3.2.3. Costs & energy

Embedded systems costs

- When building an embedded system for IoT, the two main costs to keep in mind are **energy** and **money**, which have strong implications in the embedded system design.
- Energy:
  - embedded system have an expected workload - want to minimise energy required to handle the workload (longer battery life, more efficient)
  - More powerful highly featured MCPs draw more power - can be more *power efficient*, but consume more energy for a given task.
- Money:
  - Many embedded systems makers have tight margins (e.g., cars)
  - More powerful, highly featured MCUs cost more
  - want to use devices that do just exactly what they have to, especially when putting hundreds of them in a particular device.

Embedded systems costs

- When building an embedded system for IoT, the two main costs to keep in mind are **energy** and **money**, which have strong implications in the embedded system design.
- Energy:
  - embedded system h want to minimise energy required to handle the more efficient)
  - More powerful highly power - can be more *power efficient*, but co en task.

TAKEAWAYS:
1- pick the minimal MCU that can meet your application requirements
2- optimise the code for it

- Money:
  - Many embedded systems makers have tight margins (e.g., cars)
  - More powerful, highly featured MCUs cost more
  - want to use devices that do just exactly what they have to, especially when putting hundreds of them in a particular device.

Cost examples

Part	Family	Flash	RAM	Cost	Notes
ATSAMD20E15A	Cortex-M0+	16kB	2kB	$1.37	
ATSAMD21E16B	Cortex-M0+	64kB	8kB	$1.70	LIN, USB
NRF51422	Cortex-M0	256kB	32kB	$2.44	BLE
ATSAMD20J18A	Cortex-M0+	256kB	32kB	$2.69	20 12-bit ADC
ATSAM4S2BA	Cortex-M4	128kB	64kB	$2.80	SSC, USB
ATSAMD21G18A	Cortex-M0+	256kB	32kB	$3.15	LIN, USB
ATSAM4E8EA	Cortex-M4	512kB	128kB	$7.62	CAN, Ethernet, USB, IrDA

- Generally speaking, as flash memory and RAM go up, so does cost.

Computing Energy Budget

- Many embedded systems are battery-powered and so you want to minimise the amount of energy consumed
  - make a device last longer
  - requiring fewer recharges

- **Computing the energy budget allows you to reason about the design trade-offs.**
  - is it worth to make the batter slightly larger on your fitness tracker?
  - Can we optimise software to achieve less recharges?

High level energy consumption

- Roughly speaking, the energy consumed by a device is the energy consumed while they are sleeping and the energy consumed while they are active

$$E = P_s \cdot t_s + P_a \cdot t_a$$

E : energy

$P_{s/a}$: power in sleep/active more

$t_{s/a}$: time in sleep/active mode

- In practice it is more complicated, because there could be many terms and many different active modes (radio on/off, processor speed, etc).

Energy budget: example

- Consider a NRF51422 SoC that wakes up at 1Hz and transmits a single BLE advertisement.
  - $P_s = 4.8\mu$A
  - $P_a = 14.6$mA
  - time for sending advertisement: 0.3ms

$$E = P_s \cdot t_s + P_a \cdot t_a$$

Energy budget: example

- Consider a NRF51422 SoC that wakes up at 1Hz and transmits a single BLE advertisement.
  - $P_s = 4.8\mu$A
  - $P_a = 14.6$mA
  - time for sending advertisement: 0.3ms

$$E = P_s \cdot t_s + P_a \cdot t_a$$
$$= 4.8\mu A \cdot 999.7ms + 14.6mA \cdot 0.3ms$$

Energy budget: example

- Consider a NRF51422 SoC that wakes up at 1Hz and transmits a single BLE advertisement.
  - $P_s = 4.8\mu$A
  - $P_a = 14.6$mA
  - time for sending advertisement: 0.3ms

$$E = P_s \cdot t_s + P_a \cdot t_a$$
$$= 4.8\mu A \cdot 999.7ms + 14.6mA \cdot 0.3ms$$
$$= (4798.56\mu A + 4.38mA) \cdot s$$

