

JTP: An Energy-conscious Transport Protocol for Wireless Ad Hoc Networks

(Technical Report BUCS-2006-025 / BBN-8463)

Niky Riga Ibrahim Matta
Computer Science
Boston University
Email: {inki, matta}@cs.bu.edu

Alberto Medina Jason Redi Craig Partridge
BBN Technologies
Cambridge, MA, USA
Email: {amedina, redi, craig}@bbn.com

Abstract—Within a recently developed low-power ad hoc network system, we present a transport protocol (JTP) whose goal is to reduce power consumption without trading off delivery requirements of applications. JTP has the following features: it is lightweight whereby end-nodes control in-network actions by encoding delivery requirements in packet headers; JTP enables applications to specify a range of reliability requirements, thus allocating the right energy budget to packets; JTP minimizes feedback control traffic from the destination by varying its frequency based on delivery requirements and stability of the network; JTP minimizes energy consumption by implementing in-network caching and increasing the chances that data retransmission requests from destinations “hit” these caches, thus avoiding costly source retransmissions; and JTP fairly allocates bandwidth among flows by backing off the sending rate of a source to account for in-network retransmissions on its behalf. Analysis and extensive simulations demonstrate the energy gains of JTP over one-size-fits-all transport protocols.

I. INTRODUCTION

Motivation and Scope: Wireless ad hoc networks are plagued with unique challenges: contention for the wireless medium, time-varying topology due to the variable quality of links or mobility, and battery power constraints. A primary goal of such a network is to minimize the usage of energy so as to extend its lifetime while meeting the requirements of its applications.

To this end, much of the efforts have targeted improving the routing and lower layers of the protocol stack (e.g. [1], [2]). At the transport layer, enhancements to the TCP protocol have been devised (e.g. [3]–[5]) and some new protocols have been proposed to provide TCP-like reliability semantics over wireless ad hoc networks (e.g. [6]). However, these transport protocols only had the goal of improving goodput in the face of the intrinsic characteristics of wireless multi-hop environments without consideration to energy costs [7] or to the varying levels of reliability requirements of applications.

Recently, the JAVeLEN (Joint Architecture Vision for Low Energy Networking) architecture has been developed to elevate energy efficiency as a first-class optimization metric at all protocol layers, from physical to transport [8], [9]. JAVeLEN’s design ensures that energy gains obtained in one layer would not be offset by incompatibilities and/or inefficiencies in other layers. The developed system has demonstrated vast energy savings. For each bit of user’s data delivered, the JAVeLEN system uses approximately 100 times less energy compared to a baseline OLSR-over-802.11 system [9]. Most of this

energy efficiency comes not from reducing the transmission (or reception) power of the radios, but rather from reducing ancillary costs. The radios are turned off when not being used to deliver data, and higher layer protocols are made parsimonious in their use of the network.

Our Contributions: JTP, JAVeLEN’s transport protocol, mediates between an application’s need to share information of varying importance over the network, and JAVeLEN’s goal of minimizing energy expenditure per successfully delivered bit. In this paper, we present and evaluate the design and novel features of JTP, which are summarized as follows:

- To the best of our knowledge, JTP is the first wireless ad hoc *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* [10]. JTP employs mechanisms akin to the Dynamic Packet State [11] to avoid maintaining per-flow state.
- 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 [12], RTP [13]). JTP’s modular design separates the transport protocol’s functionalities into *application-specific* and *network-specific*, allowing any application to tailor the protocol’s behavior based on its specific QoS semantics, and thus JTP acts as a “generic” transport protocol. To this end, JTP expands the notion of *per-packet energy budgets* [14], which are allocated to specify how much energy the network should invest in trying to deliver a packet, based on the packet’s individual importance to the application as well as current energy costs.
- In JTP, the receiver is fully responsible for controlling all transmission parameters, including the per-packet energy budget, the 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.
- JTP implements a caching mechanism which enables intermediate nodes along the path of a JTP connection to temporarily store traversing packets. 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 [4]. To increase the chances of “hitting” the caches, JTP employs a rate-based flow control whose goal is to avoid contention, thus making the “routing cost” at the two ends of links symmetric, and so are the computed paths taken by data and retransmission requests.

- 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.

Organization of the Paper: Section II describes related work. Section III presents the architectural elements of JTP, and its cross-layer interactions. JTP’s support for adjustable QoS is discussed in Section IV. The destination-based quality monitoring and control is described in Section V. Section VI discusses in-network caching and its implications on fairness and flow control. Section VII presents simulation results. An appendix contains analysis of stability of the rate-based JTP flow control, and of the energy gains from caching.

II. RELATED WORK

Extensive studies (e.g. [15]) 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 (e.g. [3], [4]) 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 [16]. Energy conservation in *multi-hop* ad hoc networks led us to extend this concept to retransmissions from caches anywhere along the wireless path.

End-to-end approaches: Attempt to distinguish the type of loss either explicitly (e.g. ATCP [5]) or implicitly (e.g. WTCP [17]). 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 [18], [19]. Moreover, these schemes suffer from the slow adaptation of TCP’s AIMD mechanism to the fast changing conditions of wireless links. TCP-Westwood [20] addresses this problem by augmenting AIMD with an estimate of the available bandwidth measured based on the ACK reception rate.

Receiver-based control: The sender centric approach of TCP requires frequent feedback which causes congestion and forces the sender to back off. A receiver centric flavor of TCP (RCP [21]) has been proposed, however the rate of the backward ACK stream is not reduced.

