curl user survey 2019 analysis

“there are so many useful features”

summary and analysis by Daniel Stenberg

version 1 - June 4, 2019

Table of Contents

About curl

Curl is an old and established open source project that produces the curl tool and the libcurl library. We are a small project, with few maintainers, with little commercial backing and yet we're over 21 years old and we have gathered help from almost 2,000 contributors through the years. Our products run in a vast amount of Internet connected devices, tools and services. *curl is one of the world's most widely used software components. Possibly even **the** most widely used component!*

See https://curl.haxx.se for everything not answered in this summary.

Survey Background

We do a user survey annually in an attempt to catch trends, view and longer running changes in the project, its users and in how curl fits into the wider ecosystem. As usual, we only reach and get responses from a small subset of users who voluntarily decide to fill in the questionnaire while the vast majority of users and curl developers never get to hear about it and never get an opportunity to respond. Self-selected respondents to a survey makes the results hard to interpret and judge.

This should make us ask ourselves: is this what our users think, or is it just the opinions of the subset of users that we happened to reach. We simply have to work with what we have.

This year, the survey was up 14 days from May 13 to and including May 26. This year was the 6th annual survey as the first one ran in 2014.

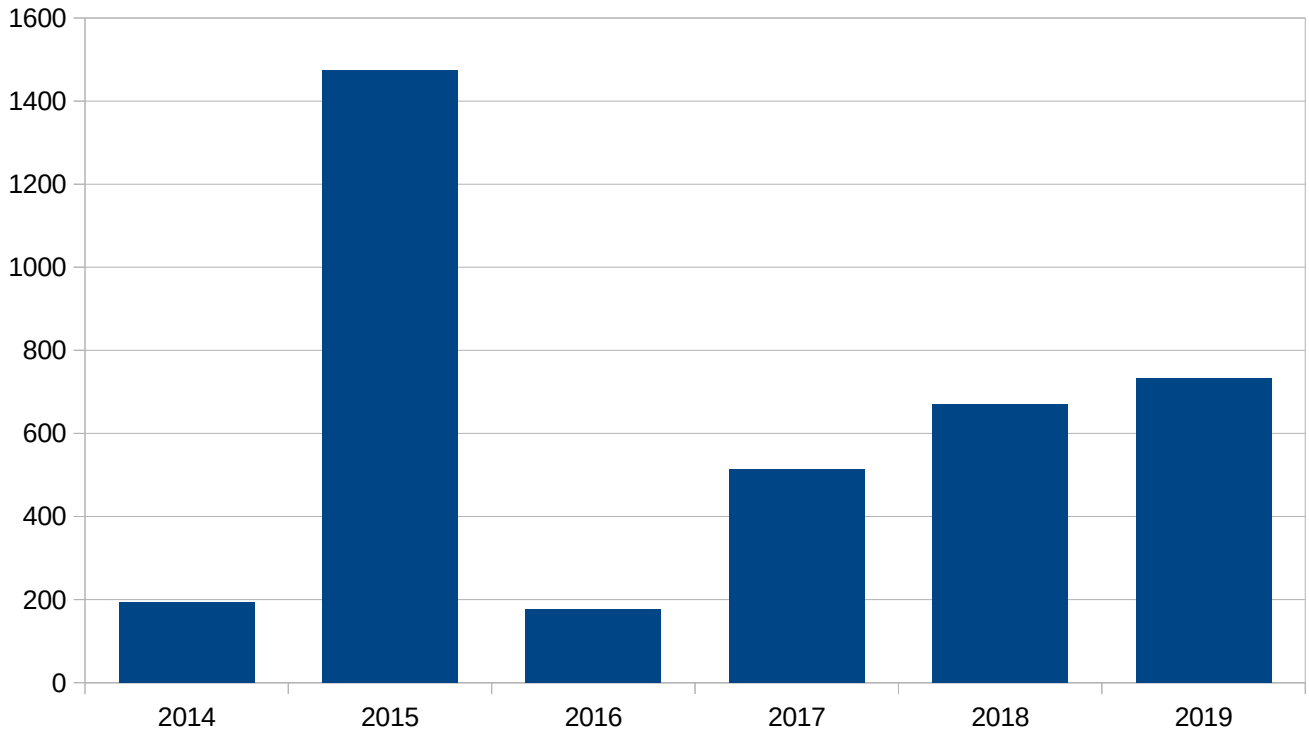
The survey was announced on the curl-users and curl-library mailing lists (with two reminders), numerous times on Daniel's twitter feed (@bagder) and on Daniel's blog (https://daniel.haxx.se/blog). The survey was also announced on the curl web site with an "alert style" banner on most pages on the site that made it hard to miss for web visitors.