Energy budget: example

- Consider a NRF51422 SoC that wakes up at 1Hz and transmits a single BLE advertisement.
  - $P_s = 4.8\mu$A
  - $P_a = 14.6$mA
  - time for sending advertisement: 0.3ms

$$
\begin{aligned}
E &= P_s \cdot t_s + P_a \cdot t_a \\
&= 4.8\mu A \cdot 999.7ms + 14.6mA \cdot 0.3ms \\
&= (4798.56\mu A + 4.38mA) \cdot s \\
&= (4.799mA + 4.38mA) \cdot s = 9.179mA \cdot s
\end{aligned}
$$

Energy budget: example

- Consider a NRF51422 SoC that wakes up at 1Hz and transmits a single BLE advertisement.
  - $P_s = 4.8\mu$A
  - $P_a = 14.6$mA
  - time for sending advertisement: 0.3ms

$$
\begin{aligned}
E &= P_s \cdot t_s + P_a \cdot t_a \\
&= 4.8\mu A \cdot 999.7ms + 14.6mA \cdot 0.3ms \\
&= (4798.56\mu A + 4.38mA) \cdot s \\
&= (\boxed{4.799mA} + \boxed{4.38mA}) \cdot s = 9.179mA \cdot s
\end{aligned}
$$

The radio is active 0.03% of the time but consumes as much as it does the 99.97% of the time when asleep

Energy budget: complication

- We assumed that system instantaneously transitioned from asleep to wake status.
  - MCU takes time to wake up
  - transceiver takes time to power up

- Transition times - high power but no work - can be significant
  - solution: wake up less often to amortise over wake periods

$$E = P_s \cdot t_s + P_a \cdot t_a + P_T \cdot t_T$$

P_T: power during transition

t_T : time to transition

Energy budget: example ++

- NRF51422 SoC waking up at 1Hz and transmits a single BLE advertisement
  - $P_s = 4.8\mu$A = 0.0048mA
  - $P_a = 14.6$mA
  - time for sending advertisement: 0.3ms
  - $P_T = 7.0$mA
  - $t_T = 0.14$ms

$$E = P_s \cdot t_s + P_a \cdot t_a + P_T \cdot t_T$$

Energy budget: example ++

- NRF51422 SoC waking up at 1Hz and transmits a single BLE advertisement
  - $P_s = 4.8\mu$A = 0.0048mA
  - $P_a = 14.6$mA
  - time for sending advertisement: 0.3ms
  - $P_T = 7.0$mA
  - $t_T = 0.14$ms

$$
\begin{aligned}
E &= P_s \cdot t_s + P_a \cdot t_a + P_T \cdot t_T \\
&= 0.0048mA \cdot 0.99956s + 14.6mA \cdot 0.0003s + 7.0mA \cdot 0.00014s
\end{aligned}
$$

Energy budget: example ++

- NRF51422 SoC waking up at 1Hz and transmits a single BLE advertisement
  - $P_s = 4.8\mu$A = 0.0048mA
  - $P_a = 14.6$mA
  - time for sending advertisement: 0.3ms
  - $P_T = 7.0$mA
  - $t_T = 0.14$ms

$$
\begin{aligned}
E &= P_s \cdot t_s + P_a \cdot t_a + P_T \cdot t_T \\
&= 0.0048mA \cdot 0.99956s + 14.6mA \cdot 0.0003s + 7.0mA \cdot 0.00014s \\
&= (0.004798mA + 0.00438mA + 0.00098mA) \cdot s
\end{aligned}
$$

Energy budget: example ++

- NRF51422 SoC waking up at 1Hz and transmits a single BLE advertisement
  - $P_s = 4.8\mu$A = 0.0048mA
  - $P_a = 14.6$mA
  - time for sending advertisement: 0.3ms
  - $P_T = 7.0$mA
  - $t_T = 0.14$ms

