Software Architecture

Implicit Asynchronous Communication Software Architecture

Buffered Message-Based Architecture Style

Eunmi Choi
Kookmin University

Buffered Message-Based Architecture Style

- Structure
  - Breaks the software system into three partitions:
    - message producers
    - message consumers
    - message service providers. 메세지를 처리해줌.
  - They are connected asynchronously by either a message queue or a message topic.
  - It is typically implemented as a message-oriented middleware (MOM) providing a reliable message service on a distributed system

세가지는 비동기적으로 연결됨. 보내놓고 자기 프로세스를 하다가 외부에서 회신을 보내주면 콜백이나 디몬이나 다른 형태로 비동기적으로 묶어둠. 메세지 큐나 메세지 토픽을 가지는 버퍼로 주로 연결함.

이런 것을 메세지 오리엔티드 미들웨어라 함. 메세지를 중간에서 처리할 수 있는 구조.

Buffered Message-Based Architecture Style

메세지 생성자와 소비자가 디커플링하게 함. 버퍼를 중신

메세지를 처리해주는 다른 요소를 충분히 릴라이어블하게 할 수 있음
버퍼를 중심으로 서버나 클라이언트가여러개
프로그램을 유동적으로 많이 붙일 수 있음.

- Characteristics
  - Messaging systems are used to build reliable, scalable, and flexible distributed applications supporting asynchronous communication.
  - Messaging system architecture is essentially a peer-to-peer client-server architecture
  - The high degree of independency of components within the messaging system(loosely coupled distributed communication between software components)
    - a message receiver does not need to be available at the same time as the message sender in order for the communication to take place.
    - In fact, the sender and receiver don't need to know each other's identity at all.
    - The receiver, however, needs to know the message format and the message destination where the message is available.

비동적으로 서로 연결할 수 있음.

MOM은 서버쪽 또는 중간에 둘 수 있음.
모든 시스템이 동일 형태의 구조를 가짐.
버퍼를 설계해줘야

메세지시스템을 중심으로 분리되고 독립적.
원할 때 필요하면 풀해서 가져가고 커플링을 나눠서 분리.. 서로의 주소를 알 필요가 없고 포트 넘버만 알면 됨. 버퍼만 알면 됨. - > 유동성 가질 수 있게 됨.
리시버 역할 : 버퍼에서 메세지를 갖다 써야하는데 이때 메세지 포맷을 알아야 함.

갈수록 dev ops :데이터에 따라오퍼레이션 함. 을 따름
많은 양의 데이터를 프로세스 분석 발췌할까가 그 다음 target가 사용할지에 중요해짐.
서비스르

Buffered Message-Based: Application

읽으심.

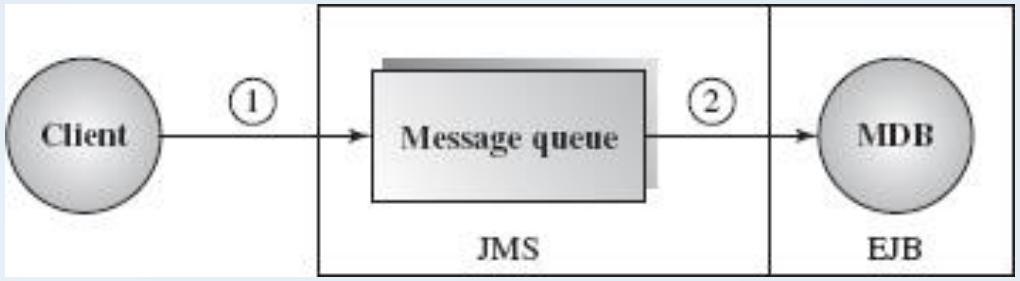
- Application Examples
  - widely used in the infrastructure for network management, telecommunication services, e-commerce, customer care, weather forecasting, supply chain management, banking systems
  - used as bridges for system merging in the enterprise integration.
  - Many platforms provide message queue mechanisms. including Unix and Microsoft MQ
  - IBM MQseries, Progress SonicMQ, JBossMQ, and FioranoMQ mplement directly the Java Message Server (JMS)
  - The BEA WebLogic JMS provides additional messaging flexibility over JMS
  - JMS is a typical API in J2EE platform that supports asynchronous messaging including Message-Driven Bean (MDB), a special type of enterprise JavaBean, that asynchronously consumes messages.

ex) 물건 만들때 필요한 재료가 공급처에서 적절히 제공해야. 적절한 타이밍에 주문을 해서 어쩌공.. 큰 어플리케이션 단계에서 일이 일어나면 프로세싱 되는 것.

