Real-Time Operating System (Day 3 Lab)

Jong-Chan Kim

Graduate School of Automotive Engineering

KMU 국민대학교 KOOKMIN UNIVERSITY

17-1. Loss (공유 자원 data loss 확인)

- Task1: Low priority, AUTOSTART
- Task2: High priority, 1 ms 주기 실행
- 공유 전역 변수 (`volatile unsigned long shared`)
  - Task1의 루프에서 `shared++`
  - Task2 주기적으로 반복 실행하며 `shared ++`
- 양 쪽에서 더한 숫자가 모두 유지되는지 확인
- Task와 ISR 사이에서도 같은 문제가 발생하는지 확인

17-1. Loss

bsw.c

```
int main(void)
{
    osEE_tc_stm_set_clockpersec();
    osEE_tc_stm_set_sr0(1000U, 1U);
```

```
#include "bsw.h"
volatile unsigned long shared = 0;
```

```
TASK(Task1)
{
    unsigned long i;
    printfSerial("Task1 Begins...\n");
    for (i = 0; i < 20000000; i++) {
        shared++;
    }
    printfSerial("Added 20000000 to shared\n");
    printfSerial("counter = %lu\n", shared);
    printfSerial("Task1 Finishes...\n");
    TerminateTask();
}
```

```
TASK(Task2)
{
    static unsigned long i = 0;
    if (i < 500) {
        shared++;
    } else if (i == 500) {
        printfSerial("Added 500 to shared\n");
    }
    i++;
    TerminateTask();
}
```

17-1. Loss

- Resource를 이용해서 Integrity Loss 문제 해결 필요

```
..............
...OS Starts...
..............
Task1 Begins...
Added 500 to shared
Added 20000000 to shared
counter = 20000256
Task1 Finishes...
```

20000000 + 500 = 20000256(!!!)

17-2. No Loss

- OSEK의 RESOURCE 기능을 이용하여 Data Loss 문제 해결

```
GetResource(S1);
shared++;
ReleaseResource(S1);
```

```
RESOURCE S1 {
    RESOURCEPROPERTY = STANDARD;
};

…

TASK Task1 {
    PRIORITY = 1;
    STACK = SHARED;
    SCHEDULE = FULL;
    AUTOSTART = TRUE;
    ACTIVATION = 1;
    RESOURCE = S1;
};
```

17-2. No Loss

- Data Integrity 문제 해결

```
..............
...OS Starts...
..............
Task1 Begins...
Added 500 to shared
Added 20000000 to shared
counter = 20000500
Task1 Finishes...
```

20000000 + 500 = 20000500(!!!)

18. Mutex

- mutex.h

```
#ifndef MUTEX_H_
#define MUTEX_H_

#define LOCKED   1
#define UNLOCKED  0

typedef struct _MutexType {
    int flag;
    EventMaskType event;
    TaskType waiting_task;
} MutexType;

void InitMutex(MutexType *mutex, EventMaskType event);

void GetMutex(MutexType *mutex);

void ReleaseMutex(MutexType *mutex);


#endif /* MUTEX_H_ */
```

Waiting/Wakeup 을 위해 Event 지정 필요

PCP 없이 Mutex 사용할 경우 문제점을 확인하기 위한 Dummy 구현

18. Mutex

- mutex.c : 다음 설명을 참고하여 Mutex함수를 구현해보기

```
#include "ee.h"
#include "bsw.h"
#include "mutex.h"

void InitMutex(MutexType *mutex, EventMaskType event)
{
    1
}

void GetMutex(MutexType *mutex)
{
    2
}

void ReleaseMutex(MutexType *mutex)
{
    3
}
```

1. InitMutex
   - Mutex를 초기화
   - 초기상태: `flag = UNLOCKED, waiting_task = 0, event = event`

2. GetMutex
   - Mutex가 이미 LOCKED 상태이면
     - 현재 Task의 ID를 `waiting_task`에 저장 (`GetTaskID`)
     - 해당 이벤트가 올 때까지 블로킹 (`WaitEvent`)
   - Mutex를 획득하면 flag를 LOCKED 상태로 변경

3. ReleaseMutex
   - Mutex가 LOCKED 상태라면
     - `flag`를 UNLOCKED 상태로 변경
     - 만약 Mutex를 기다리는 Task가 있다면 해당 Task에 이벤트를 보내서 깨움 (`SetEvent`)

