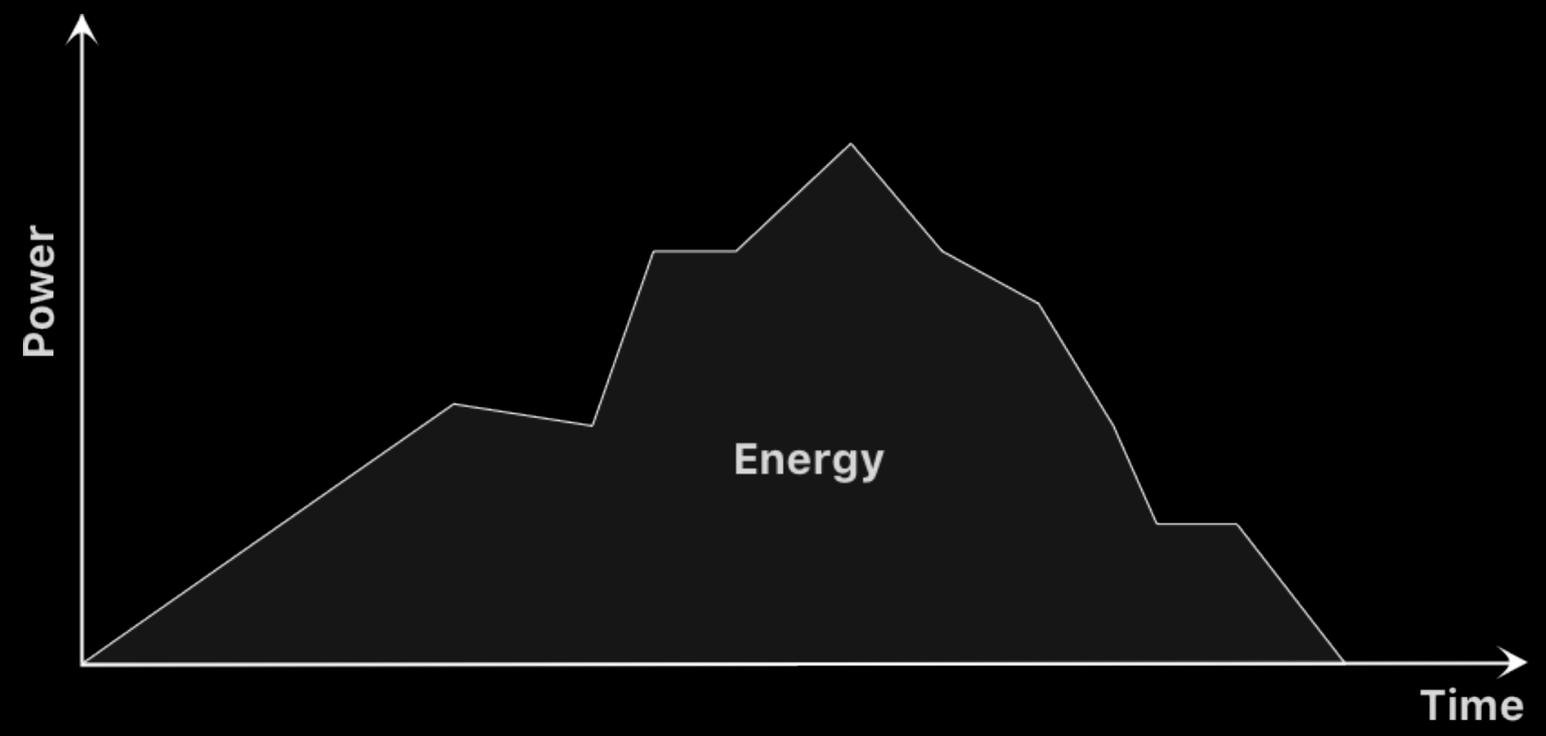
"Green" Development: is it a thing?

🍏 Aleksandra Komagorkina
🐦 @akomagorkina

"Green development is a real estate development concept that carefully considers social and **environmental impacts** of development."

— *Wikipedia*

What is Energy?



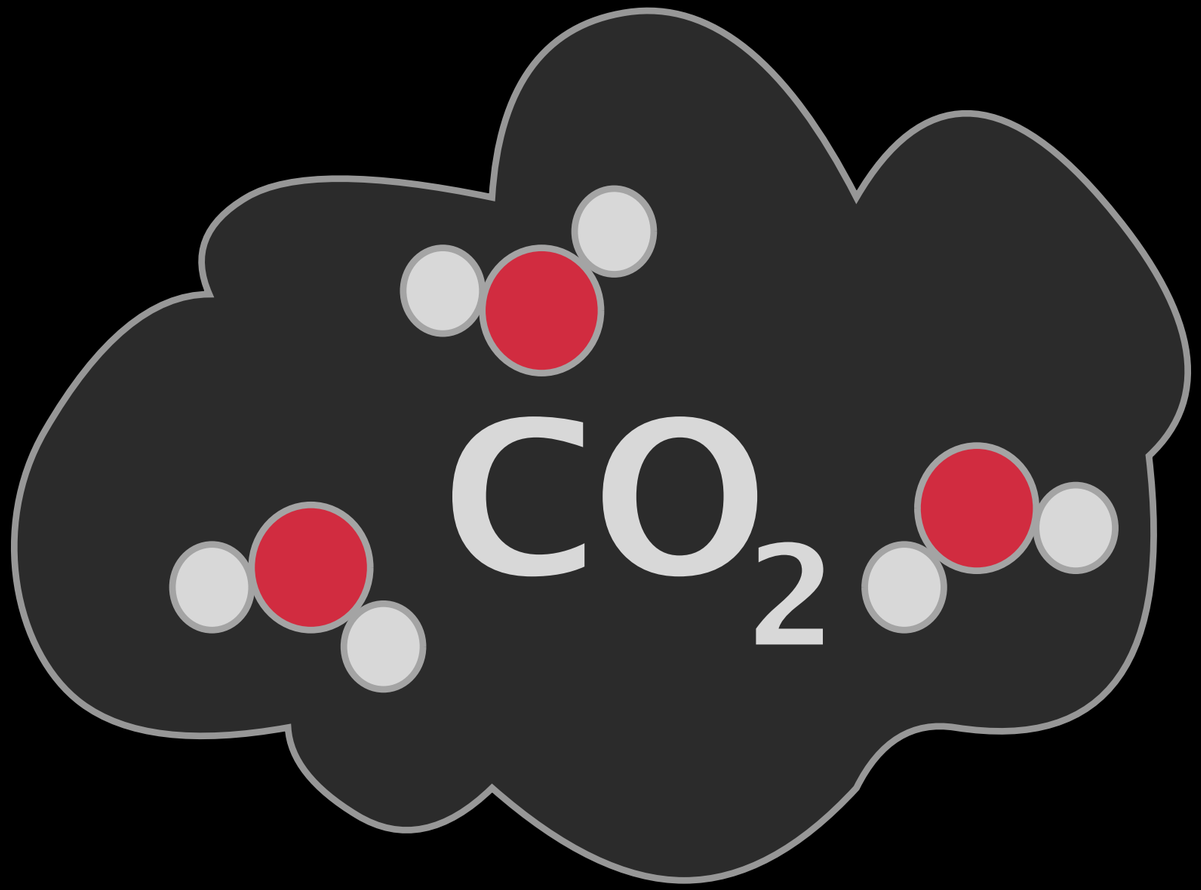
$Power$ (kW) x $Time$ (hours) = $Energy$ (kWh)

