PONTIFICIA
UNIVERSIDAD
CATÓLICA
DE CHILE

IIC3103

Taller de Integración

Profesores

Arturo Tagle / Daniel Darritchon

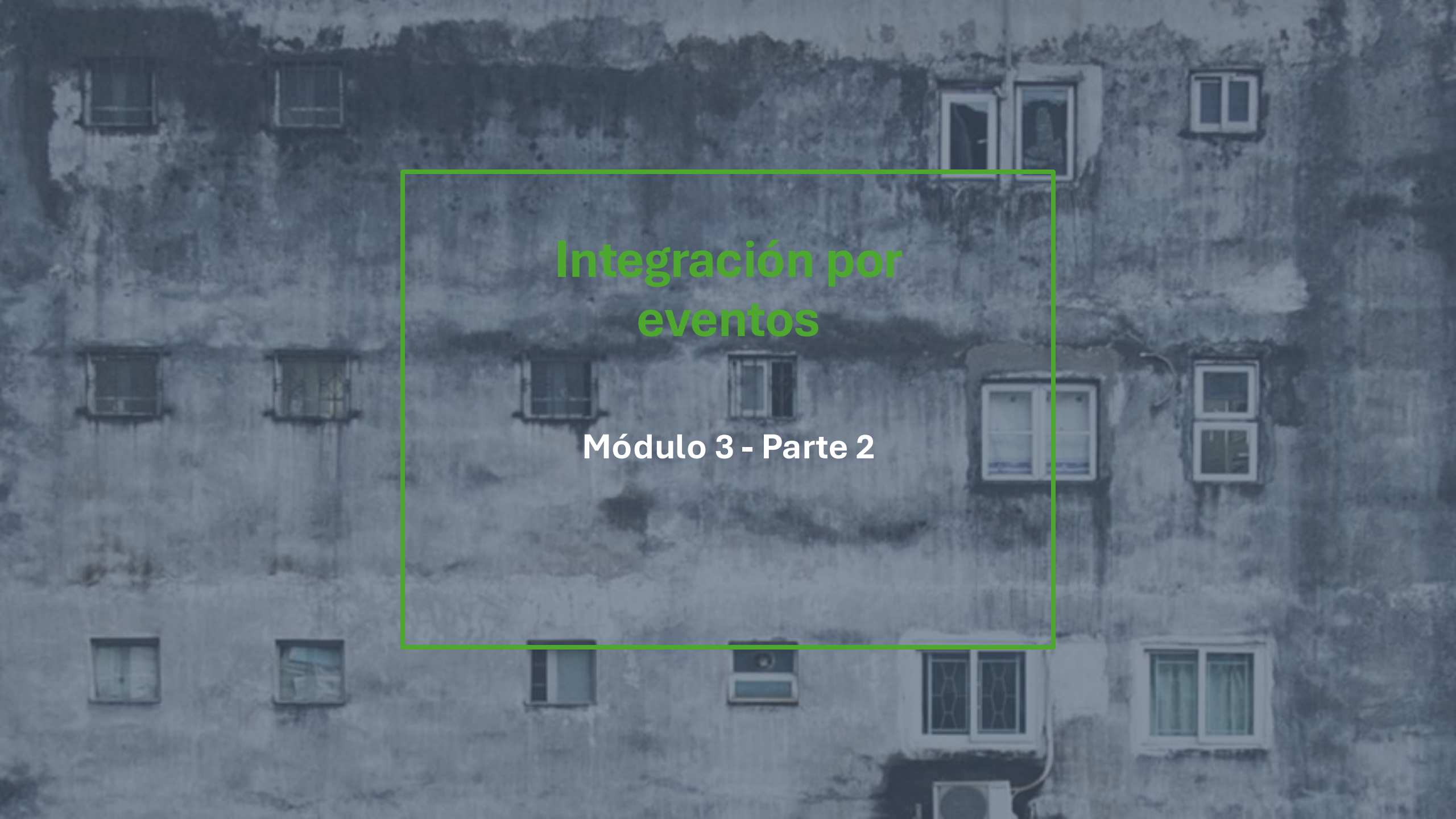
Resumen clase anterior

Características servicios y eventos

	Servicios	Eventos
Inicio de la comunicación	Cliente (pull)	Emisor del evento (push)
Participantes	Cliente y servidor	Un emisor, múltiples suscriptores
Sincronía	Comunicación síncrona	Comunicación asíncrona
Uso	Basta con conocer contrato de comunicación para realizar un request	Registro previo del suscriptor, quién notifica eventos en la medida que suceden
Tipo de datos	Estado actual de objetos	Estado en un cierto momento
Arquitectura	Cliente - servidor	Emisor - Event broker - Suscriptores

Datos en servicios y eventos

	SERVICIOS	EVENTOS
Entidad	Objeto	Hecho o cambio de estado
Esquema	Estricto, cambia poco	Flexible, distintos hechos pueden generar eventos diferentes
Qué es	Una foto del objeto en ese momento del tiempo	Un hecho o cambio de estado en el momento que se produce
Qué describe	Describe el presente	Describe una tendencia en el tiempo
Cómo se actualiza el objeto	Se actualiza, no es necesario mantener historia	Se agrega. Es necesario recorrer toda la historia para ver estado actual.
Complejidad	O(n) Lineal según cantidad de objetos	O(n*k) Según cantidad de objetos (n) y cambios notificados (k)