18. Mutex

- mutex.c

```c
#include "ee.h"
#include "bsw.h"
#include "mutex.h"

void InitMutex(MutexType *mutex, EventMaskType event)
{
    mutex->flag = UNLOCKED;
    mutex->waiting_task = 0;
    mutex->event = event;
}

void GetMutex(MutexType *mutex)
{
    if (mutex->flag == LOCKED) {
        printfSerial("  -->  BLock");
        GetTaskID(&(mutex->waiting_task));
        WaitEvent(mutex->event);
    }
    mutex->flag = LOCKED;
}
```

```c
void ReleaseMutex(MutexType *mutex)
{
    if (mutex->flag == LOCKED) {
        mutex->flag = UNLOCKED;
        if (mutex->waiting_task != 0) {
            SetEvent(mutex->waiting_task, mutex->event);
        }
    }
}
```

18. Mutex

- Mutex 선언
- Timer ISR 이용
  - Mutex 초기화
  - Task Activation
- Mutex의 동작 확인

```
MutexType s1;


ISR2(TimerISR)
{
    osEE_tc_stm_set_sr0_next_match(1000000U);
    static long c = -5;
    printfSerial("\n%4ld: ", ++c);
    if(c == -4) {
        InitMutex(&s1, Event1);
    } else if (c == 0) {
        ActivateTask(TaskL);
    } else if (c == 5) {
        ActivateTask(TaskH);
    }
}
```

18. Mutex

- Mutex.c 추가

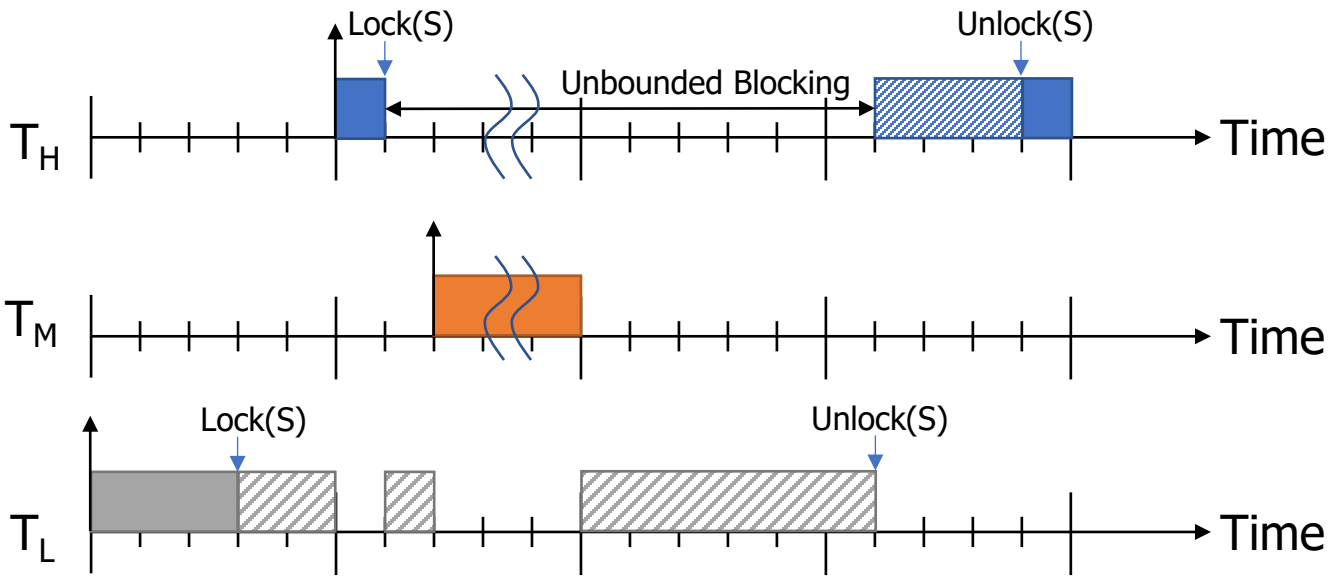
```
 APPDATA tricore_mc {
        APP_SRC="illd/src/IfxAsclin_Asc.c";
        APP_SRC="illd/src/IfxStm.c";
        APP_SRC="illd/src/IfxStm_cfg.c";
        …

        APP_SRC="illd/src/IfxScuEru.c";
        APP_SRC="illd/src/IfxVadc_Adc.c";
        APP_SRC="illd/Libraries/iLLD/TC27D/Tricore/_Impl/IfxVadc_cfg.c";
        APP_SRC="illd/Libraries/iLLD/TC27D/Tricore/Vadc/std/IfxVadc.c";

        APP_SRC="mutex.c";
        APP_SRC="bsw.c";
        APP_SRC="asw.c";
    };
```

