Tula: Balancing Energy for Sensing and Communication in a Perpetual Mobile System

Jacob Sorber, Aruna Balasubramanian, Mark D. Corner, Joshua Ennen, Carl Qualls

Abstract—Due to advances in low power sensors, energy harvesting, and disruption tolerant networking, we can now build mobile systems that operate perpetually, sensing and streaming data directly to scientists. However, factors such as energy harvesting variability and unpredictable network connectivity, make building robust and perpetual systems difficult. In this paper, we present a system, Tula, that balances sensing with data delivery, to allow perpetual and robust operation across highly dynamic and mobile networks. This balance is especially important in unpredictable environments; sensing more data than can be delivered by the network is not useful, while gathering less underutilizes the system's potential. Tula is decentralized, fair and automatically adapts across different mobility patterns. We evaluate Tula using mobility and energy traces from TurtleNet—a mobile sensor network we deployed to study Gopher tortoises—and publicly available traces from the UMass DieselNet testbed. Our evaluations show that Tula senses and delivers data at up to 85% of an optimal, oracular system that perfectly replicates data and has foreknowledge of future energy harvesting. We also demonstrate that Tula can be implemented on a small microcontroller with modest code, memory, and processing requirements.

Index Terms—Support for adaptation, store and forward networks, network management, wireless sensor networks, mobile communication systems.

✦

1 Introduction

Due to three key innovations: small programmable sensors; energy harvesting [20], [26]; and disruption tolerant networking, mobile systems are poised to answer many questions about a wide range of natural and manmade systems. Recent efforts focusing on zebras [32], whales [12], turtles [26], people [15], and vehicles [9] have shown that in-situ monitoring using embedded devices can provide unprecedented and transformational data. When these systems harvest energy from their environment and gather data in a robust manner, they can become *perpetual* and self managing, streaming data directly to scientists for decades.

However, a number of external factors make building robust perpetual systems difficult. Seasonality, habitat disruption, changes in social networks and mobility can drastically affect network connectivity and energy harvesting. Without basic parameters such as network connectivity and energy availability, it is impossible to tune power management and routing. A key premise of this problem domain is that: *node mobility, unpredictable network connectivity and uncertain energy availability represent the greatest challenges for designing perpetual systems..*

Early perpetual systems, like ZebraNet [32], focused on minimizing energy consumption rather than energy awareness and adaptation. More recent systems either use local energy adaptation techniques without considering data delivery [19], [23], [26] or use adaptation techniques for purely static networks [11]. However, adapting to both energy and network variations is considerably more difficult. In particular, a node needs to adapt and balance both its sensing and routing tasks. In a long-running system the goal is to gather as much data from nodes as the limited resources of network bandwidth and energy permit. Sensing more data than can be delivered by the network is not useful, while gathering less underutilizes the system's potential. In addition, sparse networks that depend on node cooperation for routing data [2], [27], must balance the energy devoted to sensing and routing their own data, with energy used to route data for other nodes.

In this paper, we present a system, Tula, that addresses this challenge for mobile sensor networks. A Tula node uses a distributed algorithm to balance energy allocation across three tasks—sensing, routing the node's own data and the routing data for other nodes. The Tula energy allocation ensures max-min fairness, which allows data collection from all nodes, including poorly connected nodes. The key insight in Tula is that sensing and routing are inherently dependent, and optimizing only one or the other in an energy-constrained environment is futile.

Given the allocation for sensing and routing, Tula

- *J. Sorber is with Dartmouth College, Computer Science Department, 6211 Sudikoff Lab, Hanover, NH 03755. E-mail: jacob.m.sorber@dartmouth.edu*
- *A. Balasubramanian is with the University of Washington, Computer Science & Engineering Department, 185 Stevens Way, Seattle, WA 98195. Email: arunab@cs.washington.edu*
- *M. D. Corner is with the University of Massachusetts Amherst, Computer Science Department, 140 Governors Drive, Amherst, MA 01003. Email: mcorner@cs.umass.edu*
- *J. Ennen is with Maryville College, Natural Sciences Division, 502 E. Lamar Alexander Parkway Maryville, TN 37804. Email: josh.ennen@maryvillecollege.edu*
- *C. Qualls is with the University of Southern Mississippi, Department of Biological Sciences, 118 College Drive #5018, Hattiesburg, MS 39406. Email: carl.qualls@usm.edu*

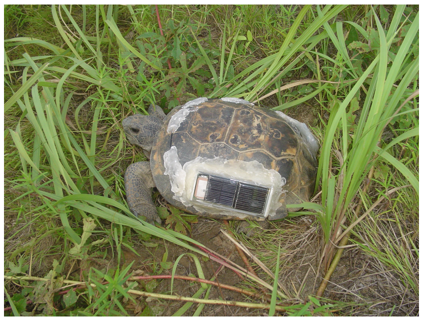
Fig. 1. A solar-powered tracking device used in TurtleNet.

uses an adaptive sensing system to collect data and a DTN routing algorithm to deliver the data. We formulate the Tula allocation problem as a constraint optimization problem (COP). Each Tula node measures energy consumption for sensing and communication and gathers data about the environment through node meetings, to locally solve the COP on an embedded device. Tula is general, and automatically adapts across mobility patterns, from static to highly mobile environments.

Our design of Tula targets mobile sensing systems broadly, and we evaluate Tula in the context of two specific systems. The primary example is TurtleNet, a mobile sensor network that we deployed to study Gopher tortoises. The TurtleNet deployment consists of 17 tortoises and we have collected energy harvesting and mobility data. The deployment has been in operation since August 2008. In our evaluation, we use traces from TurtleNet, combined with an implementation of Tula on TinyNodes [7]. We also evaluate Tula on publicly available traces obtained from the UMass DieselNet project [4].

Our evaluations over both TurtleNet and DieselNet show that Tula senses and delivers data within 75% of an optimal, oracular system that perfectly replicates data and has foreknowledge of future energy harvesting. The protocol is fair in terms of delivery rates across nodes, and comes within 95% of the optimal in terms of the max-min fairness objective. Tula not only works well for sparse mobile networks, but also for static mesh networks. Our evaluations on a synthetically generated mesh network shows that Tula adapts well to the static environment and senses and delivers data within 85% of the optimal. Tula also outperforms systems that adapt only sensing or only routing, collecting up to 27% more data and completely avoiding energy failures and energy waste. Finally, we show that Tula can be implemented on a small microcontroller with modest code, memory, and processing requirements.

2 Applications and Challenges

Devices that operate perpetually using harvested energy represent a new class of mobile system that promises to enable a wide and largely unexplored range of potential applications. This vision includes significant advances for scientists studying mobility in nature. In spite of decades-worth of study, the movements and behaviors of most animal species in the wild are completely unknown. Current methods like trapping and manual radio telemetry are labor intensive, yield few data points, and significantly increase the frequency of animal interactions with humans. By using small in-situ sensor devices to observe animal location, movement, and environmental conditions, researchers will be able to collect more data at higher temporal densities with minimal impact on behavior. This shift promises to answer long-debated questions about habitat usage, population trends, and complex interactions between different species, including humans.

2.1 TurtleNet

In light of these potential benefits, we have deployed TurtleNet, a mobile network with the goal of overcoming many of the challenges faced by perpetual sensing systems. Our deployment consists of 17 tracking devices attached to Gopher Tortoises (Gopherus polyphemus), shown in Figure 1. Each device consists of a Shockfish TinyNode [7], a solar panel, a battery, multiple sensors, and additional energy measurement hardware.

During operation, the devices record connection opportunities with neighboring nodes and periodic sensor readings, including temperature, GPS coordinates, battery level, solar energy harvested and energy consumption. Unlike traditional networks, these nodes rarely have an end-to-end connection to one of the two deployed GPRS-enabled base stations, and devices must opportunistically deliver collected data using mobile-to-mobile routing [2], [27]. When two mobile nodes are within communication range, called a connection opportunity, they exchange data. This data is stored and then forwarded during subsequent connection opportunities until it is eventually delivered to the sink. The network has been in operation since August 2008.

