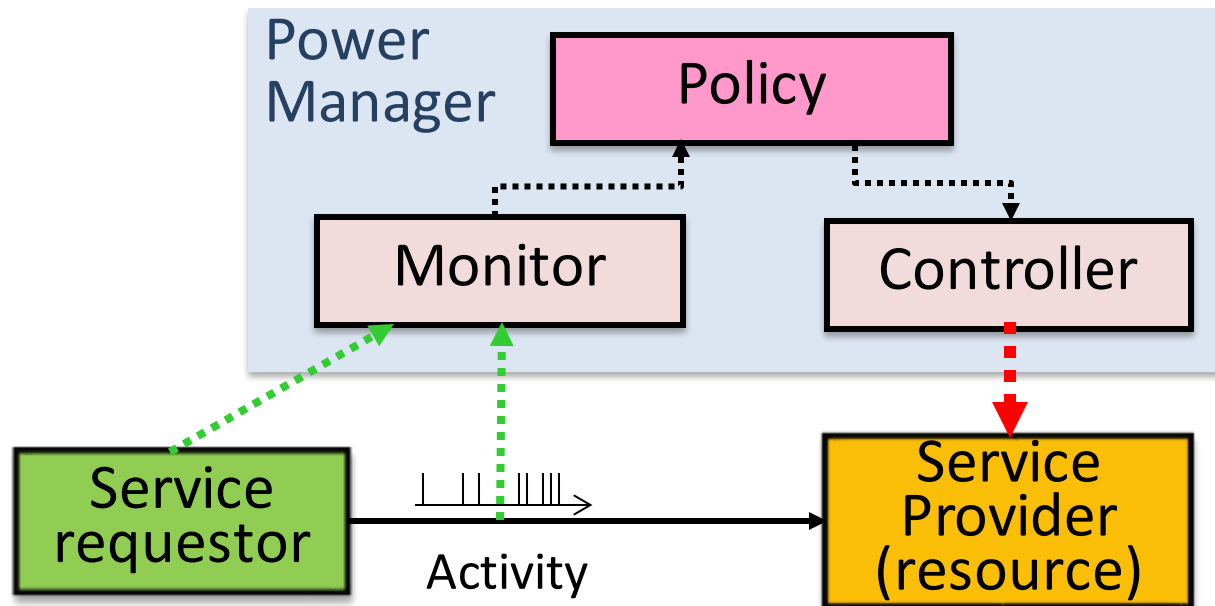


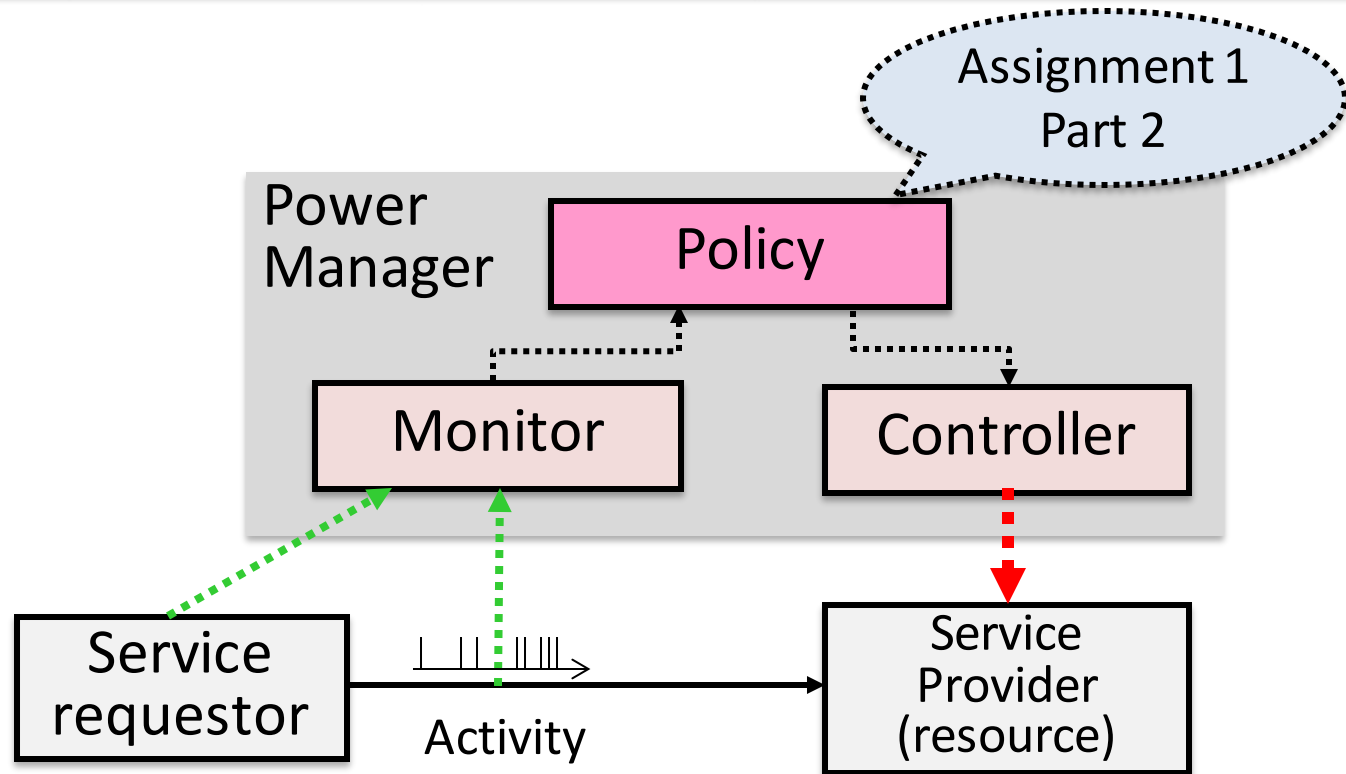
Lab 1 – Day 2
Dynamic Power Management

Recall



- Power manager (PM)
 - Monitors requestor's activity and sets state of provider according to some **policy** (implemented inside the PM)

Recall

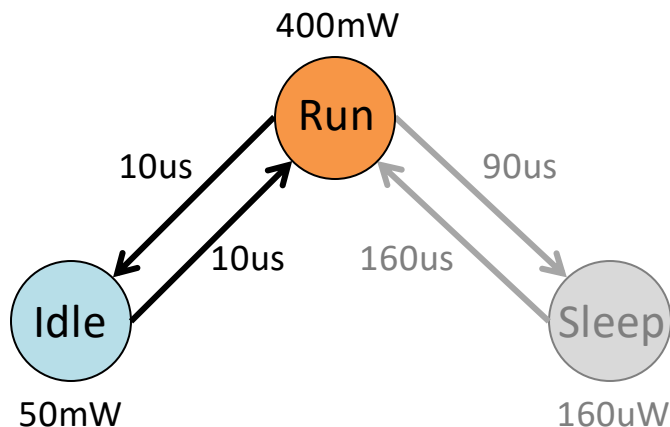


- Policy implementation
 - Extend the timeout policy

Assignment 1 – Part 2
**Extension of the timeout
policy**

DPM simulator

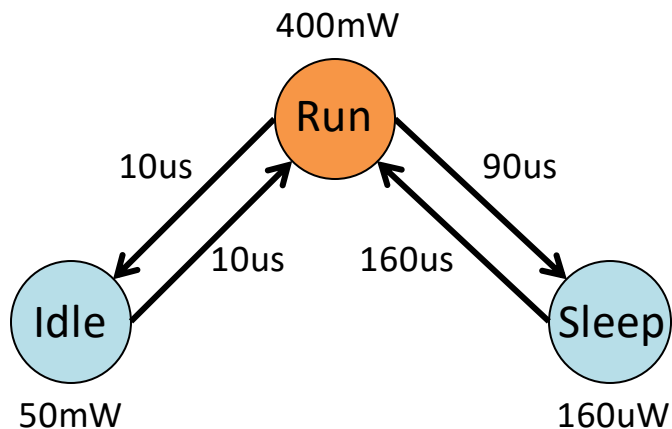
- The DPM simulator supports a timeout policy with transitions from ACTIVE to IDLE
 - Never goes to SLEEP state
 - Moving to SLEEP may save even more energy...



