interesting mechanisms explore a static tolerance, hybrid approach based on channel capacity estimates along with a static tolerance, and a hybrid capacity estimate with dynamic tolerance. In all three, when the bandwidth is repeatedly observed to be less than the tolerance, the node switches to Bluetooth. In the static version, the switch to 802.11 is simply when the bandwidth is repeatedly observed to be above tolerance. The capacity estimate schemes use the RTT from frequent pings to estimate the available channel capacity to determine when to switch to 802.11.

In contrast to improving data transmission, [9] leverages multiple, heterogeneous wireless interfaces to achieve energy savings during device association and connection setup. The authors, similar to CoolSpots, note that there are many scenarios in which both short and long range wireless interfaces are in range of an access point, and in those scenarios, it is worth leveraging the short range, low power interface whenever possible. In [9], both Bluetooth and Mica2 Mote radios are examined as the low power, short range interface while 802.11 is used as the high power interface. Further, the authors focus on device association and connection setup to achieve energy savings as this is typically more common than data transmission and costs just as much. Their approach requires that the access point broadcast a beacon over the low power interface that contains the necessary association information for 802.11. This way, a node can disable its 802.11 radio, conserving energy, and upon receiving a beacon over the less expensive interface, enable 802.11 and immediately connect to an access point without needing to waste energy scanning and acquiring the same information that was sent in the beacon. While the Mote proved to be the most effective at this, Bluetooth also resulting in energy savings, but caused an increase in connection latency due to overhead in the design of Bluetooth.

CoolSpots effectively leverages multiple, heterogeneous wireless interfaces to conserve energy during data transmission with respect to a single node. [9] also leverages multiple, heterogeneous wireless interfaces, but it explores device association and connection setup, but still only for a single node. Further, *both* only consider when a node is in range of infrastructure with both long and short range interfaces, additionally requiring that the infrastructure support both interfaces. In contrast, this thesis aims to exploit *both* the energy savings *and* the range disparity of the heterogeneous wireless interfaces to coordinate the efforts of many nodes in an ad hoc setting.

To distribute a task across a cluster of many, otherwise independent nodes, a key step is the election of a leader to coordinate their efforts. [4] provides two algorithms for leader election in mobile ad hoc networks. The first is an adaptation of the TORA algorithm [7] to support the propagation of a leader id while the second is an extension of the first to support concurrent changes to the topology during the leader election process. Both algorithms form a directed acyclic graph in which the root is known by every node in the graph and as a result, is considered the leader of the graph until a topology change in detected. Both algorithms construct the graph by having some node observe a topology change, triggering it to propagate updates to all of its neighbors. Every node, in turn, examines its neighbors status and from that, derives its new status. It suffices to say that the status contains enough information to make the updates ordered and unique, as well as allow each neighbor to adjust its place in the graph with respect to the leader's position. While concurrent updates cannot be ordered by definition, the adapted version adds a mechanism to allow one update to overtake another update by comparing the updates and effectively using a tie break.

Another approach to leader election in an ad hoc mobile setting was proposed in [10]. In their algorithm, they also construct a directed acyclic graph in the form of a spanning tree, but they do so only temporarily. Specifically, the algorithm uses a breadth first search approach to propagate an election message to every neighbor of a node that observes a topology change. Upon receipt of an election message, a node marks the sender as its parent and continues the propagation to its neighbors, excluding its own parent. Upon receiving an election message from a node other than one's parent, an ack is sent to the sender. Once all neighbors, excluding one's parent, have ack'd the election message, a node then ack's its parent's election message. The ack contains the summary of the information needed to elect the new leader. Once the election initiating node has been ack'd, it selects the leader using the information in the ack, and propagates a leader message in the same manner as the election message. A key difference in this approach from the approach in [4] is that the leader elected is considered a most-valued node in some respect (i.e., this algorithm is extrema-finding, where the implementor can define what defines extrema). This new algorithm works well on a static topology. To adapt it to handle a dynamic topology with concurrent topology changes, two more messages, probe and reply, are added to allow node's to request and reply to heartbeats. While the technique used in the initial algorithm is referred to as a diffusing computation, a dynamic topology suffers from having multiple diffusing computations at once, resulting in conflicting results to the leader election process. The modified version assigns a unique value to each diffusing computation and dictates that a node only participate in a single diffusing computation at a time and that higher valued computations cause a node to immediately stop participating in a lowered valued computation in favor of the higher valued computation. Further, all node's now send probe messages to any node that it is expecting an ack from. If a reply message is not received within a timeout period, the node that did not reply is dropped from the computation.

