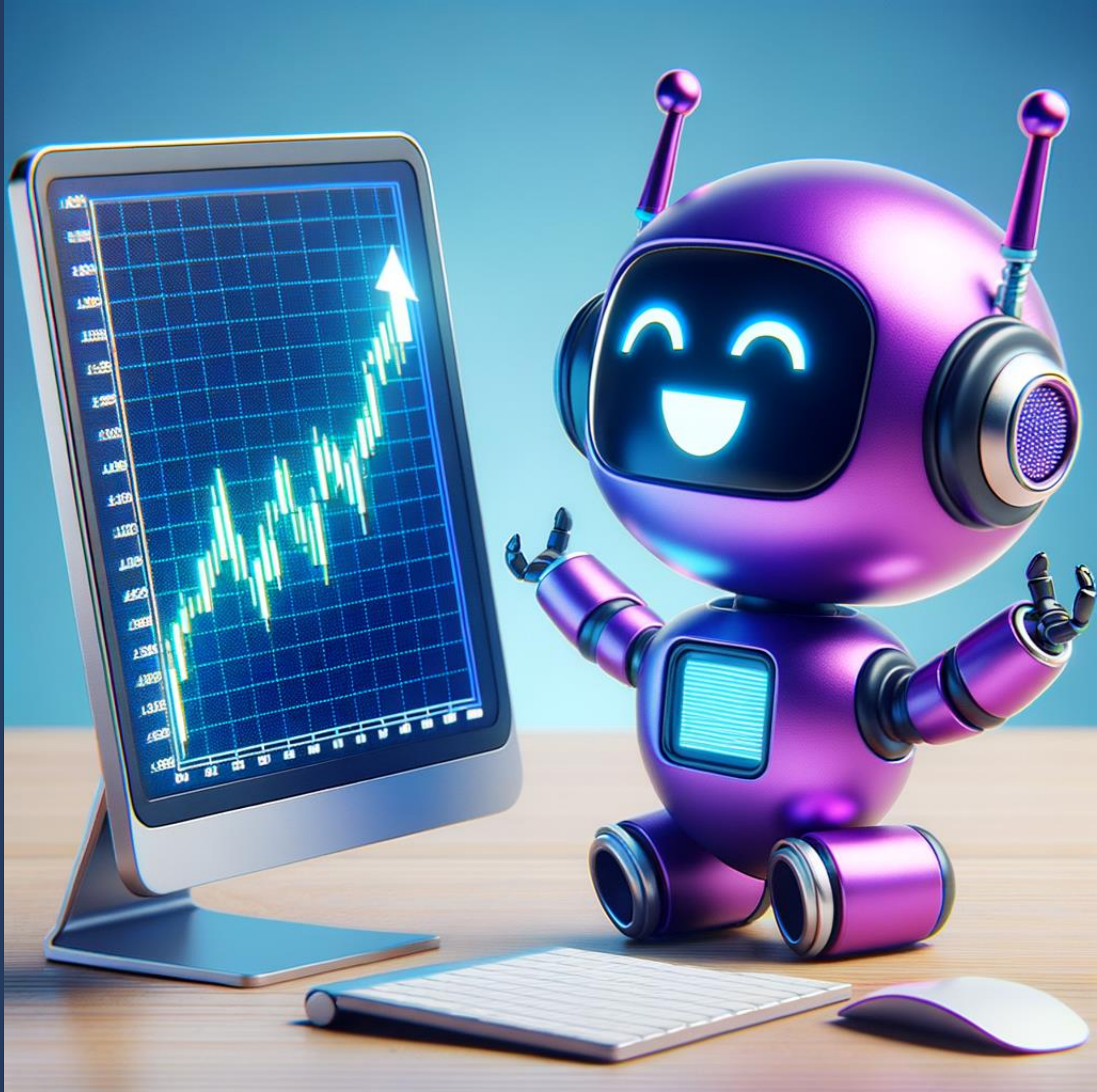
Microsoft

HttpClient & performance

Miha Zupan



Step 0:
Use modern .NET

Framework = ☹️

Step 1:
Use HttpClient

HttpWebRequest / WebClient = 🙁

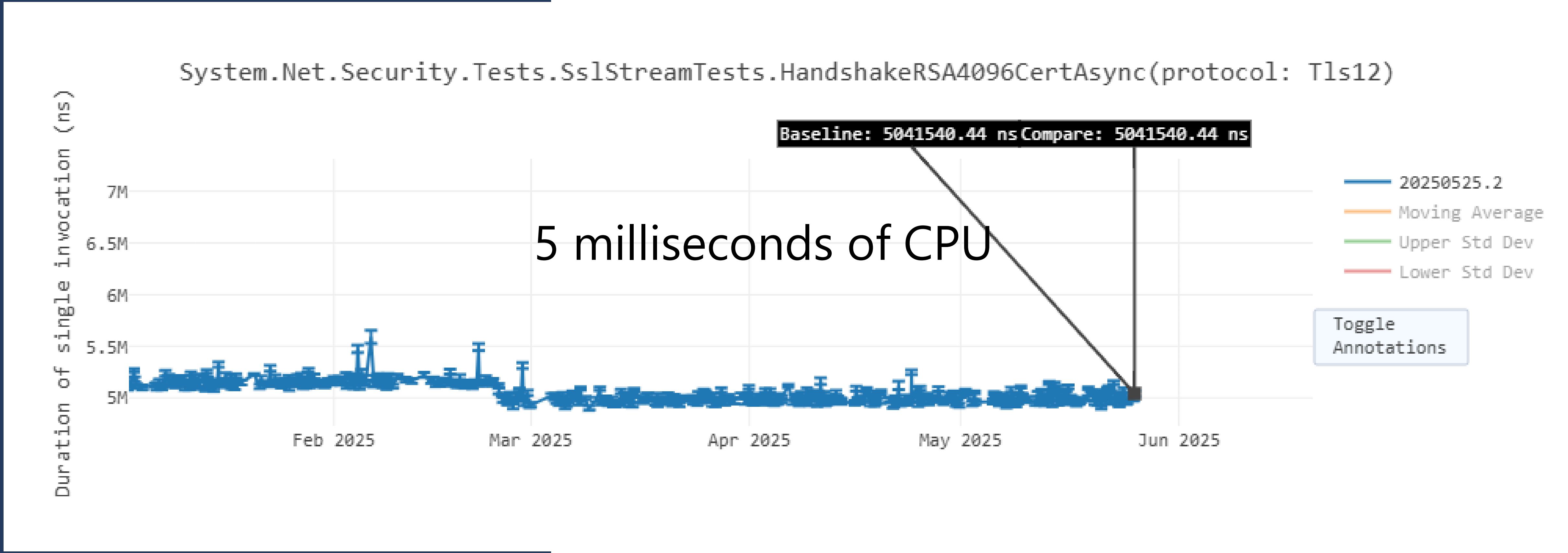
HttpWebRequest on modern .NET = 😢

See new “Migrate from HttpWebRequest” docs

Step 2:
Reusing connections

Why bother?

