Metronet: software para medición de calidad de servicio en voz y video.

Pablo Belzarena, Víctor González Barbone, Federico Larroca, Pedro Casas.

Facultad de Ingeniería, Universidad de la República, Montevideo, Uruguay.

Abstract— **El proyecto Metronet se propuso como objetivos desarrollar una metodología y un software para medir los diferentes parámetros de calidad de servicio (QoS) en aplicaciones de voz y video. El proyecto se realizó avanzando en forma casi paralela en la investigación teórica con simuladores, la construcción de prototipos para verificación en redes reales de los resultados teóricos y las fases primarias de ingeniería de software para el producto final. Luego de varias etapas de convergencia se alcanza la definición completa de la ingeniería de software del producto, pasándose a la implementación, reutilizando, corrigiendo o sustituyendo el código del último prototipo, más el agregado de código necesario para interfaz de usuario, autenticación, presentación de resultados, emisión de audio o video real para una calificación subjetiva y otros aspectos propios de interés del usuario. El desarrollo resulta más o menos lineal una vez definida la metodología de medición de estado de la red y predicción de calidad de servicio, resultado de la investigación y la verificación con prototipos. El presente trabajo describe brevemente el software de Metronet y su desarrollo, culminando en la presentación de algunos resultados experimentales de su uso en la medición y predicción de la calidad de servicio de un proveedor en un momento dado. Las conclusiones resumen la experiencia apuntando aciertos, carencias y posibles mejoras del producto y del proceso de desarrollo.**

I. Introducción y motivación.

Con el surgimiento de nuevos servicios sobre Internet, en particular voz y video, ha crecido la necesidad de medir la performance de la red. Estas medidas de performance permiten verificar la calidad de servicio requerida por las nuevas aplicaciones así como avanzar en el problema de entender el comportamiento de la red.

Los servicios multimedia en una red de conmutación de paquetes tienen problemas que afectan la calidad de servicio, como retardo, jitter, pérdidas, y en algunos casos degradación por una baja tasa de codificación.

Una red IP como Internet tiene dificultades para medir los parámetros de performance debido a que la ruta de los paquetes puede cambiar, la tasa de bits no es constante y generalmente el tráfico es en ráfagas, cuando se envían paquetes de prueba estos pueden ser filtrados o alterados por algún ISP (proveedor de servicios de Internet) del camino, no hay sincronización de relojes entre los routers y los equipos de los extremos, etc. Otro inconveniente es que generalmente los routers internos a lo largo de un camino no están bajo el control de un mismo usuario o ISP, y por lo tanto no son útiles los procedimientos de medición basados en la información obtenida de los mismos. Es por esto que las medidas de extremo a extremo han tenido gran desarrollo en los últimos años.

Este trabajo describe el desarrollo de una herramienta de software (Metronet) que permite realizar mediciones de la calidad de servicio de extremo a extremo en Internet. Esta herramienta, si bien es general para la realización y análisis de mediciones de extremo a extremo en una red, pone especial énfasis en la implementación de una metodología de medición desarrollada por el grupo de investigadores [1] que se describe brevemente en la sección siguiente.

II. Metodología de medición.

El objetivo propuesto es estimar la calidad de servicio para aplicaciones multimedia en Internet. Estas estimaciones permitirían, a modo de ejemplo:

- monitorear y verificar durante largos períodos de tiempo un Service Level Agreement (SLA) entre un usuario y un ISP o entre diferentes ISPs.
- predecir la calidad de servicio que un video o una conversación podrá alcanzar luego de recorrer un camino en Internet.
- monitorear los caminos entre un proveedor de video o TV sobre Internet o un ISP y sus usuarios, para tomar acciones como reducir la tasa de codificación o elegir un camino alternativo.

Algunas aplicaciones que realizan mediciones en Internet estiman parámetros tales como retardo o pérdidas para algún servicio enviando paquetes de

prueba con un tiempo entre envío de paquetes constante o exponencial. Sin embargo los parámetros de calidad de servicio dependen del comportamiento estadístico de cada servicio y por lo tanto, en muchos casos, este tipo de estimación es poco preciso.