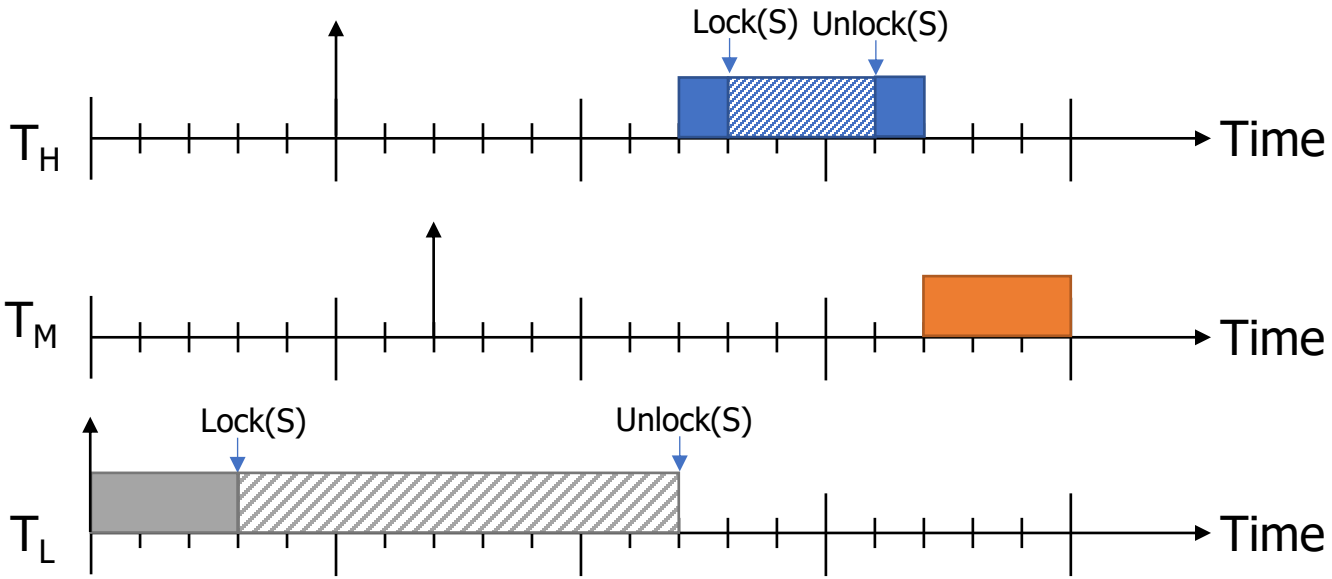
19-1. Priority Inversion

- 아래 스케쥴 재현하기



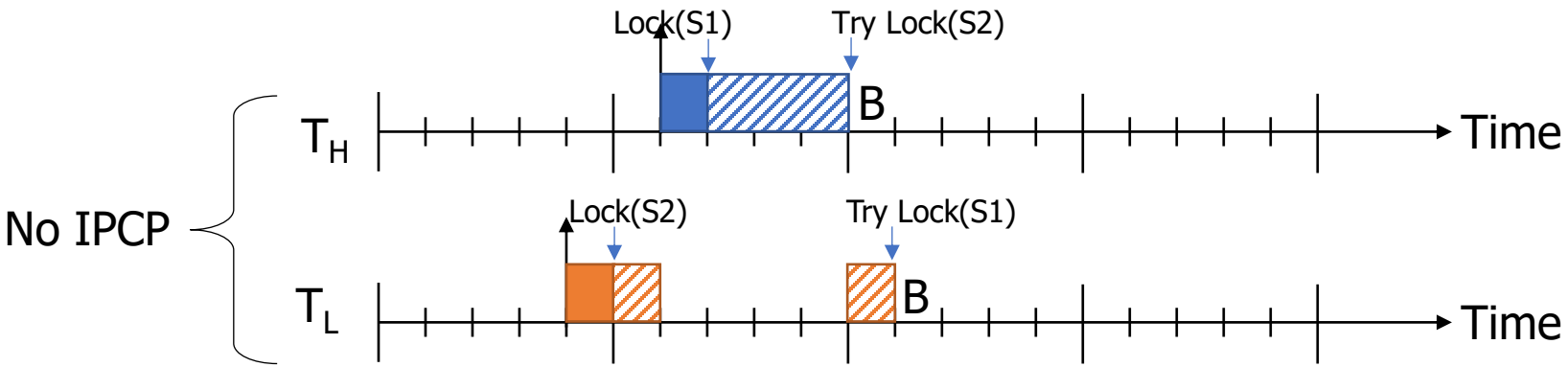
19-2. No Priority Inversion

- 아래 스케쥴 재현하기
- IPCP가 적용된 Resource를 이용하여 스케쥴 변화 확인



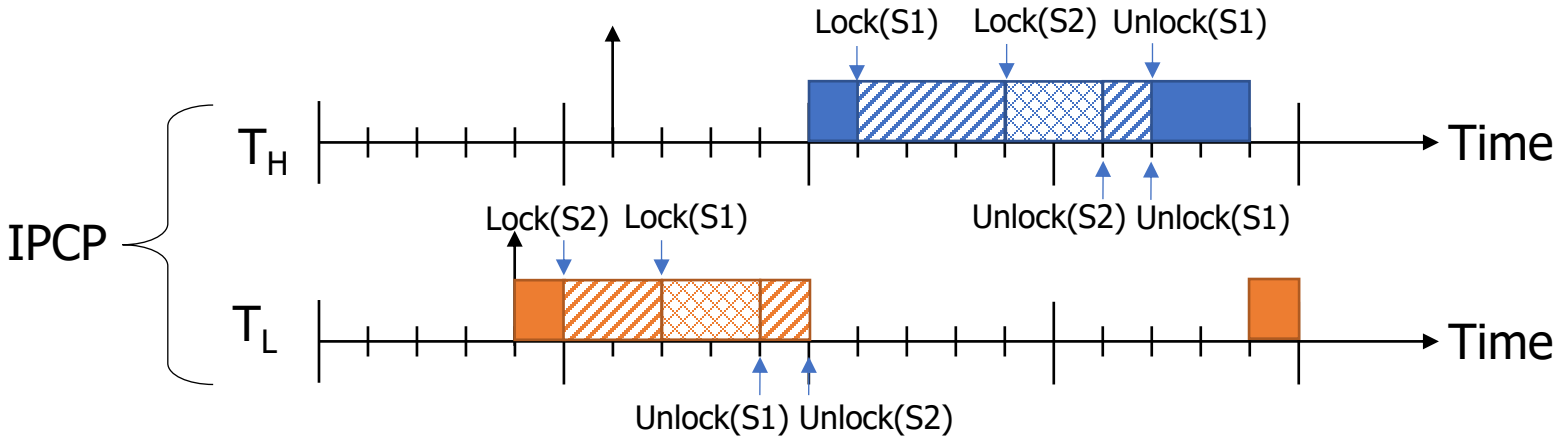
20-1. Deadlock

- Deadlock 재현하기