We used a service run by Google to perform the survey, which typically also leads to us loosing a small share of users who refuse to use services hosted by them. We feel compelled to go with simplicity, no cost and convenience of the service rather than trying to please everyone.

Number of responses

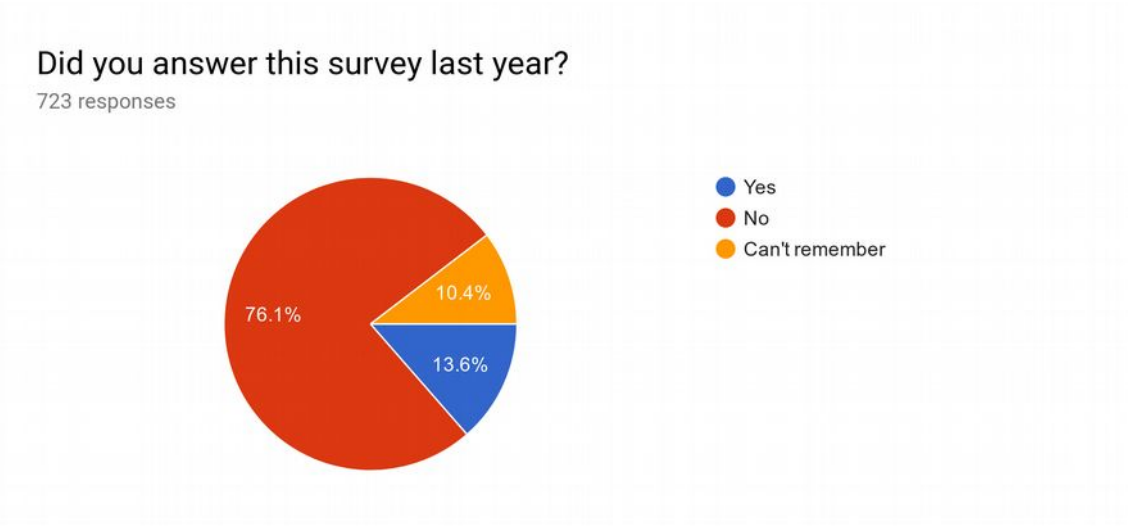
This year showed a 9.2% increase in number of survey participants compared to last year, climbing from 670 to 732 responses. Still far off from the record year 2015 when a total of 1475 entries were collected.

Of course, a single respondent to this survey can be the author of an application that uses libcurl and is used by a billion users. Or the respondent is the single user of curl in his basement. We just can't tell or know. Also, there's no way for us to compare or judge "importance" between users. We're all equals here!



Returning respondents?

To better understand the answers and the population of users who answered this year's survey, we again asked if the users who filled this in also did it last year. The answers showed us that the vast majority didn't answer last year (76.1%) or couldn't remember (10.4%), and less than every 7th respondent know they answered last year.



This result was also shockingly similar to the 2018 results which ended up 77% no and 10.5% can't remember!

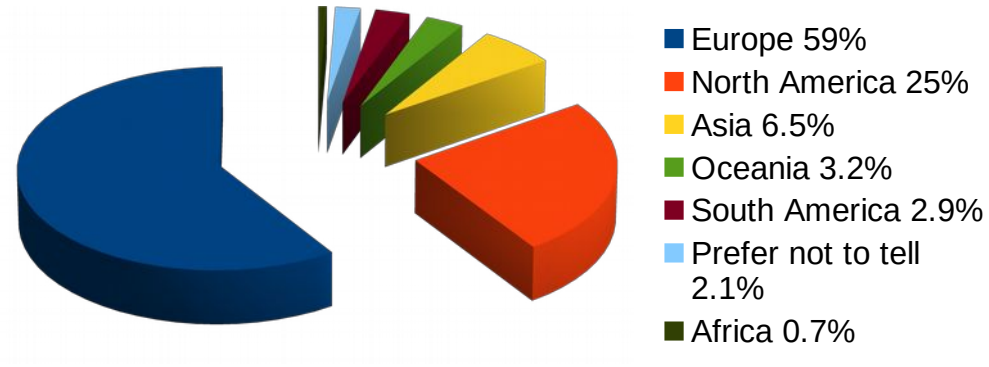
The people we reach to fill in this survey are more random than we'd like, possibly because we don't have this large really devoted users in the community. We used the same means and communication channels for the survey this year as every other year and still only about 98 humans who answered last year answered this year again... In absolute numbers, that's still up from the 83 return respondents in 2018.

