# Broadcast like UDP with the reliability of TCP

Asked 16 years, 3 months ago Modified 4 years, 9 months ago

Viewed 12k times

19

I'm working on a .net solution that is run completely inside a single network. When users make a change to the system, I want to launch an announcement and have everyone else hear it and act accordingly. Is there a way that we can broadcast out messages like this (like UDP will let you do) while keeping guaranteed delivery (like TCP)?

This is on a small network (30ish clients), if that would make a difference.

.net network-programming tcp udp

Share

Improve this question

Follow

asked Aug 28, 2008 at 2:55

Jeffrey

1,667 ● 5 ● 25 ● 41

16 Answers

Sorted by: Highest score (default)

15

Almost all games have a need for the fast-reacting properties (and to a lesser extent, the connectionless properties) of UDP and the reliability of TCP. What they do is they build their own reliable protocol on top of UDP. This gives them the ability to just burst packets to whereever and optionally make them reliable, as well.

The reliable packet system is usually a simple retry-until-acknowledged system simpler than TCP but there are protocols which go way beyond what TCP can offer.

Your situation sounds very simple. You'll probably be able to make the cleanest solution yourself - just make every client send back an "I heard you" response and have the server keep trying until it gets it (or gives up).

If you want something more, most custom protocol libraries are in C++, so I am not sure how much use they'll be to you. However, my knowledge here is a few years old - perhaps some protocols have been ported over by now. Hmm... RakNet and enet are two C/C++ libraries that come to mind.

Share Improve this answer Follow

edited Sep 1, 2008 at 4:24

answered Aug 28, 2008 at 7:16

Sander

26.3k ● 3 ● 54 ● 88

Regarding *"which go way beyond what TCP can offer"*, how is that possible since TCP is specifically made to address reliable delivery? Is there something that TCP lacks in a reliable packet system? – Pacerier Feb 12, 2015 at 4:17

1 @Pacerier TCP will withold data it has already received if there's even a little part missing from the stream. Perhaps in some cases the requirement is different - get data as soon as it arrives, even if it is out of order. – Tomas Andrle Feb 27, 2018 at 13:59

10

Take a look at sctp which has a combination of tcp and udp features. There is a windows implementation available.

Share Improve this answer Follow

answered Aug 28, 2008 at 7:24

epatel
**46k** ● 17 ● 111 ● 144

1 +1 for suggesting sctp. However sctp really hasn't had time to mature as much as TCP or UDP. A lot of drivers, nics and switches havent been super optimized for it like they have for TCP. Theres an interesting performance comparison here - datatag.web.cern.ch/datatag/WP3/sctp/tests.htm – quixver Aug 30, 2013 at 16:17

7

You could use Spread to do group communication.

Share Improve this answer

answered Aug 28, 2008 at 3:40

Follow

Cagatay
1,372 ● 1 ● 12 ● 16

I'll second this. Using somebody else's library is your best bet to get this correct. – Jay Kominek Nov 23, 2008 at 15:01

1 @JayKominek, Depends on who the "somebody else" is. – Pacerier Feb 11, 2015 at 12:36

3

@epatel - I second the SCTP suggestion (I voted up, but can't comment yet so additional stuff here).

SCTP has many great features and flexibility. You can sub-divide your connection into multiple streams, and choose the reliablity of each and whether it is ordered or not. Alternatively, with the Partially Reliability extension, you can control reliability on a per message basis.

Share Improve this answer

Follow

edited Sep 11, 2008 at 12:22
Jeff Atwood
63.9k ● 48 ● 151 ● 153

answered Sep 11, 2008 at 10:27
Andrew Johnson
7,289 ● 4 ● 25 ● 27

Broadcast is not what you want. Since there could and probably will be devices attached to this network which

2

don't care about your message, you should use Multicast. Unlike broadcast messages, which must be sent to and processed by every client on the network, Multicast messages are delivered only to interested clients (ie those which have some intention to receive this particular type of message and act on it).

If you later scale this system up so that it needs to be routed over a large network, multicast can scale to that, whereas broadcast won't, so you gain a scalability benefit which you might appreciate later. Meanwhile you eliminate unnecessary overhead in switches and other devices that don't need to see these "something changed" messages.

Share Improve this answer Follow

answered Sep 17, 2008 at 10:38

tialaramex
3,801 ● 1 ● 23 ● 23

You do not know that from the question ,, eg in a wireless domain you will be blocked out of the network while the message is being sent through the air regardless so it always more efficient to broadcast , reading it and ignoring it is trivial cost . Some use cases run dedicated networks eg mines , sensor nets , control systems etc etc – user1496062 May 21, 2015 at 2:23

1 It's not "more efficient to broadcast", it's exactly the opposite. That's my point. – tialaramex Jun 5, 2015 at 12:49

You might want to look into RFC 3208 "PGM Reliable Transport Protocol Specification".

2

Here is the abstract:

> Pragmatic General Multicast (PGM) is a reliable multicast transport
> protocol for applications that require ordered or unordered,
> duplicate-free, multicast data delivery from multiple sources to
> multiple receivers. PGM guarantees that a receiver in the group either receives all data packets from transmissions and repairs, or is able to detect unrecoverable data packet loss. PGM is specifically intended as a workable solution for multicast applications with basic reliability requirements. Its central design goal is simplicity of operation with due regard for scalability and network efficiency.

