PHASE 0 NETWORK

Exequiel Fuentes L.[1] Álex Segovia M. [1] Eric Jeltsch F. [2(*)]

RESUMEN

Hoy en día, los astrónomos pueden usar los telescopios e instrumentos de muchos observatorios distribuidos por el mundo para ejecutar sus observaciones. El mecanismo para determinar los recursos que están disponibles consume tiempo y desgraciadamente, no siempre coinciden las necesidades de los astrónomos con la información que está a disposición. En este contexto se ha determinado un proceso preliminar que hemos llamado Phase 0, el cual es automatizado a través de una herramienta de software, llamada Phase 0 Network (P0N), la que ha sido implementada sobre tecnología peer-to-peer (P2P).

Palabras claves: Java, peer-to-peer (P2P), JXTA, sistemas distribuidos.

1. INTRODUCCIÓN

Existen requisitos específicos para el proceso de propuesta en todos los observatorios para la obtención de tiempo de observación, este proceso se lleva a cabo en dos fases, conocidas en [1] como Phase I y Phase II. Los astrónomos escriben propuestas para postular a tiempo de observación en telescopios durante Phase I. Si una propuesta es aceptada, el tiempo asignado para realizar la observación debe ser planificado detalladamente en Phase II. Cada observatorio proporciona herramientas de software para cada fase. Sin embargo, existe un paso en este proceso no reconocido que consiste en descubrir los recursos distribuidos alrededor del mundo que pudieran satisfacer las necesidades de un usuario para un experimento determinado. El proceso para determinar qué telescopios e instrumentos satisfacen esta necesidad consume tiempo y no siempre carente de errores, pues la consulta debe hacerse manualmente inspeccionando manuales y sitios web. De manera que escoger la (o las) opción(es) correcta(s) es un desafío, en donde P0N es el primer paso en el proceso de propuesta.

Pueden existir varios observatorios con instrumentos que podrían permitir ejecutar una observación en particular, algunos mejores que otros, sin embargo, es dificultoso para un astrónomo conocer todos los recursos disponibles en el mundo, y es aún más dificultoso decidir si ese recurso puede permitir ejecutar la observación.

Las herramientas de software en P0N tienen como misión asistir al astrónomo en la tarea de investigar si una observación puede ser ejecutada en un observatorio, desde [2]. P0N, es un sistema de software de carácter distribuido con una infraestructura peer-to-peer (P2P) la que es usada para proporcionar el servicio, y más aún con una baja cooperación entre los observatorios.

Para tal efecto, se describen las necesidades del astrónomo en un documento XML y enviado al grupo Phase 0. Los *peers* que participan en el grupo Phase 0 evalúan la descripción como una consulta y retornan un resultado haciendo coincidir las necesidades con la configuración de los instrumentos del observatorio. La descripción de los recursos también es estructurado en un documento XML. El resultado obtenido puede entonces ser utilizado por el astrónomo para preparar su propuesta en Phase I.

Este documento presenta en la siguiente sección la tecnología P2P y las ventajas de su uso en la implementación de P0N, en la sección 3 se presentan la tecnología y los protocolos del proyecto JXTA, en donde P2P se sustenta. En la sección 4 se define la propuesta P0N y los modelos de respuesta a las consultas, las que se presentan en XML.

[1] Observatorio Gemini, Centro de Operaciones Sur, AURA, La Serena, Chile. efuentes@gemini.edu , asegovia@gemini.edu

[2] Universidad de La Serena, Departamento de Matemáticas, Av. Cisternas 1200 La Serena, Chile. ejeltsch@userena.cl
(*)financiado por DIULS (Dirección de Investigación de la Universidad de La Serena).

2. ARQUITECTURA PEER-TO-PEER

La tecnología P2P ha ido ganando terreno en los medios de comunicación y en los procesos distribuidos, vea [3]. P2P llamó mucho la atención cuando apareció *Napster* [4] que permitía compartir archivos musicales, aunque no era estrictamente una aplicación P2P. Con la misma idea de compartir archivos aparecieron otras aplicaciones como *Gnutella*[5], *eDonkey* [6], *eMule*[7], entre otras. Estas aplicaciones P2P trajeron algunos problemas legales, técnicos y económicos. Existen otras corporaciones que han enfocado la tecnología P2P a otras áreas de investigación, por ejemplo las ciencia, los proyectos *SETI@Home*[8], *Folding@Home*[9], *Genome@Home*[10] y *Grid projects*[11], los que han demostrado cómo la tecnología P2P es muy útil para resolver problemas complejos, costosos y de carácter distribuido.
Digamos que un *peer* es un participante en un sistema P2P, el que puede ser cualquier tipo de dispositivo, tal como se muestra en la figura 2.1. En una topología P2P pura, un *peer* puede comunicarse directamente con cualquier *peer* sin un administrador o control central. Al mismo tiempo, es posible hacer una variedad de combinaciones entre topologías de acuerdo a las necesidades que requiera la aplicación. Los peers son participantes activos en una aplicación, de manera que pueden proporcionar servicios a otros peers en el grupo. Una ventaja comparativa respecto de otros sistemas clásicos, es que los archivos o los procesos están completamente descentralizados. Una de las metas de la tecnología P2P es tomar ventaja de los recursos que existen en la red.