Rate-based flow control: To ameliorate the ACK compression problem, rate-based protocols (cf. [22]) have been proposed, whereby the available rate is explicitly collected and fed back to the sender (e.g. ATP [6]). These solutions still use frequent *constant rate* feedback which competes with data flows for resources.

Application-specific protocols: Transport protocols cognizant of a *certain* application’s QoS requirements have been devised (e.g. RTP [13], ITP [12]). Our JTP protocol is more generic than these proposals as it enables *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: 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 for energy savings, while still satisfying application’s requirements (e.g. [23]). These monitoring techniques are hard to apply in wireless ad hoc networks, since each node is also a router as well as an end-node. The tradeoff between throughput performance and energy costs (due to transmission power and error control) was analyzed in [24] in the context of proxy-based schemes.

For wireless ad hoc networks, we will demonstrate in this paper, that even if network nodes are 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 (e.g. PSFQ [25]). Given the goal of one-to-many reliable delivery in such sensor network realm (e.g. to (re-)program the sensors), issues that arise in ad hoc 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 (e.g. SCTP [26]) suffer from inefficiencies similar to TCP when employed in wireless ad hoc environments.

III. JTP DESIGN

Figure 1 shows the elements of the JTP architecture; the path of a JTP connection consists of the source, intermediate nodes, and the destination. The JTP packet is routed along the path, and its hop-by-hop processing and transmission parameters (e.g. available path rate, energy budget, selective negative acknowledgments) are encoded in its header. The destination executes two JTP processes: a path monitor which aggregates path quality measurements collected by JTP packets, and a path (flow) controller which updates the transmission parameters of the JTP connection upon significant and persistent change in path quality. The transmitter JTP process at the source uses these transmission parameters, as well as knowledge of mid-path recovery actions encoded in JTP acknowledgments, to control the sending of new packets.

Underlying substrate: JTP is currently implemented over the lower layers of JAVeLEN [8], [9]. The physical layer (PHY)

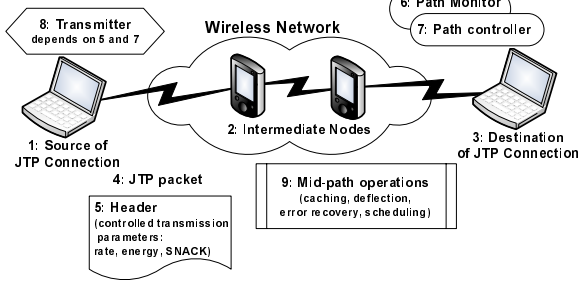


Fig. 1. Elements of JTP

uses dual *signaling-data* radios, whereby a low-power low-rate *signaling* radio wakes up a high-power high-rate *data* radio only when necessary. The MAC layer employs pseudo-random codes for implementing uncorrelated (but predictable) wakeup slotted schedules for the signaling radios. Thus, when the signaling radio of a neighbor node is predicted to be on, a node uses its signaling radio to request its neighbor to wake up its data radio for reception. We note that, together with frequency hopped OFDM radio transmissions, the probabilistic MAC of JAVELEN practically implements collision-free access. The routing layer of JAVELEN collects link-state information using *hazy-sighted scoping* [2] so a node receives more frequent link-state updates from closer nodes. JAVELEN assigns a (quantized) higher (inferior) routing metric to links with smaller signal-to-noise margins. Thus, packets are (next-hop) forwarded over minimum-energy paths.

A. Dynamic Packet State

1) **Data Packets:** As a data packet travels from the source to the destination of a JTP connection, several fields in its header get updated:

- **Available rate:** The available rate of a node represents its current available *reception* capacity determined by its current rate of unused (idle) receive wakeup slots. At each node visited, a packet is stamped with the minimum available rate measured so far along the path of the JTP connection. This available rate is then used by the JTP flow controller at the destination to update the sending rate of the source, as described in Section V. Since a JTP packet may trigger more than one MAC-level packet (slot), the available rate value must be normalized by the average number of MAC-level transmissions, which is computed as the inverse of the success probability of delivery measured for the link. Since JAVELEN provides a practically collision-free MAC layer, the maximum receiving rate of a node represents its effective share of the channel. Hence, if the JTP flow controller does not allow this available rate to drop to zero, contention-induced losses are avoided.

- **Energy budget:** The source initially assigns each packet an energy budget value, which is then reduced at every node along the path by the transmission energy actually used to send the packet successfully through each link. A packet is dropped whenever it runs out of energy. This approach provides an energy-conscious mechanism for dealing with routing loops (as opposed to the traditional hop-count TTL). Furthermore,

it implicitly implements a per-packet access control whereby a packet is dropped if faced with a sudden change in network conditions which cause higher energy expenditure. In the latter case, a retransmission of the packet can make it through when the network condition returns to normal. If the condition does not change (i.e., it is a *persistently* high energy state), the source will eventually obtain new (higher) energy requirements from the destination and will update the energy budget assigned to packets.

- **Loss tolerance:** A source node encodes the desired end-to-end loss tolerance in packet headers. Each node along the path computes the minimal number of transmission attempts to the next hop, given the remaining length of the path and packet's loss tolerance. The packet is dropped if this pre-determined maximum number of local attempts is exceeded. As described in Section IV, 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.

