Increasing Energy Efficiency in Mobile Peer Networks by Exploiting Traffic Sampling Techniques

Julian K. Buhagiar
Dept of Communications and Computer Engineering
University of Malta
Msida, MSD 2080, MALTA
julian.buhagiar@ieee.org

Carl J. Debono
Dept of Communications and Computer Engineering
University of Malta
Msida, MSD 2080, MALTA
c.debono@ieee.org

***Abstract*—Wireless infrastructures have seen a drastic increase in energy requirements as technology has shifted towards higher allocated frequencies in an attempt to provide more bandwidth to accommodate more users and provide the resources for more demanding applications and services, such as video streaming and multimedia applications. Driven by the increase in demand from an always increasing subscriber base, large cities are also forcing wireless service operators to install more base stations and access points to sustain an adequate quality of service. Mobile peer networking offers a possible solution for the later with the added benefit of providing power-efficient communication, since transmission over short distances demands less transmission power using appropriate peer connectivity algorithms [1]. Thus, substantial energy can be saved, given that the power amplifier that lies in the transmitter circuitry is the most power hungry device of each mobile node. A novel algorithm based on peer nodal hierarchies, traffic mapping, and neural networks is proposed. This algorithm invokes the use of a traffic sampling matrix to optimise delivery and routing of peer node information, allowing for more efficient power distribution. Additional optimisation is provided through a state-switching algorithm that exploits the traffic sampling algorithm to switch to a more energy-efficient state algorithm when residual neighbouring resources are available. Results show that this technique presents a remarkable power efficiency improvement over standard peer-to-peer networks.**

Keywords-energy saving, mobile peer networking, power consumption, state-switching, traffic sampling

I. INTRODUCTION

The ever increasing cost of energy is pushing network operators and designers to develop solutions for more power-efficient mobile networks. With all the different wireless technologies deployed today, mobile peer-to-peer (MP2P) networks offer a possible solution to curb energy consumption by reducing the transmission distance between nodes and hence the power requirements. MP2P networks can also provide efficient distribution and delivery of high-bandwidth services such as media downloading, making them a more attractive solution. Current mobile peer network implementations are based on well established fixed-network topologies, implying that existing peer protocols are overlaid on the existing mobile networks, thus inheriting the unique challenges present in the mobile environment, such as bandwidth asymmetry and node availability [2].

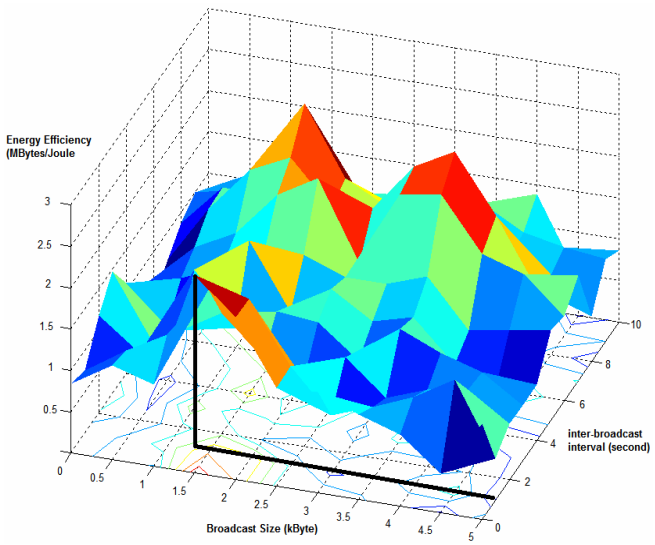

Figure 1. Energy Efficiency (in MB/Joule) for a MP2P system in node discovery mode. Assuming a mobile peer density λ of $6.25 \times 10^{-4} m^2$, for a mobile inter-broadcast interval of 1s, optimal energy efficiency is attained with a broadcast size of 1.35kB.

Inefficient use of this overlay network causes significant mobile power dissipation as the mobile peer clients enter/exit the network sporadically, and asynchronously connect with the network seeds. Optimal MP2P energy efficiency can only be derived through iterative calculations for each peer node. These involve parameter calculations such as broadcast size and inter-broadcast intervals, resulting in energy efficiency curves such as the one shown in Figure 1 [3]. Such algorithms are themselves computationally intensive, leading to a trade-off between energy efficiency and computational speed [4]. Moreover, the optimal solution changes with time as the wireless channel characteristics change and the nodes move in the network. Therefore, a solution that adapts itself with time is essential.

A promising solution is to apply enhanced traffic sampling algorithms that optimise the delivery and routing of the peer client information, thereby contributing to more efficient power dissipation. This paper presents a novel approach that exploits traffic sampling to optimise the energy profile in MP2P networks. Successful implementation of this technology can be deployed in mobile peer networking

scenarios where efficient peer node resource information routing is required, such as TCP congestion avoidance [2].

The proposed peer network solution also provides rapid information delivery based on node routing and traffic history techniques. A neural network-based algorithm is used to optimise the node routing within the peer network from traffic reports generated by the mobile clients within the network. The algorithm sets the routing paths based on the traffic requests and the busy hour traffic allocations. History is used as a starting point for the optimisation process since traffic profiles tend to be statistically similar across weeks for similar traffic payloads. Through this information, the algorithm guarantees the most efficient resource allocation on the mobile radio interface, thereby yielding an optimal power profile.

Additional energy optimisation is provided through efficient use of the observed traffic reports to influence the state-switching of each peer node. By periodic measurement of the neighbouring nodes, the band state of each mobile node can be changed to a more efficient state, depending on the amount of residual resources the node has.

The rest of the paper is organised as follows: Section II gives an overview of peer networking. Section III describes the algorithm developed followed by testing and results in Section IV. Finally, Section V provides some comments and conclusions.