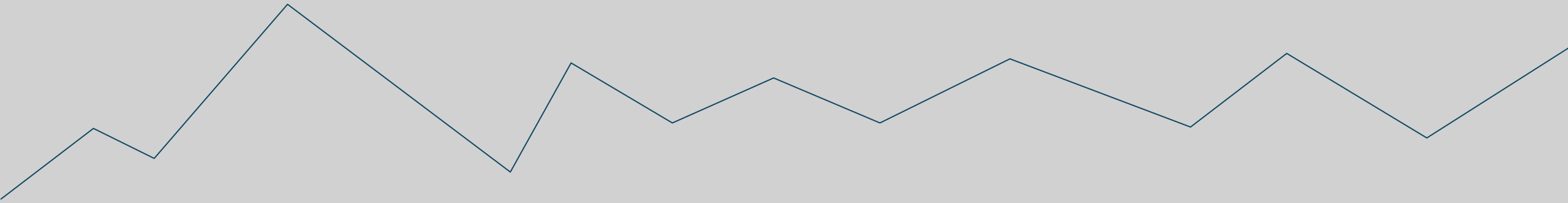


Integración por eventos

Módulo 3 - Parte 2

Concentradores de eventos

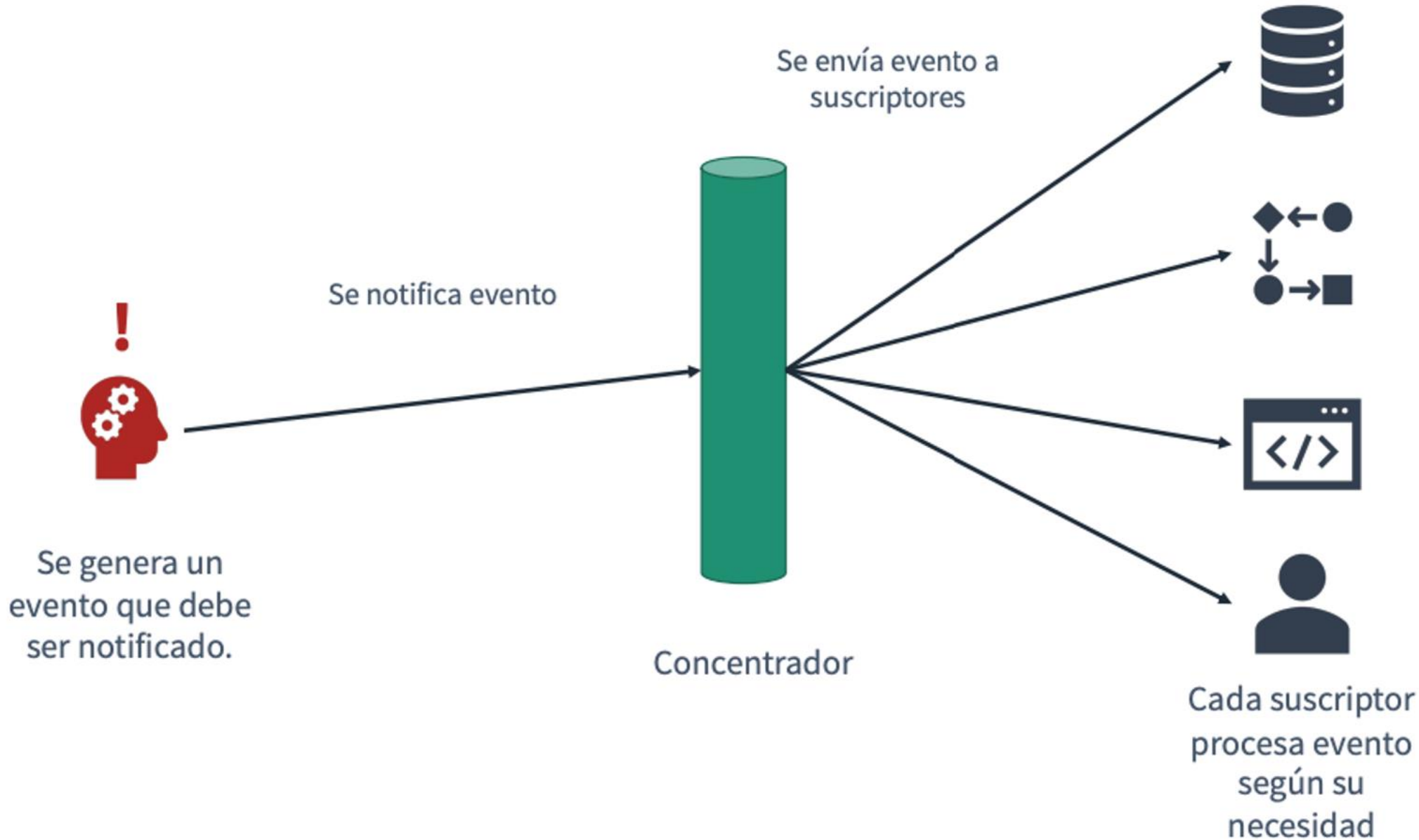
También conocidos como brokers, middlewares o bus de eventos.



CONCENTRADORES DE EVENTOS

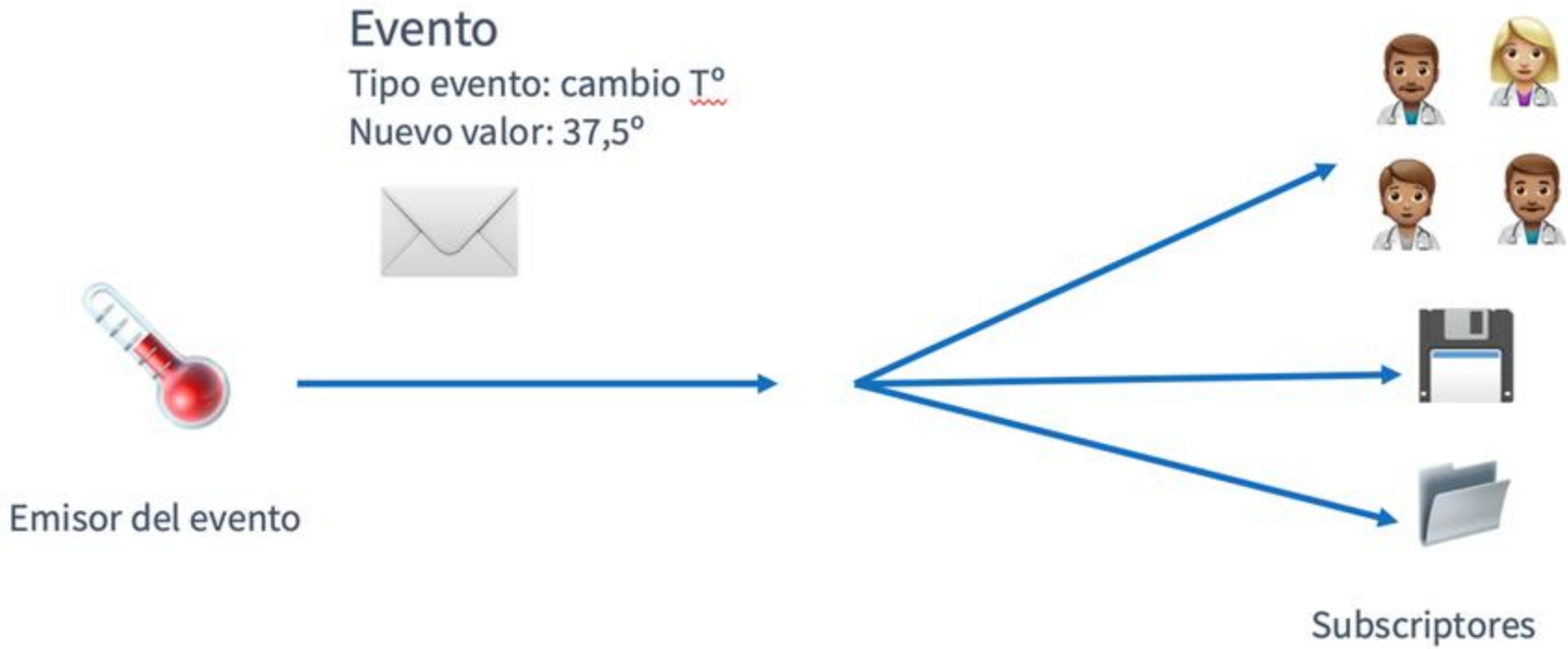
Son utilizados para facilitar, mediar y enriquecer la interacción de emisores y suscriptores en sistemas orientados a eventos.

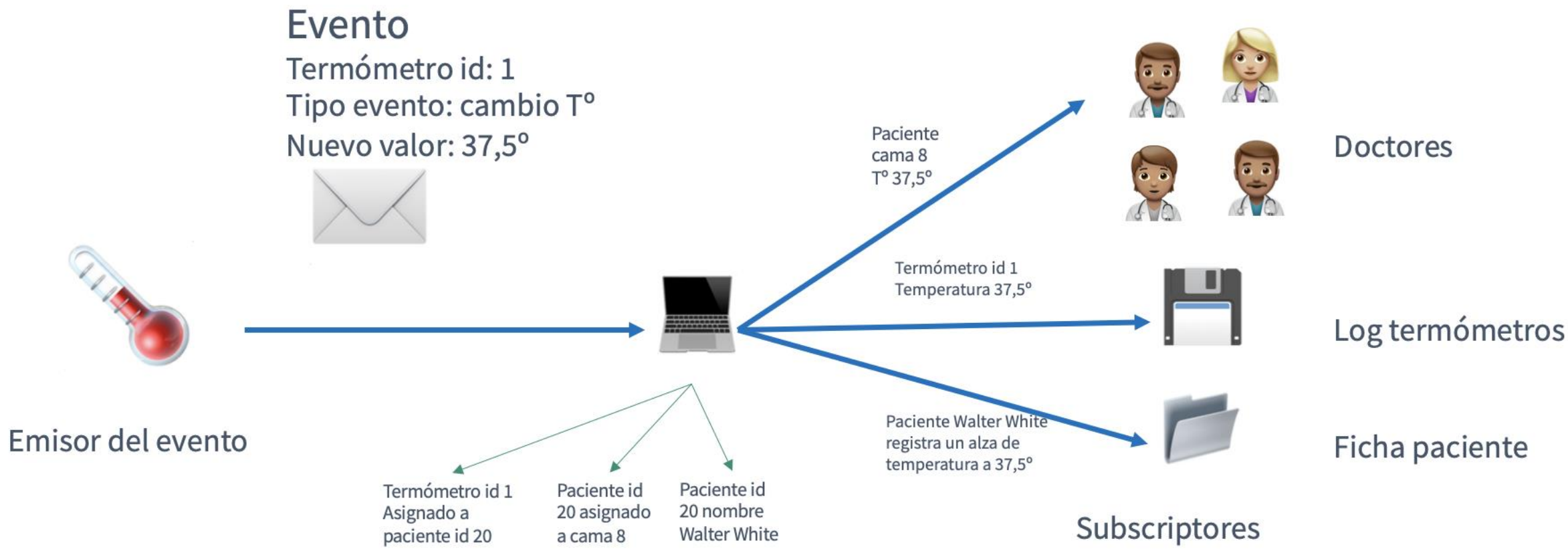
Las arquitecturas orientadas a eventos son, normalmente, arquitecturas con intermediarios. Los concentradores, son parte fundamental de estas arquitecturas.


