An Energy-conscious Transport Protocol for Multi-hop Wireless Networks

N. Riga I. Matta
Computer Science
Boston University
{inki, matta}@cs.bu.edu

A. Medina C. Partridge J. Redi
BBN Technologies
Cambridge, MA, USA
{amedina, craig, redi}@bbn.com

ABSTRACT

We present a transport protocol whose goal is to reduce power consumption without compromising delivery requirements of applications. To meet its goal of energy efficiency, our transport protocol (1) contains mechanisms to balance end-to-end vs. local retransmissions; (2) minimizes acknowledgment traffic using receiver regulated rate-based flow control combined with selected acknowledgements and in-network caching of packets; and (3) aggressively seeks to avoid any congestion-based packet loss. Within a recently developed ultra low-power multi-hop wireless network system, extensive simulations and experimental results demonstrate that our transport protocol meets its goal of preserving the energy efficiency of the underlying network.

1. INTRODUCTION

Motivation: Multi-hop wireless networks are plagued with unique challenges: contention for the wireless medium, time-varying topology due to the variable quality of links or mobility, and power constraints imposed by battery lifetimes. The focus of this paper is the last challenge: power constraints. We seek to minimize the usage of energy so as to extend the lifetime of both individual nodes and the network as a whole, while meeting the requirements of applications.

While energy efficiency has been recognized as a challenge for some time [38], until recently, very little progress has been made. In the past couple of years we have begun to see efforts to reduce the energy consumed in the radio circuit, which draws most of the battery power.

Low power [37], adjustable power [41], as well as two-stage receiver radios [31] have been designed and implemented in an effort to minimize the energy consumption while transmitting and receiving packets. In order to minimize the energy spent on "idle" listening of the channel, new MAC protocols have been proposed that implement coordinated wake-up schedules that allow nodes to be turned off for long periods of time. Also in an effort to minimize network control (e.g. routing) traffic, controlled scoping of topology information [30] has been proposed.

As an example, the JAVeLEN system [11,26] achieves dramatic results demonstrating networks that consume 100 times less energy for the same effective network goodput, compared to a typical 802.11 multi-hop wireless network running the OLSR routing protocol.

These type of highly energy-efficient systems present a challenge to network protocol designers. All high-layer protocols need to be examined and, if necessary, redesigned to make sure they are conservative in their transmissions. A parsimonious media-access protocol is of limited benefit if applications choose to be chatty and consume power with low-value transmissions.

In this paper we examine the problem of designing a transport protocol that is energy conserving and is suitable for deployment over energy-conscious systems. Extensive studies (reviewed in Section 7) have demonstrated the inadequacy of TCP to serve as a transport protocol in wireless environments. Later attempts to enhance or redesign TCP for wireless scenarios are mainly focused on improving performance metrics such as goodput and delay, but not on minimizing energy consumption.

Designing an energy-conscious transport protocol is simultaneously simple and hard. The problem is simple because, for practical purposes, we can use a simple metric to minimize: the total number of node transmissions for each packet, or each conversation.[1] The problem is hard because very little is known about minimizing transmissions across multiple nodes. There is plenty of research work on minimizing transmissions end-to-

[1]This metric admittedly ignores energy costs for computation and memory in the nodes. The usual logic for ignoring these costs is that the radio consumes far more energy.

CoNEXT'07, December 10-13, 2007, New York, NY, U.S.A.

end, especially limiting the number of acknowledgments sent, but almost nothing on intermediate transmissions. And the one key piece of work in the area, Ludwig's work on the interaction of end-to-end and hop-by-hop retransmissions [22], gives a depressing result: namely in a world where one needs both end-to-end and hop-by-hop retransmissions, it is very easy to have the two transmission mechanisms interact to cause more, rather than less, total transmissions. Our transport protocol coordinates end-to-end and hop-by-hop retransmissions by allowing the ends of the connection to explicitly control the amount of local retransmission effort, based on application's reliability requirement. JTP's objective of minimizing the number of total transmissions is not only beneficial in energy-constrained networks, but in any wireless network since limiting the overall number of link-layer transmissions effectively increases the network-wide capacity.

Our Contribution: Our transport protocol mediates between an application's need to share information of varying importance over the network, and the system's goal of minimizing energy expenditure per successfully delivered bit. Given that JAVeLEN is the state of the art in energy conserving network systems, our protocol uses JAVeLEN as the underlying architecture and thus we will refer to it as JTP (JAVeLEN Transport Protocol). Although JTP is implemented to work within JAVeLEN, it is designed to operate within any wireless (or wireline) architecture that provides an interface for JTP to both, control the number of node retransmissions made by the media-access protocol, and get at least indication of the available capacity of the channel, and of the packet loss rate. JTP's architecture strives to maximize the modularity of the system. Information is propagated through packet headers and all the necessary hop-by-hop operations are performed by a separate module that is used by the MAC.

In this paper, we present a summary of our design and the novel features of JTP as follows:

- To the best of our knowledge, JTP is the first multi-hop wireless *end-to-end* transport protocol designed to perform hop-by-hop *soft-state* operations to improve (goodput and energy) performance while preserving the *end-to-end principle* [29] (Section 2).

  JTP employs mechanisms akin to the Dynamic Packet State [34]—using packet headers to propagate information—to avoid maintaining per-flow state and maintain the modularity of the system.

- JTP exploits any energy-gain opportunities provided by the applications. Historically, transport protocols have offered a particular reliability/QoS model and the application's task was to pick the transport protocol whose model most closely met the application's needs (e.g. UDP, TCP, ITP [25], RTP [13]). JTP is designed to act as a "generic" transport protocol tailored by the application based on its specific QoS semantics. JTP uses the tolerance of the application to losses and limits the network's effort in trying to deliver a packet, based on the packet's individual importance as well as current energy costs (Section 3).

- In JTP, the receiver is fully responsible for controlling all transmission parameters; connection's sending rate, retransmission requests for missing/lost packets, as well as the frequency of such controls. To the best of our knowledge, JTP is the first transport protocol that supports *variable destination-controlled feedback* trying to keep feedback as low as the stability and reliability of the network permits (Section 5).

- JTP implements a caching mechanism which enables intermediate nodes along the path of a JTP connection to temporarily store traversing packets.[2] This enables the recovery of lost packets as close to the receiver as possible. These pipelines of caches along paths generalize the single-level caching often employed in cellular-type (single wireless hop) networks [5]. Although a system that supports symmetric routes between hosts, like the JAVeLEN system, would exploit caching benefits to its fullest, the opportunistic design of the caching system would seize any chance for locally recovering lost packets, without interfering with the end-to-end semantics of each connection.

  To allocate bandwidth fairly among flows in the presence of in-network caching and retransmissions, a JTP sender backs off its sending rate to account for "internal" retransmissions triggered from caches on its behalf (Section 4).

- JTP also employs a (congestion-avoidance) rate-based flow control—using ATM-like explicit rate feedback from the network—in an attempt to eliminate energy wastage associated with congestion-induced packet drops.

- JTP is implemented as "shared code" that can be run either in simulation or on real radio nodes. Results from simulation and from a prototype of JAVeLEN radios (Section 6) confirm the premise of JTP in reducing the energy consumed per delivered bit.

