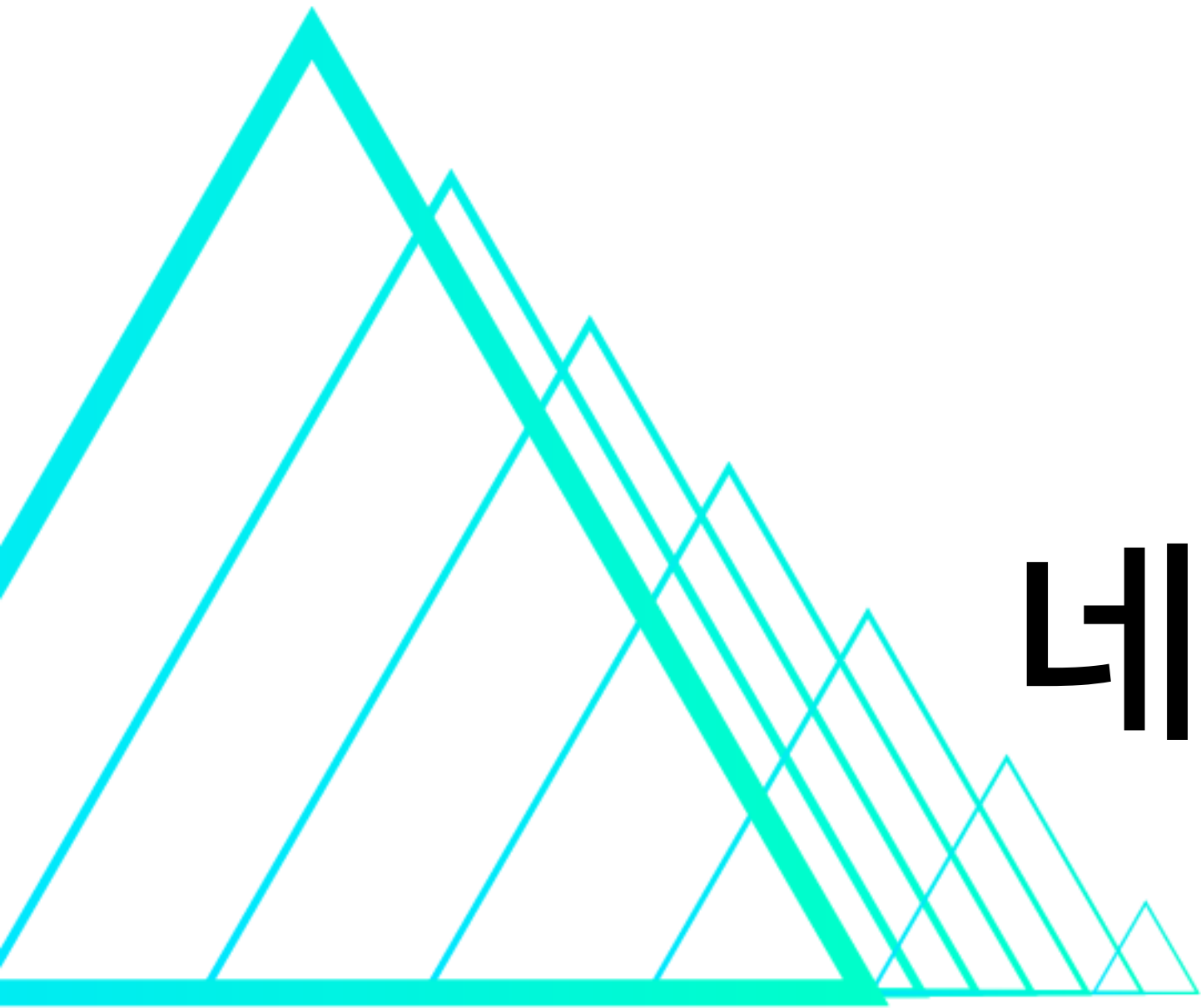
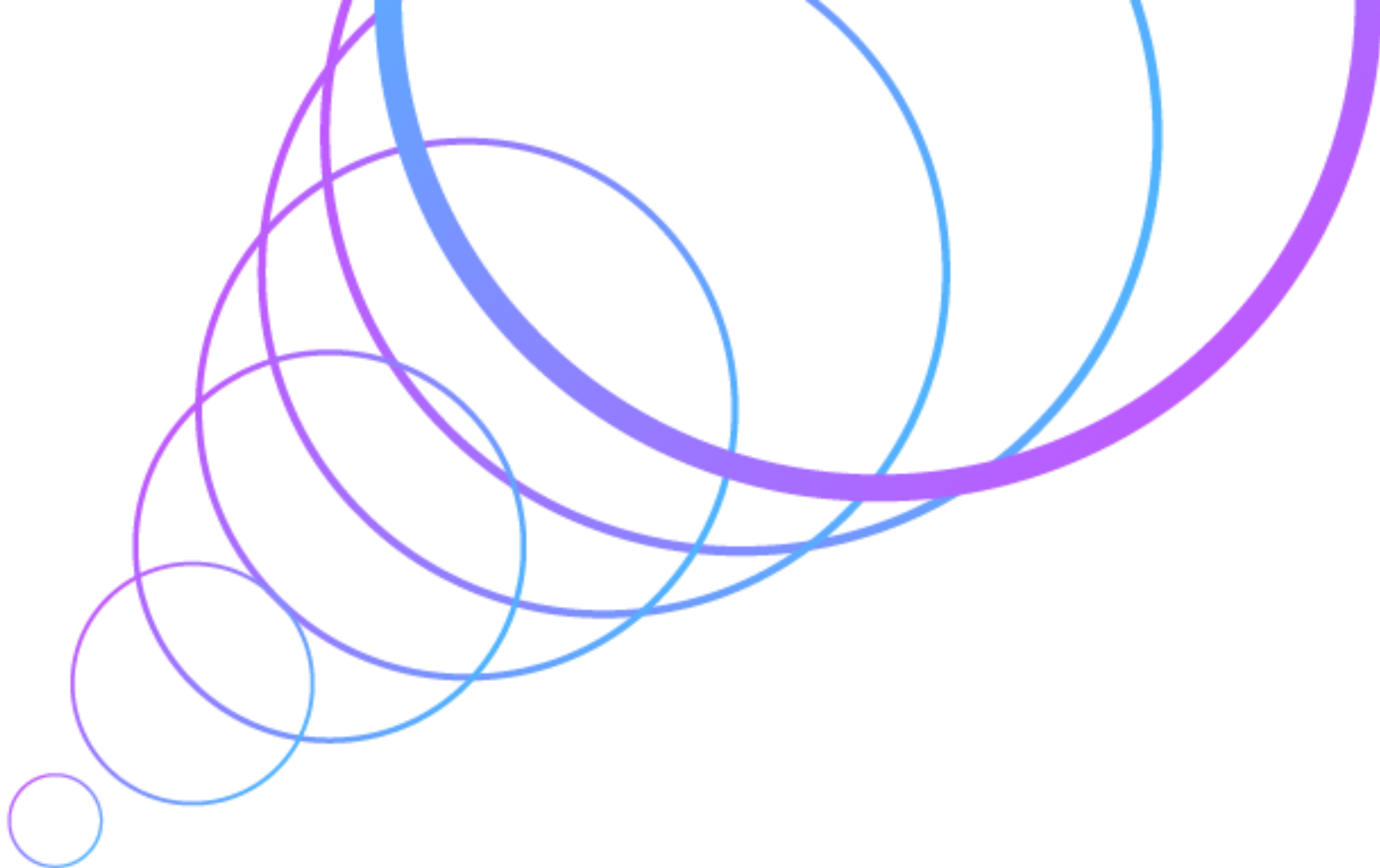




언택트 시대의 쇼핑 네이버 쇼핑라이브 개발기

장준영 NAVER 라이브커머스플랫폼

NAVER Live Commerce

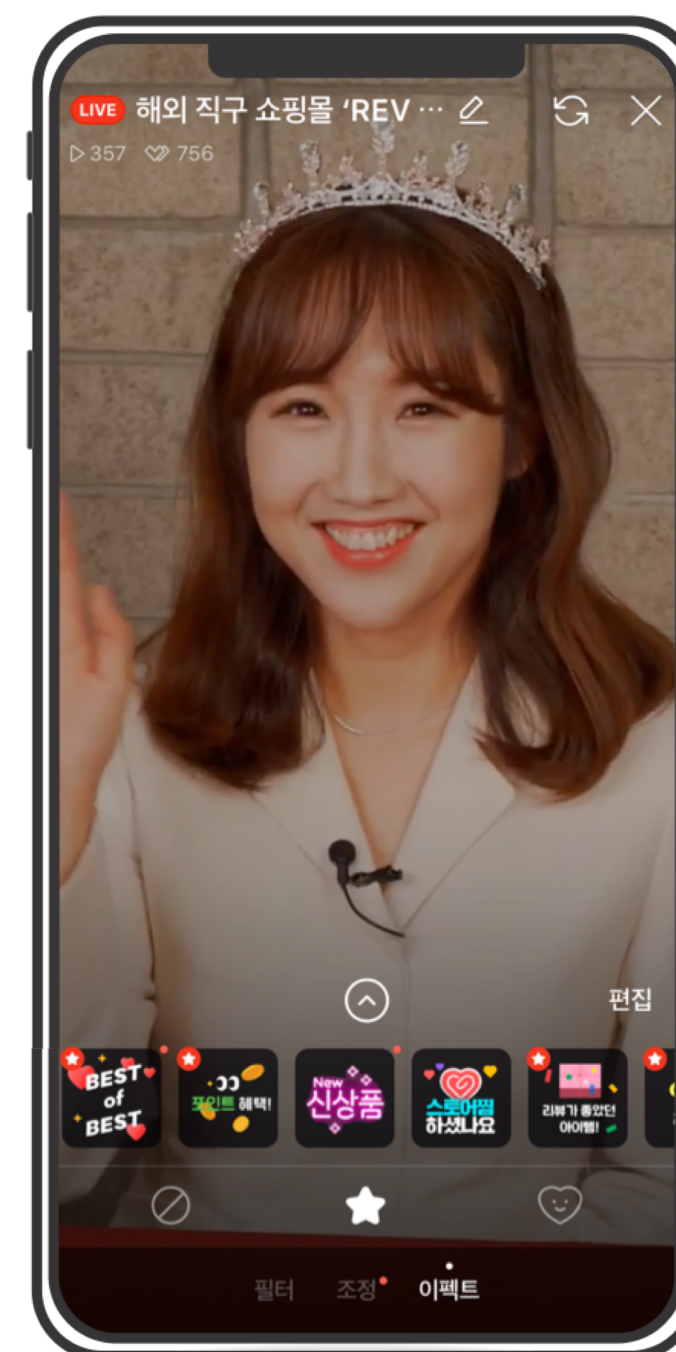
NAVER 라이브커머스

네이버 쇼핑라이브의 콘텐츠 제작/제휴/사업/운영/개발 전반을 담당합니다.

라이브커머스란? Live Streaming과 Commerce가 합쳐진 합성어로 라이브 스트리밍을 통해서 물건을 사고 파는 미디어 커머스를 일컫는다.

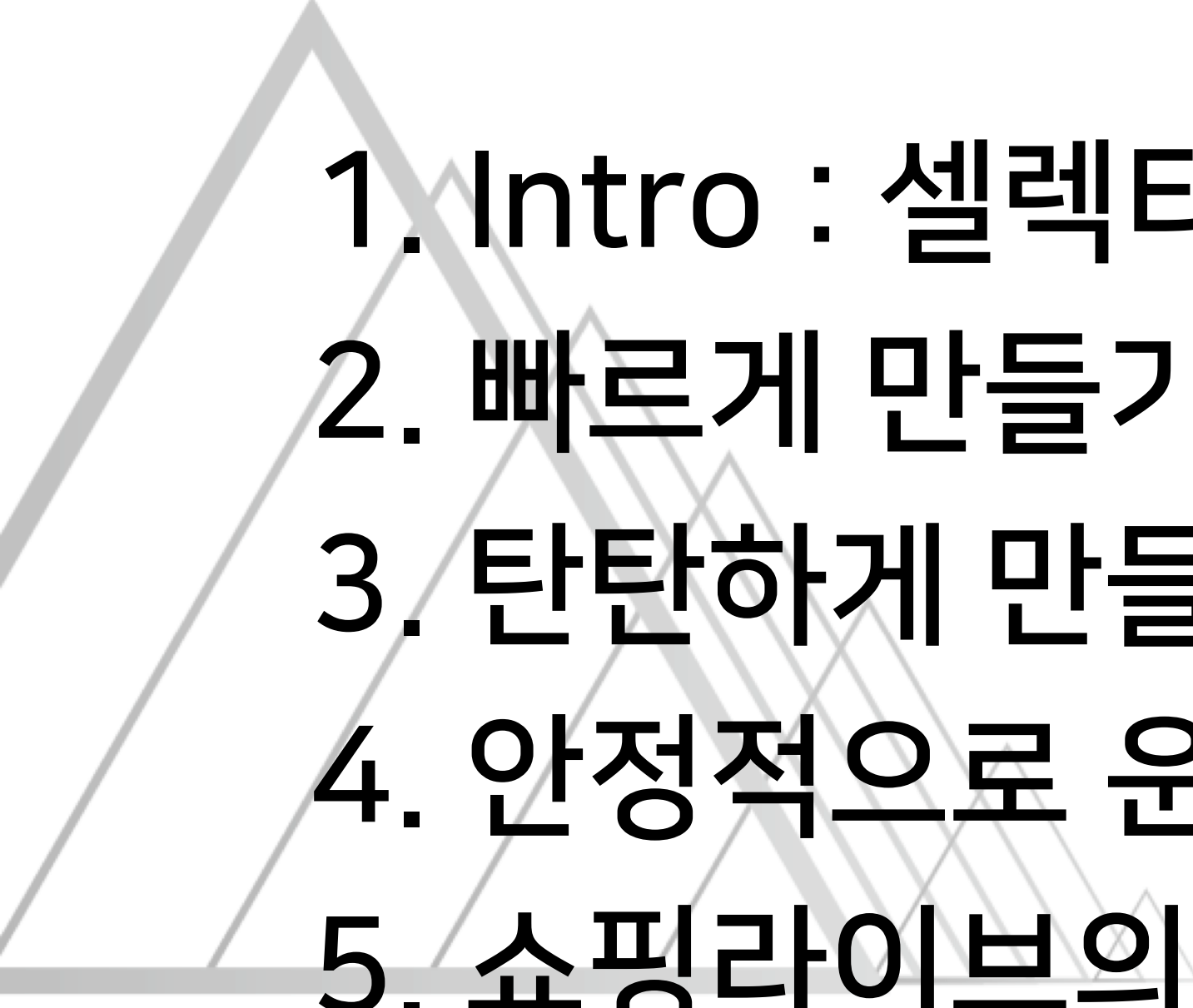


LIVE



CONTENTS



- 
1. Intro : 셀렉티브 to 쇼핑라이브
 2. 빠르게 만들기
 3. 탄탄하게 만들기
 4. 안정적으로 운영하기
 5. 쇼핑라이브의 현재와 미래
- 

1. Intro : 셀렉티브 to 쇼핑라이브

1.1 셀렉티브 - 시작

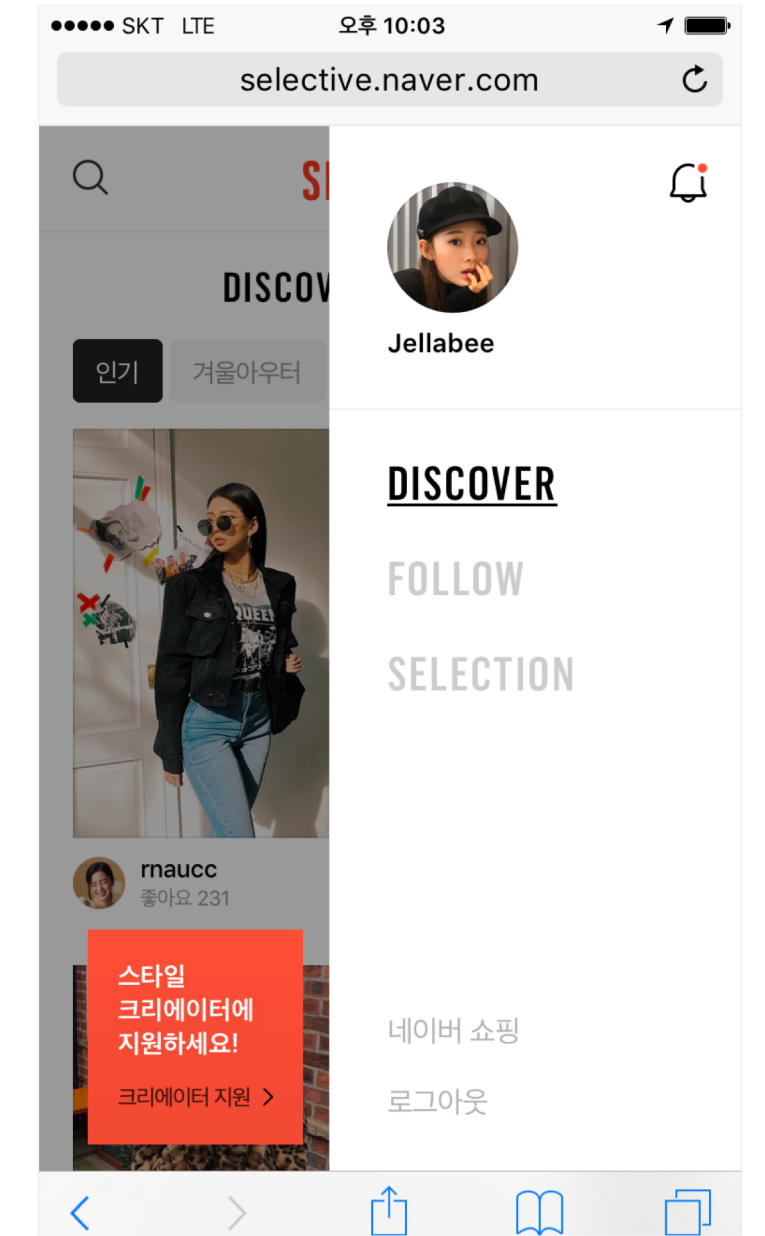
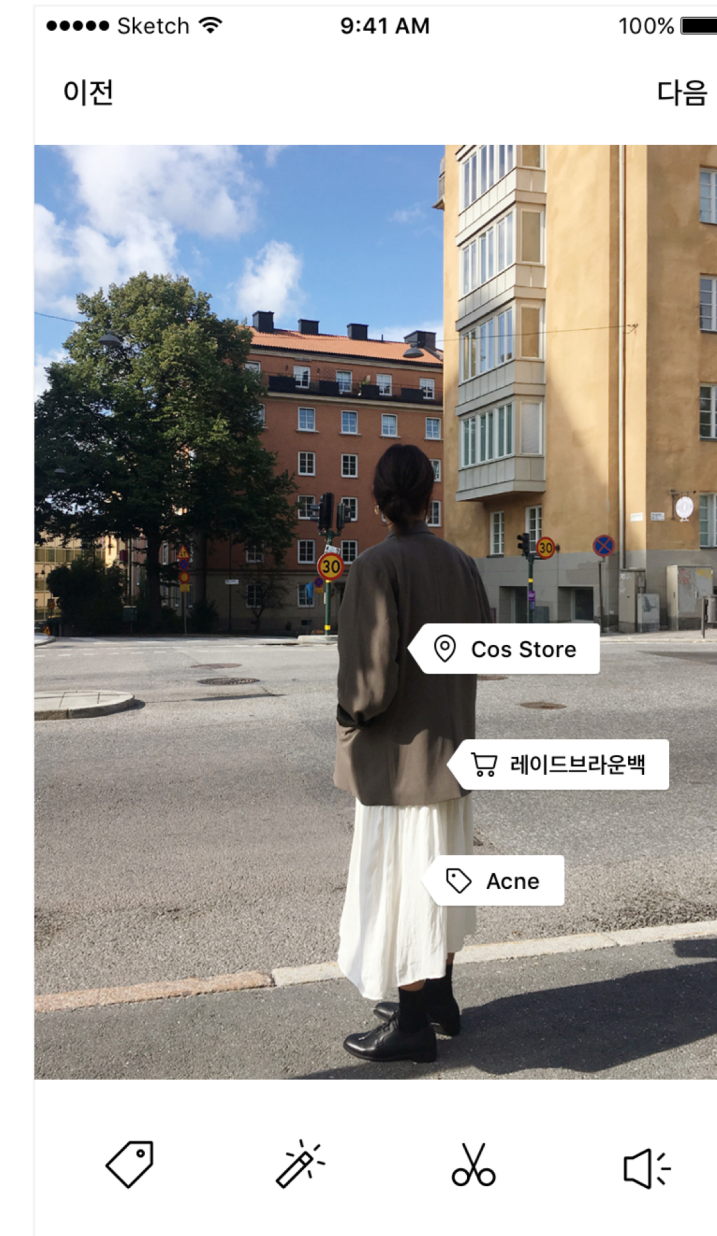
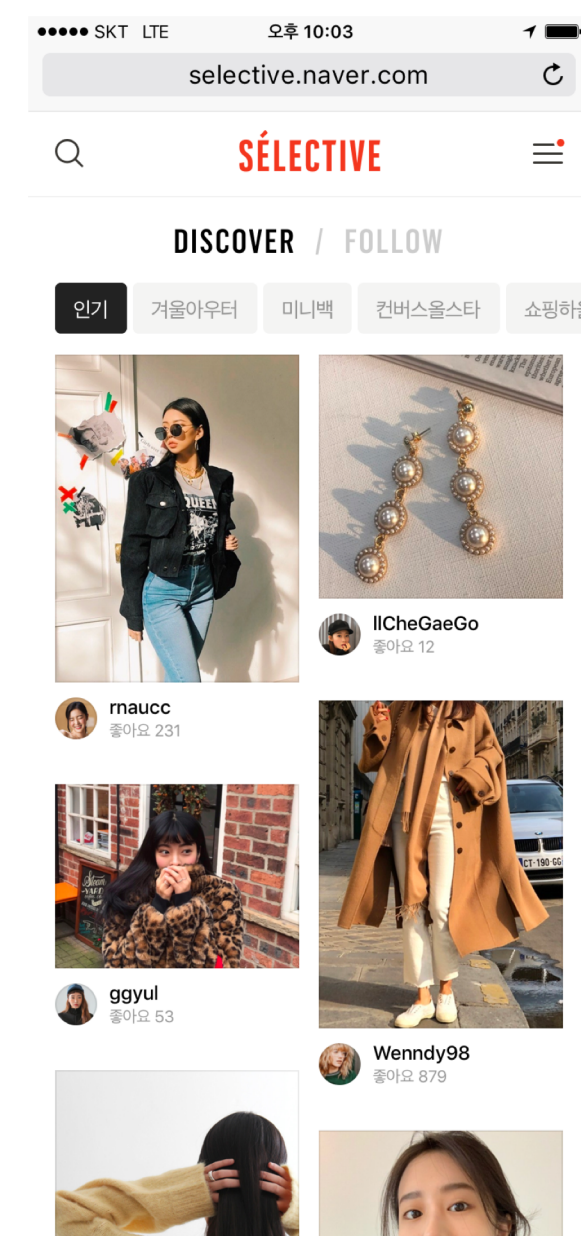
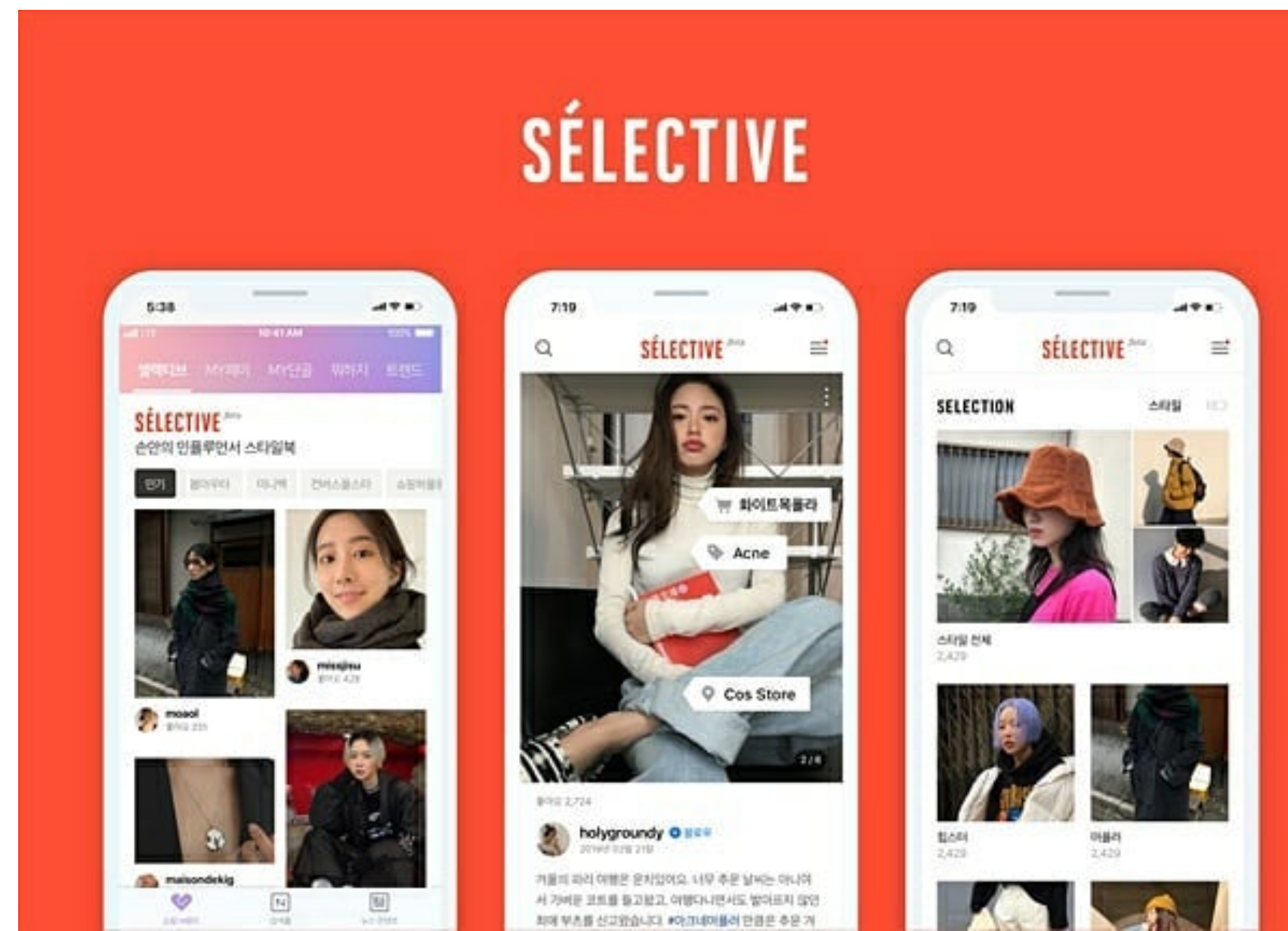
2019

2월

셀렉티브(SELECTIVE) 서비스 시작

- 2019년 2월. 패션, 스타일 콘텐츠를 한데 모아 보여주고 쇼핑으로 바로 연결해주는 “스타일 북 서비스” 출시
- 현 쇼핑라이브의 전신

무수한 고민과 노력 끝에 나온 서비스이지만 -> 성장 정체



1.2 셀렉티브 라이브 - 돌파구 찾기

2019

9월

서비스 정체기에서 찾아낸 돌파구 - 라이브커머스

- 인플루언서(판매자)와 구독자(소비자)간 실시간 화상 소통하며 동시에 제품을 판매하는 "라이브커머스" 피쳐 도입
- 당시 라이브커머스가 국내에 크게 활성화 되지 않은 상황

