ICT-eMuCo
Embedded Multi-Core Processing for Mobile Communication Systems
Service Contracts adapted to eMuCo

eMuCo

SEVENTH FRAMEWORK PROGRAMME

Service Contracts

A ***service contract*** describes the minimum resource requirements of an application. ***Negotiations*** among service contracts determine if a system can satisfy the minimum requirements of all applications.

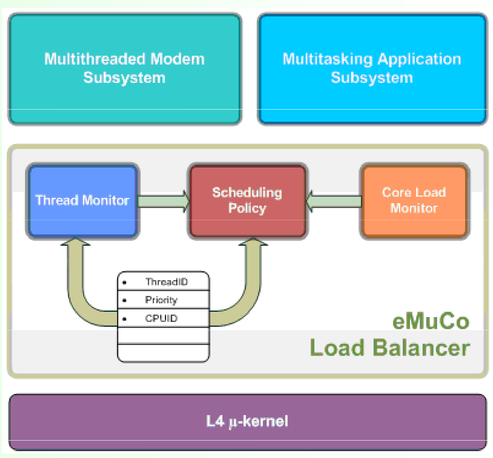
A ***service contract*** also serves the purpose of isolating the execution of its application from the others. Enforcing ***service contracts*** at runtime requires the operating system to support some form of ***resource reservation*** (e.g. TDMA, execution time server etc.).

Apart from timing requirements, ***service contracts*** in a multicore mobile embedded environment should also specify the required parallelism and the energy consumption level base on which they are derived.

eMuCo Load Balancer

eMuCo Load Balancer Architecture

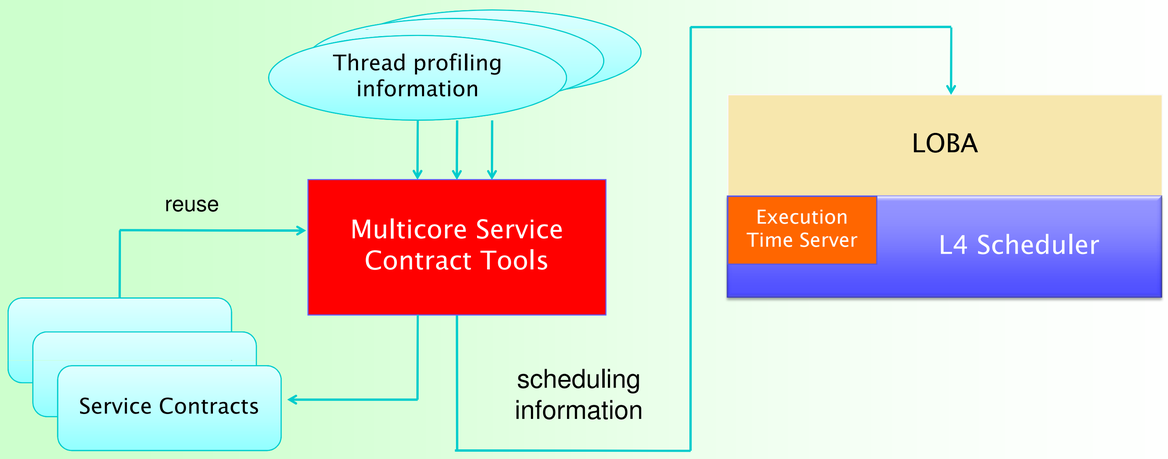
The basic functionality of a ***load balancer*** in a multicore mobile embedded system is to provide support for real-time and non-real-time applications with different performance and timing requirements. The scheduling policy of the current ***eMuCo load balancer*** determines ***statically*** how threads are allocated to CPUs.



Load Balancer Architecture

Integration with Service Contracts

In eMuCo, a hierarchical multiprocessor scheduling framework based on the fully partitioned scheduling approach and aperiodic server technique is being developed. Its corresponding schedulability analysis will be implemented in our contract negotiation tool, which decides whether all contracts can be satisfied.



Integration

Service Contract Tools

Contract Generation Tool (CG)

This tool reads thread information from a data file and generates a ***service contract*** that describes the total resource requirements (at a given CPU frequency). The input data file contains the following information:

- Thread ID (ordered according to thread priorities),
- Thread ***utilization*** (obtained by another tool called ***csamon*** at a given CPU frequency).

