

What is the difference between a port and a socket?

Asked 16 years, 2 months ago Modified 6 months ago

Viewed 544k times



1156

This was a question raised by one of the Software Engineers in my organisation. I'm interested in the broadest definition.



network-programming

sockets

port



Share

Improve this question

Follow

edited Oct 27, 2023 at 21:12



John Smith

7,399 ● 7 ● 51 ● 63

asked Sep 30, 2008 at 10:04



Richard Dorman

24.1k ● 16 ● 49 ● 50

-
- 42 Just to reiterate, sockets are not limited to network IO. They're available in all sorts of situations for streaming data between various applications. – Oli Sep 30, 2008 at 10:14
-
- 7 Could you give 2 or 3 examples of non-network IO sockets? – Harry May 5, 2021 at 12:39
-
- 2 In the realm of a network socket, a socket object is a communication tunnel dependent on a host IP address and

37 Answers

Sorted by:

Highest score (default) 

1

2

Next



Summary

1181

A TCP socket is an endpoint *instance* defined by an IP address and a port in the context of either a particular TCP connection or the listening state.



A port is a virtualisation identifier defining a service endpoint (as distinct from a service *instance* endpoint aka session identifier).



A TCP socket is *not* a connection, it is the endpoint of a specific connection.

There can be concurrent connections to a service endpoint, because a connection is identified by *both its local and remote* endpoints, allowing traffic to be routed to a specific service instance.

There can only be one listener socket for a given address/port combination.

Exposition

This was an interesting question that forced me to re-examine a number of things I thought I knew inside out. You'd think a name like "socket" would be self-explanatory: it was obviously chosen to evoke imagery of the endpoint into which you plug a network cable, there being strong functional parallels. Nevertheless, in network parlance the word "socket" carries so much baggage that a careful re-examination is necessary.

In the broadest possible sense, a port is a point of ingress or egress. Although not used in a networking context, the French word *porte* literally means *door or gateway*, further emphasising the fact that ports are transportation endpoints whether you ship data or big steel containers.

For the purpose of this discussion I will limit consideration to the context of TCP-IP networks. The OSI model is all very well but has never been completely implemented, much less widely deployed in high-traffic high-stress conditions.

The combination of an IP address and a port is strictly known as an endpoint and is sometimes called a socket. This usage originates with RFC793, the original TCP specification.

A TCP *connection* is defined by two endpoints aka *sockets*.

An endpoint (socket) is defined by the combination of a network address and a *port* identifier. Note that

address/port does *not* completely identify a socket (more on this later).

The purpose of ports is to differentiate multiple endpoints on a given network address. You could say that a port is a virtualised endpoint. This virtualisation makes multiple concurrent connections on a single network interface possible.

It is the socket pair (the 4-tuple consisting of the client IP address, client port number, server IP address, and server port number) that specifies the two endpoints that uniquely identifies each TCP connection in an internet. (*TCP-IP Illustrated Volume 1*, W. Richard Stevens)

In most C-derived languages, TCP connections are established and manipulated using methods on an instance of a Socket class. Although it is common to operate on a higher level of abstraction, typically an instance of a NetworkStream class, this generally exposes a reference to a socket object. To the coder this socket object appears to represent the connection because the connection is created and manipulated using methods of the socket object.

In C#, to establish a TCP connection (to an existing listener) first you create a *TcpClient*. If you don't specify an endpoint to the *TcpClient* constructor it uses defaults - one way or another the local endpoint is defined. Then you invoke the *Connect* method on the instance you've

created. This method requires a parameter describing the other endpoint.

All this is a bit confusing and leads you to believe that a socket is a connection, which is bollocks. I was labouring under this misapprehension until Richard Dorman asked the question.

Having done a lot of reading and thinking, I'm now convinced that it would make a lot more sense to have a class *TcpConnection* with a constructor that takes two arguments, *LocalEndpoint* and *RemoteEndpoint*. You could probably support a single argument *RemoteEndpoint* when defaults are acceptable for the local endpoint. This is ambiguous on multihomed computers, but the ambiguity can be resolved using the routing table by selecting the interface with the shortest route to the remote endpoint.

Clarity would be enhanced in other respects, too. A socket is *not* identified by the combination of IP address and port:

[...]TCP demultiplexes incoming segments using all four values that comprise the local and foreign addresses: destination IP address, destination port number, source IP address, and source port number. TCP cannot determine which process gets an incoming segment by looking at the destination port only. Also, the only one of the [various] endpoints at [a given port number] that

will receive incoming connection requests is the one in the listen state. (p255, *TCP-IP Illustrated Volume 1*, W. Richard Stevens)

As you can see, it is not just possible but quite likely for a network service to have numerous sockets with the same address/port, but only one listener socket on a particular address/port combination. Typical library implementations present a socket class, an instance of which is used to create and manage a connection. This is extremely unfortunate, since it causes confusion and has lead to widespread conflation of the two concepts.

Hagrawal doesn't believe me (see comments) so here's a real sample. I connected a web browser to <http://dilbert.com> and then ran `netstat -an -p tcp`. The last six lines of the output contain two examples of the fact that address and port are not enough to uniquely identify a socket. There are two distinct connections between 192.168.1.3 (my workstation) and 54.252.94.236:80 (the remote HTTP server)

```
TCP      192.168.1.3:63240      54.252.94.236:80
SYN_SENT
TCP      192.168.1.3:63241      54.252.94.236:80
SYN_SENT
TCP      192.168.1.3:63242      207.38.110.62:80
SYN_SENT
TCP      192.168.1.3:63243      207.38.110.62:80
SYN_SENT
TCP      192.168.1.3:64161      65.54.225.168:443
ESTABLISHED
```

Since a socket is the endpoint of a connection, there are two sockets with the address/port combination `207.38.110.62:80` and two more with the address/port combination `54.252.94.236:80`.

I think Hagraawal's misunderstanding arises from my very careful use of the word "identifies". I mean "completely, unambiguously and uniquely identifies". In the above sample there are two endpoints with the address/port combination `54.252.94.236:80`. If all you have is address and port, you don't have enough information to tell these sockets apart. It's not enough information to *identify* a socket.

