



# Reimplementing CoAP for C# with the Task-based Asynchronous Pattern

Is it worth to await?

Philip Wille

# Introduction

- Synchronous and asynchronous execution

# Introduction

- Synchronous and asynchronous execution
  - Synchronous:
  - Asynchronous:

# Introduction

- Synchronous and asynchronous execution
  - Synchronous:
    - **Waiting for completion** of method before continuing with program flow.
  - Asynchronous:

# Introduction

- Synchronous and asynchronous execution
  - Synchronous:
    - **Waiting for completion** of method before continuing with program flow.
  - Asynchronous:
    - Can **perform other tasks** while the execution is running.

# Introduction

- Synchronous and asynchronous execution
  - Synchronous:
    - **Waiting for completion** of method before continuing with program flow.
    - **Busy waiting** → Thread is marked as **blocked**.
  - Asynchronous:
    - Can **perform other tasks** while the execution is running.

# Introduction

- Synchronous and asynchronous execution
  - Synchronous:
    - **Waiting for completion** of method before continuing with program flow.
    - **Busy waiting** → Thread is marked as **blocked**.
  - Asynchronous:
    - Can **perform other tasks** while the execution is running.
    - **No busy waiting** → Thread is **free** for other tasks.

# Introduction

- Synchronous and asynchronous execution
  - Synchronous:
    - **Waiting for completion** of method before continuing with program flow.
    - **Busy waiting** → Thread is marked as **blocked**.
  - Asynchronous:
    - Can **perform other tasks** while the execution is running.
    - **No busy waiting** → Thread is **free** for other tasks.
- **T**ask-based **A**synchronous **P**attern (TAP)



# Introduction

- Synchronous and asynchronous execution
  - Synchronous:
    - **Waiting for completion** of method before continuing with program flow.
    - **Busy waiting** → Thread is marked as **blocked**.
  - Asynchronous:
    - Can **perform other tasks** while the execution is running.
    - **No busy waiting** → Thread is **free** for other tasks.
- **T**ask-based **A**synchronous **P**attern (TAP)
  - Developed by Microsoft.

# Introduction

- Synchronous and asynchronous execution
  - Synchronous:
    - **Waiting for completion** of method before continuing with program flow.
    - **Busy waiting** → Thread is marked as **blocked**.
  - Asynchronous:
    - Can **perform other tasks** while the execution is running.
    - **No busy waiting** → Thread is **free** for other tasks.
- **T**ask-based **A**synchronous **P**attern (TAP)
  - Developed by Microsoft.
  - Simple usage.

# Introduction

- Synchronous and asynchronous execution
  - Synchronous:
    - **Waiting for completion** of method before continuing with program flow.
    - **Busy waiting** → Thread is marked as **blocked**.
  - Asynchronous:
    - Can **perform other tasks** while the execution is running.
    - **No busy waiting** → Thread is **free** for other tasks.
- **T**ask-based **A**synchronous **P**attern (TAP)
  - Developed by Microsoft.
  - Simple usage.
  - Built-in in C#.

# Introduction

- Synchronous and asynchronous execution
  - Synchronous:
    - **Waiting for completion** of method before continuing with program flow.
    - **Busy waiting** → Thread is marked as **blocked**.
  - Asynchronous:
    - Can **perform other tasks** while the execution is running.
    - **No busy waiting** → Thread is **free** for other tasks.
- **T**ask-based **A**synchronous **P**attern (TAP)
  - Developed by Microsoft.
  - Simple usage.
  - Built-in in C#.
- **C**onstrained **A**pplication **P**rotocol (CoAP)

# Introduction

- Synchronous and asynchronous execution
  - Synchronous:
    - **Waiting for completion** of method before continuing with program flow.
    - **Busy waiting** → Thread is marked as **blocked**.
  - Asynchronous:
    - Can **perform other tasks** while the execution is running.
    - **No busy waiting** → Thread is **free** for other tasks.
- **T**ask-based **A**synchronous **P**attern (TAP)
  - Developed by Microsoft.
  - Simple usage.
  - Built-in in C#.
- **C**onstrained **A**pplication **P**rotocol (CoAP)
  - Subset of **H**ypertext **T**ransport **P**rotocol (HTTP).