Para el tipo de aplicaciones descritas anteriormente una forma de estimar la calidad de servicio consiste en enviar un video o conversaciones de voz durante suficiente tiempo y medir los parámetros deseados sobre esos paquetes. Buscamos una metodología para inferir la calidad de servicio de una aplicación multimedia sin tener que enviar el servicio multimedia todo el tiempo, lo cual no sería razonable. Una forma de resolver este problema sería tener un modelo para los caminos de Internet con una función o funciones conocidas que permitan calcular los parámetros de calidad de servicio conociendo:

- un modelo para el tráfico multimedia.
- un modelo para el cross traffic.
- las capacidades y tamaños de buffer de los enlaces a lo largo de un camino.

El principal problema de este método es que, mirando el camino desde los extremos, no se conoce el cross traffic, y las capacidades del enlace o tamaños de buffer. Incluso en el caso en el que podamos tener buenas estimaciones desde los extremos del cross traffic, las capacidades y los tamaños de buffer, no hay un modelo analítico para un camino de Internet que permita obtener esta función (o funciones) para calcular los parámetros de calidad de servicio.

La métrica de comportamiento Y (retardo, pérdidas, "jitter", etc.) para paquetes de tráfico multimedia es una función $Y = \Phi(X_t, V_t, C, B)$ donde X_t es el proceso estocástico de cross-traffic, V_t es el proceso estocástico de video o voz, C es la capacidad del enlace y B es el tamaño de buffer. La capacidad el enlace C y el ancho de banda B no son conocidos, pero se asumen constantes durante el tiempo de monitoreo. Se asume el proceso V_t como variable conocida en el problema de lo que resulta $Y = \Phi(X_t)$. El proceso de cross traffic X_t es dependiente y no estacionario.

En una primera etapa se estima la función Φ mediante el envío de una ráfaga de paquetes de prueba de largo fijo, enviando inmediatamente después un flujo de video breve.

La metodología desarrollada se basa en:

- estimación del cross traffic X_t usando un conjunto de paquetes de prueba livianos. Se envía una ráfaga de paquetes de prueba de largo fijo, e inmediatamente después un flujo de video breve. Estos envíos se repiten varias veces a intervalos t_{in}a partir del final del flujo de video previo. Midiendo el tiempo entre arribos t_{out} entre paquetes de prueba consecutivos se obtiene una serie fuertemente correlacionada con el proceso de cross-traffic presente en el enlace, lo que permite su estimación.
- usando la estimación del cross traffic X_t anterior y midiendo el parámetro de interés Y (pérdidas, retardo, jitter, etc.), se infiere o "aprende" la función Φ que relaciona el parámetro de interés Y con el cross traffic X_t.
- luego de culminada la etapa de aprendizaje, se envían sólo paquetes de prueba livianos para obtener una nueva estimación del cross traffic X_t; usando la función estimada $\hat{\Phi}$con este cross-traffic se estima el parámetro de calidad de servicio $\hat{Y}$ para una aplicación multimedia, sin haber enviado tráfico multimedia pesado en ningún momento.

La estimación de Φ se calcula en base a una generalización del estimador de Nadaraya-Watson. Existen resultados de convergencia para procesos estacionarios [2]. Para este trabajo se demostró una extensión de estos resultados para algunos modelos no estacionarios. El fundamento teórico completo de esta metodología puede verse en [1].

III. El proyecto Metronet.

El proyecto Metronet tuvo como objetivo encontrar una metodología y una herramienta informática para realizar medidas de extremo a extremo, de los diferentes parámetros de calidad de servicio para aplicaciones de voz y video, en forma independiente del proveedor del servicio, apto para uso con direcciones IP dinámicas, distorsionando en forma mínima el tráfico en la red a medir. En una primera etapa se desarrolló la metodología [1] ya referida en la introducción. Posteriormente se desarrolló el software que si bien plantea un framework general de mediciones de extremo a extremo, permite implementar la metodología referida. Este software además fue probado primero en una red piloto de máquinas Linux y con tráfico simulado, en una situación controlada donde los resultados predichos pueden ser fácilmente contrastados con las mediciones reales [4]. Por último se han realizado pruebas en Internet para evaluar su operatividad.

Este trabajo describe el producto de software obtenido, sus prestaciones, gestión de desarrollo y arquitectura.

