Day 1
Big Data Application & Real-time Streaming: The first chapter

Lecturer: M.S. Le Minh Tan

Introduction

- Le Minh Tan
- Master of Computer Science
- Student of HCMUTE
- Artificial Intelligence
- tanlm@hcmute.edu.vn
- https://fb.com/roggertanvus
- 0932.751.620 (Zalo)

So what’s the plan?

- Progress score: Weekly homework + 2 tests (each x2)
- Final score: Exam
- Preparation:
  - Laptop
  - electric socket
  - VMWare Workstation/Virtual Box/Hyper V/...
  - Ubuntu Desktop/Server >= 18 (LTS)
  - Visual Studio + C#
- Please check out the **subject curriculum on UTEx**.

Textbooks

- A. G. Psaltis, “Streaming Data - Understanding the real-time pipeline,” 2017.
- Gerard Maas & Francois Garillot, "Stream Processing with Apache Spark: Mastering Structured Streaming and Spark Streaming", 2020.
- M. Guller, Big Data Analytics with Spark. Berkeley, CA: Apress, 2015.
- S. Gupta, “Real-Time Big Data Analytics,” February 2016, Published by Packt Publishing Ltd.
- Z. Nabi, Pro Spark Streaming. Berkeley, CA: Apress, 2016.

You all need a group!

- Yes, each needs to be in a group for some activities.
- Steps:
  1. Gather 3-4 members.
  2. Vote for a leader.
  3. Post the list of members (ids + full names), leader id to FB group.
- Rules:
  1. No multiple groups per student.
  2. Deadline: Before class ends.
  3. Those with no group need to contact me in one week.


GUY WHO DOES MOST OF THE WORK IN THE GROUP PROJECT
ME TRYING TO LOOK USEFUL

Concepts

Real-time system

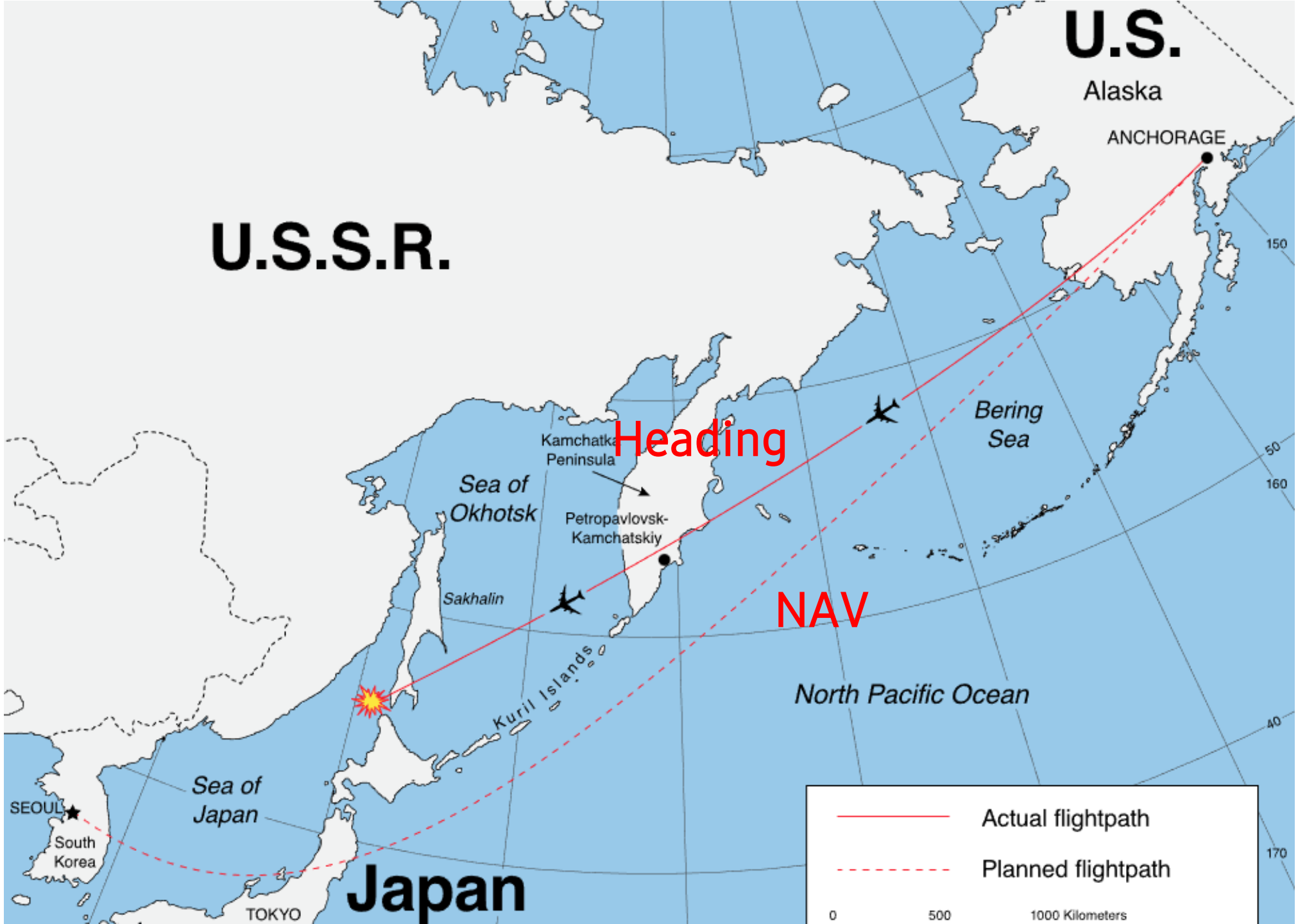
Data streaming

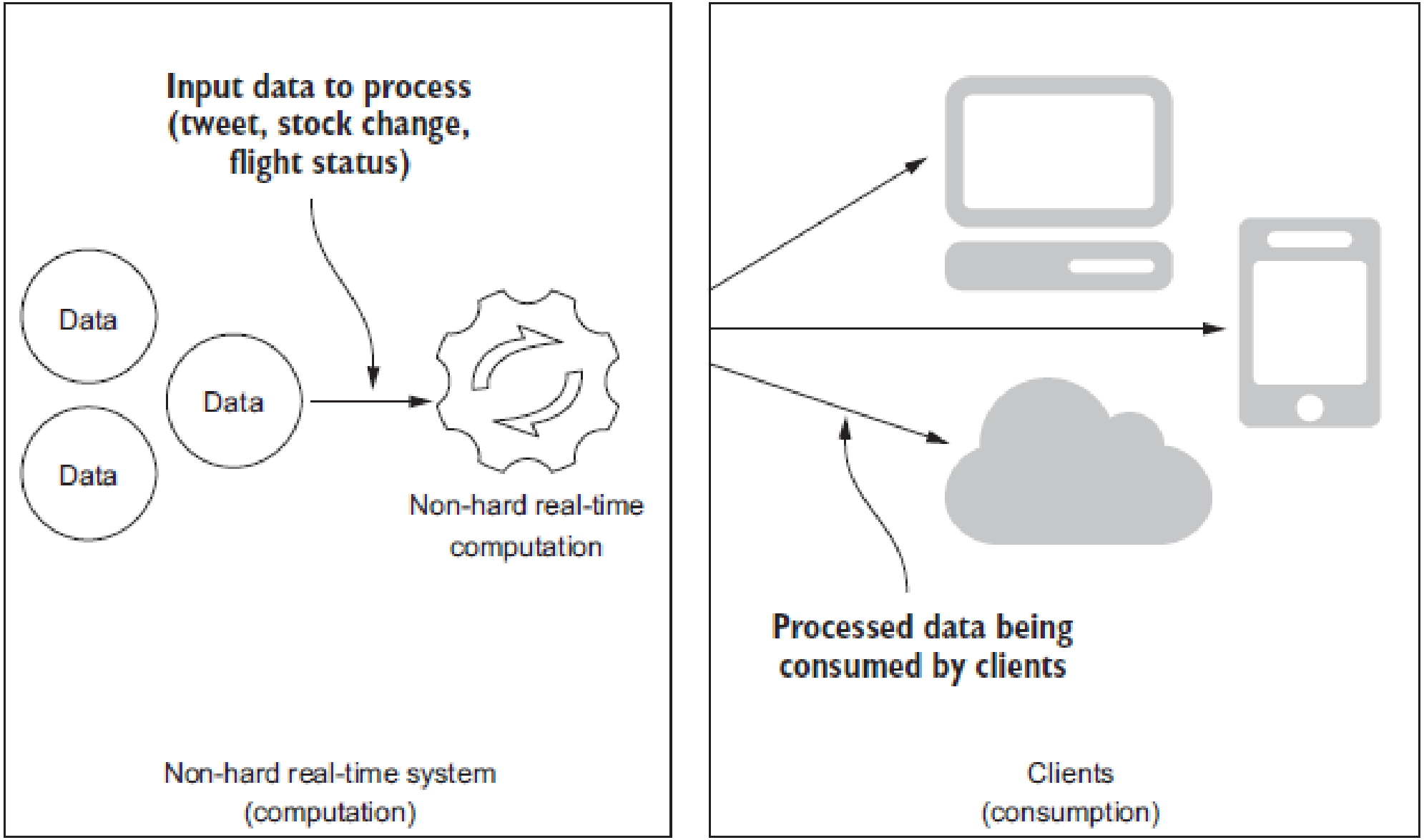
1. Where have you heard of the word “stream” from?
2. What applications/functions that need to work in real-time?