[2] We note that efficiency achieved by caching and repairing errors earlier using such in-network caching does not contradict the end-to-end argument of system design [29]—the source does *not* delete its copy of a packet until it gets an acknowledgment from the final destination that it has successfully received the packet. Furthermore, the soft-state nature of caches provides resilience to route changes.

2. JTP DESIGN

As we sought to design a protocol that minimizes the total number of node transmissions required to deliver application data, we chose to decompose that goal into three subgoals:

(1) Minimize end-to-end retransmissions. Ludwig's work showed that we needed to strike a balance between end-to-end (source) and local retransmissions. End-to-end retransmissions effectively waste all the energy already expended on getting the packet at least part way to the destination by the initial transmission. So, while occasional retransmissions from the source are required (e.g. due to intermediate node failure or topology changes), we sought to do everything we could to retransmit a lost packet from the farthest downstream, intermediate node along the packet's path which had received the packet successfully.

(2) Minimize acknowledgments. Acknowledgments are, often, pure overhead—they carry no application data. Yet, they consume roughly as much energy as a data transmission. So, consistent with reliability and other goals, we would like to minimize acknowledgments.

(3) Avoid congestion loss in the nodes. By design, a classic TCP-like congestion control induces packet drops, since loss is the only sign of congestion. The energy expended by packets that are discarded, simply to signal congestion, is wasted. In a world where energy is the key metric, congestion control should consume less energy than what discarding a packet implies. Assuming that JTP is effective in only transmitting data when necessary, JTP aims to avoid congestion instead of controlling it.

The result of these goals was the JTP architecture shown in Figure 1. JTP uses rate-based transmission controlled by the destination. Intermediate wireless nodes report on their condition in data packet headers. A Path Monitor and Path Controller at the destination collect this path performance data and adjust data rates to avoid congestion. Intermediate nodes retransmit packets on a per hop basis, cache packets, and examine end-to-end acknowledgments. If an acknowledgment indicates a packet was lost further along the path, the node will retransmit this packet, thereby avoiding an end-to-end retransmission (a la the work of Balakrishnan *et al.* [5]). Based on this design, we next look at how JTP actually works.

Before we delve into the details of JTP, we provide a brief description of the JAVeLEN system. JAVeLEN deploys a TDMA MAC which orchestrates nodes' transmissions by using pseudo-random schedules, providing collision-free access to the channel and allowing nodes to turn off their radios when they are not in use. Each node also keeps statistics about link transmissions and idle slots in order to provide estimates of the available transmission rate and of the packet loss rate on every link. JAVeLEN uses an energy conserving link-state routing algorithm [30], that provides each node with a local, possibly inaccurate, view of the network's topology.

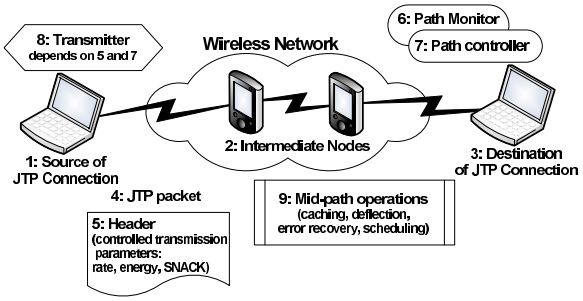


Figure 1: Elements of JTP

2.1 A Packet-based View of JTP

We begin by describing JTP in terms of packets—data and acknowledgment packets—and how they flow between source and destination. Due to space restrictions a detailed description of the packet format is omitted —the reader is referred to [27] for more details.

2.1.1 Data Packets

Data packets travel from the source to the destination of a JTP connection. In JTP, data packets contain three novel fields: available rate, loss tolerance, and energy budget.

• *Available rate*: The available rate of a link, from a node to its neighbor, represents its current available *transmission* capacity—in a TDMA MAC, like the JAVeLEN MAC, that available rate is determined by the current rate of unused (idle) time slots during which the neighbor is awake for reception; in CSMA/CA networks, a method similar to [20] can be used to estimate the available bandwidth. At each node visited, a packet is stamped with the minimum available rate collected so far along the path of the JTP connection As described in Section 5, this available rate is used by the flow controller at the destination to update the sending rate of the source. Note that due to retransmissions that may be required to get the packet to the next hop, a packet may consume more than one MAC-level transmission slot. So the available rate value must be normalized by the average number of MAC-level transmissions. Ludwig [22] pointed out that allowing multiple MAC-level transmissions can increase packet delay and also reduce the effective bandwidth that an application perceives. As we will see later, by allowing applications to control the number of MAC-level transmissions per link, JTP effectively gives applications control over the delay and effective bandwidth on every link.

• *Loss tolerance*: A source node encodes the desired end-to-end loss tolerance in packet headers. Each node along the path pre-computes the maximum number of transmission attempts to the next hop, given the re-

maining length of the path (known from the node's view of the topology) and packet's loss tolerance. The packet is dropped if this pre-determined maximum number of local attempts is exceeded. As described in Section 3, before forwarding the packet, the node updates the loss tolerance field so any left-over attempts (from the pre-determined maximum number) do *not* get used downstream, thus reducing the variability in energy consumption across nodes along the path.

• *Energy budget*: The source initially assigns each packet an energy budget value based on the energy the network would *typically* expend to deliver the packet successfully. A packet is dropped whenever the energy used exceeds the energy budget of the packet. This approach provides an energy-conscious mechanism for dealing with routing loops (as opposed to the traditional hop-count TTL) and, as we elaborate later, in conjunction with the loss tolerance field, creates a sturdy way to manage the energy expenditure per packet.

The available rate field addresses the goal of avoiding congestion loss. Because JAVeLEN provides a practically collision-free MAC layer, if the JTP flow controller ensures that it does not drive the available rate to zero, congestion-induced losses are avoided.[3]

The energy budget and loss tolerance fields manage the expenditure of energy per packet. By limiting the effort of each node to successfully deliver a packet, the loss tolerance field bounds the total transmission energy spent by each node. The energy budget on the other hand, sets an upper bound on the energy that the whole network might expend to deliver a packet to the destination.

2.1.2 Acknowledgment Packets

JTP ACK packets carry data acknowledgments, as well as transmission rate and energy budget information from the receiver to the sender. The rate at which ACKs are fed back to the sender is regulated by the receiver, based on the stability conditions of the path. For a stable path, a minimum feedback rate is determined by the application based on its requirements—for example, an application with a more stringent delay requirement would require a higher feedback rate to achieve a more timely recovery of missing data. A lower feedback rate, if tolerated by the application, allows JTP to *aggregate* ACK information in a single packet, thus reducing feedback load.

It is well known that rate-based flow control is vulnerable to the loss of feedback messages. As we elaborate in Section 5, JTP overcomes this problem by having the receiver inform the sender of its feedback rate. So if the sender does not get an ACK within the expected feedback delay, it backs off its transmission rate. Furthermore, short-term, and significant, variations in path's conditions (available rate or energy used) would be detected by the path monitoring function at the receiver, thus triggering an early feedback.

In addition to reporting rate and energy information, an ACK packet carries both positive cumulative and negative selective acknowledgments. Each intermediate node on the path examines the SNACKs and retransmits missing packets if these packets are in the intermediate node's cache—if they are, the node appropriately modifies the ACK packet so the sender is explicitly informed of such in-network retransmissions done on its behalf (cf. Section 4). Caching, and acknowledgments, are discussed in more detail in Section 4.