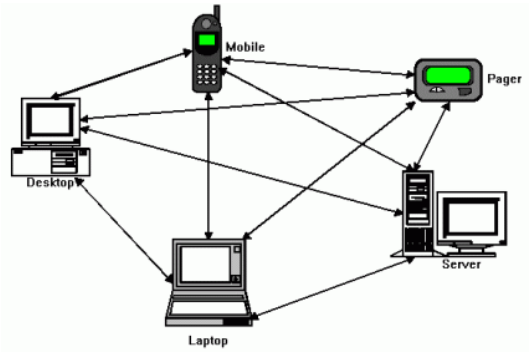

Figura 2.1 Un ejemplo de arquitectura P2P

2.1 P2P VERSUS OTRAS ARQUITECTURAS

La arquitectura *cliente-servidor* es una de las más populares para proporcionar servicios web, la que puede pensarse como un conjunto de servicios que son proporcionados a clientes que hacen uso de estos servicios. Los servidores y los clientes son tratados de manera diferente. En el contexto P0N digamos que cualquier observatorio interesado en participar podrá hacerlo sin solicitar autorización ni mantener un registro, en consecuencia no se desea un control central, por otra parte el elevado costo que significaría mantener un servidor con toda la información de todos los observatorios, es una desventaja importante. Por estas razones, la arquitectura P2P es ideal para la implementación de P0N comparada frente a la arquitectura *cliente-servidor*.
Otra arquitectura es la de *objetos distribuidos*, en este caso, no hay diferencias entre servidores y clientes, y el sistema se puede pensar como un conjunto de objetos interactuando cuya posición es irrelevante. No hay diferencia entre un proporcionador de servicios y un usuario de estos. Para permitir que los objetos interactúen se utiliza un software llamado *middleware* el cual conecta diferentes tipos de aplicaciones. Existen algunos *middleware*, como por ejemplo, *Socket*, *CORBA*[12], *RPC*, *Jini*[13], entre otros. La gran desventaja que presentan los productos *middleware* es su complejidad. P0N se puede desarrollar utilizando algunos de estos *middleware*, pero son de un bajo nivel lo que implica que para tener un prototipo funcional se deben escribir muchísimas librerías. Por último digamos que las aplicaciones P2P son ideales para grupos poco organizados, y no requiere un acuerdo entre los participantes. Los observatorios y astrónomos que estén interesados podrán participar como *peers* y descubrir a otros sin trabajar o sin cooperar activamente, si no desean. Quienes deseen proporcionar información pueden unirse sin previa aprobación o

registro. En resumen, la arquitectura P2P es ideal para compartir datos personales de astronomía o grandes archivos de datos de varios observatorios con el único requisito de pertenecer al grupo.

3. PROYECTO JXTA

P0N es una aplicación P2P construida sobre la tecnología del proyecto JXTA de Sun Microsystems [14], [15]. JXTA proporciona una plataforma P2P y un conjunto de recursos necesarios para la creación de cualquier tipo de aplicación P2P, además, JXTA es de fuente abierta, lo cual significa que los protocolos y el código fuente están disponibles sin costo y restricción de licencia.
JXTA no es una librería para los lenguajes Java o C, sino, un conjunto de protocolos que permite a los *peers* auto-organizarse y auto-configurarse para formar grupos independiente de su posición en la red, sin la necesidad de un administrador central. Los protocolos permiten que las aplicaciones sean independientes de la plataforma. Las aplicaciones pueden ser escritas en diferentes lenguajes y pueden ínter operar con otras aplicaciones JXTA sin importar su lenguaje de implementación y sistema operativo. Actualmente las aplicaciones JXTA pueden ser programadas en Java, Perl, Python, Ruby y C.