# Introduction

- Synchronous and asynchronous execution
  - Synchronous:
    - **Waiting for completion** of method before continuing with program flow.
    - **Busy waiting** → Thread is marked as **blocked**.
  - Asynchronous:
    - Can **perform other tasks** while the execution is running.
    - **No busy waiting** → Thread is **free** for other tasks.
- **Task-based Asynchronous Pattern (TAP)**
  - Developed by Microsoft.
  - Simple usage.
  - Built-in in C#.
- **Constrained Application Protocol (CoAP)**
  - Subset of **Hypertext Transport Protocol (HTTP)**.
  - Specialized for **Internet of Things (IoT)** and **Machine-to-Machine (M2M)** devices.

# Synchronous execution

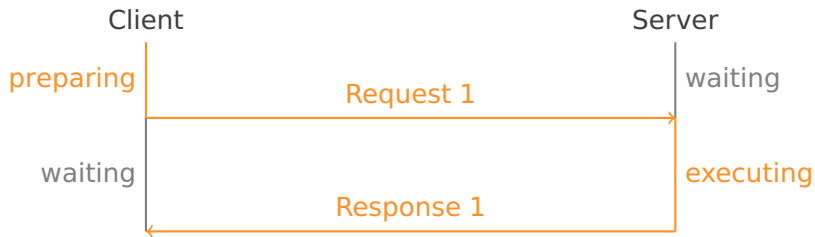


# Synchronous execution





# Synchronous execution



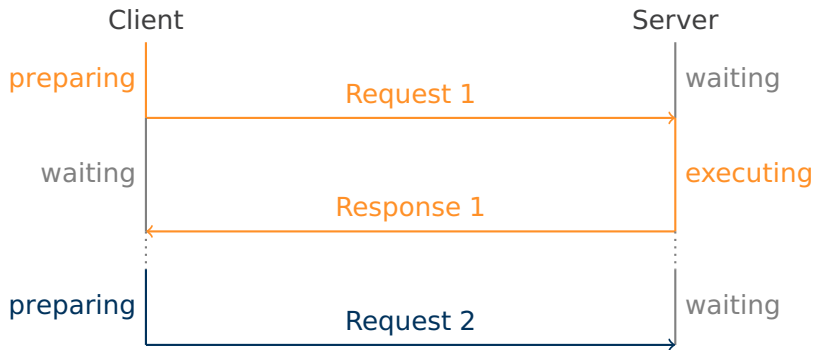
