Implementation of Real-Time Scheduling Algorithms

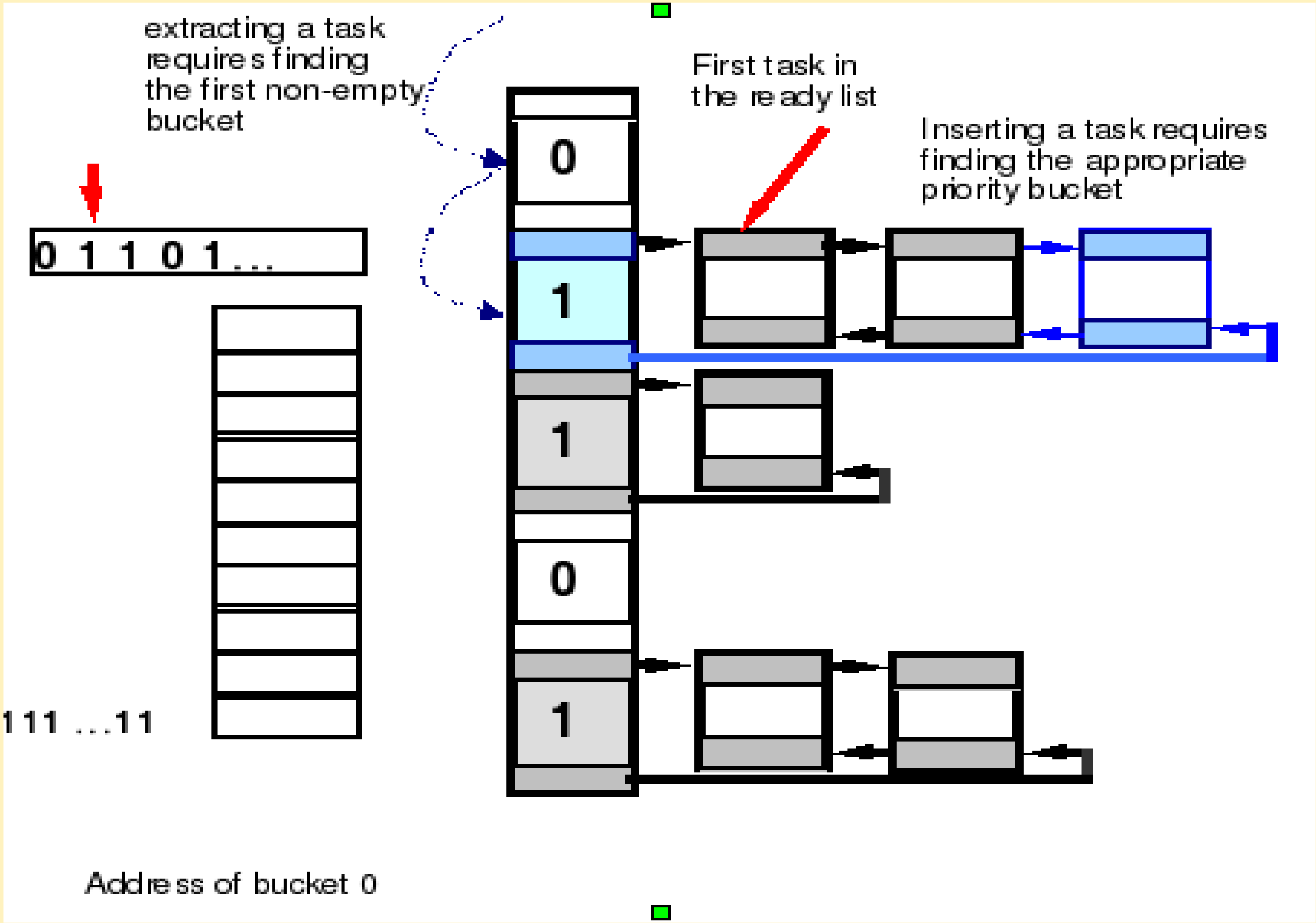
Luca Abeni
luca.abeni@unitn.it

November 2, 2014

Implementation of Fixed Priorities

- When implementing fixed priority it is possible to have an array of queues (one for each priority level)
- Insertion into the queue is $O(1)$ operation
- Extracting from the queue would entail $O(n)$ search on the different priority levels to find the first nonempty queue
- However, we can use a bitmap (i.e., an array of bits) to tag the queues that are non-empty
- Extraction becomes $O(1)$ if we have a microinstruction that returns the first 1 bit in a word (CLZ)
- If not we can use a table to implement the operation $\lceil \log w \rceil$, but we need as many entries as the bits in the table

Implementation of fixed priority - I



Implementing EDF

- EDF queueing is more complex
  - Dynamic priorities $\rightarrow$ cannot use the "bitmask trick"
  - No $O(1)$ complexity
- Can EDF be implemented with something better than $O(n)$ complexity?
- Data structure storing ordered keys, with efficient:
  - Insertion
  - Selection of the first entry
  - Removal of the first entry
  - Efficient removal of non-first entries is not too important
  - Efficient search of specific entries is not too important

Red/Black Trees

- The deadline queue can be implemented using a Red/Black tree
  - Self-balancing tree, based on nodes colouring
  - $O(log(n))$ on all the operations
- Not too bad, if n is not too large!
- $\Rightarrow$ Red/Black trees make EDF implementable in practice (without too much overhead)!