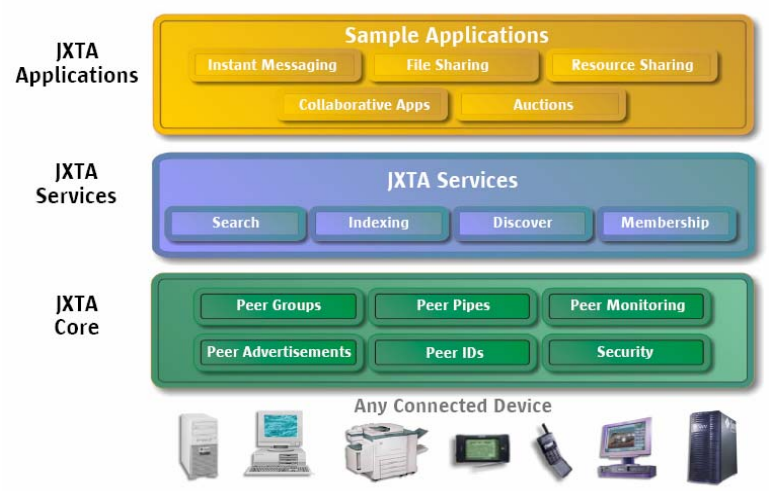

Figura 3.1 Arquitectura de software del proyecto JXTA

La arquitectura de software del proyecto JXTA está dividida en tres capas como lo muestra la Figura 3.1. La capa de más abajo tiene los mecanismos claves para las aplicaciones P2P, incluye *descubrimiento*, *transporte* (incluyendo manejo de firewalls), *creación de peers y grupos de peers* y *primitivas de seguridad*. La capa de servicios proporciona servicios de red que no pueden ser absolutamente necesarios para que una red P2P funcione, pero son comunes o deseables en un ambiente P2P. La capa de más arriba incluye la implementación de aplicaciones integradas, tales como *mensajería instantánea, compartir documentos y recursos*, y muchos otros. El límite entre servicios y aplicaciones no es rígido, todo depende del punto de vista. P0N es posible situarla en la capa de aplicaciones del proyecto JXTA.

3.1 CONCEPTOS DE JXTA

P0N utiliza conceptos propios de la tecnología P2P y otros particulares de JXTA. Aquí sólo se describirán los conceptos utilizados por P0N.
Un *peer* es cualquier dispositivo en la red que proporciona o usa un servicio, intercambia información, y generalmente interactúa con otros *peers*. En JXTA un *peer* no tiene una dirección IP específica, sino tiene una identificación única no importando donde el *peer* este localizado.
A una colección de *peers* que tienen un conjunto de intereses y servicios comunes, en JXTA se le denomina *peer group*. En JXTA un *peer group* puede formar una comunidad en que los *peers* puedan interactuar de manera segura.

En una red P2P puede haber otros tipos de peers como *rendezvous peer* y *relay peer*. Un *rendezvous peer* permite a otros *peers* descubrir los recursos y reenviar requerimientos a otros *rendezvous peers*. Un *relay peer* mantiene la información de otros *peers* y ayuda a rutear mensajes a otros *peers*, permitiendo la comunicación a *peers* detrás de un firewall.
Los *servicios* son la motivación del porqué los dispositivos se reúnen en una red P2P. Los *servicios* son proporcionados por los *peers*, estos pueden incluir transferencia de archivos, proporcionar información de estado, realizar cálculos, o básicamente hacer cualquier cosa que un *peer* desee hacer en una red P2P capaz de soportarlo.
En JXTA todos los recursos, tales como *peers*, *peer groups* y *servicios* están representados por *advertisements*. Un *advertisement* es un lenguaje neutral de metadato estructurado que describe a un recurso en la red, este es representado como un documento XML. Por ejemplo, un *peer* debe publicar un *advertisement* para informar a otros *peers* su presencia en la red.

3.2 PROTOCOLOS DE JXTA

Para permitir la comunicación se necesitan *protocolos*. En JXTA, un *protocolo* es uno o más mensajes en un formato específico. Los peers usan estos *protocolos* para descubrir a otros peers, anunciar y descubrir recursos en la red, y comunicar y rutear mensajes. P0N utiliza algunos de los *protocolos* de JXTA, los que a continuación se describen:

- **Peer Discovery Protocol (PDP).** Este protocolo proporciona a mecanismo básico para que los *peers* descubran todos los recursos JXTA publicados. Cada recurso es descrito y publicado usando un *advertisement*.
- **Endpoint Routing Protocol (ERP).** Es el mecanismo por la cual un *peer* puede descubrir una ruta para enviar un mensaje a otro peer. Si la topología de red cambia y hace que una ruta previamente utilizada no este disponible, los *peers* pueden utilizar *ERP* para encontrar una ruta alternativa.
- **Rendezvous Protocol (RVP).** Es el mecanismo en la cual los peers que implementan este protocolo manejen los detalles de propagación de mensajes entre peers.

4. PHASE 0 NETWORK

Considerando la sección anterior, P0N se puede definir como sigue: “P0N es un conjunto de *peers* unidos a un mismo grupo llamado *P0 Group*, donde los *peers* podrán realizar consultas y responder a estas”. Los *peers* se han clasificado en *astronomer peer*, *facility peer*, *rendezvous peer* y *relay peer*. Un *astronomer peer* es un *peer* simple diseñado para ser un usuario final, sólo accederá a los servicios proporcionados por otros *peers* que pertenezcan a *P0 Group*. Un *facility peer* es un *peer* simple diseñado para ser un servidor, sólo proporcionará servicios de respuesta y seguramente se encontrará detrás de un firewall. Las consultas y las respuestas estarán estructuradas en documentos XML. Estos *peers* sólo se limitarán a realizar esas funciones específicas, ellos no son responsables de manejar la comunicación. Un *rendezvous peer* es un *super peer* –un *peer* igual que otro, pero con otras funciones– que será el encargado de proporcionar información y descubrir recursos en *P0 Group*. Un *relay peer* proporciona los mecanismos para que *peers* se comuniquen con otros *peers* separados de la red a causa de un firewall y NAT.