Why do we care about energy?

Why do we care about energy?

Emissions

What are these emissions?



How much energy do you need to charge your iPhone?

iPhone XS Max battery holds a charge of * **3174 mAh**. If you fully drain and recharge your phone everyday, then over a year you would have to feed it about 4 409 watt hours, or **4,4 kWh**.

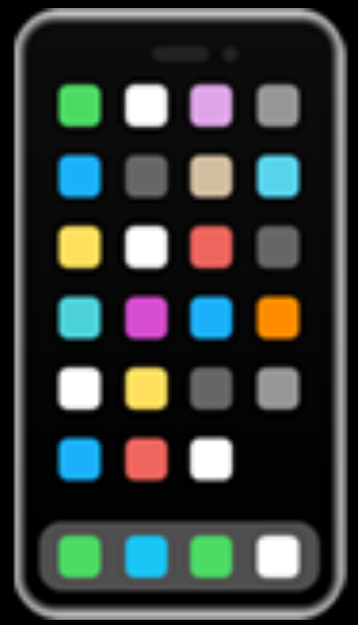
— *Forbes*

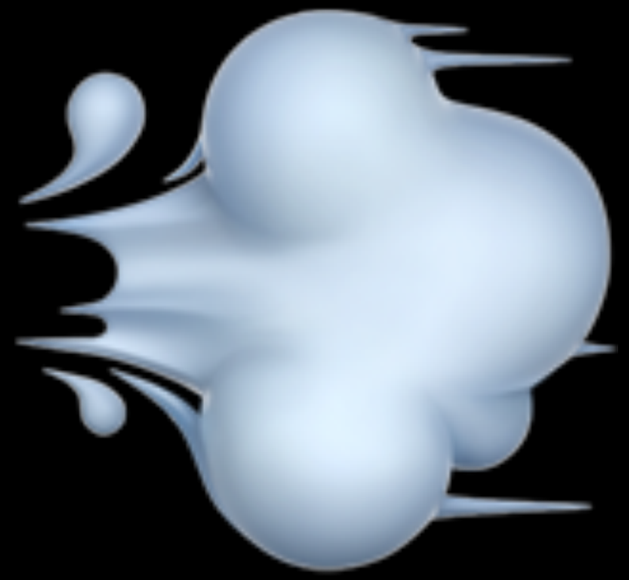
To obtain **1kWh** from coal or fuel, **800g of CO_2** will be rejected in the atmosphere during combustion of fossil fuel

— J. Bernard, Sciences et vie 214 (2001) 68

In April 2017, 728 million iPhones were in use worldwide

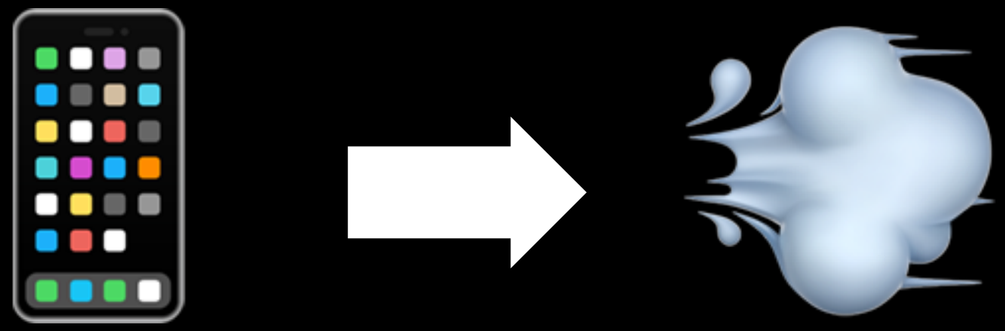
– *statista.com*





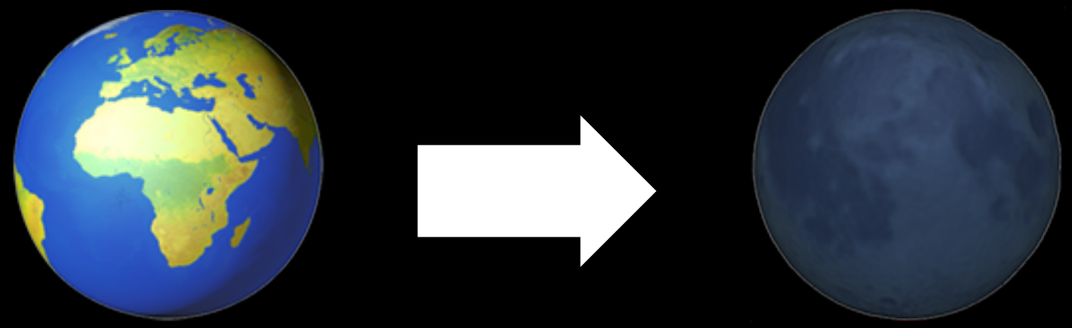
4.4 kWh

3.5 kg



4.4 kWh

3.5 kg



0.728
billion people

2.5
million tons

What can we do?