While these algorithms perform leader election in a mobile ad hoc setting, they incur a lot of overhead by constructing a directed acyclic graph and propagating changes for every topology change when the graph's only purpose is to determine which node is the current leader. This thesis is only concerned with clusters with a one hop radius given the constraint on the accuracy of the task results, and as a result does not need to construct a directed acyclic graph because each node is adjacent to the current leader. Further, for the purposes of this thesis, it is sufficient to allow partitions to occur without immediate detection and correction as they will automatically be corrected within a time bound less than or equal to the length of a single task interval. Finally, the work of this thesis is able to achieve fairness through a randomized approach, avoiding the need to define fairness in terms of an extrema and thus not requiring that leader election be extrema-finding.

Given that traditional leader election algorithms are ill suited to handle the scenario proposed by this thesis, a more unconventional approach is needed. Span [2] allows clusters of nodes to coordinate packet routing by constructing a dominating set across the clusters and having each clusterhead act as a vertex in a routing backbone. While the work in this thesis does not require that the clusterheads be able to reach one another, the approach in Span, the approximation of a dominating set, is still valid to form the clusters by selecting clusterheads and designating all one hop neighbors as part of that cluster. To evaluate this approach, a study of existing work on dominating sets is necessary. The work in [3] first establishes that the problem of finding a minimum connected dominating set is known to be *NP*-hard, and then proposes two greedy approximation algorithms as well as variants on each. The first algorithm uses a greedy heuristic to choose a start node, growing a tree outward from that node. Specifically, the maximum degree node is chosen to be the start node. Once at a given node, mark it as part of the dominating set and mark its neighbors as connected to the dominating set. The algorithm then proceeds to choose the next connected node with the highest degree of unmarked neighbors that is not marked itself to add to the dominating set. This repeats until all nodes are in connected. The second algorithm proposed consists of two phases. In the first phase, instead of growing a tree, it assembles pieces that are not connected to the rest of the set. These pieces are assembled by always choosing to mark the next node that results in the largest, non-zero, reduction in the number of pieces. The second phase connects the connected components.

As discussed previously in this section, constructing and maintaining a tree in a mobile ad hoc environment is undesirable due to the high dynamism and the message overhead. With this in mind, the previous algorithms to approximate a connected dominating set are infeasible. However, [1] acknowledges this problem and proposes a localized, distributed algorithm to operate in mobile ad hoc environments. The algorithm proposed models the mobile ad hoc network as a unit-disk graph, constructs a maximal independent set, and finally connects the components by identifying connector nodes as those nodes that are between a pair of dominators that are at most three hops away from one another. The authors then clarify that connectors stay connectors only so long as they connect at least two dominators, and if this fails to be true at any time, the connector becomes a leaf. Further, the selection of the connectors included in the connected dominating set are chosen greedily (i.e., there may be more than one path between two dominators, and the path chosen is not necessarily the shortest path). While this algorithm is much quicker to construct a connected dominating set and does so in a manner that is easier to maintain across changes to the topology, the work in this thesis does not strictly require a connected dominating set, although the approach is more desirable than traditional leader election algorithms.

