OSIsoft PI World 2018

Writing Highly Performant PI Web API Applications

Presented by **Jim Bazis, Max Drexel**

Introduction

- **Max Drexel**
- mdrexel@osisoft.com
- Software Developer
- PI Web API Team

- **Jim Bazis**
- jbazis@osisoft.com
- Team Lead
- PI Web API Team

Agenda

- Goals
- Streamsets
- Batch requests
- Advanced Batch
- Channels
- Stream updates

Overview

Goals

- Understand common sources of poor performance
- Recognize them when they occur
- Know your options to mitigate them

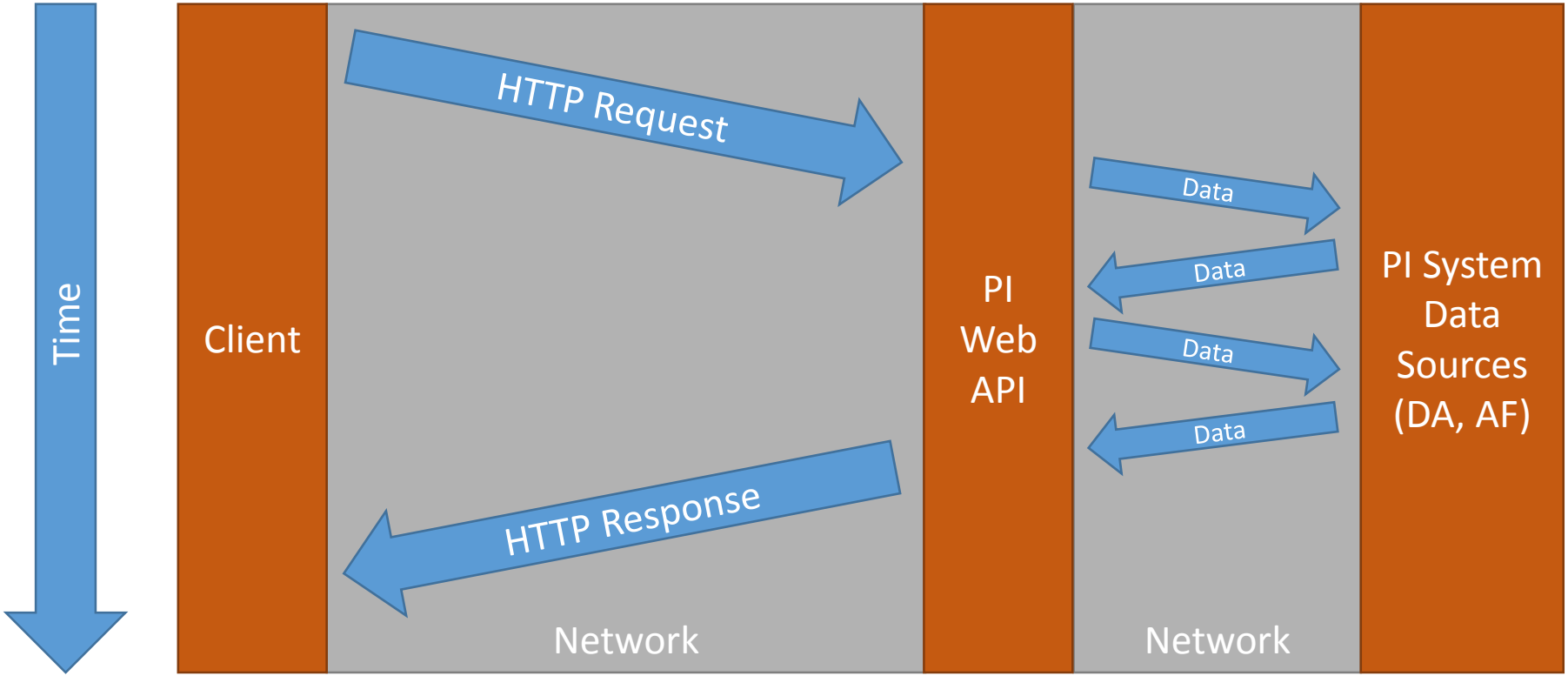
What do we mean by “Poor Performance”?

- Pages load too slowly for end users
- Too many users causes quality of service to degrade
- Can’t acquire data fast enough for application to be useful
- *And many more…*

