A Battery-Aware Algorithm for Supporting Collaborative Applications

Sami Rollins and Cheryl Chang-Yit

University of San Francisco, San Francisco, CA, 94121, USA
{srollins,cchangyit}@cs.usfca.edu

Abstract. Battery-powered devices such as laptops, cell phones, and MP3 players are becoming ubiquitous. There are several significant ways in which the ubiquity of battery-powered technology impacts the field of collaborative computing. First, applications such as collaborative data gathering, become possible. Also, existing applications that depend on collaborating devices to maintain the system infrastructure must be reconsidered. Fundamentally, the problem lies in the fact that collaborative applications often require end-user computing devices to perform tasks that happen in the background and are not directly advantageous to the user. In this work, we seek to better understand how laptop users use the batteries attached to their devices and analyze a battery-aware alternative to Gnutella's ultrapeer selection algorithm. Our algorithm provides insight into how system maintenance tasks can be allocated to battery-powered nodes. The most significant result of our study indicates that a large portion of laptop users can participate in system maintenance without sacrificing any of their battery. These results show great promise for existing collaborative applications as well as new applications, such as collaborative data gathering, that rely upon battery-powered devices.

1 Introduction

Battery-powered devices such as laptops, cell phones, and MP3 players are becoming ubiquitous. One study, conducted in the US, estimates that 73.7 percent of University students own laptop computers and 83.1 percent of 18- to 19-year-olds own an MP3 player [8]. Some Universities even require all incoming freshman to have laptops[21]. With the interest surrounding devices like the iPhone [1] and platforms such as Google's Android [7], this trend is unlikely to reverse.

There are several significant ways in which the ubiquity of battery-powered technology impacts the field of collaborative computing. First, new applications, such as collaborative data gathering, become possible. These applications enable end-user devices, such as phones and laptops, to gather and report information about their surrounding environments [3] and can be used for a variety of purposes including environmental monitoring and entertainment. Additionally, the performance applications that depend on collaborating devices for system maintenance tasks, such as communication, is impacted. Consider an application, such as Skype [18], that relies on collaborators to help connect parties that wish

to engage in a conference. Users of battery-powered devices may be hesitant to provide a service that requires them to devote precious battery, but for which they see no direct benefit. Similarly, users who do allow their battery-powered devices to provide such a service may see a decrease in the performance of the device, for example, its lifetime.

Fundamentally, the problem lies in the fact that collaborative applications often require end-user computing devices to perform tasks that happen in the background and are not directly advantageous to the user. This is in stark contrast to the underlying principle of most battery-saving techniques: enter a low-power state when possible. In this work, we seek to better understand how laptop users use the batteries attached to their devices and explore how these usage patterns can be used to guide how aggressively a device should participate in tasks from which the user does not derive direct benefit.

In particular, we focus our attention on a battery-aware alternative to Gnutella's ultrapeer selection algorithm that determines which collaborating peers are responsible for maintenance tasks such as routing and caching. The goal of our algorithm is to devote system maintenance tasks to devices that have the most stable battery characteristics. Our results are not only applicable to building a Gnutella-style overlay, but also provide insight into how to integrate battery-awareness into applications such as collaborative data gathering.

The remainder of this paper is organized as follows. First, we analyze battery data collected from 41 laptop users over a period of roughly 7 months. Our primary finding is that laptop users are often plugged into a power outlet during periods of active use. Based on this finding, we evaluate a battery-aware modification of Gnutella. Our results show that this algorithm consumes significantly less battery than the standard Gnutella algorithm and still ensures that over half the network can participate in maintenance tasks over 90 percent of the time.

2   Data Collection

The goal of this work is to understand the usage patterns of battery-powered devices and analyze the impact of those patterns on collaborative applications. Our analysis is based on data collected by 41 laptop users during the months of January through August 2007. Unlike previous studies that either collect data at a central collection point, such as a base station [11], or ask users to carry special-purpose devices for data collection [15], we asked the participants to install our data collection software on their personal laptop computers. As a result, we were able to gather a significant amount of information about the real usage patterns of our participants.

Our software is written in Java and runs on both Windows and Mac OSX. It periodically records information about the status of the laptop on which it runs, including the percentage of the battery remaining, whether the device is on AC power, and whether the device is connected to the Internet. Once per day, it uploads the information collected to our data collection server. Our software is completely passive and only collects data during periods when the laptop is already on.

We advertised our software among colleagues and friends, and on several mailing lists. The characteristics of the data are shown in Table 1.