[sim] Active time in profile = 0.300130s
[sim] Idle time in profile = 0.244066s
[sim] Total time = 0.544196s
[sim] Timeout waiting time = 0.024679s
[sim] Total time in state Run = 0.324809s
[sim] Total time in state Idle = 0.219387s
[sim] Total time in state Sleep = 0.000000s
[sim] Time overhead for transition = 0.022770s
[sim] N. of transitions = 2277
[sim] Energy for transitions = 0.022770J
[sim] Energy w/o DPM = 0.217678J, Energy w DPM = 0.163663J
[sim] 24.8 percent of energy saved.

Assignment 1 – Part 2

- Modify the timeout policy to enable transitions also to SLEEP
 - Must modify the implementation of the DPM simulator

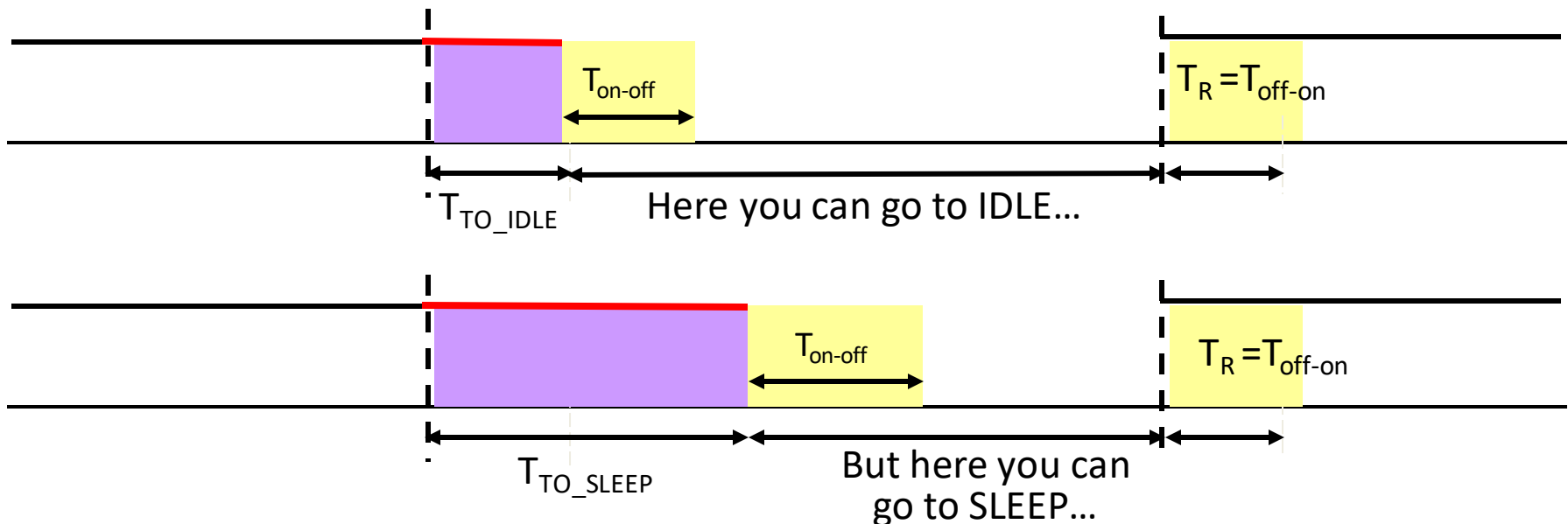
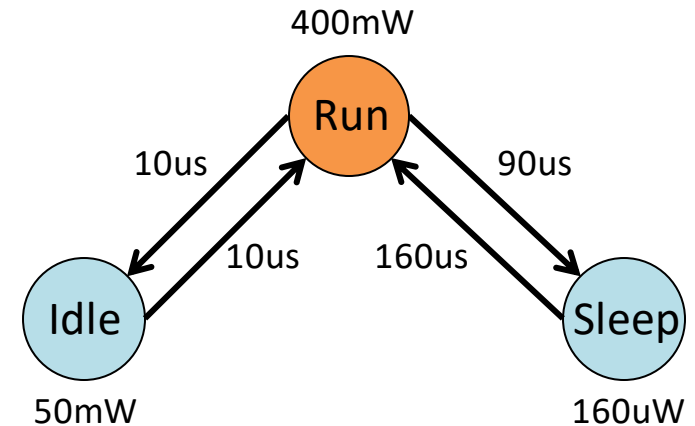


```
switch (policy) {
```

```
    case DPM_TIMEOUT:
        if(curr_time > idle_period.start +
            tparams.timeout[0]) {
            *next_state = PSM_STATE_IDLE;
        } else {
            *next_state = PSM_STATE_ACTIVE;
        }
        /* LAB 2 EDIT */
        break;
```