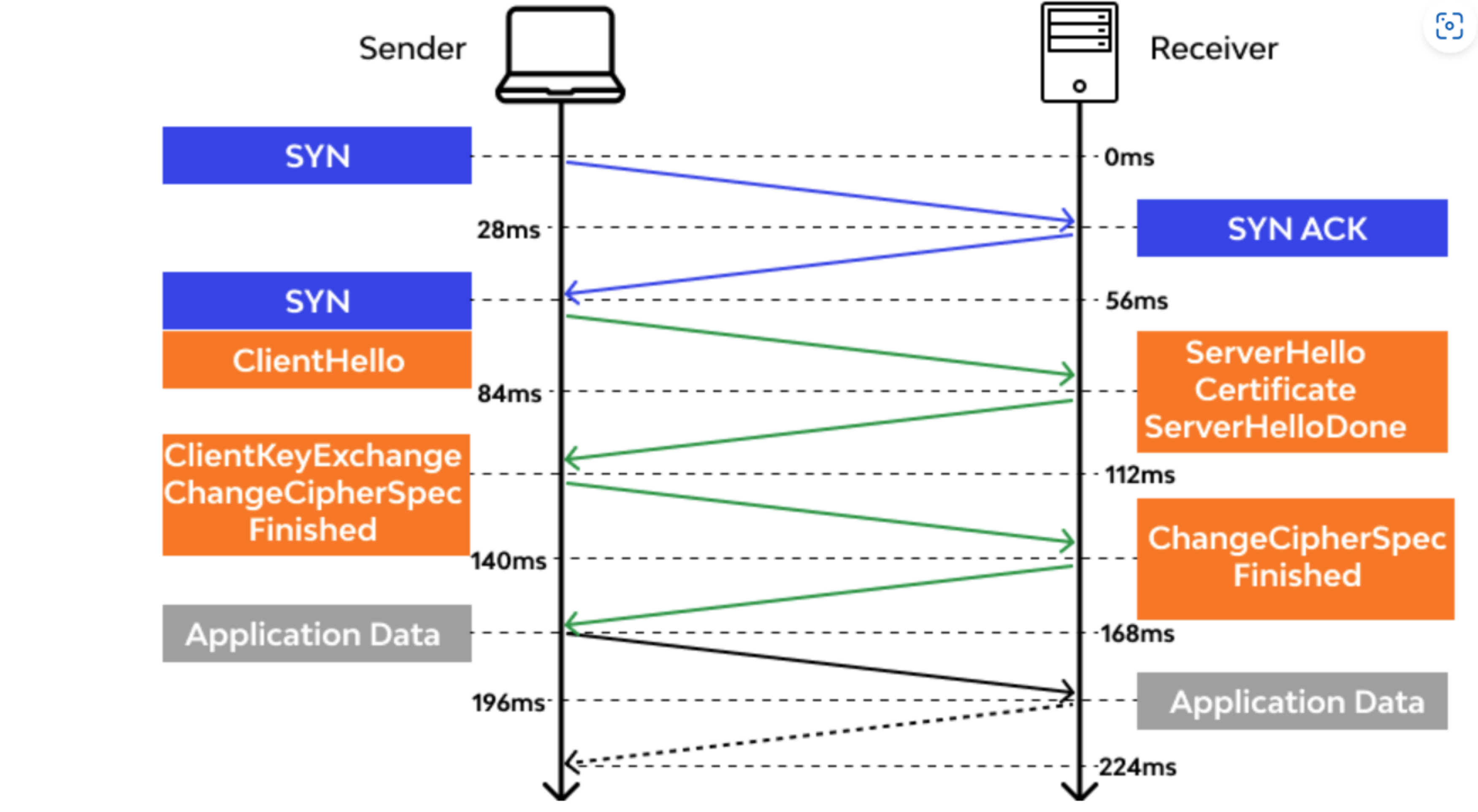
- Crypto is not cheap



5 milliseconds of CPU

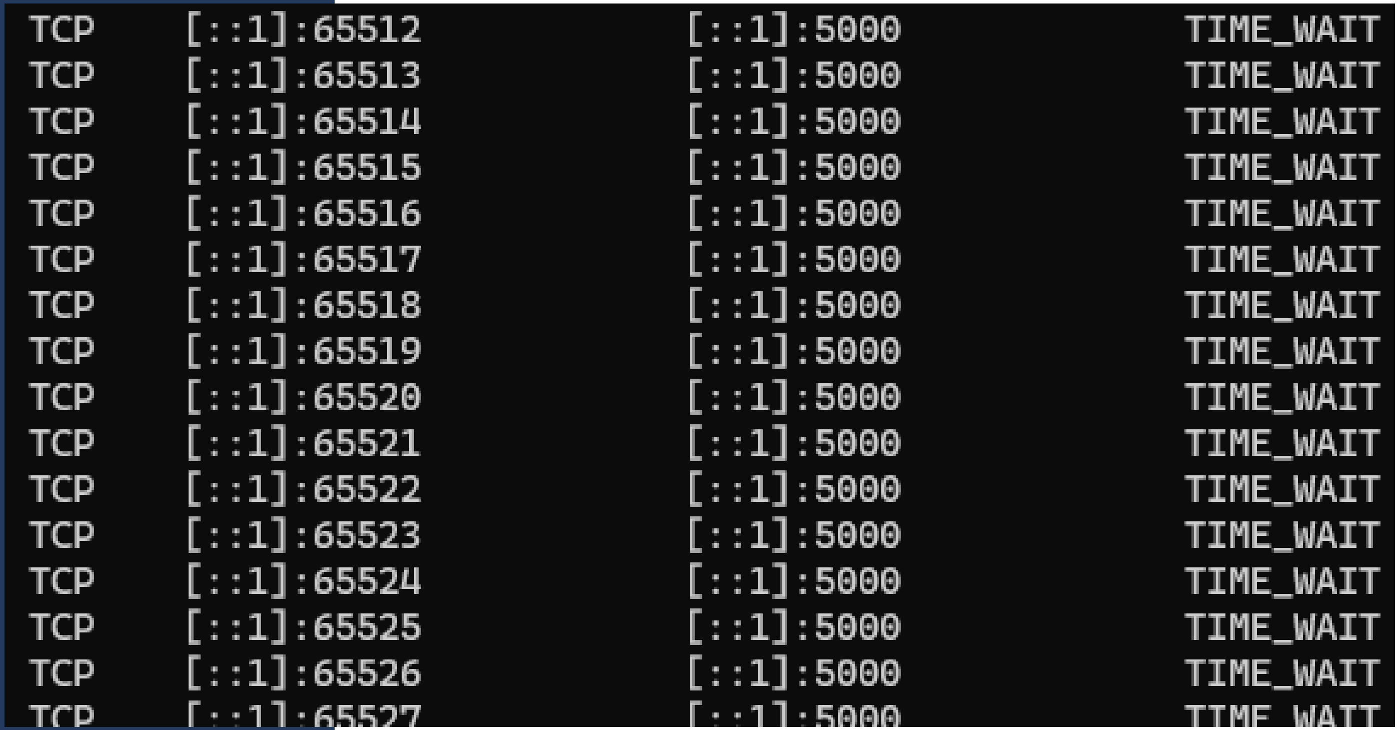
Why bother? #2

- Crypto is not cheap
- Latency



Why bother? #3

- Crypto is not cheap
- Latency
- Limited resource
  - “Only one usage of each socket address (protocol/network address/port) is normally permitted.”



```
TCP    [::1]:65512            [::1]:5000             TIME_WAIT
TCP    [::1]:65513            [::1]:5000             TIME_WAIT
TCP    [::1]:65514            [::1]:5000             TIME_WAIT
TCP    [::1]:65515            [::1]:5000             TIME_WAIT
TCP    [::1]:65516            [::1]:5000             TIME_WAIT
TCP    [::1]:65517            [::1]:5000             TIME_WAIT
TCP    [::1]:65518            [::1]:5000             TIME_WAIT
TCP    [::1]:65519            [::1]:5000             TIME_WAIT
TCP    [::1]:65520            [::1]:5000             TIME_WAIT
TCP    [::1]:65521            [::1]:5000             TIME_WAIT
TCP    [::1]:65522            [::1]:5000             TIME_WAIT
TCP    [::1]:65523            [::1]:5000             TIME_WAIT
TCP    [::1]:65524            [::1]:5000             TIME_WAIT
TCP    [::1]:65525            [::1]:5000             TIME_WAIT
TCP    [::1]:65526            [::1]:5000             TIME_WAIT
TCP    [::1]:65527            [::1]:5000             TIME_WAIT
```

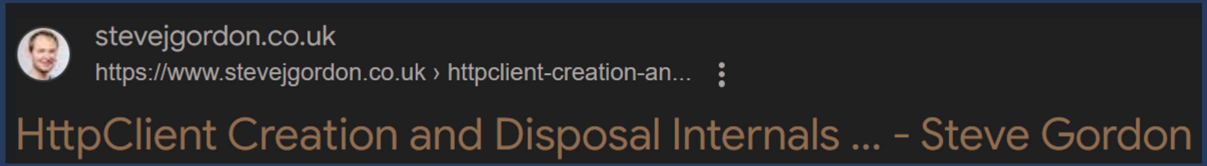
To dispose or not to dispose

Stack Overflow
https://stackoverflow.com › questions › do-httpclient-an...

Do HttpClient and HttpClientHandler have to be disposed ...

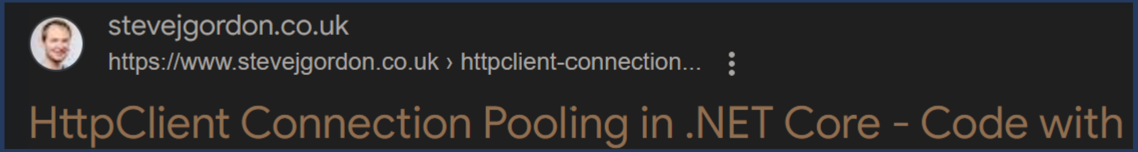
Reddit · r/csharp
6 comments · 4 years ago

Should dispose be called on an HTTPClient when it's ...



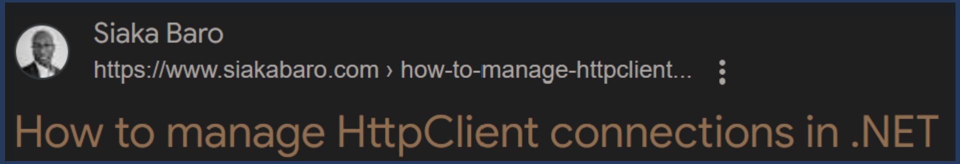
stevejgordon.co.uk
https://www.stevejgordon.co.uk › httpclient-creation-an...

HttpClient Creation and Disposal Internals ... - Steve Gordon



stevejgordon.co.uk
https://www.stevejgordon.co.uk › httpclient-connection...

HttpClient Connection Pooling in .NET Core - Code with

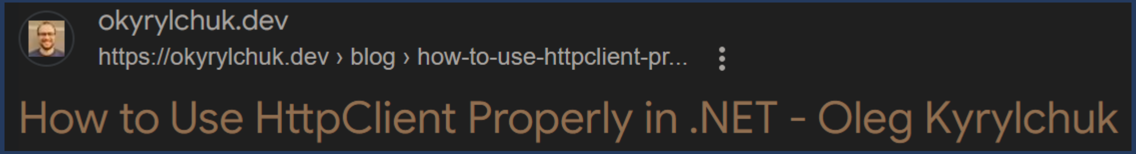


Siaka Baro
https://www.siakabaro.com › how-to-manage-httpclient...

How to manage HttpClient connections in .NET

C# Corner
https://www.c-sharpcorner.com › article › optimize-http...

Optimize HttpClient Usage in .NET Core

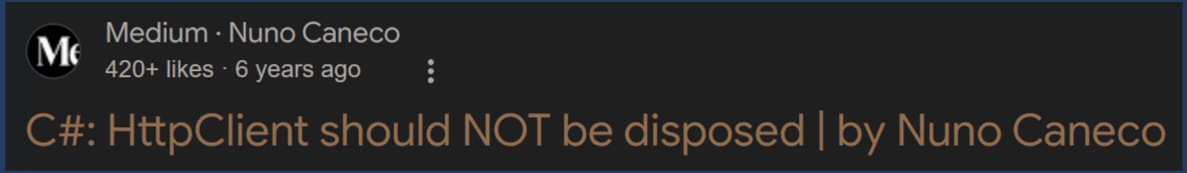


okyrylchuk.dev
https://okyrylchuk.dev › blog › how-to-use-httpclient-pr...

How to Use HttpClient Properly in .NET - Oleg Kyrylchuk

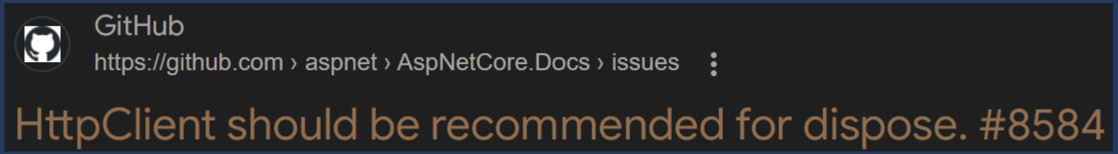
mytechramblings.com
https://www.mytechramblings.com › posts › dotnet-httpc...