Interestingly enough, even with three quarters of the respondents being new this year, some of the answers and the distribution of them are shockingly similar to previous years. This might be a sign that the opinions and views that are shared in these surveys are shared wider than just among the 732 that filled in the survey this year.

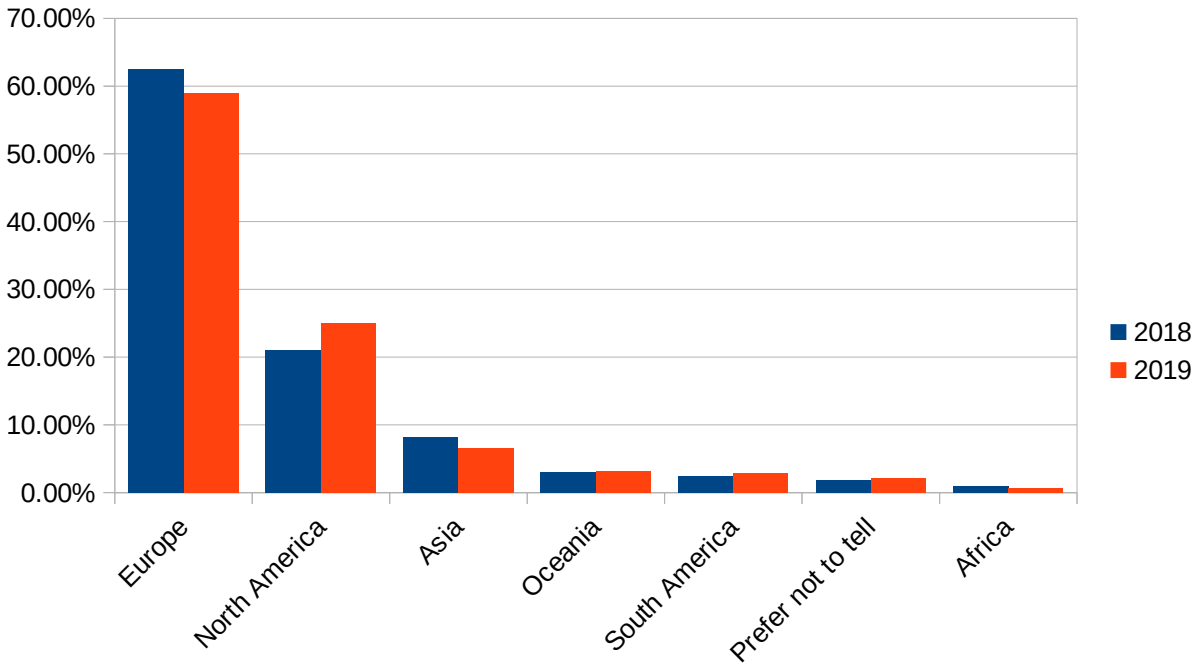
Users living where?

As mentioned in the previous section, only 13.8% of the respondents were the same as 2018. But amazingly the population distribution over the continents were almost identical!

Clearly a strong focus on Europe.



The numbers from 2018 and 2019 laid out as bars clearly show their similarities:



What kind of users?

A new question this year asked the respondent to characterize themselves. This as an attempt to better understand where the needs and desires come from.

Given the large amount of free-form write-ins to this answer, it was probably not a very well phrased question or the alternatives were a bit weak. I suppose humans don’t like to be categorized and lots of users seemed to want to put themselves into several ones rather than just one.