Addendum

Paragraph two of section 2.7 of RFC793 says

A connection is fully specified by the pair of sockets at the ends. A local socket may participate in many connections to different foreign sockets.

This definition of socket is not helpful from a programming perspective because it is not the same as a socket *object*, which is the endpoint of a particular connection. To a programmer, and most of this question's audience are programmers, this is a vital functional difference.

@plugwash makes a salient observation.

The fundamental problem is that the TCP RFC definition of socket is in conflict with the definition of socket used by all major operating systems and libraries.

By definition the RFC is correct. When a library misuses terminology, this does not supersede the RFC. Instead, it imposes a burden of responsibility on users of that library to understand both interpretations and to be careful with words and context. Where RFCs do not agree, the most recent and most directly applicable RFC takes precedence.

References



1. *TCP-IP Illustrated Volume 1 The Protocols*, W. Richard Stevens, 1994 Addison Wesley
2. [RFC793](#), Information Sciences Institute, University of Southern California for DARPA
3. [RFC147](#), The Definition of a Socket, Joel M. Winett, Lincoln Laboratory

Share Improve this answer

edited Oct 7, 2021 at 7:34

Follow

community wiki

-
- 10 Perhaps, a real world analogy to keywords socket and port would help those who up-voted the question. Still a great explanation! – [rohitverma](#) Feb 8, 2013 at 18:24
-
- 7 @rationalcoder - Read the whole answer. There is a difference between being defined by something and being identified by it. For example instances of a class are defined by the class. They are partly but not completely identified by it. – [Peter Wone](#) Dec 16, 2015 at 1:42 
-
- 7 It's *partly* identified by IP and port. That's enough to create one. But you can create another one with the same IP and port so long as the other end is different – [Peter Wone](#) Jan 10, 2016 at 9:11 
-
- 13 I didn't voted because I disagree with this statement - "*A socket is not identified by the combination of IP address and port:*" .. Read TCP RFC - tools.ietf.org/html/rfc793 .. It is very clear that socket is combination of IP and port, if you know IP and port then you have identified a socket or endpoint, if you know pair of socket i.e. client IP+port and server IP+port then you have identified a unique connection ..
– [hagrawal7777](#) Mar 6, 2016 at 0:52
-
- 10 "In the above sample there are two endpoints with the address/port combination 54.252.94.236:80. If all you have is address and port, you don't have enough information to tell these sockets apart. It's not enough information to identify a socket." Aren't those the same sockets, but different connections, between the two connections you have 3 sockets, 2 locals and one same server socket being connected to; or are they in fact two different sockets? There would be no telling them apart because they are the same, but to tell the connections apart you would need the different local sockets. – [Andrew Clavin](#) Jul 4, 2016 at 6:41
-



223



A socket consists of three things:

1. An IP address
2. A transport protocol
3. A port number



A port is a number between 1 and 65535 inclusive that signifies a logical gate in a device. Every connection between a client and server requires a unique socket.

For example:

- 1030 is a port.
- (10.1.1.2 , TCP , port 1030) is a socket.

Share Improve this answer

Follow

edited Jun 2, 2013 at 22:52



[user207421](#)

311k ● 44 ● 320 ● 488

answered Aug 2, 2012 at 17:10



[RT_](#)

6,409 ● 4 ● 17 ● 6

112 No. A socket consists of *five* things: {protocol, local address, local port, remote address, remote port}. – [user207421](#) Jul 22, 2013 at 1:30

2 @KorayTugay It's in the IP header. What makes you think the TCP layer can't see that? – [user207421](#) Aug 31, 2015 at 4:00

- 2 @EJP as I can see in the most voted answer above that a socket is NOT the connection itself but it is the end point of a connection then how come it can include both local and remote ports and ip addresses. A socket will represent only one side of the connection i.e. either local port and local ip address OR remote port and remote ip address. Kindly correct me if I'm wrong. – [RBT](#) Mar 17, 2016 at 8:56
-
- 2 @RBT The connection is defined by the tuple, and so therefore are the sockets that form its endpoints. See RFC 793. – [user207421](#) Jun 30, 2016 at 0:34 ✎
-
- 11 @EJP Still RFC 793 : "A pair of sockets uniquely identifies each connection. That is, a socket may be simultaneously used in multiple connections." If a socket already consisted of five things, how could there be" a pair of sockets" in my citation? – [Gab是好人](#) Jan 29, 2017 at 18:39 ✎
-



129



A socket represents a single connection between two network applications. These two applications nominally run on different computers, but sockets can also be used for interprocess communication on a single computer. Applications can create multiple sockets for communicating with each other. Sockets are bi-directional, meaning that either side of the connection is capable of both sending and receiving data. Therefore a socket can be created theoretically at any level of the OSI model from 2 upwards. Programmers often use sockets in network programming, albeit indirectly. Programming libraries like Winsock hide many of the low-level details of socket programming. Sockets have been in widespread use since the early 1980s.

A port represents an endpoint or "channel" for network communications. Port numbers allow different applications on the same computer to utilize network resources without interfering with each other. Port numbers most commonly appear in network programming, particularly socket programming. Sometimes, though, port numbers are made visible to the casual user. For example, some Web sites a person visits on the Internet use a URL like the following:

<http://www.mairie-metz.fr:8080/> In this example, the number 8080 refers to the port number used by the Web browser to connect to the Web server. Normally, a Web site uses port number 80 and this number need not be included with the URL (although it can be).

In IP networking, port numbers can theoretically range from 0 to 65535. Most popular network applications, though, use port numbers at the low end of the range (such as 80 for HTTP).

