

DTNLite: Reliable Data Delivery in Sensornets

Rabin Patra and Sergiu Nedevschi

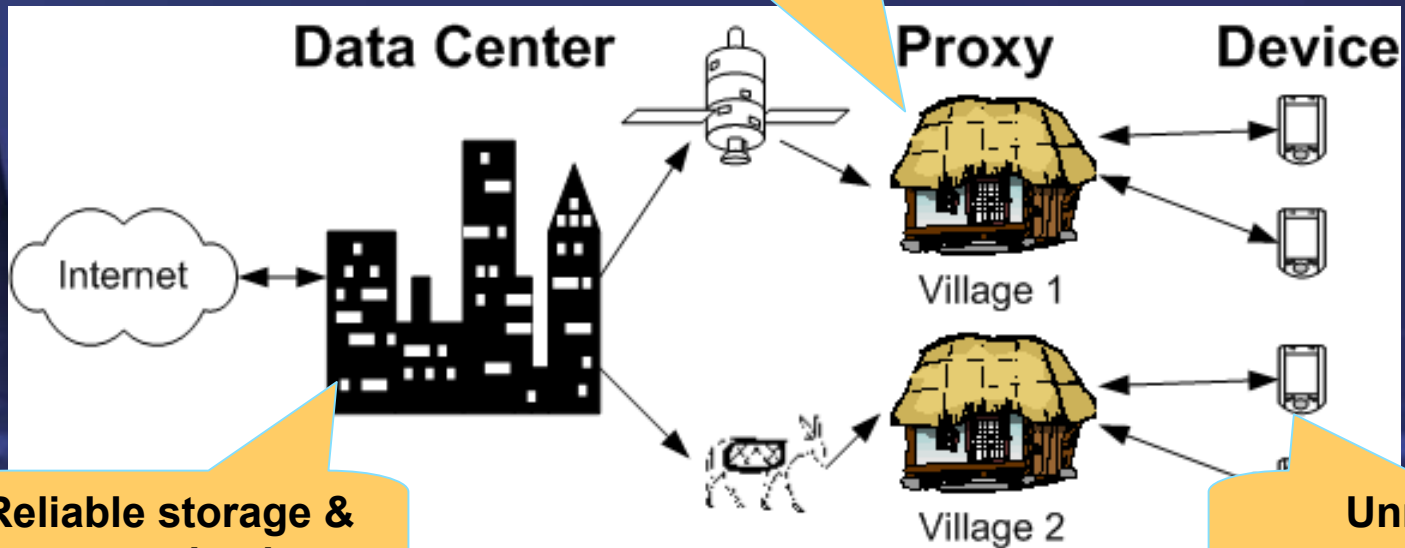
UCB Nest Retreat 2004

Why Reliable Data Delivery?

- Reliable Data Delivery – not much attention yet
 - Monitoring, directed diffusion, object tracking allow occasional data loss
- Interesting in perspective
 - Network reprogramming
 - Reliable and timely delivery of emergency event notifications
 - For large data units – reliable delivery of all packets
 - As sensornets become ubiquitous, reliability may be desired

ICT4B

- Information and communication technology for billions
- Issues
 - Cheap handheld devices
 - Cheap networking solutions
 - Proposed solution: tiered



Relaying/caching,
permanent storage

Reliable storage &
communication

Unreliable
storage/comm

DTN

- Communications between data centers and proxies – intermittent Communication mechanisms: DTN
- Delay tolerant networking:
 - Route messages efficiently and reliably in networks facing the following challenges:
 - Disconnection
 - High roundtrip delay
 - Fast evolving topologies
 - High error rates
- Links between proxies and handhelds?
 - DTNLite

Reliable Delivery Design Space

- Instruments for achieving reliability: **acknowledgement and retransmission**
- Who performs retransmission?
 - Source only
 - Unsuitable for high error rates, lots of retransmissions
 - Some nodes
 - Every node
 - Too many acks
- What is acknowledged?
 - Every packet
 - Too many acks
 - Selective acks (nacks)
 - Large delays due to reassembly at intermediary nodes

Existing Approaches

- *RMST*: Reliable MultiSegment Transport
 - Some nodes perform reassembly
 - In-network caching, selective nacks
- *PSFQ*: Pump Slow Fetch Quickly
 - Every node performs a special form of reassembly:
 - packets forwarded immediately if in-order, buffered otherwise
 - In-network caching, repair requests when out-of order packets received
 - Achieves small delay (\sim forwarding) and small number of retransmissions (\sim store-and-forward)

Stand-up to Challenges?