2.2 Challenges

On analyzing the deployment traces, we uncovered a number of key challenges. The primary difficulty in designing TurtleNet—and generally any untethered mobile network—is the continuous variation of both energy harvesting and network connectivity due to mobility. Figure 2 shows how a node's daily harvested energy varied—experiencing both day-to-day and seasonal changes. Note that within a 10-day period, daily harvesting ranged from less than 0.1kJ to more than 1.7kJ. *In order to support perpetual operation, a device must adapt its behavior over time.*

In addition to temporal variation, energy harvesting also varies considerably across the network. Figure 3

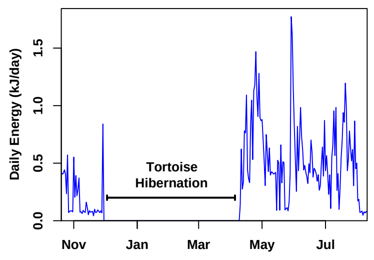

Fig. 2. Daily solar energy is shown for a TurtleNet node before and after hibernation.

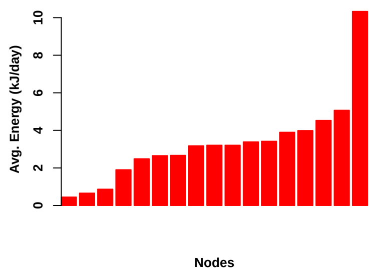

Fig. 3. The average daily energy harvested by each TurtleNet node during a 1-month trace.

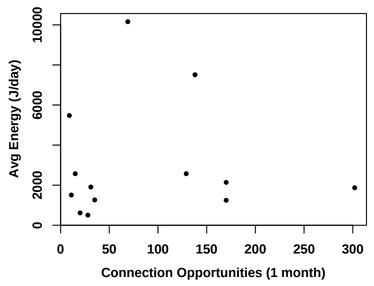

Fig. 4. Harvested energy plotted against number of meetings for each node. Energy-rich nodes are not necessarily better connected and vice versa.

shows the average daily energy harvested by all nodes in the network over a 1 month period of time, sorted to show the energy distribution. The figure shows that there is significant variation in energy harvesting across nodes. *With diverse energy budgets in the network, each node needs to balance its available energy between sensing and delivering data.*

Recall that nodes rely on other well connected nodes to store and forward their data to the destination. Unfortunately, there is very little correlation between how connected a node is and the energy it gathers, as shown in Figure 4. The well-connected hubs that are best positioned to route data may not have sufficient energy to support network demand. *Routing decisions in perpetual networks must depend on not only topology, but also the available energy.*

Finally, in TurtleNet—and most other mobile systems—connections between mobile nodes often exhibit patterns due to social habitats as shown in Figure 5. The figure shows that 40% of node meetings repeat more than 10 times a month. The meeting patterns are not completely random and can be leveraged to combat the network's uncertainty. In other words, if two peers have a connection opportunity, we can expect that the peers will have future connection opportunities.

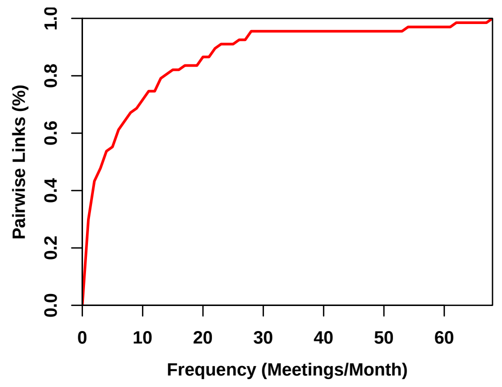

Fig. 5. CDF of the pair-wise meeting frequency during 1 month of TurtleNet operation. While some meetings occur too infrequently to be very useful, 50% of the node pairs repeat at least 5 times.

2.3 Design Goals

In this paper, we describe Tula, a system that addresses the above challenges by supporting perpetual operation and ensuring fair and efficient data collection. Specifically, our design goals are: *(i)* Perpetual operation, *(ii)* Ability to operate in sparse, resource constrained environment, and *(iii)* Fairness.

Perpetual operation requires that energy spent sensing, storing, processing, and communicating must be matched with harvested energy. In addition, the sensing rates must match with the rate of data delivery, since sensing more data than can be delivered is not useful.

Tula must operate in sparsely connected mobile networks and on energy and computationally constrained platforms. Sparse connectivity requires that nodes use DTN routing [2], [30] to route data using node mobility to overcome disruptions.

Finally, fairness in Tula is critical. Tula operates in networks with significant variation in both energy and network connectivity. Maximizing network throughput or minimizing delay, without enforcing fairness, may result in well connected and energy-rich nodes collecting and delivering their own data at a high rate, while starving nodes that are further away. Well connected nodes may also have their energy budgets depleted by high network demand.

Different models have been proposed for sharing resources among nodes and flows within a network. Wildlife tracking applications typically seek to characterize animal behavior and their interactions with the environment by collecting as much data from as many nodes as possible. This emphasis on sensing "breadth" rather than "depth" can be expressed using max-min fairness, which requires that a nodes' sending rate be improved only after all lower rates have already been maximized. In Tula we focus on achieving max-min fairness, and other fairness models are

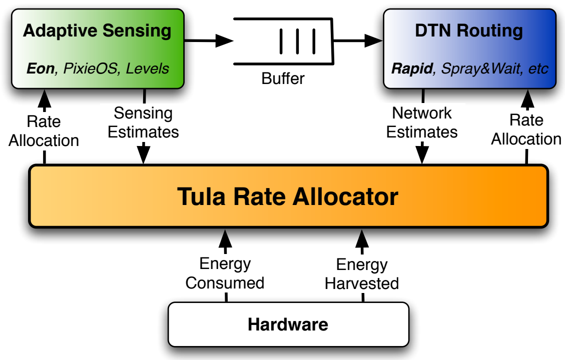

Fig. 6. The Tula architecture.

left to future work (see Section 9).

3 Tula Architecture

The Tula architecture, shown in Figure 6, consists of three main components: an *Adaptive Sensing system* for collecting sensor data, a *DTN routing algorithm* for opportunistically delivering that data, and a *Rate Allocator* that coordinates both sensing and routing activities by appropriately allocating resources.

Existing adaptive sensing systems adjust application sensing rates alone to adapt to the changes in a device's energy budget. A sensing system may have several sensors and application tasks. Existing adaptive systems, including Eon [26], PixieOS [23], and Levels [19], estimate or measure the energy costs of various application tasks and automatically adjust application behavior to match a device's energy budget.

Existing DTN routing systems opportunistically route network packets from source to destination using sporadic and uncertain device-to-device meetings. Many systems have been designed, including Rapid [2] and Spray and Wait [27], to effectively deliver data over intermittent links while responding to changing network conditions.

Unfortunately, the adaptive sensing and DTN routing systems are not designed to work together. Existing adaptive sensing systems consider only local energy constraints, ignoring the impact of sensed data on the network. Similarly, DTN routing systems assume unlimited energy and consider only bandwidth restrictions. Tula's core function is to overcome this challenge by combining the benefits of adaptive sensing and DTN routing into a single coordinated system. Our evaluation (Section 7) shows that Tula significantly outperforms systems that only adapt sensing or only adapt routing to changing energy conditions.

Rather than build a complete system from scratch, Tula abstracts the sensing and routing and controls these systems using an allocator that balances energy for sensing with that of data delivery. The Tula energy allocation is most easily understood in terms of rate: the number of packets, or bytes, that can be generated by sensing and delivered by routing, over some time period.

The allocator's objective is to maximize the rate at which sensor data is collected and delivered, while ensuring that the allocated rates are fair to all nodes. To this end, Tula must appropriately adjust *(i)* the rate of sensing, *(ii)* the routing rate for the node's own data, and *(iii)* the maximum routing rate for each neighbor's data. Given the sensing rate, Tula leverages existing sensing mechanisms that adapt the local sensing task according to available energy. Given a routing rate, Tula adapts existing DTN routing protocols to route data within the given rate.