Understand how enegry is used on the iPhone

CPU

Device wake

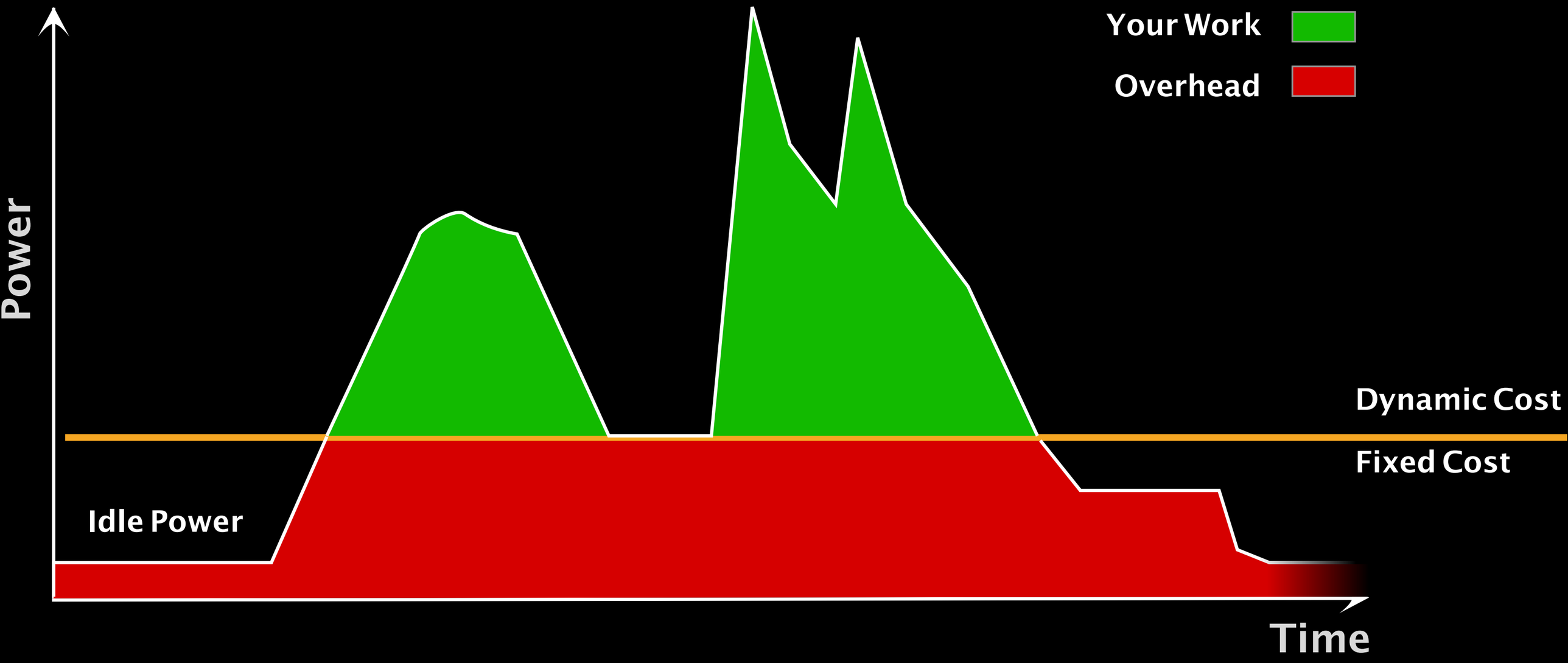
Networking operations

Graphics, animations, and video

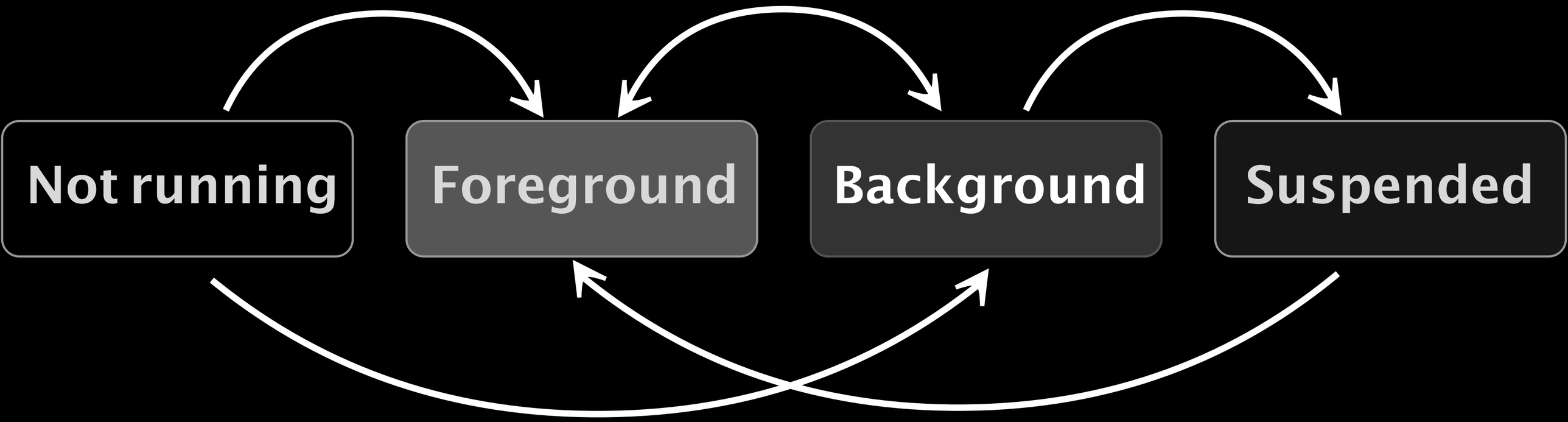
Location

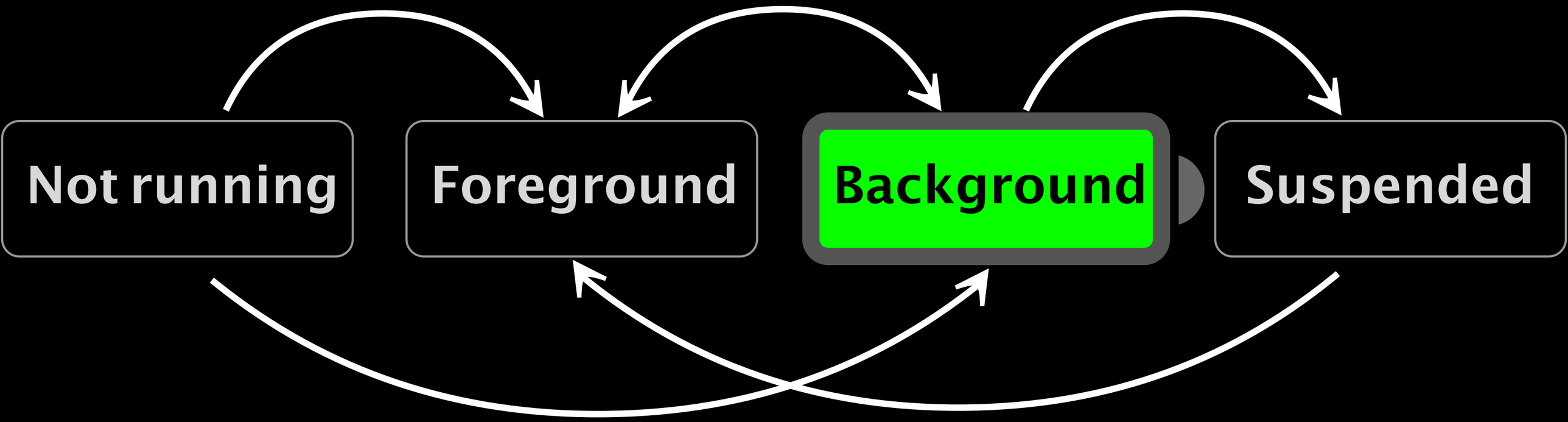
Motion

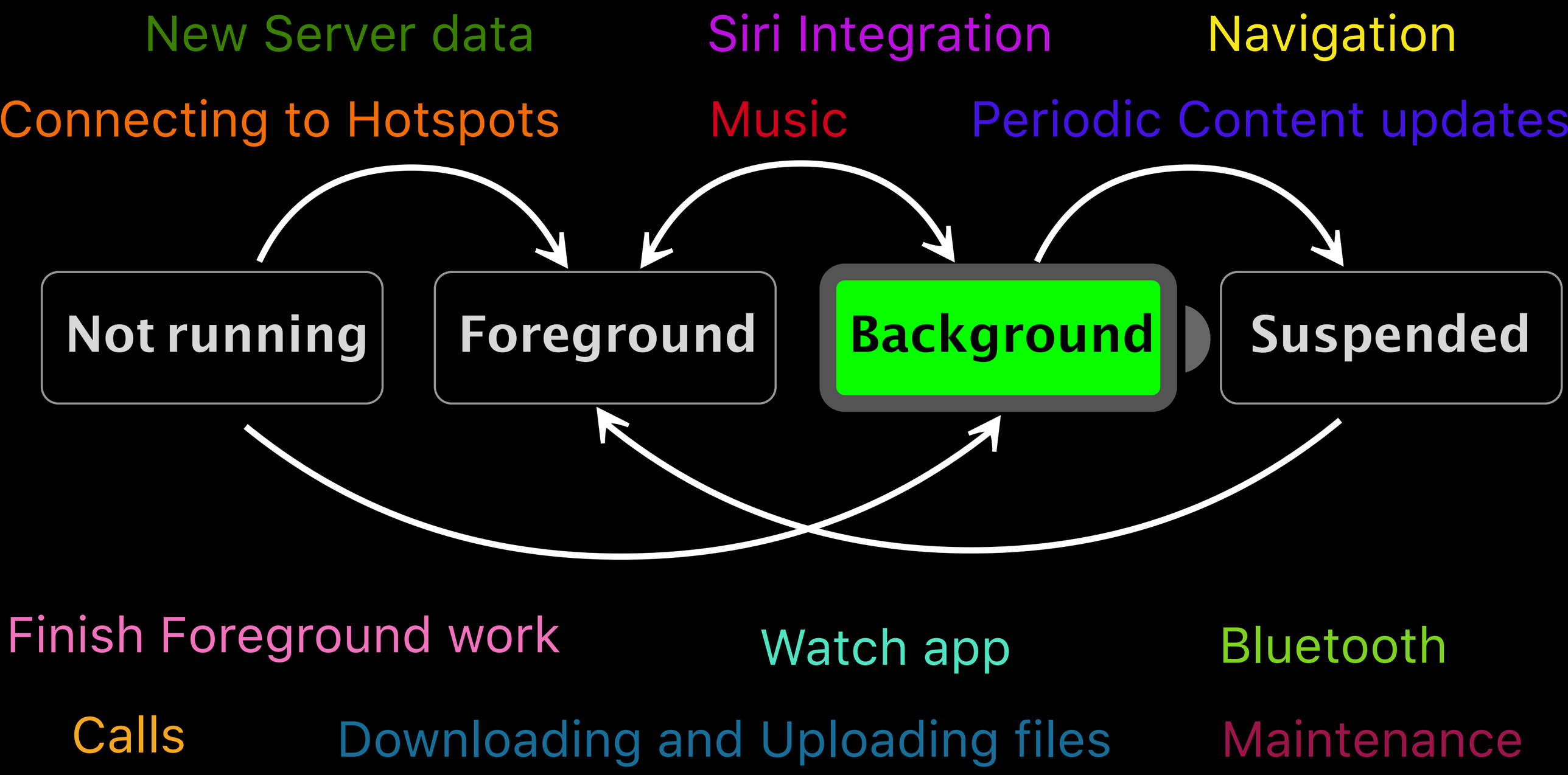
Bluetooth


Your Work
Overhead
Power
Dynamic Cost
Fixed Cost
Idle Power
Time

Background


Not running
Foreground
Background
Suspended


Not running
Foreground
Background
Suspended


New Server data
Siri Integration
Navigation
Connecting to Hotspots
Music
Periodic Content updates
Not running
Foreground
Background
Suspended
Finish Foreground work
Watch app
Bluetooth
Calls
Downloading and Uploading files
Maintenance

Energy waste in background

Energy waste in background

- Not notifying the system when background activity is complete

Energy waste in background

- Not notifying the system when background activity is complete
- Playing silent audio

Energy waste in background

- Not notifying the system when background activity is complete
- Playing silent audio
- Performing location updates

Energy waste in background

- Not notifying the system when background activity is complete
- Playing silent audio
- Performing location updates
- Interacting with Bluetooth accessories

Energy waste in background

- Not notifying the system when background activity is complete
- Playing silent audio
- Performing location updates
- Interacting with Bluetooth accessories
- Downloads that could be deferred

What can we do?

Complete background tasks

Background Task Completion

Background Task Completion

- User expects immediate completion

Background Task Completion

- User expects immediate completion
- Protect completion

=> Give the app additional time to run in the backgrond before being suspended

Background Task Completion

```
UIApplication.beginBackgroundTask(expirationHandler:)

ProcessInfo.performExpiringActivity(withReason:using:)