In the reexamination of Span with respect to dominating sets, the problem of leader election can be seen as a matter of contention for the role of leader within the cluster. Given that a cluster does not care which node performs the task, but only that one node perform it at any instant, the problem can be viewed as a form of medium access control. The MAC parallel exists because MAC protocols do not care which node is transmitting at any instant, so long as only one node is transmitting at a time and further that if any node wants to transmit then some node is allowed to transmit. The work in this thesis further parallels MAC protocols in that they too attempt to achieve fairness among nodes. Both [5] and [6] examine fairness in MAC protocols, which is directly related to the work in this thesis as it relies on a form of random back-off selection during a contention window in order to achieve load balancing across nearby nodes.

Chapter 3: Design

To allow independent nodes to distribute the execution of a task, common to the entire set of nodes, two requirements must be met: the nodes must be aware of each other, and the nodes must be able to distribute the results of the task to the entire set of nodes. Awareness is required to identify redundancy in the efforts of individual nodes. If two nodes do not know about one another, they cannot coordinate their efforts. To eliminate redundancy, nodes must be able to defer execution of the task to another node. This is only valid if that node can then distribute the results of the task to the entire set of nodes. The algorithm put forth in this thesis further requires that nodes be able to distribute the workload in networks exhibiting high dynamism such as those of mobile ad hoc networks, and finally that the workload be distributed fairly across the entire set of nodes.

The purpose of this algorithm is to allow nodes in close proximity to distribute the execution of a task across the entire set of nodes. Specifically, a task is any operation that is common to all nodes in the set, but only requires one node to execute such that the entire set can obtain the results. This is possible for any task that has results that are insensitive to the minor variations in location of the nodes in the set. This thesis defines minor variations to be equivalent to the range of the short range wireless interface supported by each node. An example of such a task is obtaining a GPS location for a given node. GPS sensors are roughly accurate to the range of a Bluetooth radio, and are far more expensive to operate. Thus it is more efficient to allow one node to obtain its GPS location and distribute the result to the other nodes within Bluetooth range. This allows the other nodes to save energy by not enabling the expensive GPS radios. However, there are many tasks fitting this description, so the rest of the thesis will generally refer to "the task" instead of any one specific example. Further, any node that is said to be executing the task is a leader.

In order to handle the high dynamism resulting from highly mobile nodes, the task coordination algorithm operates on local information, making purely local decisions at each node. Further, in order to distribute the workload of performing the task, nodes must be able to take turns. As a result, the algorithm consists of two intervals, similar to MAC protocols: the task interval and the contention interval. During the task interval, a node is either performing the task with its high power wireless interface enabled or it is listening for beacons with its low power wireless interface while its high power wireless interface is disabled. During the contention interval, a node selects a contention slot and waits for that slot to arrive before beaconing, with its low power wireless interface, to signal that it will perform the task. If it receives a beacon from another node prior to the selected slot, the node returns to listening for beacons instead of contending.

3.1 The Task Interval

To maintain coverage of the task at all times, the leader performs the task until the next leader begins performing the task (i.e., with respect to the high power wireless interface, the task interval extends to the very end of the contention interval). If this were not the case, there would be a gap in performing the task during the contention interval. However, with respect to the low power wireless interface, each node transmits beacons during the segment of the task interval that does not overlap with the contention interval. If nodes were to transmit low power beacons during the contention interval, they would be mistaken for election beacons and thus break the protocol. As this thesis is not concerned with the details of the task itself, any reference to the task interval throughout the remainder of this thesis refers to the segment of the task interval that does not overlap with the contention interval (i.e., the task interval is to be interpreted with respect to the low power wireless interface only).

During the task interval, each node in the set transmits beacons over the low power wireless interface. If the node is not a leader, these beacons only contain a synchronization value. If the node is a leader, the beacons are more frequent and not only contain the same synchronization value, but also contain any partial results of the task. The synchronization value is the beginning of the task interval of the lowest id node in the cluster at the time that node joined the cluster. This synchronization value allows new nodes to join the cluster and locally calculate the progress of the current task interval so that the node knows when the next contention interval starts. If the lowest id node or the leader leave the cluster, the synchronization value remains the same so that the remaining nodes in the cluster can still coordinate their efforts. While this offers a low overhead, local solution to these scenarios, it also creates the potential for a cluster to not have a leader.

