Dynamic Scheduling in TAO

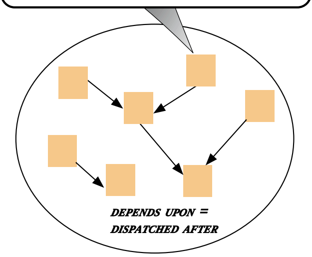
Chris Gill

Distributed Object Computing Group
Computer Science Department, Washington University, St. Louis

cdgill@cs.wustl.edu
http://www.cs.wustl.edu/∼cdgill/dynsched_DOVE.ps.gz

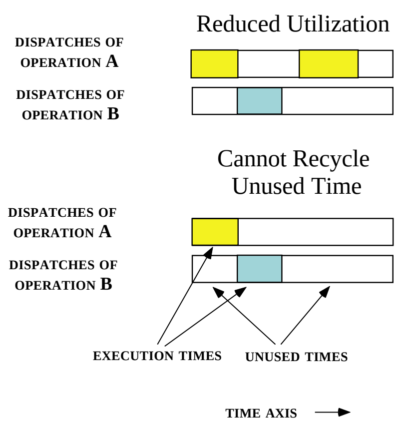
Terminology: Operation Characteristics

```
struct  RT_Info
{
  Time worstcase_exec_time_;
  Period period_;
  Criticality criticality_;
  Importance importance_;
  Dependency_Info dependencies_;
};
```



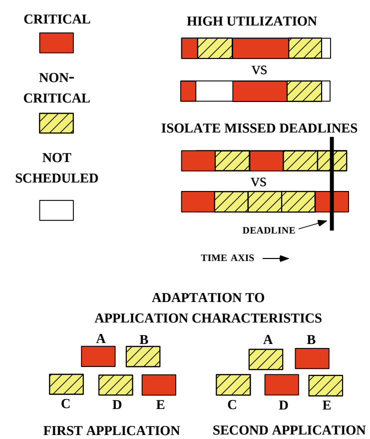
- *Criticality* is an application defined significance of the operation missing its deadline
- *Period* is the time interval between arrivals of dispatch requests for the operation
- *Execution time* is the longest time used by one execution of the operation
- *Importance* is a weaker secondary indication of the operation's significance

Limitations of Static Scheduling



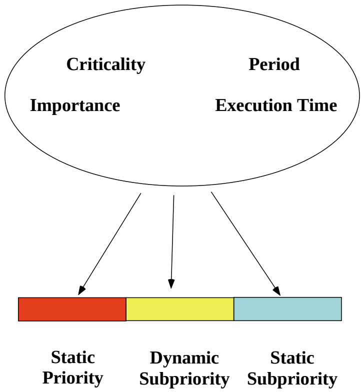
- Assigning priority by period limits achievable utilization
- Time cannot be reassigned if an operation is not called, or does not use its worst case computation time
- Goal: higher utilization
- Hypothesis: with dynamic scheduling techniques we can achieve this goal without undue overhead or instability of the schedule under load

Requirements for Hard Real-Time Dynamic Scheduling



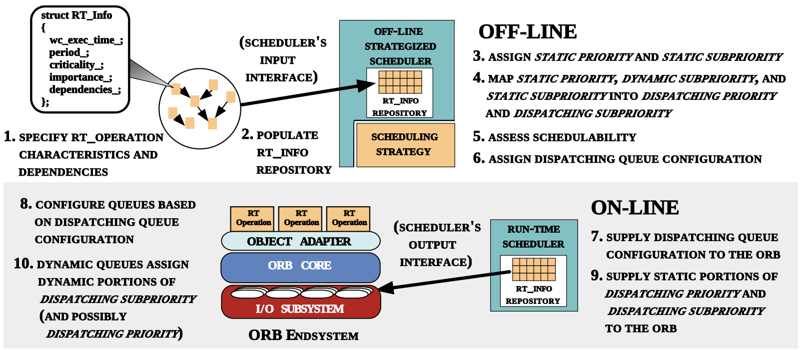
- Achieve higher *utilization*
  - Schedule more unused time
- Preserve *stability* of the schedule under load
  - Isolate missed deadlines to *non-critical* operations
- Adapt to application specified characteristics

Generalizing Priority Assignment



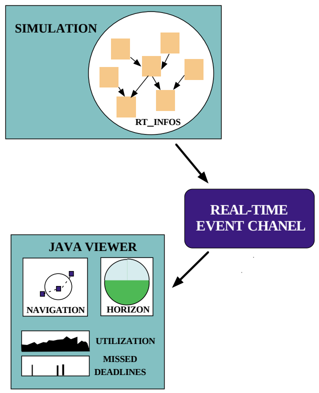
- Priority assignment is a *mapping* from operation characteristics into urgency
- Each scheduling strategy provides a distinct mapping
  - *Maximum Urgency First* (MUF)
  - *Minimum Laxity First* (MLF)
  - *Earliest Deadline First* (EDF)
  - *Rate Monotonic Scheduling* (RMS)

TAO Strategized Scheduling Architecture



- Scheduler performs on-line and off-line activities
- Scheduler provides configuration for QoS enforcement

Scheduler Effectiveness Demonstration



- **Metrics**
  - Simulation: Missed deadlines, Latency, Latency jitter, CPU utilization Scheduling overhead
  - Measured: Latency, Latency jitter
- **Can be run on different platforms: NT, Solaris, etc.**

Conclusions

- **Hybrid static-dynamic strategies can be implemented without excessive overhead**
  - Empirical end-to-end results using MUF in EC_Multiple
  - Empirical results for deadline and laxity based dispatching queues
- **Hybrid strategies can preserve the schedulability of the critical set**
  - Maximum Urgency First
- **Visualization of scheduling behavior is useful**
  - Simulations to explore alternative strategies
  - Measurements to benchmark performance