```

Background Task Completion

```
// Guarding Important Tasks While App is Still in the Foreground
func send(_ message: Message) {
  let sendOperation = SendOperation(message: message)
  var identifier: UIBackgroundTaskIdentifier!
  identifier = UIApplication.shared.beginBackgroundTask(expirationHandler: {
     sendOperation.cancel()
     postUserNotification("Message not sent, please resend")
     // Background task will be ended in the operation's completion block below
  })
  sendOperation.completionBlock = {
     UIApplication.shared.endBackgroundTask(identifier)
  }
  operationQueue.addOperation(sendOperation)
}
```

Background Task Completion

```
// Guarding Important Tasks While App is Still in the Foreground
func send(_ message: Message) {
  let sendOperation = SendOperation(message: message)
  var identifier: UIBackgroundTaskIdentifier!
  identifier = UIApplication.shared.beginBackgroundTask(expirationHandler: {
     sendOperation.cancel()
     postUserNotification("Message not sent, please resend")
     // Background task will be ended in the operation's completion block below
  })
  sendOperation.completionBlock = {
     UIApplication.shared.endBackgroundTask(identifier)
  }
  operationQueue.addOperation(sendOperation)
}
```

Background Task Completion

```
// Guarding Important Tasks While App is Still in the Foreground
func send(_ message: Message) {
  let sendOperation = SendOperation(message: message)
  var identifier: UIBackgroundTaskIdentifier!
  identifier = UIApplication.shared.beginBackgroundTask(expirationHandler: {
     sendOperation.cancel()
     postUserNotification("Message not sent, please resend")
     // Background task will be ended in the operation's completion block below
  })
  sendOperation.completionBlock = {
     UIApplication.shared.endBackgroundTask(identifier)
  }
  operationQueue.addOperation(sendOperation)
}
```

Defer the download until the better time

