seL4 and CAmkES Real Time

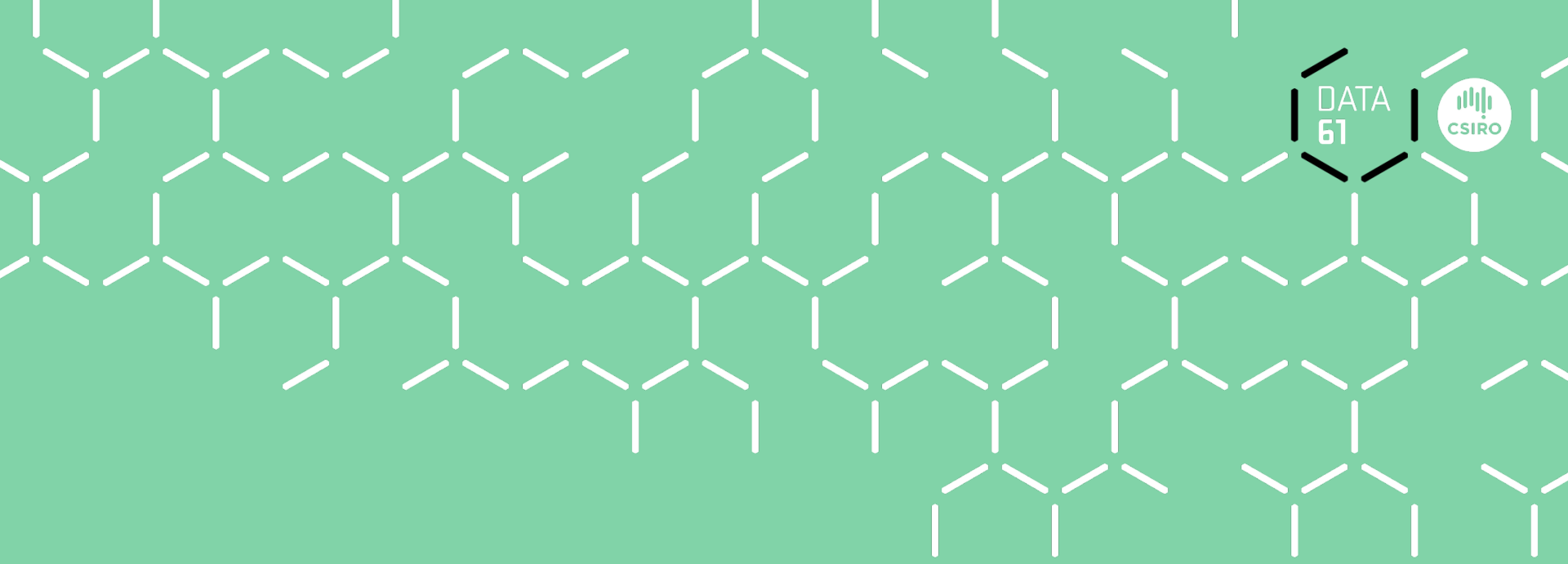
Ihor Kuz, Stephen Sherratt, Hesham Almatary