Note: The term port also refers to several other aspects of network technology. A port can refer to a physical connection point for peripheral devices such as serial, parallel, and USB ports. The term port also refers to certain Ethernet connection points, such as those on a hub, switch, or router.

ref

http://compnetworking.about.com/od/basicnetworkingconcepts/l/bldef_port.htm

ref

http://compnetworking.about.com/od/itinformationtechnology/bl/def_socket.htm

Share Improve this answer

Follow

edited Oct 27, 2023 at 21:13



John Smith

7,399 ● 7 ● 51 ● 63

answered Sep 30, 2008 at 10:07



Galwegian

42.2k ● 16 ● 113 ● 158

2 Layer 2 on the OSI model is a connection between nodes, it has no mechanism of connecting processes. I don't believe you can consider a socket existing at OSI l2. – [Antonio Haley](#) Sep 30, 2008 at 12:45

28 A circuit is a connection - a socket is an endpoint. A connection consists of 2 sockets. – [Mark Brackett](#) Sep 30, 2008 at 14:33

6 "A socket represents a single connection between two network applications." That does not match [RFC 793, Transmission Control Protocol](#) that explains: "To allow for many processes within a single Host to use TCP communication facilities simultaneously, the TCP provides a set of addresses or ports within each host. Concatenated with the network and host addresses from the internet communication layer, this forms a socket. A pair of sockets uniquely identifies each connection." – [Ron Maupin](#) Nov 17, 2019 at 18:37



With some analogy

122



Although a lot technical stuff is already given above for **sockets**... I would like to add my answer, just in case , **if somebody still could not feel the difference between ip, port and sockets**

Consider a server S,

and say **person X,Y,Z** need a service (say chat service) from that **server S**

then

IP address tells --> *who?* *is that chat server 'S' that X,Y,Z want to contact*

okay, you got "who is the server"

but suppose that server 'S' is providing some other services to other people as well,say '*S provides storage services to person A,B,C*

then

port tells ---> *which?* service you (X,Y,Z) need i.e. chat service and not that storage service

okay., you make server to come to know that 'chat service' is what you want and not the storage

but

you are three and the server *might want to identify all the three differently*

there comes the **socket**

now **socket tells--> *which one? particular connection***

that is , say ,

socket 1 for person X

socket 2 for person Y

and socket 3 for person Z

Share Improve this answer

Follow

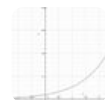
edited Dec 25, 2022 at 22:13



starball

48.3k ● 28 ● 187 ● 847

answered Jun 20, 2016 at 9:57



eRaisedToX

3,341 ● 2 ● 24 ● 30

-
- 4 So X,Y,Z would connect to the same port, i.e. same service, but have different sockets on the server side? So when, say, X sends some packet to the server, it's going to say: 'find me the (protocol, X's IP, X's port, S's IP, S's port) socket' and send to the chat app. I assume there must be a binding between some application -specific objects and socket objects? For example, when i get some data from socket-1, i want to display that as a user message, but the app needs to know messages from socket A are from User-X. – [monolith](#)
Nov 7, 2019 at 20:14
-



Firsty, I think we should start with a little understanding of what constitutes getting a packet from A to B.



A common definition for a network is the use of the [OSI Model](#) which separates a network out into a number of layers according to purpose. There are a few important ones, which we'll cover here:

- The *data link layer*. This layer is responsible for getting packets of data from one network device to another and is just above the layer that actually does the transmitting. It talks about MAC addresses and knows how to find hosts based on their MAC (hardware) address, but nothing more.
- The *network layer* is the layer that allows you to transport data across machines and over physical boundaries, such as physical devices. The network layer must essentially support an additional address based mechanism which relates somehow to the physical address; enter the Internet Protocol (IPv4). An IP address can get your packet from A to B over the internet, but knows nothing about how to traverse individual hops. This is handled by the layer above in accordance with routing information.
- The *transport layer*. This layer is responsible for defining the way information gets from A to B and any restrictions, checks or errors on that behaviour. For example, TCP adds additional information to a packet such that it is possible to deduce if packets have been lost.

TCP contains, amongst other things, the concept of [ports](#). These are effectively different data endpoints on the

same IP address to which an Internet Socket (`AF_INET`) can bind.

As it happens, [so too does UDP](#), and other transport layer protocols. They don't technically *need* to feature ports, but these ports do provide a way for multiple applications in the layers above to use the same computer to receive (and indeed make) outgoing connections.

Which brings us to the anatomy of a TCP or UDP connection. Each features a source port and address, and a target port and address. This is so that in any given session, the target application can respond, as well as receive, from the source.

So ports are essentially a specification-mandated way of allowing multiple concurrent connections sharing the same address.

Now, we need to take a look at how you communicate from an application point of view to the outside world. To do this, you need to kindly ask your operating system and since most OSes support the Berkeley Sockets way of doing things, we see we can create sockets involving ports from an application like this:

```
int fd = socket(AF_INET, SOCK_STREAM, 0); // tcp
socket
int fd = socket(AF_INET, SOCK_DGRAM, 0); // udp
socket
// later we bind...
```

Great! So in the `sockaddr` structures, we'll specify our port and bam! Job done! Well, almost, except:

```
int fd = socket(AF_UNIX, SOCK_STREAM, 0);
```

is also possible. Urgh, that's thrown a spanner in the works!

Ok, well actually it hasn't. All we need to do is come up with some appropriate definitions:

- An internet socket is the combination of an IP address, a protocol and its associated port number on which a service may provide data. So tcp port 80, stackoverflow.com is an internet socket.
- An unix socket is an IPC endpoint represented in the file system, e.g. `/var/run/database.sock`.
- A socket API is a method of requesting an application be able to read and write data to a socket.

Voila! That tidies things up. So in our scheme then,

- A port is a numeric identifier which, as part of a transport layer protocol, identifies the service number which should respond to the given request.

So really a port is a subset of the requirements for forming an internet socket. Unfortunately, it just so happens that the meaning of the word socket has been applied to several different ideas. So I heartily advise you

name your next project socket, just to add to the confusion ;)

Share Improve this answer

answered Apr 7, 2012 at 20:17

Follow



user257111

This is why bullets don't leave and won't leave Powerpoint; they work! – [Anurag Kalia](#) Mar 1, 2013 at 18:47

Very nice introduction to tcp-ip and network communication. Beginners, read this first. – [Colin](#) Mar 14, 2013 at 18:43



43

A socket = IP Address + a port (numeric address)
Together they identify an end-point for a network connection on a machine. (Did I just flunk network 101?)