Discretionary Background URL Session

```
// Set up background URL session
let config = URLSessionConfiguration.background(withIdentifier: "com.app.attachments")
let session = URLSession(configuration: config, delegate: ..., delegateQueue: ...)

// Set discretionary
config.discretionary = true
```

Discretionary Background URL Session

```
// Set up background URL session
let config = URLSessionConfiguration.background(withIdentifier: "com.app.attachments")
let session = URLSession(configuration: config, delegate: ..., delegateQueue: ...)

// Set discretionary
config.discretionary = true
```

Discretionary Background URL Session

```
// Set up background URL session
let config = URLSessionConfiguration.background(withIdentifier: "com.app.attachments")
let session = URLSession(configuration: config, delegate: ..., delegateQueue: ...)

// Set discretionary
config.discretionary = true
```

Discretionary Background URL Session

```
// Set timeout intervals
config.timeoutIntervalForResource = 24 * 60 * 60
config.timeoutIntervalForRequest = 60

// Create request and task
var request = URLRequest(url: url)
request.addValue("...", forHTTPHeaderField: "...")
let task = session.downloadTask(with: request)

// Set time window
task.earliestBeginDate = Date(timeIntervalSinceNow: 2 * 60 * 60)

// Set workload size
task.countOfBytesClientExpectsToSend = 160
task.countOfBytesClientExpectsToReceive = 4096
task.resume()
```

Discretionary Background URL Session

```
// Set timeout intervals
config.timeoutIntervalForResource = 24 * 60 * 60
config.timeoutIntervalForRequest = 60

// Create request and task
var request = URLRequest(url: url)
request.addValue("...", forHTTPHeaderField: "...")
let task = session.downloadTask(with: request)

// Set time window
task.earliestBeginDate = Date(timeIntervalSinceNow: 2 * 60 * 60)

// Set workload size
task.countOfBytesClientExpectsToSend = 160
task.countOfBytesClientExpectsToReceive = 4096
task.resume()
```

