JavaEE平台技术
消息服务器-RocketMQ

邱明 博士

厦门大学信息学院

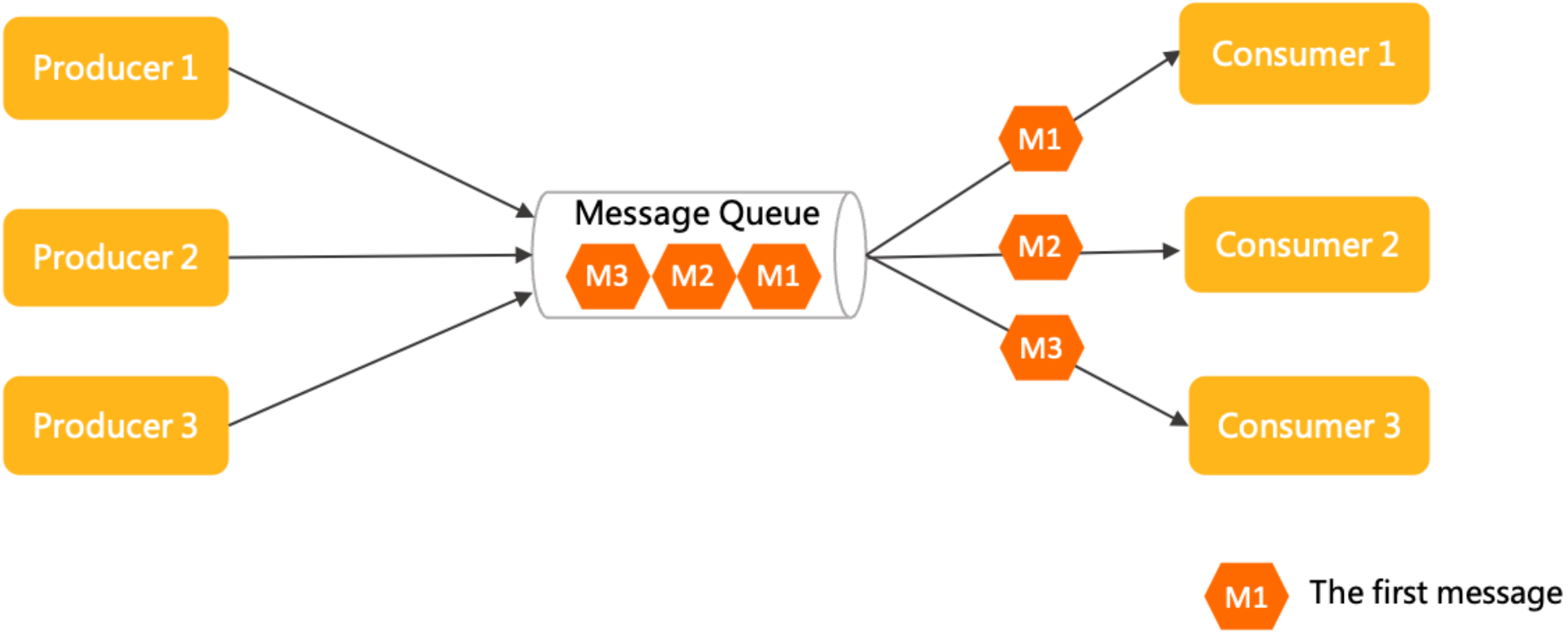
mingqiu@xmu.edu.cn

1 基本概念

- 消息服务是一个分布式异步通讯模型
- 基于发布/订阅的消息模型

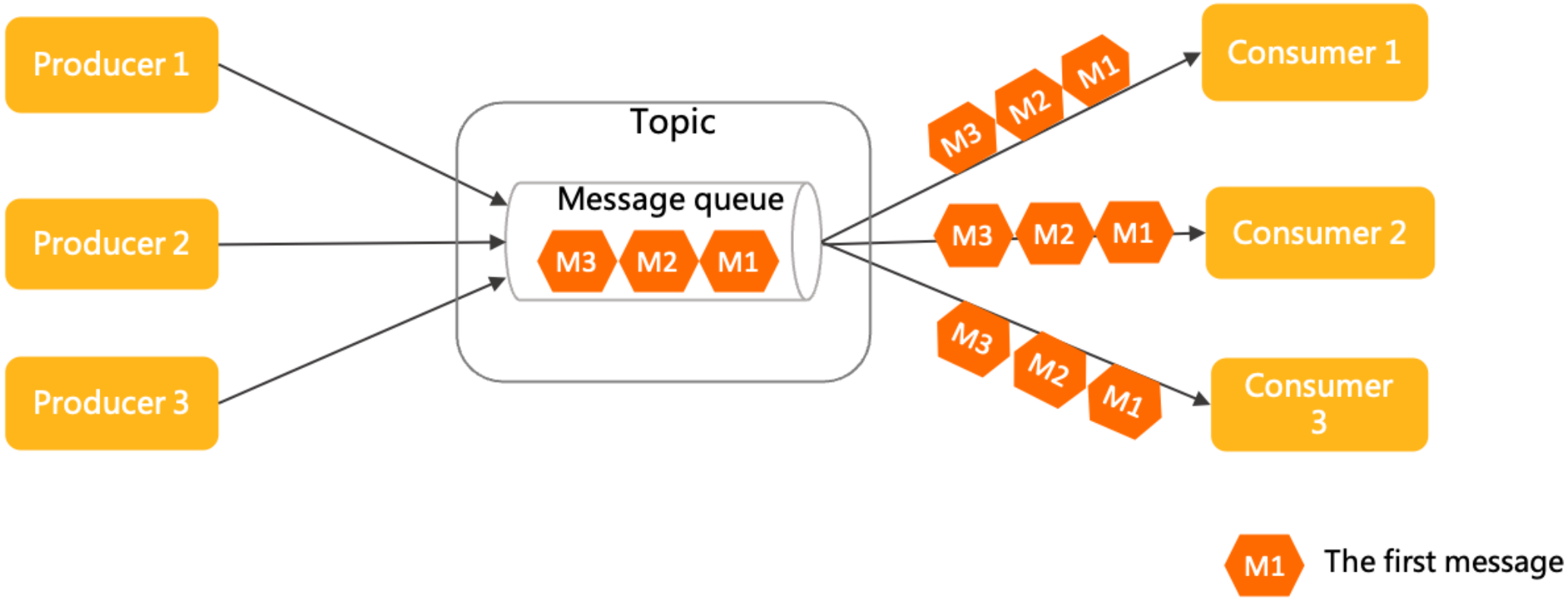
1 基本概念

- 点对点通讯模式



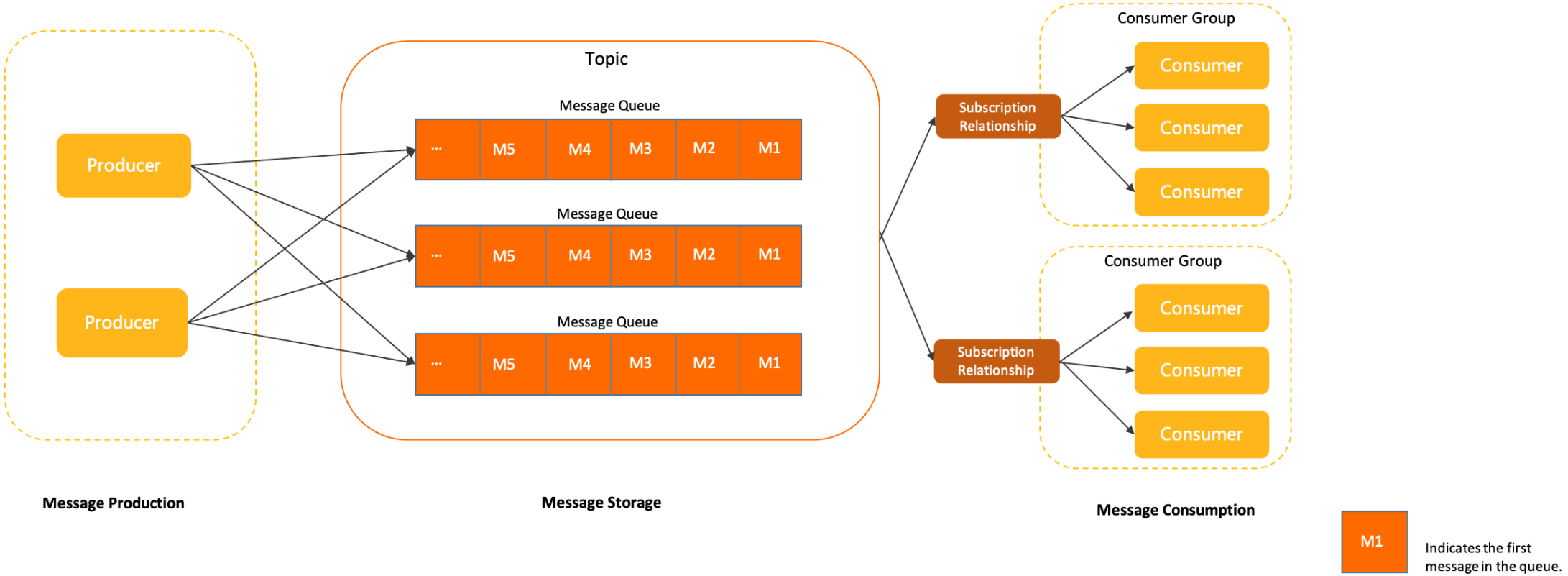
1 基本概念

- 订阅和发布通讯模式

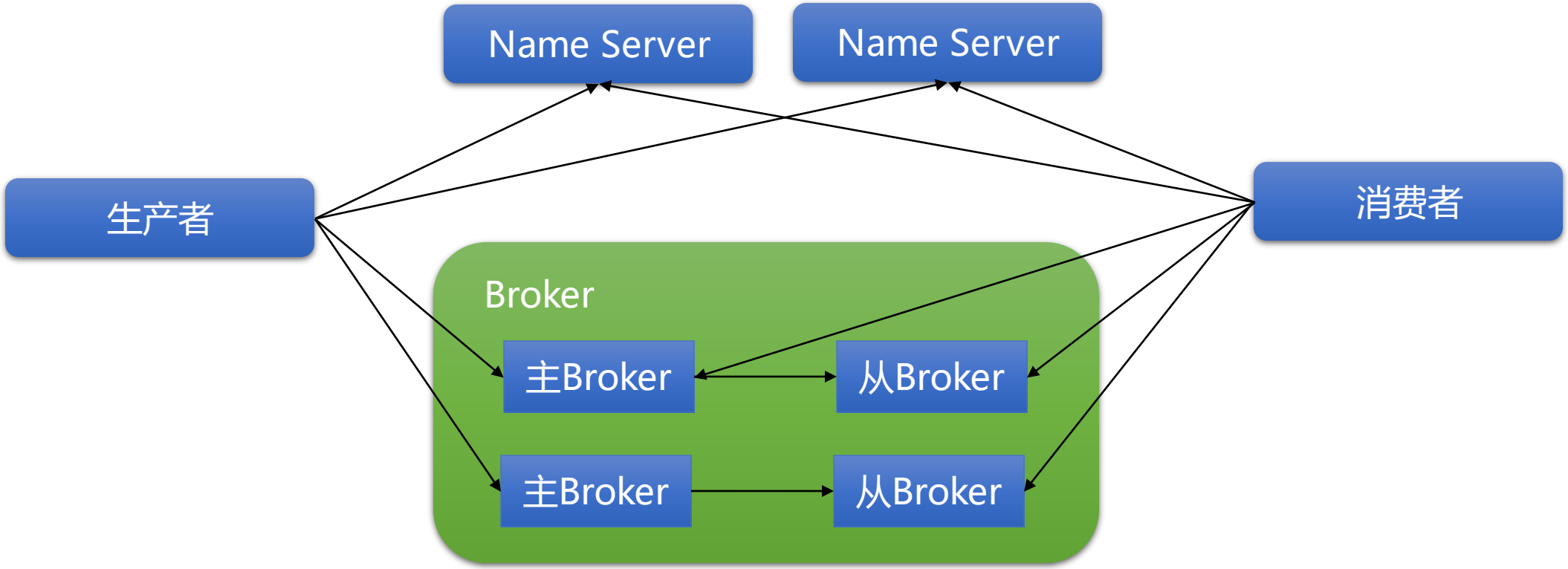


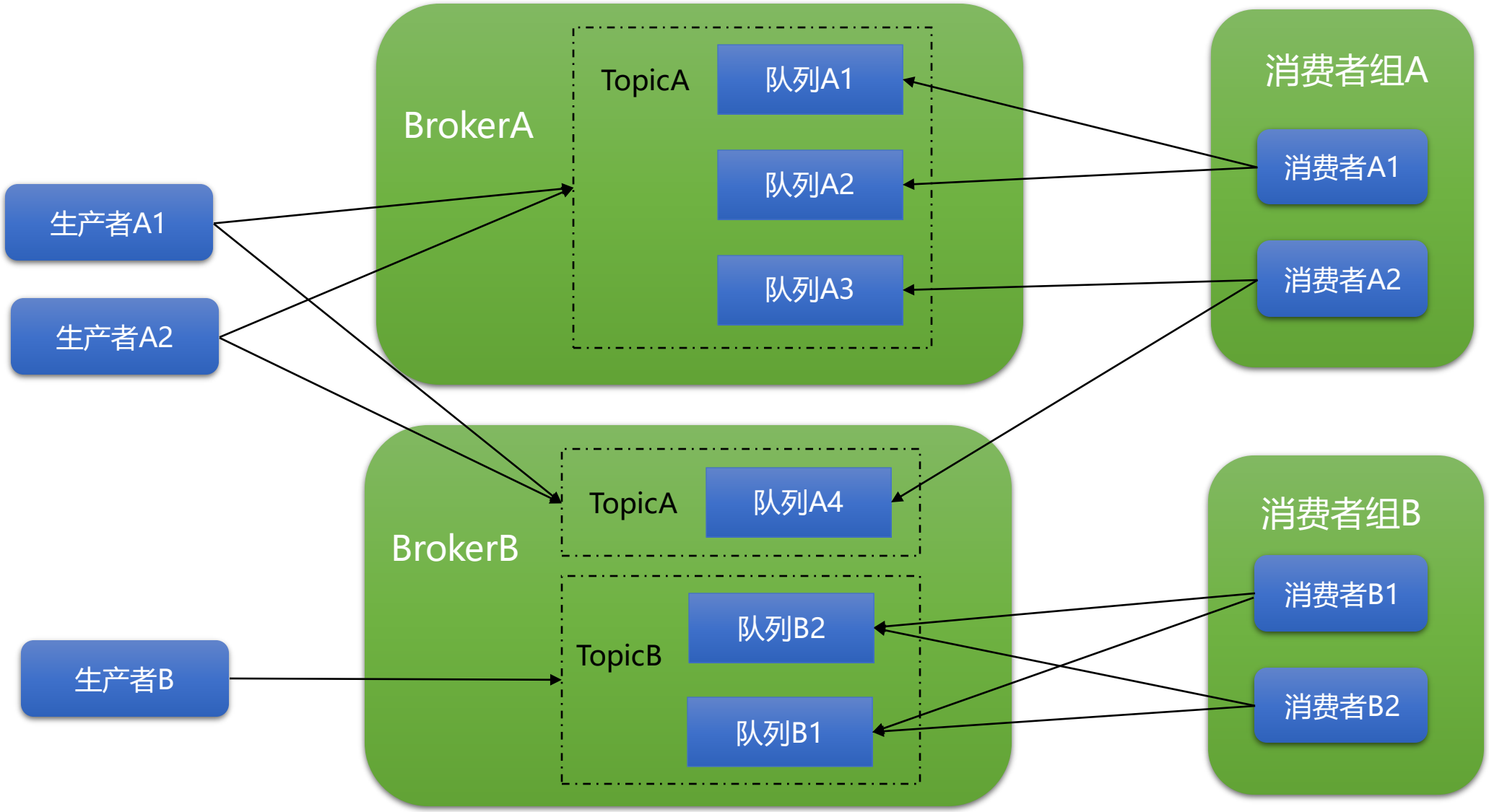
