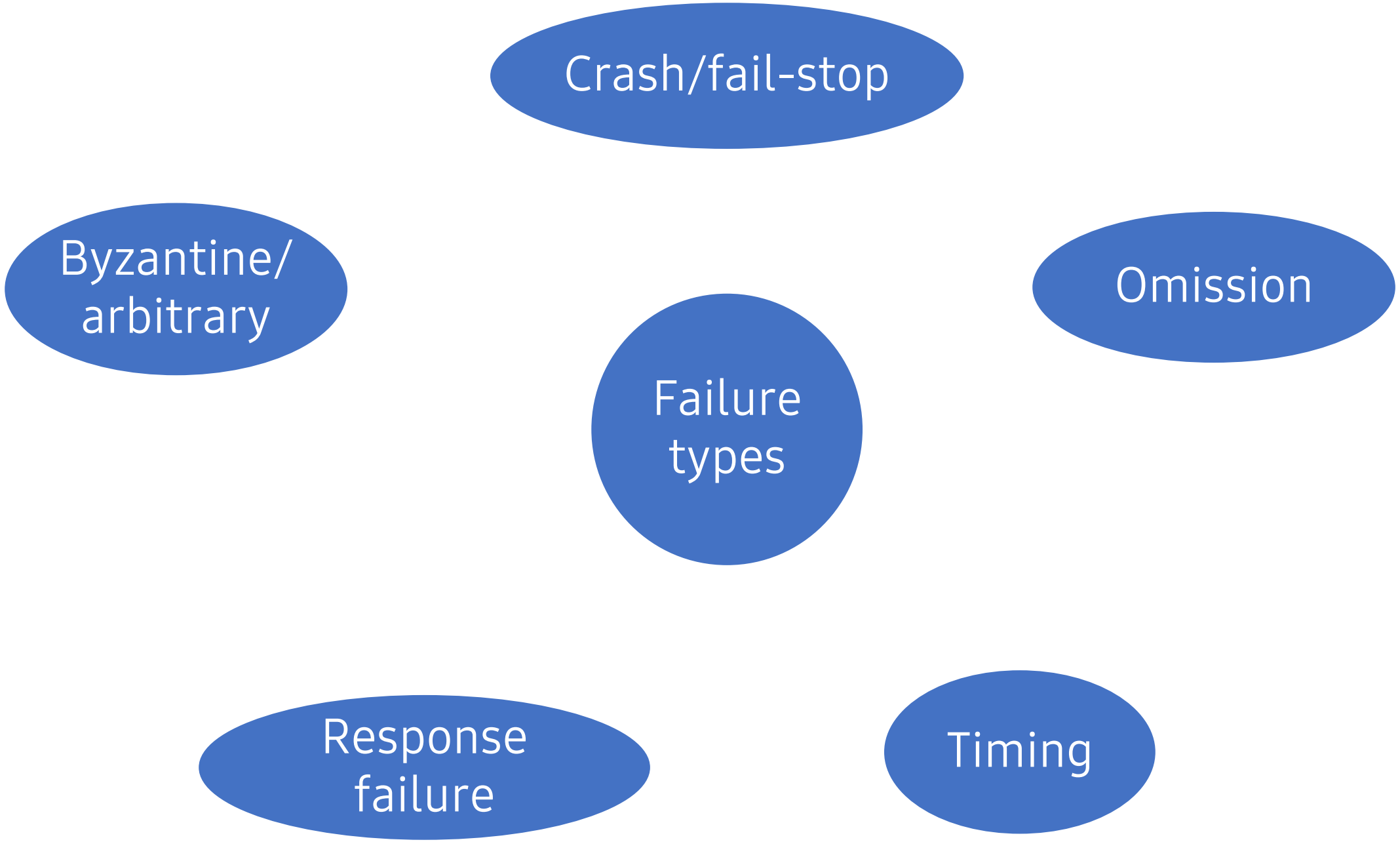
Day 2
Transporting the data from collection tier

Lecturer: M.S. Le Minh Tan

Outlines

I. Fault tolerance for collection tier
II. Message queuing tier: Core concepts
III. Security, fault tolerance and Business scenarios


Crash/fail-stop
Byzantine/
arbitrary
Omission
Failure
types
Response
failure
Timing

I. Fault tolerance for collection tier

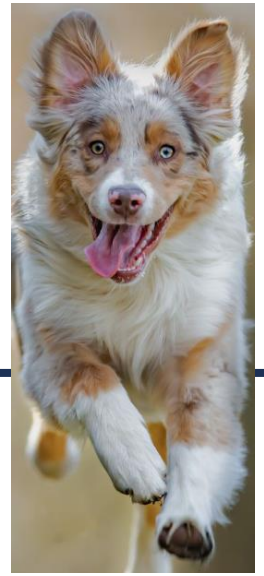
- One or more of our collection nodes may fail.
- The message may not be reproducible.
- Approaches: Checkpointing and logging.

1. Checkpointing

- Purpose: Save system state for recovery.
- Characteristics:
  - Global: Save system state **as a whole**.
  - Potential for data loss: By recovering up to the most recent state, messages that were processed and generated afterward are loss.

	Concurrent	Incremental
Features	write to page -> storing checkpoint -> resume writing.	Save only modified pages (**deduplication**).
👍	• High compability with existing infrastructure. • Version isolation.	• Less storage usage. • Less time-consumption.
👎	• Very high storage usage. • Slow checkpoint/restoration.	• Hash collision. • Use less common file systems. • Rely on previous snapshot.

1



M1

M1

M1

Con mèo

2

M2

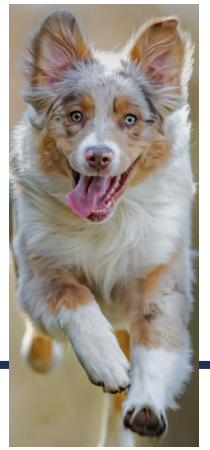
M2

M2

?