4.1 P0N BASADO EN JXTA

JXTA proporciona un marco de trabajo que permite centrarse en el contenido de los mensajes y en la implementación de los *peers*. Se escogió Java como lenguaje de programación, aunque no es un requerimiento, pues otros lenguajes pueden utilizarse con las librerías JXTA.
JXTA proporciona el concepto de *peer* y la idea de una red virtual. Cada *peer* tiene una única identificación (*peer ID*) en la forma de un *peer advertisement* que es publicado en la red JXTA. Los *peers* en JXTA almacenan los *advertisements*. La abstracción de la red virtual y el *peer ID* permite a los *peers* moverse y cambiar sus direcciones IP.
JXTA soporta grupos de *peers* seguros. P0N utiliza un grupo llamado *P0 Group*, esto facilita descubrir los *peers* que realmente interesan, proporcionando la seguridad y separación del resto de la red JXTA. En este momento, unirse a

un *peer group* requiere sólo el nombre del usuario y un password, pero las comunicaciones podrían ser encriptadas para asegurar su privacidad. Se asume una presencia constante de los *facility* peers en *P0 Group*.
Un *peer* de JXTA en un grupo, puede proporcionar servicios a todos los miembros del grupo. Cada *facility peer* puede publicar y proporcionar *advertisements* por uno o más servicios. Los *peers* que proporcionan servicios crean los *advertisements* y los publican en la red JXTA. El *advertisement* del servicio proporcionado por un *facility peer* es propagado a través de *P0 Group* sin ser intervenido. Al publicar un *advertisement* de servicio, todos los *peers* en el grupo pueden encontrarlo y hacer uso de éste.
JXTA proporciona los mecanismos para que los *peers* puedan interactuar. Todos los protocolos definidos en JXTA están implementados como servicios, dando un alto grado de abstracción. Los *peers* utilizan *Discovery Service*, que usa *Peer Discovery Protocol* para descubrir y usar los *advertisements*. Este servicio no es responsable de enviar y recibir los mensajes. *Endpoint Routing Protocol*, define el protocolo para rutear los mensajes hacia otros peers, este protocolo es utilizado por *Endpoint Service* responsable de realizar la comunicación entre dos *peers*. Otro protocolo importante es *Rendezvous Protocol*, los *peers* lo utilizan para conectarse a los *rendezvous peers* con el fin de propagar los mensajes a otros *peers*, este protocolo esta implementado por *Rendezvous Service*, que no sólo es utilizado para que los peers propaguen mensajes, sino que también para que proporcionen servicios de rendezvous, los que pueden ser modelados a través de gramáticas formales de grafos, [16].
Los *peers* pueden estar detrás de firewalls o pueden tener direcciones IP basadas en ***N**etwork **A**ddress **T**ranslation*, que es un estándar en Internet que habilita a las redes locales LAN para usar un conjunto de direcciones IP para el tráfico interno y un segundo conjunto de direcciones para el tráfico externo, cuyo propósito es proporcionar un tipo de firewall escondiendo las direcciones IP internas. En este caso, se configura un *peer* llamado *relay peer* que proporciona los mecanismos para que los *peers* puedan comunicarse.
A continuación se muestra en figura 4.1, el modelo asociado a P0N basado en JXTA. En ambos modelos se muestra un *astronomer peer* realizando consultas a seis *facility peers*, todos unidos a *P0 Group*. Al lado izquierdo se muestra el modelo virtual, sólo existen *astronomer peer* y seis *facility peers* interactuando vía P2P. Al lado derecho se muestra el modelo físico, nótese que la comunicación entre *peers* involucra *rendezvous* y *relay peers*.