August 2016

CSIRO

Overview

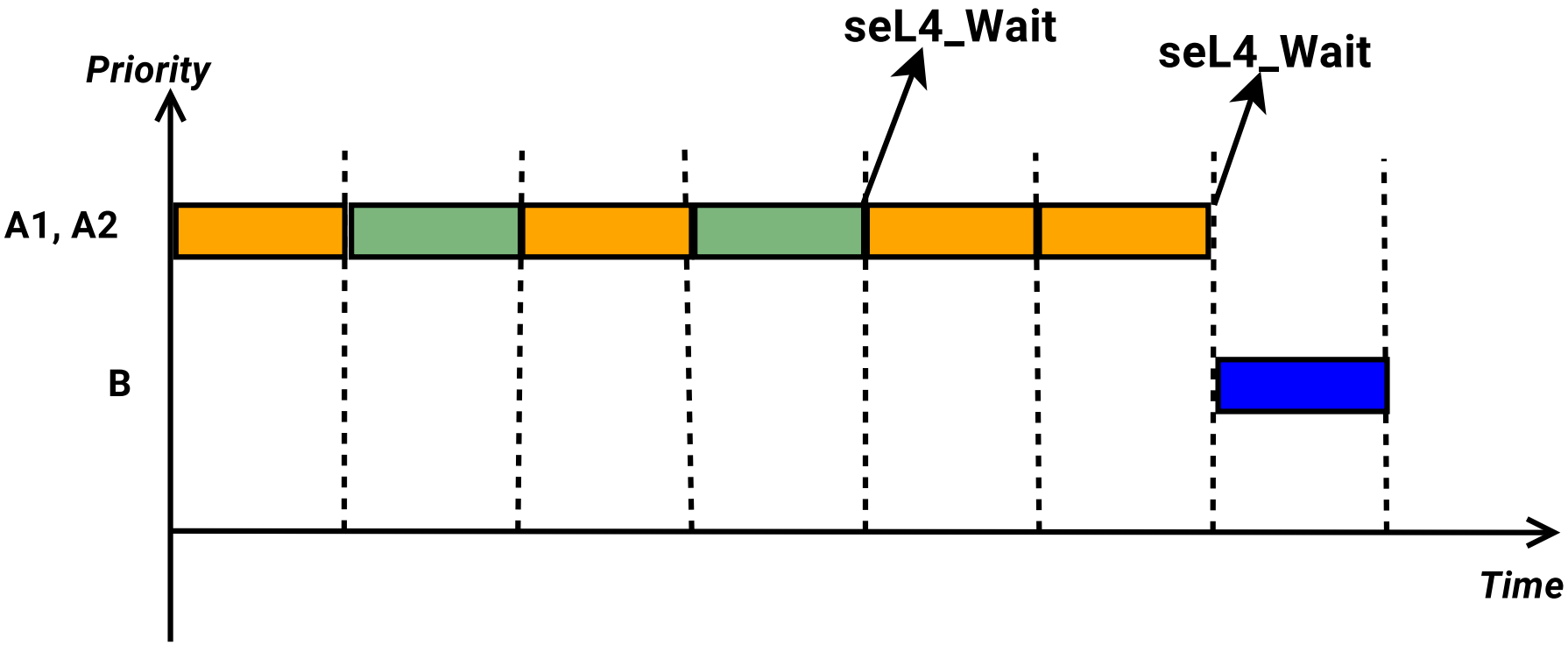
- seL4-RT: Real Time support in seL4
- CAmkES-RT: Real Time support in CAmkES
- Exercises



seL4-RT: Real Time Support in seL4

Legacy seL4 Scheduling Model

- Thread represented by TCB
- TCB has: CSpace, VSpace, IPC buffer
- Preemptive, Round Robin, 256 priority levels



Legacy seL4 Thread Creation

1. Create TCB Object
   - `seL4_Untyped_Retype()`
2. Set Cspace, Vspace, IPC buffer, priority
   - `seL4_TCB_Configure()`
3. Set stack and instruction pointer
   - `seL4_TCB_WriteRegisters()`
4. Activate thread
   - `seL4_TCB_Resume()`

New seL4 Real Time Support

Everything stays the same except…

seL4 Thread Attributes

- CSpace, VSpace
- IPC buffer
- Priority
- Timeslice

seL4-RT Thread Attributes

- CSpace, VSpace
- IPC buffer
- Priority
- Scheduling Context

Not Runnable
if NULL

Scheduling Context

- New kernel-object type
  - Bound to a TCB
  - Specifies how long a thread can be run for
- Consists of a **budget (b)** and a **period (T)** ($b \leq T$)
  - Thread consumes budget and is preempted when it exhausts the budget
  - Budget is refreshed every period
  - *kernel will not permit a thread to run for more than b out of T microseconds*

- (b, T) forms an upper bound of time usage, *not* reservation
  - ... but supports schedulability analysis that can guarantee lower bounds
  - needs to take priorities into account