2

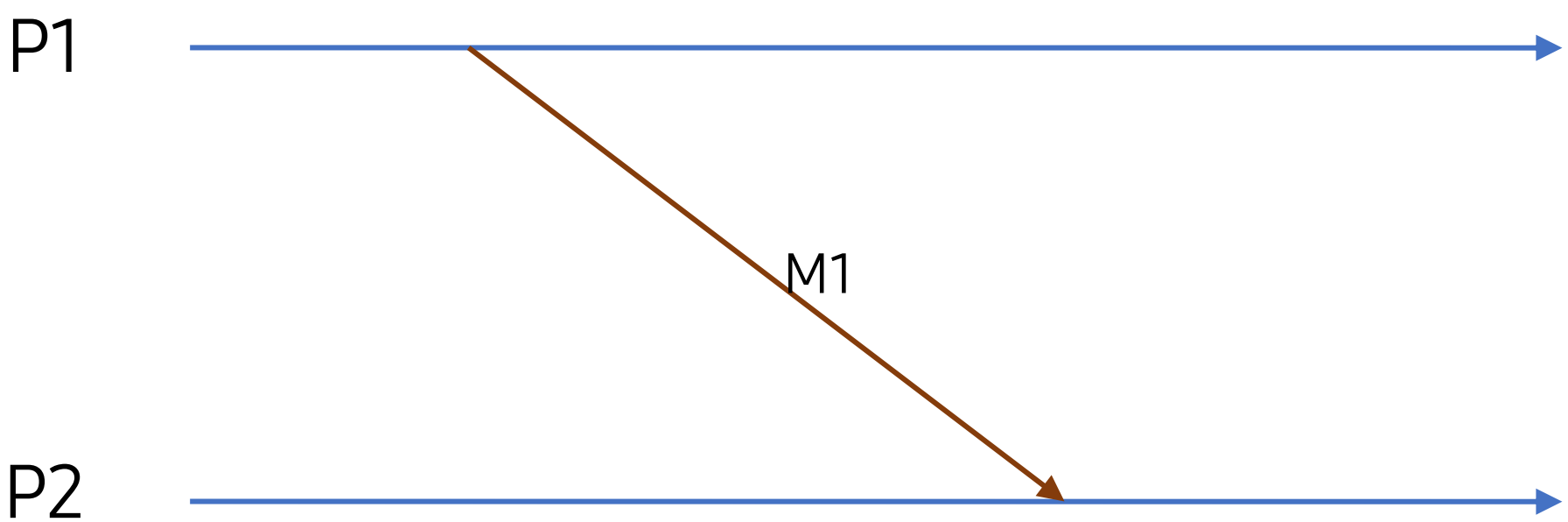
We'll start pic 2 from the beginning



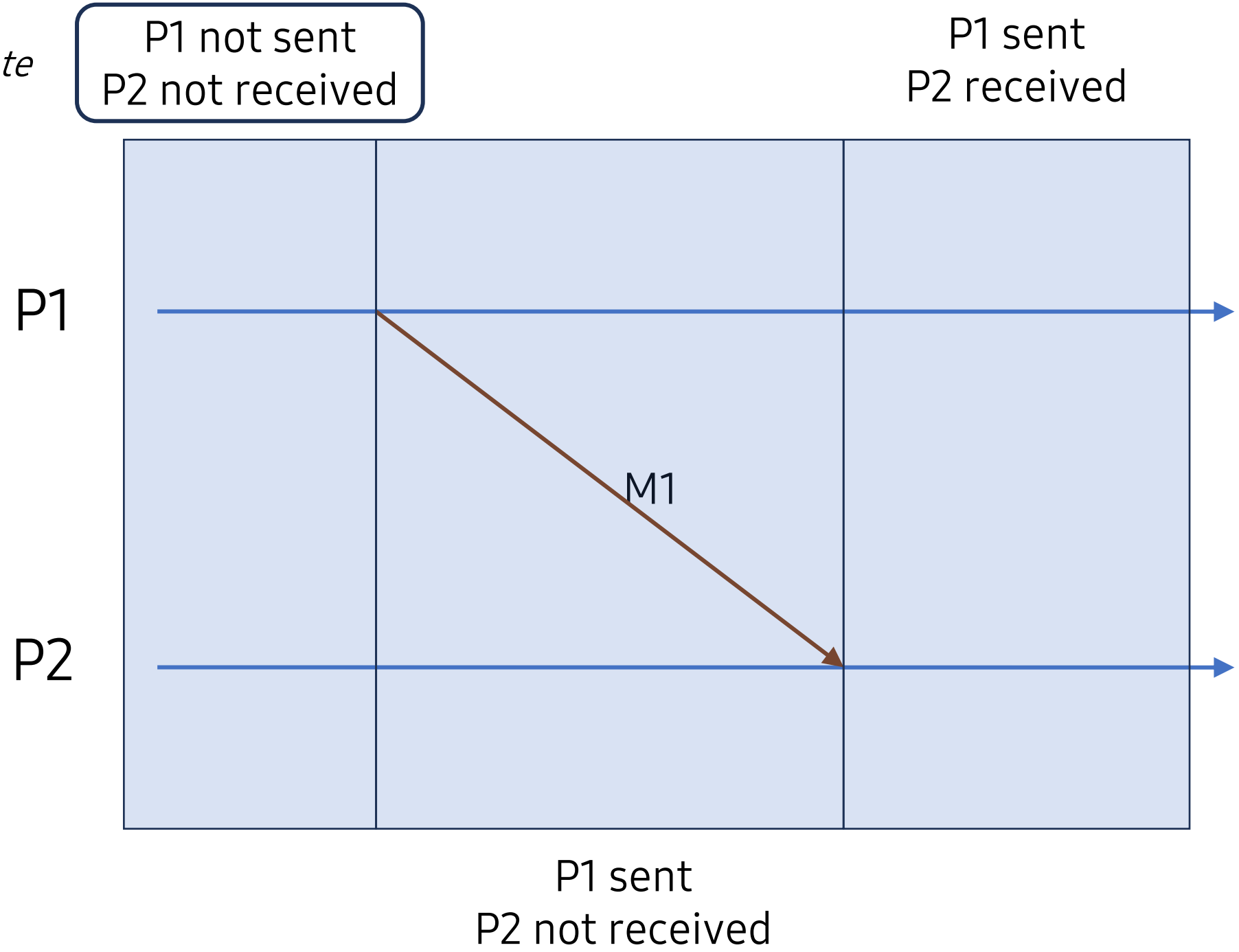
Return to pic 1…

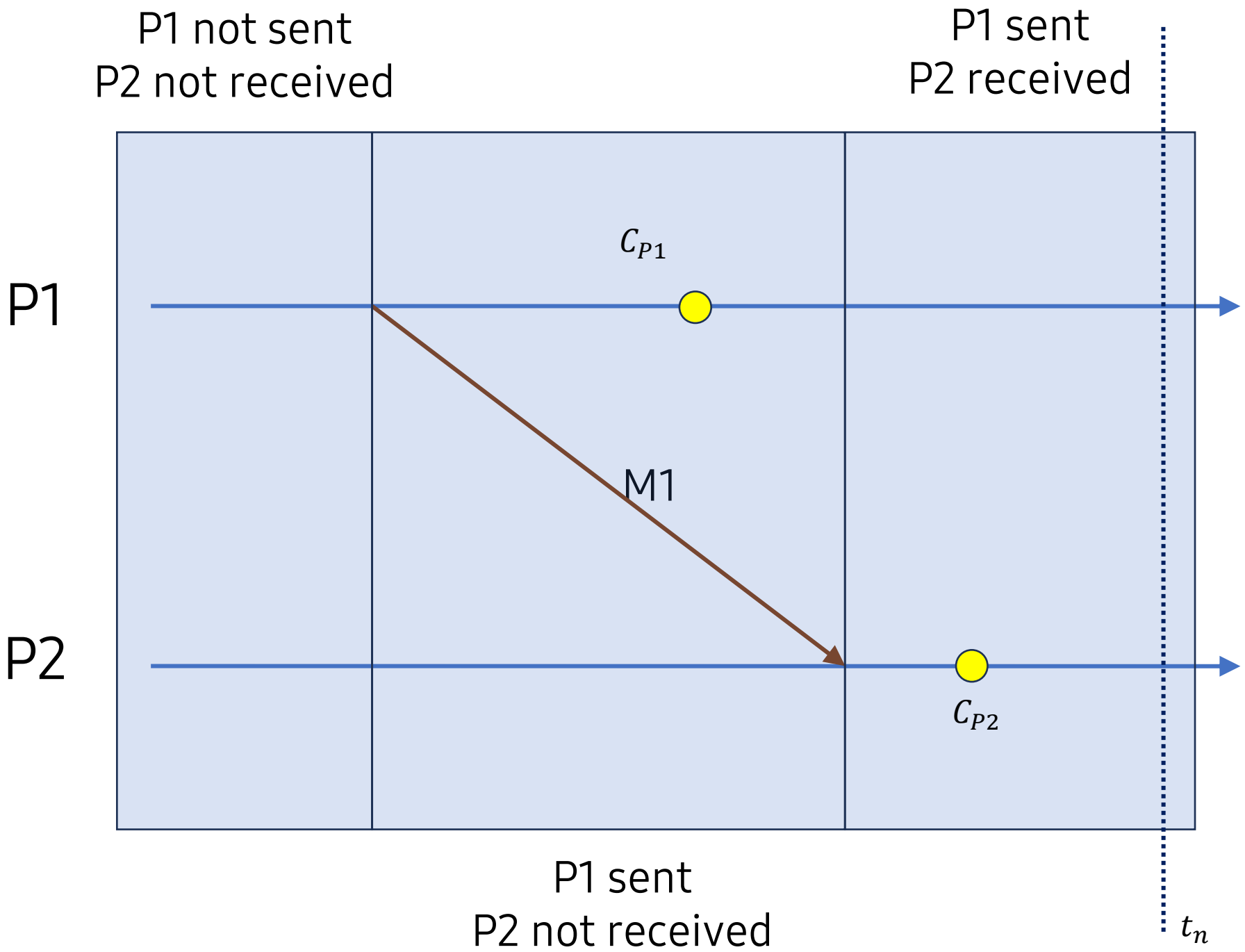
Let's try again

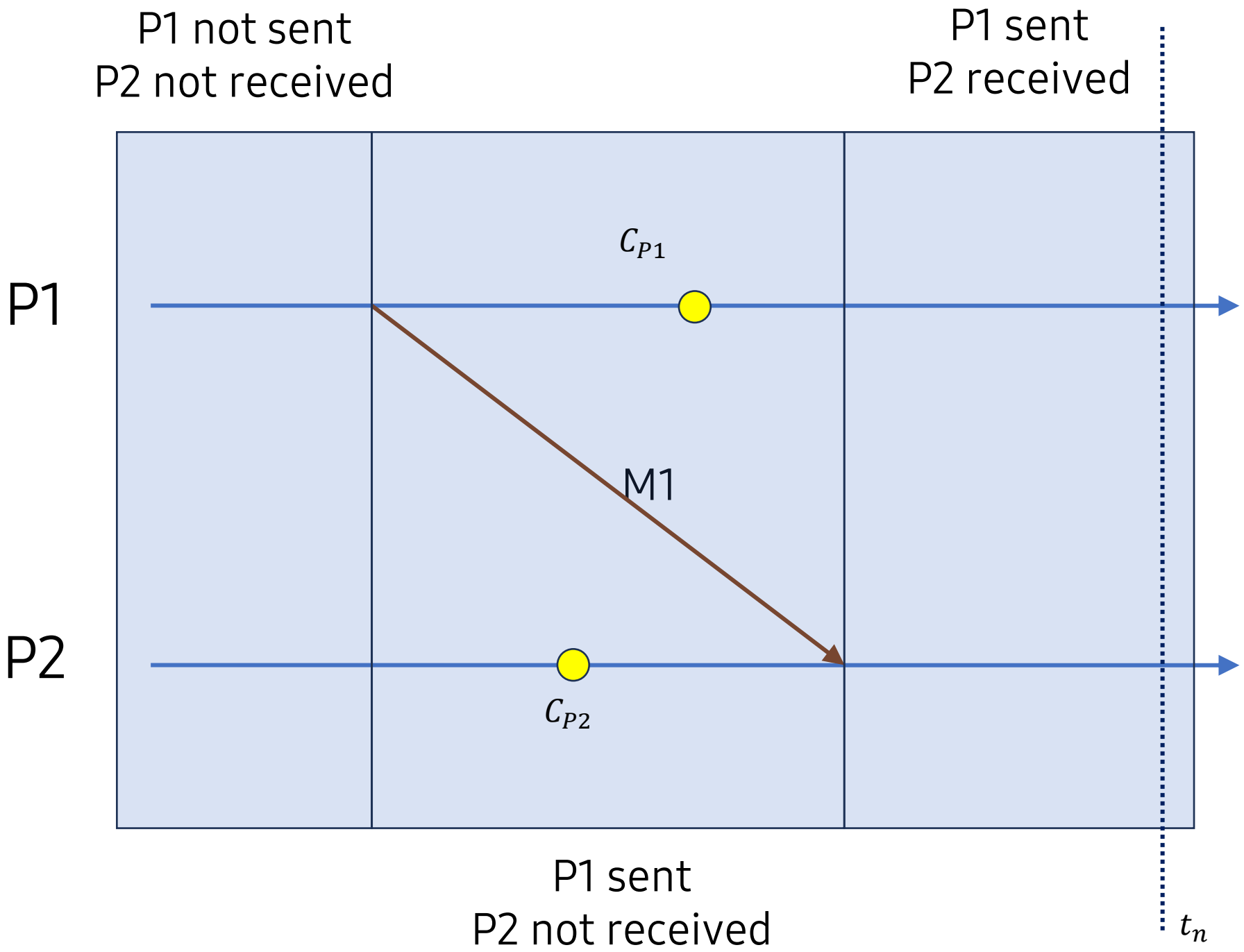
Try pic 1 again

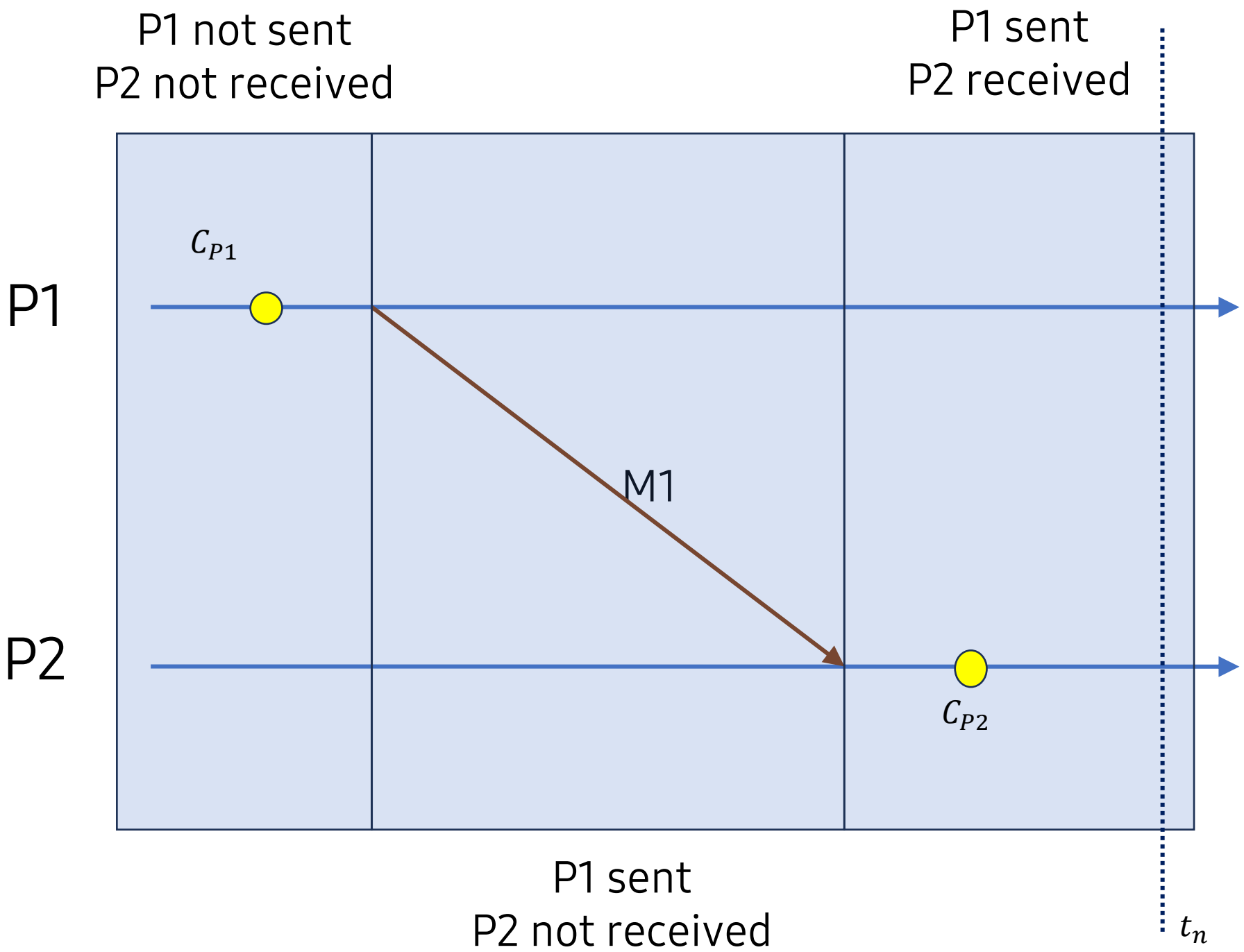

P1
M1
P2

Global state



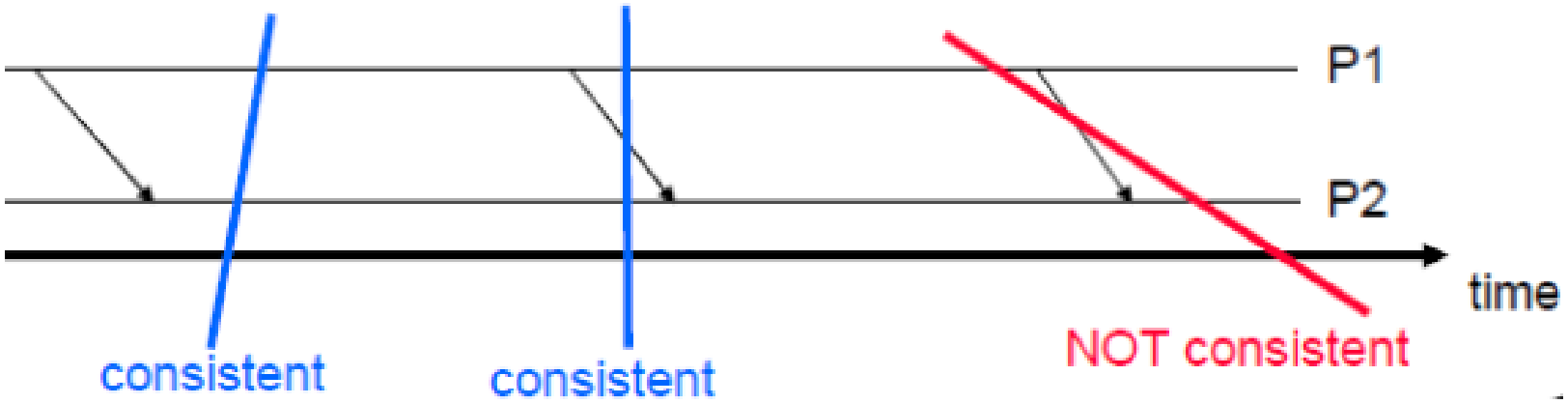

P1 not sent
P2 not received
P1 sent
P2 received
P1
P2
C_{P1}
M1
C_{P2}
P1 sent
P2 not received
t_n


P1 not sent
P2 not received
P1 sent
P2 received
P1
P2
C_{P1}
M1
C_{P2}
P1 sent
P2 not received
t_n

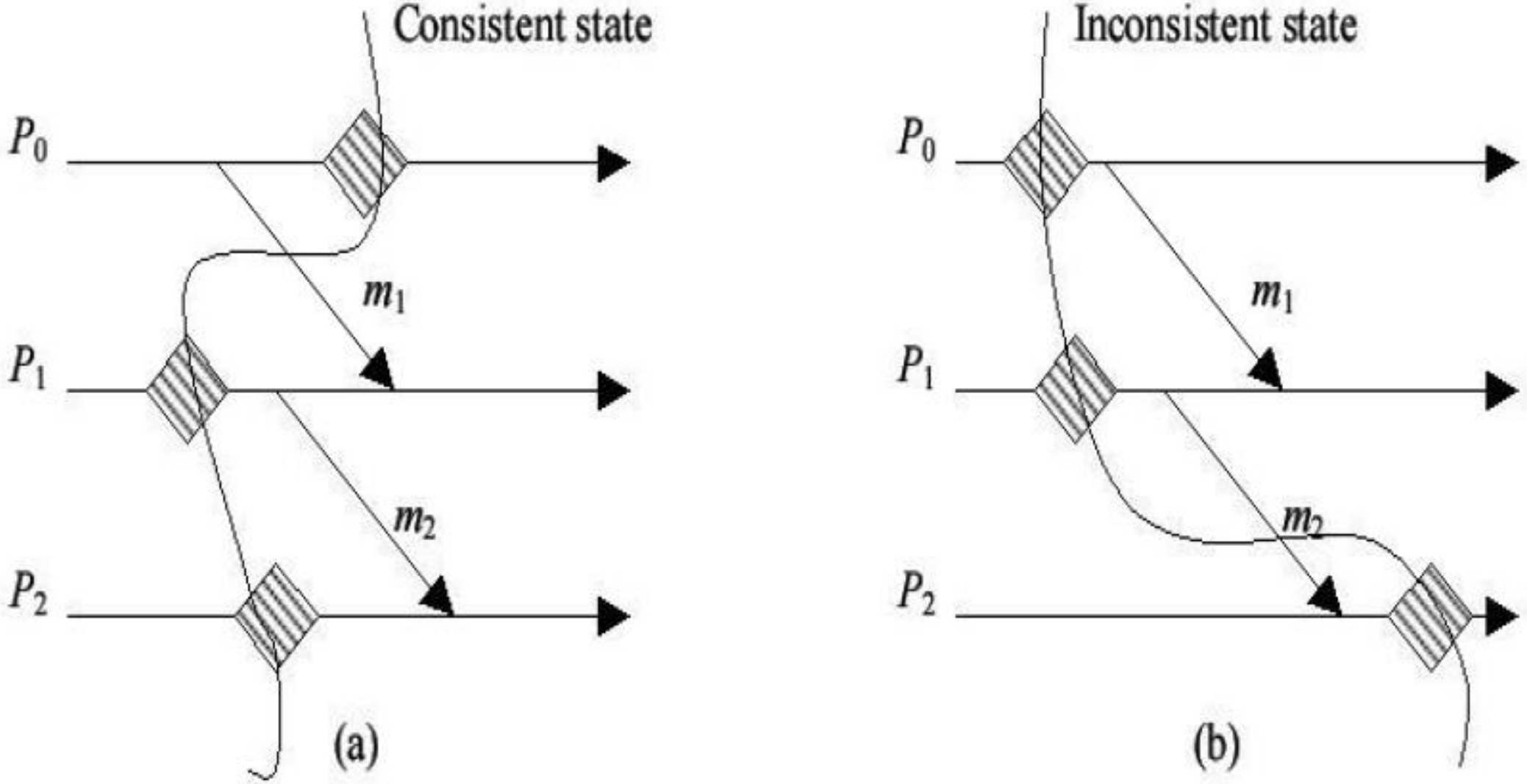

P1 not sent
P2 not received
P1 sent
P2 received
C_{P1}
P1
M1
P2
C_{P2}
P1 sent
P2 not received
t_n

Consistency

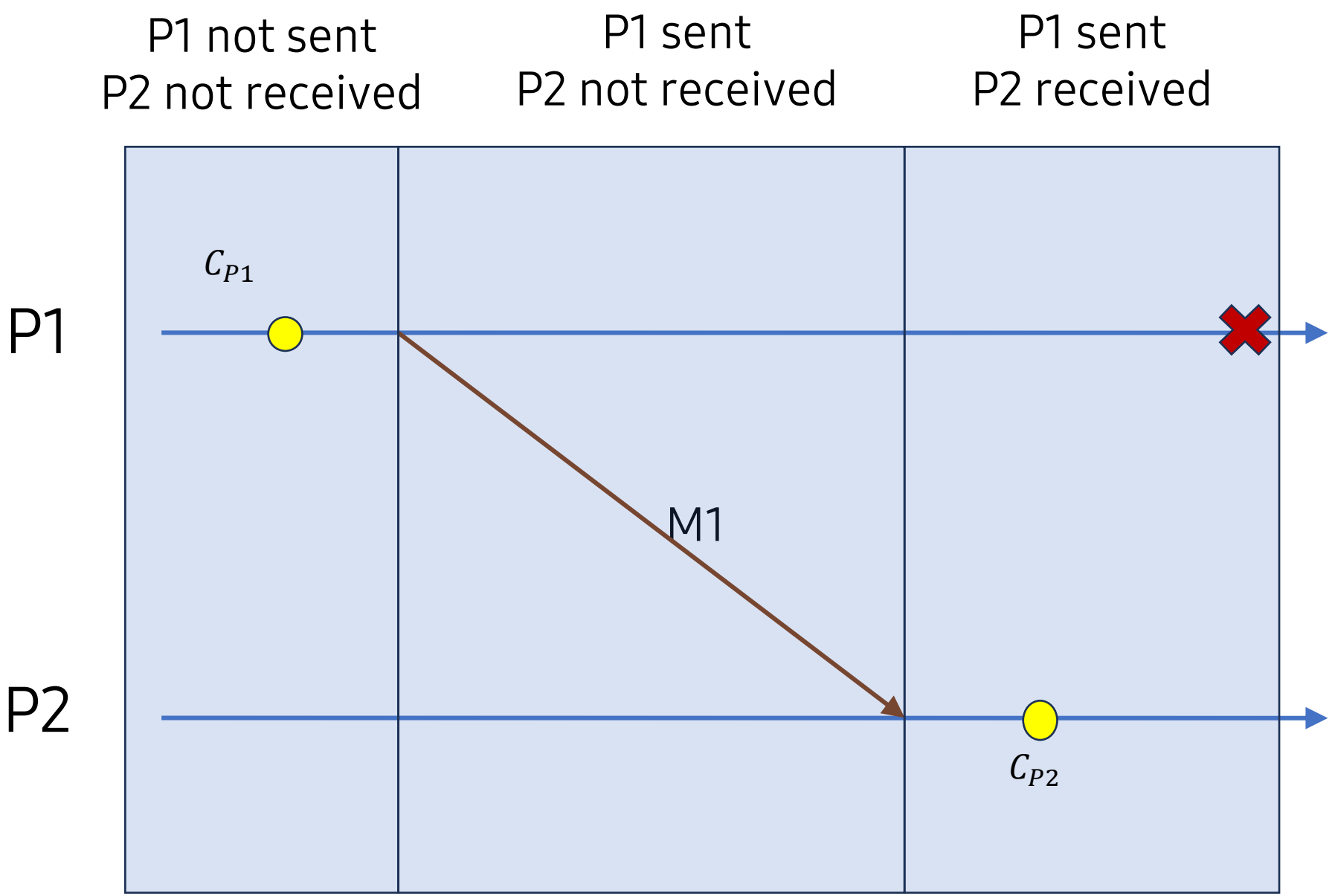
- A checkpoint is considered inconsistent if the receiver took the message, but the sender had not sent it yet.
- Consistency is defined by possibility.
- Global checkpoint is better than global snapshot, but requires:
  - Replayable message.
  - A global consistent state solution is found.