IV. Especificación del software: restricciones y requerimientos.

En el momento de la especificación del software existían algunas restricciones, formuladas en los objetivos del proyecto, orientadas esencialmente en dos sentidos:

- facilitar al usuario el acceso al software y su operación;
- cargar mínimamente la máquina del usuario, atendiendo a la diversidad de hardware, software instalado y estado de procesos. Esta restricción tiende a la universalidad en el uso del software y a una mínima incidencia del estado de la máquina cliente sobre la medición.

Más específicamente, el software debía cumplir las siguientes restricciones (o requerimientos no funcionales):

- escrito en un lenguaje de programación libremente accesible para el usuario.
- carga mínima en la máquina del usuario, limitada a la devolución de los paquetes enviados por el ISP con mínimo trámite.
- descarga transparente del código cliente necesario, sin intervención específica del usuario.
- herramientas de la máquina del usuario limitadas a un navegador y eventualmente una máquina virtual para la ejecución del código cliente.

Los requerimientos funcionales del software eran los siguientes:

- facilidad de uso.
- interfaz clara, sobria, sin excesivo detalle técnico, tanto en la especificación de las pruebas como en la presentación de resultados.
- validación de acceso del usuario por parte del ISP al que se conecta, mediante nombre y contraseña.
- posibilidad del usuario de fijar localizaciones y ser reconocidas éstas luego, para poder efectuar mediciones desde diferentes lugares y utilizar datos de verificaciones anteriores como insumo para la evaluación estadística.
- posibilidad del usuario de elegir tipo de medio (audio o video).
- presentar valores por defecto para todos los parámetros, siempre que ello sea posible, permitiendo al usuario fijar los suyos propios si lo desea.
- posibilidad del usuario de elegir características del multimedio, tales como tipo de codec, tasa de bits, tipo de conexión, cantidad de movimiento (video). Se presentan sólo aquellos valores correspondientes al tipo de multimedio elegido para la verificación.
- posibilidad del usuario de elegir características del experimento, fijando cantidad de pruebas, tiempo entre pruebas, cantidad de paquetes de prueba, tiempo entre paquetes.
- opcionalmente, el usuario puede pedir la transmisión de una muestra del medio elegido para una evaluación subjetiva, de entre una lista ofrecida.

Si bien no forma parte de la metodología de medida, se estimó de gran interés para el usuario ofrecer la posibilidad de elegir una muestra de video y verla por sí mismo, habilitando una estimación subjetiva de la calidad. La calidad percibida (PQoS) puede diferir mucho de los resultados obtenidos por medición, según las características personales del usuario: capacidad auditiva y visual, interés en el material, etc [6]. La lista de videos ofrecida intenta abarcar los tipos más frecuentes en el medio elegido, posibilitando al usuario elegir aquel que más se aproxima al medio que realmente quiere ver y oír. Las muestras tienen una duración típica de 20 a 30 segundos.

Se decidió realizar la implementación del software en Java [8], por su independencia de plataforma y su ubicuidad. El código cliente se implementó como "applet" [9], y sólo devuelve los paquetes recibidos de la dirección del ISP a esa misma dirección, con un mínimo tratamiento de estampado de fecha y hora. El hecho de realizar estas mediciones en capa de aplicación se consideró pertinente, por simular más de cerca la situación real del usuario al recibir voz o video, lo cual se realiza siempre mediante una aplicación en capa 5 del modelo simplificado de OSI .

V. Uso del software.

El usuario dirige su navegador a la dirección URL del proveedor cuya calidad de transmisión quiere evaluar. La primera página recibida es una bienvenida al programa donde se le pide usuario y contraseña, o se le ofrece bajar el software necesario (applet de Java) para instalar en el cliente, si no lo ha hecho ya.

Una vez validado, el usuario recibe una página donde se fijan todos los parámetros del experimento. El usuario debe elegir obligatoriamente la localización, si se trata de audio o video; el resto de los parámetros tienen valores por defecto que el usuario puede simplemente aceptar. Si desea recibir una muestra real de audio o video, deberá elegirla entre la lista de muestras disponibles.

Fig. 1. Bienvenida y autenticación del usuario.

En el contexto de este software el término "experimento" se aplica al proceso de medición completo, el cual consta de varias "pruebas". Esto resulta aparente en el propio contenido y distribución de la página, por lo que el manejo debiera ser totalmente intuitivo.

