universität innsbruck

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** $\rightarrow$ 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** $\rightarrow$ Thread is marked as **blocked**.
  - Asynchronous:
    - Can **perform other tasks** while the execution is running.
    - The main thread will be **notified**.

Introduction

- Synchronous and asynchronous execution
  - Synchronous:
    - **Waiting for completion** of method before continuing with program flow.
    - **Busy waiting** $\rightarrow$ Thread is marked as **blocked**.
  - Asynchronous:
    - Can **perform other tasks** while the execution is running.
    - The main thread will be **notified**.
    - **No busy waiting** $\rightarrow$ Thread is **free** for other tasks.

Introduction

- Synchronous and asynchronous execution
  - Synchronous:
    - **Waiting for completion** of method before continuing with program flow.
    - **Busy waiting** $\rightarrow$ Thread is marked as **blocked**.
  - Asynchronous:
    - Can **perform other tasks** while the execution is running.
    - The main thread will be **notified**.
    - **No busy waiting** $\rightarrow$ 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** $\rightarrow$ Thread is marked as **blocked**.
  - Asynchronous:
    - Can **perform other tasks** while the execution is running.
    - The main thread will be **notified**.
    - **No busy waiting** $\rightarrow$ 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** $\rightarrow$ Thread is marked as **blocked**.
  - Asynchronous:
    - Can **perform other tasks** while the execution is running.
    - The main thread will be **notified**.
    - **No busy waiting** $\rightarrow$ 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** $\rightarrow$ Thread is marked as **blocked**.
  - Asynchronous:
    - Can **perform other tasks** while the execution is running.
    - The main thread will be **notified**.
    - **No busy waiting** $\rightarrow$ 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.
    - The main thread will be **notified**.
    - **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#.
- **Co**nstrained **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.
    - The main thread will be **notified**.
    - **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#.
- **Co**nstrained **A**pplication **P**rotocol (CoAP)
  - Subset of **H**yper**t**ext **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.
    - The main thread will be **notified**.
    - **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#.
- **Co**nstrained **A**pplication **P**rotocol (CoAP)
  - Subset of **H**yper**t**ext **T**ransport **P**rotocol (HTTP).
  - Specialized for **I**nternet **o**f **T**hings (IoT) and **M**achine-**t**o-**M**achine (M2M) devices.