Do not memorize this!
Learn how to determine consistency

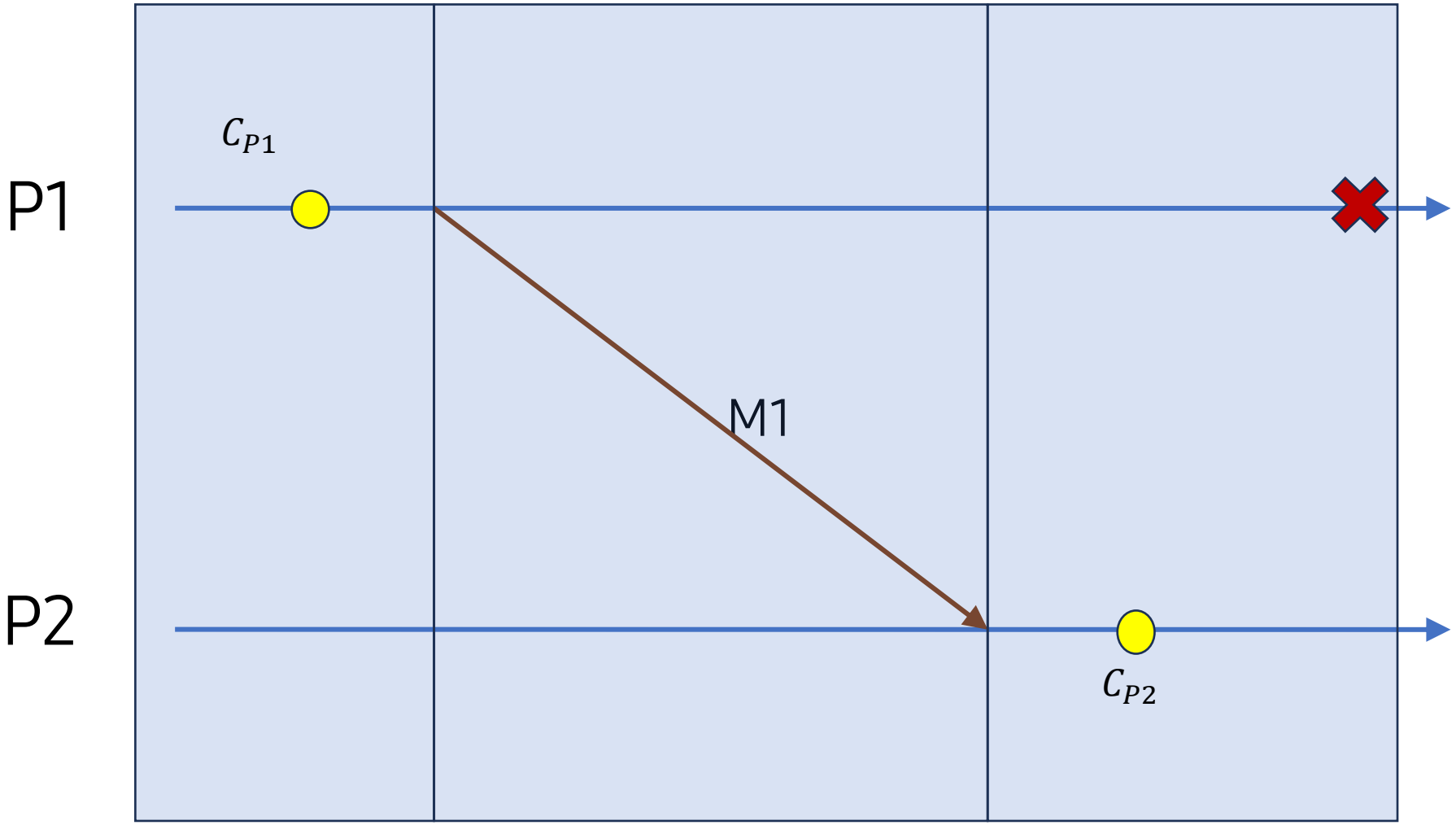


Inconsistency problem

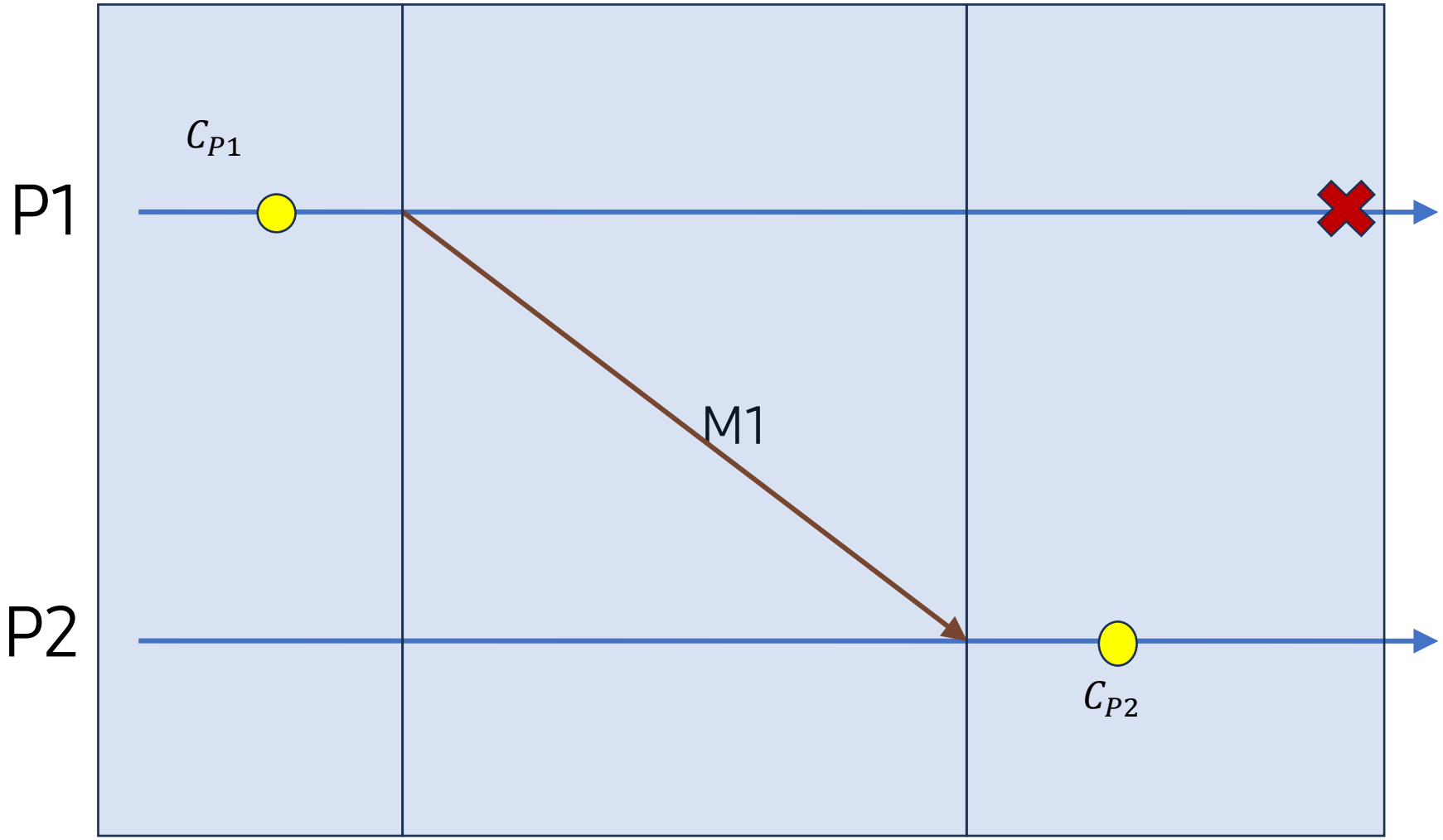


Is consistent?

Step 1: Split the timeline into multiple segments by send/receive events.

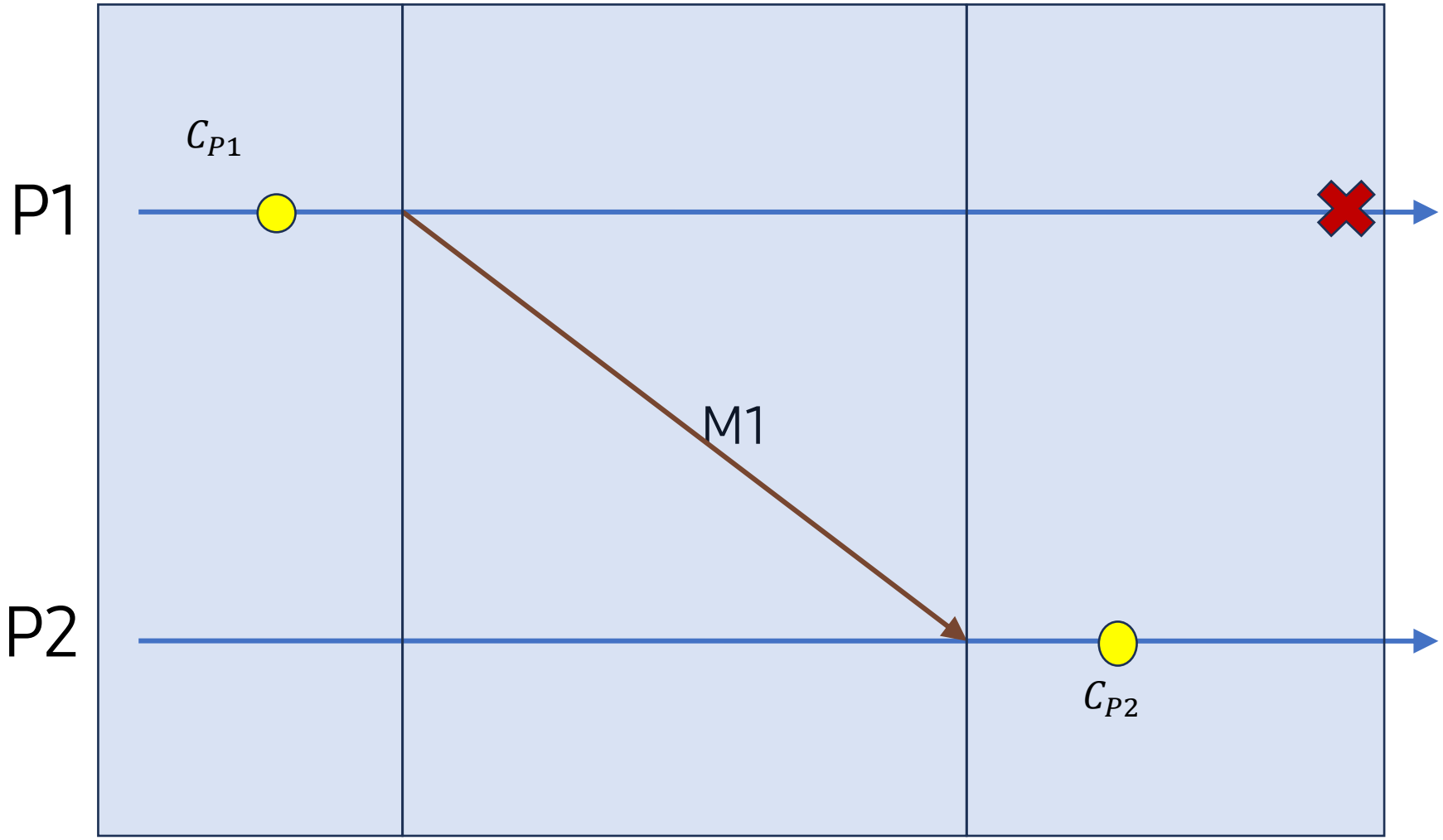


Step 2: Build a state table.

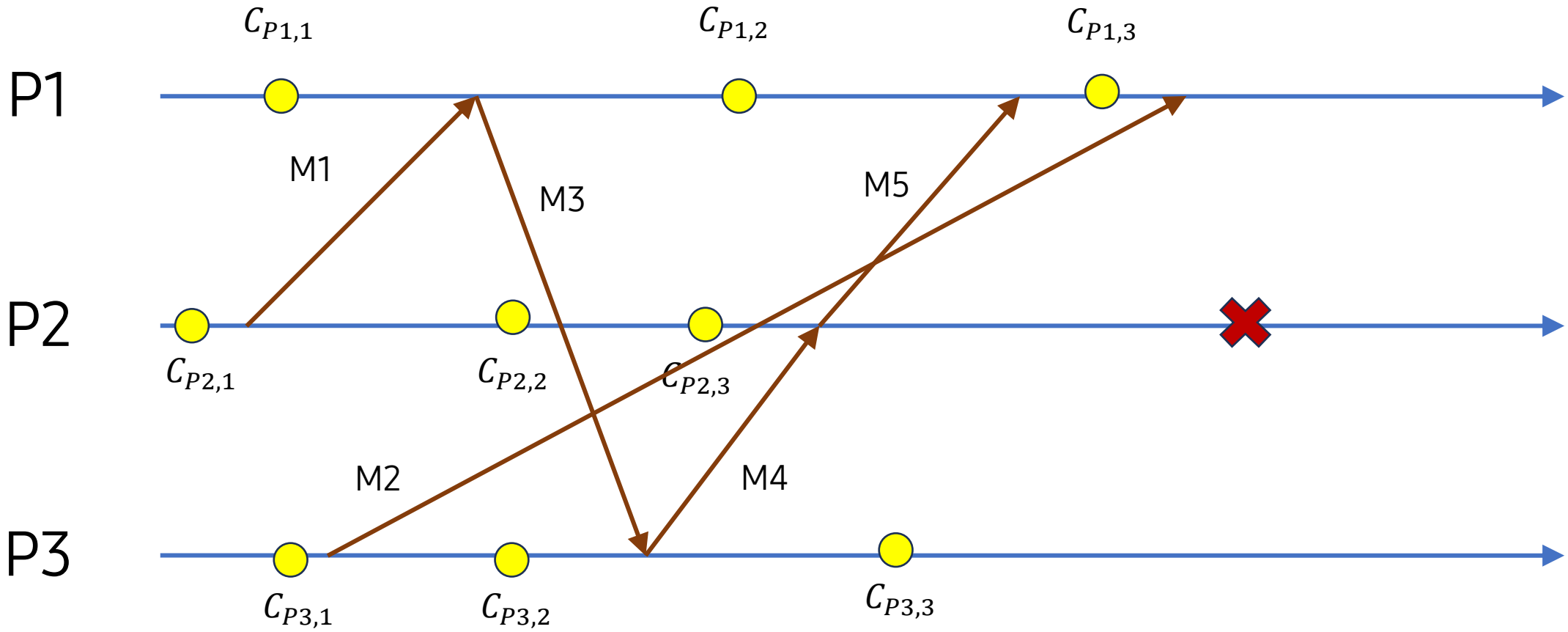


Seg	P1	P2	Glob
1			
2	s.M1		s.M1
3	s.M1	r.M1	s.M1 r.M1

Step 3: Check consistency of checkpoints.