Schedulability

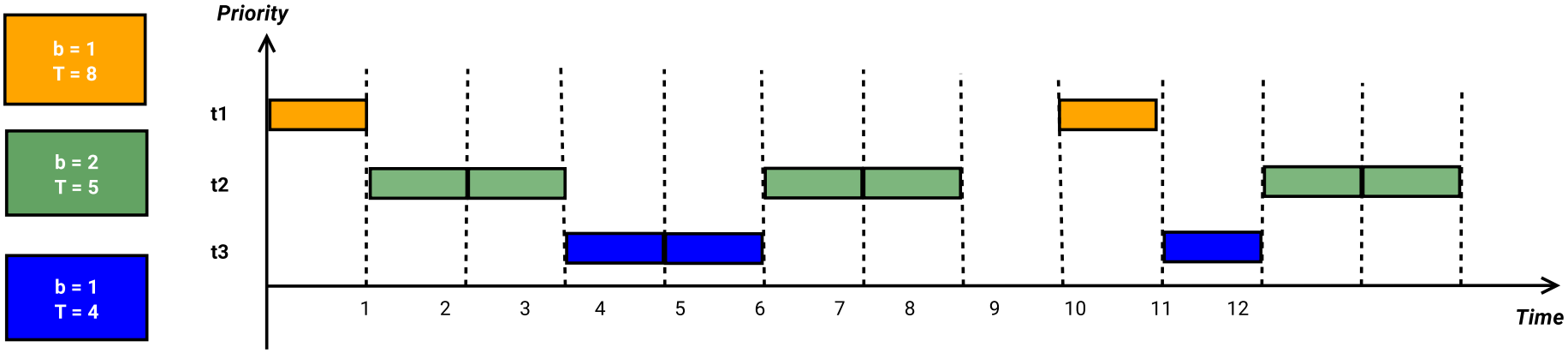
- A thread of priority P will meet its deadline if it passes RMS schedulability test
- Utilisation:

$$u = \sum_{i=1}^{n} \frac{b_i}{T_i} \leq n \times (2^{\frac{1}{n}} - 1)$$

- (where priority of $T_i \leq P$)

It is the user's responsibility to make sure the system is schedulable!

