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** → 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.
  - Small changes to synchronous code for enabling asynchronous execution.

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.
  - Small changes to synchronous code for enabling asynchronous execution.
  - 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.
  - Small changes to synchronous code for enabling asynchronous execution.
  - Built-in in C#.
- **Co**nstrained **A**pplication **P**rotocol (CoAP)

Introduction

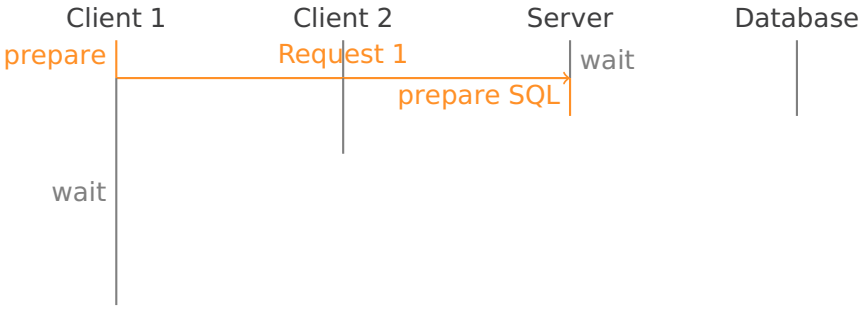
- Synchronous and asynchronous execution
  - Synchronous:
    - **Waiting for completion** of method before continuing with program flow.
    - **Busy waiting** → Thread is marked as **blocked**.
  - Asynchronous:
    - Can **perform other tasks** while the execution is running.
    - The main thread will be **notified**.
    - **No busy waiting** → Thread is **free** for other tasks.
- **T**ask-based **A**synchronous **P**attern (TAP)
  - Developed by Microsoft.
  - Small changes to synchronous code for enabling asynchronous execution.
  - Built-in in C#.
- **Co**nstrained **A**pplication **P**rotocol (CoAP)
  - Designed for **M**achine-**t**o-**M**achine (M2M) devices.

Introduction

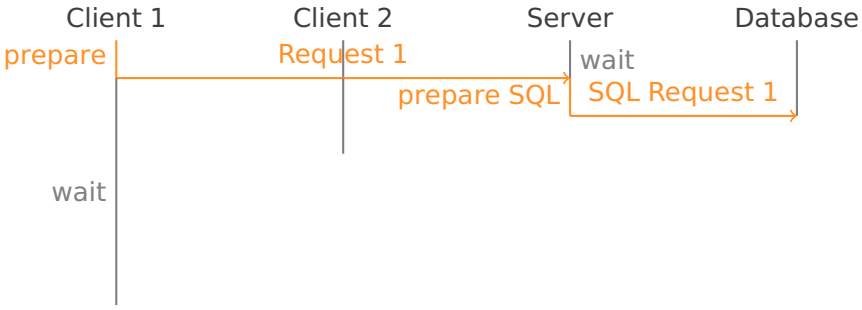
- Synchronous and asynchronous execution
  - Synchronous:
    - **Waiting for completion** of method before continuing with program flow.
    - **Busy waiting** → Thread is marked as **blocked**.
  - Asynchronous:
    - Can **perform other tasks** while the execution is running.
    - The main thread will be **notified**.
    - **No busy waiting** → Thread is **free** for other tasks.
- **T**ask-based **A**synchronous **P**attern (TAP)
  - Developed by Microsoft.
  - Small changes to synchronous code for enabling asynchronous execution.
  - Built-in in C#.
- **Co**nstrained **A**pplication **P**rotocol (CoAP)
  - Designed for **M**achine-**t**o-**M**achine (M2M) devices.
  - Request/response interaction model.

