Lecture 26 — Interval & Watchdog Timers

Patrick Lam & Jeff Zarnett
p.lam@ece.uwaterloo.ca & jzarnett@uwaterloo.ca

Department of Electrical and Computer Engineering
University of Waterloo

December 29, 2014

Concept: Timers in Embedded Systems

Today: How to use timers,
e.g. in embedded systems:

- Interval Timers
- Watchdog Timers

We'll use Java timers to implement these, and learn the difference between Android and Java timers.

Applications of Timers

Interval Timers—perform a task occasionally.

- (raising alarms; polling).

Watchdog Timers—reset a stuck system.

Part I

Interval Timers

Interval Timers

Good for recurring tasks.

Example: Light sensor polling
(but not on Android—receive sensor events instead).

Previously: `Handler` for interval timers.

Handlers for Infinitely-Recurring Interval Timers

```
Handler h = new Handler();
Runnable r = new Runnable() {
  public void run() {
    // execute the task
    h.postDelayed(this, delayInMS);
  }
};
h.postDelayed(r, delayInMS);
```

Cancelling Tasks

By the way, you can cancel an upcoming task like this:

```
h.removeCallbacks(r);
```

True Interval Timers

Handlers run when the thread is available.

True interval timers interrupt the processor.

Can simulate with `java.util.Timer`[1].

Payload runs in a different thread:

(+) more predictable timing;
(–) can't update UI from other thread.

Generally more overhead for `java.util.Timer`, not great for Android apps.

[1] `http://steve.odyfamily.com/?p=12`, accessed February 3, 2013.

Getting Around UI Thread Issues

Start the Timer like this:

```
Timer t = new Timer();
t.schedule(new TimerTask() {
  @Override
  public void run() {
    runOnUiThread(timerTick);
  }
}, firstDelayMS, repeatIntervalMS);
```

If you omit repeatIntervalMS, you get a one-shot timer.

Don't call `timerTick.run()` directly.

Programming timerTick

Put your UI-manipulating code in `timerTick`:

```
Runnable timerTick = new Runnable() {
  @Override
  public void run() {
    Toast.makeText(getApplicationContext(),
                   "ding!",
                   Toast.LENGTH_SHORT).show();
  }
};
```

Java Timer summary

To use a Java Timer:

1. instantiate a new `Timer` object;
2. schedule `TimerTask` on that `Timer`;
3. inside the `TimerTask`, invoke `Runnable` to run on the UI thread; and
4. implement the `run()` method on the `Runnable` to effect UI changes.

Part II

Watchdog Timers

Goal of a Watchdog Timer

Observe that a system appears to be stuck and take some action.

About Watchdog Timers

Commonly found in computer systems where humans cannot easily take corrective action.

If a NASA satellite is stuck, it's hard to send astronauts up to push the reset button.

If not for the ability to automatically restart, the satellite could be permanently disabled.

About Watchdog Timers

The system needs to indicate, before the limit of the timer, that everything is okay.

If the system thinks all is well, either by checking some fault conditions or simply by default, the watchdog timer is reset.

Watchdog Example

Checking if free memory is below 256 MB.

If the system detects free memory is $>$256 MB, reset timer.

If it is below 256 MB, then the watchdog timer is not reset.
The timer will fire.

Corrective action takes place: (i.e., freeing up some memory).

Implementing a Watchdog Timer

Can build a watchdog timer using an interval timer:

- Set the timer interval to be the largest allowable time for a task to take. Start the timer.
- If the timer fires, the task took too long.
- The timer event handler deals with the situation
  (e.g. by killing the task.)

Resilience for Watchdog Timers

In an embedded system:
specialized hardware.

Timer has some resilience:
runs in a separate thread.

Android Watchdog Timer

Summary:

- start the watchdog timer;
- the timer runs a task *t* on the UI thread;
- task *t* kills the activity; and
- set up a click listener to go to a web page instead.

Outcomes from Watchdog Timer

1. If you click the button soon enough, app goes to the web page;
2. Otherwise, the Activity finishes.

Multiple Stage Watchdog Timers

So far corrective action has been all-or-nothing.

We can have multiple stages – multiple levels of response.

If the first level of response does not resolve the problem:
second level of response activates; takes more drastic action.

Two Stage Watchdog Timer Example

The first watchdog timer attempts to restart `example.exe`.

The first level has a timer length of 180 seconds; every 60 seconds, `example.exe` resets the watchdog timer.

If the reset doesn't happen, the watchdog activates.

Watchdog 1: starts watchdog 2, with a length of 120 seconds. Then the first watchdog tries to restart `example.exe`.

If the first watchdog does not successfully restart the program within the 120 seconds, the second watchdog timer activates and restarts the computer.

Mars Pathfinder

Mars Pathfinder

A few days into the mission, the lander began experiencing resets.

Data was being lost.