전사적인 여러 다른 시스템을 합쳐서 하는 경우 메세지로 처리하는 것

Buffered Message-Based Architecture Style

- Point-to-Point Messaging (P2P)
  - A P2P messaging architecture is composed of message queues, senders, and receivers.
  - Each message is sent to a destination (a specific queue) which is maintained by the consumer; consumer clients extract messages from these queues.
  - The queue retains all messages it receives either until the messages are consumed or until the messages expire.
  - A sender and a receiver of a message have no timing dependencies; the receiver can still receive the message even if it was not available when the sender sent the message.
  - It is much more reliable than the simple event-listener based system discussed earlier.



Message queue of JMS

Java Bean : 하나의 오브젝트를 bean으로 표현
자바 언어의 object level을 bean으로 표현
있는형태의 bean을 메세지로 보냄. 다른 종류의프로그램에 오브젝을 보내기 좋은 형태!

Message Driven Bean (MDB) of EJB

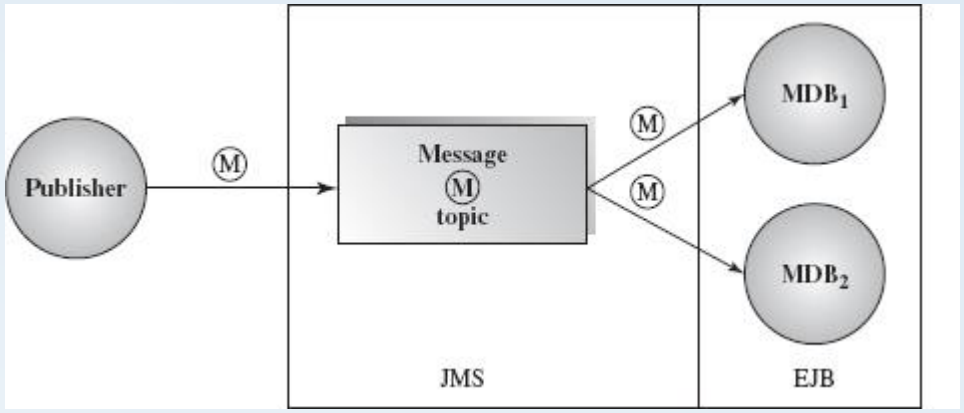
데이터가 사라지는 것을 방지하기 위해
가져갈때 오리지널은 냅두고 카피본만 가져감! 버퍼가 가득 차면
expire_time을 명시해두고 메모리가 아닌 다른 곳에 옮김
- > 여러 컨슈머가 공유 가능, 만료일 설정 가능, 타임에 대한 의존성이 없음, 무척 reliable함(버퍼를 reliable하게 관리 가능)

Buffered Message-Based Architecture Style

한 카피를 여러개 가져갈 수 있다. 메세지를 중복해 가져갈 수 있음
hub - like : 한 데이터 메세지를 다른 컨수머에게 카피카피한 구조를 갖기 때문에 broadcast, multicast하게 보낼 수 있음
프로그램을 확대하고 싶음 (데이터 하나만으로) - - > 버퍼 하나만 놓으면 됨

- Publish-Subscribe Messaging (P&S)
  - The publish-subscribe messaging architecture is a hub-like architecture where publisher clients send messages to a message topic that acts like a bulletin board.
  - Message topic publishers and subscribers are not aware of each other.
  - The system delivers the messages to all of its multiple subscribers instead of to a single receiver, as in the message queue system.
  - Publishers and subscribers have a less timing coupling dependency.