3.1 Adapting Sensing

Adaptive sensing systems change their sensing rates according to energy conditions. Adapting the sensing rate is especially important in sensor nodes that have multiple sensors and have high variability in harvest energy.

Eon [26], an adaptive sensing system that we use in Tula, uses hardware support to measure energy consumption and harvesting. Eon combines these measurements with runtime information about the application in order to estimate the system's idle power draw and the energy cost of various program tasks. Finally, Eon uses this information to determine how much energy is required to sense data at a given rate.

This relationship is communicated to the rate allocator, which uses the information to solve the allocation problem (Section 4).

3.2 Adapting Routing

The Tula rate allocator only assigns the maximum rate at which a node can route data for each of its neighbors. The actual routing decision involves other tasks including estimating routes, tracking acknowledgements, and adapting the route to changing network connectivity. In Tula, we use the Rapid [2] DTN routing protocol and adapt it to an energy constrained environment.

Rapid estimates a distance metric between each node and the sink, where the distance is the expected delivery delay for the next packet added to its buffer. This delay estimate takes into account the number of packets currently in the buffer. So, as the buffer fills the expected delay also increases. Rapid then replicates data through multiple routes based on a marginal utility heuristic. Rapid estimates network parameters including the expected delay and bandwidth by averaging over a sliding window; it then communicates the estimates to the allocator.

The routing rate assigned by the allocator is only an upper bound. The actual data that is routed through the path depends on the quality of the route. For

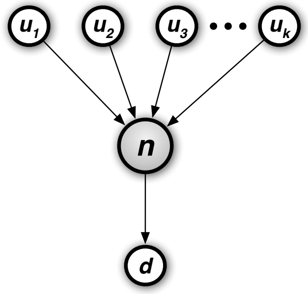

Fig. 7. A simplified example to illustrate the Tula distributed allocation algorithm. The algorithm is executed by node n, whose upstream neighbors are $u_1, u_2, \ldots u_k$

example, let nodes A and B be peers and let A's allocator assign a maximum rate at which it can route data for B. Because of changes in the network (due to mobility, interference, etc.), B may send data at a much lower rate through A; in turn A will reduce its rate allocation for B and balance the sensing rates appropriately. In other words, the routing protocol adapts to changing network conditions, that in turn affect the rate allocation. We discuss this mechanism in more detail in Section 5.1.

In addition to Rapid [2], we have also modified two other protocols, Spray and Wait [27] and a Random routing protocol; however, our experiments have not shown significant differences in overall performance. So, throughout this paper we focus only on Rapid.

4 Rate Allocation

We first describe the rate allocation algorithm by making two simplifying assumptions: *(i)* A node routes packets through only one neighbor, and *(ii)* data is only forwarded, never replicated. Later, we relax these assumptions. The network, shown in Figure 7, illustrates this simplified scenario. The allocation algorithm is described with respect to the node, n, with upstream nodes $u_1, \ldots, u_k$ routing data through n, and a single downstream neighbor d, through which data is routed toward a sink or base station.

A node's neighbor set is determined from its meeting history, and neighbors are designated as upstream and downstream based on the routing protocol's *distance* estimate. Most DTN routing protocols use a distance metric to evaluate different routes. In Rapid, the distance is the expected delivery delay. When two peers meet, the peer with a lower delivery delay is considered the downstream neighbor, and the one with the higher delivery delay is the upstream neighbor. Also, since nodes are mobile, these upstream/-downstream relationships will change over time. The impact of mobility and the implications of relaxing the definition of upstream and downstream neighbors are discussed in Sections 5.4 and 5.

Node n executes the Tula allocation algorithm in order to determine its own sensing rate r_n and the rate at which it can route data for its neighbors, $r_1, r_2, \ldots, r_k$. The allocation problem is formulated as a *Constraint Optimization Problem (COP)* with the objective of finding a max-min fair rate allocation, based on a set of input variables that are either estimated locally or exchanged between one-hop neighbors during contact opportunities. Estimates for individual variables are produced using a fixed-size history window, which may be adapted to individual applications. Tula's default window size is two weeks which we have found to work well in practice. Determining the ideal window size is left for future work.

The max-min fairness objective, as described in Section 2.3, allows us to collect as much data from as many nodes as possible.

Locally measured values, shown in Table 2, include the energy required to collect a packet of sensor data (E_s), to receive (E_r) a data packet, and to deliver (E_d) that packet to the sink. The delivery energy (E_d) includes the energy required to discover neighbors, and protocol-overhead. P is the power budget available to the node for its tasks, not including the system's idle power draw. Also included in this idle power estimate is the power required to listen for network beacons. Tula's performance depends on the use of low-power radio listening. Otherwise the energy required for radio listening would negate the advantages of adapting. All of these measurements can be obtained by a node's hardware while the device is deployed. In Tula, we use the low-level energy profiling capabilities already provided by the Eon [26] runtime system (Section 3.1).

The network variables, shown in Table 1, are exchanged either *upstream* or *downstream* through the network whenever nodes meet. The direction for each variable with respect to n is shown in the table.

Network variables are only exchanged with immediate peers and are *not* flooded across the network. These variables are described along with the COP formulation in the following paragraphs.

Objective function: The objective of the rate allocation is to achieve max-min fairness in the data collected across the nodes. A rate allocation is max-min fair if increasing any rate, r_i, requires the reduction of a lesser rate, r_j, $r_j \leq r_i$. This is also referred to as a lexicographically maximized rate assignment (Equation 1).

$$\textit{Objective: Lexicographically max.}\ \{r_1, r_2, \ldots, r_n\} \quad (1)$$

Energy conservation constraint: Perpetual operation requires that a node's harvested energy be sufficient for all sensing and networking tasks. Equation 2 ensures that n can sense and deliver its own data at a rate of r_n and receive and deliver data at a rate of r_i from each upstream neighbor u_i without exceeding its

F_i *(down)*	The fraction of node u_i's data that are sent through n
B_i *(down)*	The maximum rate at which u_i can forward data to n
O_j *(up)*	The rate at which n can route packets through its downstream neighbor, j

TABLE 1
List of inputs that are exchanged between n and its neighbors to solve the COP. Variables marked *(up)* are exchanged from n's upstream neighbors, and variables marked *(down)* are exchanged with n's downstream neighbors.

E_s	Energy required to sense a packets worth of data
E_d	Energy to deliver a packet
E_r	Energy required to receive a packet
P	Power available for sensing and routing

TABLE 2
Variables that are estimated locally by n to solve the COP

power budget, P. All variables are estimated locally using hardware instrumentation.

$$\sum_{i=1}^{k} r_i(E_r + E_d) + r_n(E_s + E_d) < P \qquad (2)$$

Downstream constraint: The total data that n can route is capped by its downstream neighbor, d. In the same way that n assigns a maximum routing rate to its upstream nodes, d likewise assigns a maximum data rate to n. Equation 3 ensures that n will never accept or collect (by sensing) data at a rate higher than the rate, O, at which it can deliver data. Node n receives the value for O from d each time they meet.

$$r_n + \sum_{i=1}^{k} r_i < O \qquad (3)$$

Upstream constraint: The objective and the first two constraints alone will result in all upstream routing rates being assigned equal to the local sensing rate. This equal division of resources is fair; however, the system will be underutilized if some upstream neighbors are unable—due to energy or bandwidth limitations—to take advantage of the allocated rate. To avoid this condition, each upstream node, u_i provides node n with an additional value: B_i is the maximum amount of data that an upstream node can send, given its energy limitations. An upstream node u_i can compute its value of B_i by solving the COP without the downstream constraint (Equation 3)

$$r_i \leq B_i \, (\forall \; i \in [1, k]) \qquad (4)$$

The COP can be solved using a well-known progressive filling algorithm [3]. The algorithm evenly adds rate to each upstream link. As rates reach their limits, they are excluded from receiving additional rate, and the process continues until either all peers are excluded or no residual energy is available. This algorithm is fast, easy to implement, and amenable to use on low-power platforms, as we show in Section 6.

