PUBLISH SUBSCRIBE MODEL

- Publish/Subscribe systems are nowadays considered a key technology for information diffusion.
- Each participant in a publish/subscribe communication system can play the role of a publisher or a subscriber of information.
- Publishers produce information in form of events, which are then consumed by subscribers.
- Subscribers can declare their interest on a subset of the whole information issuing subscriptions.
- There are two major roles:
    - Publisher
    - Subscriber
- The former provides facilities for the later to register its interest in a specific topic or event.
- Specific conditions holding true on the publisher side can trigger the creation of messages that are attached to a specific event.
- Message will be available to all the subscribers that registered for the corresponding event.
- There are two major strategies for dispatching the event to the subscribers.

Push strategy:

- It is the responsibility of the publisher to notify all the subscribers. Eg: Method invocation.

Pull strategy :

- The publisher simply makes available the message for a specific event.
- It is the responsibility of the subscribers to check whether there are messages on the events that are registered.
- Subscriptions are used to filter out part of the events produced by publishers.
- In Software Architecture, Publish/Subscribe pattern is a message pattern and a network

oriented architectural pattern

- It describes how two different parts of a message passing system connect and communicate with each other.
- There are three main components to the Publish Subscribe Model:
    - Publishers
    - Eventbus/broker
    - Subscribers

Publishers:

- Broadcast messages, with no knowledge of the subscribers.

Subscribers:

- They ‘listen’ out for messages regarding topic/categories that they are interested in without any knowledge of who the publishers are.

Event Bus:

- Transfers the messages from the publishers to the subscribers.

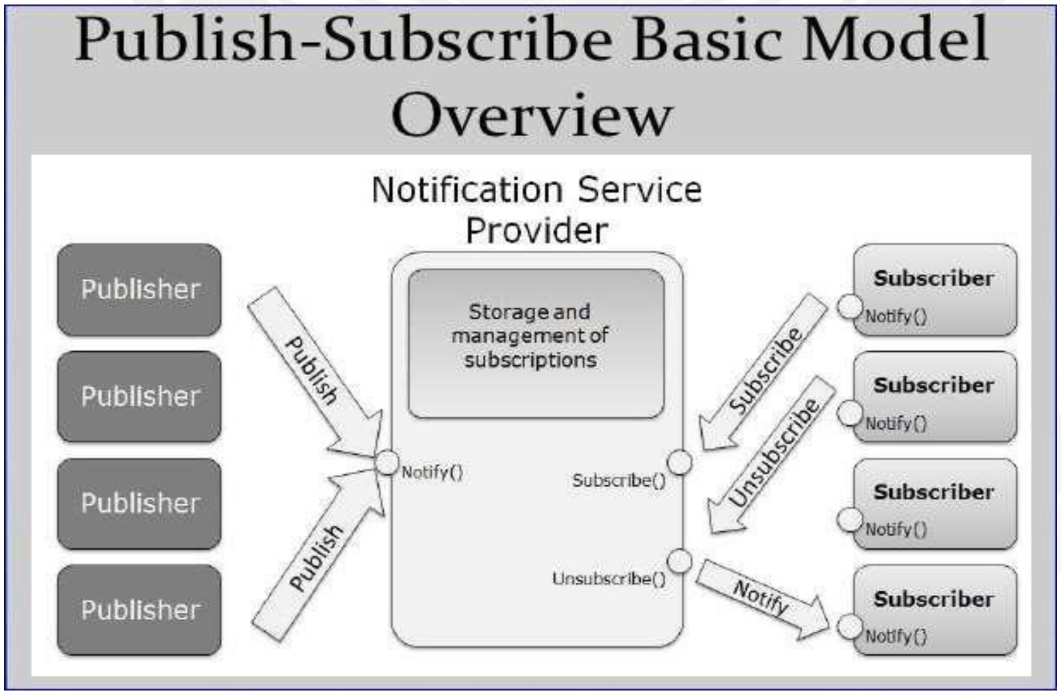

Figure 2.8 Publish Subscribe Model

- Each subscriber only receives a subset of the messages that have been sent by the Publisher.
- Receive the message topics or categories they have subscribed to.
- There are two methods of filtering out unrequired messages:
    - ➢ Topic based filter
    - ➢ Content based filter
    - ➢

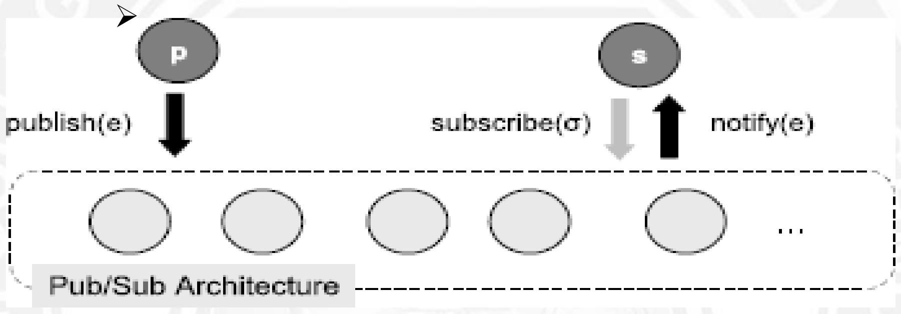

Figure 2.9 High Level View of A Publish/Subscribe System

- A generic pub/sub communication system is often referred as Event Service or Notification Service.
- System composed of a set of nodes distributed over a communication network.
- The clients of this system are divided according to their role into publishers and subscribers.
- Clients are not required to communicate directly among themselves.
- The interaction takes place through the nodes of the pub/sub system.

