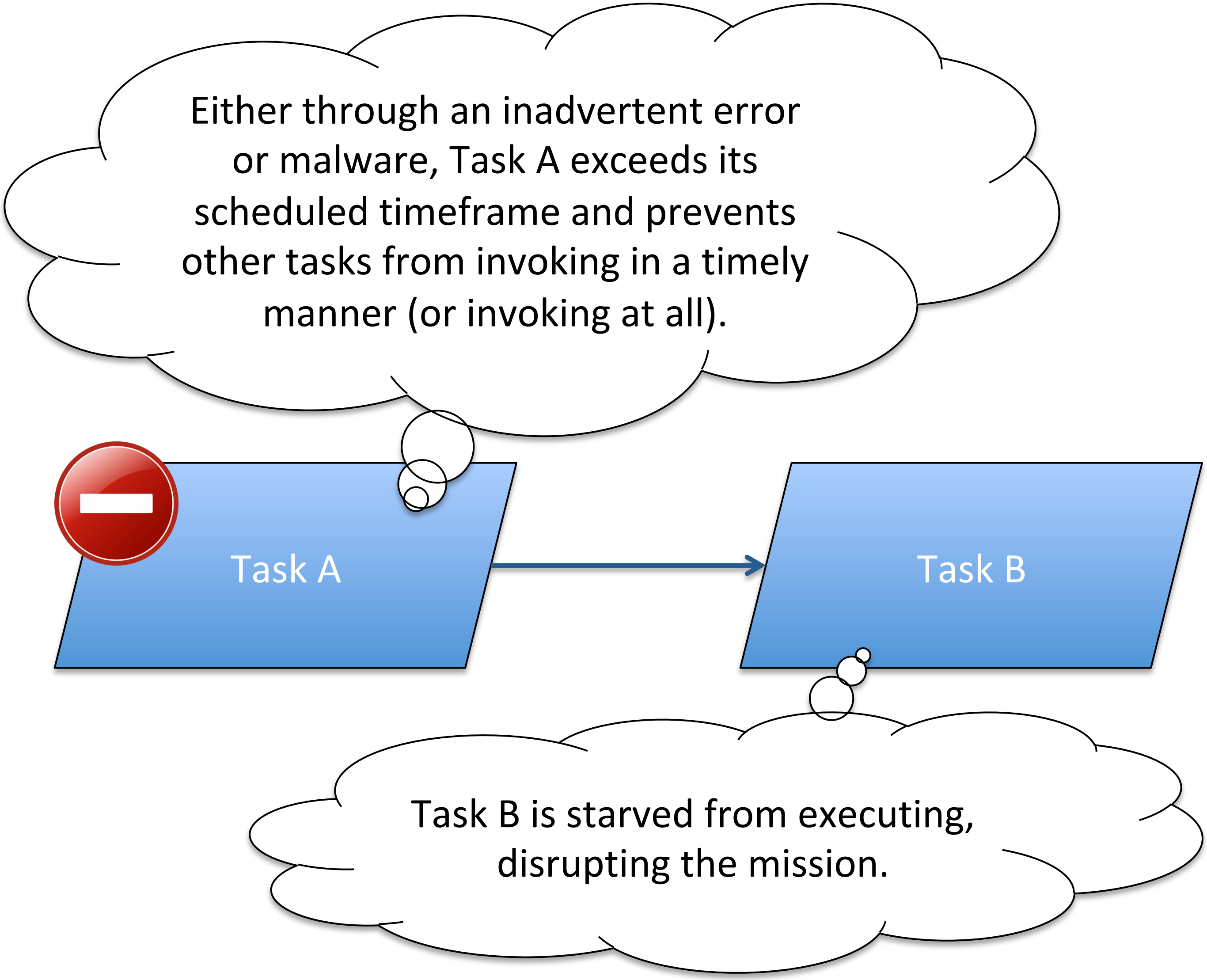
Adventium LABS

KANSAS STATE UNIVERSITY

CASE Temporal Isolation Overview

November 5, 2019

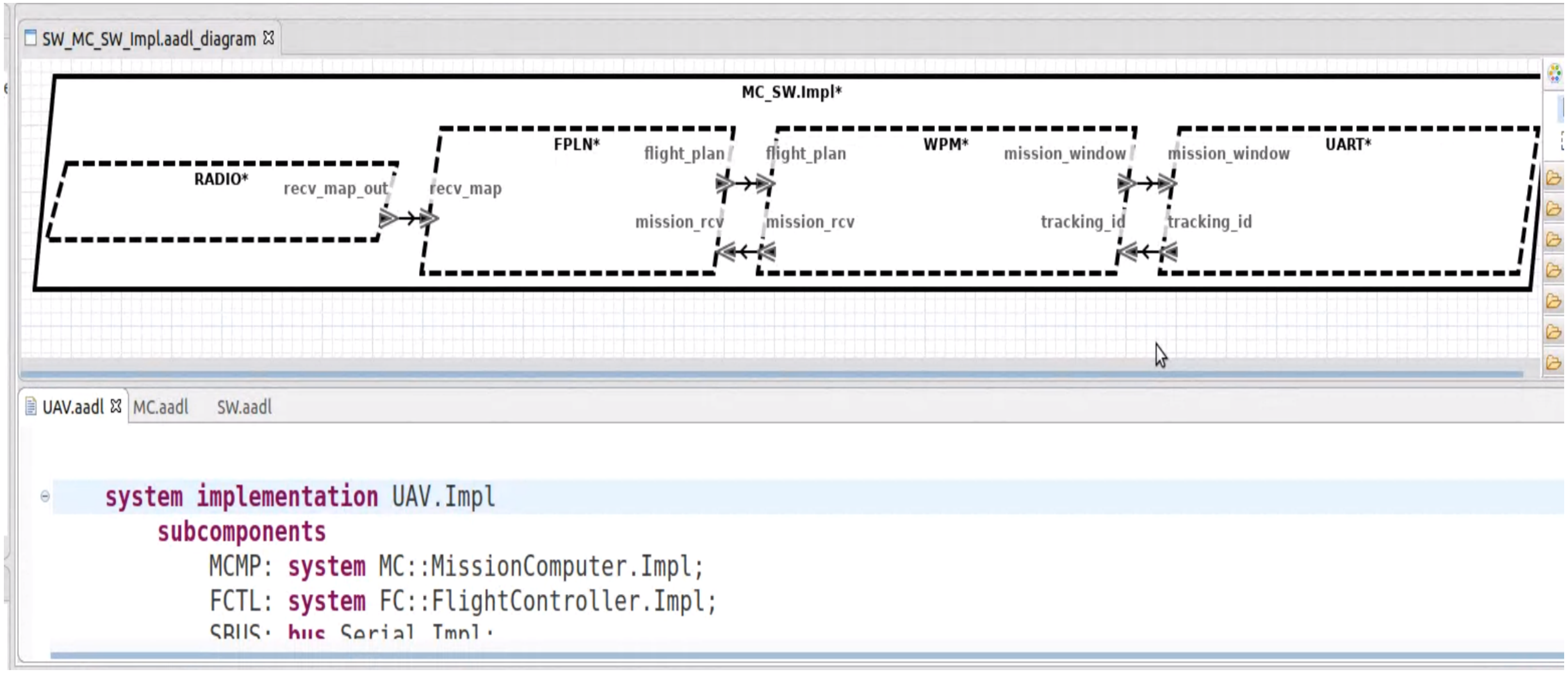
Security Scenario in Real-Time Systems



Either through an inadvertent error or malware, Task A exceeds its scheduled timeframe and prevents other tasks from invoking in a timely manner (or invoking at all).

Task B is starved from executing, disrupting the mission.

