Integrating Power Management into Distributed Real-time Systems at Very Low Implementation Cost

Bita Gorjiara[1], Nader Bagherzadeh[1], Pai Chou[1,2]

[1]Center for Embedded Systems,

University of California, Irvine, USA

[2] Dept. of Computer Science

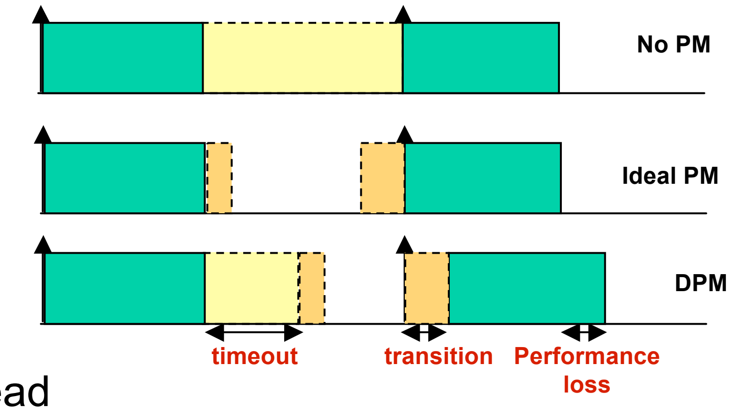
National Tsing Hua University, Taiwan

Introduction

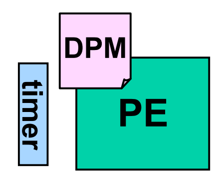
- **Dynamic Power Management (DPM) : Turning off unused components**
- **Common DPM approaches**
  - Timeout
    - Switches off after idling for a while
    - Switches ON at the arrival of an event
    - Cons:

      Energy waste due to idling,

      Performance loss due to transition overhead

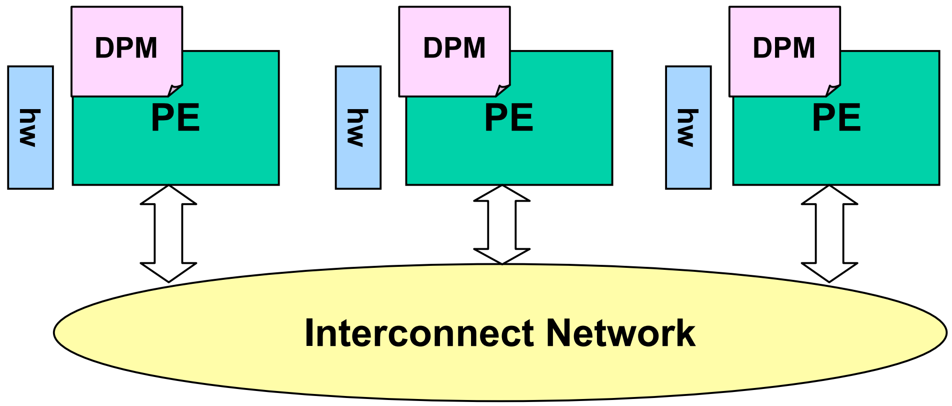


  - Predictive, stochastic [Benini00,Chung99, Hwang97, Irani03]
    - Predicts the length of idle times based on history
- **Implementation**
  - Requires additional software and hardware