Modelo virtual **Modelo físico**

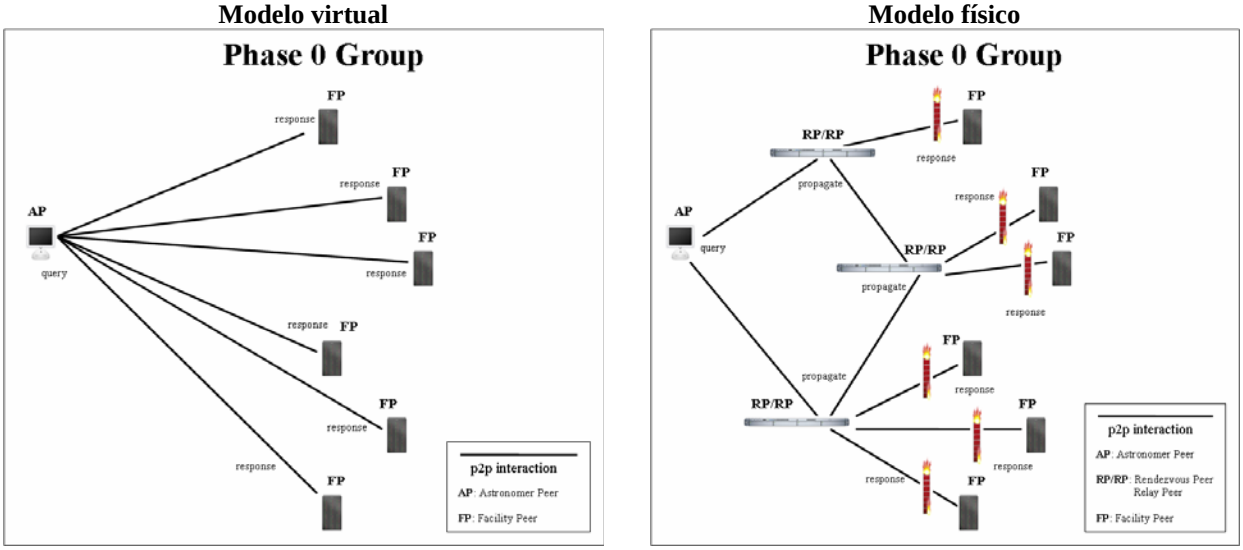

Figura 4.1 A la izquierda se muestra el modelo virtual de P0N y a la derecha el modelo físico.

4.2 LAS CONSULTAS Y RESPUESTAS EN P0N

Una suposición básica de P0N es que los objetivos de observación pueden ser descritos abstractamente, independientes de un instrumento específico o especificación de un telescopio. El proceso de hacer esta descripción no es fácil, pues requiere que los astrónomos conozcan bien su área y tengan la capacidad de describir sus metas en requerimientos técnicos. La mayoría de las propuestas de Phase I incluyen una sección de justificaciones técnicas que permiten a los astrónomos explicar en palabras, cómo un experimento puede ser hecho a través de una configuración específica del telescopio e instrumento. El éxito o el fracaso de una propuesta a menudo se concentran en esta sección. Desde el punto de vista de P0N, el nivel de detalle en la descripción debe ser adecuado para representar los aspectos de ciencia que son apropiados para hacer coincidir las metas de ciencia con las configuraciones del observatorio.
En esta sección se introduce una descripción de las tareas de ciencia basada en XML. La especificación dada aquí no está completa y sólo envuelve un subconjunto de tipos de observaciones y modos que los astrónomos utilizan. Se espera que publicando el prototipo existan personas que deseen unirse a este esfuerzo para refinar y expandir la descripción de ciencia y P0N.
Un *astronomer peer* que requiera información deberá realizar una consulta, que será enviada en forma de un documento XML. Cada *facility peer* tendrá la oportunidad de formular una respuesta en forma de un documento XML y retornarlo al *astronomer peer*.

4.2.1 Diseño de consultas y respuestas

La consulta y la respuesta están construidas sobre el formato de un documento XML.
La consulta o *query*, como se denomina en este documento, está formada por el elemento raíz *phase0query*, dentro del cual podemos distinguir los elementos *daterange, targets* y finalmente *wavelengths*, que nos permiten definir el rango de tiempo en el cual queremos observar el o los blancos deseados y el o los wavelengths que lo hacen posible. Esta información implica definir otros elementos, como son: *startdate* y *enddate* para definir el periodo de tiempo de observación, además para definir un blanco en el cielo, necesitamos especificar coordenadas mediante los elementos *ra* y *dec*, que definen los ángulos de ascensión y declinación del blanco respectivamente, y finalmente podemos definir el o los wavelengths deseados.
La respuesta o *response* como se denomina en este documento, está formada por el elemento raíz *phase0response*, lo que entregará la información disponible en los siguientes elementos:

- *facility*, que a través de sus atributos nos entrega la ubicación y nombre del observatorio consultado.
- *succes_percent*, indica una estimación de éxito para la observación propuesta en el observatorio consultado, basada en los porcentajes de wavelengths encontrados y blancos visibles.
- *target_matched_percent*, entrega una estimación de los blancos visibles desde la ubicación del observatorio consultado, basada en la calidad de visibilidad de estos.
- *no_targets_observable*, *targets_observable_at_3airmass*, *targets_observable_at_2airmass*, *targets_observable_at_1_5airmass*, entregan una cuenta de los blancos con una pobre visibilidad hasta los blancos mayormente visibles dependiendo de la masa de aire existente.
- *wavelengths_matched_percent*, entrega un porcentaje de instrumentos con los wavelengths deseados disponibles en el observatorio consultado.
- *instruments* o *instrument*, dependiendo de si el observatorio consultado entrega uno o varios instrumentos, nos entrega el nombre y URL con información útil disponible del o los instrumentos apropiados para la observación.
- *wavelength*, indica el ancho de onda del o los instrumentos retornados.
- *date_range*, el rango de fecha para la observación.

