Block, Don't Run

An Introduction to Cocoa Run Loops

Daniel Jalkut | red sweater

Sequential Programs

* Convert a JPG to GIF
* Run payroll
* Compile a source file
* Etc.

Event-Driven Programs

- Keyboard, mouse, touch, shake
- Disks and other devices
- Network activity
- Timed operations

“The purpose of a run loop is to keep your thread busy when there is work to do and put your thread to sleep when there is none.”

Threaded Programming Guide

Apple Computer

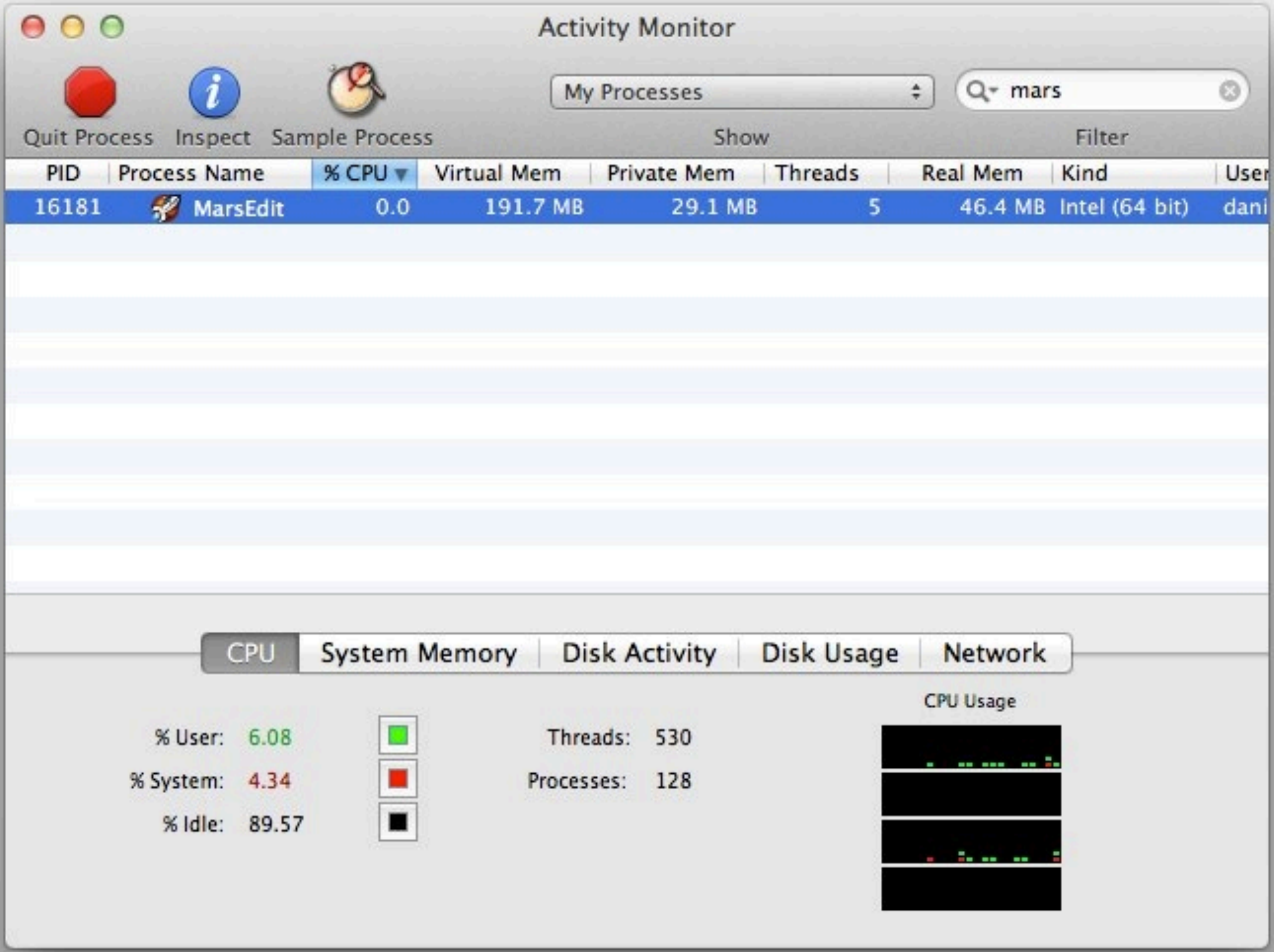
NSRunLoop

- Created for you on main thread
- One per thread
- CFRunLoop at lower-level

Good News!

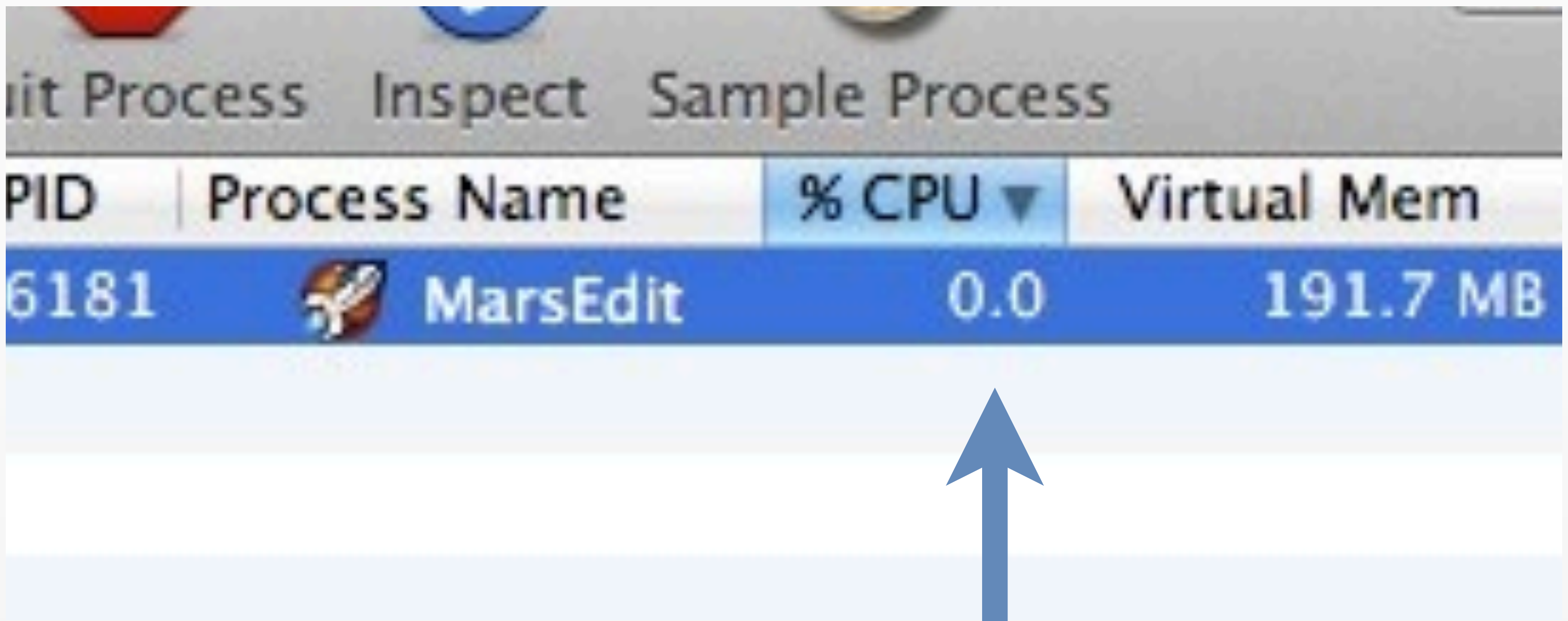
Why Learn More?

