An Energy-conscious Transport Protocol for Wireless Ad Hoc Networks
*(while maintaining the sanity of layering/modularity)*

Abraham Matta
Computer Science
Boston University

Joint work with
Niky Riga (BU)
Alberto Medina (BBN)
Jason Redi (BBN)
Craig Patridge (BBN)
Joint Architecture Vision for Low Energy Networking

- System designed for Wireless Ad Hoc Networks
  - Dynamic topology
    - Mobility, nodes join/leave network
    - Variable link quality
  - Contention for wireless medium
    - Variable link capacity
  - Power constraints

- Objective
  - Network-wide energy efficiency
  - Meeting requirements of applications
JAVeLEN Overview

- Express QoS requirements
- Varying reliability semantics
- End-to-end service
- Hop-by-hop soft-state operations
- Energy budgets
- Cross-layer interactions
- and more...
- Hazy-sighted scoping
- Energy-conserving multi-point relays
- SNR-based path selection
- Pseudo-random wake-up schedules
- Slotted protocol
- Two-stage receiver
Energy-efficient Substrate

- JAVeLEN builds on prior work on energy-conscious scheduling and routing
- High-power receiving data radios “on” only when:
  - There’s data for them to receive
  - They are scheduled to be awake
- MAC is effectively collision-free
- Routing traffic minimized & routes symmetric
- Traffic-aware protocols
  - adaptive wake-up schedules + TDM-like MAC at high load
- JAVeLEN uses 100 times less energy than OLSR/802.11
- Transport should NOT waste energy with low-value transmissions
Energy-conscious Transport

- May fit into other architectures given interface with MAC gives:
  - Control over link-layer transmissions
  - Indication of available capacity of channel

- End-to-end service: eJTP
- Hop-by-hop soft-state operations: iJTP
- eJTP has an application-dependent module to tailor transmission parameters: error recovery, per-packet energy budget, ...
- eJTP supports variable receiver regulated feedback
- eJTP uses congestion-avoidance rate-based flow control
- iJTP supports in-network caching and retransmits
- eJTP backs off sending rate to account for cache retransmits - coordinated recovery & fairness
Previous Work on TPs

- **Standard transport protocols (TCP, UDP, SCTP, …)**
  - Not suitable for wireless (cause of a loss)
  - Not energy aware

- **Application specific protocols (RTP, ITP, …)**
  - eJTP also influences in-network decisions for energy conservation

- **MANET transport protocols**
  - end-to-end approaches (ATCP, WTCP, ELFN, …)
  - Rate based flow control (ATP, …)
  - Receiver based control (RCP)
  - Not energy aware

- **Transport protocols for sensor nets (PSFQ, …)**
  - application specific - not suitable for general purpose wireless networks
Ideally…

- How many / which packets to transmit?
  - Only the absolutely necessary for the application

- # control messages?
  - Only connection establishment / termination

- # link-layer transmissions per packet?
  - Only one per link

- When to transmit a packet?
  - Only when the link quality is best
JTP Main Design Choices

- Increase application’s control
  - Application-dependent modules

- Minimize acknowledgments
  - Receiver based control

- Avoid congestion
  - Rate based flow control

- Do hop-by-hop control
  - Dynamic Packet State
    - Control energy expenditure
    - QoS
  - Caching to minimize end-to-end retransmissions
Elements of JTP

- Data packets carry: effective available rate, loss tolerance, energy budget, energy used
- ACK packets carry: cumulative ACK and SNACKs
**JTP Components**

- **eJTP runs on the ends of connections**
  - Accessible to applications through socket interface
  - Implements application-dependent transport functions
  - Implements receiver-based explicit rate-based and energy-based control
  - Implements receiver-based error control

- **iJTP runs on intermediate hops**
  - Implemented as loadable plugins of the MAC protocol
  - **DPS plug-in**: updates loss requirement, energy used, and available throughput in JTP header
  - **Caching plug-in**: retransmissions of cached packets requested by SNACK from receiver

---

2/5/2007

CCW 2007, Pittsburgh, PA 1-10
Algorithm 1 PreXmit()
1: increaseEnergyUsed(packet);
2: if (packet.energyUsed > packet.energyBudget) then
3:     dropPacket(packet);
4:   else
5:     if firstDataTransmission(packet) then
6:         lossRate = getLinkLossRate(packet);
7:         setMaxDataTransmissions(packet, lossRate);
8:         updateLossTolerance(packet);
9:     end if
10:    rate = getAvailableRate(packet);
11:    packet.rate =
12:        MIN(packet.rate, rate/AverageLayerAttempts);
13:   end if

Algorithm 2 PostRev()
1: if (packet.type == DATA) then
2:     cachePacket(packet);
3: else if (packet.type == ACK) then
4:     retransmitPackets(packet.packets)
5:     updateACK(packet);
6: end if
Adjustable Reliability

- To decrease link-layer losses over wireless
  - Increase transmission power
  - Coding
  - Multiple link-layer retransmissions
- Expend energy to deliver only “needed” packets
- iJTP controls # link-layer attempts

Low Energy
Low Reliability

High Energy
High Reliability

Application end-to-end reliability requirement \( r \)

#link-layer attempts \( N(r, l_i) \)

Loss rate \( l_i \)
Destination-based Control

- Control messages are necessary
  - Recover from losses
  - Adapt to network changes
  - ? Transport!

- Use low frequency
- Send additional feedback if significant changes are detected
- Monitor path condition using Flip Flop Filtering

Low Frequency
  - Slow Adaptation

High Frequency
  - Fast Adaptation

Graph: Switch to agile filter

2/5/2007
Loss Recovery: Fair In-network Caching

- Avoid source retransmissions
- Explicitly inform source of cache retransmissions, so it backs off

(a) JTP with Backoff

(b) JTP without Backoff
Implementation

- eJTP and iJTP are written in BBN's shared code framework
  - Operating System Adaptation Layer (OSAL): wrapper functions for uCLinux/Linux 2.6.x and OPNET 12.x
- eJTP as a Loadable Kernel Module in Linux
- iJTP as loadable plug-in of MAC in RTLinux
- Tested over JAVELEN radios, and in OPNET simulation
iJTP implements two levels of error recovery

- iJTP sets a lower number of link-layer transmission attempts for less stringent reliability requirement
- iJTP recovers from its cache, packets that application is missing and still needs
Comparison

(a) Total energy expended per application data bit delivered.
(b) Energy expended by each node normalized by total delivered bits in a 9-node linear topology.
(c) Average goodput experienced by flows in the network.

JTP achieves up to 5x energy savings over TCP-SACK, and 2x over ATP

2/5/2007
Performance under Mobility

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

(b) Goodput delivered to applications.

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

Deploying local caches is still beneficial.
Conclusion

- Early, but significant, step in energy-conscious transport
- Tests consistently show that JTP yields higher goodput and consumes less energy
- Energy-awareness leads to non-intuitive solutions
  - Needed to manage energy hop-by-hop through packet’s loss tolerance
  - Coordinate e2e and local transmissions for fairness
- Develop new iJTP plug-ins / capabilities
  - deflection routing
  - Energy-conscious cache/memory management
  - QoS-aware lightweight resource allocation
- What if some nodes are malicious or non-conformant?