Table 1. Characteristics of the data collected

|	Total Days Participating	% Days Reporting
Minimum	16	.1%
Maximum	216	96%
Mean	190	62%
Median	158	56%

All of our participants downloaded and installed the software at different times, therefore the *Total Days Participating* column indicates the minimum, maximum, mean, and median for the total number of days from the time a participant installed the software until the end of the data collection period (August 8, 2007). The *% Days Reporting* column indicates the ratio of the number of days a participant recorded data to the participant's *Total Days Participating*.

3   Churn and Battery Characteristics

Our analysis considers two elements that impact the ability of battery-powered devices to participate in maintenance tasks to support collaborative applications: the churn rate of the nodes and the battery constraints of the devices. If nodes are only briefly online, the overhead, for example network traffic, required to allocate the appropriate maintenance tasks may be too cumbersome. Further, nodes without sufficient battery are unlikely to want to contribute this valuable resource to tasks that do not directly benefit the user. Our results suggest that laptop users demonstrate rather stable usage patterns: often staying online for over 1 hour and often charging their laptops for the entire duration of their session.

Figure 1 examines the length of the *sessions* observed in our data set. A session begins when a node first reports being on and connected to the Internet and ends when it no longer reports data or when it reports being no longer connected to the Internet. Because we sample the data at 10-minute intervals, there is a small margin of error in our results. If a node records a state of disconnected at 2:00, a state of connected at 2:10, and a state of disconnected at 2:20, we assume that the node was on for 10 minutes from 2:10 to 2:20, though it is possible that the node was on for up to 20 minutes or as little as a few seconds. The figure plots the cumulative distribution function (CDF) of the percentage of sessions of a given length for all sessions observed during the data collection period.

We observe that the sessions in the 25th to 75th percentile are roughly 10 minutes to 2.5 hours in length. Roughly 22 percent of sessions are 10 minutes or shorter, and the longest sessions are well over 5 days in length. While it is clear that allocating system tasks to nodes that are online for only 10 minutes must be avoided, our results demonstrate ample opportunity to identify nodes that can support longer-lived applications.
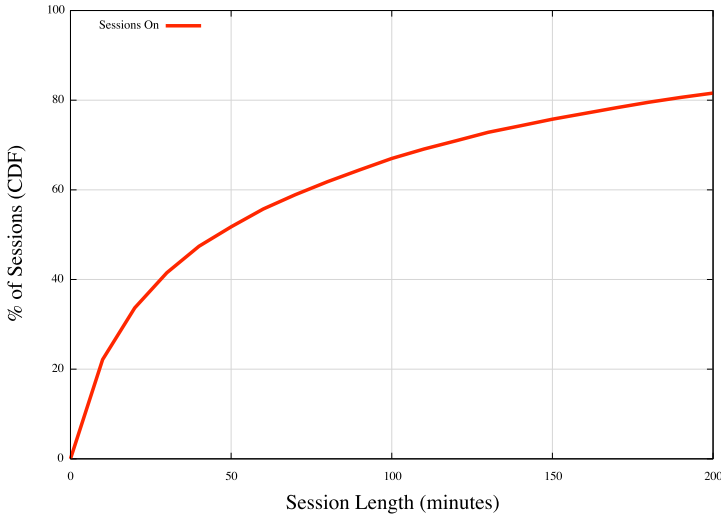
**Fig. 1.** CDF of the length of time a laptop user is typically online

Because our analysis shows that session lengths are longer than 3 hours in only about one-fifth of cases, we suspected that there may be a correlation between session length and laptop battery lifetime. Figure 2 shows this assumption to be false. It plots the CDF of the percentage of nodes online at each 10-minute time sample. Recall that not all of the participants started recording data on the same day. The percentage of nodes online is the number of nodes online at a given time divided by the total number of nodes who have reported data prior to that time. As expected, for a given percentage of nodes (x), the CDF shows that there is a higher probability that x or fewer nodes will be on and plugged in than simply on.

Examining the 25th to 75th percentile, we find that the percentage of nodes online varies from approximately 16% to 34% and the percentage of nodes online and on AC power varies from 14% to 30%. This indicates that only a small percentage of nodes are typically online and *not* plugged in. Our previous work [2] suggests that (1) the devices in our study are rarely on and not online and (2) nearly half of all recharges happen when the battery is at less than 50 percent capacity. This combination of data suggests that most battery is spent when laptops are in a suspended state.