2.2 JTP Components

As mentioned earlier in Section 1, although JTP is an *end-to-end* transport protocol, it also performs hop-by-hop operations to improve its performance. In order to alleviate the overhead of the hop-by-hop operations and increase the protocol's efficiency, JTP is divided into two main components: the *end-to-end JTP* (eJTP) and the *hop-by-hop (or, intermediate) JTP* (iJTP).

2.2.1 End-to-end JTP

The functionalities of eJTP are further categorized into *application-specific* and *network-specific* modules. This modularity makes JTP a more versatile transport protocol. In this work, we focus on bulk data transfers over multi-hop wireless networks.

• The *application-specific* module of eJTP is responsible for fragmentation and reassembly of application's data. Moreover, it expresses the QoS requirements (e.g. reliability level) to the network-specific module to influence in-network packet handling decisions, flow control, and retransmission requests only for those missing packets that are important to the application.

• The *network-specific* module manages all JTP connections and implements the path monitor and the (congestion-avoidance rate-based) controller of per-packet transmission parameters.

2.2.2 Hop-by-hop JTP

iJTP performs all the mid-path (intermediate) *cache-related* and *soft-state* operations. It manages the local cache of every node by ensuring that only valuable data packets are stored, data packets indicated in SNACKs are retransmitted, and cached data packets are evicted based on the selected caching policy.

Soft-state per-packet operations performed by iJTP at each node, include:

• *Update the available rate field*: iJTP is responsible for acquiring from the MAC layer an estimate of the avail-

[3]Even if the underlying MAC does not provide collision-free access, JTP's operation will not be affected, since collisions would only increase the link loss experienced by packets, thus increasing the number of link-layer retransmissions per packet and effectively reducing the measured available bandwidth, which in turn forces the sources to back off.

able rate to every neighbor, as well as an estimate of the packet loss rate on that link. It uses this information to estimate the *effective* available rate to each neighbor. iJTP stamps each passing packet with the lowest effective available rate (throughput) observed thus far along the path.

• *Update the loss tolerance field*: Based on the loss tolerance carried in the packet's header and the link's estimated packet loss rate, iJTP sets the number of data transmission attempts on that link, as we elaborate in Section 3. The loss tolerance field is then updated to reflect its value for the remainder of the path.

Besides the fact that iJTP must process all JTP packets that pass through a node, the above described soft-state operations require the crafting of cross-layer interactions with the MAC layer. In order not to compromise the performance of a node, by redundant copying, context switching, and message passing between JTP and the MAC layer, we implemented iJTP as a separate loadable plug-in module of the MAC protocol. iJTP is invoked whenever a packet is received or about to be transmitted over the air interface.[4] The reader is referred to [27] for a more detailed description of iJTP operations.

3. ADJUSTABLE RELIABILITY FOR ENERGY CONSERVATION

Not all applications (e.g. voice, video, images [25]) require full reliability to perform well. Given reliability targets from applications (provided by the *application* module), and knowledge of packet loss rates (provided by the MAC layer [5]), JTP adjusts the effort it puts into delivering each packet, seeking to expend only enough energy to deliver "needed" packets. Specifically, iJTP controls the number of node transmission attempts on a per packet basis. In JTP, the reliability level is expressed in terms of the loss tolerance percentage (e.g. 10% of packets can be lost) encoded in the header of each packet.

Let l_{e2e} be the end-to-end loss tolerance requested by the application. Let $n_i,\ i \in [0, H]$ be the nodes on the path from the source n_0 to the destination n_H, where H is the total number of links on the path. Let $q_i,\ i \in [0, H-1]$ denote the probability that a packet sent by node n_i will be successfully delivered to the next node n_{i+1}. In order to satisfy the end-to-end loss tolerance of the application, the following equation should hold:

$$l_{e2e} = 1 - \Pi_{i=0}^{H-1}\ q_i \tag{1}$$

The value of q_i changes depending on the number of node transmission attempts indicated to the MAC by iJTP. Let p_i denote the probability that a single node transmission from n_i to n_{i+1} fails, and let M_i denote the total number of node transmission attempts requested for a packet on link (n_i, n_{i+1}). Then $q_i = 1 - p_i^{M_i}$, from which we can compute M_i as:[6]

$$M_i = \max(1, \min(\lceil \frac{\log(1-q_i)}{\log(p_i)} \rceil, MAX\_ATTEMPTS)) \tag{2}$$

where $MAX\_ATTEMPTS$ is the maximum number of link transmissions that the MAC allows. The challenge is to dynamically adjust the values of M_i for each packet in a flow so as to satisfy the desired l_{e2e}.

If the length of the path to the destination is known, the values for q_i's can be directly computed from equation (1), and encoded in the headers of packets. However, in a network setting where the topological views at the nodes are typically not accurate, the path length is estimated based on a node's current view. Therefore, JTP carries out the computation of q_i's at each node, as the packet travels toward the destination, thus using increasingly more accurate views.

Let l_{ti} be the loss tolerance that is encoded in the packet when received by node n_i. Let H_i be the number of links from n_i to the destination. Before the node transmits the packet to its next-hop, it updates the loss tolerance field as follows:

$$\Pi_{j=i}^{i+H_i-1} q_j = 1 - l_{ti} \Rightarrow q_i \Pi_{j=i+1}^{i+H_i-1} q_j = 1 - l_{ti} \Rightarrow \quad l_{t(i+1)} = 1 - \frac{1-l_{ti}}{q_i} \tag{3}$$

Although there are many different strategies that might be employed to compute q_i on each link—e.g. imposing higher successful delivery requirement on less loaded links or on nodes with higher available energy—in this paper we assume that JTP attempts to assign the same $q_i = q$ for all the links. From Equation (1) we get:

$$q = (1 - l_{ti})^{\frac{1}{H_i}} \tag{4}$$

By dynamically adjusting the hop-by-hop success probability experienced by each packet, the end-to-end reliability requirements are met even if the topological views at different nodes are inconsistent, or the path changes. Moreover, by assigning a loss tolerance target to each individual packet, JTP enables the application to prioritize its packets (e.g. video frames of varying importance).

We tested JTP under different reliability levels. Three different levels are considered 0% (jtp_0), 10% (jtp_{10}) and 20% (jtp_{20}).

[4] Although within the JAVeLEN system, iJTP resides in the MAC, iJTP operations can be performed by eJTP using an overlay type of architecture.

[5] The MAC layer keeps statistics about successful packet transmissions and estimates the average packet loss rate experienced over each link.

[6] The success probability is equal to $\sum_{k=0}^{M_i-1} p_i^k(1-p_i) = (1-p_i^{M_i})$.

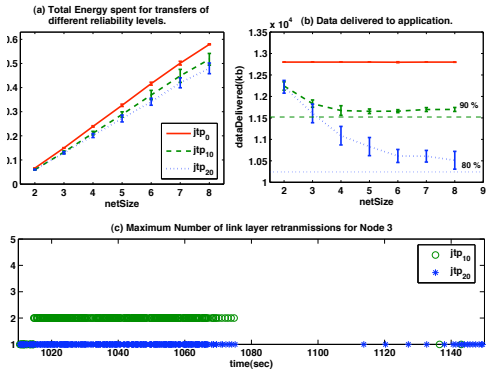


Figure 2: Different reliability levels: jtp_0, jtp_{10}, jtp_{20} have loss tolerance of $0\%, 10\%$ and 20%, respectively.

