SocketHubAppender

Sends LoggingEvent objects to a set remote a log servers, usually a SocketNode.

Acts just like SocketAppender except that instead of connecting to a given remote log server, SocketHubAppender accepts connections from the remote log servers as clients. It can accept more than one connection, and when a log event is handled, the event is sent to the set of currently connected remote log servers. Implemented this way it does not require any update to the configuration file to send data to another remote log server. The remote log server simple connects to the host and port the SocketHubAppender is running on.

However, given the nature of accepting connections on-the-fly, it cannot be guaranteed that all events will be received while the tcp connection is in process. But once connected, it should behave the same as SocketAppender.

This implementation borrows heavily from the SocketAppender implementation as an example.

The SocketHubAppender has the following properties:

*If sent to a SocketNode, remote logging is non-intrusive as far as the log event is concerned. In other words, the event will be logged with the same time stamp, org.apache.log4j.NDC, location info as if it were logged locally by the client.
*SocketHubAppenders do not use a layout. They ship a serialized LoggingEvent object to the server side.
*Remote logging uses the TCP protocol. Consequently, if the server is reachable, then log events will eventually arrive at the server.
*If no remote servers are attached, the logging requests are simply dropped.
Logging events are automatically *buffered by the native TCP implementation. This means that if the link to server is slow but still faster than the rate of (log) event production by the client, the client will not be affected by the slow network connection. However, if the network connection is slower then the rate of event production, then the client can only progress at the network rate. In particular, if the network link to the the server is down, the client will be blocked. On the other hand, if the network link is up, but the server is down, the client will not be blocked when making log requests but the log events will be lost due to server unavailability.
*If the JVM hosting the SocketHubAppender exits before the SocketHubAppender is closed either explicitly or subsequent to garbage collection, then there might be untransmitted data in the pipe which might be lost. This is a common problem on Windows based systems. To avoid lost data, it is usually sufficient to close the SocketHubAppender either explicitly or by calling the shutdown method before exiting the application.