Synchronous server

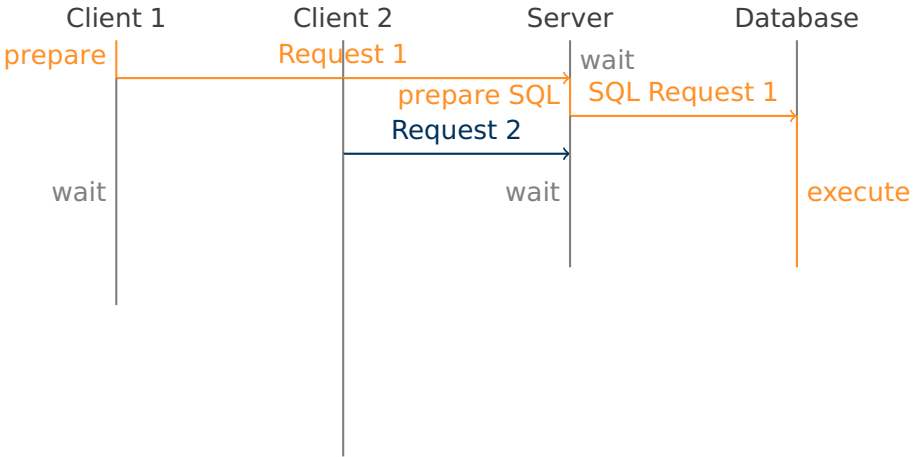
Synchronous server



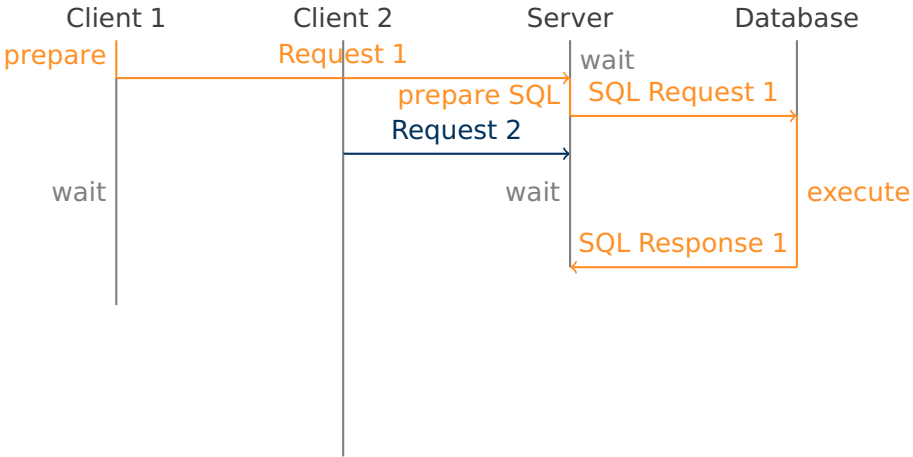
Synchronous server



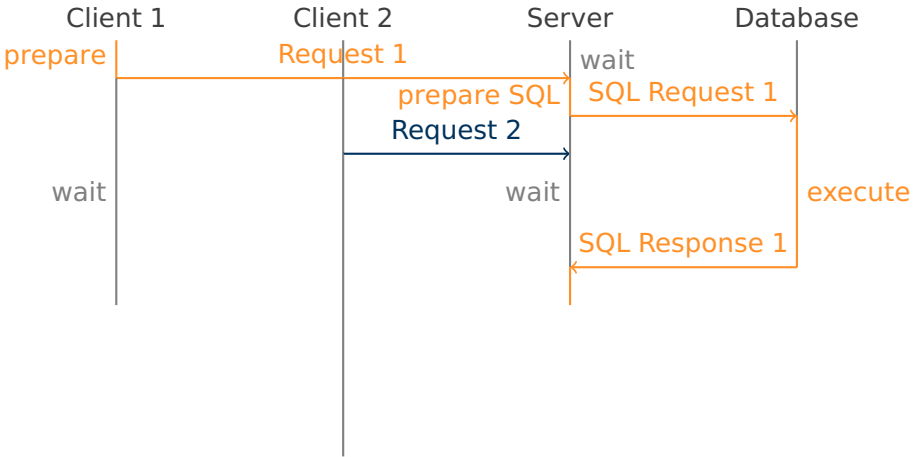
Synchronous server



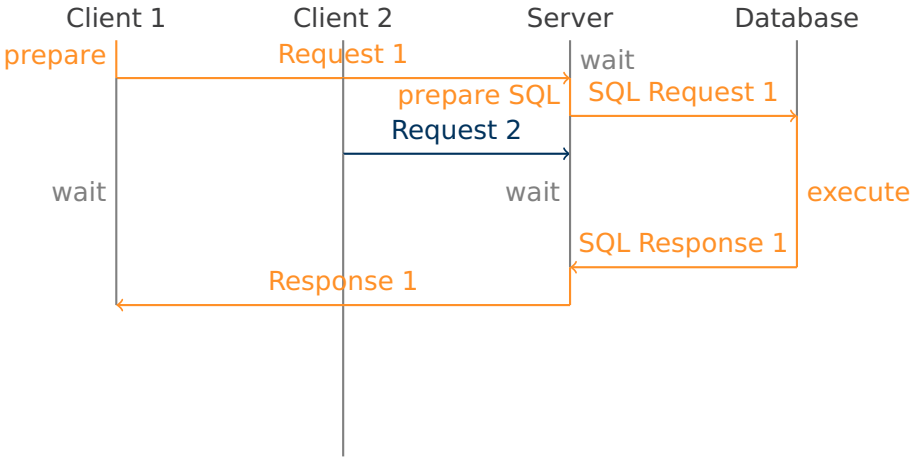
Synchronous server



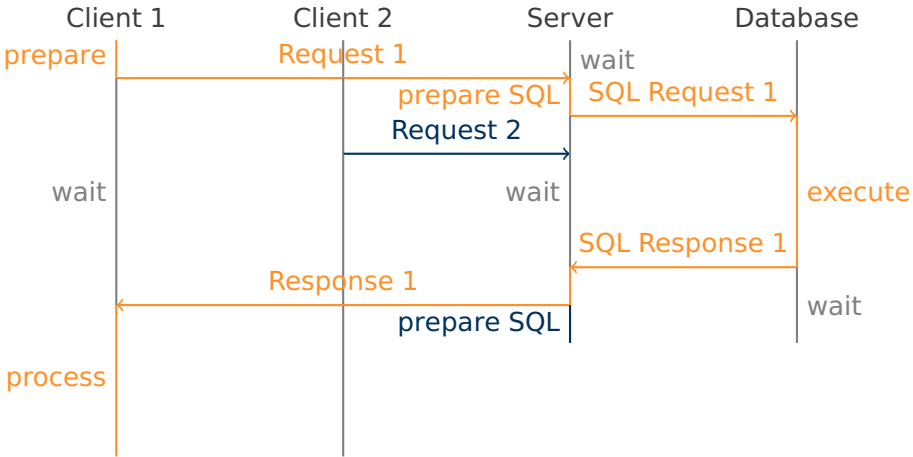
