TECH CONFERENCE
DotNet 2020

#DotNet2020

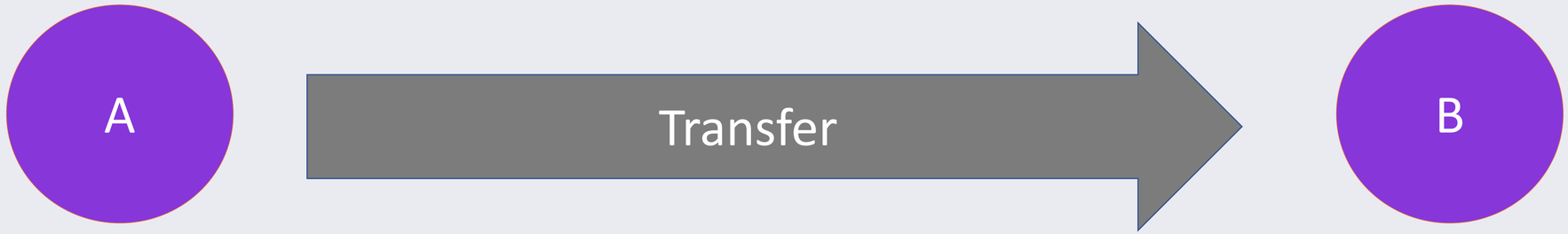
Distributed Messaging Patterns

Cecil L. Phillip AG
Senior Cloud Advocate @ Microsoft
@cecilphillip

Communication

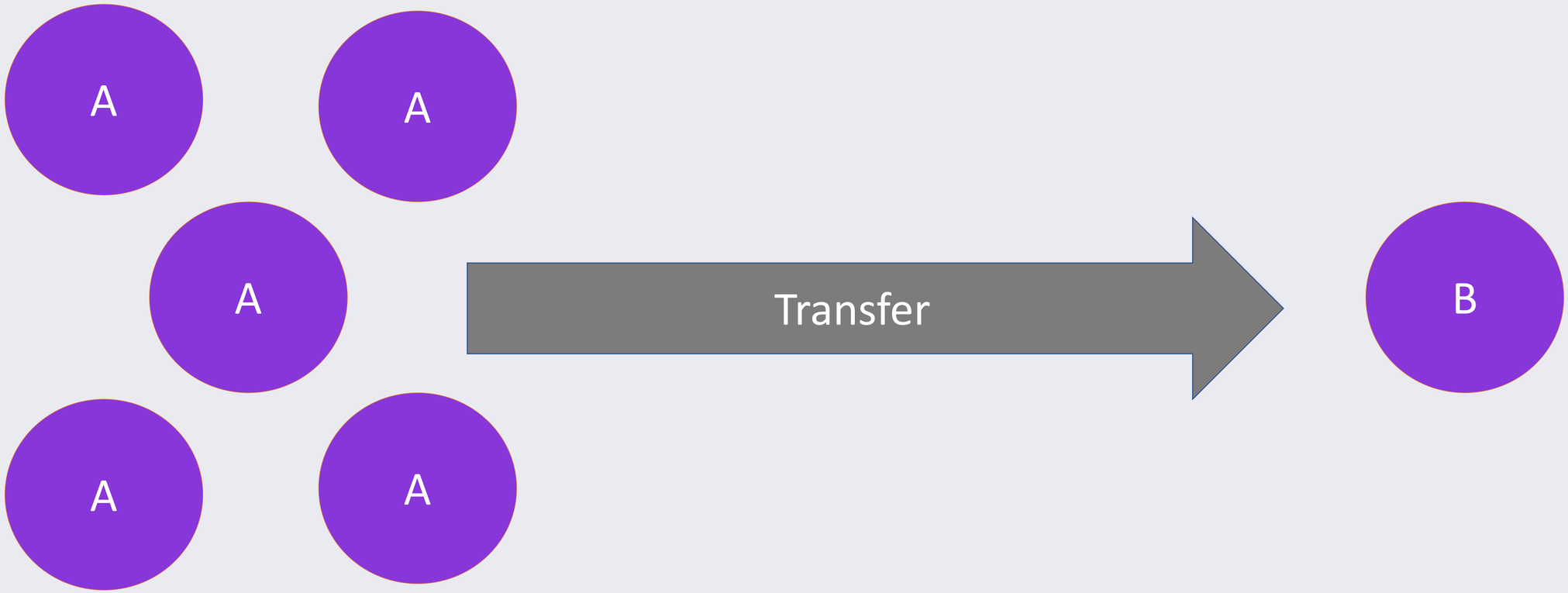
How do we share information between systems?

Direct	Brokered	Broadcast
Moving messages between two endpoints.	Moving messages via communication middleware	Distribution of messages from one source to anyone interested
“Phone”	“Mail”	“Radio”