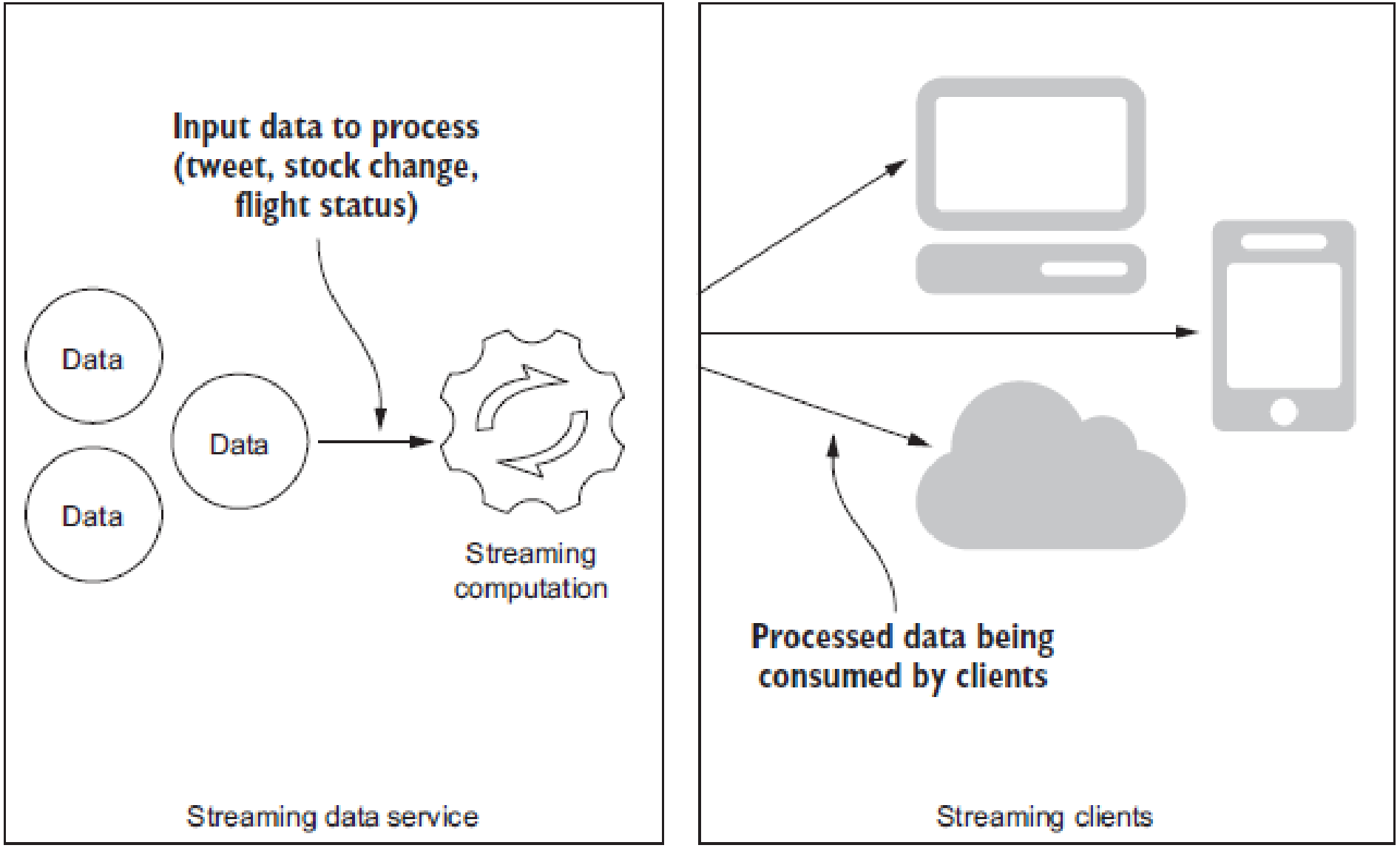
- Real-time system: A computer system where the correctness of the system behavior depends not only on the **logical results** of the computations but also on the **physical time** when these results are produced.
  - Hard: Microseconds – milliseconds, high risk.
  - Soft: Milliseconds – seconds, low risk.
  - Near: Seconds – minutes, low risk.
- Risk: Will there be any related system down or life at risk?

Case study: Korean Flight 007 accident

- Sep. 1st, 1983
  - Cold war (1947 – 1991)
- NY > Seoul
- Passengers: 246
- Crew: 23
- Fatalities: 269
- Survivors: 0

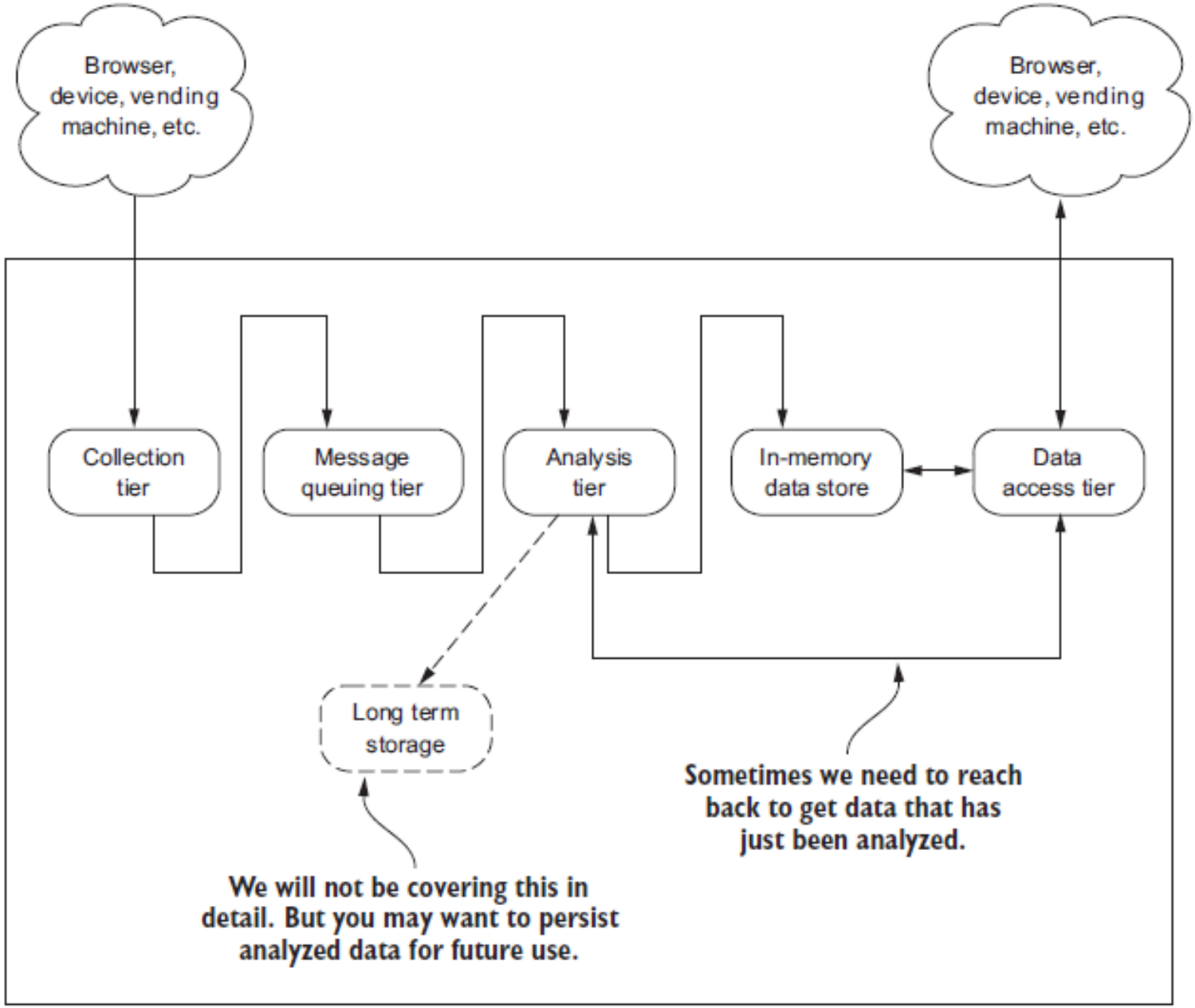

U.S.
Alaska
ANCHORAGE
U.S.S.R.
Heading
Bering Sea
Kamchatka Peninsula
Sea of Okhotsk
Petropavlovsk-Kamchatskiy
Sakhalin
NAV
Kuril Islands
North Pacific Ocean
Sea of Japan
SEOUL
South Korea
TOKYO
Japan
150
50
160
40
170
Actual flightpath
Planned flightpath
0
500
1000 Kilometers


Input data to process
(tweet, stock change,
flight status)
Data
Data
Data
Non-hard real-time
computation
Non-hard real-time system
(computation)
Processed data being
consumed by clients
Clients
(consumption)