DPM for systems

- **DPM is costly to implement at system level**

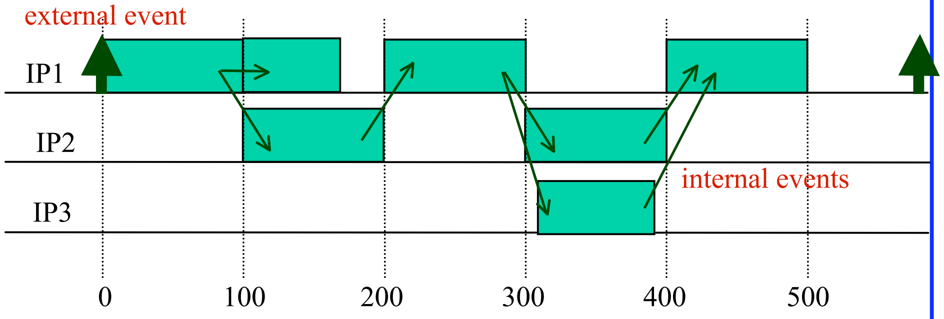


- **Not all components can handle their own DPM**

  Example: analog elements

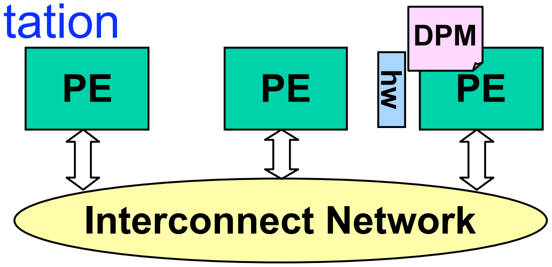
DPM for a real-time system

- **A real-time system**
  - External event come periodically
  - External events trigger internal events



  - Application information can be used to avoid component-level DPM

- **Our DPM: Application-based Power Management (APM)**
  - Exploits application and system info to predict idle durations
  - Is a centralized approach => Low-cost implementation



- **Contributions of this paper:**
  - Systematically modeling real-time systems for centralized DPM
  - Developing the power manager kernel

Outline

- **Software architecture of APM**
- **Modeling a real-time system and its services**
- **Our DPM algorithm**
- **Experimental Results**
  - APM implementation for a Software-Defined Radio (SDR) system

Services and Requests

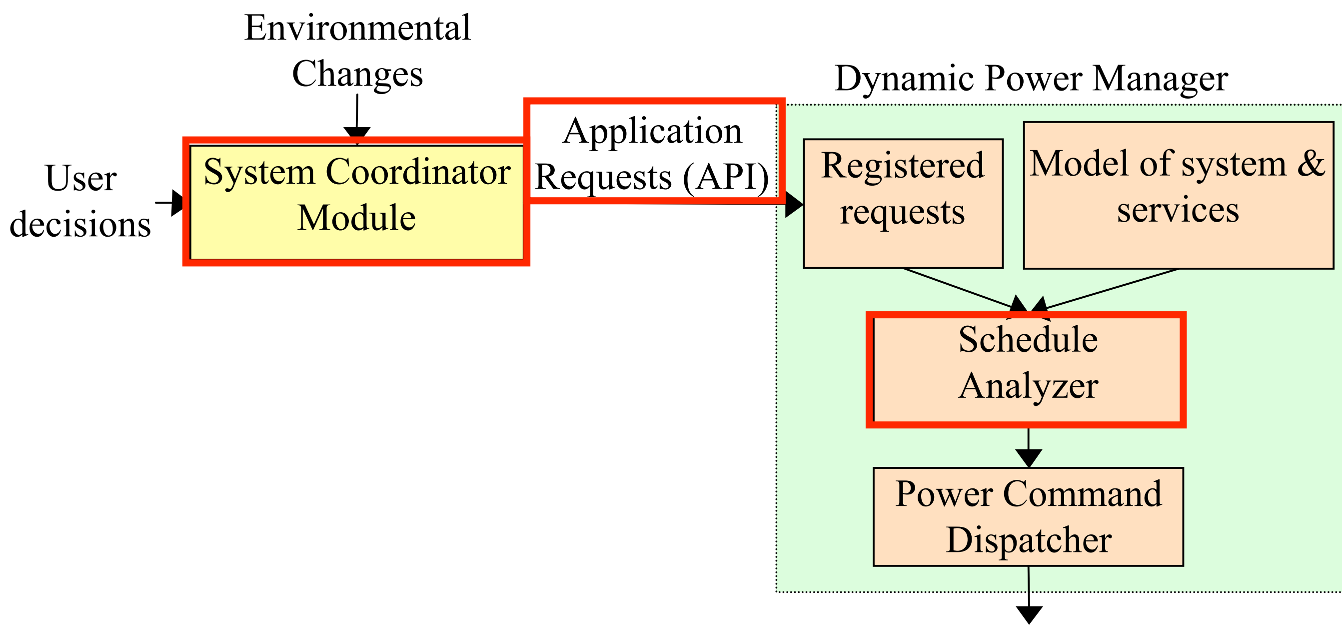
- **Service**
  - Defines a high-level behavior of the system
  - Modeled by a set of tasks, their timing and their dependencies
  - example:
    - System: An MPEG decoder system
    - Services: corresponding to each supported resolution, a service is defined
  - A system runs a finite set of services (known at design time)

