Vortex [Blue] - Tests

Este doc intenta capturar distintos casos de test para evaluar en los distintos lenguajes que se implemente y verificar que la implementación es correcta y cumple al menos con las expectativas iniciales

# Tests

He intentado empezar desde lo más básico en lo que serían casos de uso de Vortex e ir incrementando la complejidad y rareza de los casos.

Las definiciones de casos de tests están agrupadas arbitrariamente por similitud y en algunos casos vienen acompañados de gráficos para ayudar a la descripción textual.  
Los gráficos de red utilizarán la siguiente simbología:



**Vortex**: Representa la red y las librerías. Cuando un mensaje es enviado a vortex representa una asbtracción de lo que vortex hace con él

**Mensaje:** Representa un mensaje que circula en la red. Cada paso es representado como un arco distinto entre nodos

**Intereses:** Representan con distintos colores, definiciones de conjunto distintas, de manera que los que coinciden en color coinciden en el tipo de mensaje que manejan. (Dos nodos con colores distintos no deberían intercambiar mensajes)

**Emisor:** Representa un nodo en el rol de emisor. No quiere decir que es sólo emisor, sino que en ese diagrama actúa como emisor de un mensaje. (Se usa el color azul)

**Receptor esperado:** Representa un nodo al cual debería llegarle el mensaje enviado. (Se usa el color verde)

**Receptor accidental:** Representa un nodo al cual le llega el mensaje sin intención. Este tipo de receptores no debería existir en una comunicación ideal, pero es posible que suceda en la realidad.

**Nodo no afectado:** Representa un nodo de la red el cual no participa de la comunicación aunque es parte de la red física

**Conexión:** Representa una conexión física entre nodos que no es usada necesariamente para la comunicación

## A01 - Comunicar dos partes asíncronamente

Vortex debería permitir enviar y recibir cualquier objeto entre dos partes evitando una dependencia temporal entre ellas.



Condiciones a verificar:

1. El emisor debería poder enviar por vortex cualquier objeto serializable[[1]](#footnote-1)
2. El receptor debería poder recibir de vortex cualquier objeto serializable
3. El thread del emisor no debería bloquearse durante la entrega del mensaje
4. El thread del receptor debería poder ser independiente del usado para la entrega del mensaje
5. El emisor no debería recibir su propio mensaje
6. En memoria, el tiempo de entrega normal[[2]](#footnote-2) debería ser inferior a 1ms (o 1000 mensajes por segundo)
7. El emisor debería poder saber cuándo es entregado el mensaje si lo indica[[3]](#endnote-1)
8. El receptor debería recibir los mensajes en el mismo orden que los envío el emisor[[4]](#endnote-2)

## A02 – Comunicación de eventos

Vortex debería ser usable como un componente para generación y notificación de eventos. Permitiendo a las partes comunicantes indicar qué tipos de eventos generan y reciben.  
Para permitir este caso de uso, los eventos se tratan como mensajes entre las partes.



Condiciones a verificar:

1. El emisor debería poder declarar el tipo de mensajes que emite
2. El receptor debería poder especificar el tipo eventos que recibe
3. El emisor debería poder enviar mensajes aunque no estén dentro de los declarados
4. El receptor debería poder recibir mensajes no declarados por el emisor si están en el conjunto declarado como recibible
5. El emisor debería ser notificado si existe en la red al menos un receptor interesado en el tipo de mensajes declarado
6. El receptor debería ser notificado si existe en la red al menos un emisor del tipo de mensajes declarado
7. El emisor debería ser notificado si no existe ningún receptor en la red del tipo de mensajes declarado
8. El receptor debería ser notificado si no existe en la red ningún emisor del tipo de mensajes declarado
9. El emisor NO debería ser notificado si existen receptores no interesados en su tipo de mensajes
10. El receptor NO debería ser notificado si existen emisores de otro tipo de mensajes
11. El emisor NO debería ser notificado si no existen receptores de otro tipo de mensajes
12. El receptor NO debería ser notificado si no existen emisores de otro tipo de mensajes
13. El mensaje enviado por el emisor debería llegar a los receptores interesados
14. El mensaje enviado NO debería llegar a los receptores NO interesados
15. En memoria el tiempo de entrega debería ser inferior a 500 mensajes por segundos (o 2ms)

## A03 – Especificación de intereses

La declaración del tipo de mensajes enviados, así como el tipo de mensajes recibidos se hace a través de la descripción de un conjunto como una serie de restricciones que deben cumplir los mensajes de ese conjunto.