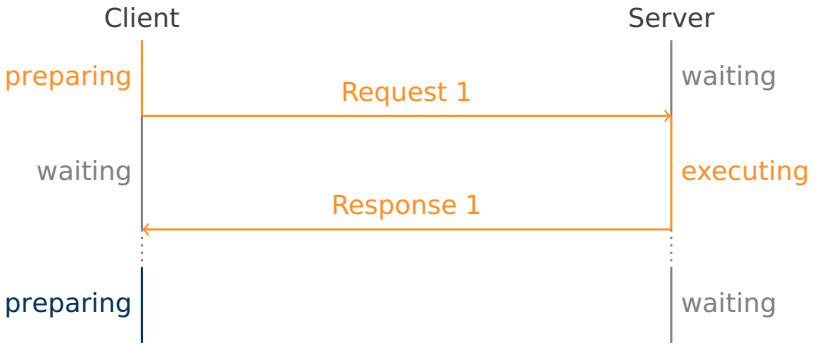
Synchronous execution

Synchronous execution

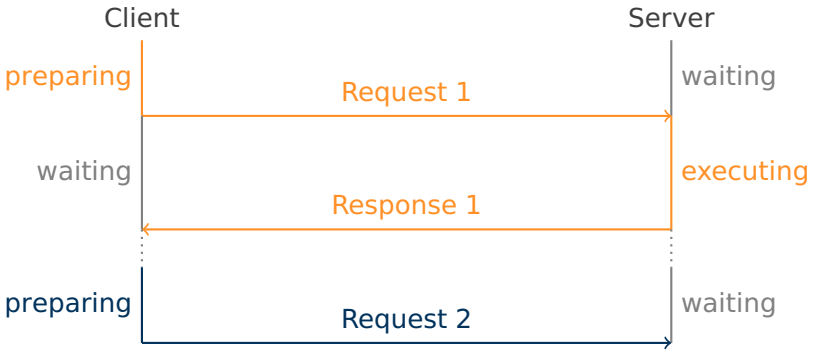
Synchronous execution

Synchronous execution

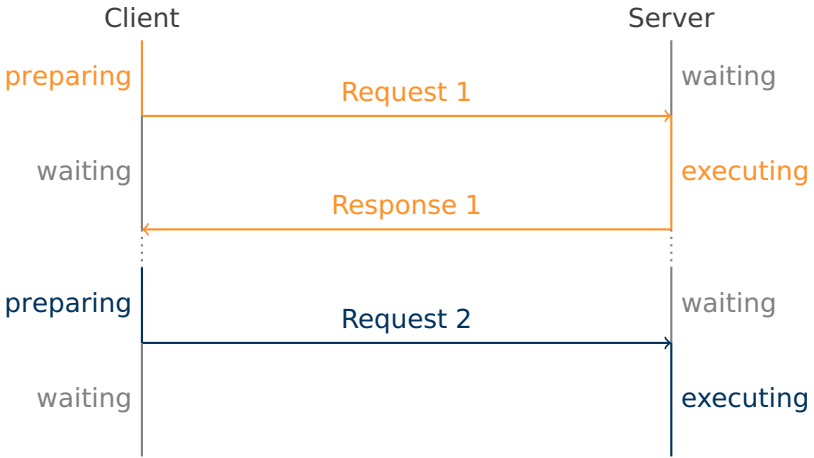
Synchronous execution



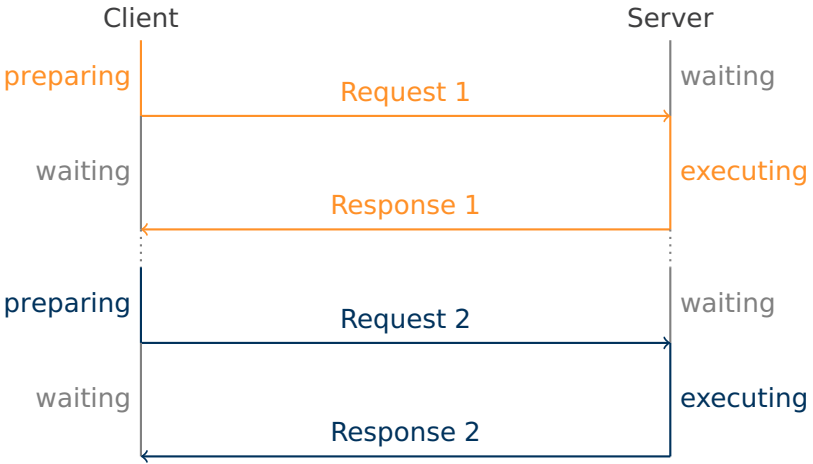
Synchronous execution



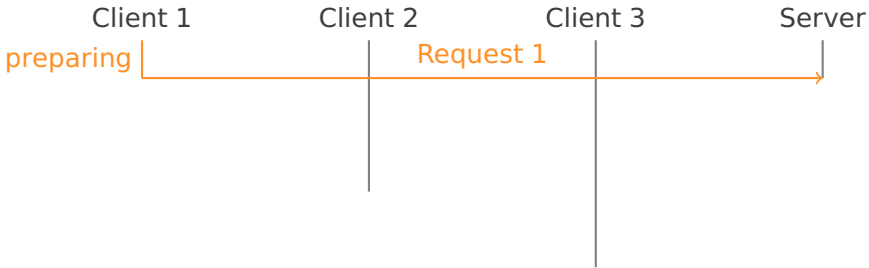
Synchronous execution



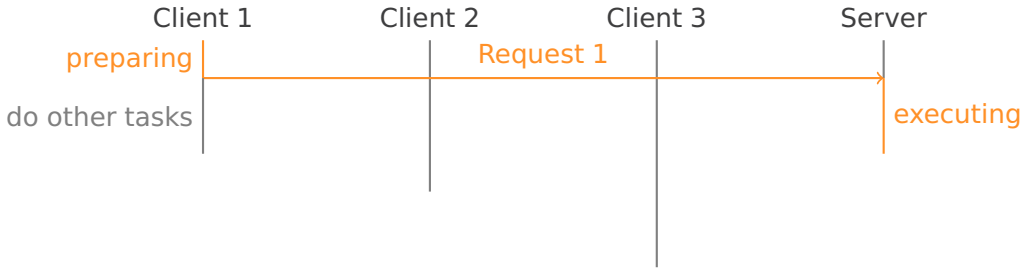
Synchronous execution



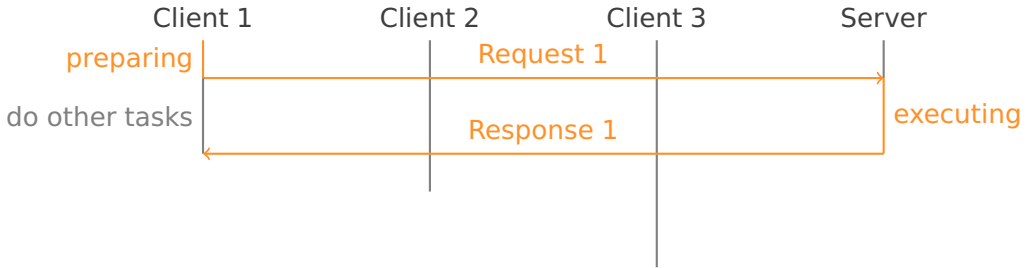
Asynchronous execution



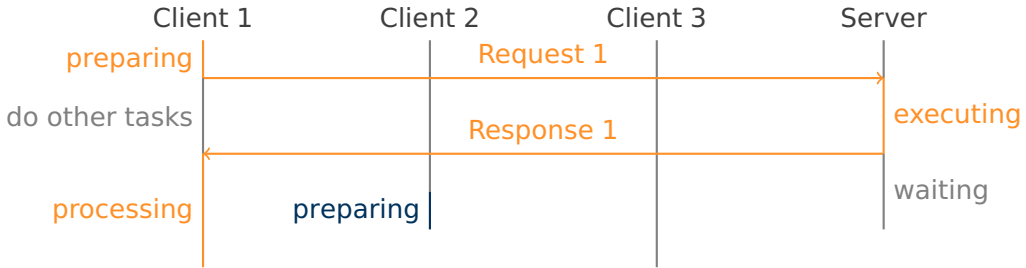
Asynchronous execution



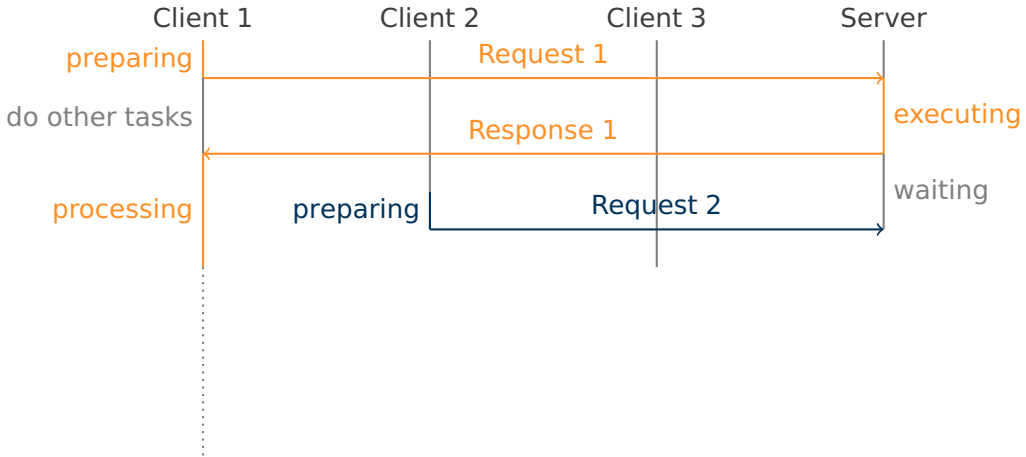
Asynchronous execution



