DEUTSCHE BÖRSE
GROUP

API != HTTP

Building services and APIs using AMQP 1.0

Jakub Scholz, Architect, Risk-IT

APIs and µServices

APIs

- APIs define how should different software components interact
  - Operating system and library APIs are used to link together different software components into single program
  - Remote APIs are used for communication between different systems using defined protocols
- We use APIs everywhere and they have been with us „forever“

µServices

- Cloud native and microservice based architectures causes a significant shift in what APIs we use
  - Applications are not anymore a single process running on single machine
  - Application is a set of services distributed across the cloud
- Most APIs between services are using standardised communication protocols

HTTP(S)

- HTTP(S) usually the first choice protocol

Advantages

- Easy to use
- Supported almost everywhere
  - Most platforms (computers, smartphones, TVs, …)
  - Most operating systems
  - Most programming languages
- Stateless
- Scalable
- Secure (Encryption, Authentication / Authorization)
- World-wide scale
  - Can be used across the world
  - Works well across organizational boundaries

HTTP(S)

- If HTTP(S) is so great, why should we consider something else?

Problems

- Limited number of supported communication patterns
  - Request – response pattern only
  - No push communication
- Often used with inefficient data encodings (JSON)
- Stateless
- Short lived TCP connections
- Limited support for request pipelining
- Synchronous communication
- No flow control

AMQP

- Open standard
- Asynchronous
- Reliable delivery

- High throughput
- Low latency
- Push messages

- Flow control
- Security
- Binary

AMQP 1.0

- Open messaging protocol
- Core protocol approved as ISO 19646
- Further enhancements are being developer and standardized
- Competing products from Apache, Red Hat, Microsoft, IBM etc.
- Used across many industries (Finance, IoT, eGovernment, …)

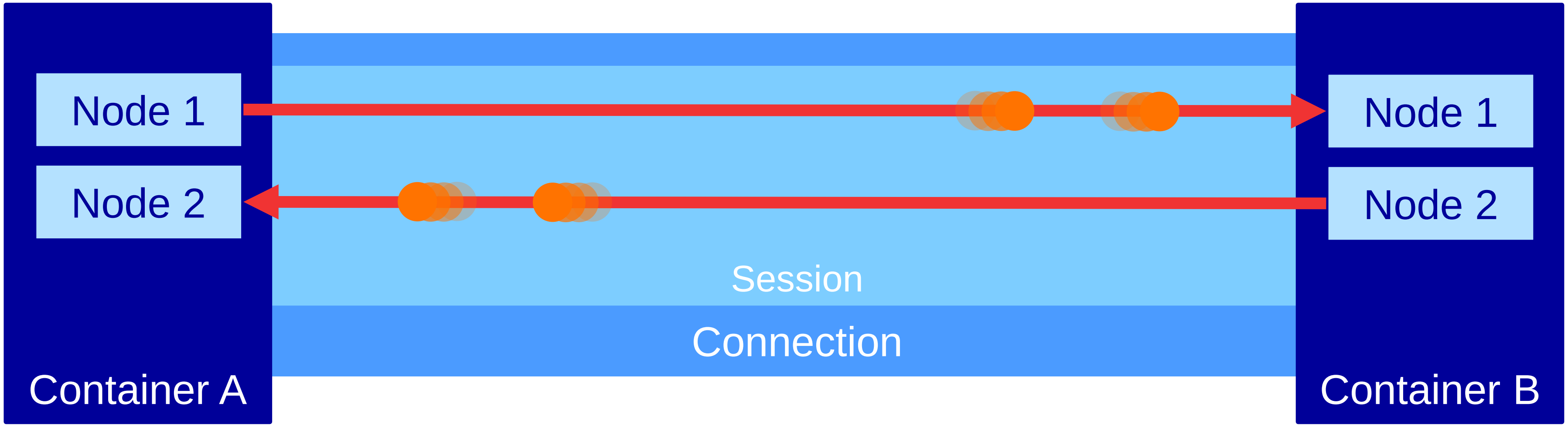
Older versions

- *Limited number of product implementing older versions*
- *Asymmetric protocol (broker, queues, …)*
- *This talk is only about AMQP 1.0*

AMQP

AMQP 1.0

- Designed as symmetric protocol
  - Defines transfer of messages from point A to point B
  - No brokers, queues, exchanges or bindings are required
  - Opens new possibilities, enables new use cases
- Containers represent different applications / clients
- Nodes are message consumers or producers which exist within a container