II. MOBILE P2P ENERGY CONSUMPTION

A. Mobile P2P Environment Challenges

Mobile peer networking is different from the traditional wired peer-to-peer networks in the following ways:

i) Availability – the mobile user moves within the network and can therefore be out of range, switched off, or under a profile which discourages peer networking (example: roaming, non-flat rate data tariff).

ii) Connectivity – the mobile user's ping times can vary significantly throughout the networking session as the system switches to a slower packet access network to maintain the service or the current location area is an inefficiently subnet, resulting in long hop times.

iii) Topology – the peer nodes are physically mobile, and sometimes can be highly mobile [6]. The traffic density can vary significantly and consequently the underlying communication network is subject to frequent topology changes and disconnections. P2P network approaches that require pre-defined data access structures such as search routing tables [5] and spanning trees [6] are impractical in the wireless environment.

iv) Throughput – The communication throughput between two peers is constrained by the wireless bandwidth available, the channel contention, and the limited connection time. Therefore, selective communication is necessary such that the most important data is communicated.

v) Cooperation – Like many other P2P systems or mobile ad-hoc networks, the ultimate success of mobile P2P networks relies heavily on the cooperation among users. In infrastructure P2P systems, incentive is provided for peers to participate as suppliers of data, compute cycles, knowledge/expertise, and other resources. In mobile ad-hoc networks, incentive is provided for mobile hosts to participate as intermediaries/routers. In mobile P2P networks, the incentive has to be provided for participation as both suppliers and intermediaries (namely brokers) [7].

B. Peer Node Reporting

In modern peer networks, such as BitTorrent [8] there are two main methods for node reporting to/from peer nodes, namely report pulling and report pushing. Report pulling implies that when a mobile peer makes an explicit request, the peer network is flooded with queries and the specific report is retrieved from the mobile peers that contain the specific report data for that particular node. Report pulling is widely used in resource discovery, route discovery in mobile ad hoc networks, and file discovery by query flooding in wired P2P networks like Gnutella [9]. Flooding in a wireless network is in fact relatively efficient compared to wired networks because of the advantages derived using wireless multicast [10].

The alternative approach for peer reporting is through pushing. In this method, reports are flooded, and consumed by peers whose query is answered by received reports. To date mechanisms exist to broadcast information in:

(i) the entire peer network

(ii) in a specific geographic area (geocast)

(iii) to any one specific mobile node (unicast/mobile ad-hoc routing)

(iv) any one arbitrary node (anycast).

Additionally, report pushing methods can be further divided into stateful methods and stateless methods. Most stateful methods are topology-based, i.e. they impose a structure of links in the network, and maintain states by data dissemination.

One specific group of stateful methods contains cluster- or hierarchy-based methods [11], in which moving peers are grouped into some clusters or hierarchies and the cluster heads are randomly selected. Reports are disseminated through the network in a cluster or hierarchy manner, which means that reports are first disseminated to every cluster head and each cluster head then broadcasts the reports to the member peers in its group.

Although cluster- or hierarchy-based methods can minimise the energy dissipation in moving peers, these methods will fail or dissipate more energy in highly mobile environments as they have to maintain a hierarchy structure and frequently need to reselect the cluster heads. Each time cluster heads are elected, energy is lost in the overhead needed in passing necessary information and in the computation of the clustering algorithm.

Another stateful algorithm utilises Location-based methods. In this method, each moving peer knows the location of itself and its neighbours through some localisation technique, such as GPS or Atomic Multilateration [12].

The simplest location-based data dissemination is Greedy Forwarding, in which each moving peer transmits a report to a neighbour that is closer to the destination than itself.

However, Greedy Forwarding can fail in some cases, such as when a report is stuck in local minima, which means that the report stays in a mobile peer whose neighbours are all further from the destination. Therefore, some recovery strategies are required, such as GPSR (Greedy Perimeter Stateless Routing [13]). Other location-based methods, such as GAF (Geographic Adaptive Fidelity [14]) and GEAR (Geographical and Energy Aware Routing [15]), take advantage of knowledge about both location and energy to disseminate information and resources more efficiently.

In stateless methods, the most basic and simplest solution is through flooding techniques, such as [16]. In flooding-based methods, mobile peers simply propagate received reports to all neighbouring mobile peers until the destination or a maximum number of hops is reached. Each report is propagated as soon as is received. Flooding-based methods have many advantages, such as no state maintenance, no route discovery, and easy deployment. However, they inherently cannot overcome several problems, such as implosion, overlap, and resource blindness. Therefore, other stateless methods are proposed, such as gossiping-based methods and negotiation-based methods.

Gossiping-based methods, such as [4], improve flooding-based methods by transmitting received reports to a randomly selected neighbour or to the neighbours that are interested in the particular content. The advantages of these methods include reducing the implosion and lowering the system overhead. However, the cost of determining the particular interests of each moving peer can be huge and transmitting reports to a randomly selected neighbour can still cause the implosion problem and waste peers' memory, bandwidth and energy. Furthermore, dissemination, and thus performance, is reduced compared to pure flooding.

Negotiation-based methods solve the implosion and overlap problem by transmitting first the id's of reports; the reports themselves are transmitted only when requested [17]. Thus, some extra data transmission is involved, which costs more memory, bandwidth, and energy. In addition, in negotiation-based methods, moving peers have to generate meta-data or a signature for every report so that negotiation can be carried out, which will increase the system overhead and decrease the efficiency.

C. Energy Consumption Model

Before participating in peer node reporting, each MP2P client specifies the energy constraint using the algorithm: "*from time(t) until time(H) the MP2P system is allowed to use fraction F of the remaining energy*". The allocated energy covers all the energy consumed by report dissemination, including the energy used for transmission, receiving, listening, and computation. If F is the energy allocation fraction, and Ω Joules of energy is left in the node, this constraint is translated into the following specification: "*The algorithm may use no more than $\Omega \cdot F$ Joules until time H*".

The pair ($\Omega \cdot F$, H) is thus the energy budget. Therefore, the lifetime demand of each individual device is accommodated [27].

Now we introduce the energy consumption model. Let the size of a message be M bytes excluding the MAC header. According to [26], the energy consumed for transmitting a message can be described using the linear equation:

$$E_n = f * M + g \quad (1)$$

Intuitively, there is a fixed component associated with the network interface state changes and channel acquisition overhead, and an incremental component which is the size of the message. Experimental results confirm the accuracy of the linear model and are used to determine values for the linear coefficients g and f. For 802.11 broadcast systems, g=266×10^{-6} Joule, and f=5.27×10^{-6} Joule/byte ([27]).