Se envía evento a suscriptores
Se notifica evento
Se genera un evento que debe ser notificado.
Concentrador
Cada suscriptor procesa evento según su necesidad

EJEMPLOS DE MIDDLEWARES DE EVENTOS

1. Sistemas de colas (MQ, ej. RabbitMQ)
2. Sistemas Pub/Sub
3. Notificaciones Push
4. Sistemas de streaming:
    1. Apache Kafka
    2. Apache Spark Streaming


Evento
Tipo evento: cambio Tº
Nuevo valor: 37,5º
Emisor del evento
Subscriptores


Evento
Termómetro id: 1
Tipo evento: cambio T°
Nuevo valor: 37,5°
Emisor del evento
Paciente
cama 8
T° 37,5°
Doctores
Termómetro id 1
Temperatura 37,5°
Log termómetros
Paciente Walter White
registra un alza de
temperatura a 37,5°
Ficha paciente
Subscriptores
Termómetro id 1
Asignado a
paciente id 20
Paciente id
20 asignado
a cama 8
Paciente id
20 nombre
Walter White

FUNCIONALIDADES

Integridad de mensajes, asegurando entregas de tipo:

- Exactamente una entrega del evento
  - Ejemplo: Procesamiento de transacciones bancarias. No se pueden duplicar transacciones.
- Por lo menos una entrega del evento, o a lo más una entrega del evento
  - Ejemplo: conteo de estadísticas de uso, donde no contar uno (o contar un evento doble) no distorsiona la muestra.
  - Ejemplo: streaming de cambios de precio de acciones, si llega dos veces un precio, no cambia el valor.

FUNCIONALIDADES

Seguridad

- Control de acceso
- Autorización a suscripción a tópicos/canales según perfil

Logs y auditoría

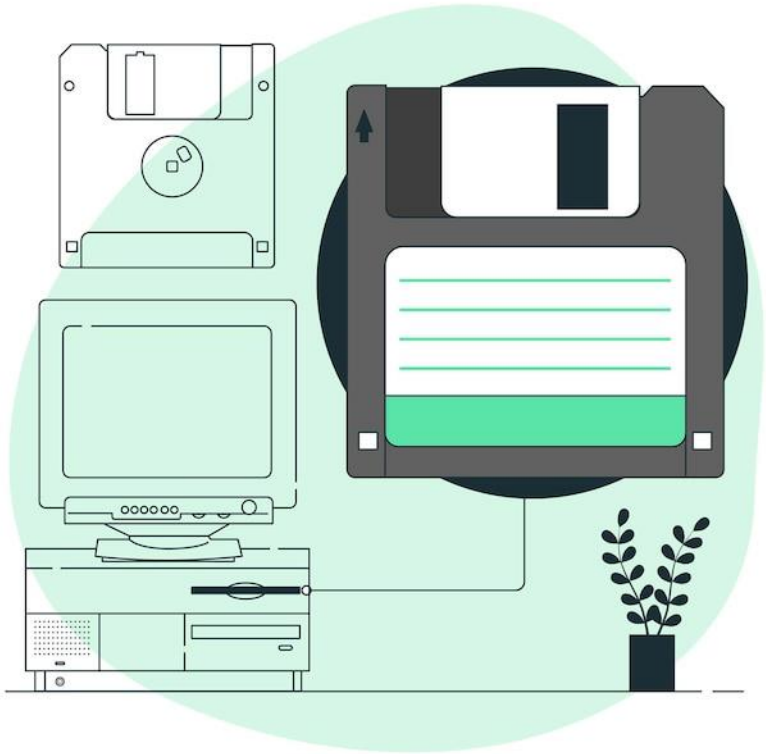
Preprocesamiento de eventos

- Filtros de datos
- Aumentar datos
- Transformación de datos

Persistencia de eventos

- Almacenamiento histórico
- Control de envejecimiento

EN RESUMEN



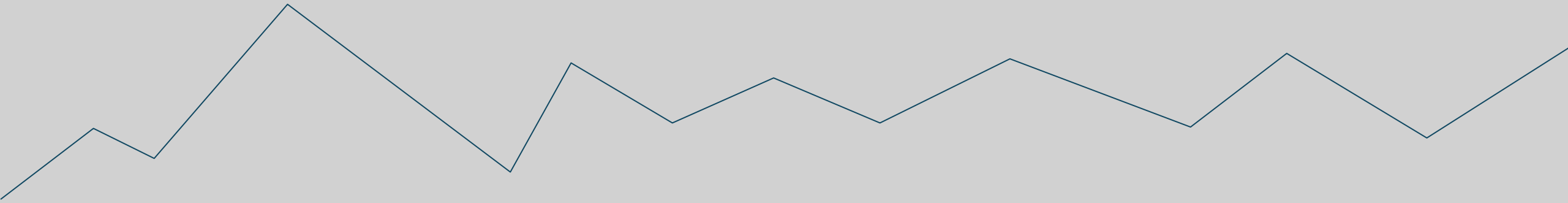
Reciben la notificación original del emisor

Propagan el evento original a los suscriptores

Manejan y mantienen el registro de suscriptores

Dos patrones de arquitectura

Topologías de procesamiento de eventos



TOPOLOGÍA

La topología es el mapa lógico de una red para intercambiar datos, en este caso, eventos. Está formada por nodos interconectados.

Estas topologías nos permiten generar patrones de arquitecturas de sistemas distribuidos y asíncronos altamente escalables.

COMPONENTES

Emisor del evento

Es quién genera el cambio de estado y notifica un evento.

Procesador de eventos (suscriptor)

El suscriptor o procesador del evento es quién tiene la lógica de negocio para procesar el mensaje. Cada procesador es autocontenido e independiente del resto.

