Capítulo 3

Arquitectura

Este sistema de manera global presenta cuatro módulos: bearerbox de Kannel, kjGateway, Java Message Service y aplicación de servicio (Ver figura 3.1).

bearerbox

Este es el módulo de kannel que se conecta a una central de sms (SMSC), es la parte básica de un ESME (External Short Message Entity). Es el nivel más bajo de conexión que concede un proveedor de servicios de texto en telefonía celular para agregadores de servicio. El protocolo más popular para este efecto es SMPP mismo que ha sido elegido para realizar las pruebas.

kjGateway

Este es el módulo principal. Es un intermediario entre kannel y JMS. Al integrarse con JMS, características como persistencia y transacciones pueden ser fácilmente configuradas.

Éste se comunica con el bearerbox usando el protocolo interno de Kannel, mismo que fue decodificado e implementado en Java para la realización de este proyecto. En la figura 3.2 podemos ver de modo un poco más detallado los módulos internos de kjGateway.

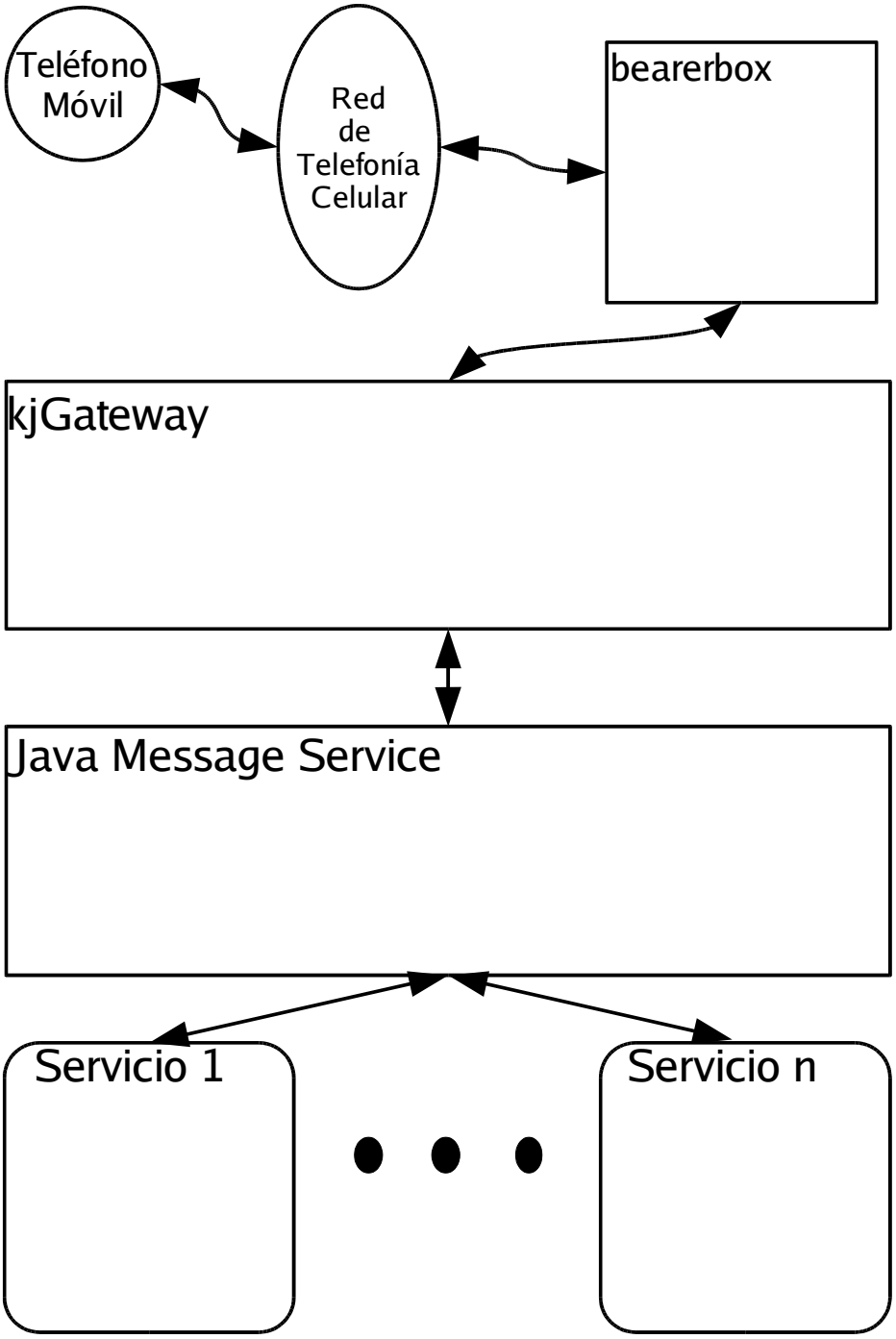

Figura 3.1: Modo en que cada componente se conecta en un sistema sms gateway basado en JMS

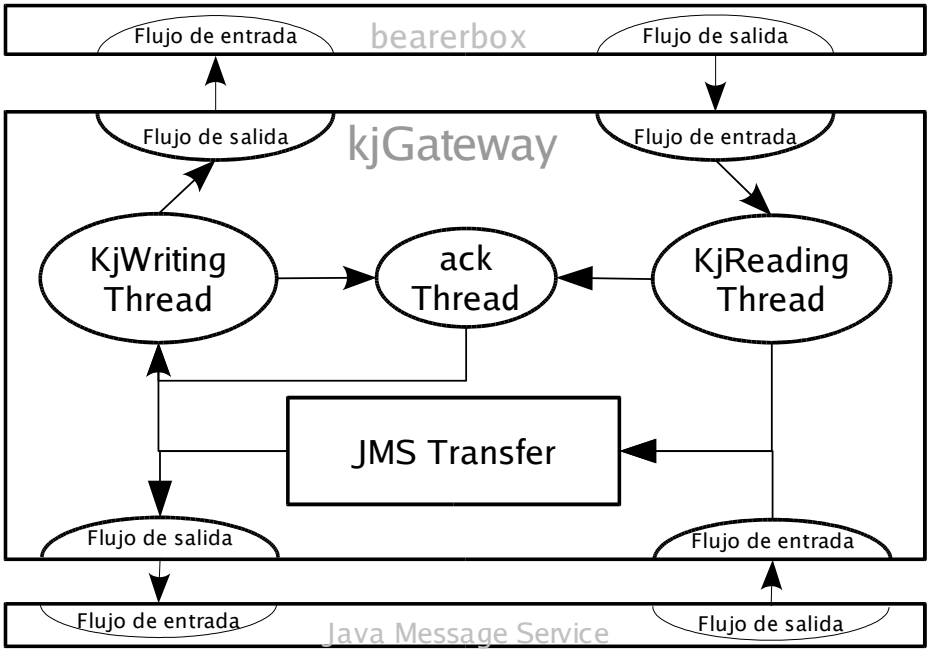

Figura 3.2: Arquitectura de kjGateway