Figures 2(a) and (b) demonstrate that by neither overachieving (e.g. TCP-like full reliability for all), nor underachieving (e.g. UDP-like no reliability for all), JTP manages to save energy and still satisfies an application's requirements, denoted by the horizontal dotted lines in Figure 2(b).[7] (See Section 6 for simulation details.) Figure 2(c) shows the maximum number of node retransmissions set by iJTP for each packet at the third node along a 4-node path. Packets that request higher reliability are retransmitted more times by the link layer in cases when the link quality is not good enough, and thus JTP directly affects the amount of network's effort in delivering each packet.[8]

4. CACHING

JTP employs in-network caching of data packets to avoid end-to-end (source) retransmissions as much as possible. Caching can be viewed as a second line of defense if all MAC-level attempts fail; for example, due to a temporary excessive degradation in link quality. Moreover packets might be lost downstream due to buffer overflows or route failures. Employing opportunistic, soft-state caching would avoid unnecessary end-to-end retransmissions. The destination, which has better knowledge of the application's requirements than mid-path nodes, may request any of these cached packets that it is missing, and are still needed by the application.

Upon receiving a traversing ACK packet, a node checks the Selective Negative ACK (SNACK) field to determine whether any packet(s) requested for retransmission exists in the node's local cache. Requested packets found in the cache are forwarded downstream toward the destination.

Besides the SNACK field, an ACK packet header also contains a *locally-recovered packets* field, used to indicate which of the packets requested for retransmission have been already locally retransmitted by some node. Upstream nodes check this field to avoid multiple retransmissions of the same packets. When the source of the transfer receives an ACK, it will only retransmit packets that remain in the SNACK field.

If the cache of a node becomes full, to insert a newly arriving packet, the packet evicted from the cache is the least recently manipulated, *i.e.* Least Recently Used (LRU) policy. The motivation is that it is unlikely that those packets not recently requested for retransmission would be ever requested in the future. A detailed study of different cache replacement strategies is the subject of future work.

In order to assess the benefits of caching, we analytically computed the gain of locally recovering lost packets. Due to space limitations we only provide a summary and the main results of the analysis—the reader is referred to [27] for more details.

The analysis aims at computing an upper bound on the gains achieved by caching so we assume a best-case scenario whereby cache sizes are infinite, and the path is symmetric, thus each lost packet will be recovered by the last downstream node which has successfully received it.

If p is the link loss probability, then with a simple probabilistic analysis the expected total number of node transmissions required by JTP in order to deliver k packets over H hops is given by:

$$E[T_{tot}^{JTP}] = k \times H \times \frac{1}{1-p} \tag{5}$$

In the case of JTP with no in-network caching (denoted JNC), the analysis is a little bit more complicated. The main idea is to compute the number of packets, S, that the source needs to send so as the destination receives k of them, and then compute the total number of node transmissions triggered by each of the S packets as:

$$E[T_{tot}^{JNC}] = \frac{k \times H}{(1-p^n)^{H-1}(1-p)} \tag{6}$$

where n is the maximum number of link attempts. Observe that the cost of JNC is $\frac{1}{(1-p^n)^{H-1}}$ times higher than that of JTP.

4.1 Fair In-network Caching

Enabling mid-path nodes to retransmit packets on behalf of sources may cause a violation of sending rate compliance imposed by the destination of a transfer. For the sake of fairness and congestion control, the source node must incorporate in-network retransmissions in its sending rate calculations. To this end, JTP

[7]The jtp_{20} plot is always above the application's requirement since the aggregate loss rate of the path is below 20%.
[8]The flow that corresponds to 0% loss tolerance is omitted from this plot since iJTP will always assign its packets the maximum number of node attempts.

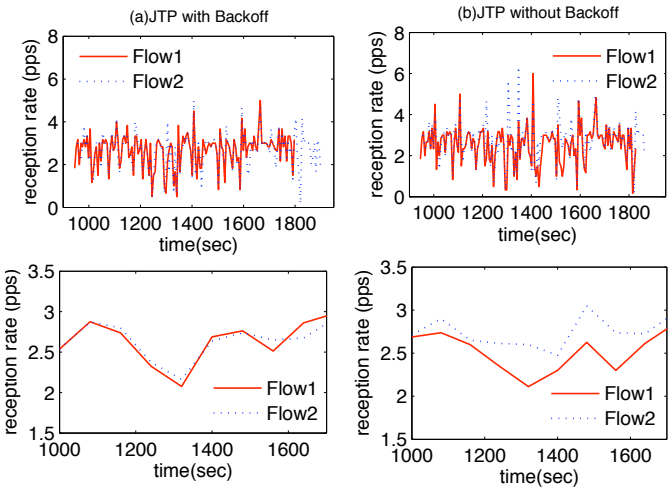


Figure 3: Short-term (top row) and long-term (bottom row) average of the reception rate for two competing flows: (a) the source backs off for locally recovered packets; (b) it does not.

forces sources to back off in accordance to the extra traffic that is induced by in-network retransmissions. When an ACK packet is received, the source uses the *locally-recovered packets* field to adjust its sending rate. Let $r(t)$ be the rate indicated to the source by a received ACK at time t. Let N be the number of packets locally recovered within the network, and let s_j, $j \in [1, N]$ be the sizes of these packets. The source computes an appropriate back-off period t_b as follows:

$$t_b = \frac{\sum_{j=1}^{N} s_j}{r(t)}$$

Figure 3 shows the short- (top plots) and long- (bottom plots) term average of the packet reception rate (throughput) at the destination for two competing flows. (See Section 6 for simulation details.) Flow 1 does not request packet retransmissions (i.e. UDP-like flow), while flow 2 requires that all its packets be delivered and thus invoking the local recovery mechanism of the in-network caches. We observe in the right plots, spikes in the reception rate of flow 2 when it does not back off its sending rate to account for its additional in-network retransmissions—the unfairness introduced is more evident from the long-term average plots.

5. DESTINATION BASED CONTROL

5.1 Path Monitoring using Flip-flop Filtering

One design concept of JTP is to adaptively minimize the frequency at which the destination node informs the source of new transmission parameters, such as a new sending rate. To this end, the end-to-end component of JTP, eJTP, collects sample measurements of the state of the connection's path, such as the minimum available rate over the links of the path. Let x_i denote the i^{th} sample of a path's metric, $\bar{x}$ the estimated average, and $\bar{R}$ the estimated range of sample values. We use principles from statistical quality control [23] to detect a *significant* change in the path's state, which then triggers the destination to send *additional* feedback signal, in addition to feedback sent regularly at a low frequency.