* Professional duty
* Performance tuning
* Memory management
* Inter-thread communication
* Keep threads alive!



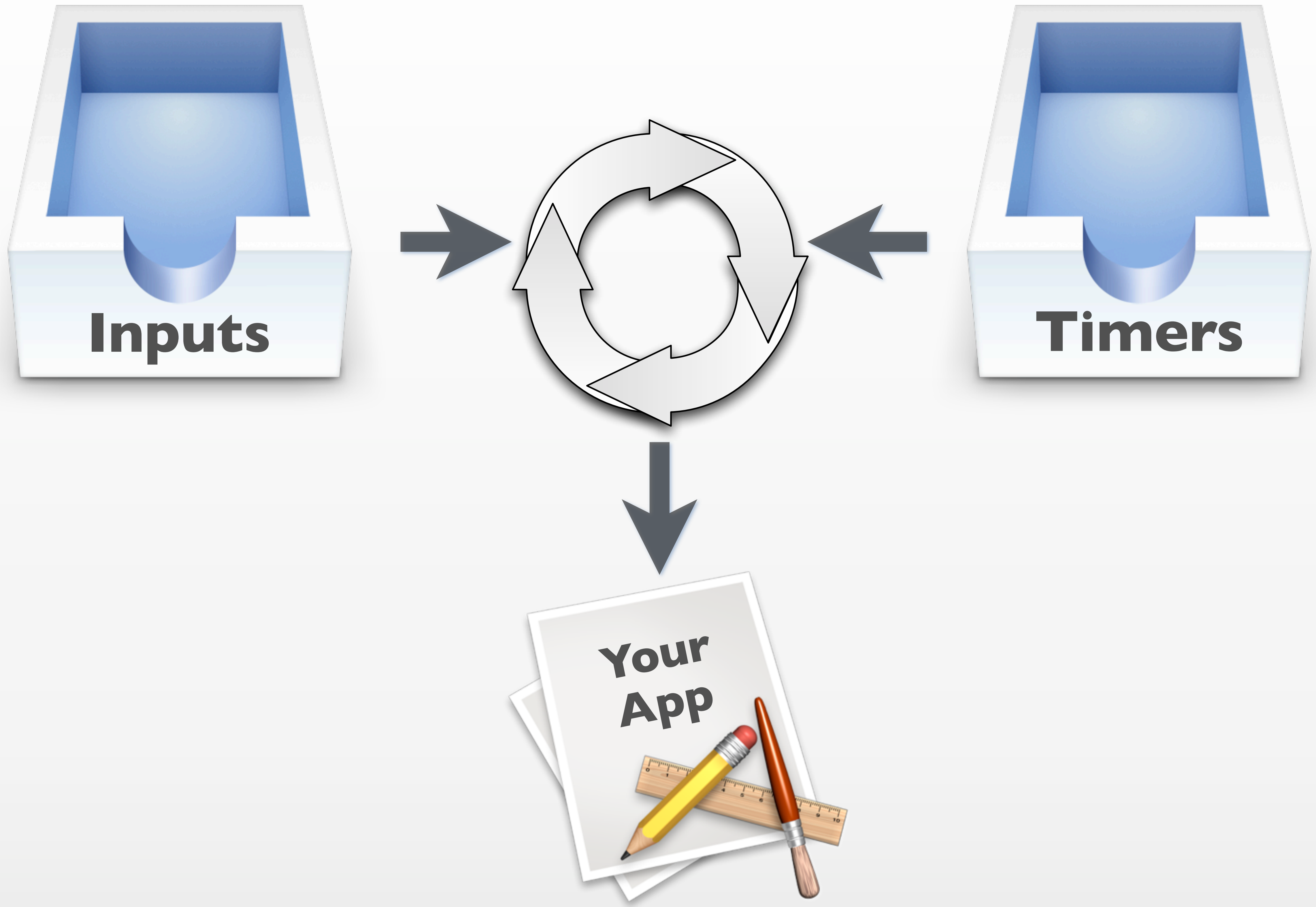
PID	Process Name	% CPU	Virtual Mem	Private Mem	Threads	Real Mem	Kind	User
16181	MarsEdit	0.0	191.7 MB	29.1 MB	5	46.4 MB	Intel (64 bit)	dani

% User:	6.08	🟩	Threads:	530
% System:	4.34	🟥	Processes:	128
% Idle:	89.57	⬛		



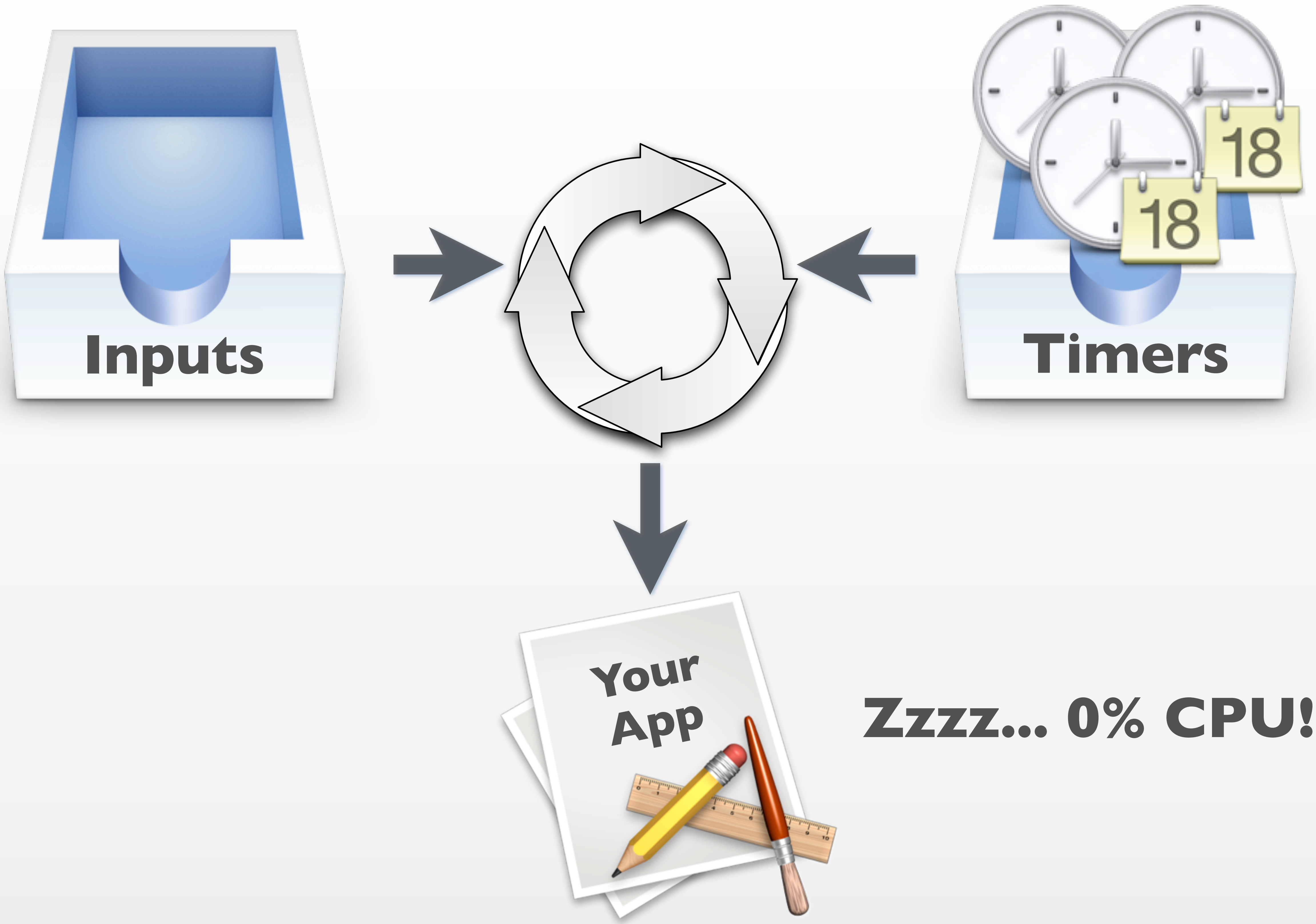
PID	Process Name	% CPU	Virtual Mem
6181	MarsEdit	0.0	191.7 MB