Share Improve this answer

answered Sep 30, 2008 at 10:05

Follow



Gishu

137k ● 47 ● 226 ● 311



8 I believe port has broader meaning than your definition.
– [Richard Dorman](#) Sep 30, 2008 at 10:09

5 And sockets are not only subject to the TCP/IP stack. See UNIX domain sockets or inter process communication sockets in general. – [matthias krull](#) May 1, 2012 at 23:04

not sure about this answer. You can use HTTP to communicate with another process via sockets without assigning a port. – [SeF](#) Apr 13, 2020 at 7:48



43



Generally, you will get a lot of theoretical but one of the easiest ways to differentiate these two concepts is as follows:

In order to get a service, you need a service number. This service number is called a port. Simple as that.



For example, the HTTP as a service is running on port 80.

Now, many people can request the service, and a connection from client-server gets established. There will be a lot of connections. Each connection represents a client. In order to maintain each connection, the server creates a socket per connection to maintain its client.

Share Improve this answer

Follow

edited Feb 14, 2022 at 0:34



[cheznead](#)

2,779 ● 7 ● 31 ● 55

answered Dec 13, 2013 at 10:09



[kta](#)

20.1k ● 7 ● 67 ● 48

Does each socket require it's own port? – [MondayPaper](#) Jul 2, 2014 at 14:45

5 I am not sure if your statement: "the server creates socket per connection to maintain it's client" is correct.

– [Rushi Agrawal](#) Jun 6, 2015 at 15:08

1 @RushiAgrawal Then I suggest you look it up. Specifically, see *man accept*. – [user207421](#) Aug 31, 2015 at 4:02

1 This implies that for each socket which the server creates per connection to maintain its client can have the same port number (such as port 80 for HTTP connections continuation) but with different IP address of the clients which the requests for connections are sent from. right? – [Randika Vishman](#) Sep 5, 2015 at 5:57

2 The server creates a socket *instance* per connection. The problem here is the English language which is ambiguous with classes and instances. – [Peter Wone](#) Jan 24, 2017 at 20:51



37

These are basic networking concepts so I will explain them in an easy yet a comprehensive way to understand in details.



- **A socket** is like a telephone (i.e. end to end device for communication)



- **IP** is like your telephone number (i.e. address for your socket)



- **Port** is like the person you want to talk to (i.e. the service you want to order from that address)
- A socket can be a client or a server socket (i.e. in a company the telephone of the customer support is a server but a telephone in your home is mostly a client)

So a socket in networking is a virtual communication device bound to a pair (ip , port) = (address , service).

Note:

- A machine, a computer, a host, a mobile, or a PC can have multiple addresses , multiple open ports, and thus multiple sockets. Like in an office you can have multiple telephones with multiple telephone numbers and multiple people to talk to.
- Existence of an open/active port necessitate that you must have a socket bound to it, because it is the socket that makes the port accessible. However, you may have unused ports for the time being.
- Also note, in a server socket you can bind it to (a port, a specific address of a machine) or to (a port, all addresses of a machine) as in the telephone you may connect many telephone lines (telephone numbers) to a telephone or one specific telephone line to a telephone and still you can reach a person through all these telephone lines or through a specific telephone line.
- You can not associate (bind) a socket with two ports as in the telephone usually you can not always have two people using the same telephone at the same time .
- Advanced: on the same machine you cannot have two sockets with same type (client, or server) and same port and ip. However, if you are a client you can open two connections, with two sockets, to a server because the local port in each of these client's sockets is different)

Hope it clears you doubts

Share Improve this answer

edited May 24, 2019 at 22:50

Follow

answered Feb 19, 2019 at 7:17



Mosab Shaheen

1,174 ● 11 ● 29

1 It is interesting to see all these understandings and analogies of sockets/ports/ip addresses under this question. And I like this answer. – [Bloodmoon](#) Apr 25, 2019 at 8:48

1 Wow! What a good explanation and examples. +1 for this. – [Peter](#) Jul 3, 2020 at 10:14



34



There seems to be a lot of answers equating socket with the connection between 2 PC's..which I think is absolutely incorrect. A socket has always been the *endpoint* on 1 PC, that may or may not be connected - surely we've all used listener or UDP sockets* at some point. The important part is that it's addressable and active. Sending a message to 1.1.1.1:1234 is not likely to work, as there is no socket defined for that endpoint.

Sockets are protocol specific - so the implementation of uniqueness that both [TCP/IP](#) and [UDP/IP](#) uses* (ipaddress:port), is different than eg., [IPX](#) (Network, Node, and...ahem, socket - but a different socket than is meant by the general "socket" term. IPX socket numbers

are equivalent to IP ports). But, they all offer a unique addressable endpoint.

Since IP has become the dominant protocol, a port (in networking terms) has become synonymous with either a UDP or TCP port number - which is a portion of the socket address.

- UDP is connection-less - meaning no virtual circuit between the 2 endpoints is ever created. However, we still refer to [UDP sockets](#) as the endpoint. The API functions make it clear that both are just different type of sockets - `SOCK_DGRAM` is UDP (just sending a message) and `SOCK_STREAM` is TCP (creating a virtual circuit).
- Technically, the IP header holds the IP Address, and the protocol on top of IP (UDP or TCP) holds the port number. This makes it possible to have other protocols (eg. [ICMP](#) that have no port numbers, but do have IP addressing information).

Share Improve this answer

answered Sep 30, 2008 at 13:26

Follow



Mark Brackett

85.6k ● 17 ● 111 ● 155

Good answer for socket. Port indeed refers to TCP or UDP, which, I want to stress, not necessarily is used on top of IP.

– [Jack](#) Jun 17, 2021 at 9:02

Short brief answer.



30



A **port** can be described as an **internal address** within a host that identifies a program or process.

A **socket** can be described as a **programming interface** allowing a program to communicate with other programs or processes, on the internet, or locally.

