Congestion Control and Fairness for Many-to-One Routing in Sensor Networks

Cheng Tien Ee

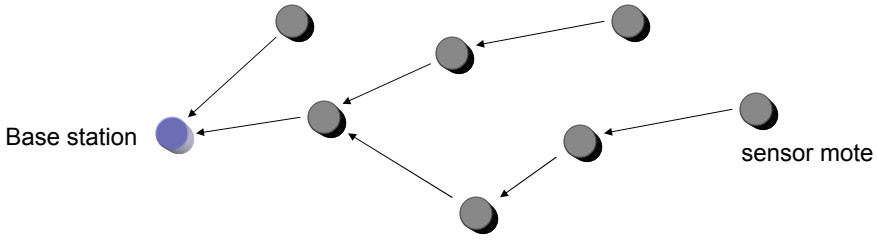
UC Berkeley, EECS

Presentation Outline

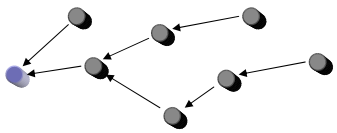
- Background
- Problems
- Congestion control
- Fairness
- Results
- Conclusion

Background

- Sensor motes gather data and send to central mote (base station)
- Motes too far from base station requires intermediate motes to relay, or route, data
- Routing structure formed is a tree, rooted at the base station

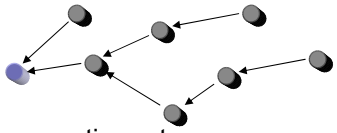


Problems



- motes closer to the base station has to transmit packets generated locally as well as those generated by downstream motes
- these motes likely to become bottlenecks in the system
- results in
  - more packets originating further away being dropped (*unfairness*)
  - loss of packets due to queue overflow and interference during transmission (*congestion*)
- unfairness may result in network not retrieving sufficient data from faraway motes to meet application requirements
- congestion wastes scarce energy resources

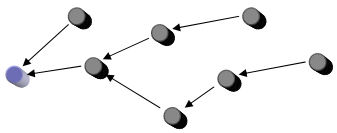
Possible Solution



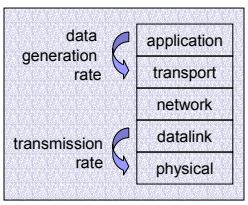
1. determine maximum application data generation rate
2. implement hop-by-hop Automatic Repeat Request (ARQ)

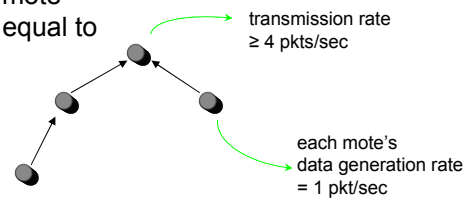
- why does this work?
  - since motes generate data at a rate network can handle, congestion (queue overflow) should not occur
  - ARQ ensures all packets ultimately reach the base station
- BUT
  - difficult to obtain maximum rate for every network configuration
  - underestimation of generation rate reduces effective bandwidth
  - what about transient congestion, eg. big metal truck goes by causing reflection etc. and thus interference
  - or temporary events that cause motes to generate more data than usual

Congestion Control

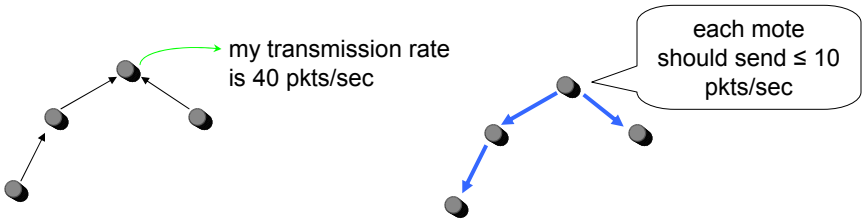


- we want the network to adapt itself
- first, distinguish between 2 different rates:
  - data generation rate: rate at which data is passed from application
  - transmission rate: rate at which packets, both locally produced and from downstream motes, are transmitted upstream
- next, transmission rate of parent mote should ideally be greater than or equal to data generation rate of all motes downstream