D. Optimal Transmission Size

Consider a broadcast of M bytes by a mobile device x. If another neighbour of destination node y transmits during some time slot of a broadcast, then a collision will occur, and the whole broadcast is considered corrupt (i.e. unsuccessfully received) at y. If N is the number of neighbours that successfully receive the message from node x, the throughput of the broadcast sent by x, denoted T_h, is defined as:

$$T_h = M * N \quad (2)$$

Intuitively, the throughput is the total amount of data successfully received by the neighbours of x. If E_n is the energy consumed at the network interface of x for sending the broadcast message, the energy efficiency of the broadcast by x, denoted P_{eff}, is defined as:

$$P_{eff} = \frac{T_h}{E_n} \quad (3)$$

In other words, the energy efficiency is the throughput produced by each unit of transmission energy consumed at node x. To compute the value of T_h, consider a mobile device x broadcasting a message at an arbitrary time slot. According to [26], the expected value of T_h can be approximated by:

$$E(Th) \approx 2\pi\lambda M \int_0^r \delta (1-p')^{\lambda r^2 (2q(\frac{\delta}{2r}) + (\pi - 2q(\frac{d}{2r})).(2T+1)) - 1} d\delta \quad (4)$$

where:

- λ is the number of devices per unit in area
- r is the transmission range of each device in (m)
- b is the data transmission speed in (bits per second)
- p' is the probability that a device starts a broadcast
- τ is the length of the MAC timeslot in (s)
- h is the size of the MAC header in bytes
- c is the time since completion of last broadcast (s)

and:

$$T = \tau(M + h).8 / b$$

$$q(a) = \arccos(a) - a\sqrt{1 - a^2}$$

Equation (4) takes into account the effect of hidden terminals as well as direct collisions. By the definition of energy efficiency, the expected value of the energy efficiency is:

$$E(P_{eff}) \approx 2\pi\lambda M$$

$$\frac{\int_0^r \delta(1-p')^{\lambda r^2 (2q(\frac{\delta}{2r})+(\pi-2q(\frac{d}{2r})).(2T+1))-1} d\delta}{f(M+g)} \qquad (5)$$

From equation (5), if τ, p', λ, h, b, r, f and g are fixed, then the energy efficiency P_{eff} as a function of the broadcast size M is a bell shaped curve. Thus, there is a value of M that maximises the energy efficiency, i.e. achieves the best trade off between the channel utilisation and broadcast reliability.

For the rest of this subsection we show that indeed, except for M, all the parameters of equation (5) can be determined by the mobile device. The parameters τ, h, r, b depend on the network, and are fixed for a given communication network technology. For example, h is equal to 47 in the wireless access protocol 802.11b [2]. f and g depend on the network interface hardware and can be calibrated as demonstrated by [25]. The density λ can be estimated based on the average number of neighbours over time (recall that each mobile device knows its neighbours via the neighbour discovery protocol), given the transmission range.

The probability p' is determined as follows. Let c be the number of seconds since the completion of the last broadcast of x until the time when the current broadcast size is to be determined. If, on average, a mobile device starts a broadcast of data every c seconds, then its probability of starting a broadcast in each medium access time slot is τ / c. Therefore, the broadcast probability p' is substituted in equation (5) by τ / c. For instance, if c = 5 seconds and τ = 20μs, then

$$p' = (20\times10^{-6})/5 = 4\times10^{-6}$$

Using $p' = \tau / c$, a formula is derived for the throughput in which the only unknown parameter is M. In some scenarios the actual p' may be lower than τ / c. This occurs because there is a delay from the time when the broadcast is triggered until the channel is sensed free and when the broadcast is actually started. In other words, the actual broadcast period may be larger than c. However, since the difference between p and p' is small, this delay is expected to be small as well and therefore can be neglected. Thus, the optimal value for M can be found, i.e. the value of M for which P_{eff} is maximised. This value is denoted by $M_{optimal}$, and can be outlined in Figure 1.

E. Proposed Solution

One method of determining P_{eff} of a broadcast is by measuring the energy efficiency, which can be defined as:

$$E_n = \frac{\sum(D_n)}{T} \qquad (6)$$

where D is the amount of mobile peer data that is correctly received and T is the unit of peer transmission energy. The data throughput received by a mobile peer node n can be expressed as a product of the neighbours that successfully receive the broadcast (N), and the size of the broadcast b:

$$T_h = Nb \qquad (7)$$

Consequently the power efficiency can be expressed as shown in equation (3). Alternatively P_{eff} can be expressed as the trade-off between the inter-broadcast interval t_b and the broadcast size b [3], as can be seen in Figure 1.

The main challenge to maximise P_{eff} is to determine which nodes are more connectable than others, and the relative connection times associated with each. This requires the construction of a reliability algorithm which builds up a table of the client nodes within the peer network with the following criteria:

(i) *availability a* in % over time window t_1,
(ii) *connectivity c* in ms over time window t_2, and
(iii) *transfer rate r* in kbps over time window t_3.

As this table is updated periodically, a list of the nodes whose performance indicators are most favourable over time t_1, t_2, and t_3 will gradually be upgraded to super-nodes. The challenge is to provide an efficient scalable traffic analysis algorithm that is susceptible to rapid variations in the mobile environment.

III. TRAFFIC SAMPLING

A. Traffic Analysis

Determining the metrics a, c and r over time t gives a reliability indication which helps build a traffic profile for each node. The observed traffic for each node i can be defined as:

$$T_i^{obs} = [a_i \quad c_i \quad r_i] \qquad (8)$$

Most wireless network operators specify their network-wide goals in terms of Origin-Destination (OD) pairs [9]. To achieve flow monitoring goals which are specified in terms of OD-pairs, the optimisation engine needs the traffic matrix and the routing information, both of which are readily available in the network [18]. The required matrix is obtained through the traffic sampling architecture outlined in Figure 2.