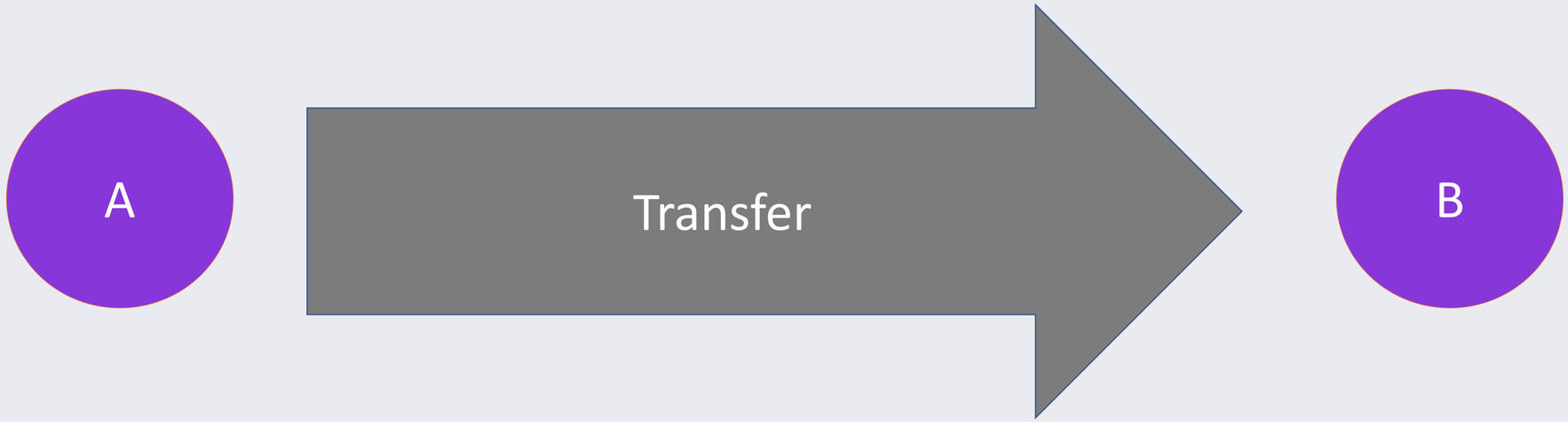


Messaging is all about getting data from here to there

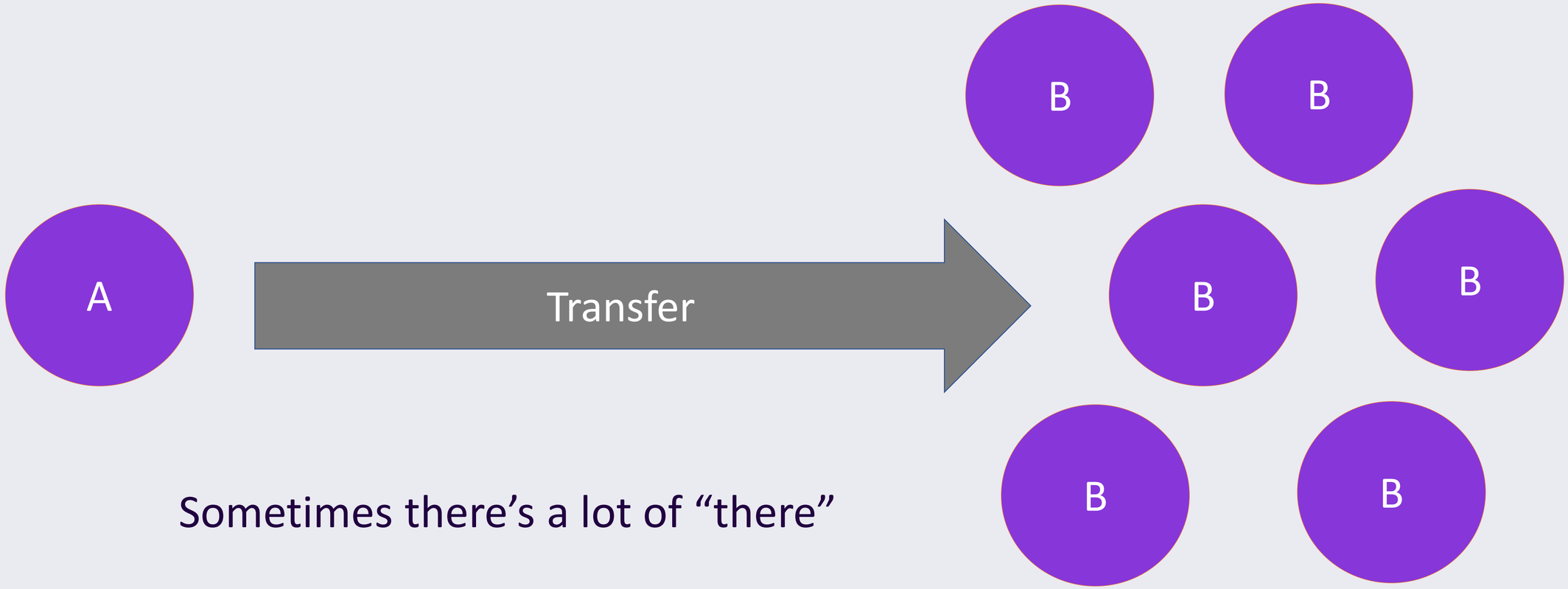
(Getting data back from there to here is the just same thing)



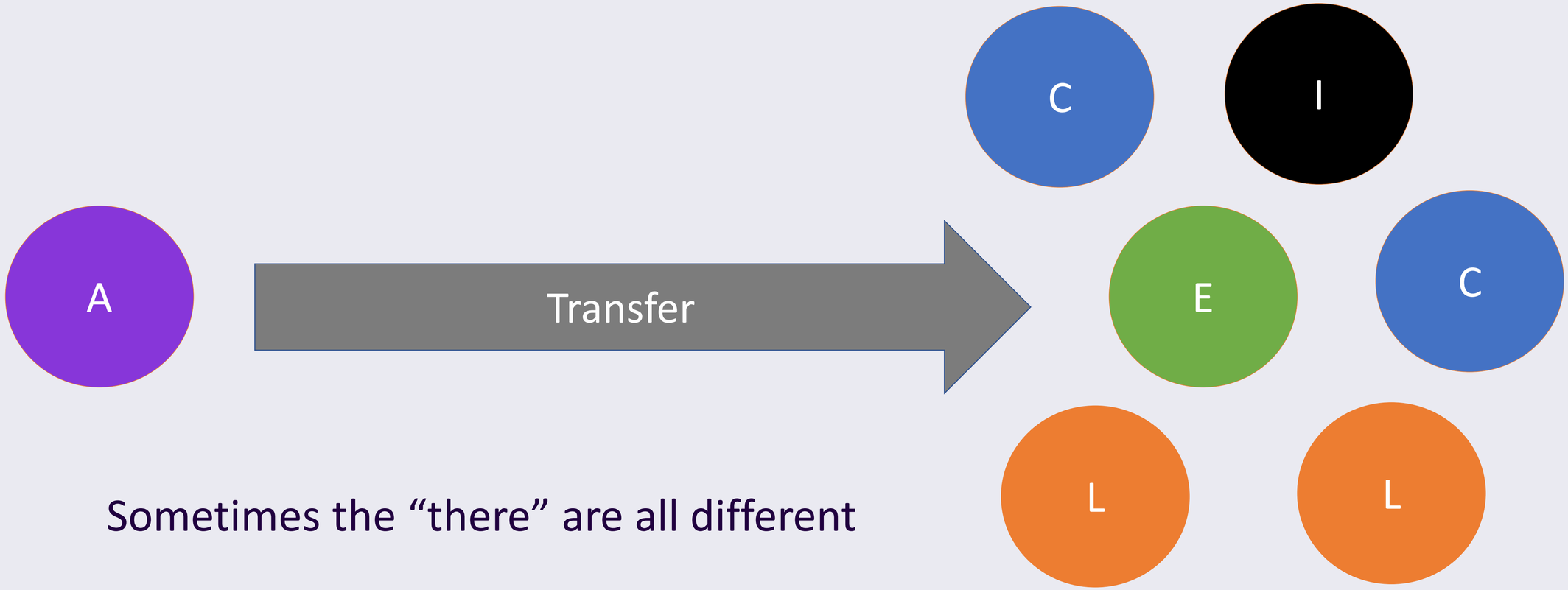
Sometimes there’s a lot of “here”