Static info

- **Request**
  - Defines properties of external events
    - Period: e.g. frame per second
    - Deadline: may be the same as period
    - Service type
  - The external events are determined at runtime based on user decisions or environmental changes
  - Multiple requests may be processed simultaneously

Dynamic info

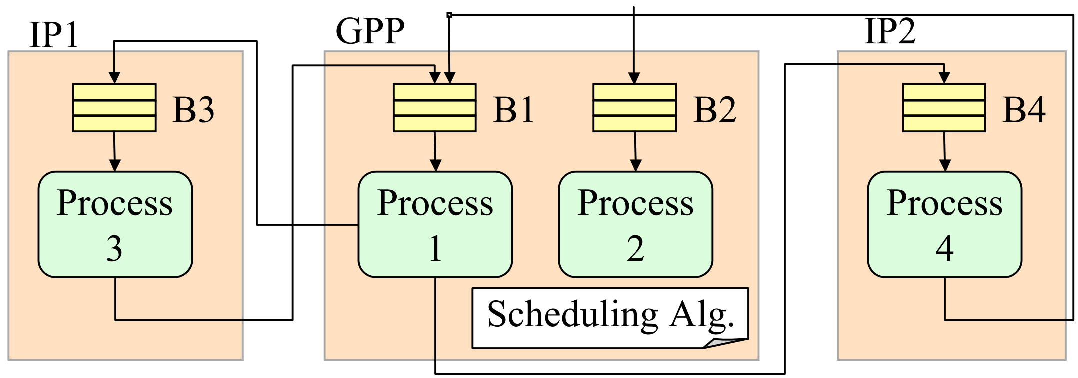
Software architecture of APM



- **System Coordinator**
  - Translates high-level application decisions to requests
  - Used API: Register/Terminate a request
- **Schedule Analyzer:**
  - Simulates system schedule for registered requests
  - Extracts idle durations and power commands

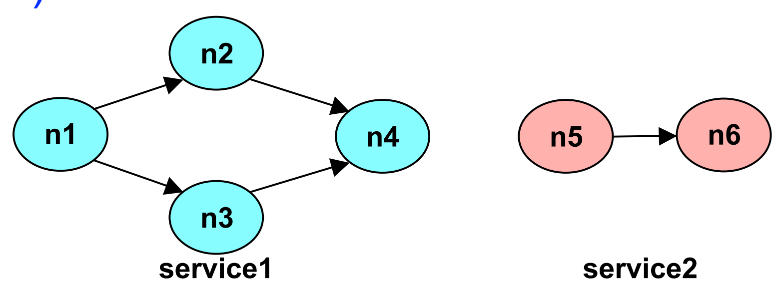
Modeling system and services

- **Properties of our model**
  - Is based on Communicating Sequential Processes (CSP)
  - Extension: functionality is abstracted by black-box tasks



- **Modeling a system**
  - Resources
  - Processes
  - Event buffers (abstracts memories, queues, …)

- **Modeling services**
  - Using task graphs

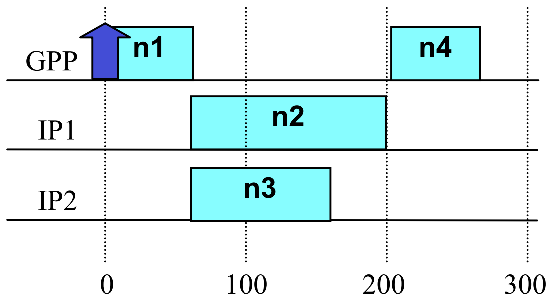


Task	Execution Delay	Resource	Process
n1	…	GPP	Process1
n2	…	IP1	Process3
n3	…	IP2	Process4
n4	…	GPP	Process1