Para detallar el diseño de este módulo, podemos observar en el diagrama de la figura 3.3 cómo es que las clases y objetos que modelan este componente se relacionan.

La clase principal desde el punto de vista del gateway es la clase `KannelJMSGateway`, que instancía a las clases `KjWritingThread`, `KjReadingThread`, `AckCycleThread`, `KannelBinding` y una implementación de la clase `JMSTransport` que es cargada de manera dinámica como se le indique desde un Objeto `Properties`.

kjReadingThread

Esta clase es un subproceso o hilo encargado de leer mensajes entrantes del bearerbox usando la conexión especificada por `KannelBinding`. Este thread hace una lectura que se bloquea hasta que recibe un mensaje completo; al desbloquearse, si el mensaje es válido, lo procesa de acuerdo a su tipo:

1. SMS - Llama a un método de la interfase `JMSTransport` para indicarle que un nuevo mensaje ha llegado.

2. ACK - Llama a un método de la clase `AckCycleThread` para indicarle que se ha recibido la confirmación de un mensaje enviado anteriomente.

3. Otros - Son ignorados ya que el prototipo no los requiere para la etapa de pruebas.

JMSTransport

Esta interfase está diseñada para intercambiar mensajes entre JMS y Kannel (Figura 3.4). Contiene los prototipos de los métodos: `gotMOMessage`, `gotMTMessage`, `addKjWritingThread` y `start`.

