Vortex [Blue]

Este doc intenta capturar los nuevos conceptos que surgen de repensar vortex e intentan definir lo que sería la segunda versión de la red

# Qué es vortex?

A partir de la experiencia previa con vortex llegamos a ciertas bases que definen un poco el espíritu de vortex en su nueva generación. Estos puede resumirse en:

“Vortex es a la vez, una red de comunicaciones entre componentes de software, el protocolo que utiliza esa red, la implementación modelo de la misma y el conjunto de servidores que la ejecuta”

### Vortex como red entre componentes: “Red Vortex”

La idea principal detrás de Vortex es que permita la comunicación entre distintas partes de los sistemas en cualquier escala.   
Tanto si es una comunicación intra, o inter-aplicación Vortex debe ofrecer las metáforas necesarias para abstraer esa diferencia y permitir que exista la interacción de componentes aún cuando fueran escritos en distintos lenguajes de programación.   
  
Este punto es clave ya que es la motivación principal de que exista Vortex.  
Creemos que la comunicación así planteada permite la reducción del acoplamiento entre componentes (ayudando a mejorar el diseño), aumenta la escalabilidad de los sistemas (permitiendo dividir la necesidad de computo en distinto hardware), aumenta la flexibilidad de los sistemas (permitiendo agregar elementos en la comunicación a posteriori sin modificar el código original), y sobretodo posibilita la existencia de sistemas distribuidos con bajo costo de desarrollo (brindando una base para la interacción de las partes).

### Vortex como protocolo: “Protocolo Vortex”

Para satisfacer las ideas detrás de vortex y algunos principios que fuimos adquiriendo con los distintos prototipos, hemos desarrollado el protocolo Vortex que busca dar soporte a la red de componentes.

Este protocolo es un trabajo en progreso y es posiblemente el principal punto de cambio durante la vida de Vortex. Con lo cual es posible que siga cambiando.  
A partir de esta definición, y a partir del éxito de la implementación posiblemente vayamos congelando la especificación de manera de poder sumar otros lenguajes sin romper lo que ya está hecho.

### Vortex como implementación: “Librerías Vortex”

Tanto el protocolo como la red empezarán funcionando con una implementación base, que son las pruebas de concepto de estas ideas.  
Se desarrollarán dos versiones, una en .Net y otra en Java, de la implementación ya que son los lenguajes que más conocemos los que hacemos Vortex. Lo cual no quita la posibilidad de otras implementaciones.

A medida que vayamos asegurando los conceptos y congelando el protocolo la idea es incorporar más implementaciones en distintos lenguajes de manera de permitir la participación de más aplicaciones en la comunicación común.  
Estas implementaciones pasarán a formar parte de un conjunto de librerías open source en distintos lenguajes que sean utilizables por código externo, de modo que utilicen la red sin necesidad de conocer los detalles.

### Vortex como servidores: “Servidores Vortex”

De las implementaciones que desarrollemos vamos a poner disponibles en la red las que podamos de manera de usarlas en aplicaciones internas. Estos servidores permitirán la comunicación entre aplicaciones a través de internet y permitirán a terceros probar el protocolo o las librerías de referencia.

# Principios básicos de Vortex

Al ir probando con distintas ideas y distintos prototipos algunos principios oprevalecieron sobreo otros y toman fuerza como esenciales de Vortex. Estos deberían ser respetadas por toda implementación.

* Diseño distribuido
* Simplicidad y agnosticismo del código cliente
* Independencia del lenguaje de programación
* Basado en teoría de conjuntos
* Basado en el mejor esfuerzo posible
* Tiempos de entrega no asegurados
* Persistencia de los mensajes no asegurados
* Ruteo de mensajes con semántica variable pero compatible
* Adaptabilidad al entorno de ejecución
* Optimización de los recursos de la red
* Control de duplicados
* Notificación de interlocutores

### Diseño distribuido

Desde el momento cero, Vortex fue concebido como una red de funcionamiento distribuido pero con la posibilidad de contar con servidores centrales conocidos.

