Time, Synchronization, and Wireless Sensor Networks
Part II

Ted Herman
University of Iowa

Presentation: Part II

metrics and techniques

single-hop beacons

regional time zones

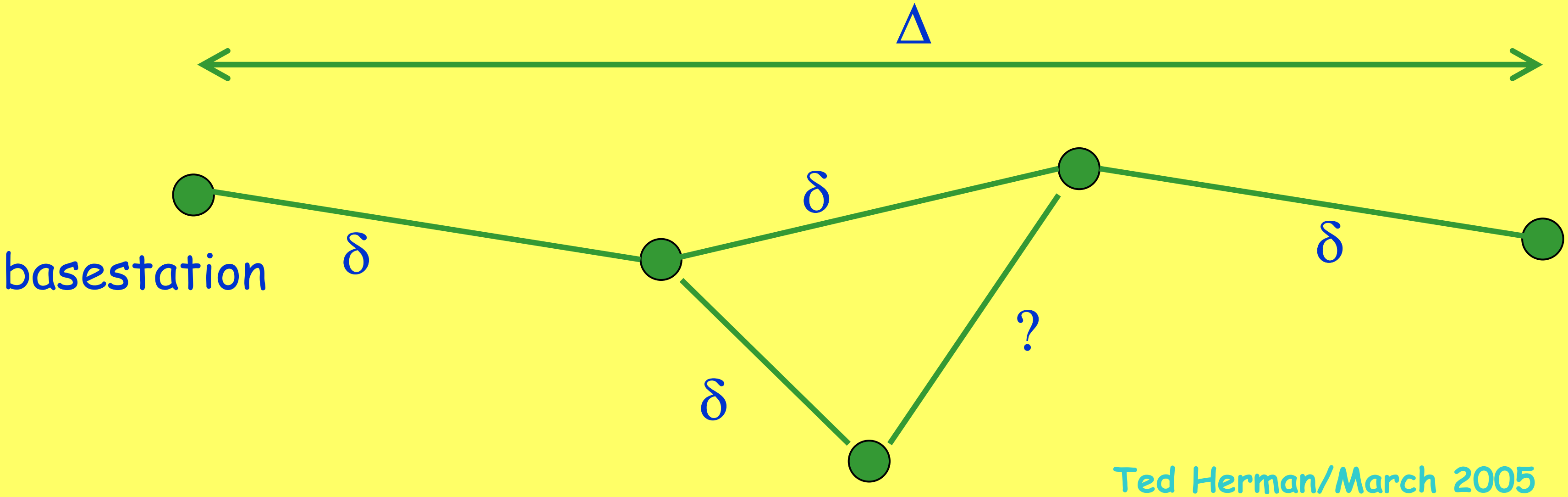
routing-structure and leader clock

uniform convergence

conclusion

Multihop Synchronization

- wireless sensor networks are multihop (sometimes ad hoc) networks
- measures of quality of synchronization:
  - δ-difference between *neighboring* clocks
  - Δ-difference between basestation and any clock
  - δ-difference along any path in routing tree

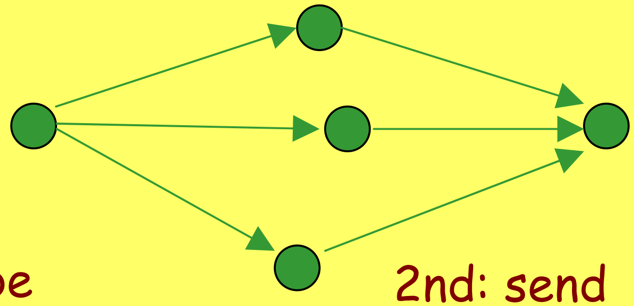


Synchronization Techniques

1. use GPS or radio beacon
   - requires special hardware, extra cost
   - $\Delta = \delta$
2. use only "regional time zones"
   - complicated time zone conversion gateways
3. use routing structure and leader clock
   - Δ = (distance) x δ
   - building, maintaining routing structure → fault tolerance issues
4. use uniform convergence to maximal clocks
   - similar metrics to routing structure, but different fault tolerance properties
5. other: biologically-inspired methods, phase "waves", time-flow algorithms (not yet practical)

How to Evaluate in Practice ?