# Synchronous execution



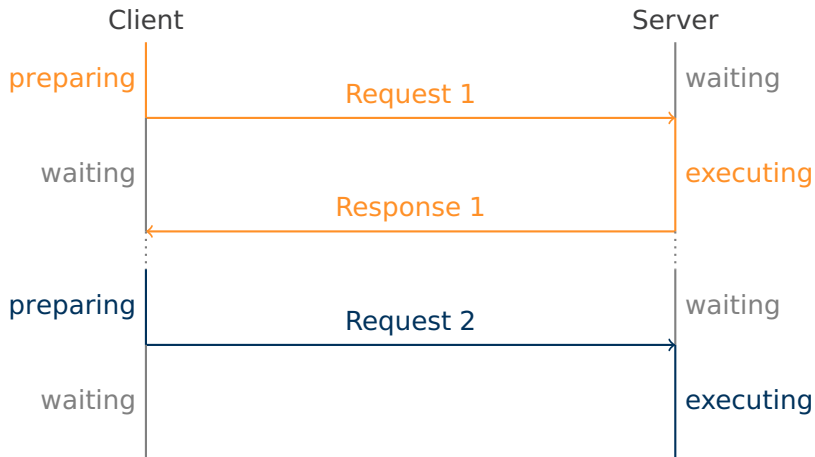
# Synchronous execution



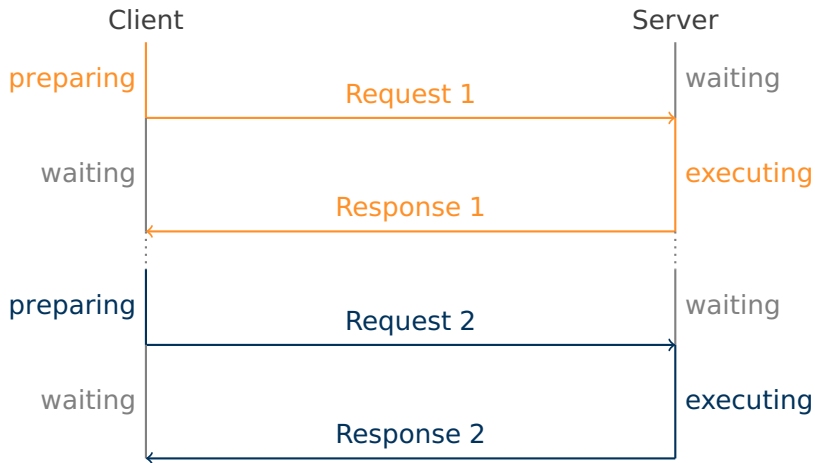
# Synchronous execution



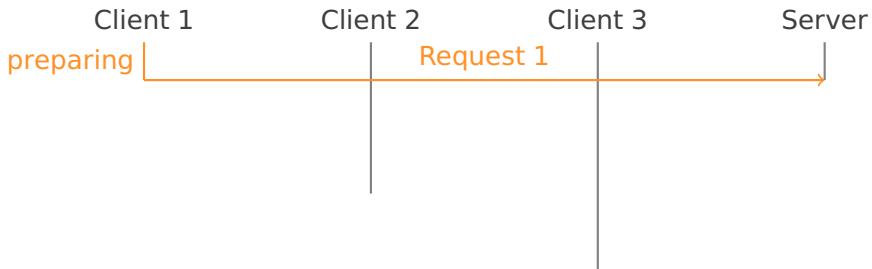
# Synchronous execution



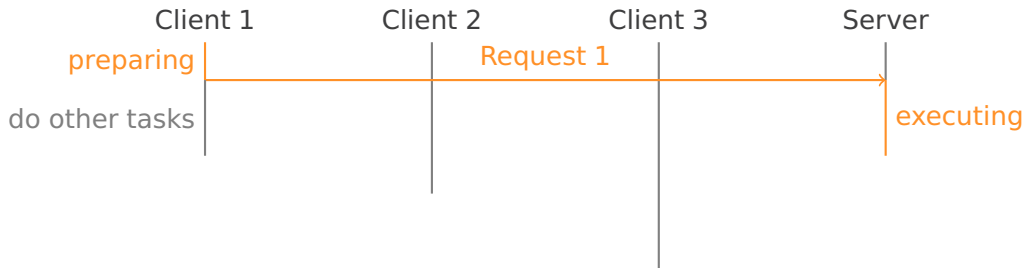
# Synchronous execution



# Asynchronous execution

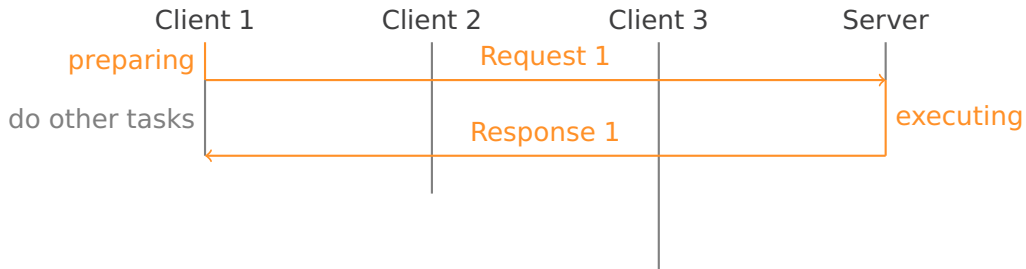


# Asynchronous execution

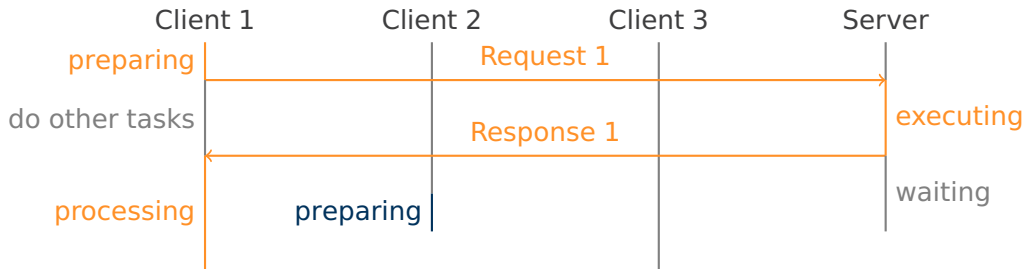