Back to .NET basics: How to properly use HttpClient



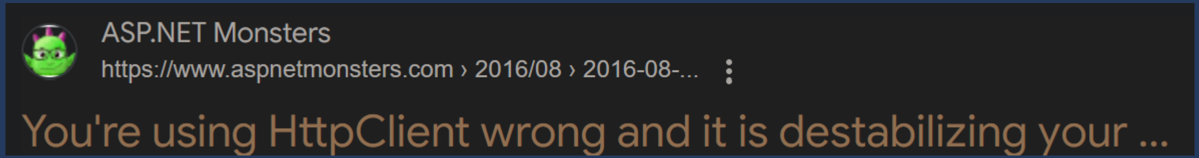
Medium · Nuno Caneco
420+ likes · 6 years ago

C#: HttpClient should NOT be disposed | by Nuno Caneco



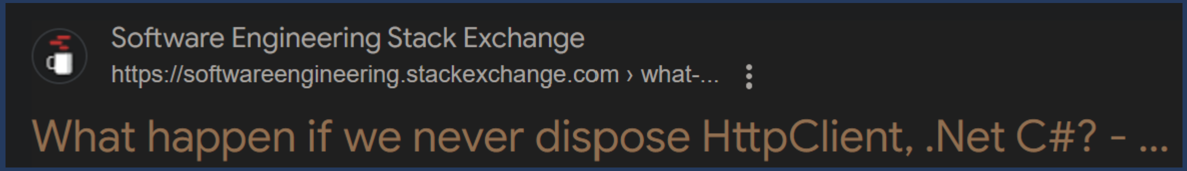
GitHub
https://github.com › aspnet › AspNetCore.Docs › issues

HttpClient should be recommended for dispose. #8584



ASP.NET Monsters
https://www.aspnetmonsters.com › 2016/08 › 2016-08-...

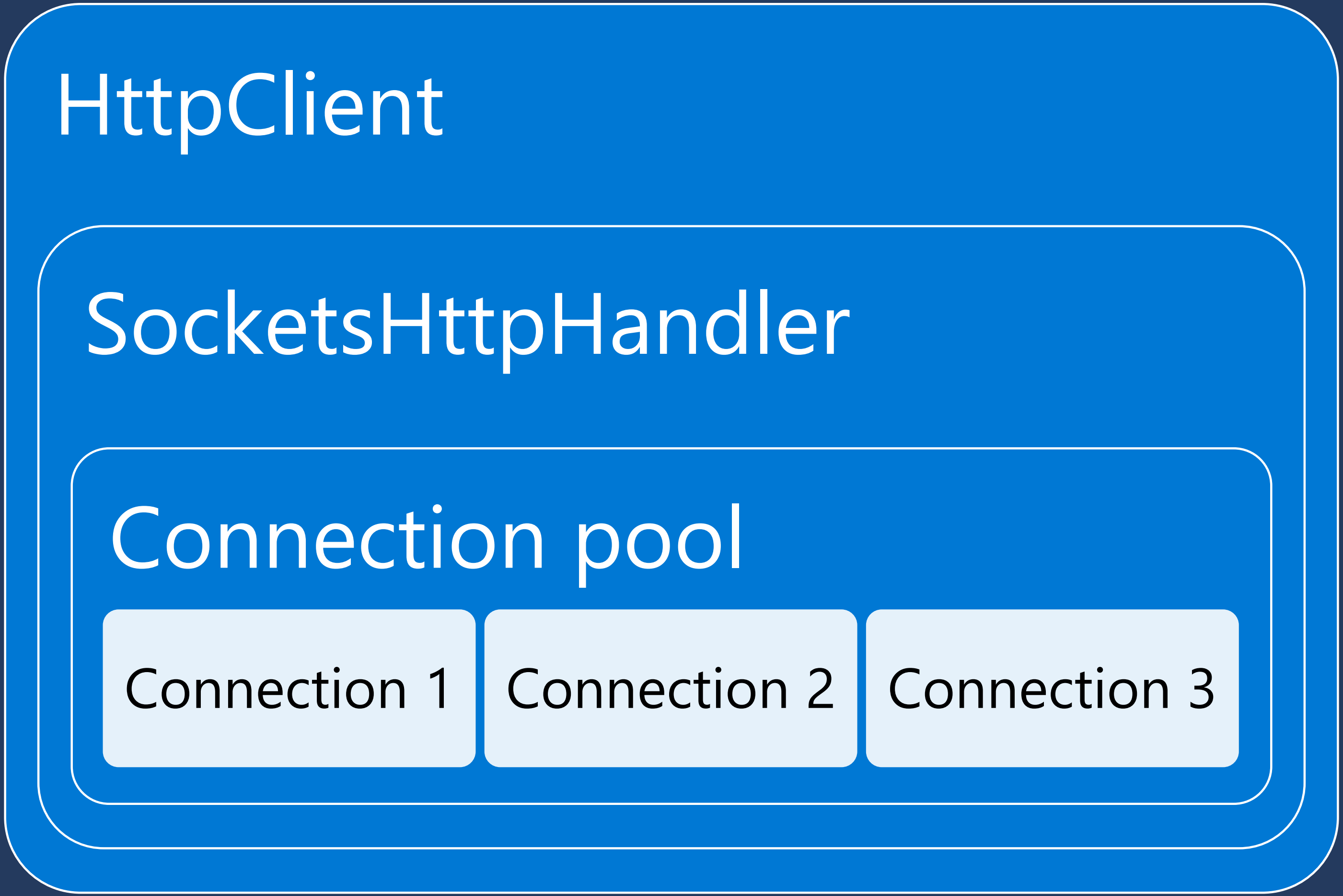
You're using HttpClient wrong and it is destabilizing your ...



Software Engineering Stack Exchange
https://softwareengineering.stackexchange.com › what-...

What happen if we never dispose HttpClient, .Net C#? - ...

Where does the connection pool live?



HttpClient

SocketsHttpHandler

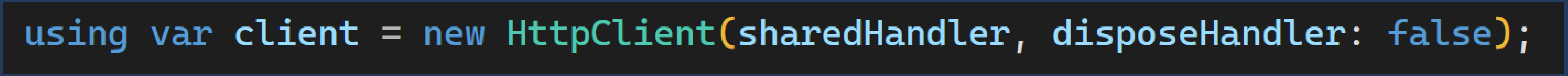
Connection pool

Connection 1

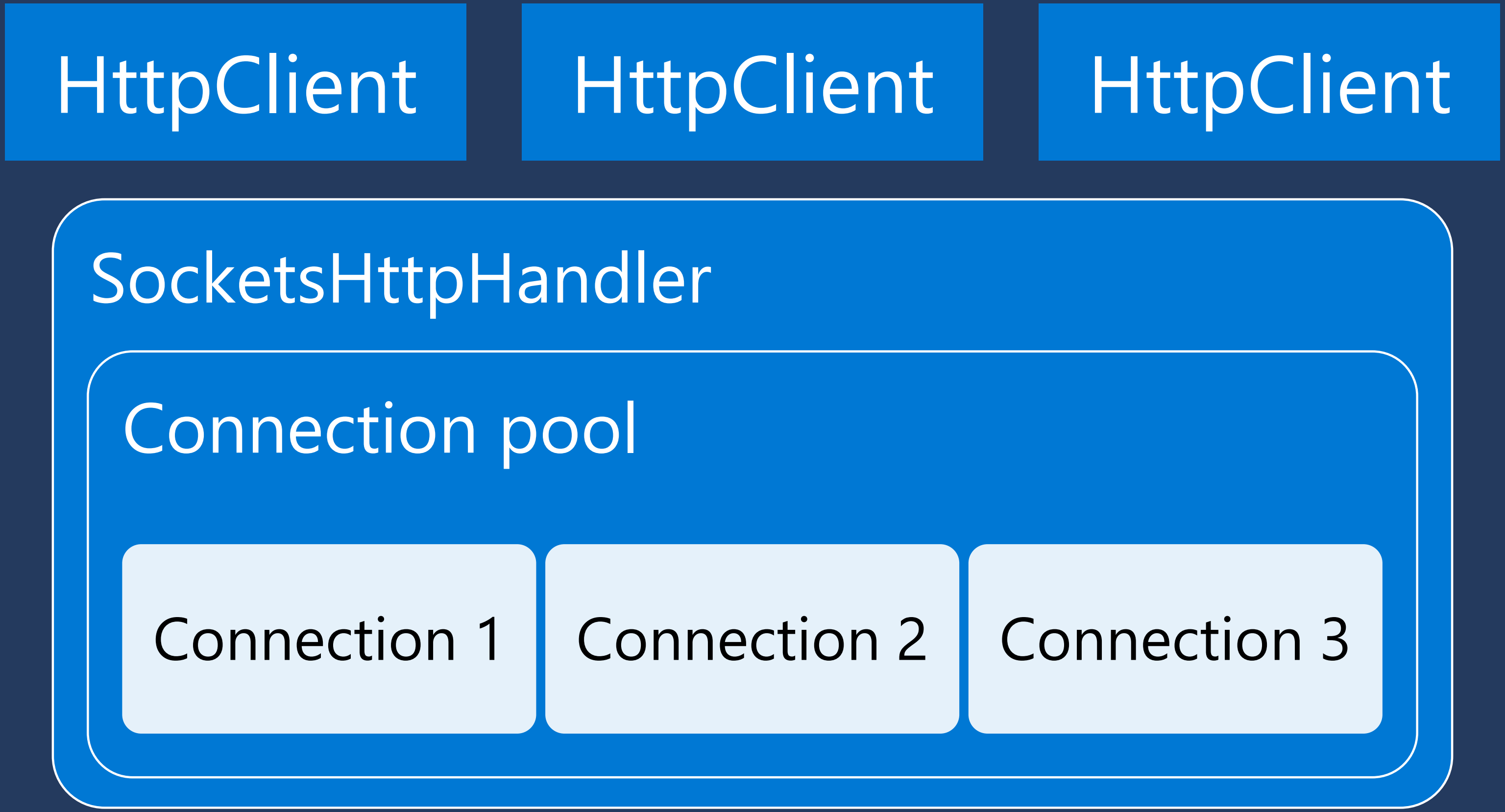
Connection 2

Connection 3

Reusing an HttpMessageHandler



```
using var client = new HttpClient(sharedHandler, disposeHandler: false);
```