To handle the traffic dynamics the following heuristic approach is used in this work. Assume that the sampling manifests for every 5-minute interval for the Fri. 9am-10am period of the current week is required. To avoid over fitting, the hourly traffic matrix for the previous week's Fri. 9am-10am period is employed after being divided by 12 (to obtain the required 5-minute interval). The resulting traffic matrix T^{old} is used as input data to compute the manifests for the first 5-minute period. At the end of this period, the flow of data from each node is collected, and the traffic matrix T^{obs} from the collected flow reports is obtained. If the fractional coverage for OD_i with the current sampling strategy is C_i and x_i sampled flows are reported, then $T_i^{obs} = x_i/C_i$. That is, the number of sampled flows are normalized by the total flow sampling rate.

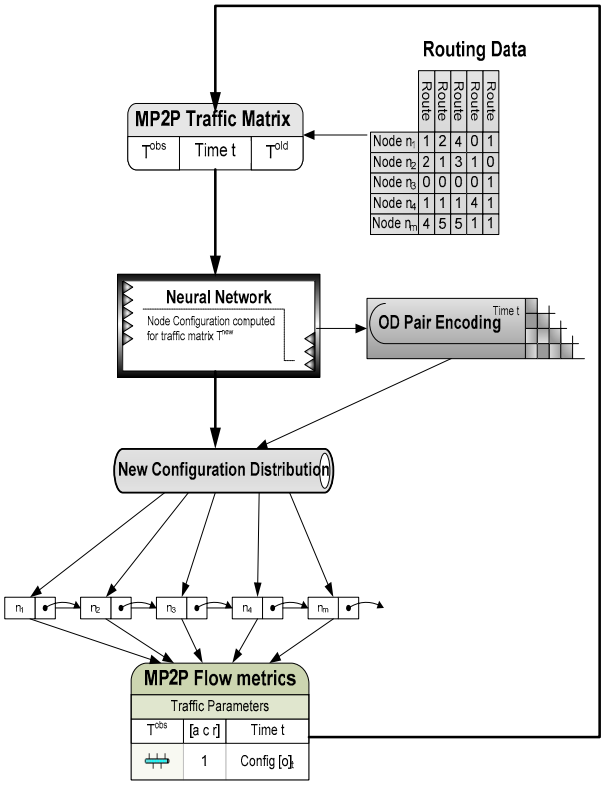

Figure 2. Schematic layout of the traffic sampling architecture. The traffic matrix compiled from the flow reports acts as an input vector into the neural network architecture, which computes the optimal configuration for each node, distributed to each respective node. New traffic measurements are then generated, collected and re-parsed into the traffic matrix for re-analysis.

Given the observed traffic matrix for the current measurement period T^{obs} and the old traffic matrix obtained from the week before T^{old}, a new traffic matrix can be computed using a conservative update policy. The resulting traffic matrix T^{new} is then used as the input to the algorithm that obtains the manifests for the next 5-minute period. Thus, the following conservative update algorithm was designed, as outlined in Algorithm 1. This algorithm determines the differences between the new and the existing traffic reports, and computes a new traffic matrix entry if the difference between the values exceeds a specific threshold. The residual resources are measured and allocated to the respective nodes according to the resource utilisation of the peer nodes reporting the traffic matrix.

Algorithm 1: P2P Traffic Sampling Algorithm

```
1:FUNCTION {P2P_Traffic_Matrix_Update (T_i^obs, T_i^old, Δ)}
2://check for significant differences between T^obs and T^old
3: δ_i = |T_i^obs − T_i^old|/T_i^old // δ_i = estimation error for ODi
4: if δ_i > Δ then
5: // compute new traffic matrix entry
6:     T_i^new = [0]
7: else
8:     T_i^new =T_i^old
9: end if
10: if T_i^obs ≥ T_i^old then
11:    T_i^new = T_i^obs
12: end if
13: if T_i^obs < T_i^old then
14: // check resource utilisation for all routers monitoring OD_i
15:    for all x_i ∈ x do
16:        C_i = resource utilisation of x_i
17:          // result: 1=free, 0=busy
18:    end for
19:    if ΣC = i then
20:    //   all   nodes   have   residual resources available
21:          T_i^new = T_i^obs
22:    else
23:    // not enough residual resources available – revert to previous matrix
24:          T_i^new = T_i^old
25:    end if
26: end if
27: return(T_i^new)
```

The challenge in deploying this solution is limited by the ability to process this information effectively for each node, and to detect when the information is new or no longer valid [19]. A neural network algorithm can provide the necessary framework to process and determine the optimal nodal information. A neural network which uses a back-propagation architecture promises to be effective in this application since training is done during each iteration by the traffic data that is emanating from each node in the network.

B. Back-propagation Neural Networks

Back-propagation neural networks are created by the generalisation of the Widrow-Hoff learning rule to multiple-layer networks and nonlinear differentiable transfer functions [17]. A set of training vectors and their corresponding target vectors are used to train the neural network until it is capable of approximating a function, associate input vectors with specific output vectors, or classify each input vector in an appropriate way as defined by the user. Any function with a finite number of discontinuities can be approximated by networks having biases, a sigmoid layer, and a linear output layer.

It has been shown that back-propagation networks tend to give reasonably good classification results when they are presented with input vectors that they have never seen [20]. This generalisation property makes it possible to train the network on just a representative set of the input/target pairs and get reasonably good results without training the network on all possible input/output pairs.

The simplest implementation of the back-propagation learning algorithm updates the weights of the network and its biases in the direction in which the performance function decreases most rapidly. This is represented by the negative of the gradient of the function. A single iteration of the algorithm is given by:

$$x_{k+1} = x_k + \alpha_k g_k \quad (9)$$

where x_k is a vector of current weights and biases, g_k is the current gradient, and α_k is the learning rate.

In most neural network training algorithms the learning rate α_k is used to determine the length of the weight update, and is known as the step-size [23]. There are two different ways in which this gradient descent algorithm can be implemented: incremental mode and batch mode. In incremental mode, the gradient is computed and the weights are updated after each input is applied to the network. On the other hand, in batch mode all the input vectors are applied to the network before the weights are updated. Both solutions eventually converge and can be adopted.