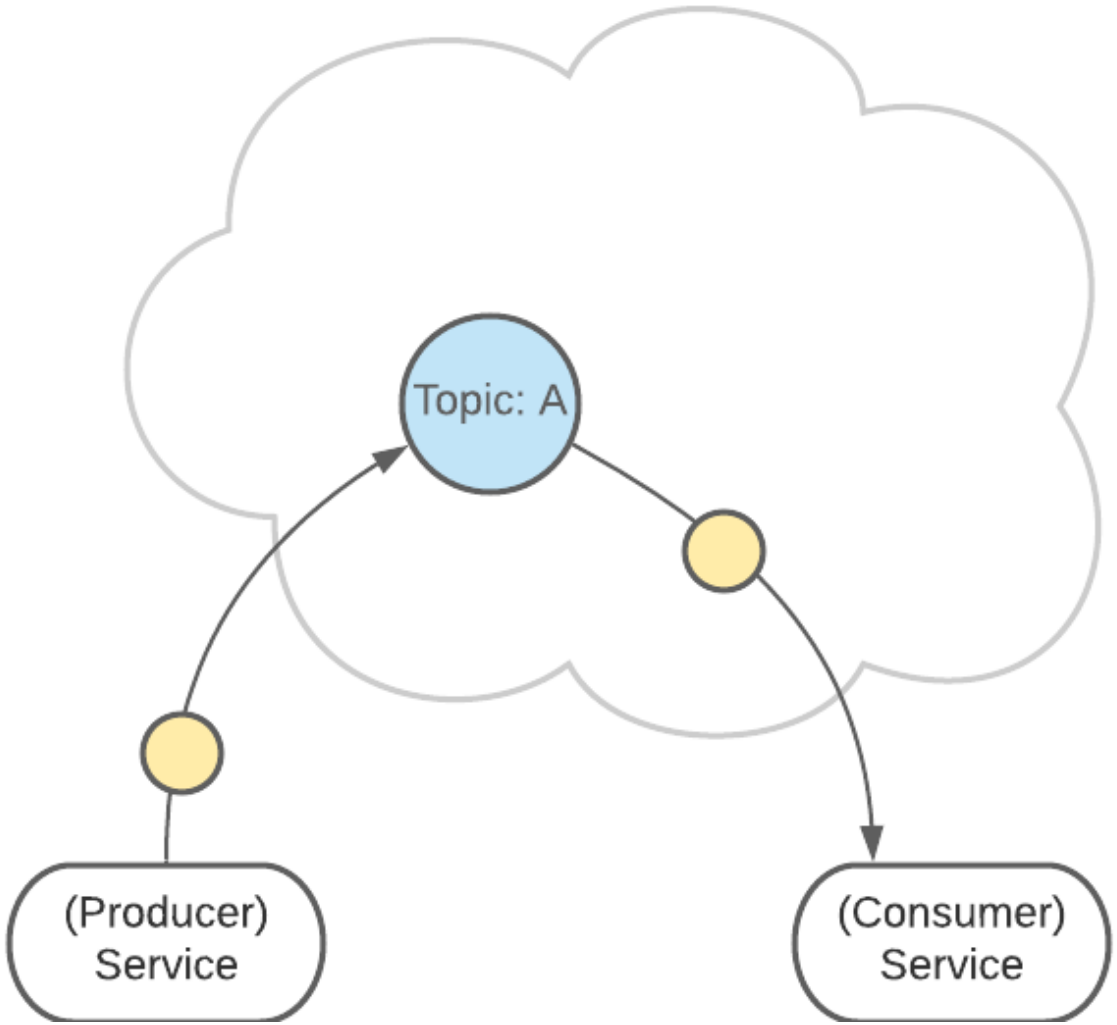
Canal de eventos, concentrador o bus de eventos (middleware)

Un canal de eventos es quién traslada el evento original al procesador de eventos o suscriptor.

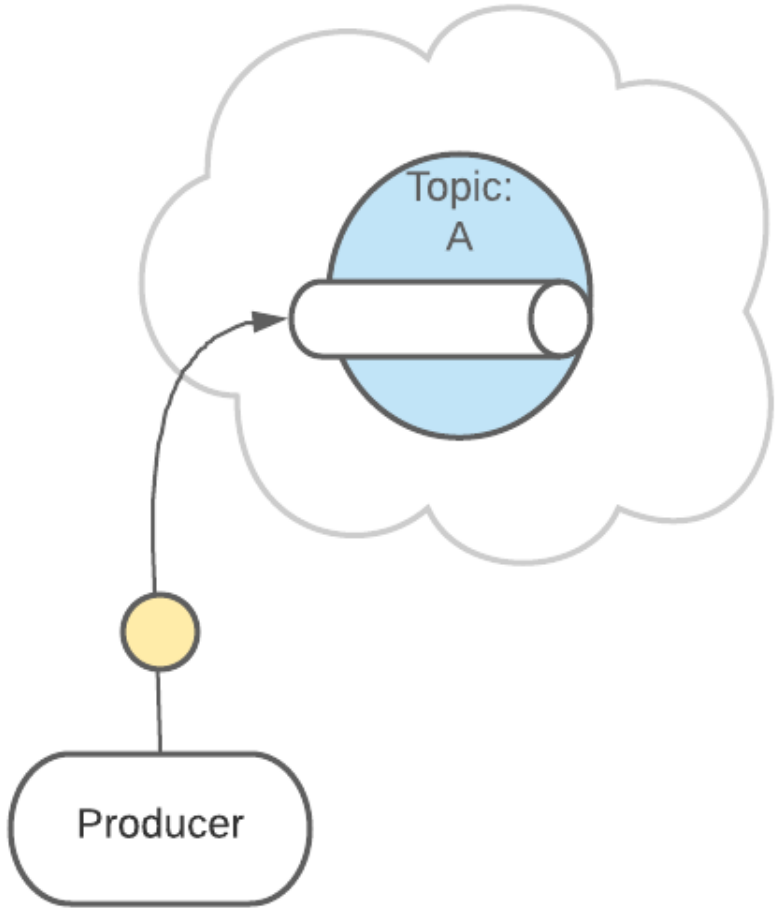
Tener distintos canales en una misma topología nos permite diferenciar tipos de clientes, formas de procesamiento, tipos de eventos, etc.

El canal puede ser liviano, sin lógica de negocio (patrón broker), o pesado, donde cierta lógica de negocio es trasladada al canal, como la decisión de quién debe procesar qué tipo de eventos (patrón mediador).

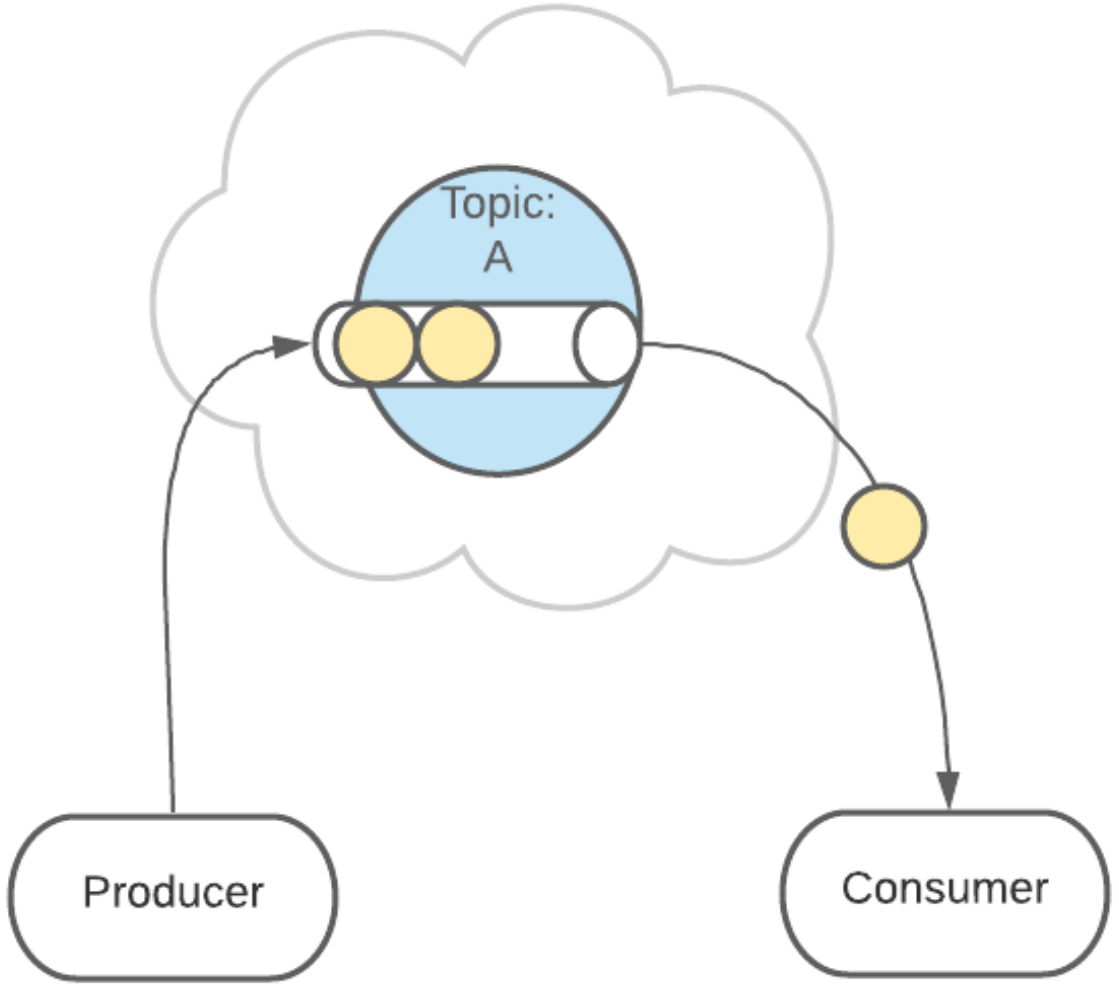
El emisor del evento publica en un canal (también llamado tópico en algunos contextos)