The press reported these failures in terms such as “software glitches” and “the computer was trying to do too many things at once”.

Mars Pathfinder

Pathfinder contained an “information bus”.

A bus management task ran frequently with high priority to move certain kinds of data in and out.

Access to the bus was synchronized with mutual exclusion locks.

Mars Pathfinder

The meteorological data gathering task ran as an infrequent, low priority task.

When publishing, it would lock the bus, do writes to the bus, and then unlock it.

The spacecraft also contained a communications task that ran with medium priority.

If an interrupt caused the information bus thread to be scheduled while the bus was locked, the problem occurred.

Mars Pathfinder

Most of the time this combination worked fine.

It was possible for the communications task to be scheduled while the information bus task was waiting for the low priority task.

The communications task, having higher priority, would prevent the lower priority meteorologicgal task from running.

... and therefore the information bus task could not run.

Mars Pathfinder

After some time had passed, a watchdog timer would go off.

Tthe data bus task had not been executed for some time.

The watchdog concluded that something had gone drastically wrong, and initiated a total system reset.

Part III

Timer Coalescing

Timer Coalescing

Using timers has an impact on power consumption.

Suppose an interval timer set up with an interval of 1000ms.

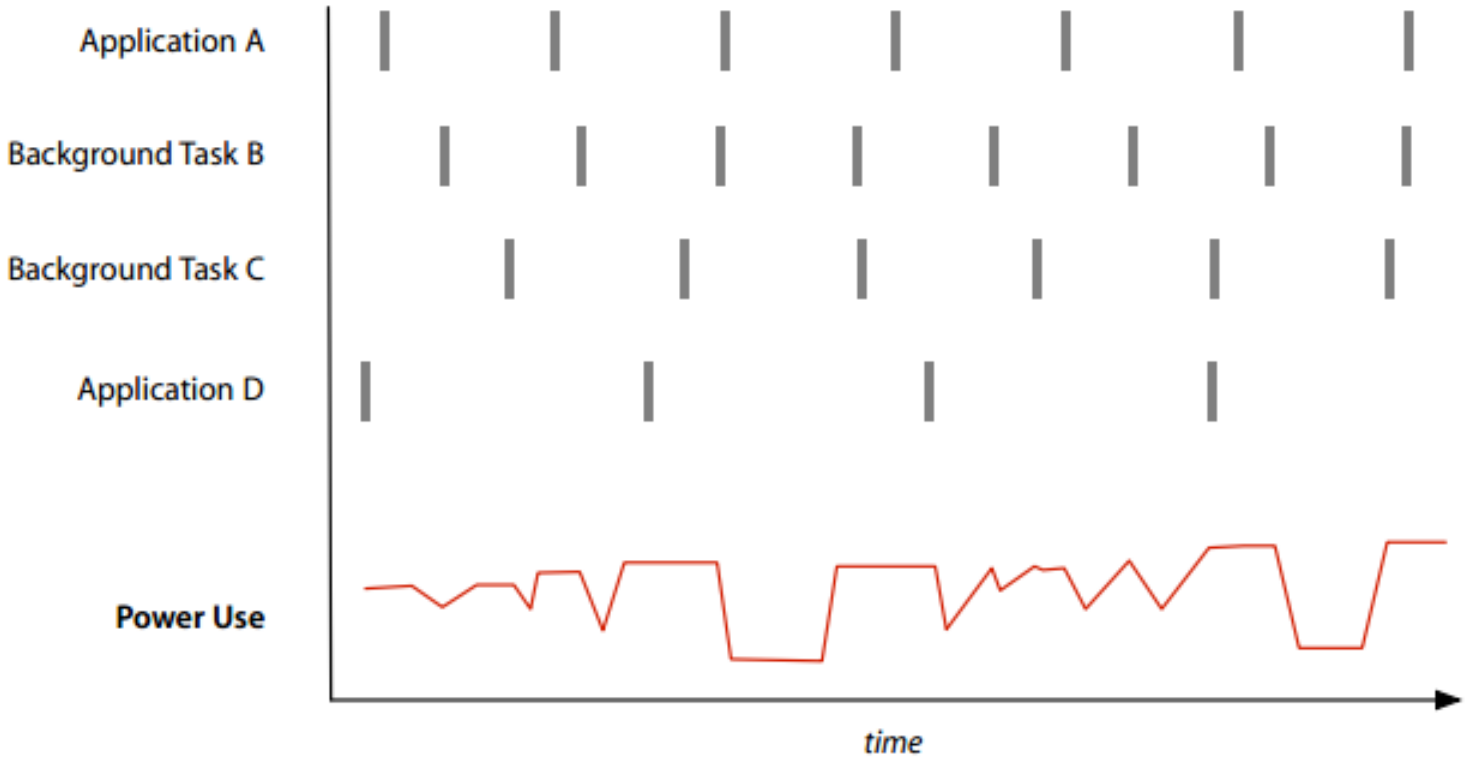
Every second the system will execute the timer's action – the system wakes up, runs the action, then goes back to sleep.