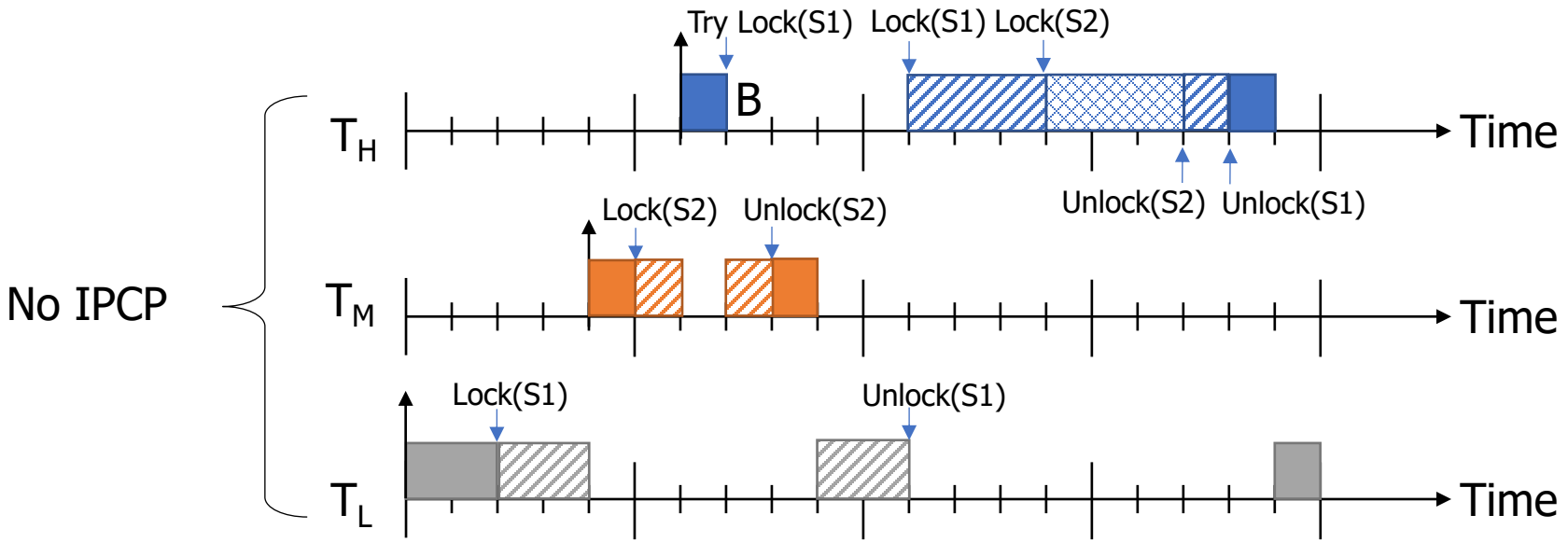
20-2. No Deadlock

- Deadlock 재현하기
- IPCP가 적용된 Resource를 이용하여 Deadlock 해결



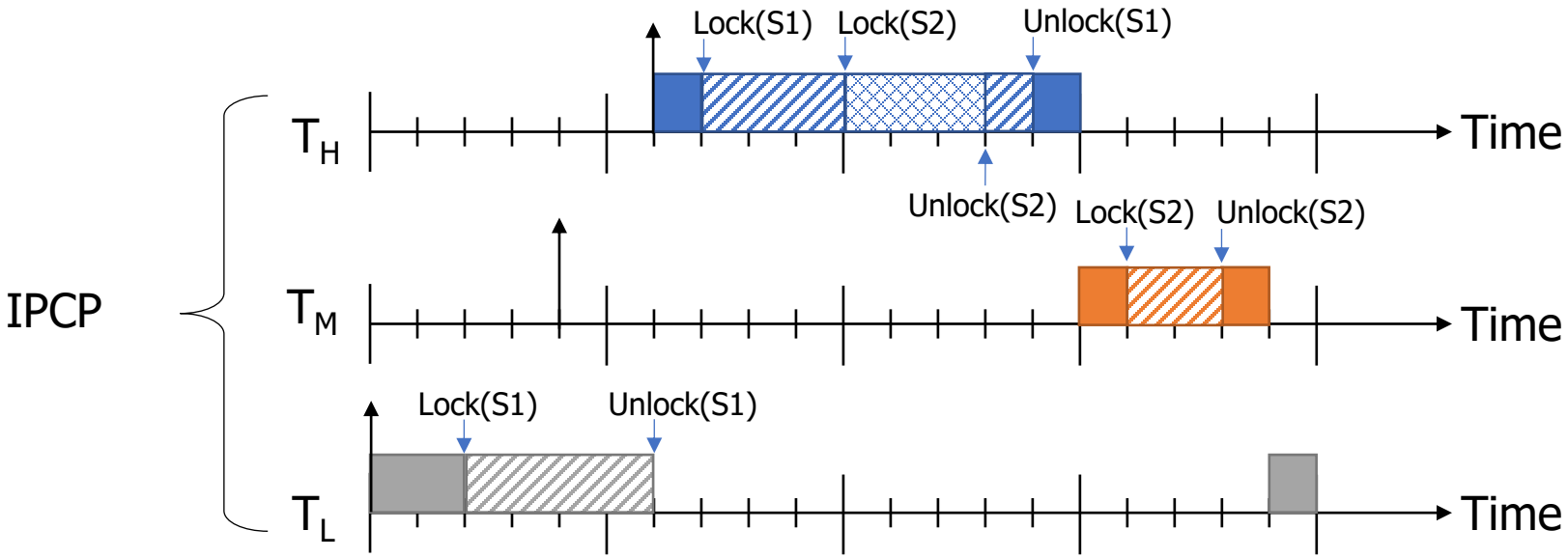
21-1. Before IPCP

- 아래 스케쥴 재현하고 IPCP 동작 확인



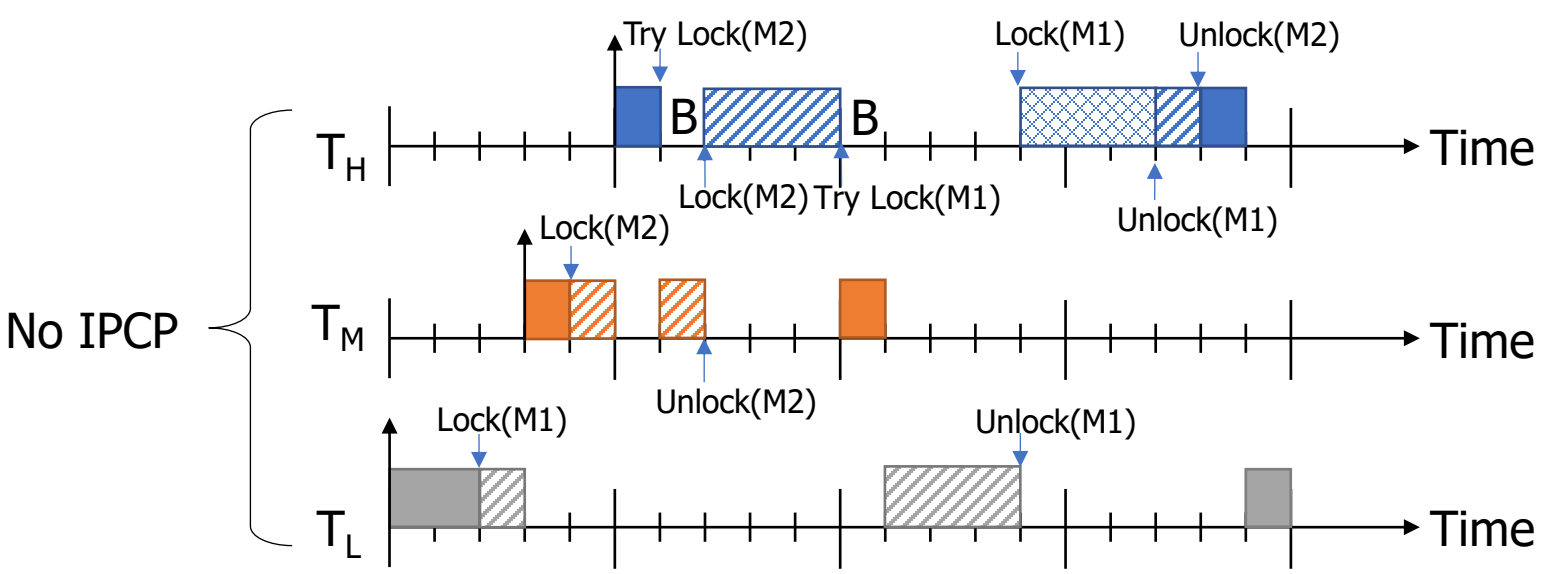
21-2. After IPCP

- 아래 스케쥴 재현하고 IPCP 동작 확인