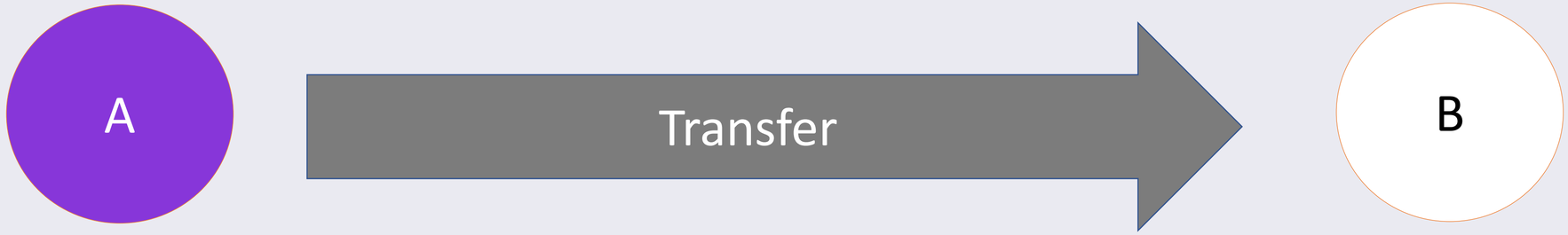
Sometimes there’s A LOT of data



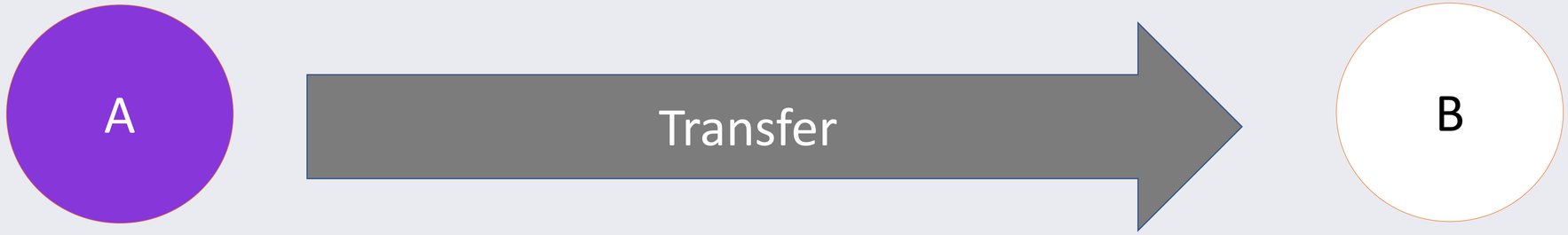
Sometimes there’s a lot of “there”



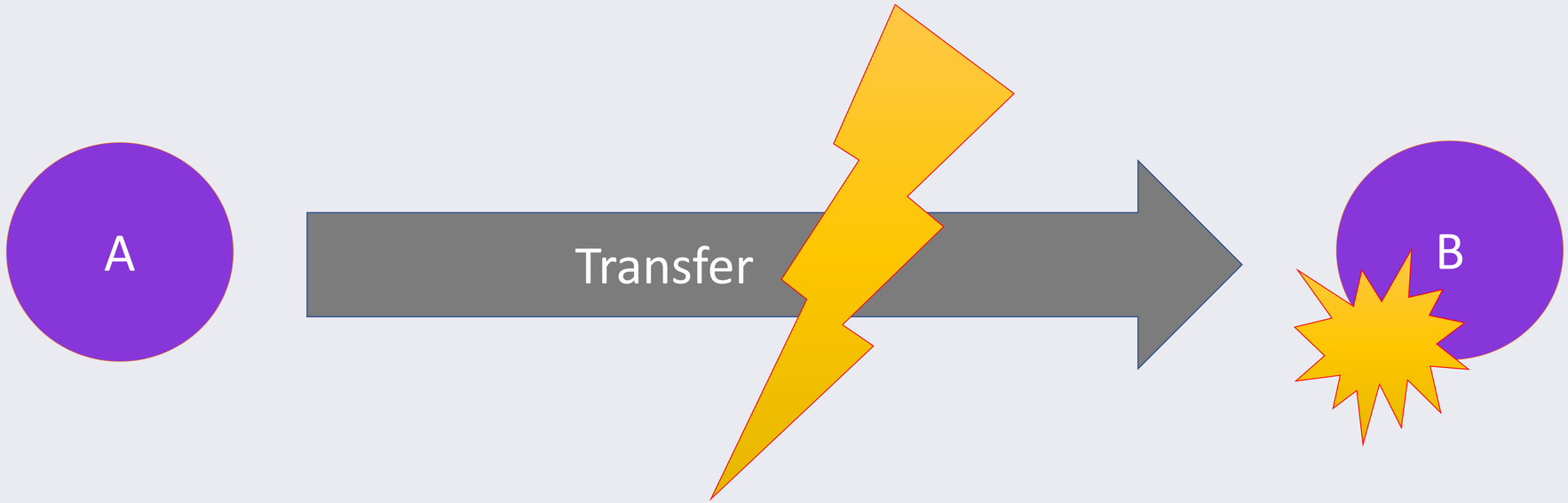
Sometimes the “there” are all different



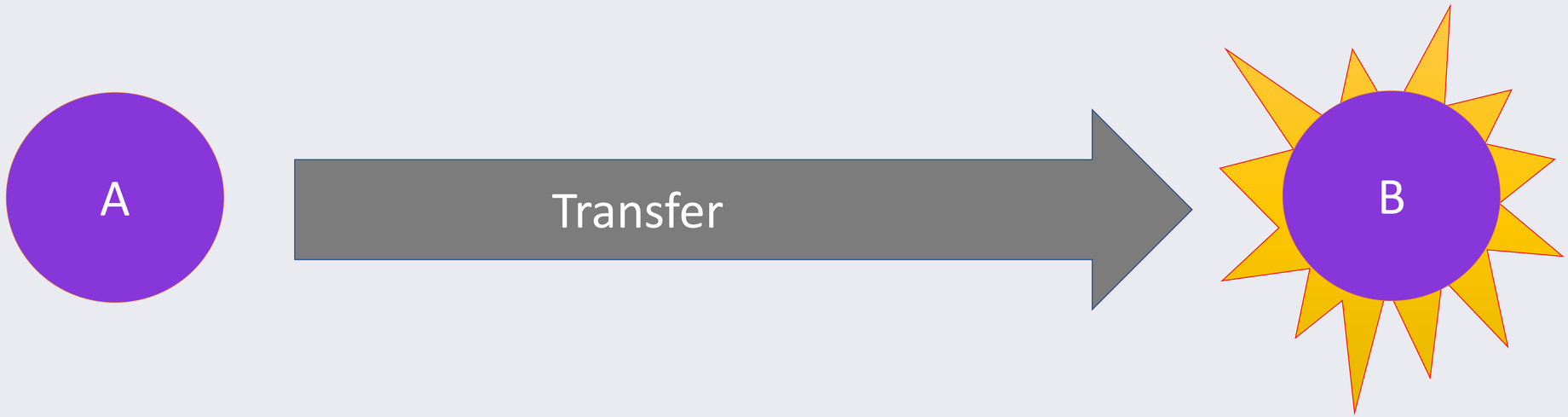
Sometimes “there” isn’t currently paying attention