1 基本概念

- Topic-队列的集合



2 RocketMQ的结构




BrokerA
TopicA
队列A1
队列A2
队列A3
生产者A1
生产者A2
消费者组A
消费者A1
消费者A2
BrokerB
TopicA
队列A4
TopicB
队列B2
队列B1
生产者B
消费者组B
消费者B1
消费者B2

3. 消息的类型

- 普通消息


```
public void sendLogMessage(Log log){

    String json = JacksonUtil.toJson(log);
    Message message = MessageBuilder.withPayload(json).build();
    logger.info("sendLogMessage: message = " + message);
    rocketMQTemplate.sendOneWay("log-topic:1",  message);
}
```

3. 消息的类型

- 延时消息

固定的延时等级 "1s 5s 10s 30s 1m 2m 3m 4m 5m 6m 7m 8m 9m 10m 20m 30m 1h 2h"

3. 消息的类型

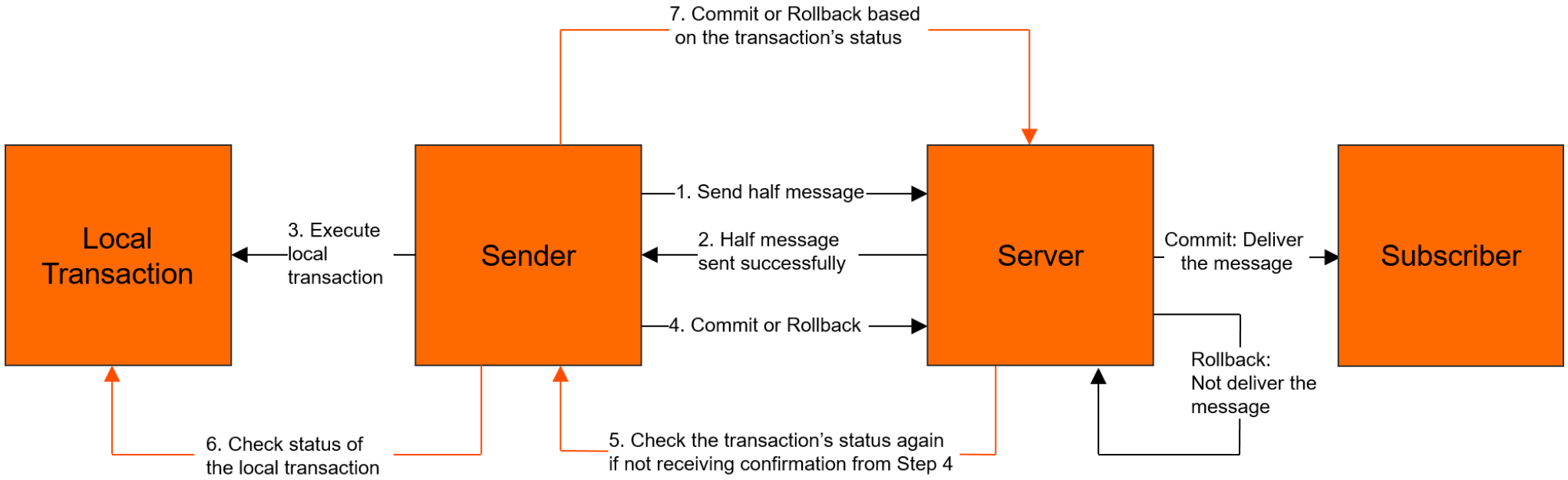
- 延时消息示例

```
public void sendOrderPayMessage(Long orderId){
    logger.info("sendOrderPayMessage: send message orderId = "+orderId+" delay ="+delayLevel+" time =" +LocalDateTime.now());
    rocketMQTemplate.asyncSend("order-pay-topic", MessageBuilder.withPayload(orderId.toString()).build(), new SendCallback() {
        @Override
        public void onSuccess(SendResult sendResult) {
            logger.info("sendOrderPayMessage: onSuccess result = "+ sendResult+" time ="+LocalDateTime.now());
        }

        @Override
        public void onException(Throwable throwable) {
            logger.info("sendOrderPayMessage: onException e = "+ throwable.getMessage()+" time ="+LocalDateTime.now());
        }
    }, timeout * 1000, delayLevel);
}
```