Synchronous server



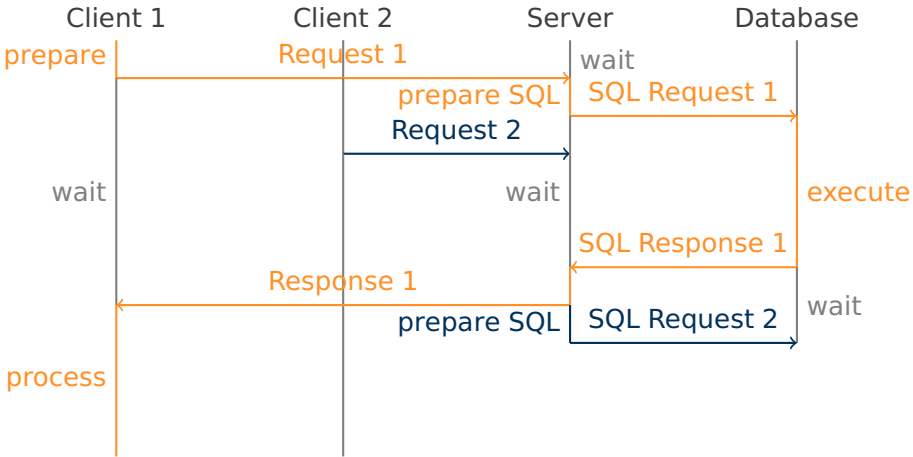
Synchronous server



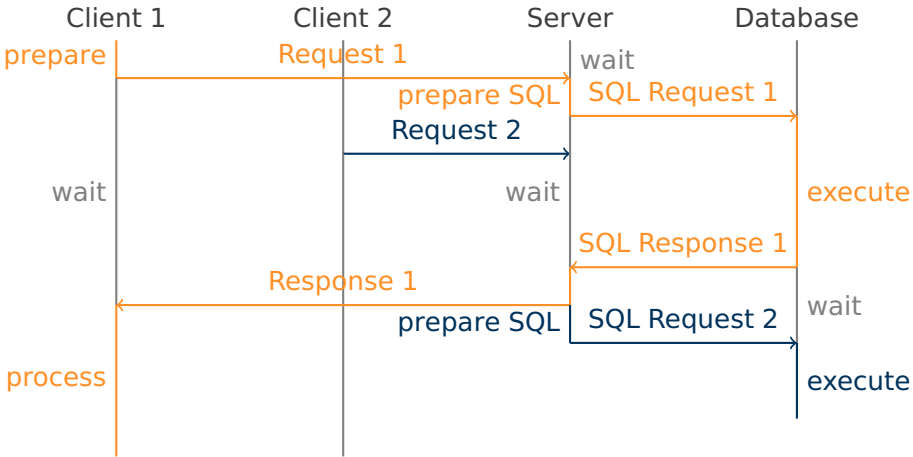
Synchronous server



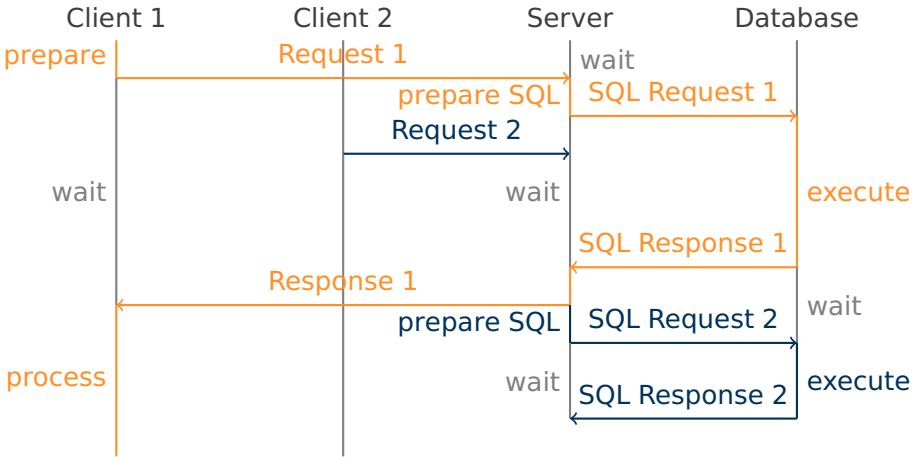
Synchronous server



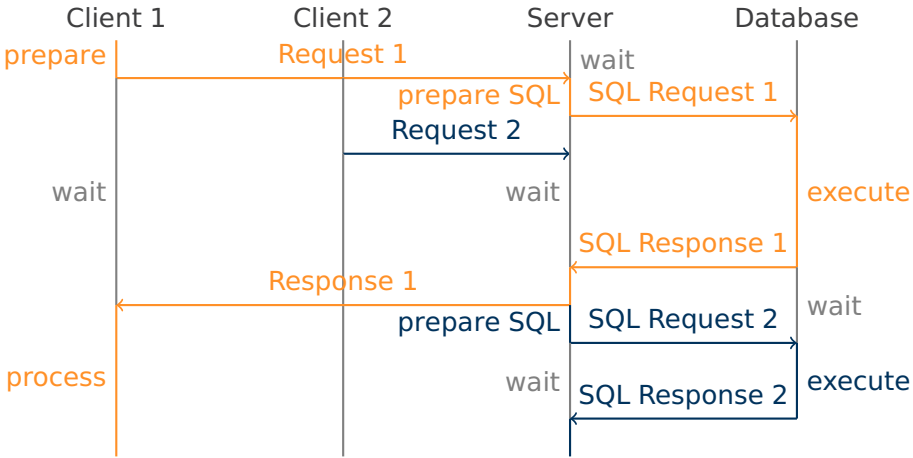
Synchronous server



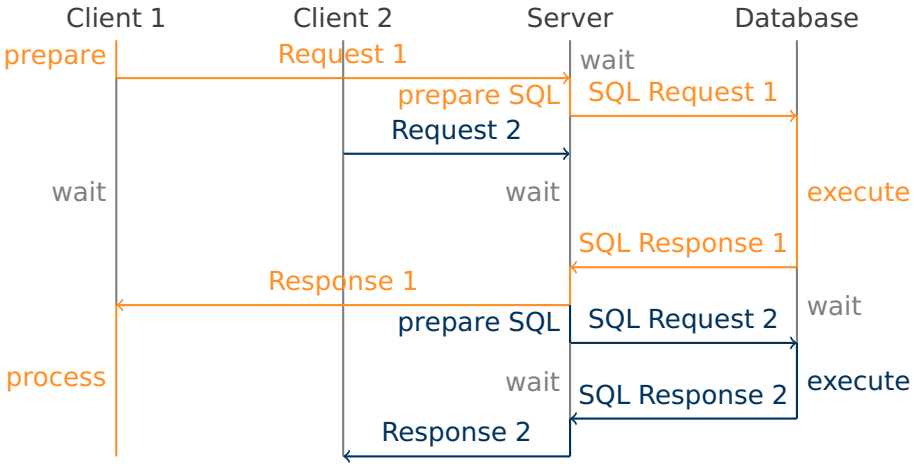