2) **Acknowledgment Packets:** JTP acknowledgment packets (ACK) carry in their headers a list of requested retransmissions from the destination. As ACKs travel back from the destination to the source, the encoded requested retransmissions are updated to reflect retransmissions triggered by cache hits at intermediate nodes. As discussed in Section VI, this informs the source to refrain from unnecessarily retransmitting those packets locally retransmitted inside the network on its behalf.

B. JTP Architecture

The functionalities of JTP are 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 wireless ad hoc networks.

- The *application-specific module* provides the following services: fragmentation/reassembly; passing QoS requirements (e.g. reliability level) to the transfer module in order to influence in-network packet-handling decisions; and flow control and retransmission requests only for those missing packets that are important to meet the reliability target of the application.

- The network-specific modules include: the *transfer module*, which manages JTP connections and implements the path monitor and the (contention-avoidance rate-based) controller of per-packet transmission parameters; the *send module* which assigns the proper radio profile for each transmitted packet based on the quality of service requirements encoded in the packet header; and the *forwarding module* which caches data packets, locally retransmits missing packets from the cache, and updates the packet's header with current local statistics on rate, transmission energy, etc.

The remaining modules provide support services such as *queuing* for managing incoming and outgoing packets, *caching*, and *routing* for managing topological information from the network layer.

C. Cross Layer Interactions

The operation of JTP requires the crafting of cross-layer interactions to enable transport tasks to expend minimal amounts of resources while satisfying the end-to-end semantics of application data transfers. Besides the common interactions with neighboring layers (i.e. the network and application layers), JTP performs direct interactions with the data link (MAC) layer and receives information from the physical layer (PHY).

JTP can influence the transmission parameters used by the MAC layer on a packet by packet basis. This is achieved through the use of radio profiles, which are registered by JTP with the network layer. The parameters that JTP can influence include:

- *Transmission power*: JTP can set the power level to be used by the sender for every packet based on measured *pathloss* of the link.
- *Number of link access attempts*: JTP can limit the number of times a (sending) node will try unsuccessfully to wake up a neighboring (receiving) node.
- *Number of data transmissions*: JTP can limit the number of retransmission attempts made by the link layer.

JTP also obtains valuable information from MAC layer, including *link metrics* such as packet loss rate and pathloss (the latter is obtained by the MAC from the PHY layer), and *nodal metrics*, such as available receiving rate (throughput).

We note that JTP can operate within any network architecture, other than JAVeLEN, as long as the aforementioned inter-layer interfaces are present. For example, JTP may obtain the available nodal throughput from an 802.11 MAC using estimation techniques such as [27]. Furthermore, although energy minimization is JAVeLEN's main objective, JTP's goal could be thought of as more general, that of minimizing resource usage for each application data bit delivered.

IV. ADJUSTABLE QOS REQUIREMENTS

In this section we take *reliability* as an example QoS metric, and we show how JTP incorporates reliability requirements of applications in order to achieve energy savings in the realm of ad hoc networks.

Not all applications (e.g. voice, images [12]) 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), JTP influences the *minimal* effort to impose on the network to deliver only "needed" packets. Specifically, JTP controls the number of link-layer transmission attempts on a per packet basis. In JTP, the reliability level is expressed in terms of a loss tolerance percentage (e.g. 10% of packets can be lost), which is 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:

$$l_{e2e} = 1 - \prod_{i=0}^{H-1} q_i \quad (1)$$

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

$$M_i = \left\lceil \frac{\log(1 - q_i)}{\log(p_i)} \right\rceil \quad (2)$$

The challenge is to dynamically adjust the values of M_i 's 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. In JAVeLEN, where views are hazy-sighted scoped [2], a node that is closer to the destination, has a more accurate view of its path to it. Thus, JTP carries out the computation of q_i 's at each node, as the packet travels toward the destination, thus using increasingly 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. Thus, we want:

$$\prod_{j=i}^{i+H_i-1} q_j = 1 - l_{ti} \quad (3)$$

For ease of exposition, assume JTP sets q_i to be the same for all the links, i.e. $q_i = q$,² then:

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

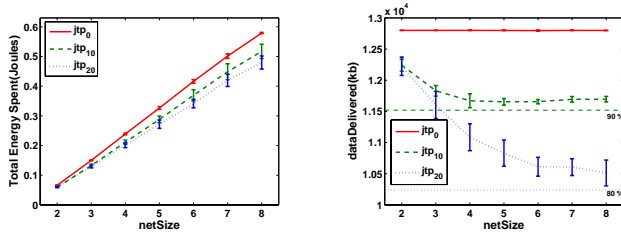
If $p_i \leq (1 - q)$ then the node attempts to transmit the packet only once. Otherwise, the number of transmission attempts, M_i , is computed using equation (2).

Before the node transmits the packet to its next-hop, it updates the loss tolerance field as follows:

$$\begin{aligned} \prod_{j=i}^{i+H_i-1} q_j = 1 - l_{ti} &\Rightarrow q_i \prod_{j=i+1}^{i+H_i-1} q_j = 1 - l_{ti} \Rightarrow \\ \prod_{j=i+1}^{i+H_i-1} q_j = \frac{1 - l_{ti}}{q_i} &\Rightarrow 1 - l_{t(i+1)} = \frac{1 - l_{ti}}{q_i} \Rightarrow \\ l_{t(i+1)} &= 1 - \frac{1 - l_{ti}}{q_i} \end{aligned} \quad (5)$$

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).