타이밍 커플링에 대한 디팬던시를 끊어줌. - > 디커플 효과



Message topics of JMS

Buffered Message-Based Architecture Style

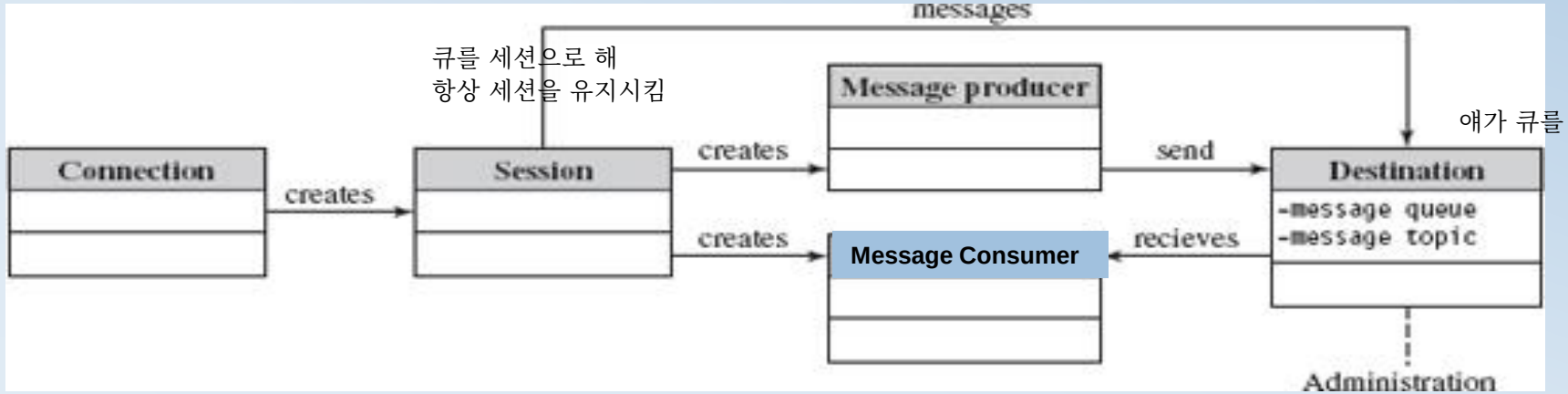
- Implementation
  - Similar to event-based connection architectures, message-based connection architecture is mainly implemented by asynchronous communications
  - A software component can register a *message listener* (similar to an event listener) with a consumer.
  - Whenever a message arrives at the destination, the provider delivers the message by calling the listener's onMessage() method, which performs the message handling operation just like the actionPerformed() method of the class ActionListener for AWT or Swing event in the Java API discussed previously.

어싱크로너스 커뮤니케이션에 의해..

연결할 때 메세지 리스너를 만들어줌. (서버의 버퍼에 대한 리스너)

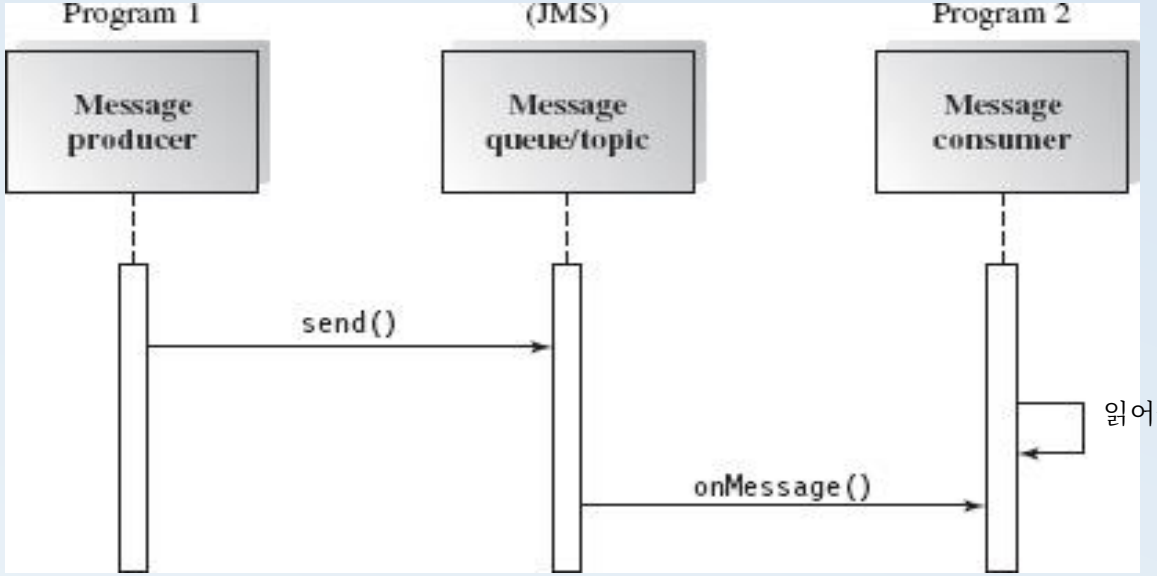
Buffered Message-Based Architecture Style