`gotMOMessage` - este método es llamado por `kjReadingThread` para notificar que ha recibido un mensaje originado en la red de telefonía, típicamente un teléfono celular.

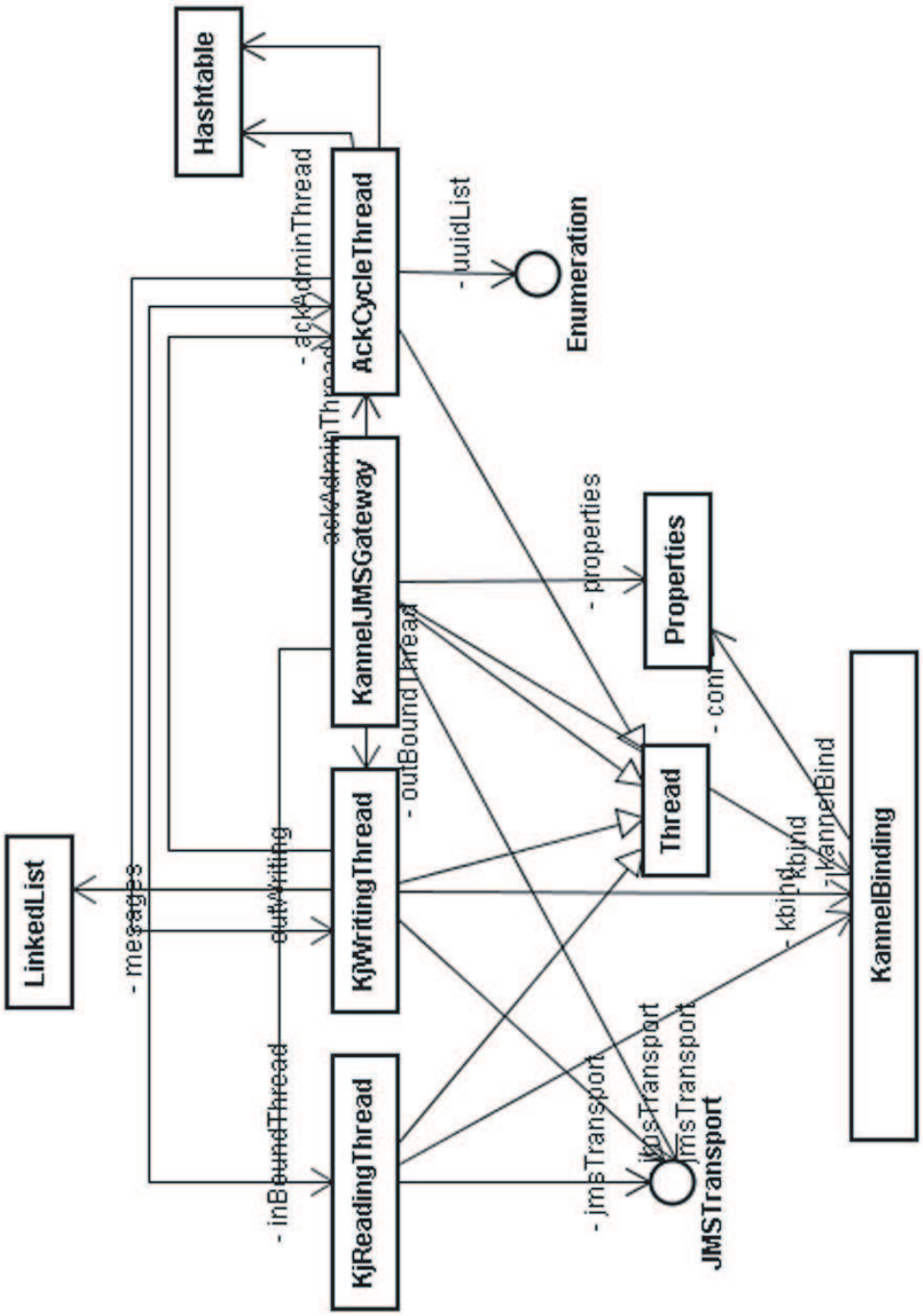

Figura 3.3: Diagrama de clases de kjGateway

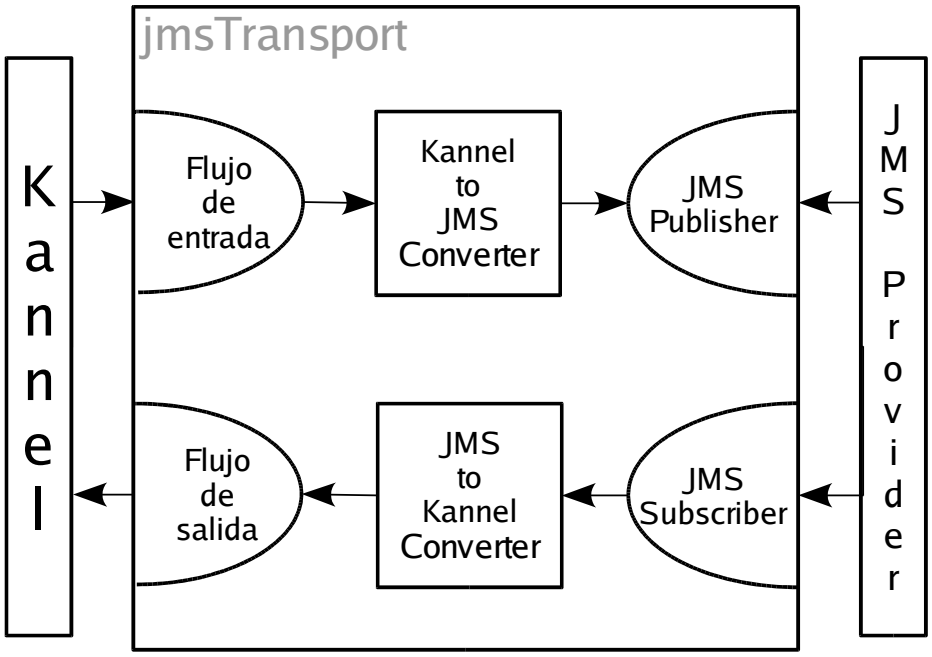

Figura 3.4: Funcionamiento de jmsTransport

`gotMTMessage` (uso futuro) - La semántica de este método se refiere a las acciones a llevar a cabo cuando un mensaje con destino a la red de telefonía ha llegado al sistema.

`addKjWritingThread` - Agrega un objeto de la clase `KjWritingThread` al objeto actual, para permitirle enviar mensajes a la red de telefonía celular.