¹The success probability is equal to $\sum_{k=0}^{M_i-1} p_i^k (1 - p_i) = (1 - p_i^{M_i})$.
²It is straightforward to require different q_i 's on different links. Such unequal allocation policy may impose higher successful delivery requirement on links that are less loaded or have higher available energy.



(a) Total Energy spent for transfers of different reliability levels. (b) Data delivered to application.

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

Figure 2(a) shows the system-wide energy expenditure for different levels of reliability over linear topologies of varying size (see Section VII for simulation details). Figure 2(b) demonstrates that by not overachieving (e.g. TCP-like full reliability for all), nor underachieving (e.g. UDP-like no reliability for all), JTP manages to save energy by satisfying given application's requirements.

V. DESTINATION BASED CONTROL

A. Path Monitoring using Flip-flop Filtering

One design concept of JTP is to adapt to a minimum the frequency at which the destination node informs the source of new transmission parameters (e.g. sending rate, per-packet energy budget). JTP collects sample measurements of the state of the connection's path, e.g. the minimum available rate over the links of the path, or the sum of per-packet transmission energy. We denote by x_i the i^{th} such sample. We use principles from statistical quality control [28] 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:

$$\begin{aligned} \bar{x} &= (1 - \alpha)\bar{x} + \alpha x_i, \text{ initially } \bar{x} = x_0 \\ \bar{R} &= (1 - \beta)\bar{R} + \beta |x_i - x_{i-1}|, \text{ initially } \bar{R} = \frac{x_0}{2} \end{aligned} \quad (6)$$

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

$$UCL = \bar{x} + 3 \frac{\bar{R}}{1.128}; \quad LCL = \bar{x} - 3 \frac{\bar{R}}{1.128} \quad (7)$$

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 consecutive number of 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, the JTP 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, the JTP 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 [19].

In our implementation, we set T as a function of the sending rate. In addition, we place a lower bound on T , say 3 seconds. Specifically,

$$T = \max(T_{Lower_Bound}, n \times \frac{1}{SendingRate}); \quad n \geq 1.^3$$

B. Contention Avoidance Mechanism

When the flip-flop path monitor triggers a new feedback message, the path (flow) controller at the destination should set the sending rate, and the energy budget to be used by the source for the subsequent packets until a new feedback is sent. The controller is more involved for the sending rate since the source should share the available rate on the path by adapting its sending rate up or down depending on whether the path is underutilized.

1) PP^2/MD Sending Rate Controller: This controller makes use of the minimum available rate measured along the path of the JTP connection. We denote by \bar{A} the average available rate measured at the JTP destination. 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:

$$r(t+1) = r(t) + K_I \frac{\bar{A}(t)}{r(t)}, \quad 0 < K_I < 1 \quad (8)$$

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

$$r(t+1) = K_D r(t), \quad 0 < K_D < 1 \quad (9)$$

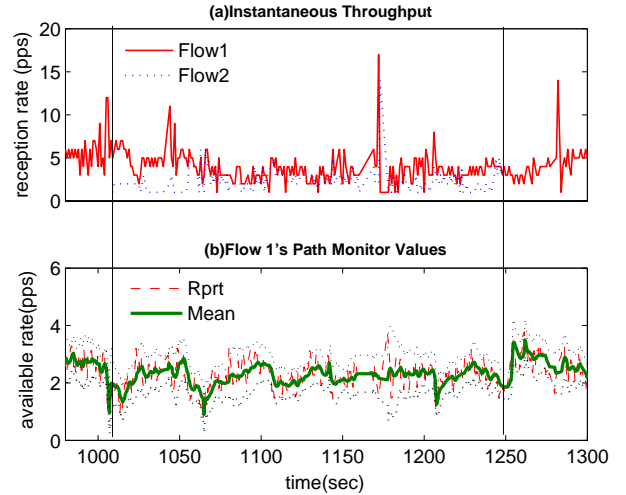


Fig. 3. Rate adaptation for two competing JTP flows.

The throughput and stability analysis of this controller can be found in the Appendix. Figure 3(a) demonstrates its

³Notice that this ensures that the destination does not send feedback/SNACK messages at a rate higher than the sending rate. Furthermore, the exact value of T_{Lower_Bound} and n is application-dependent—for example, the voice application module would use a lower value for T_{Lower_Bound} and n to meet the timing constraints of voice delivery.

convergence to fairness for two competing flows—a long-lived flow 1 is shown to adapt to a short-lived flow 2 which starts and ends at times 1000 and 1250, respectively. (See Section VII for simulation details.) Figure 3(b) demonstrates the behavior of the flip-flop path monitor of flow 1—the average available path rate, together with the control limits, as well as the instantaneous path rate values collected (reported) by JTP packets, are shown. We observe how the average value catches up with the instantaneous reported values as the monitor switches to the agile EWMA filter, so that the JTP source quickly backs off or increases its rate accordingly. We also observe other times where the monitor switches to the agile EWMA filter in response to varying network conditions, thus allowing the flow to quickly adapt.

We note that the JTP destination also limits the sending rate by its delivery rate up the stack to the receiving-side of the application. Furthermore, as discussed later, the source adapts its sending rate down to accommodate the retransmission rate of its packets by intermediate nodes.