- Message Queue/Topic in JMS



얘가 큐를 가지고 있음. 토픽도 가짐.

시퀀스 다이어그램



읽어들여서 컨슘~

Buffered Message-Based Architecture Style

읽으심.

- Message Queue/Topic in JMS
  - Message clients (the message producer and message consumer) need to create connections to the message service provider and then create their session on the message server.
    - The message destination (the queue or topic) is created by either administrator or by client programming.
  - The message producer generates a message and sends it to the destination queue or topic; the message consumer receives the message from the destination asynchronously by accepting the call OnMessage().

큐 = 버퍼

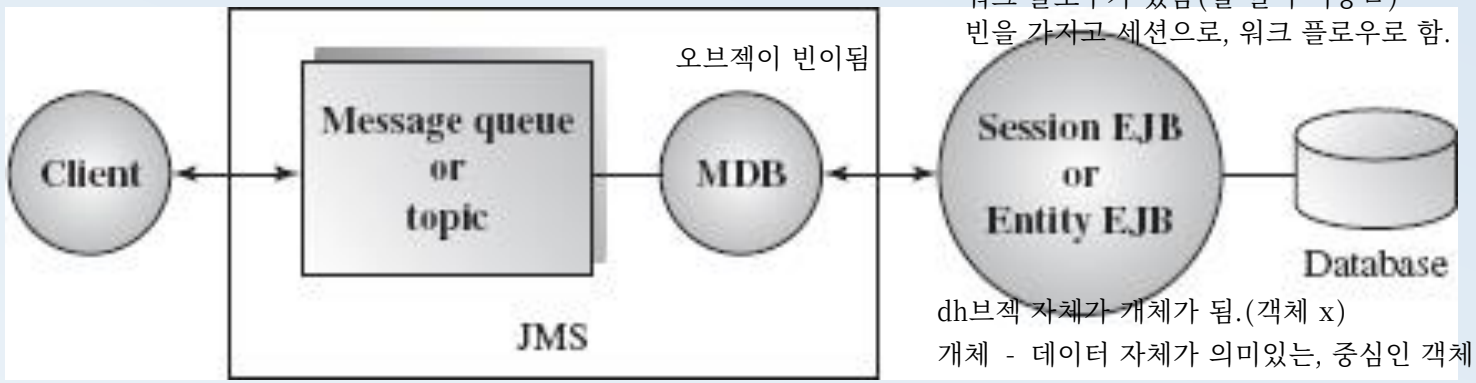
a게 세지를 컨숨할수 있음

Buffered Message-Based Architecture Style

큰 사이즈에 적합!
event에 따라 핸들링 - > 프레임웍에서 쓰는 핸들링 씀

얘는 자주 일어나지 않고 큰 일에 적합!

- Complex enterprise system with JMS
  - Ex) enterprise web online business-to-business (B2B) application
    - The inventory component sends a message to the supplier component when the inventory level for an item drops below a certain level and needs to be replenished.
    - The inventory component also sends a message to the business office component to prepare cash to purchase more inventory items and update the budgets.
    - The supplier component sends a notification message to the catalog component to update the information after the supplier component gets more items in.



워크 플로우가 있음(할 일이 이씅ㅁ)
빈을 가지고 세션으로, 워크 플로우로 함.