AMQP

- AMQP has many advantages …

Advantages

- Supported on many different platforms and operating systems
- Clients available for main programming languages
- Scalable
- Secure (Encryption, Authentication / Authorization)
- World-wide scale
  - Can be used across the world *
  - Works well across organizational boundaries *
- **Configurable reliability (at-most-once, at-least-once, exactly-once)**
- **Good support for different encoding types**
- **Long-lived connections**
- **Truly asynchronous communication**
- **Flow control and message pipelining**
- **Many different communication patterns**

**) Standardization is „work in progress“ at OASIS*

AMQP

- And some disadvantages …

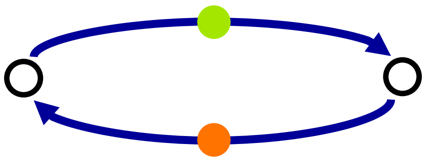
Problems

- More complex than HTTP(S)
  - Not as easy to use
  - Takes longer to learn
- Same additional enhancements to the core protocol are still „work in progress“ (Management, addressing, …)
- Might be not available in some less common programming languages
- Integration into different frameworks / toolkits is often lacking

Basic communication patterns

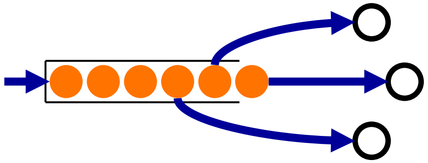
Request Response

- Fully asynchronous
- Can be combined with other patters
- Response can consist of multiple messages



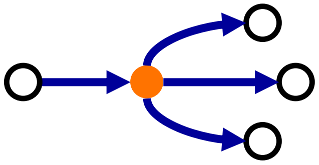
Queueing

- Pull pattern
  - Receivers pull messages „on their own pace“
  - Distribution of tasks to workers
- Provides decoupling between sender and receiver



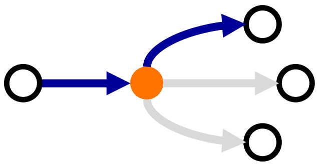
Multicast

- Send message once, deliver many times
- Automatically push updates from server to clients
  - No polling for updates needed anymore
  - No more waiting for refresh



Competing Consumers

- Multiple consumers competing for messages
  - Scalability
  - Load balancing



Demo

Live Score service

Service

- Add games
- Update scores
- See latest scores

Source: https://www.flickr.com/photos/twirlingumbrellas/9994289386/ (CC BY-NC-ND 2.0)

Live Score service

- How would you implement Live Score service with HTTP?

API

- GET /api/v1.0/score -> Get list of games and their scores
- POST /api/v1.0/score -> Add new game
- PUT /api/v1.0/score -> Update game score

HTTP

- HTTP server is easy to implement in most languages / frameworks
  - https://github.com/scholzj/livescore-demo-vertx-http
- Client have to do polling for score updates
  - No simple enough way to push updates from the server

Live Score service

- How could you implement Live Score service with AMQP?

API

/getScore	-> Get list of games and their scores
/addGame	-> Add new game
/setScore	-> Update game score
/liveScore	-> Broadcasting of score updates

AMQP

- There is no „single right way“ how to build AMQP based APIs
  - REST style
  - CRUD style
  - RPC style
- AMQP gives you possibility to push updates to clients
  - When score is updated, a message is broadcasted to all connected clients
- Binding between WebSockets and AMQP
  - Allows to use AMQP directly from the browser

Live Score service

- How do you build AMQP Server?

AMQP Server

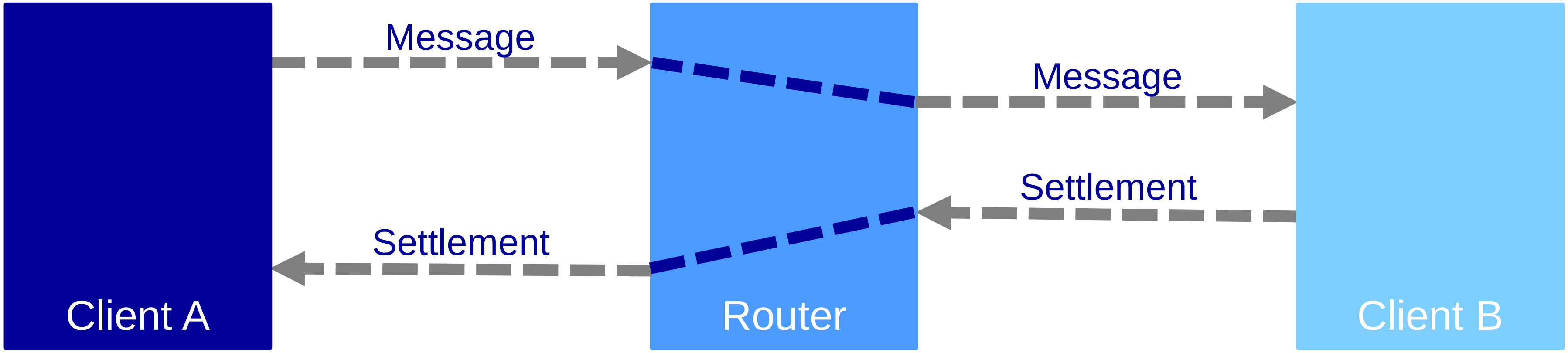
- AMQP server is not part of most languages / frameworks
- AMQP server doesn‘t have to be a message broker
  - No need to provide queues or do message routing
  - Accepts connections from clients and handles API calls
- What does AMQP server do?
  - Protocol encoding / decoding
  - Network I/O handling
  - Encryption
  - Authentication & Authorization
- Apache Qpid Proton can be used to provide basic elements
  - Native in C and Java
  - Bindings for C++, Go, Python, Perl, PHP, Ruby, JavaScript, …