- Problems due to end-to-end acks:
 - If high-roundtrip delays, storage overflow at source
 - If no end-to-end acks: unreliable due to flaky nodes
- Problems due to storage in RAM:
 - If disconnections: buffer overflow
 - If large message sizes: impossible reassembly at intermediaries

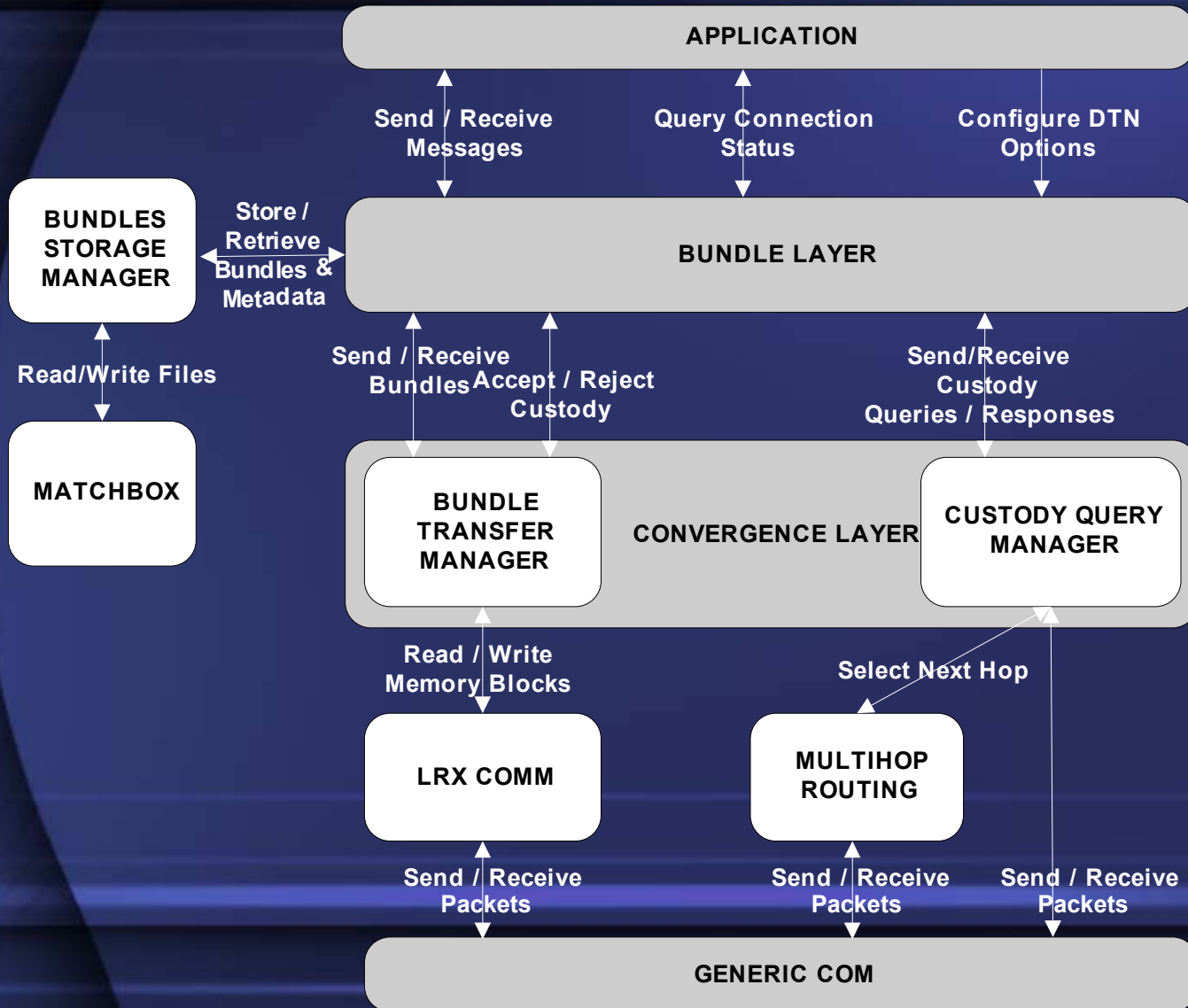
Proposed Mechanisms

- Store-and-forward using stable storage:
 - Prevents buffer overflows due to large roundtrip delays and disconnections
- Custody transfer as an alternative to end-to-end reliability:
 - ***Custody transfer: the acknowledged delivery of a message from one hop to another and the corresponding passing of reliable delivery responsibility.***
 - Storage at source freed earlier
 - Smaller end-to-end paths