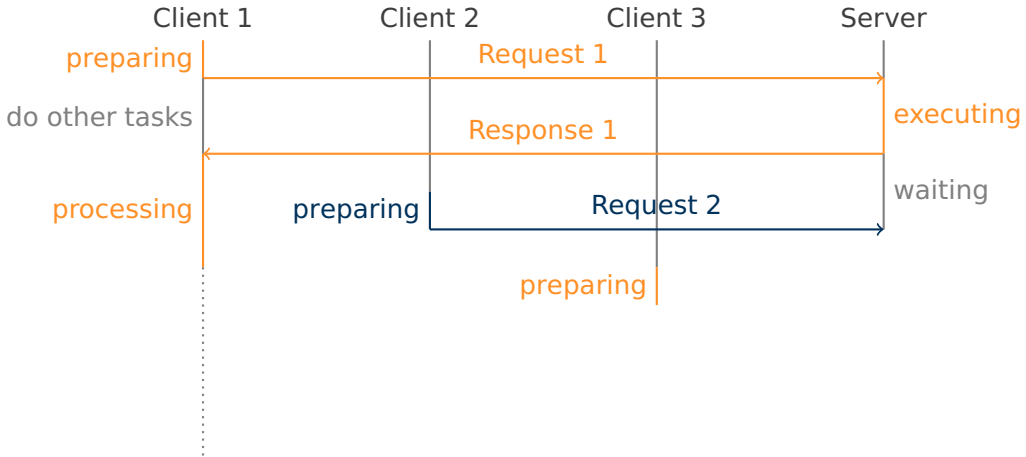
Asynchronous execution



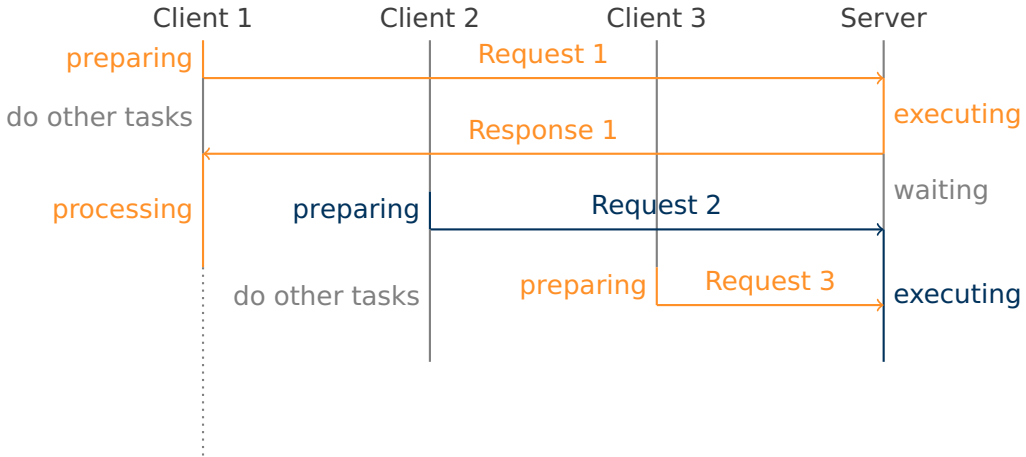
Asynchronous execution



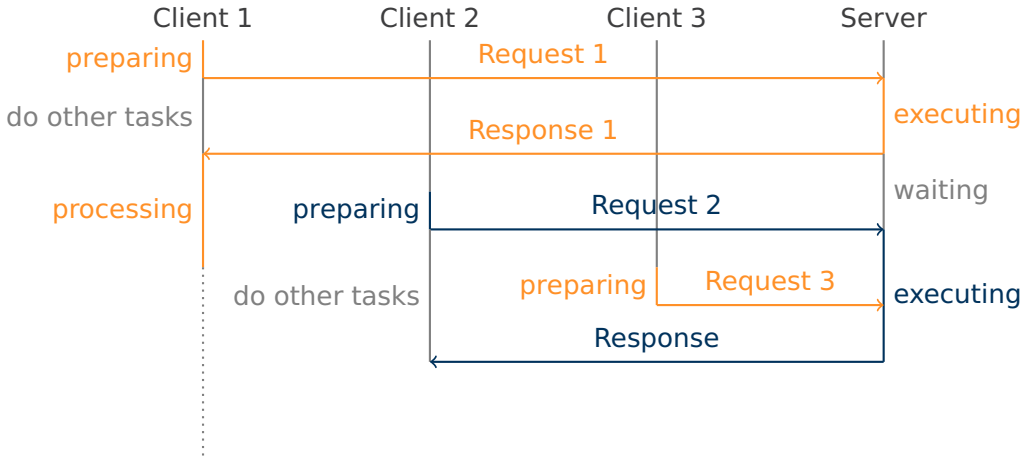
Asynchronous execution



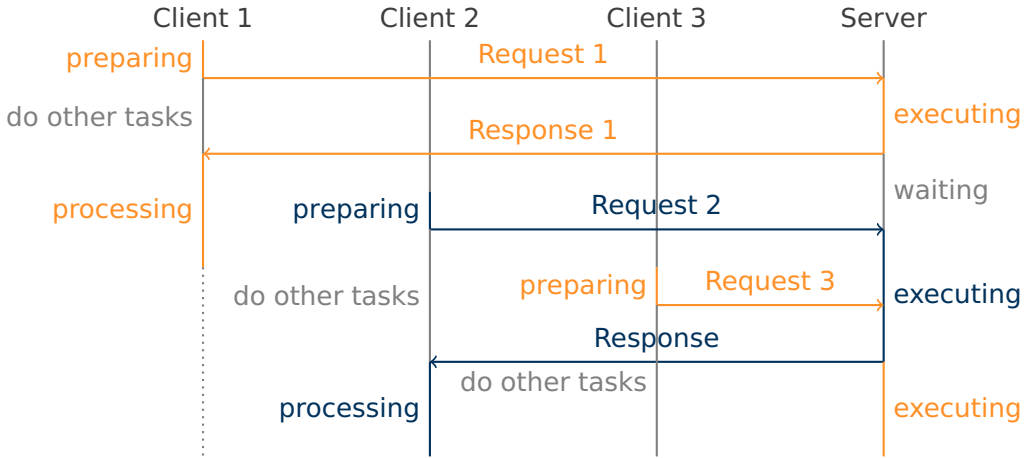
Asynchronous execution



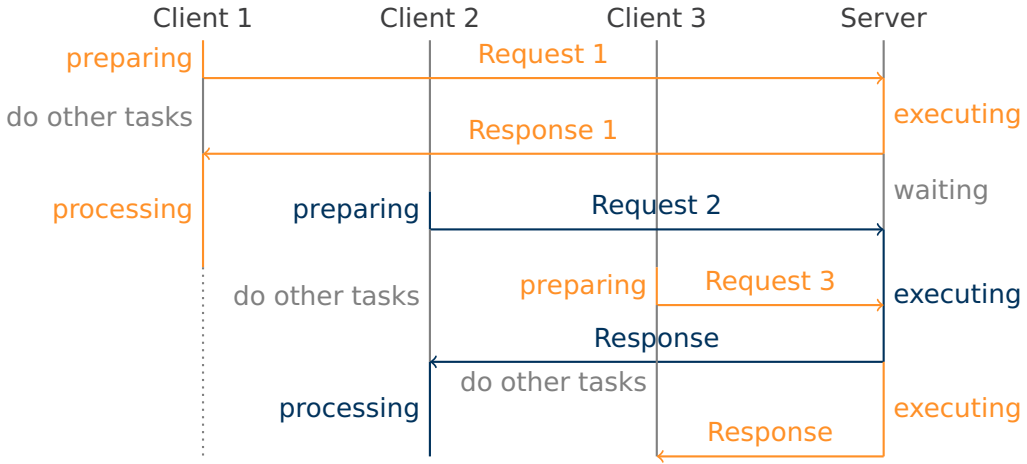
Asynchronous execution



Asynchronous execution



Asynchronous execution



Asynchronous execution in Java

```
1  public CompletableFuture<Integer> calculateAsync() throws InterruptedException {
2      CompletableFuture<Integer> completableFuture = CompletableFuture.supplyAsync(() -> {
3          Thread.sleep(1000);