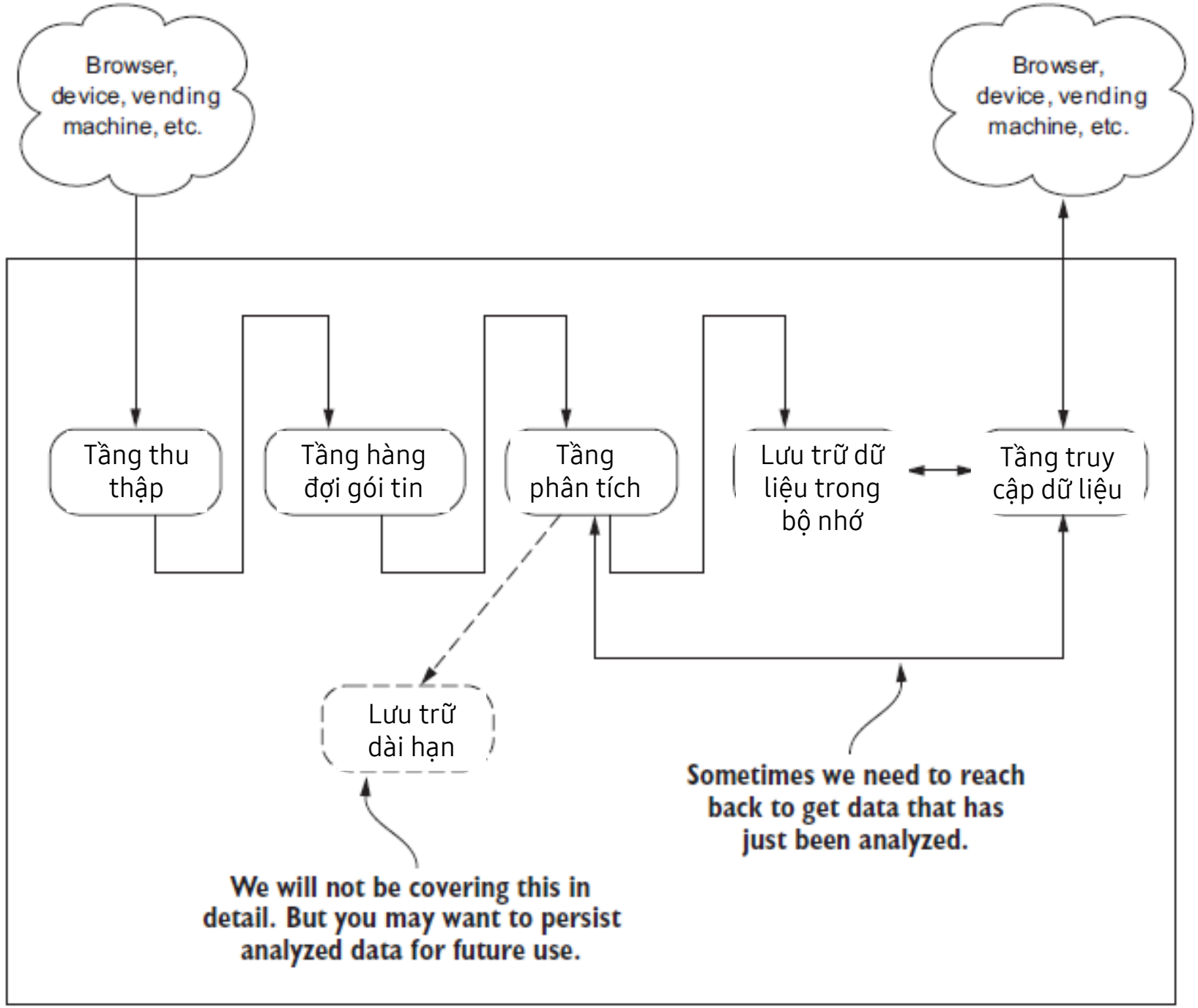


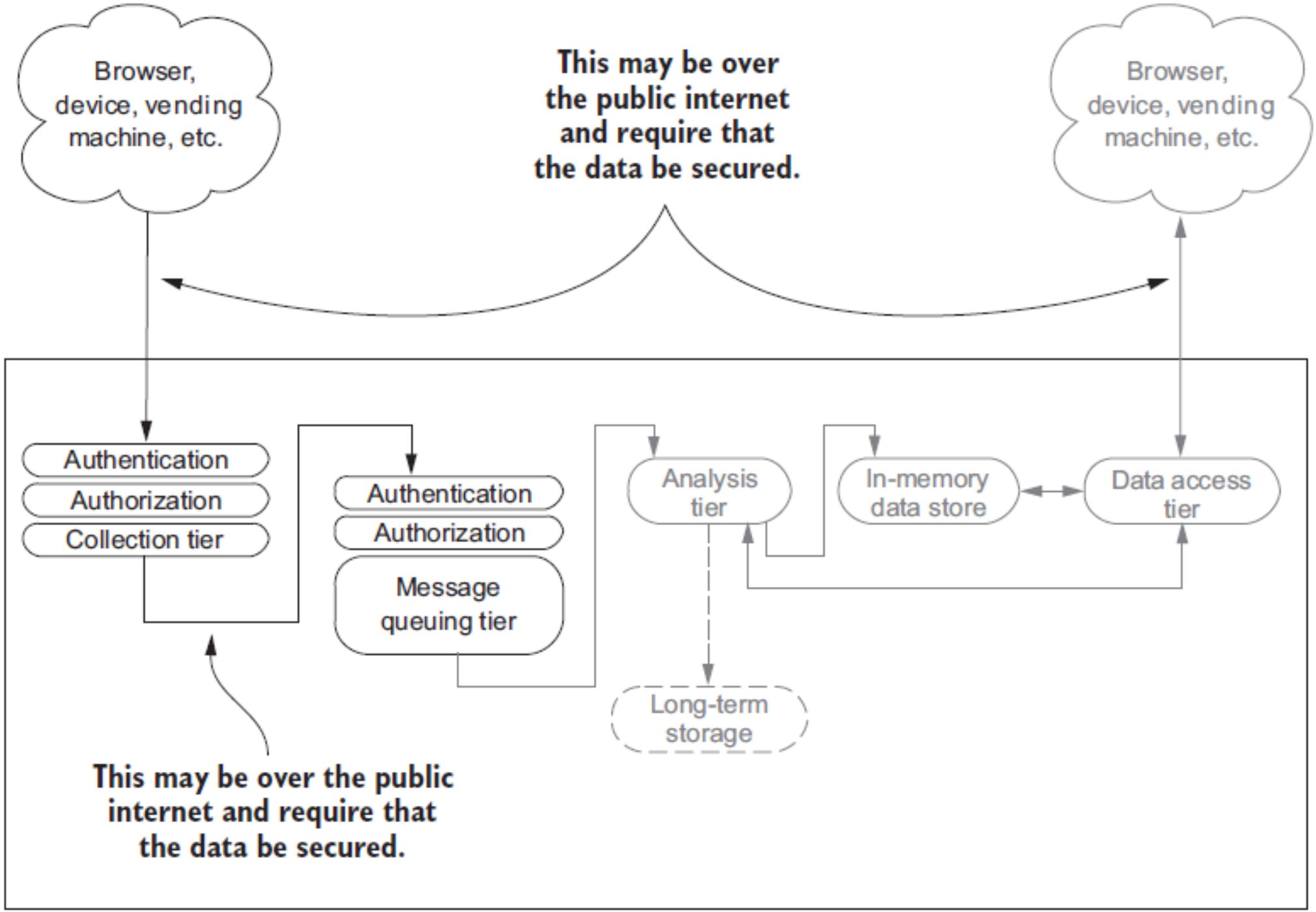
Streaming data system

A non-hard real-time system



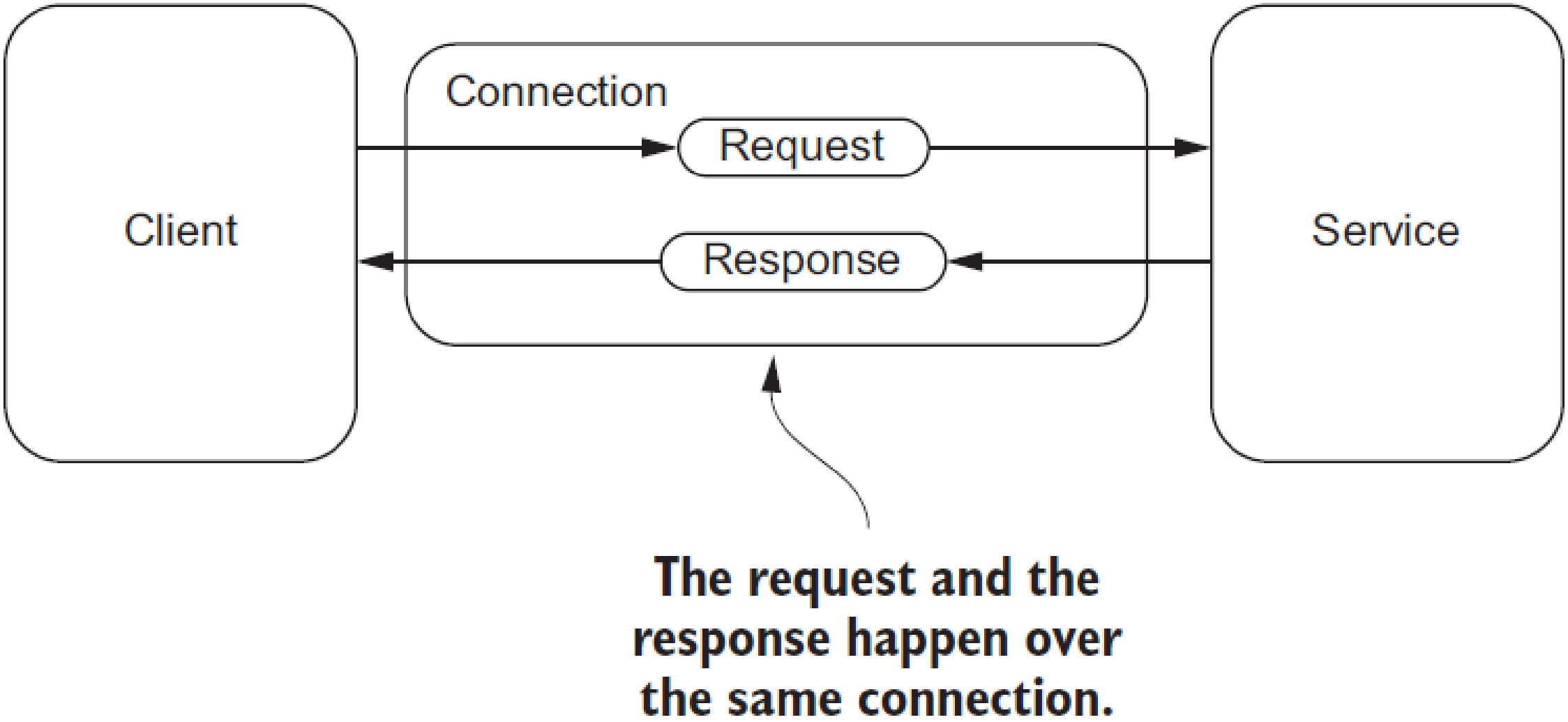
Architectural of streaming system


Browser, device, vending machine, etc.
Browser, device, vending machine, etc.
Tầng thu thập
Tầng hàng đợi gói tin
Tầng phân tích
Lưu trữ dữ liệu trong bộ nhớ
Tầng truy cập dữ liệu
Lưu trữ dài hạn
We will not be covering this in detail. But you may want to persist analyzed data for future use.
Sometimes we need to reach back to get data that has just been analyzed.