Share Improve this answer Follow

edited Oct 7, 2021 at 6:09

Community Bot
1 ● 1

answered Nov 23, 2008 at 14:53

Bruno Rijsman
3,796 ● 4 ● 35 ● 63

---

2

You could use a Message Broker, such as ActiveMQ. Publish your messages to a topic and have the clients register durable subscriptions to the topic, so that they won't miss any messages even if they are not online.

> Apache ActiveMQ is a message broker written in Java together with a full JMS client. However Apache ActiveMQ is designed to communicate over a number of protocols such as Stomp and OpenWire together with supporting a number of different language specific clients.

Client platform support includes c# and .net

Share Improve this answer Follow

answered Jan 20, 2010 at 7:14

crowne
8,524 ● 3 ● 40 ● 51

1

You could implement your own TCP-like behaviour at the application layer.

So for instance, you'd send out the UDP broadcast, but then expect a reply response from each host. If you didn't get a response within X seconds, then send another and so on until reaching some sort of threshold. If the threshold is reached (i.e. the host didn't respond at all), then report an error.

To do this though, you'd need a pre-defined list of hosts to expect the responses back from.

Share Improve this answer Follow

answered Aug 28, 2008 at 5:19

Steve M
10.6k ● 12 ● 53 ● 63

---

1

Create a TCP server. Have each client connect. In your TCP protocol with the clients, create each packet with a 2-byte prefix of the total size of the following message.

Clients then call `read(max_size=2)` on the socket to determine the size of the next message, and then `read(max_size=s)` to collect the message.

You get reliable, ordered messages, simple. You don't need a messaging framework for this one.

Share Improve this answer Follow

answered Jan 20, 2010 at 6:54

Matthew Herrmann
432 ● 3 ● 3

There are 30-ish clients, up to a few hundred in the future. How are you going to support that? – Pacerier Feb 12, 2015 at 4:19

1

You'd **definitely** want to look into Pragmatic General Multicast:

> While TCP uses ACKs to acknowledge groups of packets sent (something that would be **uneconomical over multicast**), PGM uses the concept of Negative Acknowledgements (NAKs).

For further G-diving, the term you're looking for is reliable multicast. Also take a look at Multipath TCP.

Share Improve this answer Follow

edited Mar 4, 2020 at 20:06

answered Feb 12, 2015 at 4:44

Pacerier

89.5k ● 111 ● 383 ● 644

0

What you can do is that after the broadcast have the **clients** initiate the tcp connections. Otherwise you just have to keep a list of all clients and initiate the connections to each client yourself.

Share Improve this answer Follow

answered Aug 28, 2008 at 2:59

paan
7,182 ● 8 ● 40 ● 44

0

I think there are three options, broadly speaking:

1. Instead of broadcasting UDP, you could create an entity (a thread, process, server, service, or whatever the thing is that exists in your solution) that keeps a list of subscribers and sends unicast UDP messages to them.
2. Use UDP multicast, but you'll have to write some sort of mechanism that would take care of reliable delivery for you (i.e., retries, timeouts, etc). This also means you have to get a reply from your clients.
3. If you're not afraid of experimental transport protocols, you might look here for suggestions.,

Share Improve this answer Follow

answered Aug 28, 2008 at 3:07

Ben Collins
20.7k ● 18 ● 128 ● 190

0

Yoy should take a look at the Norm (NACK-Oriented Reliable Multicast) specification. You can find information about Norm here.

> The NORM protocol is designed to provide end-to-end reliable transport of bulk data objects or streams over generic IP multicast routing and

> forwarding services. NORM uses a selective, negative acknowledgement (NACK) mechanism for transport reliability and offers additional protocol mechanisms to conduct reliable multicast sessions with limited "a priori" coordination among senders and receivers

It somewhat very well known in the military world.

[Norm specs.]

[Norm Source]

Share Improve this answer Follow

answered Aug 28, 2008 at 7:24

Jorge Córdoba
**52.1k** ● 11 ● 82 ● 130

---

0

Why build something from scratch if you can use library? Especially for such small project?

Try use Emcaster which itself using reliable multicast messaging - PGM, is written in .NET and with full source. You will get nice pub/sub engine with topic filtering readily available. Or you can learn from code how to do it and base your own extension on it.

Share Improve this answer Follow

answered Dec 9, 2008 at 23:29

Libor
**105** ● 4

0

I think the most irritating feature of TCP in these scenarios is the ability/way of sorting incoming packets to their original order - the concept of a stream. You cannot read a byte until the byte before it arrived.

If you can live without it, you have your chance to have your protocol, fast and reliable, but not for ordering of packets! It's simply impossible to manage both of them, because you cannot order your bytes until you receive an other copy of a lost packet, that's the main tradeoff.

Share Improve this answer Follow

answered Dec 9, 2008 at 23:55

Szundi
317 ● 2 ● 10

-3

do a RDP multicast.

Share Improve this answer Follow

edited Nov 24, 2008 at 4:45

answered Aug 28, 2008 at 2:57

Tim Williscroft
3,735 ● 25 ● 37

There is no TCP multicast, I'd vote this down, but I don't have enough reputation. – tialaramex Sep 17, 2008 at 10:39