<u>Elements of a Publish/Subscribe System</u>

- A publisher submits a piece of information e (i.e., an event) to the pub/sub system by executing the publish(e) operation.
- An event is structured as a set of attribute-value pairs.
- Each attribute has a *name, a simple character* string, and a *type.*
- The type is generally one of the common primitive data types defined in programming

languages or query languages (e.g. integer, real, string, etc.).

- On the subscriber’s side, interest in specific events is expressed through subscriptions.
- *A* subscription is a filter over a portion of the event content (or the whole of it).
- Expressed through a set of constraints that depend on the subscription language.
- A subscriber installs and removes a subscription from the pub/sub system by executing the subscribe() and unsubscribe() operations respectively.
- An event e matches a subscription if it satisfies all the declared constraints on the corresponding attributes.
- The task of verifying whenever an event e matches a subscription is called matching.

<u>Semantics of a Publish/subscribe System</u>

- When a process issues a subscribe/unsubscribe operation, the pub/sub system is not immediately aware of the occurred event.
- The registration (resp. cancellation) of a subscription takes a certain amount of time, denoted as Tsub, to be stored into the system.

This time encompass the update of the internal data structures of the pub/sub system and the network delay due to the routing of the subscription.

Three properties:

- Safety (Legality): A subscriber cannot be notified for an information it is not interested in.
- Safety (Validity): A subscriber cannot be notified for an event that has not beenpreviously published.
- Liveness: The delivery of a notification for an event is guaranteed only for those subscribers that subscribed at a time at least Tsub before the event was published.

<u>Quality of Service in Publish/Subscribe Systems</u>

- Reliable delivery
- Timeliness
- Security and trust

Reliable delivery

- Reliable delivery of an event means determining the subscribers that have to receive a published event, as stated by the liveness property and delivering the event to all of them.

Timeliness

- Real-time applications often require strict control over the time elapsed by a piece of information to reach all its consumers.
- They are typically deployed over dedicated infrastructures or simply managed environments where synchronous message delivery can be safely assumed.

Security and trust

- A subscriber wants to trust authenticity of the events it receives from the system.
- Generated by a trusty publisher and the information they contains have not been corrupted.
- Subscribers have to be trusted for what concerns the subscriptions they issue.
- Since an event is in general delivered to several subscribers, the producer/consumer trust relationship that commonly occur in a point-to-point communication, in pub/sub system must involve multiple participants

Subscription Models

- Topic based Model
- Type based Model
- Concept based Model
- Content based Model

Topic-based Model

- Events are grouped in topics.
- A subscriber declares its interest for a particular topic to receive all events pertaining to that topic.
- Each topic corresponds to a logical channel ideally connecting each possible publisher to all interested subscribers.
- Requires the messages to be broadcasted into logical channels.
- Subscribers only receive messages from logic channels they care about (and have

subscribed to).

Type based Model

- Pub/sub variant events are actually objects belonging to a specific type, which can thus encapsulate attributes as well as methods.
- Types represent a more robust data model for application developer.
- Enforce type-safety at the pub/sub system, rather than inside the application.
- The declaration of a desired type is the main discriminating attribute.

Concept based Model

- Allows to describe event schema at a higher level of abstraction by using ontologies.
- Provide a knowledge base for an unambiguous interpretation of the event structure, by using metadata and mapping functions.

Content based Model

- System allows subscribers to receive messages based on the content of the messages.

Subscribers themselves must sort out junk messages from the ones they want.

<u>Benefits</u>

Loose coupling

- The publisher is not aware of the number of subscribers, of the identities of the subscribers, or of the message types that the subscribers are subscribed to.

Improved security

- The communication infrastructure transports the published messages only to the applications that are subscribed to the corresponding topic.
- Specific applications can exchange messages directly, excluding other applications from the message exchange.

Improved testability.

- Topics usually reduce the number of messages that are required for testing.

Separation of concerns

- Due to the simplistic nature of the architecture, developers can exercise fine grained separation of concerns by dividing up message types to serve a single simple purpose each.
- Eg. data with a topic “/cats” should only contain information about cats.

Reduced cognitive load for subscribers

- Subscribers need not concern themselves with the inner workings of a publisher.
- They do not even have to access to the source code.
- Subscribers only interact with the publisher through the public API exposed by the publisher.

<u>Drawbacks</u>

Increased complexity.

Publish/Subscribe requires you to address the following:

- To design a message classification scheme for topic implementation.
- To implement the subscription mechanism.
- To modify the publisher and the subscribers.

Increased maintenance effort.

- Managing topics requires maintenance work.
- Organizations that maintain many topics usually have formal procedures for their use.

Decreased performance

- Subscription management adds overhead.
- This overhead increases the latency of message exchange, and this latency decreases performance.

Inflexibility of data sent by publisher

- The publish/subscribe model introduces high semantic coupling in the messages passed by the publishers to the subscribers.
- Once the structure of the data is established, it becomes difficult to change.
- In order to change the structure of the messages, all of the subscribers must be altered to accept the changed format

Instability of Delivery

- The publisher does not have perfect knowledge of the status of the systems listening to the messages.
- For instance, publish/subscribe is commonly used for logging systems.
- If a logger subscribing to the ‘Critical’ message type crashes or gets stuck in an error state, then the ‘Critical’ messages may be lost!
- Then any services depending on the error messages will be unaware of the problems with the publisher.

<u>Applications</u>

Used in a wide range of group communication applications including

- Software Distribution
- Internet TV
- Audio or Video-conferencing
- Virtual Classroom
- Multi-party Network Games
- Distributed Cache Update

It can also be used in even larger size group communication applications, such as broadcasting and content distribution.

- News and Sports Ticker Services
- Real-time Stock Quotes and Updates
- Market Tracker
- Popular Internet Radio Sites