Sometimes “there” isn’t currently paying attention



Sometimes there’s trouble



Sometimes “there” is VERY BUSY

Messaging Protocols

IETF HTTPS & WebSocket Protocol

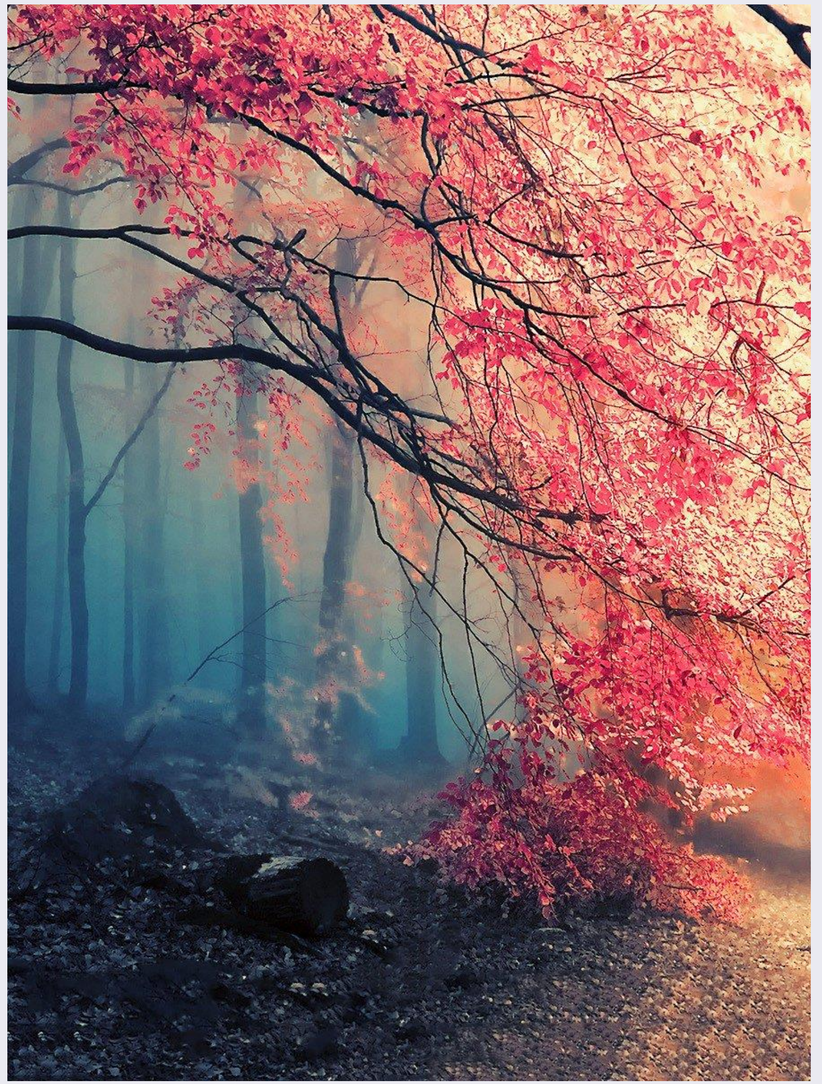
- Request-response application protocol
- OSS: Nginx, Apache HTTP Server, Kestrel, YARP, Envoy

OASIS AMQP 1.0

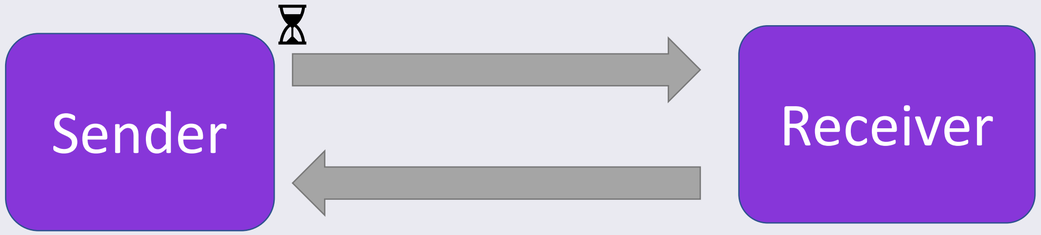
- Symmetric, reliable message transfer protocol with support for multiplexing and flow control
- OSS: Apache ActiveMQ, Apache Qpid Broker-J, Apache Qpid Dispatch Router, Apache Camel, Pivotal RabbitMQ

OASIS MQTT 3.x/5.x

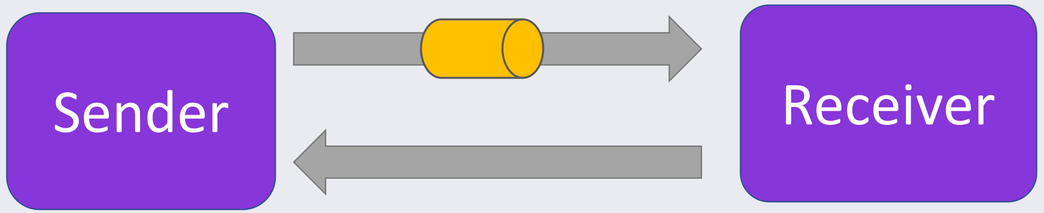
- Reliable publish-subscribe protocol for telemetry transfer and state synchronization
- OSS: Apache ActiveMQ, Eclipse Hono, Pivotal RabbitMQ, Eclipse Mosquitto, HiveMQ, VerneMQ, etc.



Synchronous vs. Asynchronous



- Sender sends request and then waits for an immediate answer
- May happen via asynchronous I/O, but the logical thread is preserved.