4.2.2 Ejemplos de consultas y respuestas

A continuación se muestra un ejemplo de una posible consulta y respuesta. El ejemplo está basado en los instrumentos disponibles en el Observatorio Gemini Sur. La información de los instrumentos de Gemini Sur está disponible en el sitio Web de Gemini, [17]. Una simple consulta y respuesta se muestra en la figura 4.2 (la información entregada en la respuesta es sólo un ejemplo). Las líneas 1 y 2 de A y B muestran la declaración requerida por XML y el tipo de documento del que se trata.

A. Consulta	B. Respuesta
1. <?xml version="1.0"?> 2. <!DOCTYPE phase0query> 3. <phase0query> 4. <daterange> 5. <startdate>**041014**</startdate> 6. <enddate>**051014**</enddate> 7. </daterange> 8. <targets> 9. <target> 10. <ra>**109.30999999999999**</ra> 11. <dec>**-7.126388888888888**</dec> 12. </target> 13. </targets> 14. <wavelengths> 15. <optical /> 16. </wavelengths> 17. </phase0query>	1. <?xml version="1.0" ?> 2. <!DOCTYPE phase0response> 3. <phase0response> 4. <facility site="**Cerro Tololo**" name="**GEMINI-SOUTH**"> 5. <success_percent value="**100.0**" /> 6. <targets_matched_percent value="**100.0**" /> **7.** < targets_observable_at_3airmass value="**0**" /> 8. <targets_observable_at_2airmass value="**0**" /> 9. <targets_observable_at_1_5airmass value="**1**" /> 10. <no_targets_observable value="**0**" /> 11. <wavelengths_matched_percent value="**100.0**" /> 12. <instruments> 13. <instrument url="**http://www.gemini.edu/sciops/instruments/nirs/nirsIndex.html**" name="**Gnirs**"> 14. <wavelength>**optical**</wavelength> 15. </instrument> 16. </instruments> 17. <date_range final_date="**051014**" init_date="**041014**" /> 18. </facility> 19. </phase0response>

Figura 4.2 Al lado izquierdo un ejemplo de consulta y al derecho una respuesta a esa consulta.

Los autores han implementado librerías escritas en Java para la creación del grupo con sus respectivos servicios, y los *peers*, además se han realizado las pruebas con algunos astrónomos para testear el funcionamiento de la infraestructura desarrollada.

4.3 DISEÑO DE P0N

Esta sección resume la manera cómo se modeló Phase 0 Network. La figura 4.3 muestra los pasos que los *peers* ejecutan para tomar un rol, ya sea, *rendezvous peer*, *facility peer* o *astronomer peer*. Primero cada *peer* debe unirse al grupo, el *peer* tomará los servicios proporcionados por este. Luego tomará un único rol, que especificará que servicios prestará. Luego deberá conectarse a un *rendezvous peer* para que sus mensajes sean propagados a través del grupo.

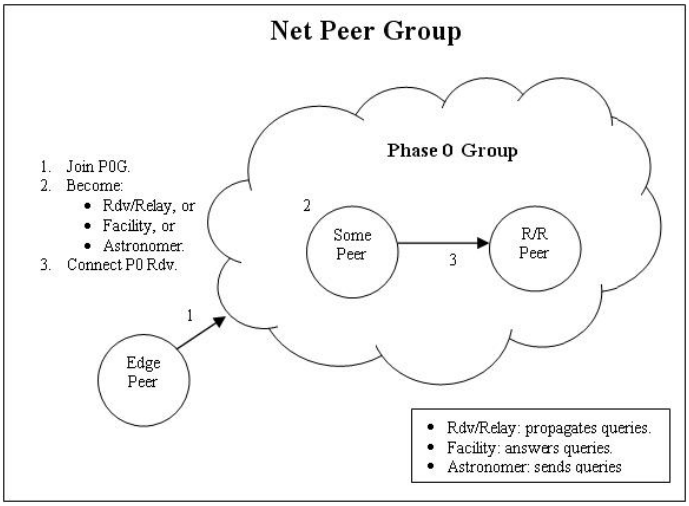

Figura 4.3 Los peers en Phase 0

Otra parte importante del modelo de Phase 0 Network es la consulta y la respuesta, se presentan dos diagramas de clases que describen gráficamente el modelo de dato.