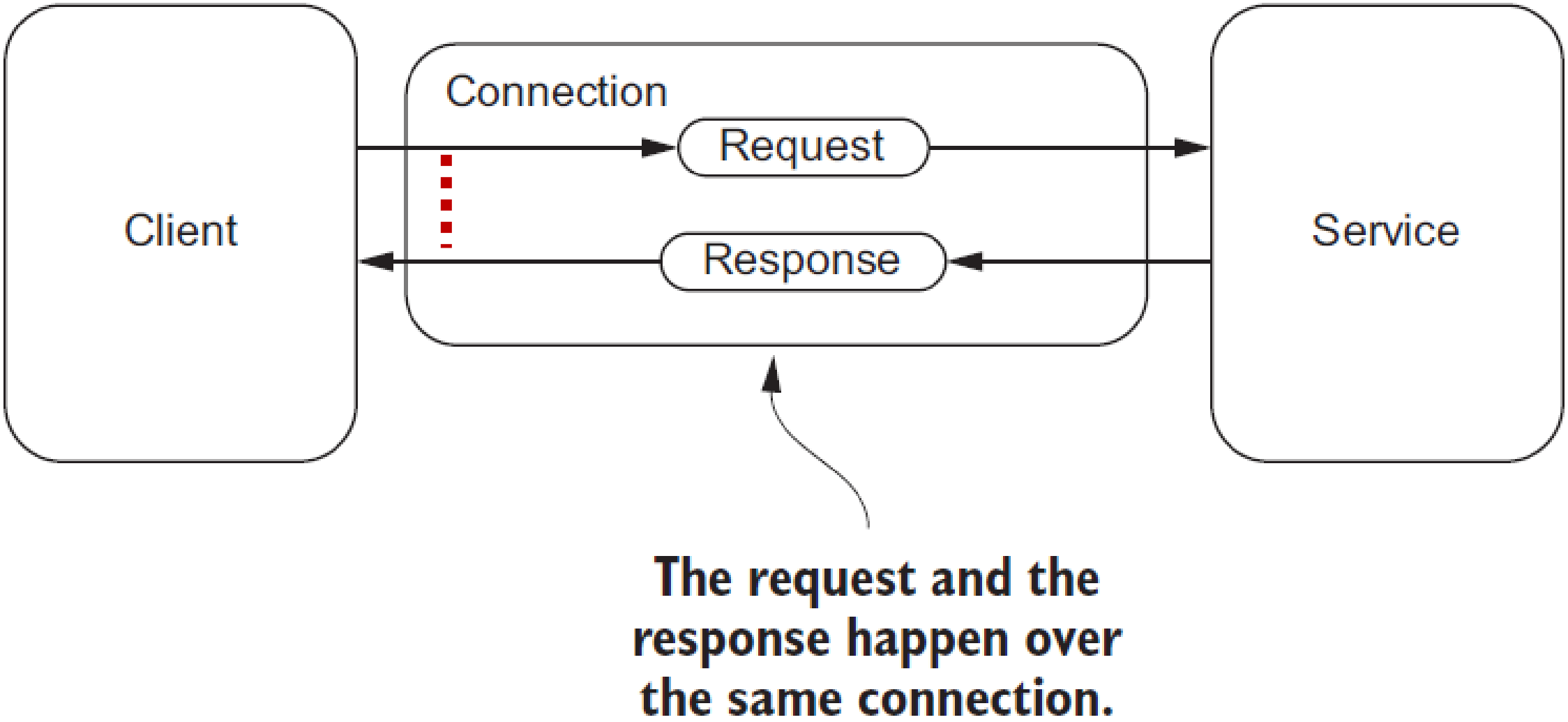

Browser, device, vending machine, etc.
This may be over the public internet and require that the data be secured.
Browser, device, vending machine, etc.
Authentication
Authorization
Collection tier
Authentication
Authorization
Message queuing tier
Analysis tier
In-memory data store
Data access tier
Long-term storage
This may be over the public internet and require that the data be secured.

Data ingestion: Common interation patterns

- Our concern: Collection tier.
- Multiple common interaction patterns:
    1. Request/response pattern
    2. Request/acknowledge pattern
    3. Publish/subscribe pattern
    4. One-way pattern
    5. Stream pattern



1. Basic request/response pattern



1. Basic request/response pattern

Prepare the material for soup

Put breads in

Check status
Update temperature,...

Checking status

Prepare the material for soup

Put breads in

Check status
Update temperature,...

Sending signal

Prepare the material for soup

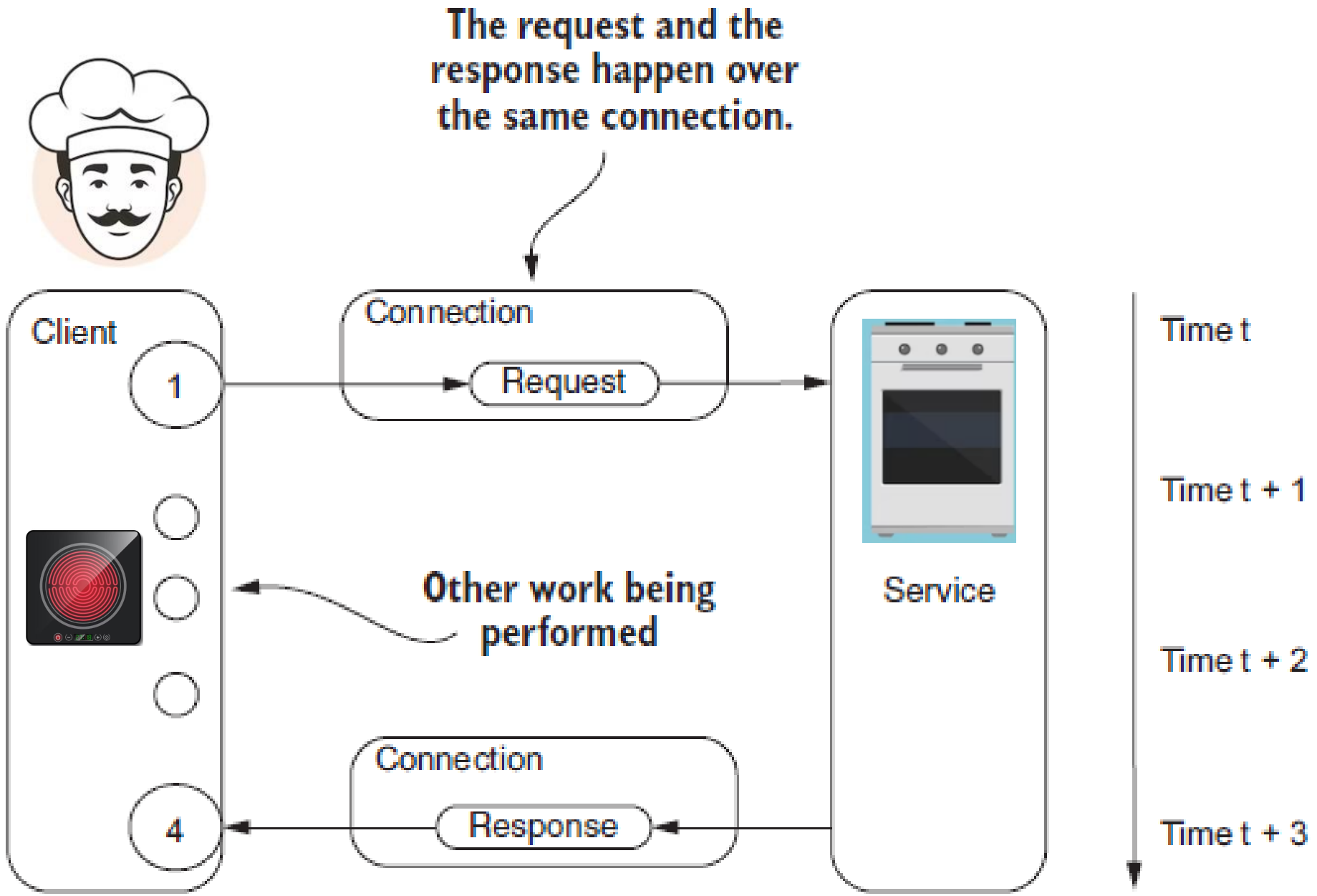
Put breads in

Check status
Update
temperature,...

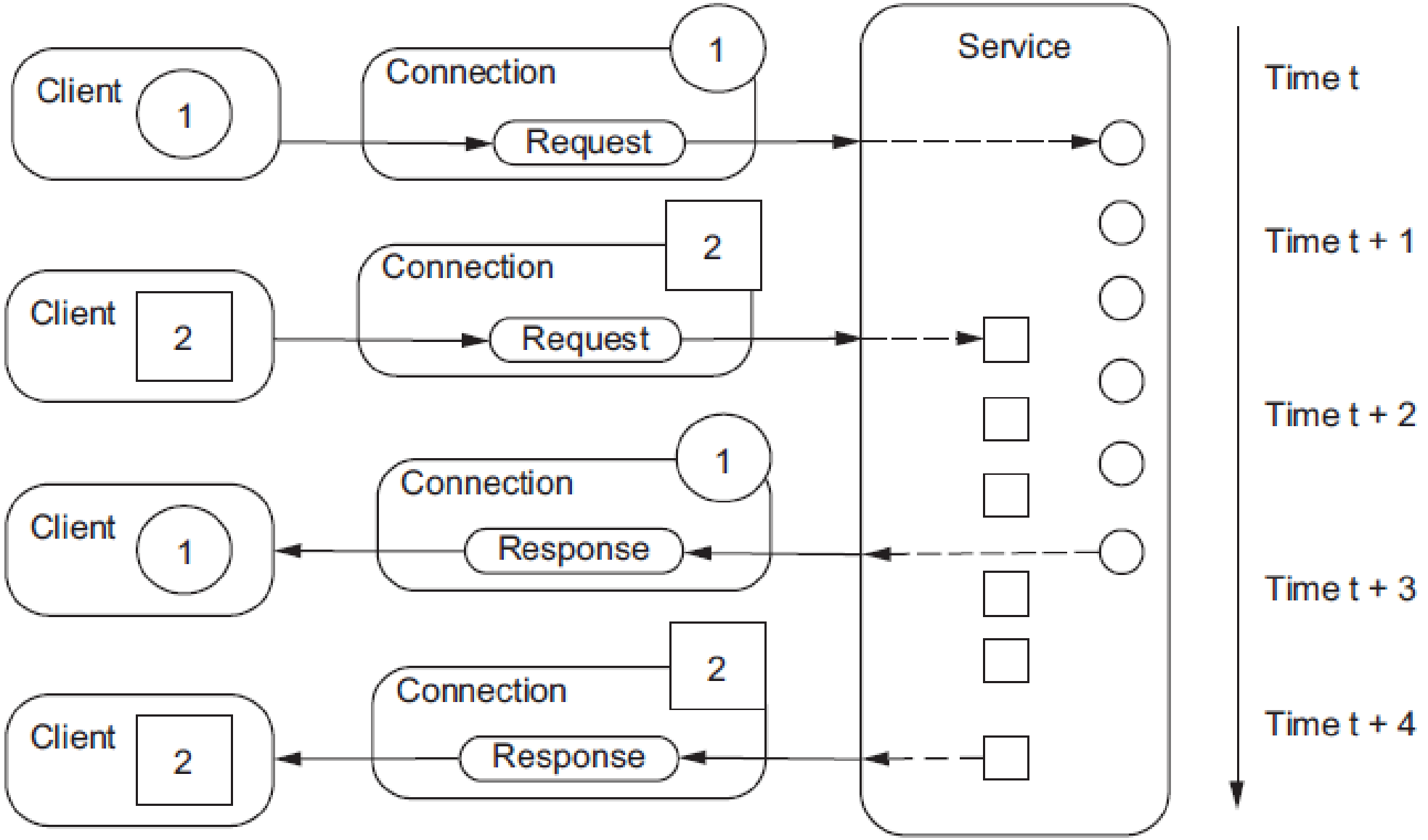
Have the signal come yet?

Check status
Update
temperature,...

