casual

uab 2024

What is casual?

In general terms casual is a distributed application server. A platform to build applications on top of, that can interact with each other in a distributed manner, with or without transactional context.

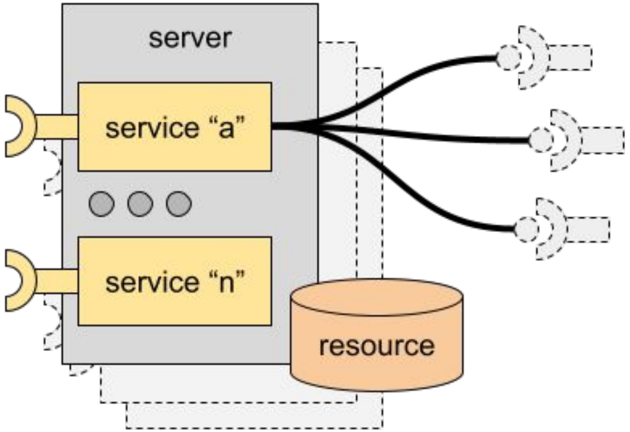
Users build logical applications that consist of one or more servers that advertise arbitrary named services. Servers are deployed in a domain. Domains can be connected with each other, on different machines, in any topology.

casual

…What is casual?

A server is just an executable with an entry point for each advertised service. A server is scaled by spawning processes of the executable. Hence, servers can be dynamically scaled to meet load requirements. Resources such as databases can also be associated with a server.

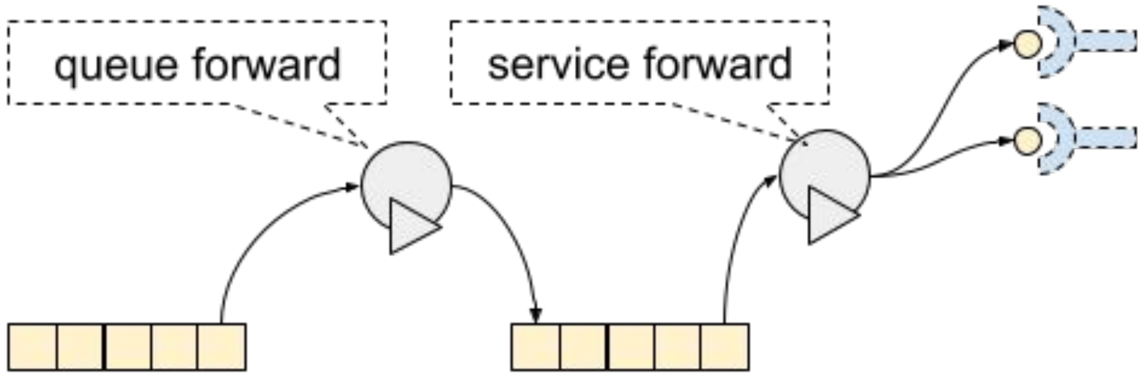
A service can call other services, and casual will find where these services reside in the topology.



casual

…What is casual?

casual provides a queue implementation together with *queue-to-service* and *queue-to-queue forward* functionality.



Why use casual?

The semantics are simple and easy to reason about.

Applications can be scaled to fit any need. Asynchronous service calls give massive concurrency without the hassle of threads.

With the building blocks: services, queues, queue-to-service and queue-to-queue all relevant communication patterns can be constructed.

Manage casual in a UNIX-friendly way that enables interoperability with familiar tools.

Specifications

casual conforms to the following specifications:

- xa
- xatmi
- tx

These specifications have been proven in low latency, high throughput transactional systems for several decades.