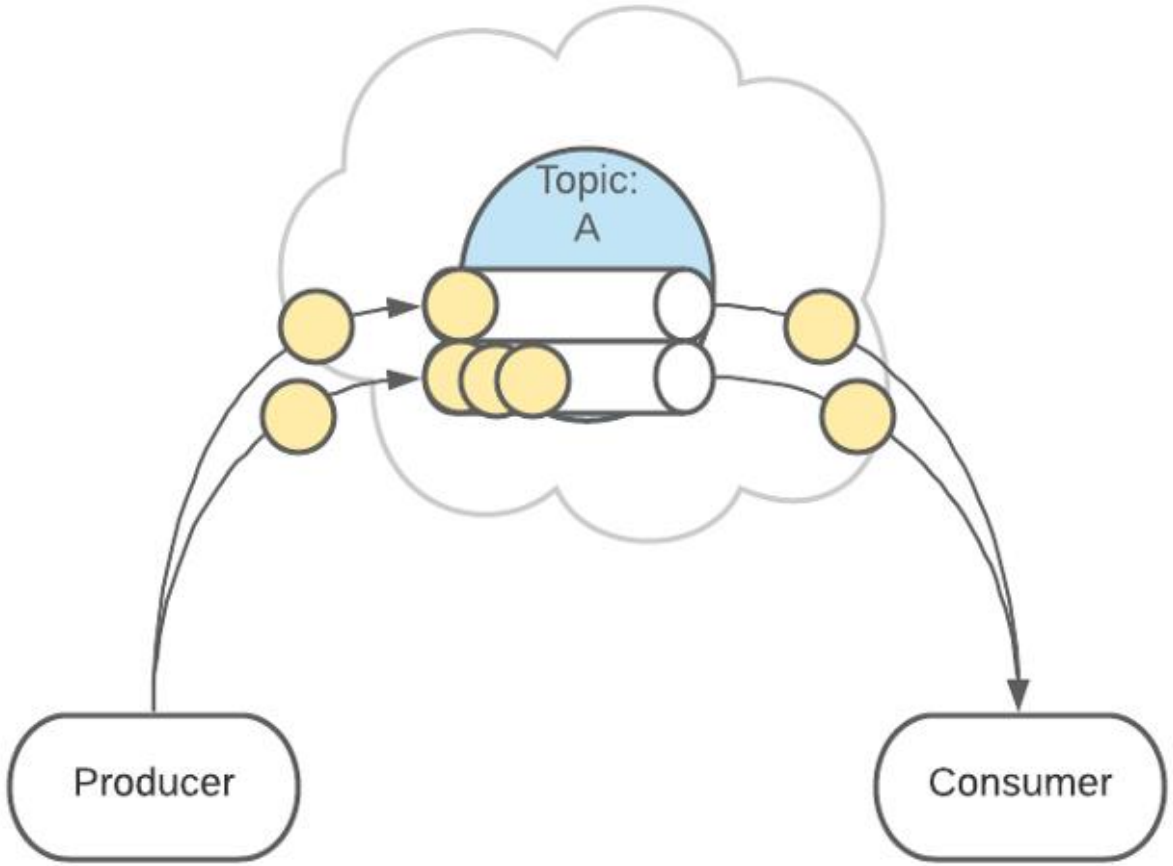
Un tópico normalmente tiene una cola de mensajes, donde se guarda el evento hasta que es consumido por el suscriptor



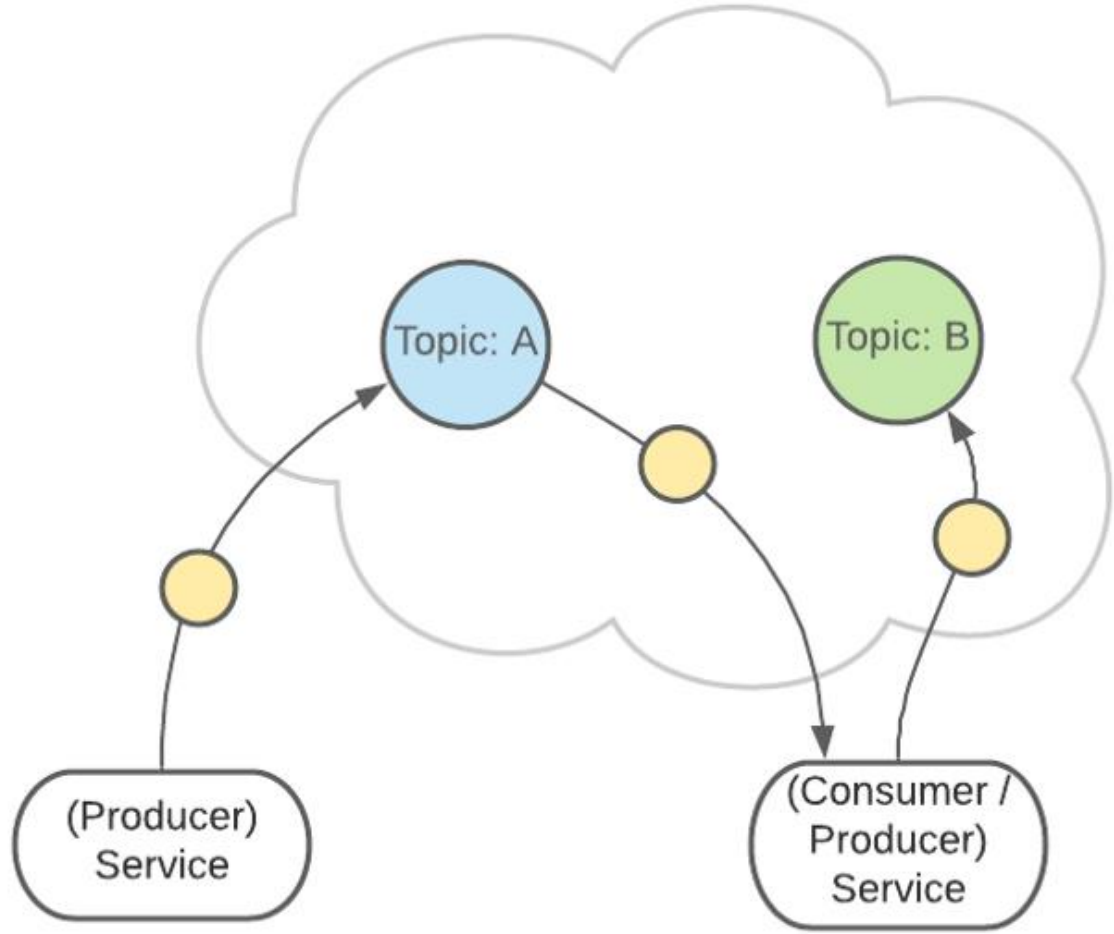
La cola permite almacenar temporalmente mensajes mientras el suscriptor procesa otros mensajes.