2) *Energy Budget Controller*: Let $e_{UCL}(t)$ be the current upper control limit of the flip-flop path monitor for the energy consumption of individual packets. The energy budget e that is reported back to the source is computed as follows:

$$e(t+1) = \beta e_{UCL}(t), \beta > 1 \quad (10)$$

where β is a parameter defined by the *application module* that denotes the importance of each packet, since the energy budget controls the extra effort the network should invest in the delivery of each packet under transient surges in energy consumption, or in the case of route failures. β should be greater than one so that the path monitor is able to detect outliers.

VI. CACHING

JTP employs in-network caching of data packets to avoid end-to-end (source) retransmissions as much as possible. Upon receiving a traversing ACK packet, a node checks whether any packet(s) requested for retransmission, as expressed in the Selective Negative ACK (SNACK) field, exist in its local cache. Requested packets found in the cache are forwarded downstream toward the destination.⁴

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

⁴We note that efficiency achieved by catching and repairing errors earlier using such in-network caching does not contradict the end-to-end argument of system design [10]—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.

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. Energy vs. Path Length

As the lengths of paths from sources to destinations increase, the gain obtained from locally recovering packets increases as well, making JTP more scalable. To evaluate the benefits of in-network caching in JTP, we ran experiments with linear topologies of varying size, activating and deactivating the caching functionality (see Section VII for simulation details). Figure 4(a) shows the percentage of energy overhead introduced if in-network caching were not used.

Figure 4(b) shows the energy expenditure of each node on the path for an 8-node linear topology. As expected, nodes closer to the source suffer more from the end-to-end retransmissions, and thus experience higher energy overhead of up to 23% when caching is *not* used. The use of local recovery from caches also results in a more equal energy allocation over all the nodes along the path, since the burden of recovering lost packets is distributed among all nodes. In an arbitrary topology network, where flows are randomly distributed between nodes, the role of in-network caching is essential for equally saving energy at each node.

An analysis of in-network caching gain can be found in the Appendix.

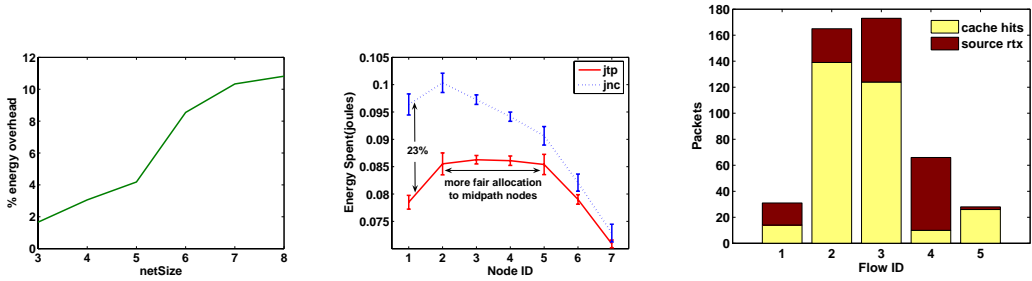
B. Controlling Routing Symmetry

In order for caching to yield maximum energy savings, communication paths must be symmetric. Recall that JAVeLEN employs next-hop routing to forward packets using increasingly more accurate topological views toward destinations. Next-hop routing also has lower overhead compared to source routing approaches. Thus, one way to force routes to be symmetric is to ensure that link metrics are symmetric. Recall that the link metric in JAVeLEN is a quantized function of signal-to-noise ratio. To increase metric symmetry, JTP employs a contention-avoidance flow rate control mechanism, thus it minimizes contention perceived at both nodes of a link. Together with quantization, this maximizes the chances of symmetric link costs and thus of symmetric routes.

Figure 4(c) shows the number of cache hits and source retransmissions for 5 JTP flows over a 25-node random topology. (See Section VII for simulation details.) The symmetry of paths is demonstrated by the in-network cache hits experienced by flows, resulting in more than two-third of the lost packets recovered from caches rather than from sources. This in turn, results in lower energy consumption per delivered application data bit compared to TCP and ATP—e.g. JTP consumed 1.95×10^{-5} Joules per bit, whereas TCP consumed 2.5×10^{-5} . Furthermore, JTP delivered more data to the applications—JTP delivered 3.5Mb, whereas TCP delivered 1.8Mb.

C. 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



(a) Percentage of energy savings from using local caching. (b) Energy spent per node in an 8-node linear topology. (c) Cache hits vs. end-to-end retransmissions for a 25-node random topology.

Fig. 4. Energy Gains from In-network Caching

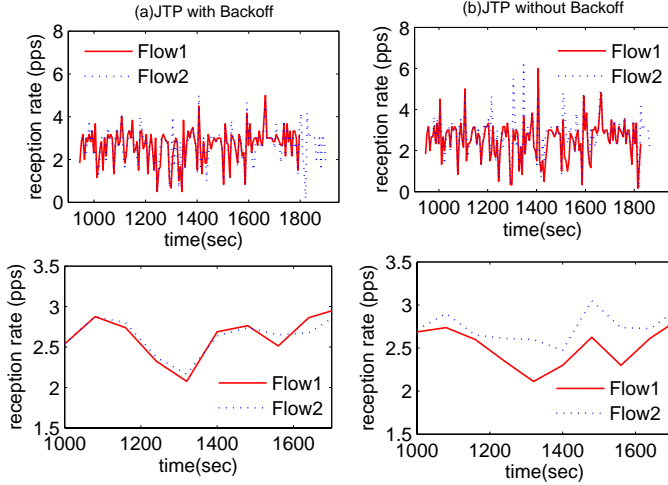


Fig. 5. Short- and long-term average of the reception rate for two competing flows: (a) the source backs off for locally recovered packets; (b) it does not.

and congestion control, the source node must incorporate in-network retransmissions in its sending rate calculations. To this end, JTP 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:

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

Figure 5 shows the short- (top plots) and long- (bottom plots) term average of the packet reception rate (throughput) at the destination for two competing flows. Flow 1 does not request packet retransmissions (i.e. UDP-like flow), while flow 2 requires for all its packets to 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.