HttpClient

HttpClient

HttpClient

SocketsHttpHandler

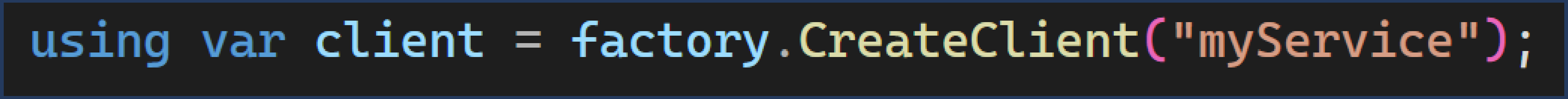
Connection pool

Connection 1

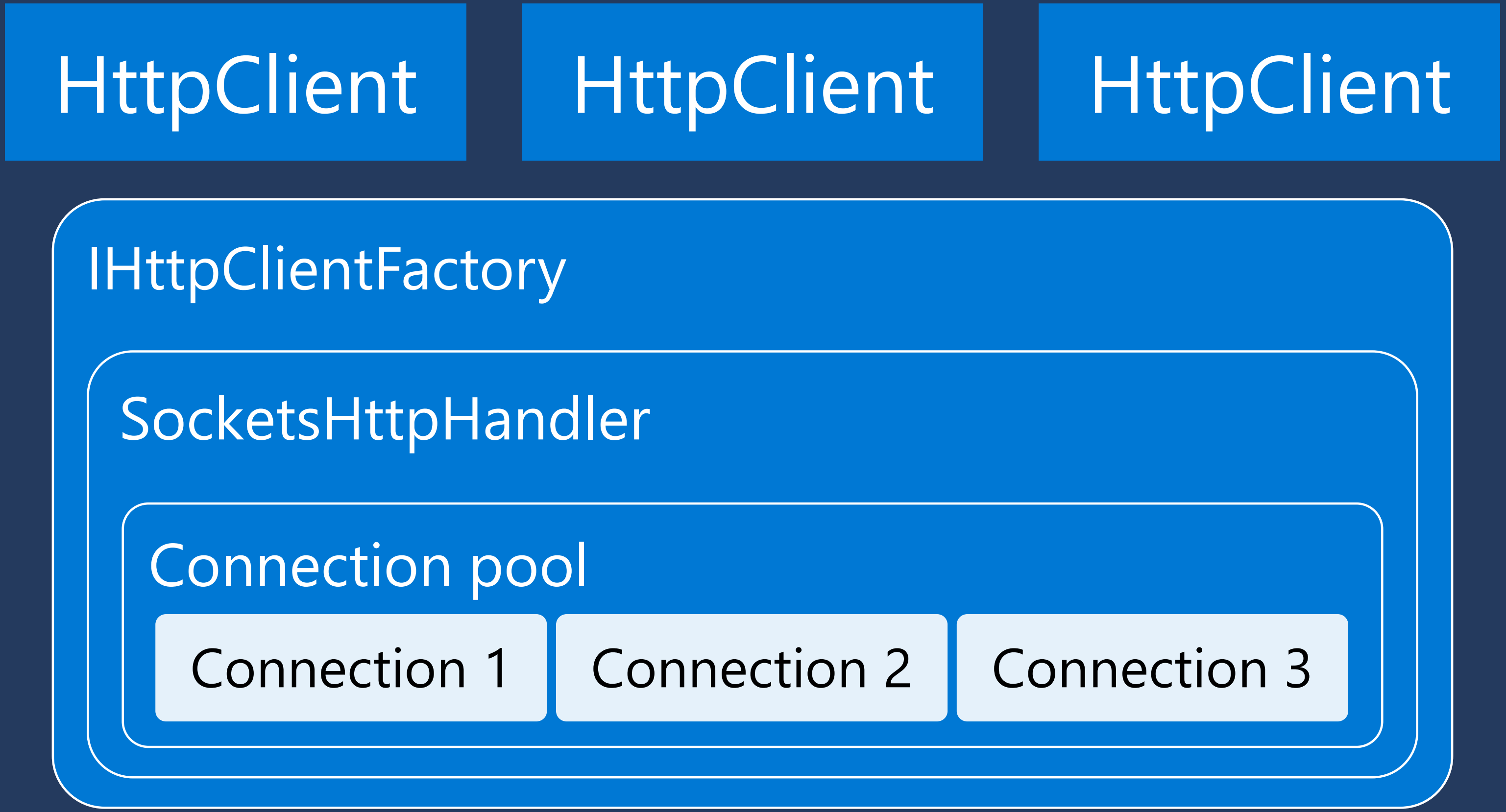
Connection 2

Connection 3

HttpClientFactory



```
using var client = factory.CreateClient("myService");
```



HttpClient

HttpClient

HttpClient

IHttpClientFactory

SocketsHttpHandler

Connection pool

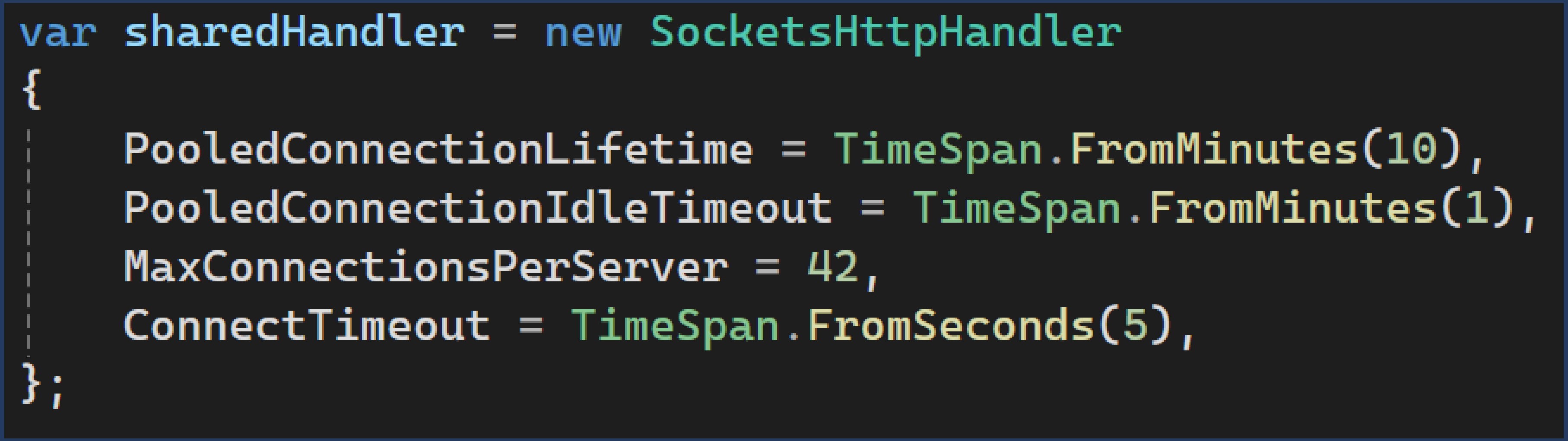
Connection 1

Connection 2

Connection 3

What does the connection pool do?

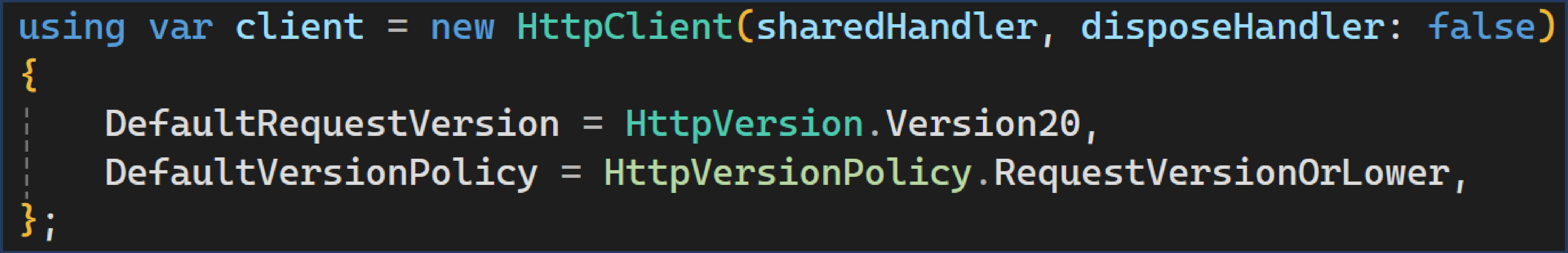
- Keep track of connections, establish new ones
- Dealing with proxies
- Enforce connection limits, lifetime, idle timeout
- Detect dead connections



```
var sharedHandler = new SocketsHttpHandler
{
    PooledConnectionLifetime = TimeSpan.FromMinutes(10),
    PooledConnectionIdleTimeout = TimeSpan.FromMinutes(1),
    MaxConnectionsPerServer = 42,
    ConnectTimeout = TimeSpan.FromSeconds(5),
};
```