- can use GPS for independent evaluation
  - useful to evaluate skew, not so useful for fast evaluation of offset synchronization
- self-sampling: nodes calculate difference between clock and time in a timesync message
  - large difference → lack of synchrony
- probes: single-hop broadcast, timestamped by all who receive, then transmit recorded timestamps and observe differences in the timestamps

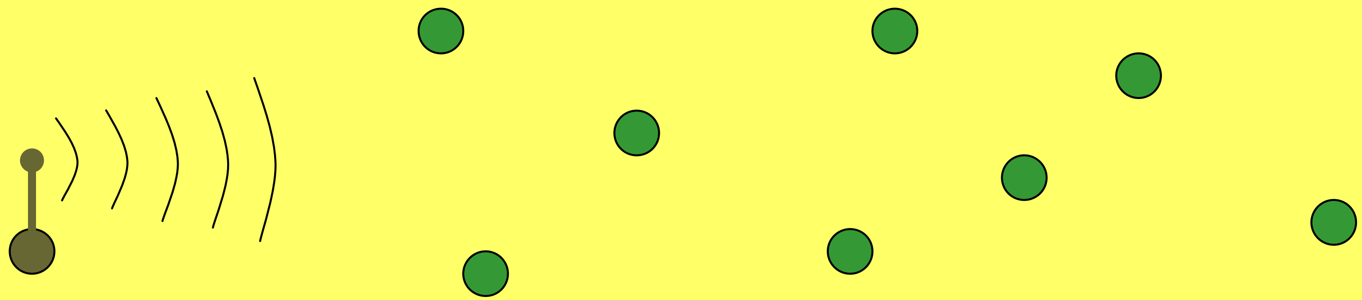


Presentation: Part II

- metrics and techniques
- **single-hop beacons**
- regional time zones
- routing-structure and leader clock
- uniform convergence
- conclusion

Single-Hop Beacon

- excellent performance
- single point of failure
- concerns of power, legality, stealth, assurance



- practical for open area, limited scale
- special hardware: tall antenna, strong signal
- basically using standard sensor hardware

