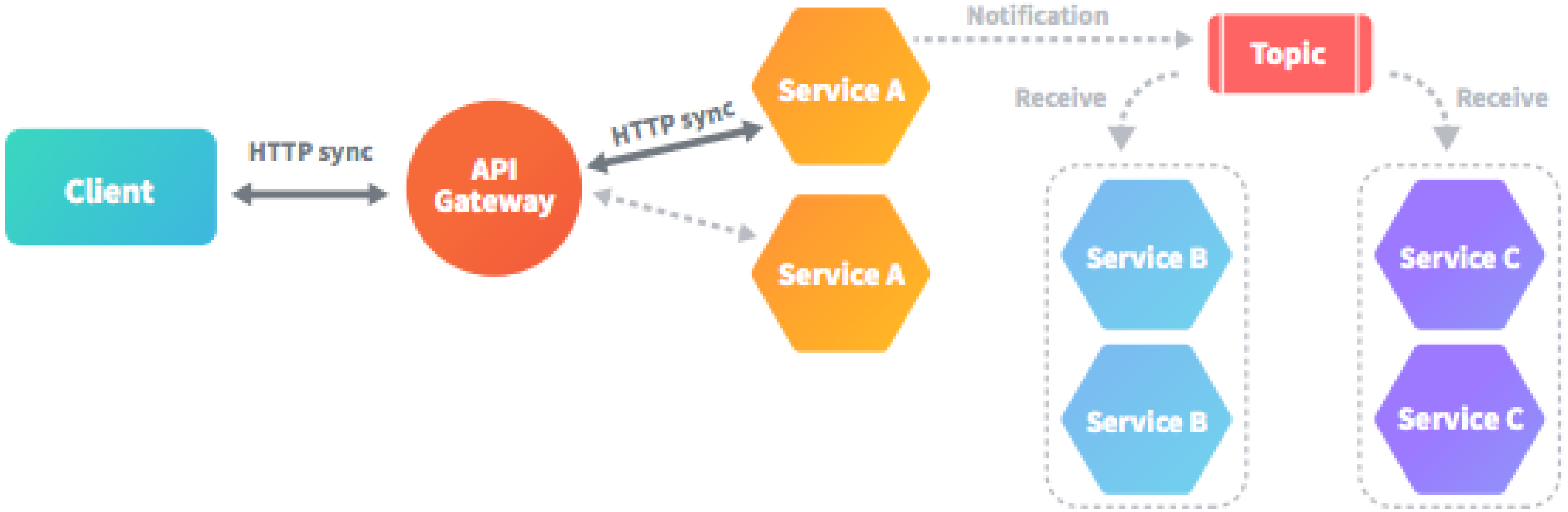
Apache Kafka

Bùi Minh Hoàng Vũ
Trần La Nhật Phương

Microservices

- Applications are built as loosely coupled services
- The services are heterogeneous in nature
- To work together, they need a way to communicate
    - Synchronous One-To-One
    - Asynchronous
        - Callback
        - Polling
        - Message Queue


Client
HTTP sync
API Gateway
HTTP sync
Service A
Service A
Notification
Topic
Receive
Receive
Service B
Service B
Service C
Service C

Apache Kafka

- A distributed streaming platform
- Data within Kafka is stored:
  - Durably
  - In order
  - Can be read deterministically
- Can be distributed between servers

Publish-Subscribe Model

- The sender does not direct messages to a receiver
- The message is classified instead
- The receiver subscribes to classes of messages