Objective

Lexicographically max. $\left\{ \frac{r_1}{F_1}, \frac{r_2}{F_2}, \ldots, \frac{r_{n-1}}{F_{n-1}}, r_n \right\}$ (5)

Constraints

Energy conservation

$$\sum_{i=1}^{k} r_i(E_r + E_d) + r_n(E_s + E_d) < P \qquad (6)$$

Downstream

$$r_n + \sum_{i=1}^{k} r_i \leq \sum_{j=1}^{m} O_j \qquad (7)$$

Upstream

$$r_i \leq B_i \, (\forall i \in [1, k]) \qquad (8)$$

Fig. 8. Energy allocation problem formulation solved by node n. The goal is to estimate r_n, the local sensing rate and r_i, the rate at which n can route packets for each of its neighbors u_i

Similar optimization formulations have been used to enforce max-min fairness in sensor networks and support perpetual operation [11], [22]. These works assume that the routes are static and that network connectivity is predictable. However, in the target environment that we consider in Tula, nodes are mobile, requiring dynamic routing. In addition, the network is sparse and connectivity is unpredictable. Both these factors add significant complications to the straightforward formulation, as we discuss in the next section.

5 Incorporating Routing

The simplified Tula rate allocation makes assumptions that do not hold in practice, especially when using DTN routing to navigate a sparsely connected network. In any sparse network a node's information about the network may be incomplete and is often outdated. Consequently, each node's local solution to the COP is only a rough approximation of the global solution.

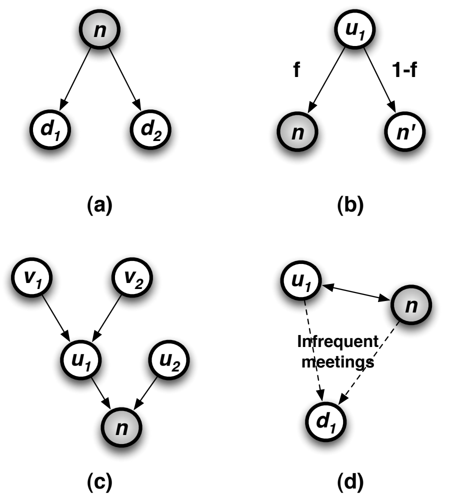

Fig. 9. Scenarios that complicate the simple Tula allocation algorithm

Additionally, Tula must address the following specific challenges: *(i)* Routing through multiple nodes: Nodes have multiple downstream paths, making it harder to determine the fraction of data to be forwarded through each neighbor, *(ii)* Replication: Multiple copies of the packet exist in the network, making the straightforward delivery estimation difficult, *(iii)* Transitive routing: A node may be routing data for several of its upstream neighbors, making the fairness estimation complicated, and *(iv)* Routing through an upstream neighbor: Although our formulation restricts routing only through downstream neighbors, in sparse environments it is not always easy to differentiate between upstream and downstream neighbors.

Below, we describe how Tula handles these complications and present the resulting modified COP in Figure 8.

5.1 Routing through multiple nodes

DTN routing algorithms often rely on multiple downstream nodes to route packets. When presented with multiple downstream options, as shown in Figure 9(a), the routing algorithm on n determines which packets will be routed through d_1 and which will go through d_2. These routing decisions are limited, however, by the routing rates allocated by the downstream nodes. Therefore, the total data that n can route is now the sum of the rates allocated by each downstream neighbor, $d_1, d_2 \ldots d_m$. We account for this limit in the COP by replacing the maximum downstream rate, O, with the maximum downstream rates allocated by m downstream neighbors, $O_1, O_2 \ldots O_m$. The new downstream constraint incorporating these variables is shown in Equation 7.

This change also impacts downstream nodes, illustrated in Figure 9(b), where n receives only a fraction, f, of the packets routed by u_1. The remaining fraction, $1 - f$, of the packets are routed through another node n'. Using the original COP, both n and n' would allocate resources to u_1 as though each were routing all of its data—clearly defeating Tula's efforts at fairness. Node n avoids this by allocating rate to u_1, proportionally to f. To accomplish this, we introduce a variable, F_i, which represents the fraction of all data routed by a node u_i through n. Equation 5 shows the modified objective function using F_i to allocate rates fairly to fractional network flows. A node receives F_i values from its upstream neighbors, which keep track of these values by maintaining a limited routing history.

In addition to ensuring fairness, the F_i values also provide a mechanism by which an upstream node's routing protocol can express demand to a downstream node's rate allocator. For example, if the routing protocol on u_1 diverts packets from a less promising n to a more promising n', the fraction of data routed by u_1 through n will decrease, signaling the rate allocator at n to reduce its allocation. Alternatively, if u_1 wants to route more packets through n, it will communicate an increased F_1 value to n. This signals n to increase its allocation, so long as the increase does not violate the fairness model.

Another result of allowing multiple downstream routes is that allocation decisions at node, n, can indirectly effect nodes that are not on n's upstream or downstream path. Consequently, solving the optimization problem optimally would, in the worst case, require each node to have complete knowledge of the network. Obtaining this global knowledge is costly in static, fully connected networks, and impossible in a sparse mobile network, like TurtleNet. Fortunately, in practice, we have found that ignoring these indirect effects does not noticeably degrade performance.

5.2 Replication

Network uncertainty can often be masked by replicating the same data over different paths. Replication adds robustness to the network and has been shown to improve delivery rates in disruption-prone environments [2], [27]. Replicating data, if not accounted for, will also unfairly skew rate allocations in much the same way as routing through multiple downstream nodes.

Consider the scenario illustrated in Figure 9(a). If n sends all of its data to d and a duplicate copy to d', n might be tempted to send $F_n = 1.0$ to both d_1 and d_2, since each downstream node routed all of its data. In this case, n receives twice its fair allocation.

Instead, a Tula node n incorporates the replication rate in its estimation of the fraction F_n. To this end, n computes F_n as the fraction of its total network transmissions including replicas. In the previous example, n sends $F_n = 0.5$ to both d_1 and d_2, assuming that both paths are favored equally.

5.3 Transitive routing

In sparse mobile networks, a node can be several hops away from the sink. In the TurtleNet testbed, for example, some devices were as much as 5 hops away from the base station. In order to deliver data, these nodes must route data transitively as shown in Figure 9(c). In the example, nodes v_1 and v_2 route through u_1, while nodes u_1 and u_2 route through n. Assuming energy and connectivity constraints are equal, and n only considers u_1 and u_2 when allocating resources, n will assign similar rates to u_1 and u_2, even though u_1 is routing 3 nodes' data and u_2 only routes for itself.

Accounting for transitivity requires no change to the COP. Rather the variable, F_i is extended to include upstream traffic. Node u_1 sends the total fraction of traffic that it routes though n, including its own and all its upstream nodes. In the example, if v_1 and v_2 route all their data through u_1, and u_1 in turn routes all its data through n, it will send $F_1 = 3$ to n. The value F_i needs to only be communicated to the immediate downstream neighbor. Nodes v_1 and v_2 communicate their respective values to u_1, and u_1 aggregates the values with its fraction to estimate F_1.

To sum up, the variable F_i represents the fraction of total packets a node sends—including its own and others' packets and accounting for replication—to its downstream neighbor.

5.4 Routing through an upstream neighbor

Under most conditions, data packets are routed toward the sink through downstream nodes; however, at times it makes sense to route data to a node that is farther from the destination, as illustrated in Figure 9(d). In the example, nodes n and u_1 meet each other frequently, but each rarely meets a shared downstream node d. According to the routing algorithm's distance metric, n is slightly closer to d than u_1. Therefore, n is downstream from u_1, and u_1 will route its data through n. Since u_1 is nearly as likely to meet d as n, node n can significantly increase the probability of delivering data in a timely manner by routing data through u_1 as well. Unfortunately, this is not permitted by our current definition of upstream and downstream nodes. We have observed this scenario often in TurtleNet, and we expect it to occur in any network with social groups and non-uniform mixing.