Figure 3 takes a closer look at the number of sessions observed where the node is either on AC power the entire session, not on AC power the entire session, or plugs in or unplugs during the session. There are nearly 3 times as many sessions where a node is on AC power versus not on AC power for the entire session. We also note that further analysis indicated that for the sessions that exhibited a state change, in about half of all cases over 80 percent of the session was spent on AC power. This suggests that there is ample opportunity to design systems that run on battery-powered nodes yet only utilize battery of those nodes that can spare it.
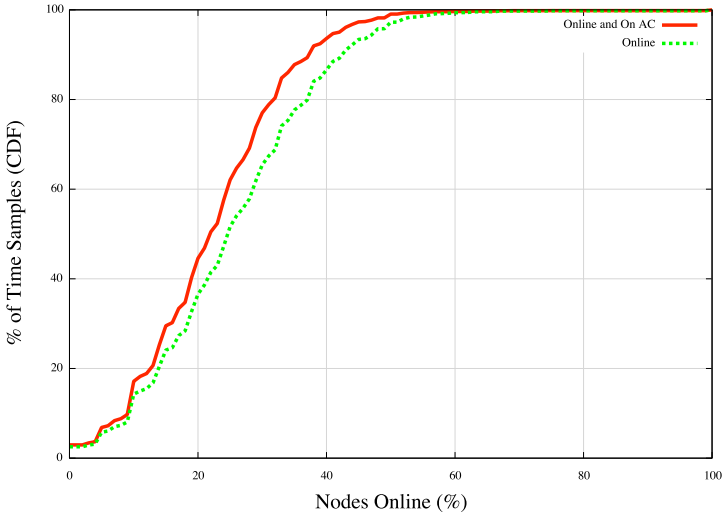
Fig. 2. CDF of the percentage of nodes online and the percentage of nodes online and plugged in at each time sample
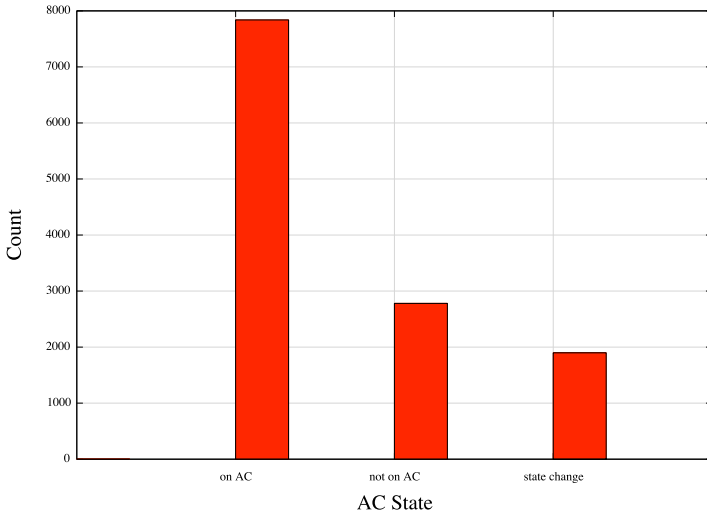
Fig. 3. Count of the number of sessions where a node is (a) plugged in for the entire session, (b) unplugged for the entire session, or (c) plugs in or unplugs at least once during the session

3.1   Discussion

The interesting result of this study is that while the laptops often exhibited what we refer to as *desktop replacement* behavior, for example often running on AC power during times of use, their behavior does differ from a traditional desktop.

Sessions are shorter than would be expected from an always-connected desktop and there are situations where the nodes do drain their batteries during times of use. The question remains, is it prudent to assume that laptops should be exempt from maintaining the infrastructure to support collaborative applications, or can we design the system such that a laptop contributes only a reasonable set of resources? The remainder of this paper addresses this question by evaluating a battery-aware algorithm for building a Gnutella overlay.

4 Battery-Aware Supernode Selection

Resource constraints are a concern for any distributed system in which end-user devices are called upon to contribute to the whole of the system. One technique used by the P2P community is to balance system load by creating a hierarchical network where resource-plentiful nodes are promoted to *ultra* or *super* status [6,14]. Gnutella [13], for example, supports two types of peers: ultrapeers and leaves. Ultrapeers are full participants in the network and must relay messages, perform searches, and cache information. Leaves merely consume the services of the network through ultrapeers. When a node joins the Gnutella network, it is initially designated as a leaf. If a node has the appropriate bandwidth and connectivity characteristics, it will try to promote itself to ultrapeer status after it has been online for at least 1 hour.

In this section, we explore a battery-aware alternative to Gnutella's mechanism for selecting ultrapeers. Our algorithm favors selection of supernodes that are likely to be plugged in, and attempts to avoid choosing supernodes that will be unplugged for any portion of their session. Our results indicate that this algorithm nearly always selects nodes that will remain plugged in for the entire period they serve as a supernode.