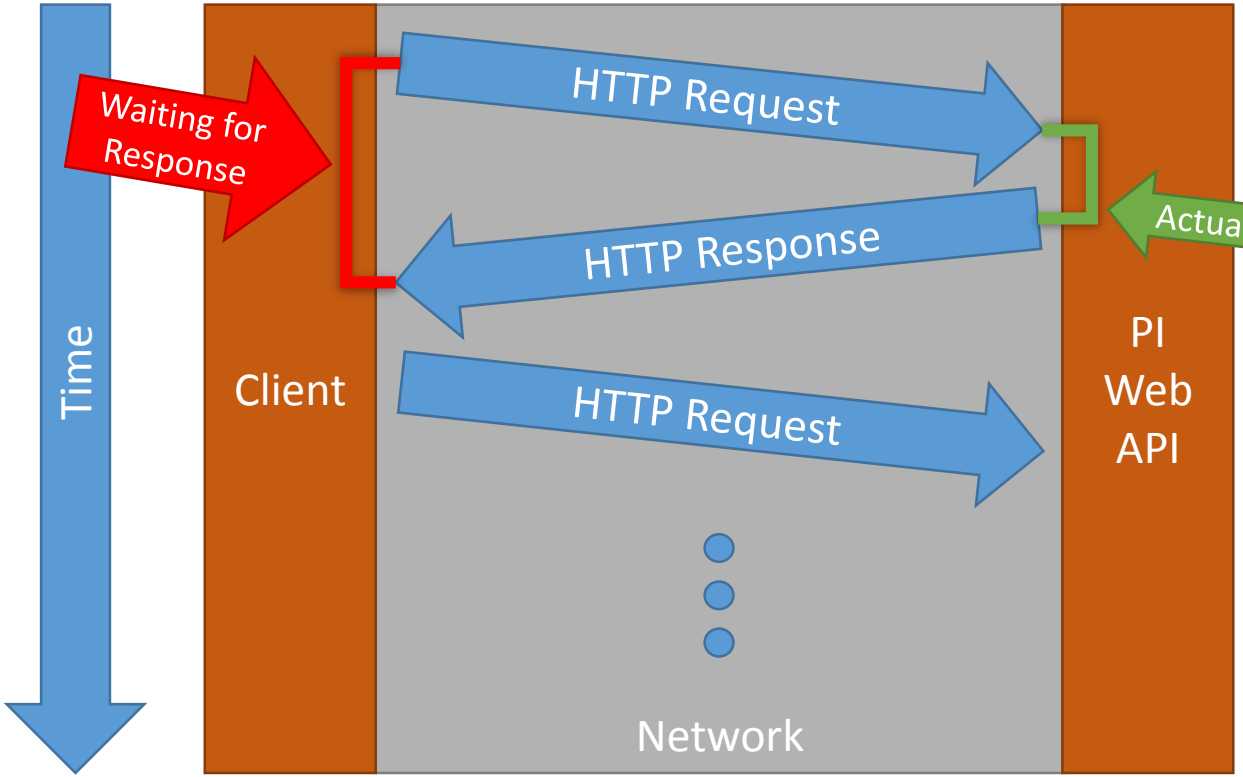
LIVE CODING

Example Application

PI Web API HTTP request lifecycle



What are the constraints of this lifecycle?



- Every request incurs a performance penalty
- Network latency & bandwidth
- Ethernet, TCP/IP, and TLS: reduces throughput to 87.7% under ideal conditions
- Calls to other services (identity provider, AF server, etc.)
- We don't control these

What *do* we have control over?

- The resources we interact with
- The way we interact with them
  - Which endpoints we call
  - How frequently we call them
  - What communication mechanism we use

Optimize!

- Requests can’t escape latency – so make fewer requests
- Bandwidth is finite – use less of it
- Server has finite resources – use them more efficiently