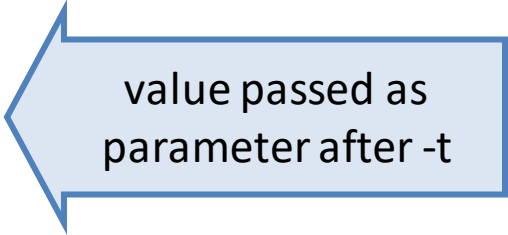
Assignment 1 – Part 2

- Use two timeouts...
 - Note that there is no explicit transition between IDLE and SLEEP!
 - Add the transition in the PSM



The DPM simulator

- `print_command_line()` (src/utilities.c)
 - Prints the command line to invoke the tool
- `parse_arg()` (src/utilities.c)
 - Parses the inputs you provide via command line
 - For the timeout policy:
 - *selected_policy = DPM_TIMEOUT;
 - tparams->timeout[0] = atof(argv[++cur]);
 - May have to be extended to take 2 timeouts in input (one for IDLE and one for SLEEP)



value passed as
parameter after -t

The DPM simulator

- `dpm_decide_state()` (src/dpm_policies.c)
 - Provided system state
 - Workload status (active/idle)
 - Current simulation time
 - Adopted policy (i.e., timeout)
 - Determines what is the state at the next time step

The DPM simulator

- Extend TIMEOUT case to support also transition to sleep

```
switch (policy) {
```

```
    case DPM_TIMEOUT:
```

```
        if(curr_time > idle_period.start + tparams.timeout[0]) {
```

```
            *next_state = PSM_STATE_IDLE;
```

```
        } else {
```

```
            *next_state = PSM_STATE_ACTIVE;
```

```
        }
```

```
        /* LAB 2 EDIT */
```

```
        break;
```

But.. What about SLEEP?

Wait for the timeout and then go to IDLE

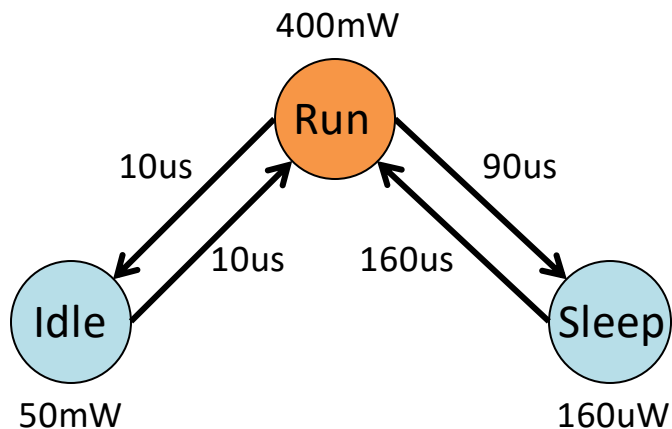
The DPM simulator

- `dpm_simulate()` (src/dpm_policies.c)
 - Emulates policy application to the PSM given the workload
 - For all instants in the current (active and idle) period
 - Invoke `dpm_decide_state()` to apply the policy and determine transitions
 - Check the state returned and update energy consumption accordingly

It emulates the execution of the CPU againts the WorkLoad provided. It computes the energy consumption and prints the results.