4          return 1;
5      });
6      return completableFuture;
7  }
8
9  public static void main(string[] args) {
10     CompletableFuture cf = calculateAsync();
11     while (!cf.isDone) {
12         System.out.println("CompleteableFuture is not finished yet...");
13     }
14     long result = cf.get();
15 }
```

Listing 1: Asynchronous usage in Java

Asynchronous execution in C#

```
1  public async Task<int> CalculateAsync() {
2      await Task.Delay(1000).ConfigureAwait(false);
3      return Task.FromResult(1);
4  }
5
6  public static Task Main(string[] args) {
7      var result = await CalculateAsync().ConfigureAwait(false);
8  }
```

Listing 2: Asynchronous usage in C#

Short introduction into CoAP

- URI scheme based
  - "coap:" "//" host [":" port] path-abempty ["?" query]
  - "coaps:" "//" host [":" port] path-abempty ["?" query]
- REST-like
  - GET, PUT, POST, DELETE, PATCH, ...
- Specialized for using with constrained nodes and constrained networks.
- Works with HTTP.
- Fulfills requirements of environments like energy, building automation, and other machine-to-machine (M2M) applications.
- Several implementations for many programming languages like C# (CoAP.NET), Java (Californium), Python (CoAPthon), C (FreeCoAP) ...