Roughly 10% of energy goes to transitioning from sleep to active mode

$$E = P_s \cdot t_s + P_a \cdot t_a + P_T \cdot t_T$$

$$= 0.0048mA \cdot 0.99956s + 14.6mA \cdot 0.0003s + 7.0mA \cdot 0.00014s$$

$$= (0.004798mA + 0.00438mA + \boxed{0.00098mA)} \cdot s$$

Minimising energy consumption

- To minimise **sleep energy**, put microcontroller into lowest possible state
  - in the previous example we only saw “sleep” mode, but microcontrollers have different many low-power states and power-saving features.
  - the state the microcontroller is in has complex implications to software

- To minimise **active energy**, minimise time peripherals and MCU are active
  - perform operations in parallel to minimise active time
  - cluster/batch operations to minimise transition times (e.g., instead of sending one packet every second you can send 10 packets every ten seconds).
  - minimize clock rate.

Sleep states: SAM4L running modes

Mode	Current	Wakeup Latency

Sleep states: SAM4L running modes

- Four basic running modes:

Mode	Current	Wakeup Latency

Sleep states: SAM4L running modes

- Four basic running modes:
  1. RUN - MCU executes instructions, everything can be active

Mode	Current	Wakeup Latency
Run @48MHz	14.5 mA	-

Sleep states: SAM4L running modes

- Four basic running modes:
  1. RUN - MCU executes instructions, everything can be active
  2. SLEEP - MCU runs no instructions, clock/peripherals can be active

Mode	Current	Wakeup Latency
Run @48MHz	14.5 mA	-
Sleep	50μA	0.25μs

Sleep states: SAM4L running modes

- Four basic running modes:
  1. RUN - MCU executes instructions, everything can be active
  2. SLEEP - MCU runs no instructions, clock/peripherals can be active
  3. WAIT - no instructions, only the 32KHz clock active for peripherals (slowest rate clock, lowest power)

Mode	Current	Wakeup Latency
Run @48MHz	14.5 mA	-
Sleep	50μA	0.25μs
Wait	6μA	1.5μs

Sleep states: SAM4L running modes

- Four basic running modes:
  1. RUN - MCU executes instructions, everything can be active
  2. SLEEP - MCU runs no instructions, clock/peripherals can be active
  3. WAIT - no instructions, only the 32KHz clock active for peripherals (slowest rate clock, lowest power)
  4. RETENTION - no instruction, only 32KHz clock, no active peripherals (can be waken by external interrupts)

Mode	Current	Wakeup Latency
Run @48MHz	14.5 mA	-
Sleep	$50\mu A$	$0.25\mu s$
Wait	$6\mu A$	$1.5\mu s$
Retention	$3\mu A$	$1.5\mu s$

Active state: parallelism

- Parallelism is one technique to enhance efficiency in active mode

Sequential

```
loop {
    sample_sensor();
    radio_on();
    send_value();
    sleep();
}
```