Areas of Improvement

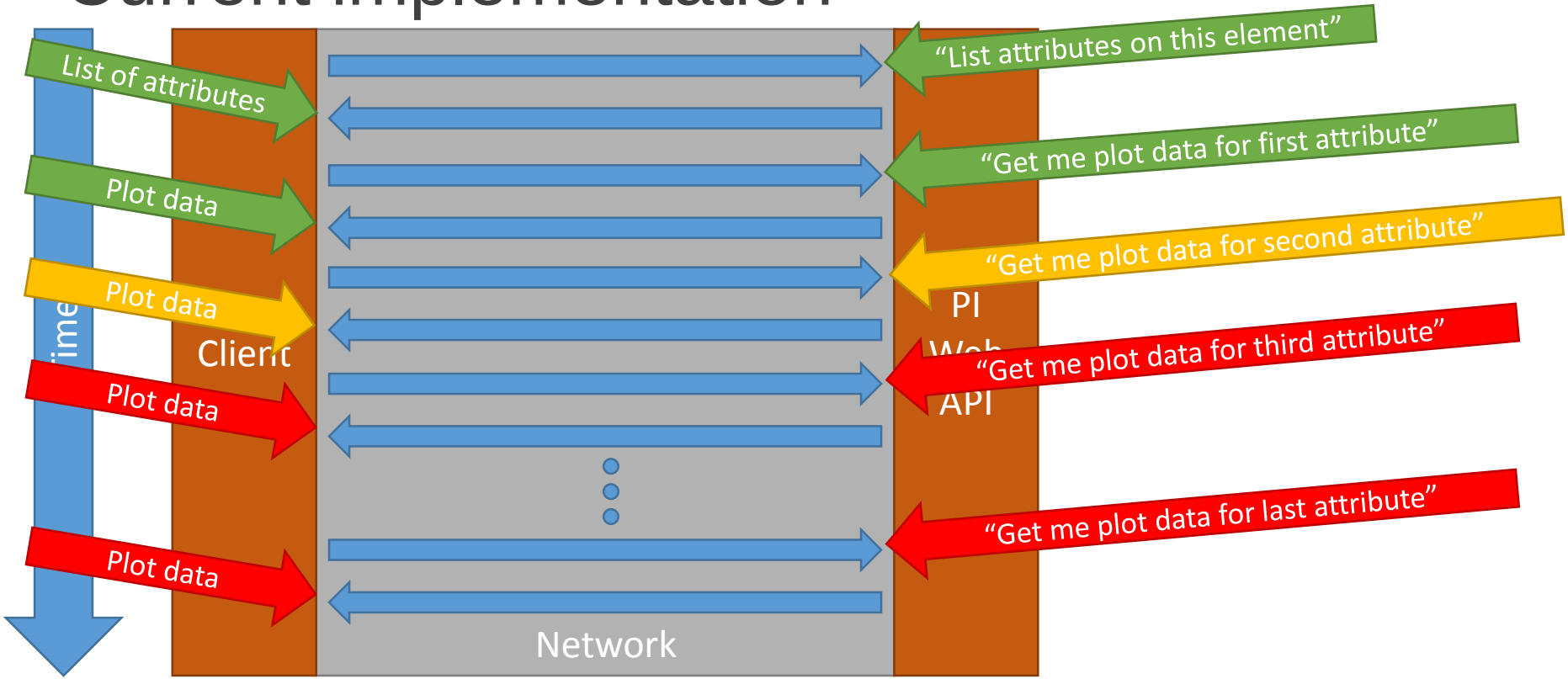
Endpoints & Resources

PI Web API Endpoints

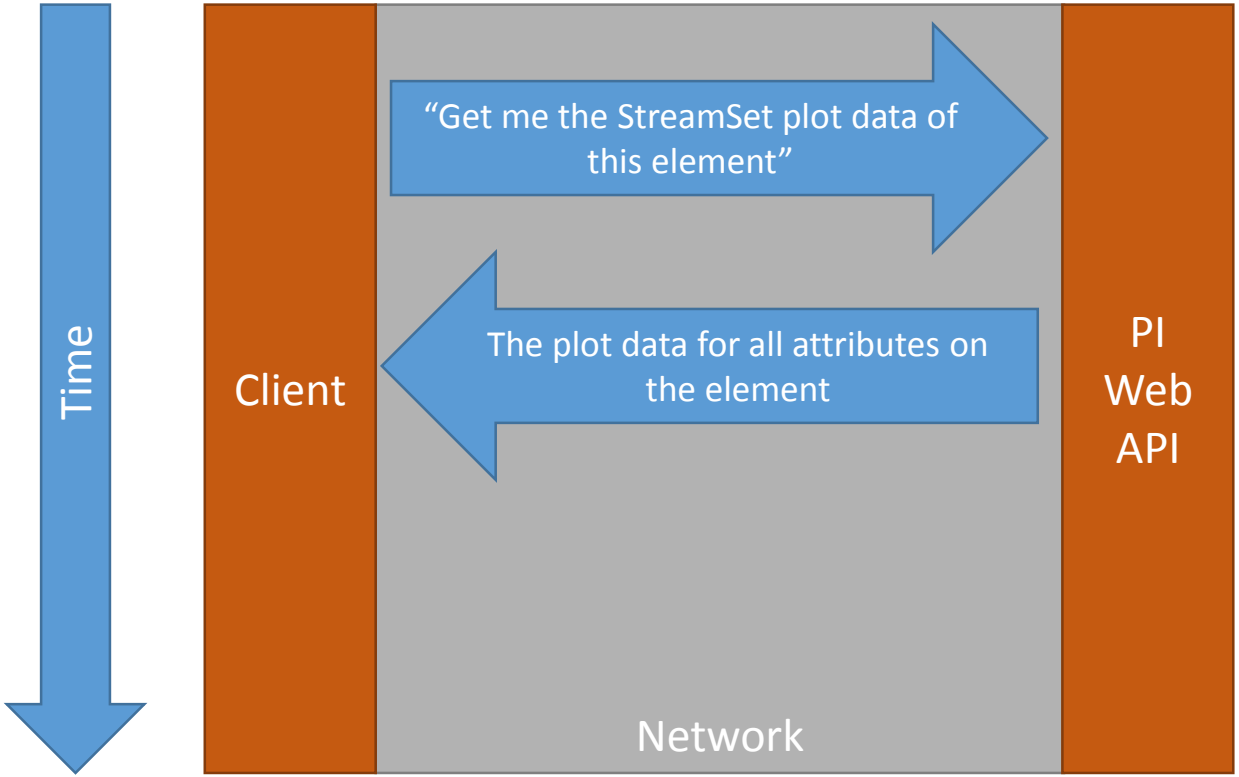
- Some endpoints are designed to improve performance
  - *What* are they?
  - *How* do I use them?
  - *Which* one is appropriate for my use case?

