

Re: how to control UDP sending Speed?

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework/2007-03/msg00318.html>

- *From:* "Peter Duniho" <NpOeStPeAdM@xxxxxxxxxxxxxxxxxxx>
 - *Date:* Sat, 17 Mar 2007 17:32:20 +0800
-

On Sat, 17 Mar 2007 10:54:51 +0800, Victor <vvhh2002@xxxxxxxxxxxxxxxxxxx> wrote:

Thanks, Pete

I just want program a UDP stream sending application for a IPQAM device that is needed by Digial TV VOD Broadcast system. The speed of my server and a IPQAM device is 10000Mbps.

The IPQAM device only can accept a constant bit rate udp TS(mpeg2 transport stream) data stream for broadcasting TV.

What is the network configuration here? Does the IPQAM device receive the UDP datagrams directly? What is its relationship to the sender? Are they on the same computer? On the same LAN? Or communicating over the Internet and/or some other telecommunications link? Does the sender get any feedback whatsoever regarding the state or status of the IPQAM device, and/or any other components in the network involved in transmitting the data?

What does it mean when you write "only can accept a constant bit rate UDP TS data stream"? I ask, because there is no such thing as a genuinely "constant bit rate UDP stream". UDP isn't stream-oriented in the first place, and as a protocol that's part of TCP/IP there's definitely no ensuring that the UDP datagrams arrive in an exact frequency (for that matter, UDP doesn't guarantee delivery, doesn't guarantee unique delivery, and doesn't guarantee in-order delivery). It seems likely to be that if the IPQAM device is actually receiving the UDP datagrams directly, it *must* have some sort of buffering. So you shouldn't really need for the transmitted datagrams to be sent at a perfect delivery sequence. They just need to arrive fast enough to keep the IPQAM device's buffer(s) filled (or nearly so), without going too fast (causing UDP datagrams to be dropped).

If the IPQAM isn't receiving the UDP datagrams directly, then obviously there's some other component that is and is passing them along. That other component should be able to do the buffering and even out the flow of data to the IPQAM device proper.

Keep in mind that I have no idea what an "IPQAM device" is. :) All I know is what you've written here, which is that it's some sort of video receiver that uses UDP.

As far as your implementation goes, I would offer a few thoughts: one is that you probably don't need the high-resolution timer, since network i/o is a) not that granular anyway, and b) operates in the realm of milliseconds, not microseconds. The second is that rather than looping, it would probably make more sense to just sleep for a given number of milliseconds. This will avoid too much CPU overhead and weird thread

Re: how to control UDP sending Speed?

starvation bugs (which ironically can cause your network i/o to become more irregular, even as you are trying to even things out and reduce it). Yes, it's true that the Windows thread scheduler does not guarantee that you'll be woken up exactly after the time you specify in Sleep(), but as long as you are calculating a new wait interval each time (and it appears that you are), then you'll be pretty close and will maintain a proper average with a minimal of overhead.

It seems to me that if you send an amount of data with each interval that is somewhere between 10% and 50% of the IPQAM device's network i/o buffers, while at the same time ensuring that you are waiting for some reasonable amount of time (ie tens of milliseconds), things ought to work reasonably well. Of course, these two parameters may not be mutually compatible, depending on how large the IPQAM device's buffers are. You may have to fiddle with the specifics to get something that works just right.

As far as the question of writing your own network driver goes, I don't really know. You'll have to learn all about the low-level definitions of UDP and IP (on which UDP is built), as well as the intricacies of driver development under Windows (see the Windows DDK). I think that the network driver is a user-mode driver, which should simplify things, but I'm not sure about that, and even if it is, you still have the issue of interacting with the network hardware driver at a lower level. Bottom line is that it's gonna be complicated if you go that route. The other thing is that even if you develop your own driver, you may not have control over the network components between your driver and the IPQAM device. Just because you send data out at a perfectly regulated interval, that will not ensure that the data arrives at the device at a perfectly regulated interval. The best-case scenario will be where that device is actually on the same computer, and even there the way that i/o happens on Windows will prevent you from achieving a perfectly regulated delivery.

And of course, I will reiterate: unless you are sending data one byte at a time, the blocks of data you send are still going to be transmitted at the nominal rate of the network connection. If you are sending 1K blocks once per second, then that will achieve a 1K/sec data rate, but each 1K block is going to be sent at whatever rate the network is (100Mbps, for example).

Since I don't know what an "IPQAM device" is, I cannot comment with much authority on its design. But, assuming it is designed to be even minimally aware of TCP/IP networks, there should be no reason at all that your data has be sent at *exactly* some rate. TCP/IP networks just don't work that way, and so the designers of the device surely have anticipated this and provided for some buffering to allow for somewhat uneven delivery. You should be able to target an average, and as long as you do that correctly, things should work fine.

If I were you, I would look more closely at the implementation of the average solution first, to see if it can be improved, rather than diving into a much more complicated proposal such as writing your own network driver.

Pete

.