1.7

Release notes.

file:///Users/lazan/git/casual/1.7/casual/documentation/release-notes.md

Demo

Managers

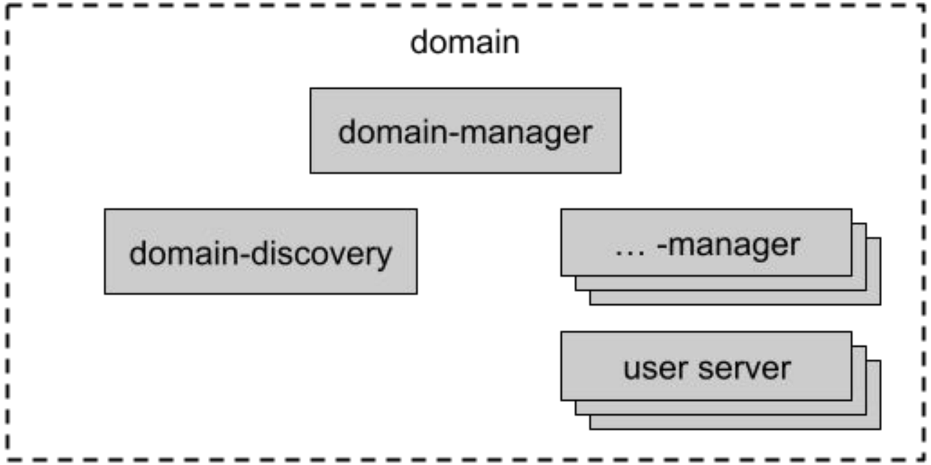
casual consists of a few managers. A manager is in essence an executable that has a certain area of responsibility.

- casual-domain-manager
- casual-service-manager
- casual-queue-manager
- casual-gateway-manager
- casual-transaction-manager

The name of the manager indicates their main responsibility. Some of the managers delegate responsibility to subprocesses.

casual-domain-manager

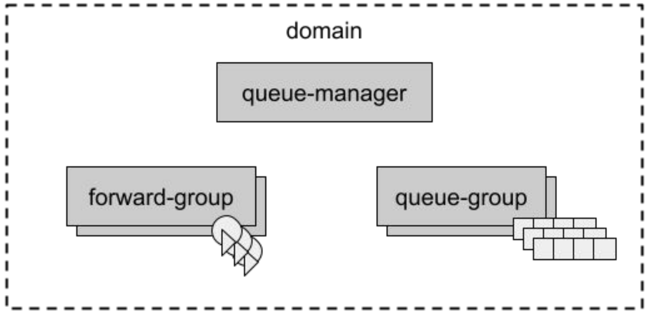
- handle the domain configuration
- startup and shutdown of the domain
- scale instances and other configuration updates
- answer process lookup requests
- casual-domain-discovery
    - coordinates service/queue discovery to and/or from other domains



casual-service-manager

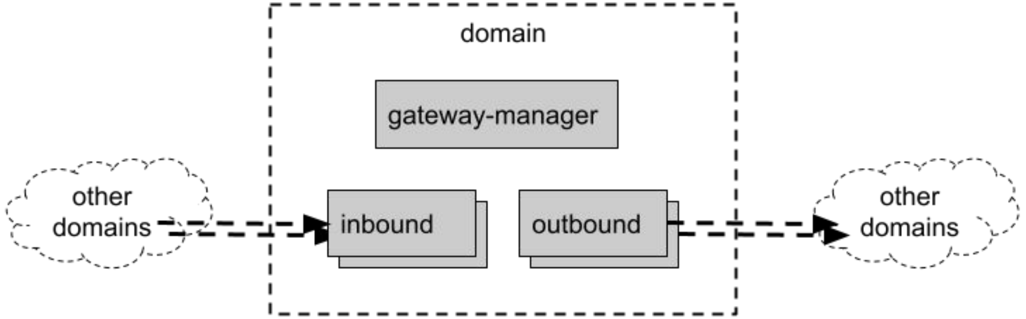
- keep track of addresses to services, including remote
- answer service lookup requests

