An Adaptive Scheduling Method for Quasi-Periodic Sensor Traffic

Sharat C. Visweswara, Apurva C. Goel, Rudra Dutta
North Carolina State University

This research was partly supported in part by NSF grant # EEC-0332271
Outline

- Context
- Problem articulation
- Algorithm description
- Numerical results
Sensor Networks

- Class of ad hoc networks, for monitoring and controlling the environment
- Distinct characteristics
  - Egress traffic pattern
  - Immobile sensor nodes (sometimes)
  - Critical dependence on battery life
Power Saving Approaches

- Various approaches at different layers
  - Manipulate transmitter power, physical layer
  - MAC layer scheduling, switch-off transmitter
  - Optimize routing, networking layer
  - Data compression, other application layer techniques

- Cross layer approaches

- Minimizing idle listening
  - Switch off transceiver
  - Time to sleep computed from other layer considerations
Problem Definition

- Schedule transceiver switch-offs,
  - For *quasi-static* traffic
  - When synchronization cannot bound drift
  - Adapting to variations in mean period
  - For sensors with different period
  - In distributed, ad-hoc manner

- Motivating application
  - Structural health monitoring of bridges

- Contributions
  - Appropriate metric for performance
  - Model metric behavior with sleep
  - Design adaptive algorithm
Basic Algorithm Template

- Alternate between sleeping and waking
- Sleep
  - Until timer fires
- Awake
  - Stay awake until packet is received
  - Compute next wakeup time, set timer
  - Go to sleep
- Simplifying assumptions
  - Single sink
  - Continuous monitoring sensors
  - Roughly uniform density, representative circular area
  - Very low data rate
Modeling Variations in Period

- Quasi-static traffic pattern
  - Period is not precise - model with PDF
  - Unimodal, peaked PDF of inter-arrival times
  - Tail likely to be light, but may be long
  - Otherwise general

- Successive inter-arrival times
  independent
  - Follows from lack of synchronization
  - Important consequences
“Utility” and “Benefit”

- Usual metric: lifetime of network
  - However, assumed “all else being equal”
  - With light-but-long-tailed pdf, some loss inevitable
  - But lifetime can be increased indefinitely at the cost of loss
  - Need new metric reflecting both

![Quasi-periodic Traffic Diagram]

- No sleep, No loss.
- Some sleep, little loss.
- More sleep, lots of loss.
- 100% sleep, 100% loss.
“Utility” and “Benefit”

- What is the purpose of the network?
  - Get sensor data to sink

- How does lifetime help?
  - More time transferring data, at same rate

- Hence:
  - *Utility* of network: total data transferred over lifetime
  - *Benefit* of sleep algorithm: factor by which utility is increased, over base case (no sleep)

- Data from all sensors and times considered equally valuable

\[
\text{benefit} = \frac{1 - \text{loss}}{1 - \text{sleep}}
\]

<table>
<thead>
<tr>
<th>Loss: loss at lossiest node</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep: sleep at most awake node</td>
</tr>
</tbody>
</table>
The Effect of Uncertainty

- Consider the characteristic of a flow obtained by merging two periodic flows (with “good” period ratio)
- Deviations can accumulate
The Consequence of Uncertainty

- R cannot forward packets from A & B as they come
  - Unless D is prepared to keep PDF info about every node it forwards data for
  - MUST shape traffic
  - Must buffer data (aggregation, but only simple sense)

- R cannot ignore the fact that some packets come from A and some from B
  - Must keep PDF info about every “upstream neighbor”
  - But at least this is scalable
Three Phase Algorithm

- **Passive phase**
  - Do not sleep, determine own “base” period from sensing period (already shaping traffic)
  - Motivated by freshness considerations
  - Also respect application-specified end-to-end delay bounds if specified (may need “dummy” transmissions)

- **Learning phase**
  - Observe behavior of upstream neighbors (moving target)
  - Build histograms, estimated PDFs
  - Reduce own period if necessary

- **Operating phase**
  - Start sleeping between expected reception times
  - Parameterized by allowable loss factor \( k \)
  - Do NOT modify own period or histograms
Core Algorithm

Algorithm 1 Calculate Transmission Period

\[ \{n.d_{\text{max}}: \text{node } n\text{'s delay bound}\} \]
\[ \{n.d: \text{node } n\text{'s depth}\} \]
\[ \{n.t_{g_{\text{tx}}}: \text{node } n\text{'s data generation period}\} \]
\[ \{n.t_{t_{\text{tx}}}: \text{node } n\text{'s mean transmission period}\} \]
\[ \{U: \text{set of all upstream neighbors}\} \]
\[ d_{\text{max}} \leftarrow \min\{u.d_{\text{max}}\} \forall u \in U \]
\[ \text{if } \text{self}.d_{\text{max}} < d_{\text{max}} \text{ then} \]
\[ d_{\text{max}} \leftarrow \text{self}.d_{\text{max}} \]
\[ \text{end if} \]
\[ t_{t_{\text{tx}}} \leftarrow \min\{u.t_{t_{\text{tx}}}\} \forall u \in U \]
\[ \text{self}.t_{t_{\text{tx}}} \leftarrow \min\{d_{\text{max}}, t_{t_{\text{tx}}}\} \]