2개월

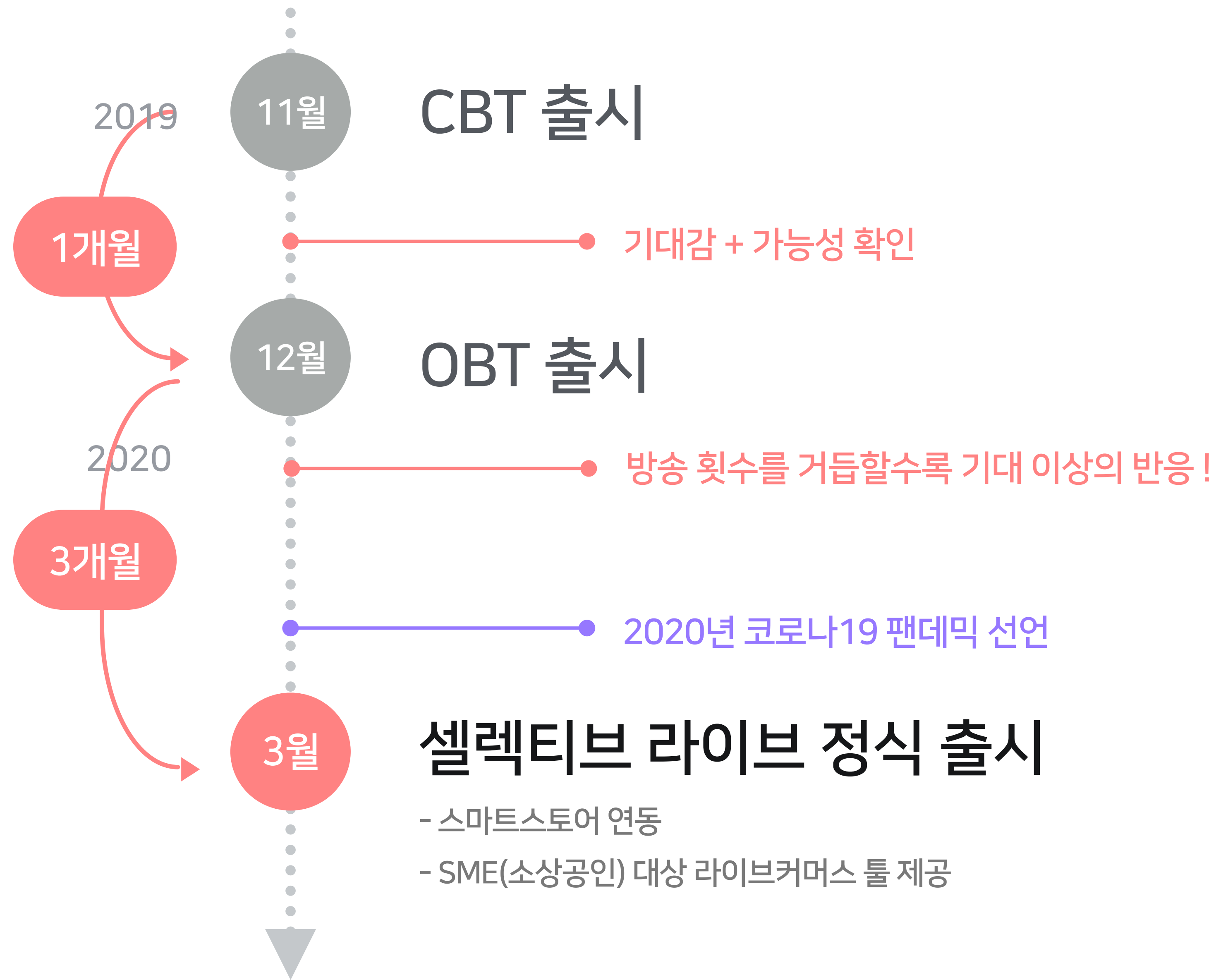
11월

빠른 시장 테스트를 위해 CBT 준비

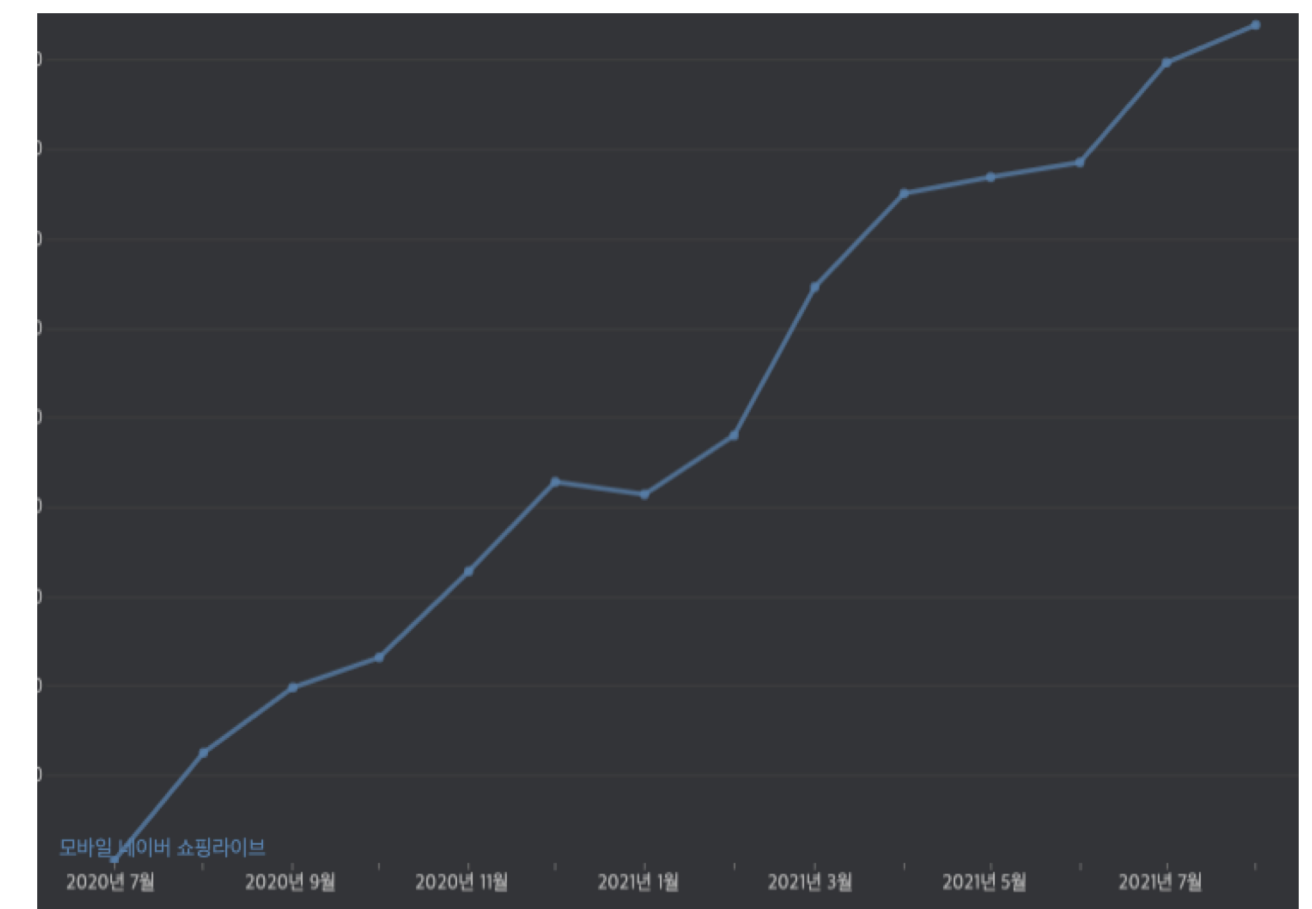
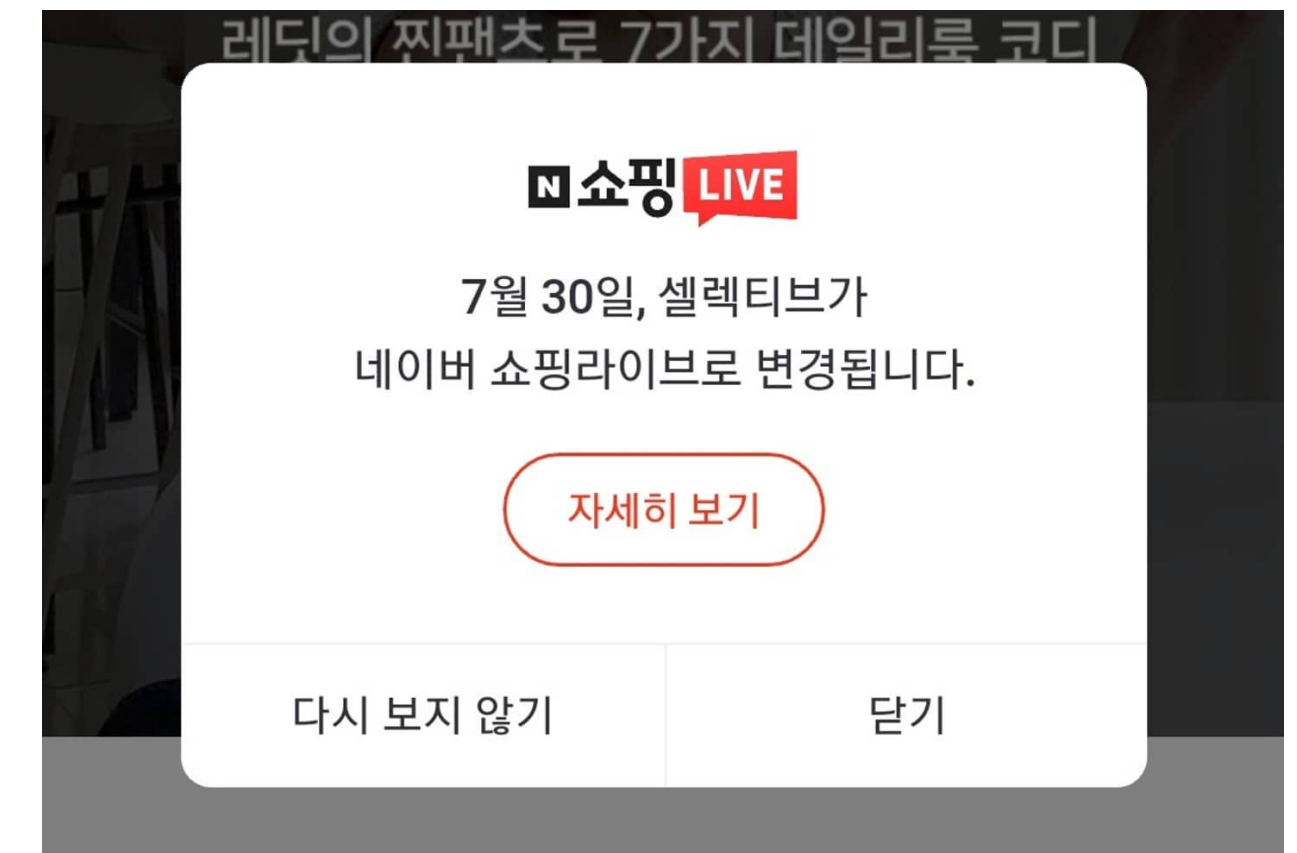
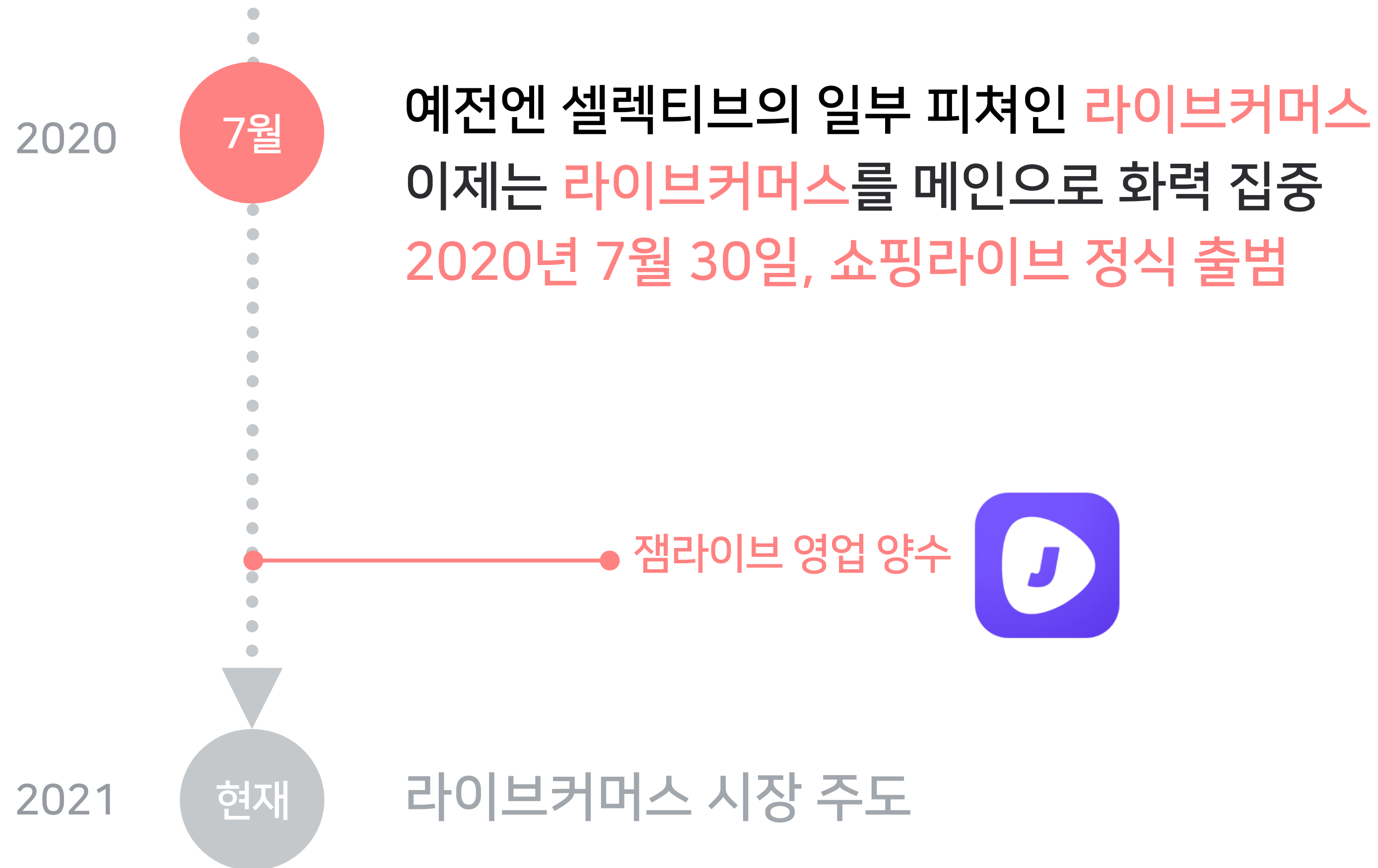
2019.11.30 첫 라이브 테스트 진행 in 판교 현대백화점



1.2 셀렉티브 라이브 - 돌파구 찾기



1.3 쇼핑라이브 - 성공적인 피봇팅

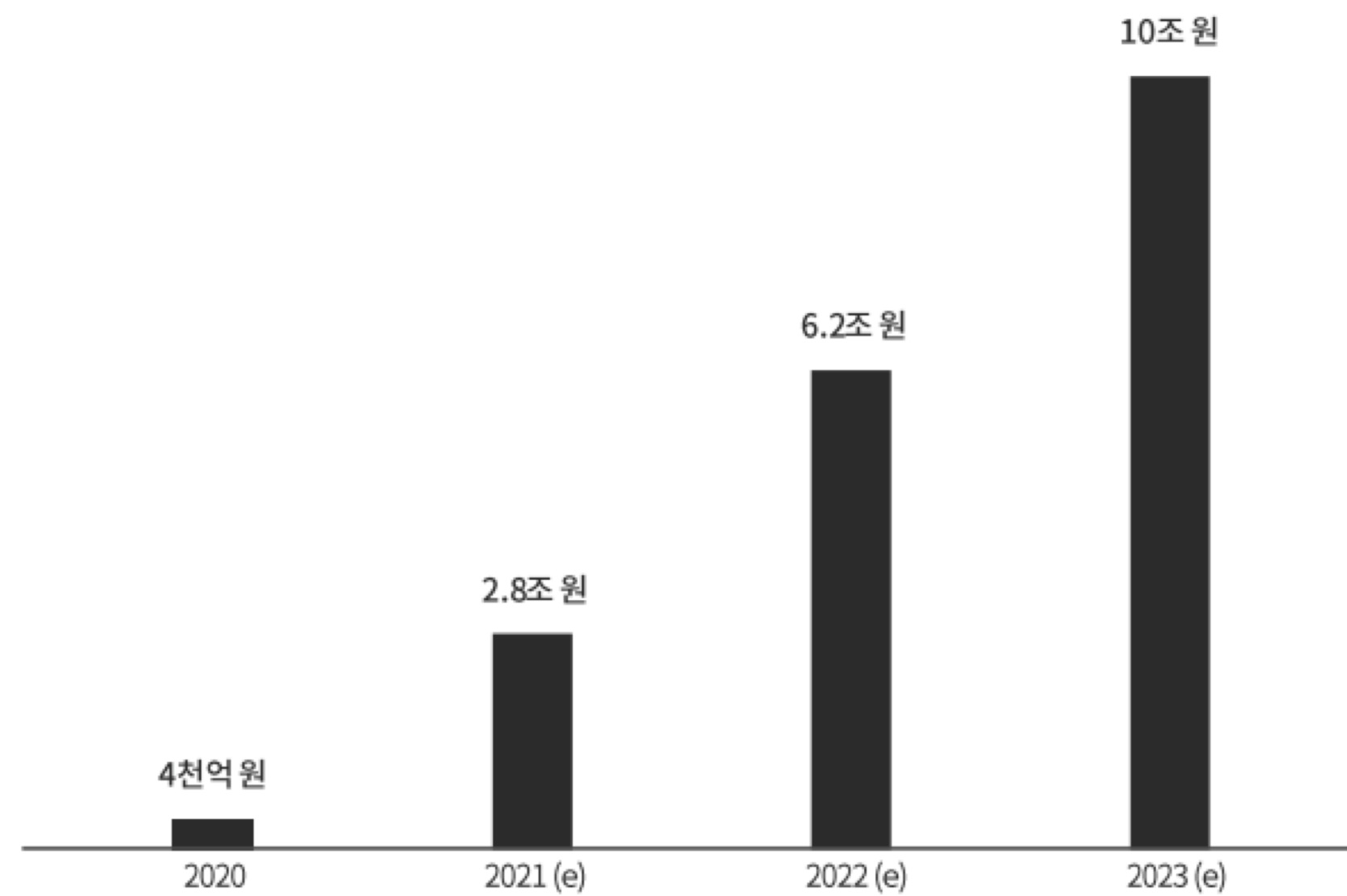


출시후 월간 트래픽 성장 추이

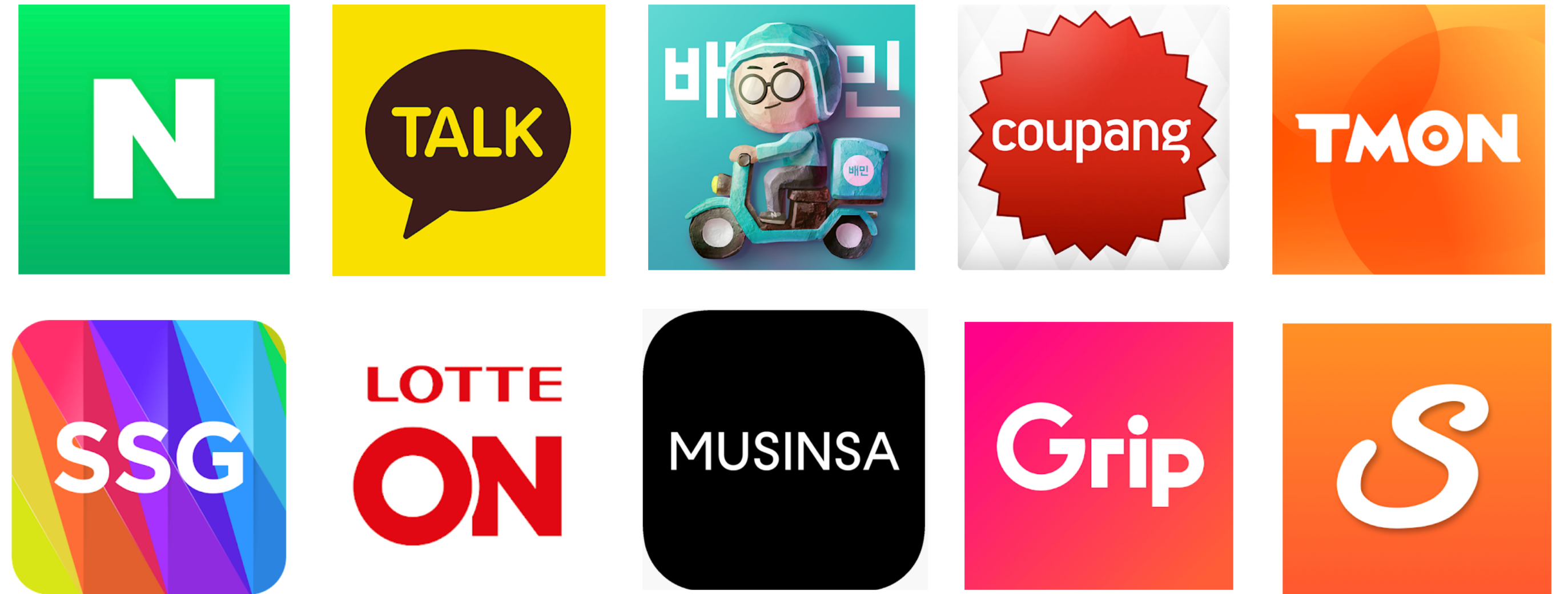
쇼핑 패러다임의 변화 - 라이브커머스

국내 라이브커머스 시장 규모 및 전망

교보증권 리서치센터, 2021



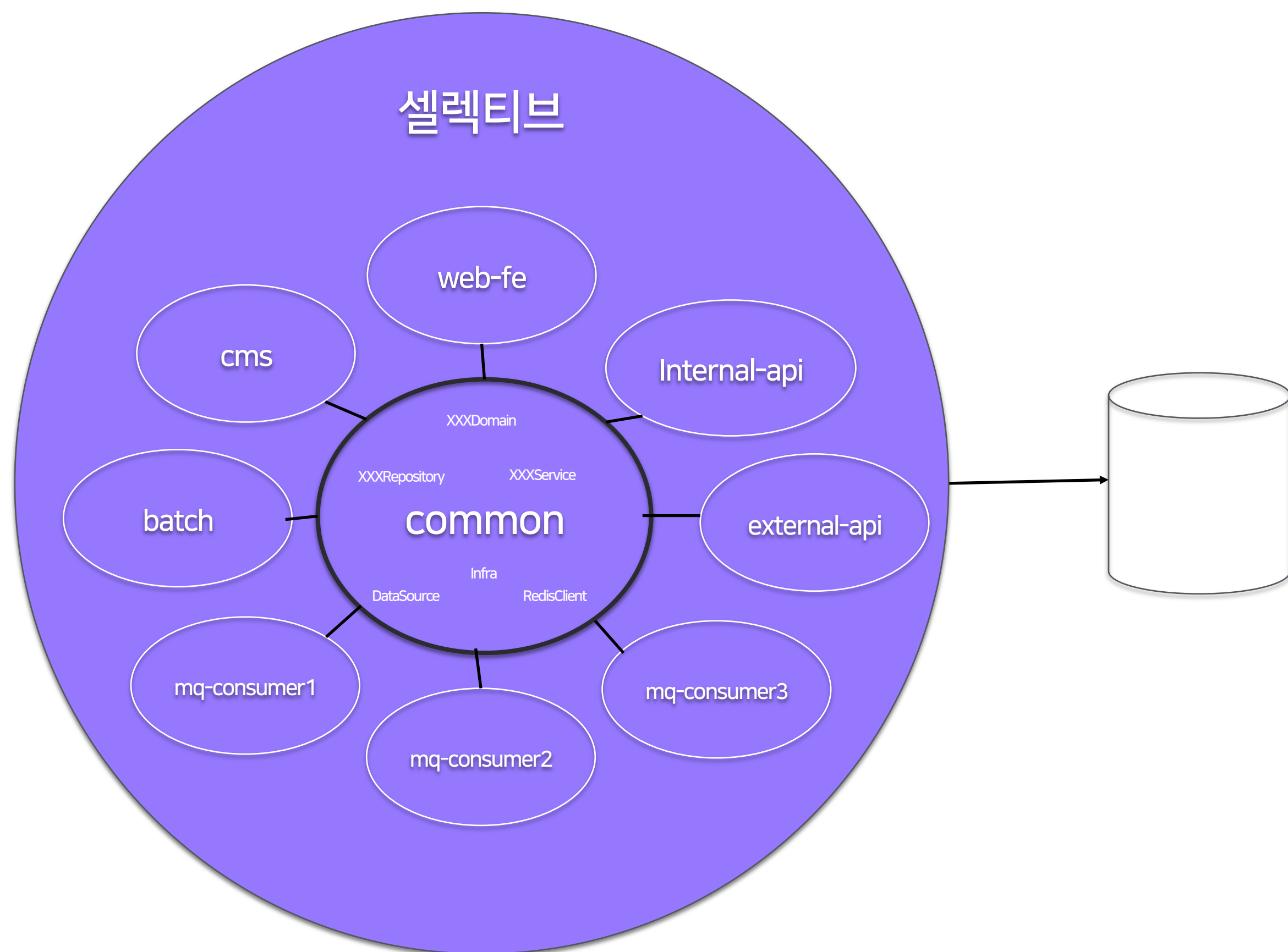
라이브커머스 제공 업체 현황



2. 빠르게 만들기

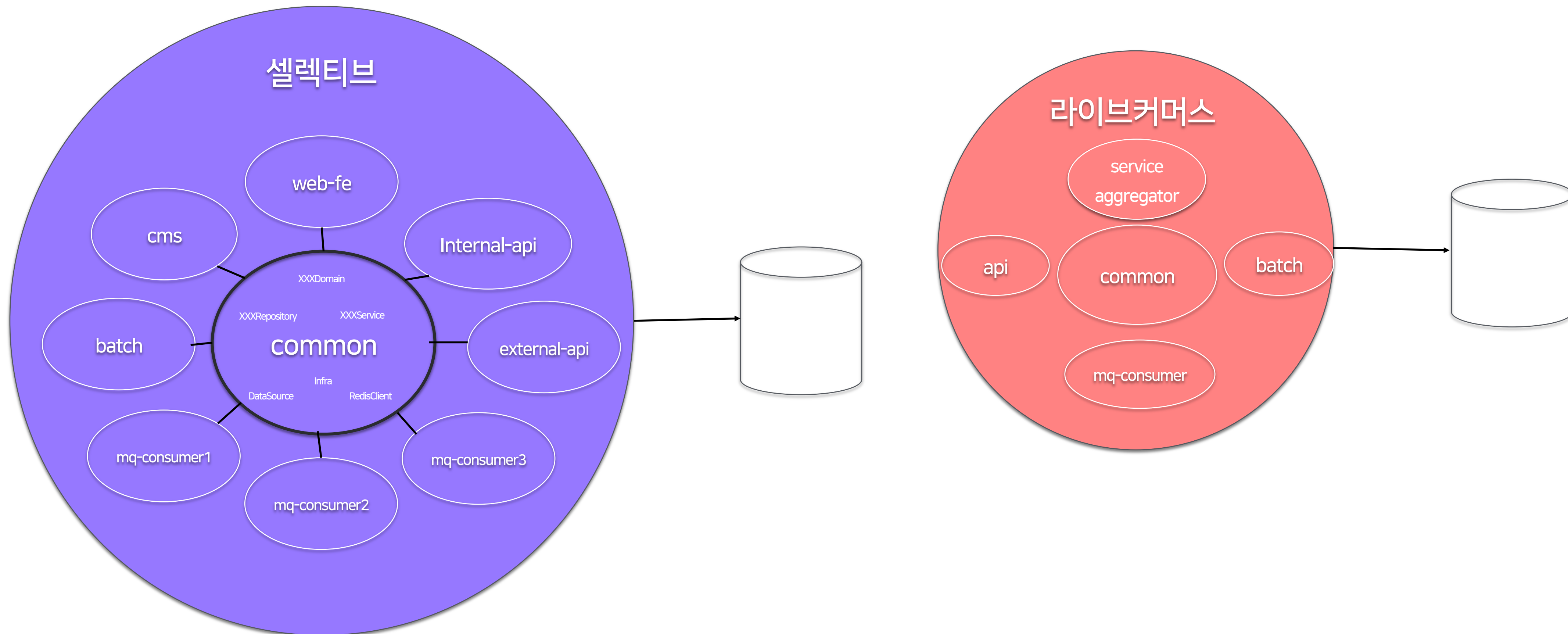
2.1.1 셀렉티브 (초기 시스템)