➔ Decoupled Communication

➔ Dynamic Scaling

➔ Fault Tolerance

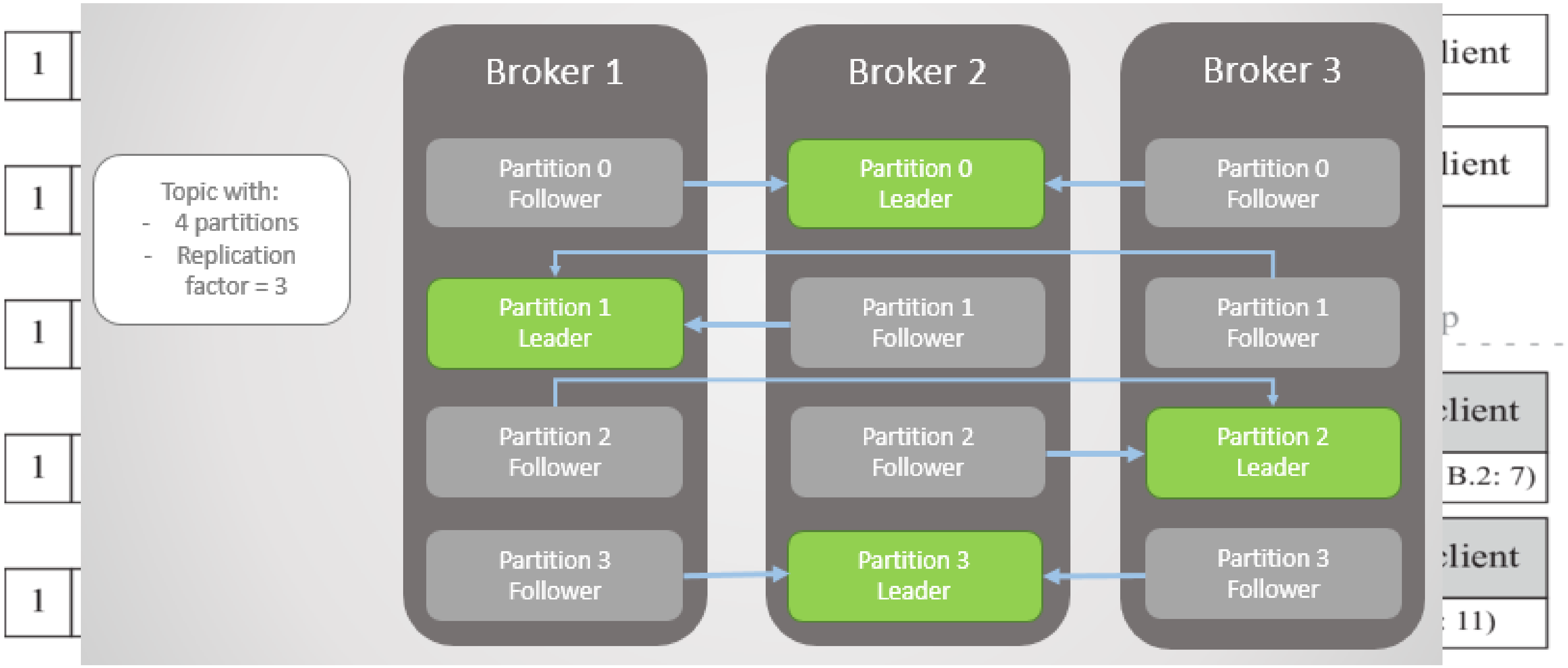
Core Concepts

- Message: A unit of data. Can optionally have a key.
- Schema: JSON, XML, Avro,…
- Topics: Organize events into categories
- Partitions: Only order within a partition is guaranteed.
- Brokers: Store and manage events.
- Producers and Consumers

Data Flow

- A producer publish a message to a specific topic
- Kafka Brokers store and replicate the message
  - Messages are assigned to partitions using the key
  - Messages are appended to the end of partitions
- Consumers read messages from specific partitions
- Messages are not deleted upon read