`start` inicializa un objeto de esta clase tomando las propiedades que recibe como parámetro. Típicamente, este método implementará los mecanismos necesarios para permitir que las funciones básicas del mismo estén vigentes.

kjWritingThread

Este subproceso se encarga de escribir mensajes para el bearerbox, ya sea de contenido o de control. Cuando envía un mensaje de contenido, manda una copia del mismo mensaje al AckCycleThread en espera de una confirmación de entrega. El principal

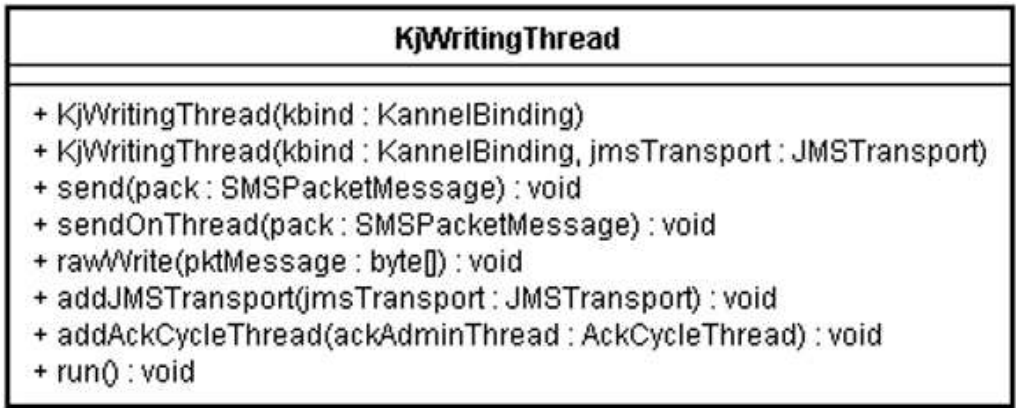

Figura 3.5: Diagrama de clase de kjWritingThread

método que implementa esta clase es `send`.