Common 모듈을 의존하는 멀티 모듈 프로젝트 구조



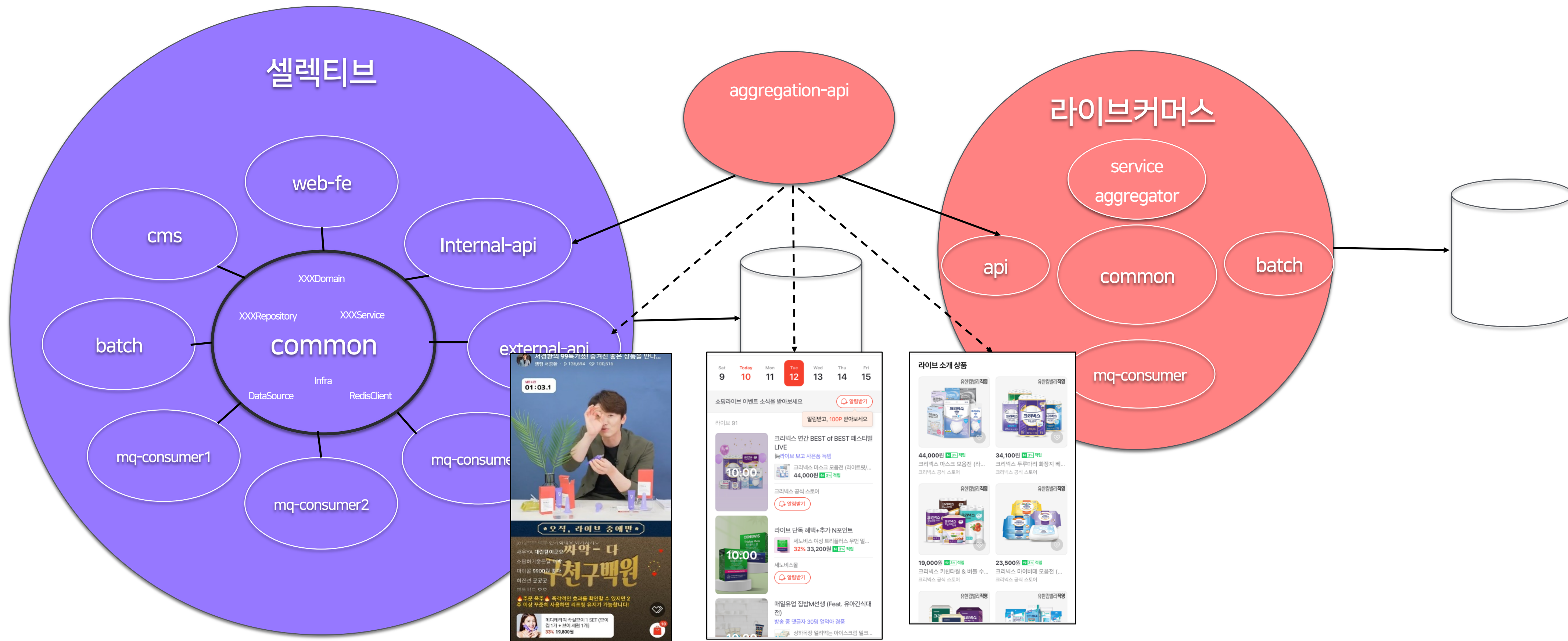
2.1.2 셀렉티브 X 라이브커머스

라이브커머스 도메인/비즈니스를 담당하는
셀렉티브와 유사한 형태의 멀티 모듈 프로젝트 추가 구성



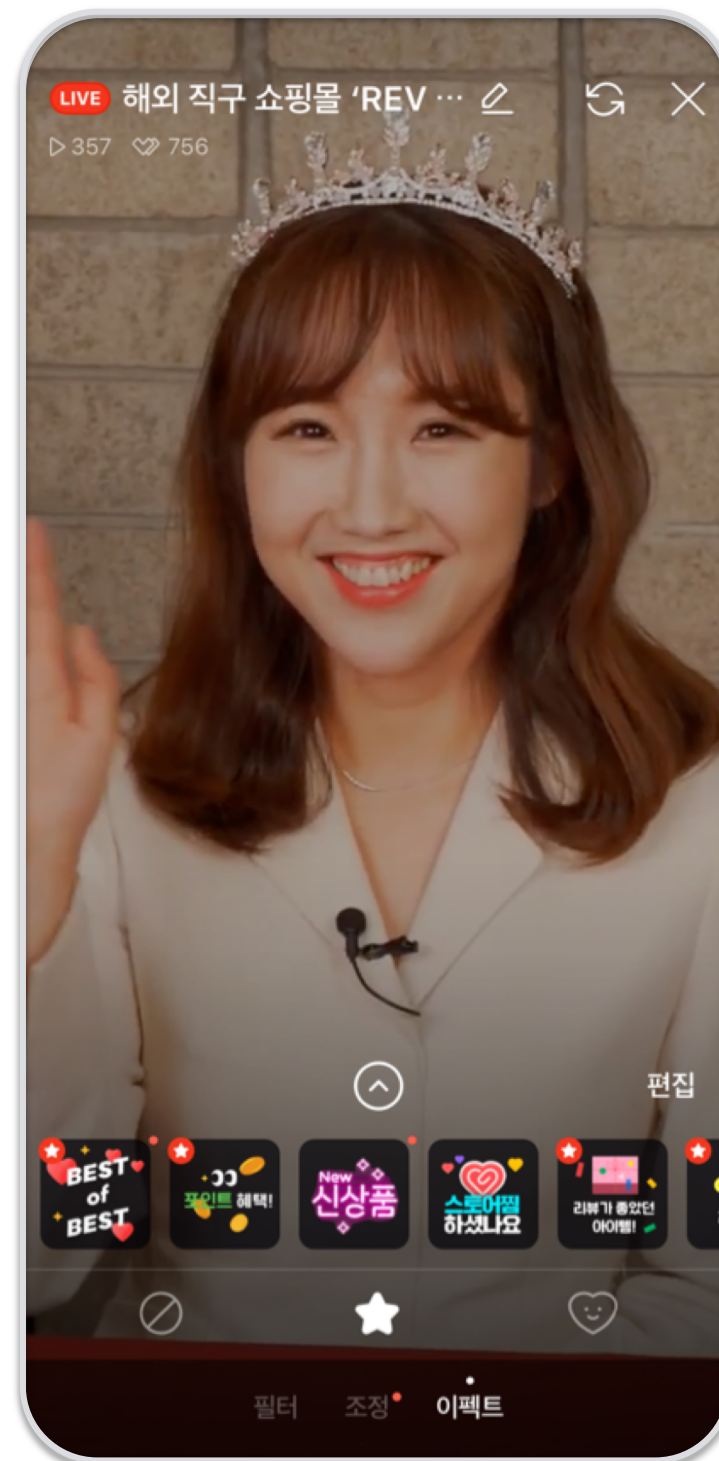
2.1.2 셀렉티브 X 라이브커머스

라이브커머스 도메인/비즈니스를 담당하는
셀렉티브와 유사한 형태의 멀티 모듈 프로젝트 추가 구성



2.2 라이브커머스 주요 Feature

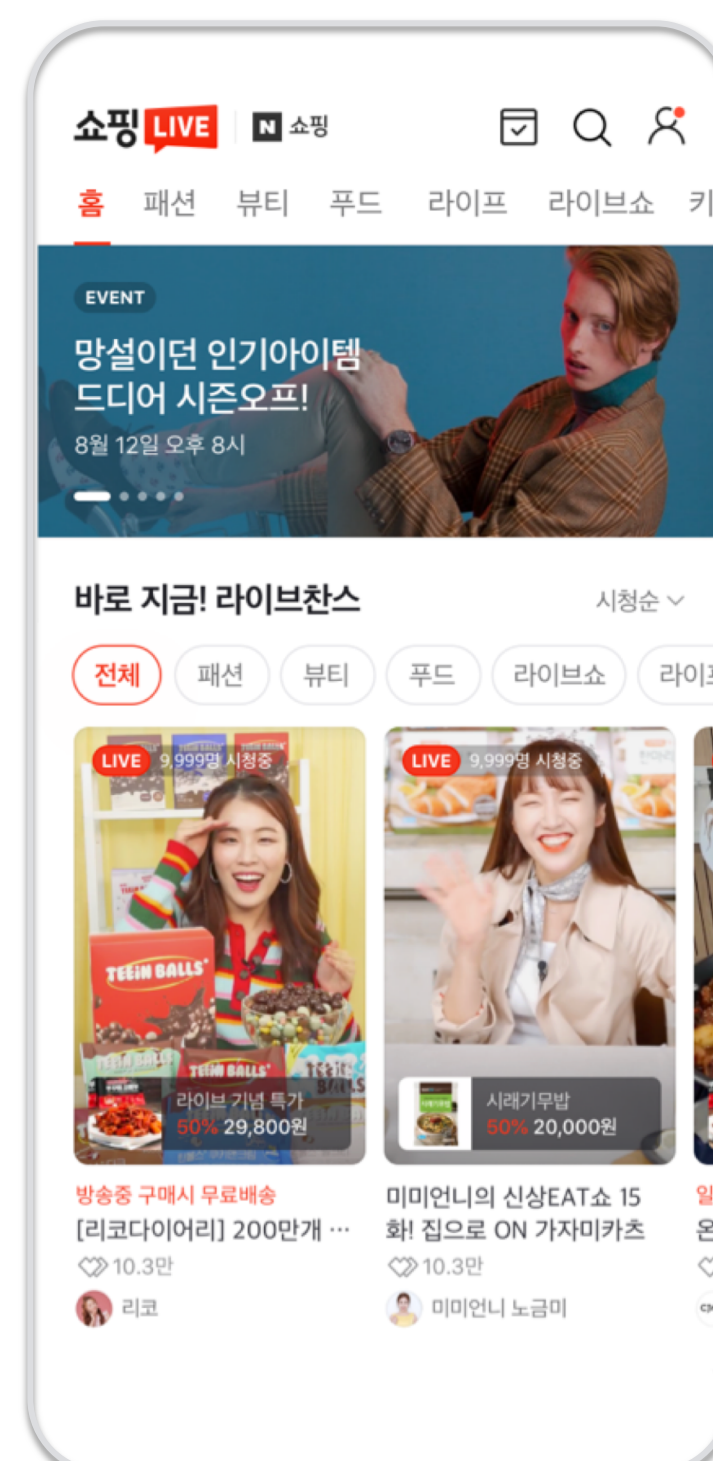
1. 라이브 동영상 송출



2. 라이브커머스 뷰어



3. 방송 전시



4. 방송 관리



2.3 라이브커머스 뷰어 Spec



2.3 라이브커머스 뷰어 Spec

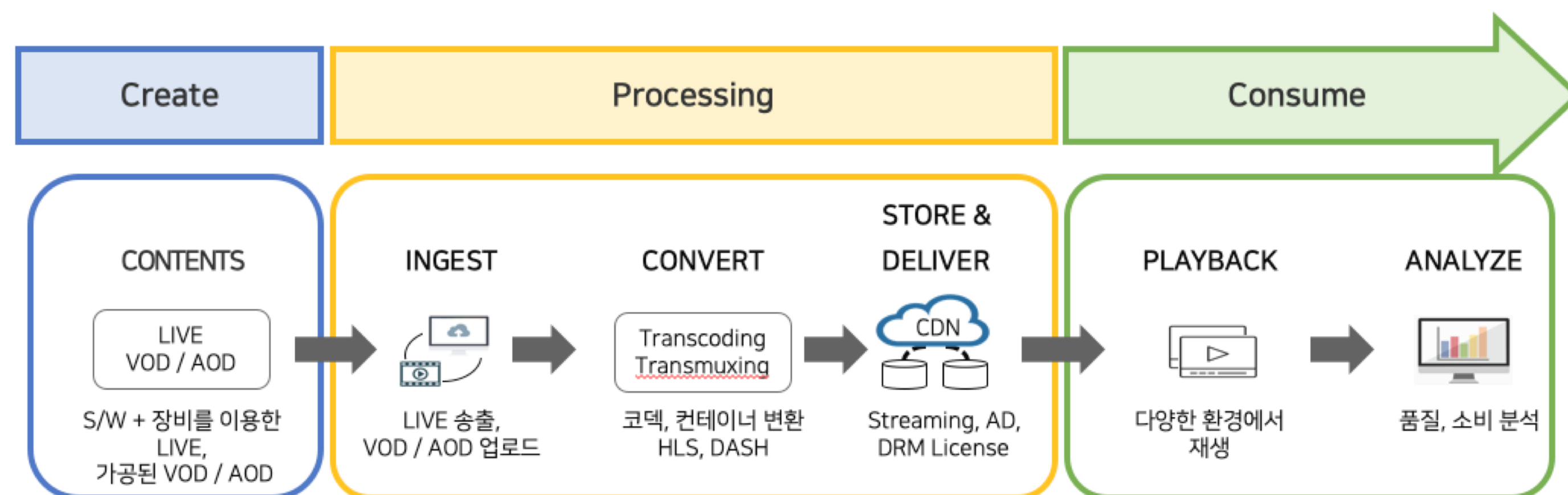


큰 Feature 들을

빠르게 개발하려면..?

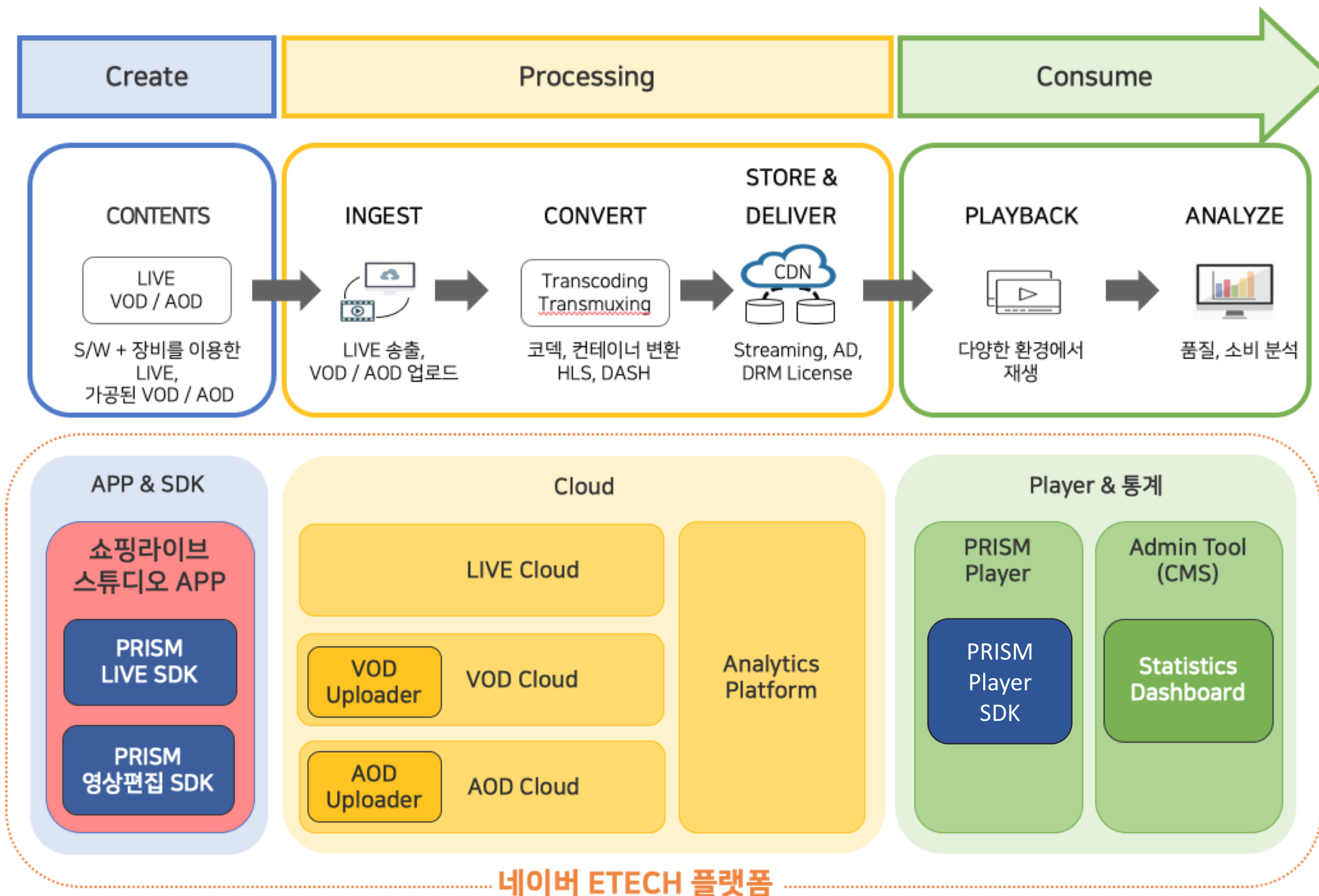
2.4.1 뷰어 - 동영상 Workflow

동영상 송출에서 재생까지의 Workflow



2.4.1 뷰어 - 동영상 Workflow

동영상 송출에서 재생까지의 Workflow



ETECH 동영상 플랫폼 기반으로

빠르게 라이브 송출 도구 및 재생 기능 개발

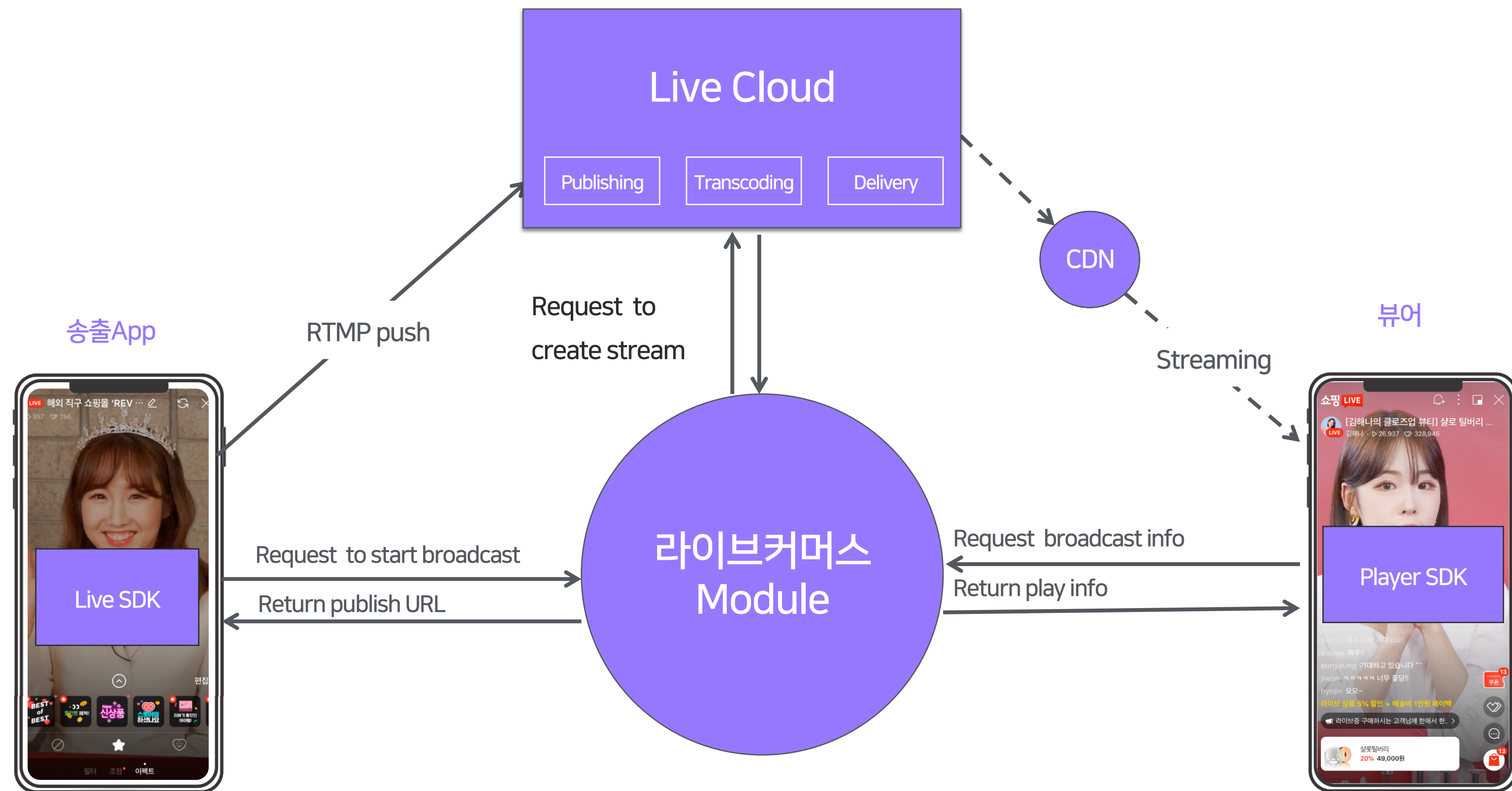
송출 Server - Live Cloud

송출 Client - Live SDK

재생 Client - Player SDK

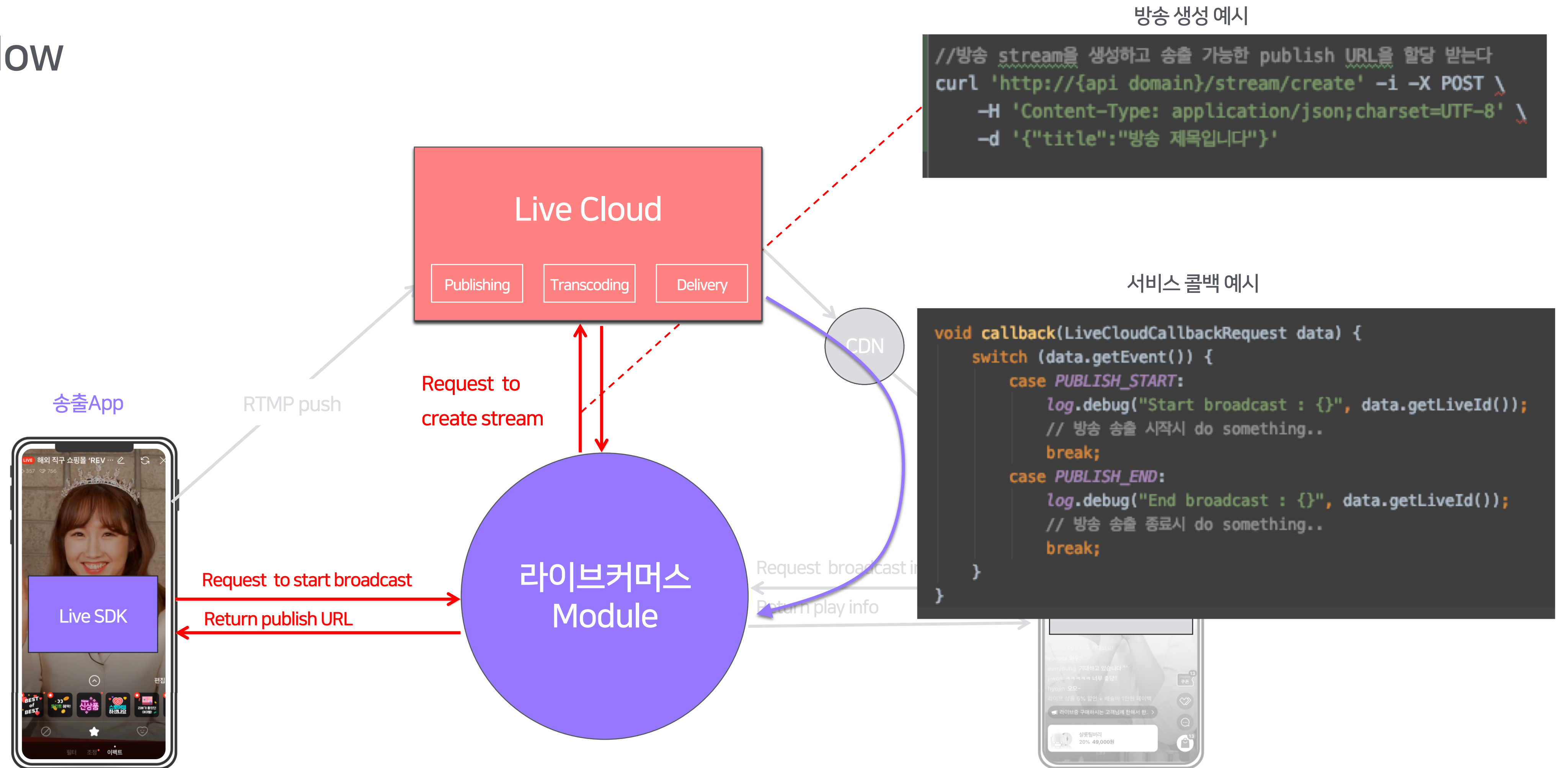
2.4.2 뷰어 - 라이브 송출 및 재생

서비스 연동 Flow



2.4.2 뷰어 - 라이브 송출 및 재생 > 생성

서비스 연동 Flow

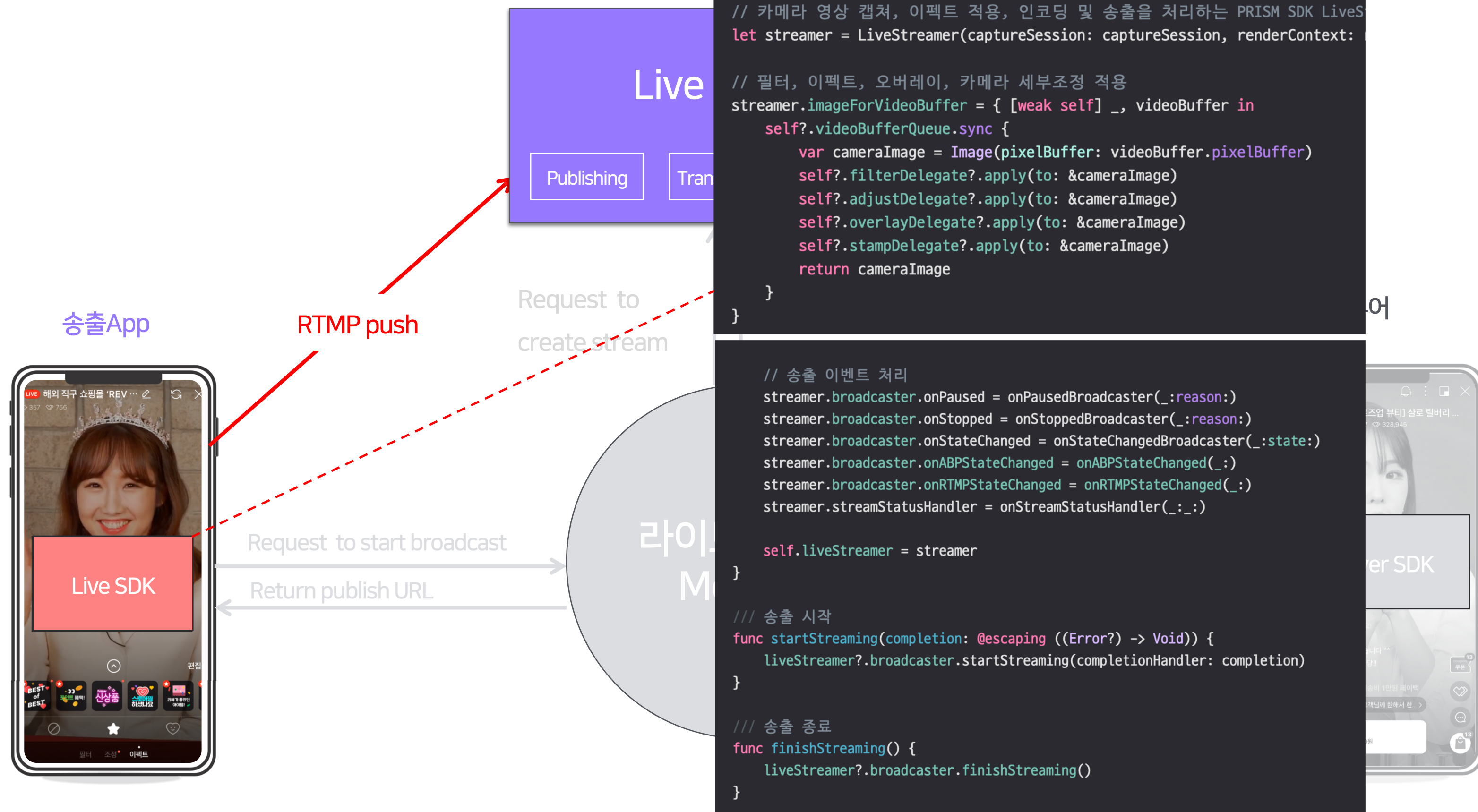


2.4.2 뷰어 - 라이브 송출 및 재생 > 송출

카메라 원본 영상 획득 → 이펙트 적용 → 인코딩 → 송출

SDK class의 instance를 생성 및 설정

송출 SDK 적용 예시

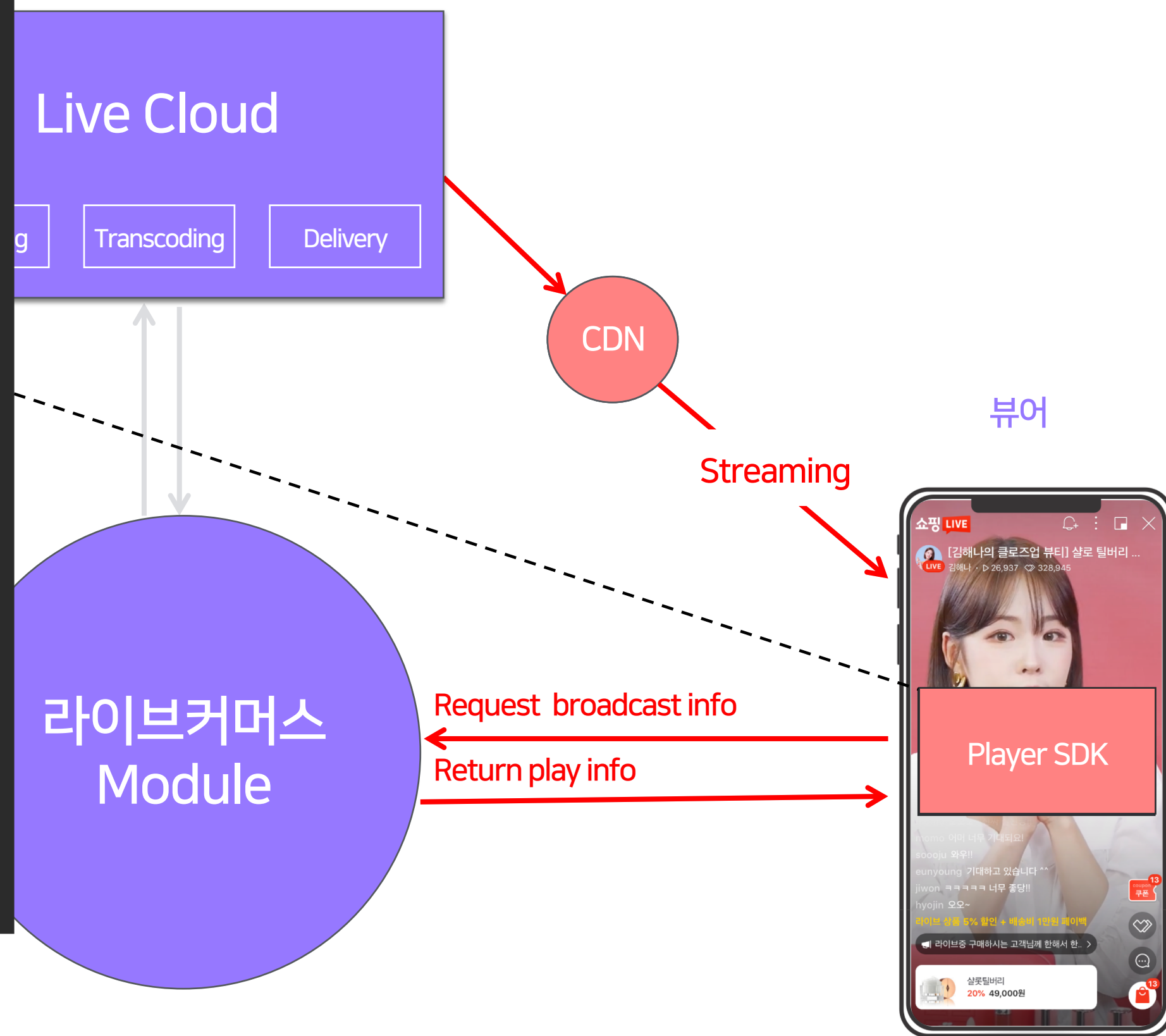


2.4.2 뷰어 - 라이브 송출 및 재생 > 재생

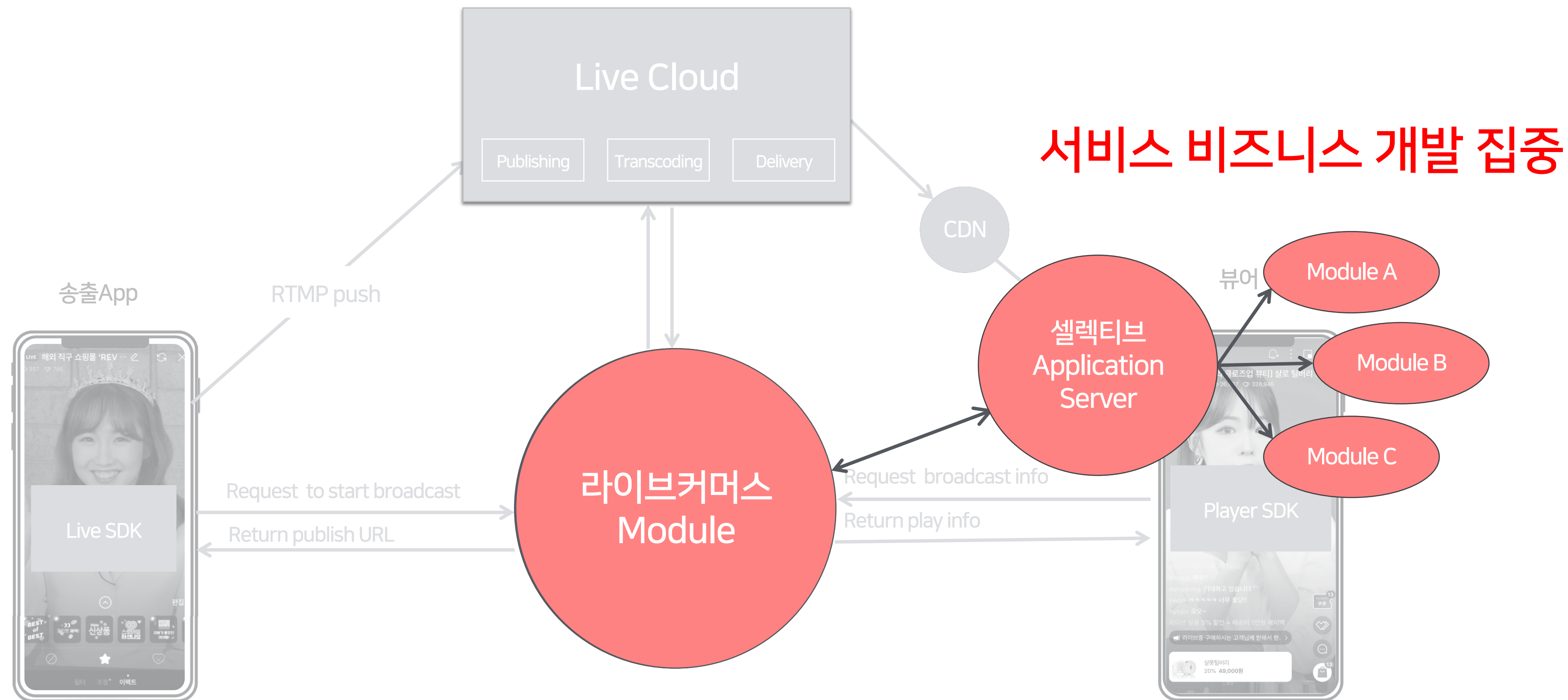
라이브 클라우드 재생 정보(playback) 사용하여 재생

뷰어 예제 코드: playback 처리 예시

```
fun ShoppingLivePlayerInfo.toSource(): Source? {
    var startTime: Long? = null
    val source = when {
        playback.isNotBlank() -> {
            // 라이브 클라우드 재생 정보를 통째로 전달받아 재생하는 케이스
            TextSource(playback)
        }
        playUrl.isNotBlank() -> {
            // URL 로 직접 재생하는 케이스
            startTime = startTimeMillis
            UriSource(
                title = title,
                url = playUrl,
                live = !replay
            )
        }
        else -> {
            return null
        }
    }
}
```