Share Improve this answer

edited Jul 27, 2014 at 14:55

Follow



david

33.5k ● 12 ● 67 ● 88

answered Mar 19, 2014 at 1:31



Andy

2,497 ● 1 ● 26 ● 27

6 The word 'internal' in the port description sounds rather like 'non-public' to me. – [Jonas N](#) Jan 21, 2016 at 15:25

So Could we say :Sockets runs inside Ports ? or Ports runs inside Sockets ? – [Gucho Ca](#) Jun 6, 2016 at 19:52

1 @GuchoCa We can't say that either sockets or ports run at all, let alone one inside the other. Unclear what you're asking. – [user207421](#) Nov 29, 2016 at 0:32



24



They are terms from two different domains: 'port' is a concept from TCP/IP networking, 'socket' is an API (programming) thing. A 'socket' is made (in code) by taking a port and a hostname or network adapter and combining them into a data structure that you can use to send or receive data.



Share Improve this answer

answered Sep 30, 2008 at 10:08

Follow



Roel

19.6k ● 7 ● 64 ● 97

-
- 1 For the most general answer, strike "made by taking a port and a hostname or network adapter and combining them into a." For example, a UNIX socket is (in code) a data structure (or object) that you can use to send or receive data.
– [Josiah Yoder](#) Sep 22, 2016 at 16:51
-



17

After reading the excellent up-voted answers, I found that the following point needed emphasis for me, a newcomer to network programming:



TCP-IP connections are bi-directional pathways connecting one address:port combination with another address:port combination. Therefore, whenever you open a connection from your local machine to a port on a remote server (say `www.google.com:80`), you are also associating a new port number on your machine with the connection, to allow the server to send things back to you, (e.g. `127.0.0.1:65234`). It can be helpful to use `netstat` to look at your machine's connections:

```
> netstat -nWp tcp (on OS X)
Active Internet connections
Proto Recv-Q Send-Q Local Address
Foreign Address      (state)
tcp4          0      0 192.168.0.6.49871
17.172.232.57.5223    ESTABLISHED
...
```

answered Mar 14, 2013 at 19:57



Colin

2,099 ● 25 ● 36



14



A socket is a communication endpoint. A socket is not directly related to the TCP/IP protocol family, it can be used with any protocol your system supports. The C socket API expects you to first get a blank socket object from the system that you can then either bind to a local socket address (to directly retrieve incoming traffic for connection-less protocols or to accept incoming connection requests for connection-oriented protocols) or that you can connect to a remote socket address (for either kind of protocol). You can even do both if you want to control both, the local socket address a socket is bound to and the remote socket address a socket is connected to. For connection-less protocols connecting a socket is even optional but if you don't do that, you'll have to also pass the destination address with every packet you want to send over the socket as how else would the socket know where to send this data to? Advantage is that you can use a single socket to send packets to different socket addresses. Once you have your socket configured and maybe even connected, consider it to be a bi-directional communication pipe. You can use it to pass data to some destination and some destination can use it to pass data back to you. What you write to a

socket is send out and what has been received is available for reading.

Ports on the other hand are something that only certain protocols of the TCP/IP protocol stack have. TCP and UDP packets have ports. A port is just a simple number. The combination of source port and destination port identify a communication channel between two hosts. E.g. you may have a server that shall be both, a simple HTTP server and a simple FTP server. If now a packet arrives for the address of that server, how would it know if that is a packet for the HTTP or the FTP server? Well, it will know so as the HTTP server will run on port 80 and the FTP server on port 21, so if the packet arrives with a destination port 80, it is for the HTTP server and not for the FTP server. Also the packet has a source port since without such a source port, a server could only have one connection to one IP address at a time. The source port makes it possible for a server to distinguish otherwise identical connections: they all have the same destination port, e.g. port 80, the same destination IP (the IP of the server), and the same source IP, as they all come from the same client, but as they have different source ports, the server can distinguish them from each other. And when the server sends back replies, it will do so to the port the request came from, that way the client can also distinguish different replies it receives from the same server.

[Share](#) [Improve this answer](#)

[edited Jun 17, 2021 at 13:25](#)

[Follow](#)

answered Sep 30, 2008 at 10:24



Mecki

133k ● 34 ● 259 ● 269

-
- 5 This is incorrect. A socket is not an endpoint. A socket is defined by two endpoints. Each endpoint is defined by a network address and a port. The purpose of ports is to differentiate multiple endpoints on the same network address, so that multiple concurrent sockets can be supported.
– [Peter Wone](#) Sep 30, 2008 at 11:18

-
- 1 I notice that RFC793 (original TCP spec) does refer to the combination of a network address and a port as a socket, so I can see where you got this, but it's still incorrect inasmuch as a socket is necessarily defined by two endpoints.
– [Peter Wone](#) Sep 30, 2008 at 11:28

-
- 3 On reflection the literature is contradictory and I apologise. Very strictly speaking communication does not occur until a TCP connection is established between two endpoints (aka sockets) each of which is identified by a network address and a port. I give up. – [Peter Wone](#) Sep 30, 2008 at 12:29

-
- 1 @PeterWone I believe you can't define a socket by two endpoints: what about a server socket waiting for an incoming connection? It is alone, and still it's a socket. And you can't even define a socket related to network.. you may have sockets over files. Yes, network address + port is a socket, but I intend a socket as a superset. – [Jack](#) Jun 17, 2021 at 9:00

@Jack yes in this comment I have used the term socket incorrectly, and so have you, as your own example of a listening socket demonstrates. My own answer above discusses at length and with references the correct nomenclature and exactly what it means. What we have here in the comments on this question called a socket is a



13



A socket is a special type of file handle which is used by a process to request network services from the operating system. A socket address is the triple: {protocol, local-address, local-process} where the local process is identified by a port number.



In the TCP/IP suite, for example:



{tcp, 193.44.234.3, 12345}

A conversation is the communication link between two processes thus depicting an association between two. An association is the 5-tuple that completely specifies the two processes that comprise a connection: {protocol, local-address, local-process, foreign-address, foreign-process}

In the TCP/IP suite, for example:

{tcp, 193.44.234.3, 1500, 193.44.234.5, 21}

could be a valid association.