Example: seL4-RT Scheduling



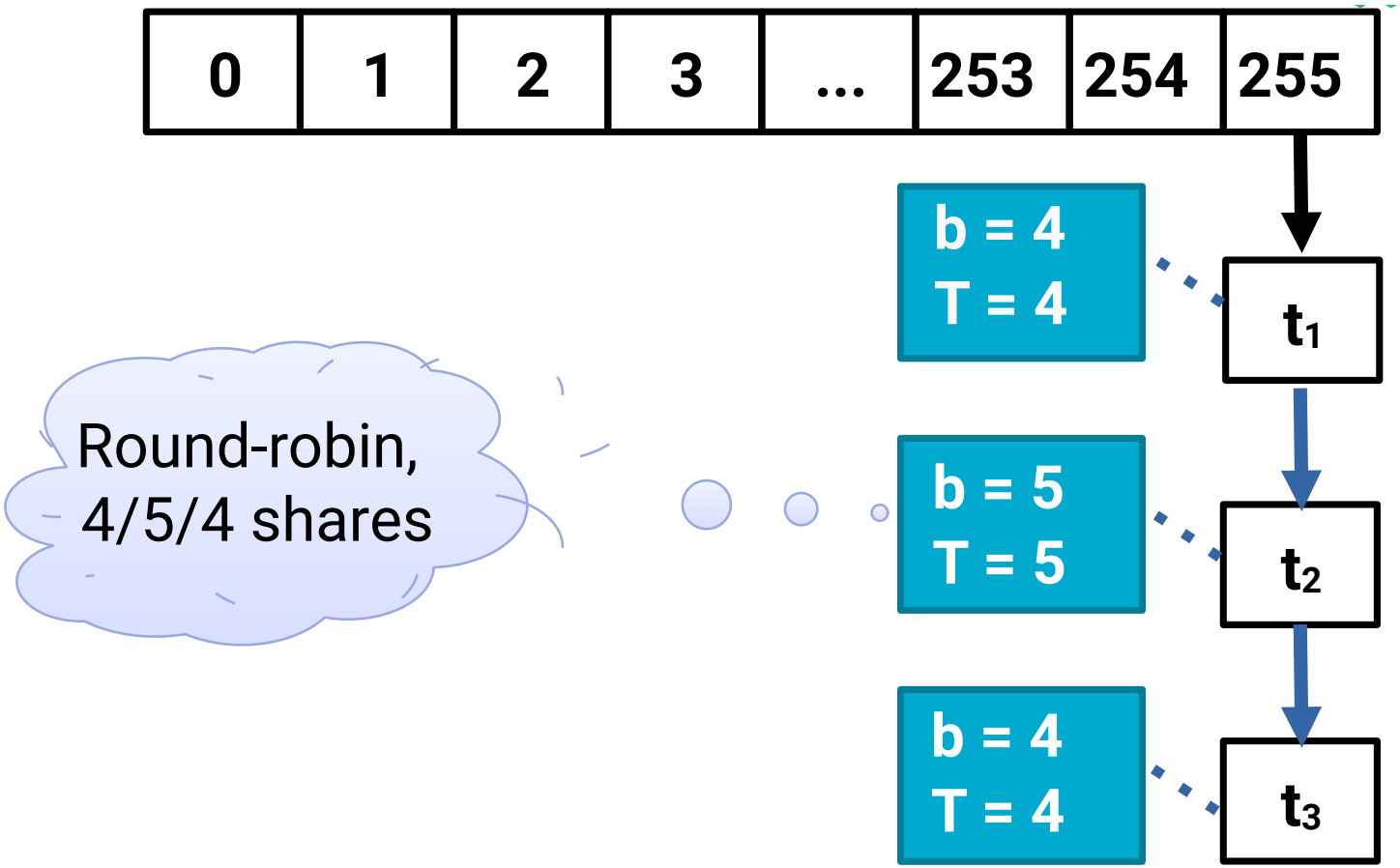
- Utilisation:

$$u = \frac{1}{8} + \frac{2}{5} + \frac{1}{5} = 0.775$$

$$n \times (2^{\frac{1}{n}} - 1) = 3 \times (2^{\frac{1}{3}} - 1) = 0.77976$$

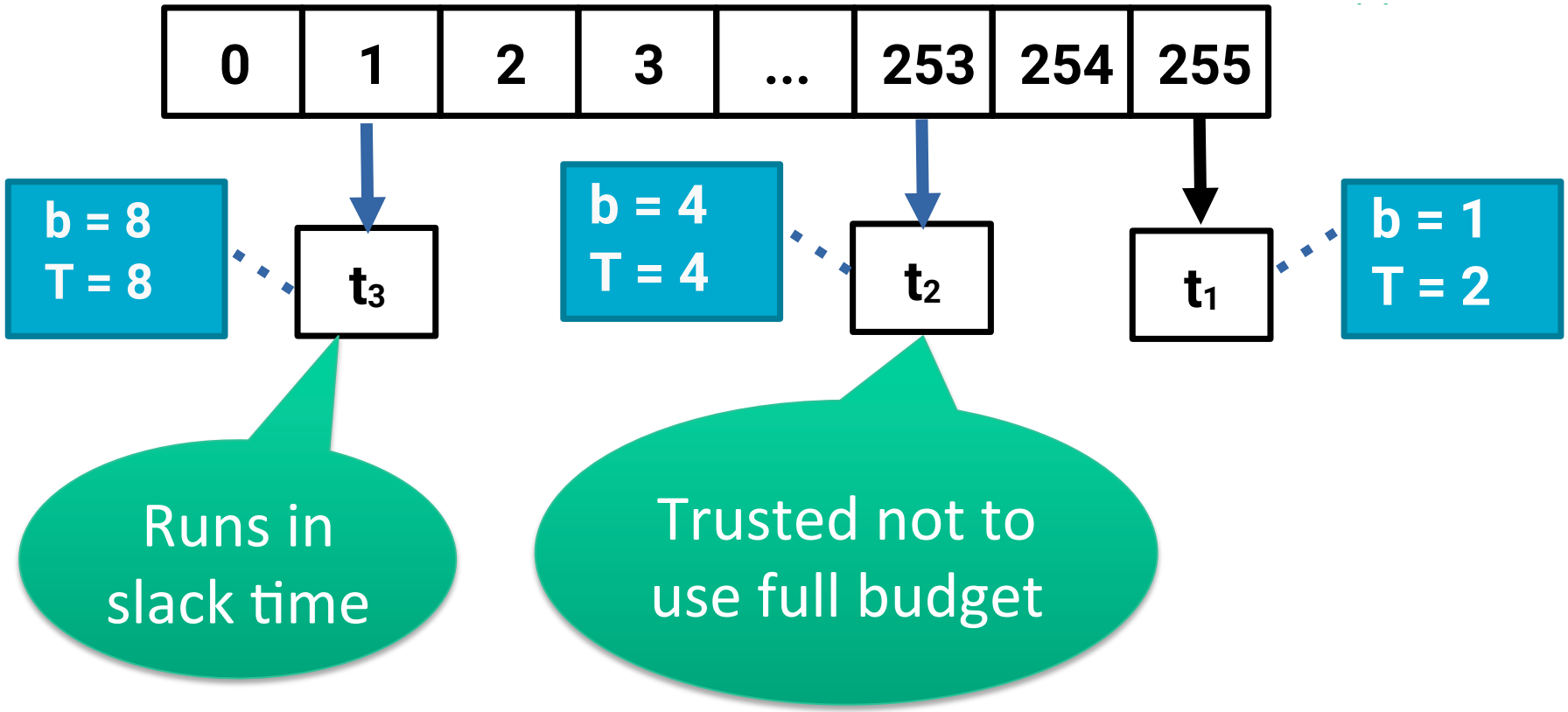
$$0.775 \leq 0.77976$$

Overcommitting: Full Budget



- utilisation: u= 4/4 + 5/5 + 4/4 = 300%
- emulates legacy model where period corresponds to time slice

General Budget



seL4-RT Thread Creation

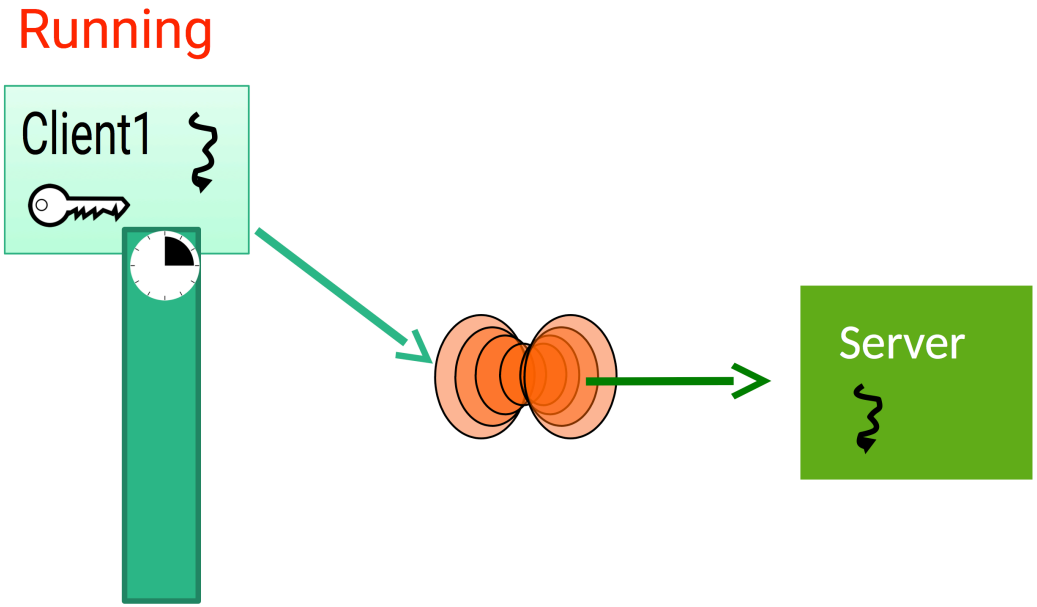
1. Create TCB Object
   - `seL4_Untyped_Retype()`
2. Set Cspace, Vspace, IPC buffer, priority
   - `seL4_TCB_Configure()`
3. Set stack and instruction pointer
   - `seL4_TCB_WriteRegisters()`
4. Create Scheduling Context Object
   - `seL4_Untyped_Retype()`
5. Assign Budget and Period
   - `seL4_SchedControl_Configure()`
6. Bind Thread to Scheduling Context
   - `seL4_SchedContext_bindTCB()`
7. Activate thread
   - `seL4_TCB_Resume()`