In this work, we focus exclusively on the construction of the overlay; application-layer metrics are the subject of future work. In addition, we do *not* assume that our underlying network is a mobile, ad-hoc network. Our data does not provide location information, therefore our nodes are either connected to the Internet and can reach all other connected nodes, or are disconnected from the Internet and unable to participate in overlay construction. Finally, we assume that our nodes meet the bandwidth and connectivity characteristics required to serve as a supernode.

4.1 Experimental Setup

Our evaluation is based on simulated analysis of the data discussed in Section 2. We have developed our own simulator that operates on the following types of events:

- **join a** - A join for node a is generated at the time that a first reports being connected to a network.
- **leave a** - A leave for node a is generated if a fails to report data or reports being disconnected from the Internet.

- **acchange *a*** - An acchange for node *a* is generated if *a* goes from a plugged in to an unplugged status or vice versa.
- **batpercentchange *a*** - A batpercentchange for node *a* is generated if *a*'s percentage of battery remaining changes.
- **superstatuschange *a*** - A superstatuschange for node *a* is generated if *a* promotes itself to be a supernode or demotes itself from supernode status to leaf status. We consider several algorithms for promotion and demotion of supernodes in the following section.

4.2   Supernode Selection Algorithms

Periodically, a node determines whether to promote itself to supernode status or demote itself to leaf status. Our evaluation compares three algorithms: random, uptime, and battery. The random algorithm gives us a baseline for comparison. We consider two uptime algorithms: the standard Gnutella approach and a slightly modified and more aggressive Gnutella-style algorithm. Finally, the goal of the battery algorithm is to choose nodes with the most stable battery characteristics. The intuition is that nodes with higher battery are more likely to exhibit *desktop replacement*-style behavior and will be online and plugged in for significant periods of time; nodes with lower battery, though they may be plugged in, may be less stable.

- **Random** - At five-minute intervals each online node determines randomly, using a uniform distribution, whether to change its supernode status.
- **Uptime** - At five minute intervals each online node will promote itself to supernode status if it has been up (online) longer than a given threshold. Our results compare an aggressive threshold of 5 minutes and the 1-hour threshold used by Gnutella [13]. Once a node has promoted itself, it will remain a supernode until it goes offline.
- **Battery** - At five minute intervals each online node executes the following algorithm:

```
if(!supernode and (on ac power and bat > threshold))
    become supernode
elif(supernode and !(on ac power and bat > threshold))
    become leaf
```

Unless otherwise noted, we use a threshold of 75 percent.

4.3   Results

Our primary goal is to minimize the amount of time a supernode spends unplugged. Figure 4 illustrates the amount of time a supernode spends on battery power. It plots the CDF of the percentage of time a supernode spends in an
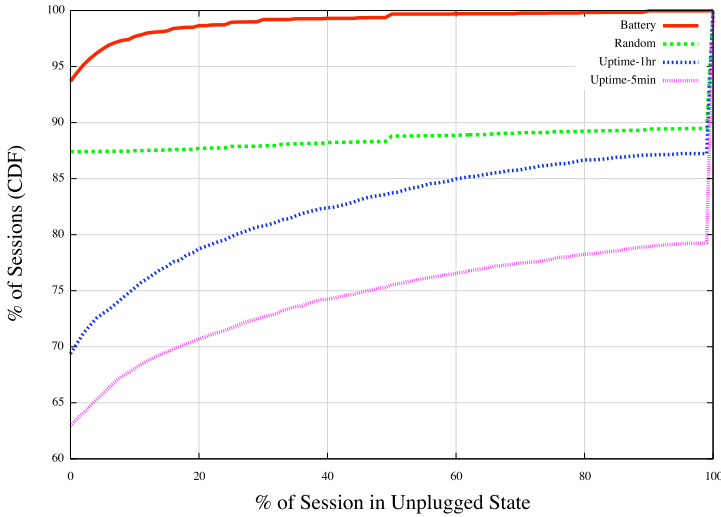
Fig. 4. CDF of the percentage of time that a supernode is not plugged in

unplugged state. Over 62 percent of the time, a node is plugged in during the
entire period that it serves as a supernode. The Battery algorithm significantly
outperforms the others; over 93 percent of the time a supernode is plugged in
during its entire session. Using the Battery algorithm, the only time a supernode
can be unplugged is the 5-minute interval between the periodic checks of battery
status. By reducing the interval to 1-minute, we can actually ensure that a
supernode is never unplugged.