CASE Simple UAV Model



```
system implementation UAV.Impl
    subcomponents
        MCMP: system MC::MissionComputer.Impl;
        FCTL: system FC::FlightController.Impl;
        SBUS: bus Serial.Impl;
```

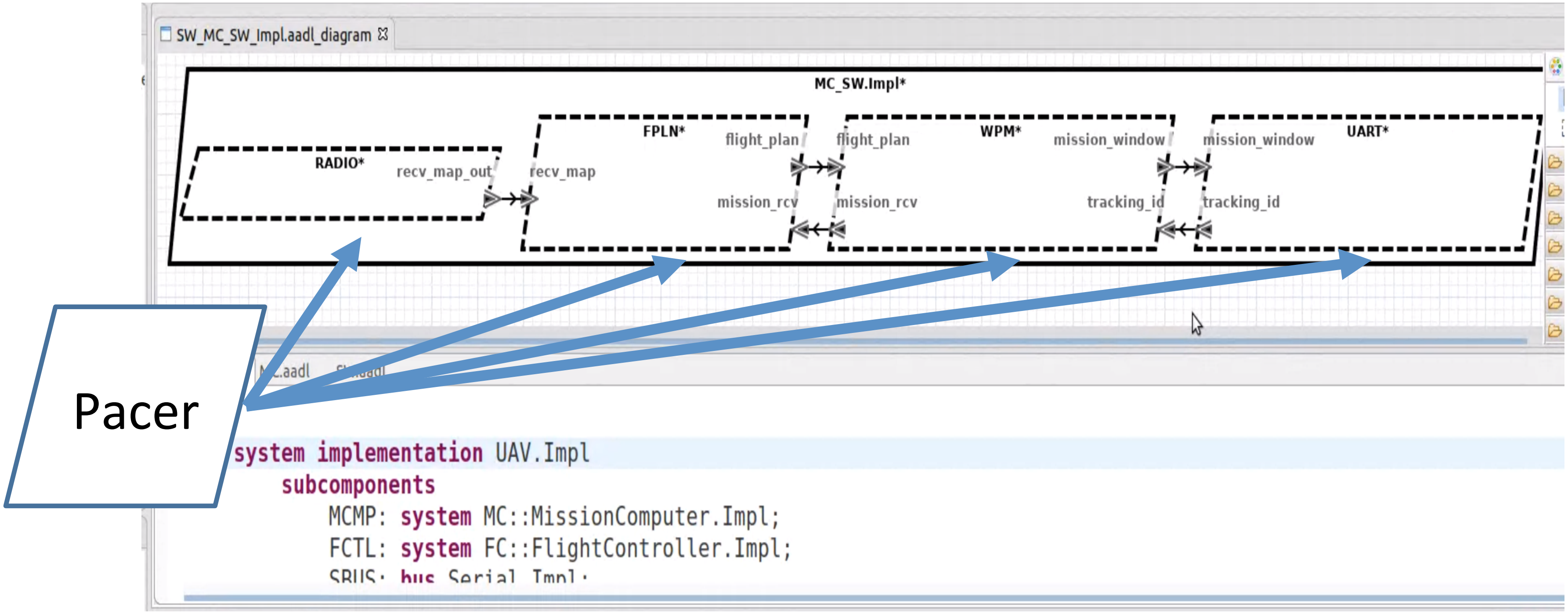
- Feed-forward design
- No temporal partitioning
    - Any component can take down processing chain merely with a busy loop
    - No ability to prevent or mitigate