Seg	P1	P2	Glob
1			
2	s.M1		s.M1
3	s.M1	r.M1	s.M1 r.M1



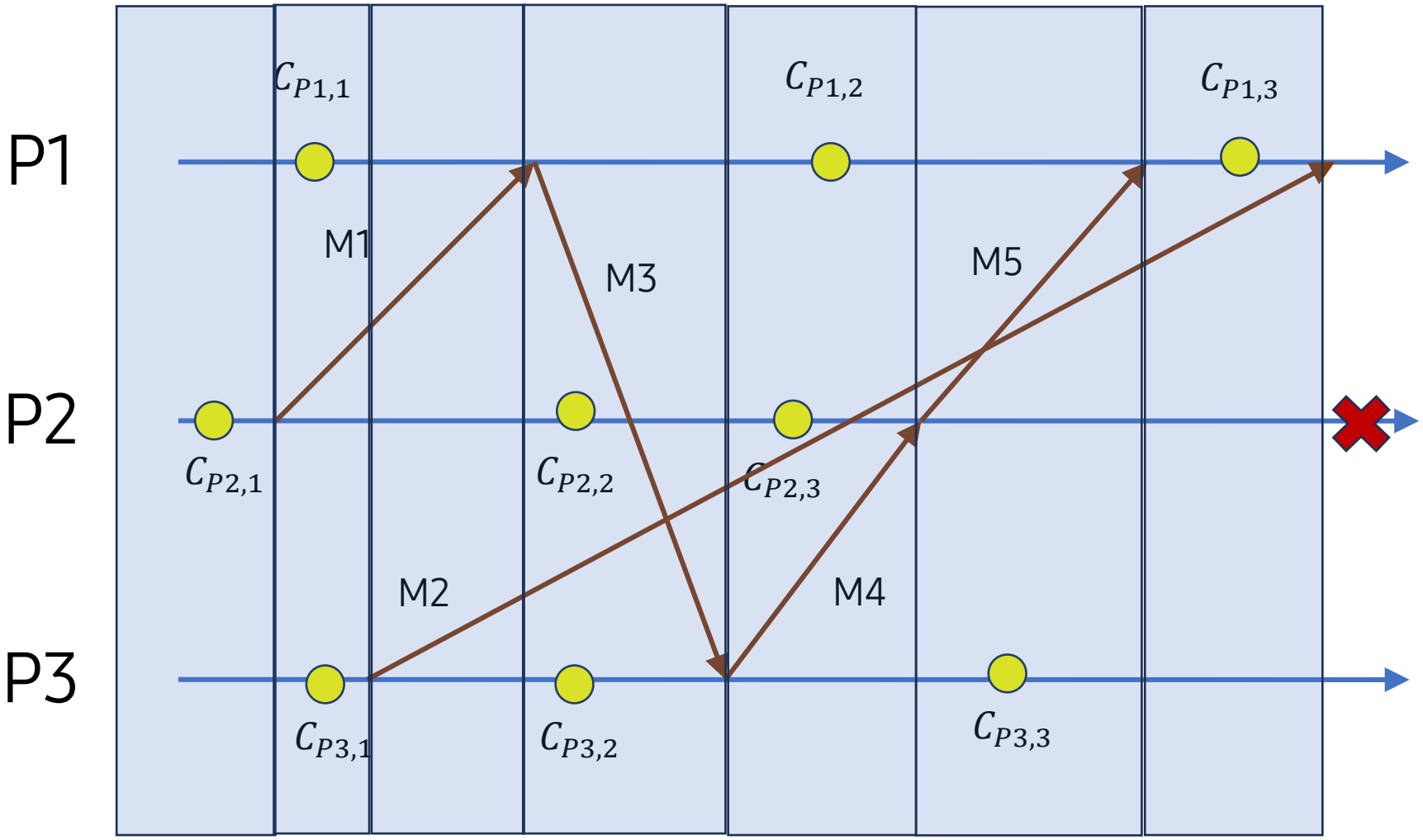
Are the following states consistent or inconsistent?

A. $\{C_{P1,1}, C_{P2,3}, C_{P3,2}\}$

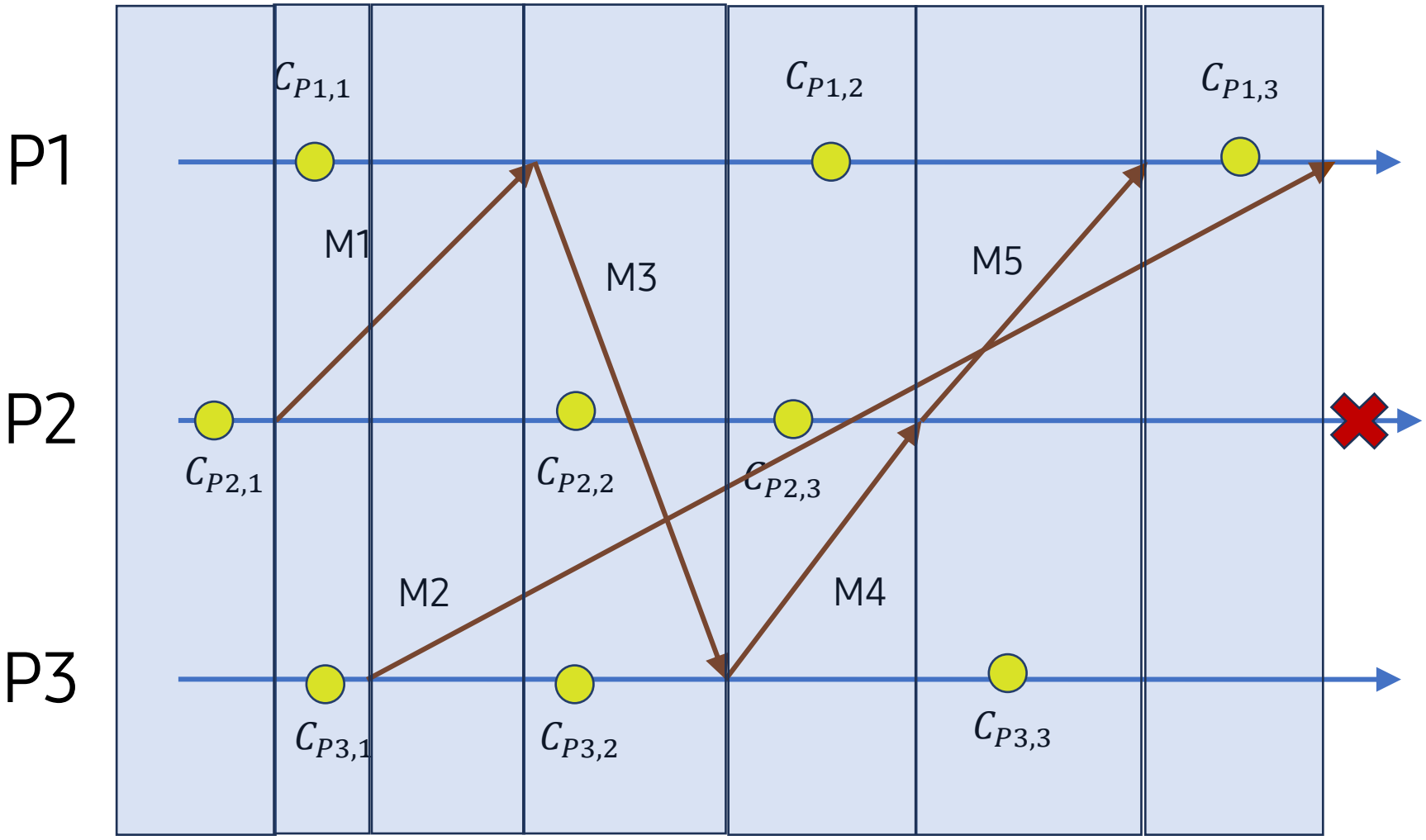
B. $\{C_{P1,2}, C_{P2,3}, C_{P3,3}\}$

C. $\{C_{P1,3}, C_{P2,1}, C_{3,1}\}$

D. $\{C_{P1,1}, C_{P2,2}, C_{P3,3}\}$

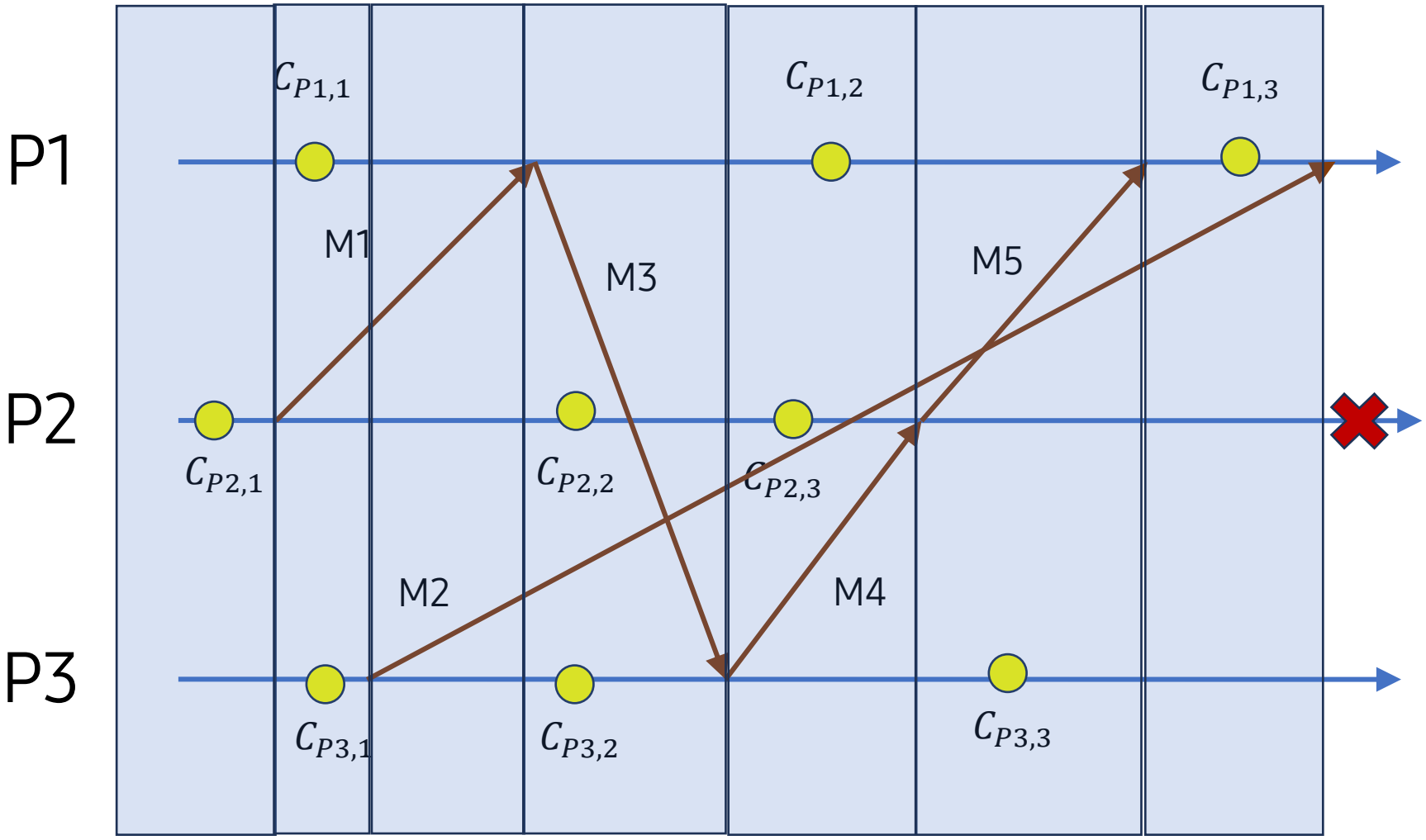


Seg	P1	P2	P3	Glob
1				
2				
3				
4				
5				
6				
7				



$C_{P1,1}, C_{P2,3}, C_{P3,2}$: 2, 5, 4
$C_{P1,2}, C_{P2,3}, C_{P3,3}$: 5, 5, 6
$C_{P1,3}, C_{P2,1}, C_{3,1}$: 7, 1, 2
$C_{P1,1}, C_{P2,2}, C_{P3,3}$: 2, 4, 6

Seg	P1	P2	P3	Glob
1				
2		s.M1		s.M1
3		s.M1	s.M2	s.M1 s.M2
4	r.M1 s.M3	s.M1	s.M2	sr.M1 s.M2 s.M3
5	r.M1 s.M3	s.M1	s.M2 r.M3 s.M4	sr.M1 s.M2 sr.M3 s.M4
6	r.M1 s.M3	s.M1 r.M4 s.M5	s.M2 r.M3 s.M4	sr.M1 s.M2 sr.M3 sr.M4 s.M5
7	r.M1 s.M3 r.M5	s.M1 r.M4 s.M5	s.M2 r.M3 s.M4	sr.M1 s.M2 sr.M3 sr.M4 sr.M5



$C_{P1,1}, C_{P2,3}, C_{P3,2}$: 2, 5, 4 ✓
$C_{P1,2}, C_{P2,3}, C_{P3,3}$: 5, 5, 6 ✓
$C_{P1,3}, C_{P2,1}, C_{3,1}$: 7, 1, 2 ✗
$C_{P1,1}, C_{P2,2}, C_{P3,3}$: 2, 4, 6 ✗

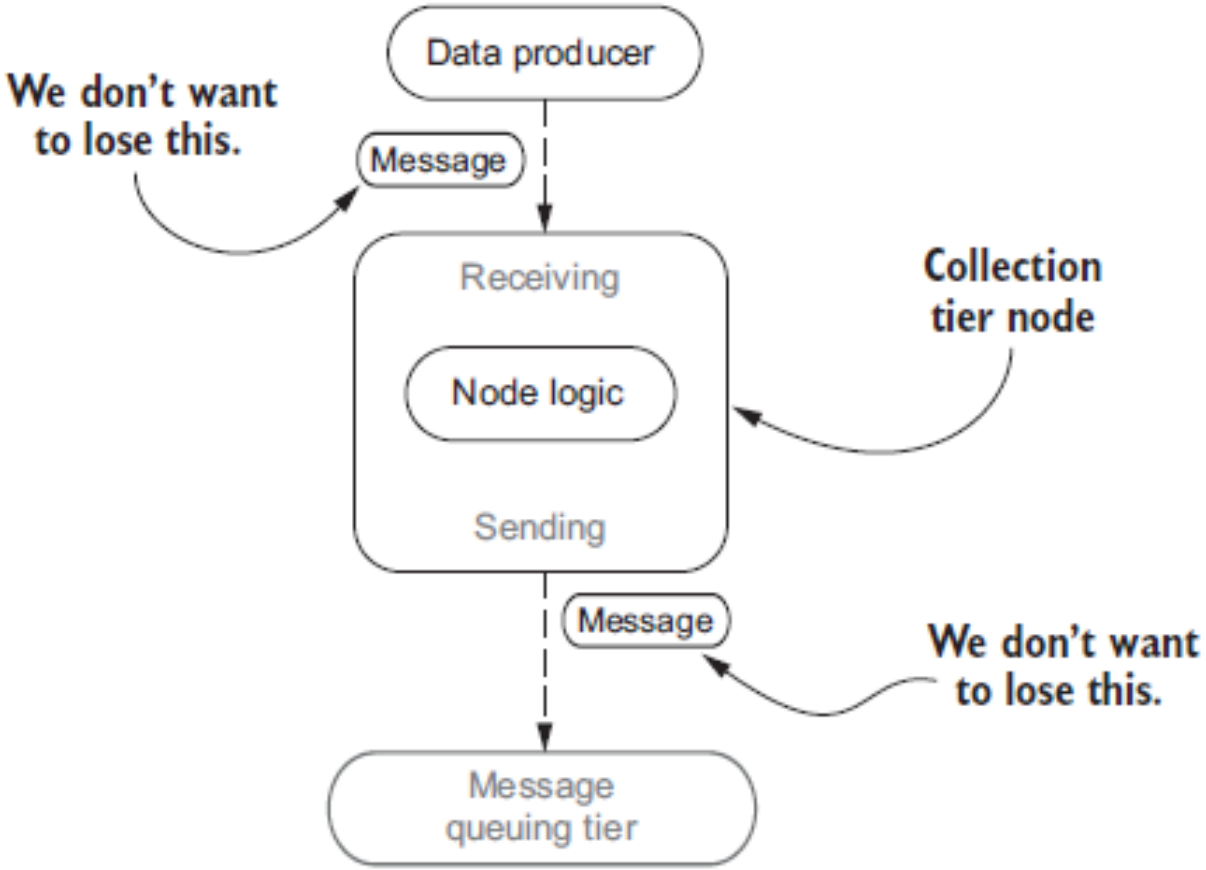
Seg	P1	P2	P3	Glob
1				
2		s.M1		s.M1
3		s.M1	s.M2	s.M1 s.M2
4	r.M1 s.M3	s.M1	s.M2	sr.M1 s.M2 s.M3
5	r.M1 s.M3	s.M1	s.M2 r.M3 s.M4	sr.M1 s.M2 sr.M3 s.M4
6	r.M1 s.M3	s.M1 r.M4 s.M5	s.M2 r.M3 s.M4	sr.M1 s.M2 sr.M3 sr.M4 s.M5
7	r.M1 s.M3 r.M5	s.M1 r.M4 s.M5	s.M2 r.M3 s.M4	sr.M1 s.M2 sr.M3 sr.M4 sr.M5

More notes

- It doesn’t need a total failure to restore global state.
- Each process may have different number of checkpoints at different milestone.