Sending signal

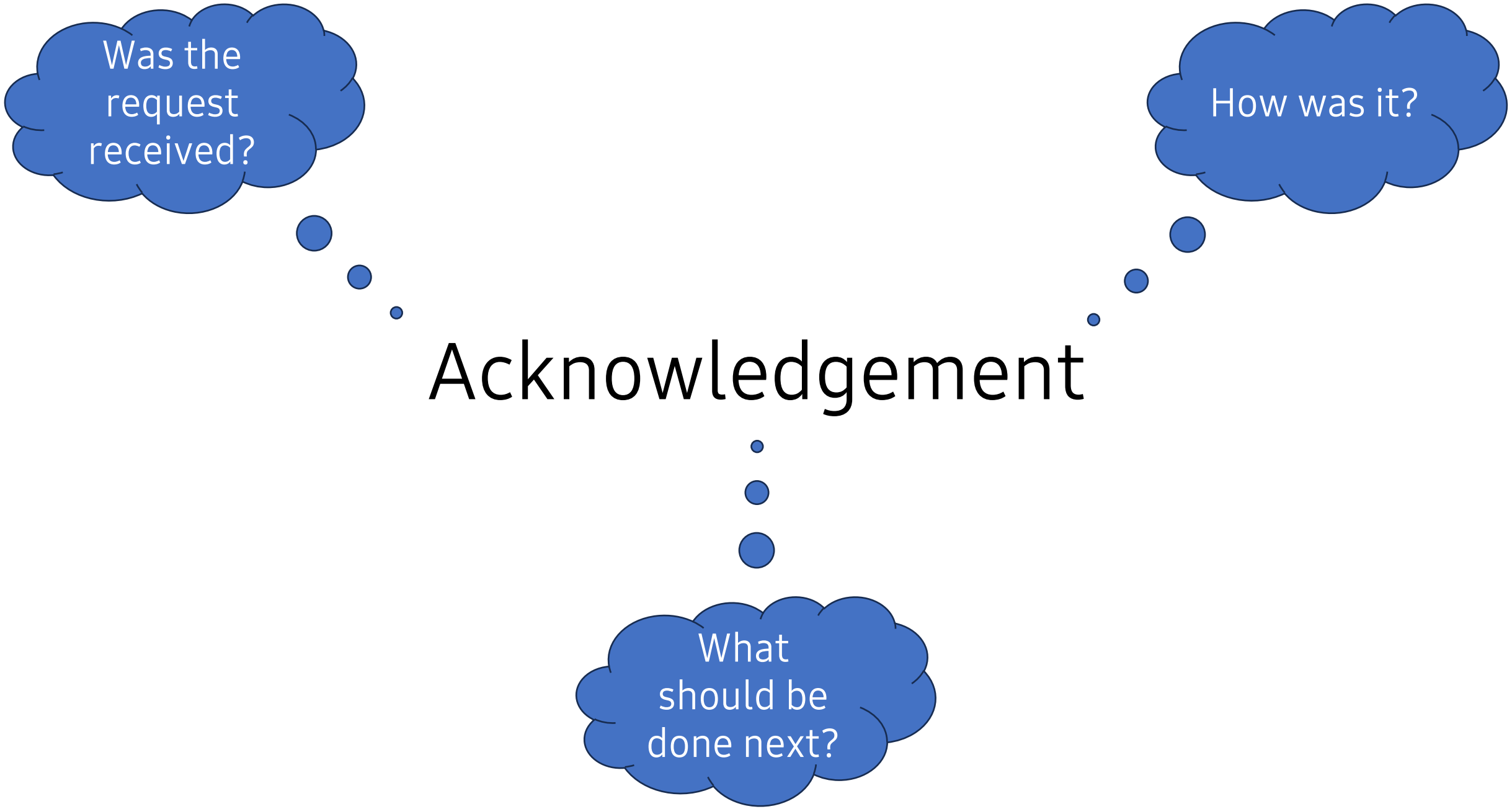

The request and the response happen over the same connection.
Client
1
Connection
Request
Service
Time t
Time t + 1
Other work being performed
Time t + 2
Connection
Response
4
Time t + 3