Plot Values

Current implementation



PI Web API features: Stream Sets

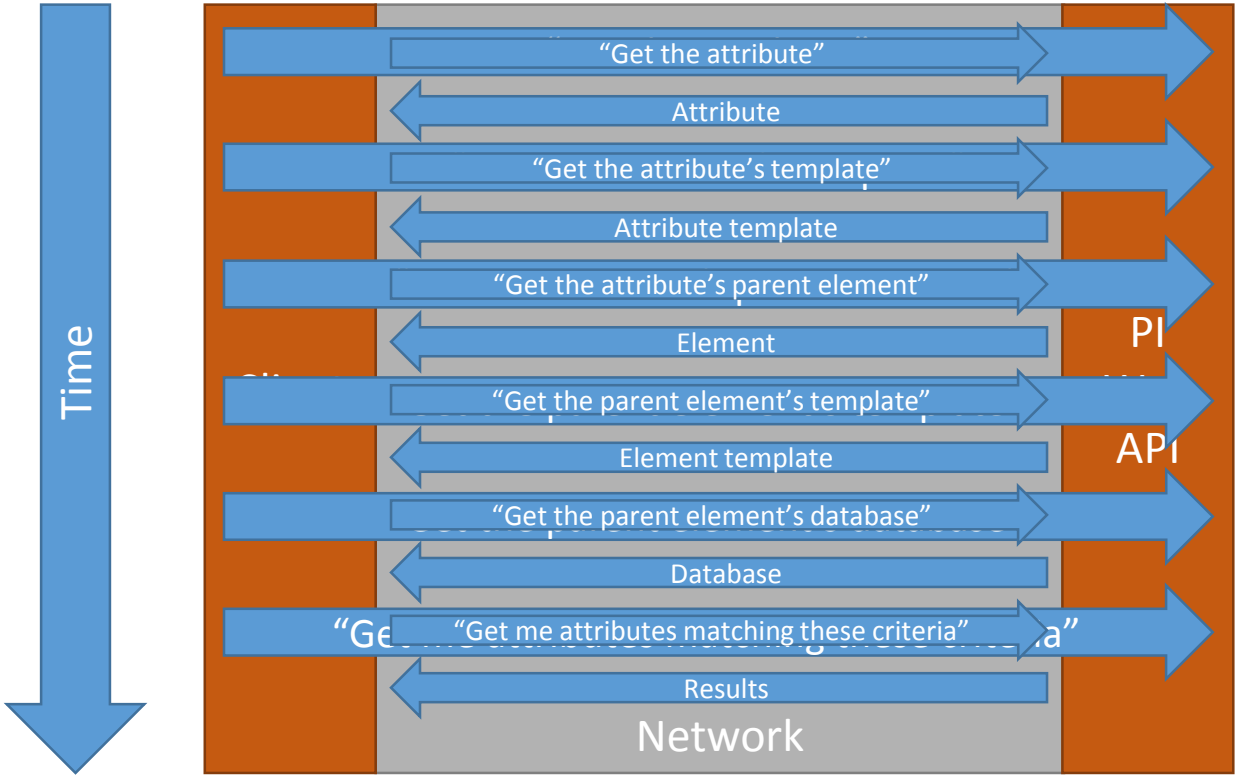


- Use a StreamSet request
- Reduced to a single HTTP request
- Many round-trips removed
- PI Web API can optimize backend calls for even better performance