Presentation: Part II

metrics and techniques

single-hop beacons

regional time zones

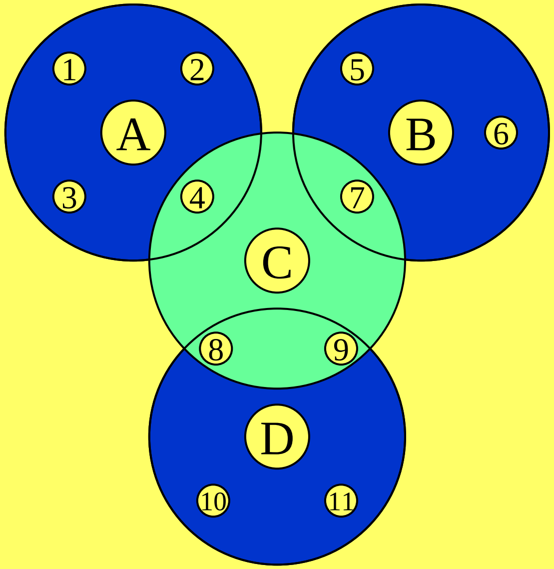
routing-structure and leader clock

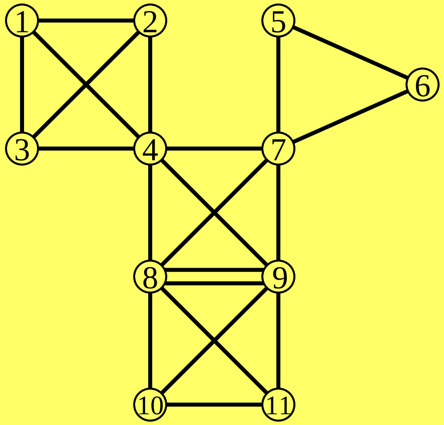
uniform convergence

conclusion

Regional Time Zones

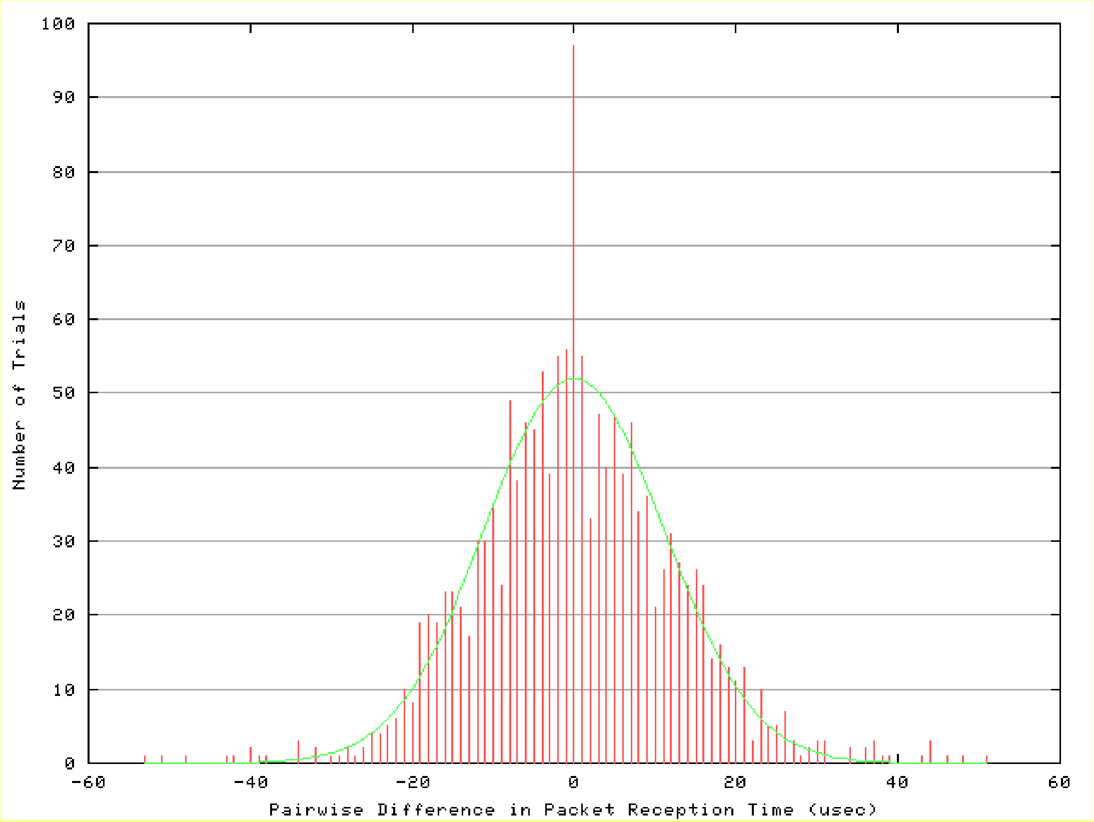
- proposed for RBS
  (Reference Broadcast Synchronization [Elson, 2003])
- use only "regional time zones"
- conversion adds complexity --- but useful if timesync not needed everywhere