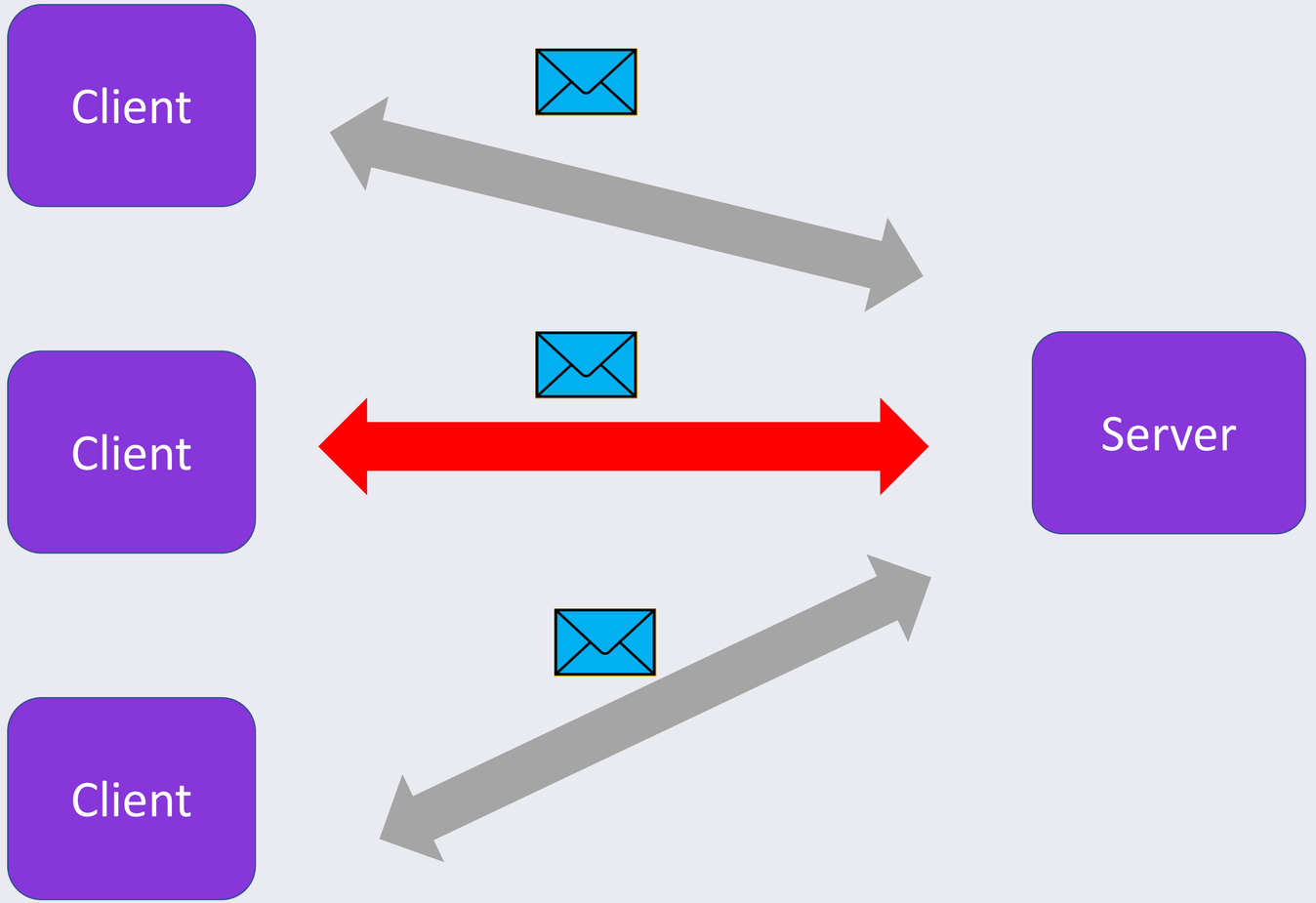


- Sender sends a message and proceeds to do other things.
- Replies may flow back on a separate path.

Synchronous Patterns

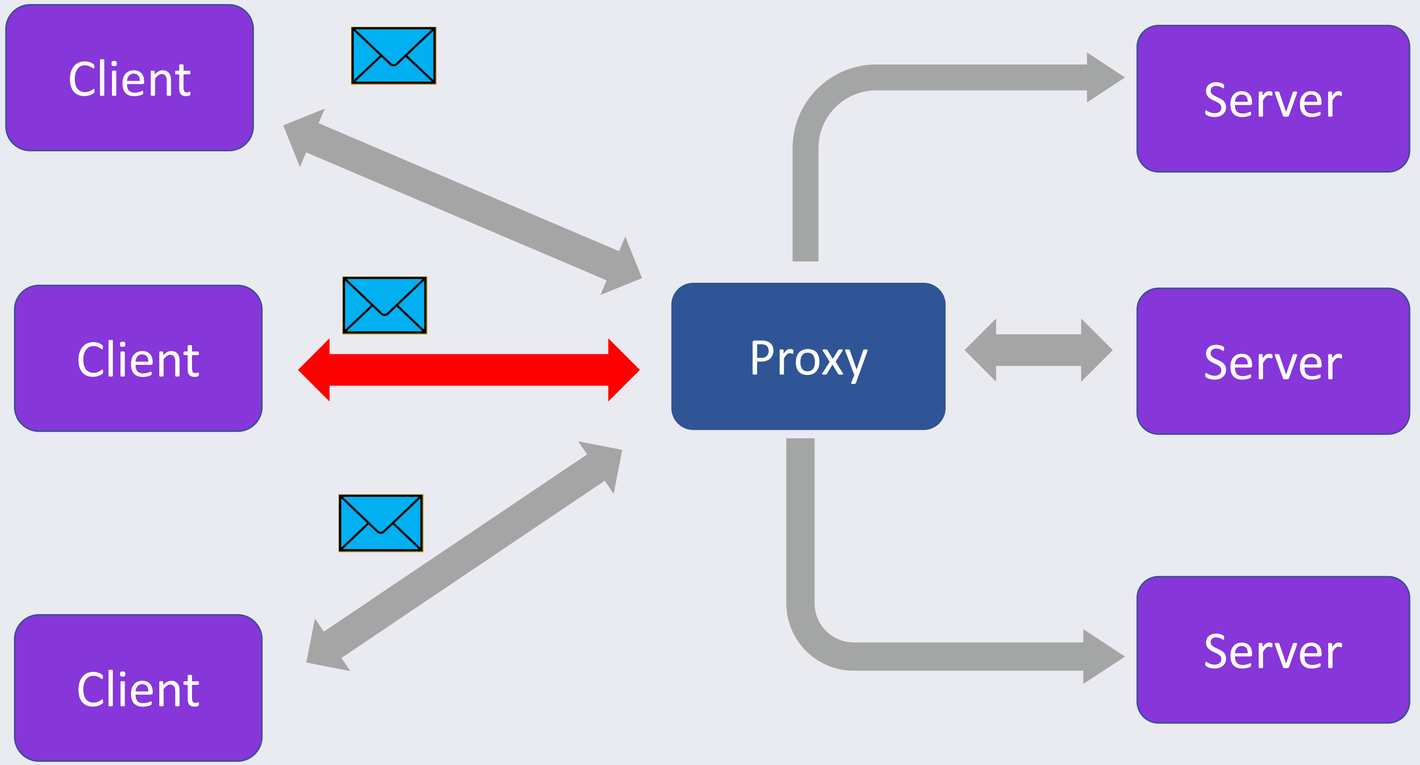
Throttling

- Request rates can be defined on per-client based on a given criteria
- Throttling state can be communicated via the transport protocol
- Clients should implement an adaptive retry strategy to get their work submitted