VII. PERFORMANCE EVALUATION

A. Simulation Parameters

In this section, we present results from the extensive evaluation and testing of JTP in the OPNET simulation environment [29]. The results shown are the average of five (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. Two types of topology are considered:

Static Linear Topologies: These were used to evaluate performance for various path lengths. The source and the destination of two competing flows were 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 low (good quality) and high (bad quality). Each link is in bad state approximately 10% of the time. The average duration of the bad period is 3 seconds.

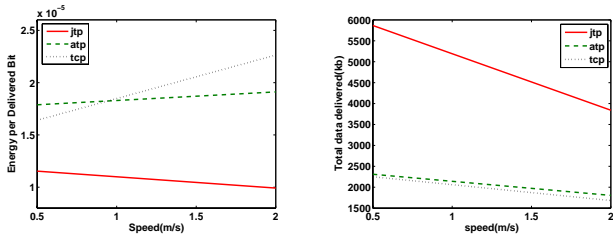
Dynamic Random Topologies: In these experiments, 25 nodes were randomly placed in a field of $1000m \times 500m$. Nodes move at speed varying from $0.5m/s$ to $2m/s$. There are 5 flows with random source-destination pairs.

Since none of the following protocols, against which we compare JTP, support adjustable reliability, JTP results are for 0% loss tolerance.

TCP-SACK: In order to have a more competitive performance, we compare against a rate-based flavor of TCP-SACK, whereby the rate of each flow is set by the well-known throughput equation of TCP [30]. Thus we remove the artifacts from the window-induced burstiness of the data and ACK streams. The SACK version helps TCP selectively retransmits 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 feedback from the receiver. The feedback period is set to be larger than RTT as suggested for ATP [6].

UDP: In some results we also include the UDP protocol, as a constant-rate, minimum-requirement, protocol that does not request for retransmissions and have no sending rate adaptation mechanism.



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

Fig. 7. Results for Random Topologies with Mobility.

B. Performance Metrics

- *Energy per delivered bit*: This measure captures the system-wide energy consumed to deliver each data bit to applications.
- *Variance in nodes' energy consumption*: A lower value indicates a more uniform energy consumption, and thus for a given initial total energy equally distributed among the nodes, a lower value extends the “lifetime” until the first node dies.
- *Goodput*: This measure captures the total rate at which the network delivers data to the applications, and thus represents how efficient the network resources are used.

C. Results

1) *Linear Topologies*: Figure 6(a) shows the energy per delivered bit for each protocol for varying network sizes. JTP is the most energy efficient protocol—even more efficient than UDP, which is a minimum-effort protocol with no control messages, nor retransmissions. This indicates that the extra energy induced by JTP in the network yields disproportionate gains to the applications.

Figure 6(b) shows the total energy consumed by each node in an 8-node linear topology. Figure 6(c) shows the variance in energy consumed by each midpath node for various network sizes. It is clear that JTP not only minimizes the energy consumption but also alleviates, through in-network caching, the unfairness in energy consumption introduced by end-to-end (source) retransmissions.

Figure 6(d) demonstrates that JTP not only provides great energy savings, but also achieves higher goodput. Without sacrificing system's performance, JTP minimizes feedback control messages, which in a wireless network environment, effectively “steal” bandwidth from users' data.

2) *Random Topologies*: Figure 7(a) shows that JTP consumes the least energy per delivered application data bit in a mobile environment. Figure 7(b) shows that the total data delivered to applications, as expected, drops as the speed of nodes increases. As mobility increases, the number of network control messages induced to keep the topology connected increases. Thus, transport protocols which generate frequent feedback traffic, such as ATP and TCP, suffer due to increased contention.

VIII. CONCLUSIONS AND FUTURE WORK

We presented the design and evaluation of an energy-conscious transport protocol (JTP) that targets low-power

ad hoc networks. JTP combines several features, notably application-determined per-packet energy budgeting, lightweight multi-hop caching support, and variable destination-controlled feedback. In this paper, we focused on bulk data transfers with varying reliability requirements. Future work includes other applications (e.g. voice, images), and other in-network energy-aware algorithms (e.g. for cache replacement, scheduling, and short-term “deflection” routing).

APPENDIX

Stability Analysis of PI²/MD Sending Rate Controller

Consider a single JTP flow adapting its sending rate over a fixed-capacity channel. For analytical tractability, let's ignore the EWMA computation of the available rate, that is, if $r(t) < C$, then the JTP source adapts its rate as follows:

$$r(t+1) = r(t) + K_I \times \frac{(C - r(t))}{r(t)} \quad (11)$$

On the other hand, if $r(t) > C$, then the JTP source adapts its rate as follows:

$$r(t+1) = K_D \times r(t) \quad (12)$$

Observe that the system remains non-linear, with two operating regions determined by whether the sending rate $r(t)$ is less than or greater than the capacity C . We next consider each of these two regions, and prove stability by showing that the value of a positive Lyapunov function $V(r)$ decreases with each iteration.