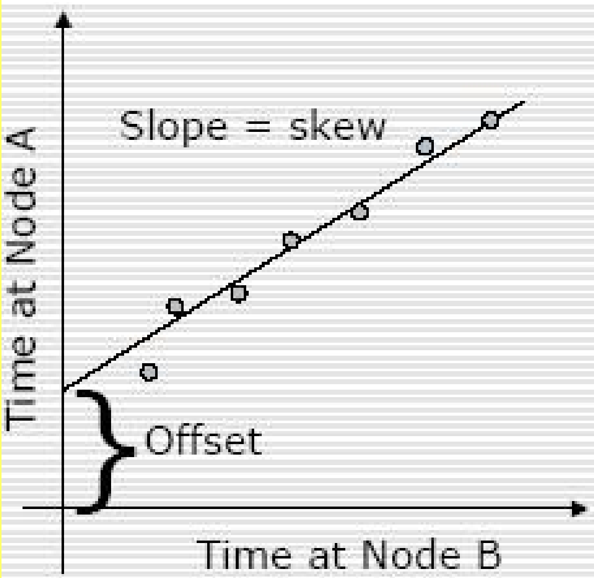
RBS Statistics

- multiple reference beacons, receiver-receiver synchronization forms distribution of noise



Noise Filtering

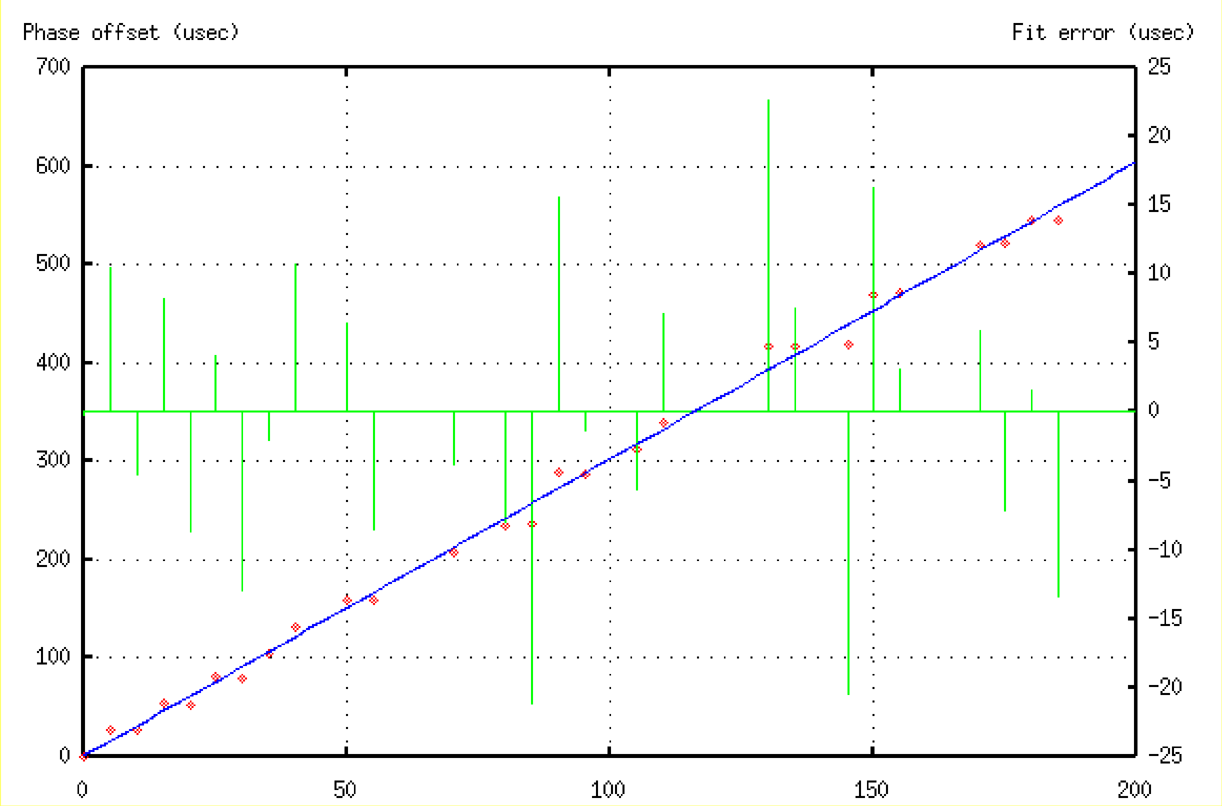
- elimination of noise by knowledge of distribution & error-minimizing hypotheses