However, most gradient descent algorithms are often too slow for practical real-time problems [24]. The basic algorithm adjusts the weights in the steepest descent direction. It turns out that, although the function decreases most rapidly along the negative of the gradient, this does not necessarily produce the fastest convergence. In the conjugate gradient algorithms a search is performed along conjugate directions, which generally produces faster convergence than steepest descent directions [25].

Most of the conjugate gradient algorithms have the step-size adjusted after each iteration. A search is made along the conjugate gradient direction to determine the best step-size that minimises the performance function along that line. The conjugate gradient algorithm used starts by searching in the steepest descent direction (g) on the first iteration of the observed traffic input vector T^{obs}, thus:

$$T^{obs}{}_0 = - g_0 \quad (10)$$

A line search is then performed to determine the optimal distance to move along the current search direction. Therefore, combining (9) and (10) yields:

$$x_{k+1} = x_k + \alpha_k T^{obs}{}_k \quad (11)$$

The next search direction is determined such that it is conjugate to the previous search directions. The general procedure for determining the new search direction is to combine the new steepest descent direction with the previous search direction as:

$$T^{obs}{}_k = -g_k + \beta_k T^{obs}{}_{k-1} \quad (12)$$

where β_k is a constant.

The various versions of the conjugate gradient algorithm are distinguished by the way in which β_k is computed. One solution is to use the Fletcher-Reeves algorithm which updates this constant using:

$$\beta_k = \frac{g_k^T g_k}{g_{k-1}^T g_{k-1}} \quad (13)$$

This is the ratio of the norm squared of the current gradient to the norm squared of the previous gradient [20]. The conjugate gradient algorithms require only a little more storage than the simpler gradient decent ones. Thus, these algorithms work well in neural networks that utilise a large number of weights.

The conjugate gradient algorithms are usually much faster than variable learning rate back-propagation, although the results vary from one problem to another [24]. Furthermore, the computational intensity required is comparable to the standard algorithms, making them ideal for implementation in power-efficient environments such as MP2P clients [1].

A tuned neural network algorithm is essential in determining the optimal peer nodal information to process node routing and thus obtain accurate traffic reports in T^{obs}. Consequently, from equation (13), any delays in reporting and processing by the back-propagation algorithm may result in estimation errors in T^{obs} and consequently δ_i. As the back-propagation network may incur significant delays in training, when subject to a large input dataset [20], additional ongoing efforts are underway to investigate optimisation methods, such as the use of Self-Organising Maps (SOMs) neural architecture in training.

C. State-switching optimisation

Additional optimisation to the traffic sampling algorithm is obtained through the efficient use of the observed traffic matrix T^{obs} to influence the state-switching of each peer node [19]. By periodic measurement of the neighbouring nodes, the band state of each mobile node can be changed for $node_i$ to a more efficient state depending on the amount of residual resources.

Algorithm 2: State Switching Algorithm

```
1:FUNCTION {Check_load(node_i, t_i^current, t_i^old)}
2://check time window hysteresis expiry t^exp
3: δ_i = |t_i^current - t_i^old| // δ_i = time elapsed from last measurement for node_i
4: if δ_i > t^exp then
5:     //check resource utilisation for all neighbouring nodes x monitoring node_i
6:     for all x_i ∈ x do
7:         C_i = resource utilisation of x_i
8:         // result: 1=free, 0=busy
9:     end for
10:    if (ΣC = i) then
11:    //all neighbouring nodes have residual resources, reduce band
12:        Reduce_Band(node_i)
13:    else
14:    if (ΣC < i and ΣC > y) then
15:    //at least y neighbouring nodes have residual resources, keep existing band
16:        F_i^new = F_i^old
17:    else
18:    //    no residual resources available, increase band
19:        Increase_Band(node_i)
20:    end if
21: end if
22: return(F_i^new)

23:FUNCTION {Reduce_Band(node_i)}
24:    //Query node band state energy table
25:    F_i^old=lookup(node_i, phone_energy_table)
26:    //Subtract one band energy state
27:    F_i^new = F_i^old - 1
28: return(F_i^new)

29:FUNCTION {Increase_Band(node_i)}
30:    //Query node band state energy table
31:    F_i^old=lookup(node_i, phone_energy_table)
32:    //Add one band energy state
33:    F_i^new = F_i^old + 1
34: return(F_i^new)
```

Figure 3 outlines the allowed state transitions while Algorithm 2 provides a high-level description of the algorithm. It is important to note that state-switching algorithms can only be supported where the mobile client operating system (OS) supports active band selection through the use of APIs, such as provided by Symbian [21] and Android [22]. Otherwise, the client MP2P software may resort to the use of the traffic sampling algorithm only [28].

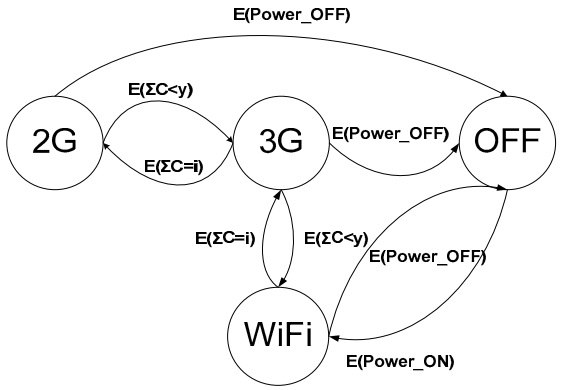

Figure 3. Allowed state transitions by the band switching algorithm, from [OFF] to [WiFi] and intermediate bands. E() represents an event and indicates the cause of the transition. Wherever $y < \Sigma C < i$, the state is transition-less.

IV. TESTING

In order to determine the energy dissipation across the different algorithms a mobile peer network was first designed and simulated using MATLAB®. The mobile peer network was designed using algorithms derived from the BitTorrent technology [8].