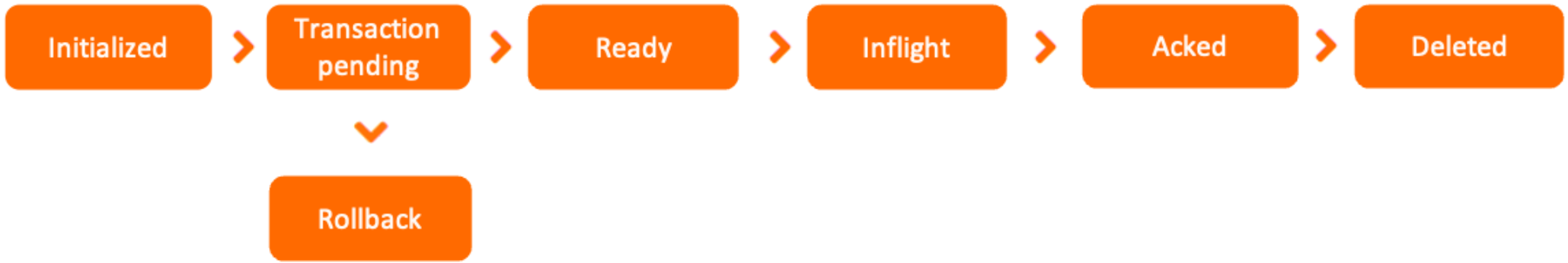
3. 消息的类型

- 事务消息


7. Commit or Rollback based on the transaction's status
Local Transaction
3. Execute local transaction
Sender
1. Send half message
2. Half message sent successfully
4. Commit or Rollback
Server
Commit: Deliver the message
Subscriber
Rollback: Not deliver the message
6. Check status of the local transaction
5. Check the transaction's status again if not receiving confirmation from Step 4