- Over short time-scales, relative skew between clocks can be modeled as a constant frequency difference + noise.
- System of linear equations
  - $Y_1 = ax_1 + b$
  - $Y_2 = ax_2 + b$
  - ...
  - $Y_n = ax_n + b$
- Solve using linear least squares for minimum mean square error (MMSE) criterion
  - $X^T X a = X^T y$

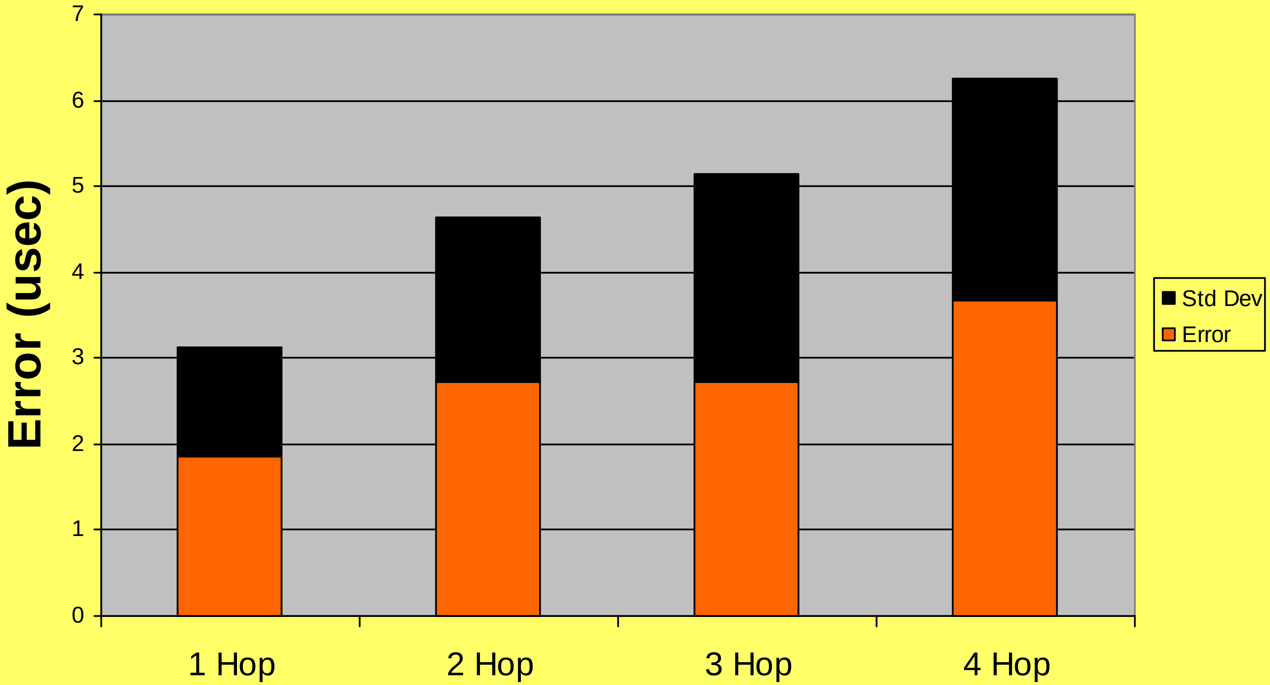
RBS Statistical Technique

- linear regression used to obtain best offset
- outlier removal would improve results
- linear regression also useful to correct skew



Multihop RBS results

- some results after conversion over multiple regions



- *better* than worst case ← some errors positive, some errors negative, so some errors cancel