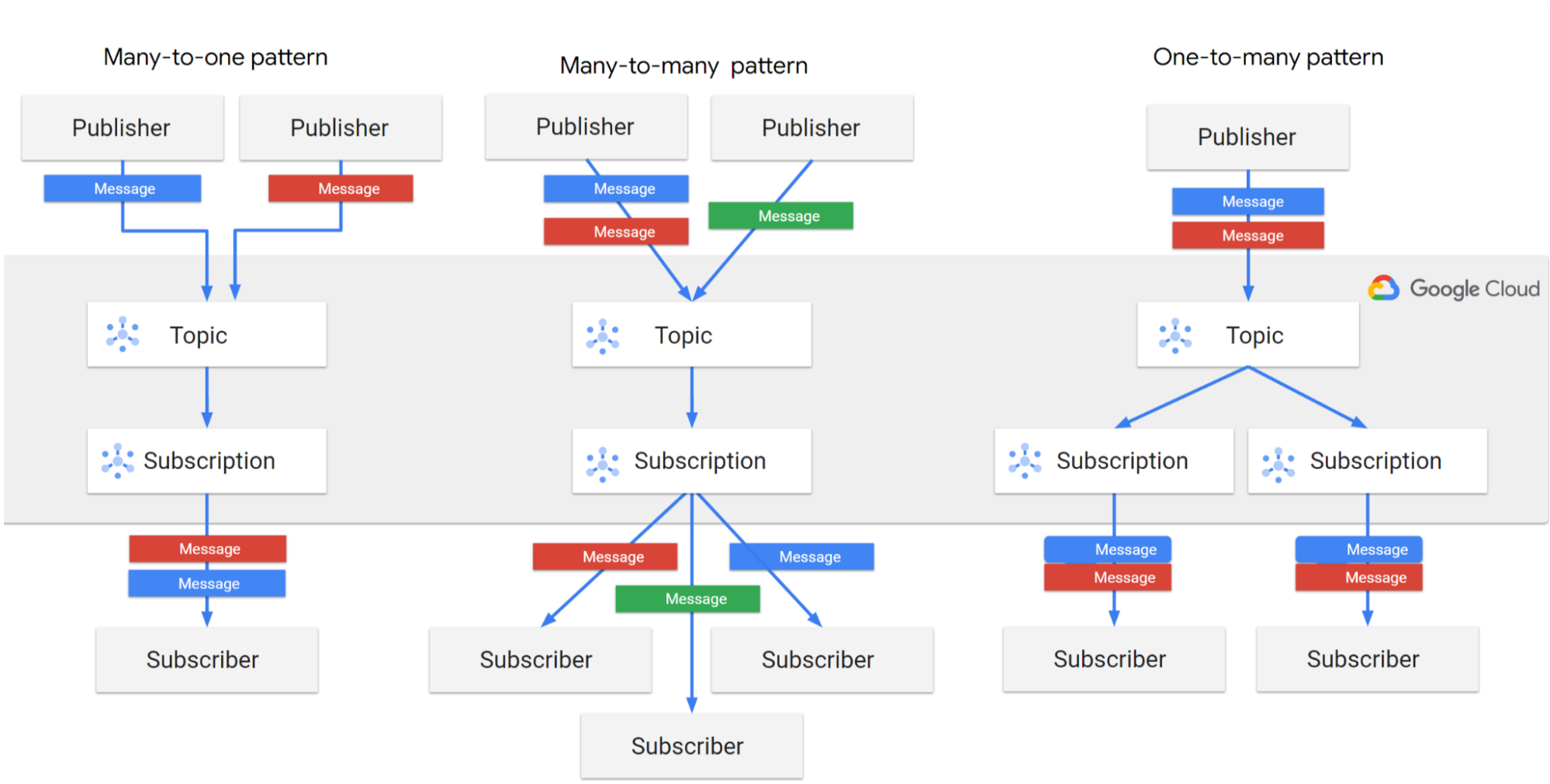


Algunos sistemas manejan múltiples colas por tópico, por ejemplo, para mensajes prioritarios.



Un suscriptor también puede ser emisor de eventos



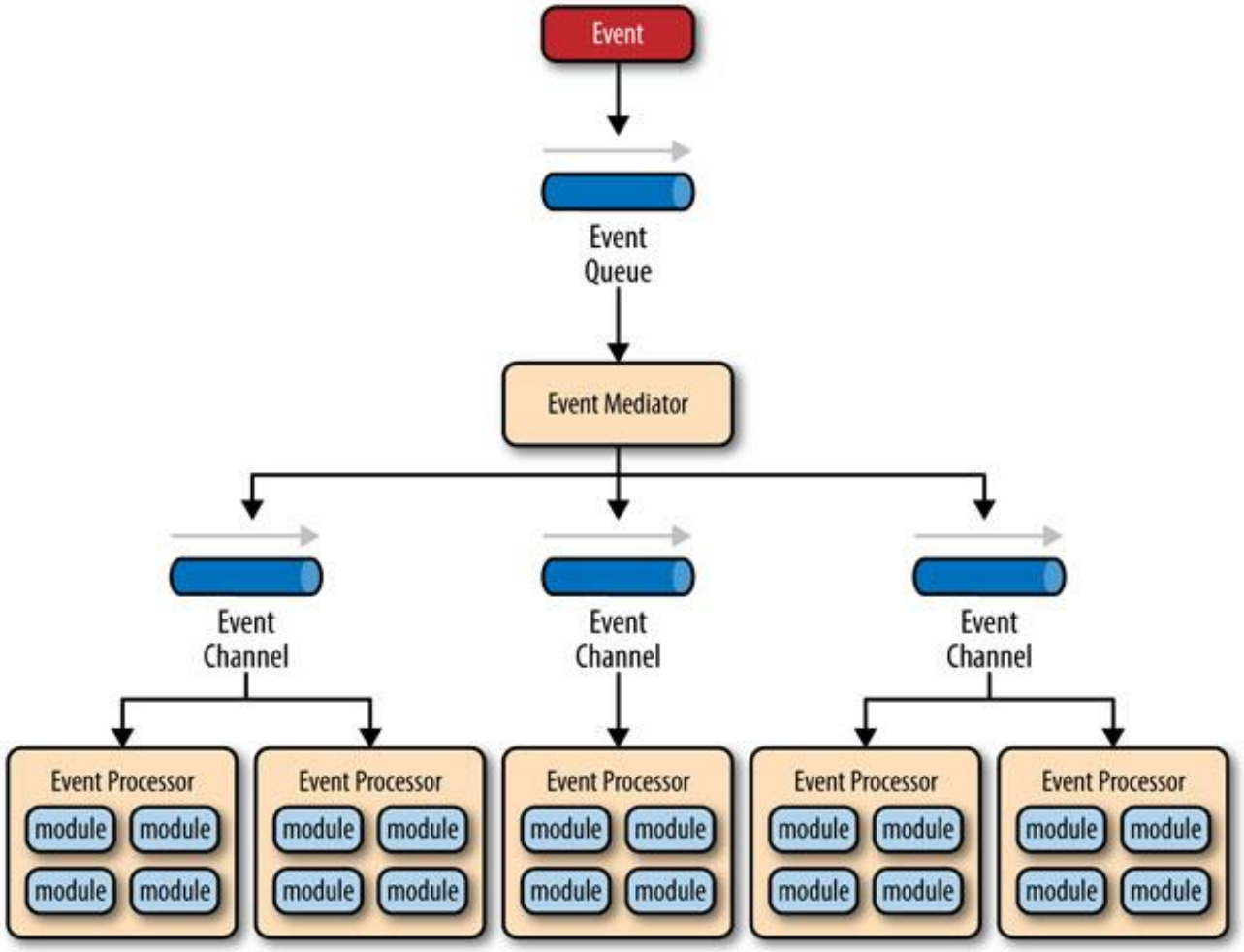

Many-to-one pattern
Publisher
Publisher
Message
Message
Topic
Subscription
Message
Message
Subscriber
Many-to-many pattern
Publisher
Publisher
Message
Message
Message
Topic
Subscription
Message
Message
Message
Subscriber
Subscriber
Subscriber
One-to-many pattern
Publisher
Message
Message
Topic
Subscription
Subscription
Message
Message
Message
Message
Subscriber
Subscriber
Google Cloud