In synthesis...

- Update the `parse_arg()` and `dpm_decide_state()` functions to support transitions also to SLEEP
 - Need an appropriate timeout for SLEEP...



```
switch (policy) {
```

```
    case DPM_TIMEOUT:
        if(curr_time > idle_period.start +
           tparams.timeout[0]) {
            *next_state = PSM_STATE_IDLE;
        } else {
            *next_state = PSM_STATE_ACTIVE;
        }
        /* LAB 2 EDIT */
        break;
```

Assignment 1 – Part 2

- Report assignment:

Compare for different workload the two different implementations of the simulation (the original and the modified one)

- Comparison between example timeout and extended timeout policies

- What changes with IDLE→SLEEP?

- Timeout vs. energy saving

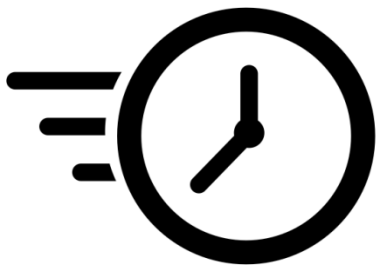
- How does energy saving change w.r.t.

- The workload distribution?

- The adopted timeouts?

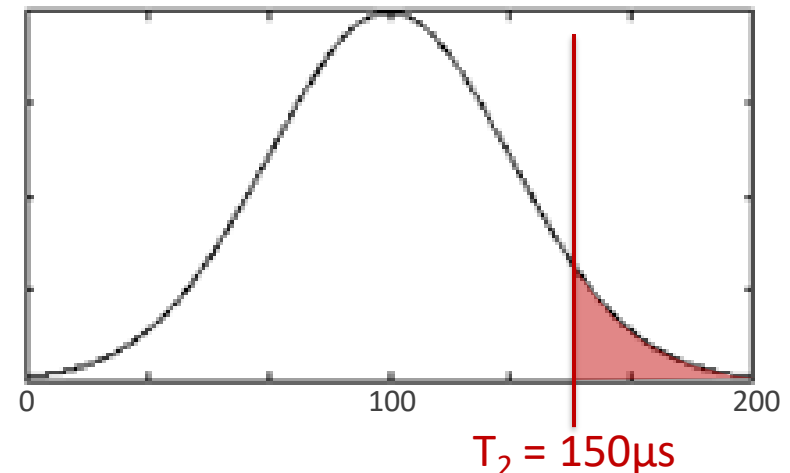
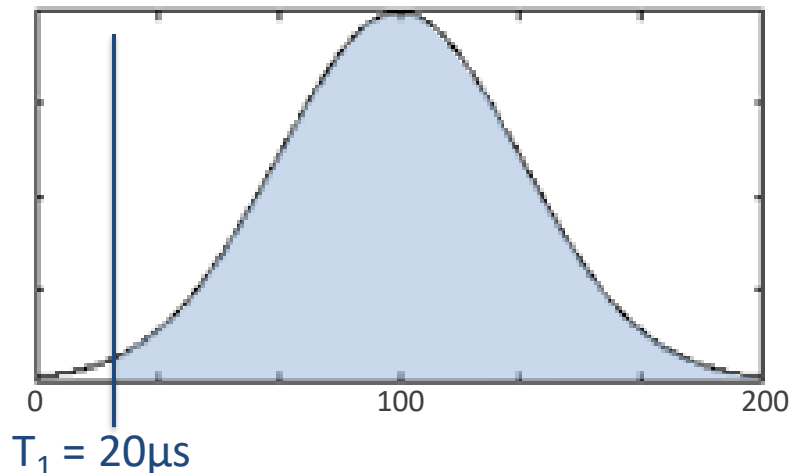
- » E.g., timeout is exactly T_{BE} , lower than T_{BE} , higher than T_{BE}

- Timing overheads to perform transitions?