Great work, Jalkut!

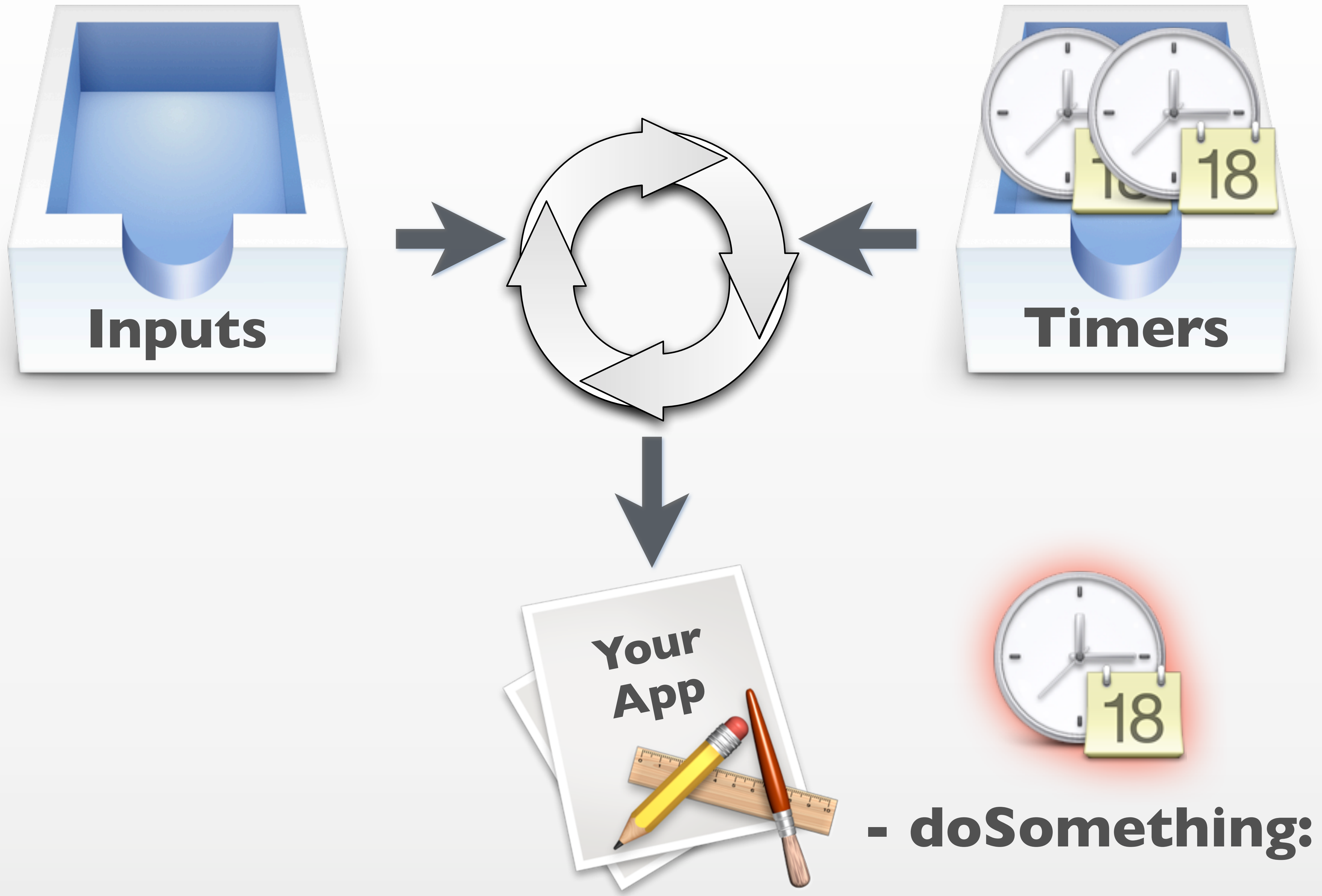

Inputs
Timers
Your App

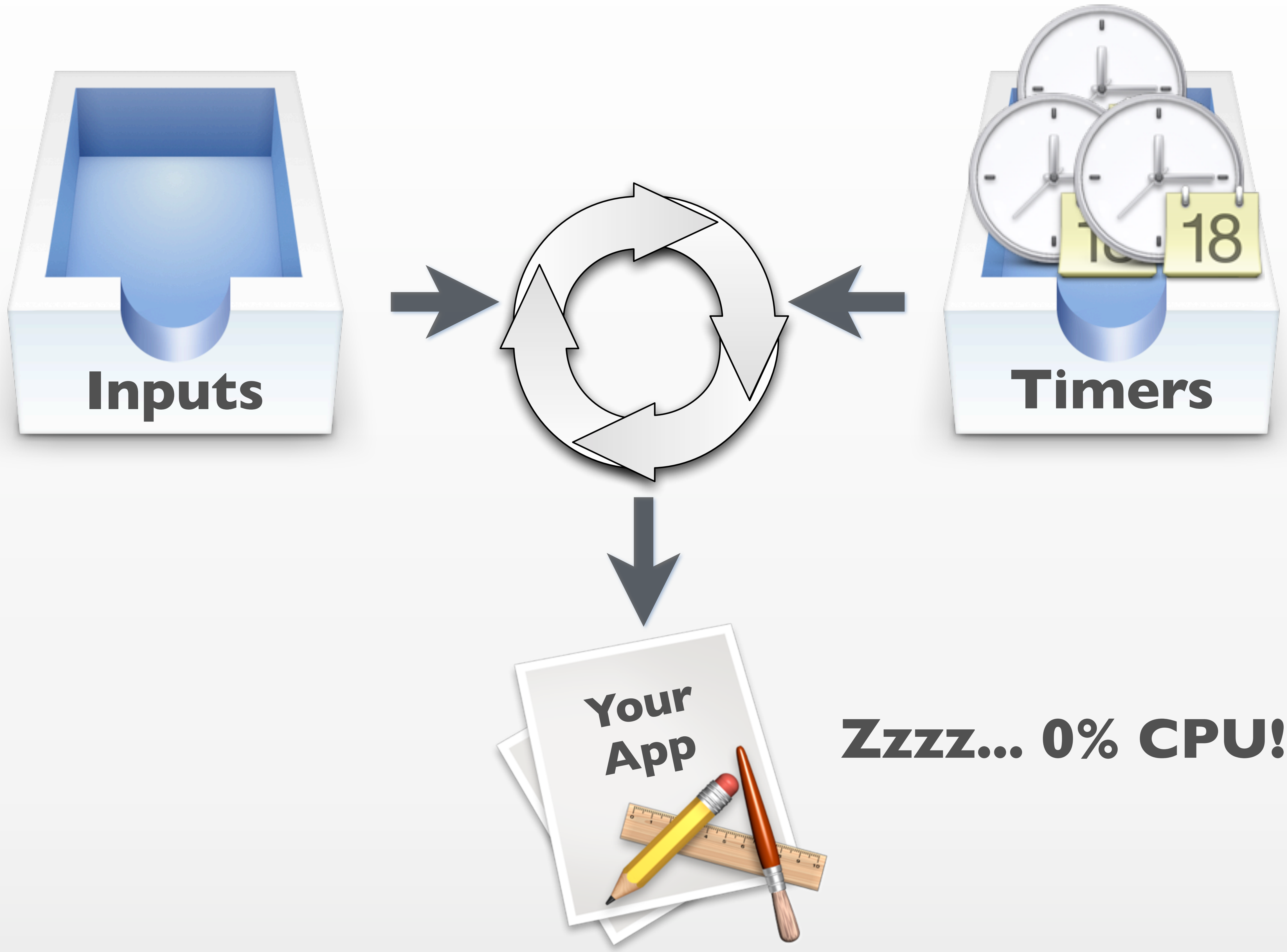
Timer Sources

* “Stuff you kind of control”
* May be auto-repeating
* Not highly accurate



Zzzz... 0% CPU!

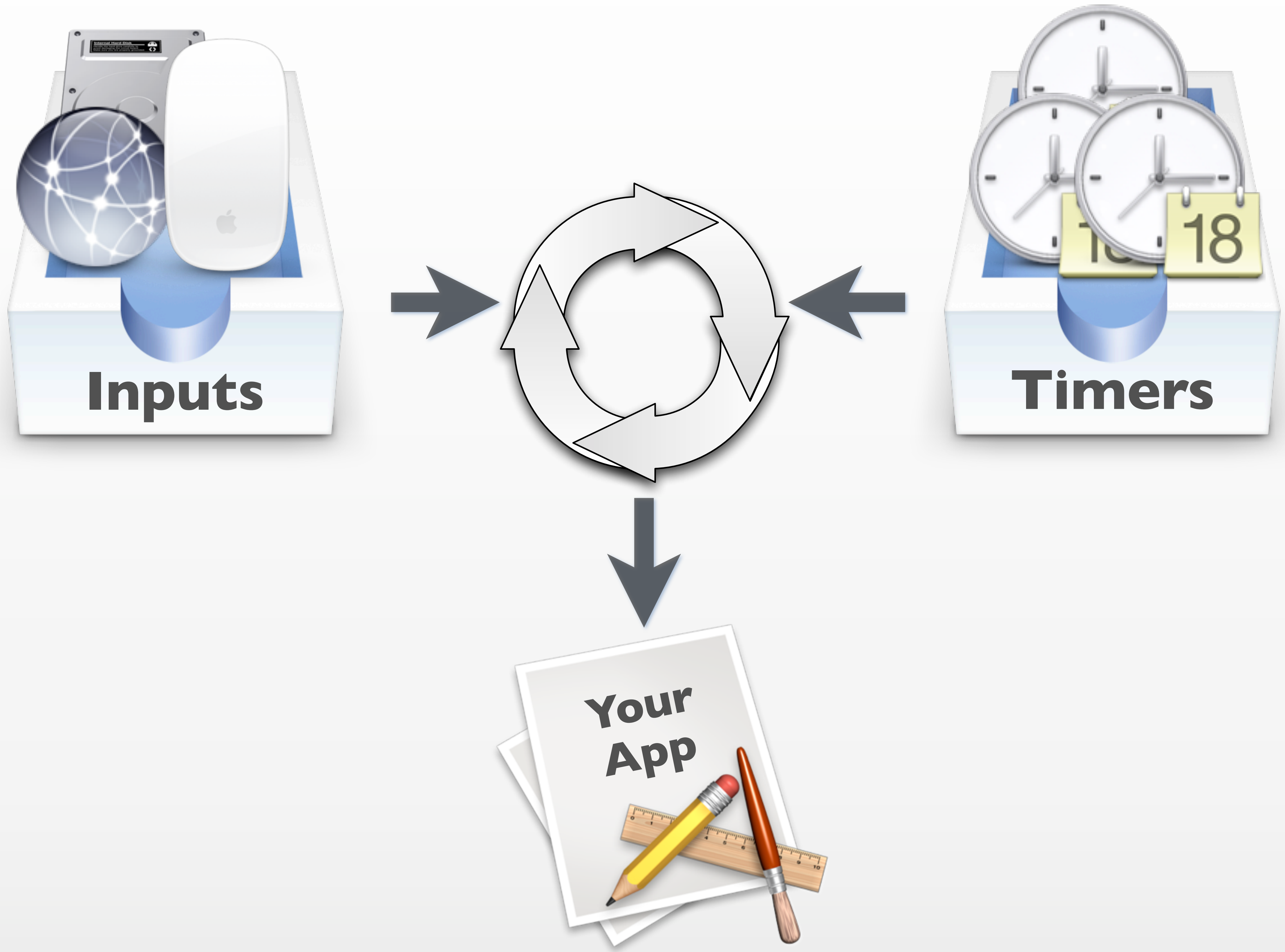

Inputs
Timers
18
Your App
18
- doSomething:

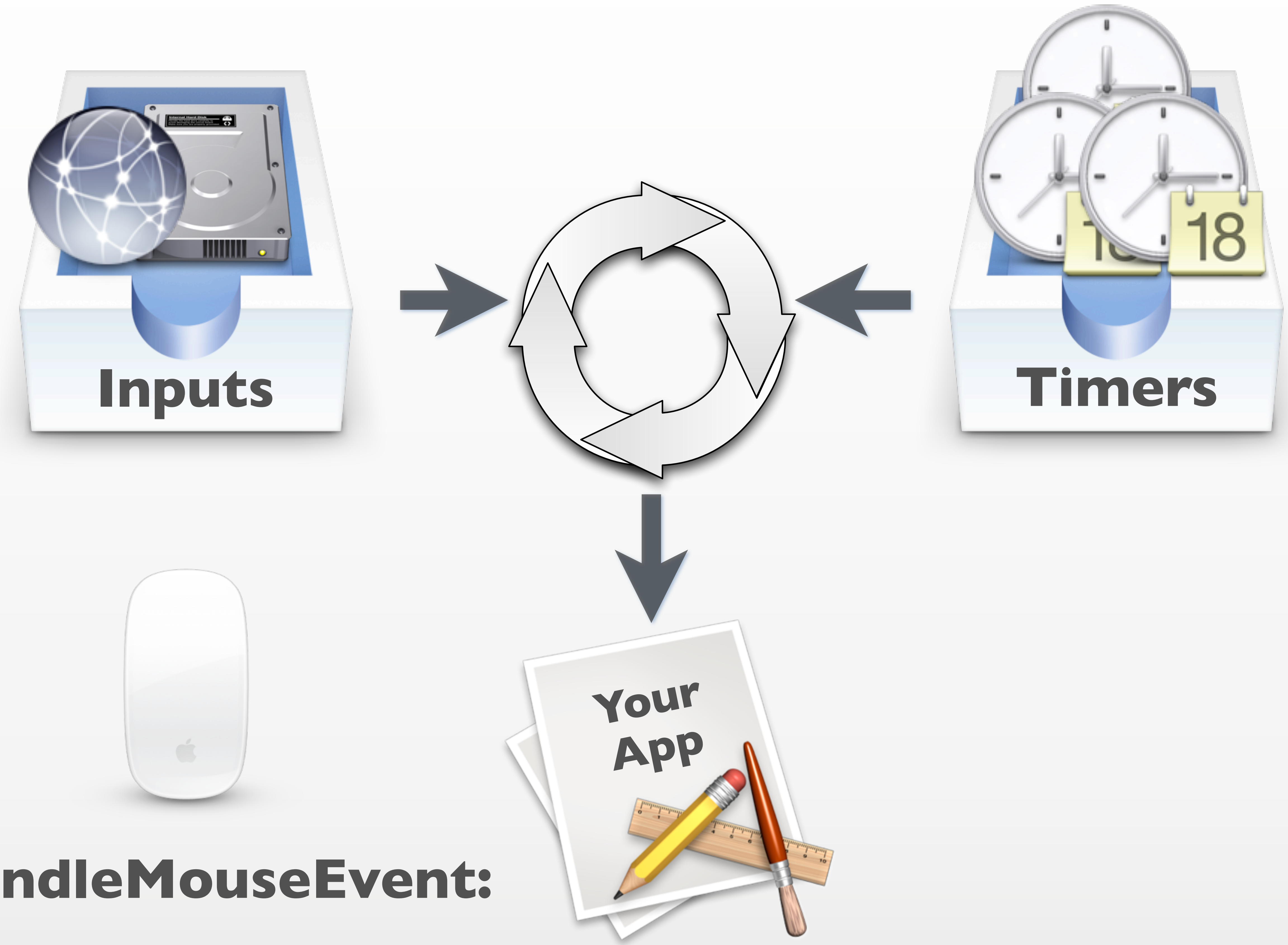


Zzzz... 0% CPU!