En cuanto el usuario decide comenzar la prueba, en el servidor del ISP se buscan registros históricos para ese usuario y localización, si existen se evalúa su utilidad, diseñando el patrón de pruebas en consecuencia [1]. El servidor emite paquetes de sondeo o paquetes de video, el cliente los devuelve, se miden tiempos, se contabilizan pérdidas, elaborando todos estos datos y guardándolos en una base de datos.

Si el usuario pidió una muestra del medio, se le envía para su audición y visualización en el propio navegador. Se presenta finalmente una página de resultados con la evaluación realizada. El usuario puede repetir el experimento, en cuyo caso vuelve a la página de parámetros, donde encuentra los últimos fijados, o terminar la aplicación. Pasado un cierto tiempo, el servidor cierra la sesión, y si el usuario quiere repetir el experimento deberá reingresar al sistema con su nombre y contraseña.

VI. Resultados experimentales.

Para verificar el funcionamiento del software y los algoritmos implementados se realizó el siguiente ensayo:

- para verificar que el resultado vale cuando los procesos no son estacionarios parte del experimento se realizó con una capacidad de 200 Kbps y parte con 160 Kbps.
- el cross traffic generado fue video codificado a 50 Kbps.
- el video de test fue un video simulado con la misma codificación a 50 Kbps.

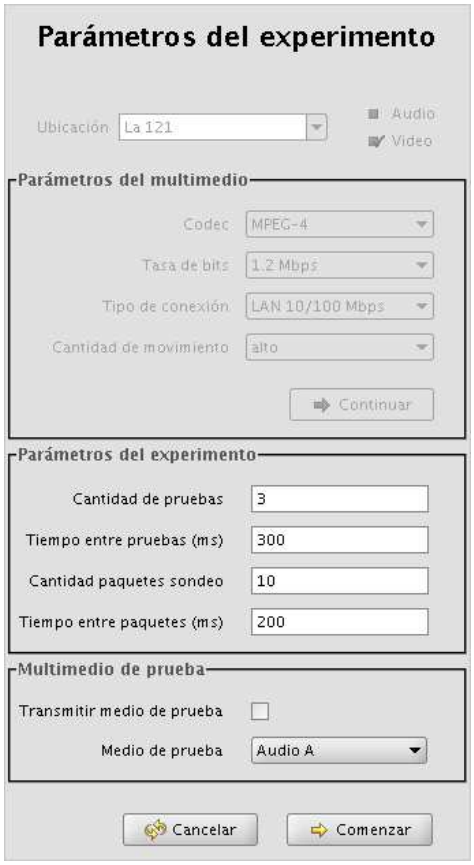

Fig. 2. Ingreso de parámetros del experimento

Se compararon los parámetros de performance predichos por los algoritmos y sus valores reales medidos. La figura 3 muestra estos resultados para el caso de RTT. Como se puede apreciar cuando varía la capacidad del enlace el RTT varía bruscamente. Como el algoritmo es "de aprendizaje" se puede ver que va mejorando su aproximación a medida que tiene más muestras, incluyendo una mejora en el seguimiento de las variaciones bruscas.

VII. Gestión del proyecto.

El desarrollo del proyecto se realizó siguiendo en forma más o menos paralela varias líneas de trabajo:

- investigación teórica, esencialmente creación de modelos de experimentación, simulación de su comportamiento y tratamiento estadístico de resultados, generación de información de base o histórica para inferencia estadística.
- construcción de prototipos, ensayos de diversos métodos de medición, tratamiento de los datos de medidas, comparación con los resultados teóricos.