21-3. Before/After IPCP

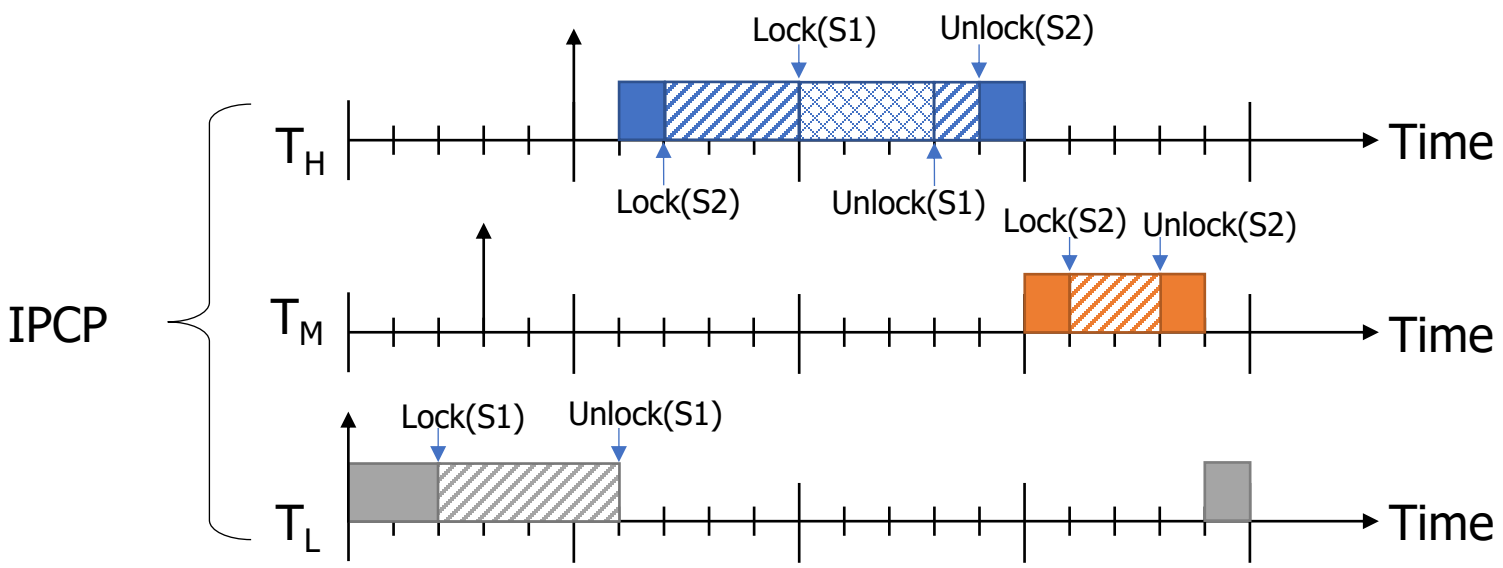
- Mutex로 아래 시스템 구현




```
 0: <TaskL begins.>
 1:
 2: TaskL : Try Lock(S1). TaskL : Get Lock(S1).
 3: <TaskM begins.>
 4: TaskM : Try Lock(S2). TaskM : Get Lock(S2).
 5: <TaskH begins.>
 6: TaskH : Try Lock(S2).   -->  BLock
 7: TaskM : Release Lock(S2). TaskH : Get Lock(S2).
 8:
 9:
10: TaskH : Try Lock(S1).   -->  BLock
11: <TaskM ends.>
12:
13:
14: TaskL : Release Lock(S1). TaskH : Get Lock(S1).
15:
16:
17: TaskH : Release Lock(S1).
18: TaskH : Release Lock(S2).
19: <TaskH ends.>
20: <TaskL ends.>
```