Example request in CoAP

Figure: GET request

Example request in CoAP

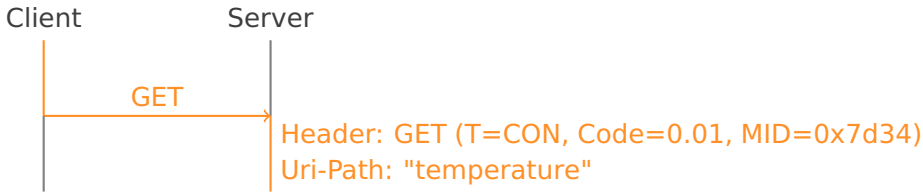


Figure: GET request

Example request in CoAP

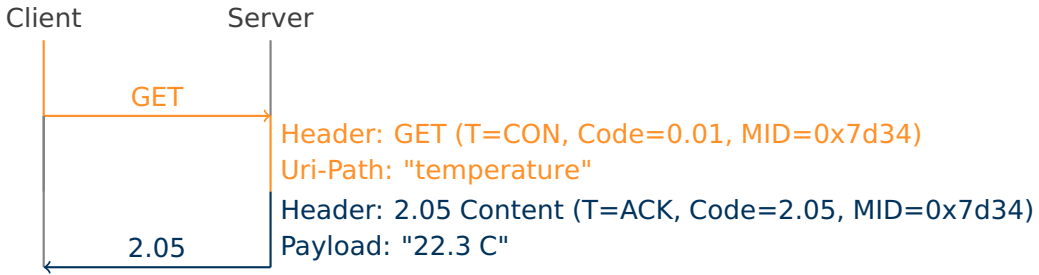


Figure: GET request

Example request in CoAP

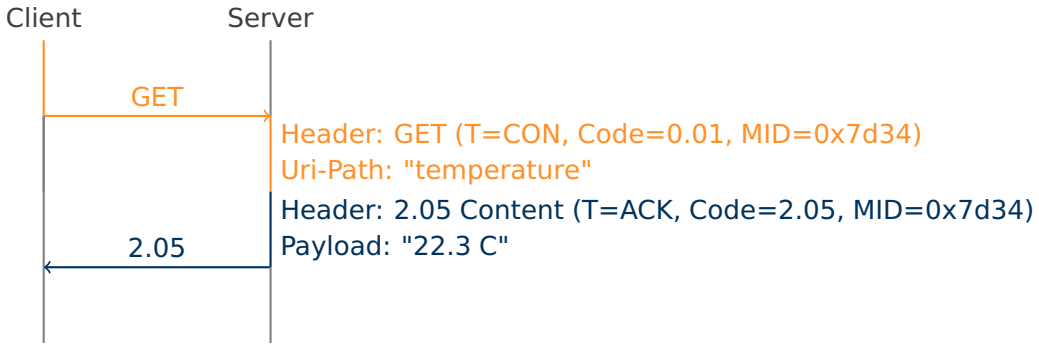


Figure: GET request

Example request in CoAP

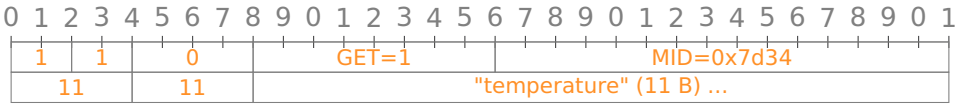


Figure: Confirmable Request

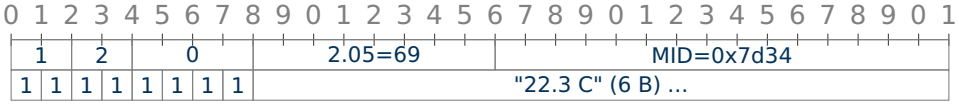


Figure: Piggybacked Response

CoAP.NET

- Implementation of CoAP for C#.
- Development inactive.
- Partially asynchronous.
- Memory leaks.
- Poor diagnostic capabilities.
- Only .NET Framework.

Goal thesis

- Goals
  - Rewrite CoAP.NET to fully asynchronous version.
  - Fixing memory leaks.
  - Enhancing diagnostic capabilities.
  - Upgrading to .NET Standard 2.0.
  - Several improvements.
- Supervisors
  - assoc. Prof. Dr. Michael Felderer (University Innsbruck).
  - Andreas Dânek (World-Direct eBusiness solutions GmbH)