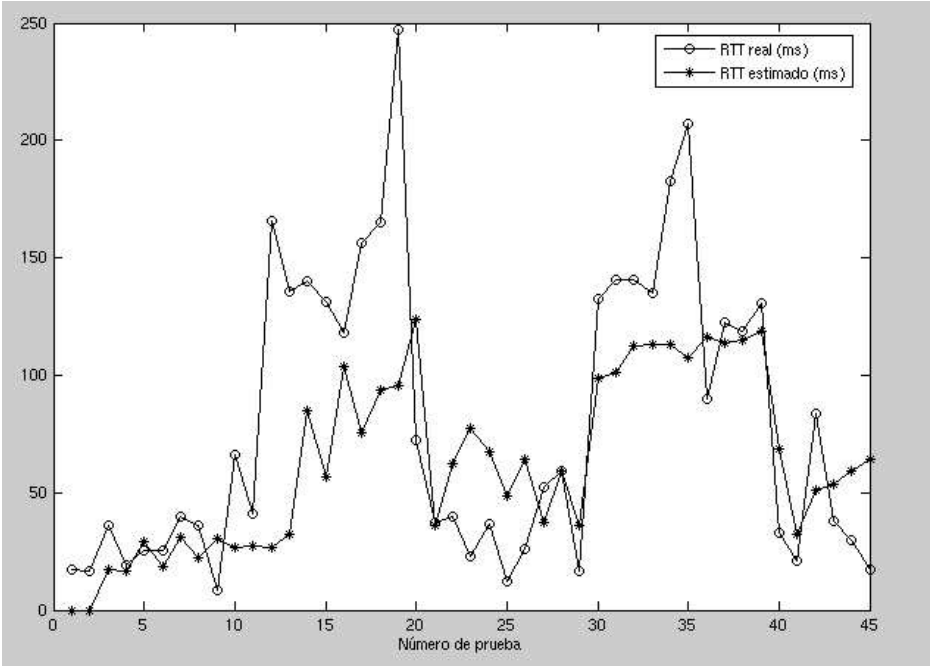

Fig. 3. Resultados del ensayo.

- desarrollo de la ingeniería de software para la obtención del modelo de diseño correspondiente al producto final, con modelado tardío de las herramientas de calidad de servicio según se fueran definiendo, dejando para último momento, como componente desacoplado, el predictor de calidad.

Se avanza hacia el producto final en diferentes instancias de convergencia de las líneas anteriores:

- cuando queda establecido el método de medición a usar, resultado de la adopción del modelo de investigación con mejor aproximación tanto en la simulación como en la verificación práctica en redes reales mediante prototipo. En este momento se completa el modelado del software correspondiente al envío de tráfico de prueba.
- cuando quedan finalmente definidos los algoritmos de predicción de la calidad de servicio esperada. Aquí se modela el componente de predictor de calidad, completándose el paquete de herramientas de calidad de servicio.

Una vez completa la ingeniería de software, se pasa a la implementación definiendo en código, pero sin contenido, todas las clases de diseño. Para el contenido, se reutiliza, reconstruye o sustituye el código usado en el prototipo para los componentes vinculados a la medición; se agrega código para los componentes relativos a la interfaz de usuario, autenticación, emisión de medio real y otros no necesarios para el prototipo.

La verificación del software queda ahora limitada solamente a las pruebas necesarias para asegurar la calidad del software en cuanto a la implementación de una metodología de medición y predicción de calidad de servicio ya definida y validada en etapas anteriores del desarrollo a través de los diversos prototipos construidos.

VIII. Modelo de Arquitectura.

Es un modelo cliente servidor clásico: quien quiere medir la calidad de servicio actúa como cliente, sólo devolviendo paquetes; el proveedor de servicio actúa como servidor, autenticando al usuario, emitiendo los paquetes, recibiendo las devoluciones, realizando los cálculos y devolviendo al cliente los resultados.

La figura 4 muestra el despliegue de paquetes en el cliente y el servidor. El nodo cliente es realmente liviano; la carga de procesamiento, memoria y repositorio de datos aparece soportada en el nodo servidor.

IX. Ingeniería del software.

Para la construcción del modelo del software como producto final se empleó un paradigma cercano clásico en cascada, con la estrategia ya anotada

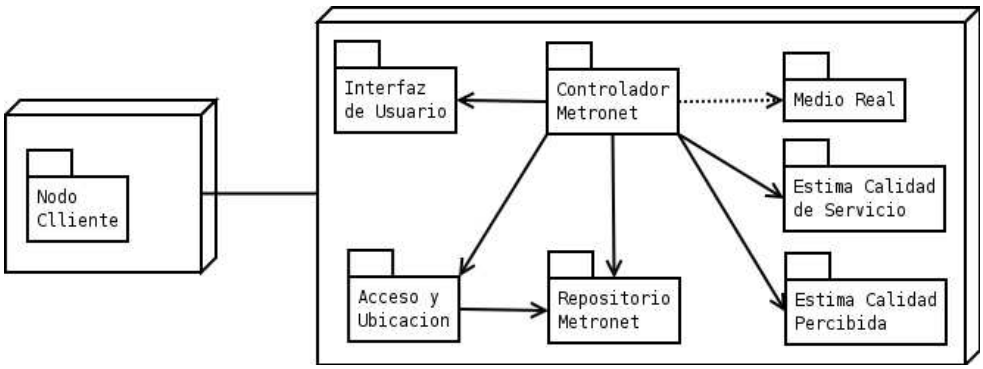

Fig. 4. Diagrama de Componentes en cliente y servidor.