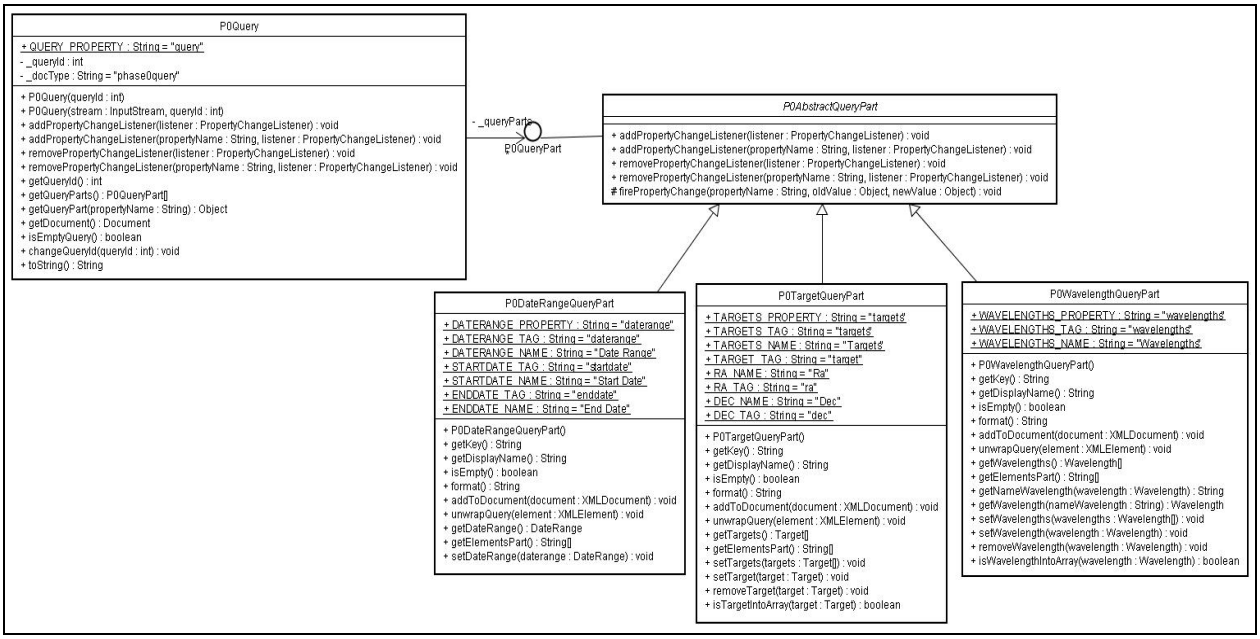

Figura 4.4 El diagrama de clase para la consulta

Dada la complejidad de la consulta se resolvió dividir esta en diferentes partes. Por otro lado, el modelo también debe proporcionar información acerca de los cambios que se hagan sobre la consulta. La clase P0Query es instanciada cada vez que se cree una nueva consulta, la clase P0Query utilizada la interfaz P0QueryPart que representa cada una de las partes de la consulta. La clase abstracta P0AbstractQueryPart implementa algunos de los métodos de la interface P0QueryPart que permiten determinar cuando ocurre un cambio sobre la instancia. Cada parte es implementada por tres clases P0DateRangeQueryPart, P0TargetQueryPart y WavelengthQueryPart.