To that end, we estimate the EWMA's $\bar{x}$ and range $\bar{R}$ as follows:

$$\begin{aligned} \bar{x} &= (1-\alpha)\bar{x} + \alpha x_i, \text{ initially } \bar{x} = x_0 \\ \bar{R} &= (1-\beta)\bar{R} + \beta \mid x_i - x_{i-1} \mid, \text{ initially } \bar{R} = \frac{x_0}{2} \end{aligned} \quad (7)$$

$\bar{R}$ is used to estimate the deviation around $\bar{x}$. $\bar{R}$ is calculated *only* from samples x_i within the following upper and lower control limits:

$$UCL = \bar{x} + 3\frac{\bar{R}}{1.128}; \quad LCL = \bar{x} - 3\frac{\bar{R}}{1.128} \quad (8)$$

Under normal operation, *stable* EWMA filters are employed, i.e. the weights α and β are small so short-term variations are filtered out. As long as x_i lies within the control limits, the state of the connection's path is considered *stable* and feedback is only reported to the source at low frequency, say every T seconds. Otherwise, x_i is considered an *outlier*. A certain number of *consecutive* outliers is used as indication of significant *and* persistent change in the state of the path, which would then trigger an immediate feedback to the source node. At this point, eJTP at the destination switches to an *agile* EWMA filter where a larger α value is used, so that $\bar{x}$ catches up with the actual value. Once x_i falls back again within the control limits, eJTP at the destination switches back to the stable filter for this connection. This usage of both stable and agile filters is known as a *Flip-flop Filter* [6].

In our implementation, we set T as a function of the sending rate and of the cache size used in the network. Specifically,

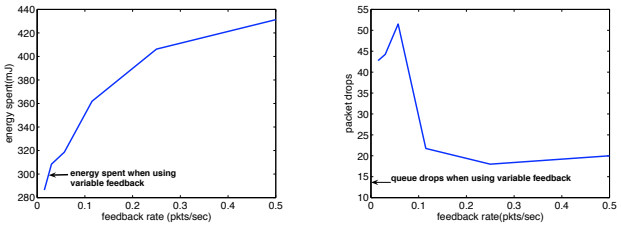
$T = \max(T_{Lower\_Bound}, n \times \frac{1}{SendingRate})$; $n \geq 1$.

Notice that this ensures that the destination does not send feedback/SNACK messages at a rate higher than the sending rate. The value of T_{Lower_Bound} is dependent on the size of in-network caches, since if packets requested for retransmission by a feedback message have already been evicted from the cache then the energy savings achieved by infrequent feedback messages would be offset by the energy consumed by packets that have to be retransmitted from the source. If C is the effective cache size and RTT is the round-trip time for the connection then[9]

$$T_{Lower\_Bound} \leq C \times SendingRate - RTT$$

Figure 4 depicts the energy gains achieved by using

[9]Estimating the effective cache size for each flow is hard, so the value for the parameter C should be set conservatively.



(a) Total energy expended in the system.

(b) Total number of packet drops in the queues of the system.

Figure 4: The gains achieved by using variable feedback rate.

variable-rate feedback instead of a constant rate. In this experiment, a linear topology of 8 nodes is used, with one long-lived flow competing with several short-lived flows. (See Section 6 for simulation details.) We vary the rate of constant-rate feedback—as expected, as the feedback rate increases, the total energy consumed (Figure 4(a)) increases since more feedback/ACK packets are generated; and for low feedback rates, we observe a high number of packet drops in the queues of intermediate nodes (Figure 4(b)) because the sender of the long-lived flow does not back off fast enough causing congestion in the system. Using variable-rate feedback, JTP not only achieves low energy consumption, but also minimizes packet drops in the system since whenever the system load increases, it sends a timely feedback forcing the sender to back off.

5.2 Congestion Avoidance Mechanism

When the flip-flop path monitor triggers a new feedback message, the path controller at the destination should set the sending rate, to be used by the source for the subsequent packets until a new feedback is received.

5.2.1 PI²/MD Sending Rate Controller

This controller at the destination sets the sending rate using the minimum available rate collected along the path of the JTP connection. Let $\bar{A}$ denote the average available path rate measured at the eJTP destination, and $\delta \geq 0$ indicate the target available path rate.[10] If $\bar{A} > \delta$ then the source increases its sending rate r in proportion to the current available capacity and, to improve fairness among competing flows, inversely proportional to the current sending rate:

$$r(t+1) \quad = \quad r(t) + K_I \, \frac{\bar{A}(t)}{r(t)}, \; 0 < K_I < 1 \qquad (9)$$

On the other hand, if there is little available rate ($\bar{A} < \delta$), then the source decreases its sending rate multiplicatively:

$$r(t+1) \quad = \quad K_D \; r(t), \; 0 < K_D < 1 \qquad (10)$$

We note that the eJTP destination also limits the sending rate by its delivery rate up the stack to the receiving-side of the application.

In order to analyze the stability of the controller, we considered a single JTP flow adapting its sending rate over a fixed-capacity channel. The details of the analysis are omitted due to space limitations—refer to [27] for details. We used Lyapunov functions to analyze stability of this non-linear system. We showed that $K_I > 0$ and $K_D < 1$ are sufficient conditions for convergence. We also proved that the controller is efficient and it will converge even for low frequency of sending-rate update albeit at a slower pace.

5.2.2 PI²/MD Fairness under Dynamic Conditions

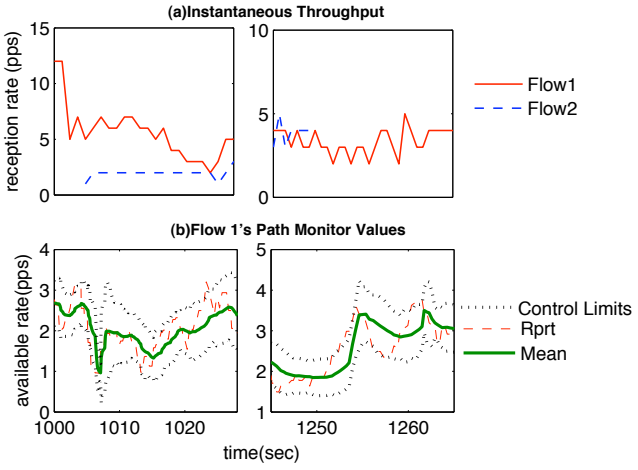


Figure 5: Rate adaptation for two competing JTP flows.

Figure 5 demonstrates the behavior of the flip-flop path monitor of a long-lived flow 1 when competing with a short-lived flow 2 which starts and ends at times 1000 and 1250, respectively. We observe (in the zoomed-in bottom plots) how the average value of available rate catches up with the instantaneous reported values as the monitor switches to the agile EWMA filter, so that the JTP source of flow 1 quickly backs off or increases its rate accordingly. The top plots show the fair convergence of flow rates when flow 2 is present.

6. PERFORMANCE EVALUATION

Earlier in the paper, we have shown simulation results that demonstrate the operation and performance of the various mechanisms in JTP. In this section, we present additional results from the evaluation and testing of JTP.

[10] In this paper we use $\delta = 0$, but it is possible to use more conservative values, guaranteeing that there would be available bandwidth within the network for a better start up for new flows.

JTP is written as "shared code" so it can run on any operating-system/hardware platform. To date, adaptation layers have been written to interface the shared code of JTP with either the OPNET simulation environment [1] or the Linux OS over JAVeLEN radios [11, 26]. We start by showing OPNET simulation results, then Linux testbed results.

