Assignment 7: EIP Reading Task

Assignment description

The following glossary is created by using definitions found in: Enterprise Integration Patterns by Gregor Hohpe and Bobby Woolf.

Table of contents

- Message
- Message Channels
- Transformation / Message Translator
- Pipes and Filters
- Pipeline Processing
- Parallel Processing
- Message Route
- Message Endpoints
- Publish–Subscribe Pattern
- Message Oriented Middleware

Message

A message is an atomic packet of data that can be transmitted on a channel. A message consits of two parts:

1. *Header*
   Information that describes the data being transmitted, its origin, its destination etc.
2. *Body*
   The data which is being transmitted.

Message Channels

A messaging application transmits data through a Message Channel, which is a virtual pipe that connects a sender to a receiver. You as a developer must determine how your applications

need to communicate and then create the channels to facilitate it.
The application should have different channels for different types of information.

Transformation / Message Translator

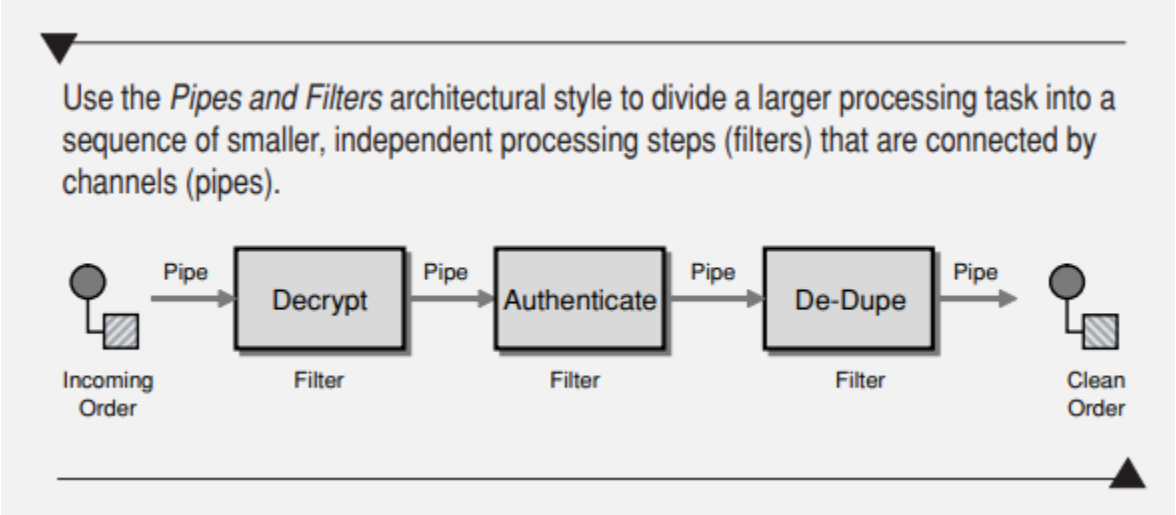
Different applications may not use the same format for data. Therefore a message must be transformed from one format to another.
Different kinds of translators:

Layer	Deals With	Transformation Needs (Example)	Tools/ Techniques
Data Structures (Application Layer)	Entities, associations, cardinality	Condense many-to-many relationship into aggregation.	Structural mapping patterns, custom code
Data Types	Field names, data types, value domains, constraints, code values	Convert ZIP code from numeric to string. Concatenate First Name and Last Name fields to single Name field. Replace U.S. state name with two-character code.	EAI visual transformation editors, XSL, database lookups, custom code
Data Representation	Data formats (XML, name-value pairs, fixed-length data fields, EAI vendor formats, etc.)	Parse data representation and render in a different format.	XML parsers, EAI parser/ renderer tools, custom APIs
	Character sets (ASCII, UniCode, EBCDIC)	Decrypt/encrypt as necessary.	
	Encryption/compression		
Transport	Communications protocols: TCP/IP sockets, HTTP, SOAP, JMS, TIBCO RendezVous	Move data across protocols without affecting message content.	*Channel Adapter* (127), EAI adapters

Pipes and Filters

If a message needs to be authenticated or transformed, you need to chain multiple processings steps together using channels. *Pipes and Filters* describes a fundamental

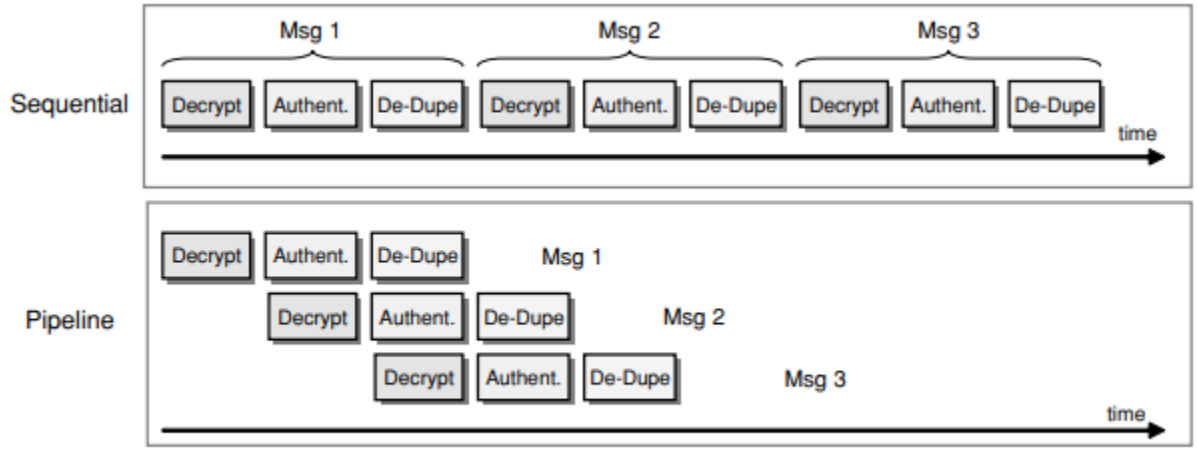
architectural style for messaging systems: Individual processing steps (**filters**) are chained together through the messaging channels (**pipes**).



