Integrated Scheduling of Multiple System Resources

Prashant Pradhan

State University of New York, Stony Brook

Motivation

- **Servers in an internet services infrastructure**
  - **Each server shared by many principals**
  - **Each principal requires a performance guarantee (and pays for it)**
- **Characteristics of a principal's task :**
  - **Requires multiple resources (CPU, link, disk, memory)**
  - **Has a single, *global* performance requirement**

Motivation

- **Example :**
  - **Periodic task :**

    Disk->CPU->Link, Disk->CPU->Link ….
  - **Work required from each resource :**

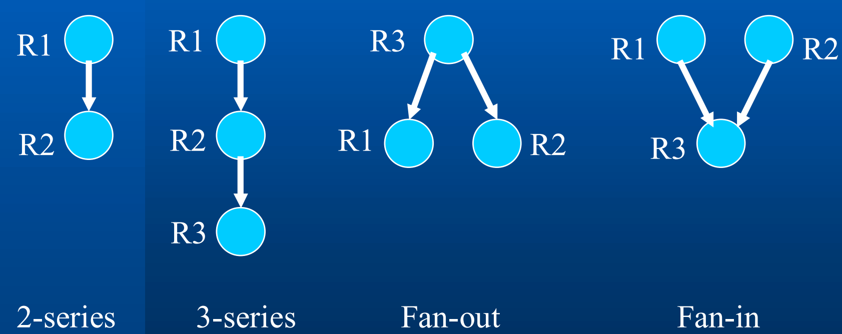
    100 bytes to be read from disk, 1000 cycles to process data, 100 bits to be sent on the network
  - **Performance requirement :**

    1000 requests/sec

Motivation

- **Only the sytem has complete knowledge of the performance requirements of all principals**
  - ***Hence the system must translate global performance requirements to per-resource allocations optimally (maximize the number of principals admitted)***

Resource Usage Topologies

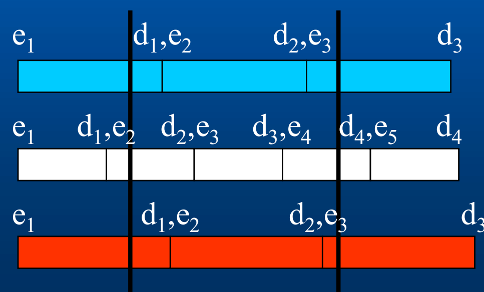


Single Resource Scheduling

- **EDF : optimal dynamic priority algorithm**
  - **Schedulability condition by Figueira and Pasquale**

Single Resource Scheduling

- EDF : optimal dynamic priority algorithm
  - Schedulability condition by Figueira and Pasquale



Single Resource Scheduling

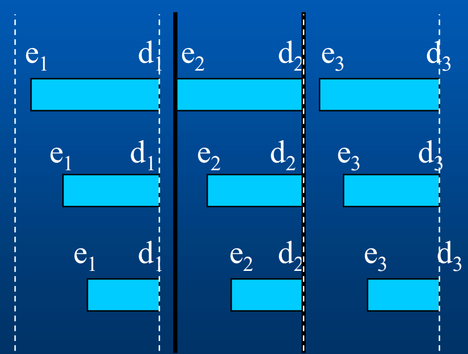
- RMS : optimal static priority algorithm
  - Analysed by Liu and Layland

Key Ideas

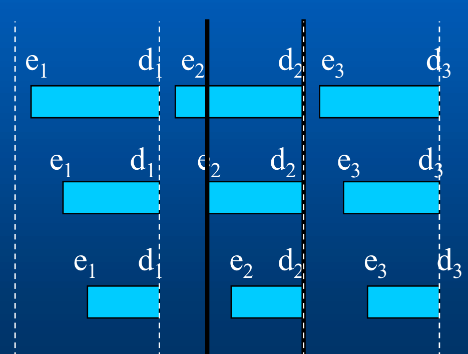
- Using an optimal scheduling algorithm in R1 is sufficient but not necessary to schedule the task set
  - use this freedom to optimize admission control in R2
- The key to performing tight admission control in R2 is to accurately characterize the time a task spends waiting in R1
  - determines eligibility in R2

Key Ideas

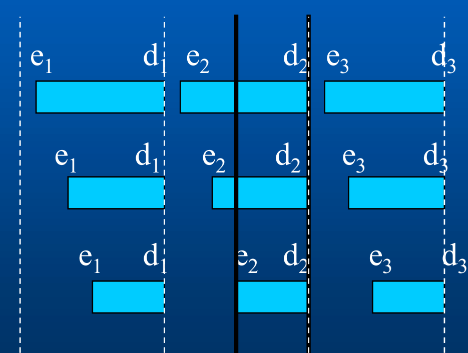
Key Ideas



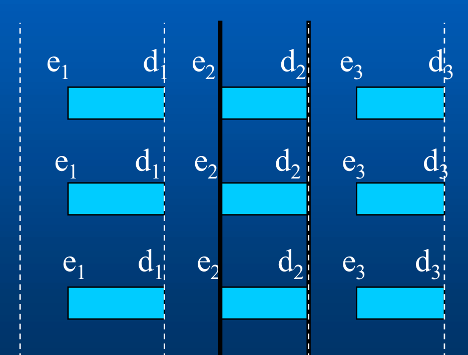
Key Ideas



Key Ideas



Key Ideas



Our Approach

- Static priority in R1, EDF in R2
  - accurately characterizes the waiting time in R1
  - dynamic priority must take into account the pending work in the two resources *at run time*

Scheduling R1

- Admission control in R2 requires :

  W1/d1 + W2/d2 + …. <= C2

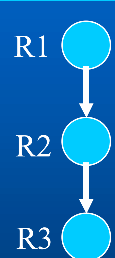
- The denominator of a task's term must be reduced by its waiting time in R1
  - minimize left hand side
- Pick tasks one-by-one : the next one picked is the one that causes that largest incremental increase in the left hand side

Scheduling R1

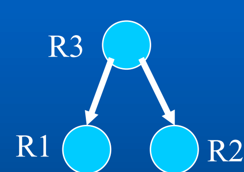
- Now we have two priority orderings for R1 :
  - O1 : RMS
  - O2 : computed to optimize R2

- Calculate a final ordering for R1
  - pick a task from O2 if it does not violate any deadlines in R1
  - if it does, pick a task from O1
  - using the newly picked task, check for admission control in R2
  - till all tasks are admitted

Extending to Other Topologies



Consider R2 and R3 as one resource. Their final ordering becomes O2 for the 3-series



While picking the ordering O2, normalize the sensitivity of a task by the capacity of its resource

Deadline Splitting

- A task's deadline d is split into two pieces d1 and d2
- EDF is performed in each resource
- The optimization criterion is to balance the utilization of all resources
  - allocate more relaxed deadline to a heavily utilized resource
- Problem : lack of tight admission control

Ongoing Work

- Optimality
- Backtracking
- Including memory as a resource

References

- Figueira N.R. and Pasquale J.; "A Schedulability Condition for Deadline-Ordered Service Disciplines", IEEE/ACM Transactions on Networking, Vol.5, No.2, April 1997.
- Liu C.L. and Layland J.W.; "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment", Journal of the ACM, Vol.20, No.1, January 1973, pp. 46-61.
- Chiueh T. and Gopalan K.; "Integrated Real-Time Resource Scheduling", ECSL Technical Report No. 56 (available from www.ecsl.cs.sunysb.edu/tr/TR56.ps.gz).