3. 消息的类型

- 事务消息


Initialized
Transaction pending
Ready
Inflight
Acked
Deleted
Rollback

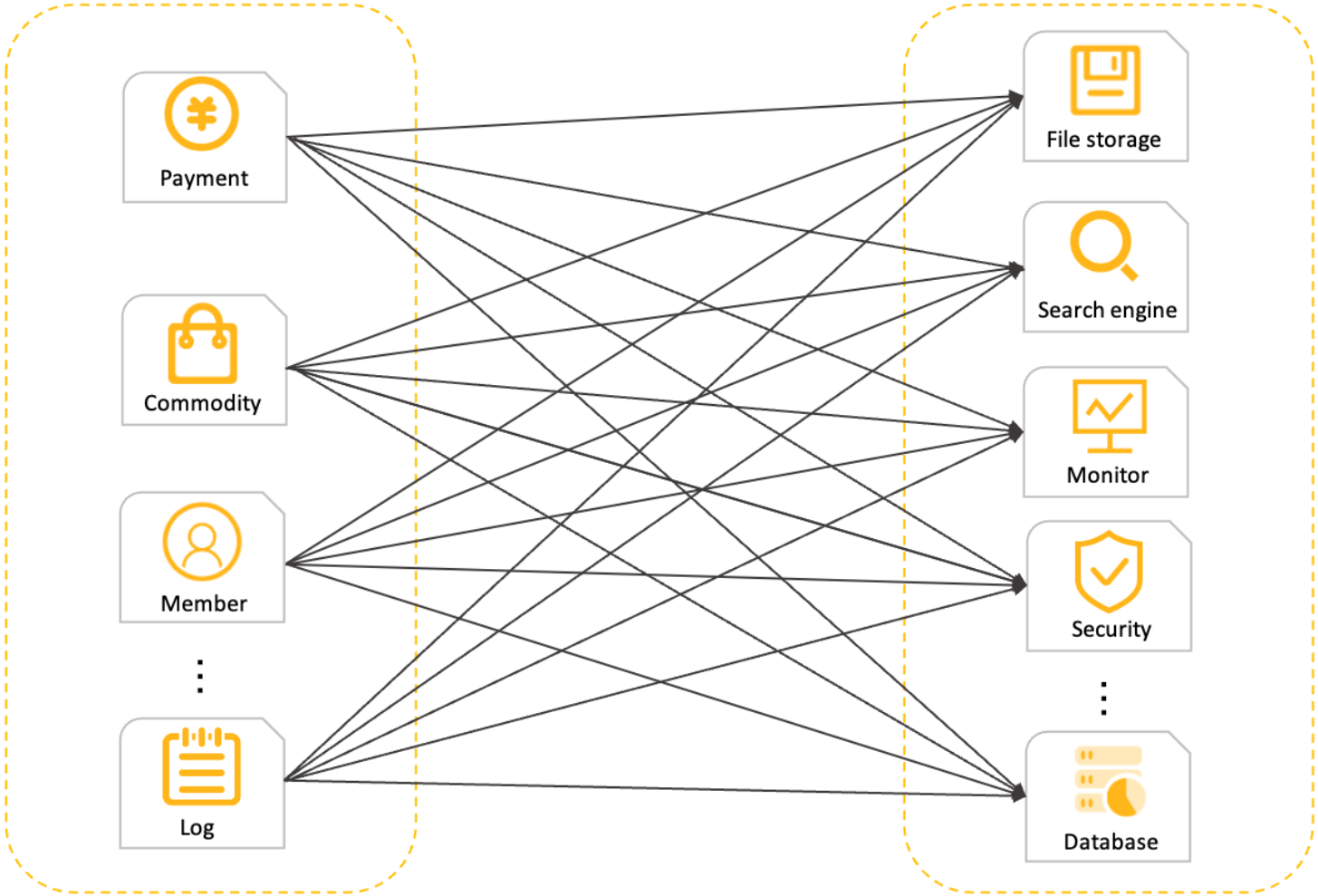
4. 消息的消费者

```
@Service
@RocketMQMessageListener(topic = "log-topic", selectorExpression = "1", consumeMode = ConsumeMode.CONCURRENTLY, consumeThreadMax = 10, consumerGroup = "log-group")
public class LogConsumerListener implements RocketMQListener<String>, RocketMQPushConsumerLifecycleListener {
    private static final Logger logger = LoggerFactory.getLogger(LogConsumerListener.class);
    @Override
    public void onMessage(String message) {
        Log log = JacksonUtil.toObj(message, Log.class);
        logger.info("onMessage: got message log =" + log);
    }

    @Override
    public void prepareStart(DefaultMQPushConsumer defaultMQPushConsumer) {
//        defaultMQPushConsumer.setConsumeFromWhere(ConsumeFromWhere.CONSUME_FROM_TIMESTAMP);
//        defaultMQPushConsumer.setConsumeTimestamp(UtilAll.timeMillisToHumanString3(System.currentTimeMillis()));
        logger.info("prepareStart: consumergroup =" + defaultMQPushConsumer.getConsumerGroup());
    }
}
```

