Goals for lecture

• Sensor networks
• Finish overview of scheduling algorithms
• Mixing off-line and on-line
• Design a scheduling algorithm: DCP
  – Will initially focus on static scheduling
• Useful properties of some off-line schedulers

Lab two?

• Everybody able to finish?
• Any problems to warn classmates about?
• 18 motes should be arriving tomorrow
  – No equipment sign-out required for next motes lab
• Linux vs. Windows development environments

Sensor networks

• Gather information over wide region
• Frequently no infrastructure
• Battery-powered, wireless common
• Battery lifespan of central concern

Low-power sensor networks

• Power consumption central concern in design
• Processor?
  – RISC µ-controllers common
• Wireless protocol?
  – Low data-rate, simple: Proprietary, Zigbee
• OS design?
  – Static, eliminate context switches, compile-time analysis

Low-power sensor networks

• Power consumption central concern in design
• Runtime environment?
  – Avoid unnecessary dynamism
• Language?
  – Compile-time analysis of everything practical

Multi-rate tricks

• Contract deadline
  – Usually safe
• Contract period
  – Sometimes safe
• Consequences?
Scheduling methods

- Clock
- Weighted round-robin
- List scheduling
- Priority
  - EDF, LST
  - Slack
  - Multiple costs

Linear programming

- Minimize a linear equation subject to linear constraints
  - In P
- Mixed integer linear programming: One or more variables discrete
  - NP-complete
- Many good solvers exist
- Don’t rebuild the wheel

MILP scheduling

Each task has a unique start time

\[ \forall p \in P, \sum_{t=0}^{t_{\text{max}}} \text{start}(p,t) = 1 \]

Each task must satisfy its precedence constraints and timing delays

\[ \forall \{p_i, p_j\} \in E, \sum_{t=0}^{t_{\text{max}}} \text{start}(p_i) \geq \text{start}(p_j) + d_j \]

Other constraints may exist

- Resource constraints
- Communication delay constraints

Force directed scheduling

- Calculate EST and LST of each node
- Determine the force on each vertex at each time-step
- Force: Increase in probabilistic concurrency
  - Self force
  - Predecessor force
  - Successor force

Self force

\[ F_i \text{ all slots in time frame for } i \]
\[ F'_i \text{ all slots in new time frame for } i \]
\[ D_t \text{ probability density (sum) for slot } t \]
\[ \delta D_t \text{ change in density (sum) for slot } t \text{ resulting from scheduling self force} \]

\[ A = \sum_{t \in F'_i} D_t \cdot \delta D_t \]
Predecessor and successor forces

**pred** all predecessors of node under consideration
**succ** all successors of node under consideration

Predecessor force

\[ B = \sum_{b \in \text{pred}(x)} D_b \cdot \delta D_t \]

Successor force

\[ C = \sum_{c \in \text{succ}(x)} D_c \cdot \delta D_t \]

Intuition

Total force: \[ A + B + C \]

- Schedule operation and time slot with minimal total force
- Then recompute forces and schedule the next operation
- Attempt to balance concurrency during scheduling

Force directed scheduling

- Limitations?
- What classes of problems may this be used on?

Implementation: Frame-based scheduling

- Break schedule into (usually fixed) frames
- Large enough to hold a long job
  - Avoid preemption
- Evenly divide hyperperiod
- Scheduler makes changes at frame start
- Network flow formulation for frame-based scheduling
- Could this be used for on-line scheduling?

Examples: Mixing on-line and off-line

- Book mixes off-line and on-line with little warning
- Be careful, actually different problem domains
- However, can be used together
- Superloop (cyclic executive) with non-critical tasks
- Slack stealing
- Processor-based partitioning

Problem: Vehicle routing

- Low-price, slow, ARM-based system
- Long-term shortest path computation
- Greedy path calculation algorithm available, non-preemptable
- Don’t make the user wait
  - Short-term next turn calculation
- 200 ms timer available

Problem space genetic algorithm

- Let’s finish off-line scheduling algorithm examples on a bizarre example
- Use conventional scheduling algorithm
- Transform problem instance
- Solve
- Validate
- Evolve transformations
Examples: Mixing on-line and off-line

- Slack stealing
- Processor-based partitioning

Scheduling summary

- Scheduling is a huge area
- This lecture only introduced the problem and potential solutions
- Some scheduling problems are easy
- Most useful scheduling problems are hard
  - Committing to decisions makes problems hard: Lookahead required
  - Interdependence between tasks and processors makes problems hard
  - On-line scheduling next Tuesday

Bizarre scheduling idea

- Scheduling and validity checking algorithms considered so far operate in time domain
- This is a somewhat strange idea
- Think about it and tell/email me if you have any thoughts on it
- Could one very quickly generate a high-quality real-time off-line multi-rate periodic schedule by operating in the frequency domain?
- If not, why not?
- What if the deadlines were soft?

Reading assignment

- Read Chapter 7