Discretionary Background URL Session

```
// Set timeout intervals
config.timeoutIntervalForResource = 24 * 60 * 60
config.timeoutIntervalForRequest = 60

// Create request and task
var request = URLRequest(url: url)
request.addValue("...", forHTTPHeaderField: "...")
let task = session.downloadTask(with: request)

// Set time window
task.earliestBeginDate = Date(timeIntervalSinceNow: 2 * 60 * 60)

// Set workload size
task.countOfBytesClientExpectsToSend = 160
task.countOfBytesClientExpectsToReceive = 4096
task.resume()
```

Discretionary Background URL Session

```
// Set timeout intervals
config.timeoutIntervalForResource = 24 * 60 * 60
config.timeoutIntervalForRequest = 60

// Create request and task
var request = URLRequest(url: url)
request.addValue("...", forHTTPHeaderField: "...")
let task = session.downloadTask(with: request)

// Set time window
task.earliestBeginDate = Date(timeIntervalSinceNow: 2 * 60 * 60)

// Set workload size
task.countOfBytesClientExpectsToSend = 160
task.countOfBytesClientExpectsToReceive = 4096
task.resume()
```

Discretionary Background URL Session

```
// Set timeout intervals
config.timeoutIntervalForResource = 24 * 60 * 60
config.timeoutIntervalForRequest = 60

// Create request and task
var request = URLRequest(url: url)
request.addValue("...", forHTTPHeaderField: "...")
let task = session.downloadTask(with: request)

// Set time window
task.earliestBeginDate = Date(timeIntervalSinceNow: 2 * 60 * 60)

// Set workload size
task.countOfBytesClientExpectsToSend = 160
task.countOfBytesClientExpectsToReceive = 4096
task.resume()
```

Also in background: BackgroundTasks Framework

- Background Processing Tasks
- Background App Refresh Task

More: **Advances in App Background Execution, WWDC2019**

Monitor energy usage in the Debugger

NSSpainDemo 〉 @akomagorkina

Running NSSpainDemo on @akomagorkina

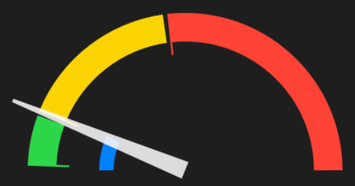
Energy Report

NSSpainDemo PID 1455

CPU	0%
Memory	21,6 MB
Energy Impact	High
Disk	Zero KB/s
Network	Zero KB/s

Energy

Average Energy Impact

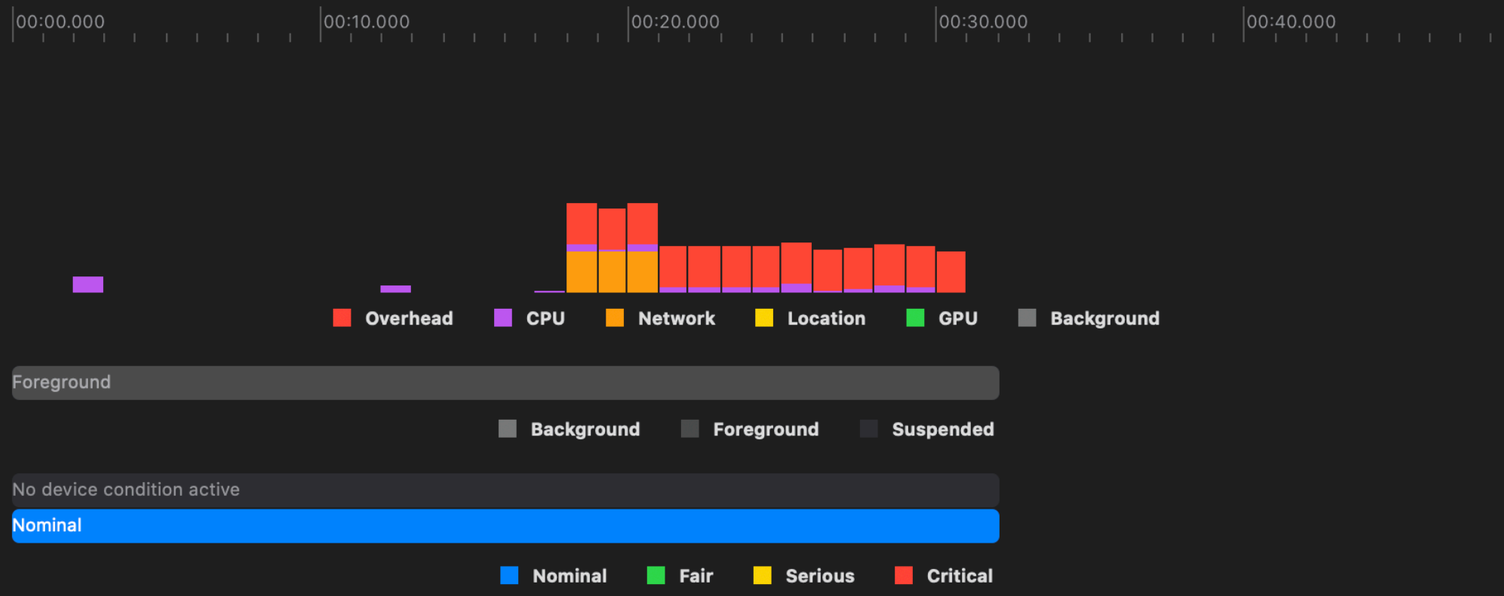


High
Energy Impact

Average Component Utilization

- **Overhead** 70,3 %
- **CPU** 13,4 %
- **Network** 16,3 %
- **Location** 0 %
- **GPU** 0 %

Energy Impact



Overhead

Overhead represents energy use as a result of bringing up radios and other system resources your app needs to perform work.

