

Re: Suggestions for custom application-layer protocol?

Source: <http://linux.derkeiler.com/Newsgroups/comp.os.linux.embedded/2005-05/0127.html>

From: Tom Anderson (*twic_at_urchin.earth.li*)

Date: 05/26/05

Date: Thu, 26 May 2005 15:51:23 +0100

On Wed, 25 May 2005, Mark wrote:

- > *I need to implement a simple application layer protocol that will be*
- > *used to communicate between an embedded device (single-board computer*
- > *running Linux) and a monitoring terminal (running Windows). There isn't*
- > *alot of data being passed around, mostly status information collected by*
- > *the embedded device, and control messages from the monitoring terminal.*
- > *Messages will consist of between 1 and 10 fields of data.*
- >
- > *I want the protocol to be text based rather than binary since the data*
- > *throughput is low. I also want the protocol to be based on TCP/IP.*
- > *Does anyone here have any suggestions on the design of a simple*
- > *protocol? Are there simple, standard ways of formatting text messages*
- > *to be sent over a TCP socket (such as comma-separated)? I would prefer*
- > *to avoid the complexities of XML.*

I've written a bunch of little protocols like this. I think it really is as simple as it sounds – use TCP, frame messages by terminating with a line break (i'd tolerate CR, LF or CR+LF, and consistently emit CR + LF, i think), split each message into fields with tab characters, use the first field for a command or status code, and put your data in the other fields. If you want to be able to have multiple commands in flight at once, use the first field for a transaction tag (as in IMAP), and the second field for command or status.

- > *Since either the embedded device or the monitoring terminal can initiate*
- > *a message, is it preferable to have TCP servers running on both sides?*
- > *Or is it better to simply leave a TCP connection open between client and*
- > *server?*

The latter. Engineer the protocol to be stateless, though, so if you lose the connection, you can just throw up a new one and carry on.

As for versioning, if you think implementations of this might persist in the wild for several years, or if anyone else is going to implement it, by all means do some version negotiation, but otherwise, if you control b