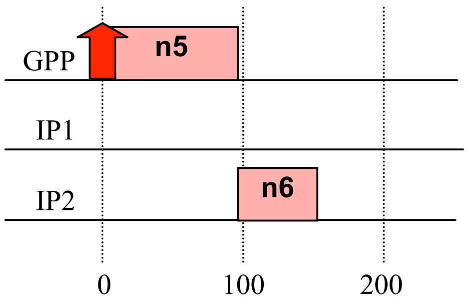
APM algorithm

- **Uses timing information of services**

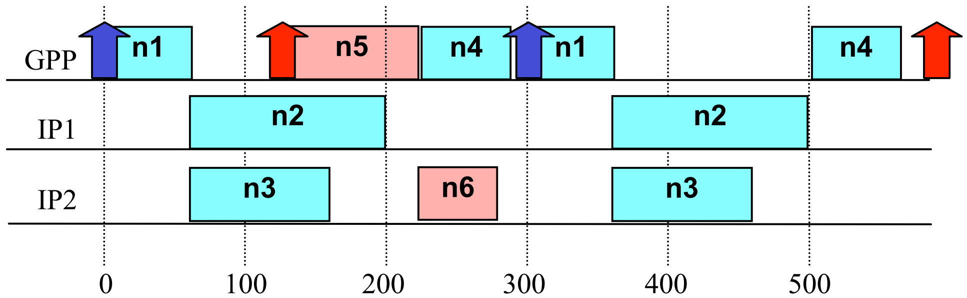
Timing diagram of Service1



Timing diagram of Service2

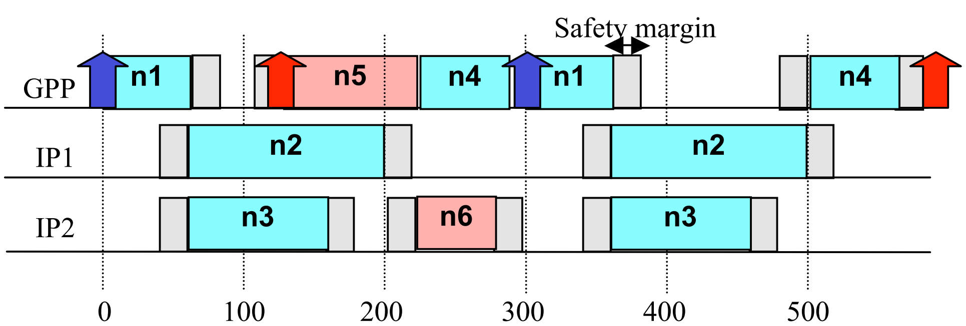


- **Uses a discrete event simulator**
- **Computes system schedule for the registered requests**