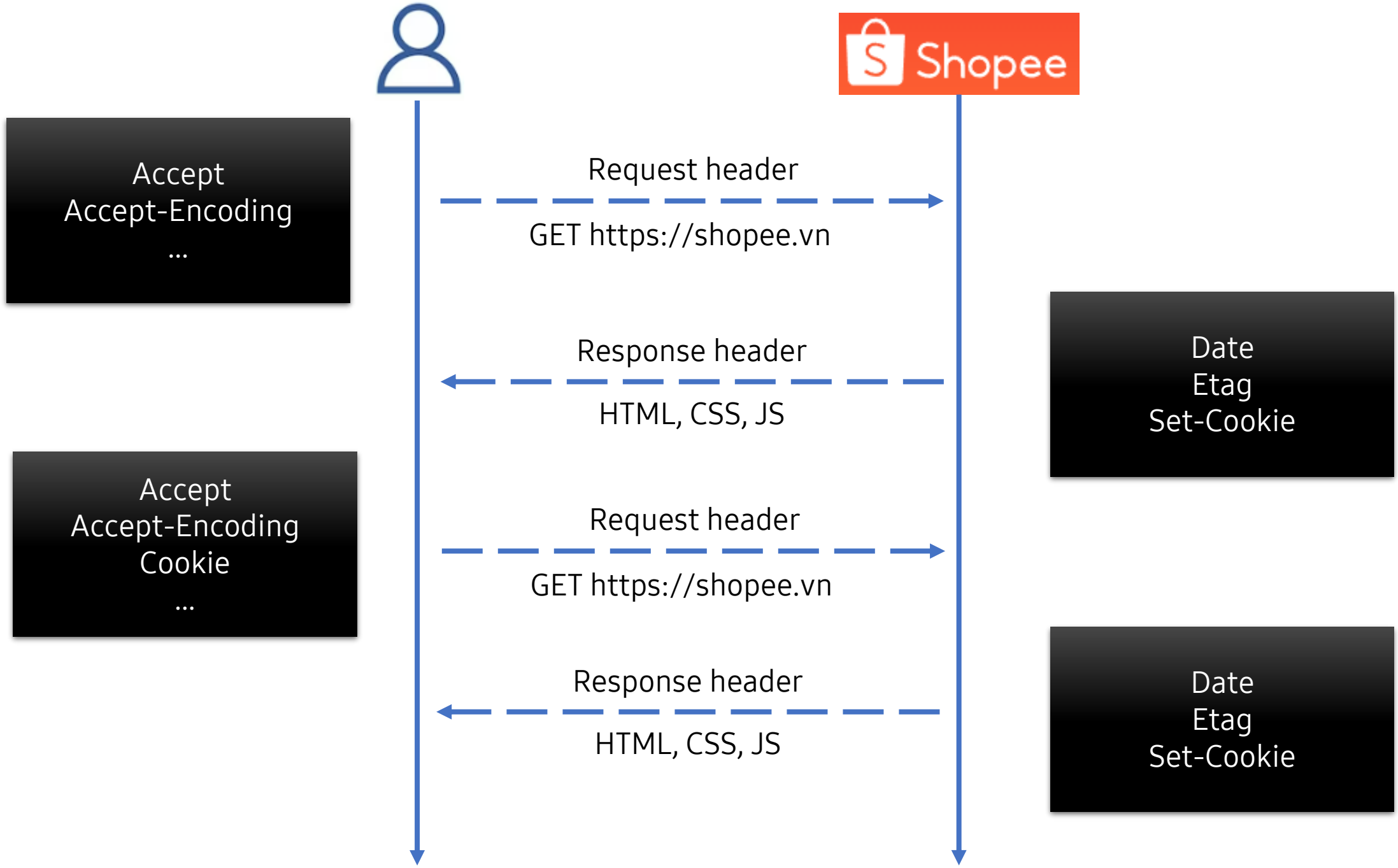
Half-async

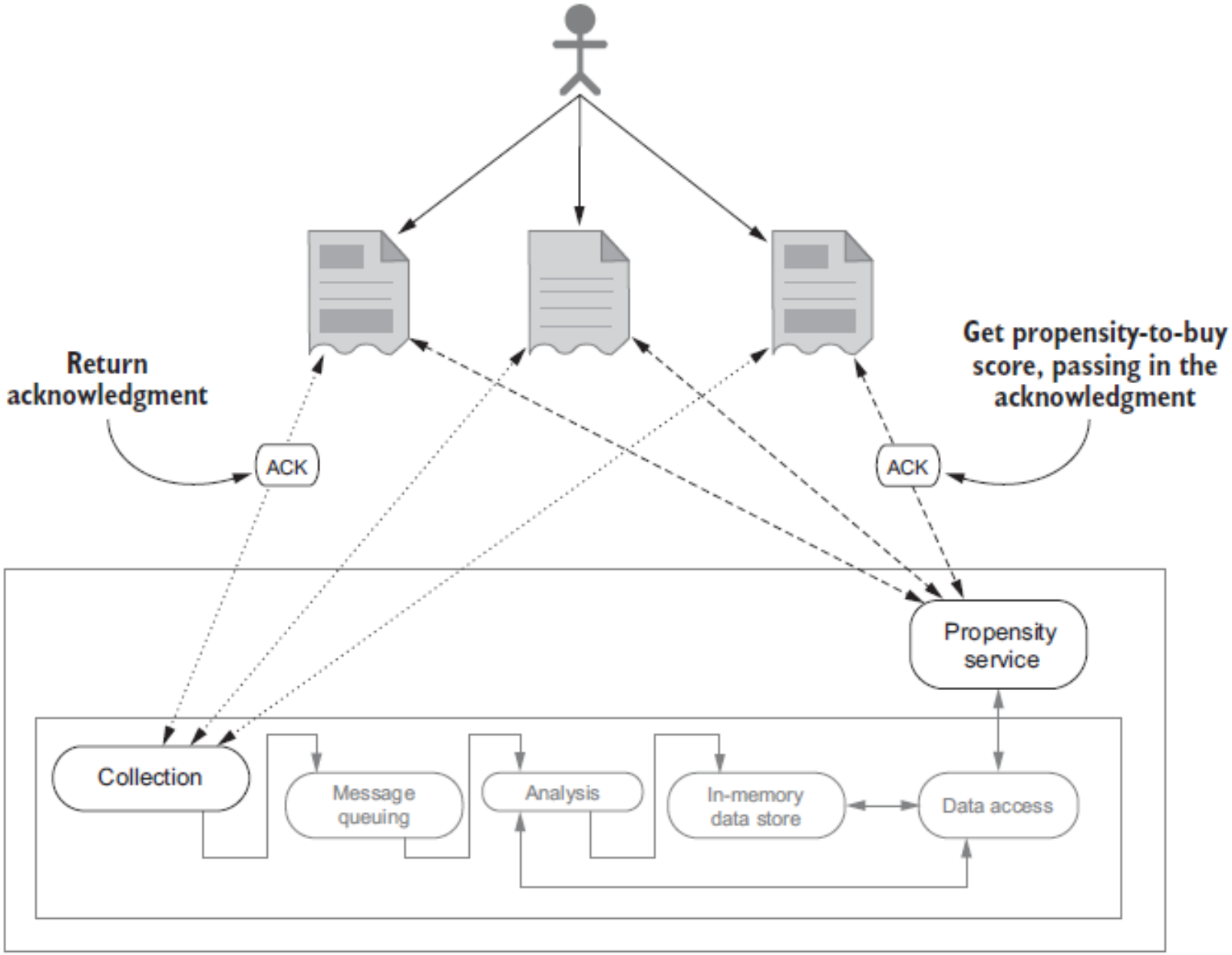


Service-side pattern

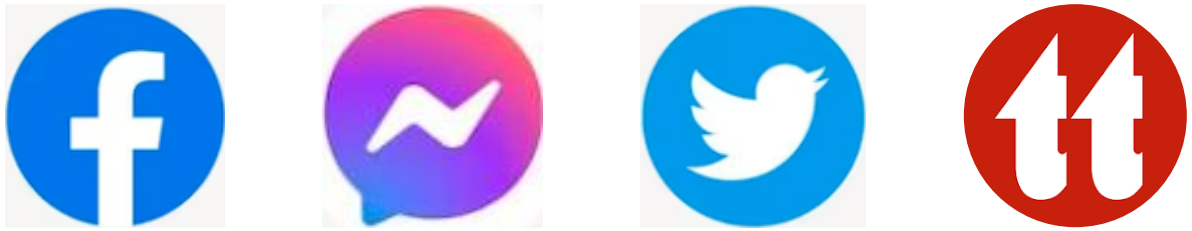
Can be half-async or full-async

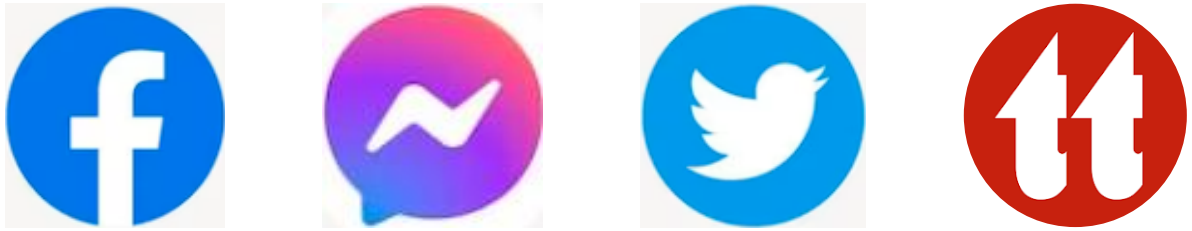

Was the request received?
How was it?
Acknowledgement
What should be done next?

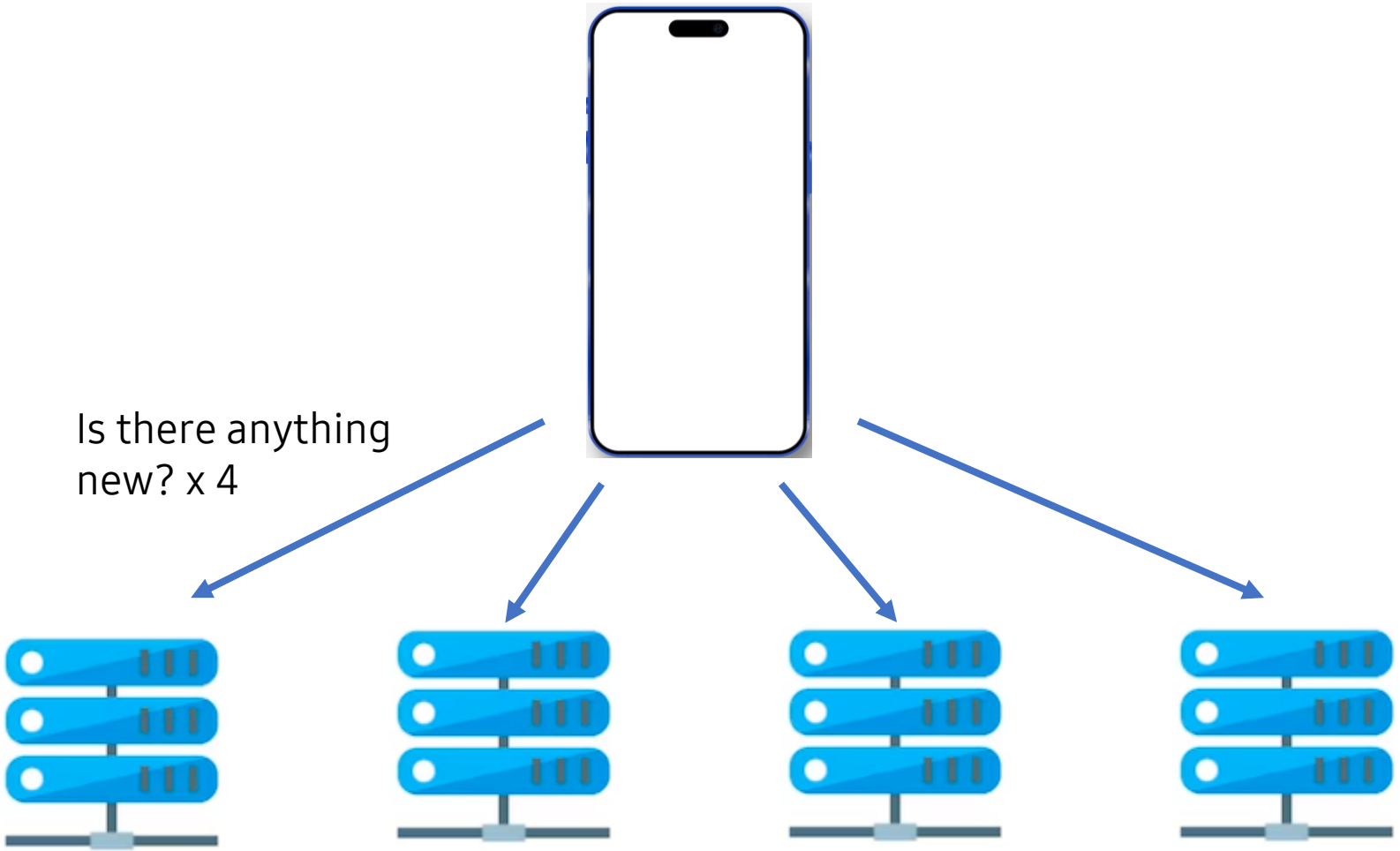

Shopee
Accept
Accept-Encoding
...
Request header
GET https://shopee.vn
Response header
HTML, CSS, JS
Date
Etag
Set-Cookie
Accept
Accept-Encoding
Cookie
...
Request header
GET https://shopee.vn
Response header
HTML, CSS, JS
Date
Etag
Set-Cookie