Synchronous server



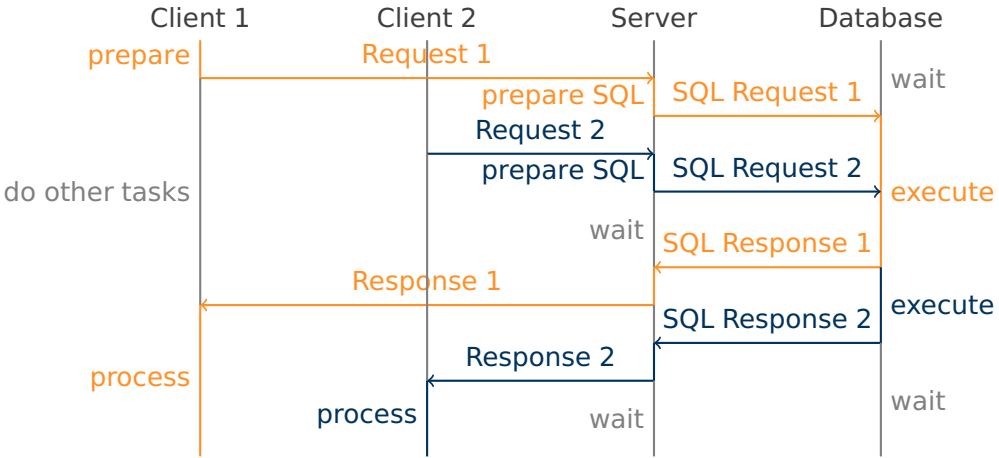
Synchronous server



Synchronous server



Asynchronous server



Synchronous execution in C#

```
1  public string Download(Uri uri) {
2      var client = new DownloadClient();
3      var result = client.Download(uri);
4
5      return result;
6  }
```

Listing 1: Synchronous usage in C#

Execute synchronous code asynchronously in C#

```
1  public async Task<string> DownloadAsync(Uri uri) {
2      var client = new DownloadClient();
3      var result = await Task.Run(() => client.Download());
4
5      return result;
6  }
```