Esta clase (Figura 3.5) puede funcionar en forma de hilo y en forma de objeto normal, la diferencia es que cuando funciona como thread, se usa el método `sendOnThread` para enviar mensajes. Al llamar a este método, los mensajes se agregan a una lista ligada (`LinkedList`) que funciona como un buffer y se intentan enviar los mensajes de la cola en un ciclo que se repite en un lapso de tiempo determinado. En el caso contrario, los mensajes son enviados tan pronto como sea posible, bloqueando el método hasta que se logre enviar el mensaje; esto significa que si existe previamente un buffer para escribir, el método podría bloquearse indefinidamente.

AckCycleThread

Esta clase es un subproceso que administra el reenvío de mensajes, y su tiempo de vida dentro de la aplicación, es decir, cuanto tiempo hay que esperar antes de reenviar un mensaje o si se ha recibido una confirmación de su entrega.

Cada vez que un mensaje es enviado, esta clase es notificada en espera de un acuse de recibo (acknowledge) de la parte receptora, en caso de no recibir tal mensaje en un periodo determinado de tiempo, intentará reenviar dicho mensaje nuevamente hasta

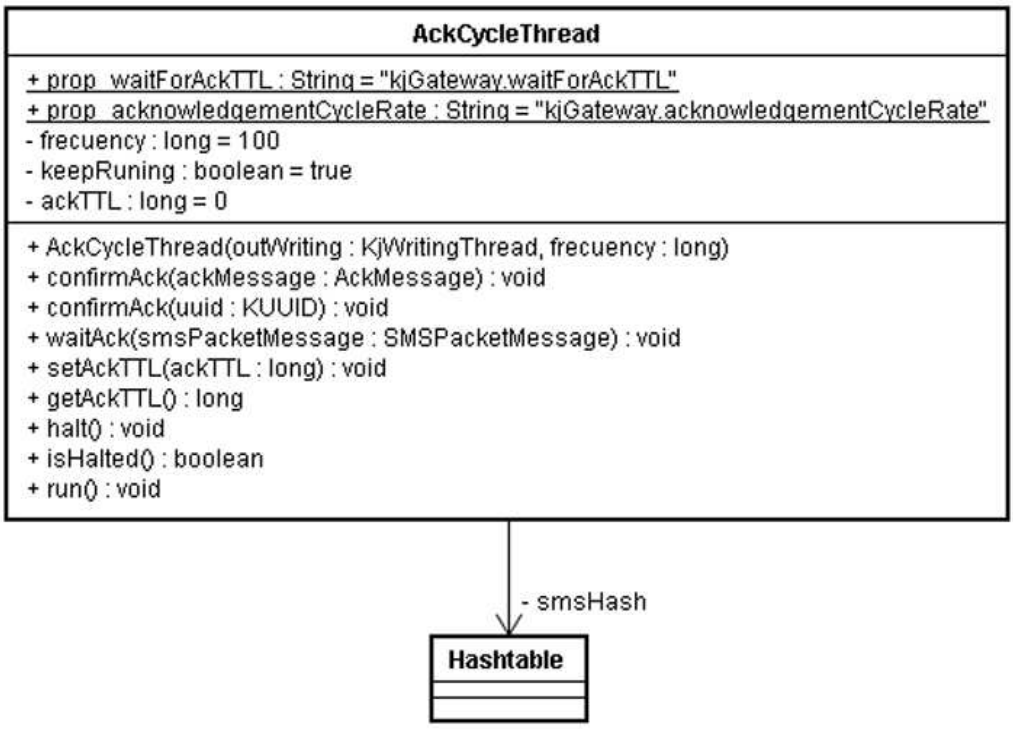

Figura 3.6: Diagrama de clase de AckCycleThread

expirar su tiempo de vida o hasta recibir dicha respuesta. Figura 3.6.