3.2 The Contention Interval

The contention interval is the period of time between task intervals. During the contention interval, it is the responsibility of the cluster to elect the next leader. The contention interval is subdivided into slots, called contention slots. The number of slots in the contention interval is said to be the length of the contention interval. In the simple contention interval, each node in the cluster randomly selects a slot in the contention interval (i.e., an integer value between zero and contention interval length, exclusive). Each node waits until the slot it chose arrives. If the slot arrives and no other node has chosen an earlier slot, the node transmits a beacon indicating that it is the new leader. If another node has chosen an earlier slot, the beacon sent by that earlier node indicates that a leader is chosen and all other nodes are to stand down, returning to non-leader status for the following task interval. Further, it is possible to have multiple leaders if multiple nodes choose the same contention slot. However, with a well-chosen contention interval length, multiple nodes choosing the same contention slot has a low probability, and if the task intervals are short, there is less time spent without a leader in the event that the leader leaves. However, these circumstances designate this algorithm as a best effort approach. Finally, if a node with a lower id than that of the synchronization node joins the cluster, all of the nodes in the cluster must resynchronize.

However, this is potentially unfair because the same node or subset of nodes may be repeatedly elected by randomly choosing the lowest slot, putting more work on themselves and less on the remaining nodes. To avoid this scenario, an exponentially decaying back-off mechanism is added. This mechanism requires that each node maintain an additional parameter, the contention back-off value, to act as a history of its contributions to the efforts of the cluster. The contention back-off value changes for each node after every contention interval based on the results of that contention interval. Each node initializes the back-off value to the length of the contention interval. If the node is not elected during a contention interval, it halves its contention back-off value to increase the probability of being elected during subsequent contention intervals. If the node is elected during the contention interval, the node resets its contention back-off value to the initial value, thus reducing its probability of being elected multiple times consecutively.

3.3 Implementation

The proposed algorithm was implemented for Android platform and was deployed to a dozen Google G1 and G2 phones. While the implementation is a full implementation of the algorithm in this thesis, it differs considerably in behavior due to practical limitations of modern smartphones. Specifically, the algorithm outlined in this thesis assumes broadcast capability for the low power, short range radio. However, modern smartphones typically only have a Bluetooth radio for short range communication and Bluetooth lacks broadcast capability. The effect of this is that the communication between devices in a cluster must be explicit between nodes in that the only way to simulate broadcast with Bluetooth is to independently connect to each neighbor. The main consequence of explicit communication is that the synchronization value is unnecessary. This is because the only way to know the contention interval begins is for the leader to connect to every neighboring node and ask each node which slot it picked. The leader then picks the lowest slot chosen and reconnects to each neighbor, announcing which node is the new leader. If multiple nodes choose the same contention value, the leader randomly chooses one of them as the new leader. If the current leader leaves the cluster, leaving no device to initiate the contention interval, the devices in the cluster timeout after expecting a connection and each elects itself, causing the entire cluster to consist of leaders. This is remedied by having nodes eliminate each other when two leaders receive each others' beacons. Specifically, if a leader receives a beacon from another leader, the leader with the lower id continues and the other leader immediately stops performing the task. This is necessary to avoid multiple leaders trying to initiate a contention interval simultaneously.

The impact of these behavioral changes is both positive and negative. The positive impacts are that there is a time bound on how long the cluster proceeds without a leader, and further that there will only ever be at most one leader in a cluster, allowing for the short time required for multiple leaders to converge. However, the explicit communication negatively impacts the efficiency of the protocol in that connections can fail requiring them to time out before the next can be established. In the worst case, all connections fail and the leader spends an amount of time proportional to the number of nodes in the cluster multiplied by the timeout value attempting to elect a new leader, only to elect itself again.