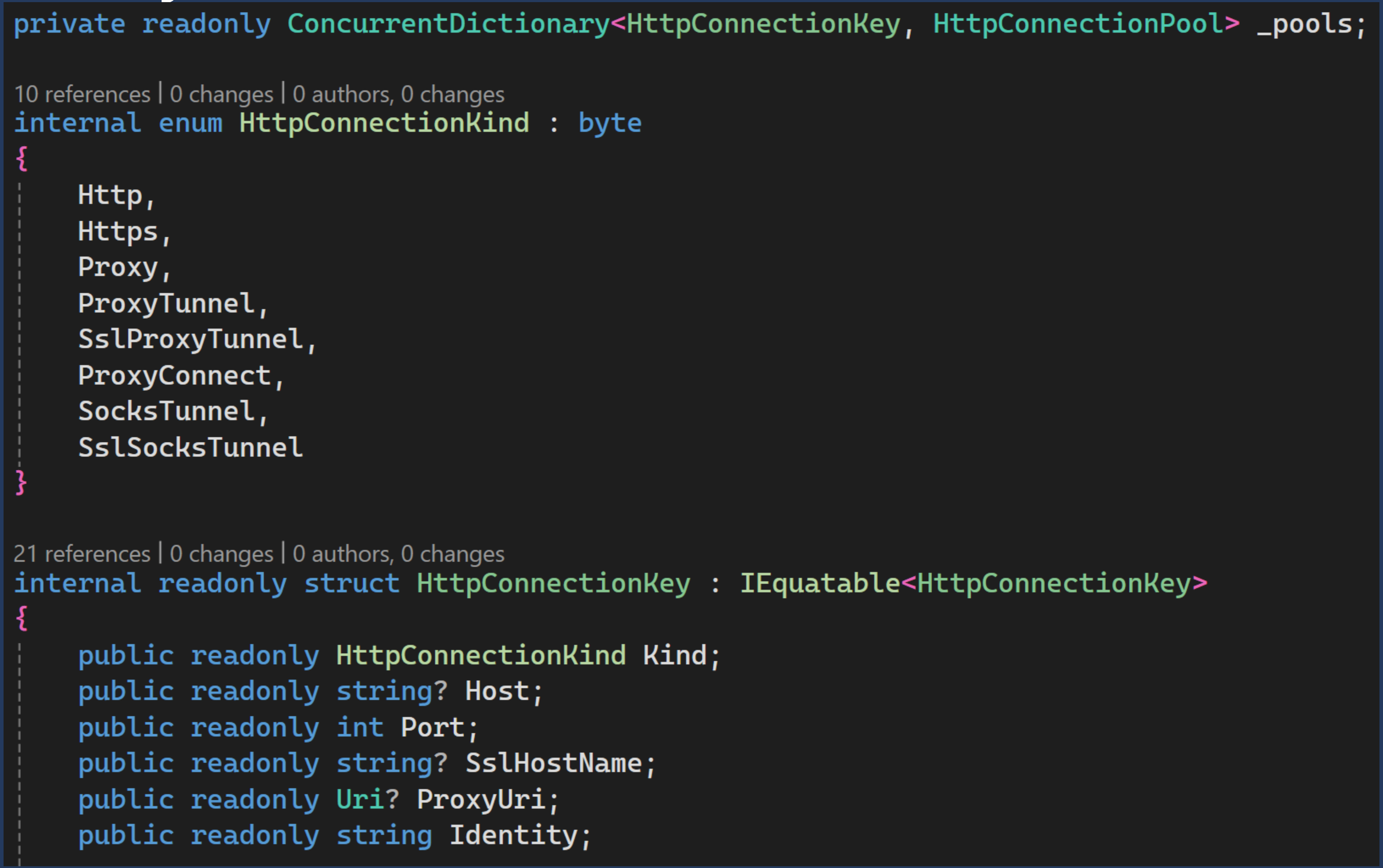
What does it do (cont.)?

- HTTP version upgrades/downgrades
- Track shared endpoint configuration
- Handle connection-based authentication
- Emit useful diagnostics
- Be fully thread-safe & fast



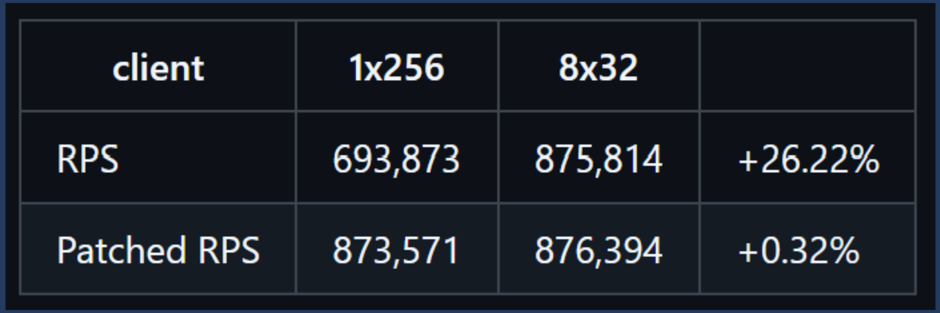
```
using var client = new HttpClient(sharedHandler, disposeHandler: false)
{
    DefaultRequestVersion = HttpVersion.Version20,
    DefaultVersionPolicy = HttpVersionPolicy.RequestVersionOrLower,
};
```

How many are there?

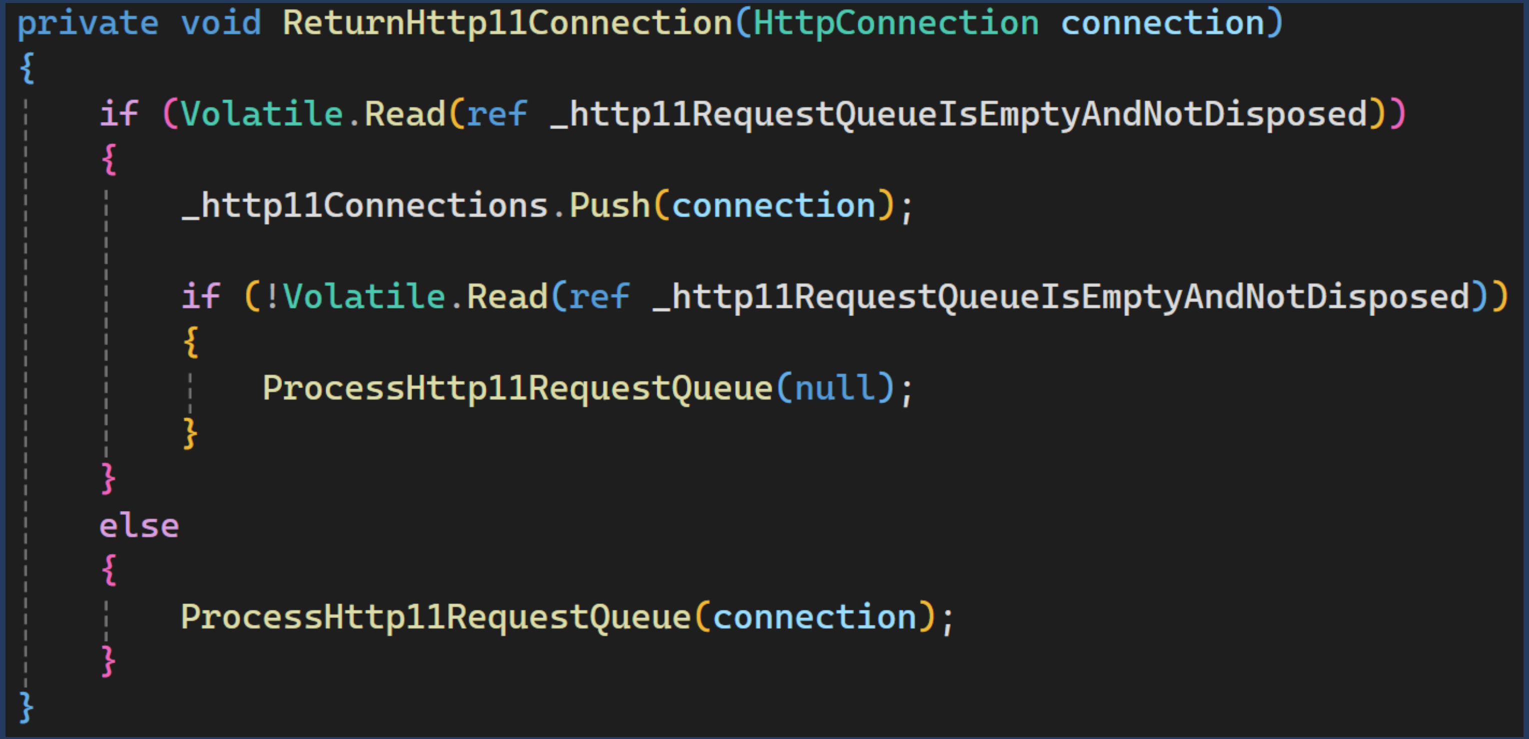


```
private readonly ConcurrentDictionary<HttpConnectionKey, HttpConnectionPool> _pools;

10 references | 0 changes | 0 authors, 0 changes
internal enum HttpConnectionKind : byte
{
    Http,
    Https,
    Proxy,
    ProxyTunnel,
    SslProxyTunnel,
    ProxyConnect,
    SocksTunnel,
    SslSocksTunnel
}

21 references | 0 changes | 0 authors, 0 changes
internal readonly struct HttpConnectionKey : IEquatable<HttpConnectionKey>
{
    public readonly HttpConnectionKind Kind;
    public readonly string? Host;
    public readonly int Port;
    public readonly string? SslHostName;
    public readonly Uri? ProxyUri;
    public readonly string Identity;
```

Lock contention improvement in .NET 9



client	1x256	8x32	
RPS	693,873	875,814	+26.22%
Patched RPS	873,571	876,394	+0.32%

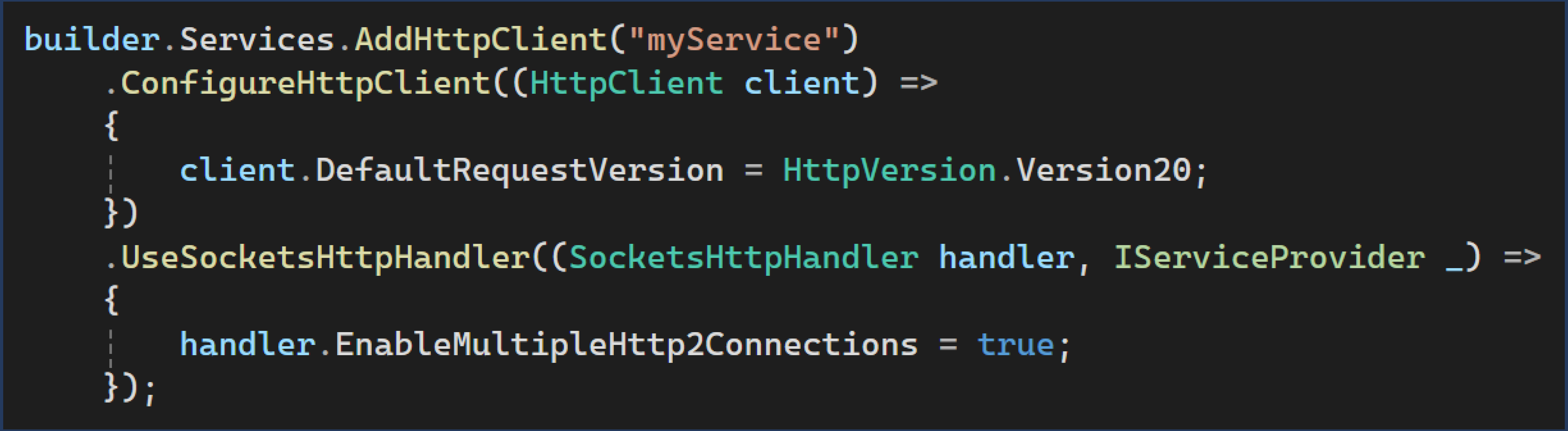


```
private void ReturnHttp11Connection(HttpConnection connection)
{
    if (Volatile.Read(ref _http11RequestQueueIsEmptyAndNotDisposed))
    {
        _http11Connections.Push(connection);

        if (!Volatile.Read(ref _http11RequestQueueIsEmptyAndNotDisposed))
        {
            ProcessHttp11RequestQueue(null);
        }
    }
    else
    {
        ProcessHttp11RequestQueue(connection);
    }
}
```