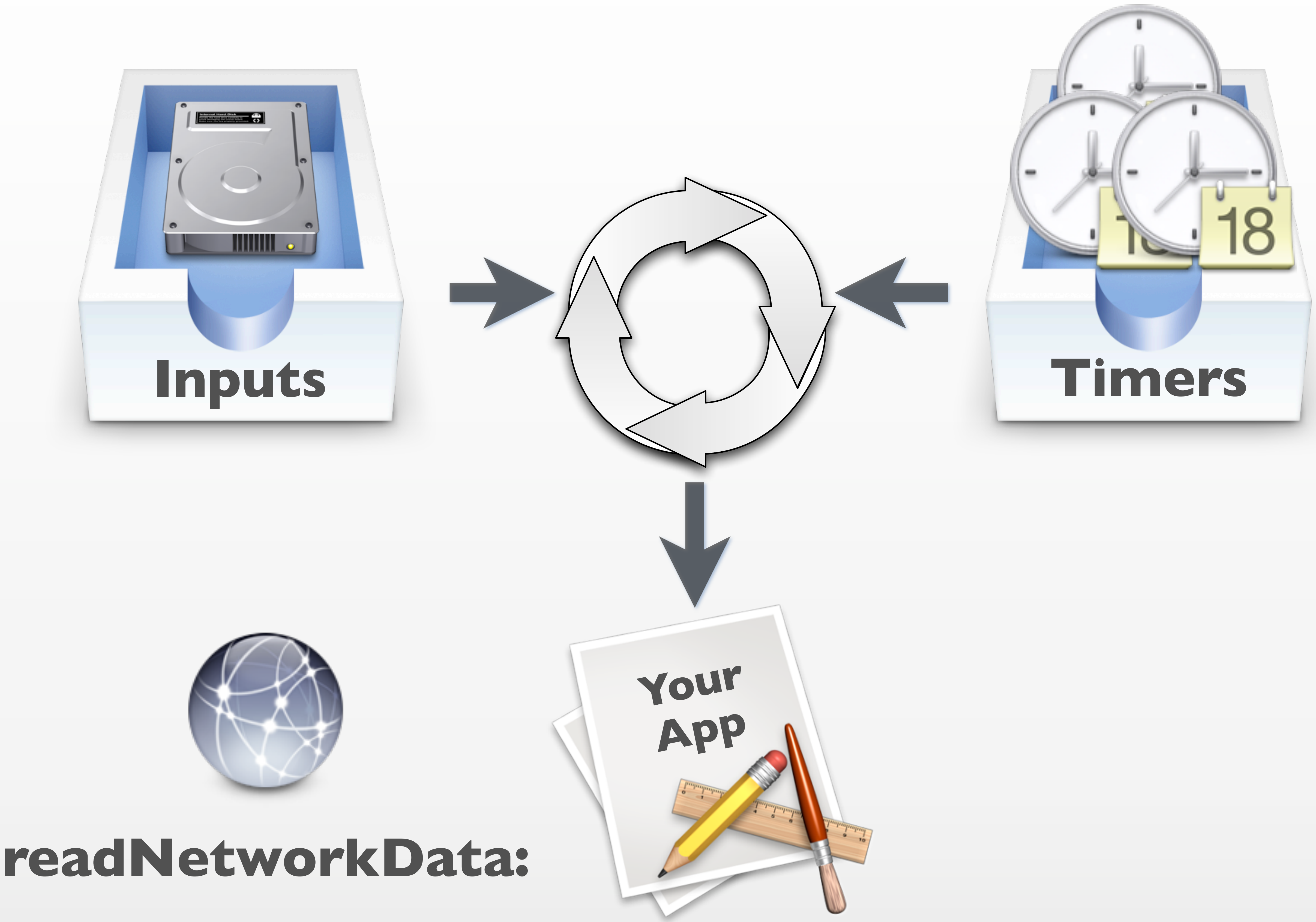
Input Sources

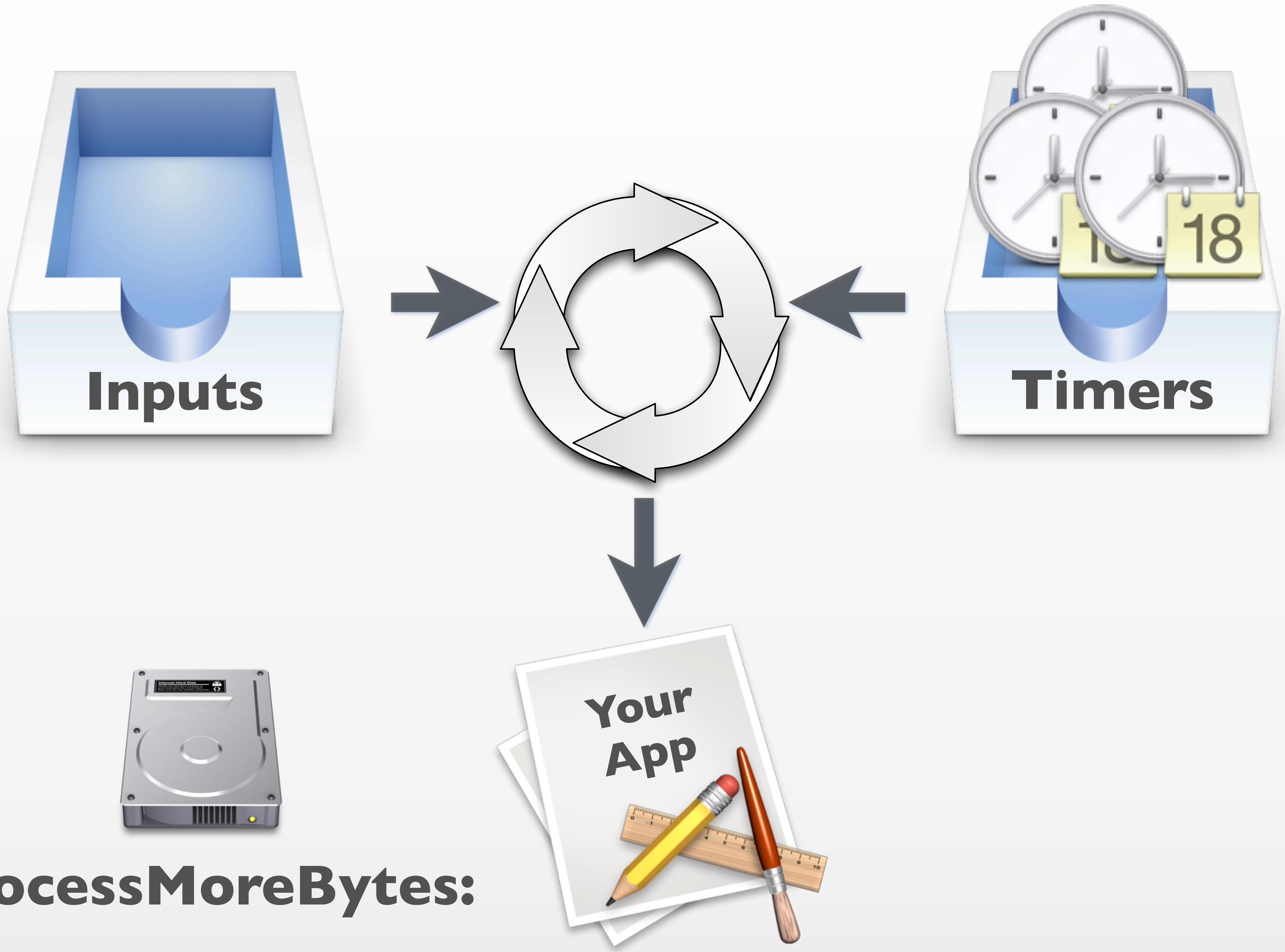
* “Stuff you don’t really control”
* Events (hardware, network, etc)