Presentation: Part II

metrics and techniques

single-hop beacons

regional time zones

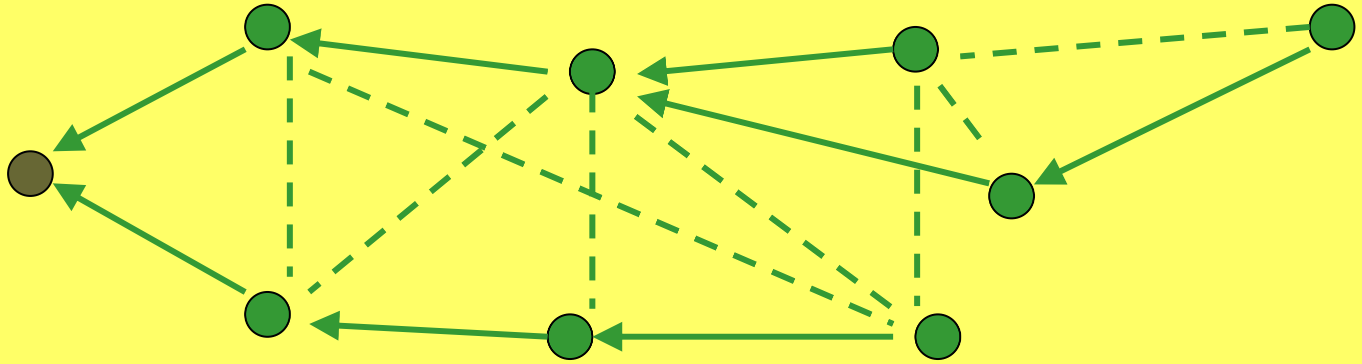
routing-structure and leader clock

uniform convergence

conclusion

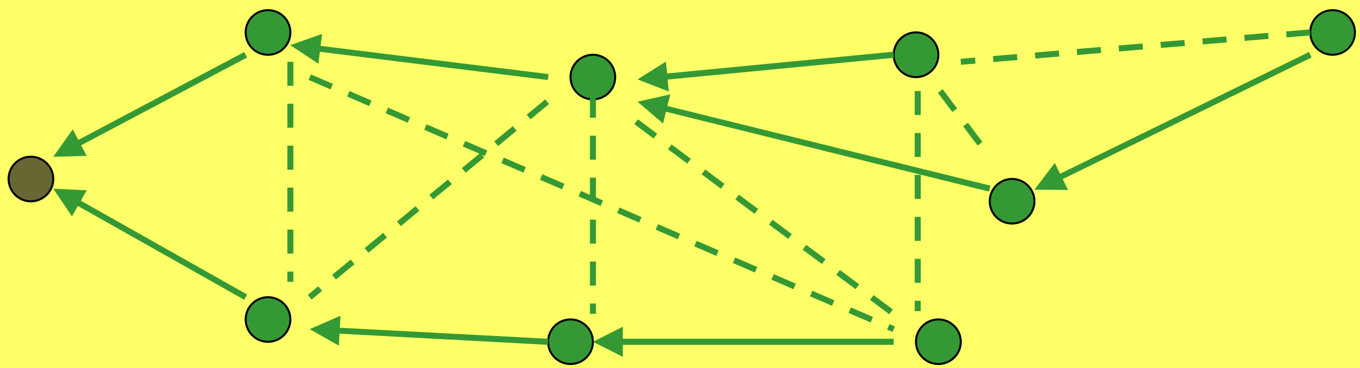
Rooted Spanning Tree

- popular routing structure
  - basestation at root
  - selection of links in tree based on Quality metrics
- other routing types: "fat tree", mesh, geographic