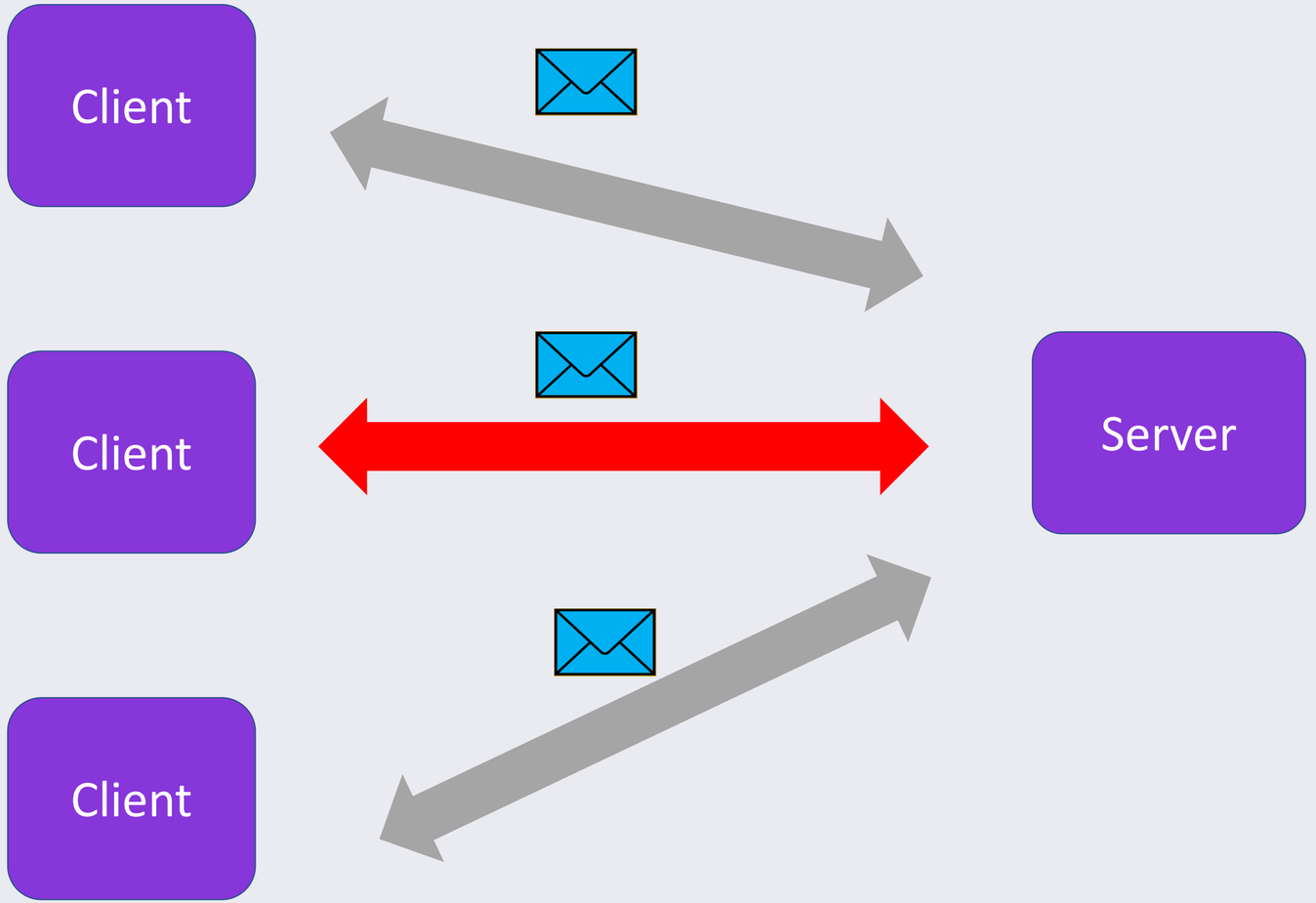
Load balancing w/ a Reverse Proxy

- Messages are distributed between multiple servers
- Clients only need direct access access the proxy
- The proxy can be enhanced with additional functionality via middleware



Retry Strategies

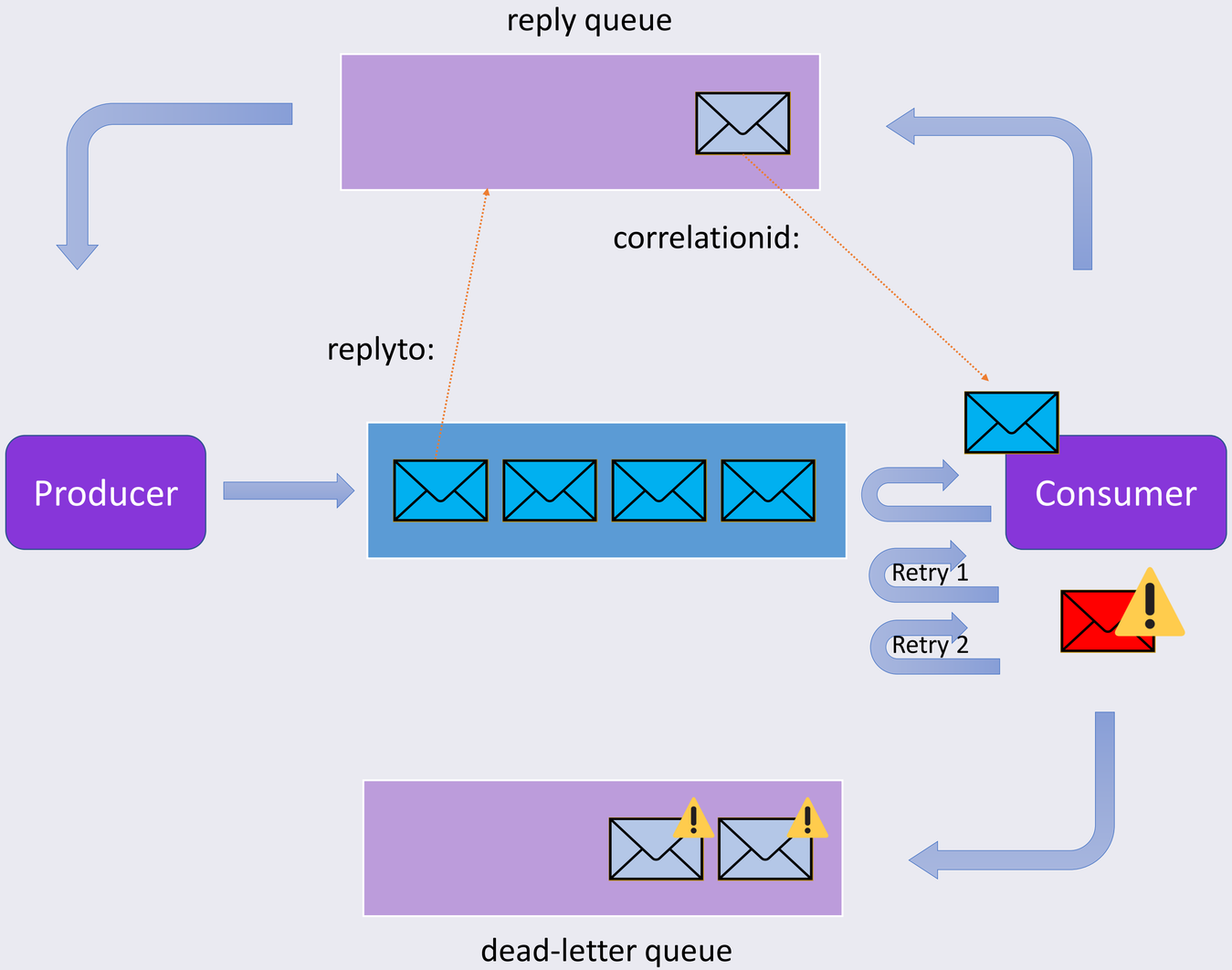
- Connectivity between systems can unreliable and faults can occur for various reasons
- Use reasonable delays between retries to prevent overloading the system
- Know when to stop retrying



Asynchronous Patterns

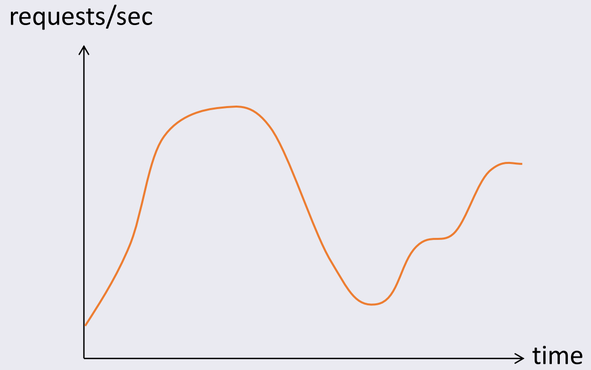
Long-Running Work

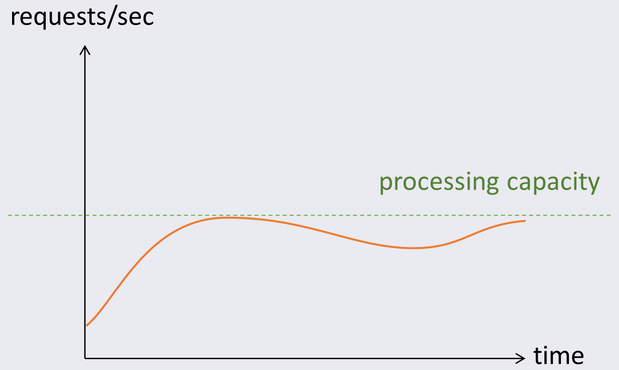
- Processing at the consumer may take very long (minutes to hours)
- Producers entrust jobs into a queue.
- Bad jobs are moved into a dead-letter queue for inspection.
- Flow back to the producer is performed through a reply queue