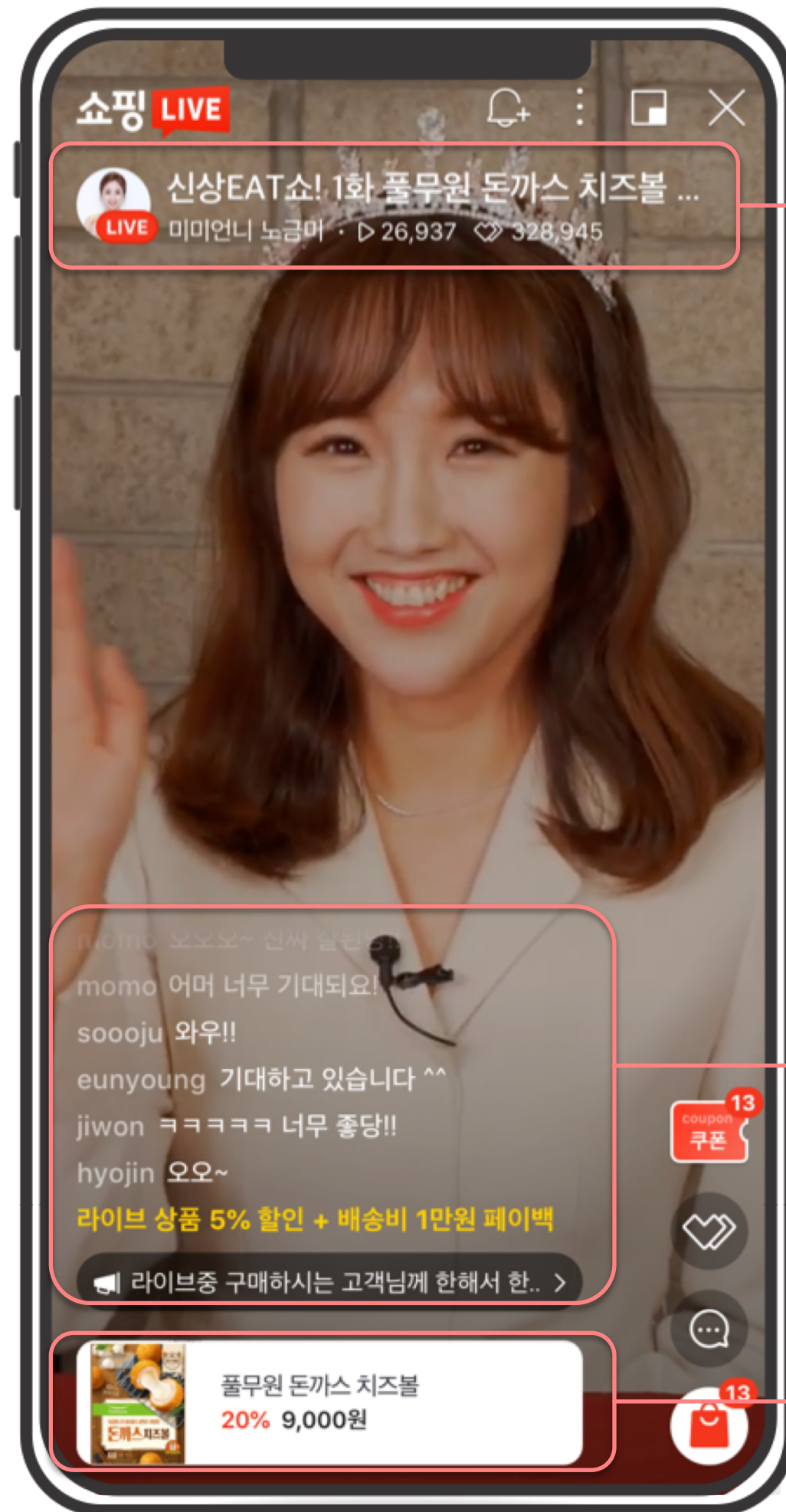


2.4.2 뷰어 - 라이브 송출 및 재생



2.5 뷰어 - 채팅 (실시간 메시징)

실시간 메시징 처리 대상



방송정보
시청 수
좋아요 수

채팅/공지

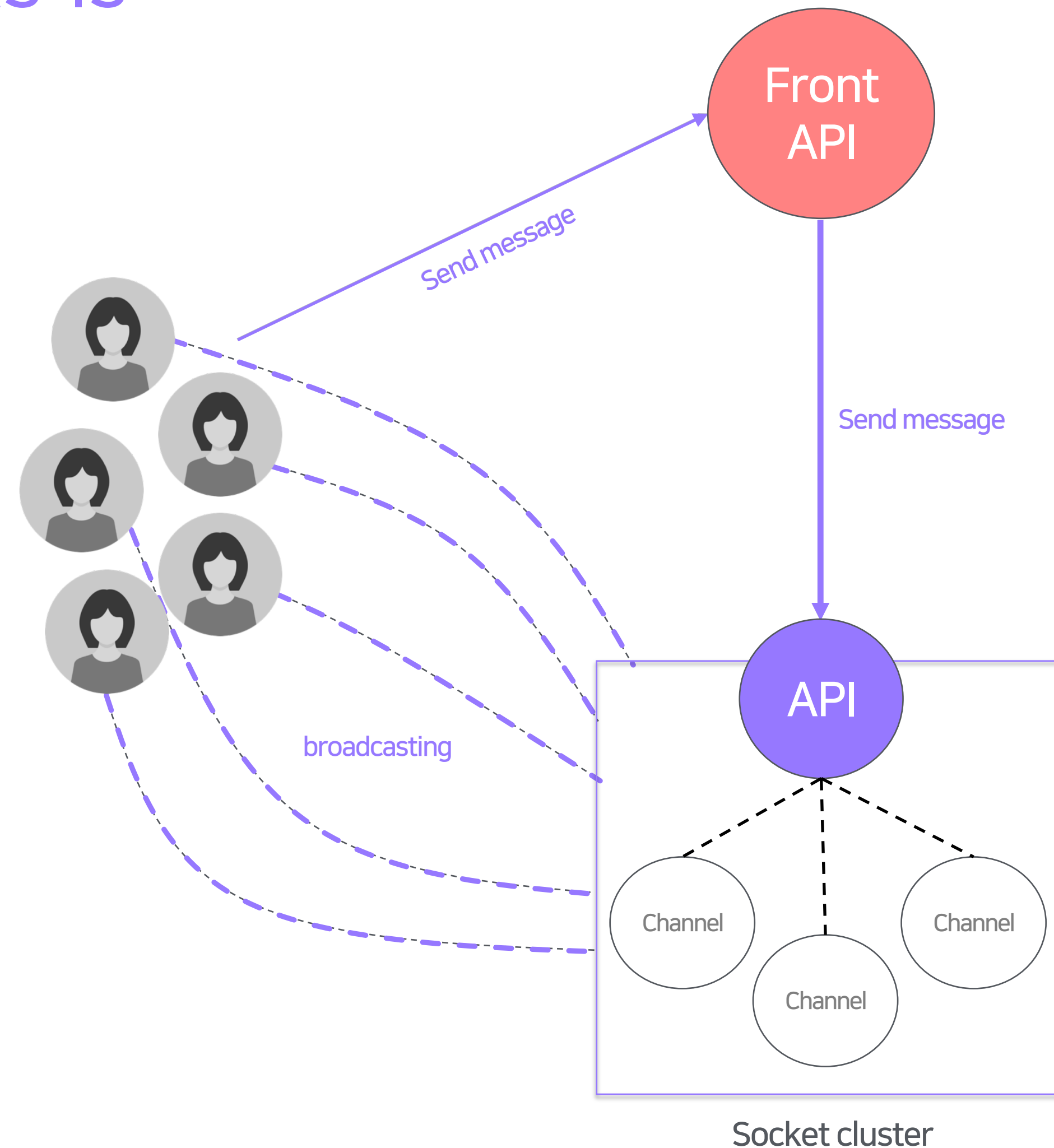
상품 정보

- socket cluster server 필요

session-io

2.5 뷰어 - 채팅 (실시간 메시징)

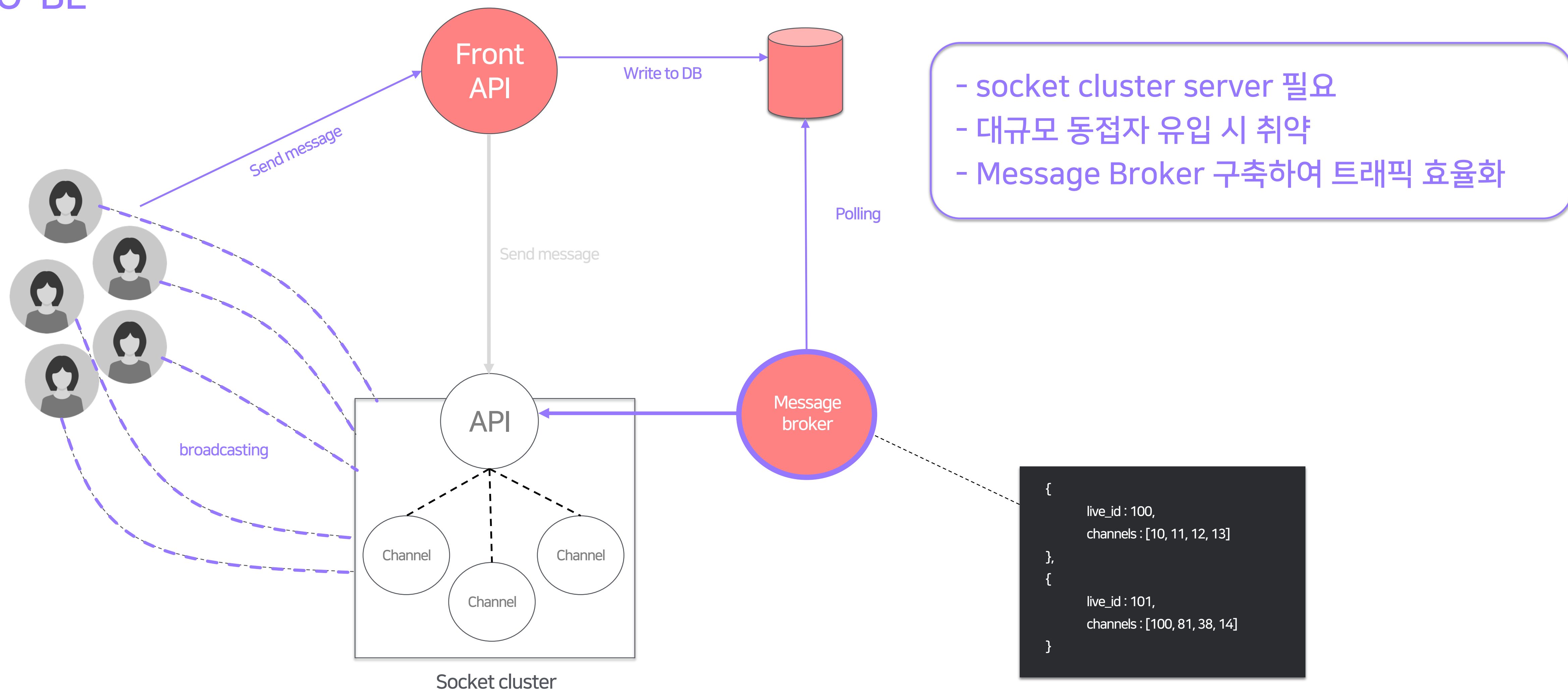
AS-IS



- socket cluster server 필요
- 대규모 동접자 유입 시 취약

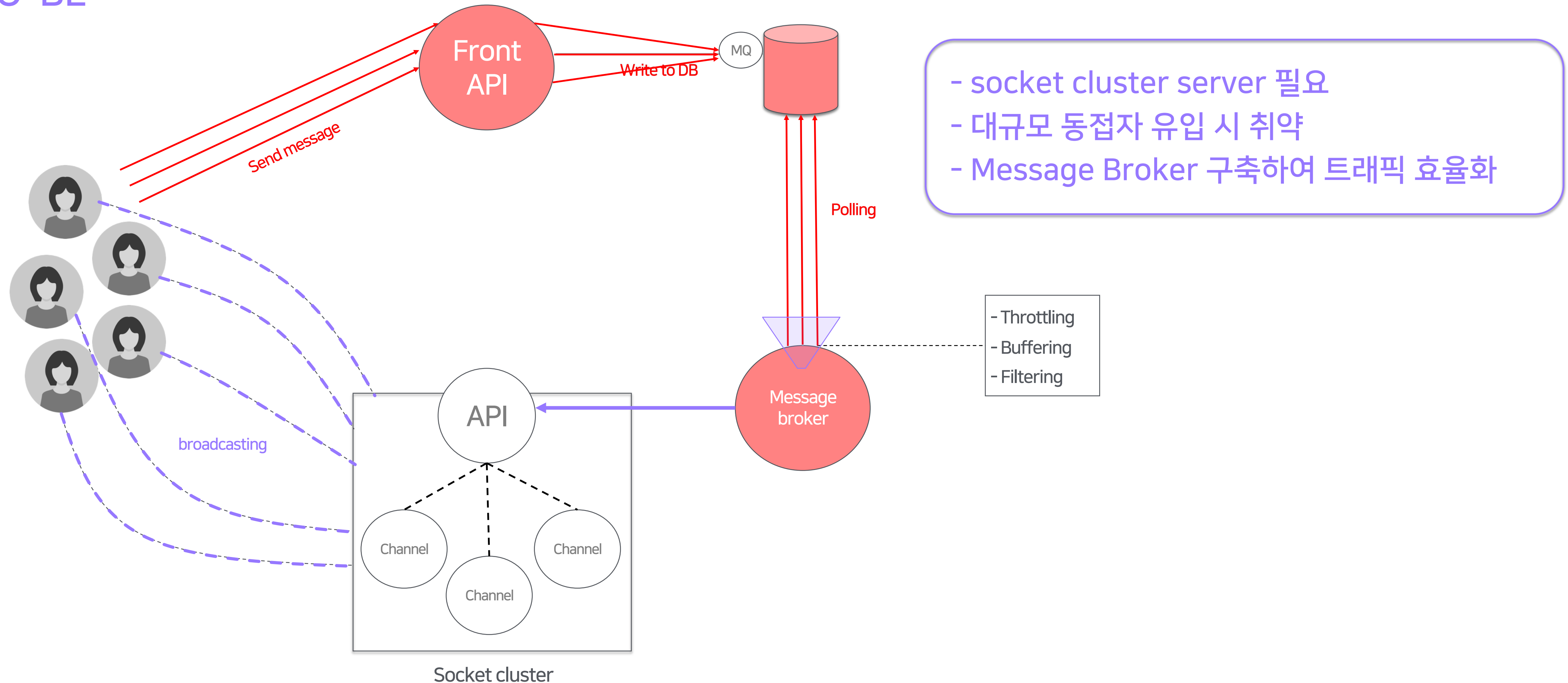
2.5 뷰어 - 채팅 (실시간 메시징)

TO-BE



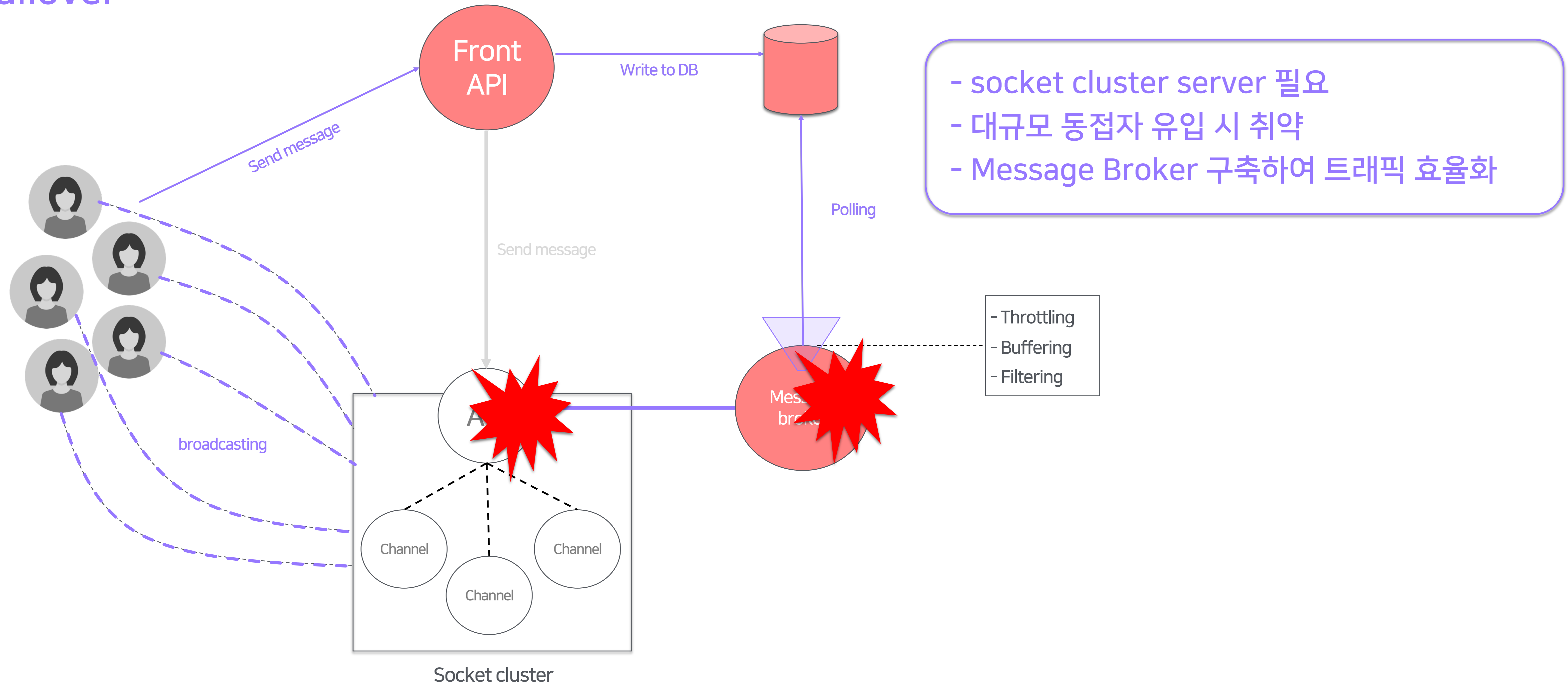
2.5 뷰어 - 채팅 (실시간 메시징)

TO-BE



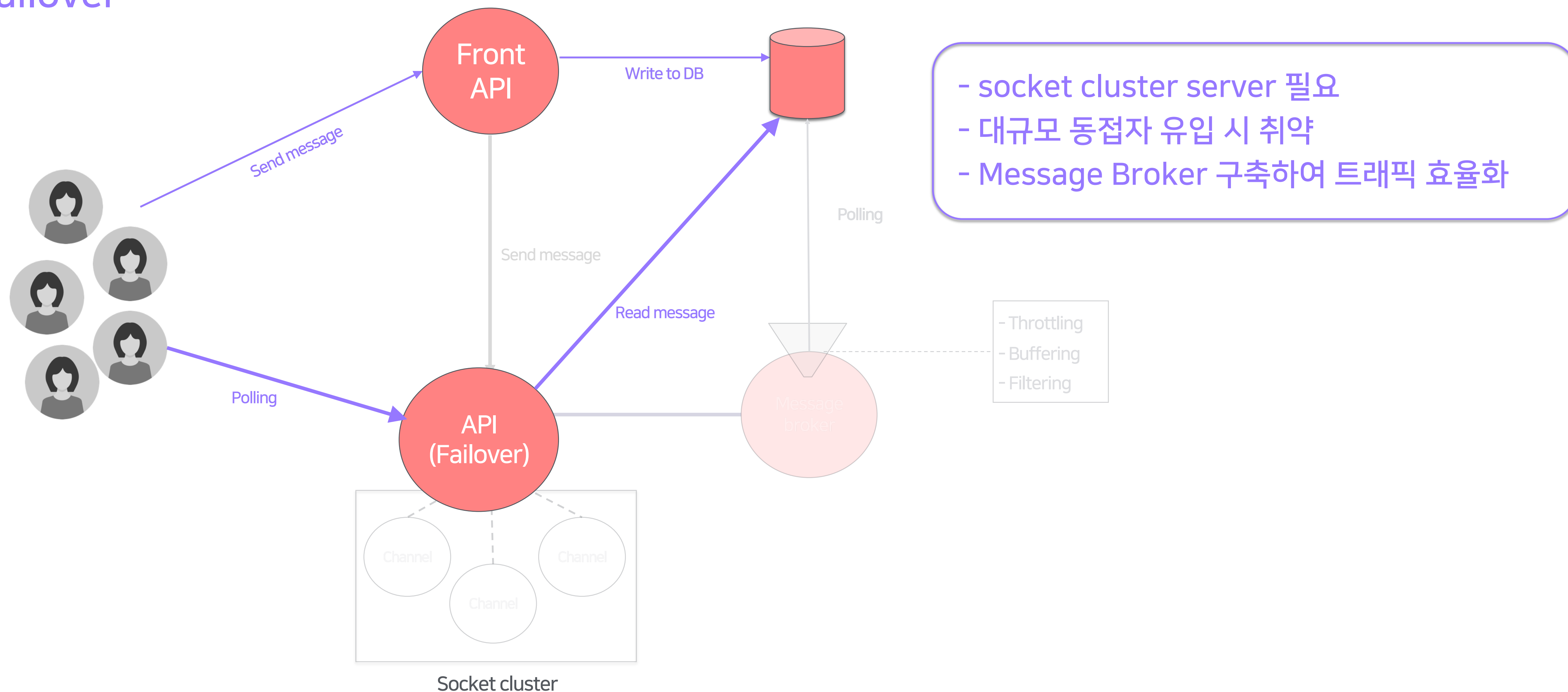
2.5 뷰어 - 채팅 (실시간 메시징)

Failover



2.5 뷰어 - 채팅 (실시간 메시징)

Failover



2.5 뷰어 - 채팅 (실시간 메시징)

장애 예방 도구

Message broker 설정값 조정

자수	
댓글 폴링 주기	
좋아요 폴링 주기	
누적 시청자수 폴링 주기	
그룹카운트 폴링 주기	
댓글 강제 폴링 설정값	
좋아요 강제 폴링 설정값	
누적시청자수 강제 폴링 설정값	
그룹라이브 카운트 강제 폴링 설정값	

Polling 모드 세부 설정

폴링 여부	☐
폴링시 댓글 폴링 주기	
폴링시 extra count 폴링 주기	

3. 탄탄하게 만들기

초기 시스템

라이브커머스 주요 피쳐들을 모듈화 하여 빠르게 개발했지만 (라이브커머스 Module)

서비스가 성장할수록

많은 피쳐들이 생기고

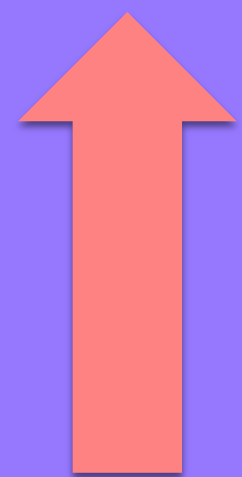
트래픽이 증가하고

기존 서비스 본진과의 (선택티브, 잼라이브) 결합도(Coupling) 증가

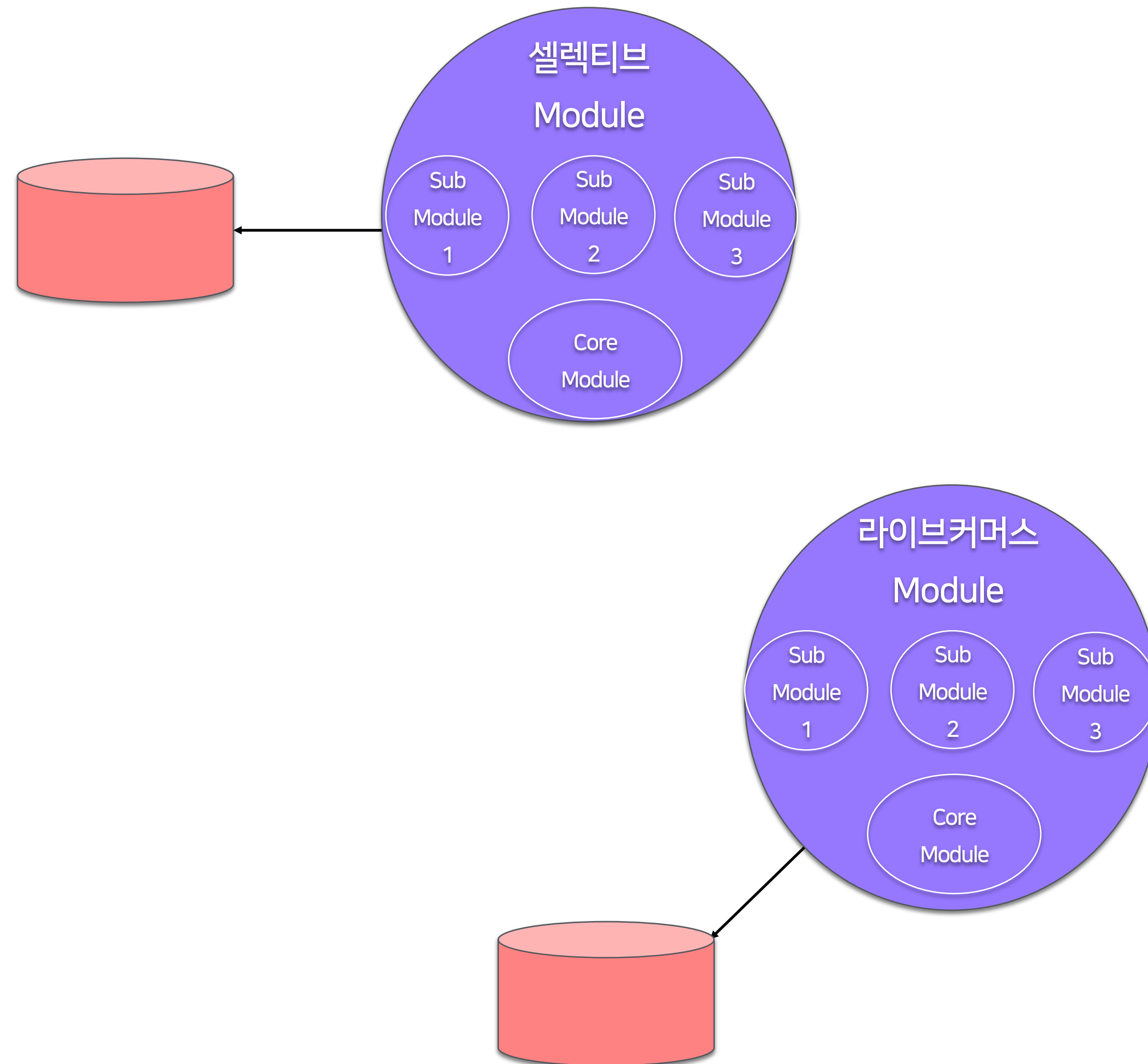
복잡도

휴먼 에러 발생 가능성

장애 리스크



3.1 초기 시스템 구성

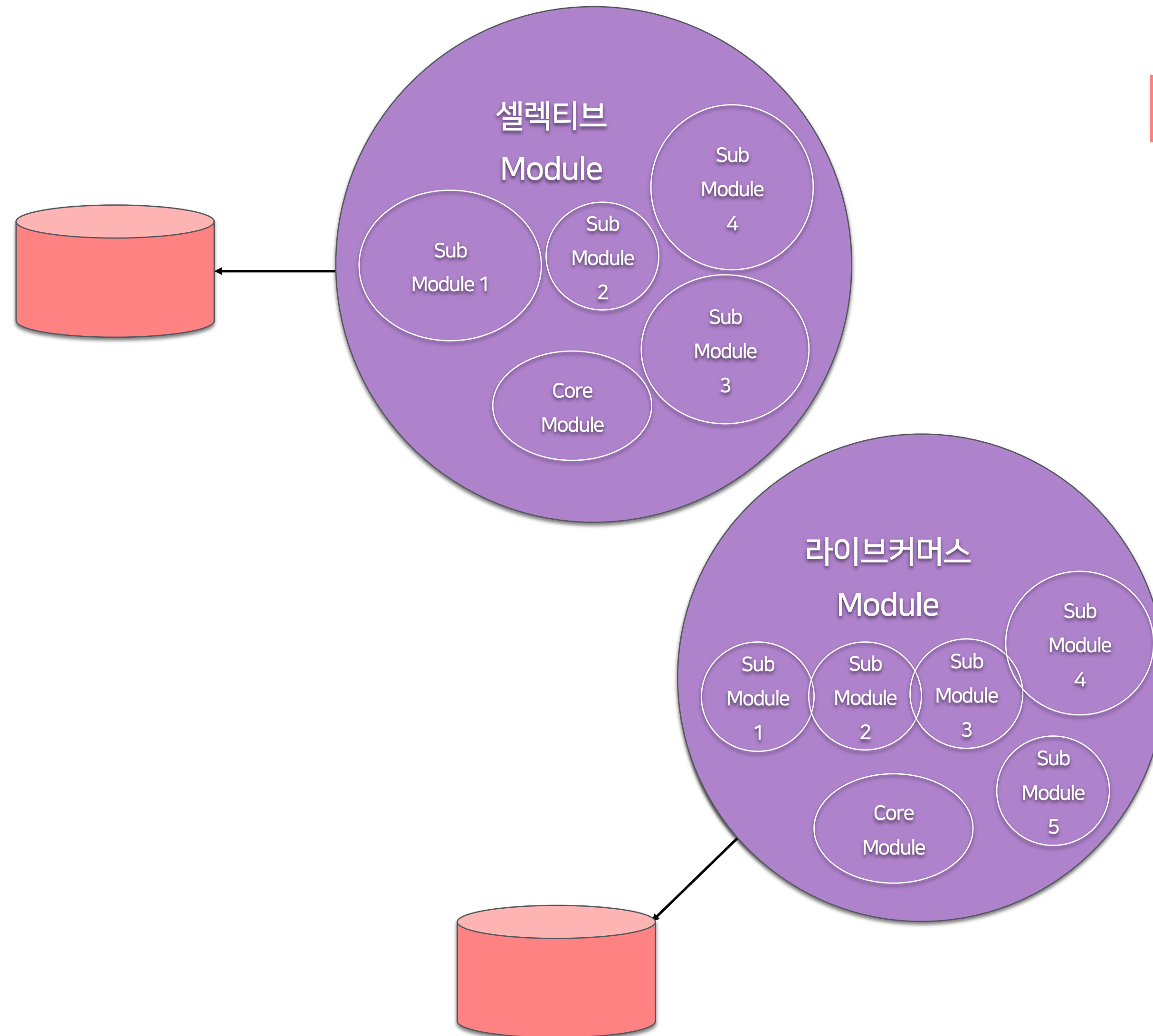


초기에는

선택티브 서비스 Module <-> 라이브커머스 Module 간

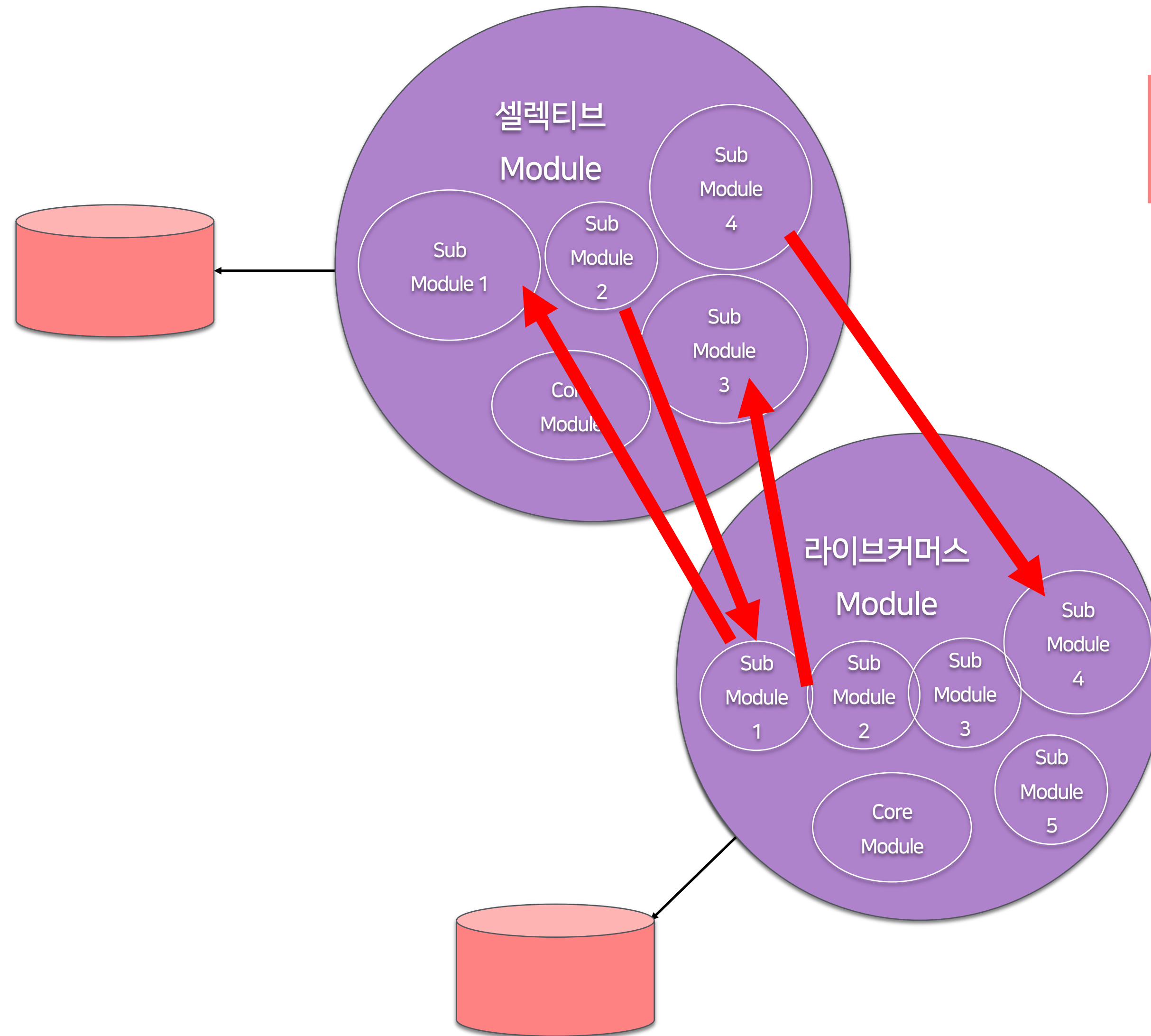
- 의존도가 높지 않고
- Sub Module (API, FE, Batch 등) 수가 많지 않음
- 코드량도 상대적으로 적음
- 개발자가 많지 않음