del desarrollo paralelo coordinado de investigación, prototipado y modelado. Las herramientas UML usadas fueron también clásicas, limitadas a casos de uso, modelo de análisis y modelo de diseño [3]. Los resultados obtenidos de la investigación teórica y su verificación con el prototipado fueron los factores que hicieron posible este desarrollo lineal y relativamente simple; el producto a construir estaba definido casi completamente, salvo algunas partes muy localizadas y desacopladas. Seguidamente se revisan brevemente algunos aspectos propios de la concepción del software.

La figura 5 muestra los paquetes en el nodo cliente. El paquete Rebotador es el applet de Java cuya función esencial es devolver al proveedor los paquetes que éste le envía, ya sean de sondeo o de video simulado, visualizadas en el detalle de análisis en la misma figura.

El diagrama de análisis de la figura 6 muestra el software de Metronet en un único diagrama, destacando la arquitectura de tres capas, con las interfaces sobre la izquierda, el repositorio (DBMS) en la parte inferior, el control principal del software, Estimador Flujo MMedia, en el centro, y el resto de las funciones como paquetes controlados por Estimador Flujo MMedia. Esta separación resultó particularmente útil en el caso de los dos paquetes de estimación (Estima Calidad de Servicio y Estima Calidad Percibida), donde son más probables los cambios. Incidentalmente, esta cualidad permitió la definición algorítmica tardía de estos paquetes sin retardar desmesuradamente el tiempo de desarrollo.

El diagrama de la figura 7 muestra el desacoplamiento entre los métodos de medida y las herramientas de medición y cálculo. Es relativamente fácil agregar nuevas herramientas, así como nuevos métodos en base a herramientas existentes.

El método de medida Medio con Histórico se aplica cuando el usuario ya ha realizado experimentos desde esa misma ubicación, existen datos de medidas almacenados, y se estima que esos datos son relevantes para la situación actual de la red, en cuyo caso se utilizan, reduciendo las pruebas de envío y recepción de paquetes. Si los datos históricos no existieran, no fueran suficientes o resultaran inadecuados para la situación actual de la red, se deriva automáticamente al método de medida Medio sin Histórico, basado netamente en pruebas de envío y recepción de paquetes para el momento actual.

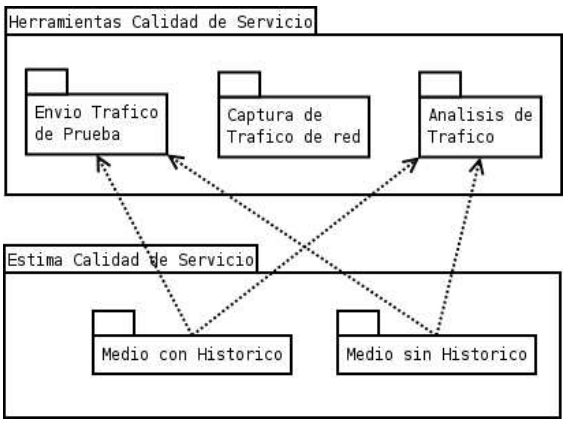

Fig. 7. Métodos y herramientas: separación y relaciones.

La figura 8 muestra las clases principales con las que se implementaron las herramientas de medida, así como algunos de los métodos.

