Energy Aware Real-Time Scheduling Policy with Guaranteed Security Protection

Wei Jiang¹, Ke Jiang², Xia Zhang¹, Yue Ma³

¹School of Information and Software Engineering
University of Electronic Science and Technology of China

²Department of Computer and Information Science
Linköping University, Sweden

³Department of Computer Science and Engineering
University of Notre Dame, USA
Introduction

- Embedded system design concerns

- Safety/
- Reliability
- Limited resources
- Real-time
- High QoS
- Energy
- Security
The Design Problem
- Security- & Energy-aware Real-Time Application
- System execution goal
  - Complete the App. with minimal energy
  - Satisfy the security and real-time requirements
- NP-hard to find the best solution

The Method
- Dynamic Programming based Approximate optimization framework
Outline

- Motivational application
- System model
- Problem formulation
- Approximation based Dynamic Programming
- Experimental results
- Conclusion
Motivational application
System Model

- Architecture model
  - Mono-processor, Battery powered

- Application model
  - A set of periodic security- & energy-aware tasks
  - Security risk constraint
  - Scheduling by classic method, e.g. RM/EDF

- Task model
  - A mandatory and optional part (Security improve)
  - Task attributes: \((BE_i, P_i, L_i, S_i, S_i^{DM}, V_i, SR_i)\)
Security Overhead Model

- Measure energy & time of security algorithms

- Samsung S3C2440 @400MHz
  - 64MB SDRAM
  - uC/OSII with Cryptlib v3.4

- SCB-68 Connector

- NI PXI-1042Q

- LabVIEW program to acquire data

- A real embedded platform
- NI instrument
- LabVIEW based data acquisition
- Nearly non-intrusive measurement
## Security Overhead Model

### Measurement results

<table>
<thead>
<tr>
<th>Ciphers</th>
<th>time (ms/KB)</th>
<th>Energy (mJ/KB)</th>
<th>Sec. Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC4</td>
<td>0.0063</td>
<td>2.0237</td>
<td>1</td>
</tr>
<tr>
<td>RC5</td>
<td>0.0125</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLOWFISH</td>
<td>0.0170</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IDEA</td>
<td>0.0196</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SKIPJACK</td>
<td>0.0217</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3DES</td>
<td>0.0654</td>
<td>21.0914</td>
<td>6</td>
</tr>
</tbody>
</table>

Energy/time ratio: 320 mJ/S

POW (power)

### Execution time of each task

\[ Exe_i = BE_i + \theta(S_i) \times L_i \]

### Energy consumption of each task

\[ En_i = POW \times (BE_i + \theta(S_i) \times L_i) \]
**Security Risk Model of each task**

- **Definition**
  - **Security risk (SR)** is the product of security failure probability and consequence impact of security failure.
  
  $$SR_i = Pr_o^{{risk}} * V_i$$

- **Failure probability**
  - $$Pr_o^{{risk}} = \begin{cases} 
  0, & \text{if } S_i \geq S_i^{DM} \\
  1 - e^{-\lambda_i(S_i^{DM} - S_i)} & \text{otherwise}
  \end{cases}$$

- More reasonable than other linear security QoS definitions like ref. [8, 10]
Problem formulation

- **Original problem**

  Min  \( \text{Energy} = \sum_{i=1}^{N} \left( \frac{HP}{p_i} \right) * E_n_i \)

  Subject to

  \( \sum_{i=1}^{N} \left( \frac{HP}{p_i} \right) * SR_i \leq RB \)

  \( \sum_{i=1}^{N} \left( BE_i + \theta(S_i) * L_i \right) / P_i \leq UB_x \)

  \( S_{min} \leq S_i \leq S_{max} \)

- **Energy**

  \( \text{Energy} = \sum_{i=1}^{N} \left( \frac{HP}{p_i} \right) * E_n_i \)

  \( = HP * POW * \sum_{i=1}^{N} \left( BE_i + \theta(S_i) * L_i \right) / P_i \)
Problem formulation

- Reduced problem

\[
\begin{align*}
\text{Min} \quad & \sum_{i=1}^{N} (BE_i + \theta(S_i) \times L_i)/P_i \\
\text{S.T.} \quad & \sum_{i=1}^{N} \left( \frac{HP}{P_i} \right) \times SR_i \leq RB \\
& \sum_{i=1}^{N} (BE_i + \theta(S_i) \times L_i)/P_i \leq UB_x \\
& S_{\text{min}} \leq S_i \leq S_{\text{max}}
\end{align*}
\]