Es decir, que para su regulación la red no cuenta con ninguna autoridad, o entidad central. Cada parte de la red debe cumplir con algunas reglas básicas de manera de permitir la comunicación del resto y con el resto.

En su diseño actual no existen mecanismos para regular el comportamiento de las partes. Se asume una “buena fe” necesaria para hacer posible la red.

### Simplicidad y agnosticismo del código cliente

Al diseñar Vortex para ser utilizado como librería externa buscamos que el código cliente necesite saber lo menos posible de lo que está pasando tras bambalinas.  
En su versión más reducida debería verse a vortex como un bus de mensajes. Un elemento al cual se le entrega un objeto a modo de mensaje, y que permite obtener otros objetos esperados a modo de respuesta.

No debería ser necesario saber cómo funciona vortex para usarlo. Al menos no en una aplicación simple. Si es necesario controlar alguno de los factores que modifican el comportamiento default de Vortex entonces es necesario que la API permita formas de modificarlo sin necesidad de conocer todos los detalles a la vez.

Este es claramente uno de los puntos más díficiles dado que nosotros vamos aprendiendo y diseñando a la vez. Probablemente en las primeras versiones se escapen más detalles de los necesarios, pero al menos ese primer caso de uso debería ser muy simple.

Con el tiempo iremos mejorando la API para lograr un buen balance entre simplicidad y control.

### Independencia del lenguaje de programación

Otro de los puntos clave de Vortex es que no debe casarse con ningún lenguaje de programación. Tampoco debe asumir condiciones de ejecución especiales ya que su diseño está pensado para ser ejecutado en micro-controladores como Arduino.

Debido a que no es posible satisfacer esta condición al 100% de los lenguajes existentes, y que tampoco es necesario, intentamos adoptar metáforas y decisiones en el protocolo que sea implementables en los lenguajes y condiciones que conocemos y que podemos probar.  
Es posible, que se nos pasen algunos elementos que dificulten o quizás imposibiliten la implementación en algunos entornos o lenguajes. Tales casos no son voluntarios, y con la necesidad de extender la red a tales entornos es posible que modifiquemos las definiciones originales.

### Basado en teoría de conjuntos

Para posibilitar la intercomunicación entre lenguajes de programación, con Vortex proponemos un lenguaje común que permita hablar de conjuntos y elementos.  
Como parte del protocolo definimos un lenguaje que deben utilizar las partes comunicantes para decirle a Vortex la información necesaria para entregar los mensajes a todos los receptores correctos y a nadie más.

Este lenguaje intenta ser una abstracción común basada en la teoría de conjuntos lo suficientemente simple e intuitiva como para ser utilizada fácilmente desde código cliente.  
Los clientes que emitan mensajes a la red, deben describir como conjunto el tipo de mensajes que envían, y como elementos describir cada conjunto que envíen. Los receptores de la red deben describir como conjunto sólo el tipo de mensajes que desean recibir.

Para que el ruteo de los mensajes sea correcto el lenguaje debe permitir dos tipos de conceptos: conjuntos y elementos; y dos tipos de relaciones: pertenencia e intersección.  
Con la pertenencia es posible evaluar si un mensaje emitido como elemento, pertenece al conjunto definido por un receptor, y por lo tanto debe ser entregado. La intersección, permite saber si el conjunto de los mensajes emitibles por un emisor es interesante (al menos en parte) para un receptor, y por lo tanto ambos pueden comunicarse entre sí.

Para representar los conjuntos en este lenguaje utilizaremos restricciones/filtros que son combinables con expresiones booleanas. Estas restricciones serán evaluables en la red durante el ruteo de manera de que cada nodo pueda determinar hacia donde deben viajar los mensajes.  
O dicho de otra manera, quienes son los interesados en los mensajes que reciben.

### Basado en el mejor esfuerzo posible

Debido a su ambicioso objetivo de intercomunicar distintos componentes en distintas escalas, en distintos entornos de ejecución y bajo distintas configuraciones de redes físicas, no es posible dar por sentada la topología de la red en ningún momento.