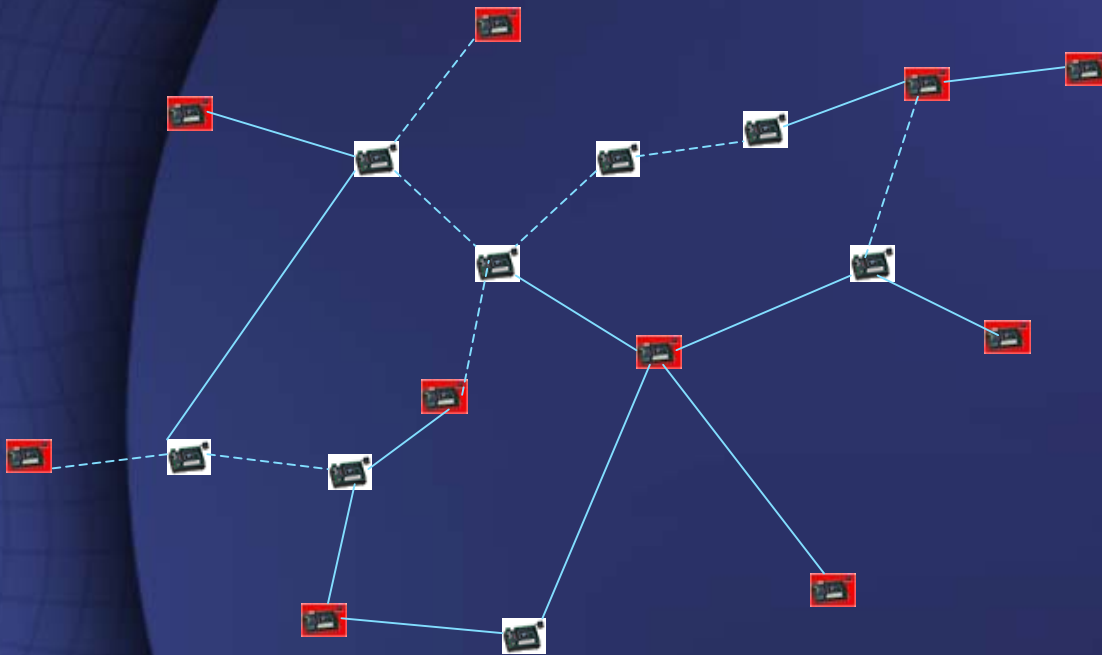
Design Decisions

- Support custody transfer
- Assume underlying tree-based multi-hop routing
- Use Application Data Units or “bundles” for custody transfers
 - reliable in-order packet delivery within a bundle
- Layered architecture :
 - separation between custody transfer mechanism and underlying multi-hop transmission mechanism
- Using Matchbox file-system for persistent storage
- Next custody hop discovery using custody query mechanism

DTNLite Architecture



Custody Transfer



- Custodians form an overlay
- Reliable message transfer between 2 custody hops:
 - Any of the reliable delivery protocols discussed
 - Handshaking for custody exchange
- Not fate sharing, end-to-end argument violation?
 - Content more important than node individuality
 - All nodes equally likely to fail

Issues with Custody Transfer

- Duplicate management
 - Wasted storage
 - Problems with aggregation, in-network processing
 - Duplicate-sensitive operations: *avg*
 - Duplicate-insensitive operations: *max*
 - Possible solution: broadcasting deletion packets with lists of already delivered ids?
- Custody transfer and congestion
 - Drop possible only at source
 - Care when accepting messages for transfer
- Choosing next custody hop