6.1 Simulation Results

In order to thoroughly evaluate the performance of JTP, two types of topology are considered. Initially, JTP was tested on *static linear topologies* in order to study the effect of path length between sender and receiver on performance. We also evaluated JTP on *random topologies*, with and without mobility of nodes, to show its performance in realistic scenarios.

In order to provide a comprehensive and fair comparison against existing transport approaches for multi-hop wireless networks [9,21,35,42], JTP is compared against two representative protocols.

• **TCP-SACK:** TCP-SACK is chosen as a representative for all window-based approaches and as the most commonly implemented protocol in working systems. In order to have a more competitive performance, we use a rate-based flavor of TCP-SACK, whereby the rate of each flow is set by the well-known throughput equation of TCP [24]. Thus we remove artifacts from the window-induced burstiness of data and ACK streams, similar to what TCP pacing [2] does. Moreover, we used delayed ACKs (one ACK every two packets) in an attempt to reduce the rate of the ACK stream. The SACK version helps TCP selectively retransmit lost packets only.

• **ATP-like:** In order to compare against the class of explicit rate-based transport protocols, we implemented a protocol which adjusts the sending rate based on explicit feedback collected by intermediate nodes, supports only end-to-end recovery, and has constant-rate feedback from the receiver. The feedback period is set to be larger than RTT as suggested for ATP [35]. ATP adjusts its sending rate based on explicit rate feedback from the path and thus takes into account real congestion capturing the behavior of TCP CLAMP [3].

Given that TCP-SACK and ATP only support 100%-reliability transfers, we will consider only bulk transfers with 0% loss tolerance. Unless otherwise stated, the configuration parameters used throughout the experiments are listed in Table 1. In this prototype implementation the JTP header is 28 bytes and the JTP ACK header is 200 bytes. [11]

In this paper, we show two different performance metrics to compare the efficiency of each protocol.

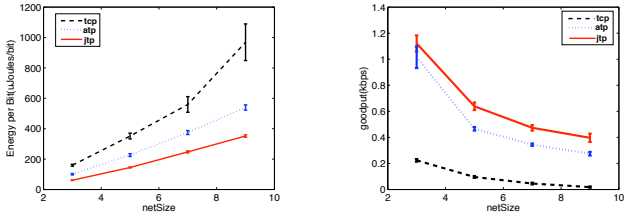
• *Energy per delivered bit*: This measure captures the system-wide energy consumed to deliver each data bit to applications. Since our goal is to evaluate the energy consumption of transport protocols, we are only concerned with the energy actually spent to transfer packets of the transport layer, and thus we will not consider the energy consumed for network maintenance by the lower layers. To this end, a monitor is placed in the link layer that computes the energy spent for the transmission of each transport-layer packet based on the transmission power, the radio's datarate and the packet's length.

• *Goodput*: This measure captures the total rate at which the network delivers new data to the applications, and thus represents how efficient the network resources are utilized.

[11]The JTP headers, and especially the ACK headers are not optimized in this prototype implementation.

Table 1: Parameters' default value

Parameter	Value	Parameter	Value
MAX_ATTEMPTS	5	JTP Pkt Size	800 bytes
Cache Size	1000 pkts	T_{Lower_bound}	10s



(a) Total energy expended per application data bit delivered.

(b) Average goodput experienced by flows in the network.

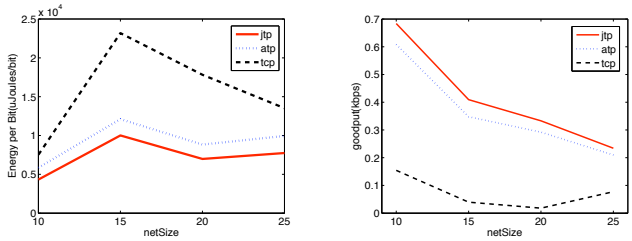
Figure 6: Results for Linear Topologies.

6.1.1 Static Linear Topologies

In these experiments the source and the destination of two competing flows are placed at the two ends of the network. To capture the varying quality of wireless links, the value of the average *pathloss* of each link alternates between a good state (low loss) and a bad state (high loss). Each link is in bad state approximately 10% of the time. The average duration of the bad period is 3 seconds. The results shown are the average of twenty (independent) runs along with 95% confidence intervals. Each simulation run lasted for 2500 seconds, and flows were started randomly after a warm-up period of 900 seconds.

Figure 6(a) shows the energy per delivered bit for each protocol for varying network sizes. JTP significantly outperforms all the other protocols. As the path length increases, ATP ends up expending twice as much energy as JTP to deliver one bit, while TCP-SACK expends almost five times more energy for the delivery of each bit.

Figure 6(b) demonstrates that JTP not only provides great energy savings, but also achieves higher goodput. Without sacrificing system's performance, JTP minimizes the amount of feedback control messages, which



(a) Total energy expended per application data bit delivered.

(b) Average goodput experienced by flows in the network.

Figure 7: Results for Static Random Topologies.

in a wireless network environment, effectively "steal" bandwidth from users' data.

6.1.2 *Random Topologies*

We begin by testing JTP over static random topologies. Nodes are randomly distributed in a two-dimensional field. In order to avoid getting disconnected topologies, the field size is set to ensure that the network is connected with high probability. The source and destination nodes of 5 simultaneous flows are chosen randomly. The presented results are the average of 10 independent runs, of $4000s$ each both for the static and the mobile scenarios. Given that the placement of the nodes and flows are chosen at random, the system-wide performance might vary significantly. In order to meaningfully compare across different protocols, we ensured that all the protocols run under the same conditions in the same run.

Figures 7(a) and (b) show the energy per delivered bit and the goodput achieved by various protocols for varying network size. JTP outperforms both ATP and TCP in both metrics.

In the next set of experiments, we tested JTP's performance in a mobile setting for a 15-node network. Each node moves within the field at various speeds (low: $0.1m/s$, moderate: $1m/s$, fast: $5m/s$). We used the random way point mobility model in which each node chooses a random direction and moves in that direction for an average distance of $47m$. There is an average pause of $100s$ between movements for each node.

Figures 8(a) and (b) show the energy per delivered bit and the goodput achieved by the three protocols. Figure 8(c) presents the relation between end-to-end and locally recovered packets—the values presented are normalized by the total data delivered to the applications. This graph shows that even in mobile environments, where the path between two nodes is constantly changing, deploying local caches is beneficial—we observe in-network retransmissions which result in energy gains and better distribution of retransmission effort across nodes.

6.2 Linux Results

In order to verify our simulation results, we implemented and tested JTP in a real JAVeLEN system. In this system, the MAC is running in RTLinux, while the non real-time part of the system, like the applications, are running on top of Linux. In these experiments, we used 14 such systems and we ran JTP, TCP-SACK and ATP. Each experiment lasted for 30 minutes. During these 30 minutes, flows were generated in each node with an average interarrival time of 400sec and average transfer size of 100KB. A summary of the results is shown in Table 2. Given that in the real system the pathloss of the links is not controlled but it is only determined by the in-door multipath fading, the links are more stable and their quality is much better, which results in lower energy consumption for all protocols. Nevertheless, JTP still outperforms both ATP and TCP in both metrics. Notice that the goodput achieved by TCP is higher than that achieved in simulation due to the low packet loss rate.