Es por tal motivo que Vortex (al menos hasta el momento), no puede garantizar la entrega de un mensaje a los receptores necesarios. Sin embargo es un principio fundamental que Vortex va a realizar su mejor esfuerzo por entregar el mensaje a destino.

Esto significa que si el emisor está conectado a la red físicamente, y los receptores también durante el tiempo de ruteo del mensaje, Vortex garantiza la entrega. Cualquier desconexión intermedia o posterior a la emisión del mensaje aumenta el riesgo de la no-entrega del mensaje, pero dependiendo de las condiciones es posible que aún así, sea entregado (aunque no se garantiza).

Como consecuencia de este principio los clientes saben que mientras exista conexión física, los mensajes no se pierden, y todos los interesados los reciben. Pero las partes comunicantes no deben asumir que su interlocutor está siempre disponible en la red, y deben tomar las medidas necesarias para retransmitir o asegurar la entrega si así lo requiriera su semántica.

### Tiempos de entrega no asegurados

De la misma manera que no se puede asegurar la topología de la red, tampoco pueden asegurarse plazos de entrega de los mensajes, ni tiempos de las respuestas. Si Vortex tiene éxito como medio de comunicación base entre lenguajes es posible que su uso sea creciente y por lo tanto se agreguen más participantes a las comunicaciones existentes (de hecho es algo que alentamos y buscamos).

Por tal motivo los comunicantes no deben realizar asunciones de los tiempos de la red. Lo ideal sería que no dependan de los tiempos, y que si lo hacen que tengan mecanismo de adaptación (ya que la red muta con el tiempo) para no quedar incomunicados incorrectamente.

Vortex en su diseño actual no incluye herramientas ni mecanismos para esta adaptación y queda delegado al código cliente si así lo requiriera su semántica.

### Persistencia de los mensajes no asegurados

Los mensajes enviados a la red vivirán en la misma mientras se realiza su ruteo. No existen mecanismos como cola de mensajes o persistencia de los mismos como partes de Vortex.

Lo que no impide que existan comunicantes especializados que brinden estos servicios para permitir la entrega de mensajes ante eventuales desconexiones.

Este tipo de funcionalidad (como otra) es posible agregarla a la red agregando nodos que, utilizando el mismo paradigma de comunicación, aumenten la semántica disponible por clientes.

### Ruteo de mensajes con semántica variable pero compatible

Uno de los nuevos principios de vortex es la posibilidad de contar con distintos niveles de implementación del protocolo, facilitando la participación de clientes más simples.

Basándonos en filtros como un conjunto de restricciones que permiten definir un conjunto, los participantes del ruteo en Vortex pueden optar por no implementar todas las restricciones posibles, dejando pasar los mensajes indecisos. De esta forma en la comunicación pueden participar nodos con una implementación simple, pero que agregan cierto nivel de ruteo, y pueden agregarse nodos que puedan filtrar con mayor precisión que el resto.

Esto permite la convivencia de distintas implementaciones sin la pérdida de los mensajes. Lo único que debe cumplirse en toda la red, es que la semántica de las restricciones no sea contradictoria entre distintas implementaciones. Existirá un conjunto base de restricciones que serán definidas como parte del protocolo, y cada implementación podrá implementar un subconjunto sin modificar su definición. También podrán extender estos conjuntos pero haciendo el mejor esfuerzo de consensuar su implementación con el resto de la red.

### Adaptabilidad al entorno de ejecución

Así como los conjuntos también es posible que distintos nodos de la red tengan soporte para distintos medios físicos. Cada medio tiene sus ventajas y desventajas, y sus formatos específicos para la transmisión de los datos.

Vortex no impone restricciones respecto del medio físico pero si requiere la posibilidad de transmitir la semántica original del mensaje más los datos de descripción del mensaje que son evaluables con los conjuntos de filtros.   
Cada medio puede implementar la transmisión de datos como sea más conveniente, pero debe ser capaz de reconstruir el mensaje original al momento de rutearlo en otro nodo.