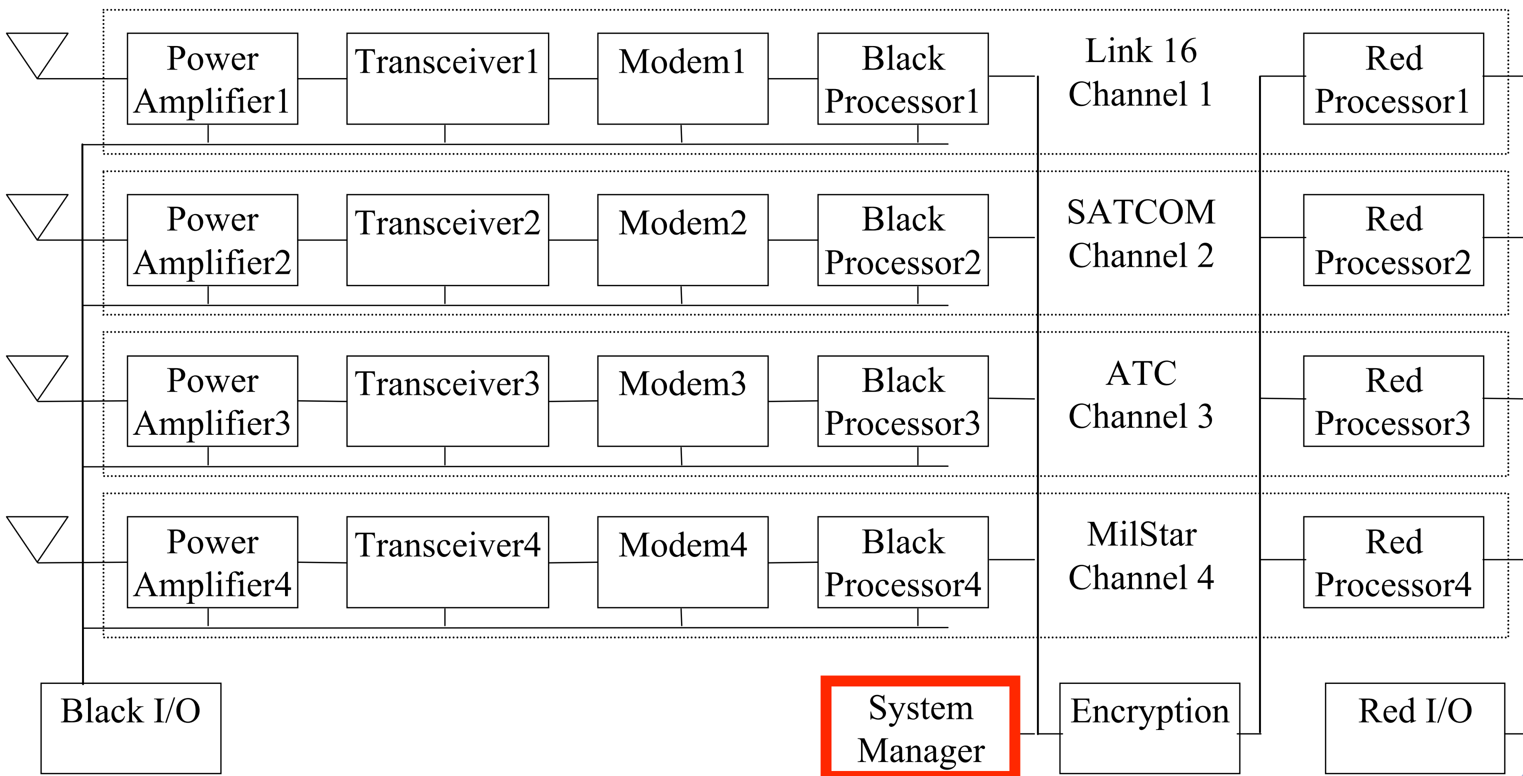
Schedule deviation

- **The real schedule may deviate from the computed schedule due to**
  - Variation in execution delay of tasks
  - Jitter in arrival of external events
- **Solution: a safety margin is added to the computed schedule**
  - Margin must be tuned for a given system
  - Has some energy penalty



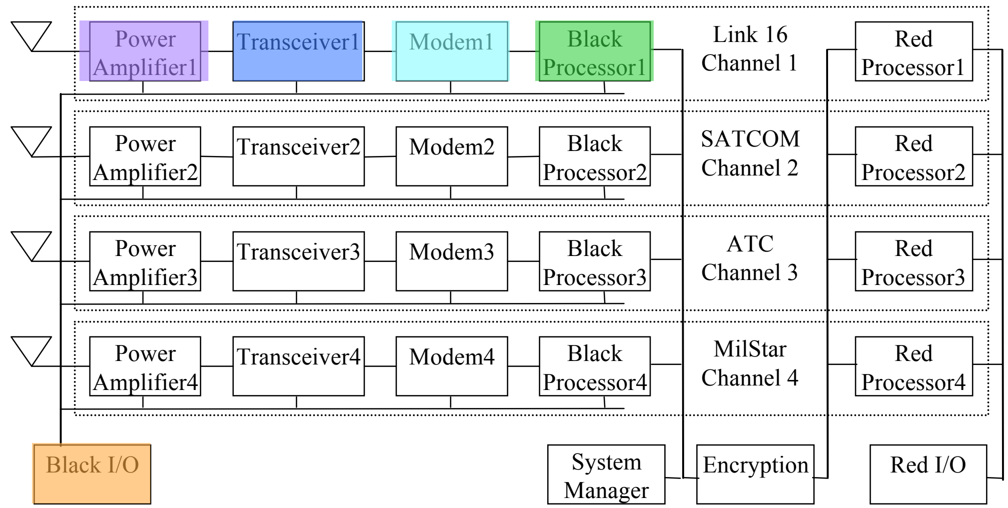
Case Study

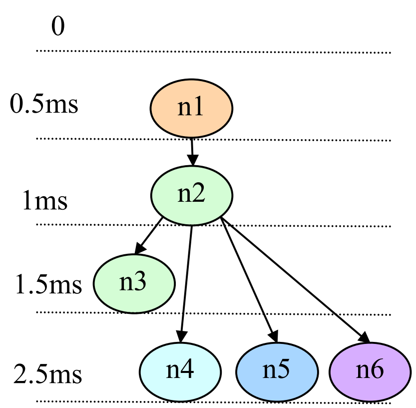
- **A software-defined radio system**
  - Used to control and monitor a UAV airplane
  - Has four channels, 23 components (analog and digital)
  - Our power manager runs on System Manager processor