One solution is to relax our definition and allow nodes to be both upstream and downstream peers of each other. This solution, however, suffers from the *count-to-infinity* problem, where a node unknowingly becomes its own downstream peer. This can be solved by exchanging link information for all upstream paths, however, it significantly increases Tula's complexity—requiring, in the worst case, that all nodes maintain information about all other nodes in the network.

Therefore, we use a simpler heuristic in Tula, which has worked well in practice. We allow a node to only replicate its own data to upstream peers, but disallow forwarding other nodes' data. This simple heuristic avoids the *count-to-infinity* problem by ensuring that data is never routed back down the path from which it came.

6 Implementation

We developed two implementations of Tula: a NesC [14] version that runs on a microcontroller platform and a trace-based simulator for repeatable experimentation.

6.1 NesC implementation

The goal of the NesC implementation is two-fold. First, it demonstrates that Tula can be implemented in the memory and processor constrained microcontroller platform. Second, it allows us to measure the energy required for each component of Tula—sensing, routing data, solving the COP and exchanging meta data for the routing algorithm. We then instantiate the simulator with real energy measurements. We plan to deploy the full implementation of Tula in our TurtleNet testbed.

The NesC implementation is a fully functioning implementation running on the ShockFish Tinynode [7]. The implementation incorporates all of the design features, including the energy/rate allocator, the Eon runtime platform and the Rapid DTN layer. We adapt the Rapid implementation to run on a memory constrained platform. Rapid normally exchanges metadata about the delay of each packet. Instead, we reduce the meta-data and exchange only the per-node delay and meta-data about a short packet history. We refer to this reduced version as *RapidLite*.

We implemented the allocator in 390 lines of NesC code, and RapidLite in 1172 lines of NesC code. The Eon runtime computes the energy budget of a sensor node by keeping track of the harvested energy and the energy spent for sensing and communication.

6.2 Trace-based simulator

Simulation based on real data collected in situ from deployed systems is the most practical method for conducting realistic, fair, and reproducible comparisons between different approaches. Our simulator takes mobility and harvested energy traces from a variety of sources, including traces from our TurtleNet deployment and from UMass DieselNet [4].

The simulator periodically solves the Tula COP and performs sensing and routing based on the rates set by the COP. Connection opportunities are simulated according to the mobility traces. Nodes exchange sensed data and meta information during a connection opportunity. Sensed data is routed based on the RapidLite algorithm. The simulator assigns energy to

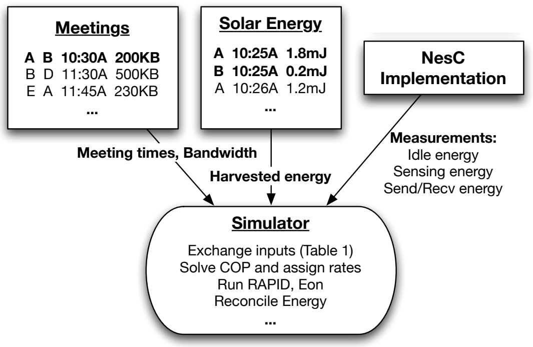

Fig. 10. Mobile connectivity, and energy traces from real deployments and energy measurements from our implementation are used to replay previous deployment scenarios using different approaches. This trace-based approach provides an evaluation of Tula that is both *realistic* and *reproducible*.

Sensor	Sense/Send ratio
GPS(Max)	2.0×10^{4}
GPS(Avg)	5.0×10^{3}
GPS(Min)	5.0×10^{3}
Accel. (ADXL330)	6.5×10^{-3}
Mag.(HMC1053)	7.2×10^{-1}

TABLE 3
Energy to sense vs. send for common sensors, and the XE1205 low-power radio

nodes according to a harvested energy trace. A node accounts for energy consumption due to processing, sensing, and communication using measurements obtained from our implementation.

7 Evaluation

Tula adapts sensing and routing rates to provide max-min fairness in the network. We compare the performance of Tula with three different kinds of approaches: *(i) Optimal*, An optimal adaptive policy based on an oracle *(ii) Static policies* that set static sensing and routing rates, and *(iii) Semi-adaptive policies* that either adapt their sensing rate or routing rate, but not both. Our evaluation compares these policies in terms of network performance, energy management, and fairness.

7.1 Methodology

Comparing perpetual systems running in the wild is a challenge—mobility, network conditions, and energy harvesting vary and are not reproducible—and the disparity between real system behavior and simulation results is often unacceptably large. We provide a *realistic* and *reproducible* evaluation of Tula by taking mobility, connectivity, and energy traces from real deployments and then replaying those deployments using different approaches.

The example in Figure 10 illustrates this approach. Meetings (e.g. node A meets node B at 10:30AM and network conditions allow the pair to send up to 200KB of data) and energy harvesting events (e.g. node A harvests 1.8mJ of energy at 10:25AM) are simulated as they occurred during the original deployment. Using our NesC implementation we measure the idle energy costs of the system as well as the energy required for individual tasks, like sensing, receiving, and transmitting data. These measurements allow us to accurately account for energy consumption throughout the simulation.

In each experiment, nodes have 512kB of storage, 250mAhr batteries, and the Tula allocator is run every 2 hours. Running the allocator more frequently increases Tula's responsiveness, but increases processing overhead. In our experience, varying this interval results in only minor changes to system performance. To evaluate alternate allocation policies, we replace the Tula COP with an allocator that enforces a static, semi-adaptive or optimal allocation policy.

Nodes can be configured to use a variety of sensors. Table 3 shows the ratio of sensing and sending cost of three different sensors: GPS, Accelerometer, and Magnetometer.

Trace collection

We conduct the trace-based simulations using three traces: TurtleNet, DieselNet and a synthetic mesh trace. The TurtleNet traces include 45 days of data from 17 tracking devices deployed on Gopher tortoises. The data contains measured solar energy (from frequent on-board voltage and current samples), connection opportunities, and the bandwidth available during a connection opportunity. More information about the design of the TurtleNet hardware can be found in [26]. The DieselNet traces are publicly available traces from the UMass DieselNet vehicular network [4] collected from 20 mobile nodes for 55 days in 2008.

The DieselNet bus traces do not contain energy harvesting information, since, like most vehicular networks, the devices rely on the vehicle's battery for power. Energy harvesting in vehicular networks promises to reduce installation time and cost. In order to add solar information to the bus data, we combine historical solar energy traces [1] with the DieselNet bus schedules to estimate the energy harvested at each bus over time. This approach is reasonable only because buses are either parked in the garage (i.e., no energy harvested) or driving on open roads with a clear view of the sky.

Finally, in order to assess the applicability of Tula to fixed networks, we also simulate a synthetic mesh network configuration of 16 nodes arranged in a 4x4 grid topology. We randomly assign energy traces from our TurtleNet data to these mesh nodes.

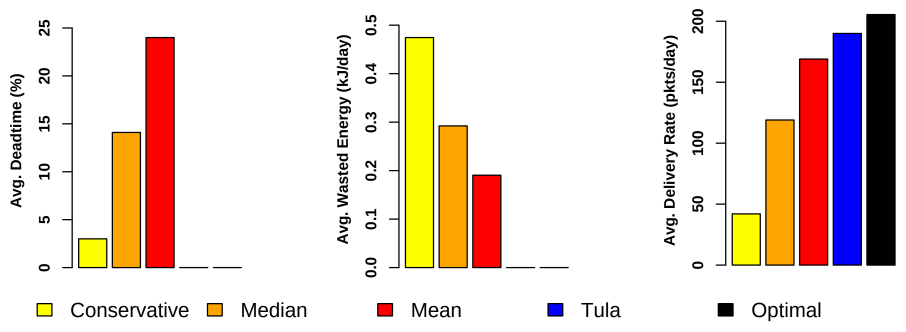

Fig. 11. Comparison of three static allocation policies, Tula and Optimal, using the TurtleNet mobility trace. The policies are compared across three metrics: battery dead time, energy wasted since the battery was full and could not charge, and average delivery rate. Tula avoids dead time and wasted energy successfully, and delivers within 92% of the oracle-based optimal policy.