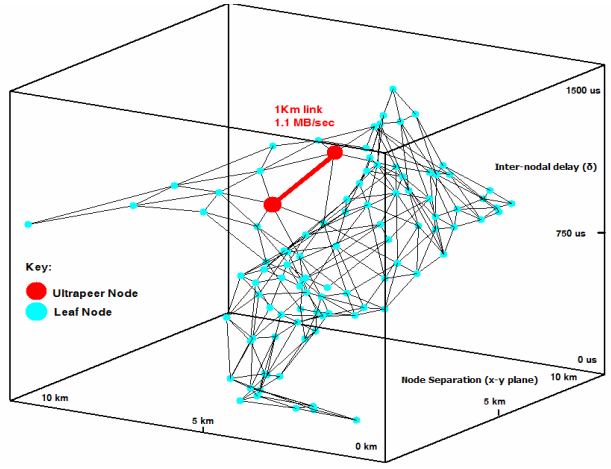

Figure 4: Mobile peer network simulation structure, showing node separation in the XY plane, and uTP inter-nodal delay (δ) in the Z plane. The two ultrapeer nodes are connected by a UMTS Release 5 wireless link with HSPA, operating at 1.4MBps (nominal rate 1.1MB/s).

The traffic sampling algorithm, utilising the back-propagation neural architecture consisting of a two-layer network, having 11 neurons in the hidden layer and 2

neurons in the top layer, and utilising a tan-sigmoid and a linear transfer function for the hidden and top layer respectively, was implemented.

Stochastic models of traffic sampling periods correlated with peer node power consumption were analysed to determine the optimum observation model for traffic sampling. High-frequency windows (1-4 min/sample) produce more frequently updated peer nodes, however this comes at a significant detriment to power consumption due to frequent polling. Conversely more conservative windows (6-15 min/sample) are more energy efficient, but they are impractical to intercept mobile peer clients with high mobility and/or cell state transitions. A 5-minute window was selected as the best trade-off between power-consumption and node data validity. Thus, the traffic sampling algorithm was built using the 5-minute observation model, and the resulting traffic matrix was used as input to the neural network.

The testing process consisted of a series of UDP connections, and high-level file transfers between two peer mobile nodes, with a 1km separation running on an emulation software [23]. These were connected across uTP, or the UDP-based implementation of the BitTorrent protocol [8], through the UMTS RNC as shown in Figure 4. The access network topologies were simulated using two-tiered wireless propagation algorithms [2]. Background packet traffic from the underlying UMTS network was incorporated to increase the network loading and investigate the proposed algorithm's energy profile under various traffic scenarios.

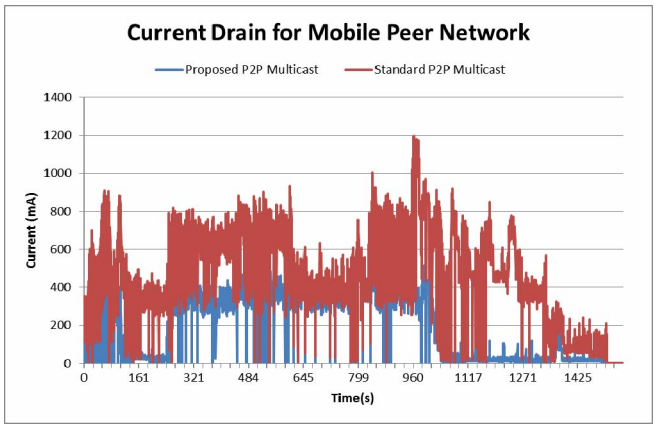

Figure 5(a). Current profile for mobile peer connectivity in mA, showing the difference in current drain between the standard mobile BitTorrent algorithm and the proposed traffic-sampled Mobile BitTorrent method.

The two peer nodes were configured to transmit (and receive respectively) a file having a size of 1.024 GB across the peer network over the uTP layer, whilst all other leaf nodes were configured as source-sink pairs to simulate the aggregate traffic on the network, generated according to a Poisson distribution. All queuing effects were simulated on the ultra-peer link using the FIFO queuing algorithm [25]. The network link capacity was set to 1.4Mbps and the round-trip propagation delay for the two ultra peer nodes (δ) was set to $1000\mu s^{-1}$. The simulation window was 1500 seconds with a sampling time of 10 samples/second. The entire tests were affected over a period of 7 days with hourly measurements during the UMTS network busy hour. After the observation period expired, the results were collated with a confidence interval of at least 95% across all tests.

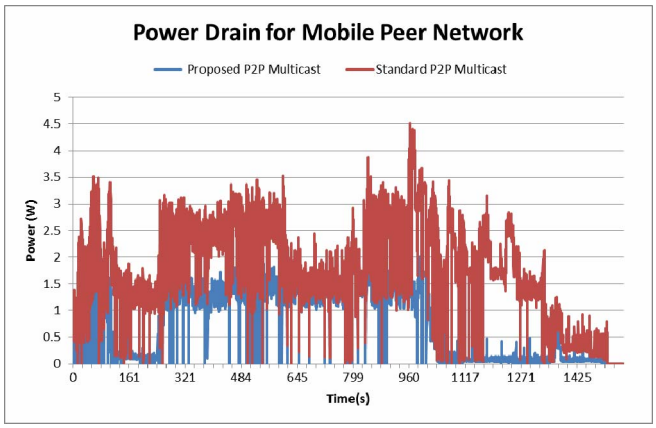

Figure 5(b). Power profile for mobile peer connectivity in W, showing the difference in power dissipation between the standard mobile BitTorrent algorithm and the proposed traffic-sampled Mobile BitTorrent algorithm.

In order to measure the effective mobile CPU load and resulting current and voltage dissipation, the mobile components were emulated with resource load monitoring capabilities to gauge the effective portion of processing cycles devoted to peer traffic routing. The resulting profile was fed into the mobile emulation software to determine the resulting drain on current and voltage respectively [22-23].

Figures 5 (a) and (b) show the difference in current drain between the standard and traffic sampled mobile peer client algorithms. The resulting drain is approximately 50% of the nominal current used by the standard algorithm. The optimal traffic sampling also resulted in the file transfer completing in 960 seconds, well ahead of the 1505 seconds taken by the standard algorithm.