- Min. Utilization
- Risk constraint
- Utilization constraint
- Security level constraint

*We don’t need to consider Energy dimension!*
Proposed Optimization Technique

- Markov dynamic programming procedure
- Approximating policies and analysis
- Round Nearest approximating algorithm
- Low Time complexity
Markov decision-making procedure

- Multi-stage decision procedure
  - N-Stage (One task, one stage)
  - Decision variable: $S_i$ (sec level)
  - State definition: $(\xi_{ik}, \gamma_{ik}, S_{ik})$

Accumulated utilization ratio of first $i$ tasks
Accumulated risk of first $i$ tasks
Specific level for $k$-th state

Min utilization state
Best Solution: $(1, 3, 2, \ldots, 3)$
Multi-stage decision-making procedure

- Number of states increases exponentially! How next?
- Approximation of Knapsack problem
  - Scale risk into a series of discrete integers by $\Delta$
  - Replace states with same risk by lowest utilization one
  - States denoted by a $N \times M$ matrix, $M = \lceil RB / \Delta \rceil$

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>...</th>
<th>$M$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_1$</td>
<td>$(\xi_{1,1}, \gamma_{1,1}, S_{1,1})$</td>
<td>$(\xi_{1,2}, \gamma_{1,2}, S_{1,2})$</td>
<td>...</td>
<td>$(\xi_{1,M}, \gamma_{1,M}, S_{1,M})$</td>
</tr>
<tr>
<td>$T_2$</td>
<td>$(\xi_{2,1}, \gamma_{2,1}, S_{2,1})$</td>
<td>$(\xi_{2,2}, \gamma_{2,2}, S_{2,2})$</td>
<td>...</td>
<td>$(\xi_{2,M}, \gamma_{2,M}, S_{2,M})$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$T_{N-1}$</td>
<td>$(\xi_{N-1,1}, \gamma_{N-1,1}, S_{N-1,1})$</td>
<td>$(\xi_{N-1,2}, \gamma_{N-1,2}, S_{N-1,2})$</td>
<td>...</td>
<td>$(\xi_{N-1,M}, \gamma_{N-1,M}, S_{N-1,M})$</td>
</tr>
<tr>
<td>$T_N$</td>
<td>$(\xi_{N,1}, \gamma_{N,1}, S_{N,1})$</td>
<td>$(\xi_{N,2}, \gamma_{N,2}, S_{N,2})$</td>
<td>...</td>
<td>$(\xi_{N,M}, \gamma_{N,M}, S_{N,M})$</td>
</tr>
</tbody>
</table>
Approximating policies and analysis

- **Round to Ceiling (RC)**
  \[ RC(SR_i) = \left\lceil \frac{SR_i}{\Delta} \right\rceil \]
  \[ RC(2.2) = 3 \]
  \[ Err = 0.8 \]

- **Round to Floor (RF)**
  \[ RF(SR_i) = \left\lfloor \frac{SR_i}{\Delta} \right\rfloor \]
  \[ RF(2.8) = 2 \]
  \[ Err = 0.8 \]

- **Round Randomly (RR)**
  \[ RR(SR_i) = \begin{cases} 
  \left\lceil \frac{SR_i}{\Delta} \right\rceil \text{ with probability } \rho_1 = \frac{SR_i}{\Delta} - \left\lfloor \frac{SR_i}{\Delta} \right\rfloor \\
  \left\lfloor \frac{SR_i}{\Delta} \right\rfloor \text{ with probability } \rho_2 = \left\lceil \frac{SR_i}{\Delta} \right\rceil - \frac{SR_i}{\Delta}
  \end{cases} \]

- **Round to Nearest (RN)**
  \[ RN(SR_i) = SR_i^\Delta = \begin{cases} 
  \left\lceil \frac{SR_i}{\Delta} \right\rceil, \text{ if } \frac{SR_i}{\Delta} - \left\lfloor \frac{SR_i}{\Delta} \right\rfloor \geq 0.5 \\
  \left\lfloor \frac{SR_i}{\Delta} \right\rfloor, \text{ if } \frac{SR_i}{\Delta} - \left\lfloor \frac{SR_i}{\Delta} \right\rfloor < 0.5
  \end{cases} \]
  \[ RN(2.2) = 2 \]
  \[ RN(2.8) = 3 \]
Approximating policies and analysis

- How to determine $\Delta$, given $(1 + \beta) \cdot RB$ approximation?
- Overall Deviation for $N$ tasks is:
  - $OD^{RC} \geq -N\Delta$
  - $OD^{RF} \leq N\Delta$
  - $-N\Delta \leq OD^{RR} \leq N\Delta$
  - $-N\Delta/2 \leq OD^{RN} \leq N\Delta/2$
- For $(1 + \beta)$ approximation, $|OD| \leq \beta \cdot RB$