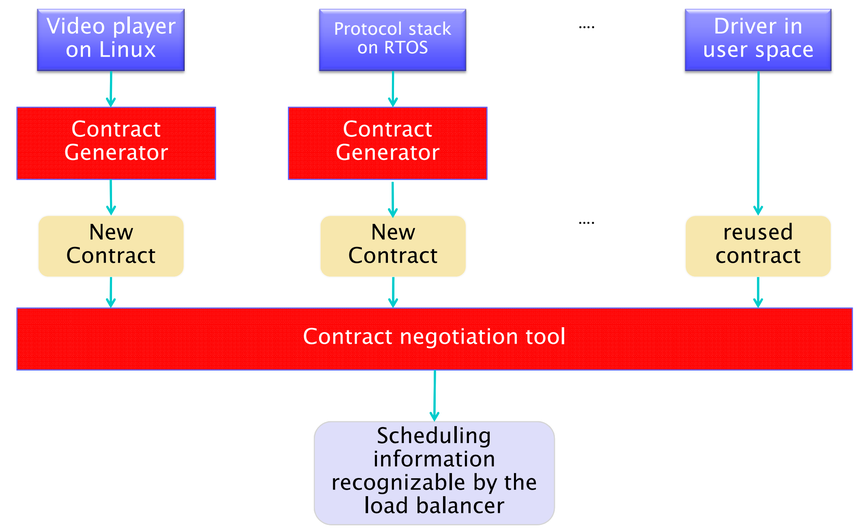
The ***service contract*** generated by this tool should contain:

- the ***CPU frequency*** for which this contract is generated,
- the number of ***dedicated CPUs*** required,
- thread allocation to these CPUs (ordered according to thread priorities),
- the number of ***servers*** required and the ***utilization*** of each server,
- thread allocation to these servers (ordered according to thread priorities).

Contract Negotiation Tool (CN)

This tool reads a set of ***service contracts*** that are generated by ***CG*** for the same ***CPU frequency*** and performs a negotiation for a given platform. The output data file contains the scheduling information needed by the ***eMuCo load balancer***:

- The ***CPU frequency*** assumed by all the concerned contracts,
- CPU ID and (if dedicated) the IDs of threads that will run on this CPU (ordered according to priorities),
- CPU ID and (if not dedicated) the IDs of servers that will run on this CPU (ordered according to priorities),
- The ***period*** and ***budget*** of each server,
- IDs of threads that will run under each server (ordered according to priorities).



example of eMuCo contract tool chain

Formal Verification of Service Contracts

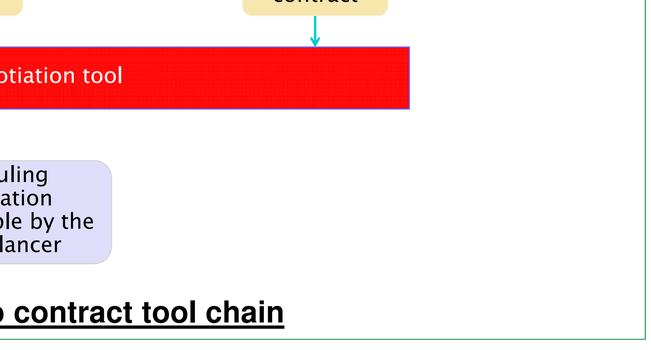
Schedulability analysis based on real-time multiprocessor scheduling theory is often ***pessimistic***. One way to overcome this problem and perform exact schedulability analysis is represented by ***formal verification***. Starting from the ***hierarchical multiprocessor scheduling*** solution proposed in eMuCo, the ***timed automata theory*** is applied for specifying the service contracts for applications and ***model checking*** techniques are employed as a technique for deciding on their schedulability on a multiprocessor system.

Using ***timed automata*** model, investigations on the possibility of using a global scheduling policy with migration of tasks between cores have been undertaken. The main ***challenges*** with this model checking based approach to ***schedulability analysis*** are:

- modeling ***task preemption*** due to the impossibility to stop clocks,
- reducing ***model checking time***,
- ***scalability*** with respect to the size of the applications which can be analyzed.

The ways the eMuCo model addresses these challenges are:

- ***task preemption*** problem was solved by using a discrete time formalism where execution time for each task is measured using an integer-valued clock,
- ***model checking time*** is known to grow exponentially with the number of clocks in the model and the constants they are compared with so this problem was solved by using just one real-valued clock and comparing it just with one constant,
- increased ***scalability*** is achieved by modeling the application as a whole and not each of its tasks independently.



www.emuco.eu

ICT-eMuCo is a European project supported under the Seventh Framework Programme (7FP) for research and technological development