Leader Clock at Root

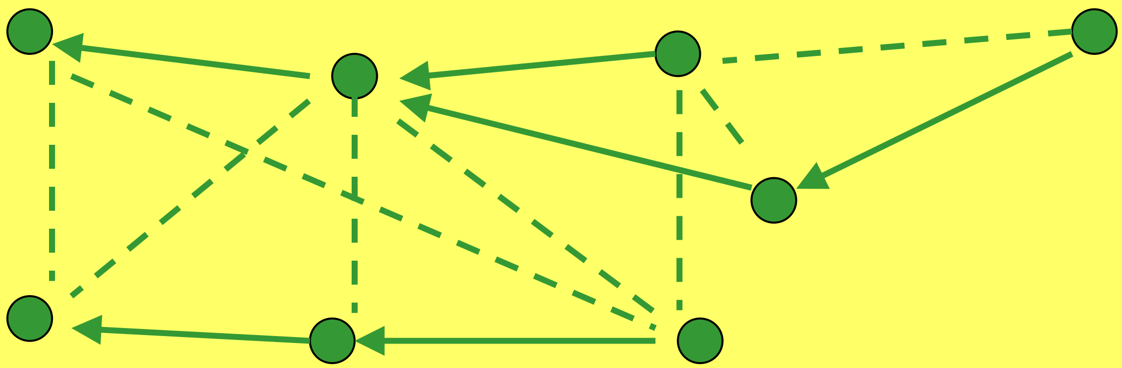
- everyone follow parent in tree
  - periodic timesync message to neighbors
  - collect many samples from parent (ignore others)
  - use linear regression to follow parent offset & skew



Leader Failure

- leader doesn't need to be basestation
  - if leader fails, recovery phase elects new leader
- leader election: leader is sensor node having smallest Id, parent is closest node to leader
- what happens when a node or link fails?
  - much like routing table recovery, look for new path to leader, eventually reach "threshold timeout" and then elect a new leader

no leader ...



Evaluation of Leader Tree

- generally excellent synchronization
  - however, strange cases can lead to $\delta \approx \Delta$
- low overhead, simple implementation
- rapid set-up for on-demand synchronization (if we use basestation as root)
- suited to sensor networks where links are stable & failures are infrequent
- does not handle sensor mobility

Presentation: Part II

metrics and techniques

single-hop beacons

regional time zones

routing-structure and leader clock

uniform convergence

conclusion

Uniform Convergence

- basic idea: instead of a leader node, have all nodes follow a "leader value"
  - leader clock could be one with largest value
  - leader clock could be one with smallest value
  - leader value could be mean, median, etc
- local convergence → global convergence
- send periodic timesync messages, use easy algorithm to adjust offset

```
if (received_time > local_clock)
   local_clock = received_time
```

Uniform Convergence Advantages

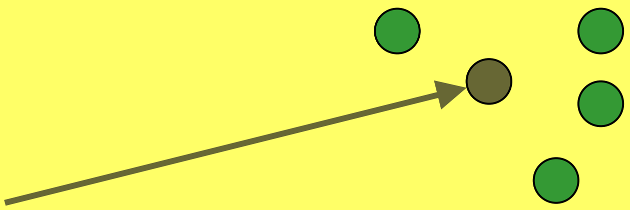
- fault tolerance is automatic
- each node takes input from <u>all neighbors</u>
- mobility of sensor nodes is no problem
- extremely simple implementation
- self-stabilizing from all possible states and system configurations, partitions & rejoins
- was useful in practice for "Line in the Sand" demonstration

Uniform Convergence Challenges

- even one failure can contaminate entire network (when failure introduces new, larger clock value)
- more difficult to correct skew than for tree
- how to integrate GPS or other timesource?
  - we can use a hierarchy of clocks for application
- what does "largest clock" mean when clock reaches maximum value and rolls over?
  - rare occurrence, but happens someday
  - transient failures could cause rollover sooner …

Preventing Contamination

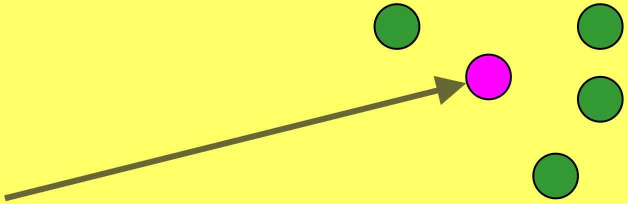
- algorithm: build picture of neighborhood



  - node p collects timesync messages from all neighbors
  - are they all reasonably close?
    - yes → adjust local clock to maximum value
    - no → cases to consider:
      - more than one outlier → no consensus, adjust to maximum value
      - only one outlier from "consensus clock range" →
        - if p is outlier, then p "reboots" its clock
        - if other neighbor is outlier, ignore that neighbor
  - handles single-fault cases only