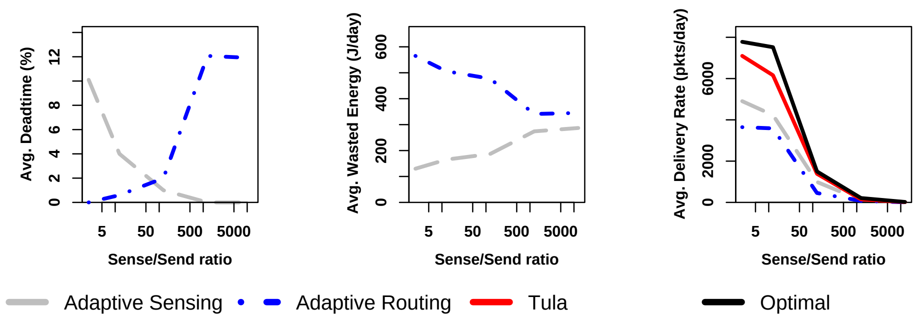

Fig. 12. Comparison of two semi-adaptive allocation polices, Tula and Optimal. The comparison is performed for different sensor applications with varying sensing-to-routing energy ratio using TurtleNet mobility data.

Optimal rate allocation using an oracle

In order to determine the optimal max-min fair rate assignment, we formulate each experimental scenario as a set of linear programs, which can be solved using a general purpose LP solver.

Our LP formulation extends the approach used by Fan et al. [11] and we have added support for temporal changes in network connectivity, rate adaptation, and storage limitations, by breaking up the linear program into discrete one-hour time segments. We use the connectivity information from each trace to determine how much data can be sent between each pair of nodes during that hour.

The solver has complete knowledge of future energy harvesting and connectivity for each time segment. The solution from the LP formulation is the maximum max-min fair rate assignment that does not sacrifice node lifetimes or data deliveries. Achieving this rate, in practice, is not feasible since it has global knowledge of the entire network and does not have to exchange information between peers. Furthermore, all connections in a segment are assumed to occur simultaneously, while in reality connections may be ordered. As a result, this "optimal" result will sometimes be more optimistic than the true optimal. Still, it provides a useful reference by which to measure system performance.

7.2 Network Performance

While Tula adaptively allocates energy for both sensing and routing, there is a wide range of alternative approaches that could be employed. In this section, we compare the performance of Tula with two classes of allocation policies: *Static* and *Semi-adaptive*.

Static rate allocation policies

One challenge in designing a static allocation policy is determining what sensing rate should be assigned to the nodes. Due to variation in energy harvesting, setting one sensing rate across all nodes will result in some nodes dying and other nodes having surplus energy. To conduct a fair comparison, we examine a range of behaviors. First, using the oracle-based optimal allocator, we determine the optimal rate allocation for each node. In a real deployment scenario, rate assignments would have to be made based on the system designer's best guess.

We examine the performance of three static rates: *conservative*, a rate that is sustainable by 90% of the nodes in the network; the *median* rate, sustainable by 50% of the nodes; and the *mean* rate, which can be achieved by only 25% of the nodes. For this experiment, nodes are configured to use the GPS sensor, and simulated using the TurtleNet mobility trace.

The results of this comparison are shown in Figure 11. We compare the performance with respect to three metrics: aggregate dead time, total wasted energy, and delivery rate. The aggregate dead time is the total time that nodes in the network have no energy. The total wasted energy is the energy that could not be stored due to limited battery size, even when solar energy was available for harvesting. Dead time is typically a result of over-utilizing energy, while wasted energy is a result of under-utilizing the available energy.

Using the *conservative* rate, nodes are dead only 3% of the time, however, on average nearly 500J of energy—enough to collect nearly 200 sensor readings—are wasted daily per node. The *mean* rate wastes much less energy, but on average nodes are dead for 25% of the time. The *median* rate provides an unsatisfying tradeoff between the two extremes, resulting in mediocre performance across all three metrics.

The results show that in a network with wide variations in energy availability and connectivity, a static scheme will perform poorly, regardless of the rate that is assigned. In contrast, by adapting per-node sensing rates, Tula is able to *completely* avoid dead time and wasted energy, resulting in 11% more data collected than using the mean rate, and within 92% of the optimal result.

Semi-adaptive rate allocation policies

Next, we make a similar comparison with two alternate adaptive policies—a policy that adapts only the routing rate or only the sensing rates.

In the *adaptive sensing* policy, routing decisions are made without any energy restrictions. However, the node adapts its sensing rates according to the remaining energy. In contrast, in the *adaptive routing* policy, sensing rates are fixed, and routing decisions are made adaptively using the energy that remains after sensing. The adaptive routing policy requires a fixed sensing rate, and we set the rate to the *conservative* rate described previously. Recall that the *conservative* rate is a rate sustainable by 90% of the nodes. Setting the static rate to other values results in similar or worse trade-offs.

Unlike the static allocation, the performance of partially adaptive rate allocation depends on the sensor. For example, if nodes only obtain accelerometer readings, the sensing cost is low enough that adapting the sensing rate does not provide benefits. Alternatively, if the sensor application obtains GPS readings, adapting

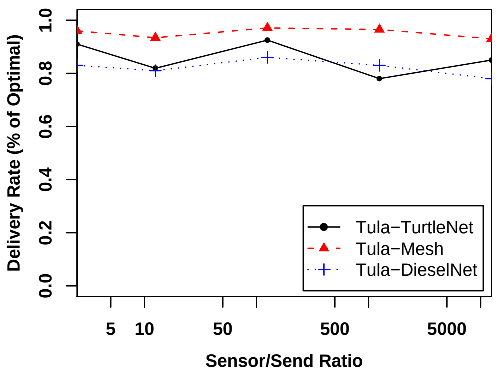

Fig. 13. Delivery rate of Tula normalized to the optimal delivery rate over three networks configurations: TurtleNet, a static 4x4 grid mesh network, and the DieselNet vehicular traces.

the sensing rate is important to ensure that a node does not exhaust its battery. Accordingly, we compare the performance of the different allocation policies for a range of sensors with varying energy requirements (as shown in Table 3).

Figure 12, illustrates the chief shortcoming of these partially adaptive approaches—their ability to adapt to changes is limited by the consumption of the static tasks. As the energy for sensing increases (left of the graph), the dead time when using adaptive routing policy increases from 0 to 12%. On the other hand, when the energy for sensing is low, more packets are sensed and routed. As a result, the average dead time of the adaptive sensing policy increases to 10%.

Both the adaptive sensing and adaptive routing policy waste between 200 J to 600 J (equivalent to 15–40 GPS readings or) daily depending on the policy and the sensor. In contrast, Tula optimizes both sensing and routing and the policy incurs no dead time and no wasted energy. In terms of delivery rate, Tula collects on average 30-50% more data than both the semi-adaptive techniques.

Network performance over DieselNet and Mesh

Figure 13 shows the delivery rates achieved by Tula for three different network configurations. Tula achieves a delivery rate of within 75% of the Optimal policy over TurtleNet and DieselNet, even without future knowledge of the harvest energy or node meeting schedules.

On a static mesh network, Tula is able to sense and deliver data within 85% of the Optimal policy for a range of sensors, showing that Tula can adapt well to different topologies. More importantly, in the absence of mobility, the rate set by the Tula allocation policy converges close to the optimal rate.

7.3 Fairness

The objective of the Tula allocation policy is to set rates such that data is sensed and delivered to the sink

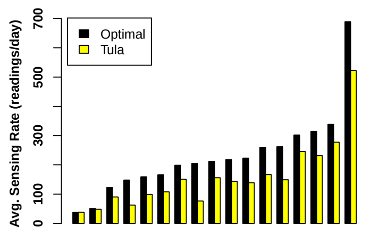

Fig. 14. TurtleNet traces: Average per-node delivery rate

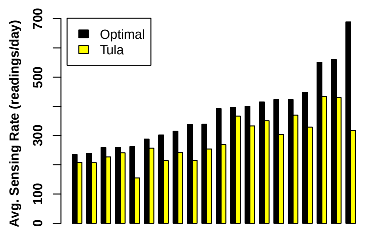

Fig. 15. DieselNet traces: Average per-node delivery rate

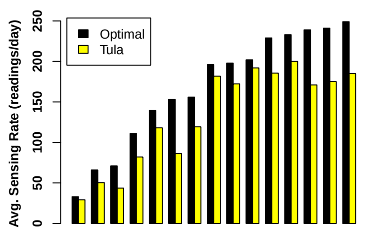

Fig. 16. Mesh: Average per-node delivery rate

at a max-min fair rate. In this section, we evaluate the fairness of Tula using two metrics. We compare the per-node delivery rate of Tula with Optimal. Recall that the optimal oracle-based rate allocator is also designed to set max-min fair rates.