CASE Temporal Isolation Requirements

- Put each thread in its own seL4 domain
  - Fixed deterministic cyclic schedule
  - Communications happen outside of domain
- Add pacing events to start threads
- Future:
  - Demonstrate UAV comms with data ports instead of event data ports.
  - Permits adding liveness detection, graceful degradation, re-initialization, etc.

Immediate Goal: Use existing seL4 domain schedules.
Long Term Goal: Use MCS version of seL4

CASE Temporal Isolation



```
system implementation UAV.Impl
    subcomponents
        MCMP: system MC::MissionComputer.Impl;
        FCTL: system FC::FlightController.Impl;
        SBUS: bus Serial.Impl;
```

The Pacer component executes in its own domain and enforces strictly periodic thread launches at the start of each domain time slice.

Simple UAV Domain Schedule

In ***domain_schedule.c***

```
const dschedule_t ksDomSchedule[] = {
  { .domain = 0, .length = 100 }, // any other seL4 threads
  { .domain = 1, .length = 50 }, // RADIO 25ms (1 tick ~= 2ms)
  { .domain = 2, .length = 100 }, // FPLN 50ms
  { .domain = 3, .length = 150 }, // WPM 75ms
  { .domain = 4, .length = 80 },  // UART 40ms
  { .domain = 5, .length = 20 },  // pacer 10ms
  };
```

Component Mapping to Domains

in ***PROC_SW.camkes***

```
configuration {
 // [2019/09/20:JCC] Allocation of components to domains
 // Leave any misc seL4 threads in domain 0 by default
 RADIO._domain = 1;
 FPLN._domain = 2;
 WPM._domain = 3;
 UART._domain = 4;
 pacer._domain = 5;

 // Enforce access controls on the sampling port to be one-way
 RADIO.recv_map_access = "W";
 FPLN.recv_map_in_access = "R";
 FPLN.flight_plan_access = "W";
 WPM.flight_plan_in_access = "R";
 WPM.mission_rcv_access = "W";
 FPLN.mission_rcv_in_access = "R";
 WPM.mission_window__access = "W";
 UART.mission_window_in_access = "R";
 UART.tracking_id__access = "W";
 WPM.tracking_id_in_access = "R";
}
```

Thread Requirements

In ***MC.aadl***

```
system implementation MissionComputer.Impl
    subcomponents
      RADIO_HW: device Radio.Impl;
      UART_HW: device UART.Impl;
      PROC_HW: processor MC_Proc.Impl;
      MEM_HW: memory MC_Mem.Impl;
      BUS_HW: bus MC_Bus.Impl;
      PROC_SW: process SW::MC_SW.Impl;
...
    properties
      Scheduling_Protocol => (FixedTimeline) applies to PROC_HW;
      Compute_Execution_Time => 20 ms .. 25 ms applies to PROC_SW.RADIO;
      Compute_Execution_Time => 25 ms .. 50 ms applies to PROC_SW.FPLN;
      Compute_Execution_Time => 25 ms .. 75 ms applies to PROC_SW.WPM;
      Compute_Execution_Time => 40 ms .. 40 ms  applies to PROC_SW.UART;

end MissionComputer.Impl;
```

The Pacer Component

In ***Pacer.camkes:***

```
component Pacer {
    control;
    emits TickTock tick;
    consumes TickTock tock;
    // For this example, all threads run at the
    // same period as the pacer. To
    // To support multi-rate there would
    // multiple periodic events.
    emits Period period;
}
```

The Pacer Behavior

In **Pacer.c:**

```
int run(void) {
  int i = 0;
  while (1) {
    printf("Pacer %d\n", ++i);
    tick_emit();
    period_emit();
    tock_wait();
  }
  return 0;
}
```

Temporal Isolation Status

- Initial demonstration with simple UAV example with data ports, executing on QEMU and XU4.
- Working on integrating trusted build generation within HAMR.
- Working on a before-after demonstration that better shows the security needs for temporal isolation.
- Investigating other possible scheduling paradigms within this same CAmkES domain framework, such as aperiodic scheduling.