3.1 초기 시스템 구성

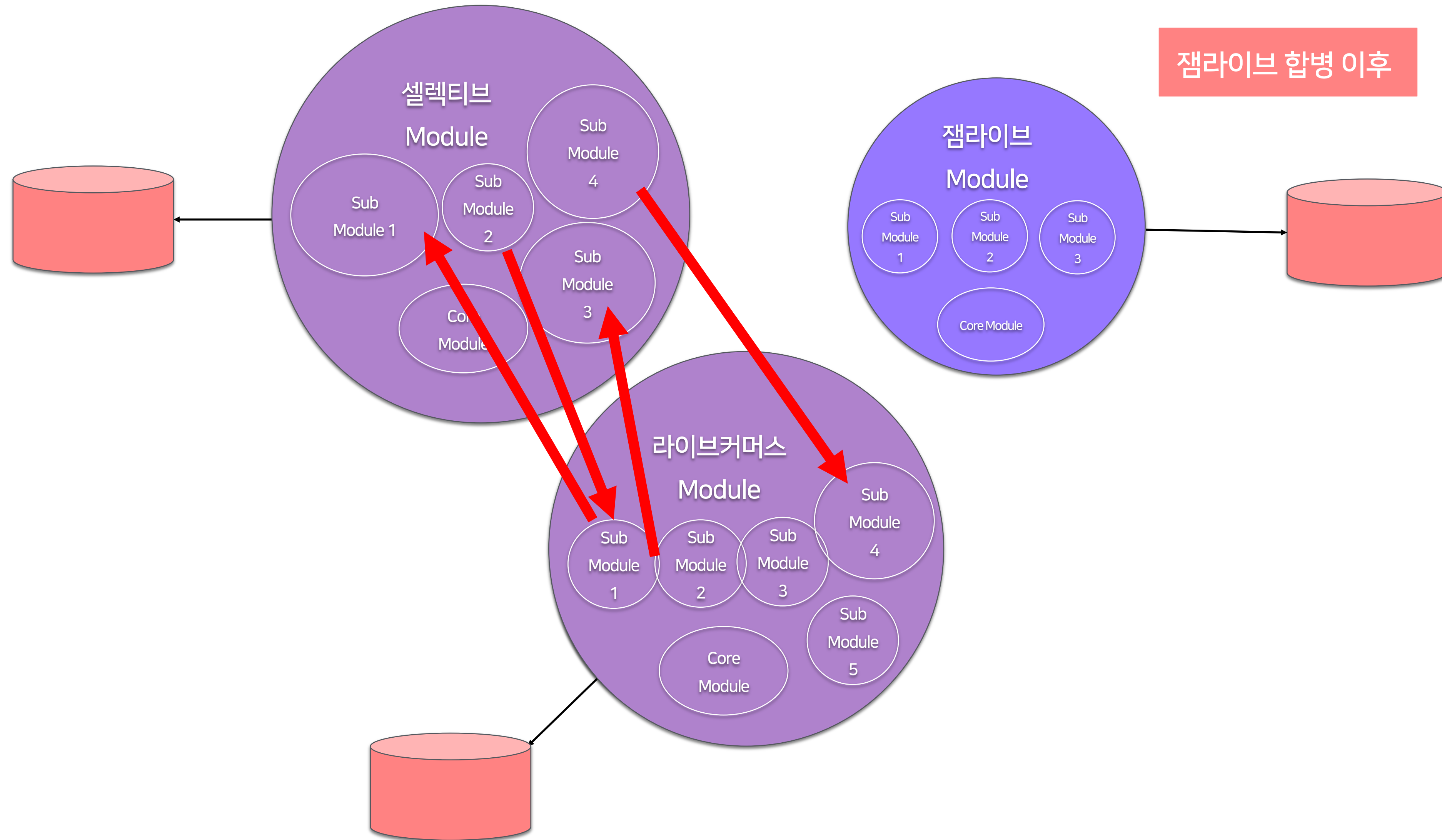


피처가 많아지면서 Module들은 점점 비대해지고..

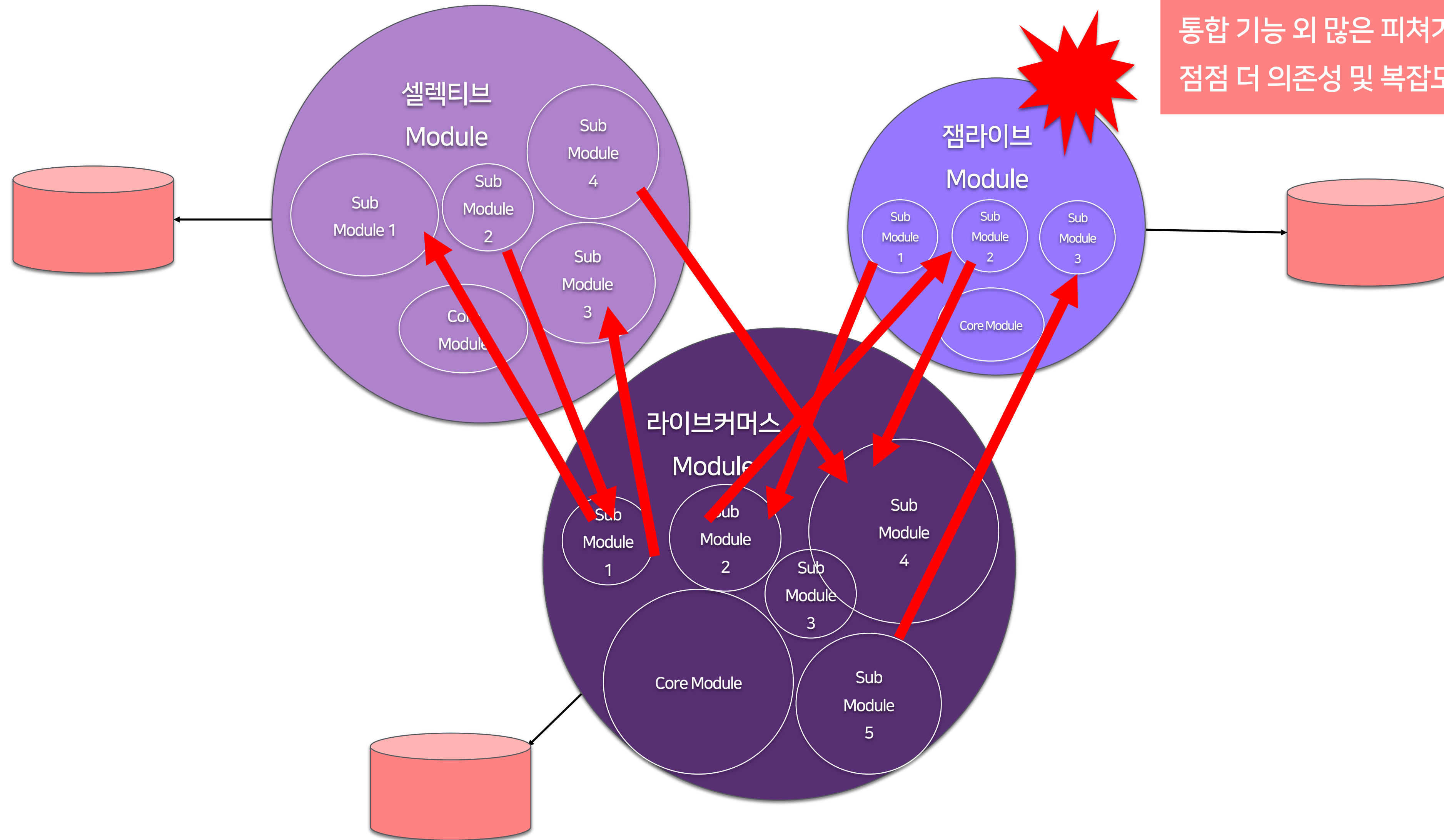
3.1 초기 시스템 구성



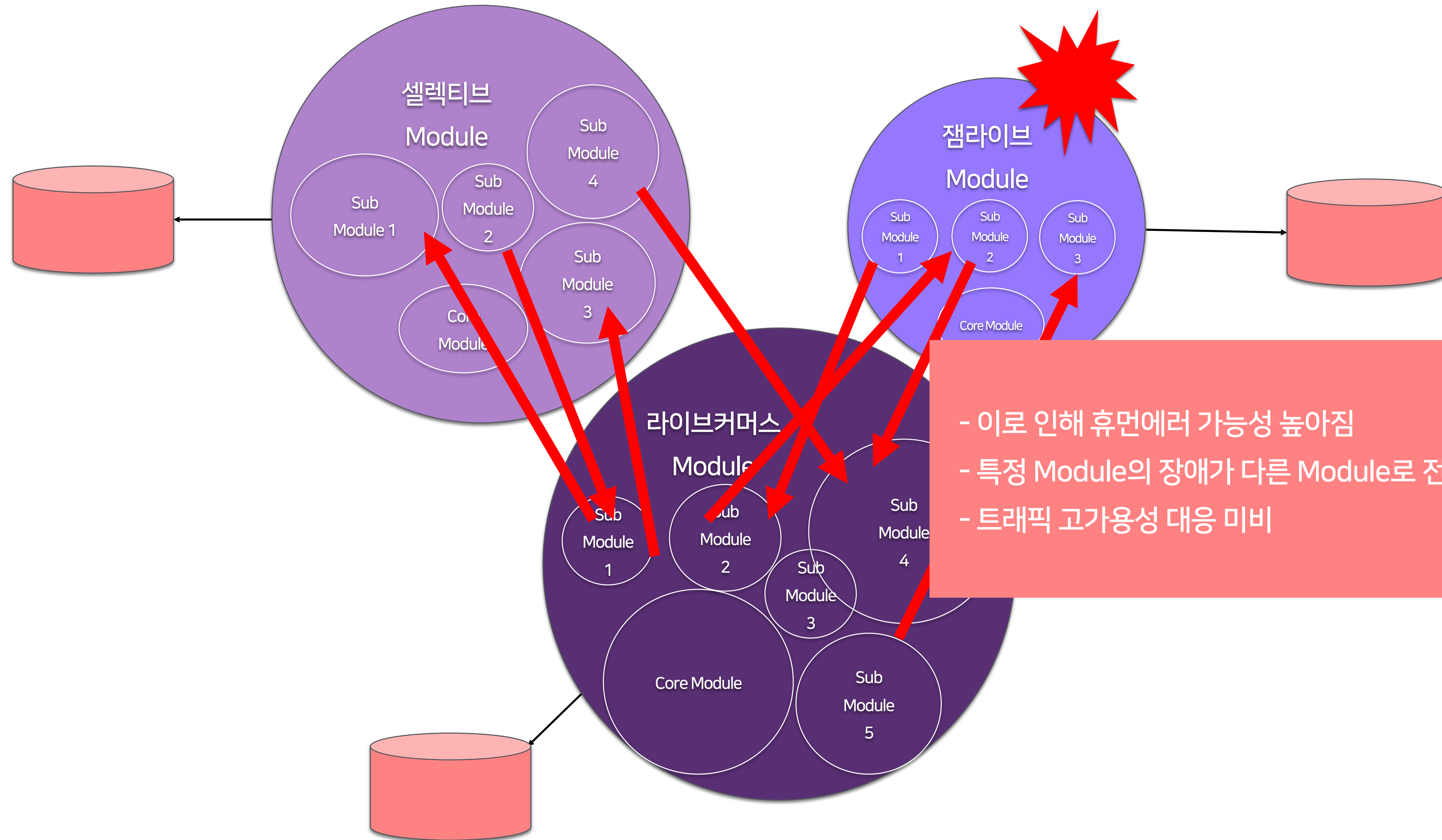
피처가 많아지면서 Module들은 점점 비대해지고..
모듈간 의존도가 증가하기 시작



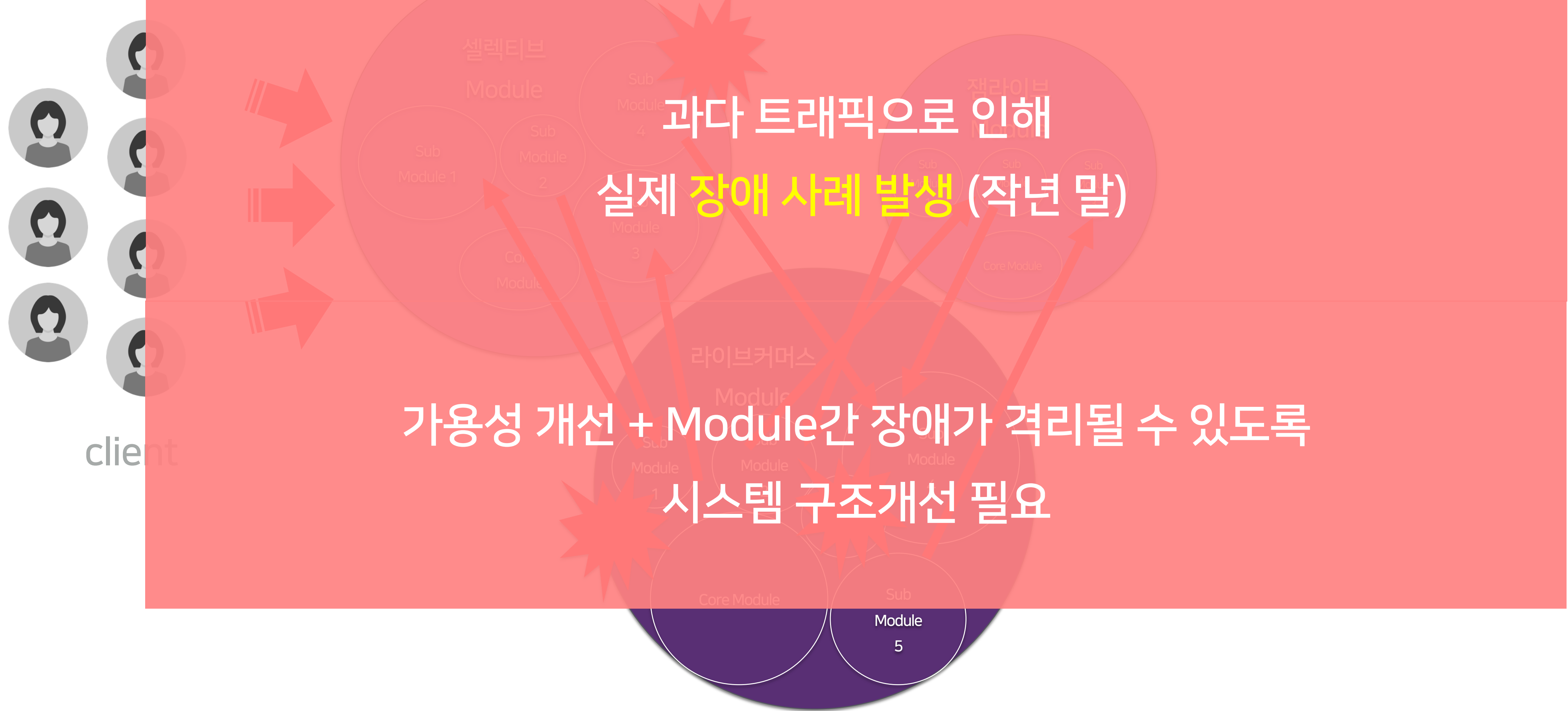
3.1 초기 시스템 구성



3.1 초기 시스템 구성



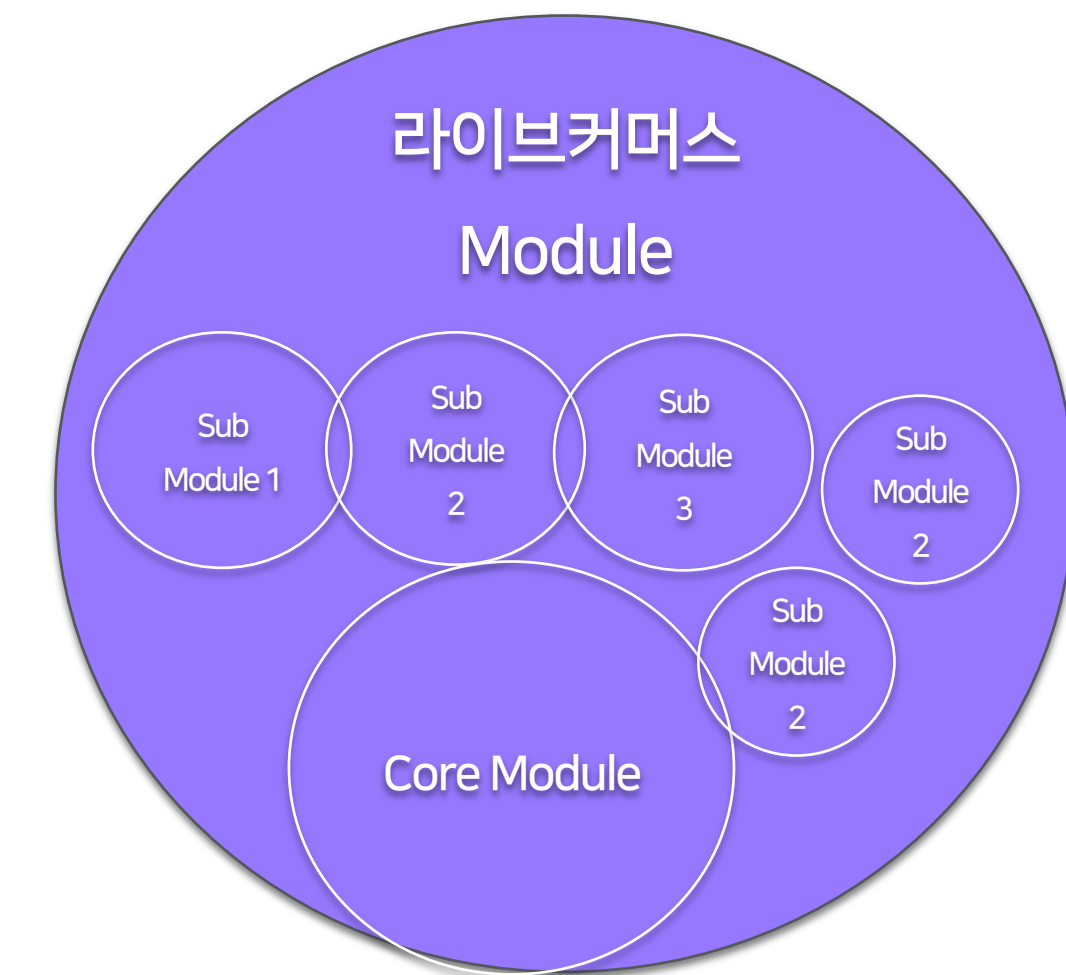
- 이로 인해 휴먼에러 가능성 높아짐
- 특정 Module의 장애가 다른 Module로 전이될 위험성 높아짐
- 트래픽 고가용성 대응 미비



3.2 시스템 구조 개선

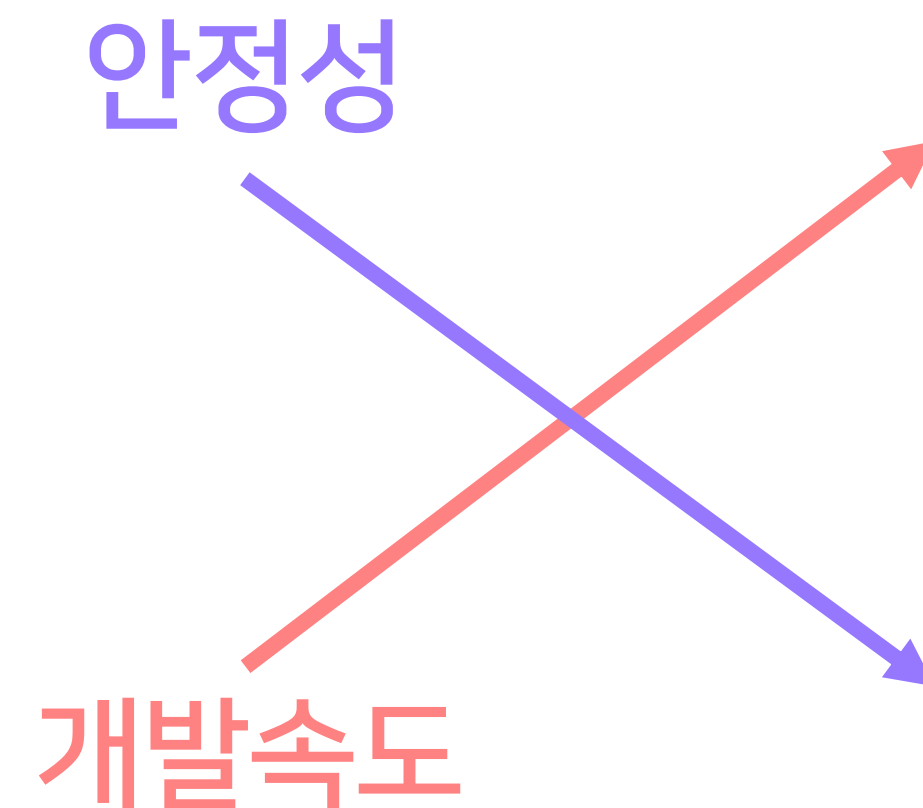
필요한 이유

- 프라임 타임에 시스템 장애 발생시 피해 규모 상당
 - ↳ 쇼호스트 섭외 비용, 스튜디오 대관료, 인건비 등 큰 피해 발생
 - => 장애가 발생하면 안된다
- 커진 개발팀 규모
 - ↳ 선택티브 라이브 초기 대비 인원 2배 증가
 - => 개발 / 협업 방식 변경 필요



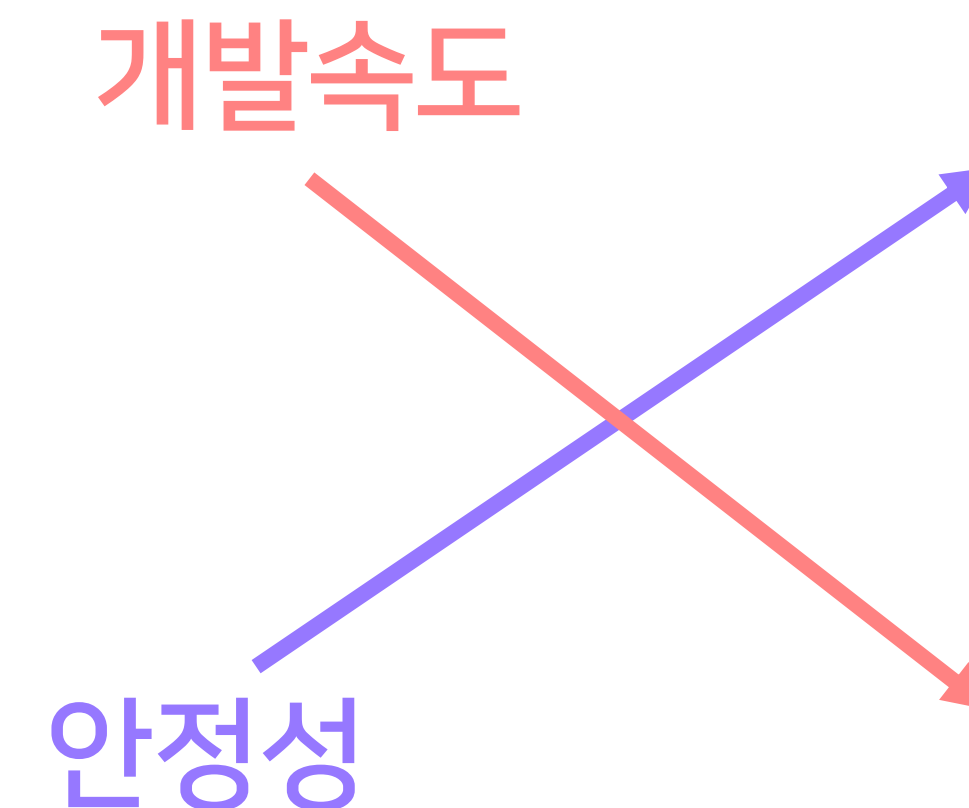
3.2 시스템 구조 개선 - 개발속도 vs 안정성

고민



“서비스가 고속 성장중이라
개발이 늦어지면 안되는데”

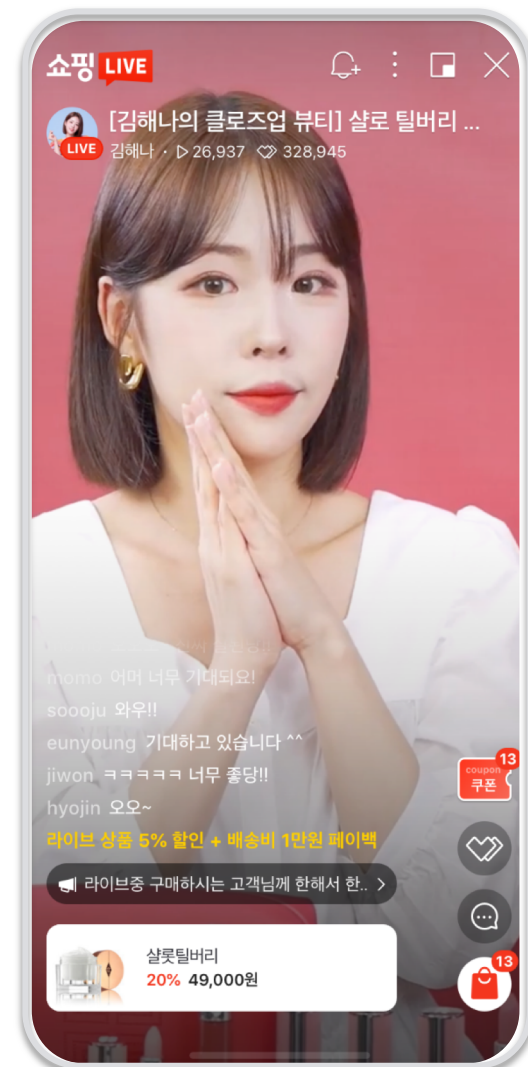
VS



“이럴때일수록 서비스 신뢰성 & 평판이 중요한데
개발이 좀 늦어지면 어때”

3.2 시스템 구조 개선 - 어디부터?