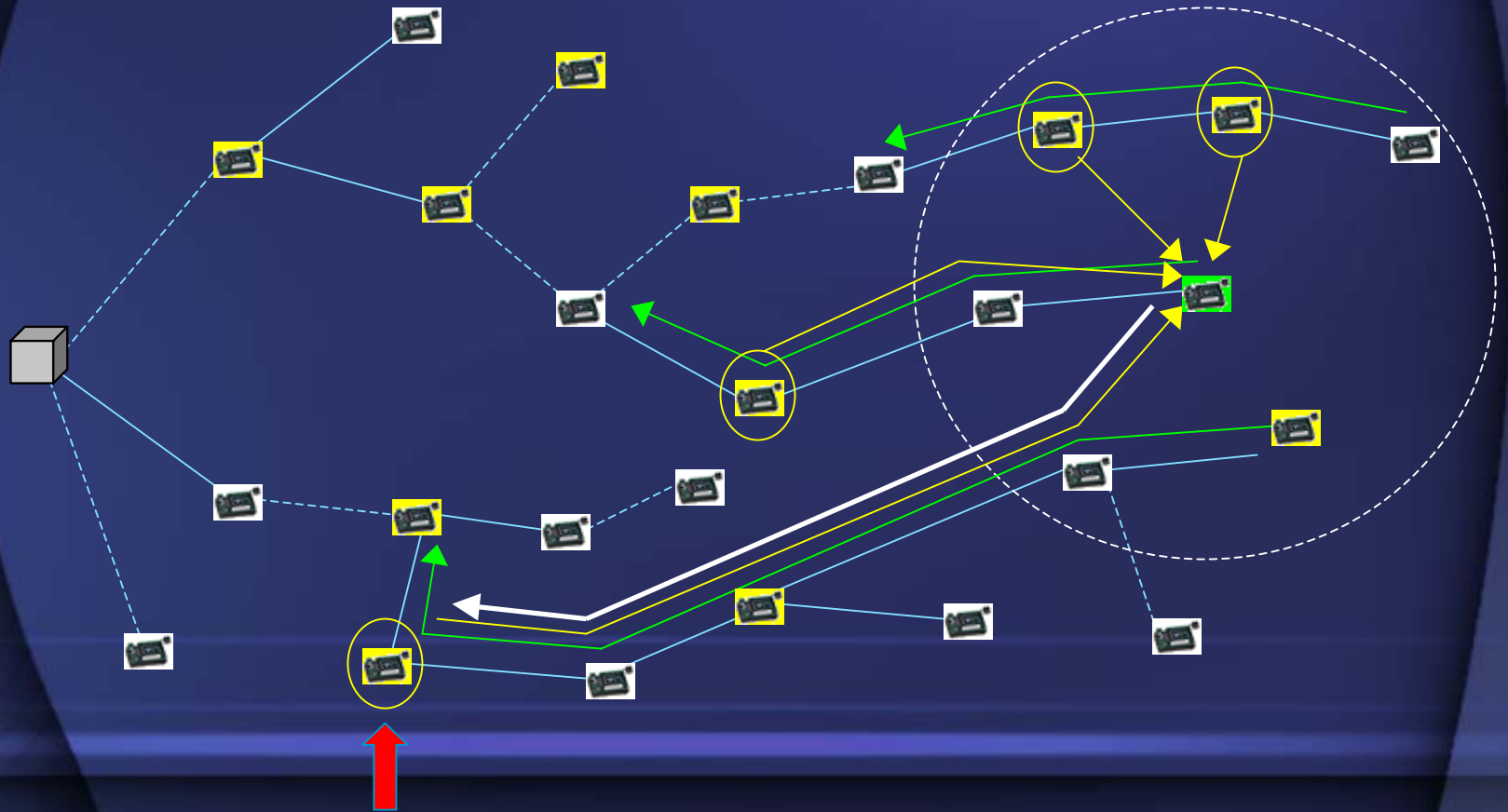
Choosing the Next Hop

- Simple metrics maintained locally:
 - Energy level remaining
 - Average delivery time
 - Average energy consumption for message delivery
- Maintaining metrics – 2 approaches
 - Distance-vector like
 - Maintained by basestation

Custody Query Mechanism

- Send query towards destination:
 - Unicast
 - Flood
 - Limited flood, followed by unicast
- Candidates answer with query responses
- Current custodian chooses best candidate, initiates custody transfer to candidate, using source routing