X. Conclusiones.

La determinación de calidad de servicio en voz sobre IP no puede considerarse, a la fecha, un tema donde se haya dicho la última palabra, ni mucho menos. Aunque se lleva a cabo una intensa investigación sobre el tema no se encontró un

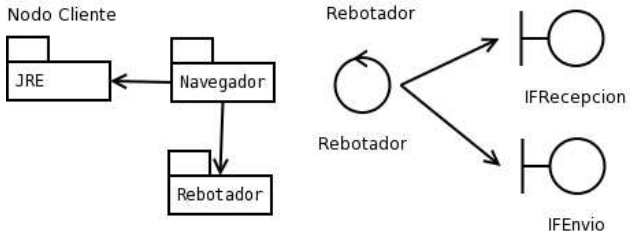

Fig. 5. Nodo cliente: diagrama de paquetes y detalle de análisis del rebotador.

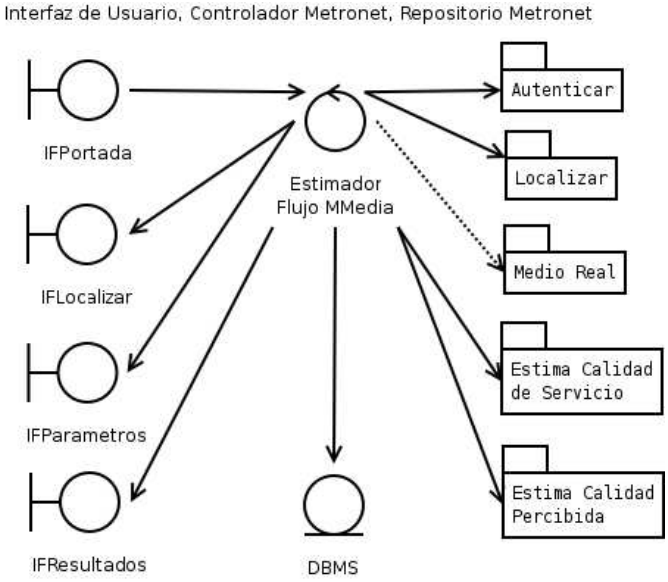

Fig. 6. Arquitectura de tres capas.

software de medición de calidad de servicio de características similares al presentado contra el cual pudieran hacerse comparaciones.

En cuanto al diseño del software, Metronet está atado a los resultados de una investigación sobre el tema y una metodología de medición propuesta. Por ello se buscó mantener desacoplados los módulos donde los cambios son más probables, reuniéndolos en el paquete denominado HerramientasQoS. Aún dentro de este paquete se intentó mantener desacoplados sus componentes. Existió un esfuerzo conciente de preparación para el cambio mediante el uso del paradigma de arquitectura en tres capas ("three-tier architecture")[3]. La gestión del proyecto se llevó a cabo coordinando el desarrollo paralelo de la investigación y el software, lo cual permitió uniformizar la carga de trabajo en las diferentes especialidades del equipo y alcanzar la meta en los plazos previstos.

Una limitación menor para el uso práctico es la necesidad de contar con una máquina virtual Java en el equipo del cliente. Dada la facilidad de obtener este software sin costo en Internet la alternativa de manejar diferentes versiones del software para la plataforma del usuario pareció una complicación mayor que la de obtener la máquina virtual Java, muchas veces ya existente en las instalaciones de las máquinas cliente.

Aunque el trabajo en un área dinámica como la estimación de calidad de servicio en redes IP hace esperar la aparición de metodologías nuevas y mejores, el diseño del software permite una adaptación fácil, concentrando los cambios en puntos bien específicos, no afectando el resto del sistema.

El software presentado, por su facilidad de instalación y uso en la máquina del cliente, permite encarar un esquema de verificación extenso, involucrando un amplio espectro de usuarios, desde clientes individuales o corporativos de proveedores Internet hasta organismos independientes de verificación de contratos de servicio. Se espera así cubrir una variedad de situaciones y topologías de red en