LIVE CODING

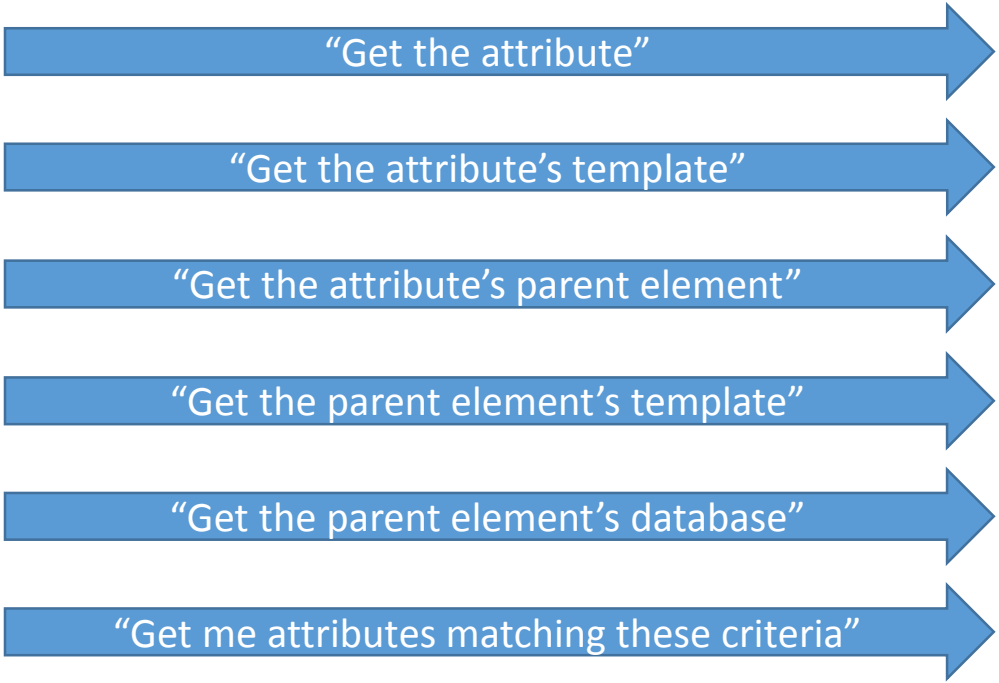
Streamsets

Different example, same problem



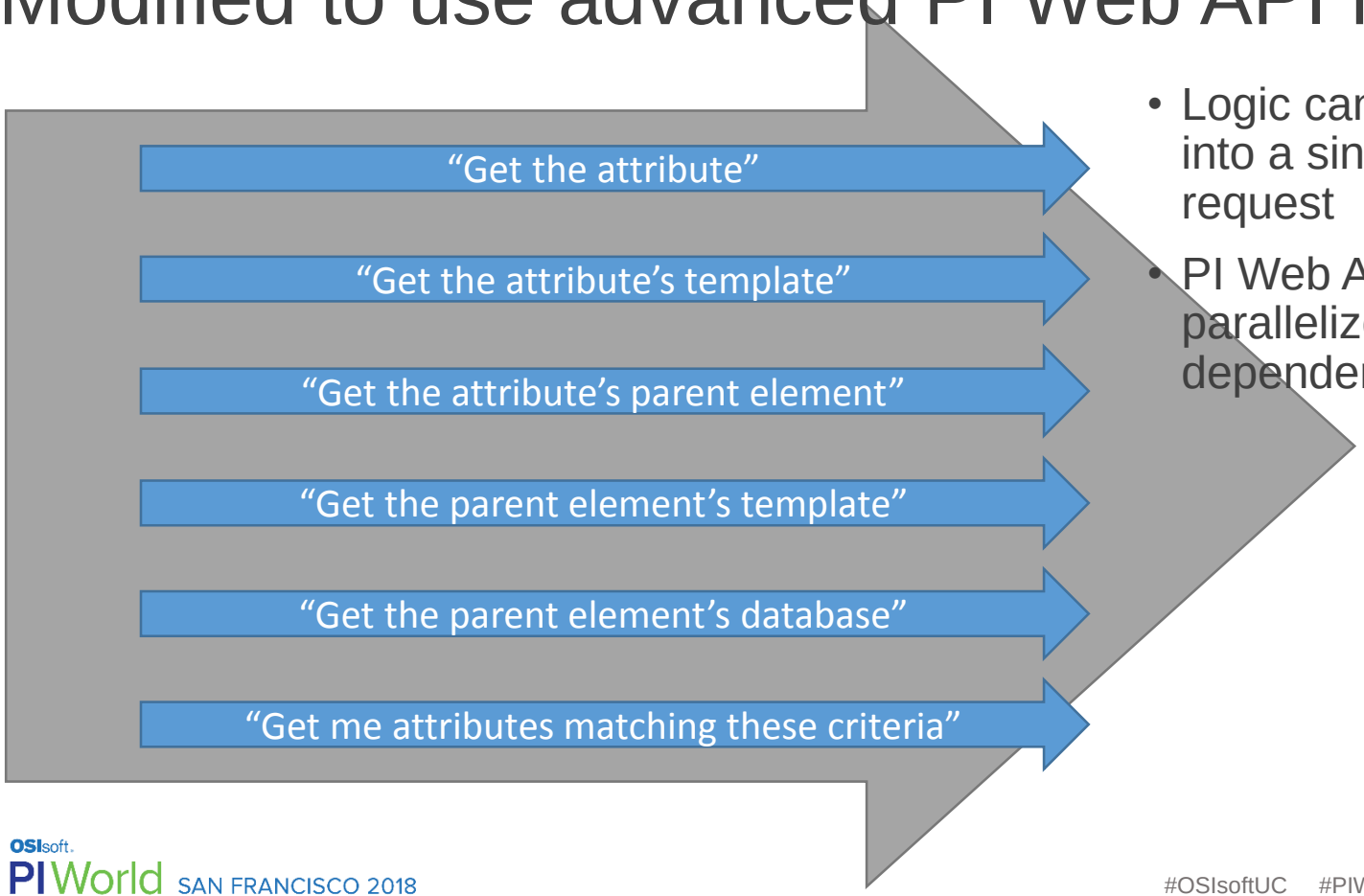
- "Given an data point on a car, get the same data point on the other cars"
- "Given an attribute, get all attributes using the same attribute template"
- Robust implementation ends up taking 6 requests