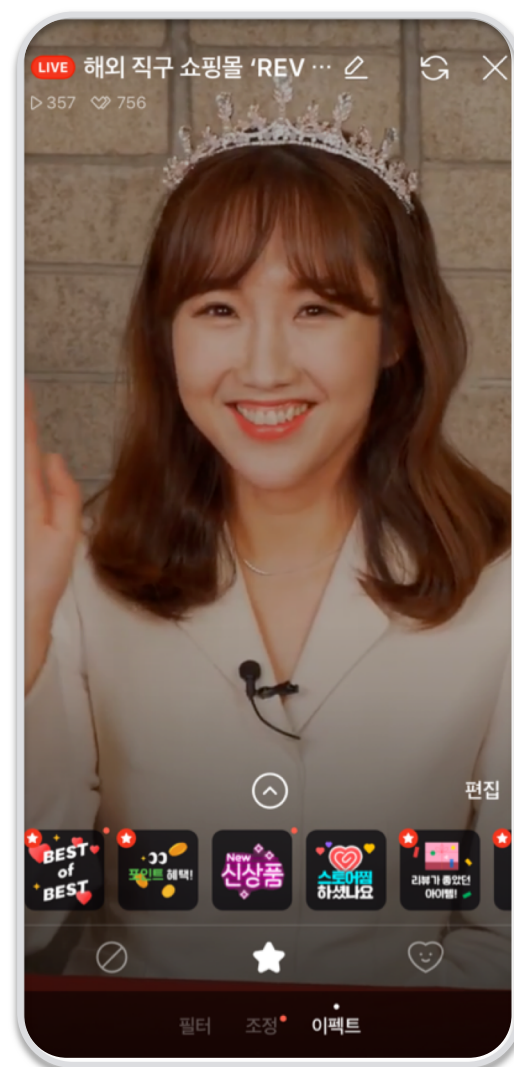
뷰어



실시간
뷰어

VOD
뷰어

송출

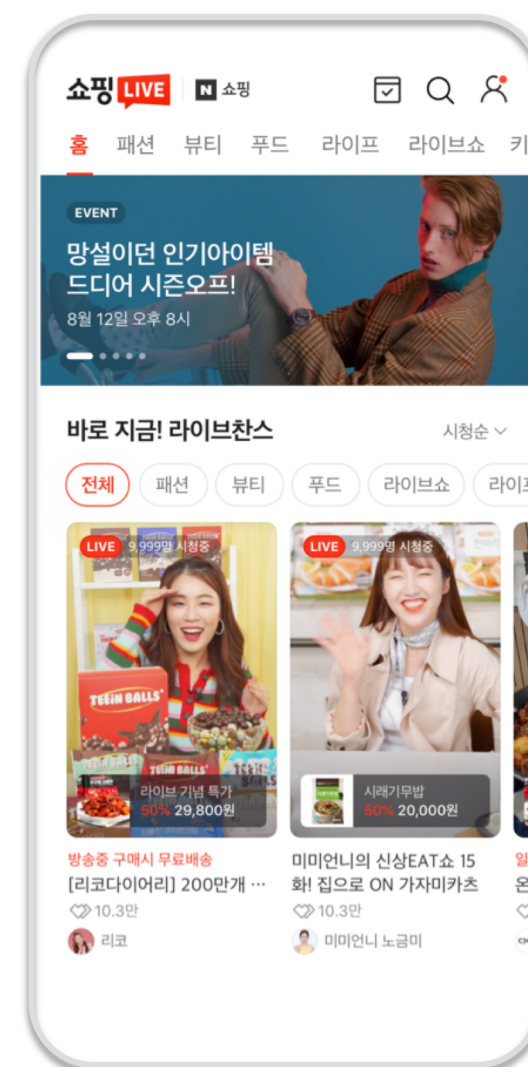


쇼핑
라이브
App

판매자
센터
App

Prism
App

서비스



홈

MY

캘린더

외부
제공



판

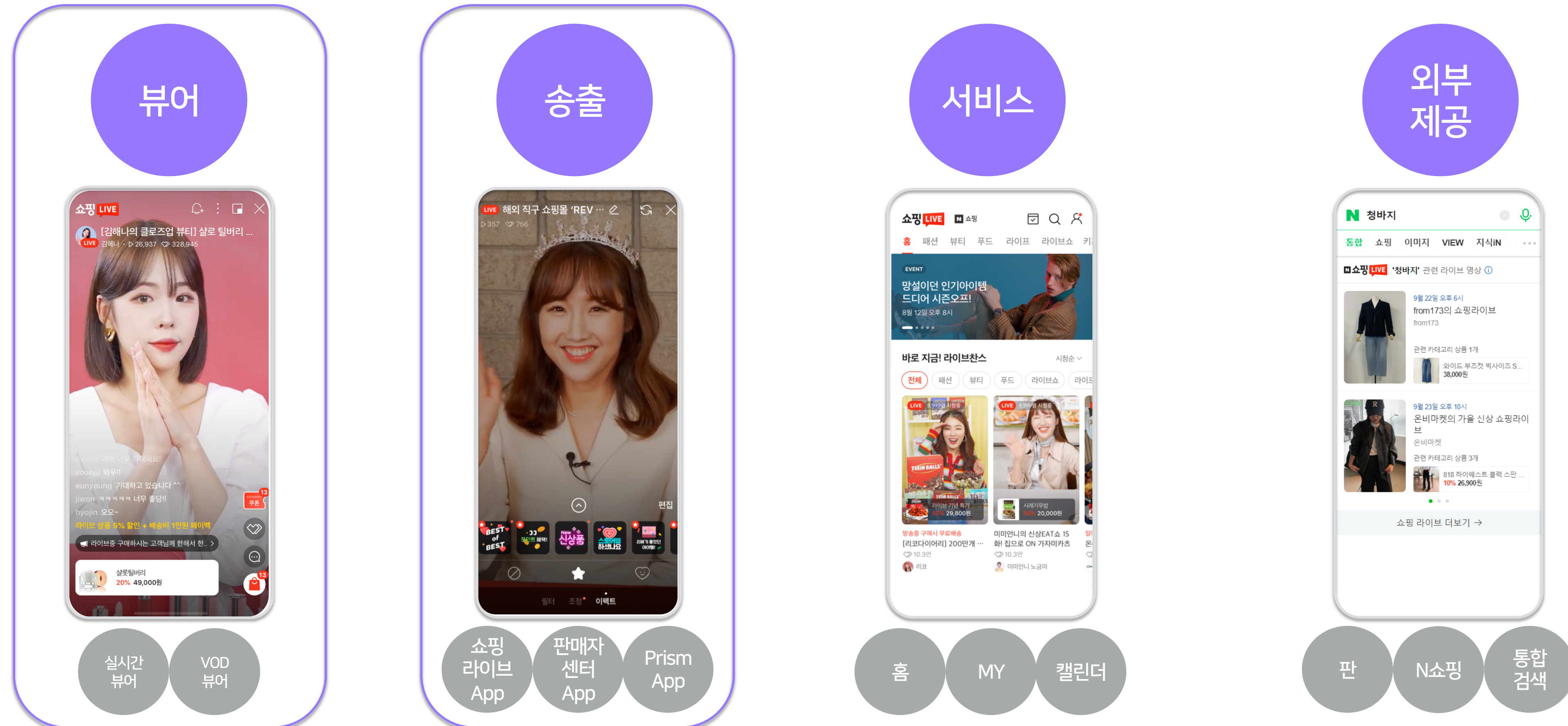
N쇼핑

통합
검색

3.2 시스템 구조 개선 - 어디부터?

Step 1

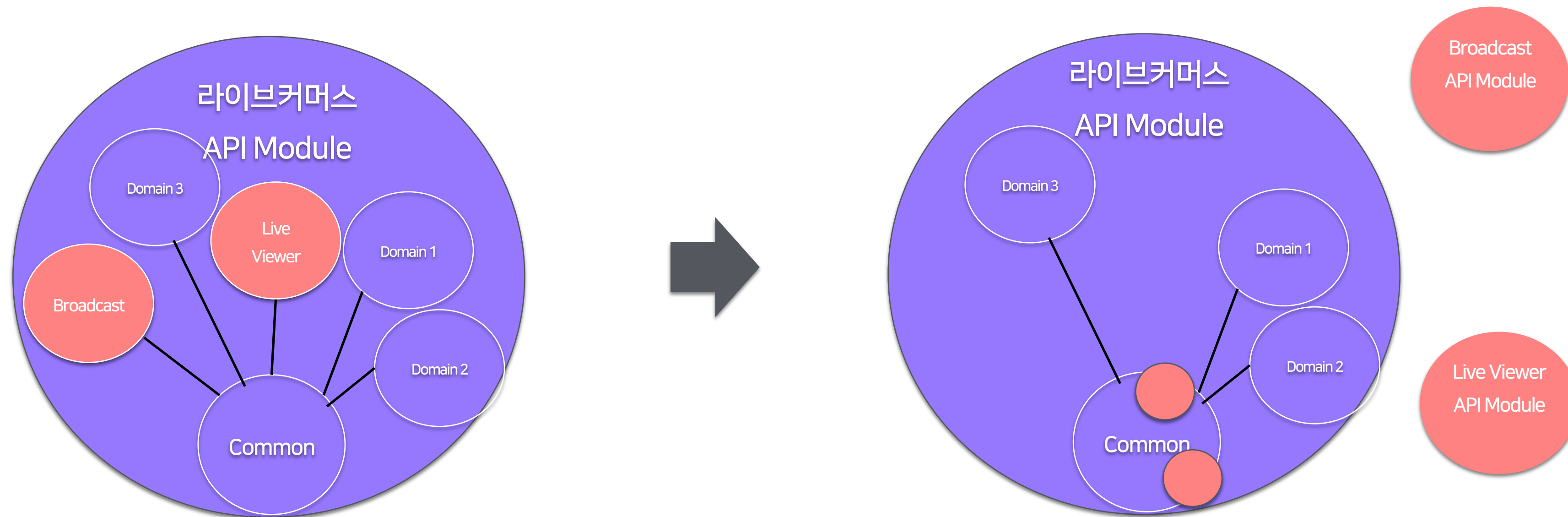
전체 트래픽의 90% 를 차지하는 **방송(뷰어/송출)**



3.2 시스템 구조 개선 - #1 트래픽 격리

Step 1

전체 트래픽의 90% 를 차지하는 **방송(뷰어/송출)** 서버 모듈 분리

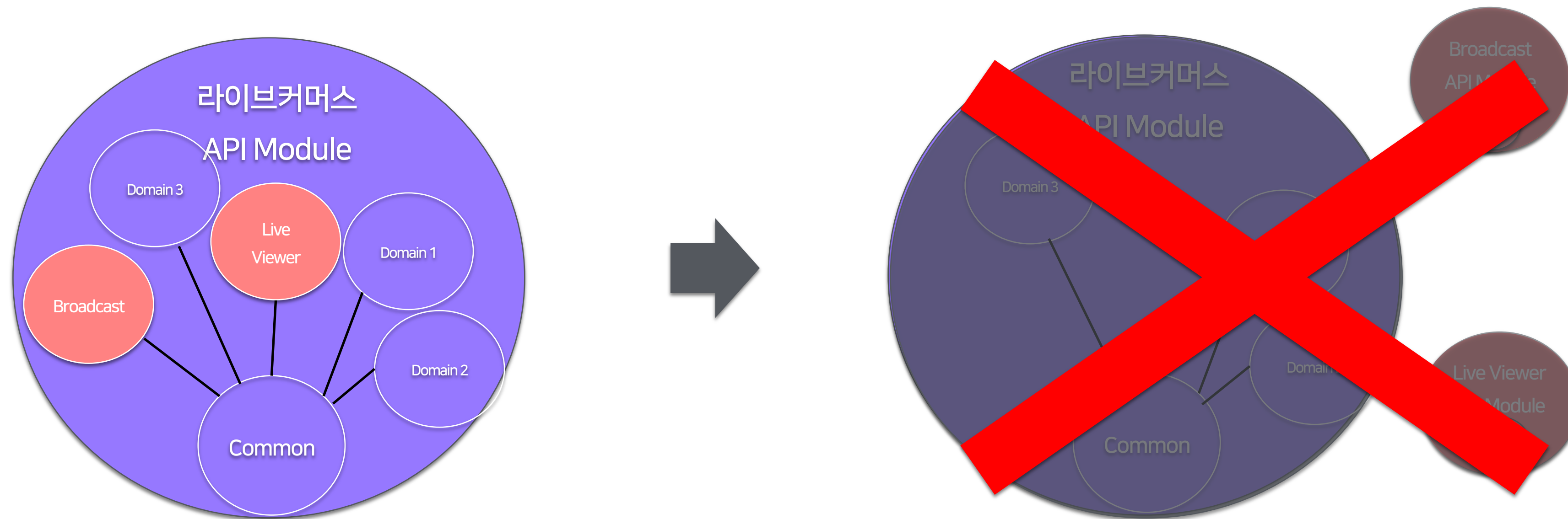


소스 코드 + 서버 모두 분리

3.2 시스템 구조 개선 - #1 트래픽 격리

Step 1

전체 트래픽의 90% 를 차지하는 **방송(뷰어/송출)** 서버 모듈 분리



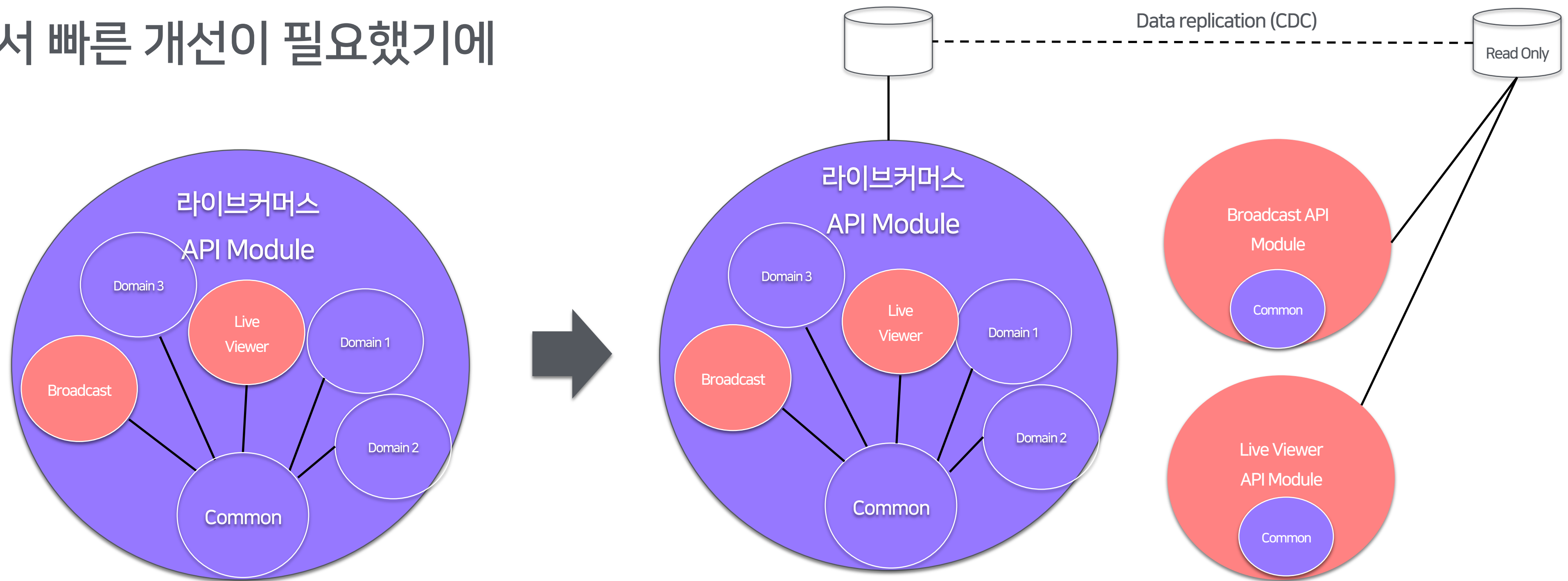
소스 코드 + 서버 모두 분리

하고 싶었지만 Common 모듈 의존도가
높은 상태라 단시간엔 어려운 상황

3.2 시스템 구조 개선 - #1 트래픽 격리

Step 1

안정성 측면에서 빠른 개선이 필요했기에



기존 시스템 구성에서

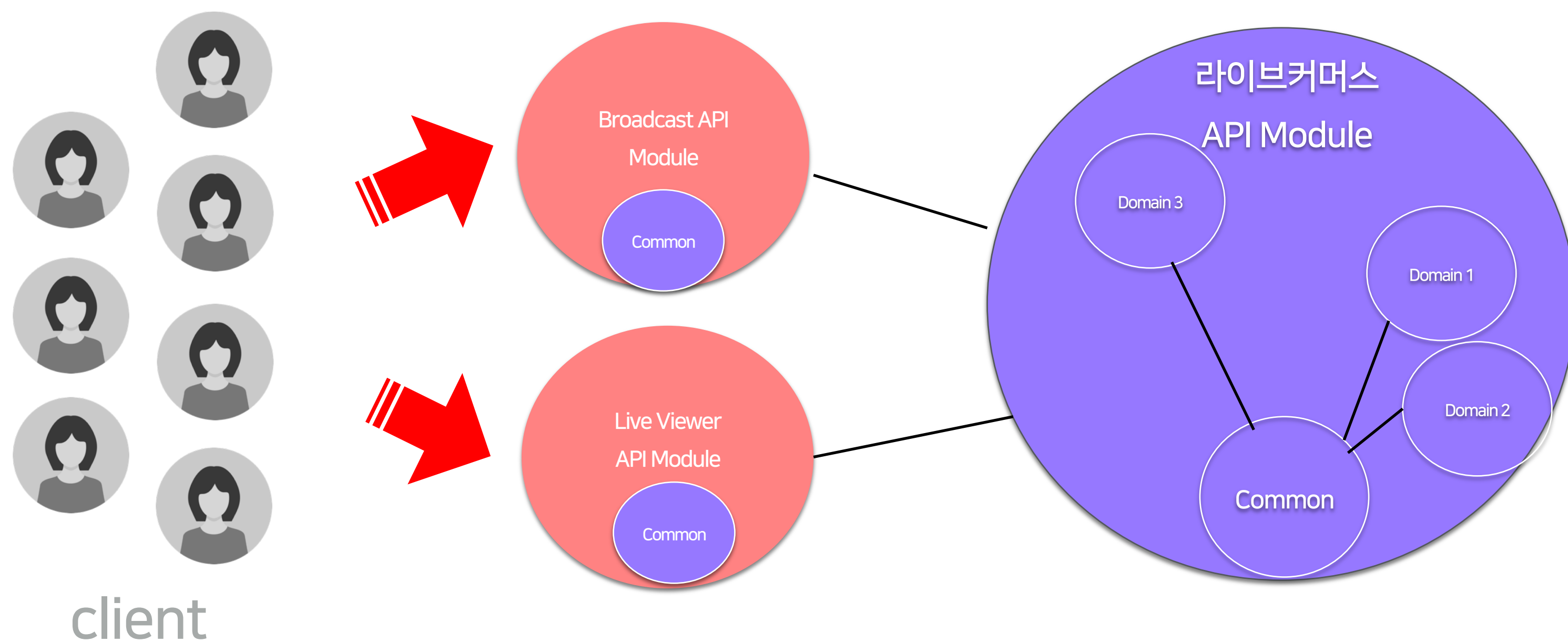
서버만 Physical 분리하고

소스 코드는 Logical 분리만 하는 것으로 결정

(이후 스텝에서 개선)

3.2 시스템 구조 개선 - #1 트래픽 격리

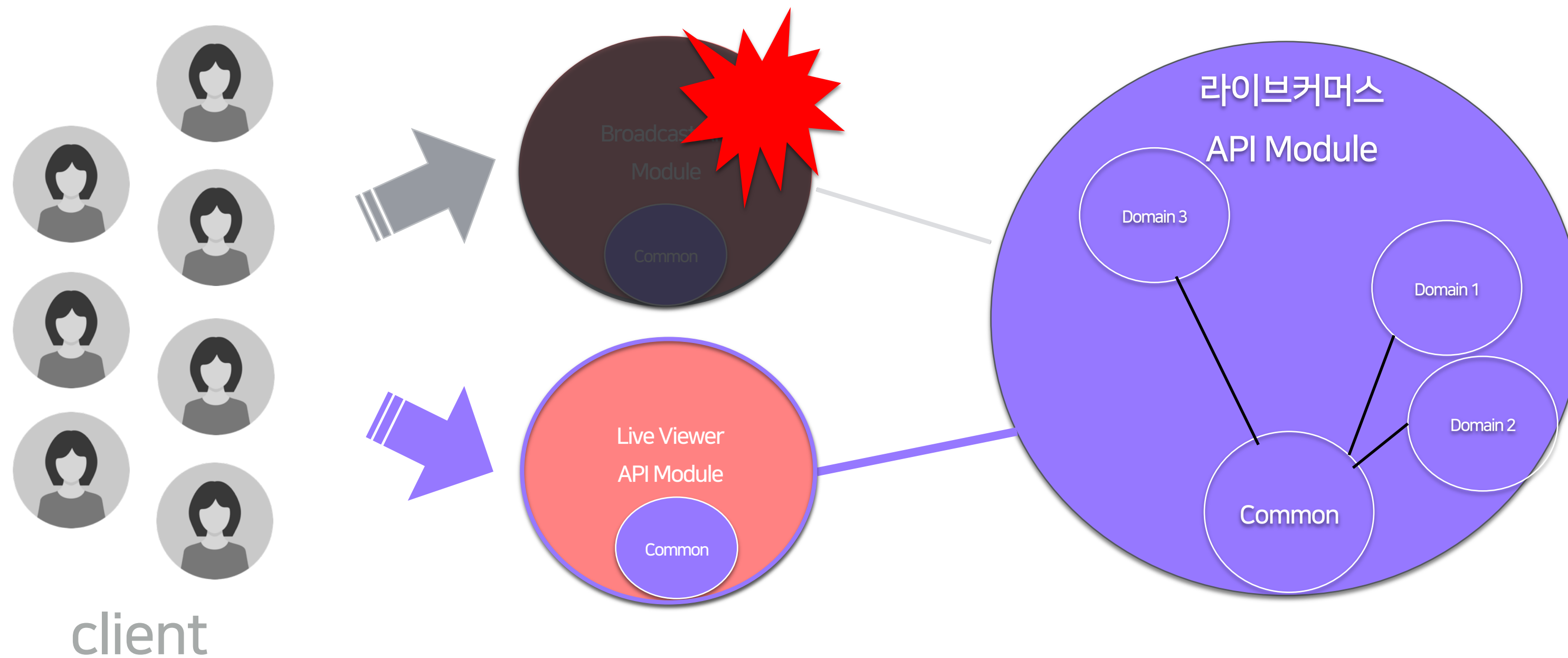
Step 1 결과



3.2 시스템 구조 개선 - #1 트래픽 격리

Step 1 결과

특정 서버(모듈)의 장애 발생시 다른 서버들에(주요 기능 단위) 영향을 주지 않게 됨



3.2 시스템 구조 개선 - #2 서버 IDC 이중화

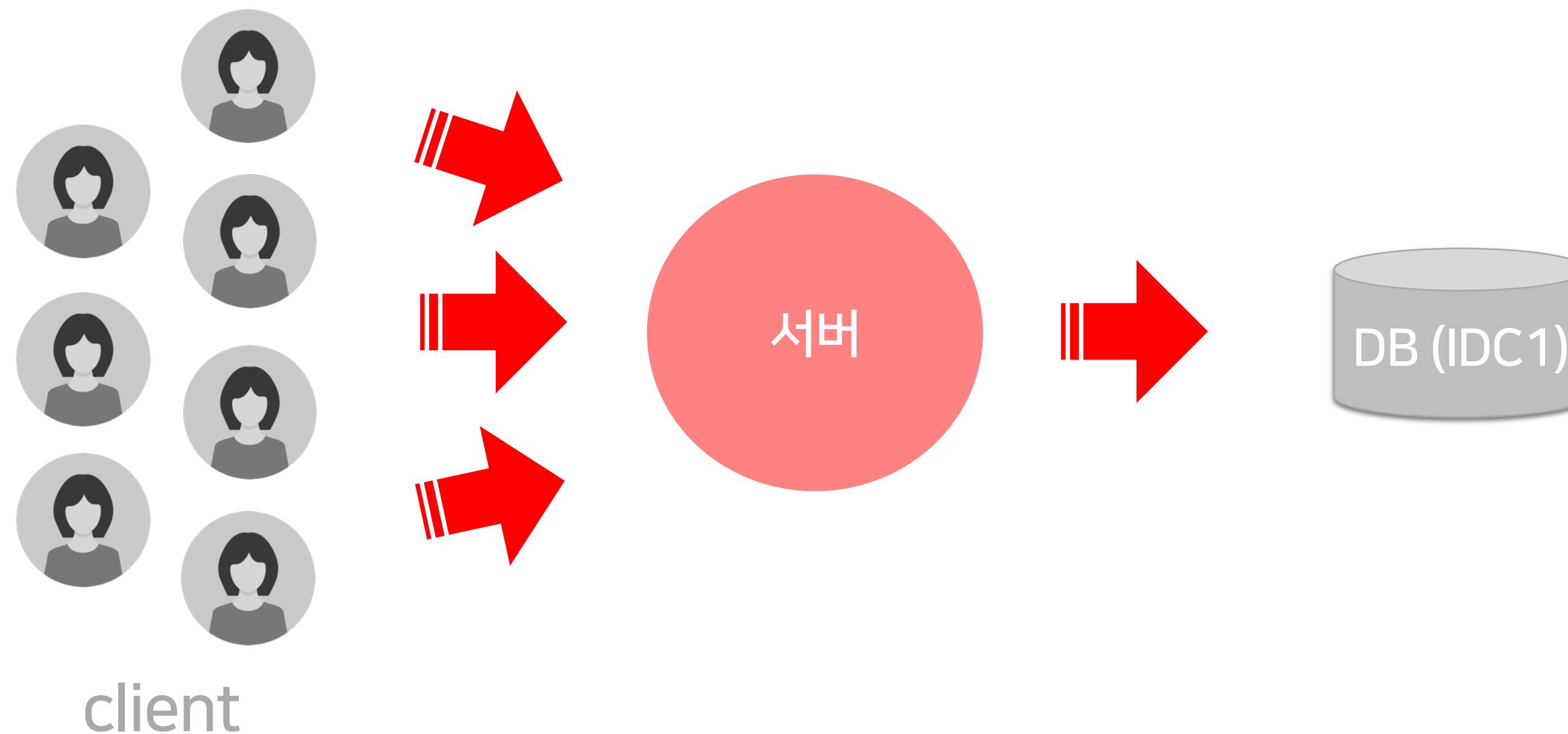
Step 2

DB 및 어플리케이션 서버 IDC 이중화 구성

3.2 시스템 구조 개선 - #2 서버 IDC 이중화

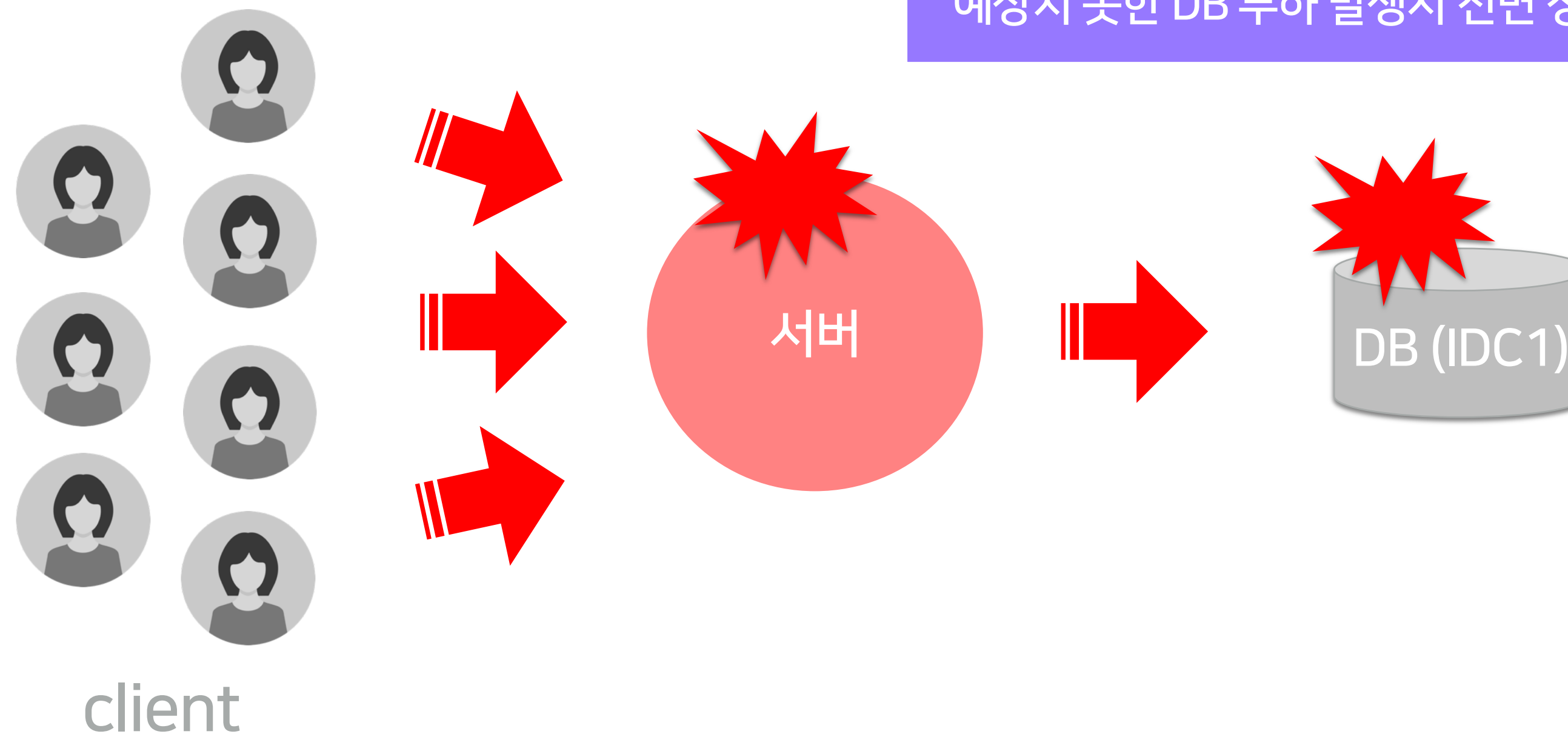
AS-IS

DB 및 어플리케이션 서버 단일 IDC 구성



3.2 시스템 구조 개선 - #2 서버 IDC 이중화

DB 및 어플리케이션 서버 단일 IDC 구성

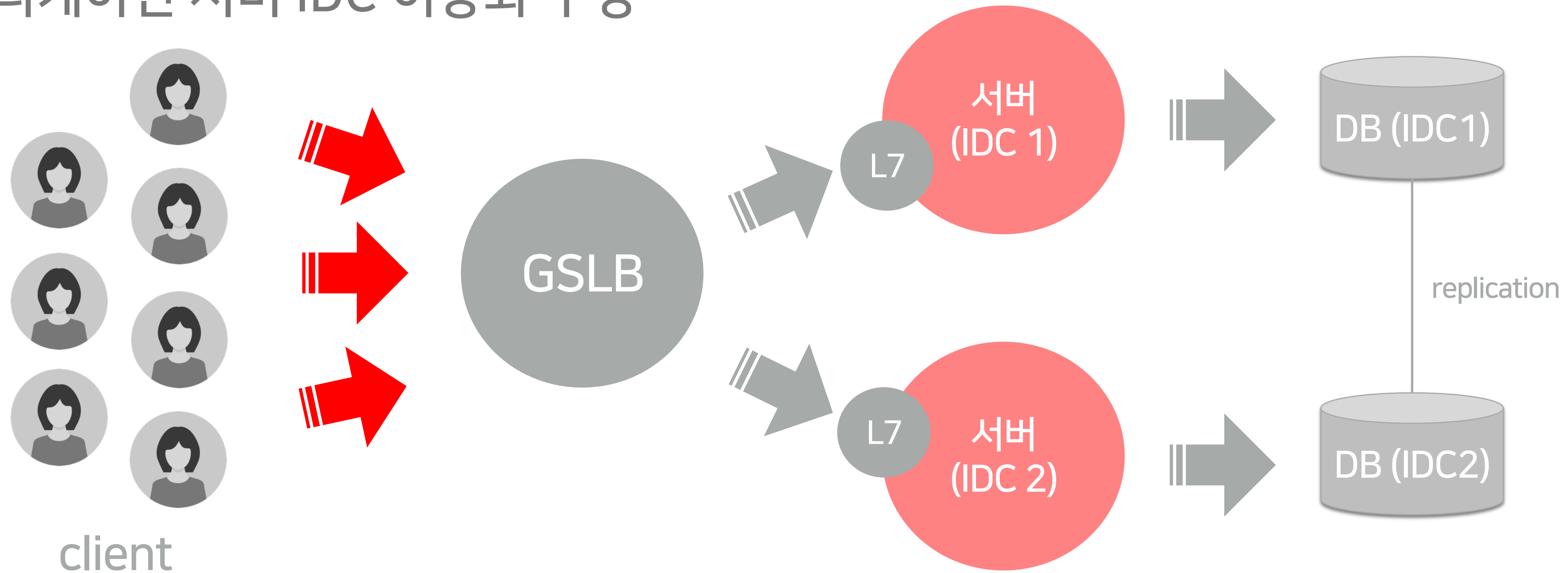


3.2 시스템 구조 개선 - #2 서버 IDC 이중화

TO-BE

IDC 이중화

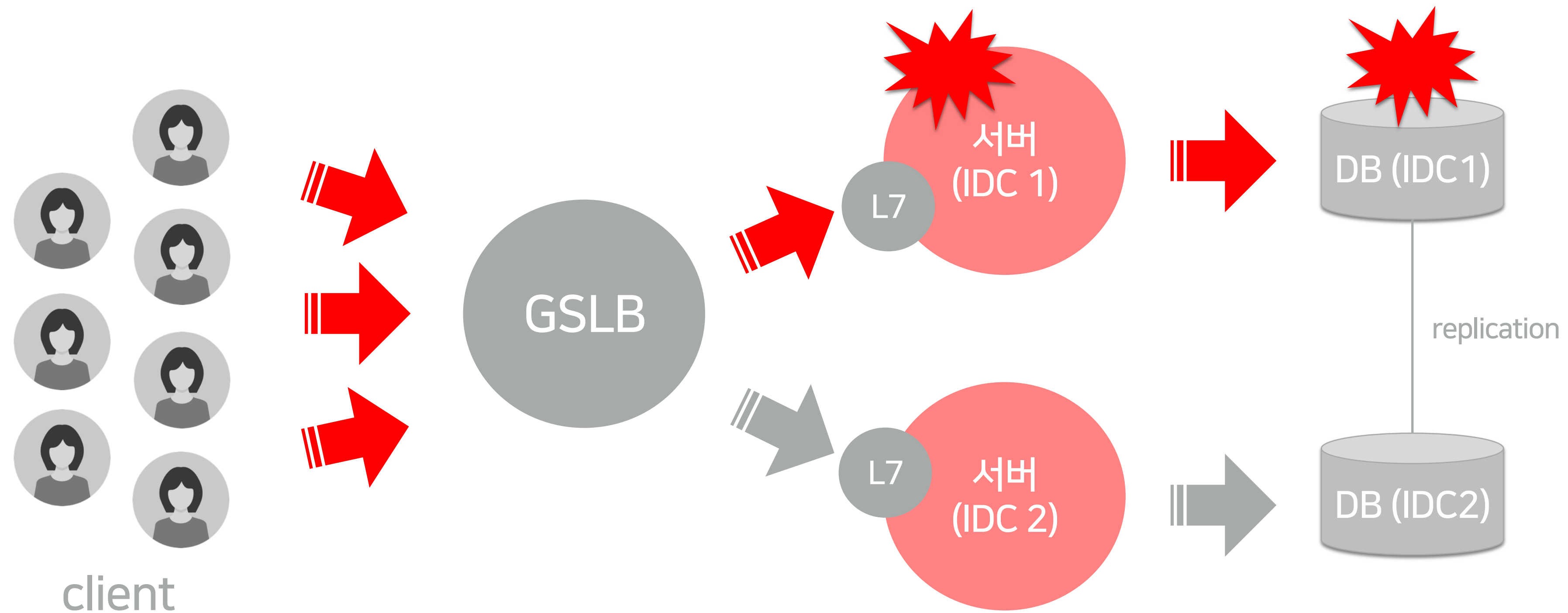
- GSLB(global server load balancing)
- DB 및 어플리케이션 서버 IDC 이중화 구성