Topología de mediador

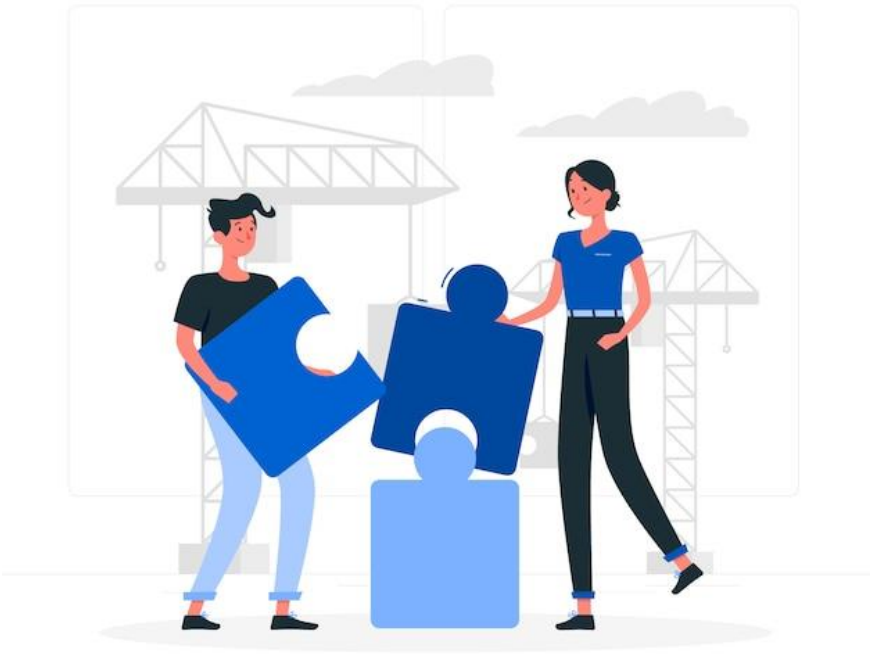
Se recibe el evento inicial, el cual es repartido por distintos canales según cierta lógica de negocio.

Esta topología es útil cuando se debe seguir un workflow o secuencia específica de eventos.

Alta trazabilidad del evento, ocurren “transacciones” que debe ser esperadas para seguir el flujo.



FLUJO DE EVENTOS



1. Un emisor emite un evento a una cola.
2. La cola lleva el mensaje al mediador.
3. El mediador orquesta la interacción asíncronamente con los canales para ejecutar cada paso.
4. El procesador recibe el evento y ejecuta la lógica de negocios específica.

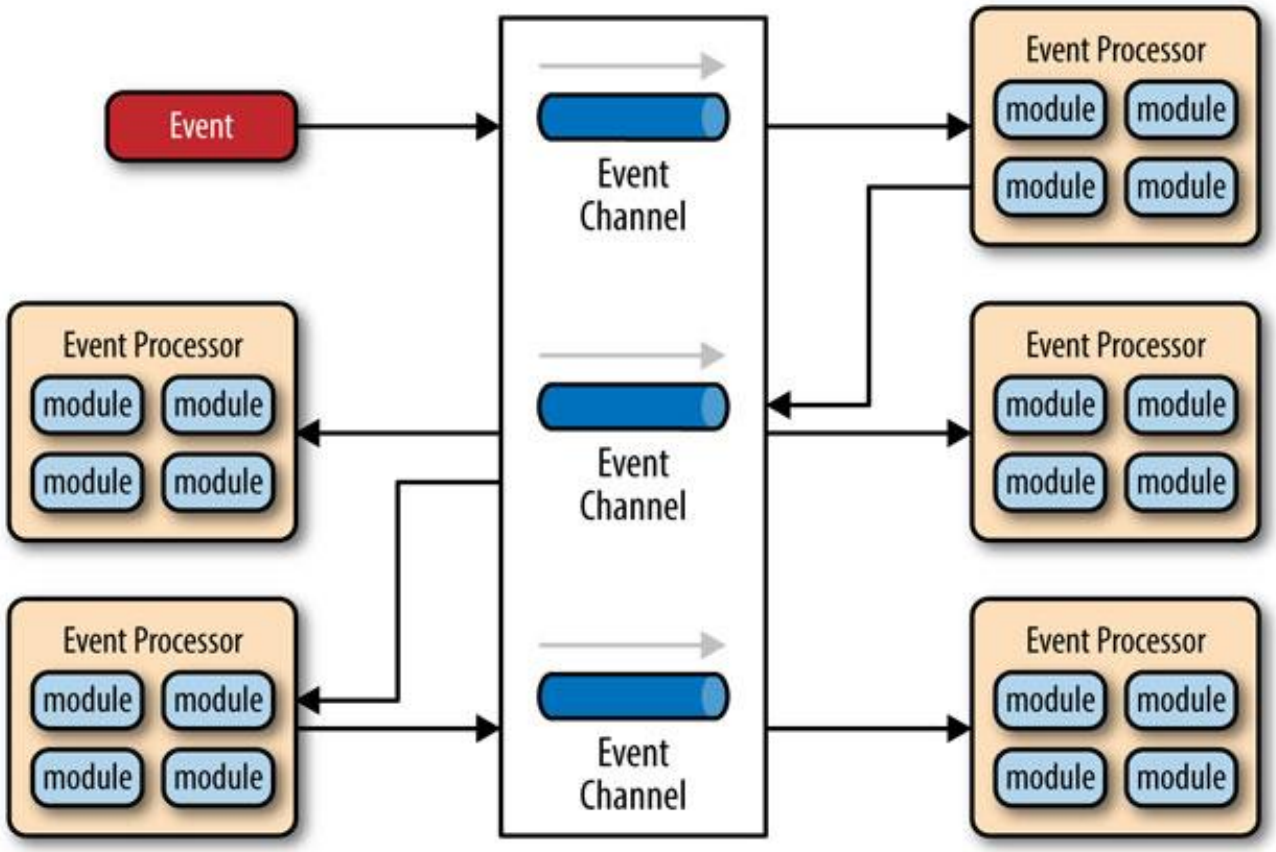
Topología de broker

En esta topología, no hay un mediador central: el evento es distribuido a lo largo de distintos procesadores.

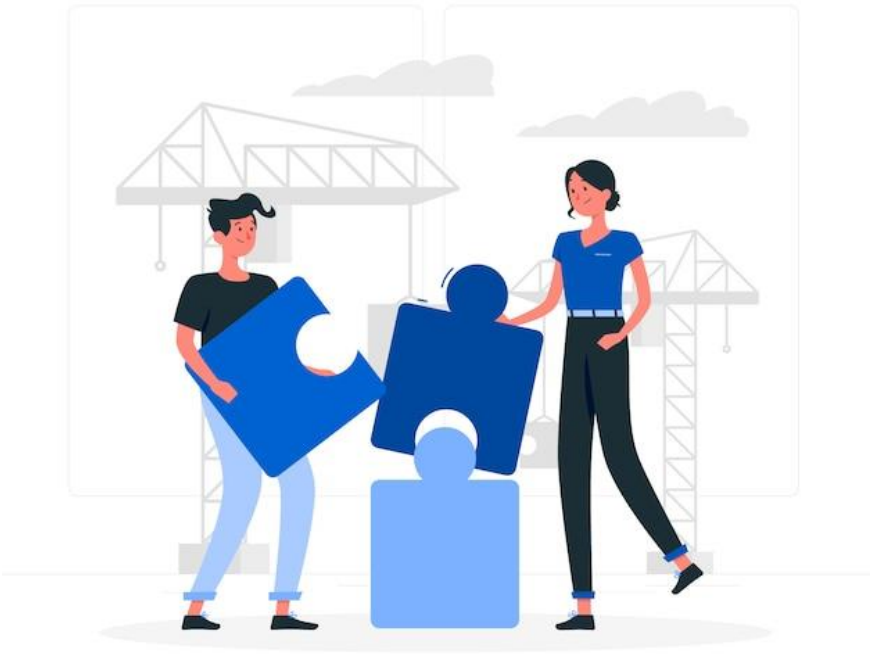
El performance mejora porque varias tareas pueden ser procesadas en paralelo.

