Since sharing resource access in most of modern operating systems must ensure mutual exclusions without deadlocks, resource synchronization protocols have been widely used to serve this need. Without any aid of protocols, a higher priority job may be blocked by a lower-priority job, which is called priority inversion as shown in the following figure.

In uniprocessor systems, locking protocols have been widely used to mitigate the effect of priority inversion based on priority inheritance, e.g., the priority inheritance protocol (PIP), the priority ceiling protocol (PCP), and the stack resource policy (SRP), which are already shown reasonably good at handling synchronization and mutual exclusions.

Nowadays, multi-core systems have become the mainstream processors. To schedule real-time tasks on multi-core systems, there have been three widely adopted paradigms: partitioned, global, and semi-partitioned scheduling. With the above task scheduling paradigms, a nature question expectedly arises: What is the best resource synchronization protocols for multi-core systems?

Since none of the resource sharing protocol can dominate all the others so far even without considering the real-world overhead, the problem is still an ongoing topic in the researches. Recently, Shi et al. tried to answer this problem on a real-time operating system (RTOS) LITMUS-RT. In this thesis, we follow a similar flow on another RTOS named RTEMS. Thanks for [1], one of the complicated protocol is implemented already. With this good basis, we just have implement two more protocols in RTEMS. Afterwards, we can analyze the overhead and the performance on a real multi-core system.

In this thesis, the student will first study the design of MrsP in [1] with the source code in the latest version RTEMS. We will provide a basis frame work and guide the student how to implement a resource sharing protocol in RTEMS. After finishing the implementations, the student should validate the protocols and evaluate the efficiency of the implementations.

Other suggestions and related topics are also welcome. Please do not hesitate to make an appointment.

Required Skills:

- Good Knowledge of C
- Knowledge of Embedded System and Linux
- Real Time Scheduling knowledge is beneficial

Acquired Skills after the thesis:

- Knowledge of resource sharing protocols on multi-core systems and RTEMS.
- Design, Analysis, Implementation of system software on real-time embedded system

[1] Catellani et al., Challenges in the Implementation of MrsP, LNCS 2015
[2] Shi et al., Implementation and Evaluation of Multiprocessor Resource Synchronization Protocol (MrsP) on LITMUSRT, OSPERT 2017