3.2 시스템 구조 개선 - #2 서버 IDC 이중화

Step 2 결과

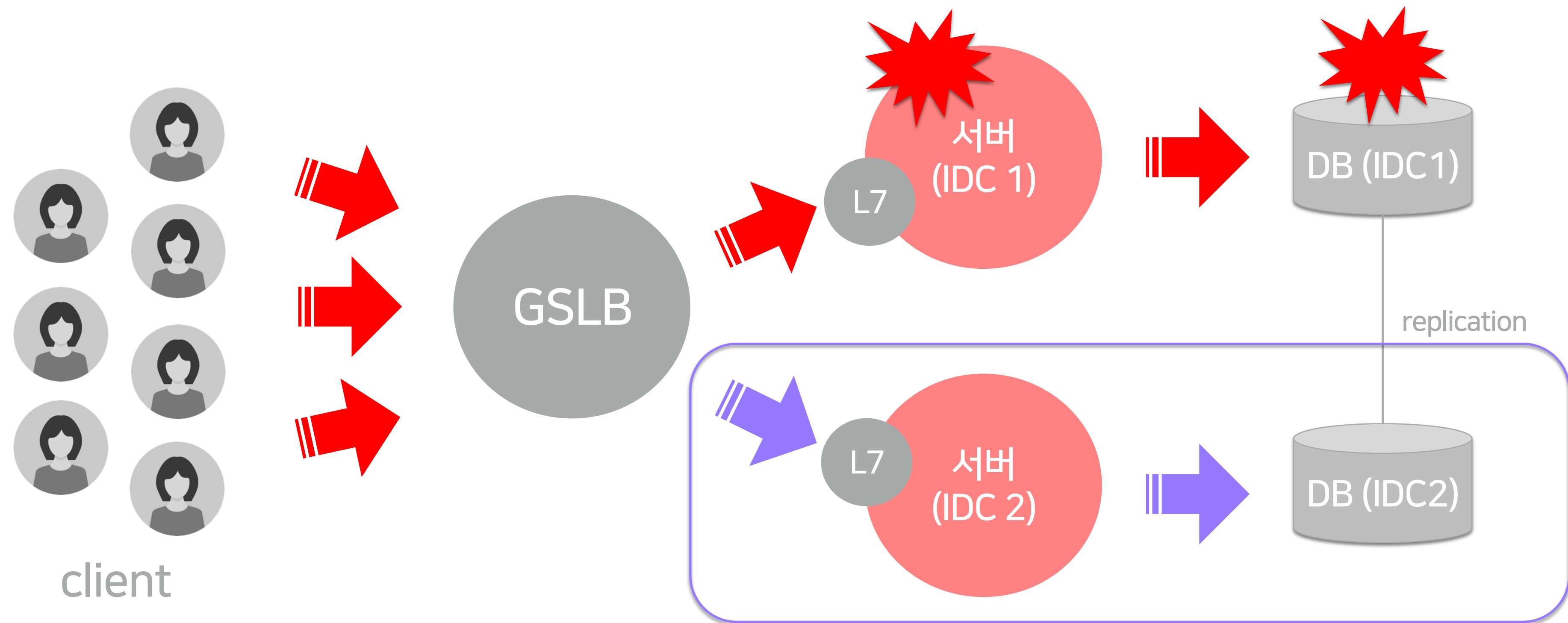
한쪽 IDC 스위치 장애 or DB/서버 장비 장애 발생시



3.2 시스템 구조 개선 - #2 서버 IDC 이중화

Step 2 결과

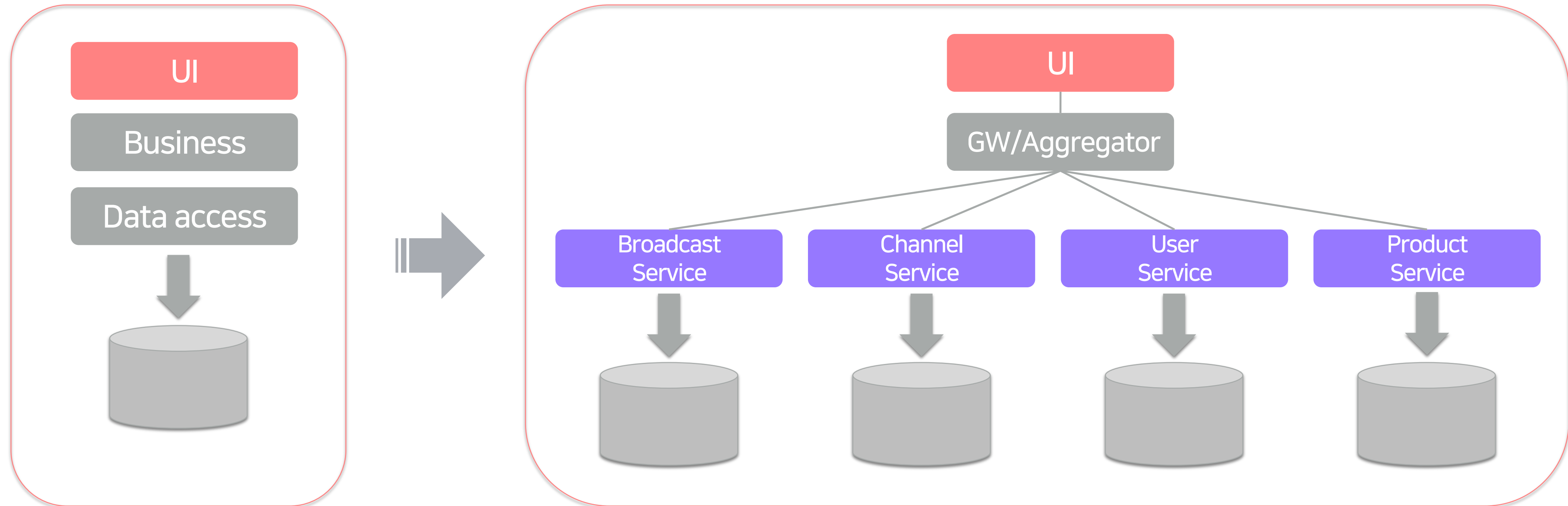
다른 IDC 를 통해 정상적으로 서비스 가능
+ 평상시엔 부하 분산 효과



3.2 시스템 구조 개선 - #3 MSA

Step 3

기존 Common 모듈 -> 주요 도메인 Service 단위로 분리 구성

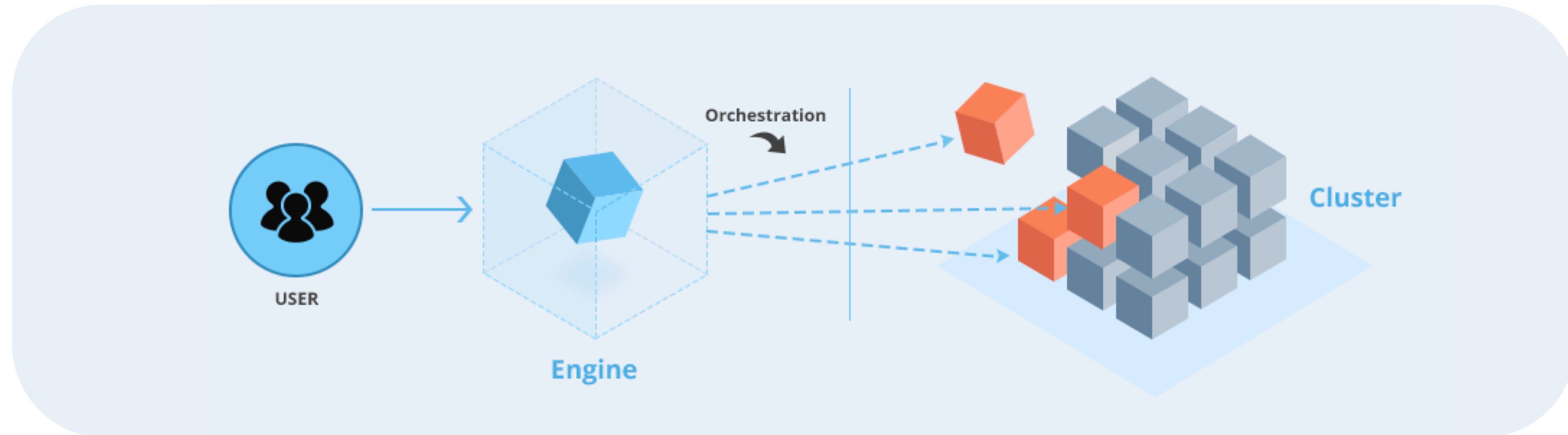


주요 도메인 서버 : 방송, 채널, 회원, 상품.. 등 현재 진행형

3.2 #4 고가용성 개선 - Auto Scaling

Step 4

Auto Scaling

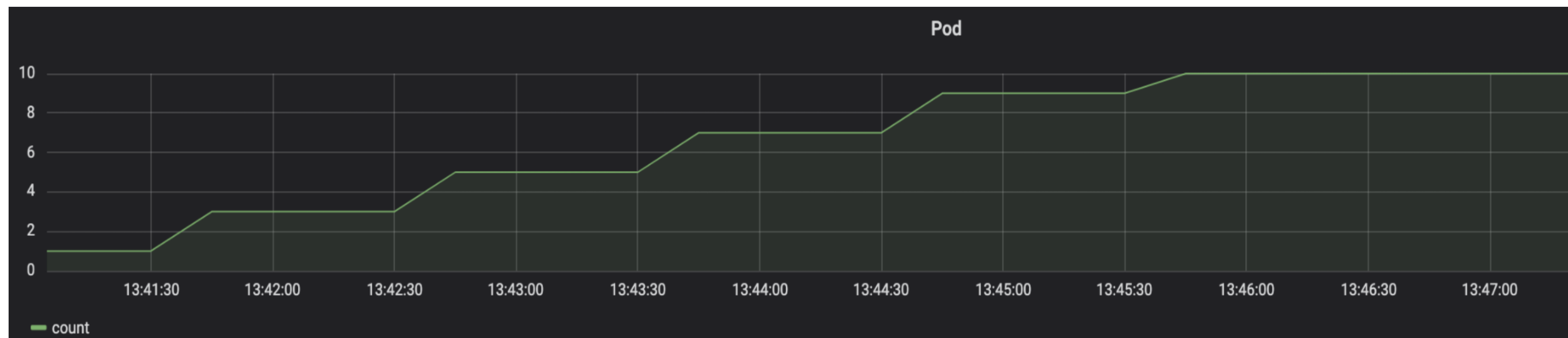


3.2 #4 고가용성 개선 - Auto Scaling

Step 4

HPA > 점진적 트래픽 상승시

- 인스턴스(POD) 수량이 점진적으로 증가(Auto Scaling)하면서 대부분 이상 없음



HPA : Horizontal Pod Autoscaler

3.2 #4 고가용성 개선 - Auto Scaling

HPA > 트래픽 급증시

- 오토 스케일링 전/중 트래픽 급증시 서비스 장애로 이어질 수 있음



3.2 #4 고가용성 개선 - Auto Scaling

HPA > **트래픽 급증시**

- 오토 스케일링 전/중 트래픽 급증시 서비스 **장애로 이어질 수 있음**



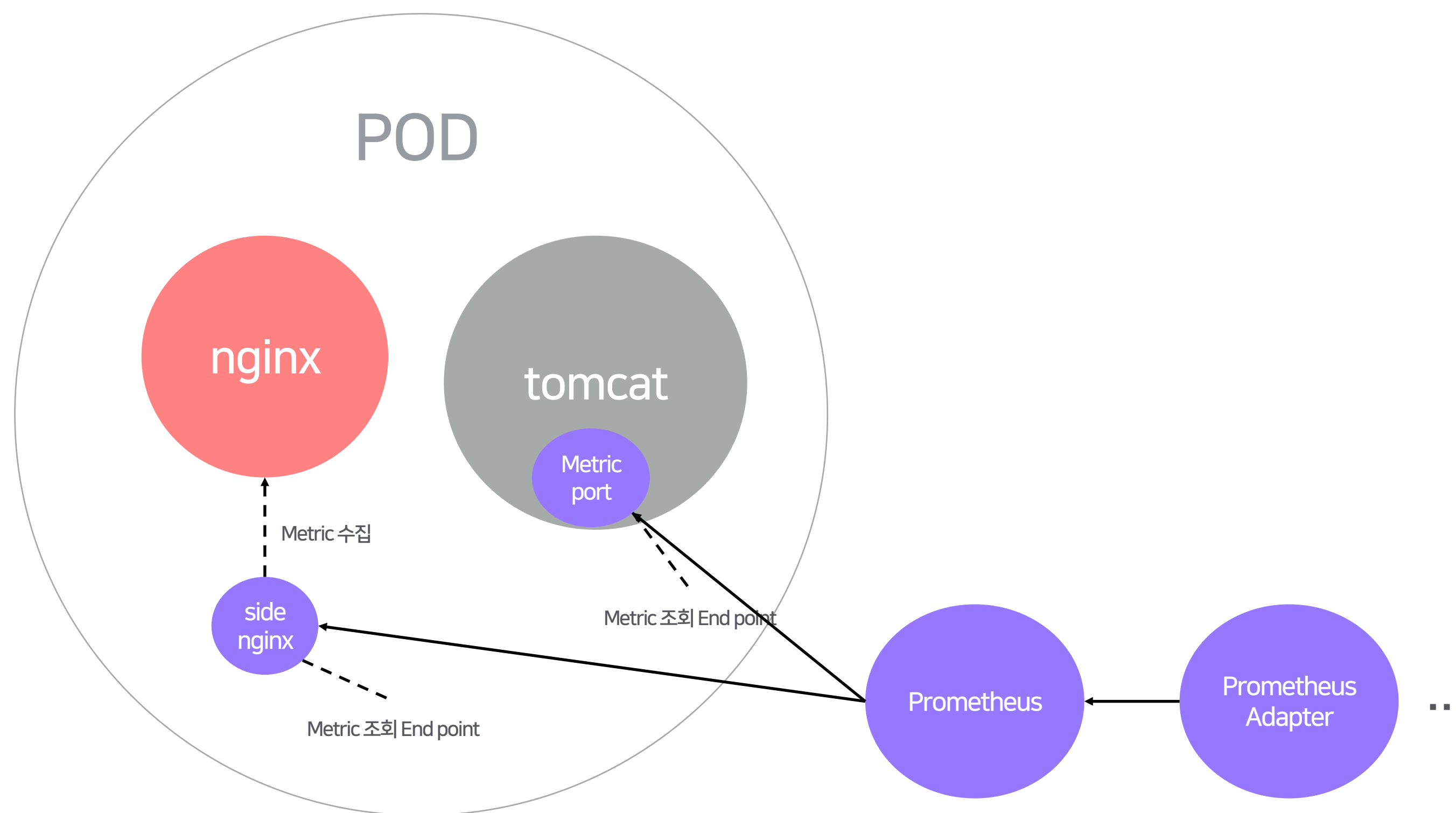
서비스 특성에 따라 적절한 대응방안 고려

- HPA With Custom Metrics
- Rate Limiting

3.2 #4 고가용성 개선 - Auto Scaling

HPA With Custom Metrics

- busy_thread (Tomcat)
- requests_total (Nginx)
- session_count (동접자)



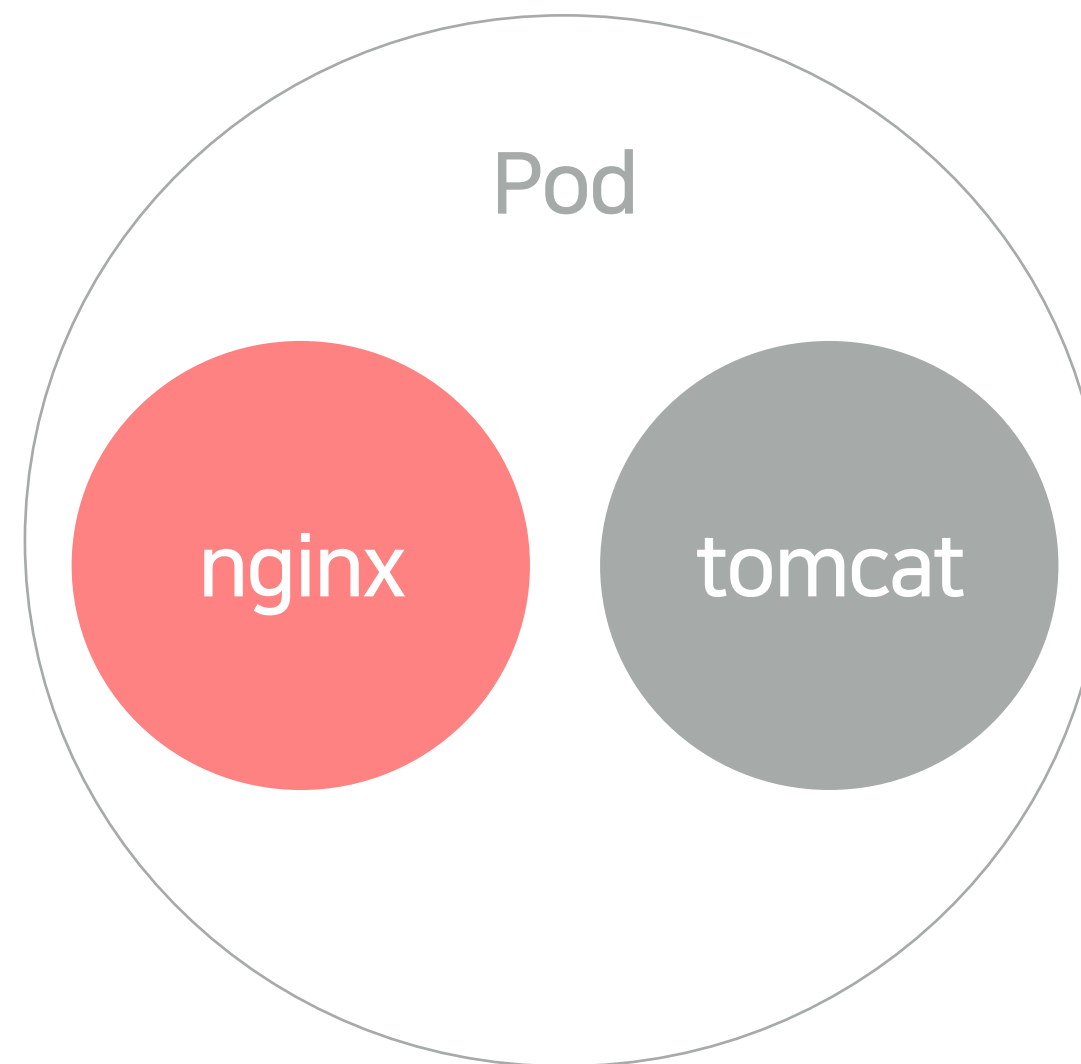
Custom Metric 예시

```
metrics:
- type: PromQL
  promQL:
    metric:
      name: nginx_http_requests_total
      query: 'sum by (pod)(rate(nginx_http_requests_total[container="nginx-metrics"])[1m])'
    target:
      type: AverageValue
      averageValue: 500
- type: PromQL
  promQL:
    metric:
      name: tomcat_threads_busy_threads
      query: 'sum by (pod,container)(tomcat_threads_busy_threads[container="app"])'
    target:
      type: AverageValue
      averageValue: 150
...
```


3.2 #4 고가용성 개선 - Auto Scaling

Rate Limiting

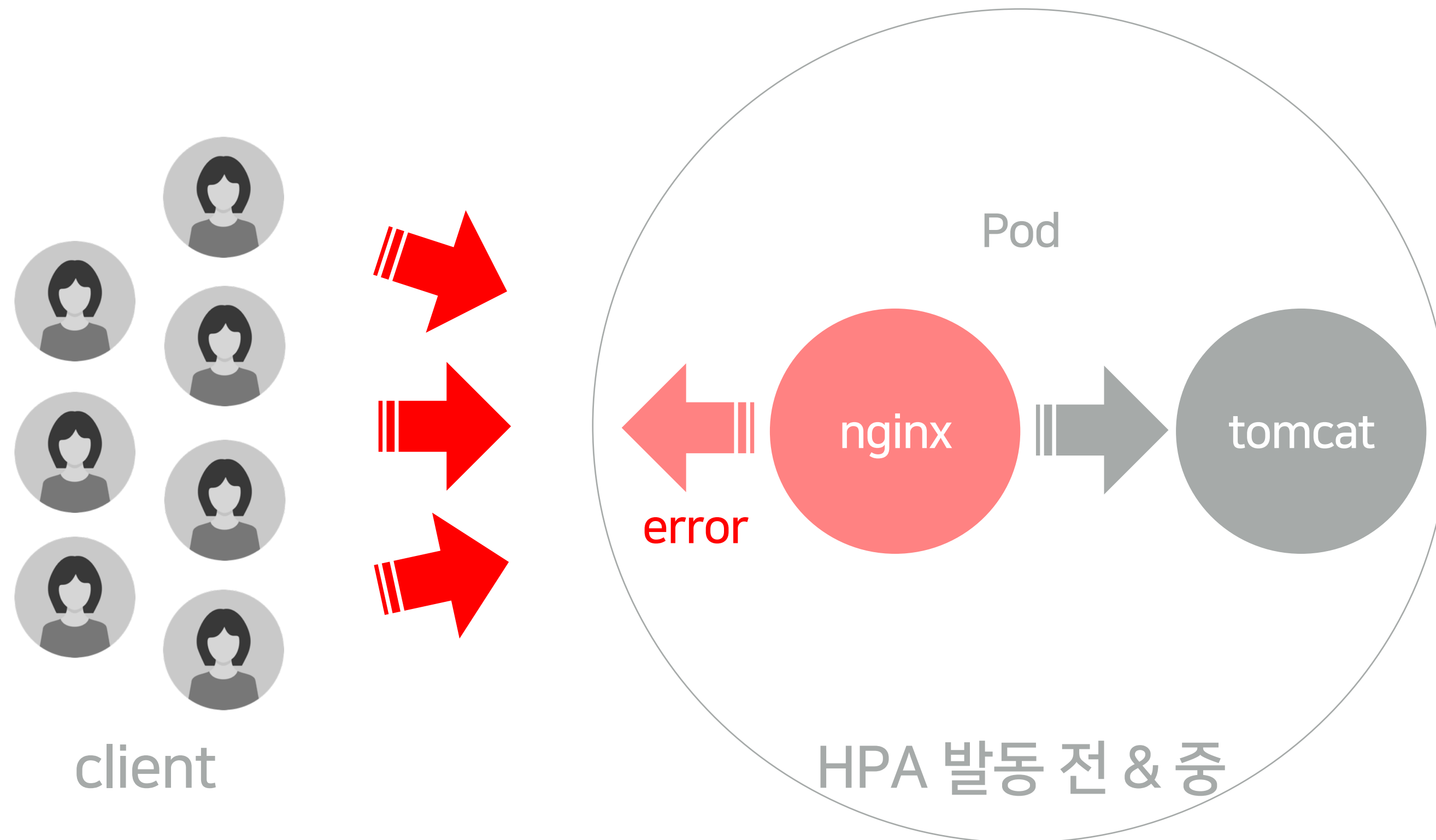
- HPA 적용 서버에 대해 rate limit 설정 추가



3.2 #4 고가용성 개선 - Auto Scaling

Rate Limiting

- HPA 적용 서버에 대해 rate limit 설정 추가



```
limit_req_zone $server_addr zone=server_addr:10m rate=50r/s;
server {
    location /a {
        proxy_pass http://backend;

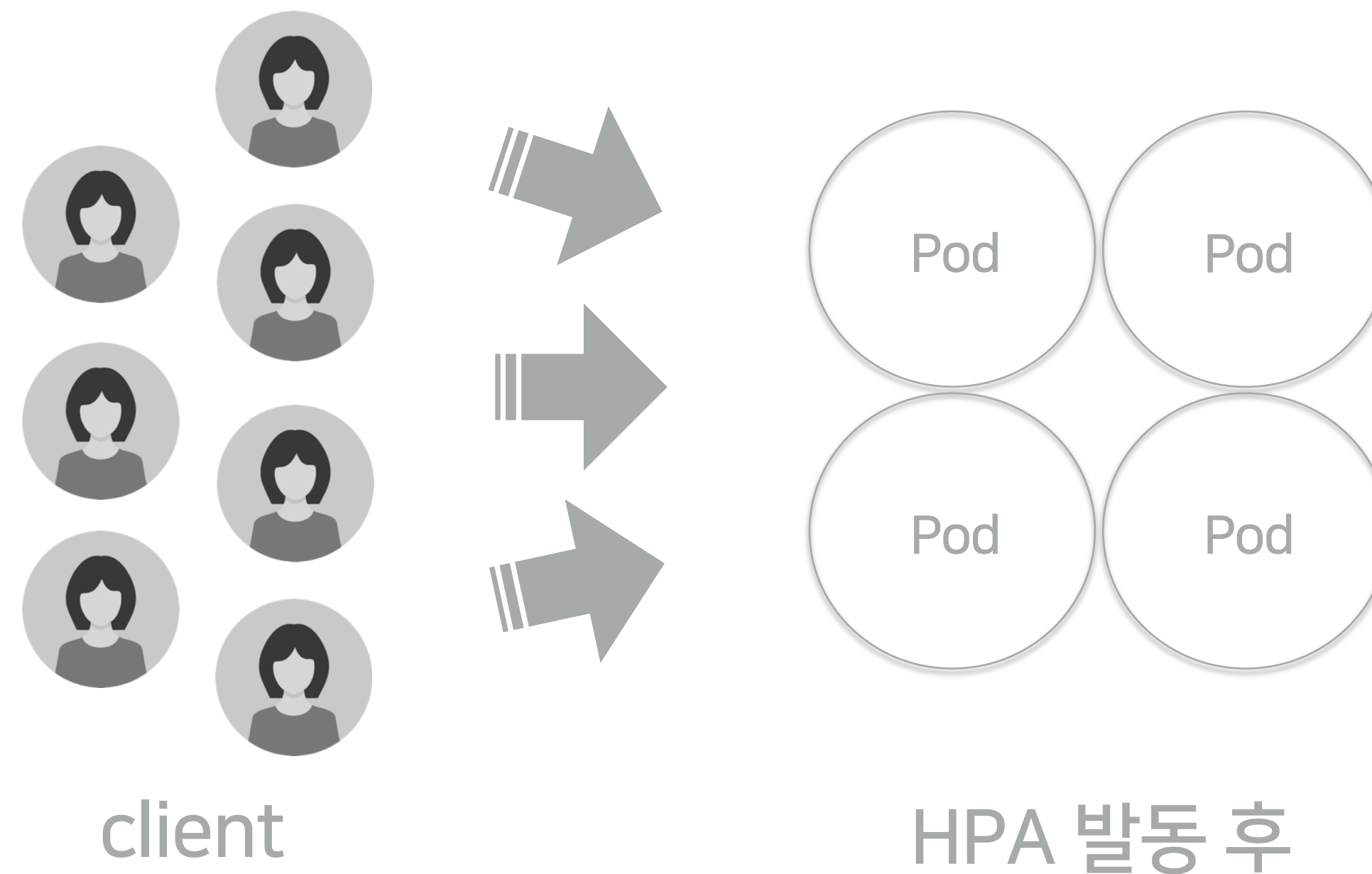
        limit_req zone=server_addr burst=100 nodelay;
        limit_req_dry_run off;
    }
    location /b {
        proxy_pass http://backend;

        limit_req zone=server_addr burst=150 nodelay;
        limit_req_dry_run off;
    }
}
```

3.2 #4 고가용성 개선 - Auto Scaling

Rate Limiting

- HPA 적용 서버에 대해 rate limit 설정 추가



3.3 Summary

- 모놀리스 구조 개발 > 빠른 개발 및 테스트



- 주요 기능 단위 서버 분리 구축 > 장애 격리



- IDC 이중화 > 고가용성



- 주요 도메인 단위 서버 분리 구축 > 개발 생산성, 안정성



- AutoScaling, RateLimiting > 트래픽 방어, 고가용성 확대

4. 안정적으로 운영하기

4.1 장애 예방을 위한 노력 - 테스트

트래픽 테스트

차수	동접수	xxx-api 1대당 TPS	CPU	rate-limit 발동	HPA 발동	xxx-web POD 수	xxx-api POD 수
1차	000	000	00%	X	X	1	1
2차	000	000	00%	X	X	1	1
3차	000	000	00%	X	O	1	1 → 4 → 3
4차	000	000	00%	O	O	1	1 → 10 → 4
5차	000	000	00%	O	O	1 → 3 → 2	1 → 22 → 6
6차	000	000	00%	O	O	1 → 5 → 3	1 → 30 → 8

4.1 장애 예방을 위한 노력 - 테스트

각 상황별 장애 시나리오 검증 및 테스트

장애상황	영상재생	채팅	공지	상품 목록	쿠폰
A master DB 섯다운	○	⦿	X 발송 불가 조회 가능	⦿	⦿
A slave DB 섯다운	⦿	⦿	⦿	⦿	⦿
B slave DB 섯다운	⦿	⦿	⦿	⦿	⦿
A redis 섯다운 (A IDC)	○	○	○	○	○
C Service API 섯다운	○	⦿	○	X 갱신 불가 캐시 조회가능	⦿
D API 섯다운	⦿	⦿	⦿	⦿	○
A+B+C 섯다운	⦿	X 작성 불가 조회 가능	X	X	X
view api 섯다운 (퍼초)	가려져아				