Passive Thread

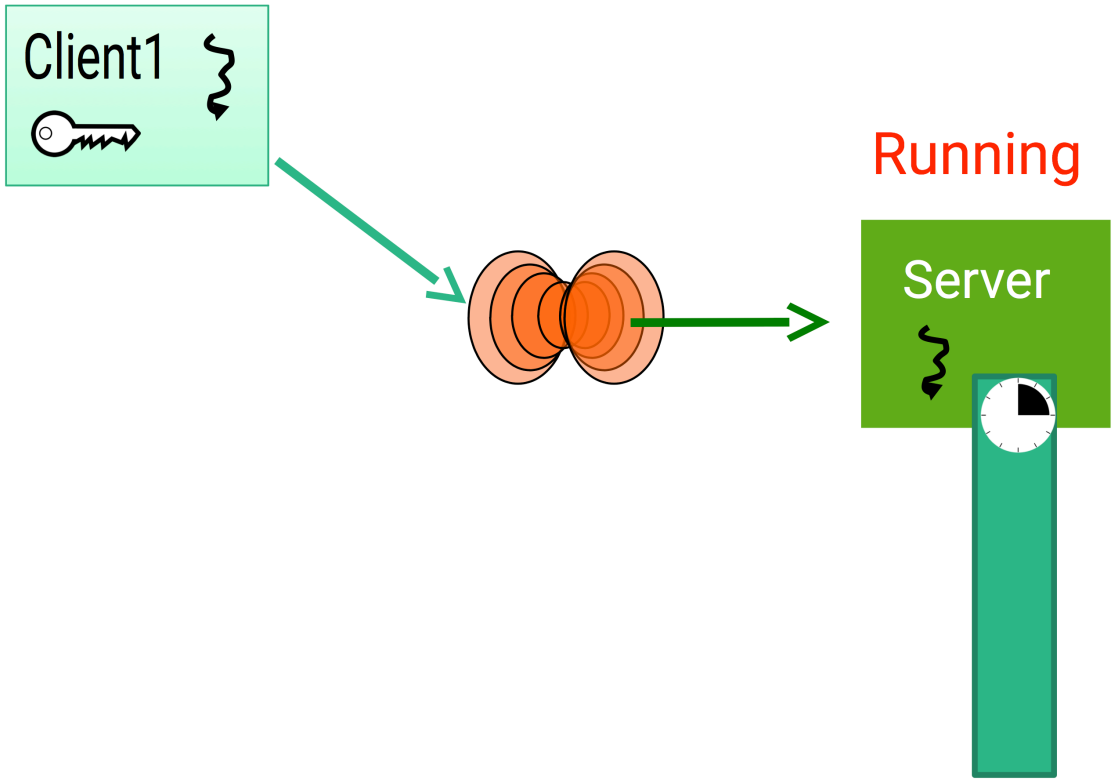
- Thread without a scheduling context
- Use `seL4_SchedContext_UnbindTCB()` to remove scheduling context
- Different from suspending a thread
  - Suspended thread will be run again if it is the next runnable thread
- Cannot execute without the action of another thread
  - To give it a scheduling context again