Table 2: JAVeLEN system results

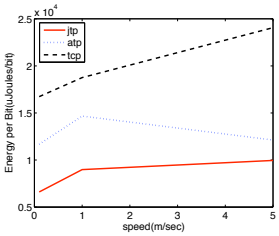
	Energy per delivered bit (mJ/bit)	Average goodput (kbps)
JTP	0.0054	0.63
ATP	0.0068	0.44
TCP	0.0105	0.17

7. RELATED WORK

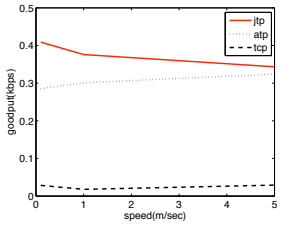
Extensive studies [14] have demonstrated the inadequacy of TCP to serve as a transport protocol in wireless environments. Enhancements have mainly focused on alleviating the effects of assuming that packet losses are only due to congestion.

Proxy-based approaches: Focus on hiding wireless losses from the TCP sender [4,5] by retransmitting from caches at the wireline-wireless boundary. Ludwig has shown that, if not designed carefully, end-to-end and in-network retransmissions, used together, can cause worse performance than either alone [22]. Energy conservation in *multi-hop* wireless networks led us to extend this concept to retransmissions from caches anywhere along the wireless path. As we showed in this paper, JTP gives the ends of the connection *explicit* control over the amount of local retransmission effort. In addition, redundant source retransmissions are avoided by explicitly informing the sender of local (cache) retransmissions done on its behalf.

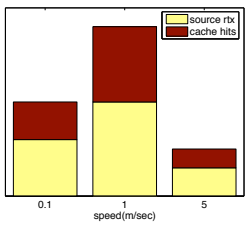
End-to-end approaches: Attempt to identify the type of loss either explicitly such as ATCP [21], or implicitly such as WTCP [33]. Even perfect knowledge of the reason of packet loss (*e.g.* congestion-induced vs. transmission error) at the sender, often, does not improve throughput performance [6, 19]. Moreover, these schemes suffer from the slow adaptation of TCP's AIMD mechanism [16] to the fast changing conditions of wireless links. TCP-Westwood [8] addresses this problem by augmenting AIMD with an estimate of the available bandwidth measured based on the ACK reception rate.



(a) Total energy expended per application data bit delivered.



(b) Goodput delivered to applications.



(c) Relation between end-to-end and locally recovered packets.

Figure 8: Results for Random Topologies with Mobility.

Other approaches tried to alleviate the effects of bursty TCP traffic by clamping the congestion window [3] or by pacing TCP packets [2]. Although these approaches significantly improve TCP performance they still rely on packet loss to identify congestion. Our JTP protocol is rate-based and avoids congestion altogether.

Receiver-based control: The sender centric approach of TCP requires frequent feedback which may cause congestion and force the sender to back off. A receiver centric flavor of TCP, such as RCP [15], has been proposed, however the rate of the backward ACK stream is not reduced. Our JTP protocol is also receiver-based, but the feedback rate is dynamically adjusted based on the stability and reliability conditions of the forward path.

Rate-based flow control: To ameliorate the ACK compression problem, rate-based protocols [10] have been proposed, whereby the available rate could be explicitly collected and fed back to the sender [35]. These solutions still use frequent *constant-rate* feedback which competes with data flows for resources. Our JTP protocol attempts to reduce the feedback rate constrained by the conditions and cache sizes on the forward path.

Application-specific protocols: Transport protocols cognizant of a *certain* application's QoS requirements have been devised, such as RTP [13] and ITP [25]. Our JTP protocol further generalizes such proposals by enabling *any* application to not only influence the flow and error control mechanisms but also in-network decisions regarding the handling of its packets.

Energy-conscious scheduling and routing: In all aforementioned research, energy consumption has not been examined. Approaches have been proposed to monitor and even shape application's data to turn on and off the network interface on end-nodes for the purpose of energy savings, while still satisfying the application's requirements [18]. These monitoring techniques are hard to apply in multi-hop wireless networks, where each node is both, a router and an end-node.

In the context of proxy-based schemes, the tradeoff between throughput performance and energy costs (due to transmission power and error control) was analyzed in [7]. For multi-hop wireless networks, several power-aware MAC scheduling and routing protocols have been proposed [28,32,36,40]. JAVeLEN [11,26] builds on this body of prior work. We demonstrated in this paper, that even if network nodes are parsimonious in their use of energy (*e.g.* nodes turned off when there is no data to transmit or receive), an energy-aware transport protocol, such as JTP, can achieve greater energy gains by turning on the radios only when it is absolutely necessary. To this end, JTP minimizes control traffic and avoids data transmissions that are unnecessary for meeting given delivery requirements of applications.

Sensor protocols: Energy-aware transport protocols have been proposed in the realm of sensor networks, such as PSFQ [39]. Given the goal of one-to-many reliable delivery in such sensor network realm (*e.g.* to program the sensors), issues that arise in multi-hop wireless networks regarding the fair allocation of resources among flows and the reduction of in-network overhead have not been considered.

Other wireline protocols: Other protocols, proposed for wireline networks, such as SCTP [12] and XCP [17], suffer from inefficiencies similar to TCP when employed in multi-hop wireless environments.

8. CONCLUSIONS AND FUTURE WORK

We presented JTP, an energy-conscious transport protocol, that is rate-based receiver-oriented with variable feedback, coordinated in-network error recovery, and application-aware per packet handling. Our results show that JTP outperforms traditional transport protocols (TCP and UDP) and a representative multi-hop wireless transport protocol (ATP) in every respect. For all network sizes and mobility settings evaluated, JTP consistently provided higher goodput and consumed less energy per node and over the entire network.

Our research also contributes in the form of lessons learned. For example, we have demonstrated the need to have multi-level error recovery, via the MAC and caching, that is explicitly controlled by the ends of connections. This shows that problems in energy-awareness can yield non-intuitive solutions.

JTP provides a fresh perspective that offers many opportunities for future work. We are currently investigating energy-awareness in cache/memory management, and the support of other applications such as data collection and image transfers.

9. REFERENCES

[1] OPNET. http://www.opnet.com/.

[2] A. Aggarwal, S. Savage, and T. Anderson. Understanding the Performance of TCP Pacing. In *Proceedings of INFOCOM*, pages 1157–1165, 2000.

[3] L. Andrew, S. Hanly, and R. Mukhtar. CLAMP: A System to Enhance the Performance of Wireless Access Networks. In *Proceedings of GLOBECOM*, 2003.

[4] A. Bakre and B. R. Badrinath. I-TCP: Indirect TCP for Mobile Hosts. In *Proceedings of ICDCS*, 1995.

[5] H. Balakrishnan, S. Seshan, E. Amir, and R. Katz. Improving TCP Performance Over Wireless Networks. In *Proceedings of ACM MobiCom*, 1995.

[6] D. Barman and I. Matta. Effectiveness of Loss Labeling in Improving TCP Performance in Wired/Wireless Networks. In *Proceedings of ICNP'2002: The 10th IEEE International Conference on Network Protocols*, Paris, France, November 2002.