- *$r(t) < C$ Region*: Define $V(r) = C - r$. Then:

$$\begin{aligned} V(r(t+1)) - V(r(t)) &= (C - r(t+1)) - (C - r(t)) = \\ &= C - \left(r(t) + K_I \times \frac{C - r(t)}{r(t)}\right) - (C - r(t)) \\ &= -K_I \times \left(\frac{C}{r(t)} - 1\right) < 0 \end{aligned}$$

Thus, the only condition for $V(r)$ to decrease is that $K_I > 0$, regardless of the exact value of K_I . Of course, the exact value of K_I determines the tradeoff between speed of convergence and quality of the steady-state behavior—a higher value of K_I leads to faster convergence but higher oscillations.

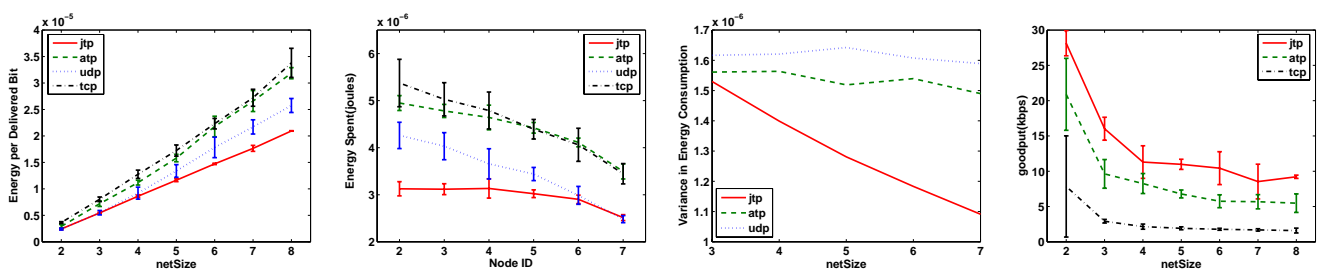
- *$r(t) > C$ Region*: Define $V(r) = r - C$. Then:

$$\begin{aligned} V(r(t+1)) - V(r(t)) &= (r(t+1) - C) - (r(t) - C) = \\ &= K_D \times r(t) - C - r(t) + C = -r(t) \times (1 - K_D) < 0 \end{aligned}$$

Observe that, for $V(r)$ to decrease, it is required that $K_D < 1$.

Thus, $K_I > 0$ and $K_D < 1$ are sufficient conditions for convergence. Furthermore, at steady-state as $t \rightarrow \infty$, substituting $r(t+1) = r(t)$ in Equation (11), we have $r(t) \rightarrow C$, hence the rate control is efficient.

Observe that in the case of lower frequency of sending-rate update, the above analysis still applies, *i.e.* the system converges albeit at a slower pace.



(a) Total energy expended per application data bit delivered. (b) Energy expended by each node normalized by total delivered bits in an end-to-end node linear topology. (c) Variance in energy consumption across nodes of a path for varying path length. (d) Average system-wide delivery rate.

Fig. 6. Results for Linear Topologies.

Analysis of In-network Caching Gain

We analyze the cost in terms of the total number of link-layer transmissions required to successfully deliver a total of k packets over H links from the source to the destination. Let n be the maximum number of link-layer transmissions allowed. Let p be the probability that one link-layer transmission fails; p is the same for all the links. The probability that a packet is successfully transmitted over link i is $P_s = 1 - p^n$. The probability that a packet is successfully transmitted from the source to the destination is $P_{e2es} = (1 - p^n)^H$.

- **JTP with in-network caching:** 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 node which has successfully received it.

We compute the expected total number of link-layer transmissions, denoted by $E[T_{tot}^{JTP}]$. The presence of in-network caching implies that each packet will be retransmitted over each link for as many times as needed, until it is successfully delivered to the next node. Thus, the expected number of link-layer transmissions follows a geometric distribution with mean:

$$E[T_i^{JTP}] = \frac{1}{1-p}$$

So each packet needs on average $H \times E[T_i^{JTP}]$ link-layer transmissions to be delivered at its destination. The expected total number of link-layer transmissions required by JTP in order to deliver k packets is thus given by:

$$E[T_{tot}^{JTP}] = k \times H \times \frac{1}{1-p} \quad (13)$$

- **JTP without caching:** In the case of JTP with no in-network caching (henceforth denoted by JNC), over each link, the packet is transmitted at most n times. If its transmission still fails, then it must be retransmitted from the source.

Denote by $M > k$, the random variable representing the number of packets sent by the source until k packets are successfully delivered at the destination, then $E[M] = \frac{k}{P_{e2es}}$.

When a packet is received at a node, the average number of link-layer transmissions that it triggers is:

$$\begin{aligned} E[T_i^{JNC}] &= (1-p) + 2(1-p)p + \dots + n(1-p)p^{n-1} + np^n \\ &= \frac{1-p^n}{1-p} \end{aligned}$$

Given that the link success probability is P_s , the probability that a packet makes it over i links is P_s^i , which then triggers $E[T_i^{JNC}]$ link-layer transmissions. Thus, the total number of link-layer transmissions for JNC is given by:

$$\begin{aligned} E[T_{tot}^{JNC}] &= \sum_{i=0}^{H-1} E[M] \times P_s^i \times E[T_i^{JNC}] \\ &= \frac{k(1-p^n)(1-(1-p^n)^H)}{(1-p^n)^H(1-p)p^n} \approx \frac{k \times H}{(1-p^n)^{H-1}(1-p)} \quad (14) \end{aligned}$$

For $H = 1$, equation (14) degenerates to (13). Observe that the cost of JNC is $\frac{1}{(1-p^n)^{H-1}}$ times higher than that of JTP.