Partitions & Replication

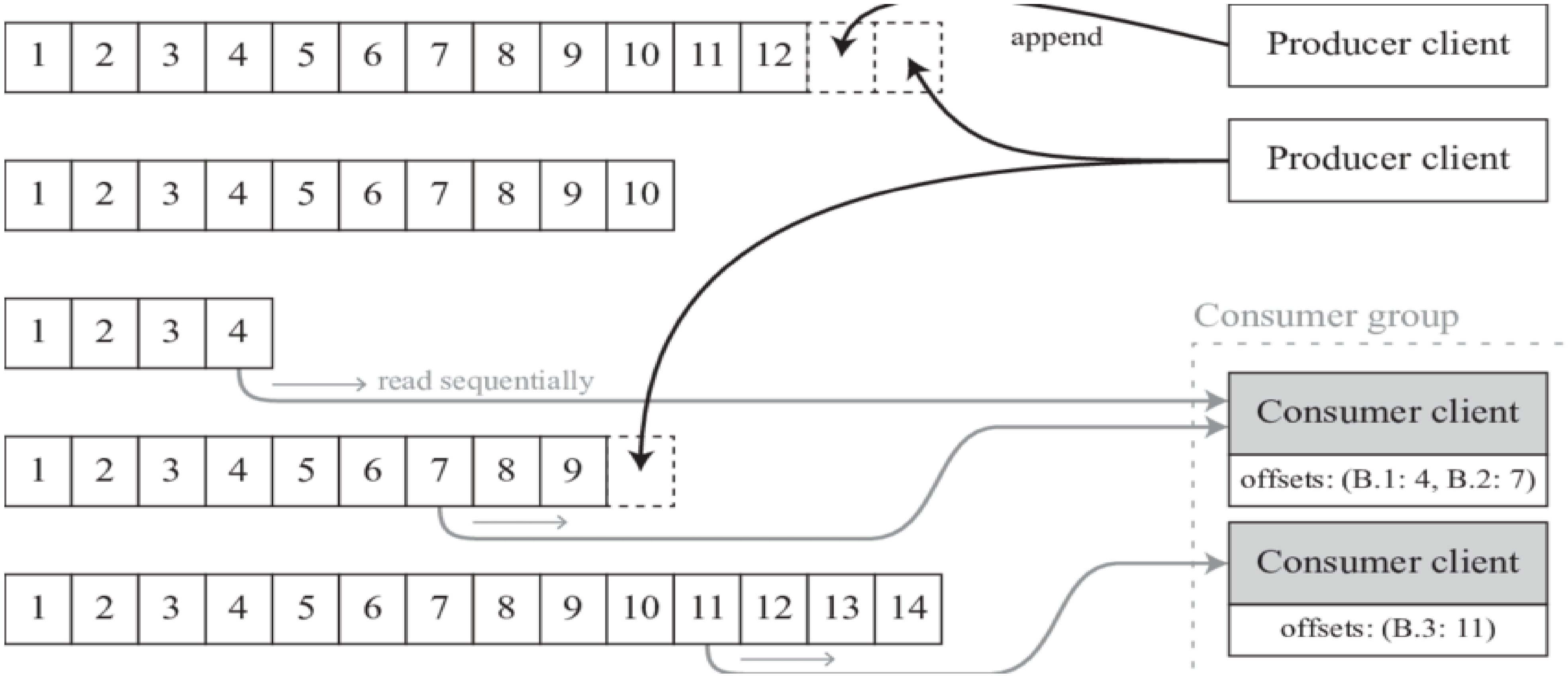


Producers

- Create & send messages to Topics
- Can use keys to influence partitioning
- Message acknowledgment:
    - No Ack
    - Leader Ack
    - All Replicas Ack

