SOA VS EDA

Service Oriented Architecture (SOA)

SOA is a way of designing software system where independent software components provide services to end-users or other software components.

A service can be defined as a unit of functionality that is self-contained, discoverable and can be dynamically invoked.

SOA has three basic participants:

- service consumer (Contract who needs to invoke the service)
- services provider (who provides service to the contractor)
- service registry. (Registry where all service provides enrolled)

The interaction among these participants involve publish, find and bind operations

Service provider defines the service description and publishes it to a service registry. Using the find operation, service requester/consumer discovers the service description on service registry. Finally, the service requester/consumer invoke the service from service provider using the bind operation.

SOA follows request and reply patterns

it is focused on synchronous communication approach

Event Driven Architecture (EDA)

EDA is an architecture design where services of independent software components communicate through event notifications.

EDA as "a methodology for designing and implementing applications and systems in which events transmit between decoupled software components and services".

EDA uses publish-subscribe architecture to enable the communication among services and to end-users. It has three main components: event emitter, event broker and event subscriber.

An event emitter detects events and posts an announcement to the event broker. The event broker collects all the triggered events and forwards them to interested subscribers.

Finally, event subscribers receive event notifications and respond accordingly. Event subscribers are able to further trigger the event to other services.

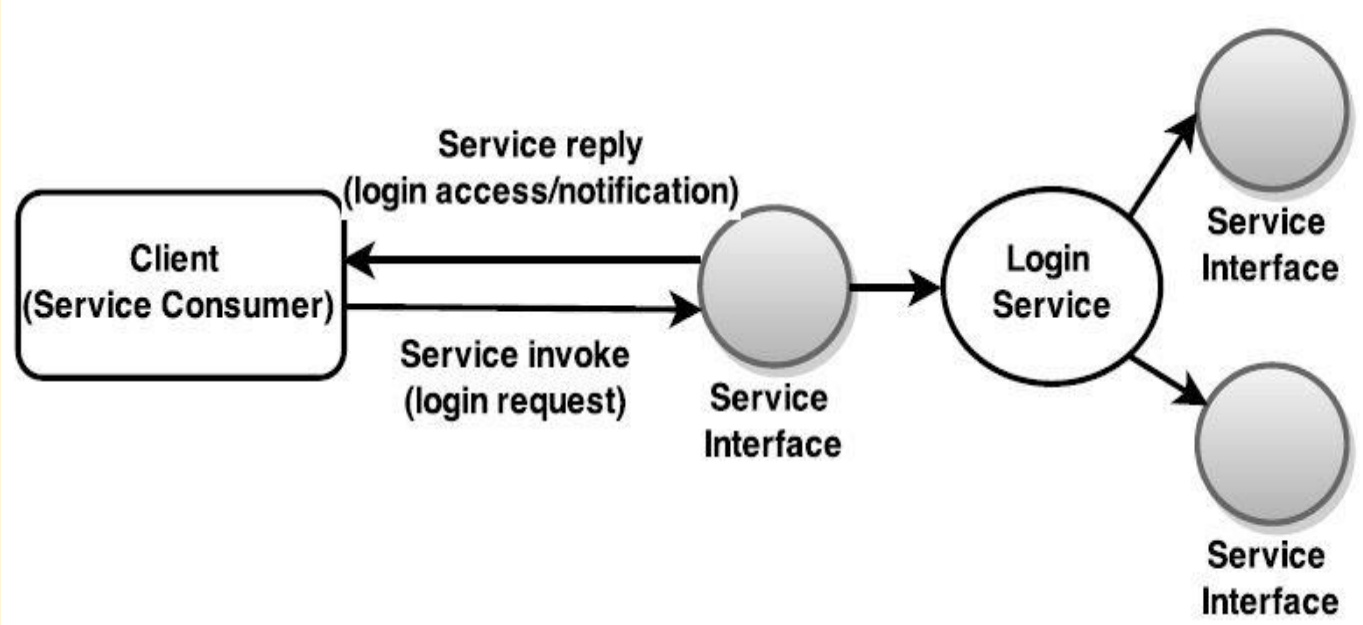
EDA follows publish and subscribe pattern

it is focused on asynchronous communication approach

Explanation:

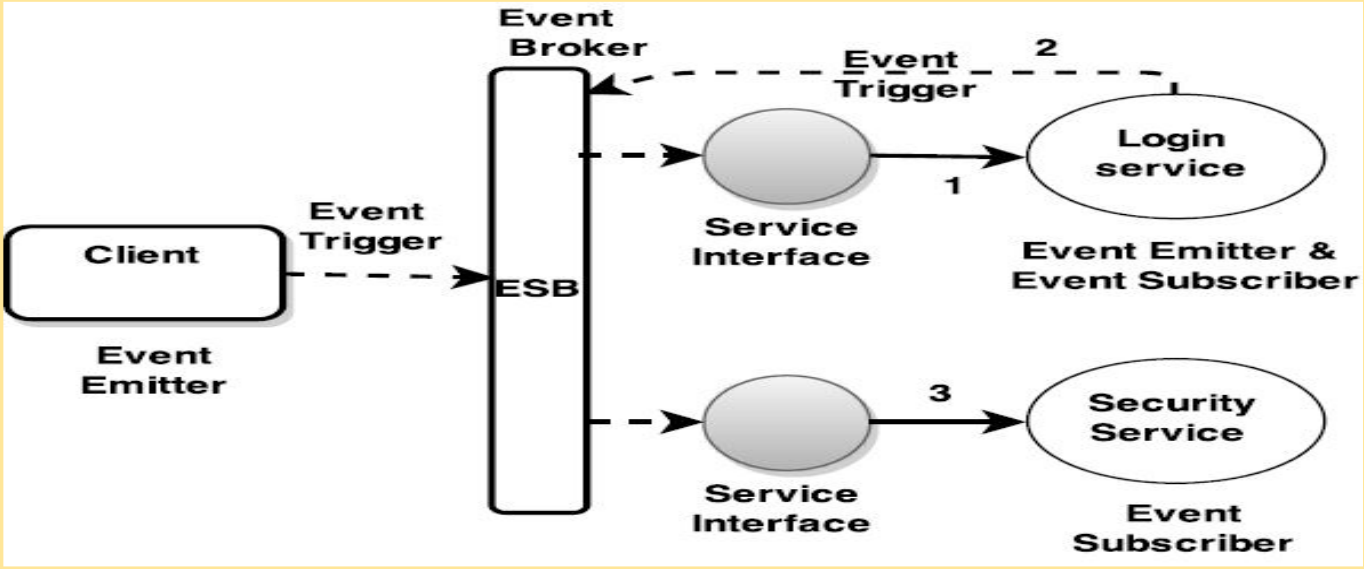
Explaining with a simple example of an 'authentication process'. This process verifies the login credentials of users in order to ensure that right person get access to computer system.

The 'authentication process' scenario from the SOA respective, where a client, which is a service consumer, requests a login service to get access to the system through the service interface. In response, the login service, which is a service provider, can invoke other services through service interfaces in order to send a random code to an e-mail/mobile for further verification. The service consumer is bounded to wait for and act on the response/reply from the service provider before access is possible. Thus, even though, the service consumer and service provider do not have interdependencies yet both are bounded during a communication.



Service Oriented Architecture of 'Authentication Process'.

The ‘authentication process’ from the EDA perspective. The client request the login service by the providing login credentials. In case of incorrect credentials, the login service will trigger the event to security service through the enterprise service bus (ESB). In addition to providing the connection among services, ESB is also able to function as an event broker. Event broker analyse the triggered event and forward it to security service. In this scenario, login service is event emitter while the security service is event subscriber.



Event driven Architecture of ‘Authentication Process’.

On the other hand, the security service (i.e. event subscriber) might take further security action by temporary blocking the account. In this scenario, the event emitter (login service) is not bounded to listen to the event subscriber’s action/response (security service). Moreover, the role of event emitter and event subscriber are not mutually exclusive. A service (e.g. login service) can be event subscriber as well as event emitter at the same time.

From high level of abstraction, it can be said that the pattern to invoke the services in SOA and EDA are different as in SOA user command and other services can invoke the service while in EDA the service can be invoked with real-time event.

Both architectures require an underlying infrastructure, a bus to carry requests in applications network, and some business processing rules.

analogy of the human body, where eyes and ears are similar to EDA as sensing the events and sending it to brain, while hands and feet are similar to SOA as providing movement on request of sense neurons.

Category	**SOA**	**EDA**
Interaction approach	Service needs to be available when service consumer request it	Event's subscriber doesn't need to be available when event is triggered
Invocation approach	One service consumer can initiate one service at a time	Event can trigger many subscribers at a time
Interaction pattern (from consumer)	Service consumer request specific service and wait for response	Event emitter triggers the event and doesn't wait for response. Emitter doesn't have knowledge on who are its subscribers
Interaction pattern (from provider)	Service provider response back with services/notification	Event subscriber takes action but event emitter is not necessarily aware of it

THANK YOU!

DO Follow For More Such Content

Cheers,

Hanuma.