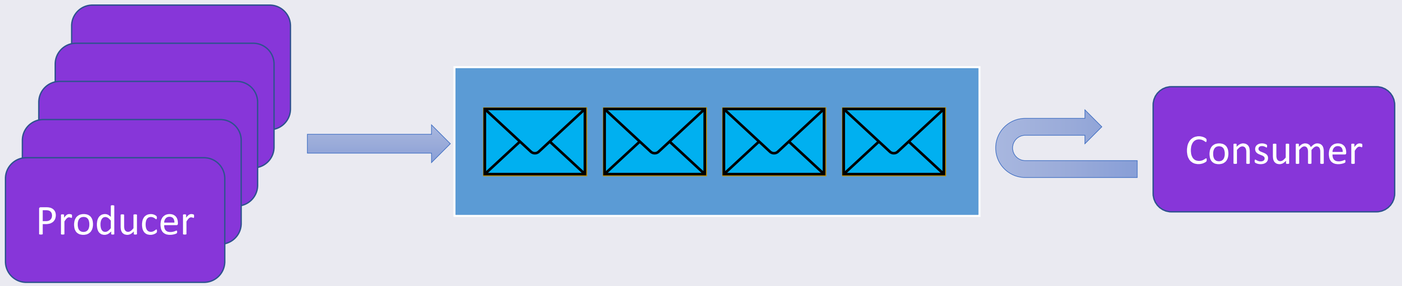


Load Leveling

- Queues act as an inbox for requests to a consumer.
- Consumer pulls work when it has capacity for processing.
- Consumers process at their own pace.
- No “too busy” errors, easier resource governance.