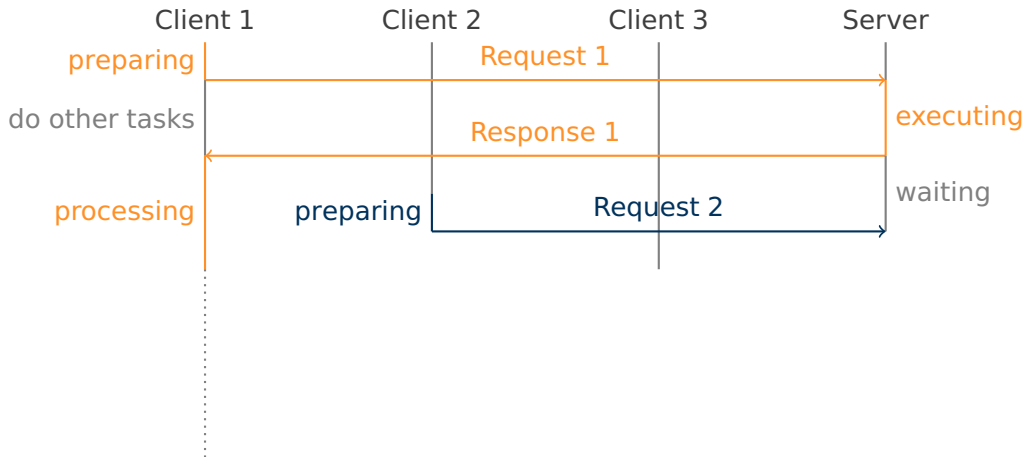
# Asynchronous execution



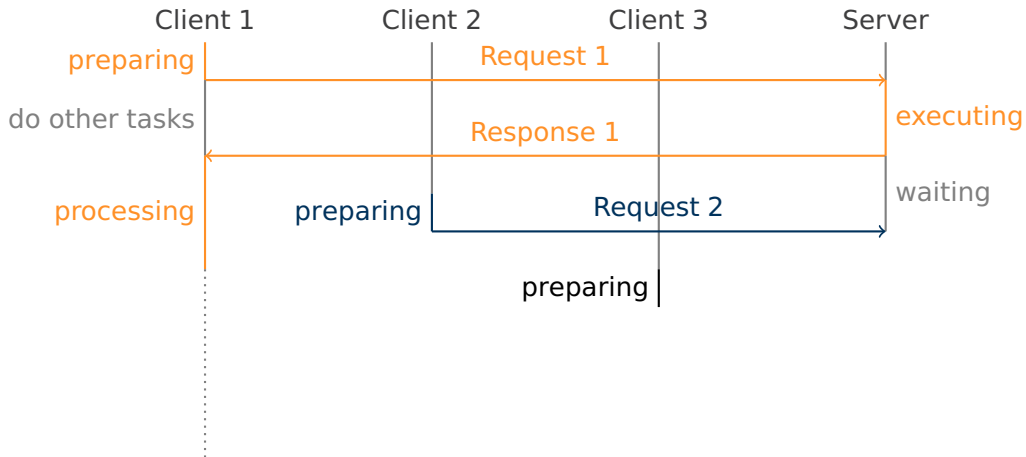
# Asynchronous execution



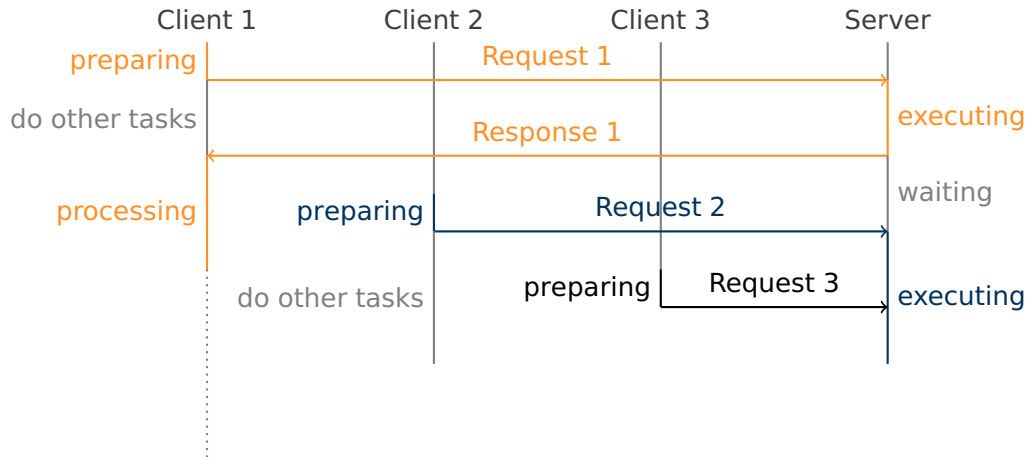
# Asynchronous execution



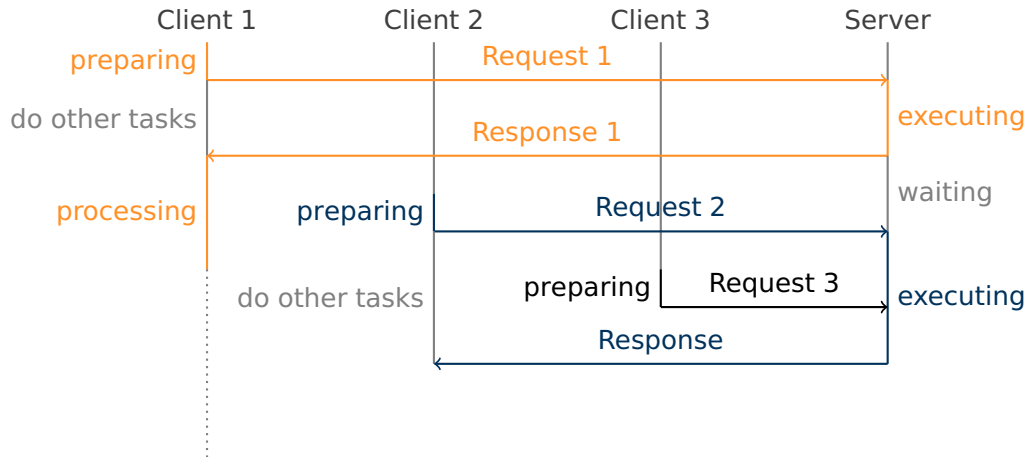
# Asynchronous execution



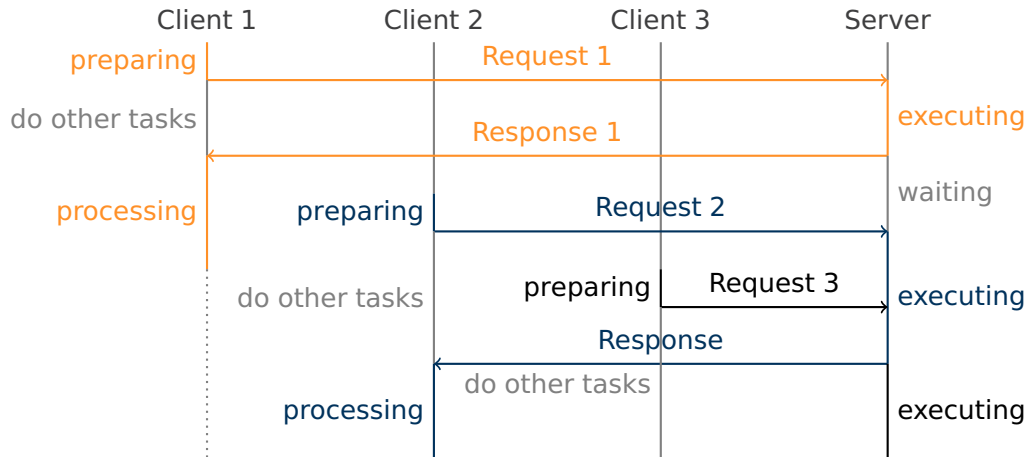
# Asynchronous execution



# Asynchronous execution



# Asynchronous execution



# Asynchronous execution