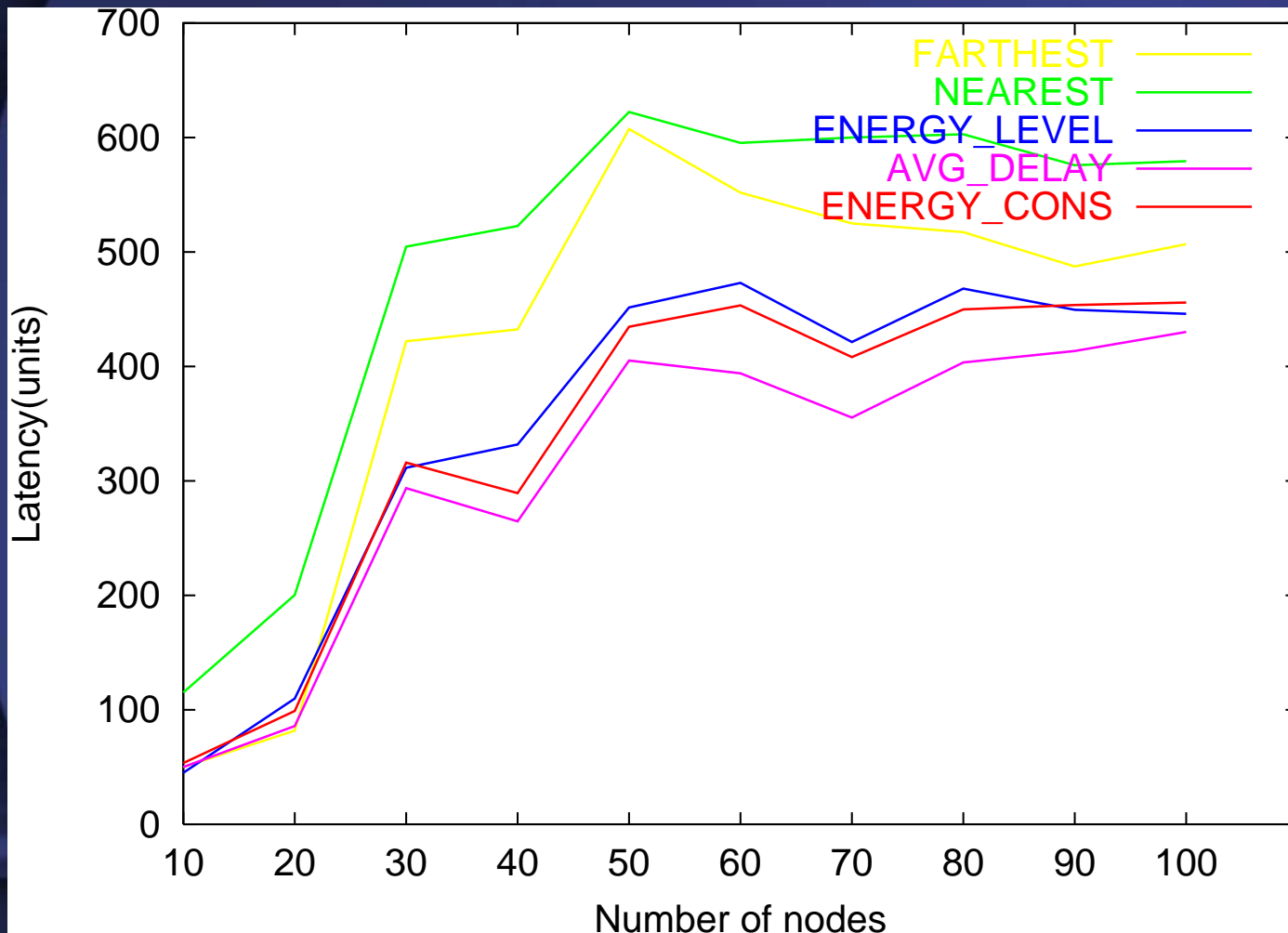
Example Custody Query



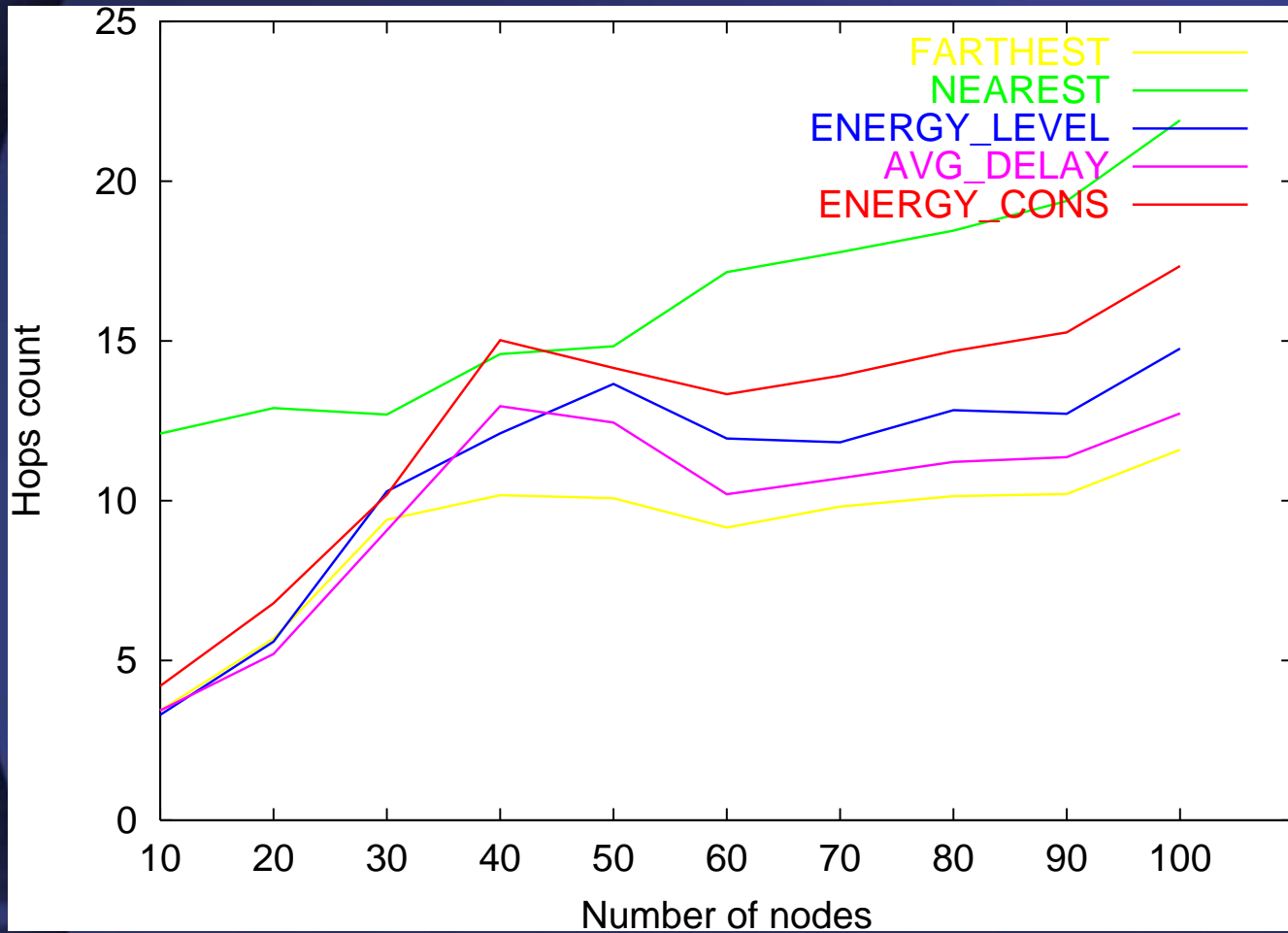
Evaluating Custody Transfer Policies

- Setup:
 - Discrete event based simulator developed for simulating routing algorithms for networks with scheduled disconnections
 - Adapted for unscheduled disconnections and collisions
 - Connectivity model: radius of connectivity
 - Link go up at exponentially separated intervals
 - Time link stays up: determined by link length
 - Simplified collision model: can only happen if packets destined for same node
 - Flooding level: 3
- Evaluated policies: farthest, nearest, max energy level, min average delivery time, min average energy consumption