Las restricciones son expresables a través de operadores que al ser evaluados devuelve un valor de verdad (TRUE o FALSE) y pueden ser combinadas entre sí con los operadores booleanos básicos: AND, OR y NOT, resultando en un único valor de verdad final que indica si el mensaje pertenece o no al conjunto.

Cada nodo puede implementar el conjunto de operadores que más le convenga[[5]](#footnote-3) y no existe un mínimo[[6]](#footnote-4). Pero debe asegurarse que en la red los operadores son implementados con la misma semántica en todos los nodos, caso contrario el ruteo no es fiable. Cada nodo puede también agregar operadores nuevos que otros nodos pueden utilizar, pero debe asegurarse que su semántica no tiene contradicciones en la red[[7]](#footnote-5).

Las restricciones del conjunto serán evaluadas sobre una meta-descripción del mensaje, no sobre el mensaje directamente. Esta meta-descripción estará expresada en un lenguaje común a todos los nodos describiendo un objeto como conjunto de propiedades con valores que pueden ser también sub-objetos.

Si el nodo evalua un mensaje con una descripción para la cual no tiene todos los operadores, debe realizar su mejor esfuerzo lógico para determinar si el mensaje pertenece o no, pero ante la incertidumbre (si es indecidible) debe dejar pasar el mensaje. Esto permite la convivencia de nodos con distinto nivel de discriminación semántica

Condiciones a verificar:

1. Se debería poder evaluar si una descripción de mensaje pertenece a una definición de conjunto
2. Se debería poder evaluar si una definición de conjunto intersecta con otra definición
3. Si no declara conjunto de envío, el emisor tendrá implícito el conjunto universal (puede enviar cualquier cosa)
4. Si no declara conjunto de recepción, el receptor tendrá implícito el conjunto universal (puede recibir cualquier cosa)
5. Se debería poder filtrar por la existencia de propiedades. Sólo da true si el mensaje contiene el atributo indicado en su descripción
6. Se debería poder filtrar por valor equivalente de una propiedad. Sólo da true si la propiedad del mensaje tiene un valor equivalente al indicado. (Operador “=” )  
   Si el valor es numérico, la propiedad debe tener el mismo número.  
   Si el valor es textual, la propiedad debe tener el mismo texto (considerando espacios)  
   Si el valor es un objeto, la propiedad debe tener un objeto equivalente (mismas propiedades con mismos valores).  
   Si el valor es un array o colección, la propiedad debe tener un array o colección con valores equivalentes en el mismo orden
7. Se debería poder filtrar por un conjunto de valores equivalentes en una propiedad. Sólo da true si el valor de la propiedad es equivalente a alguno de los indicados
8. Se debería poder filtrar utilizando operadores relacionales sobre una propiedad. [<,>,=,!=,<=, >=]
9. Se debería poder filtrar utilizando los operadores booleanos: [||,&&,!]
10. Se debería poder componer operadores para indicar el orden de evaluación. Por ejemplo si se indican 3 expresiones, cuales 2 se evalúan primero[[8]](#footnote-6).
11. En memoria se debería poder filtrar por jerarquía de clases. Es decir, evaluar si el mensaje es instancia de una clase o interfaz.
12. Se debería poder filtrar utilizando expresiones regulares. Si la propiedad es de texto, debería ser posible aplicar una expresión regular para evaluar si matchea.
13. Si un nodo no posee la capacidad de evaluar un operador, pero puede determinar el valor de verdad del conjunto debería descartar o dejar pasar el mensaje que corresponda
14. Si un nodo no posee la capacidad de evaluar un operador impidiendo determinar el valor de verdad del conjunto, debería dejar pasar el mensaje

## A04 – Modificación de intereses on-line

Un nodo de la red puede modificar sus intereses en el tiempo, y el resto de la red debería adaptarse a esos cambios de la mejor manera posible

Condiciones a verificar:

1. Si un receptor cambia su conjunto de mensajes debería recibir los mensajes que cumplen con él, y no los que no
2. Si un nodo emisor reduce el conjunto de mensajes emitidos debería ser notificado si pierde todos sus receptores
3. Si un nodo emisor aumenta el conjunto de mensajes emitidos debería ser notificado si comienza a tener receptores
4. Si un receptor reduce el conjunto de intereses debería ser notificado si pierde todos sus emisores
5. Si un receptor aumenta el conjunto de sus intereses debería ser notificado si comienza a tener emisores