casual-queue-manager



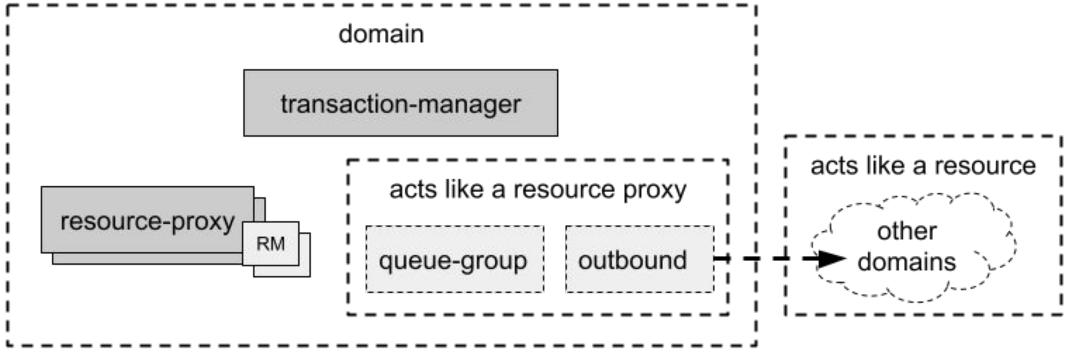
- keep track of addresses to queues, including remote
- answer queue lookup requests
- manage queue-groups and forward-groups
- casual-queue-group
  - manages configured queues for the group
- casual-queue-forward-group
  - a forward dequeues from a queue and call/enqueue to a service or queue
  - manages configured service/queue-forwards for the group

casual-gateway-manager



- manage inbound and outbound groups
- casual-gateway-outbound-group
  - connects to configured addresses
  - route internal messages/request to the correct connection
- casual-gateway-inbound-group
  - listen on configured addresses
  - route external messages/requests to the correct recipient in the domain
- casual-gateway-outbound/inbound-reverse-group
  - same as regular outbound/inbound but the connection phase is reversed. outbound listen, inbound connects.

casual-transaction-manager



- coordinate distributed transactions
- manage resource proxies
- casual-resource-proxy
  - interact with linked resource via XA
- act as a resource on behalf of other domains

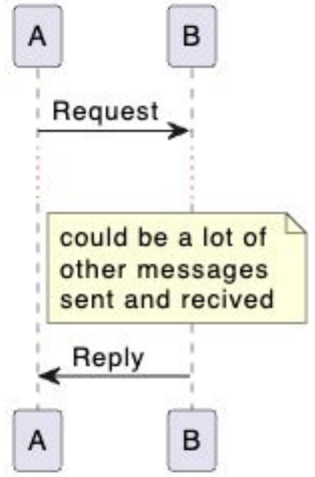
Communication model

Communication between entities (processes) is done with messages passing. Every entity has an inbound-ipc-device that others can pass messages to.

All communication is done asynchronously, hence there could be a lot of messages in flight within any given time.

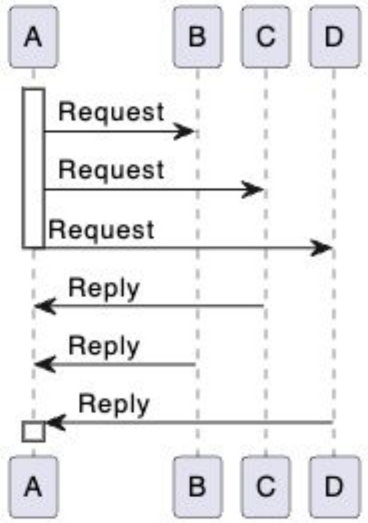
We’ll go through the most significant communication patterns. These will help build a mental model that increases understanding of how casual works.

Request-Reply



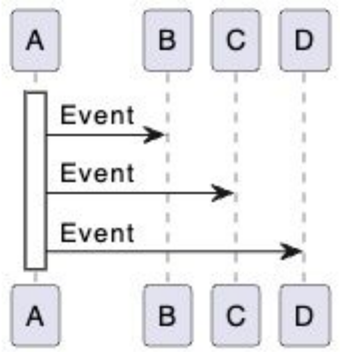
Caller sends a message to the ipc-address of the callee entity. The request message has the ipc-address of the caller. Callee will reply to the caller address.

Fan-Out



Some scenarios require requests to multiple entities, for instance when the transaction-manager sends prepare/commit/rollback to all involved resources for a transaction.

Events



Some messages are one way propagation, which we mostly call **events**. For instances when a process dies, domain-manager sends process-died events to all entities that have registered that they are interested in the event.