Chapter 4: Evaluation

The successful implementation of the algorithm on the Android platform is, in itself, an example of the applicability and effectiveness of the algorithm in that it can be implemented on commodity hardware without significant modifications. Further, clusters ranging in size from two to sixteen devices were successfully able to distribute a critical task among themselves, demonstrating both energy savings and successful distribution of the results of the critical task. Finally, the Android implementation was able to successfully hand-off the critical task across devices making purely local decisions.

To measure the effectiveness of load balancing (i.e., fairness) through the use of the exponentially decaying back-off mechanism, both the theoretical algorithm as well as the Android implementation were tested to investigate any potential effects of the practical application on theoretical fairness. Further, tests were designed specifically to investigate the effect of the contention interval length on fairness, as indicated by the distribution of the leadership role across all nodes in a given cluster. Average latency, defined as the number of contention slots from the start of a contention interval to the contention slot that resulted in leader election, was also measured during testing to further evaluate the impact of contention interval length on efficiency. Several trials were run on cluster sizes of four, eight, and sixteen nodes. Each cluster size was tested with contention interval lengths starting at four slots and

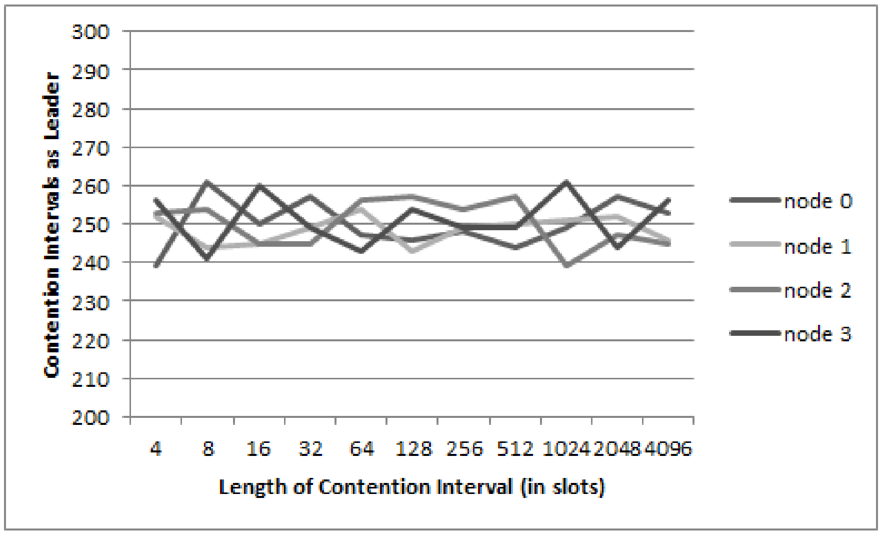

Figure 4.1: The effect of contention interval length on fairness in a cluster of four nodes. Each contention interval length was tested on 1000 consecutive contention intervals.

increasing by powers of two up to 4096 slots. Each cluster was subjected to 1000 consecutive contention intervals at each contention interval length. Figures 4.1, 4.2, and 4.3 summarize these trials. In each figure, each series is a unique node in the cluster. Note that the figures in this thesis are plotted on a log scale due to the contention interval lengths tested.

The leadership role is distributed fairly evenly over all of the nodes in a cluster independent of contention interval length as seen in the figures. It is worth noting that these trials were run against the implementation of the algorithm and not the theoretical algorithm. The significance of this is, as previously mentioned, that the implementation does not allow multiple leaders to be elected by randomly choosing one of the several nodes elected while the theoretical version allows the nodes to proceed as leaders simultaneously. The difference is demonstrated by running the exact same set of trials on an eight node cluster, recording the number of times each node participates in the random tie break instead of which node was chosen to lead. While the random tie breaking mechanism creates fairness, it does so irrespective of the back-off range selection if the length of the contention interval is too small