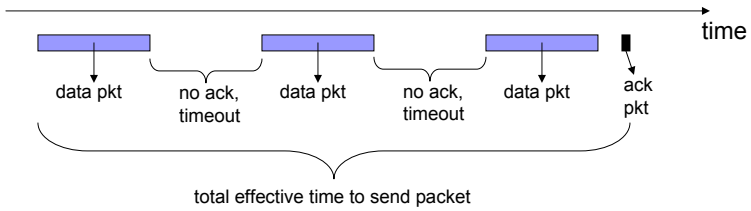


- basic idea: each mote
  - individually determines local maximum transmission rate
  - divide *transmission rate* by total number of downstream motes to give *data generation rate*
  - disseminate `min(own_rate, parent_rate)` downstream

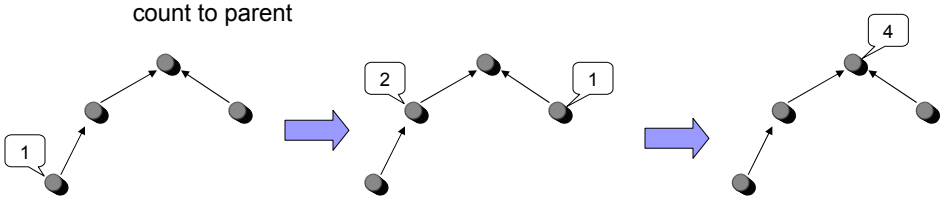


- why do this at each mote, why not just the base station?
  - because each mote's environment is different, may not be able to support parent mote's rate due to, for instance, interference

- so we need to:
  - determine effective transmission rate
  - determine number of downstream motes
  - disseminate data generation rate downstream
- possibly several ways to determine effective transmission rate
  - MAC layer keeps transmitting until reception has been confirmed, for instance via ACK packet
  - transmission rate is then inverse of total effective time to send packet



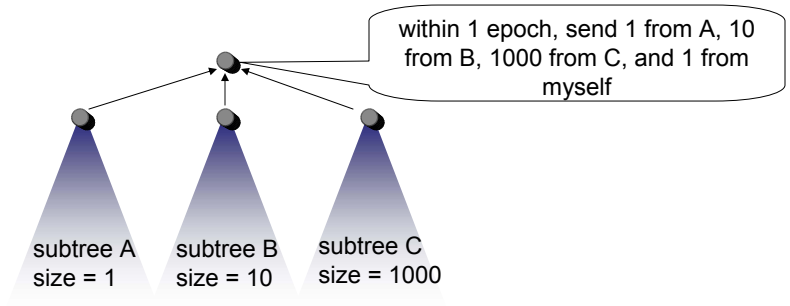
- determine number of downstream motes
  - use data aggregation technique for counting motes (Sam Madden's work)
  - each mote sums children's count, adds 1 (for itself), then transmits count to parent



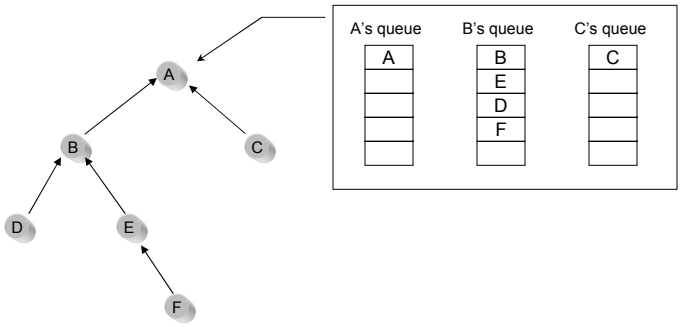
- disseminate data generation rate downstream
  - via control packets, or
  - piggy-backed on data packets