Sus principales métodos son `waitAck` y `confirmAck`. `waitAck` recibe un mensaje de texto para el que hay que esperar un acknowledge. `confirmAck` recibe un acknowledge que en caso de coincidir con un mensaje, confirmará su recepción.

El funcionamiento de esta clase está basado en un Hashtable que contiene los mensajes indexados usando su correspondiente UUID, cuya confirmación es esperada. La llamada de `waitAck` agrega una entrada en dicho Hashtable y la llamada de `confirmAck` lo elimina. Cada determinado lapso de tiempo (`frecuency`) se intentan reenviar los mensajes aún contenidos en el Hashtable.

kjProtocol API

Las conexiones con Kannel usan una implementación de un protocolo de bajo nivel

creado por los primeros desarrolladores de Kannel, a ésta se le ha llamado kjProtocol. Contiene las clases esenciales (Figura 3.7) para leer y escribir mensajes del mismo modo que lo hacen los módulos de Kannel, permitiendo así el desarrollo de componentes compatibles con Kannel bajo Java.

`BasicPacket`

Esta clase 3.8 hereda los atributos `length` y `type` para los demás tipos de paquetes, de modo que los métodos que requieran saber el tipo o longitud de un mensaje, puedan hacerlo sin tener que escribir un método sobrecargado para cada tipo de paquete existente.

`BasicKannelProtocolMessage`

Esta interfase especifica los métodos `getMessage` y `setMessage` que son usados por métodos que leen o escriben paquetes de un flujo de datos conectado con un módulo de kannel, en este caso con un bearer box.

Aplicación de servicio

Son módulos que se conectan usando JMS como protocolo de conexión para intercambiar mensajes SMS entre la aplicación o servicio y los clientes de la red de telefonía. Algunos ejemplos de servicios que pueden ser ofrecidos son los micro pagos por servicios materiales o la venta de contenido de personalización para celulares. En el capítulo 4.3 se presenta de manera más detallada el diseño e implementación de una aplicación de servicio de ejemplo.

Conclusión

En este capítulo, hemos mostrado de modo general como es que los principales módulos están relacionados con las clases que fueron implementadas