Existirán nodos conectados a medios distintos que hagan de puente entre esos medios posibilitando el uso de vortex con distintos dispositivos.

### Optimización de los recursos de la red

En la implementación de la red será contemplado un algoritmo de ruteo suficientemente bueno como para que permita la optimización del camino elegido para los mensajes, pero sin poner en riesgo la entrega de los mensajes.

Actualmente tenemos un algoritmo candidato que debe ser probado en distintas condiciones. Probablemente busquemos varias opciones antes de elegir una. Para poder evaluar el desempeño del algoritmo seguramente implementemos medidas de cuán óptimo es, de manera que esas medidas estarán disponibles como parte de la implementación modelo, pero no serán parte del protocolo.

### Control de duplicados

Quizás la optimización más importante sea la búsqueda de algoritmos que no generen mensajes duplicados por hiperconectividad. En el peor de los casos, el algoritmo que mayor redudancia tiene hará circular los mensajes en caminos alternativos hacia los destinos.

En tal escenario cada nodo tendrá la capacidad de eliminar los mensajes duplicados de manera que el código cliente no tenga que ocuparse de solucionar ese problema.  
Si encontramos algoritmos mejores es posible que no todos los nodos discriminen duplicados. En cualquier caso, es responsabilidad de Vortex como conjunto evitar que lleguen mensajes duplicados a un mismo destino.

### Notificación de interlocutores

Otro principio nuevo es la obligación de Vortex de notificar a emisores y receptores “compatibles” de su contraparte. Si bien la notificación no garantiza permanencia (ya que un comunicante puede conectarse y desconectarse antes que la notificación primera llegue a destino), si se garantiza que a un nodo conectado se le notificará de los eventos ocurridos en la red que son de relevancia según el conjunto de filtros que especificó.

Estas notificaciones permiten que las partes comunicantes sean pasivos, y se vuelvan activos en la comunicación sólo cuando existe la posibilidad de una respuesta.

Por supuesto, para que las notificaciones sean efectivas, las partes comunicantes deben hacer su mejor esfuerzo por definir qué reciben y qué emiten como filtros.

# Problemas no resueltos de Vortex

Tal como está planteado Vortex tiene algunos problemas sin resolver. No parecen problemas irresolubles pero aún no hemos encontrado buenas alternativas. Cada problema es lo suficientemente complejo como para tratarlo por separado, y en otro doc se definen tests para verificar que la implementación en cuestión los resuelve suficientemente bien.

* **Mensajes duplicados**  
  La red debe arbitrar los medios para evitar que un mismo mensaje llegue más de una vez a destino sólo porque la red tiene loops. La solución hasta el momento era asignar un ID por mensaje por emisor, quizás esto pueda cambiar
* **Elección de mejor camino**  
  De la mano con evitar mensajes duplicados elegir el mejor camino optimiza los recursos de la red y evita que se dupliquen mensajes innecesariamente, pero también asegura que el mensaje llega aunque se produzcan caídas de nodos.  
  Tal vez convenga evaluar distintos algoritmos para grafos
* **Lenguaje común para conjuntos y elementos**Aún no tenemos bien definido como sería el lenguaje que permita representar los conjuntos y sus elementos. Probablemente sea algo parecido a Json pero necesitamos resolver cómo se calcula la intersección y la pertenencia
* **Definición de mejor protocolo para cada medio**Hasta ahora teníamos una implementación en HTTP de difícil uso, y que probablemente se usara en los menos de los casos (cuando no hay conexión directa).  
  Falta definir el protocolo físico para utilizar en cada medio. Probablemente lo hagamos una vez que en memoria funcione como queremos. Habría que hacer una versión HTTP y socket binario por lo menos.
* **Definición de tests de aceptación de una implementación**Sería deseable que exista alguna forma de descripción de qué se espera de una implementación de vortex. Primero como elemento de evauación para unificar las implementaciones que estamos haciendo y definir un set mínimo de comportamiento esperado. Pero también como referencia para cualquiera que quiera implementar su versión y asegurar que es compatible con las implementaciones existentes