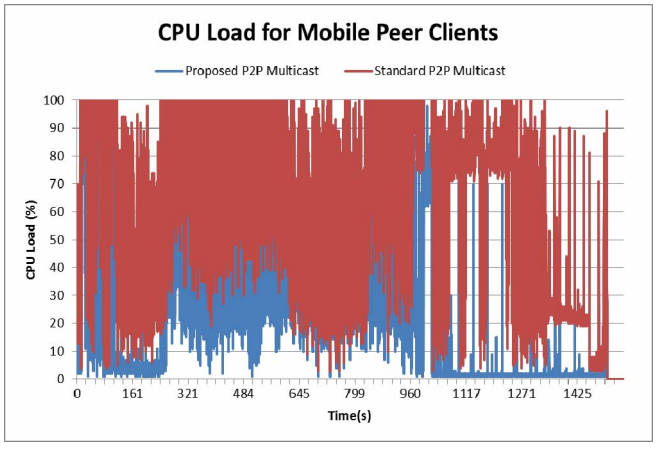

Figure 5(c). CPU load for the mobile peers, showing the difference in load between the standard mobile BitTorrent algorithm and the proposed traffic-sampled Mobile BitTorrent algorithm.

Figure 5(c) illustrates the discrepancy in CPU load for the proposed sampled BitTorrent client compared to the standard algorithms. The additional CPU cycles using the standard algorithm are the result of iterative routing of peer traffic through congested peer nodes, and/or redundant peer information, leading to a higher net CPU load on the mobile client and contributing to redundant routing information in each peer client that is discarded due to invalid or untimely data. Such situations occur when the routing information changes during data transmission between peer nodes, and is symptomatic of high latency or poor peer client localisation [2]. In this case, there is significant processing load efficiency using the traffic-sampled peer protocol, which is due to the rapid information delivered because of the lower network delay.

To further validate the performance improvements in a typical service deployment when compared to similar algorithms, a series of file transfer tests was performed across the network topologies. Table 1 illustrates the performance of the mobile simulator [22-23] for a number of 1GB file transfers across the peer networks using the Greedy Perimeter Stateless Routing (GPSR), Geographic Adaptive Fidelity (GAF) and Geographic and Energy Aware (GEAR) algorithms, when compared with the proposed Traffic Sampling (TS) technique. The performance exhibited by the TS model compares favourably, particularly in the file transfer times and processor loads, compared with the alternative methods presented in [12-15].

Although GPSR has a smaller memory footprint, the average RTT is still longer than the TS method because the rest of the node paths are unknown and have to be discovered. The TS algorithm deduces the link of each next-step based on the last measured performance of each node, contributing to a shorter RTT, which compares more favourably to other research efforts reported in [13] and [14], as can be seen from the results given in Table 1. This effectively reduces the CPU loading, thereby leading to a lower current and power drain on the power supply of the mobile peer client.

TABLE I. MOBILE PEER NETWORK PERFORMANCE

		P2P Key Performance Indicators (KPIs)			
	Test	***GPSR***	***GAF***	***GEAR***	***TS***
1	Memory Footprint (kB)	618	1125	1982	1238
2	File Transfer Time (s)	1142	1223	1890	960
3	Round-trip-time (RTT)	7s	9s	6s	2s
4	CPU Load (%)	32%	58%	89%	42%

A second series of tests was performed to determine the relative performance of the traffic sampling mechanism as a distribution of the physical peer location. As the peer network is mobile there is a significant variance with fixed peer networks, resulting in issues outlined in Section II. Consequentially a recursive test was performed across the 10km square test network with each peer node effecting a UDP burst pattern in alternate send/receive mode.

Fig. 6 shows the error *z(k)* as a function of location across the mobile peer network. Whilst the stability of the traffic sampling controller is consistent throughout the network, slightly higher errors are exhibited in areas where there are fewer peer nodes. This is due to the inverse relationship of the controller error with the background traffic and is similar to the performance shown in Fig. 6.

The Traffic Sampling mechanism displays significantly higher performance under loaded traffic conditions, and is thus suited for mobile peer networks, as these tend to exhibit higher instances of local congestion due to insufficient resources [2]. Judicious use of this algorithm enables the deployment of peer technologies in mobile networks without causing a packet traffic overload to the network, and therefore contributes to more efficient resource allocation in mobile peer networks.

An observational note is that the CPU load on the mobile client, and consequently the power drain, is highly correlated to the size of the peer algorithm memory footprint. One factor is due to the additional processor run-time loads involved in general memory management (pointer allocation, matrix operations, etc.) of the mobile operating system [21, 22].

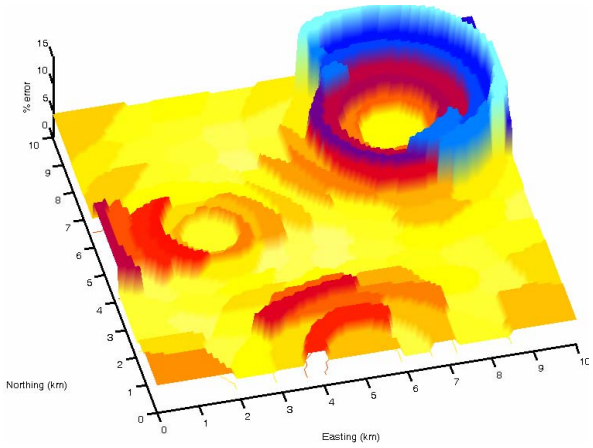

Figure 6: Traffic Sampling controller distribution error z(k) throughout the mobile peer network. Whilst the stability of the traffic sampling controller is consistent throughout the network, slightly higher errors are exhibited in areas where there are fewer peer nodes. This is due to the inverse relationship of the controller error with the background traffic.

V. CONCLUSIONS

This paper has presented a solution that exploits traffic data and neural networks in wireless peer networks to optimise the power consumption profile in mobile clients. Results have shown that the solution also provides better packet allocation efficiency on the radio interface compared to other algorithms [12-15], for the same number of peer clients. The significant improvement has been obtained by consuming fewer mobile CPU cycles by providing and

allocating resources while taking into account the conditions of the radio environment, together with the availability of traffic channels at any specific time or area within the wireless network.

The net result from the effective use of this method is to deploy the algorithm across multi-technology topologies in a step towards MP2P convergence. The simulation results have shown that the proposed method promises high power efficiency across network switching topologies as different technologies are exploited to provide various mobile services using these overlay networks.

