I. Sensor Networks Background

Large number of small wireless devices
- CPU + memory + radio + sensors all on one chip
- Moore’s Law applied to make things smaller, not faster
  - is this really Moore’s Law?
- going toward “smart dust”

Major issues:
- power: battery, other options? In all cases, need to minimize power usage
- utilization should be low
  - enables soft real-time systems
  - required for power reasons
  - sufficient?
- individual motes can’t know everything
  - typically know only local information, e.g. neighbors
  - much like P2P systems!
- must deal with failures
  - motes fail
  - links are very flaky
  - questionable power => failures
  - need a probabilistic approach
- whole new network stack (not TCP)
  - why not TCP?
  - not even IP routing (why not?)
  - must exploit broadcast (and snooping)
  - must think about multiple paths for fault tolerance
  - must think about aggregation (limited bandwidth)
  - can really optimize across layers!
  - communication effectiveness not just based on distance...
- time sync is useful but hard
o sensors are noisy
  • especially if they are cheap
  • Can you get one good sensor out of lots of cheap sensors?
  • must be calibrated -- very hard to do well
  • sensors drift with time and often temperature
  • sensors interfere with each other
o event driven
  • underlying system is event driven
  • sensors, message arrivals, timers are the sources of events
  • relation to Macedon?

II. TinyOS

Component model:
  o wire up components
  o interface to interface
  o events/commands
  o wiring can be checked statically
  o easy to do interposition, replacement

Static memory allocation -- not fundamental, but seems useful
  o some apps do their own
  o exchange resources (like buffers)
  o prevents overflow, malloc errors
  o fits with underutilization model

There is no “OS” per se
  o application specific set of components
  o some common services (e.g. routing), but easily customized
  o no need to virtualize hardware! (or is there?)

Programmed in nesC, which is a C variant that supports components/wiring/interfaces
  o also detects many data races (but not all)

III. Single Hop

Based on active messages

small messages only -- need to build up streams, large transfers
sometimes the radio is in hardware, sometimes not
sometimes link-layer acks, sometimes not
low-power listening
variety of MAC layers, mostly CSMA, some work in time division

IV. Multi-hop Communication

Tree based
- very common for data collection
- may support aggregation
- uneven power use
- root may be bottleneck
- simple ways to build trees using broadcast

Dessimiation: broadcast or epidemic
- flooding is simple but inefficient
- need reliable broadcast, which is hard
- epidemic seems to work well, but depends on density
- can broadcast first, then fill in epidemically

any-to-any routing (harder)
- hard to do general-purpose routing
- one solution: up and down a tree
- landmark routing: route to landmark and then to destination (less state)
- geographic routing

Need to track viable neighbors
- the set changes over time
- asymmetric links?
- need to know how to choose a subset
- need to know about potential replacement neighbors
- need to cleanly support snooping
- snooping is at odds with turning the radio off...

Also need to deal with fragmentation, retransmission
V. Network Services

Power Management
- hard problem -- it is application specific and it touches all parts of the system (like security or correctness)
- easy part: an interface for turning components on/off
- hard part: when do to so without breaking anything!
- common use: low duty cycle globally synchronous applications (all on or all off)

Time sync
- also hard -- interferes with regular work (kind of like garbage collection does)
- app probably needs to control both the granularity and when the re-sync occurs..
- lots of neat algorithms for this; one based on broadcast to sync receivers with each other

VI. Other Stuff

Absent abstractions: cluster formation, receive queues

cross layer optimization: very important, in part due to application specific OS

not end-to-end (generally) so far...