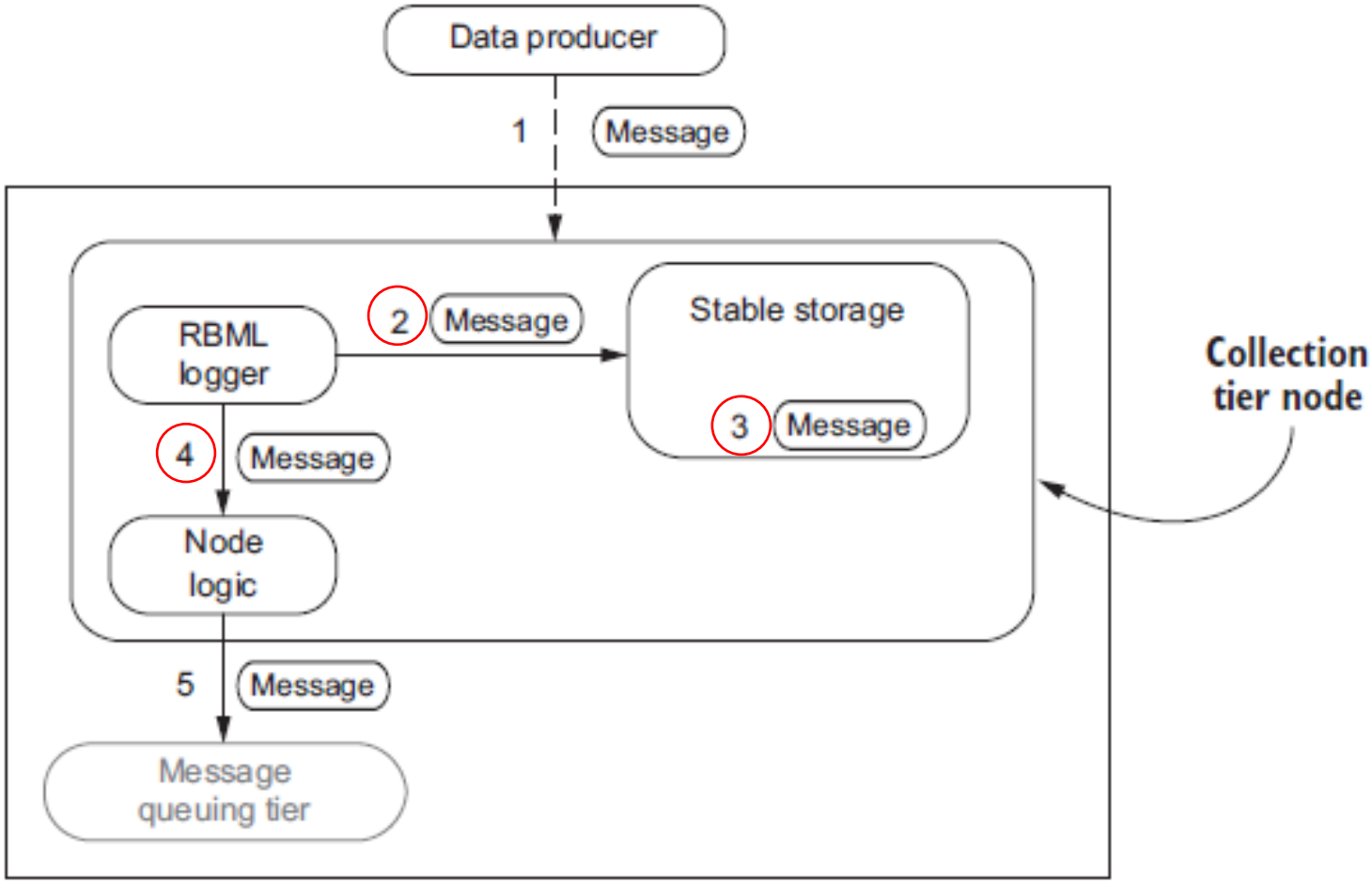
2. Logging

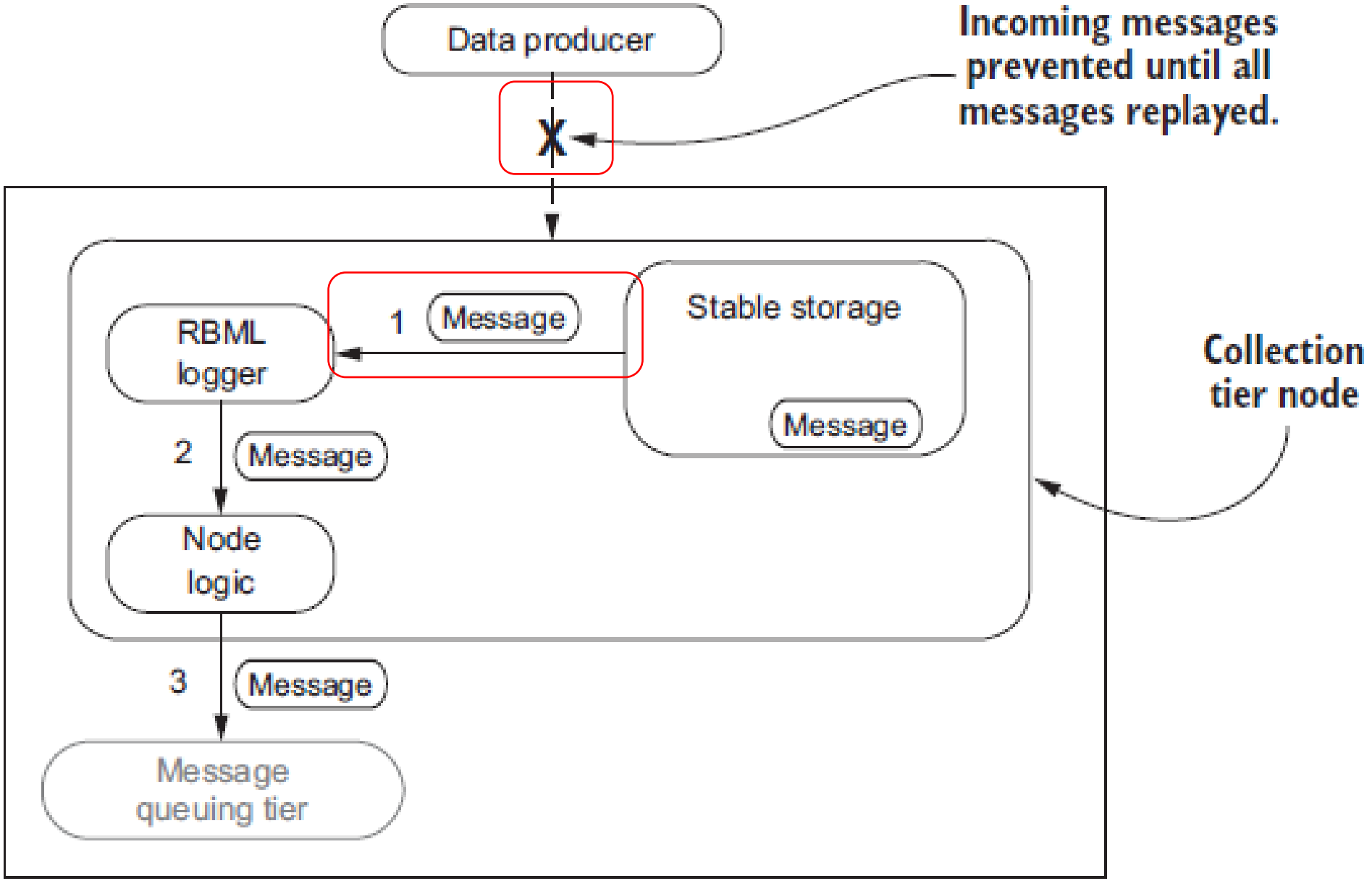
- Each tier in the system independently records all messages it receives and plays them back (**replays**) when needed.
- Techniques: RBML, SBML & HML.



Receiver-based message logging (RBML)

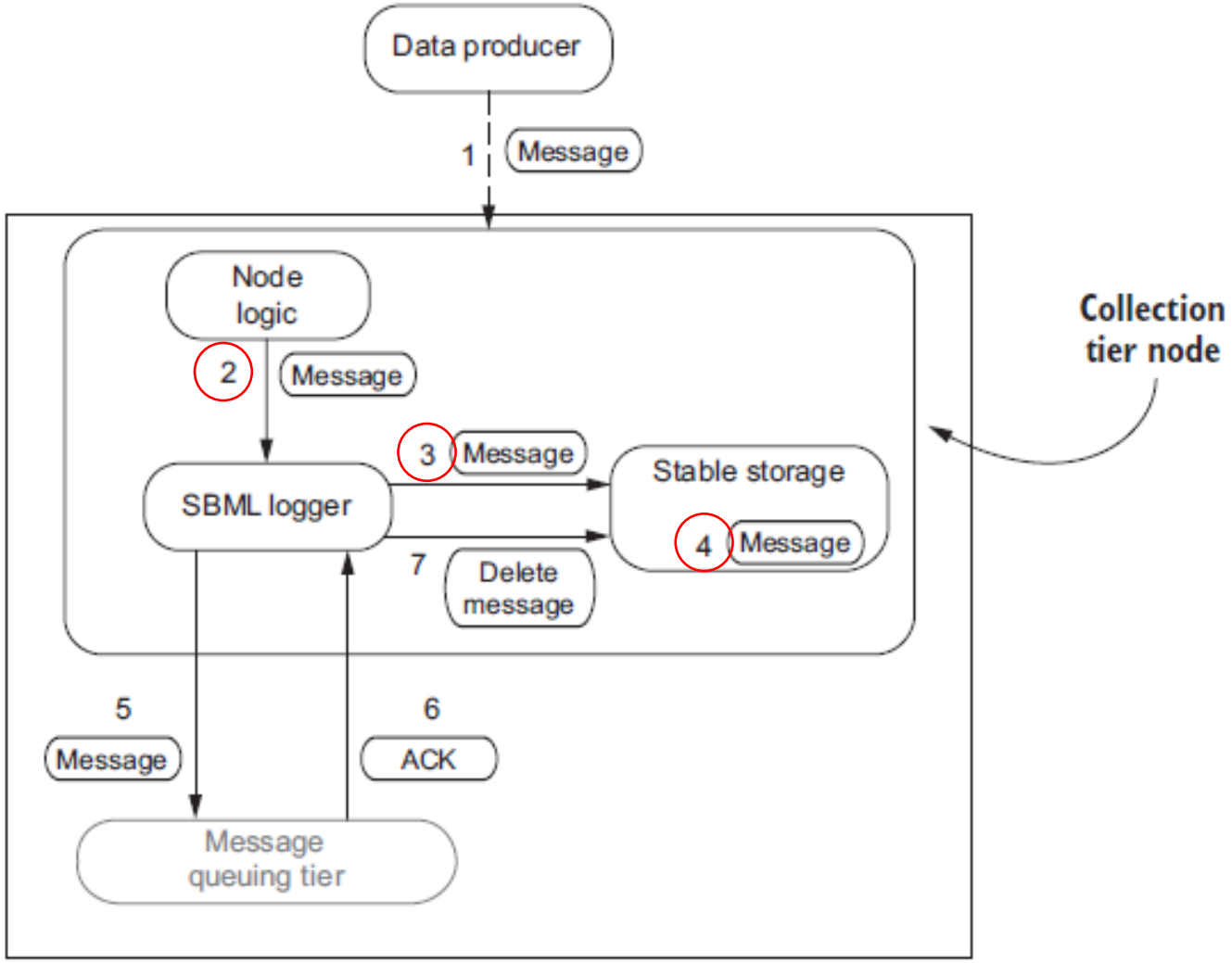
- Method: Writing received message to stable storage **before any action is taken on it.**

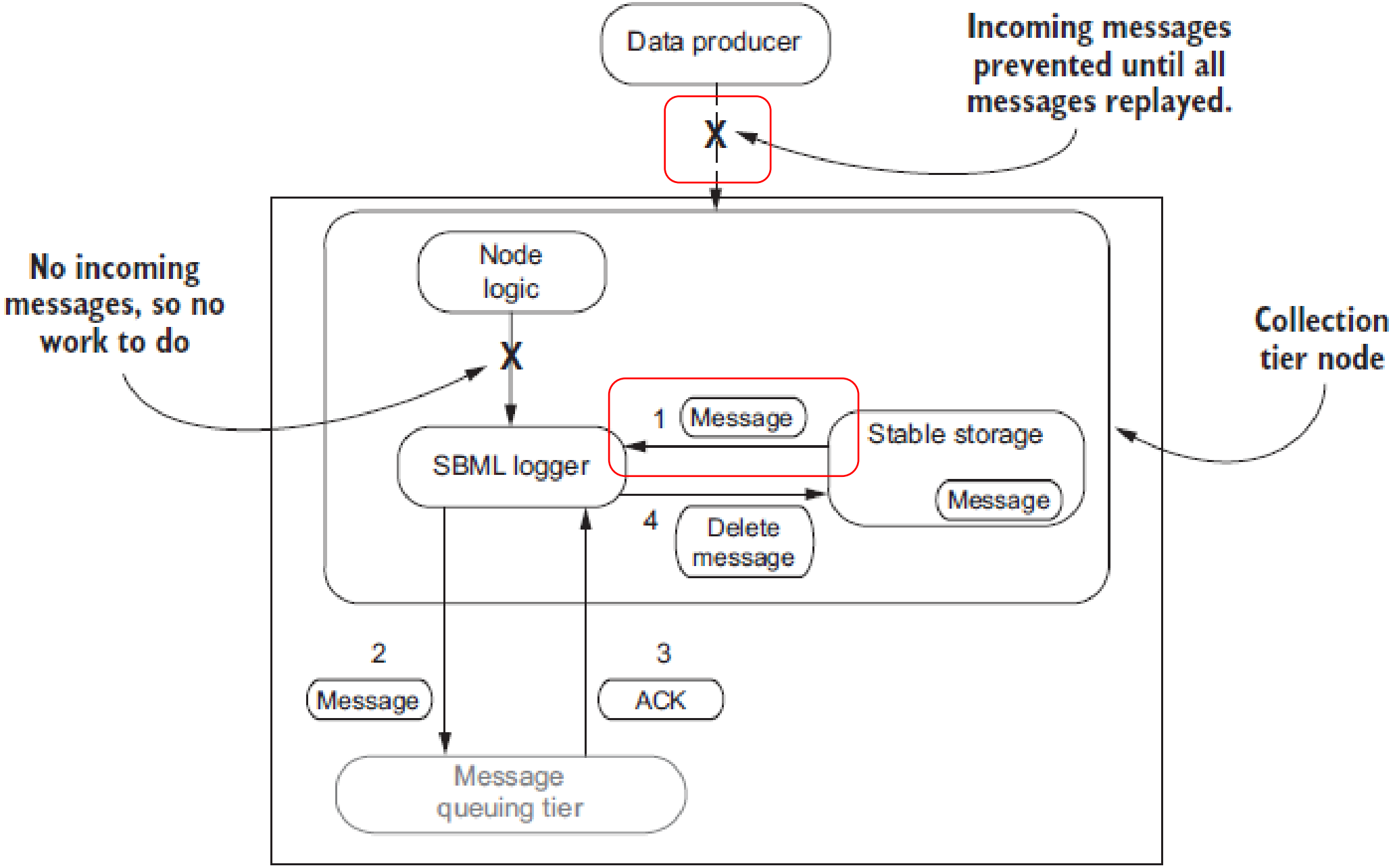



Data producer
X
Incoming messages prevented until all messages replayed.
RBML logger
1
Message
Stable storage
Message
2
Message
Node logic
3
Message
Message queuing tier
Collection tier node

