This is gonna be a convenient way to synchronize notes on different laptops.

What’s the difference between socket and web server?

Usually web server is application,works on application level.Port ,packet ,protocol have already been defined by the application. Socket works on network level ,for which Port ,packet ,protocol need to be designed by user.

Socket Programming VS. Web service?**[[1]](#footnote-1)**

Webservices are generally speaking "easier" to do, thanks to the tremendous interest in them and the support for them in developer tools and through libraries and frameworks.

However, especially if your payload is small (think messages the size of a typical SMS or tweet), the overhead you create with webservices is prohibitive: bytes sent over a wireless network like GPRS or UMTS are still very expensive, compared to bytes carried over cable or ADSL. And web services carry several layers of invisible info around that the end customer will also have to pay.

So, if your use case is based on short messages, I'd at least advise to do some bandwidth simulation calculations, and base your decision on bandwidth savings vs increased complexity of your app.

While looking at sockets, also have a look at UDP: if you can live with the fact that basically you throw someone a packet, and without designing some ack mechanism into your protocol you'll never be sure the message arrived, it's very efficient because there is no traffic to create and maintain a connection, and even pretty long messages can very well be transported inside 1 UDP packet.

EDIT based on comment:

• stream socket: not sure how you define streams, but streams and messages are two very distinct concepts for me, a stream is a typically longer sequence of data being sent, whereas a message is an entity that's sent, and optionally acknowledged or answered by the receiver.

• bandwidth simulation: the easiest way to understand what I'm talking about is to get [Wireshark](http://www.wireshark.org/) and add up everything that gets transported across the net to do a simple request of a very short string - you'll see several layers of administrative info (ie invisible, just there to make the different protocol layers work) that are all traffic paid for by the end user. Then, write a small mock service using UDP to transport the same message, or use a tool like [netcat](http://netcat.sourceforge.net/), good tutorial [here](http://www.adamsinfo.com/netcat-tutorial-for-linux-windows-howto-nc/), and add up the bytes that get transported. You'll see pretty huge differences in the number of bytes that are exchanged.

EDIT2, something I forgot to mention: mobile networks used to be open, transparent networks with devices identified by public IP addresses. There's a rapid evolution towards NATed mobile networks which has its impact on how devices inside and outside this "walled garden" can communicate ([NAT traversal](http://en.wikipedia.org/wiki/NAT_traversal)). You'll need to take this into account when designing your communication channel.

As for the use of streams for a chat application: it offers some conceptual advantages, but you can very well layer a chat app on top of UDP, look [here](http://www.codeproject.com/KB/IP/ChatAppAsynchUDPSocks.aspx) or [here](https://decibel.ni.com/content/docs/DOC-16654)

1. <http://stackoverflow.com/questions/7237459/socket-programing-vs-web-service> [↑](#footnote-ref-1)