source

Pipeline Processing

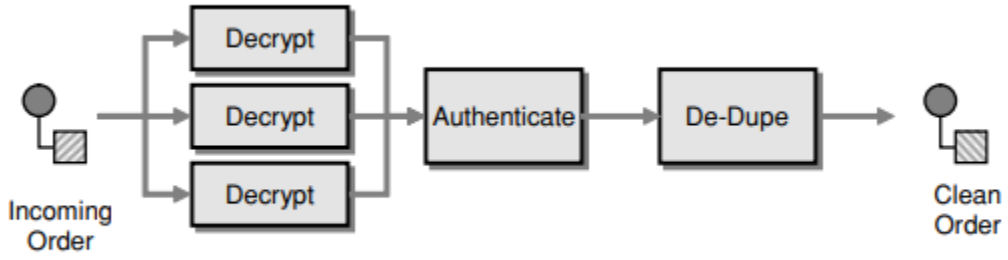
Connecting components with asynchronous Message Channels allows each unit in the chain to operate in its own thread or its own process. When a unit has completed processing one message, it can send the message to the output channel and immediately start processing another message.



Pipeline Processing with Pipes and Filters

Parallel Processing

With *Pipeline Processing* you are limited by the slowest process in the chain. You can deploy multiple parallel instances of that process. You need to be aware that with this method, messages can be processed out of order.

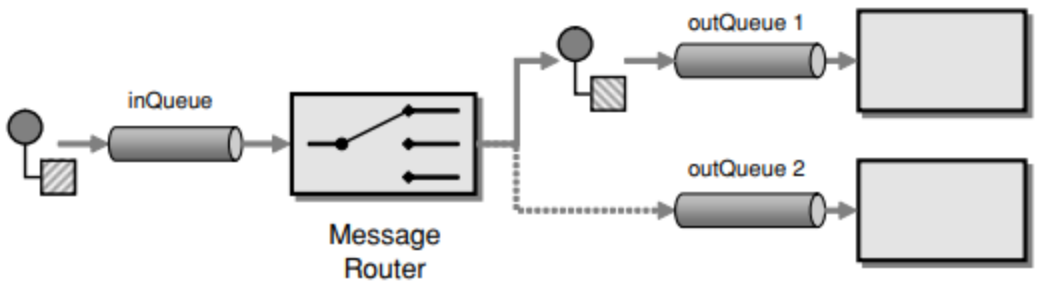


Increasing Throughput with Parallel Processing

Message Route

A message may have to go through several channels to reach its final destination. The sender might not know which channel that will be the final receiver, so it sends its message to a router, which takes the place of a filter in the Pipes and Filters architecture. The router then directs the message to the final receiver or at least to the next router.
The Message Router differs from the basic notion of Pipes and Filters in that it connects to multiple output channels (i.e., it has more than one output port)

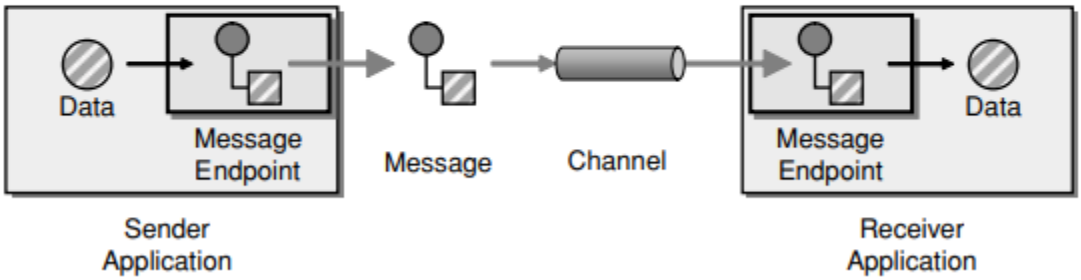
Insert a special filter, a *Message Router*, which consumes a Message from one Message Channel and republishes it to a different Message Channel, depending on a set of conditions.



Message Endpoints

A Message Endpoint is either used to send messages or receive them, but one instance does not do both. An endpoint is channel-specific, so a single application would use multiple endpoints to interface with multiple channels. The Message Endpoint code.

> Connect an application to a messaging channel using a *Message Endpoint*, a client of the messaging system that the application can then use to send or receive Messages.



Publish–Subscribe Pattern

Publish-subscribe pattern is a messaging pattern, where the senders of the message (**publisher**), do not program the messages to be sent directly to specific receivers (**subscribers**), but instead categorize published messages into classes.
In a **topic-based** system, messages are published to "topics" or named logical channels. Subscribers in a topic-based system will receive all messages published to the topics to which they subscribe. The publisher is responsible for defining the topics to which subscribers can subscribe.
In a **content-based** system, messages are only delivered to a subscriber if the attributes or content of those messages matches constraints defined by the subscriber. The subscriber is responsible for classifying the messages.[1]

Message Oriented Middleware

Message Oriented Middleware (**MOM**) is a concept, where you pass data between applications using a coommuncation channel that carries messages. Messages are usually sent and received asynchroonously. [2]

Middleware categories:

- Remote Procedure call (RPC)
- Object Request Broker (ORB)

---

1. https://en.wikipedia.org/wiki/Publish–subscribe_pattern ↩?
2. https://www.oreilly.com/library/view/enterprise-service-bus/0596006756/ch05.html ↩?