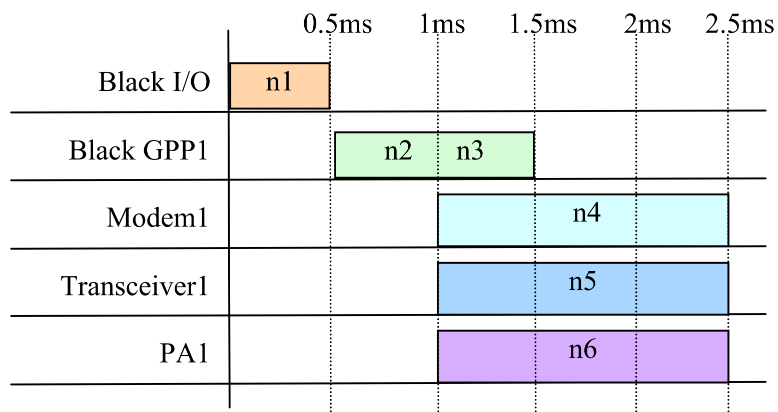
SDR services: unencrypted send

- **The receive is the reverse**





Task graph

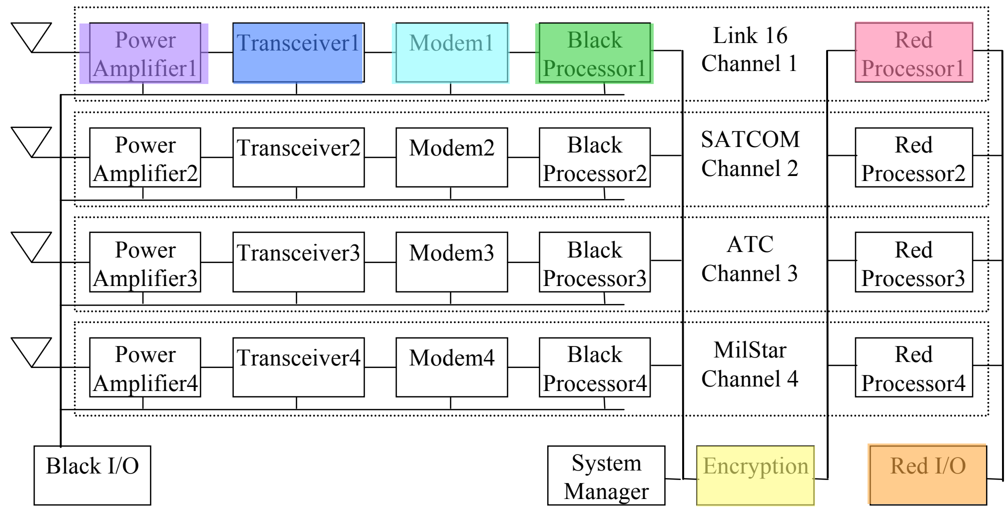


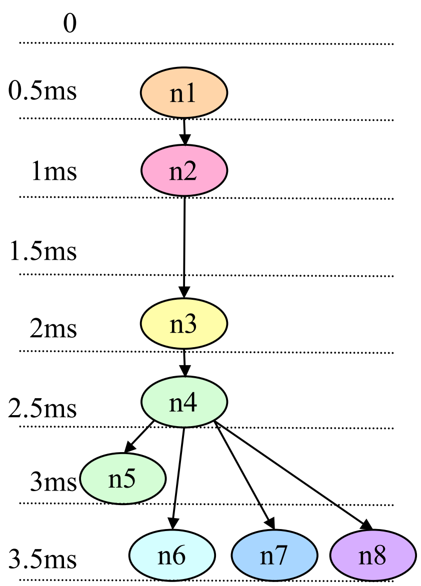
Timing diagram

Service: Unencrypted Send, Channel 1

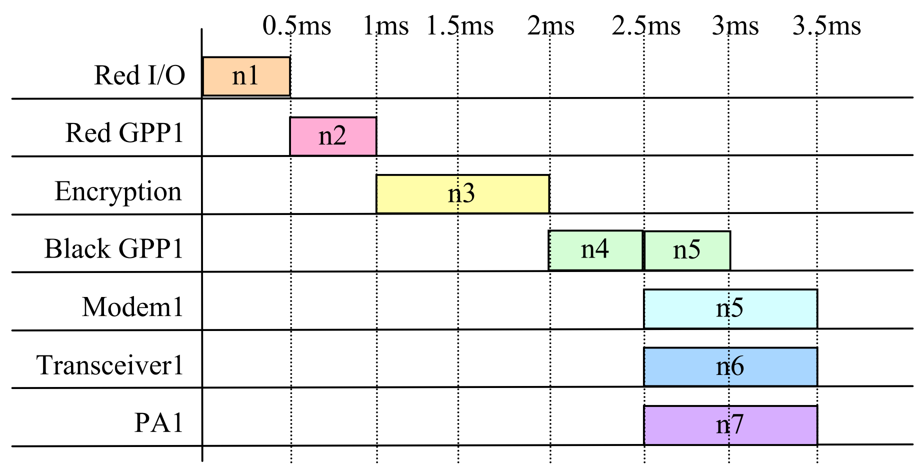
SDR services: encrypted send

- **For each channel**
  - Two send
  - Two receive
- **Total services: 16**





Task graph



Timing diagram

Service: encrypted Send

DPM and event loss

- **The real schedule may deviate from the computed schedule due to**
  - Variation in execution delay of tasks
  - Jitter in arrival of external events
- **In SDR, this causes event loss if shutdown while processing a message**
  - All the wireless devices can tolerate some loss
  - In our application up to 1% message loss is acceptable