Listing 2: Synchronous code executed asynchronously in C#

Event-based execution in C#

```
1  public class DownloadResult {
2      string result = null;
3
4      bool IsComplete() {
5          return result != null;
6      }
7
8      public void SetComplete(string result) {
9          this.result = result;
10     }
11
12     public string GetResult() {
13         return this.result;
14     }
15 }
```

Listing 3: DownloadResult to get string

Event-based execution in C#

```
17  public DownloadResult Download(Uri uri) {
18      var client = new DownloadClient();
19      var result = new DownloadResult();
20
21      client.DownloadComplete += (content) => result.SetComplete(content);
22      client.StartDownload(uri);
23
24      return result;
25  }
```

Listing 4: Usage of events in C#

Asynchronous execution in C#

```
1  public async Task<string> DownloadAsync(Uri uri, CancellationToken ct) {
2      var client = new DownloadClient();
3      var result = await client.DownloadAsync(uri, ct);
4
5      return result;
6  }
```

Listing 5: 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

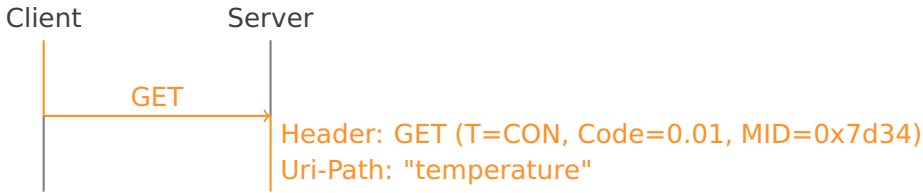


Figure: GET request

Example request in CoAP

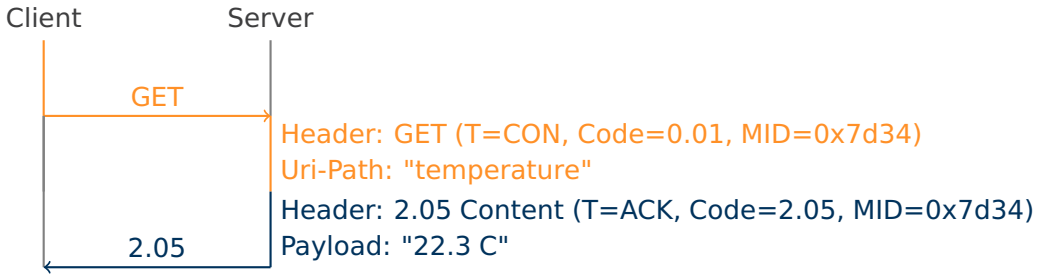


Figure: GET request

Example request in CoAP

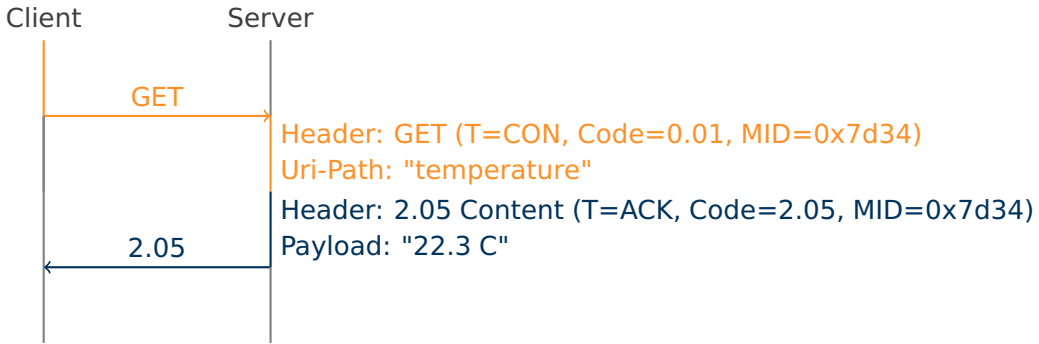


Figure: GET request

CoAP.NET

- Implementation of CoAP for C#.
- Development inactive.
- Partially asynchronous.

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)