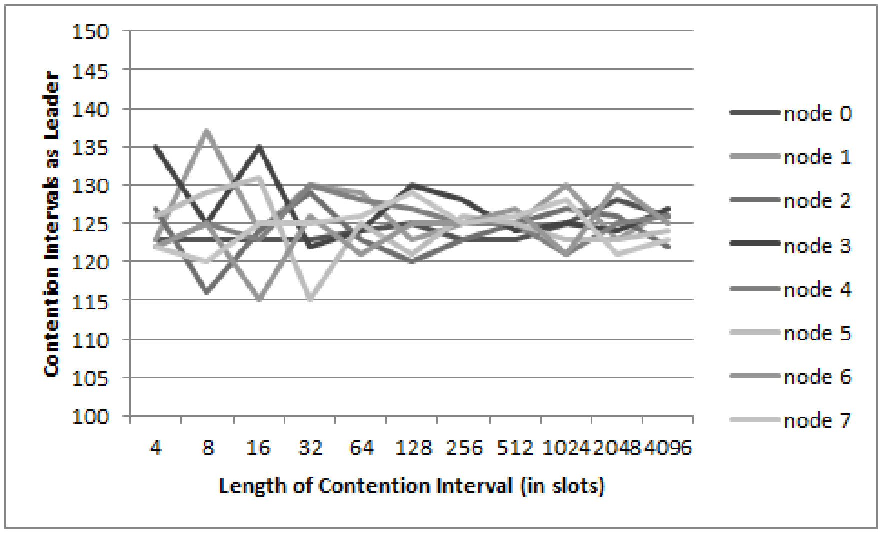

Figure 4.2: The effect of contention interval length on fairness in a cluster of eight nodes. Each contention interval length was tested on 1000 consecutive contention intervals.

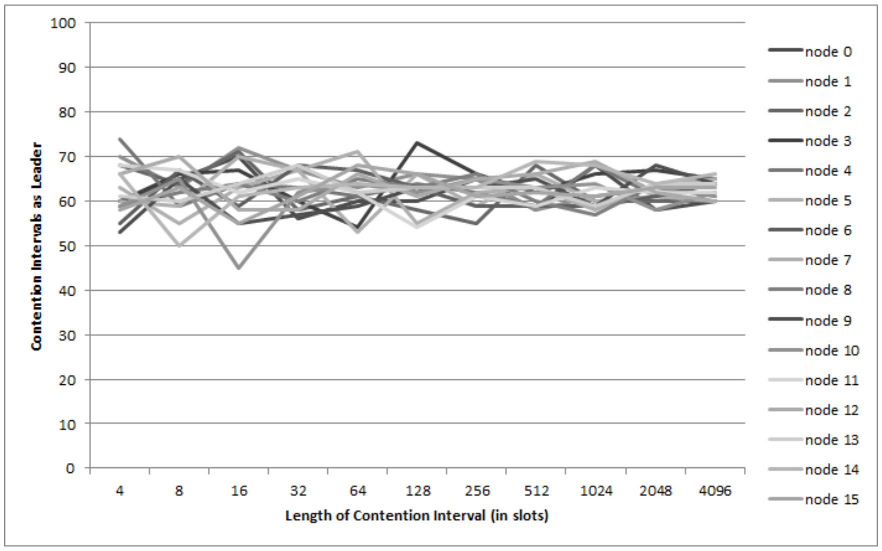

Figure 4.3: The effect of contention interval length on fairness in a cluster of sixteen nodes. Each contention interval length was tested on 1000 consecutive contention intervals.