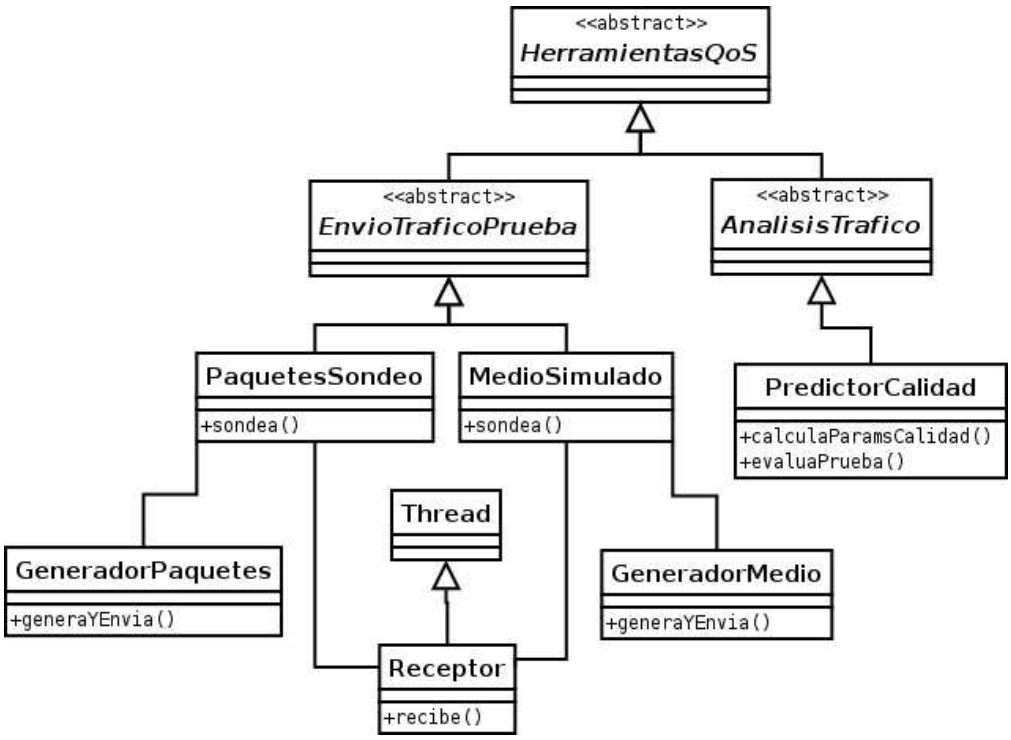

Fig. 8. Herramientas para calidad de servicio, clases principales.

la verificación de las predicciones. Esta tarea está actualmente en ejecución.

AGRADECIMIENTOS.

A Laura Aspirot y Paola Bermolen, por su participación en la discusión y comentarios al análisis y diseño del sistema.

CRÉDITOS.

El proyecto Metronet fue desarrollado en el Instituto de Ingeniería Eléctrica, Facultad de Ingeniería, Universidad de la República (UdelaR), Montevideo, Uruguay. Participaron en él, por orden alfabético de apellidos, Laura Aspirot, Pablo Belzarena, Paola Bermolen, Pedro Casas, Víctor González Barbone, Federico Larroca, Gonzalo Perera.

REFERENCES

[1] Aspirot, L; Belzarena, P.; Perera, G.; Bazzano, B. End-to-end Quality of Service Prediction Based on Functional Regression. ARTES, Facultad de Ingeniería, Universidad de la República, Montevideo, Uruguay, 2005; HET-NET 2005. http://iie.fing.edu.uy/investigacion/grupos/artes/publicaciones/hetnet05.pdf

[2] Ferraty, F., Goia, A., y Vieu P. Functional nonparametric model for time series: a fractal approach for dimension reduction. Test, Vol 11, No. 2, pp 317-344; 2002.

[3] Grady Booch, James Rumbaugh, Ivar Jacobson. El lenguaje unificado de modelado. Addison-Wesley, 1999. ISBN 84-7829-028-1.

[4] Instituto de Ingeniería Eléctrica, Facultad de Ingeniería, UdelaR. Página del servidor para realizar las pruebas: http://164.73.38.191/MetroNet

[5] Python. http://www.python.org.

[6] Pedro Casas, Ignacio Irigaray y Diego Guerra. "Calidad Percibida en Voz y Video sobre IP". Trabajo de tesis de graduación, Biblioteca Central de la Facultad de Ingeniería, UdelaR. http://iie.fing.edu.uy/investigacion/grupos/artes/pqos

[7] Tanenbaum, Andrew S. Computer Networks. 4th. Edition. Pearson Education, 2003. ISBN 0-13-066102-3.

[8] Sun Microsystems. Sun Developer Network (SDN), http://java.sun.com.

[9] Sun Microsystems. The Java Tutorial. Lesson: Applets. http://java.sun.com/docs/books/tutorial/deployment/applet/index.html.