Algorithm 2 Calculate Sleep Time

\[ \{S: \text{shape, treated as c.d.f}\} \]
\[ \{\text{last}: \text{last arrival time}\} \]
\[ \{k: \text{allowable loss factor}\} \]
\[ \{\text{ETA}: \text{estimated time of arrival of next tx}\} \]
\[ \text{min} \leftarrow \infty \]
\[ \text{for each } un \text{ do} \]
\[ \text{find } t_k \mid un.S(t_k) = k \]
\[ \text{ETA} = un.\text{last} + t_k \]
\[ \text{if } \text{ETA} < \text{min} \text{ then} \]
\[ \text{min} \leftarrow \text{ETA} \]
\[ \text{end if} \]
\[ \text{end for} \]
\[ \text{if } \text{min} < \text{now} \text{ then} \]
\[ \text{sleep} \leftarrow 0 \]
\[ \text{else} \]
\[ \text{sleep} \leftarrow \text{min} \text{ – now} \]
\[ \text{end if} \]
\[ \text{return min} \]
Adapting to Changing PDFs

- Original approach - “freeze” PDF estimate histogram once formed
  - Appropriate when PDFs are known or assumed not to change

- What if the PDFs themselves change slowly?
  - Different time scale of variation
  - May be up, down, or oscillating
  - May be intermittent

- Sequence numbering may or may not be feasible

- Third phase of the algorithm must be adjusted
  - Must continue to observe upstream node behavior
Observed Behavior in Phase 3

- Sleeping (and losing packets) skews observed histogram

Before sleeping:

- Inter-arrival time vs. Probability

After sleeping:

- Inter-arrival time vs. Probability

Timeline:

- Go to sleep: $t_s$
- Tx missed: $t_m$
- Wake up: $t_w$
- Tx caught: $t_r$

Large inter-arrival time
Adaptation Algorithms

- **With sequence numbers**
  - If packet(s) are missed, at least we know exactly how many are missed
  - Problem: how to know how to divide the inter-arrival times?
  - Constraints imposed by sleep period
  - PDF can help, BUT
  - Costly operation, and equal division turns out to be very good

- **Without sequence numbers**
  - Property of histogram that does not change is *mode*
  - Watching modes alerts node to when PDF is changing
  - Must switch to phase 2
  - Alternative: use the observed mode shift to infer actual PDF change
Determining Optimum Loss Factor

- Simple two-hop analysis shows oscillatory nature
- Quantitative model for more hops around optimal region
Simulation Results

- Discrete event list simulation
- Uniformly distributed 200-node topologies, max depth 7, placed in a grid with some perturbation
- 21-bin histogram, last 400 observations
- Normal distribution with mean=500, \( \sigma=10 \), bounded and renormalized to \( \pm 3\sigma \), thus \( \beta=0.1 \)
- Simple some-shortest-path routing is used
  - We randomly choose some shortest-hop neighbor
    - In reality, could balance loads
    - Averaging over different runs and all nodes of same tier has same effect
      - Under reasonably uniform density, first-hop nodes define lifetime
- We estimate lifetime from sleep, assuming nodes stay alive over simulation period
  - In reality nodes would die one by one
  - Number of dead/disconnect nodes jumps up at the time predicted
Performance of Basic Algorithm

Sleep

Loss

Copyright Rudra Dutta, NCSU, Fall, 2004
Benefit

- Sleep from depth 1
- Loss from depth 7
Effect of Depth on Loss

\[ L_d = 1 - (1 - F(\alpha))^{d-1} \]
Effect of In-degree on Sleep

\[ S = (1 - \eta F(\alpha))(1 - \eta(1 - \alpha)) \]

Sleep in Tree Topologies

Actual

Estimated
Degree in Uniform Topologies

Degree vs. Depth

Sleep vs. Depth

Ideal average degree at depth \(d\): \[
\eta_d = \frac{(2d + 1)}{(2d - 1)}
\]
Adaptation Algorithms: Results

- Effect of slowing down
- Effect of speeding up
- Both slowing down and speeding up
- In simulations, change contained at one depth to observe effect on other depths

- No adaptation (base case)
- Adaptation with sequence numbers
- Adaptation without sequence numbers (mode watching)
Effect of Slowdown: No Adaptation

- Sleep reduces slightly at one depth below the depth where the change occurs
- Losses are increased only slightly
Adaptation With Sequence Numbers

Effect of Slowdown

- Adapts well: little decrease in sleep
- Slight increase in loss: tends to overcompensate
Adaptation Without Sequence Numbers

- Sleep reduced at depth one hop downstream from depth where change occurs
- Effect percolates downstream
- Slight reduction in loss: due to staying awake
Effect of Slowdown: Overall

- Nodes anywhere in the network slow down
- Adaptation with sequence numbers performs best
- Adaptation improves benefit
Effect of Speedup: No Adaptation

- Drastic drop in sleep and increase in loss
- Effect of speed up greater than effect of slowdown
Adaptation With Sequence Numbers

Effect of Speedup

- Some drop in sleep, increase in loss
- Effect percolates downstream
Adaptation Without Sequence Numbers

Effect of Speedup

- Significant drop in sleep and increase in loss
- Nodes anywhere in the network speed up
- Adaptation using sequence numbers works best
- In general, there is drop in sleep and increase in loss
- Adaptation makes a large difference to benefit
Nodes speed up or slow down by a factor in the range [-10%, +10%]

Effect of speeding up dominates
Conclusion

- Adaptive distributed algorithm can increase network utility in presence of quasi-static traffic of various periodicity by allowing some traffic loss
- Allowing more loss is counter-productive beyond a point
- Increasing the period of a node allows its downstream neighbour to sleep more
- In realistic, uniform topologies, degree is small and only the first two depths are critical
  - Could be leveraged to obtain further benefit
An Adaptive Scheduling Method for Quasi-Periodic Sensor Traffic

Sharat C. Visweswara, Apurva C. Goel, Rudra Dutta
North Carolina State University

This research was partly supported in part by NSF grant # EEC-0332271