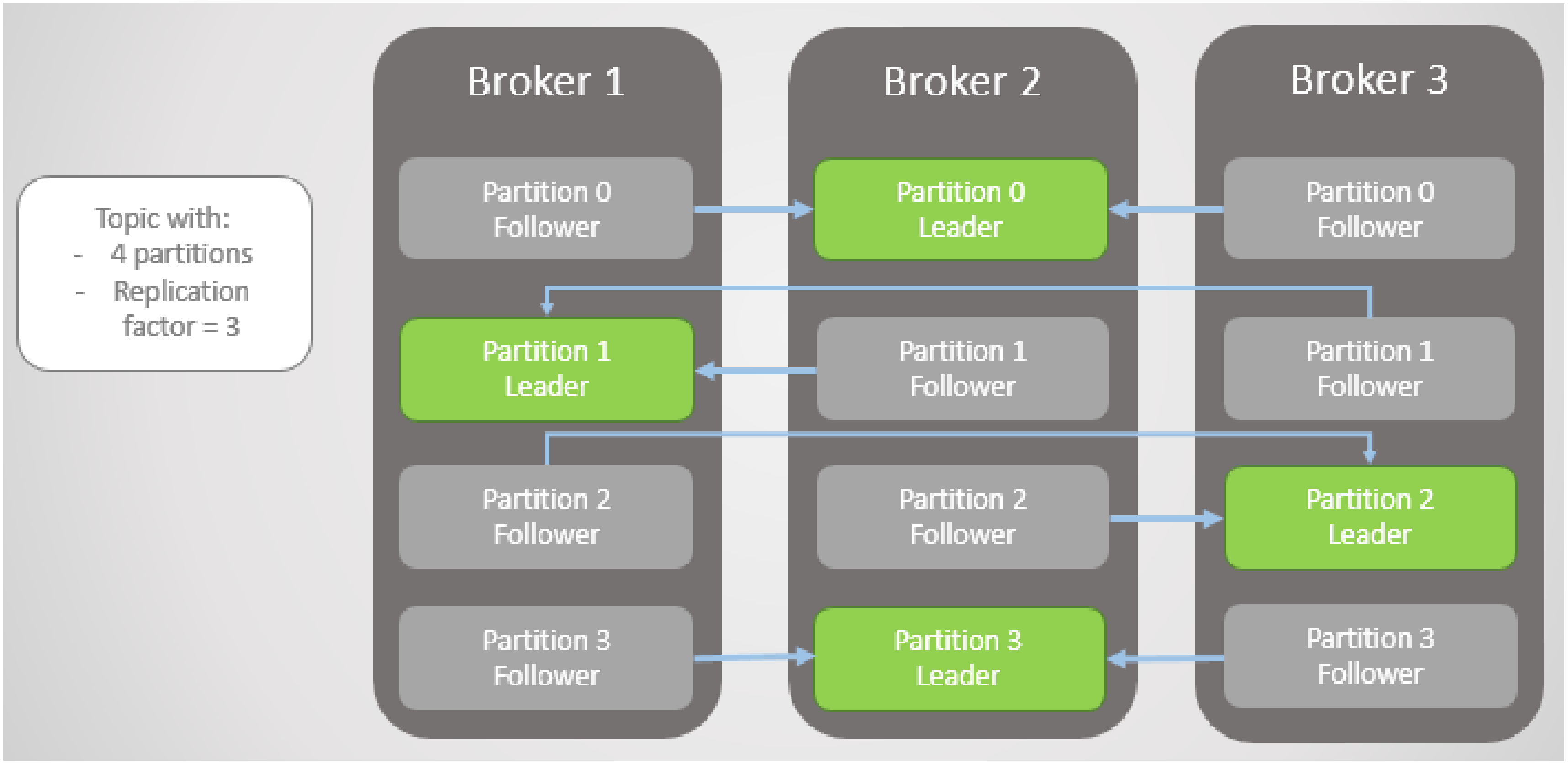
Consumers

- Subscribe to topics and process messages
- Can be organized into groups
  - Each partition in a topic is consumed by only one consumer within the group
- Consumers maintain an offset
- Dynamic addition or removal of consumers


1 2 3 4 5 6 7 8 9 10 11 12
append
Producer client
Producer client
1 2 3 4 5 6 7 8 9 10
1 2 3 4
read sequentially
Consumer group
Consumer client
offsets: (B.1: 4, B.2: 7)
1 2 3 4 5 6 7 8 9
Consumer client
offsets: (B.3: 11)
1 2 3 4 5 6 7 8 9 10 11 12 13 14

Brokers and Clusters

- A single Kafka server is called a Broker
- Brokers are designed to operate as part of a Cluster
  - One broker will function as the cluster’s controller
  - Each partition is owned by a single cluster
- Retention: Messages only expire after some time (7 days) or the partition reach a certain size (1GB)


Topic with:
- 4 partitions
- Replication factor = 3
Broker 1
Broker 2
Broker 3
Partition 0 Follower
Partition 0 Leader
Partition 0 Follower
Partition 1 Leader
Partition 1 Follower
Partition 1 Follower
Partition 2 Follower
Partition 2 Follower
Partition 2 Leader
Partition 3 Follower
Partition 3 Leader
Partition 3 Follower

Use cases

- Activity Tracking
- Messaging
- Metrics and Logging
- Commit Logging
- Stream Processing

Pros & Cons

- ✔ Multiple producers, multiple consumers
- ✔ Disk-based retention
- ✔ Scalable
- ✔ High performance

- ✘ Learning curve
- ✘ Resource overhead
- ✘ Operational Complexity
- ✘ Overkill for small data

The Kafka Ecosystem

- Kafka Connect
- Kafka Streams
- Schema Registry
- MirrorMaker
- ZooKeeper – KRaft Mode

Thank you!