Interestingly, the Uptime algorithm with a threshold of 5 minutes performs
worst; using this algorithm, nearly 22 percent of the time a supernode is un-
plugged for greater than 80 percent of its session. The Uptime algorithm with a
threshold of 1 hour performs much better, but still not as well as Random. In
other words, it is safer to choose randomly from nodes that are online than to
select the node that has been up the longest. Further analysis of the data indi-
cates that the reason for this behavior is that a small number of nodes are online
for long periods of time, but alternate between a plugged in and unplugged state
during that time. The Uptime algorithm favors these nodes, but the Random
algorithm does not select them. Improvements to the Uptime algorithm, such
as enabling a node to demote itself once its battery drops below a certain level,
yield better performance. However, the Uptime algorithm with these battery-
aware improvements still does not match the performance of the straightforward
Battery algorithm.

Though the Battery algorithm is the clear choice for minimizing battery spent
by supernodes, the potential concern is that too few nodes will ultimately pro-
mote themselves and the system maintenance tasks will be neglected. Figure 5
considers the ratio of supernodes to non-supernodes in the network. It plots the
CDF of the percentage of time a given ratio of supernodes is seen in the network.
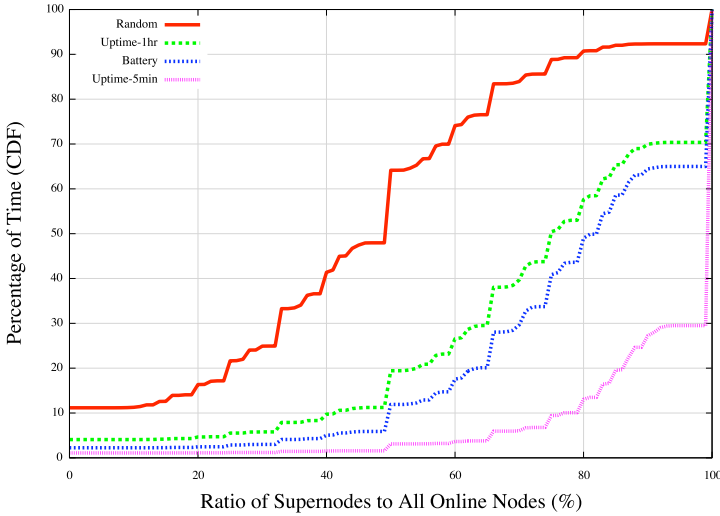
Fig. 5. CDF of the ratio of supernodes to non-supernodes

The Uptime algorithm with a 5-minute threshold performs best. Using this algorithm the probability with which fewer than 80 percent of the online nodes will carry supernode status is just over 10 percent, which means that nearly 80 percent of the network carries supernode status nearly 90 percent of the time. This ratio is a bit extreme; ideally, the number of supernodes should be just large enough to handle the burden of maintaining the network.

In the case of the Battery algorithm, only a small percentage of the time, less than 3 percent, are there no supernodes in the network. In fact, over 90 percent of the time there is at least a 1-to-1 ratio of supernodes to non-supernodes. This demonstrates that, even using a battery-conservative algorithm, a significant number of laptops can contribute to the system at low cost.

We also note that we ran our experiments using a Battery algorithm with a threshold of 90 percent and a Battery algorithm that used no threshold— it simply promoted a node when it was plugged in and demoted a node if it unplugged. The only metric where the threshold had a noticeable impact was the ratio of supernodes metric. The 90 percent algorithm performed slightly worse and the no threshold algorithm performed slightly better. This suggests that it may be useful to use threshold as a parameter to tune the supernode ratio to meet the demands of a given application.

Our final experiment considers the impact of the supernode selection algorithm on the churn rate of the network. Figure 6 illustrates the length of the time a node typically serves as supernode. It plots the CDF of the length of time between the promotion of a supernode and the demotion of the same supernode. The Uptime algorithm with a threshold of 1 hour performs best; roughly 35 percent of the sessions are shorter than an hour. In the Battery case, only 55
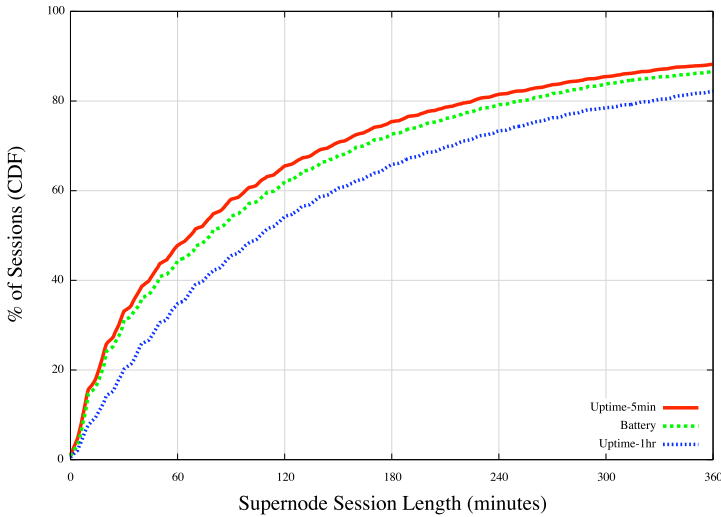
Fig. 6. CDF of the length of the session for a supernode