21-3. Before/After IPCP

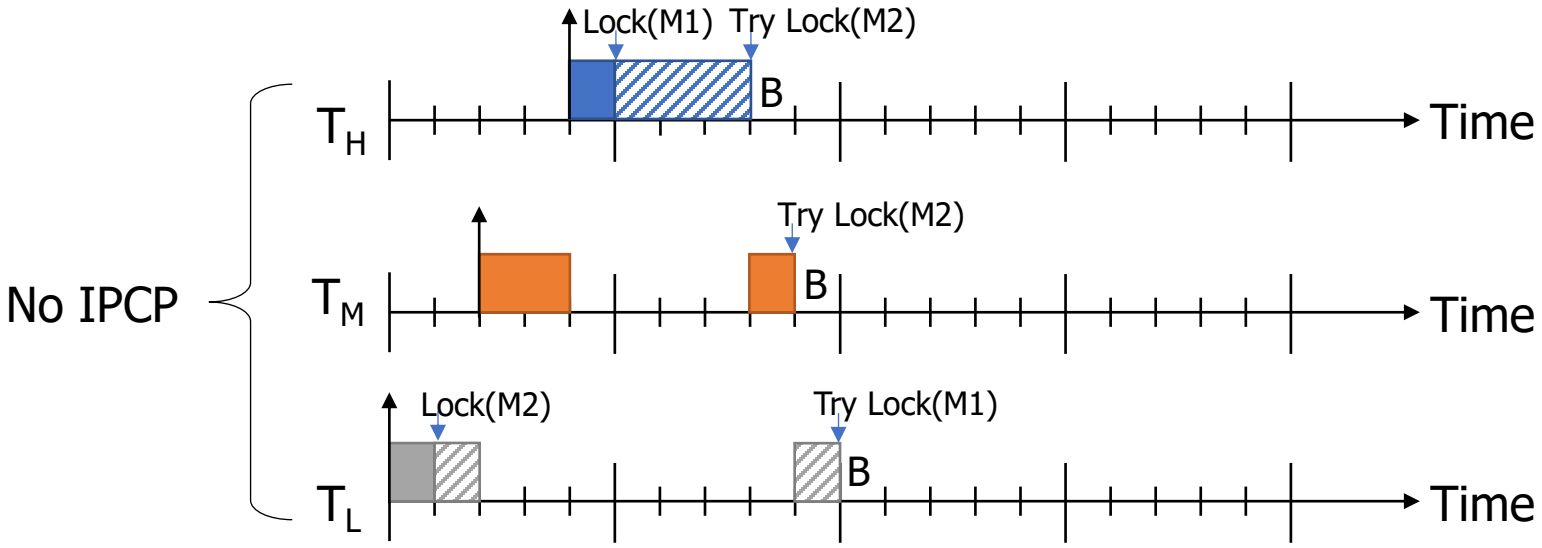
- Mutex를 Resource로 교체, 차이 비교




```
 0: <TaskL begins.>
 1:
 2: TaskL : Try Lock(S1). TaskL : Get Lock(S1).
 3:
 4:
 5:
 6: TaskL : Release Lock(S1). <TaskH begins.>
 7: TaskH : Try Lock(S2). TaskH : Get Lock(S2).
 8:
 9:
10: TaskH : Try Lock(S1). TaskH : Get Lock(S1).
11:
12:
13: TaskH : Release Lock(S1).
14: TaskH : Release Lock(S2).
15: <TaskH ends.> <TaskM begins.>
16: TaskM : Try Lock(S2). TaskM : Get Lock(S2).
17:
18: TaskM : Release Lock(S2).
19: <TaskM ends.>
20: <TaskL ends.>
```