Energy Report

Energy

Average Energy Impact

Low

Energy Impact

Xcode instruments for optimisation

Measure energy impact with new Instruments:

Measure energy impact with new Instruments:

- **XCTest Metrics**
  Performance of measure blocks

Measure energy impact with new Instruments:

- **XCTest Metrics**
  Performance of measure blocks
- **MetricKit**
  Framework for battery and performance metrics collection

Measure energy impact with new Instruments:

- **XCTest Metrics**
  Performance of measure blocks
- **MetricKit**
  Framework for battery and performance metrics collection
- **Xcode Metrics Organizer**
  Aggregated battery, performance, and I/O metrics in Xcode

Collecting Metrics Using XCTest

```
// This test measures Launch Time
func testAppLaunchTime() {
  measure(metrics: [XCTOSSignpostMetric.applicationLaunch]) {
  XCUIApplication().launch()
 }
}
```

Adopting MetricKit to receive metrics

```
import MetricKit
// Conform to MXMetricManagerSubscriber protocol
extension AppDelegate: MXMetricManagerSubscriber {

    func subscribeToMetrics() { // Call from didFinishLaunching...
      let shared = MXMetricManager.shared
      shared.add(self)
    }

    // Receive daily metrics
    func didReceive(_ payloads: [MXMetricPayload]) {
      // Process metrics
    }
}
```

Adopting MetricKit to receive metrics

```
import MetricKit
// Conform to MXMetricManagerSubscriber protocol
extension AppDelegate: MXMetricManagerSubscriber {

    func subscribeToMetrics() { // Call from didFinishLaunching...
      let shared = MXMetricManager.shared
      shared.add(self)
    }

    // Receive daily metrics
    func didReceive(_ payloads: [MXMetricPayload]) {
      // Process metrics
    }
}
```

Adopting MetricKit to receive metrics

```
import MetricKit
// Conform to MXMetricManagerSubscriber protocol
extension AppDelegate: MXMetricManagerSubscriber {

    func subscribeToMetrics() { // Call from didFinishLaunching...
      let shared = MXMetricManager.shared
      shared.add(self)
    }

    // Receive daily metrics
    func didReceive(_ payloads: [MXMetricPayload]) {
      // Process metrics
    }
}
```

Adopting MetricKit to receive metrics

```
import MetricKit
// Conform to MXMetricManagerSubscriber protocol
extension AppDelegate: MXMetricManagerSubscriber {

    func subscribeToMetrics() { // Call from didFinishLaunching...
      let shared = MXMetricManager.shared
      shared.add(self)
    }

    // Receive daily metrics
    func didReceive(_ payloads: [MXMetricPayload]) {
      // Process metrics
    }
}
```

Adopting MetricKit to receive metrics

```
import MetricKit
// Conform to MXMetricManagerSubscriber protocol
extension AppDelegate: MXMetricManagerSubscriber {

    func subscribeToMetrics() { // Call from didFinishLaunching...
      let shared = MXMetricManager.shared
      shared.add(self)
    }

    // Receive daily metrics
    func didReceive(_ payloads: [MXMetricPayload]) {
      // Process metrics
    }
}
```

Adopting MetricKit to receive metrics

```
import MetricKit
// Conform to MXMetricManagerSubscriber protocol
extension AppDelegate: MXMetricManagerSubscriber {

    func subscribeToMetrics() { // Call from didFinishLaunching...
      let shared = MXMetricManager.shared
      shared.add(self)
    }

    // Receive daily metrics
    func didReceive(_ payloads: [MXMetricPayload]) {
      // Process metrics
    }
}
```

Adopting MetricKit to receive metrics

```
import MetricKit
// Conform to MXMetricManagerSubscriber protocol
extension AppDelegate: MXMetricManagerSubscriber {

    func subscribeToMetrics() { // Call from didFinishLaunching...
      let shared = MXMetricManager.shared
      shared.add(self)
    }

    // Receive daily metrics
    func didReceive(_ payloads: [MXMetricPayload]) {
      // Process metrics
    }
}
```

Adopting MetricKit to receive metrics

```
import MetricKit
// Conform to MXMetricManagerSubscriber protocol
extension AppDelegate: MXMetricManagerSubscriber {

    func subscribeToMetrics() { // Call from didFinishLaunching...
      let shared = MXMetricManager.shared
      shared.add(self)
    }

    // Receive daily metrics
    func didReceive(_ payloads: [MXMetricPayload]) {
      // Process metrics
    }
}
```

What else can we do better?

Adopt Low Data Mode

Suggested techniques for conforming to low data mode:

Suggested techniques for conforming to low data mode:

- Reduce image quality

Suggested techniques for conforming to low data mode:

- Reduce image quality
- Reduce pre-fetching (of unused or rarely used resources)

Suggested techniques for conforming to low data mode:

- Reduce image quality
- Reduce pre-fetching (of unused or rarely used resources)

Suggested techniques for conforming to low data mode:

- Reduce image quality
- Reduce pre-fetching (of unused or rarely used resources)
- Sync less often using locally cached data more heavily.

Suggested techniques for conforming to low data mode:

Suggested techniques for conforming to low data mode:

- Mark background tasks as discretionary

Suggested techniques for conforming to low data mode:

- Mark background tasks as discretionary

Suggested techniques for conforming to low data mode:

- Mark background tasks as discretionary
- Disable auto-play

Suggested techniques for conforming to low data mode:

- Mark background tasks as discretionary
- Disable auto-play

Suggested techniques for conforming to low data mode:

- Mark background tasks as discretionary
- Disable auto-play
- Do not block user-initiated work, even though low data mode is on.

And we can do even more


9:41
1x
Temperature
iPhone needs to cool down
before you can use it.
Emergency

Understanding device contidions

How does the heat damage the batteries?

🌡️ Hot temperatures can cause permanent damage to batteries.

⚡ Components like the voltage indicator can be affected by heat

🧪 As batteries heat up, chemical reactions inside will also occur faster

New tools in Xcode 11 for various temperatures

How to work with thermal state conditions?

How to work with thermal state conditions?

- Register for `ProcessInfo.thermalStateDidChangeNotification`

How to work with thermal state conditions?

- Register for `ProcessInfo.thermalStateDidChangeNotification`
- Use the `ProcessInfo.ThermalState` cases to react to thermal state changes

How to work with thermal state conditions?