A passive thread will NOT be schedulable again until it receives a scheduling context

Scheduling Context donation



Scheduling Context donation



Scheduling Context Donation

- Implicit donation via `seL4_Call()` & `seL4_NBSendRecv()`
  - … iff the receiver does not currently have a scheduling context
  - Calling thread's scheduling context will be donated
  - Calling thread will no longer be schedulable
- Scheduling context returned when `seL4_Reply()` or `seL4_ReplyRecv()` is invoked
  - Scheduling context might *NOT* return if passive thread doesn't reply
- Use `seL4_SchedContext_BindNotification()` to donate via notification object

New APIs

- `seL4_SchedControl_Configure()`
  - Invoke SchedControl object to set budget and period of a SchedContext
- `seL4_SchedContext_Yield()`
  - Give up the rest of the budget until the next budget refresh
- `seL4_SchedContext_BindTCB()`
  - Give a TCB a SchedContext
- `seL4_SchedContext_UnbindTCB()`
  - Take a SchedContext away from a TCB
- `seL4_SchedContext_BindNotification()`
  - Allow a SchedContext to be donated via a Notification
- `seL4_SchedContext_UnbindNotification()`
  - Stop donation via a Notification

Warning: Priority Type Change

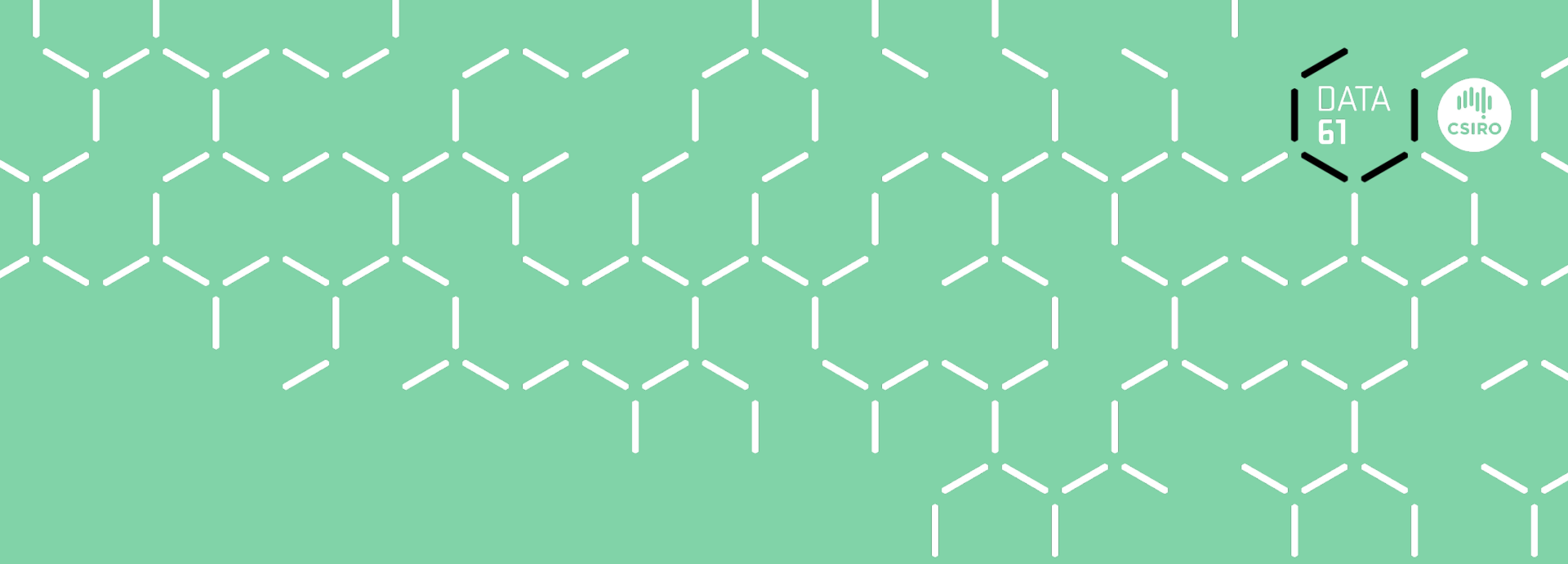
seL4 priority

- `uint8_t priority`
- Thread runs at a fixed priority

seL4-RT priority

- `seL4_Prio_t priority`
  - `uint8_t priority`
  - `uint8_t max_priority`
- Thread can change (own or other's) priority
  - ceiling defined by max_priority value

seL4-RT Exercises



Real Time Support in CAmkES

Threads in CAmkES

- Control thread (`run()` function)
- All interfaces run in separate threads

How CAmkES Supports seL4-RT

New pre-defined attributes

- Scheduling context (sc) attribute
  - By default each thread has its own scheduling context
  - Passive thread: explicitly sets scheduling context to “none”
- Budget attribute
  - Set a thread’s default scheduling context’s budget
- Period attribute
  - Set a thread’s default scheduling context’s period
- Defaults
  - Period = budget = 10,000 (microseconds)

Using Real Time Attributes

Configure scheduling context for:

- All threads in one component
  - $(instance).budget
  - $(instance).period
- Control thread
  - $(instance)._budget
  - $(instance)._period
- Specific interface thread
  - $(instance).$(interface)_budget