오브젝이 빈이됨

dh브젝 자체가 개체가 됨.(객체 x)
개체 - 데이터 자체가 의미있는, 중심인 객체
가 가

Buffered Message-Based: Application

읽으심

- Applicable domains
  - Suitable for a software system where the communication between a producer and a receiver requires buffered message-based asynchronous implicit invocation for performance and distribution purposes.
  - The provider wants components that function independently of information about other component interfaces so that components can be easily replaced.
  - The provider wants the application to run whether or not all other components are running simultaneously.
  - The application business model allows a component to send information and to continue to operate on its own without waiting for an immediate response.

busy - wait 할 필요 없고 필요하면 걍 가저ㅓ감

Buffered Message-Based: Benefits

- **Benefits:**
  - *Anonymity:* provides high degree of anonymity between message producer and consumer. The message consumer does not know who produced the message (user independence), where the producer lives on the network (location independence), or when the message was produced (time independence).
  - *Concurrency:* supports concurrency both among consumers and between producer and consumers.
  - *Scalability and reliability of message delivery:* Reliability mechanisms include: control level setting of message acknowledgement; message persistence setting without loss; message priority level setting; and message expiration setting.
  - Supports batch processing.
  - Supports loose coupling between message producers and consumers, and between legacy systems and modern systems for integration development.

읽으심

그냥 중간에 있는 버퍼만 기억하고 거기에 넣어주면 다른 쪽에서 필요에 의해 가져감. 유저 독립성, 위치 시간 독립성

동시 다발적인 일이 일어남. 콘슈머와 프로듀서 사이에서.
콘커런트 : 다른 종류의 일을 같은 시간대에서 처리할 수 있도록 함.

스캐일어빌리티 : 하다의 일에 대한 성능이 좋아지는 것 ???
메세지 펄시스턴스 : 로그를 띄울 수 이씨 때문에 결함이 생기지 않음(무결성). 메세지가 지속이 됨.
메세지가 중요한 것에 따라, 의미에 따라 세팅할 수 있음. - > 우선순위에 따라 가능..

레가시 시스템 : 옛날 시스템. 이전 버전.
기존과 연동이 가능하도록 맞춰줘야 함. - > 상호 호환적 특성을 살릴 수 있음.

Buffered Message-Based: Limitations

단점이라기보단 Risk라는 표현이 더 맞을 것!

- **Limitations:**

버퍼를 쓸때 가장 중요하고 어려운 일이 큐의 사이즈(캐패시티)를 정하는 것. 개발자가 상황에 맞춰 가장 좋은 구조로 세팅해주어야 함.

  - *Capacity limit of message queue:* This is not an inherent limitation but an implementation issue that can be minimized if the queue is implemented as a dynamic data structure (e.g., linked lists). However, there is an absolute limit based on available memory. It is also difficult to determine the numbers of agents needed to satisfy the loose couplings between agents.

관리하는 쪽과 소비하는 쪽에 다 세파레이트 콘설ㄴ함. 이종의 시스템에 있어 서로연동하기 좋은 구조를 가진다는 뜻. 설계시 복잡한 것을 고민해야 함.

  - Complete separation of presentation and abstraction by control in each agent generates development complexity since communication between agents only takes place between the control of agents.
  - Increased complexity of the system design and implementation.

- **Related architecture:**
  - Event-based system, layered, multi-tier, and MVC

Summary of Implicit invocation architectures

읽으심

Summary of Implicit invocation architectures

- Applications
  - commonly found in user interface design, e-commerce application design, and other distributed transactional business applications where the system involves many pairs of producer-consumer models that are loosely and asynchronously connected.

읽으심.

Summary of Implicit invocation architectures

- Characteristics
  - The coupling between the sender and the receiver in message-based implicit invocation connected system is even looser than the event-based connected system because there are no time constraints dependency.
  - In an event-based invocation system the event handler must be ready in order to be able to intercept an event and take an action upon that event.

이벤트로 할지 버퍼로 할지!
개골! - > 이벤트 핸들링 : 어떤 일이 반복해서 일어나고 빨리 해야 하는 것
버퍼 - > 메세지 : 가끔씩 일어나는 이종의 시스템