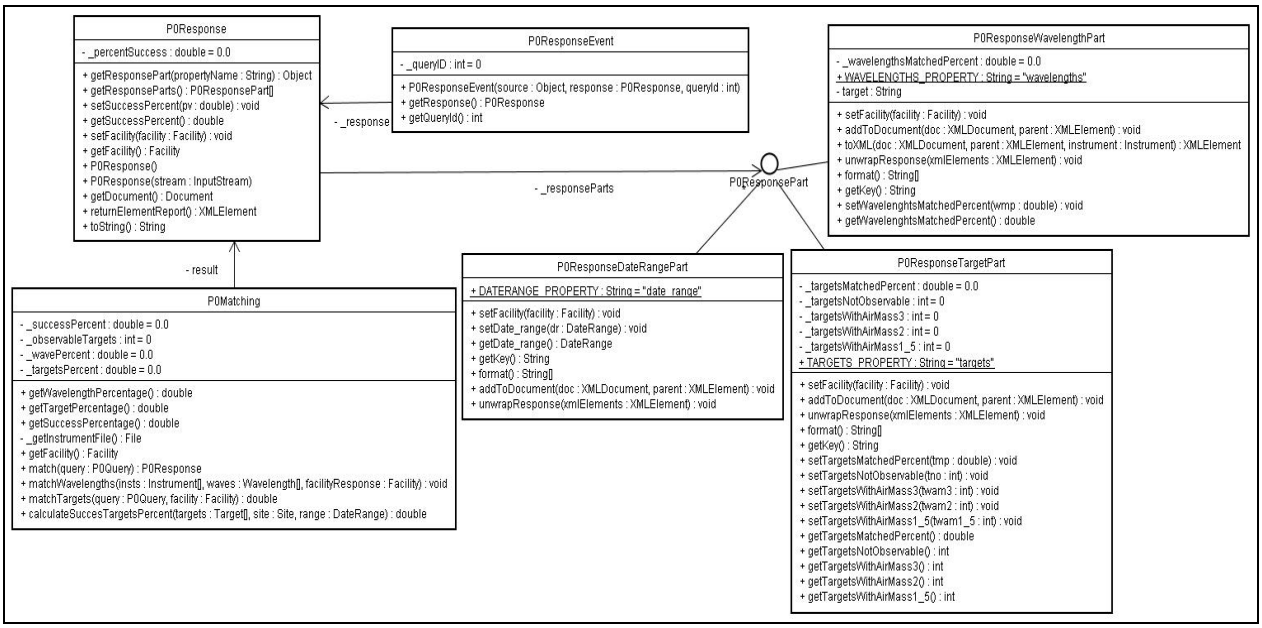

Figura 4.5 El diagrama de clase para la respuesta

El modelamiento de la respuesta, ver figura 4.5, debe permitir manejar cada parte de la consulta y permitir desarrollar una respuesta a la consulta, nótese que este proceso es por parte del facility peer. Por otro lado, el astronomer peer debe enterarse que ha recibido una respuesta y poder hacer uso de esta. En el proceso de generar una respuesta a una consulta se crea una instancia de la clase P0Response, esta crea una instancia de la clase P0Matching para hacer coincidir la consulta con los recursos del observatorio, y utiliza la interfaz P0ResponsePart que representa a cada parte de la respuesta. Cada parte de la respuesta es implementa por las clases P0ResponseDateRangePart, P0ResponseTargetPart y P0ResponseWavelengthPart. En el proceso de recibir una respuesta se crea una instancia de la clase P0Response y se genera un evento para el cual se crea una instancia de la clase P0ResponseEvent. Cada parte de la respuesta es inicializada de acuerdo el mensaje recibido.

5. CONCLUSIONES

Phase 0 es una de las partes que ya cuenta con un adecuado apoyo tecnológico de herramientas de software. En este contexto P0N es una propuesta para que la calidad y la eficiencia del proceso de observación sean realizadas. En consecuencia, más astrónomos y científicos se verán apoyados por las herramientas de Phase 0, puesto que la información recabada servirá para mejorar las propuestas en la Phase I. Al mismo tiempo, P0N entregará una manera de mostrar las capacidades de un observatorio a lo ancho de la audiencia internacional. P0N además, proporciona la oportunidad de integrar todos los soportes que ponen a disposición los observatorios, tales como sitios web y manuales. Los *peers* permitirán que los observatorios puedan localizar y compartir todo el conocimiento de sus instrumentos para el beneficio de todos.

AGRADECIMIENTOS

Este proyecto ha sido desarrollado durante la permanencia de dos de los autores [1], en Observatorio Gemini AURA Inc. Los autores desean agradecer a los Sres. Shane Walker, Software Developer para Gemini Sur y Kim Gillies (creador del concepto de P0N), Chief Programmer para Gemini, por la constante cooperación prestada en este proyecto. Finalmente, agradecemos al Observatorio Gemini AURA Inc. por la utilización de la formidable infraestructura que puso a disposición para la ejecución de las diversas etapas de este proyecto.

REFERENCIAS

[1] Observatorio Gemini; Phase I y Phase II; http://www.gemini.edu/sciops/ObsProcess/ObsProcIndex.html.
[2] Kim Gillies, Shane Walker, "Design for a phase 0 network", SPIE Vol. 4844, p. 232-241, Diciembre 2002; http://www.gemini.edu/documentation/preprints/pre86.html.
[3] Peer-to-peer; http://openp2p.com/.
[4] Napster; http://www.napster.com/.
[5] Gnutella; http://www.gnutella.com/.
[6] Edonkey; http://www.edonkey2000.com/.
[7] Emule; http://www.emule.com/.
[8] SETI@Home, Búsqueda de Inteligencia Extraterrestre, http://setiathome.ssl.berkeley.edu/download.html.
[9] Folding@Home, Computación distribuida; http://www.stanford.edu/group/pandegroup/folding/.
[10] Genome@Home, The Human Genome Project; http://www.stanford.edu/group/pandegroup/genome/.
[11] Grid computing; http://www.grid.org/home.htm.
[12] CORBA; http://www.omg.org.
[13] Jini; http://wwws.sun.com/software/jini/.
[14] Proyecto JXTA, http://www.jxta.org.
[15] JXTA, Brendon J. Wilson, Editorial New Riders; http://www.brendonwilson.com/projects/jxta/.
[16] Gnanamalar David, Frank Drewes, Hans-Jörg Kreowski: "Hyperedge Replacement with Rendevous." TAPSOFT 1993: 167-181
[17] Observatorio Gemini, Lista de instrumentos, http://www.gemini.edu/sciops/instruments/instrumentIndex.html