La tolerancia de fallos es alta porque si tenemos un servicio procesador abajo, eso no afectará a los demás.

La trazabilidad es más compleja.



FLUJO DE EVENTOS



1. Se recibe el evento y se redirige a los suscriptores.
2. Cada suscriptor procesa el evento.
3. En caso de ser necesario, un suscriptor puede emitir un nuevo evento para gatillar otro flujo.

PROCESAMIENTO DE EVENTOS

PUSH

- El middleware determina cuando se debe enviar un evento al suscriptor.
- El suscriptor no tiene que tener ninguna información del middleware porque recibe todos los datos.
- La forma más común es que el evento se envía como un request a través de un webhook.

PULL

- El procesador determina cuándo ir a buscar un evento.
- El suscriptor debe mantener la información para ir a consultar la cola o middleware que tiene los eventos.

DIFICULTADES EN EL PROCESAMIENTO DE EVENTOS

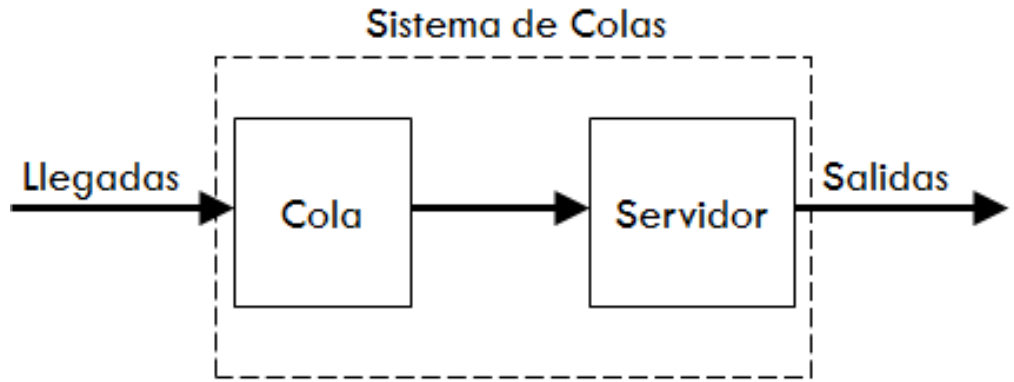
Transaccionabilidad

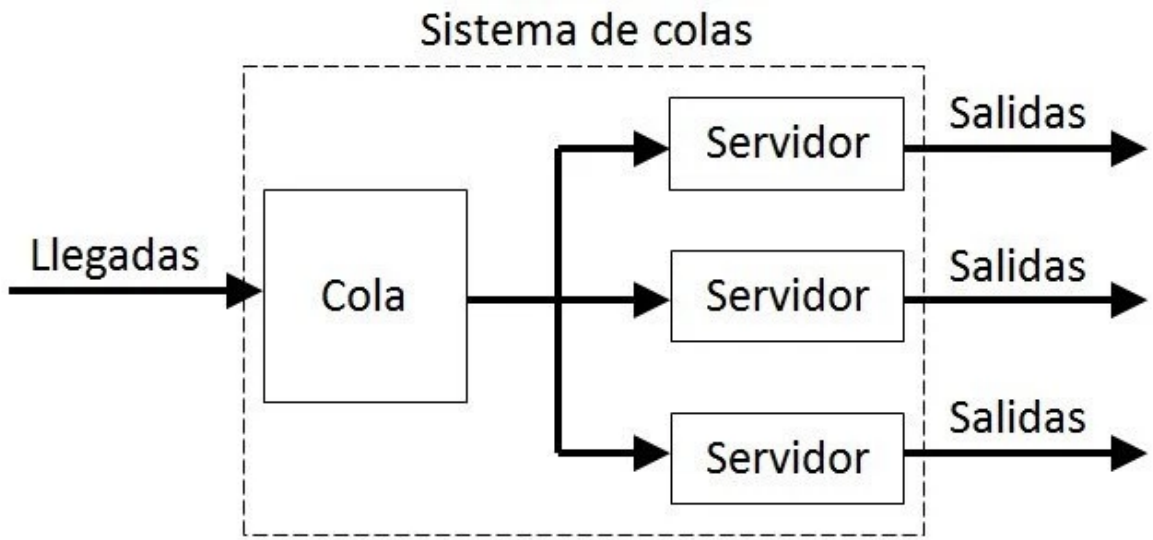
Es muy difícil generar transacciones atómicas en estas arquitecturas, ya que el procesamiento es, por definición, descentralizado y asíncrono.

Consistencia eventual

El procesamiento asíncrono hace más complejo saber cuándo tendré cierto dato actualizado.

¿Se puede modelar como sistemas de colas?


Sistema de Colas
Llegadas
Cola
Servidor
Salidas


Sistema de colas
Llegadas
Cola
Servidor
Servidor
Servidor
Salidas
Salidas
Salidas

ECUACIÓN DE COLAS

$$L_q = \lambda W_q$$

Largo medio de la cola = tasa de entrada por tiempo medio de procesamiento

EJEMPLO

¿Cuál será el largo medio de la cola si los eventos llegan a una tasa de 5 eventos/seg y el procesamiento de cada evento demora 7 segs?

El largo medio de la cola es 35 eventos (5 eventos/segundo * 7 segundos)

PONTIFICIA
UNIVERSIDAD
CATÓLICA
DE CHILE

IIC3103

Taller de Integración

Profesores

Arturo Tagle / Daniel Darritchon

BIBLIOGRAFÍA

Topologías de procesamiento de eventos

Roumeliotis, R. (2015, February 19). Variations in event-driven architecture. Retrieved April 26, 2021, from http://radar.oreilly.com/2015/02/variations-in-event-driven-architecture.html

Stepro, T. (2021, February 25). Visualizing Kafka. Medium. Retrieved September 27, 2021, from https://timothystepro.medium.com/visualizing-kafka-20bc384803e7

Wickramarachchi, A. (2017, September 16). Event driven architecture pattern. Retrieved April 26, 2021, from https://towardsdatascience.com/event-driven-architecture-pattern-b54fc50276cd

Damian, S. (n.d.). What is event streaming? A deep dive. Retrieved April 26, 2021, from https://ably.com/topic/event-streaming

Laotshi. (2020, June 15). Event-drive architecture pattern. Medium. https://laotshi.medium.com/event-drive-architecture-pattern-9bb65f4be9ad

Ilustraciones

Diseñadas por freepik.es

BIBLIOGRAFÍA

Principios de diseño de eventos

Wetzler, M. (2020, July 1). Analytics for hackers: How to think about event data - keen - event streaming platform. Keen.io blog. Retrieved September 27, 2021, from https://keen.io/blog/analytics-for-hackers-how-to-think-about-event-data/

Higginbotham, J. (2018, May 2). 5 principles for designing evolvable event streams. Medium. Retrieved September 27, 2021, from https://medium.com/capital-one-tech/5-principles-for-designing-evolvable-event-streams-f32e90dcbb79

Webhooks

Gavrailova, G. (2019, May 8). ¿Qué es un webhook y por qué lo necesitas? - email marketing software. Mailjet. Retrieved September 27, 2021, from https://es.mailjet.com/blog/news/que-es-webhook/

MÁS BIBLIOGRAFÍA

Event-driven architecture

- Event-drive architecture, https://reactiveprogramming.io/blog/es/estilos-arquitectonicos/eda
- Understanding Event-Driven Architectures (EDA): the paradigm of the future, https://medium.com/drill/understanding-event-driven-architectures-eda-the-paradigm-of-the-future-7ae632f056bb
- Event-Driven Architecture | EDA | Software Architectural Patterns, https://www.youtube.com/watch?v=gIL8rW_eyww

Sistemas de streaming

- Apache Kafka Vs Apache Spark: Know the Differences, https://www.knowledgehut.com/blog/big-data/kafka-vs-spark