Live Score service

- What if you don‘t need a server?

AMQP Router

- Use message router as your AMQP server
  - Router never takes ownership of messages
  - No decoupling of clients or storing of messages
  - Messages are routed directly between producer and consumer
  - Settlements are routed back from consumer to producer
  - Apache Qpid Dispatch

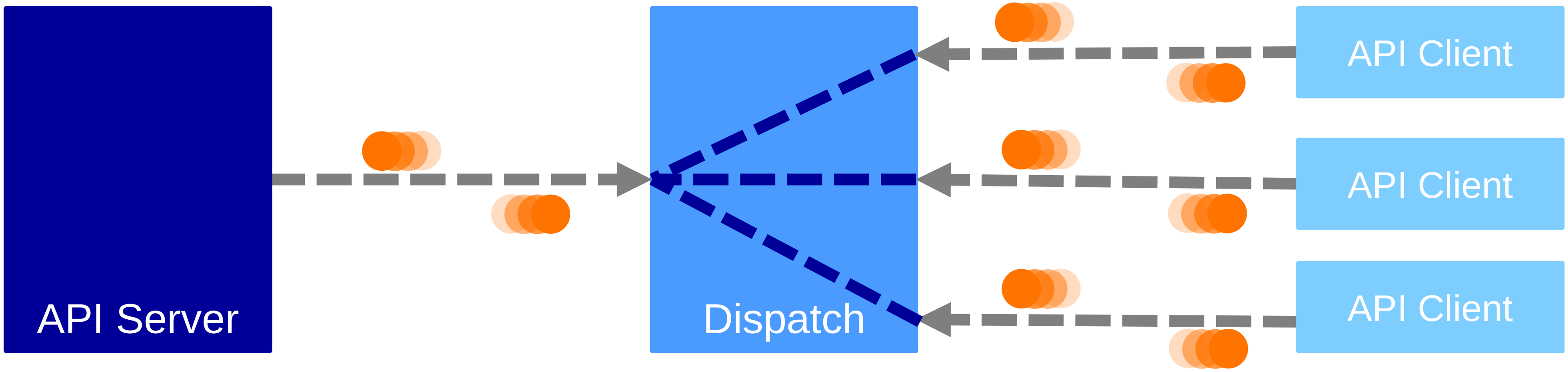


Live Score service

- What if you don‘t need a server? (continued)

AMQP Client

- Combining a router with AMQP client creates AMQP based API
  - Router accepts AMQP connections from clients and takes care of networking and security
  - API server acts as another AMQP client and connects to the router
  - API calls are received by the router and forwarded to the client
  - Client handles API calls and responds to them through the router
  - If needed, both the router and API server can be scaled



Which one to choose?

Which one to choose

HTTP

- All you need is something simple
- Many clients with light usage
- External users used to HTTP

Source: https://www.flickr.com/photos/aai/6936657289/ (CC BY-NC-ND 2.0)

Which one to choose

AMQP

- Make use of the advanced technical features
- Make use of the advanced communication patterns
- Smaller number of clients with heavy traffic

Source: https://www.flickr.com/photos/ozlady/3509831100/ (CC BY-NC 2.0)

Jakub Scholz

Thank you

Slides and demos:	https://github.com/scholzj/dbg-pti-building-services-and-api-using-amqp
License:	CC-BY-SA-4.0

Twitter:	@scholzj
Blog:	http://blog.effectivemessaging.com
GitHub:	https://github.com/scholzj