https://github.com/dotnet/runtime/pull/99364

HTTP/2 and HTTP/3

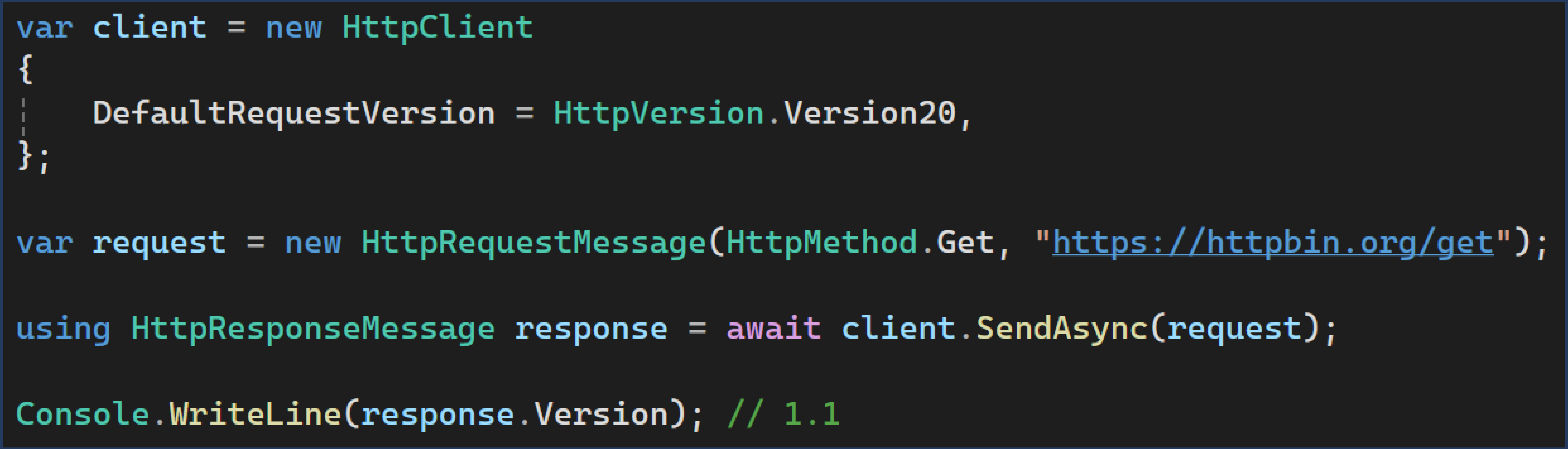
- Default is HTTP/1.1 only



```
builder.Services.AddHttpClient("myService")
    .ConfigureHttpClient((HttpClient client) =>
    {
        client.DefaultRequestVersion = HttpVersion.Version20;
    })
    .UseSocketsHttpHandler((SocketsHttpHandler handler, IServiceProvider _) =>
    {
        handler.EnableMultipleHttp2Connections = true;
    });
```

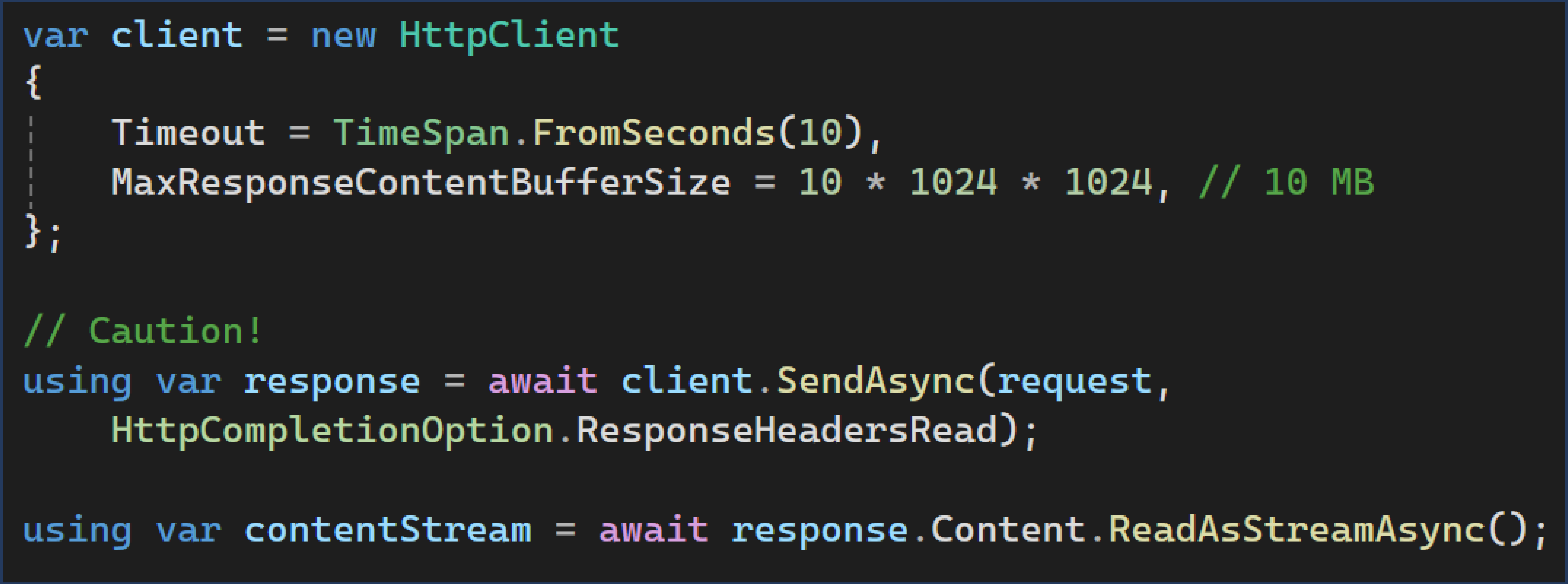
HTTP/2 and HTTP/3

- Don’t forget about HttpRequestMessage.Version



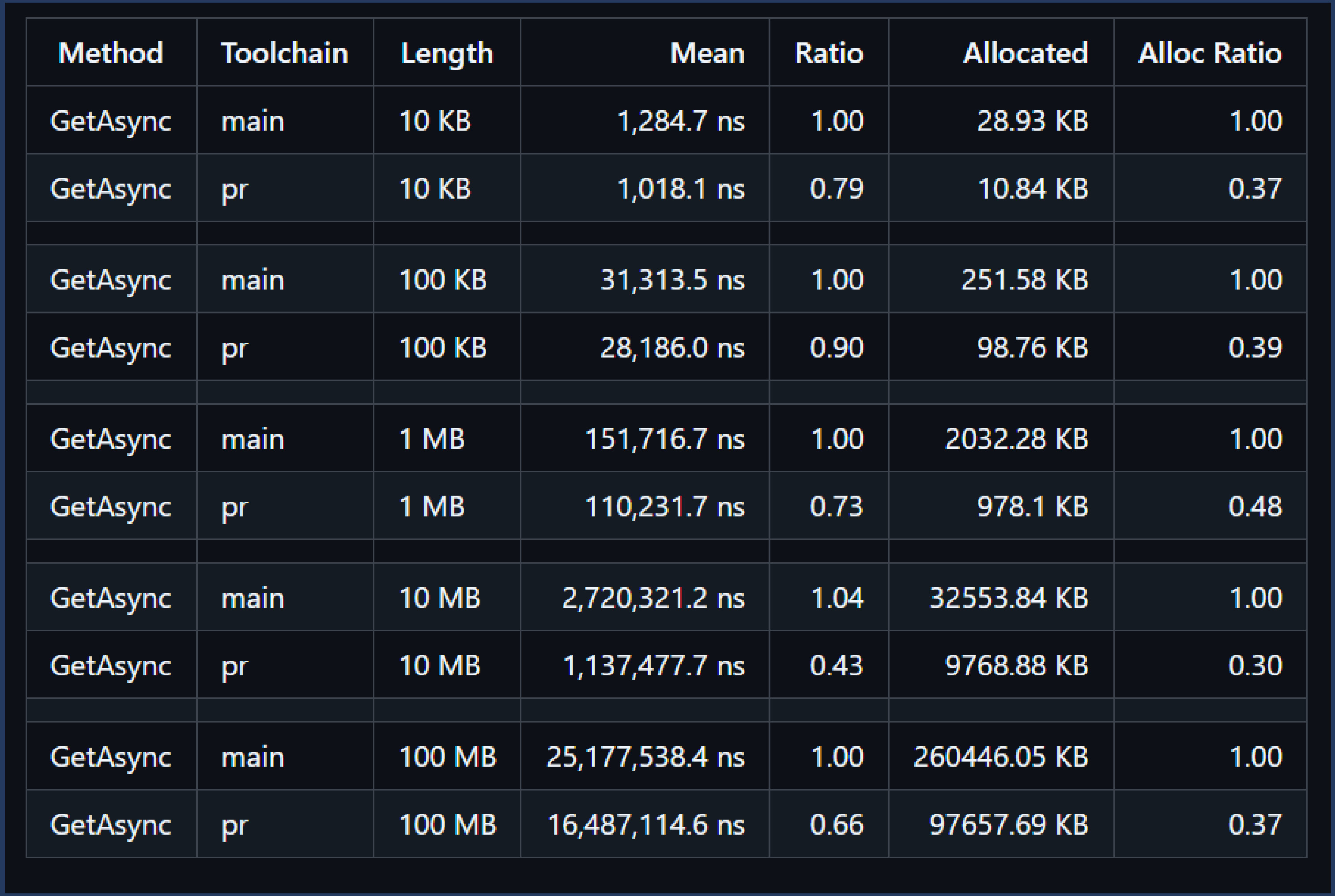
```
var client = new HttpClient
{
    DefaultRequestVersion = HttpVersion.Version20,
};

var request = new HttpRequestMessage(HttpMethod.Get, "https://httpbin.org/get");

using HttpResponseMessage response = await client.SendAsync(request);

Console.WriteLine(response.Version); // 1.1
```