Figures 14, 15 and 16 show the per-node delivery rate of Tula compared to optimal for the three network configurations: TurtleNet, DieselNet and Mesh, respectively. For all three networks, the per-node delivery rate of Tula is close to the optimal per-node rate. For example, in TurtleNet, nearly all of the nodes achieve a delivery rate within 75% of the optimal. Similar performance is seen for both Mesh and DieselNet.

7.4 Overhead

Finally, we quantify the overhead of Tula using measurements from our TinyNode-based implementation. Current draw is measured using a NI-PCI 6251 DAQ, measuring the voltage drop across a low-tolerance current-sense resistor. Simultaneously measuring the voltage across the device's battery allows us to compute the energy consumption. The measurements are shown in Table 4.

Apart from the core sensing and networking tasks, energy is incurred when periodically solving the Tula COP and computing the device's energy budget. However, both tasks only consume roughly as much energy as transmitting 2-3 packets over the device's radio. In addition to energy costs, our implementation of Tula requires 1.5kB of RAM and 22kB of additional program space in addition to the space requirements of the Eon runtime system. These size requirements are easily met by nearly all current microcontroller-based platforms.

Tula Energy/Time Overhead

Operation	Energy	Time
Solve COP	$0.9-2.3$mJ	$0.5-1.35$s
Compute Energy Budget	$1.4-1.8$mJ	$0.8-1.0$ms

Memory Overhead

RAM overhead	1.5kB
Additional code size	22kB

TABLE 4
Measurements of Tula overhead.

8 Related Work

Tula builds on a large body of previous work in several fields: challenged networking, rate allocation and fairness, mobile sensor networks, and adaptation. In many ways this synthesis is too large to cover here, so we provide the most relevant work.

8.1 Mobile sensor deployments

Previous mobile sensor deployments have shared many of the same goals as Tula. ZebraNet [32], for example, initially explored the use of in-situ sensing devices for wildlife tracking. These first devices were large (>0.5kg) and masked energy variations with large batteries and solar panels—too large for most animals to carry; however, they set the stage for future mobile sensing systems, like Tula.

Of course, mobile sensor systems are not limited to wildlife tracking. The Pothole Patrol [9] project used mobile sensors in vehicles to provide cities with valuable road-quality information. Like most vehicle-based networks, these devices receive power from the vehicles.

8.2 Low power sensor networks

Energy scarcity is a first class design concern for wireless sensor networks. Low-power hardware platforms with energy harvesting support [20], [26] as well as algorithms for estimating and predicting energy harvesting and consumption [8], [17], [26] are crucial components of all perpetual systems. Additionally, a variety of energy-aware networking techniques have been proposed for use with low-power sensors, including energy-aware clustering [31], aggregation, and traffic shaping to extend device lifetimes [25]. However, previous research on energy aware sensor networks has either focused on static network topologies, ignored networking concerns, or neglected the challenges of perpetual operation.

8.3 Challenged networks

A wealth of previous research has focused on building disruption tolerant networks with sparse connectivity.

Research in this area has provided a range of protocols [2], [16], [21], [27], which opportunistically forward and replicate packets to mobile peers. Most DTN solutions have targeted vehicular [5] and personal device networks [15]. Whether tapping into a vehicle's battery or relying on user-facilitated recharges, previous solutions assume a steady and unlimited energy supply, and neglect the challenge of energy scarcity which is central to any untethered system.

8.4 Fair network rate allocation

A variety of fairness policies have been proposed [3], [18], along with many techniques for enforcing those policies in wireless networks [13], [29], [33]. Of these approaches, the most closely related work, which has been concurrently with Tula, provides both centralized and distributed algorithms for enforcing max-min fairness in networks that have rechargeable sensors [11], [22]. However, the authors assume that the routes in the network are static, and that the energy profile of a node is known in advance. Tula enforces max-min fairness in networks with unpredictable network connectivity and dynamically changing energy constraints.

A myriad of techniques aim to improve performance and enable new applications by providing additional coordination across traditionally independent network layers [6], [28], when legacy abstractions fail to meet the needs of emerging systems and environments. Tula is also a cross-layer approach providing a tight link between the application and network in order to address the combined challenges of mobility, heterogeneity and perpetual operation.

9 Discussion

The model we have used for sensing and routing is intentionally simple and straightforward and works well in practice. However, Tula also has its limitations. For example, while using local optimizations to approximate a global optimization improves efficiency, it may introduce additional prediction error. Also, using a fixed history window to determine upstream/downstream relationships means that pathological mobility patterns exist, under which Tula's routing will perform poorly (e.g., oscillation).

While potentially problematic for Tula, these conditions have not occurred in any of the real mobility patterns we have studied. We expect Tula's pathological cases to be rare in real mobile networks, although the solutions may provide academically interesting opportunities for future work.

We have also focused on a simple network model—each node streams raw data to a sink. There are other models used by sensor network applications including aggregation [10] and querying [24]. Incorporating these alternative models in Tula, requires the system to estimate the effect that aggregation and querying have on network load. In the case of querying, network load and the flexibility with which Tula can adapt would also depend on the nature of the query itself. Supporting aggregation, would require the system to estimate the amount of compression achieved by aggregation at each hop in the network.

Finally, this work could also be extended to support other notions of fairness (e.g., proportional fairness) and to allow timely or critical data to be given higher priority in the network. We expect these extensions to be straightforward; however, in the interest of brevity and simplicity, we leave them as future work.

10 Conclusions

In this paper, we present Tula, a system which balances sensing with packet delivery for energy harvesting mobile sensor networks. Tula represents a first step in managing the resources of constrained nodes, balancing sensing and communication, while maintaining a cooperative system for delivering data. Our evaluation of Tula, using mobility and energy traces from our TurtleNet deployment, shows that Tula collects and delivers data within 75% of an optimal oracular policy. In addition, we have shown that Tula successfully enforces a max-min fairness policy and is suitable for use on low power sensing platforms. As the scale and complexity of mobile sensing systems increases, proven techniques for estimating, predicting, and efficiently sharing network and energy resources will continue to be an essential key to their success.

References

[1] http://weather.cs.umass.edu/.

[2] A. Balasubramanian, B. N. Levine, and A. Venkataramani. DTN Routing as a Resource Allocation Problem. In *Proc. ACM Sigcomm*, August 2007.

[3] J.-Y. Boudec. Rate adaptation, congestion control and fairness: A tutorial, 2000.

[4] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine. MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networks. In *IEEE INFOCOM*, April 2006.

[5] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine. MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networks. In *Proc. IEEE INFOCOM*, April 2006.

[6] M. Conti, G. Maselli, G. Turi, and S. Giordano. Cross-layering in mobile ad hoc network design. *Computer*, 37(2):48–51, Feb 2004.

[7] H. Dubois-Ferriere, R. Meier, L. Fabre, and P. Metrailler. TinyNode: A comprehensive platform for wireless sensor network applications. In *Proceedings of the fifth international conference on Information processing in sensor networks (Poster)*, pages 358–365, Nashville, TN, USA, April 2006.

[8] A. Dunkels, F. Osterlind, N. Tsiftes, and Z. He. Software-based on-line energy estimation for sensor nodes. In *EmNets '07: Proceedings of the 4th workshop on Embedded networked sensors*, pages 28–32, New York, NY, USA, 2007. ACM.