Timer Coalescing

What if there are multiple interval timers?

The system might never actually get to sleep;
constantly being woken up by interval timers.

Timer Coalescing



Typically, numerous applications and background processes use timers with different intervals.

Timer Coalescing Semantics

A critical insight about timers.

For a 1000ms timer, the system does not promise that the timer will run exactly 1000 ms from now.

What it really says is that the timer task will execute no sooner than 1000 ms from now.

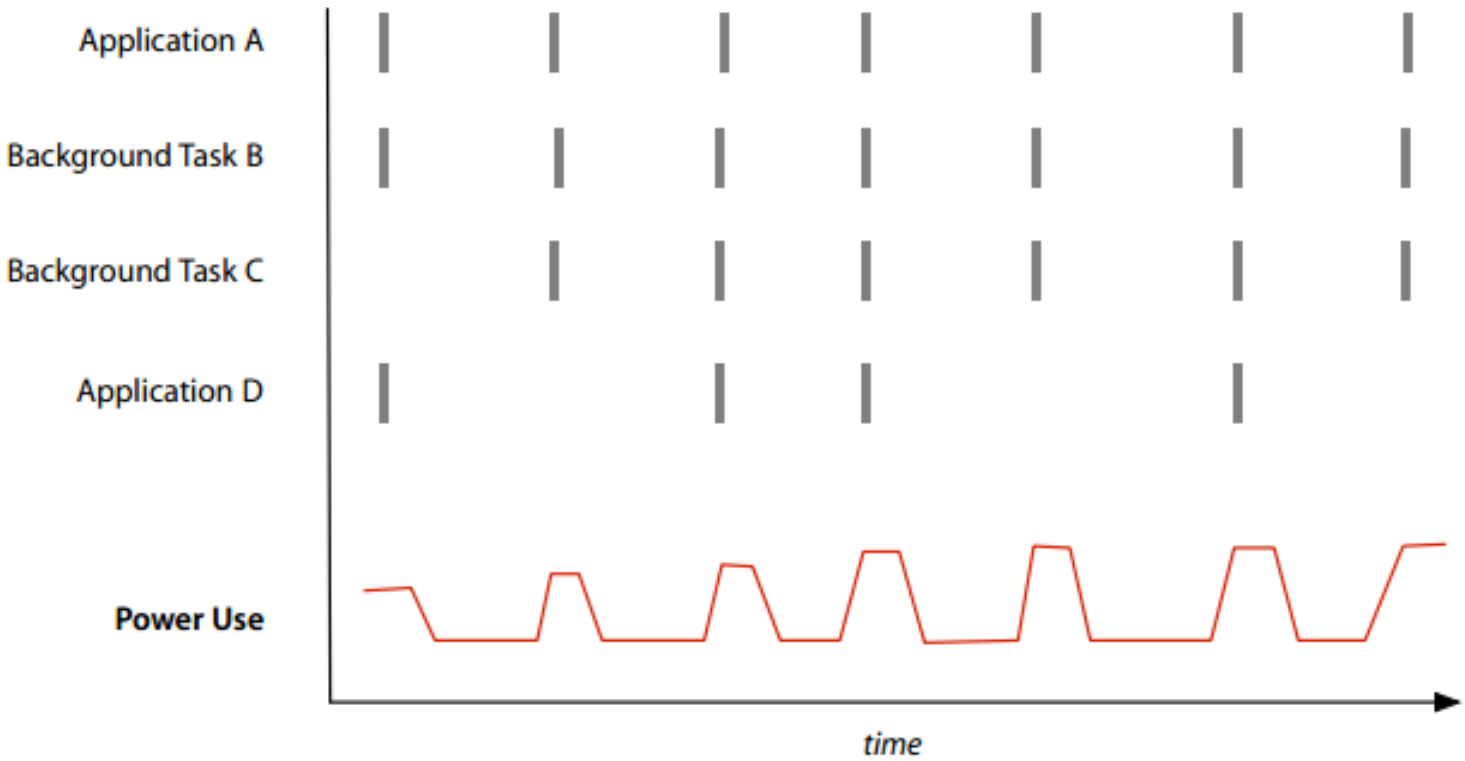
It could be 1004 ms or 1200 ms later;
yet it could not be 997 ms or 980 ms later.

Timer Coalescing Semantics

The OS can do something clever using these semantics.

Delay some timers a little bit so that they line up.

Timer Coalescing



Timer Coalescing shifts timers of multiple applications to execute at the same time.

Timer Coalescing Semantics

Now, this image is not a hundred percent accurate.

Some of the lines were moved to the left (back in time) to make them line up.

This is just artistic license; in reality the lines could only be shifted to the right.

Most modern operating systems do this, including Windows and Mac OS X.