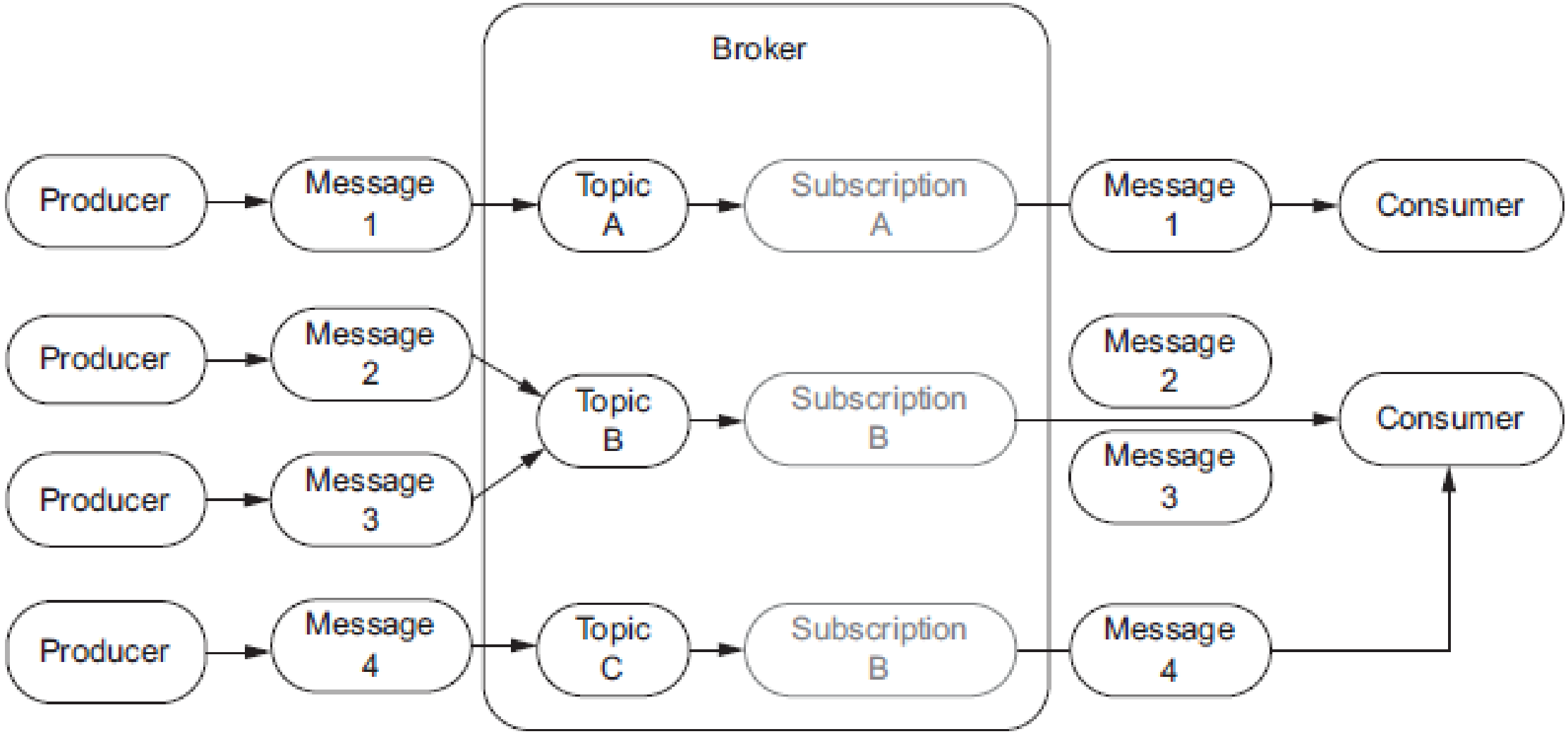


2. Request/acknowledge pattern

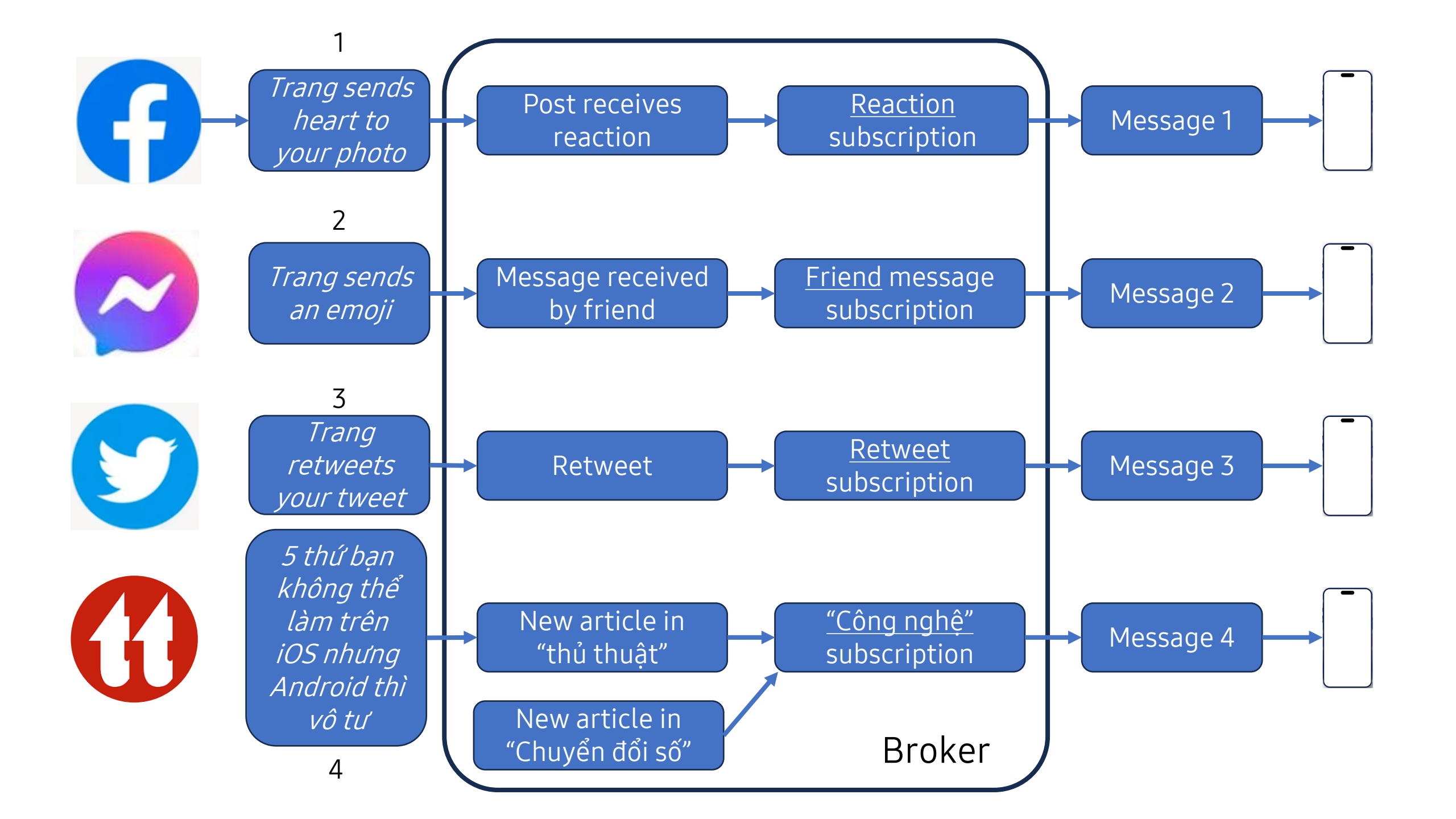





Is there anything new? x 4



3. Publish/subscribe pattern
(Receive message without request)


1
Trang sends heart to your photo
Post receives reaction
Reaction subscription
Message 1
2
Trang sends an emoji
Message received by friend
Friend message subscription
Message 2
3
Trang retweets your tweet
Retweet
Retweet subscription
Message 3
5 thứ bạn không thể làm trên iOS nhưng Android thì vô tư
4
New article in "thủ thuật"
"Công nghệ" subscription
Message 4
New article in "Chuyển đổi số"
Broker

Firebase

Cloud Messaging

- Benefits
  - Reduce power usage
  - Reduce server load
  - Improve scalability
  - Loose coupling
- Drawbacks
  - No delivery assurance
  - Order and uniqueness of messages is not guaranteed
  - Work by assumption as consumer status is blurred out

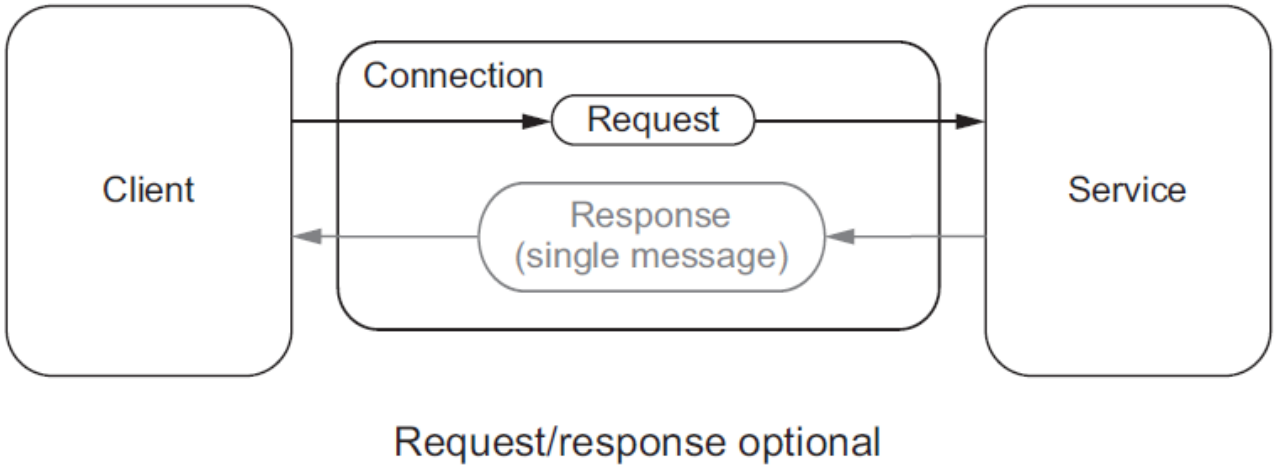
Notes!

- Multiple topics can attach to one subscription.
- Multiple subscriptions can attach to one topic.
- Subscription can have a filter to decide whether message will be sent or not.

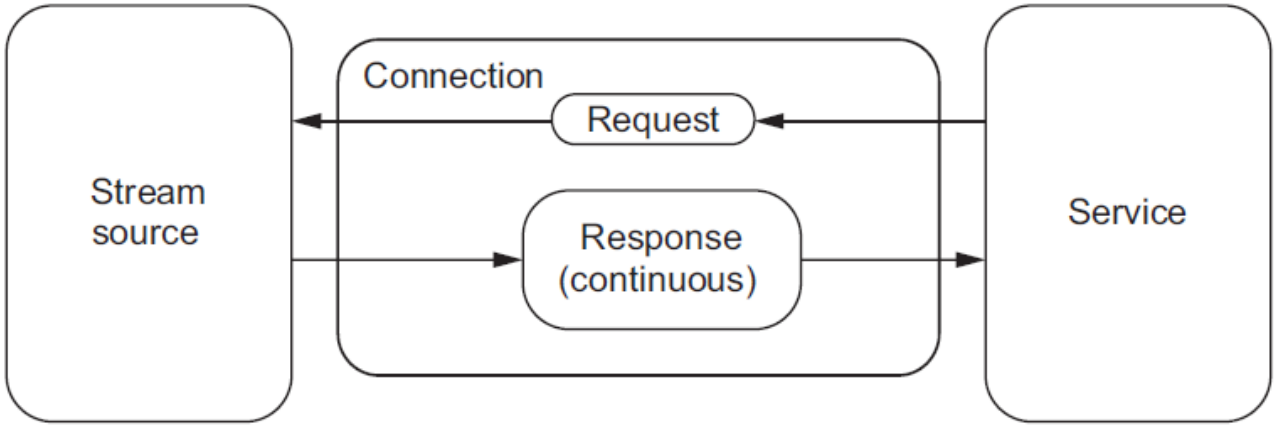
One-way pattern

- Send request, but ignore response.
- Especially useful for IoT applications.
- Also known as “fire and forget”.