* Inter-thread messages
* Mach-port messages (kernel level IPC)


Inputs
Timers
Your App

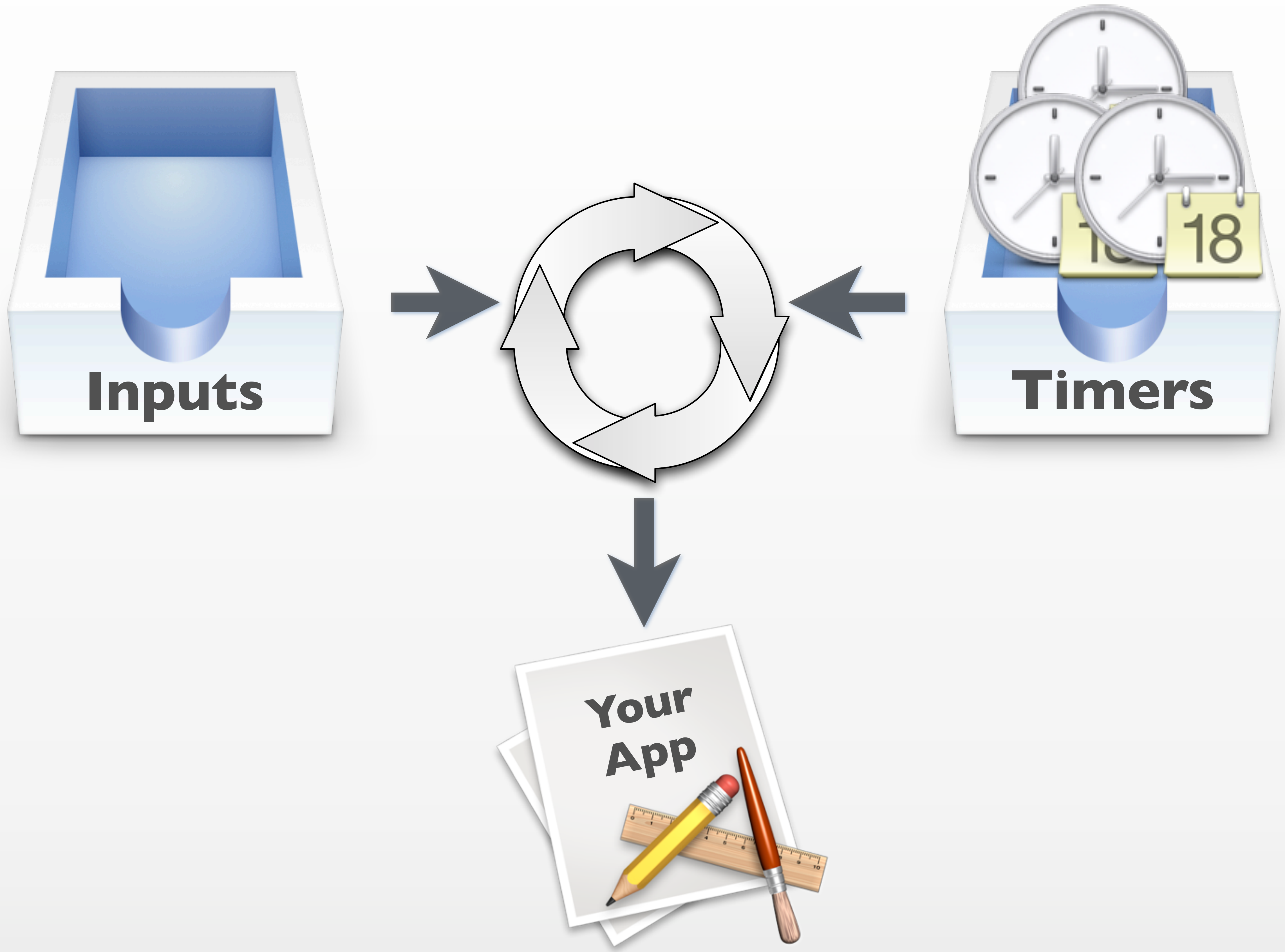


handleMouseEvent:


Inputs
18
Timers
Your App
readNetworkData:

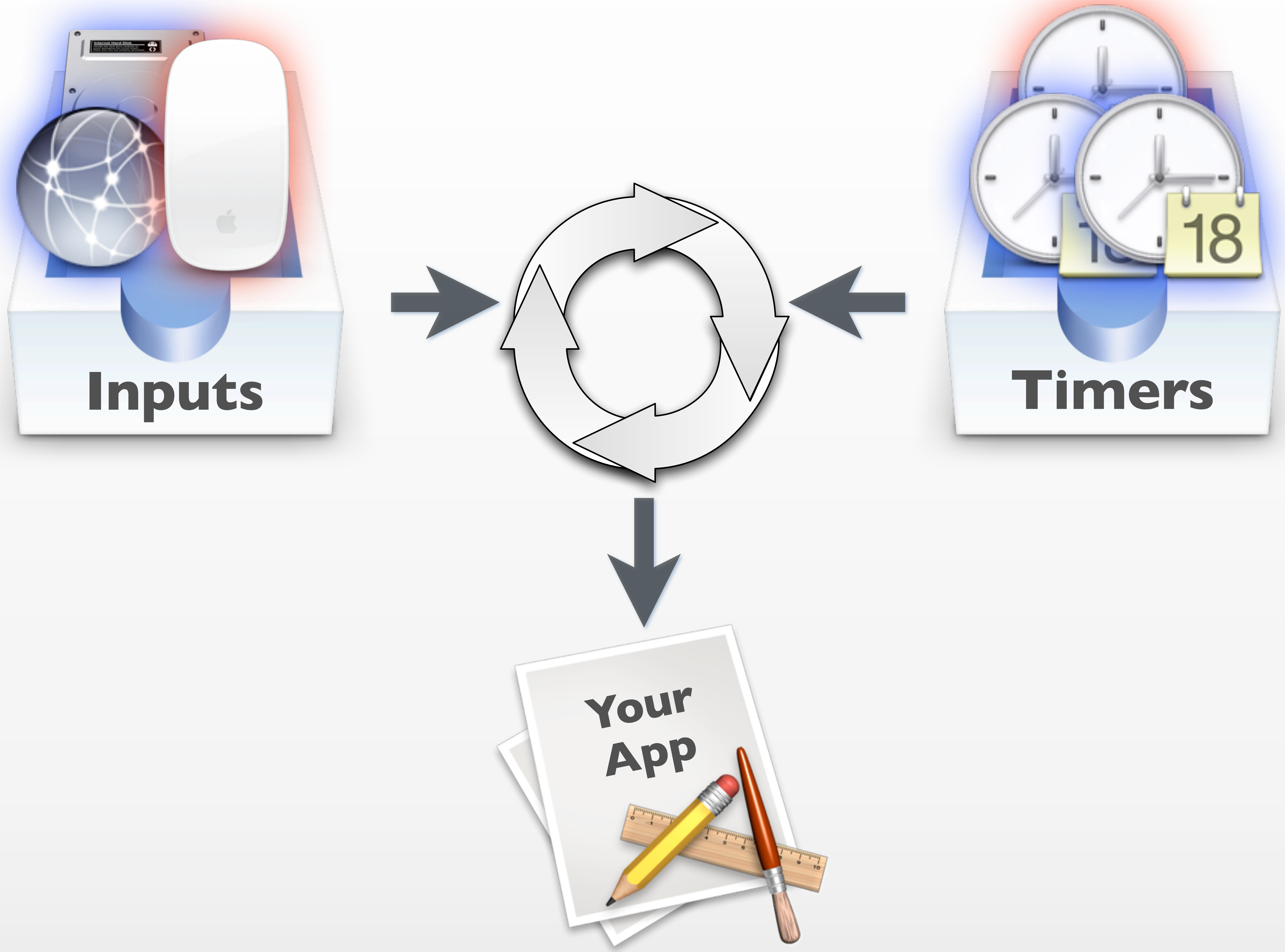


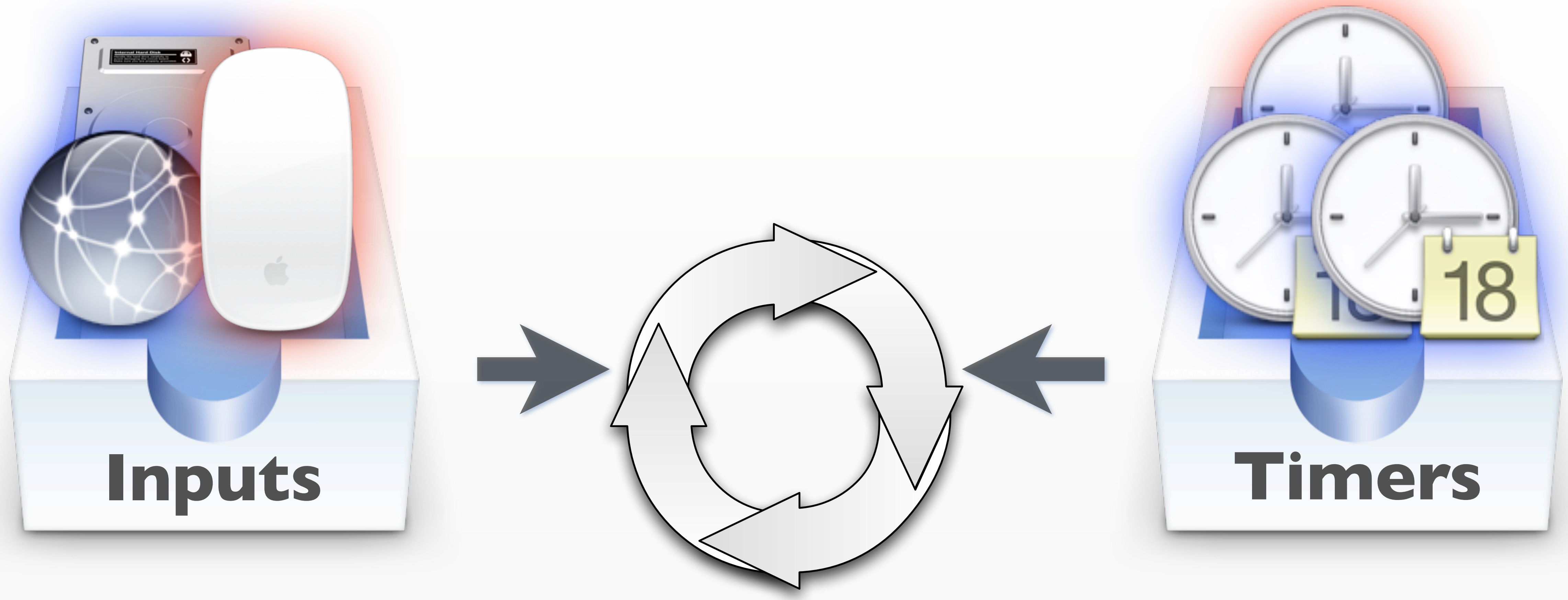
processMoreBytes:


Inputs
Timers
18
Your App

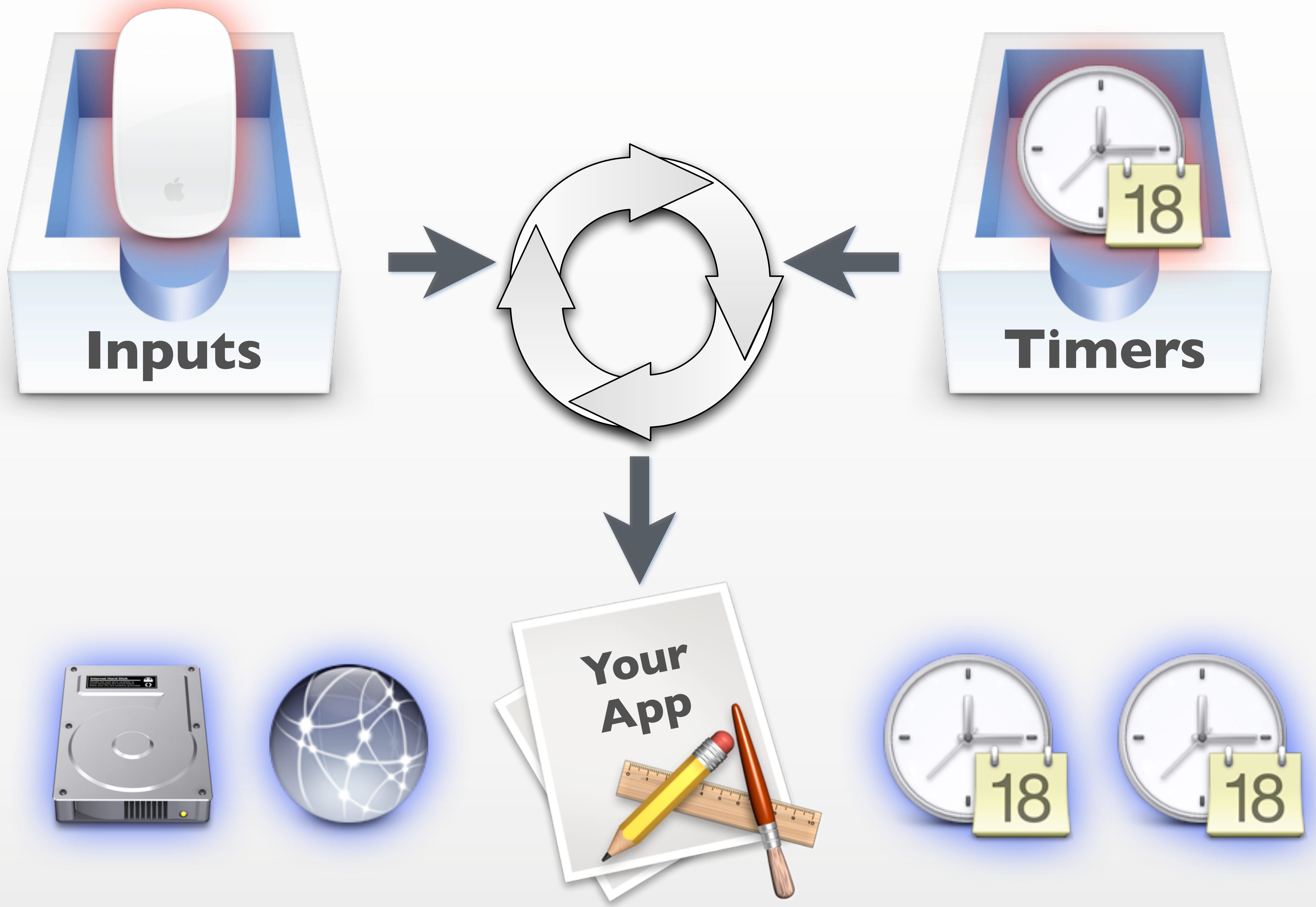
Modes

- A filter mechanism for inputs and timers
- NSDefaultRunLoopMode
- NSEventTrackingRunLoopMode
- NSModalPanelRunLoopMode
- @“com.your-company.runloop.mode”
- NSRunLoopCommonModes
  - A “magical” meta mode comprising many


Inputs
Timers
18
Your App



```
[myRunLoop runMode:kBlueMode untilDate:...];
```


Inputs
Timers
18
Your App
18
18

Practical: Performance

* **Finish main thread work in spurts**
* **Use asynchronous APIs when possible**

```
[[NSURLDownload alloc] initWithRequest:myReq delegate:self];
```

* **Don’t Use Threads! ...unless you need to**

```
[NSThread detachNewThreadSelector:@selector(mySlowTask)
          toTarget:self withObject:nil];
```

* **See also: GCD, libdispatch**

Practical: Memory

* **Preserve objects until end of loop cycle**

```
return [[someObject retain] autorelease];
```

* **Use local autorelease pools**

```
for (i = 0; i < 1000; i++) {
   NSAutoreleasePool* ap = [[NSAutoreleasePool alloc] init];
   [someClass doSomethingThatBurnsAutoreleasedObjects];
   [ap release];
}
```

* **See also: Automatic Reference Counting**

Practical: Thread Safety

* **Don’t use threads**
* **Run main-thread only operations**

```
[myUIController performSelectorOnMainThread:dangerousSEL...];
```

* **Micro-postpone time-sensitive operations**

```
SEL mySEL = @selector(updateUserInterface:);

[myView performSelector:mySEL withObject:nil afterDelay:0];
```

Practical: Run Free Or Die

* **All threads must confront death...**

```
main() {
     NSApplicationRun();
     // OK, we're about to die
}
```

* **Access the "currentRunLoop" for a thread**

```
NSRunLoop* myRL = [NSRunLoop currentRunLoop];
```

* **Sleep until some input or timer**

```
[myRL runUntilDate:[NSDate distantFuture]];
```

* **Tickle the run loop while we do stuff...**

```
[myRL runUntilDate:[NSDate dateWithTimeIntervalSinceNow:0.1]];
```

Run Loop Summary

- ♥ The heart of your Cocoa app
- * Efficient CPU sharing on a single thread
- * Facilitates inter-thread communication
- * Client code executes quickly, returns control

Advanced Study

* Apple’s Threaded Programming Guide
  http://bit.ly/applerunloops
* Mike Ash “Friday Q & A”
  http://bit.ly/ashrunloops

red sweater

Daniel Jalkut
jalkut@red-sweater.com
www.red-sweater.com
@danielpunkass