Sender-based message logging (SBML)

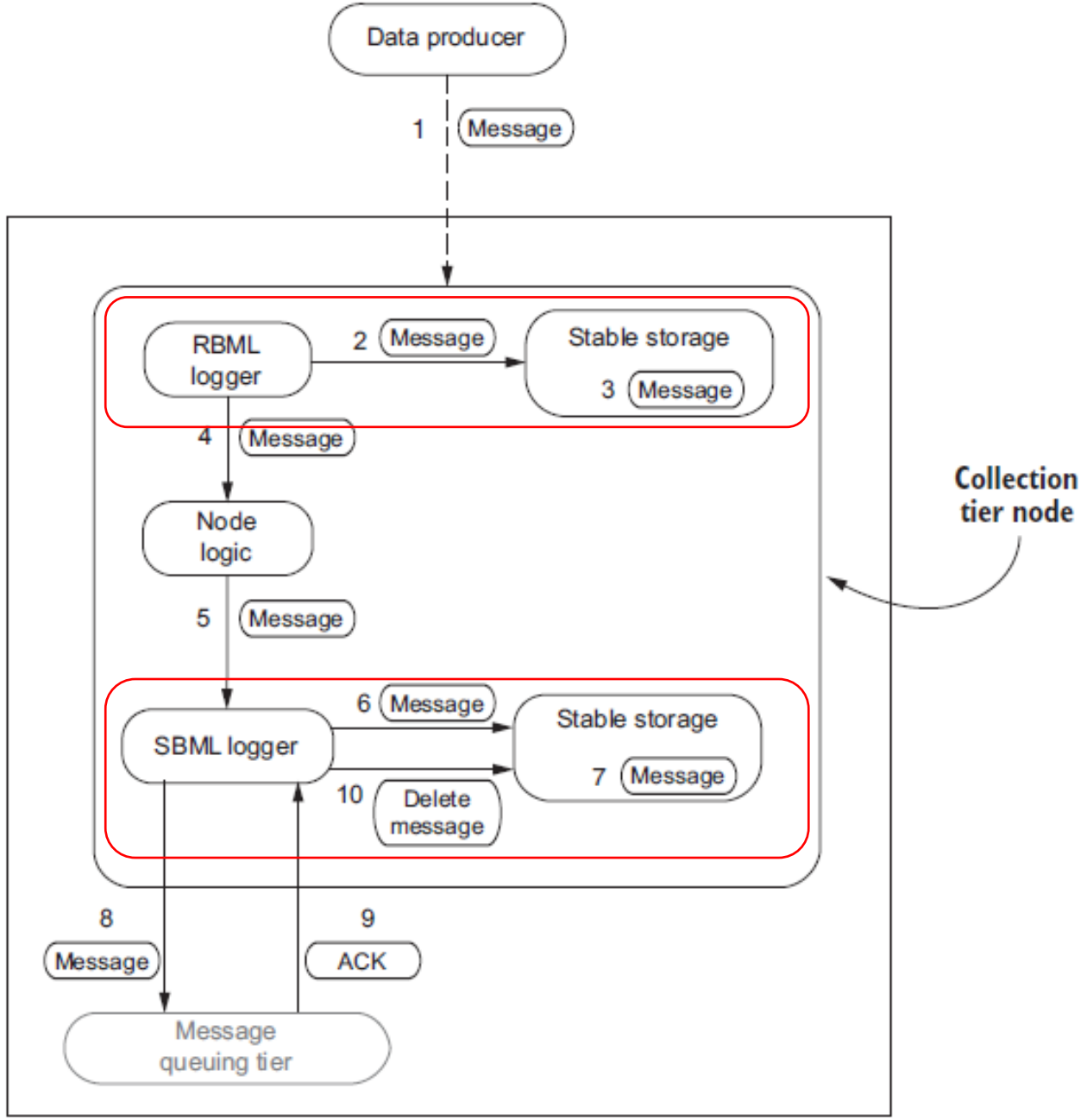
- Method: Writing received message to stable storage **before it is sent**.



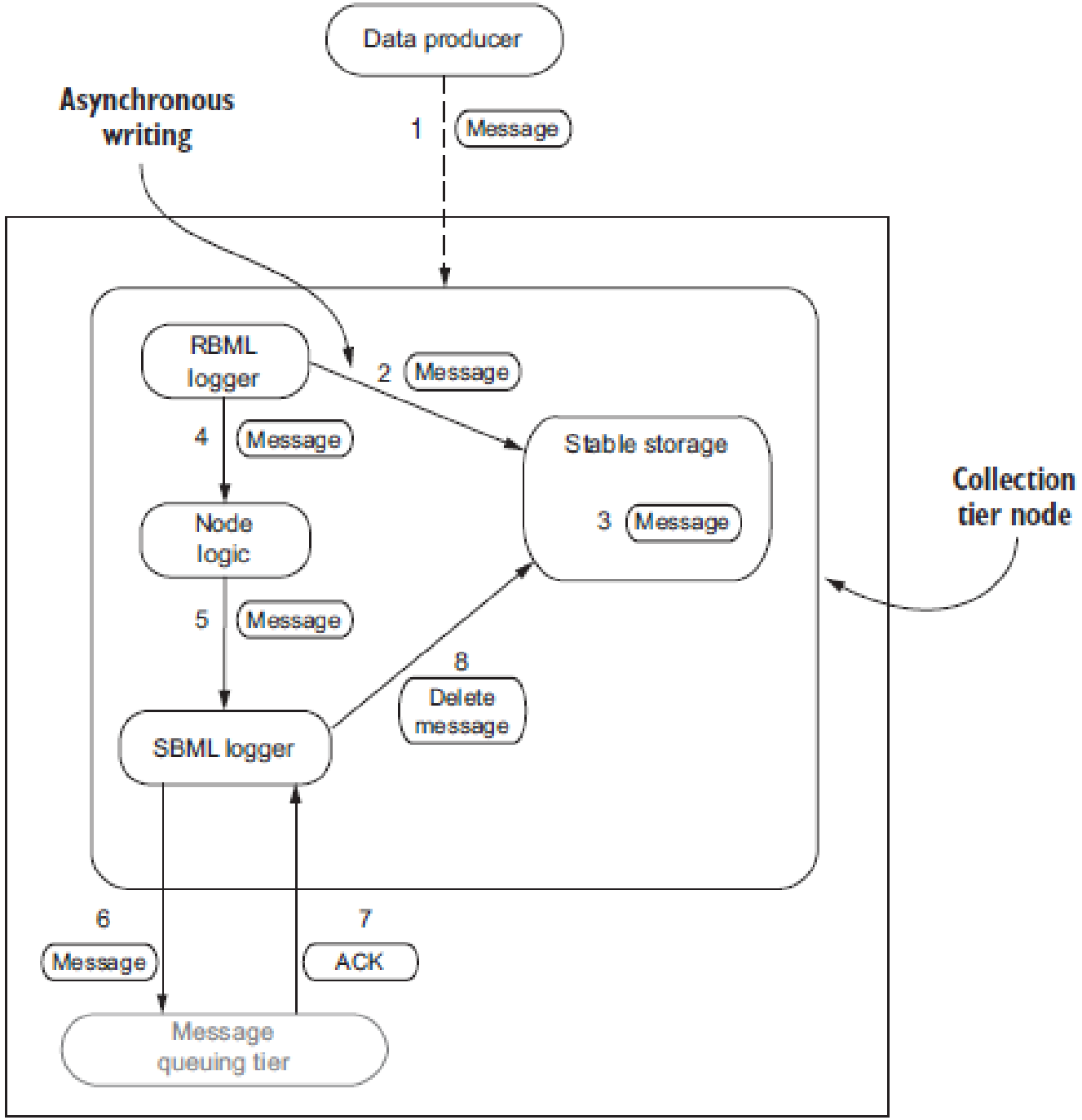

Data producer
Incoming messages prevented until all messages replayed.
X
No incoming messages, so no work to do
Node logic
X
Collection tier node
1
Message
SBML logger
Stable storage
Message
4
Delete message
2
3
Message
ACK
Message queuing tier

But what are the
benefits and **drawbacks**?

- Can we combine them?

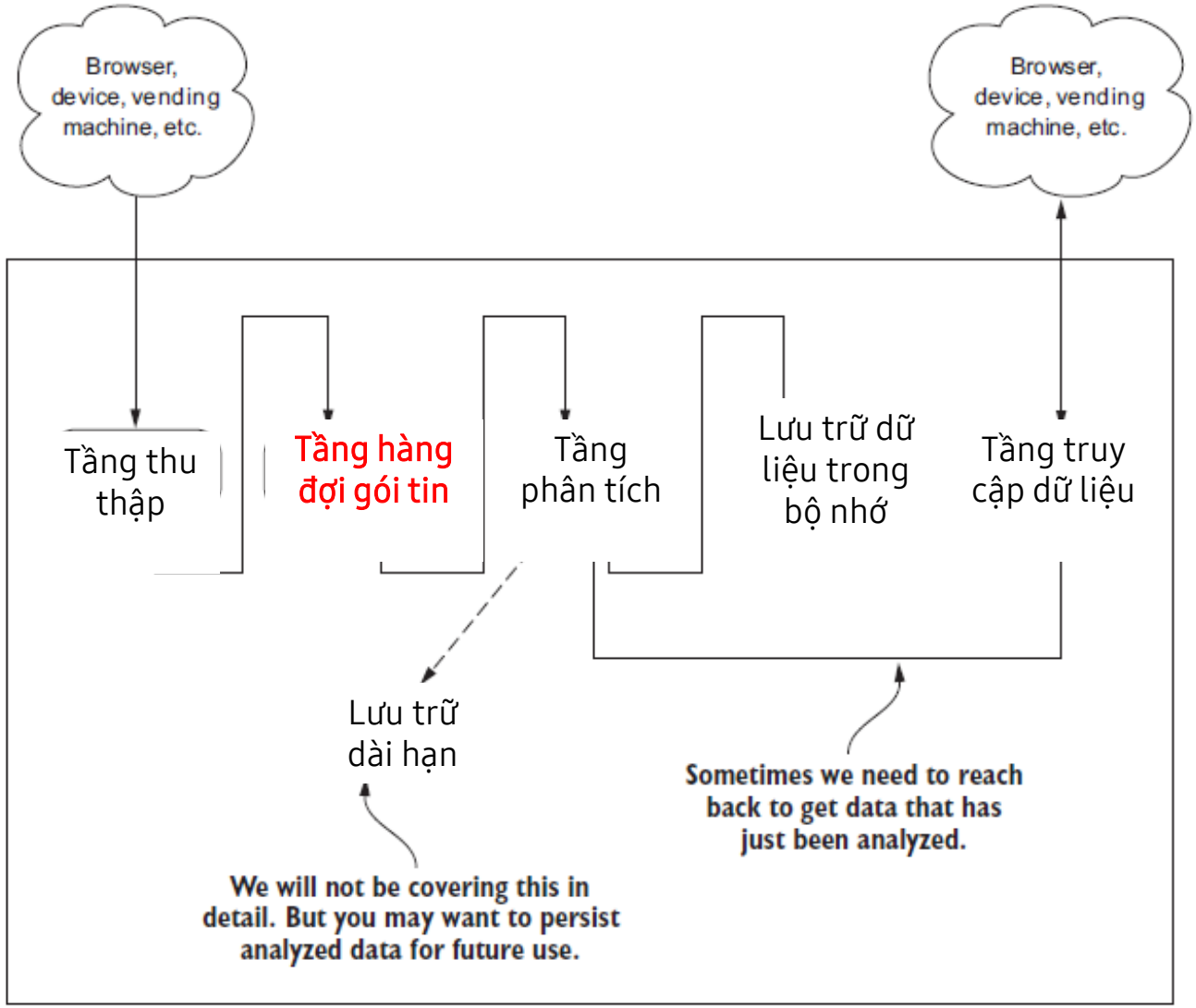


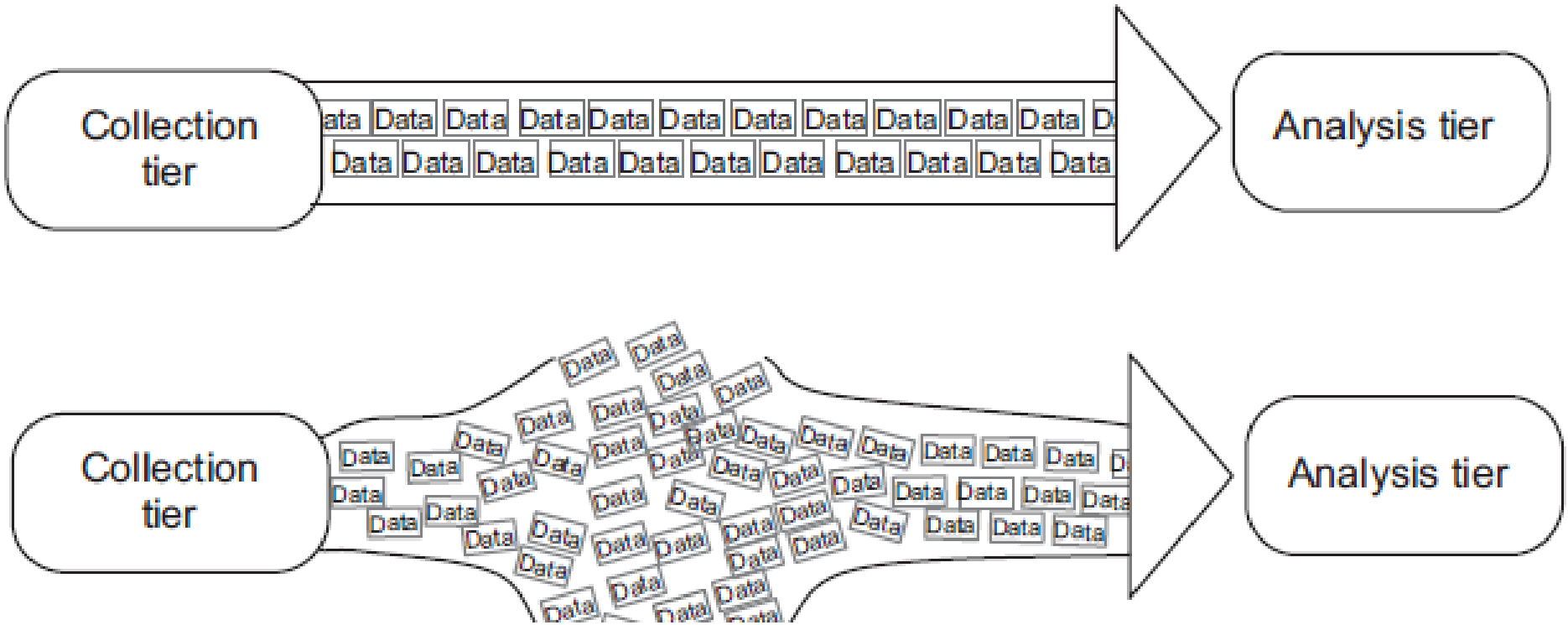
- A better approach is “hybrid message logging” (HML).
- Faster recovery process & reducing the impact of failure-free executions.