Fairness

- to be fair ⇒ at base station, same number of packets received from each mote
- basic idea: within each period of time (or *epoch*), transmit number of packets from each subtree equal to size of that subtree



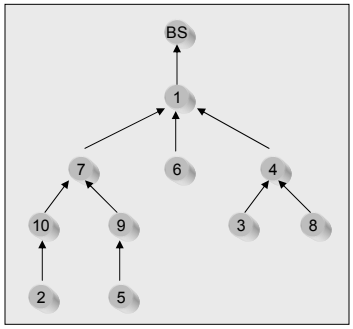
- requires:
  - per child queue (does not depend on size of subtree, so can be small and constant)
  - FIFO queues
  - subtree size (obtained as before)
- proof of correctness (by induction): see paper for more details



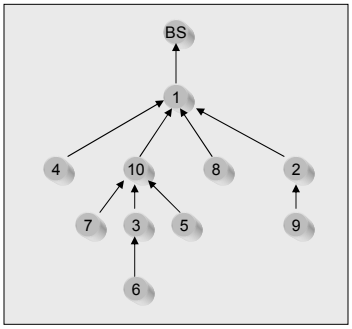
Results

- simulated algorithms
  - wrote simulator program
    - models interference
    - packet-level simulation
    - uses MACA as MAC protocol
  - check out results in paper
- also implemented on mica2dot motes
  - MAC protocol: MACA, with modified ACKs
  - 10 motes deployed indoors, within 15 feet of one another
  - let motes arbitrarily construct routing tree (algorithm is independent of tree structure)
  - compare with round-robin servicing of queues

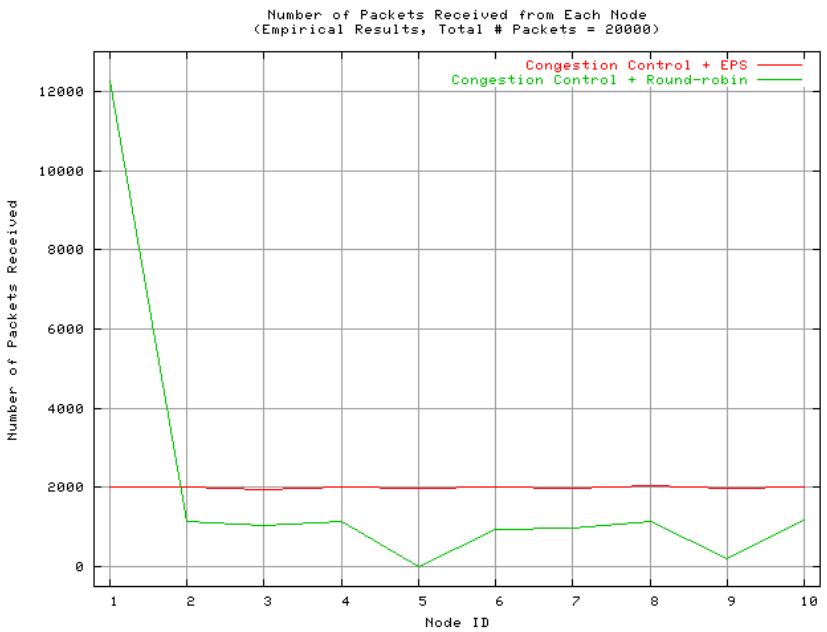
Routing Topologies



for Epoch-based Proportional Selection (EPS)



for round-robin servicing of queues



Conclusion

- network adapts itself automatically
  - impossible to manually configure 1000s of motes
- exact, same, simple code runs in each mote
  - reduces programming, debugging time
- very scalable
  - size of queues can be small, constant
  - state required increases linearly with number of neighbors
- congestion control and fairness can be implemented in transport layer, thus can be used with different MAC protocols
  - little or no modifications to MAC