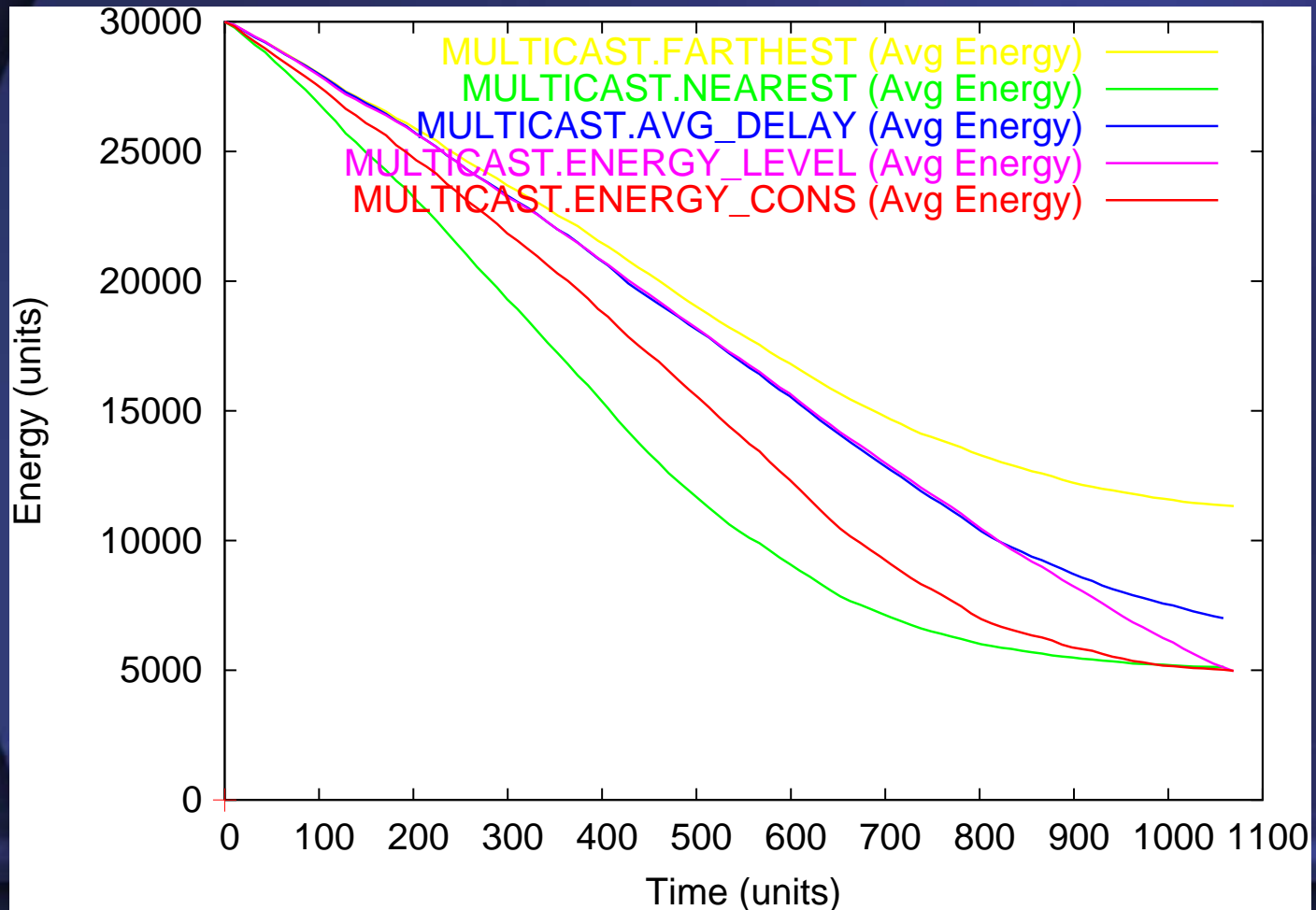
Average Latency Per Message



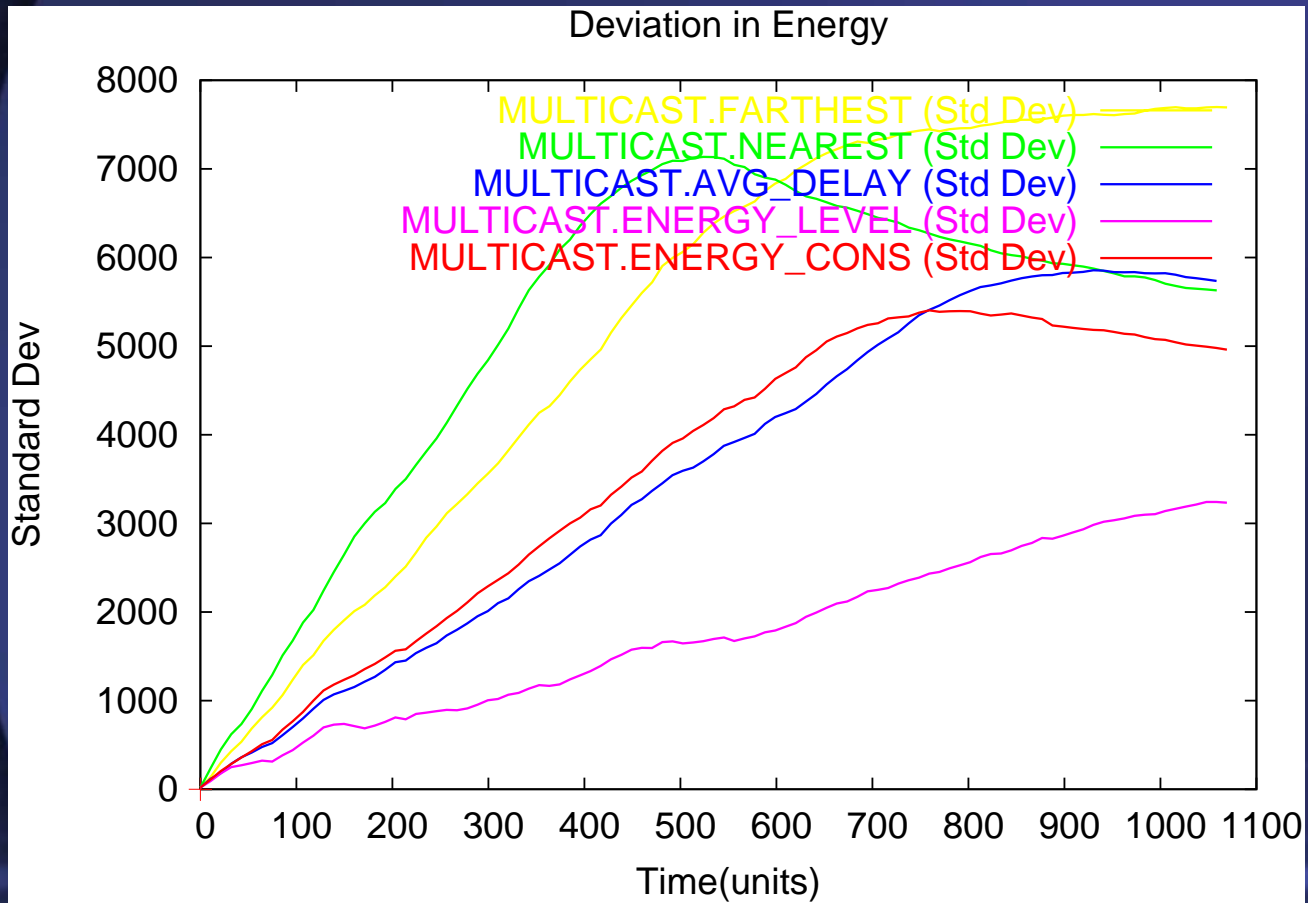
Hops Per Message



Average Node Energy



Standard Deviation



Conclusions & Future Work

- DTNLite implementation for TinyOS (to be delivered this month)
- Evaluations:
 - possible to optimize for a particular objective (energy, delay) using purely local information.
 - different selection policies able to exploit network asymmetries in connectivity and resources
- Future work:
 - comparative evaluation of our architecture with other reliable transfer protocols that have been proposed
 - perform extensive testing of the DTNLite implementation for TinyOS, to measure the overhead and performance of the custody transfer mechanism.
 - address other issues in reliable custody transfer (routing, duplicate detection, data aggregation)
 - Better metrics for custody hop selection