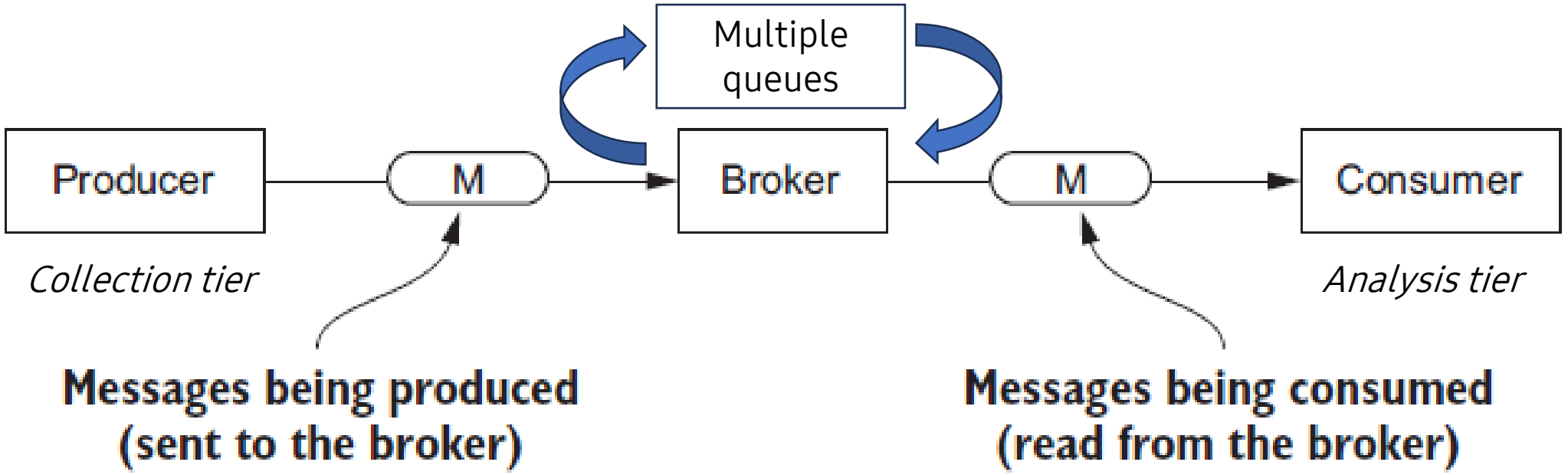


Fault tolerance in reality

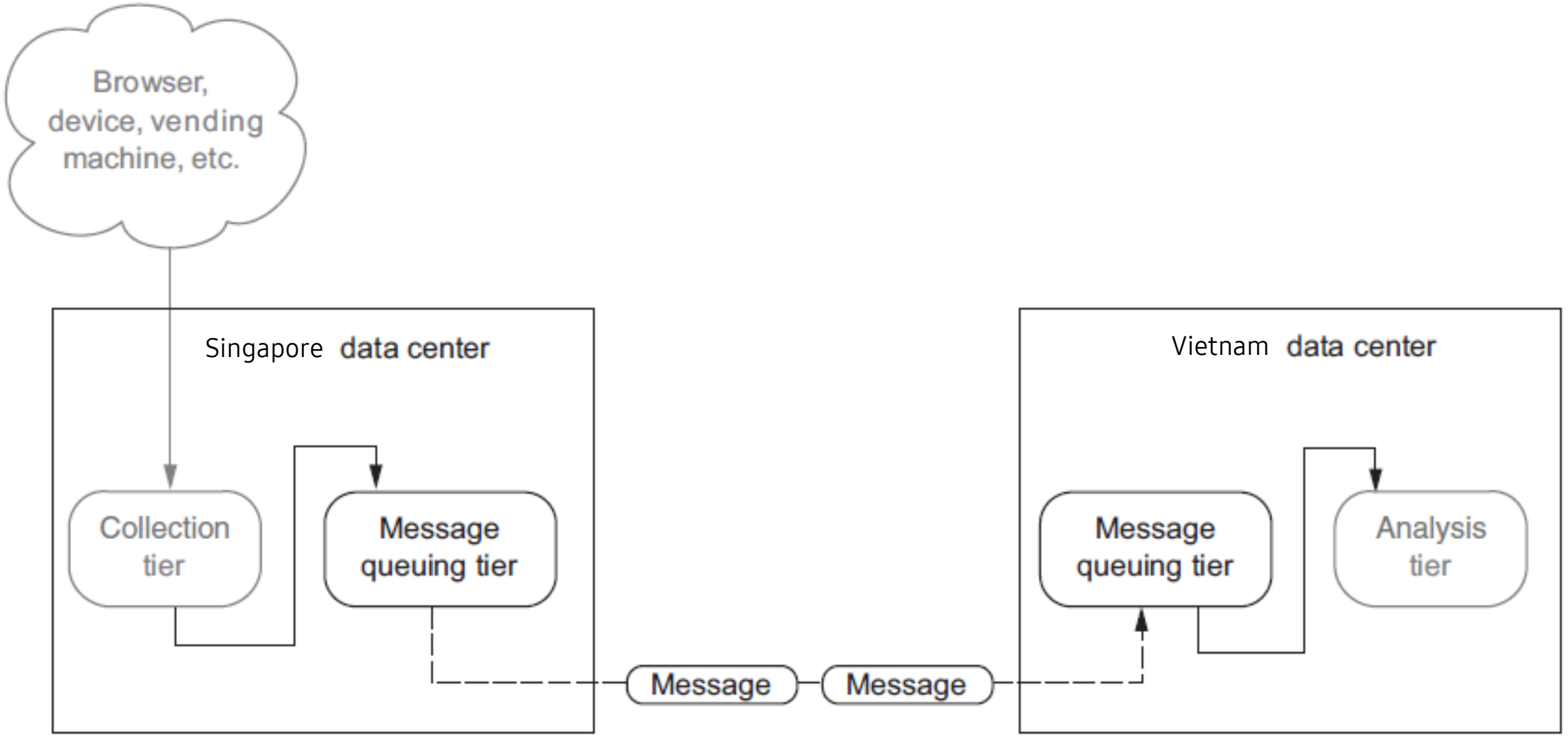
II. Message queuing tier

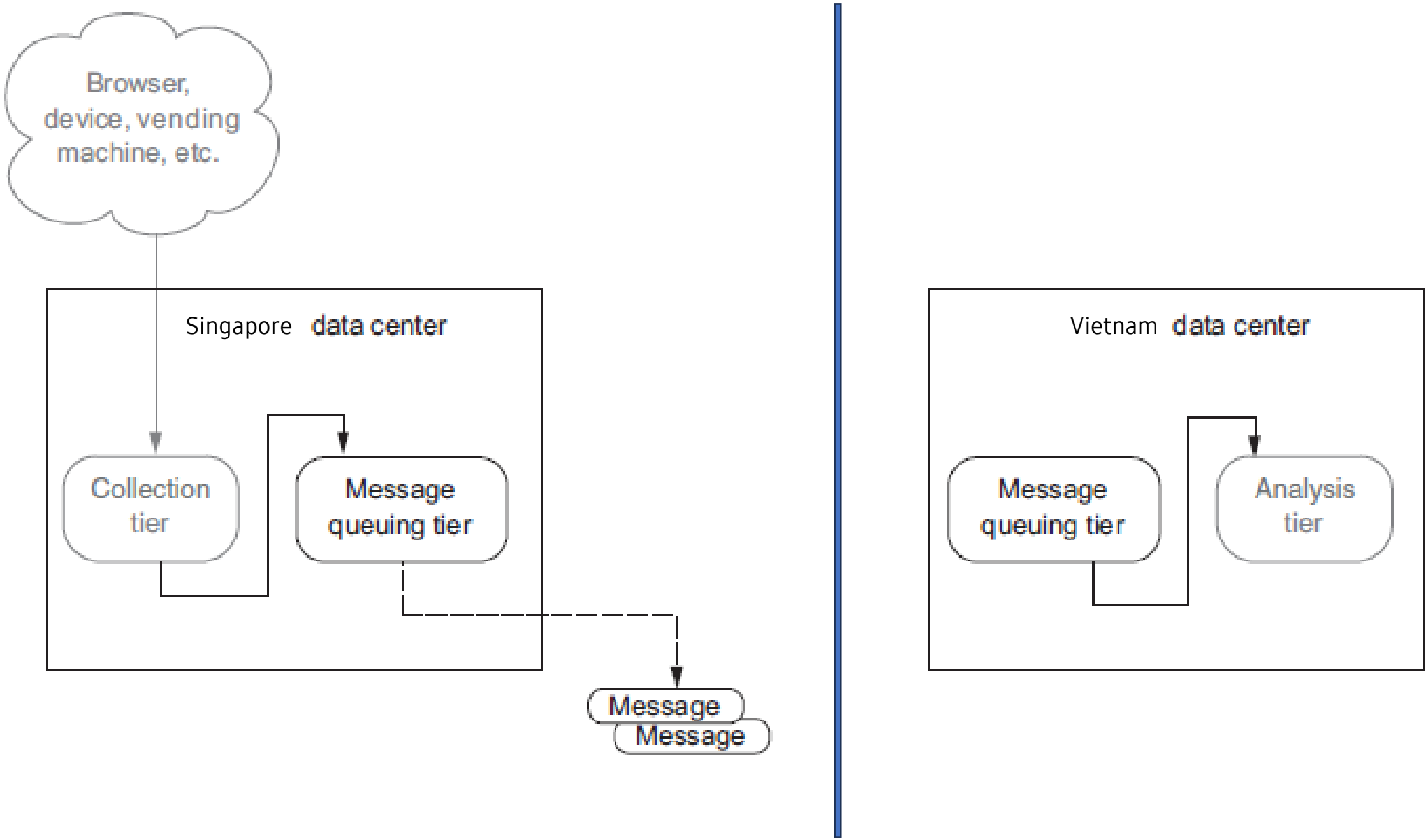


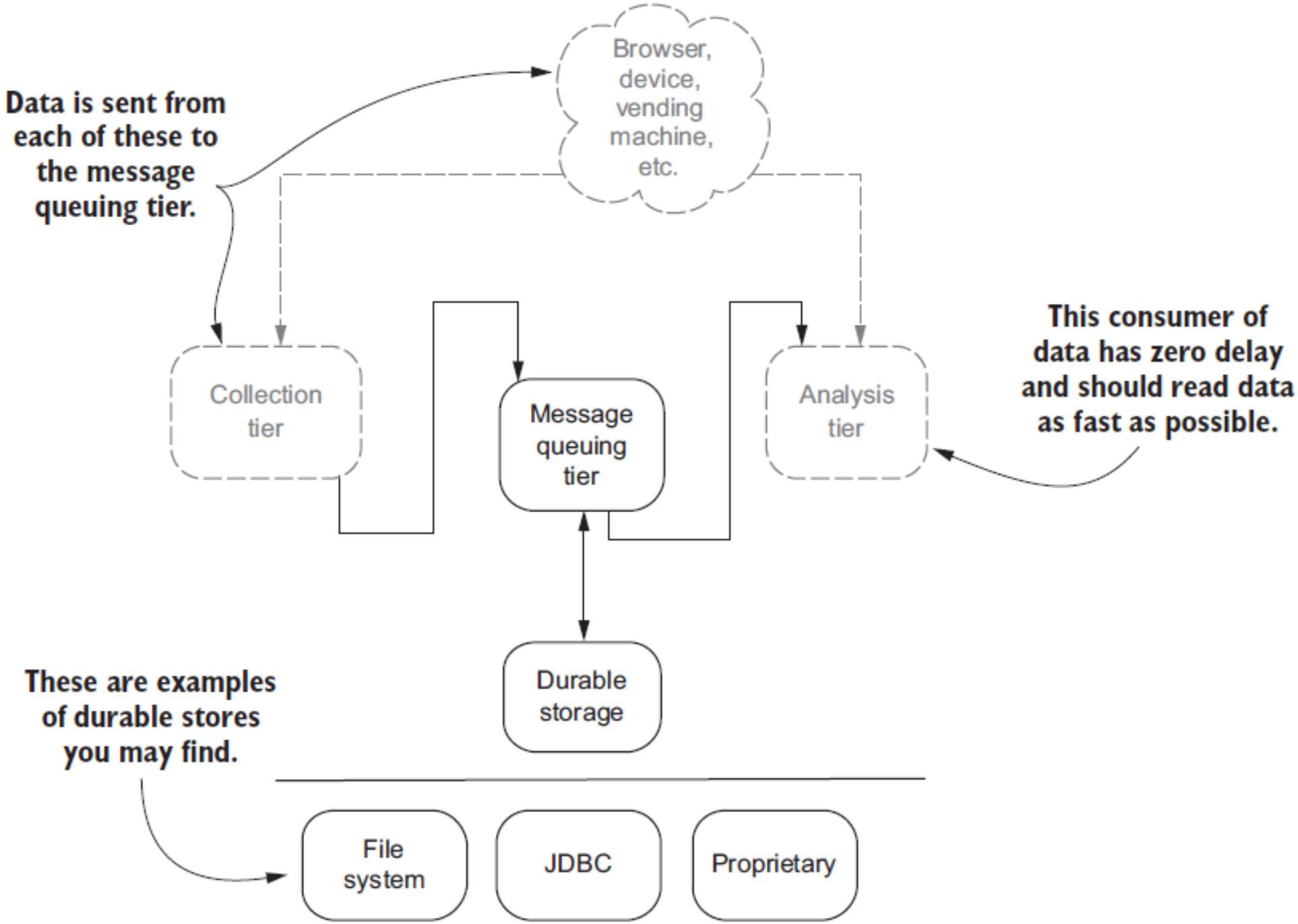

Collection tier
Data
Analysis tier
Collection tier
Data
Analysis tier

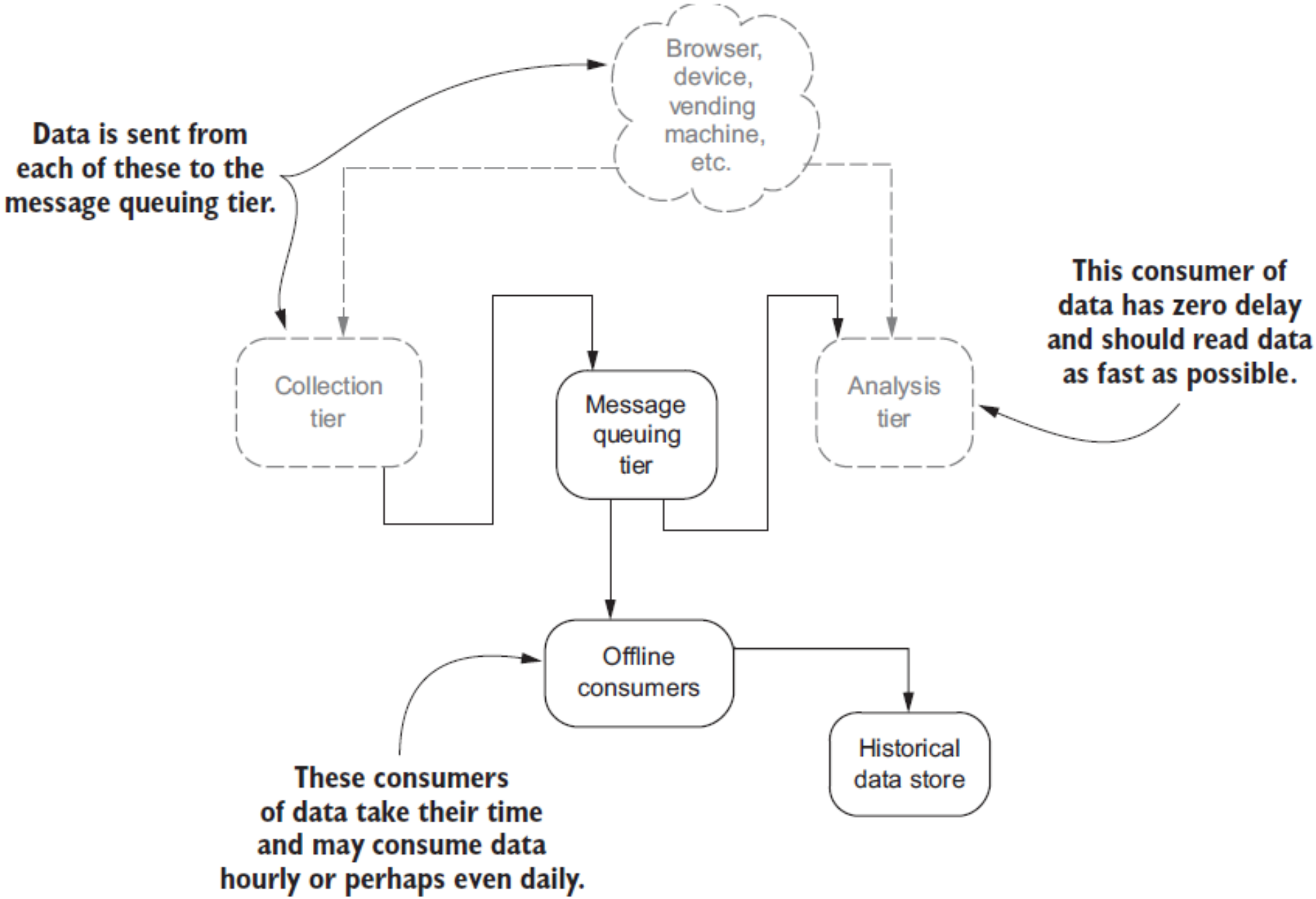

Multiple queues
Producer
M
Broker
M
Consumer
Collection tier
Analysis tier
Messages being produced (sent to the broker)
Messages being consumed (read from the broker)