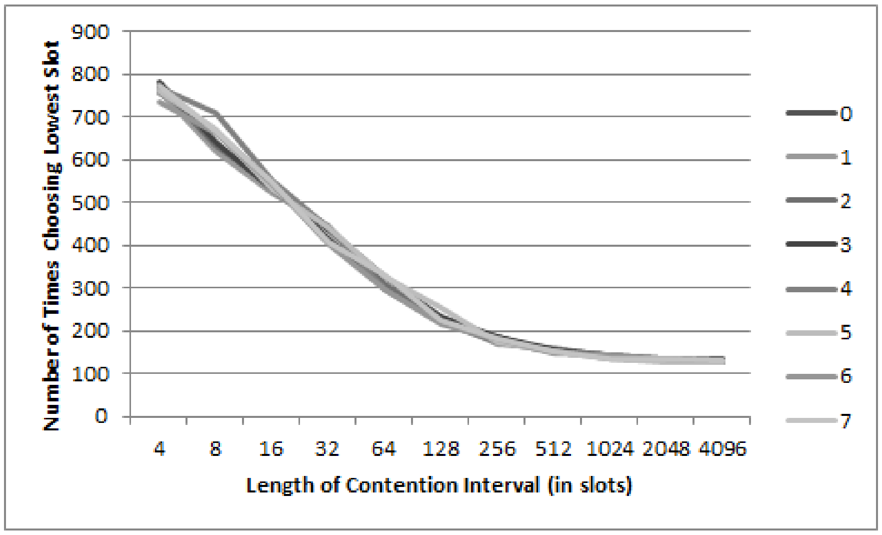

Figure 4.4: The effect of contention interval length on fairness in a cluster of eight nodes, allowing multiple leaders simultaneously. Each contention interval length was tested on 1000 consecutive contention intervals.

for the size of the cluster. This is an expected result, as there are too few contention slots to allow the nodes to spread out and benefit from the back-off range adjustments, resulting in several nodes choosing the lowest slot simultaneously. This is best demonstrated by the contention interval of length four with cluster size eight in that the expected value of nodes per contention slot is two, thus yielding an expected leader count of two per task interval when averaged over all rounds of the trial. These effects can be seen in Figure 4.4. As a final point of interest, although the two versions of the algorithm behave with significant difference under ill conditioned contention interval lengths, both versions achieve fairness levels of approximately 0.999 according to Jain's fairness index. This is intuitive in that the implementation creates artificial fairness by randomly choosing a leader among those eligible while the theoretical version achieves fairness because all nodes are equally likely to be elected as a redundant leader.

Finally, the effect of too large a contention interval is that nodes spend a longer than necessary time waiting for their contention slot to arrive because the nodes are so far spread

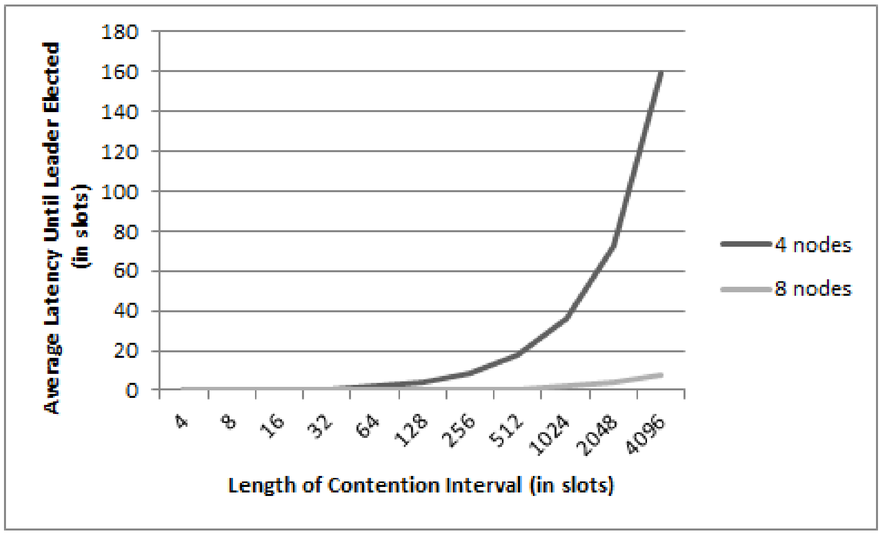

Figure 4.5: Average latency from the start of a contention interval to the election of a new leader on clusters of size four and eight nodes. Each contention interval length was tested on 1000 consecutive contention intervals.