Response buffering



```
var client = new HttpClient
{
    Timeout = TimeSpan.FromSeconds(10),
    MaxResponseContentBufferSize = 10 * 1024 * 1024, // 10 MB
};

// Caution!
using var response = await client.SendAsync(request,
    HttpCompletionOption.ResponseHeadersRead);

using var contentStream = await response.Content.ReadAsStreamAsync();
```

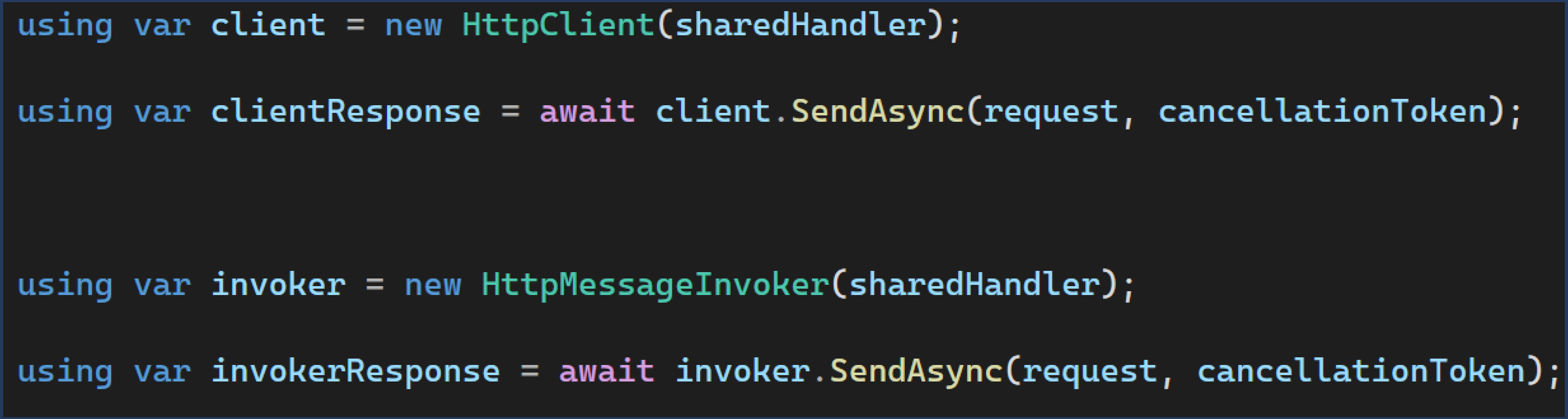
“Rework HttpClient content buffering” (.NET 10)



Method	Toolchain	Length	Mean	Ratio	Allocated	Alloc Ratio
GetAsync	main	10 KB	1,284.7 ns	1.00	28.93 KB	1.00
GetAsync	pr	10 KB	1,018.1 ns	0.79	10.84 KB	0.37
GetAsync	main	100 KB	31,313.5 ns	1.00	251.58 KB	1.00
GetAsync	pr	100 KB	28,186.0 ns	0.90	98.76 KB	0.39
GetAsync	main	1 MB	151,716.7 ns	1.00	2032.28 KB	1.00
GetAsync	pr	1 MB	110,231.7 ns	0.73	978.1 KB	0.48
GetAsync	main	10 MB	2,720,321.2 ns	1.04	32553.84 KB	1.00
GetAsync	pr	10 MB	1,137,477.7 ns	0.43	9768.88 KB	0.30
GetAsync	main	100 MB	25,177,538.4 ns	1.00	260446.05 KB	1.00
GetAsync	pr	100 MB	16,487,114.6 ns	0.66	97657.69 KB	0.37

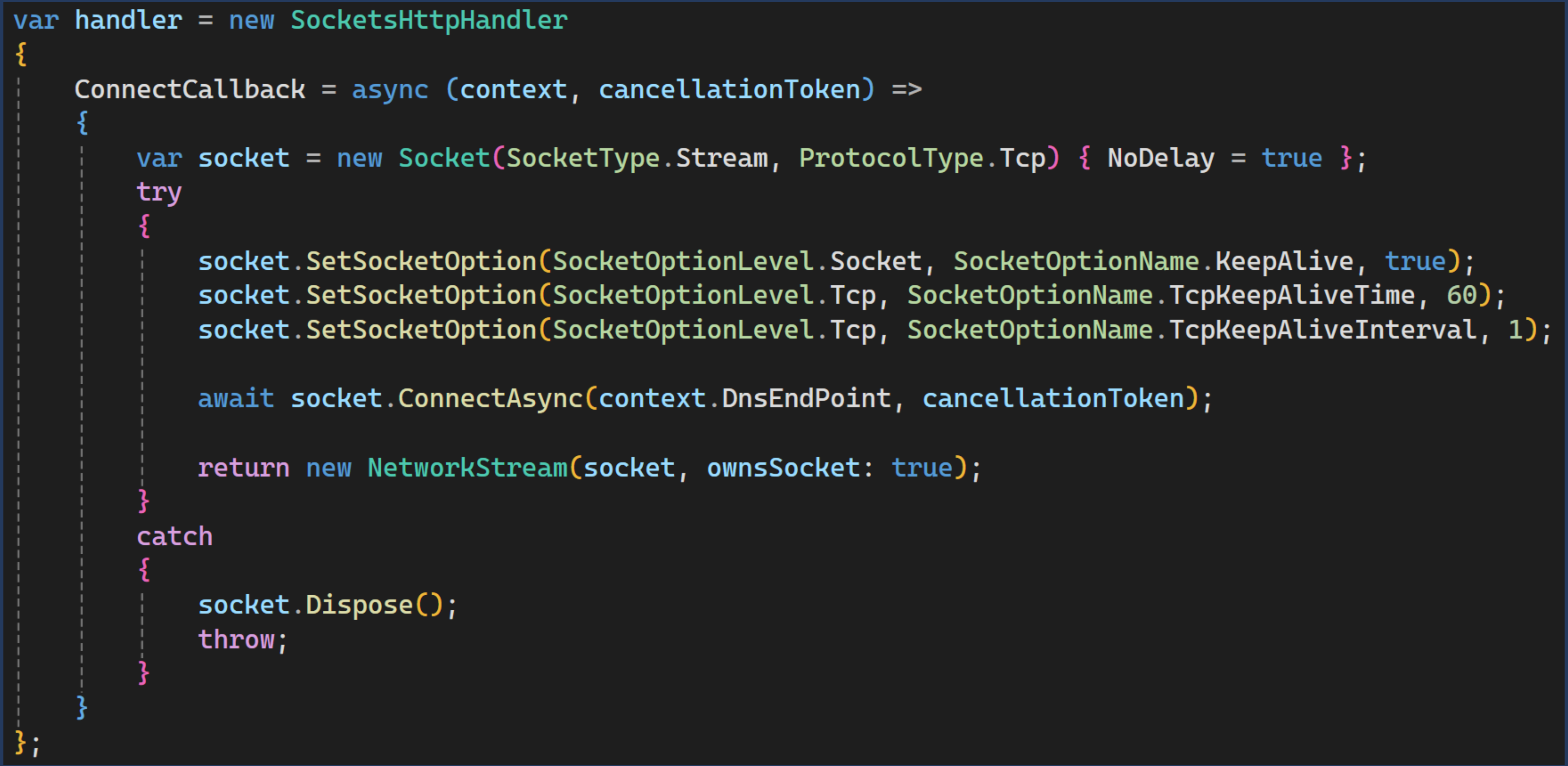
https://github.com/dotnet/runtime/pull/109642