21-4. Before/After IPCP

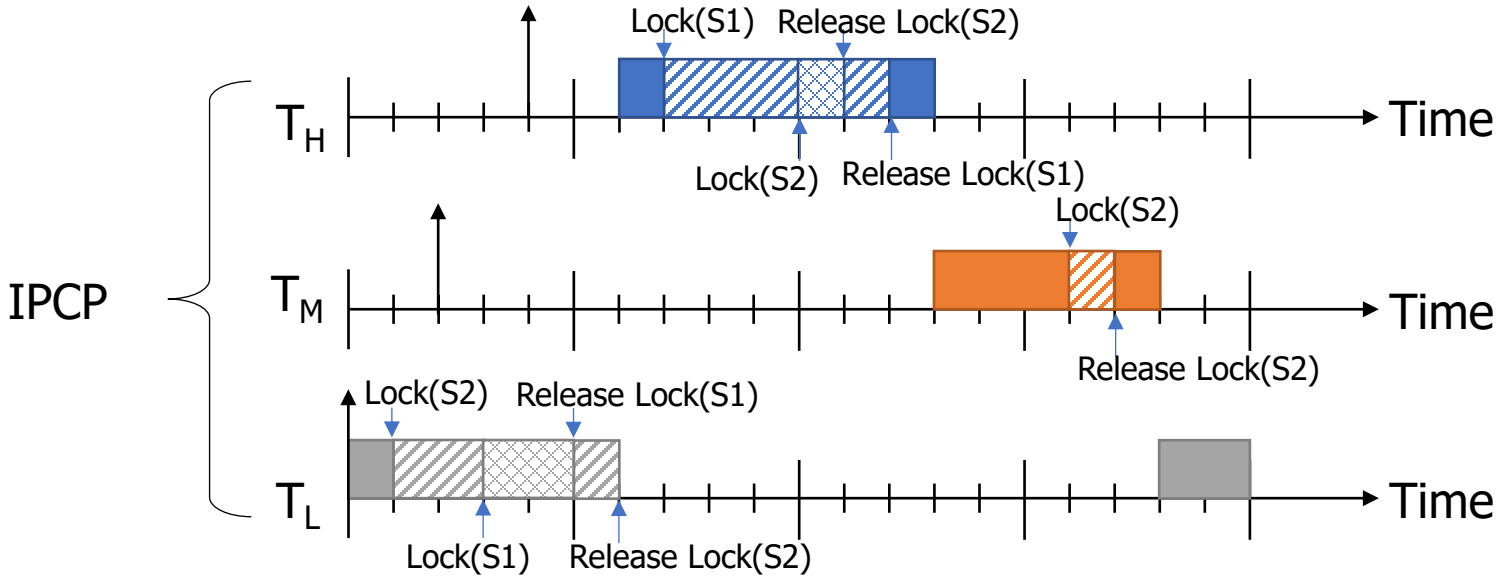
- Mutex로 아래 시스템 구현




```
 0: <TaskL begins.>
 1: TaskL : Try Lock(S2). TaskL : Get Lock(S2).
 2: <TaskM Begins.>
 3:
 4: <TaskH begins.>
 5: TaskH : Try Lock(S1). TaskH : Get Lock(S1).
 6:
 7:
 8: TaskH : Try Lock(S2).    -->  BLock
 9: TaskM : Try Lock(S2),    -->  BLock
10: TaskL : Try Lock(S1).    -->  BLock
11:
12:
13:
14:
15:
16:
17:
18:
19:
20:
```

21-4. Before/After IPCP

- Mutex를 Resource로 교체, 차이 비교




```
 0: <TaskL begins.>
 1: TaskL : Try Lock(S2). TaskL : Get Lock(S2).
 2:
 3: TaskL : Try Lock(S1). TaskL : Get Lock(S1).
 4:
 5: TaskL : Release Lock(S1).
 6: TaskL : Release Lock(S2). <TaskH begins.>
 7: TaskH : Try Lock(S1). TaskH : Get Lock(S1).
 8:
 9:
10: TaskH : Try Lock(S2). TaskH : Get Lock(S2).
11: TaskH : Release Lock(S2).
12: TaskH : Release Lock(S1).
13: <TaskH ends.> <TaskM Begins.>
14:
15:
16: TaskM : Try Lock(S2), TaskM : Get Lock(S2).
17: TaskL : Release Lock(S2).
18: <TaskM ends.>
19:
20: <TaskL ends.>
```

Questions