out over the entire window. This is clearly visible in Figure 4.5 with a cluster of four nodes and a contention interval of length 4096, yielding an average of 167 slots until a node is elected leader over 1000 contention intervals. In the Android implementation, this turns out not to matter because the nodes do not actually wait the number of slots chosen because the existing leader selects the next leader in constant time with respect to the number of slots. However, upon further investigation, a cluster of eight nodes has negligible latency up to a contention interval of length 4096. If this is cross referenced with the data from the other figures, it is clear that fairness can be achieved in a cluster of size eight with a contention interval considerably smaller than 4096, eliminating concerns of latency entirely.

Chapter 5: Conclusion

The contributions of this thesis are both the examination of shared-fate tasks with respect to sensory input of mobile devices and an algorithm to distribute and load balance these tasks across clusters of varying size. The implementation and deployment of this algorithm is a further contribution in that it demonstrates its potential for practical application on modern, commodity hardware. It is left to future work to further investigate the nature of these shared-fate tasks as well as the range of application of the algorithm proposed in this thesis. Finally, it is left to future work to investigate the energy savings achieved through use of the algorithm in this thesis.

Bibliography

[1] Khaled M. Alzoubi, Peng-Jun Wan, and Ophir Frieder. Message-optimal connected dominating sets in mobile ad hoc networks. In *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, MobiHoc '02, pages 157–164, New York, NY, USA, 2002. ACM.

[2] Benjie Chen, Kyle Jamieson, Hari Balakrishnan, and Robert Morris. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. In *ACM Wireless Networks Journal*, pages 85–96, 2001.

[3] S. Guha and S. Khuller. Approximation algorithms for connected dominating sets. *Algorithmica*, 20:374–387, 1998. 10.1007/PL00009201.

[4] Navneet Malpani, Jennifer L. Welch, and Nitin Vaidya. Leader election algorithms for mobile ad hoc networks. In *Proceedings of the 4th international workshop on Discrete algorithms and methods for mobile computing and communications*, DIALM '00, pages 96–103, New York, NY, USA, 2000. ACM.

[5] Thyagarajan Nandagopal, Tae-Eun Kim, Xia Gao, and Vaduvur Bharghavan. Achieving mac layer fairness in wireless packet networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, MobiCom '00, pages 87–98, New York, NY, USA, 2000. ACM.

[6] Timucin Ozugur, Mahmoud Naghshineh, Parviz Kermani, C. Michael Olsen, Babak Rezvani, and John A. Copeland. Balanced media access methods for wireless networks. In *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, MobiCom '98, pages 21–32, New York, NY, USA, 1998. ACM.

[7] Vincent D. Park and M. Scott Corson. A highly adaptive distributed routing algorithm for mobile wireless networks, 1997.

[8] Trevor Pering, Yuvraj Agarwal, Rajesh Gupta, and Roy Want. Coolspots: reducing the power consumption of wireless mobile devices with multiple radio interfaces. In *Proceedings of the 4th international conference on Mobile systems, applications and services*, MobiSys '06, pages 220–232, New York, NY, USA, 2006. ACM.

[9] Trevor Pering, Vijay Raghunathan, and Roy Want. Exploiting radio hierarchies for power-efficient wireless device discovery and connection setup. In *Proceedings of the 18th International Conference on VLSI Design held jointly with 4th International Conference on Embedded Systems Design*, VLSID '05, pages 774–779, Washington, DC, USA, 2005. IEEE Computer Society.

[10] Sudarshan Vasudevan, Jim Kurose, and Don Towsley. Design and analysis of a leader election algorithm for mobile ad hoc networks. In *Proceedings of the 12th IEEE International Conference on Network Protocols*, pages 350–360, Washington, DC, USA, 2004. IEEE Computer Society.