percent of sessions exceed the 1-hour mark. As we observed in Figure 4, the behavior of the Uptime algorithm is such that a few long-running nodes are selected as supernodes, but are unplugged and plugged back in several times during the session. Therefore, the churn rate is lower, but the battery used during the session is potentially detrimental to the node. Further, the improved churn rate may not be sufficiently significant to warrant the risk.

We investigated further to determine the percentage of the battery remaining at recharge for the long-running supernodes. For each node that transitioned from an unplugged to plugged-in state more than once during a single session, we record the percentage of the battery remaining each time the laptop is plugged in. In some cases, the battery gets as low as 3 percent, with a mean of 71 percent and a median of 80 percent. Though those nodes that only unplug briefly would presumably not be impacted by any additional battery devoted to overlay maintenance, there are a number of cases where additional battery usage would likely impact the user. The conservative, battery-aware approach yields a slightly higher churn rate, but ensures that the battery life of the device is not impacted and produces acceptable supernode-to-non-supernode ratios.

4.4   Additional Observations

We also considered an algorithm that uses *global knowledge* of the battery and uptime characteristics of all online nodes to select supernodes such that a ratio of 1 supernode to 4 non-supernodes is maintained. The intuition behind this algorithm is that the *local-knowledge* Uptime and Battery algorithms described above select more supernodes than necessary (see Figure 5). By maintaining a fixed ratio, fewer nodes will be promoted to supernode status and, as a result, fewer nodes will spend battery on supernode duties.
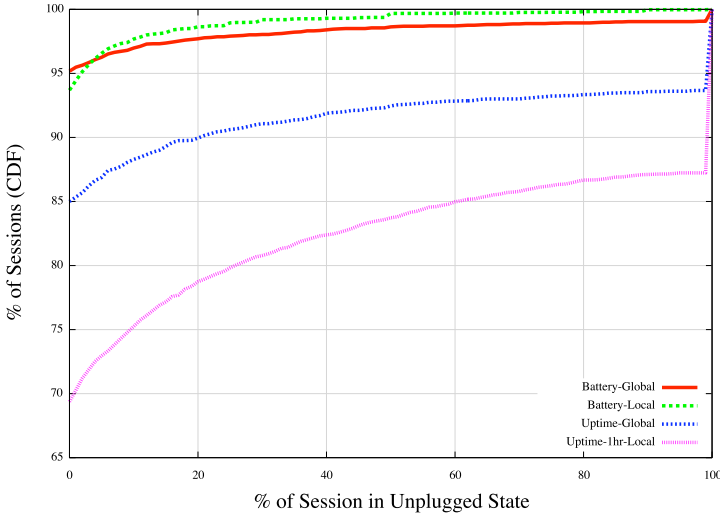
Fig. 7. CDF of the percentage of time that a supernode is not plugged in using a global-knowledge algorithm

To elect a new supernode, the global-knowledge algorithm chooses the online node with the highest battery, for the Battery algorithm, or longest uptime, for the Uptime algorithm. Once a node becomes a supernode, it remains so until it goes offline. Clearly, this algorithm requires more overhead since global state must be collected before any change to the network topology.

Figure 7 considers the amount of time a supernode spends on battery power. It plots the CDF of the percentage of time a supernode spends in an unplugged state for both the global-knowledge and local-knowledge Battery and Uptime algorithms. Despite the increased complexity of the global algorithm, the local Battery algorithm performs best overall. The global Uptime algorithm does perform significantly better than the local Uptime algorithm, because the ratio of supernodes to non-supernodes is lower. However, it is clearly advantageous to integrate battery characteristics into the supernode selection algorithm.

4.5   Discussion

The most significant result of our study is that a large portion of the laptop users we have observed can participate in P2P overlay maintenance and construction without sacrificing any of their battery. Additionally, the top-performing battery-aware algorithm is a straightforward algorithm that uses local knowledge to determine whether a node should choose to promote itself to supernode status. These results demonstrate significant potential to support collaborative applications in mobile environments where the participating nodes are also responsible for the maintenance of the system.