[9] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan. The pothole patrol: using a mobile sensor network for road surface monitoring. In *MobiSys '08: Proceeding of the 6th international conference on Mobile systems, applications, and services*, pages 29–39, New York, NY, USA, 2008. ACM.

[10] K.-W. Fan, S. Liu, and P. Sinha. Structure-free data aggregation in sensor networks. *IEEE Transactions on Mobile Computing*, 6(8):929–942, 2007.

[11] K.-W. Fan, Z. Zheng, and P. Sinha. Steady and fair rate allocation for rechargeable sensors in perpetual sensor networks. In *SenSys '08: Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 239–252, New York, NY, USA, 2008. ACM.

[12] A. Fioravanti-Score, S. V. Mitchell, and J. M. Williamson. Use of Satellite Telemetry Technology to Enhance Research and Education in the Protection of Loggerhead Sea Turtles. In *19th Annual Symposium on Sea Turtle Biology and Conservation*, 1999.

[13] V. Gambiroza, B. Sadeghi, and E. W. Knightly. End-to-end performance and fairness in multihop wireless backhaul networks. In *MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking*, pages 287–301, New York, NY, USA, 2004. ACM.

[14] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesc language: A holistic approach to networked embedded systems. In *PLDI '03: Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation*, pages 1–11, New York, NY, USA, 2003. ACM.

[15] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot. Pocket Switched Networks and Human Mobility in Conference Environments. In *Proc. ACM Workshop on Delay-Tolerant Networking*, pages 244–251, Aug. 2005.

[16] S. Jain, K. Fall, and R. Patra. Routing in a Delay Tolerant Network. In *Proc. ACM SIGCOMM*, pages 145–158, August 2004.

[17] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava. Power management in energy harvesting sensor networks. *ACM Transactions on Embedded Computing Systems*, May 2006.

[18] H. Kushner and P. Whiting. Convergence of proportional-fair sharing algorithms under general conditions. *Wireless Communications, IEEE Transactions on*, 3(4):1250–1259, July 2004.

[19] A. Lachenmann, P. J. Marrón, D. Minder, and K. Rothermel. Meeting lifetime goals with energy levels. In *Proc. of the 5th ACM Conference on Embedded Networked Sensor Systems*, pages 131–144, 2007.

[20] K. Lin, J. Hsu, S. Zahedi, D. C. Lee, J. Friedman, A. Kansal, V. Raghunathan, and M. B. Srivastava. Heliomote: Enabling long-lived sensor networks through solar energy harvesting. In *Proceedings of ACM Sensys*, November 2005.

[21] A. Lindgren, A. Doria, and O. Scheln. Probabilistic Routing in Intermittently Connected Networks. In *Proc. Workshop on Service Assurance with Partial and Intermittent Resources*, August 2004.

[22] R.-S. Liu, P. Sinha, and C. E. Koksal. Joint energy management and resource allocation in rechargeable sensor networks. In *Proceedings of IEEE Infocom*, 2010.

[23] K. Lorincz, B.-r. Chen, J. Waterman, G. Werner-Allen, and M. Welsh. Resource aware programming in the pixie os. In *SenSys '08: Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 211–224, New York, NY, USA, 2008. ACM.

[24] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tinydb: an acquisitional query processing system for sensor networks. *ACM Trans. Database Syst.*, 30(1):122–173, 2005.

[25] C. Schurgers and M. Srivastava. Energy efficient routing in wireless sensor networks. *Military Communications Conference, 2001. MILCOM 2001. Communications for Network-Centric Operations: Creating the Information Force. IEEE*, 1:357–361 vol.1, 2001.

[26] J. Sorber, A. Kostadinov, M. Garber, M. Brennan, M. D. Corner, and E. D. Berger. Eon: A Language and Runtime System for Perpetual Systems. In *Proc. ACM SenSys*, Sydney, Australia, November 2007.

[27] T. Spyropoulos, K. Psounis, and C. Raghavendra. Spray and Wait: An Efficient Routing Scheme for Intermittently Connected Mobile Networks. In *Proc. ACM Workshop on Delay-Tolerant Networking*, pages 252–259, Aug. 2005.

[28] V. Srivastava and M. Motani. Cross-layer design: a survey and the road ahead. *Communications Magazine, IEEE*, 43(12):112–119, Dec. 2005.

[29] L. Tassiulas and S. Sarkar. Maxmin fair scheduling in wireless networks. *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 2:763–772 vol.2, 2002.

[30] A. Vahdat and D. Becker. Epidemic Routing for Partially-Connected Ad Hoc Networks. Technical Report CS-2000-06, Duke University, July 2000.

[31] O. Younis and S. Fahmy. HEED: A hybrid, energy-efficient, distributed clustering approach for ad-hoc sensor networks. *IEEE Transactions on Mobile Computing*, 4(4), October 2004.

[32] P. Zhang, C. M. Sadler, S. A. Lyon, and M. Martonosi. Hardware design experiences in zebranet. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 227–238, New York, NY, USA, 2004. ACM.

[33] J. Zhu, K.-L. Hung, and B. Bensaou. Tradeoff between network lifetime and fair rate allocation in wireless sensor networks with multi-path routing. In *MSWiM '06: Proceedings of the 9th ACM international symposium on Modeling analysis and simulation of wireless and mobile systems*, pages 301–308, New York, NY, USA, 2006. ACM.

Jacob Sorber is a Postdoctoral associate at Dartmouth College, with research interests broadly in mobile computing, embedded systems, and sensor networks. Current research topics include efficient and usable security and privacy for mobile health-related systems (mHealth), and energy management for RFID-scale devices. He earned a Ph.D. from the University of Massachusetts Amherst.

Aruna Balasubramanian is a Postdoctoral fellow at University of Washington. She received her Ph.D from the University of Massachusetts Amherst. Her research interests are broadly in systems and networking, with a focus on mobile and vehicular networks. Her current research work involves architecting energy efficient and high performance mobile systems with an emphasis on end-user driven design and evaluation. Her postdoctoral work is supported by the Computing Innovation Fellowship. She is the recipient of a Microsoft Graduate Research Fellowship.

Mark D. Corner is an expert in the areas of mobile device operating systems, power management, networking, file systems, and security. He is a tenured Associate Professor in the Department of Computer Science at the University of Massachusetts Amherst and has been there since 2003. At UMass he teaches the Operating Systems course and a course on Usability. He has authored dozens of publications on mobile systems, holds two patents, and has been awarded millions of dollars in federal and industrial research grants. He was selected for DARPA's prestigious Computer Science Study Panel in 2009. He was the recipient of an NSF CAREER award in 2005, Best Paper awards at USENIX FAST 2007, ACM Multimedia 2005, and ACM Mobicom 2002. Prof. Corner also serves on the editorial board of IEEE Pervasive. He has chaired the mobile systems community's premier conference MobiSys, and the premier workshop, HotMobile. Mark Corner is a co-founder, and VP of Engineering at Fiksu Inc., which provides advertising optimization for mobile applications.

Joshua Ennen is a visiting professor of Biology at Maryville College, Maryville, Tennessee. He previously work as a Wildlife Biologist for U. S. Geological Survey. He obtained his Bachelor of Arts from Maryville College, Maryville, Tennessee. His Master of Science was obtained from Austin Peay State University, Clarksville, Tennessee. He received his Doctor of Philosophy from the University of Southern Mississippi. His research has focused mainly on reproductive ecology, movement behavior, systematics, and population genetics of chelonians including Gopherus polyphemus, Gopherus agassizii, Graptemys gibbonsi, Graptemys pearlensis, Graptemys flavimaculata, Graptemys oculifera, and Sternotherus minor peltifer.

Carl Qualls is an Associate Professor in the Department of Biological Sciences at the University of Southern Mississippi. He earned his PhD from the University of Sydney. Carl is a herpetologist with research interests in ecology, conservation biology, and life history evolution. Current research topics include reproductive ecology, phylogeography, and conservation genetics of the gopher tortoise, and a study of translocation methods for endangered frogs. Papers from these and other projects appear in numerous biological and herpetological journals.