[7] Dhiman Barman, Ibrahim Matta, Eitan Altman, and Rachid El Azouzi. TCP Optimization through FEC, ARQ and Transmission Power Tradeoffs. In *Proceedings of WWIC 2004: The 2nd International Conference on Wired/Wireless Internet Communications*, Frankfurt (Oder), Germany, February 2004.

[8] C. Casetti, M. Gerla, S. Mascolo, M. Sanadidi, and R. Wang. TCP Westwood: Bandwidth Estimation for Enhanced Transport Over Wireless Networks. In *Proceedings of ACM MobiCom*, 2001.

[9] K. Chen, K. Nahrstedt, and Nitin Vaidya. The Utility of Explicit Rate-Based Flow Control in Mobile Ad Hoc Networks. In *Proceedings of WCNC*, 2004.

[10] D. D. Clark, M. L. Lambert, and L. Zhang. NETBLT: a High Throughput Transport Protocol. In *SIGCOMM '87: Proceedings of the ACM Workshop on Frontiers in Computer Communications Technology*, pages 353–359, New York, NY, USA, 1987. ACM Press.

[11] J. Redi et al. Joint Architecture Vision for Low Energy Networking (JAVeLEN)–DARPA-ATO Connectionless Networking Program, 2004. BBN Technichal Report.

[12] R. Stewart et al. Stream Control Transmission Protocol, 2000. RFC 2960.

[13] H. Schulzrinne et.al. RTP: A Transport Protocol for Real-Time Applications, 1996. RFC 1889.

[14] Z. Fu, X. Meng, and S. Lu. How Bad TCP Can Perform in Mobile Ad Hoc Networks. In *Proceedings of ISCC*, 2002.

[15] H. Hsieh, K. Kim, Y. Zhu, and R. Sivakumar. A Receiver-Centric Transport Protocol for Mobile Hosts With Heterogeneous Wireless Interfaces. In *Proceedings of ACM MobiCom*, 2003.

[16] Van Jacobson. Congestion Avoidance and Control. In *Proceedings of ACM SIGCOMM '88*, pages 314–329, August 1988.

[17] Dina Katabi, Mark Handley, and Charlie Rohrs. Congestion Control for High Bandwidth-delay Product Networks. In *Proceedings of ACM SIGCOMM '02*, pages 89–102, 2002.

[18] R. Kravets and P. Krishnan. Application-driven Power Management for Mobile Communication. *Wireless Networks*, 6:263–277, 2000.

[19] Rajesh Krishnan, James P. G. Sterbenz, Wesley M. Eddy, Craig Partridge, and Mark Allman. Explicit Transport Error Notification (ETEN) for Error-prone Wireless and Satellite Networks. *Comput. Networks*, 46(3):343–362, 2004.

[20] Y. Liaw, A. Dadej, and A. Jayasuriya. Estimating Throughput Available to a Node in Wireless Ad-hoc Network. In *Proceedings of MASS 2004*, pages 555–557, Fort Lauderdale, Florida, 2004.

[21] J. Liu and S. Singh. ATCP: TCP for Mobile Ad Hoc Networks. *IEEE Journal on Selected Areas in Communications*, 19(7):1300–1315, 2001.

[22] R.E. Ludwig. *Eliminating Inefficient Cross-Layer Interactions in Wireless Networking*. PhD thesis, Technischen Hochschule Aachen, April 2000.

[23] Douglas C. Montgomery. *"Introduction to Statistical Quality Control"*. John Wiley & Sons, New York, 5th edition edition, 2005.

[24] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Throughput: A Simple Model and its Empirical Validation. In *Proceedings of ACM SIGCOMM*, Vancouver, British Columbia, September 1998.

[25] S. Raman, H. Balakrishnan, and M. Srinivasan. ITP: An Image Transport Protocol for the Internet. In *Proceedings of ICNP'2000: The 8th IEEE International Conference on Network Protocols*, 2000.

[26] J. Redi, S. Kolek, K. Manning, C. Partridge, R. Rosales-Hain, R. Ramanathan, and I. Castineyra. JAVeLEN—An Ultra-Low Energy Ad Hoc Wireless Network. *Ad Hoc Networks*, 6(1):108–126, 2008.

[27] N. Riga, I. Matta, A. Medina, C. Partridge, and J.Redi. An Energy-concious Transport Protocol for Multi-hop Wireless Networks , 2007. BU Technichal Report.

[28] R. Rozovsky and P. R. Kumar. SEEDEX: A MAC Protocol for Ad Hoc Networks. In *Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2001.

[29] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end Arguments in System Design. *ACM Trans. Comput. Syst.*, 2(4):277–288, 1984.

[30] C. Santivanez, R. Ramanathan, and I. Stavrakakis. Making Link-State Routing Scale for Ad Hoc Networks. In *Proceedings of MobiHoc*, pages 22–32, 2001.

[31] E. Shih, P. Bahl, and M. J. Sinclair. Wake on Wireless: An Event-driven Energy Saving Strategy for Battery Operated Devices. In *Proceedings of the 8th Annual International Conference on Mobile Computing and Networking (Mobicom)*, 2002.

[32] S. Singh, M. Woo, and C.S. Raghavendra. Power Aware Routing in Mobile Ad Hoc Networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (Mobicom)*, 1998.

[33] P. Sinha, N. Venkitaraman, R. Sivakumar, and V. Bharghavan. WTCP: A Reliable Transport Protocol for Wireless Networks. In *Proceedings of ACM MobiCom*, 1999.

[34] I. Stoica and H. Zhang. Providing Guaranteed Services Without Per Flow management. In *Proceedings of ACM SIGCOMM*, 1999.

[35] K. Sundaresan, V. Anantharaman, H-Y. Hsieh, and R. Sivakumar. ATP: A Reliable Transport Protocol for Ad-hoc Networks. In *Proceedings of ACM MobiHoc*, Annapolis, ML, 2003.

[36] B. Tavli and W. Heinzelman. MH-TRACE: Multi-Hop Time Reservation using Adaptive Control for Energy Efficiency. *IEEE Journal on Selected Areas of Communication*, 22(5):942–953, June 2004.

[37] K.H. Torvmark. Low Power Systems Using the CC1010. Chipcon Application Note AN017.

[38] V. Tsaoussidis and I. Matta. Open Issues on TCP for Mobile Computing. *Wireless Communications and Mobile Computing*, 2(1):3–20, 2001.

[39] C. Wan, A. T. Campbell, and L. Krishnamurthy. PSFQ: A Reliable Transport Protocol for Wireless Sensor Networks. In *Proceedings of WSNA*, 2002.

[40] W. Ye, J. Heidemann, and D. Estrin. Medium Access Control with Coordinated Adaptive Sleeping for Wireless Sensor Networks. *IEEE/ACM Transactions on Networking*, 12(3), June 2004.

[41] H. Yousefizadeh, L. Zheng, and H. Jafarkhani. Rate Constrained Power Control in Space-Time Coded Fading Ad-Hoc Networks. In *Proceedings of IEEE Global Communications Conference (Globecom-04)*, November 2004.

[42] H. Zhai, X. Chen, and Y. Fang. Rate-Based Transport Control for Mobile Ad Hoc Networks. In *Proceedings of WCNC*, 2005.