HttpMessageInvoker



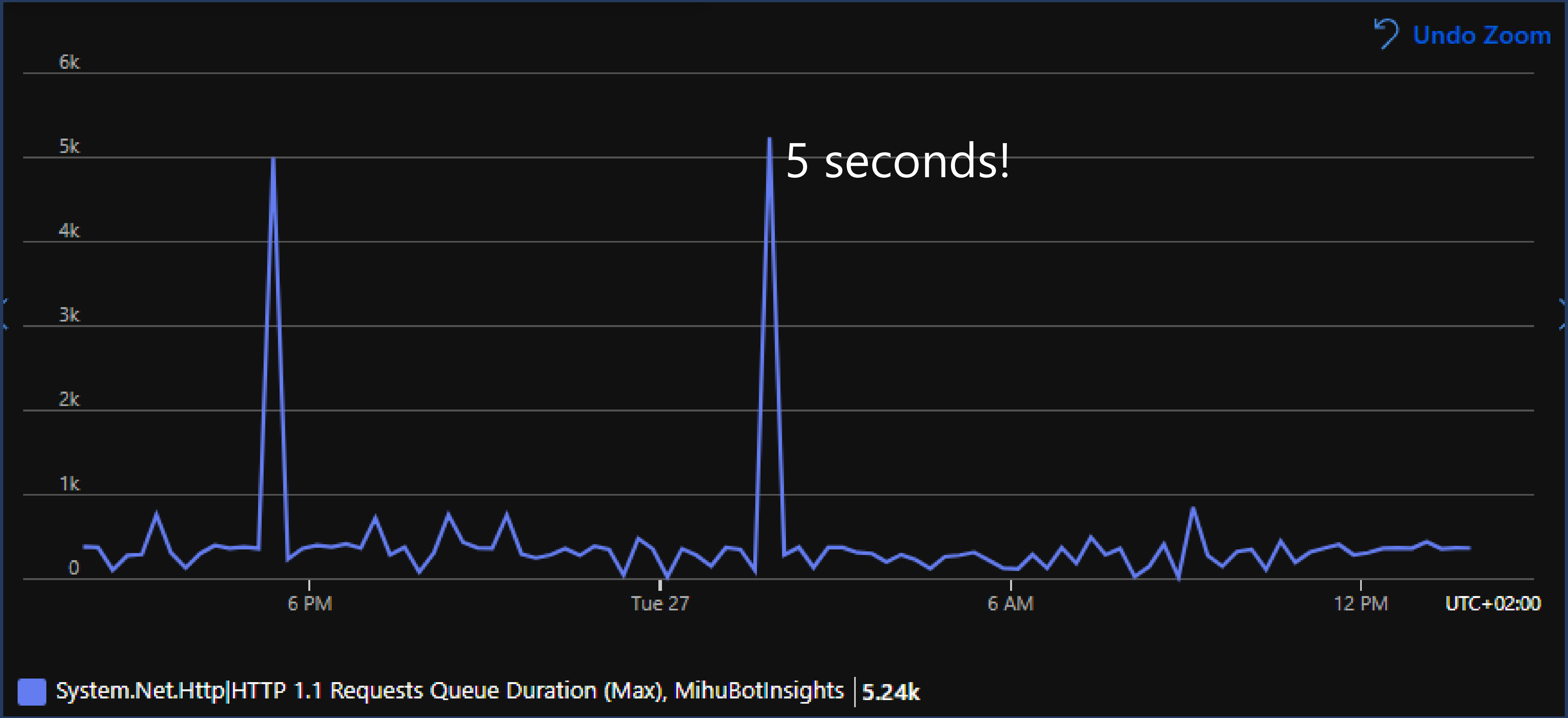
```
using var client = new HttpClient(sharedHandler);

using var clientResponse = await client.SendAsync(request, cancellationToken);



using var invoker = new HttpMessageInvoker(sharedHandler);

using var invokerResponse = await invoker.SendAsync(request, cancellationToken);
```

ConnectCallback



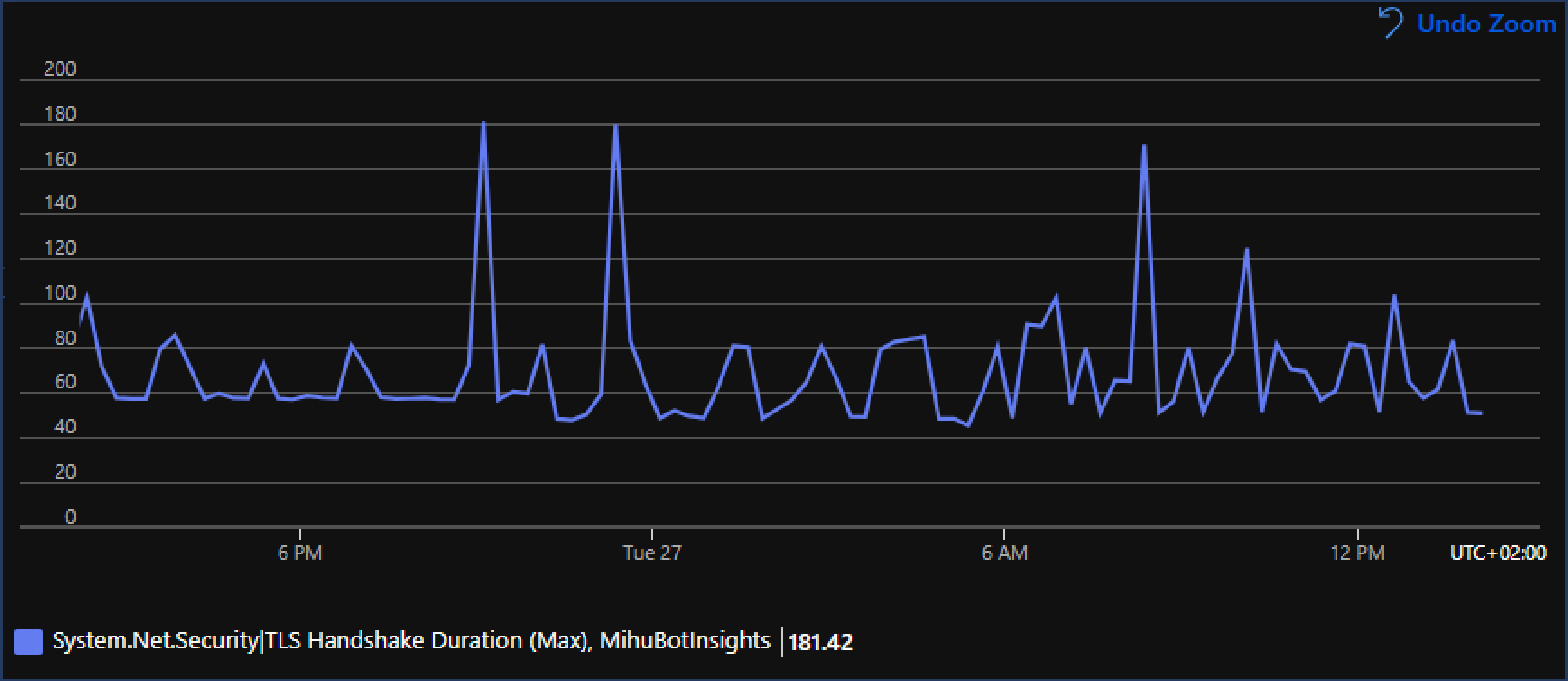
```
var handler = new SocketsHttpHandler
{
    ConnectCallback = async (context, cancellationToken) =>
    {
        var socket = new Socket(SocketType.Stream, ProtocolType.Tcp) { NoDelay = true };
        try
        {
            socket.SetSocketOption(SocketOptionLevel.Socket, SocketOptionName.KeepAlive, true);
            socket.SetSocketOption(SocketOptionLevel.Tcp, SocketOptionName.TcpKeepAliveTime, 60);
            socket.SetSocketOption(SocketOptionLevel.Tcp, SocketOptionName.TcpKeepAliveInterval, 1);

            await socket.ConnectAsync(context.DnsEndPoint, cancellationToken);

            return new NetworkStream(socket, ownsSocket: true);
        }
        catch
        {
            socket.Dispose();
            throw;
        }
    }
};
```

Telemetry – HTTP request queue

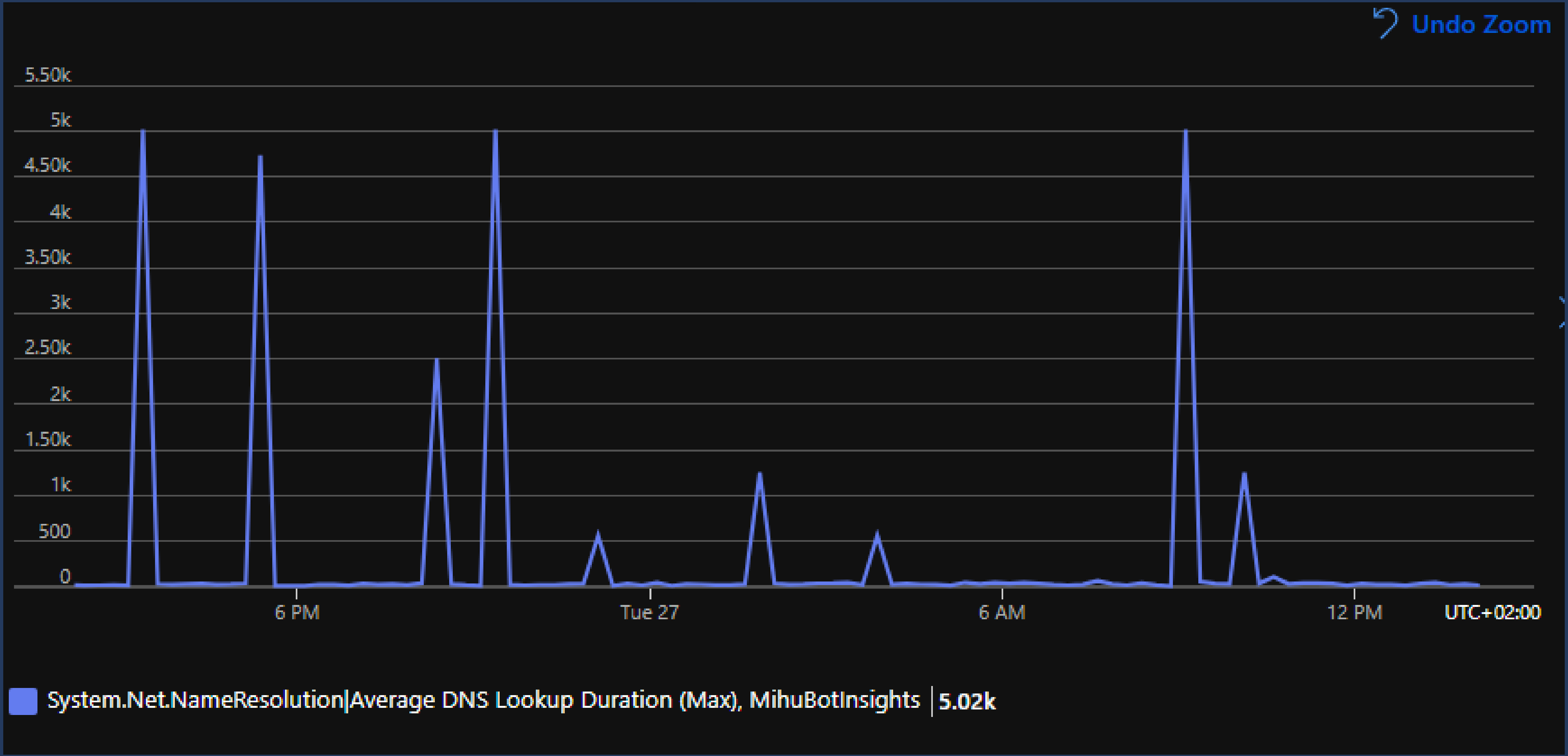


5 seconds!

Telemetry – TLS handshake



Telemetry – DNS



Thank you