- **Safety margins are added to the computed schedule to reduce the loss**

Experiment setup

- **Simulation environment**
  - Developed to study different aspects of the system
    - Different jitter and safety margin values are used
    - Event loss is captured
  - Is modeled in SystemC
  - Uses state-based power estimation [Bergamaschi03]
  - Three variations:
    - Without DPM
    - With ideal DPM
    - With APM
- **Testbench**
  - Actual communication profile of SDR during a 10-hour mission
  - 300,000 messages
  - Rate and type of messages varies at runtime
- **Hardware implementation**
  - Our DPM is added to the SDR system
  - Power is measured and compared to simulation model

Results

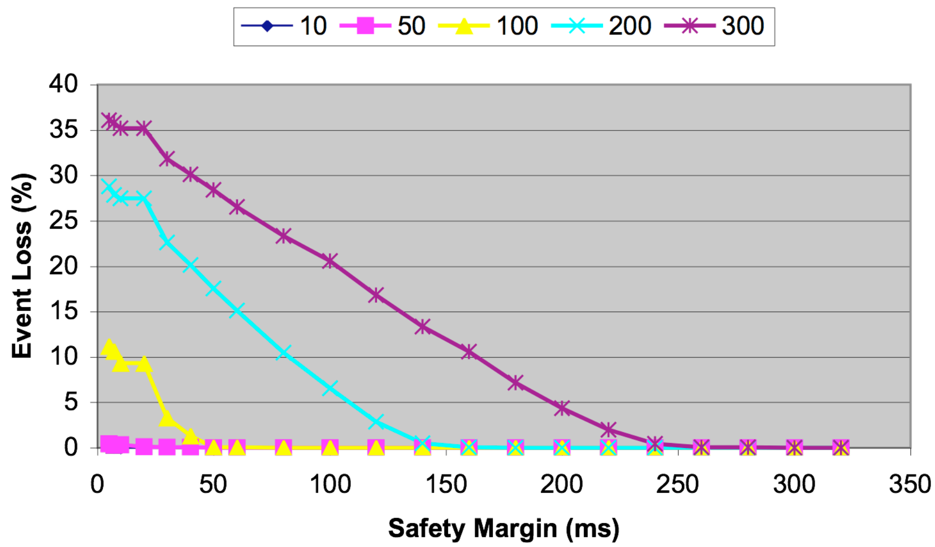
- **Energy consumption**
  - No DPM: 7.29MJ
  - Ideal DPM: 0.95MJ => 88% savings
  - APM: varies for different safety margins