## A05 – Nodos con semántica heterogénea

Los nodos que conforman la red pueden tener distinta implementación de los operadores pero los mensajes deberían llegar igual al destino correcto



Condiciones a verificar:

1. Si un nodo intermedio tiene un subconjunto de los operadores usados, los mensajes deberían llegar a receptores similares. Dicho de otra manera, si el nodo intermedio no puede discriminar la diferencia, el mensaje es entregado a ambos (C y D)
2. Si a un nodo intermedio se le agrega la capacidad de evaluar un operador, entonces su ruteo debería mejorar. (sólo llegar a C)

## A06 – Respuesta a cambios en la red

Cuando la topología de la red se modifica, ya sea agregando o quitando nodos, Vortex debe realizar el mejor esfuerzo para garantizar la entrega de mensajes a los receptores esperados.

Condiciones a verificar:

1. Si un receptor se conecta antes de que se desconecte el último, el emisor no debería ser notificado del cambio  
   
2. Si un receptor se desconecta después que se desconecte el último, el mensaje enviado desde el emisor debería llegar al nuevo receptor
3. En una larga cadena de nodos si un receptor se conecta en la punta al mismo tiempo que el emisor manda un mensaje, el mensaje debería llegar al nuevo conectado sólo si existía un camino previo antes (si no, es posible que llegue pero no puede asegurarse).[[9]](#footnote-7)
4. Si un camino entre dos nodos está optimizado, al conectarse un tercero lejano, debería recibir el mensaje enviado por el emisor  
   
5. Si un camino entre dos nodos está optimizado y se pierde un nodo intermedio, el mensaje debería llegar si hay caminos alternativos
6. Si un camino entre dos nodos se pierde y no hay caminos alternativos el mensaje no debería llegar
7. Si se pierde el camino entre receptor y emisor ambos deberían ser notificados que perdieron a su contraparte[[10]](#footnote-8)
8. Si un nodo intermedio se agrega, permitiendo conectar dos redes independientes, los mensajes debería llegar de una a otra
9. Si se crea un camino entre receptor y emisor agregando un nodo intermedio ambos deberían ser notificados de la existencia del otro[[11]](#footnote-9)
10. Si una red de tipo triángulo, pierde la conexión directa de una de las puntas, el mensaje debería llegar por la conexión indirecta  
    

## A07 – Topologías varias

En distintas configuraciones de redes (ya que es una red con topología emergente) Vortex debería hacer correcta entrega de los mensajes

Condiciones a verificar:

1. En un red con nodo intermedio el mensaje debería llegar desde el emisor al receptor pasando por el intermedio

1. Un objeto serializable es cualquier que pueda ser representado en un formato apropiado y ser reconstruido en otro ambiente sin perder la semántica. Obviamente esta definición es discutible, porque depende de la capacidad de serializar. Lo importante es destacar que ciertos objetos puede ser serializados fácilmente (si están conformados por datos) y otros requieren otros mecanismos o quizás no sean serializables si son conexiones a base de datos, archivos, etc. [↑](#footnote-ref-1)
2. Normal se refiere al tiempo que le lleva una vez que está inicializado y andando normalmente [↑](#footnote-ref-2)
3. No van a estar en la etapa inicial. Quizás se introduzcan a futuro, o quizás nunca hagan falta [↑](#endnote-ref-1)
4. No van a estar en la etapa inicial. Quizás se introduzcan a futuro, o quizás nunca hagan falta [↑](#endnote-ref-2)
5. Dependiendo del hardware y del lenguaje de programación es posible que algunos operadores sea difíciles de implementar. En la medida de los posible deberían implementarse la mayoría para permitir una mejor semántica en el ruteo de mensajes [↑](#footnote-ref-3)
6. De hecho, un nodo puede no implementar ningún operador y sería válido (en ese caso actúa como relay) [↑](#footnote-ref-4)
7. Es decir, un operador no puede dar distintos valores de verdad para el mismo mensaje en distintos nodos de la red [↑](#footnote-ref-5)
8. El quivalente a utilizar paréntesis [↑](#footnote-ref-6)
9. Como es posible que el mensaje sea descartado en el medio si no hay camino previo, es posible que nunca llegue a destino. En cambio si ya existía uno, debería entregarse si la conexión se produce antes de que llegue el mensaje al nodo intermedio [↑](#footnote-ref-7)
10. Obviamente sólo si son los últimos [↑](#footnote-ref-8)
11. Obviamente sólo si son los primeros [↑](#footnote-ref-9)