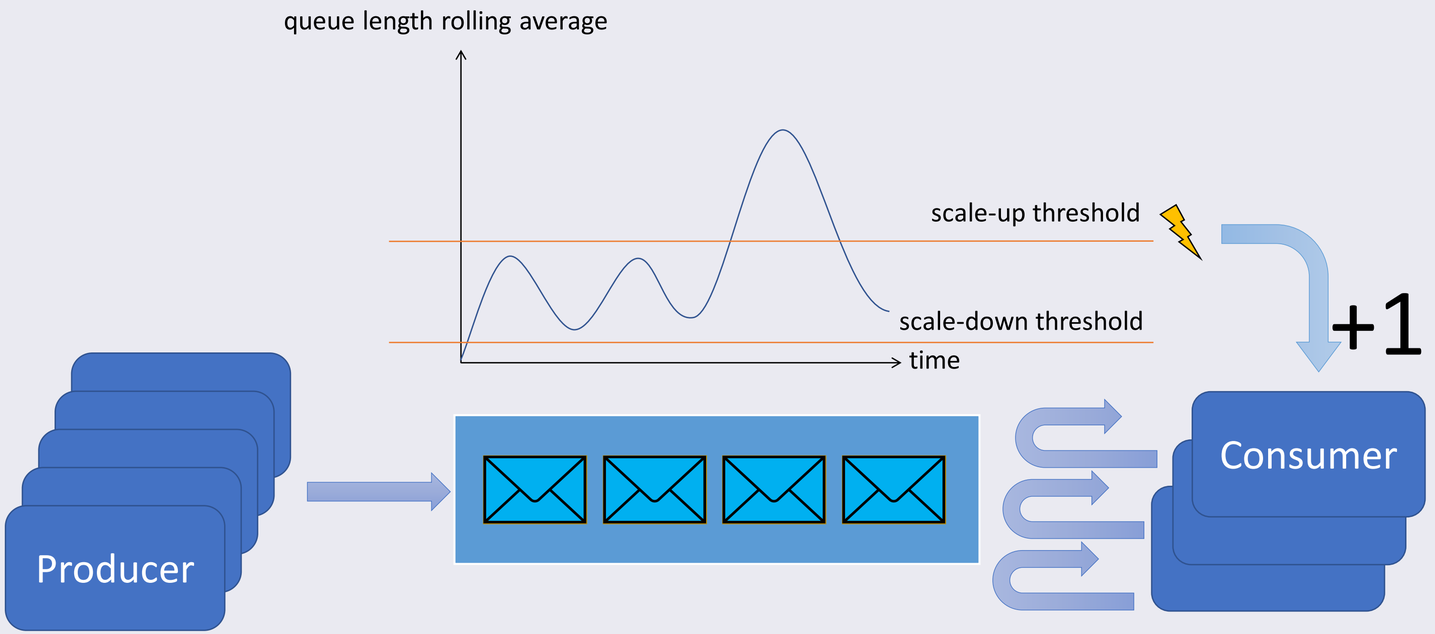






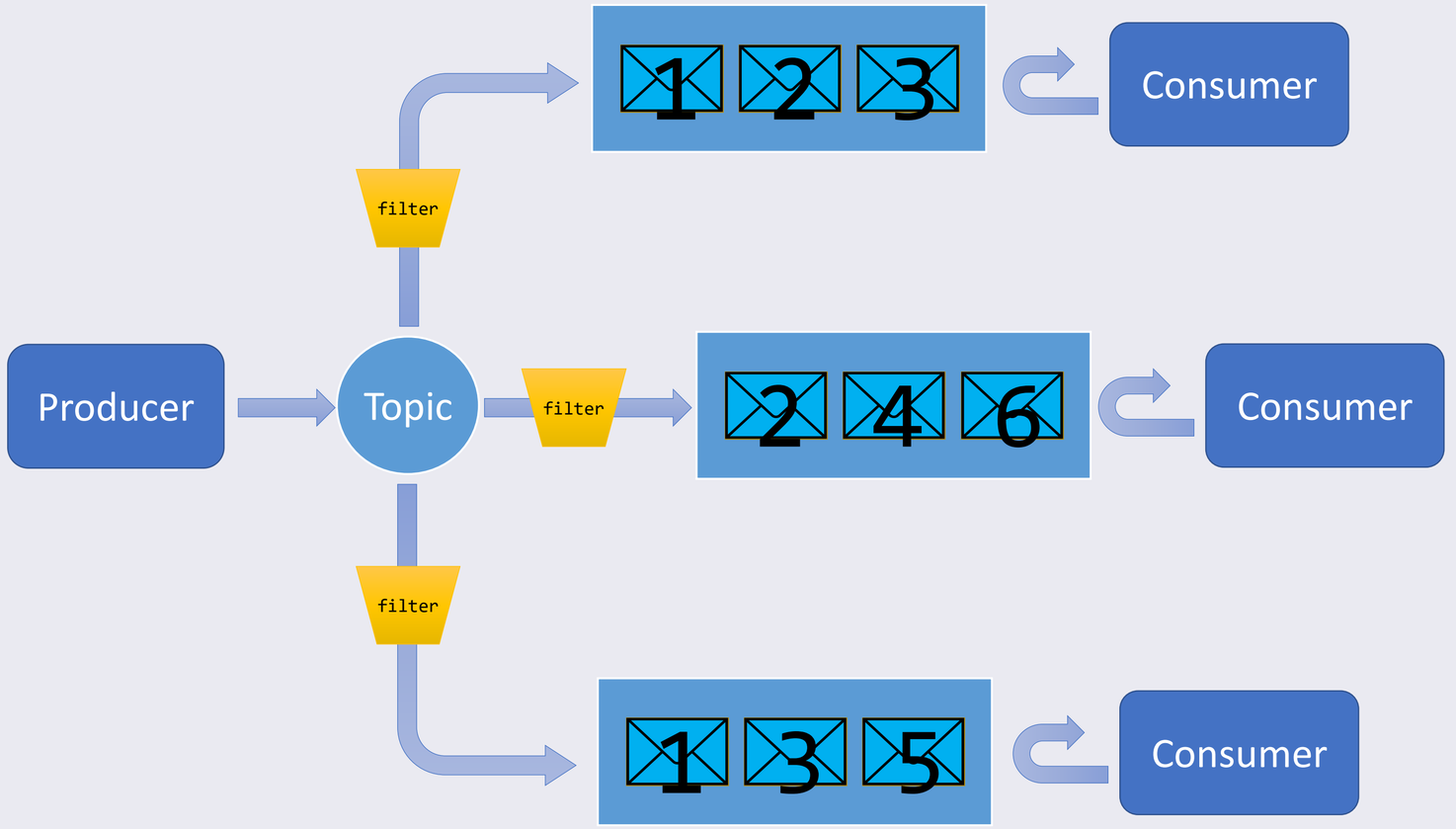
Load Balancing (and Auto Scaling)

- Multiple consumers compete for messages
- Truly load-aware job balancing
- Queue-length can be observed and more consumers can be added to to manage load



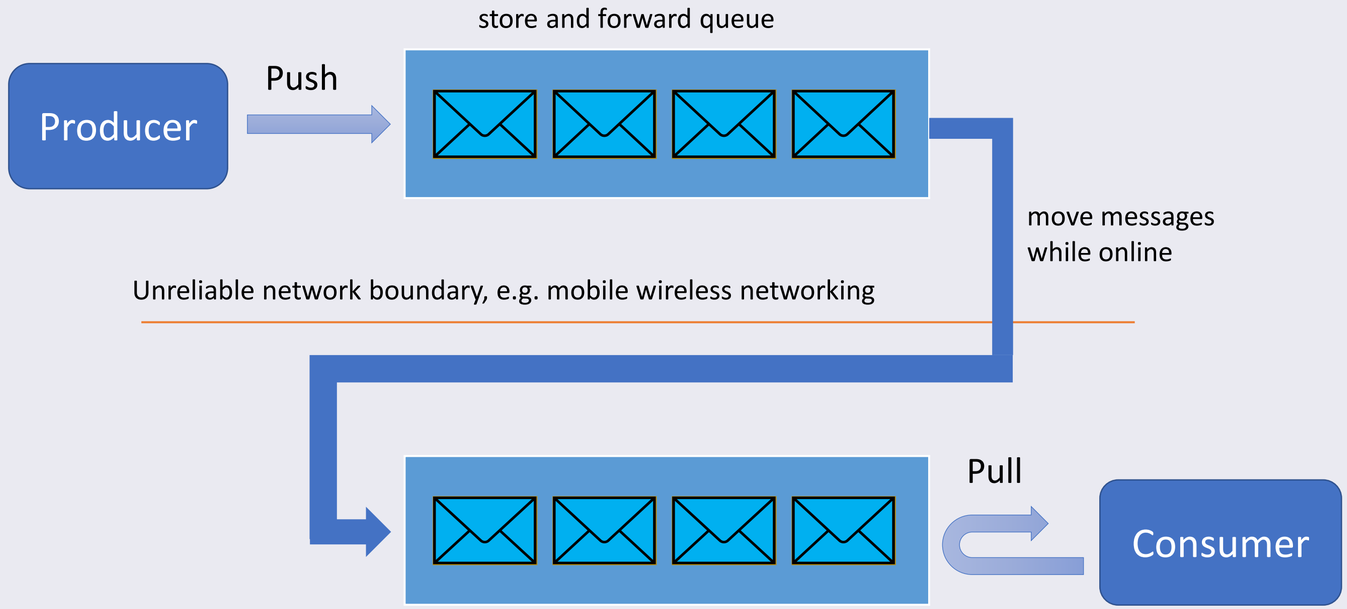
Publish-Subscribe

- Directing one input message to zero or more interested parties (subscribers).
- Every subscriber can obtain a copy of every published message.
- Subscribers may provide filters that select a subset of the published messages.



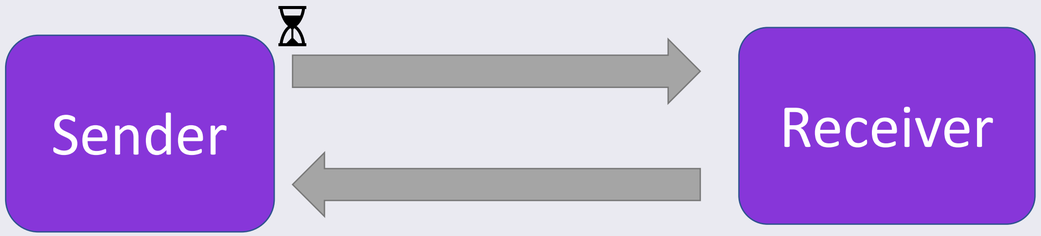
Sparse Connectivity

- Common scenario with mobiles devices and IoT applications.
- Mobile users switch networks, go out of range, hit bandwidth caps, etc.
- Using a local store/forward queue makes communication paths more robust.

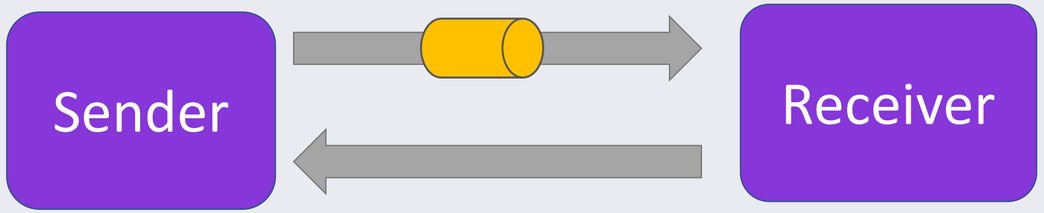


Recap

Synchronous



Asynchronous



Samples and Documentation

https://www.theurlist.com/distributed-messaging-patterns

Thanks and …
See you soon!

Thanks also to the sponsors.
Without whom this would not have been posible.

plain concepts

Microsoft

axazure

plasticscm

CEU | Universidad San Pablo