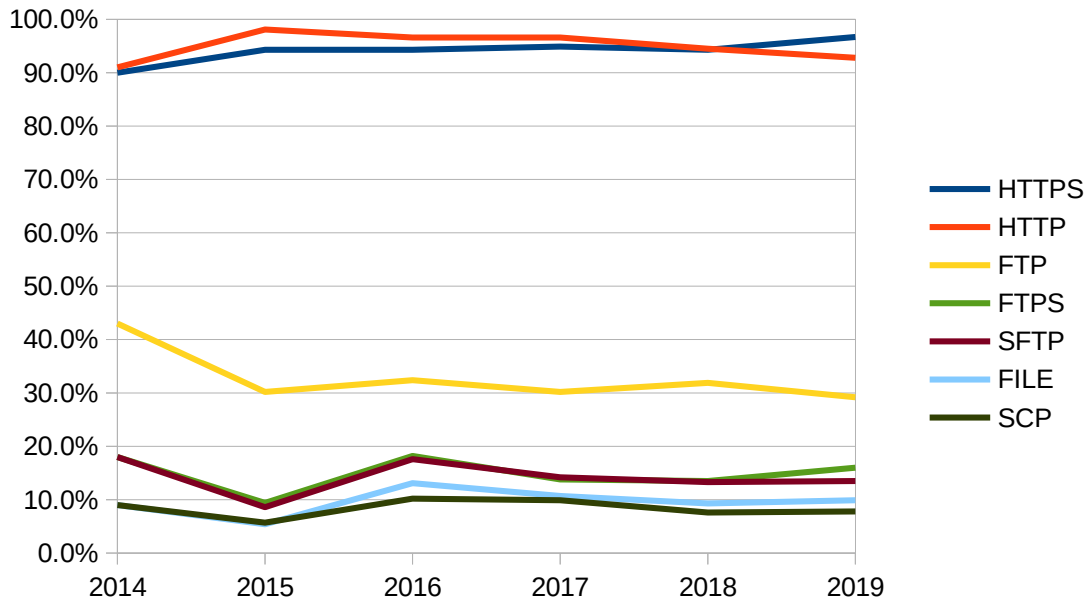
Backend developer	35.8%
Sysadmin	15.3%
App developer	14.8%
Web developer	12.1%
Shell user	8.5%
Prefer not to tell	2.8%
curl developer	1% (7 individuals)
Various free-form answers	9.7%

What protocols

N = 727

curl and libcurl are known foremost and primarily for their HTTP(S) support and capabilities, but the current curl version supports up to 23 different protocols. Several of them are conditioned on build-time requirements so far from all curl installations actually have them all enabled.

2019 is the first time HTTPS is marked as the most commonly used protocol after having been second to HTTP the five previous years. All the other protocols dwarf in comparison to these two. FTP continued its slow decay and is on an all-time low at 29.2% (down 2.7% from last year). Similar to last year, only five protocols are used by more than 10% of the user base. The development of the top-7 protocols look like this:

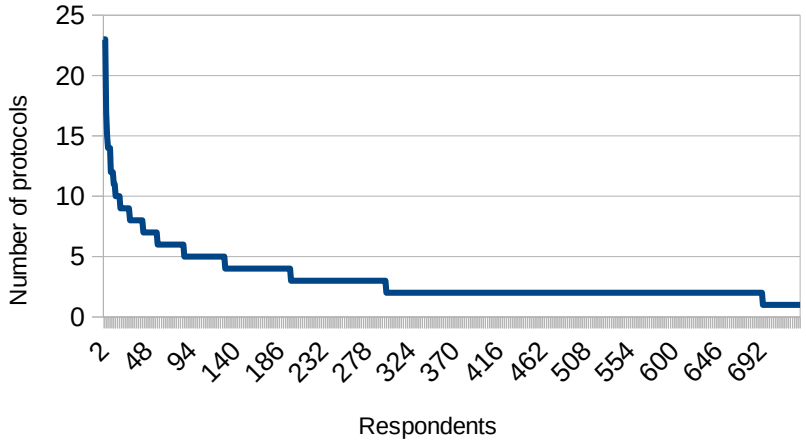


The complete protocol distribution for 2019 is shown in the table on the right here.