A half-association is either: {protocol, local-address, local-process}

or

{protocol, foreign-address, foreign-process}

which specify each half of a connection.

The half-association is also called a socket or a transport address. That is, a socket is an end point for communication that can be named and addressed in a network. The socket interface is one of several application programming interfaces (APIs) to the communication protocols. Designed to be a generic communication programming interface, it was first introduced by the 4.2BSD UNIX system. Although it has not been standardized, it has become a de facto industry standard.

Share Improve this answer

edited May 20, 2015 at 16:53

Follow

answered Jun 9, 2014 at 12:18



Krishna

5,634 ● 2 ● 29 ● 33

This answer is the one that did it for me. I guess it is because no one else mentioned the word association. Good explanation. – [rationalcoder](#) Dec 16, 2015 at 5:57

There is no process number in any of your examples. The word you are looking for is 'port'. – [user207421](#) Jun 28, 2016 at 18:16

Read the first para.. It's mentioned clearly there. Let me know of any ambiguity by quoting the exact phrase.. Would be helpful for me to improvise. – [Krishna](#) Nov 3, 2018 at 10:09

I read it. The correct formulation would be "A socket address is the triple: {protocol, local-address, local-port-number}". A

process can own multiple ports, which makes your formulation invalid. – [user207421](#) Mar 29, 2022 at 9:06



13



A socket address is an IP address & port number

| | |
|----------------------|------------------|
| 123.132.213.231 | # IP address |
| :1234 | # port number |
| 123.132.213.231:1234 | # socket address |

A connection occurs when 2 sockets are bound together.

Share Improve this answer

Follow

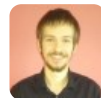
edited Jun 20, 2020 at 9:12



Community Bot

1 • 1

answered Jan 24, 2017 at 17:04



Zaz

48.6k • 15 • 88 • 105

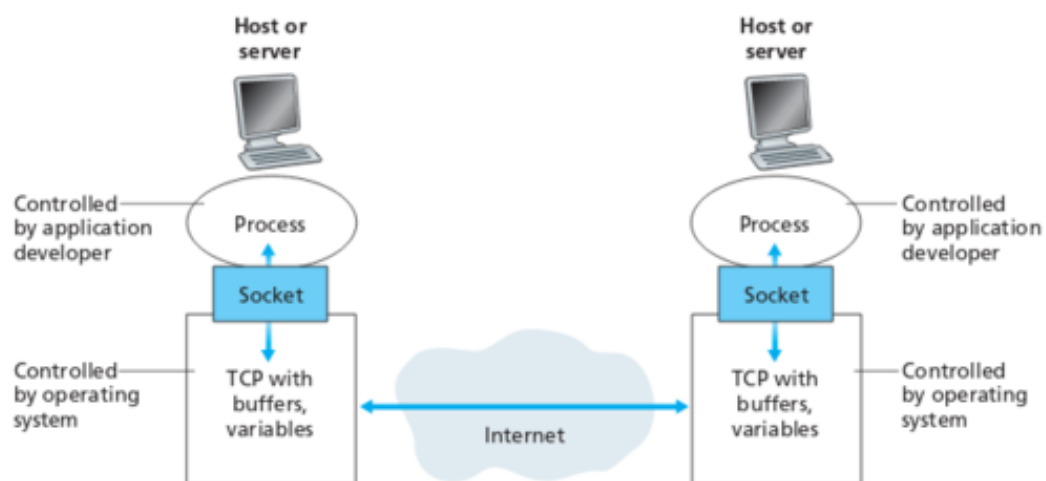
There is no such thing as binding two sockets together. The word 'bound' means something else with ports. – [user207421](#) Apr 24, 2019 at 5:44

This is wrong, a socket does not need an IP address nor a port number, only some network sockets need them. Neither do Unix domain sockets nor socketCAN sockets need them. – [12431234123412341234123](#) Jun 28, 2021 at 16:36

11

An application consists of pair of processes which communicate over the network (client-server pair). These processes send and receive messages, into and from the network through a software interface called **socket**.

Considering the analogy presented in the book "Computer Networking: Top Down Approach". There is a house that wants to communicate with other house. Here, house is analogous to a process, and door to a socket. Sending process assumes that there is a infrastructure on the other side of the door that will transport the data to the destination. Once the message is arrived on the other side, it passes through receiver's door (socket) into the house (process). This illustration from the same book can help you:



Sockets are part of transport layer, which provides logical communication to applications. This means that from application's point of view both hosts are directly connected to each other, even though there are numerous routers and/or switches between them. Thus a socket is not a connection itself, it's the end point of the connection. Transport layer protocols are implemented only on hosts, and not on intermediate routers.

Ports provide means of internal addressing to a machine. The primary purpose is to allow multiple processes to send and receive data over the network without interfering with other processes (their data). All sockets are provided with a port number. When a segment arrives to a host, the transport layer examines the destination port number of the segment. It then forwards the segment to the corresponding socket. This job of delivering the data in a transport layer segment to the correct socket is called **de-multiplexing**. The segment's data is then forwarded to the process attached to the socket.

Share Improve this answer

edited Oct 4, 2016 at 8:26

Follow

answered Oct 4, 2016 at 8:14



Ugnés

769 ● 1 ● 10 ● 24



9



The port was the easiest part, it is just a unique identifier for a socket. A socket is something processes can use to establish connections and to communicate with each other. Tall Jeff had a great telephone analogy which was not perfect, so I decided to fix it:



- ip and port ~ phone number
- socket ~ phone device
- connection ~ phone call
- establishing connection ~ calling a number

- processes, remote applications ~ people
- messages ~ speech

Share Improve this answer

edited Oct 11, 2015 at 23:19

Follow

answered Oct 11, 2015 at 22:57



inf3rno

26.1k ● 12 ● 119 ● 205

Good clarification (especially when you consider the telephone switching history that's part of the foundation of networking terminology..) – [o3wæi](#) Apr 3, 2016 at 14:11

Have a look at a `netstat` display some time. All sockets accepted from a listening socket share the same port. *Ergo* a port is not a unique identifier for a socket. – [user207421](#) Jan 29, 2017 at 1:19



7



A socket is a structure in your software. It's more-or-less a file; it has operations like read and write. It isn't a physical thing; it's a way for your software to refer to physical things.

