Measuring Distributed and Local Priority Inversions in Real-Time ORBs

Victor Giddings
Objective Interface Systems, Inc.
victor.giddings@ois.com

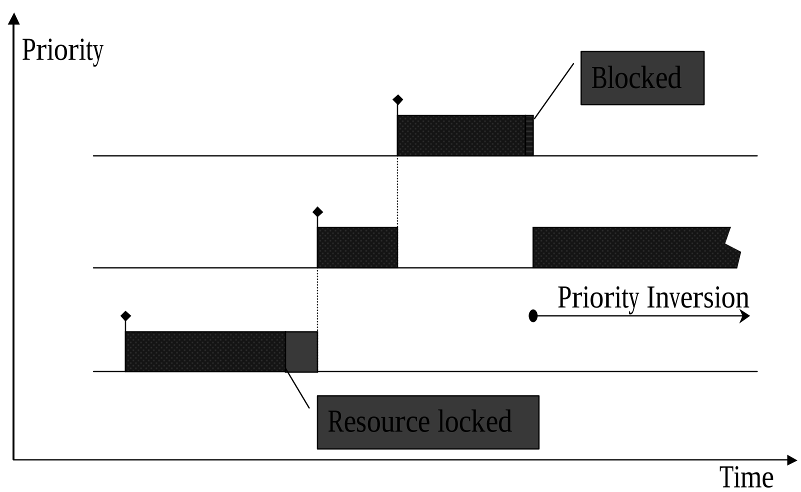
Background Real-Time Scheduling

- Tasks
  - Units of schedulable work
  - Assigned priorities
- Real-Time Scheduling
  - Preemptive – Lower priority tasks suspended when higher priority task enabled
  - Assignment of priority related to the tightness of deadlines, e.g., higher frequency implies higher priority
  - Fails if tasks miss deadlines

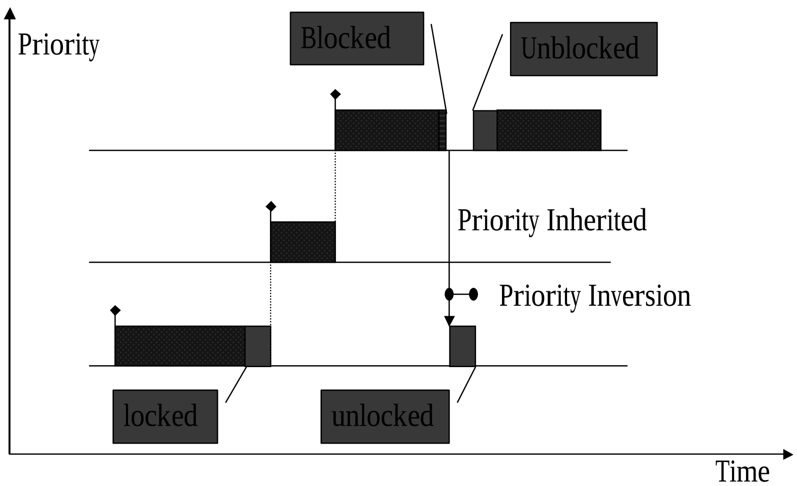
What Is Priority Inversion?

- Whenever the highest priority task that is otherwise enabled is not executing
  - Usual cause – contention for limited quantity resource with a lower priority task
  - For example, a lower priority task has locked a resource while in a critical section

Unbounded Priority Inversion



Priority Inheritance



Bounded Priority Inversion

- **Priority inheritance helps to bound priority inversion**
- **But reduces schedulability of application**

$$R_i = C_i + B_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

R_i – maximum response time of task i
C_i – maximum computation time of task i
B_i – maximum blocking time of task i
T_i – period of task i
$hp(i)$ – higher priority tasks than task i

Bounded Priority Inversions

$$B_i = \max_{\{k,s \mid k \in lp(i) \wedge s \in used\_by(k) \wedge ceil(s) \geq pri(i)\}} cs_{k,s}$$

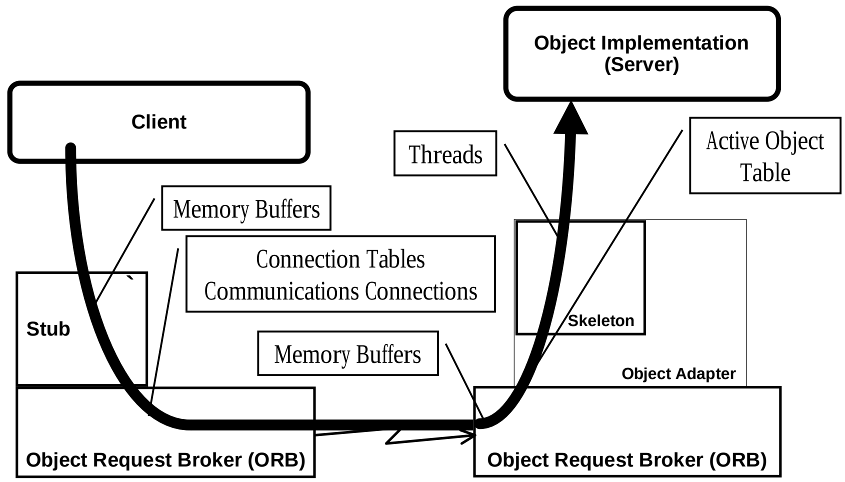
lp(i) – lower priority tasks
used_by(i) – critical section is used by task *i*
ceil(s) – priority ceiling of critical section
pri(i) – priority of task *I*
$cs_{k,s}$ – execution time of task *k* in critical section *s*

- **Priority inversions may be counted multiple times**

Distributed Priority Inversions

- **Occur when request processing on a remote node must contend for resources with other requests**
- **May require distributed priority inheritance to resolve**

Sources of Priority Inversions in ORBs



Avoiding Priority Inversions in ORBs

- **ORBs avoid priority inversion by avoiding resource contention**
  - **Buffer Pools**
  - **Multiple Communications Connections**
  - **Thread Pools**
- **Where contention remains, resource management must be priority aware**

Measuring Priority Inversion

- Possible to build "constructive model"
- Need
  - Extensive benchmark or estimation of ORB internal operation
  - Detailed model of ORB behavior
- However
  - Requires extensive disclosure of ORB internals
  - Result would be obsoleted by ORB revision
- Measurement seems to be required

Measuring Priority Inversion Previous Work

- Schmidt et all (WUSTL)
  - Measured Round Trip Latency of High Priority Task
  - Added lower priority tasks
  - Yields - Figure of Merit
    - Useful for comparing ORB implementations
    - Slope may be useful as indicator of blocking factor
  - Present form may be too limited for schedulability analysis

Measuring Priority Inversion Previous Work

- ❖ **Rttaskdemo (OIS)**
  - ❒ **Shipped as part of ORB*express* RT**
  - ❒ **Instrumented multi-priority benchmark**
  - ❒ **Yields**
    - ♦ **Throughput and variability at each priority level**
    - ♦ **Demonstrates lack of unbounded inversions**
    - ♦ **Unclear applicability to schedulability analysis**

Measuring Priority Inversion Previous Work

- ❖ **HARTStone Benchmark (SEI) –**
  - ❒ **Series of**
    - ♦ **Sets of Periodically Scheduled Tasks**
    - ♦ **Performing Synthetic Loads**
  - ❒ **Varied**
    - ♦ **Phasing**
    - ♦ **Loads**
  - ❒ **Tracked deadlines**

Measuring Priority Inversion Proposed Method

- **CORBA HARTstone**
  - **Run HARTstone series**
    - **Compare remote and local operations**
    - **Estimate blocking factor from regression analysis**