Protocol	2019
HTTPS	96.7%
HTTP	92.8%
FTP	29.2%
FTPS	16.0%
SFTP	13.5%
FILE	9.9%
SCP	7.8%
SMTP	5.5%
LDAP	5.0%
IMAPS	4.8%
TELNET	4.8%
IMAP	4.7%
LDAPS	4.5%
SMTPS	4.5%
TFTP	4.0%
SMB	3.4%
POP3	3.2%
GOPHER	2.8%
POP3S	2.8%
RTSP	2.1%
RTMP	1.9%
SMBS	1.4%
DICT	1.0%

Only 5.5% (40) of the users users marked they use a single protocol. Around half of the population uses two. Three users claim they use all 23. Median 2, average 3.2.

While HTTP and HTTPS are dominant and are used by virtually every user, we can also see that almost 50% of the population is also using one or more of the other protocols.

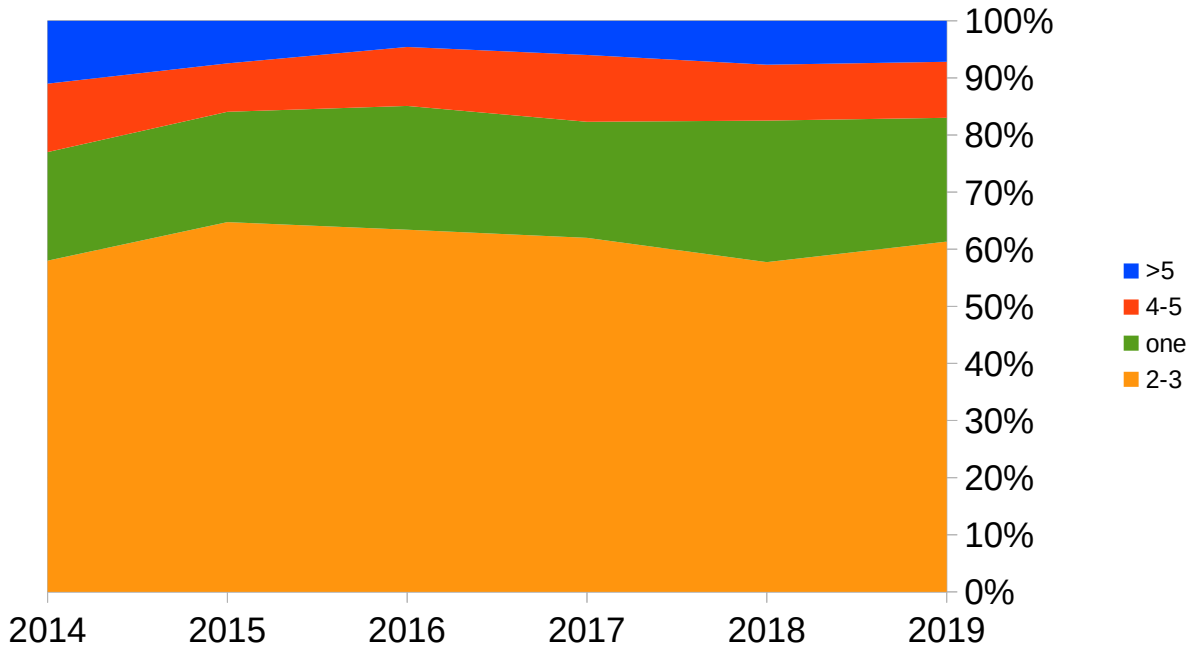


Multiple platforms

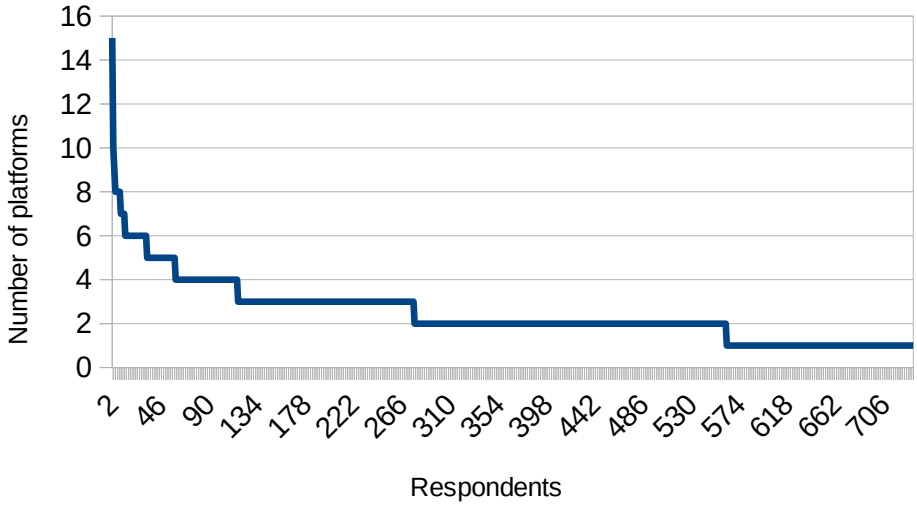
N = 725

curl is highly portable and can be made to run on a large variety of platforms. We also know that this is consistently mentioned as one of curl's primary sales pitches and voted "best area". There's a considerable effort being put into keeping this functionality across as many platforms as possible and this graph shows that it is generally appreciated and used.