This work introduces a number of additional questions we would like to further explore. First, our current data set does not enable us to quantify the negative

impact of promoting to supernode status those nodes which are running on battery. In other words, we cannot quantify how much battery a node would devote to its supernode duties, nor can we determine the impact on the overall lifetime of the device. Our results indicate that even our extremely conservative approach performs well. However, we plan to explore less conservative algorithms that *predict* how much battery a node will have to devote to the system as a whole, for example using prediction strategies similar to those proposed by Mickens and Noble [16].

We also plan to evaluate the performance of our strategy in the context of a real application. In particular, we would like to implement our algorithm in the context of a collaborative data gathering application that enables mobile nodes to share information they have witnessed with other interested parties.

5 Related Work

This work is the first, to our knowledge, that uses real traces to analyze a battery-aware algorithm for building a P2P infrastructure to support collaborative applications. Our primary goal is to bridge the gap between studies of battery usage and studies that use purely synthetic data to analyze the performance of P2P algorithms for mobile environments.

The goal of the Hybrid Chord Protocol (HCP) [22] is to enable resource-constrained nodes to participate in P2P networks. Their approach is to build a structured P2P network that differentiates between *static* and *temporary* nodes. Resource-plentiful static nodes then take on the bulk of the network maintenance overhead. Unlike our work, this work does not consider battery as a metric for differentiating between nodes. Our analysis, based on real data, provides additional insight into how battery impacts the behavior of *temporary* nodes. An interesting extension of our work would be to investigate how our battery-aware metrics could be integrated into HCP.

Approaches such as Ekta [17] and the approach proposed by Landsiedel, Gotz, and Wehrle [12] are also related to our work. Both projects address the problems associated with building distributed hash tables in mobile ad-hoc networks, but neither directly addresses the challenge of dealing with the limited battery lifetime of the mobile nodes. The body of work on sensor networks and disruption tolerant networks (DTNs) is also related to our work, and much of this work focuses on energy constraints. For example Jun et al. [9] consider power management in DTNs and Span [4] proposes an energy-efficient algorithm for overlay maintenance in ad hoc wireless networks. However, these approaches are exclusively designed for a MANET environment. We take a different view by considering the real usage patterns of laptop users.

There is a broad body of work that considers overlay maintenance and supernode selection for P2P applications. Lo et al. [14], compare several algorithms for scalably selecting supernodes in P2P networks and Garces-Erice et al. [6] investigate hierarchical DHTs. However, this work does not consider performance in the context of real usage patterns. Stutzbach and Rejaie [20] consider the impact

of churn by analyzing usage data for several existing P2P networks. However, this work, as well as the work of Lo et al., is targeted toward non-battery powered, standard desktop computers. It does not consider how to integrate battery as a metric for overlay maintenance.

Several studies have attempted to capture usage patterns of mobile users. Kim and Kotz [10] investigate ways in which to model user mobility based on traces of user visits to a set of access points. Similarly, Song and Kotz [19] use realistic traces to simulate several opportunistic routing algorithms. In both of these cases, the goal is to investigate mobility patterns of users, not battery usage patterns. In fact, the trace data collected for the studies is collected via an access point and does not contain battery usage data.

There have been a few studies that specifically collect traces of battery usage. The study conducted by McNett and Voelker [15] collects mobility and battery data from a set of PDA devices given to a class of college freshmen. However, the results of the study again focus on mobility of the users, not their battery usage. MyExperience [5] more carefully considers battery usage of mobile phone users, but does not address the impact with respect to building systems.

6 Conclusion

In this work, we investigate the impact of battery life on the infrastructure to support collaborative applications. We identify the unique characteristics of laptop users by analyzing data collected by 41 users over a period of 7 months. Our results suggest that laptop users often behave similar to desktop users—staying online and plugged in to a power outlet for long periods of time—but also demonstrate periods of running on battery power or staying online for short stretches. We then evaluate a conservative, battery-aware alternative to the Gnutella ultrapeer selection algorithm. The key element of our approach is the selection of battery-plentiful supernodes that can bear the brunt of the system maintenance and overhead. Our results indicate that, using this algorithm, battery-powered nodes sacrifice little or none of their limited battery resources for the sake of overlay construction and maintenance. These results show great promise for existing collaborative applications as well as new applications, such as collaborative data gathering, that rely upon battery-powered devices.