Special Case: restarting node

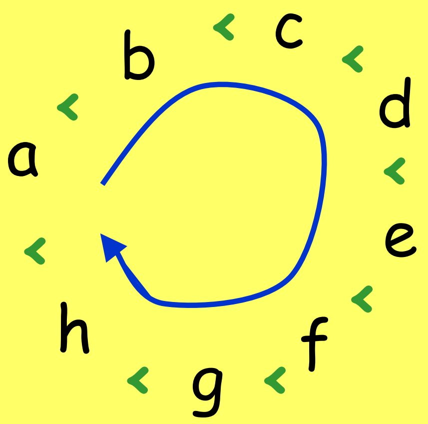
- algorithm: again, build picture of neighborhood



- node p joining network or rebooting clock
- look for "normal" neighbors to trust
  - normal neighbors → copy maximum of normal neighbors
  - no normal neighbors → adjust local clock to maximum value from any neighbor (including restarting ones)
  - after adjusting to maximum, node becomes "normal"

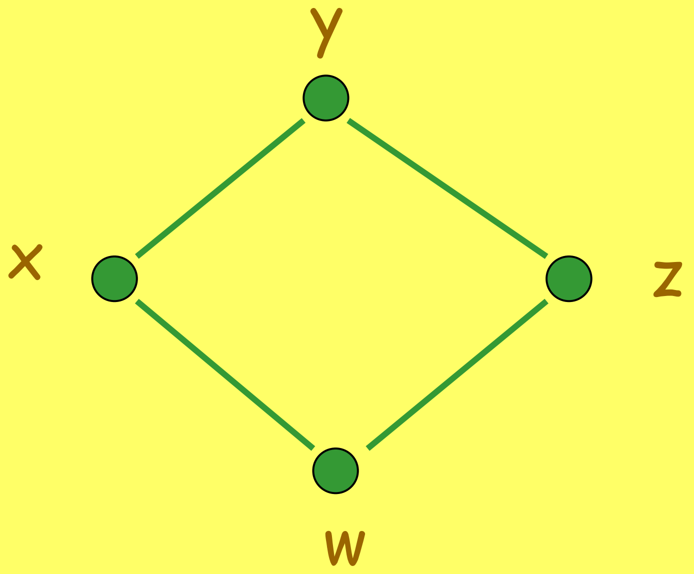
Clock Rollover

- p's clock advances from 2^{32}-1 back to zero
- q (neighbor of p) has clock value 2^{32}-35
  - question: what should q think of p's clock?
- proposal: use (<,max) cyclic ordering around domain of values [0,2^{32}-1]



Bad Case for Cyclic Ordering

- network is in "ring" topology
- values (w,x,y,z) are about ¼ of 2^{32} apart in domain of clock values → in ordering cycle
- maybe, each node follows larger value of neighbor in parallel → never synchronizing!



<u>a solution to this problem</u>

reset to zero when neighbor clocks are too far apart, use special rule after reset

Presentation: Part II

metrics and techniques

single-hop beacons

regional time zones

routing-structure and leader clock

uniform convergence

conclusion

Conclusion

- Part I
  - we saw how time sync has different needs & opportunities in wireless sensor networks than for traditional LAN/WAN/Internet
  - propagation delay often insignificant
  - special techniques to deal with radio/MAC/system delays

Conclusion

- Part II
  - some quite varied alternatives for how to synchronize in multihop networks
  - single-hop beacon (like GPS) good for some situations
  - time sync strategies can be similar to routing protocol structures (trees, zones)
  - time sync is a "local" property, so notions like uniform convergence may be useful

Conclusion

- Some Open Problems
  - how to choose a timesync algorithm based on application requirements ?
  - how to conserve energy in timesync ?
  - are there special needs for coordinated actuation, long-term sleeping, sentries, and low duty cycles ?
  - what kind of tools are helpful to use complicated timesync ideas, but make application design simple ?