A port is a device-like thing. Each host has one or more networks (those are physical); a host has an address on each network. Each address can have thousands of ports.

One socket only may be using a port at an address. The socket allocates the port approximately like allocating a

device for file system I/O. Once the port is allocated, no other socket can connect to that port. The port will be freed when the socket is closed.

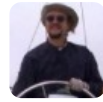
Take a look at [TCP/IP Terminology](#).

Share Improve this answer

edited Dec 31, 2008 at 11:10

Follow

answered Sep 30, 2008 at 10:09



[S.Lott](#)

391k ● 82 ● 517 ● 788

-
- 3 This description of socket is pretty off base. A socket is about the connection between a pair of tuples where a tuple refers to an IP ADDR & Port pair. Additionally many sockets CAN connect to the same port. How do you think a web server takes multiple connections on port 80? This is a poor answer – [Tall Jeff](#) Sep 30, 2008 at 12:15
-
- 2 Sorry. Multiple sockets are not connected to port 80. One socket is connected and spawns additional sockets where the real transfer happens. See opengroup.org/onlinepubs/009695399/functions/listen.html. – [S.Lott](#) Sep 30, 2008 at 12:39
-
- 1 Actually, the description at opengroup.org/onlinepubs/009695399/functions/connect.html is better. The peer socket returned by a connection is NOT on port 80. – [S.Lott](#) Sep 30, 2008 at 12:41
-
- 1 This post is incorrect in several particulars and misleading in several respects. – [Peter Wone](#) Oct 2, 2008 at 2:12
-
- 1 @Peter Wone: Which particulars? Which aspects? Hoping to learn from my mistakes. – [S.Lott](#) Oct 2, 2008 at 10:01
-



7



from [Oracle Java Tutorial](#):

A socket is one endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent to.

Follow



[user207421](#)

311k ● 44 ● 320 ● 488

answered Apr 12, 2014 at 14:03



[yondoo](#)

304 ● 1 ● 7 ● 8

That's only a tutorial, and certainly not a normative reference.

– [user207421](#) Aug 31, 2015 at 4:04

"A socket is one endpoint of a two-way communication link"

Is that not a socket definition, Not a java tutorial??

– [prayagupadhyay](#) Jun 7, 2017 at 6:03

@prayagupd Of course it's a definition, but it's from a tutorial, not a specification. – [user207421](#) Jun 23, 2017 at 20:08 ✎



7



Port and socket can be compared to the Bank Branch.



The building number of the "Bank" is analogous to IP address. A bank has got different sections like:

1. Savings account department
2. Personal loan department
3. Home loan department
4. Grievance department

So 1 (savings account department), 2 (personal loan department), 3 (home loan department) and 4 (grievance

department) are ports.

Now let us say you go to open a savings account, you go to the bank (IP address), then you go to "savings account department" (port number 1), then you meet one of the employees working under "savings account department". Let us call him SAVINGACCOUNT_EMPLOYEE1 for opening account.

SAVINGACCOUNT_EMPLOYEE1 is your socket descriptor, so there may be SAVINGACCOUNT_EMPLOYEE1 to SAVINGACCOUNT_EMPLOYEEEN. These are all socket descriptors.

Likewise, other departments will be having employees working under them and they are analogous to socket.

Share Improve this answer

Follow

edited Dec 19, 2018 at 10:16



Pang

10.1k ● 146 ● 85 ● 124

answered Apr 11, 2017 at 12:21



Shridhar410

107 ● 1 ● 6



5



A socket is a data I/O mechanism. A port is a *contractual* concept of a *communication protocol*. A socket can exist without a port. A port can exist without a specific socket (e.g. if several sockets are active on the same port, which may be allowed for some protocols).



A port is used to determine which socket the receiver should route the packet to, with many protocols, but it is not always required and the receiving socket selection can be done by other means - a port is entirely a tool used by the protocol handler in the network subsystem. e.g. if a protocol does not use a port, packets can go to all listening sockets or any socket.

Share Improve this answer

answered Sep 30, 2008 at 10:08

Follow



Sander

26.3k ● 3 ● 54 ● 88



5



Port:

A port can refer to a physical connection point for peripheral devices such as serial, parallel, and USB ports. The term port also refers to certain Ethernet connection points, such as those on a hub, switch, or router.



Socket:

A socket represents a single connection between two network applications. These two applications nominally run on different computers, but sockets can also be used for interprocess communication on a single computer. Applications can create multiple sockets for communicating with each other. Sockets are bidirectional, meaning that either side of the connection is capable of both sending and receiving data.

Share Improve this answer

answered Sep 30, 2008 at 10:23

Follow



[balaweblog](#)

15.4k ● 28 ● 76 ● 95

A TCP or UDP port does not refer to anything physical, or to Ethernet connection points either. You haven't answered the question. – [user207421](#) Jun 23, 2017 at 20:11

@user207421 I don't need anything about TCP nor UDP in the question. – [12431234123412341234123](#) Jun 28, 2021 at 16:43

What you don't need is anything about 'physical connection point'. This is not correct. A port is a logical entity that doesn't refer to anything physical at all. – [user207421](#) Mar 29, 2022 at 9:10



Relative TCP/IP terminology which is what I assume is implied by the question. In layman's terms:

5



A PORT is like the telephone number of a particular house in a particular zip code. The ZIP code of the town could be thought of as the IP address of the town and all the houses in that town.