Research is currently ongoing to investigate peer node transmission across smaller inter-nodal differences to reduce power consumption, using an algorithm based on a derivative of the state-switching method used in this work. This will allow for more efficient use of the radio spectrum as well as the peer node burst rates with peer nodes having relatively close proximity.

Additional research is also envisaged in investigating the performance of the neural network algorithm in high mobility scenarios, that is where peer users are moving at speeds in excess of $60ms^{-1}$, and to determine an optimal feedback mechanism to respond efficiently across multiple radio access network topologies. These methods will allow for the mobile power dissipation to be more efficient in a technology- and mobility-agnostic environment.

REFERENCES

[1] J.K. Buhagiar, C.J. Debono "Exploiting Traffic Sampling Techniques to Optimize Energy Efficiency in Mobile Peer Networks," in Proc. of the 2009 First International Conference on Advances in P2P Systems (AP2PS 2009), pp. 72-77, October 2009.

[2] J.K. Buhagiar, C.J. Debono "Exploiting Adaptive Window Techniques to Reduce TCP Congestion in Mobile Peer Networks," in Proc. of the 2009 IEEE Wireless Communications and Networking Conference (WCNC 2009), April 2009.

[3] Y. Luo, O. Wolfson, "Mobile P2P Databases", in NSF Workshop on Data Management for Mobile Sensor Networks (MobiSensors), Pittsburgh, PA, January 2007.

[4] A. Datta, S. Quarteroni, and K. Aberer, "Autonomous Gossiping: A Self-Organizing Epidemic Algorithm For Selective Information Dissemination in Wireless Mobile Ad-Hoc Networks," in The International Conference on Semantics of a Networked World, 2004.

[5] A. Helmy, Efficient Resource Discovery in Wireless AdHoc Networks: Contacts Do Help. Book Chapter in Resource Management in Wireless Networking by Kluwer Academic Publishers, May 2004.

[6] R. Krishnan, M. Smith, R. Telang, The economics of peer-to-peer networks, Carnegie Mellon University, 2002.

[7] J. Hellerstein, W. Hong, S. Madden, K. Stanek "Beyond Average: Toward Sophisticated Sensing with Queries," in Proc. of the 2nd Int. Workshop on Information Processing in Sensor Networks, 2003.

[8] BitTorrent [Online]. http://en.wikipedia.org/wiki/BitTorrent © 2009

[9] Gnutella [Online]. http://en.wikipedia.org/wiki/Gnutella © 2009

[10] V. Sekar, M. Reiter, W. Willinger, H. Zhang, "Coordinated Sampling: An Efficient Network-wide Approach for Flow Monitoring," Technical Report, CMU-CS-07-139, Computer Science Dept., Carnegie Mellon University, 2007.

[11] D. Kim, M. Lee, L. Han, H. In "Efficient Data dissemination in Mobile P2P ad-hoc networks for ubiquitous computing", in Proc. of Int. Conf. on Multimedia and Ubiquitous Engineering, pp 384-389, 2008.

[12] A. Visvanathan, J. H. Youn, and J. Deogun, "Hierarchical Data Dissemination Scheme for Large Scale Sensor Networks," in IEEE International Conference on Communications (ICC'05), pp. 3030-3036, May 2005.

[13] B. Karp and H. T. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Sensor Networks," in The 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'00), pp. 243-254, Aug 2000.

[14] Y. Xu, J. Heidemann, and D. Estrin, "Geography-informed Energy Conservation for Ad hoc Routing," in The ACM International Conference on Mobile Computing and Networking, pp. 70-84, Rome, Italy, July 2001.

[15] Y. Yu, R. Govindan, and D. Estrin., "Geographical and Energy Aware Routing: A Recursive Data Dissemination Protocol for Wireless Sensor Networks," Technical Report UCLA/CSD-TR-01-0023,UCLA, May 2001.

[16] R. Oliveira, L. Bernardo, and P. Pinto, "Flooding Techniques for Resource Discovery on High Mobility MANETs," Workshop on Wireless Ad-hoc Networks, 2005.

[17] J. Kulik, W. Heinzelman, and H. Balakrishnan, "Negotiation-Based Protocols for Disseminating Information in Wireless Sensor Networks," Wireless Networks, vol. 8, pp. 169-185, 2002.

[18] H. Ballani, P. Francis, "CONMan: A Step Towards Network Manageability", in Proc. of ACM SIGCOMM, 2007.

[19] V. Sekar, M. Reiter, W. Willinger, H. Zhang, R. Kompella, D. Andersen, "cSAMP: A System for Network-Wide Flow Monitoring," in Proc. of the 5th USENIX Symposium on Networked Systems Design and Integration, 2008.

[20] S. Lawrence, C. L. Giles, A. C. Tsoi, A. D. Back, "Face Recognition: A Convolutional Neural Network Approach," IEEE Transactions on Neural Networks - Special Issue on Neural Networks and Pattern Recognition, Vol. 8, No. 1, pp 98-113, 1997.

[21] Symbian OS Emulator 5th Edition, Symbian C++ SDK © 2009 Symbian Foundation.

[22] Google Android OS Emulator 1.5r2, 2.0 © 2008-9 Google, http:// code.google.com/android

[23] Control Systems Toolbox MATLAB © Mathworks 2009.

[24] Neural Networks Toolbox MATLAB © Mathworks 2009.

[25] P2P Multicast Library (PML), © Sourceforge, 2009, http://pml.sourceforge.net

[26] L. Feeney, M. Nilson, "Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment," INFOCOM, 2001.

[27] O. Wolfson, B. Xu, R. Michael Tanner, "Mobile Peer-to-peer Data Dissemination with Resource Constraints," in Proc of the International Conference on Mobile Data Management, pp 16-23, 2007.

[28] J.K. Buhagiar, C.J. Debono "Optimizing Multicast Protocols to Reduce Energy Dissipation in Mobile Peer Networks," in Proc. of the 2010 IEEE Wireless Communications and Networking Conference (WCNC 2010), April 2010.