ACKNOWLEDGMENT

This work was funded by the Defense Advanced Research Projects Agency (DARPA/ATO) under AFRL contract number FA8750-06-C-0199. Content of this work does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

We would like to thank Isidro Castineyra for earlier discussions on the design of JTP. Thanks to Daniel Coffin, Will Tetteh, Fabrice Tchakountio, and Keith Manning for their help in interfacing JTP with the MAC and network layers. Thanks to Jerry Burchfiel, Prithwish Basu, and Bill Watson for their feedback which helped us improve the presentation. We would also like to thank all other members of BBN's Mobile Networking Systems department, especially Henry Yeh.

REFERENCES

- [1] I. Jawhar and J. Wu, "Quality of Service Routing in Mobile Ad Hoc Networks," *Resource Management in Wireless Networking (Springer)*, vol. 16, pp. 365–400, 2005.
- [2] C. Santivanez, R. Ramanathan, and I. Stavrakakis, "Making Link-State Routing Scale for Ad Hoc Networks," in *Proceedings of MobiHoc*, 2001, pp. 22–32.
- [3] A. Bakre and B. R. Badrinath, "I-TCP: Indirect TCP for Mobile Hosts," in *Proceedings of ICDCS*, 1995.
- [4] H. Balakrishnan, S. Seshan, E. Amir, and R. Katz, "Improving TCP Performance Over Wireless Networks," in *Proceedings of ACM MobiCom*, 1995.
- [5] J. Liu and S. Singh, "ATCP: TCP for Mobile Ad Hoc Networks," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 7, pp. 1300–1315, 2001.
- [6] 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.

- [7] V. Tsaoussidis and I. Matta, "Open Issues on TCP for Mobile Computing," *Wireless Communications and Mobile Computing*, vol. 2, no. 1, pp. 3–20, 2001.
- [8] J. R. et al., "Joint Architecture Vision for Low Energy Networking (JAVeLEN)—DARPA-ATO Connectionless Networking Program," 2004, BBN Technichal Report. [Online]. Available: www.jasonredi.com
- [9] 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*, 2006, to appear.
- [10] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-end Arguments in System Design," *ACM Trans. Comput. Syst.*, vol. 2, no. 4, pp. 277–288, 1984.
- [11] I. Stoica and H. Zhang, "Providing Guaranteed Services Without Per Flow management," in *Proceedings of ACM SIGCOMM*, 1999.
- [12] S. Raman, H. Balakrishnan, and M. Srinivasan, "ITP: An Image Transport Protocol for the Internet," in *Proceedings of Intl. Conference on Network Protocols*, 2000.
- [13] H. S. et.al, "RTP: A Transport Protocol for Real-Time Applications," 1996, RFC 1889.
- [14] F. Ye, S. L. G. Zhong, and L. Zhang, "GRAdient Broadcast: A Robust Data Delivery Protocol for Large Scale Sensor Networks," *Wireless Networks*, vol. 11, no. 3, pp. 285–298, 2005.
- [15] Z. Fu, X. Meng, and S. Lu, "How Bad TCP Can Perform in Mobile Ad Hoc Networks," in *Proceedings of ISCC*, 2002.
- [16] R. Ludwig, "Eliminating Inefficient Cross-Layer Interactions in Wireless Networking," Ph.D. dissertation, Technischen Hochschule Aachen, April 2000.
- [17] P. Sinha, N. Venkitaraman, R. Sivakumar, and V. Bharghavan, "WTCP: A Reliable Transport Protocol for Wireless Networks," in *Proceedings of ACM MobiCom*, 1999.
- [18] R. Krishnan, J. P. G. Sterbenz, W. M. Eddy, C. Partridge, and M. Allman, "Explicit transport error notification (eten) for error-prone wireless and satellite networks," *Comput. Networks*, vol. 46, no. 3, pp. 343–362, 2004.
- [19] 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.
- [20] 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.
- [21] 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.
- [22] 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*. New York, NY, USA: ACM Press, 1987, pp. 353–359.
- [23] R. Kravets and P. Krishnan, "Application-driven Power Management for Mobile Communication," *Wireless Networks*, vol. 6, pp. 263–277, 2000.
- [24] D. Barman, I. Matta, E. Altman, and R. E. 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.
- [25] C. Wan, A. T. Campbell, and L. Krishnamurthy, "PSFQ: A Reliable Transport Protocol for Wireless Sensor Networks," in *Proceedings of WSNA*, 2002.
- [26] R. S. et al., "Stream Control Transmission Protocol," 2000, RFC 2960.
- [27] Y. Liaw, A. Dadej, and A. Jayasuriya, "Estimating Throughput Available to a Node in Wireless Ad-hoc Network," in *Proceedings of MASS 2004*, Fort Lauderdale, Florida, 2004, pp. 555–557.
- [28] D. C. Montgomery, "Introduction to Statistical Quality Control", 5th ed. New York: John Wiley & Sons, 2005.
- [29] "OPNET," <http://www.opnet.com/>.
- [30] 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.