- Register for `ProcessInfo.thermalStateDidChangeNotification`
- Use the `ProcessInfo.ThermalState` cases to react to thermal state changes
- Switch off background and unneeded functionality when thermal state is elevated

Actions on ProcessInfo.ThermalState

Thermal State	Recommendations	System Actions
Nominal	No corrective action needed	
Fair	Slightly elevated thermal state, apps can proactively start energy-saving measures	Photos analysis pauses
Serious	System performance is impacted, reduce CPU, GPU, and I/O usage	ARKit and FaceTime reduce FPS rate, Restore from iCloud is paused
Critical	Reduce CPU, GPU, and I/O usage, and stop using peripherals such as camera	ARKit and FaceTime drop FPS rate

Subscribe to thermal state condition changes

```
NotificationCenter.default.addObserver(
      self,
      selector: #selector(reactToThermalStateChange(_:)),
      name: ProcessInfo.thermalStateDidChangeNotification,
      object: nil
)

@objc
func reactToThermalStateChange(_ notification: Notification) {
  print(ProcessInfo.processInfo.thermalState)
}
```

Subscribe to thermal state condition changes

```
NotificationCenter.default.addObserver(
      self,
      selector: #selector(reactToThermalStateChange(_:)),
      name: ProcessInfo.thermalStateDidChangeNotification,
      object: nil
)

@objc
func reactToThermalStateChange(_ notification: Notification) {
  print(ProcessInfo.processInfo.thermalState)
}
```

```
var thermalState = ProcessInfo.ThermalState.nominal {
  didSet {
   switch thermalState {
      case .nominal, .fair: // All good
         configuration.userFaceTrackingEnabled = true
         sceneView.rendersMotionBlur = true
      case .serious: // Something went wrong
         configuration.userFaceTrackingEnabled = false
         sceneView.rendersMotionBlur = true
      case .critical: // PANIC
         configuration.userFaceTrackingEnabled = false
         sceneView.rendersMotionBlur = false
      }
    }
 }
```

```
var thermalState = ProcessInfo.ThermalState.nominal {
  didSet {
    switch thermalState {
      case .nominal, .fair: // All good
        configuration.userFaceTrackingEnabled = true
        sceneView.rendersMotionBlur = true
      case .serious: // Something went wrong
        configuration.userFaceTrackingEnabled = false
        sceneView.rendersMotionBlur = true
      case .critical: // PANIC
        configuration.userFaceTrackingEnabled = false
        sceneView.rendersMotionBlur = false
      }
    }
  }
```

```swift
var thermalState = ProcessInfo.ThermalState.nominal {
  didSet {
   switch thermalState {
      case .nominal, .fair: // All good
         configuration.userFaceTrackingEnabled = true
         sceneView.rendersMotionBlur = true
      case .serious: // Something went wrong
         configuration.userFaceTrackingEnabled = false
         sceneView.rendersMotionBlur = true
      case .critical: // PANIC
         configuration.userFaceTrackingEnabled = false
         sceneView.rendersMotionBlur = false
      }
    }
 }
```

```swift
var thermalState = ProcessInfo.ThermalState.nominal {
  didSet {
   switch thermalState {
      case .nominal, .fair: // All good
         configuration.userFaceTrackingEnabled = true
         sceneView.rendersMotionBlur = true
      case .serious: // Something went wrong
         configuration.userFaceTrackingEnabled = false
         sceneView.rendersMotionBlur = true
      case .critical: // PANIC
         configuration.userFaceTrackingEnabled = false
         sceneView.rendersMotionBlur = false
      }
    }
 }
```

```
var thermalState = ProcessInfo.ThermalState.nominal {
  didSet {
   switch thermalState {
      case .nominal, .fair: // All good
         configuration.userFaceTrackingEnabled = true
         sceneView.rendersMotionBlur = true
      case .serious: // Something went wrong
         configuration.userFaceTrackingEnabled = false
         sceneView.rendersMotionBlur = true
      case .critical: // PANIC
         configuration.userFaceTrackingEnabled = false
         sceneView.rendersMotionBlur = false
      }
    }
 }
```

```swift
var thermalState = ProcessInfo.ThermalState.nominal {
  didSet {
   switch thermalState {
      case .nominal, .fair: // All good
         configuration.userFaceTrackingEnabled = true
         sceneView.rendersMotionBlur = true
      case .serious: // Something went wrong
         configuration.userFaceTrackingEnabled = false
         sceneView.rendersMotionBlur = true
      case .critical: // PANIC
         configuration.userFaceTrackingEnabled = false
         sceneView.rendersMotionBlur = false
      }
   }
 }
```

To sum up

To sum up

- Think about Background

To sum up

- Think about Background
- Minimize Networking

To sum up

- Think about Background
- Minimize Networking
- Measure as much as possible

With great power comes great responsibility

Further links & reading

Further links & reading

- @StuFFmc at iOSDevUK 2019: Save the environment with Xcode

Further links & reading

- @StuFFmc at iOSDevUK 2019: Save the environment with Xcode
- WWDC2019: Improving Battery Life and Performance

Further links & reading

- @StuFFmc at iOSDevUK 2019: Save the environment with Xcode
- WWDC2019: Improving Battery Life and Performance
- WWDC2019: Advances in App Background Execution

Further links & reading

- @StuFFmc at iOSDevUK 2019: Save the environment with Xcode
- WWDC2019: Improving Battery Life and Performance
- WWDC2019: Advances in App Background Execution
- WWDC2019:Designing for Adverse Network and Temperature Conditions

Further links & reading

- @StuFFmc at iOSDevUK 2019: Save the environment with Xcode
- WWDC2019: Improving Battery Life and Performance
- WWDC2019: Advances in App Background Execution
- WWDC2019:Designing for Adverse Network and Temperature Conditions
- Energy Efficiency Guide for iOS Apps

Thank you! 🍏

@akomagorkina