Acknowledgments

References

1. Apple, `http://www.apple.com/iphone/`
2. Banerjee, N., Rahmati, A., Corner, M.D., Rollins, S., Zhong, L.: Users and batteries: Interactions and adaptive energy management in mobile systems. In: Krumm, J., Abowd, G.D., Seneviratne, A., Strang, T. (eds.) UbiComp 2007. LNCS, vol. 4717, pp. 217–234. Springer, Heidelberg (2007)
3. Burke, J., Estrin, D., Hansen, M., Parker, A., Ramanathan, N., Reddy, S., Srivastava, M.B.: Paticipatory sensing. In: World Sensor Web Workshop, Boulder, Colorado, USA (October 2006)
4. Chen, B., Jamieson, K., Balakrishnan, H., Morris, R.: Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. Wirel. Netw. 8(5), 481–494 (2002)
5. Froehlich, J., Chen, M.: MyExperience: A system for in situ tracing and capturing of user feedback on mobile phones. In: Proc. Int. Conf. Mobile Systems, Applications, and Services (MobiSys) (June 2007)
6. Garces-Erce, L., Biersack, E., Felber, P., Ross, K.W., Urvoy-Keller, G.: Hierarchical peer-to-peer systems. In: Euro-Par 2003, Klagenfurt, Austria (2003)
7. Google, `http://code.google.com/android/`
8. Inside Higher Ed. Students 'evolving' use of technology (September 2007), `http://www.insidehighered.com/news/2007/09/17/it`
9. Jun, H., Zhao, W., Ammar, M.H., Zegura, E.W., Lee, C.: Trading latency for energy in densely deployed wireless ad hoc networks using message ferrying. Ad Hoc Netw., 5(4), 444–461 (2007)
10. Kim, M., Kotz, D.: Modeling users mobility among wifi access points. In: WiT-MeMo 2005: Papers presented at the 2005 workshop on Wireless traffic measurements and modeling, Berkeley, CA, USA, pp. 19–24. USENIX Association (2005)
11. Kotz, D., Essien, K.: Analysis of a campus-wide wireless network. Wirel. Netw. 11(1-2), 115–133 (2005)
12. Landsiedel, O., Götz, S., Wehrle, K.: A churn and mobility resistant approach for dhts. In: MobiShare 2006: Proceedings of the 1st international workshop on Decentralized resource sharing in mobile computing and networking, pp. 42–47. ACM, New York (2006)
13. Limewire, `http://wiki.limewire.org/index.php?title=Ultrapeers`
14. Lo, V., Zhou, D., Liu, Y., GauthierDickey, C., Li, J.: Scalable supernode selection in peer-to-peer overlay networks. In: HOT-P2P 2005: Proceedings of the Second International Workshop on Hot Topics in Peer-to-Peer Systems, Washington, DC, USA, pp. 18–27. IEEE Computer Society, Los Alamitos (2005)
15. McNett, M., Voelker, G.M.: Access and mobility of wireless pda users. SIGMOBILE Mob. Comput. Commun. Rev. 9(2), 40–55 (2005)
16. Mickens, J.W., Noble, B.D.: Exploiting availability prediction in distributed systems. In: NSDI 2006: Proceedings of the 3rd conference on 3rd Symposium on Networked Systems Design & Implementation, Berkeley, CA, USA, P.6. USENIX Association (2006)
17. Pucha, H., Das, S.M., Hu, Y.C.: Ekta: An efficient dht substrate for distributed applications in mobile ad hoc networks. In: WMCSA 2004: Proceedings of the Sixth IEEE Workshop on Mobile Computing Systems and Applications, Washington, DC, USA, pp. 163–173. IEEE Computer Society, Los Alamitos (2004)

18. Skype, `http://www.skype.com/`
19. Song, L., Kotz, D.F.: Evaluating opportunistic routing protocols with large realistic contact traces. In: CHANTS 2007: Proceedings of the second workshop on Challenged networks CHANTS, pp. 35–42. ACM, New York (2007)
20. Stutzbach, D., Rejaie, R.: Understanding churn in peer-to-peer networks. In: IMC 2006, Rio de Janeiro, Brazil (October 2006)
21. University of Oregon, `http://engr.oregonstate.edu/students/wireless/`
22. Zoels, S., Schubert, S., Kellerer, W., Despotovic, Z.: Hybrid DHT design for mobile environments. In: Joseph, S., Despotovic, Z., Moro, G., Bergamaschi, S. (eds.) AP2PC 2006. LNCS, vol. 4461, pp. 19–30. Springer, Heidelberg (2008)