A SOCKET on the other hand is more like an established phone call between telephones of a pair of houses talking to each other. Those calls can be established between houses in the same town or two houses in different towns. It's that temporary established pathway between the pair of phones talking to each other that is the SOCKET.

answered Sep 30, 2008 at 12:12

**Tall Jeff**

9,984 ● 7 ● 46 ● 61

-
- 3 A socket is an endpoint. It exists before a connection is established (TCP), or in the absence of a connection (UDP). Ergo it is not itself the connection. – [user207421](#) Jun 28, 2016 at 18:19
-



4



In a broad sense, Socket - is just that, a socket, just like your electrical, cable or telephone socket. A point where "requisite stuff" (power, signal, information) can go out and come in from. It hides a lot of detailed stuff, which is not required for the use of the "requisite stuff". In software parlance, it provides a generic way of defining a mechanism of communication between two entities (those entities could be anything - two applications, two physically separate devices, User & Kernel space within an OS, etc)

A Port is an endpoint discriminator. It differentiates one endpoint from another. At networking level, it differentiates one application from another, so that the networking stack can pass on information to the appropriate application.

Follow



Harty

316 ● 3 ● 12



4



Socket is an abstraction provided by kernel to user applications for data I/O. A socket type is defined by the protocol it's handling, an IPC communication etc. So if somebody creates a TCP socket he can do manipulations like reading data to socket and writing data to it by simple methods and the lower level protocol handling like TCP conversions and forwarding packets to lower level network protocols is done by the particular socket implementation in the kernel. The advantage is that user need not worry about handling protocol specific nitigrities and should just read and write data to socket like a normal buffer. Same is true in case of IPC, user just reads and writes data to socket and kernel handles all lower level details based on the type of socket created.

Port together with IP is like providing an address to the socket, though its not necessary, but it helps in network communications.

Share Improve this answer

Follow

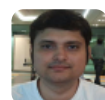
edited Aug 3, 2021 at 10:59



rofrol

15.2k ● 8 ● 86 ● 83

answered Oct 16, 2013 at 10:08



Navneet Singh

41 ● 1



3



A port denotes a communication endpoint in the TCP and UDP transports for the IP network protocol. A socket is a software abstraction for a communication endpoint commonly used in implementations of these protocols (socket API). An alternative implementation is the XTI/TLI API.

See also:

Stevens, W. R. 1998, UNIX Network Programming: Networking APIs: Sockets and XTI; Volume 1, Prentice Hall.

Stevens, W. R., 1994, TCP/IP Illustrated, Volume 1: The Protocols, Addison-Wesley.

Share Improve this answer

answered Sep 30, 2008 at 10:28

Follow



VoidPointer

17.8k ● 15 ● 58 ● 58



3



Socket is SW abstraction of networking endpoint, used as the interface to the application. In Java, C# it is represented by object, in Linux, Unix it is a file.

Port is just a property of a socket you have specify if you want to establish a communication. To receive packet from a socket you have to bind it to specific local port and NIC (with local IP address) or all NICs (INADDR_ANY is specified in the bind call). To send packet, you have to specify port and IP of the remote socket.

answered Oct 22, 2018 at 14:59



Dražen Grašovec

792 ● 10 ● 18



2



Already theoretical answers have been given to this question. I would like to give a practical example to this question, which will clear your understanding about Socket and Port.

I found it [here](#)



This example will walk you thru the process of connecting to a website, such as Wiley. You would open your web browser (like Mozilla Firefox) and type `www.wiley.com` into the address bar. Your web browser uses a Domain Name System (DNS) server to look up the name `www.wiley.com` to identify its IP address is. For this example, the address is `192.0.2.100`.

Firefox makes a connection to the `192.0.2.100` address and to the port where the application layer web server is operating. Firefox knows what port to expect because it is a well-known port . The well-known port for a web server is TCP port 80.

The destination socket that Firefox attempts to connect is written as socket:port, or in this example, 192.0.2.100:80. This is the server side of the connect, but the server needs to know where to send the web page you want to view in Mozilla Firefox, so you have a socket for the client side of the connection also.

The client side connection is made up of your IP address, such as 192.168.1.25, and a randomly chosen dynamic port number. The socket associated with Firefox looks like 192.168.1.25:49175. Because web servers operate on TCP port 80, both of these sockets are TCP sockets, whereas if you were connecting to a server operating on a UDP port, both the server and client sockets would be UDP sockets.

Share Improve this answer

answered Jun 26, 2013 at 12:57

Follow



Omkar Ramtekkar

31 ● 2

Very poor quality citation. The third paragraph misuses the word 'socket' as though it meant 'IP address'. It doesn't.

– [user207421](#) Jun 28, 2016 at 18:24



A single port can have one or more sockets connected with different external IP's like a multiple electrical outlet.

2



```
TCP      192.168.100.2:9001
155.94.246.179:39255  ESTABLISHED      1312
TCP      192.168.100.2:9001      171.25.193.9:61832
ESTABLISHED      1312
TCP      192.168.100.2:9001
178.62.199.226:37912  ESTABLISHED      1312
TCP      192.168.100.2:9001
188.193.64.150:40900  ESTABLISHED      1312
TCP      192.168.100.2:9001
198.23.194.149:43970  ESTABLISHED      1312
TCP      192.168.100.2:9001      198.49.73.11:38842
ESTABLISHED      1312
```

Share Improve this answer

answered Aug 31, 2015 at 3:43

Follow



guest

46 ● 2



2

Uff.. too many people linking socket concept to a two-endpoints communication, mostly on TCP/IP protocol.

But:



- **NO** - Socket is not related to a two-endpoint communication. It's the local endpoint, which can or cannot be connected on the other side (Think about a server socket listening for incoming connection)
- **NO** - Socket it's not strictly related to TCP/IP. It is defined with a protocol, which can be TCP/IP, but can be anything else. For example you can have socket that communicates over files. You can also implement a new protocol yourself to have a communication over USB lamp which sends data by flashing: that would still be a socket from the application point of view.

Regarding the port concept it's correct what you read on other answers. Port is mostly intended to be the number value (2 bytes, 0-65535) in TCP or UDP packet. Just let me stress that TCP or UDP not necessarily are used on top of IP. So:

- **NO** - it's not correct saying that port is part of TCP/IP or UDP/IP. It's part of TCP or UDP, or any other protocol which define and use it. IP has no knowledge of what a port is.

Share Improve this answer

answered Jun 17, 2021 at 9:13

Follow



Jack

1,596 ● 1 ● 13 ● 24

1

2

Next



Highly active question. Earn 10 reputation (not counting the [association bonus](#)) in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.