  - $(instance).$(interface)_period
- Scheduling context attribute
  - $(instance).$(interface)_sc

```
Component Server {
  provides Simple a;
  provides Simple b;
}
Assembly {
  composition {
     component Server s;
     component Client c;
  }
  configuration {
     c.budget = 1000;
     c._period = 5000;
     s.a_period = 5000;
     s.b_sc = “none”;
  }
}
```

Porting Existing Apps to CAmkES-RT

- All threads in a component instance need a budget and a period
  - except passive components.
  - The default budget and period are 10000 µs
- Passive component only make sense for the "receiving" side of RPC or Event interface
  - Don’t make control thread passive!
- Always use interface-specific attribute to configure passive component
- Passive component only works with seL4RPCCall and seL4Notification connectors
  - Will do scheduling context donation

CAmkES-RT Exercises: server-1

- Get RT tutorial exercises
  - `mkdir sel4-rt-tutorials; cd sel4-rt-tutorials`
  - `repo init -u https://github.com/sel4proj/sel4-tutorials-manifest.git -m sel4-rt-tutorials.xml`
  - `repo sync`
- Server-1
  - Goal: create and initialise a passive thread, do scheduling context donation
- TODOs:
  - TODO 1: initialise the new TCB
  - TODO 2: Create a cap to store a sched context
  - TODO 3: initialise a scheduling context
  - TODO 4: Bind the scheduling context
  - TODO 5: Make this server passive
  - TODO 6: Get signal from server saying it initialised
  - TODO 7: Unbind the server's scheduling context

CAmkES-RT Exercises: server-2

- Server-2
  - Goal: experiment with budget and period settings for an active thread
- TODOs
  - TODO_4: Play around with the parameters and run the app. Try and see if you can explain what happens

- CAmkES RT Exercises
  - camkes-rt-1
    - Make a CAmkES app that does what server-1 does
  - camkes-rt-2
    - Make a CAmkES app that does what server-2 does

More info

- http://sel4.systems/Info/Docs/seL4-manual-1.0.0-rt-dev.pdf
  - Documentation of seL4-RT kernel
- https://github.com/seL4/seL4/tree/rt
  - “rt” branch of kernel
- https://github.com/seL4/sel4test/tree/rt
  - “rt” branch of sel4test
- https://github.com/seL4/camkes-tool
  - CAmkES-RT support is in camkes master branch
- https://github.com/seL4-projects/camkes-sc-tests
  - Examples and tests for CAmkES-RT