Modified to use advanced PI Web API features



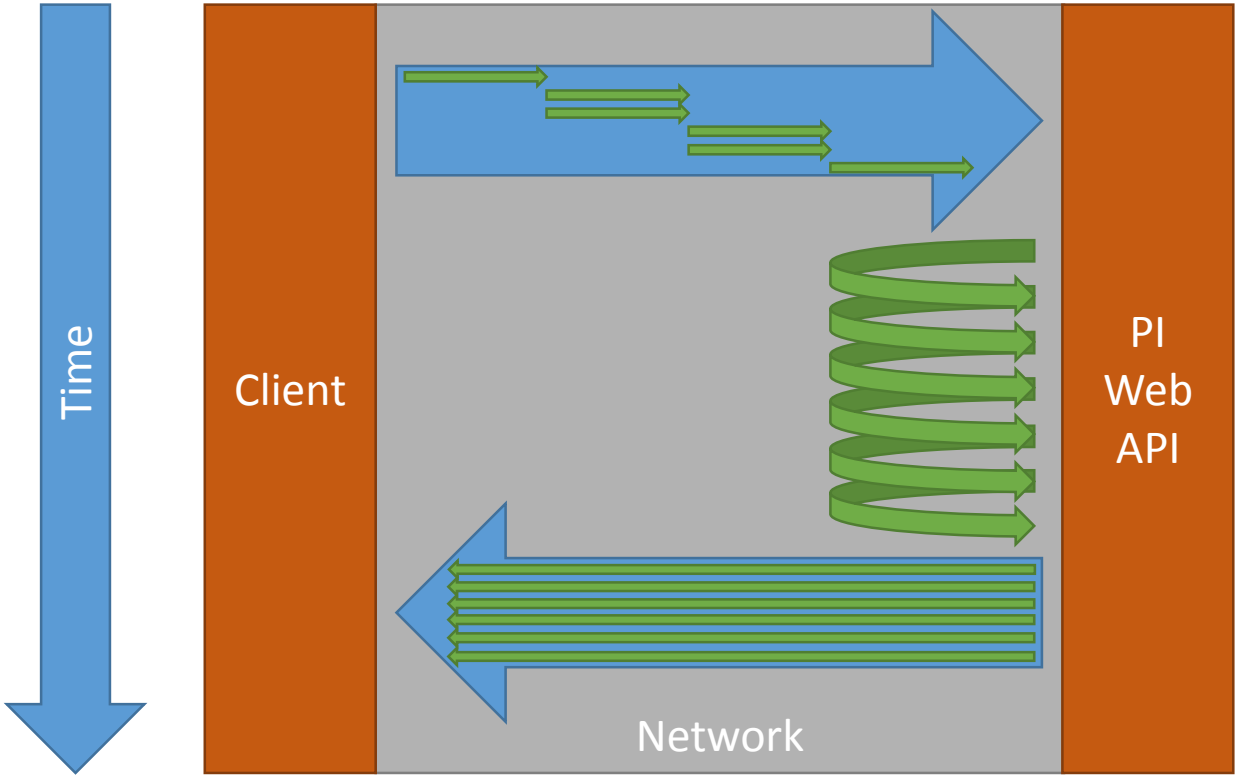
- Logic can be bundled into a single Batch request

Modified to use advanced PI Web API features



- Logic can be bundled into a single Batch request
- PI Web API can parallelize non-dependent requests

Modified to use advanced PI Web API features



- Batch subrequests are executed without needing to traverse the network
- Results of the subrequests are sent as a single response
- Now we only need one round-trip: five removed! Free performance!