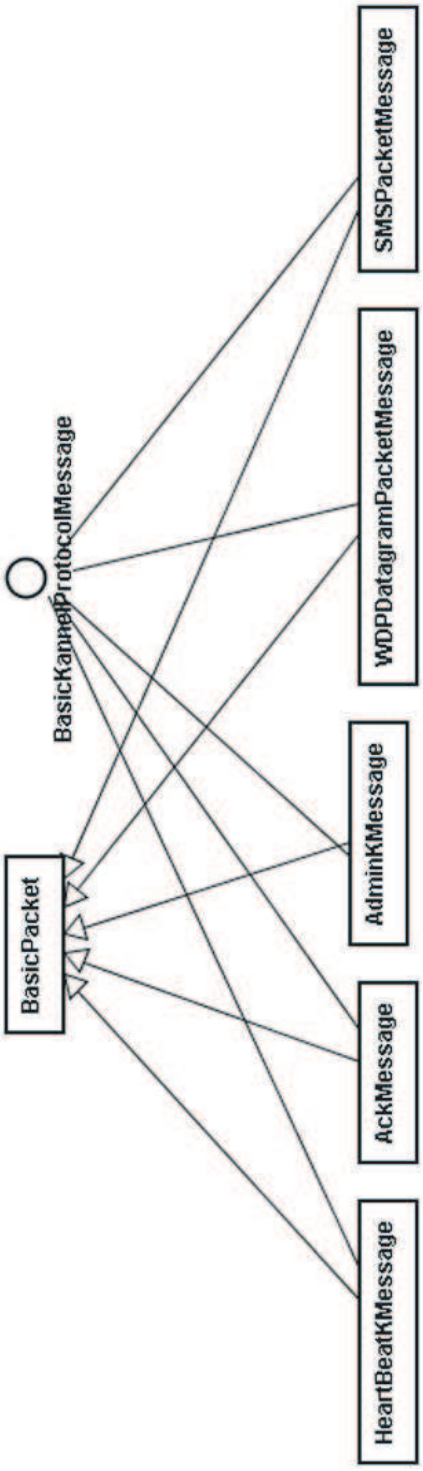

Figura 3.7: Diagrama de clases principales del API kjProtocol

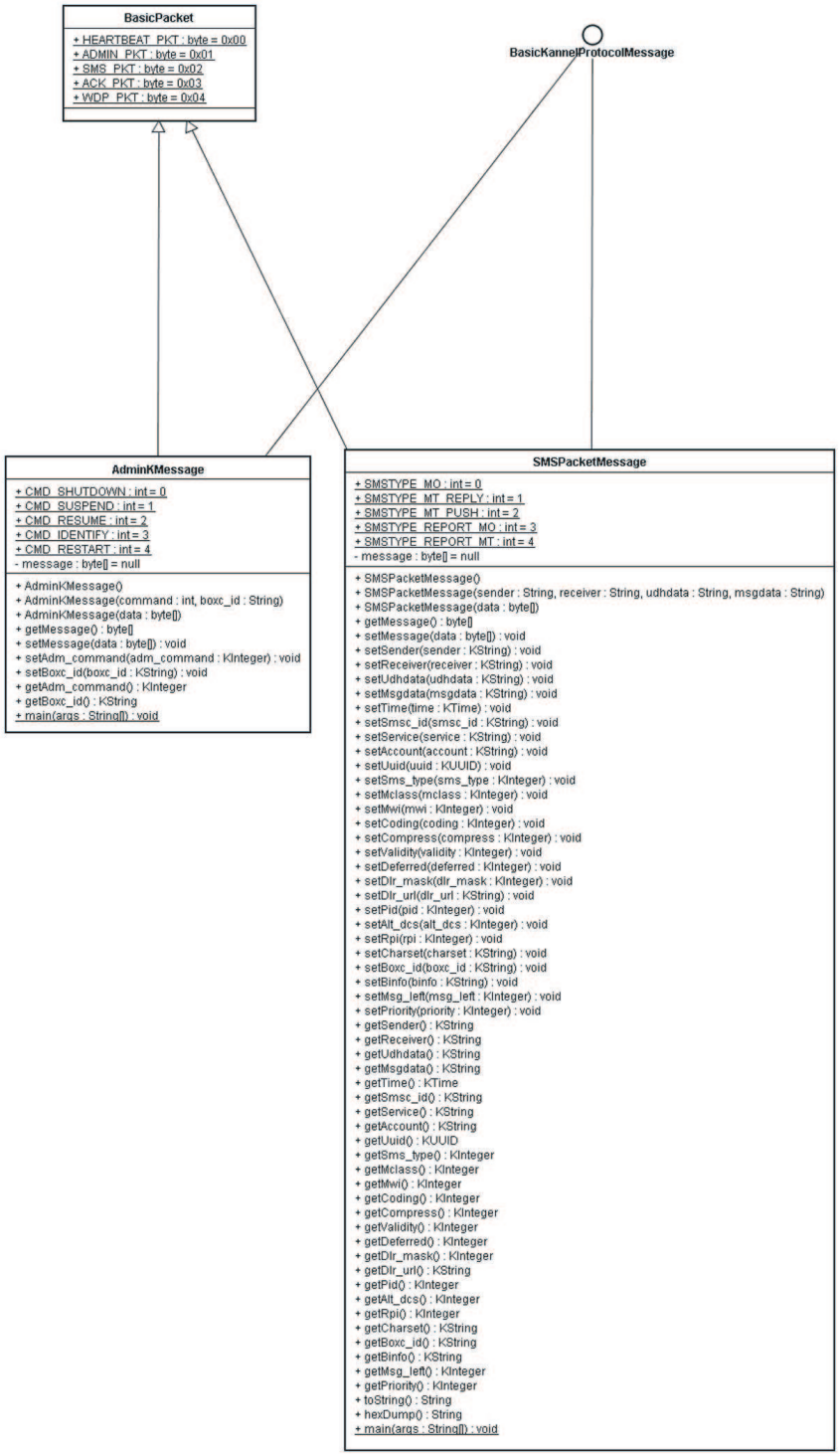

Figura 3.8: Diagrama de clases para SMSPacketMessage, AdminKMessage, BasicPacket y BasicKannelProtocolMessage