Streaming pattern



Request/response optional



Streaming

Scaling the interaction patterns

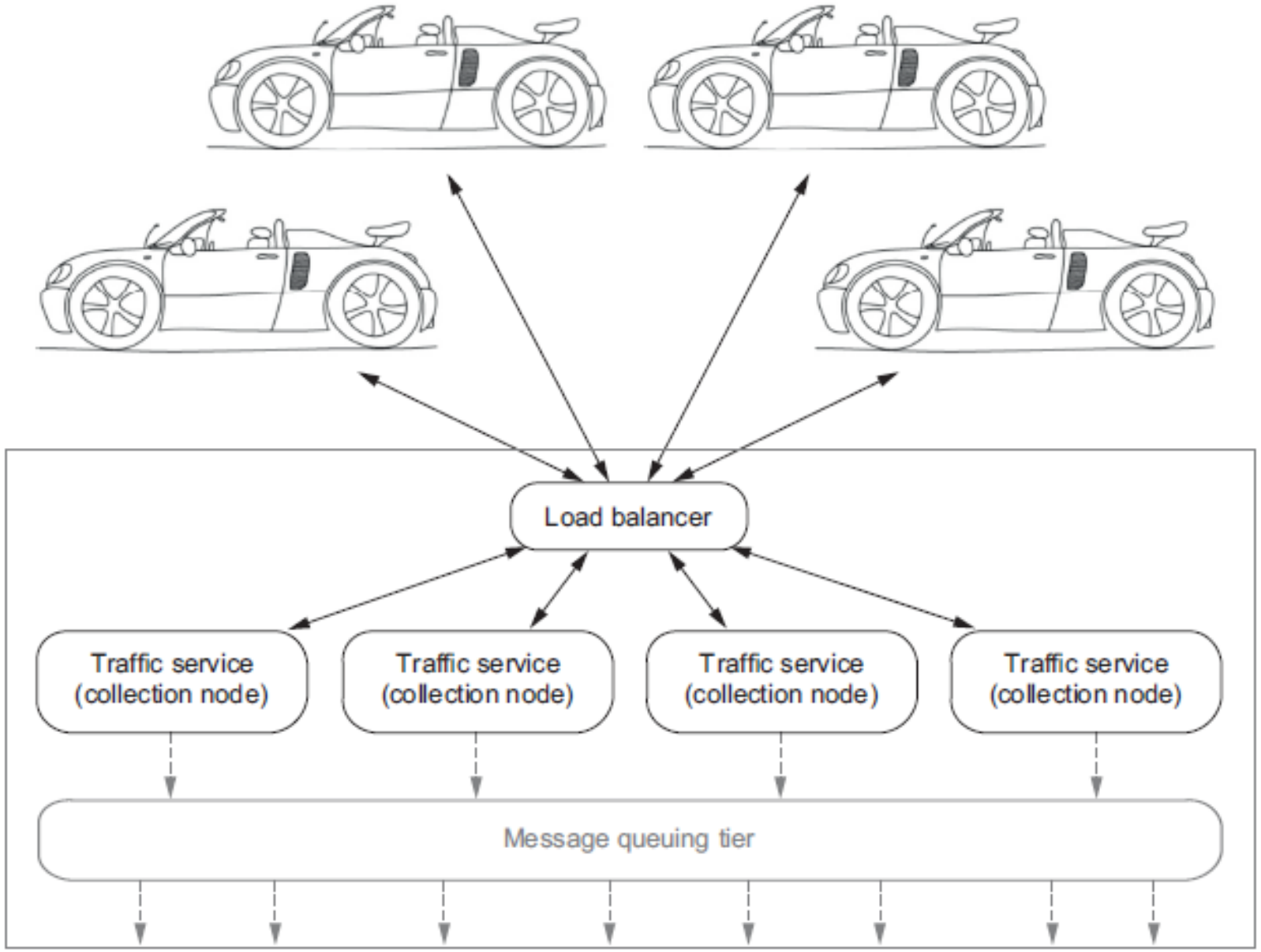
- Regardless of tier, there are 2 ways of scaling a service:
  - **Vertical scaling**: Increase capacity of existing hardware (including virtual) or software.
    - Capacity limitation.
  - **Horizontal scaling**: Add servers.
    - No capacity limitation.
    - Support auto-scaling.
    - **=> The goal for modern-day system design.**

Example 1
Smart vehicles that send/receive traffic data.

A. Request/response

B. Request/acknowledge

C. Publish/subscribe



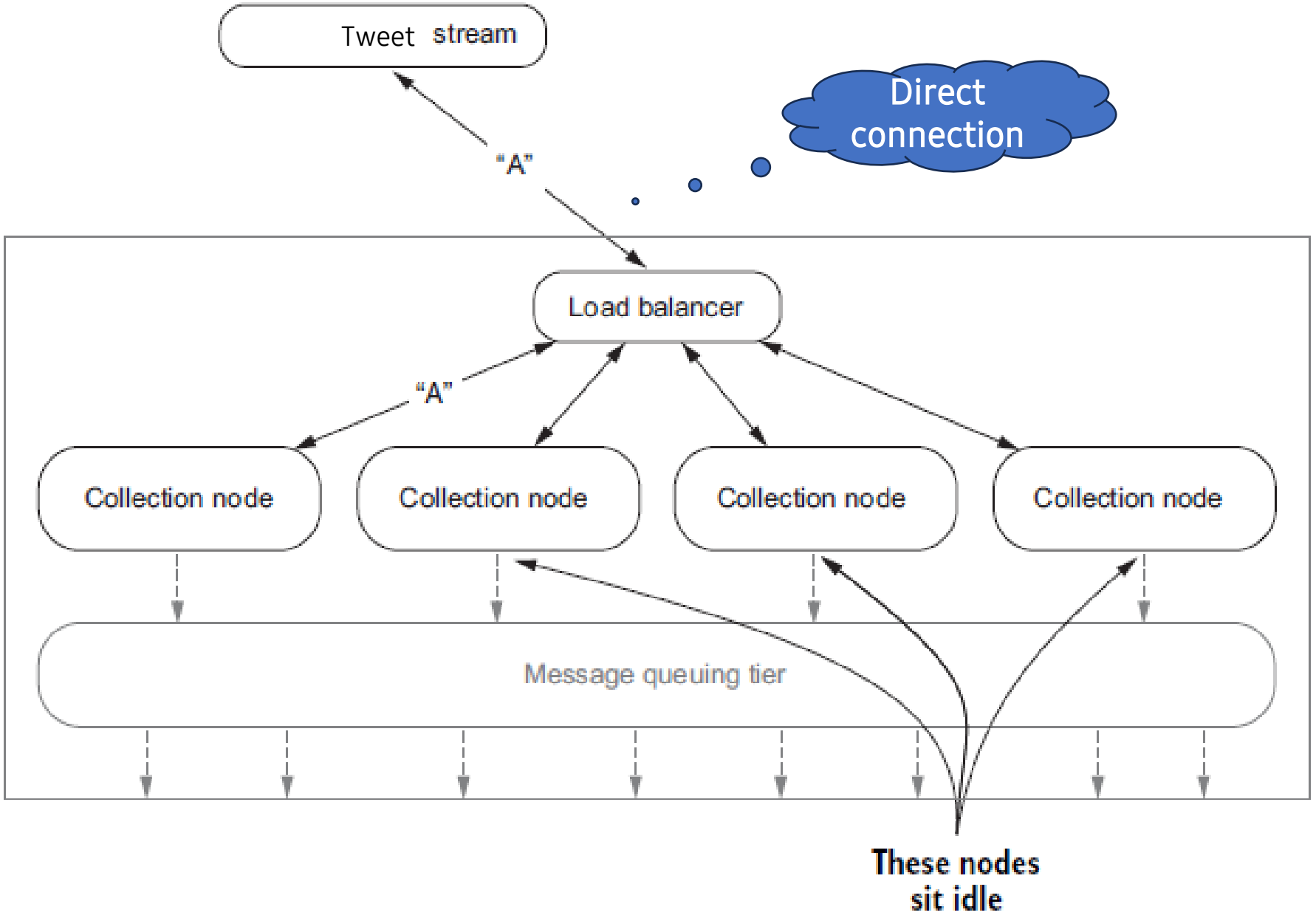
Request/response optional pattern

Example 2
A popularity KPOP award that counts tweets.

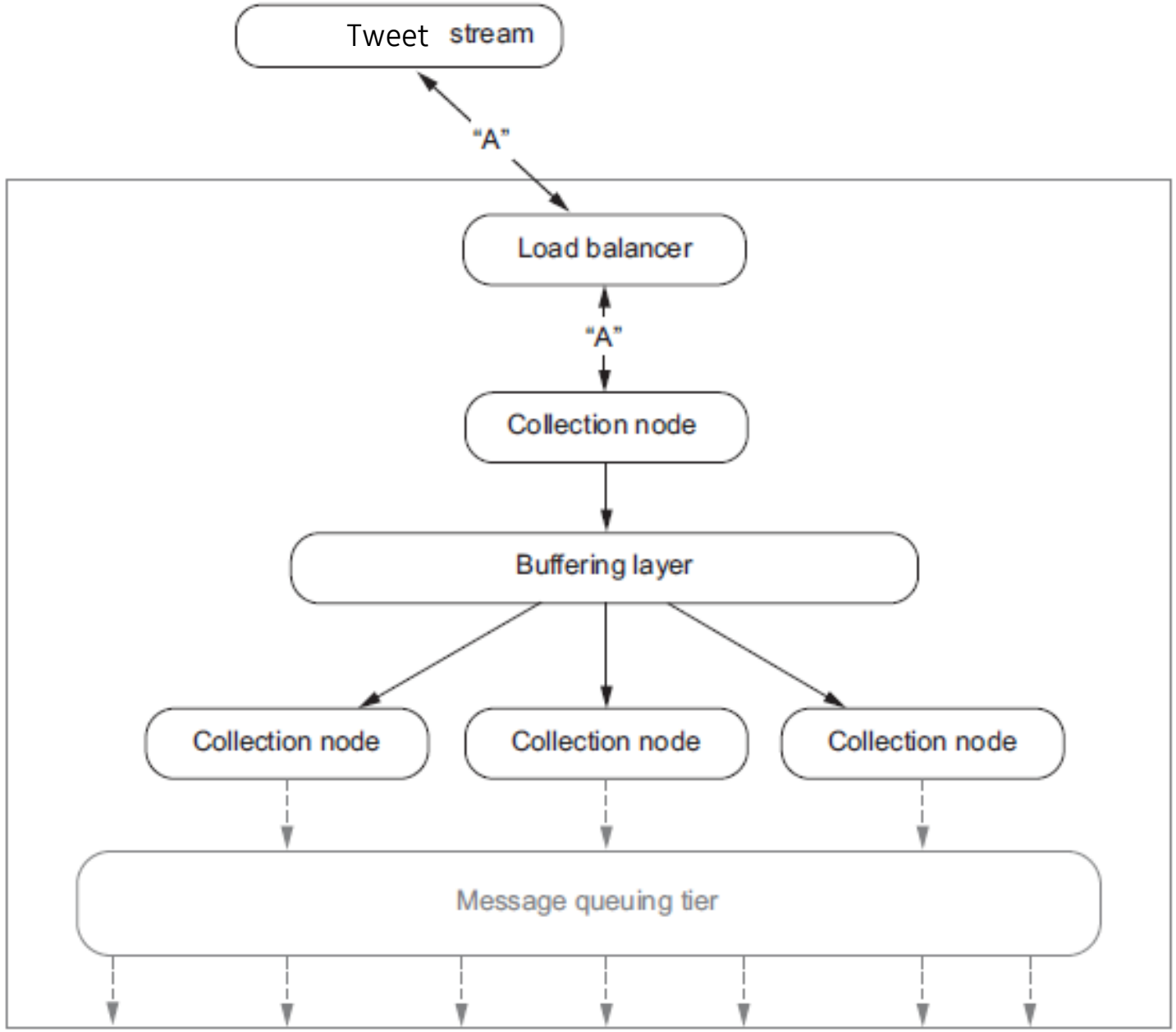
A. Request/response

B. Publish/subscribe

C. Stream


Tweet stream
"A"
Direct connection
Load balancer
"A"
Collection node
Collection node
Collection node
Collection node
Message queuing tier
These nodes sit idle

Collection tier: Collecting tweets and sending messages



Stream pattern with buffering layer

Homework

1. Within one A4 page, Watch this video and answer these questions in Vietnamese [group]:
   - What are the human errors? List at least 3 of them. (6 pts)
   - What is the technical solution to avoid these kinds of tragedy? (4pts)
2. Preparation: Do research about **fault tolerance** [individual].

Tips: Check out section 2.3, page 28, Real-Time Systems: Design Principles for Distributed Embedded Applications