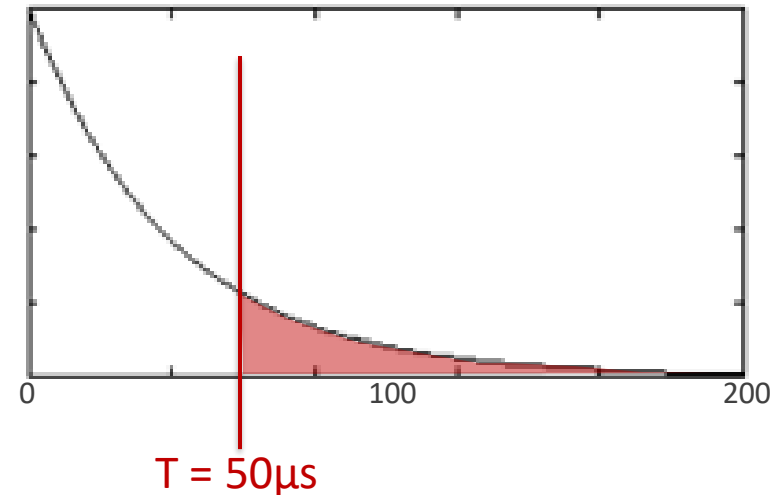
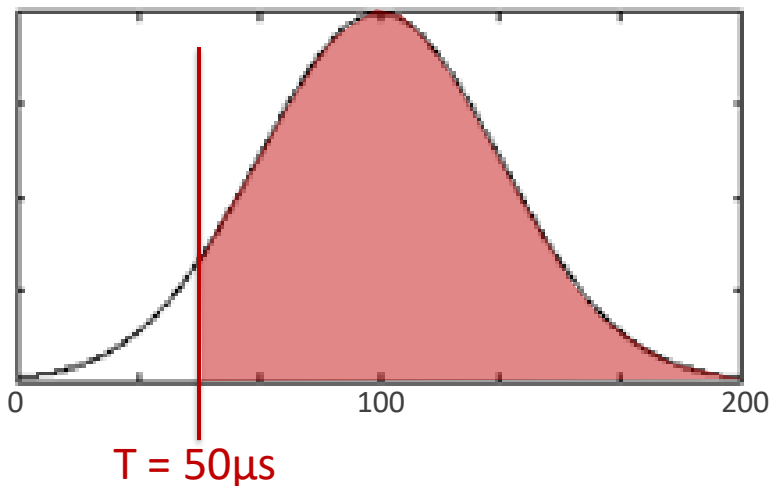
Assignment 1 – Part 2

- Example: how does energy saving change w.r.t. the adopted timeout?
 - Remember: distribution histogram
→ per each time value, the number of samples that have that value
 - How many idle periods will last longer/shorter?



Assignment 1 – Part 2

- Example: how does energy saving change w.r.t. the workload distribution?
 - How many idle periods will last longer/shorter?
 - Normal distribution and exponential distribution have very different behaviors...



Assignment 1 – Part 2

- Example:
 - Why do I get these power savings?
 - Why is policy 2 worse than policy 1 on distribution 4?
 - Why is it better on distribution 2?
 - Why is policy 1 better on distrib. 2 than on distrib. 5?
 - Why is sometimes the power saving negative?
 - ...?

The one that has

	POLICY 1	POLICY 2
DISTRIB 1	0%	0%
DISTRIB 2	25%	27%
DISTRIB 3	0%	-2%
DISTRIB 4	10%	9%
DISTRIB 5	5%	11%