Durability


Browser, device, vending machine, etc.
Singapore data center
Collection tier
Message queuing tier
Message
Message
Vietnam data center
Message queuing tier
Analysis tier

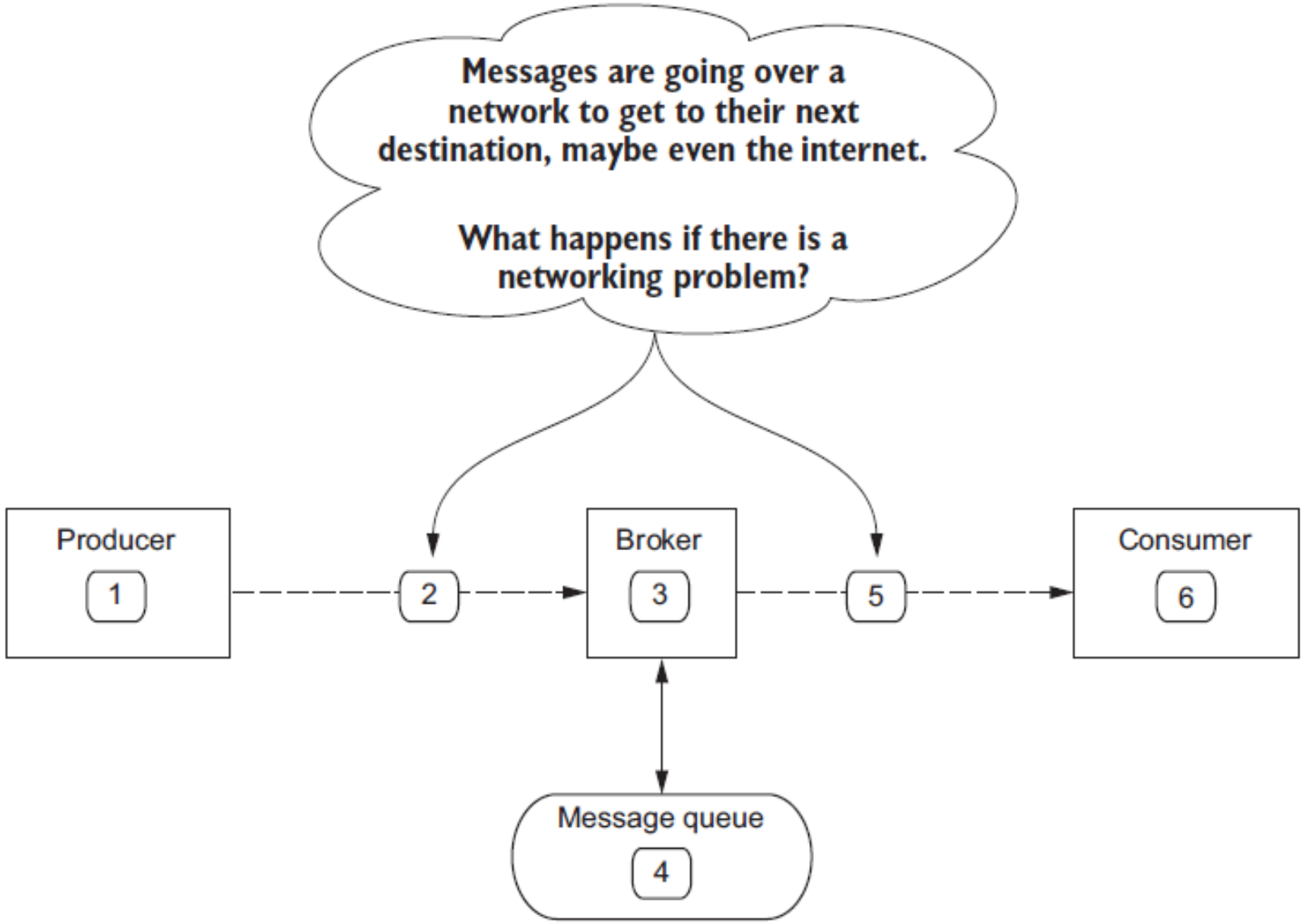

Browser, device, vending machine, etc.
Singapore data center
Collection tier
Message queuing tier
Message
Message
Vietnam data center
Message queuing tier
Analysis tier


Data is sent from each of these to the message queuing tier.
Browser, device, vending machine, etc.
Collection tier
Message queuing tier
Analysis tier
This consumer of data has zero delay and should read data as fast as possible.
Durable storage
These are examples of durable stores you may find.
File system
JDBC
Proprietary


Browser, device, vending machine, etc.
Data is sent from each of these to the message queuing tier.
Collection tier
Message queuing tier
Analysis tier
This consumer of data has zero delay and should read data as fast as possible.
Offline consumers
Historical data store
These consumers of data take their time and may consume data hourly or perhaps even daily.

- Message queue:
  - **RabbitMQ**
  - ActiveMQ
  - HornetQ
  - NSQ
  - ZeroMQ
  - **Apache Kafka**

- So, which one should I pick? (semantics/guarantees)
  - At most once: Message **may** get lost, **no reread** by a consumer.
  - At least once: Message **never** get lost, **can be reread** by a consumer.
  - Exactly once: Message **never** get lost, **can be reread** by a consumer **once and only once**.



1. Producer
2. Network wire/line
3. Broker
4. Message queue
5. Network wire/line
6. Consumer

Exactly once

At least once

At most once

- More complexity
- Faster performance
- Weaker guarantees

Exactly once

- More complexity

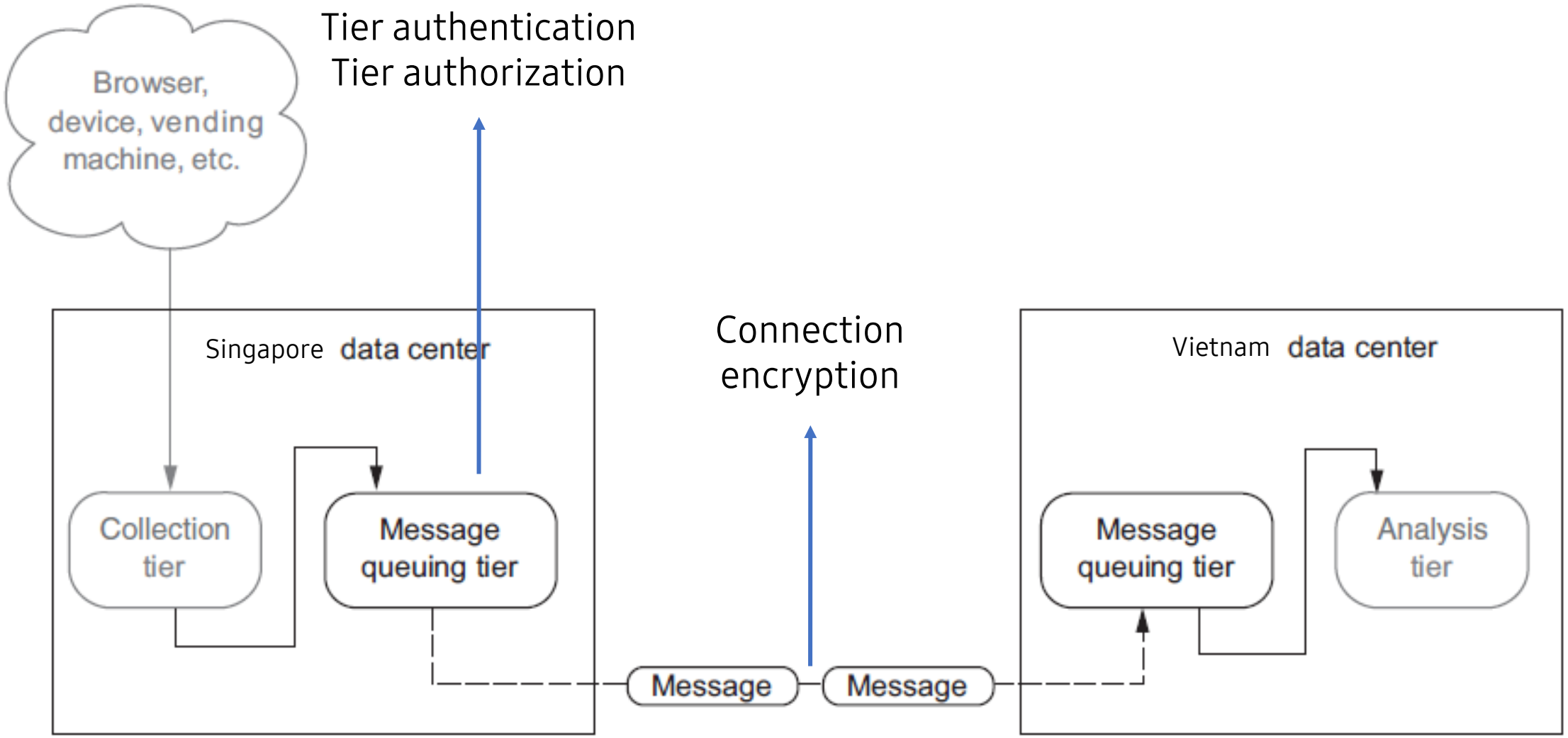
At least once

At most once

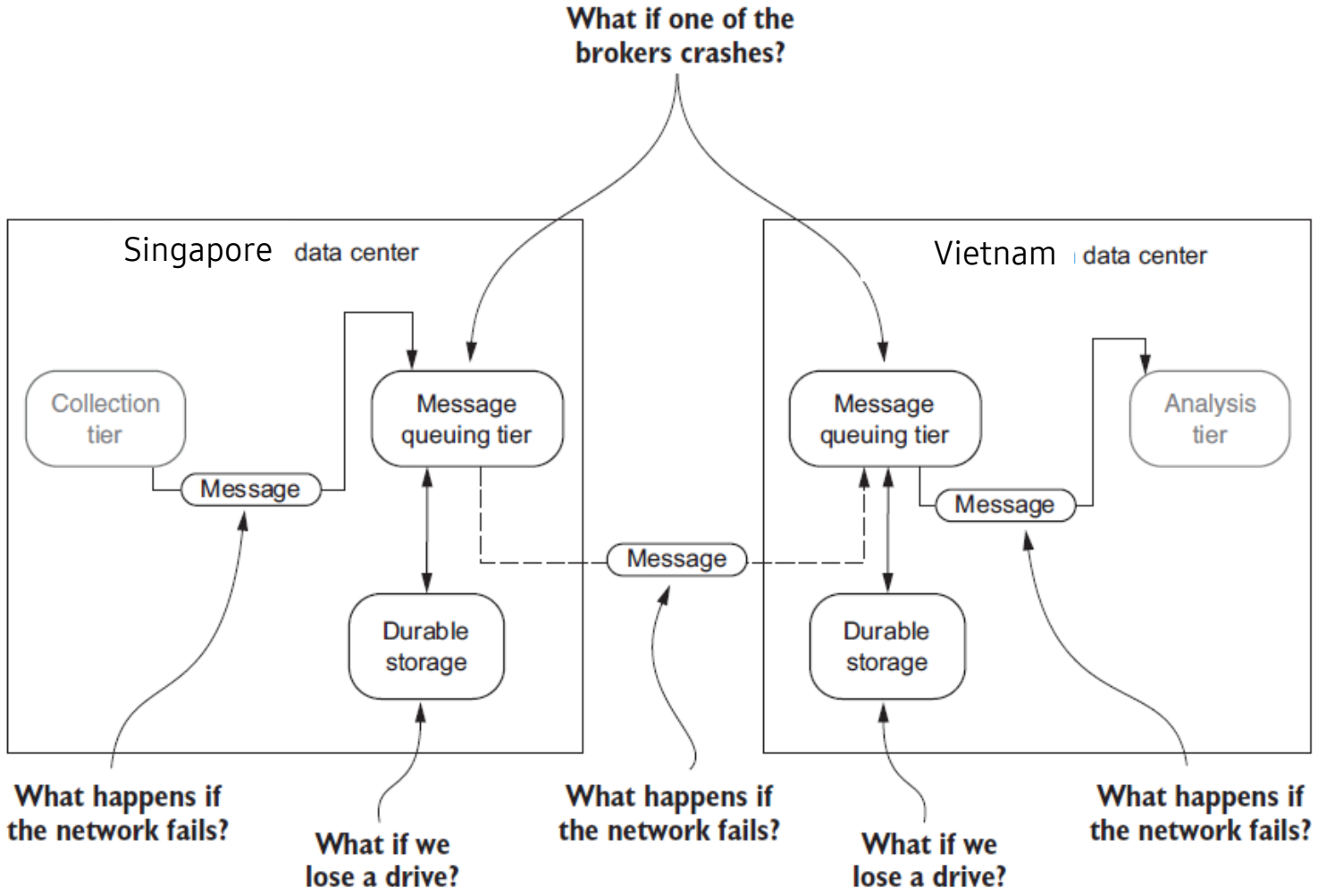
- Faster performance
- Weaker guarantees

III. Security, fault tolerance and Business scenarios

- Security can be improved by:
  - Tier authentication
  - Tier authorization
  - Connection encryption
  - Storage encryption


Tier authentication
Tier authorization
Browser, device, vending machine, etc.
Singapore data center
Collection tier
Message queuing tier
Connection encryption
Message
Message
Vietnam data center
Message queuing tier
Analysis tier

What may happen if something fails?



Issues

- Broker crashes
- Network connection is interrupted
- Drive is off

Business scenarios

- The impact of collection tier and analysis tier interruption?
- How many system days off can the business tolerate losing?
- Does the business require storing historical data?
- Suitable type of streaming system semantics?

Question	Discussion
Foodi – A food ordering system for Android app *The streaming system collects users' behaviors when viewring a restaurant page: Viewing duration, scroll rate, viewed dishes,...*	
The impact of collection tier and analysis tier interruption?	Not being able to recommend restaurant or dishes efficiently. This won't stop the recommendation system however.
How many system days off can the business tolerate losing?	The model becomes outdated over three months, which may afftect user experience (check *concept drift*).
Does the business require storing historical data?	Due to privacy agreements, no related data is stored consistently.
Suitable type of streaming system semantics?	At most once.

Question	Discussion
Description of system in one or two lines *How is the streaming system used?*	
The impact of collection tier and analysis tier interruption?	What will happen if there is interruption?
How many system days off can the business tolerate losing?	Estimate number of days the system can be off and give explanation.
Does the business require storing historical data?	Do they? And why?
Suitable type of streaming system semantics?	At most once/At least once/Exactly once. Taking account of all above answers, this should be reasonable.

Homework (group)

- Imagine you're going to build streaming systems for 2 businesses, build 2 scenario question tables for both of them.

  Notes: Don't use textbooks' examples (p.54-57) as well as mine.

Question	Discussion	
Description of system in one or two lines *How is the streaming system used?*		1.5pts
The impact of collection tier and analysis tier interruption?	What will happen if there is interruption?	1pts
How many system days off can the business tolerate losing?	Estimate number of days the system can be off and give explanation.	1pts
Does the business require storing historical data?	Do they? And why?	1pts
Suitable type of streaming system semantics?	At most once/At least once/Exactly once. Taking account of all above answers, this should be reasonable.	0.5pts