LIVE CODING

Batch

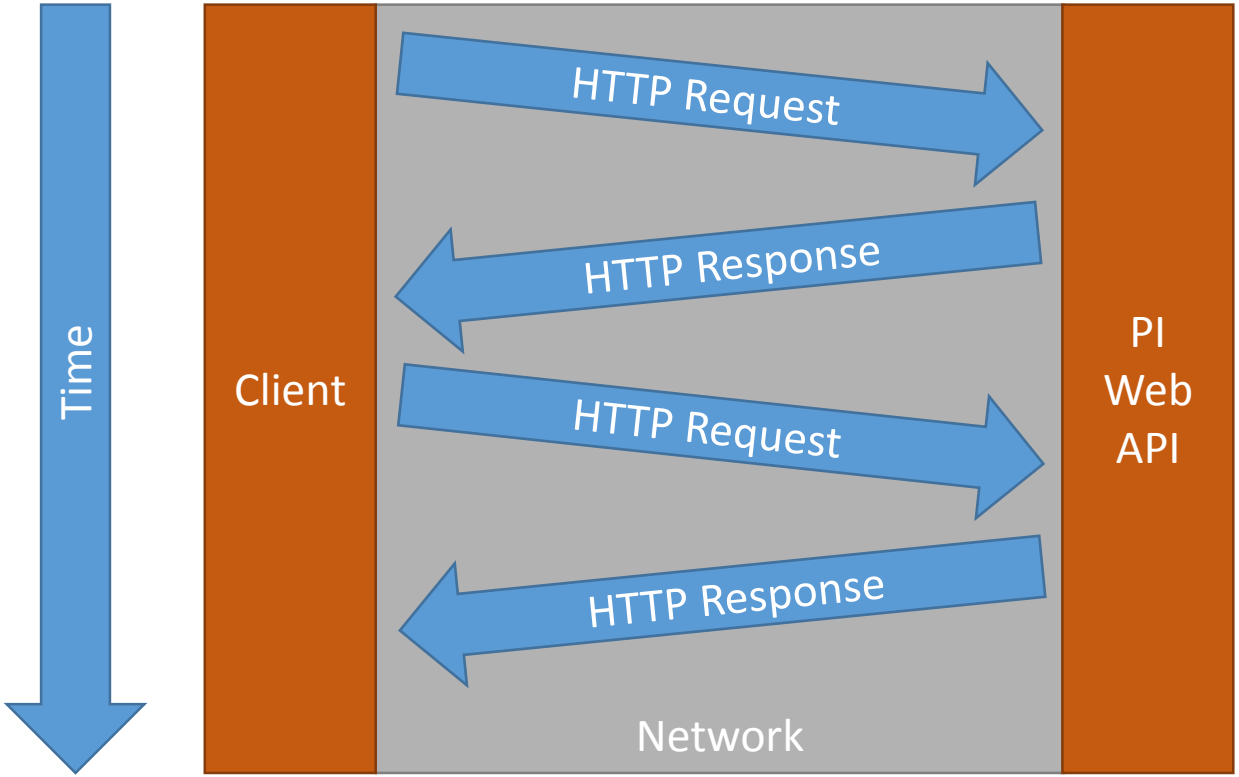
Areas of Improvement

Communication Mechanisms

Communication mechanisms

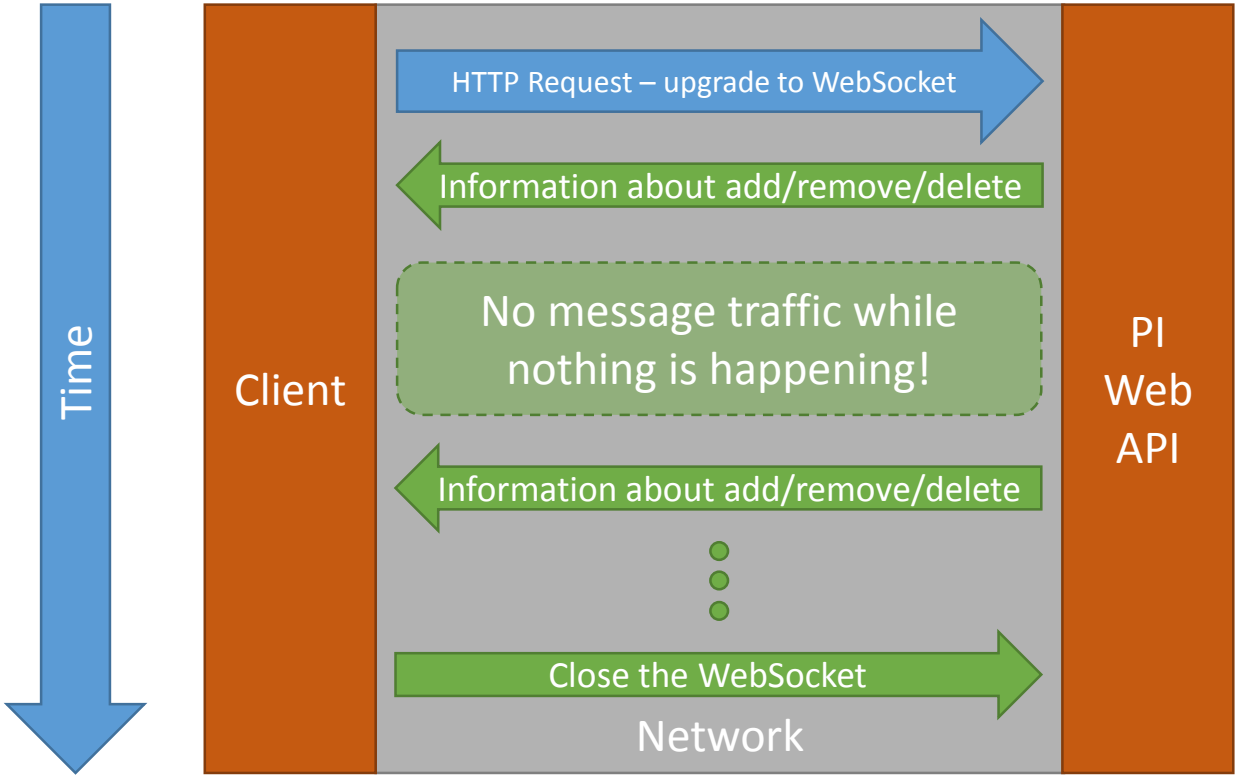
- What are the options?
- Which one should I use?
- How do they impact my application?