Time: sum of time of each operation

Parallel

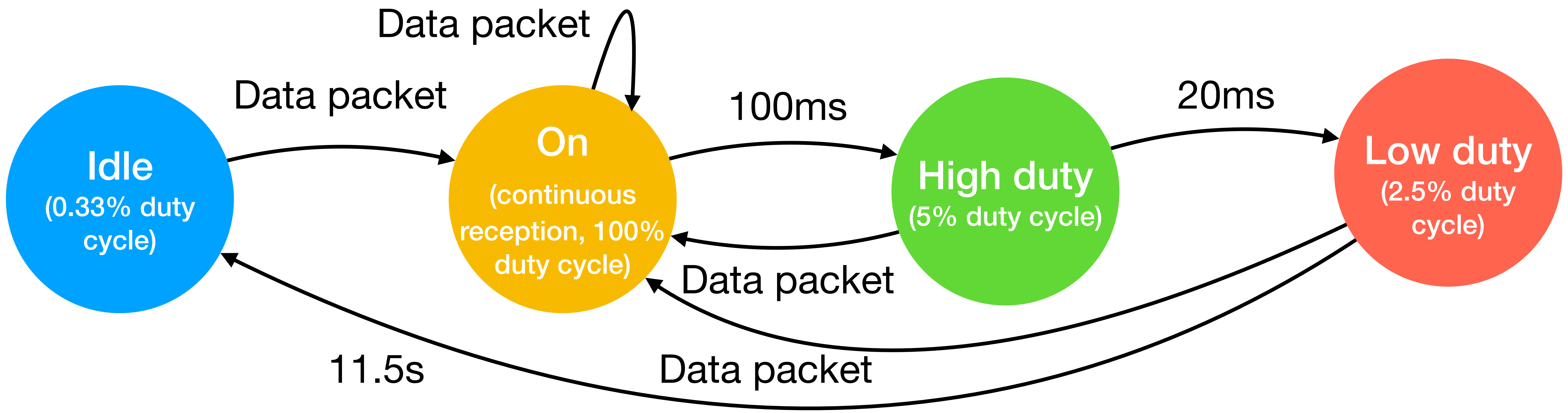
```
loop {
    parallel {
        sample_sensor();
        radio_on();
    }
    send_value();
    sleep();
}
```

Time: max(time for sample sensors and turning radio on) + time for sending values and enter sleep mode

Active state: Batching (1)

- If your system has very high transition costs (probably because it takes long to go from sleep to active mode), you can improve the efficiency of your system by **batching a whole bunch of operations** to amortise transition costs between them.

- Example: IoT system with LTE (Long Term Evolution) radio - often go through a variety of states depending on the network traffic they experience.



Duty cycle: % of time over a given period when the device is active

Active state: Batching (2)

- Assume to have an IoT device with an LTE radio. Assume time for sending a small packet is 5ms, i.e., 0.005s. The devices senses every 30 seconds and can decide whether to transmit the packet with data every 30s or transmit two packets every 60s.

- Sending one packet every 30 s:

$$0.105s \cdot 100\% + 0.02s \cdot 5\% + 11.5s \cdot 2.5\% + (30 - 11.625)s \cdot 0.33\,\%$$
$$\sim (0.105 + 0.001 + 0.2875 + 0.06)s/30s = 0.4535s/30s \sim 1.5\,\%$$

- Sending two packets every 60 s:

$$\sim 0.110s \cdot 100\% + 0.02s \cdot 5\% + 11.5s \cdot 2.5\% + (60 - 11.63)s \cdot 0.33\,\%$$
$$\sim (0.11 + 0.001 + 0.2875 + 0.16)s/60s = 0.5585s/60s \sim 0.93\,\%$$

Active state: Batching (2)

- Assume to have an IoT device with an LTE radio. Assume time for sending a small packet is 5ms, i.e., 0.005s. The devices senses every 30 seconds and can decide whether to transmit the packet with data every 30s or transmit two packets every 60s.

- Sending one packet every 30 s:

$$0.105s \cdot 100\% + 0.02s \cdot 5\% + 11.5s \cdot 2.5\% + (30 - 11.625)s \cdot 0.33\,\%$$
$$\sim (0.105 + 0.001 + 0.2875 + 0.06)s/30s = 0.4535s/30s \sim \boxed{1.5\,\%}$$

- Sending two packets every 60 s:

$$\sim 0.110s \cdot 100\% + 0.02s \cdot 5\% + 11.5s \cdot 2.5\% + (60 - 11.63)s \cdot 0.33\,\%$$
$$\sim (0.11 + 0.001 + 0.2875 + 0.16)s/60s = 0.5585s/60s \sim \boxed{0.93\,\%}$$

About 38% less time spent active