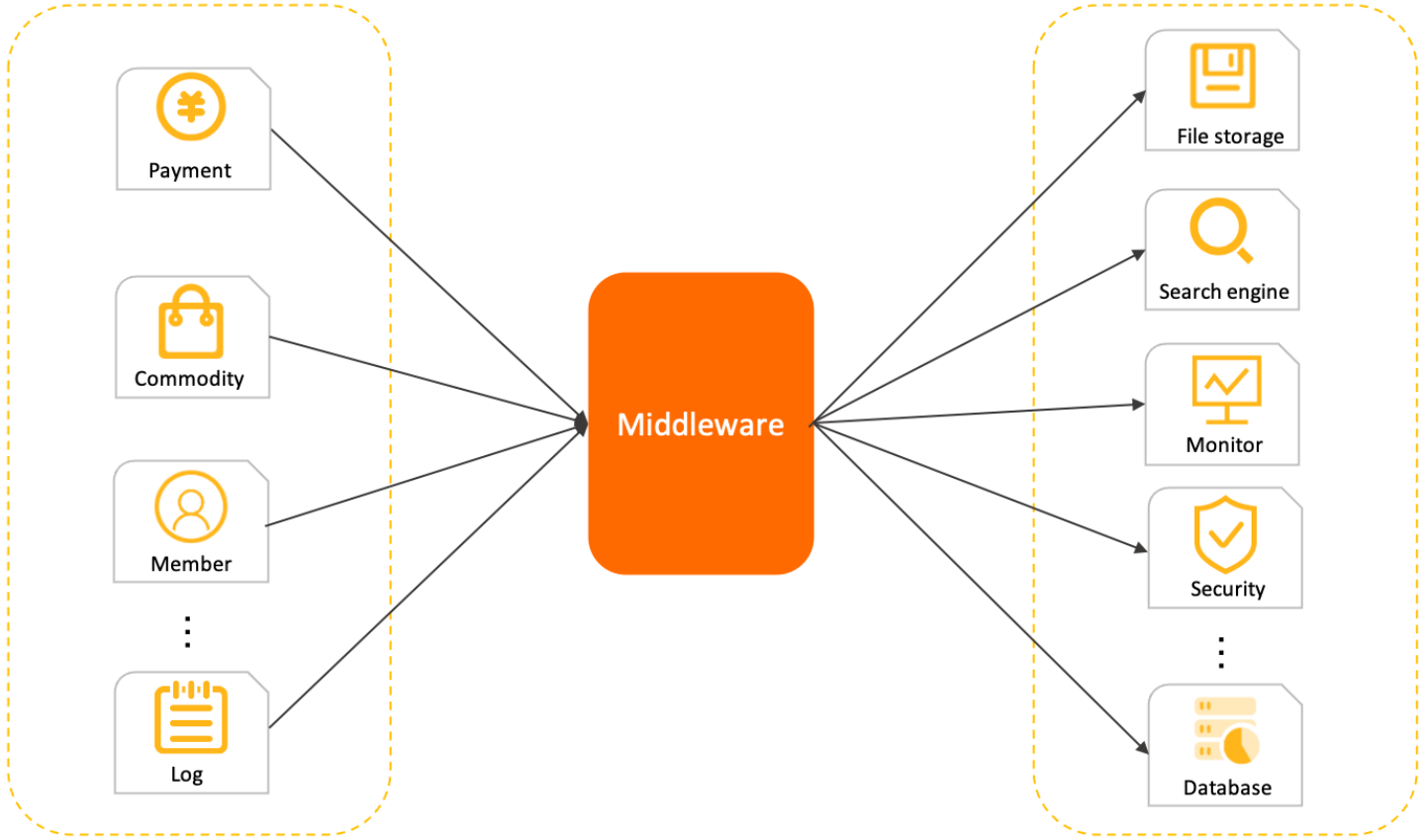
4 消息服务器的作用

- 系统解耦



4 消息服务器的作用

- 系统解耦



4 消息服务器的作用

- 系统解耦