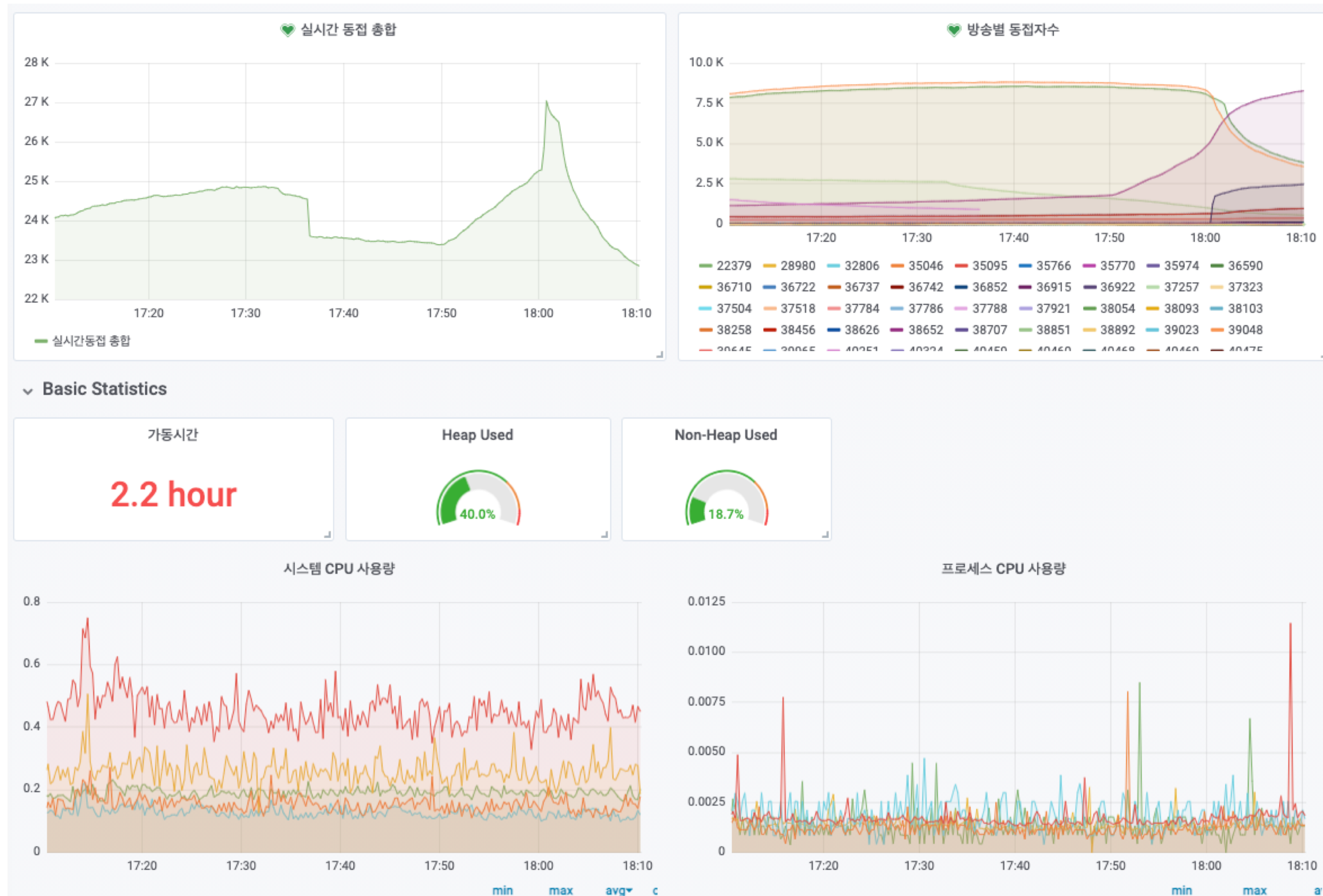
테스트 시나리오 파일

파일

- >  Scenario_v0.5.xlsx
- >  Scenario_v0.6.xlsx
- >  Scenario_v0.7.xlsx
- >  Scenario_v0.8.xlsx
- >  Scenario_v0.9.xlsx
- >  Scenario_v0.10.xlsx
- >  Scenario_v0.11.xlsx

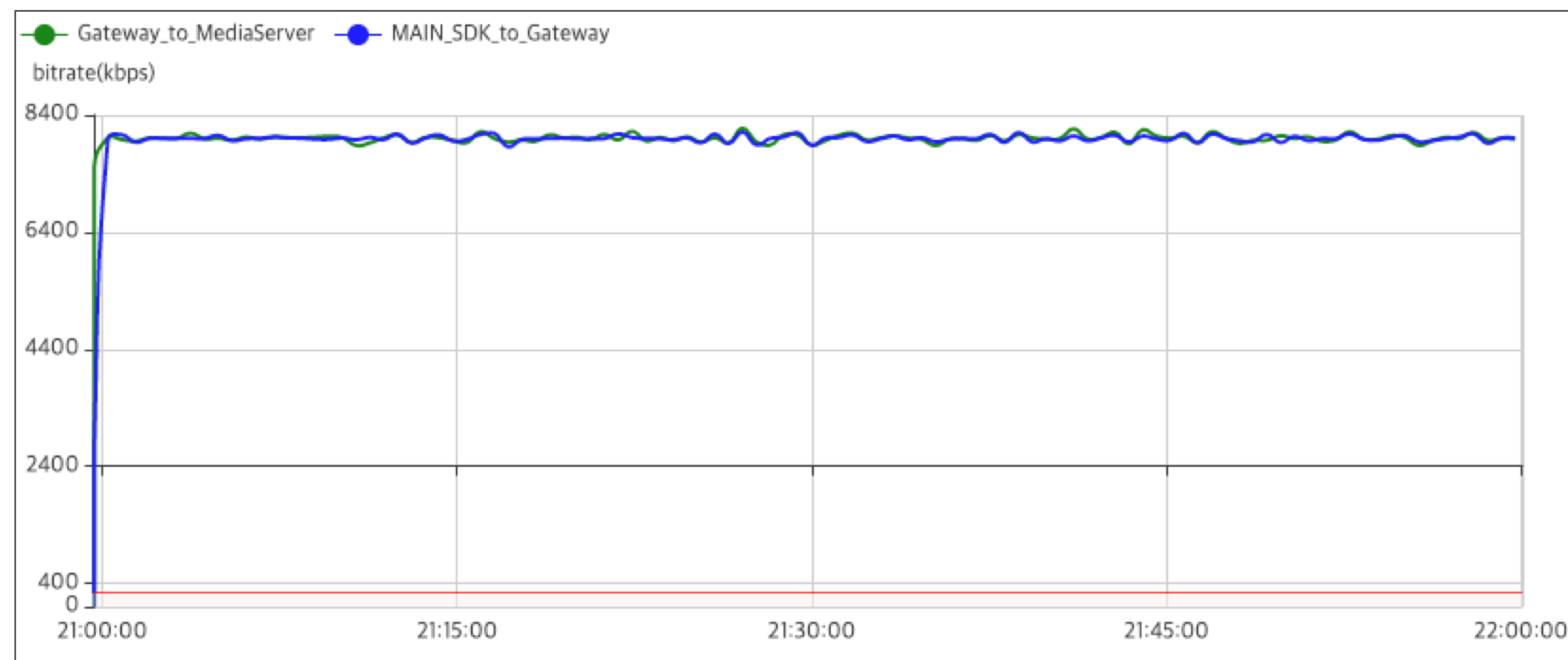
4.1 장애 예방을 위한 노력 - 모니터링

실시간 동접자 현황 및 트래픽 증가 모니터링

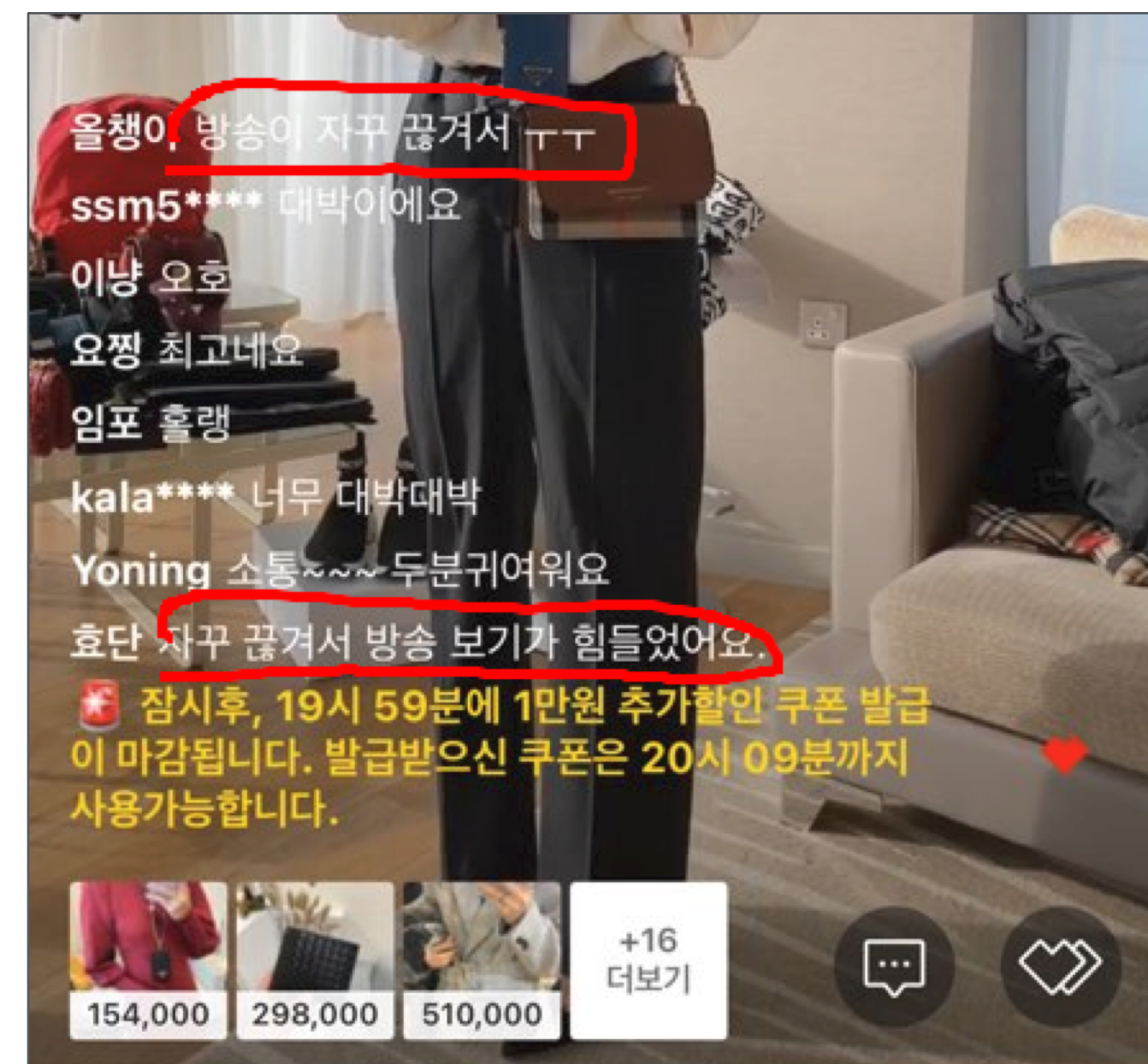


4.1 장애 예방을 위한 노력 - 모니터링

방송 모니터링



송출 품질 모니터링

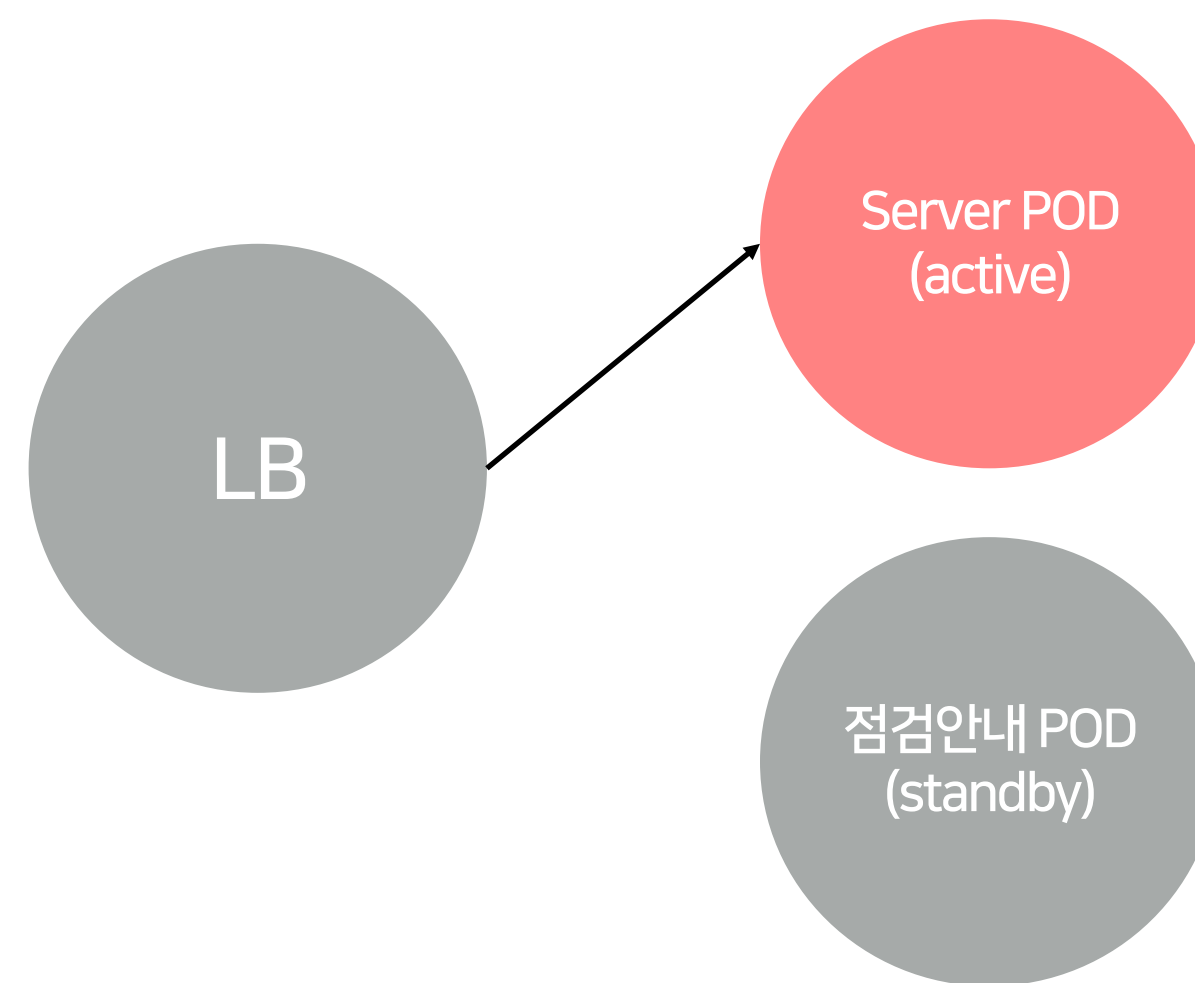


채팅 실시간 분석

4.2 장애 발생시 대응 방안

내부 장애 발생시 - **서버 전면 장애**

- 트래픽 우회 및 점검안내 용도 standby POD 항시 대기

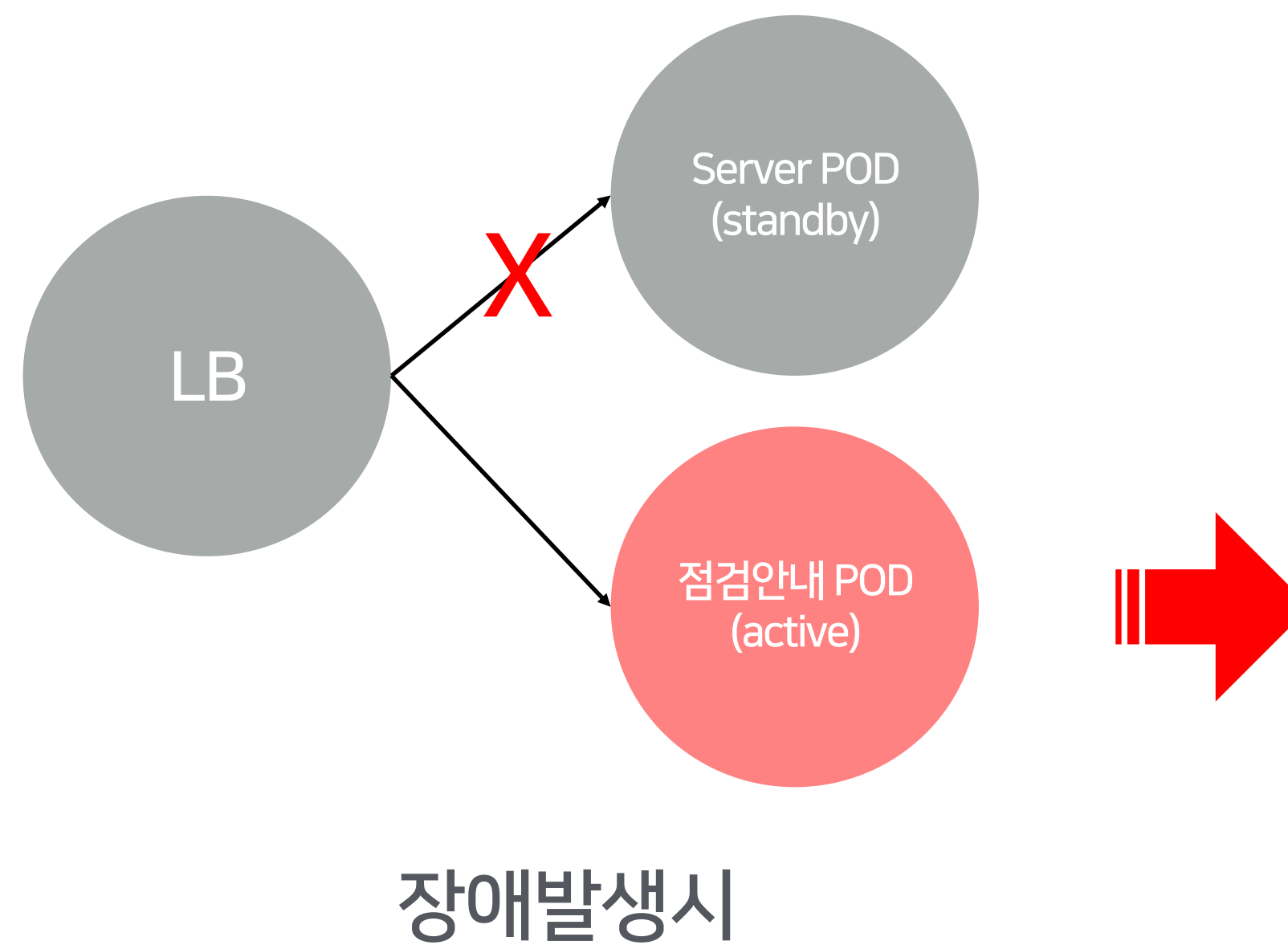


평상시

4.2 장애 발생시 대응 방안

내부 장애 발생시 - **서버 전면 장애**

- 트래픽 우회 및 점검안내 용도 standby POD 항시 대기



쇼핑 **LIVE**

☑ 🔍 👤

< 공지사항

[장애공지] 네이버페이 간헐적 장애 안내
2020.12.14

안녕하세요. 네이버 쇼핑라이브 서비스팀입니다.
서비스 내 아래와 같이 간헐적 장애가 발생하고 있습니다.

- 장애 내용 : 네이버페이 장애로 인한 결제페이지 연결지연
- 장애 발생 : 12월 14일 오전 11시 20분 ~

서비스 이용에 불편을 드려 다시 한번 사과드리며,
보다 안정적인 서비스를 위해 최선을 다하겠습니다.

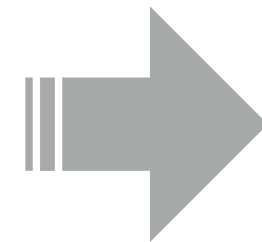
네이버 쇼핑라이브 서비스팀

4.2 장애 발생시 대응 방안

내부 장애 발생시 - 특정 API 장애



client



A API

B API

C API

D API

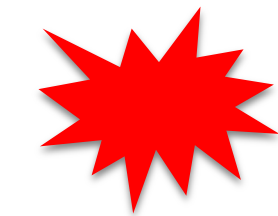
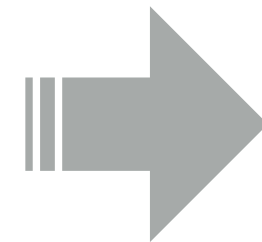
4.2 장애 발생시 대응 방안

내부 장애 발생시 - 특정 API 장애

- 다른 연동 서버에 영향을 최소화 해야한다.
- 페이지 영향이 없어야 한다.



client



A API

B API

C API

D API

4.2 장애 발생시 대응 방안

내부 장애 발생시 - 특정 API 장애상황
- API/Web 응답 우회 툴

real-view-api ▾

REAL-VIEW-API 설정 적용 초기화

설정 적용 컴포넌트 선택

응답값 수정: 매칭되는 요청의 응답값을 수정합니다.

패턴 /aaa/**	GET ▾ http 메소드	컨텐츠 타입 application/json;ch	HTTP 코드 999	응답 수정값 {"message" : "DEVIEW 반가워요"}	추가
----------------------------------	--------------------------------------	--	-----------------------------------	--	---------------

API 예시

4.2 장애 발생시 대응 방안

내부 장애 발생시 - 특정 API 장애상황 - API/Web 응답 우회 툴

real-view-api ▾
설정 적용 컴포넌트 선택

REAL-VIEW-API 설정 적용 초기화

응답값 수정: 매칭되는 요청의 응답값을 수정합니다.

패턴 /aaa/**	GET ▾ http 메소드	컨텐츠 타입 application/json;ch	HTTP 코드 999	응답 수정값 {"message": "DEVIEW 반가워요"}
---------------	-------------------	-------------------------------	----------------	--------------------------------------

API 예시

real-view-web ▾
설정 적용 컴포넌트 선택

REAL-VIEW-WEB 설정 적용 초기화

리다이렉트: 웹 페이지 요청을 리다이렉트 처리합니다.

전체 페이지 리다이렉트	패턴 /**	Redirect https://shoppinglive.n	추가
수동 지정	패턴 /bbb/devview.html	Redirect https://www.abc.com	추가

Web 예시

4.3 Summary



5. 쇼핑라이브의 현재와 미래

5. 쇼핑라이브의 현재

	서비스 신뢰성	재생 기술	송출 기술	기타 기술
현재	#고가용성 #IDC이중화 #트래픽제어 #모니터링 #장애대응	#줌확대 #초고속피드 #ULL #멀티CDN #타임머신	#이펙트 #오버레이 #HEV #LIVE_TO_VOD #이중회선	#상품노출구간추출 #음성인식자동자막

5. 쇼핑라이브의 미래

현재

서비스 신뢰성	재생 기술	송출 기술	기타 기술
#고가용성 #IDC이중화 #트래픽제어 #모니터링 #장애대응	#줌확대 #초고속피드 #ULL #멀티CDN #타임머신	#이펙트 #오버레이 #HEV #LIVE_TO_VOD #이중회선	#상품노출구간추출 #음성인식자동자막

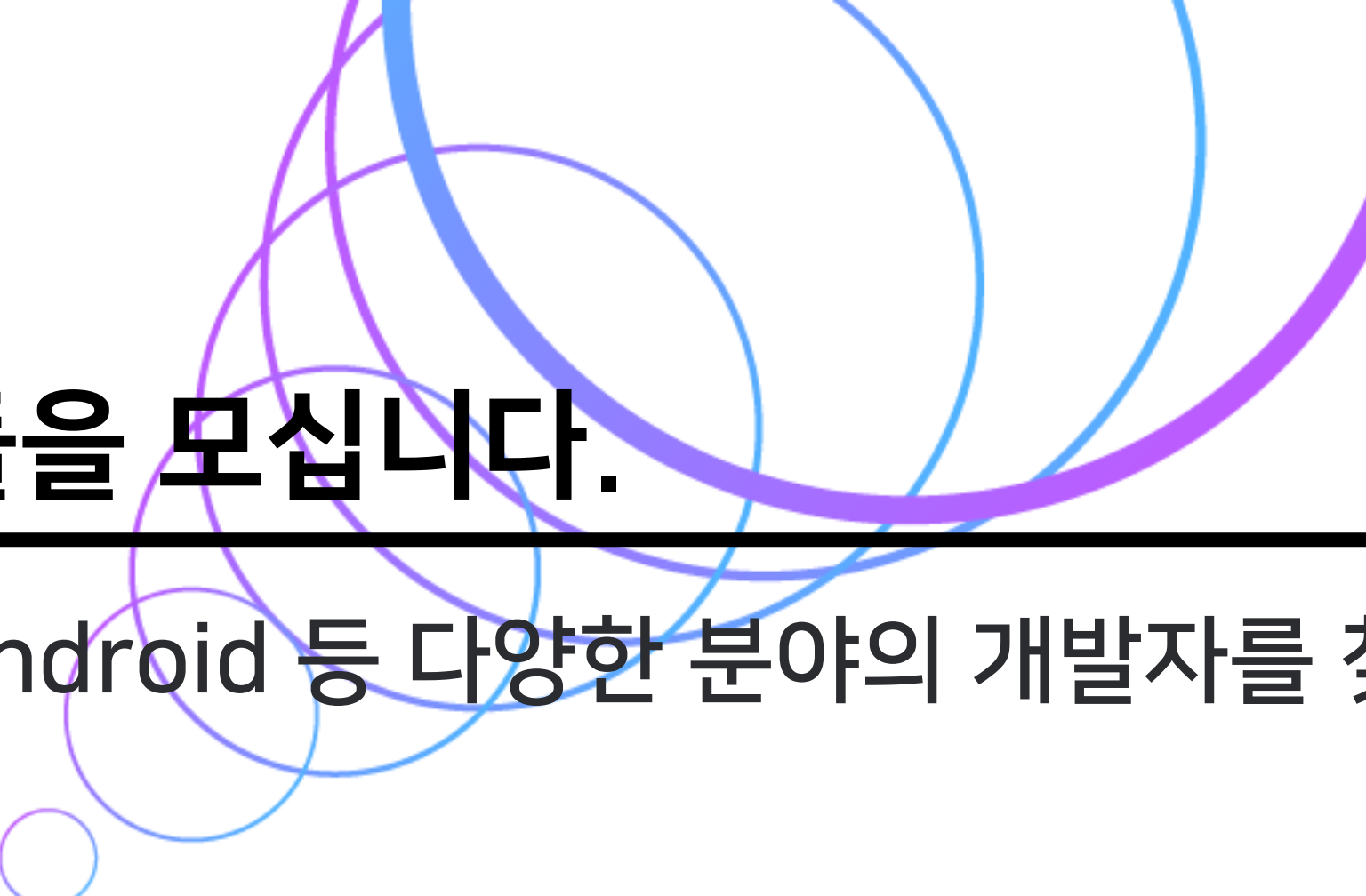


미래

#장애탐지 #장애복구자동화	#비디오커머스 #실시간이벤트	#실시간크로마키 #카메라프로	#AR라이브커머스 #콘텐츠추천 #데이터인사이트
#쇼핑라이브App		#제휴사OpenAPI	



NAVER
Live Commerce



함께 성장할 동료분들을 모십니다.

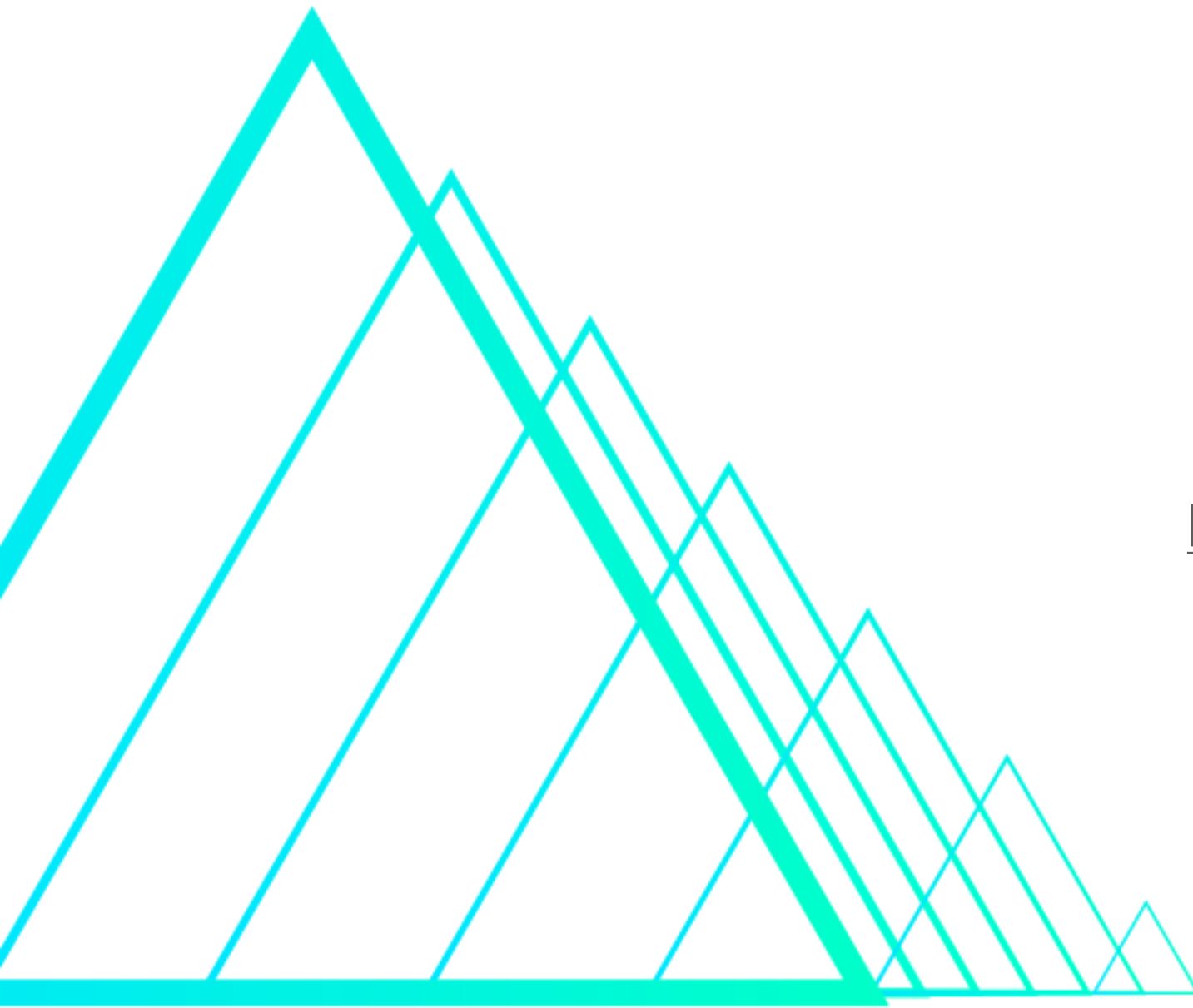
FE / BE / Data / iOS/ Android 등 다양한 분야의 개발자를 찾고 있습니다.

라이브커머스 플랫폼 직무 소개

<https://naver-career.gitbook.io/kr/service/shopping/livecommerce>

라이브커머스 플랫폼 채용 문의

dl_livecmc_recruit@navercorp.com





Thank You