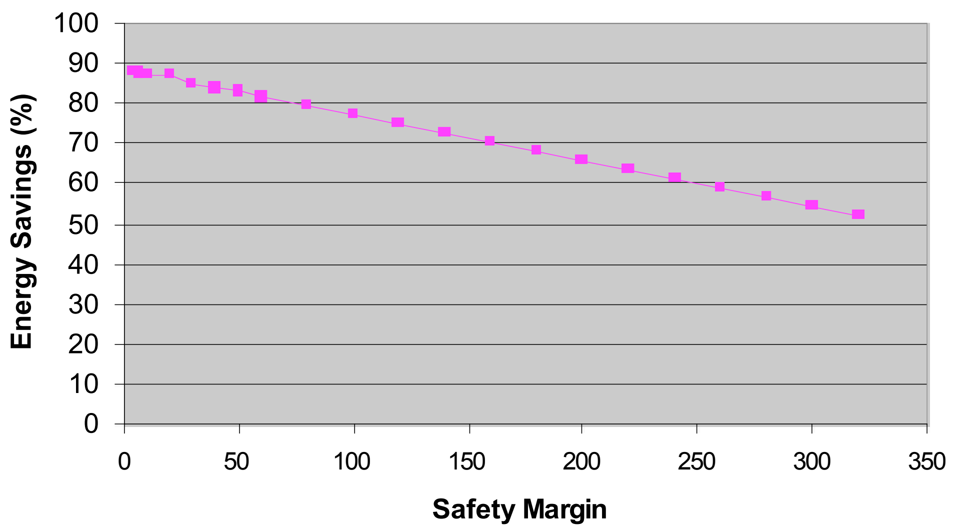
Event loss (%)

Safety Margin (ms)	Jitter (ms)					Energy saving (%)
	10	50	100	200	300	
5	0.32	0.44	11.21	28.79	36.06	87.8
7	0.08	0.26	10.66	27.87	35.79	87.6
10	0	0.32	9.3	27.48	35.19	87.2
20	0	0.12	9.3	27.48	35.19	87.2
30	0	0.04	3.3	22.62	31.81	84.9
40	0	0.08	1.35	20.16	30.14	83.8
50	0	0	0.04	17.57	28.43	82.7
60	0	0	0.08	15.11	26.56	81.5
80	0	0	0	10.5	23.34	79.3
100	0	0	0	6.56	20.56	77.0
120	0	0	0	2.86	16.82	74.8
140	0	0	0	0.52	13.36	72.5
160	0	0	0	0.04	10.62	70.2
180	0	0	0	0	7.16	68.0
200	0	0	0	0	4.37	65.7
220	0	0	0	0	1.99	63.5
240	0	0	0	0	0.44	61.2
260	0	0	0	0	0.04	59.0
280	0	0	0	0	0.04	56.7
300	0	0	0	0	0	54.4
320	0	0	0	0	0	52.2

Results

- **Energy consumption**
  - No DPM: 7.29MJ
  - Ideal DPM: 0.95MJ => 88% savings
  - APM: varies for different safety margins





Results summary

- **The minimum safety margin corresponding to the jitter values**

Jitter (ms)	Min. safety margin (ms) (less than 1% event loss)	Energy saving (%)
10	5	87.8
50	5	87.8
100	40	82.7
200	140	72.5
300	240	61.2

- **Hardware measurements**
  - Safety margin = 140ms
  - Savings = 68%
  - Simulation error: 5%

Runtime overhead of APM

- **APM runs on System Manager PE (PowerPC 500MHz, 256MB RAM, 16W)**
- **Total APM processing time**
  - 9 mins for a 10-hour mission
  - On average, for every 80 seconds of the mission, one second of DPM computation
- **Energy consumption of APM**
  - 8.6KJ
  - Very low energy consumption

  (1% of the energy consumption of the system with APM)

Conclusion

- **Application-based power management (APM)**
  - Is a low-cost centralized DPM
  - Targets real-time systems
  - Anticipates idle durations using high-level system modeling and simulation
  - Reacts to application changes quickly
  - Accounts for event jitter and task delay variation
  - Achieved 60-87% energy savings for SDR