HTTP Requests



- Pros:
  - Easy to use
  - Widely supported
  - Can make use of existing infrastructure (load balancers, analytics, caching, etc.)
  - Low hardware overhead
  - Low software overhead
- Cons:
  - Need to continually issue requests to find out about changes (polling)

WebSockets (using the Channels feature)

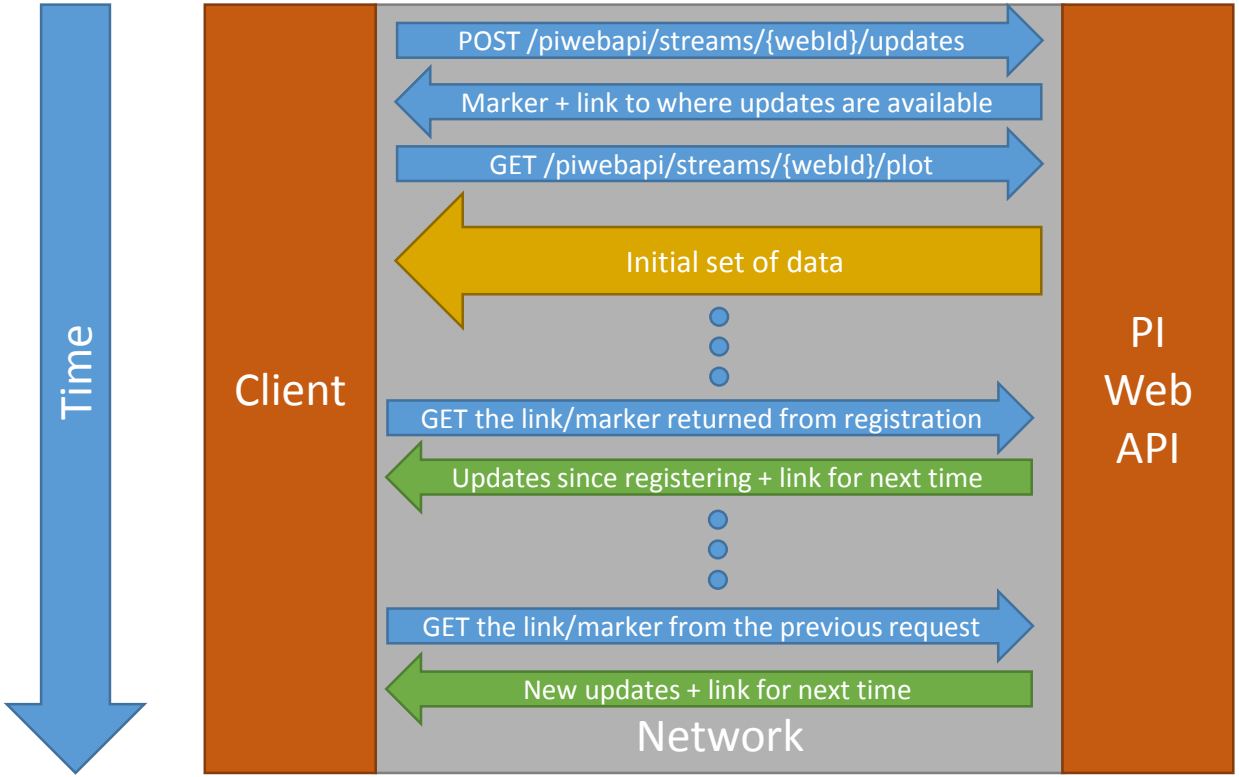


- Pros:
  - Get informed of changes as they occur: no polling needed
  - Lower latency
  - Less protocol overhead: never need more than 14 bytes per frame (vs. HTTP headers – still suffer from TCP/TLS/etc.)
  - Asynchronous model – not wasting hardware or network resources
- Cons:
  - Need client support
  - Underlying TCP connection still has network traffic
  - *PI Web API specific*: Does not support Claims Based Authentication

LIVE CODING

Channels

Stream Updates (CTP)



- Registers the stream or streamset to be monitored for changes
- Every time you request the updates, you get the changes since the time you registered and a new link to use next time
- Pros:
  - Operates over HTTP – get all the benefits of normal HTTP requests (infrastructure, library support, etc.)
  - Response sizes are much smaller than polling (only getting changes)
  - Uses less server & network resources than polling
  - Works with Claims Based Authentication
- Cons:
  - Client application needs to actively check for changes (not as easy as Channels)
  - Registrations are per Web API instance (need sticky sessions)

LIVE CODING

Stream Updates

감사합니다

谢谢

Danke

Merci

Gracias

Thank You

ありがとう

Спасибо

Obrigado

Contact Information:
- Jim Bazis: jbazis@osisoft.com
- Max Drexel: mdrexel@osisoft.com

Source Code:
- https://github.com/osisoft
- PI-Web-API-PIWorld-2018-highly-performant-applications

New AFSearch Functionality

- Introduced as part of PI Web API 2017 R2
- Uses AF Search syntax
- As of PI Web API 2018, the following search types are supported:
  - Analyses
  - Analysis Templates
  - Attributes
  - Elements
  - Event Frames
  - Notification Rules
  - Notification Rule Templates
- Much better performance, especially for use cases where users frequently re-execute searches
- Uses fewer resources across the PI System