- Max $\Delta$ of $RN$ is twice larger than $RF$, $RC$ and $RR$!
- Reduce the number of states by a half in decision-making procedure!
Round Nearest approximating algorithm

**Algorithm 1** RN-based approximation algorithm

1: Step 1: Schedulability test  
2: if \( \sum_{i=1}^{N} (HP/P_i)SR_i(S_{\text{max}}^\text{max}) > RB \) or \( \sum_{i=1}^{N} \text{Exec}_i(S_{\text{min}}^\text{min})/P_i > UB_{\text{x}} \) then  
3: Return. /*Given task set is not schedulable*/

4:  
5: Step 2: Initialization  
6: Compute the grouping factor \( \Delta = 2/\beta RB/N \) and \( M = \lfloor RB/\Delta \rfloor \)  
7: Initialize state matrix \( \Omega_{N \times M} \) with each element \( \Omega_{i,j} = (0, 0, 0) \)  
8: Initialize \( \Omega_1 \) by calculate \( (\xi_1, \gamma_1, S_1) \) with each \( S_1 \in [S_{\text{min}}, S_{\text{max}}] \)

9:  
10: Step 3: Update the state matrix in N-Stage decision procedure  
11: for \( i = 2 \) to \( N \) do  
12: while \( (\xi_{i-1}, \gamma_{i-1}, S_{i-1}) \neq (0, 0, 0) \) in \( \Omega_{i-1} \) do  
13: for \( S_i' = S_{\text{min}} \) to \( S_{\text{max}} \) do  
14: Calculate temporary state \( (\xi_i', \gamma_i', S_i') \)  
15: if \( \xi_i' > UB_{\text{x}} \) or \( \gamma_i' > RB \) then  
16: Ignore this state and break /*Schedulability or security violated*/  
17: if state \( \Omega_{i,j'}, (j = \gamma_i') \) is not existed then  
18: \( \Omega_{i,j} = (\xi_i', \gamma_i', S_i') */\text{ Store new state}*/  
19: else if \( \xi_i < \xi_i' \) in \( \Omega_{i,j} \) then  
20: \( \Omega_{i,j} = (\xi_i', \gamma_i', S_i') */\text{ Keep state with smaller utilization}*/  

21:  
22: Step 4: Find the minimal energy consumption solution  
23: Find \( \Omega_{N,j}^* \) with minimal utilization ratio \( \xi_N^* \)  
24: Obtain the final security assignment decision \( (S_1, S_2, \cdots, S_N) \) by backtracking  
25: \( \text{Energy}^* = \xi_N^* \cdot HP \cdot POW /*\text{The minimal energy}*/

17
Experimental results

- **Experiment setup**
  - Two group simulations, each with three synthetic sets
  - Basic execution time of each task: 5~10 ms
  - Period: 300~500 ms
  - Confidential data size: 100~400 KB
  - Security demand: 6~8
  - Security impact/loss of each task: 5~10 $
  - Security coefficient $\lambda$: 1~3

- **Compared algorithms**
  - RRAA: Round Randomly approximating algorithm
  - RCAA [16]: Round to Ceiling approximating algorithm
  - GRDY: Assigned security level in greedy fashion
  - SEAS [8]: Gradually increase the security by small risk/energy ratio
Impacts of Risk Bound (RB)

\[ RB = MIR + \alpha \ast (MAR - MIR), \quad \beta = 0.05 \]

- **Minimal Risk**
  - RNAA saves 14.5%, 5.9%, 4.3% of GRDY, RCAA, SERS

- **Maximal Risk**
  - RNAA satisfies the risk slack ratio and its complexity is half of RCAA, RRAA.
Impacts of Risk Slack Ratio ($\beta$)

Given $RB = MIR + 0.7 \times (MAR - MIR)$

RNAA saves 19.3%, 16.3%, 10% of GRDY, RCAA, SERS

RNAA is the best among them, lowest energy with guaranteed little risk deviation!
Conclusions

- A new scheduling optimization problem for security- and energy-critical real-time applications
  - Minimal energy with real-time and risk constraints
  - Multi-dimensional knapsack problem (NP-hard)

- Efficient techniques
  - Problem reduced (energy dimension)
  - Approximating dynamic programming
  - Half complexity of traditional approx. DP algorithms

- Experiments show the good performance
Thanks for your time!

^_^