The distribution of users using curl on a number of different platforms remains fairly similar over the years. Only one out of five uses curl on a single platform. One out of six uses it on 4 or more. 7% of the users marked *more than 5*.



One user claims using it on 15 platforms, but the average is 2.46 platforms and the median is 2.

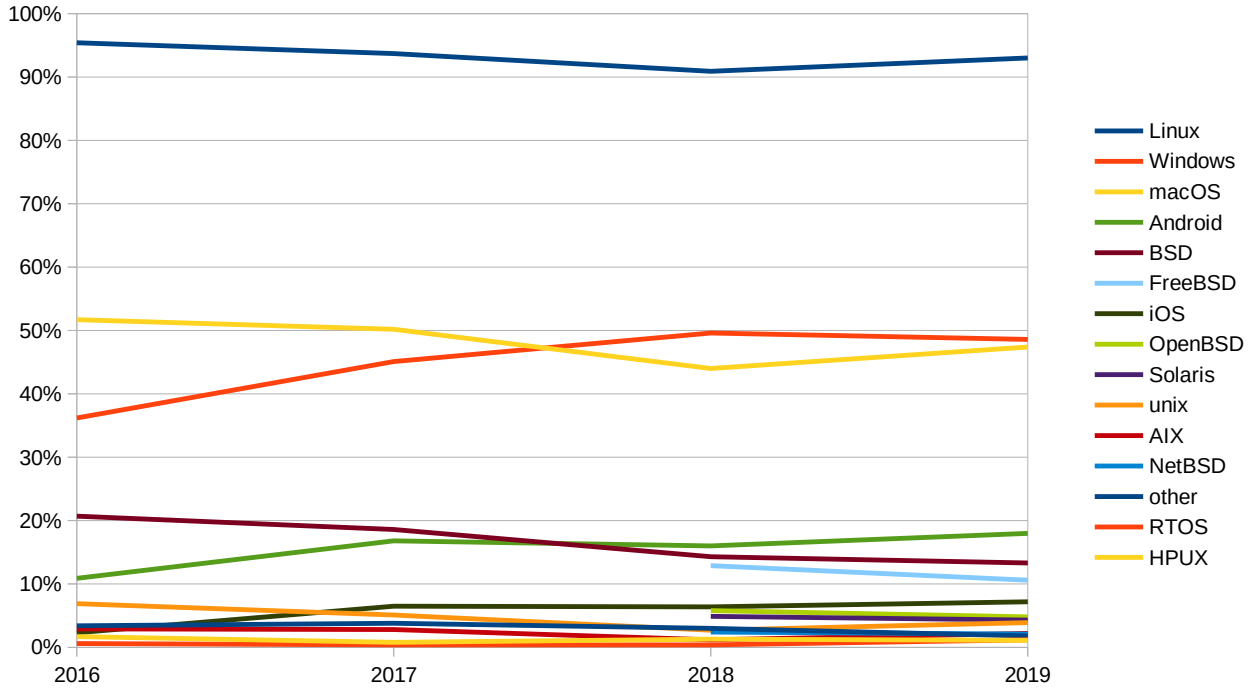


What platforms

N = 726

Last year Windows surpassed macOS as the curl platform number two. Linux still dominates, and curl is used on Linux by almost every user; 93% this year. Perhaps notable is that the Android user share increased yet again to an all-time high of 18%.

(BSD in the graph below is FreeBSD + OpenBSD + NetBSD sumed up, just because we asked about “BSD” generically in the early years so this makes it comparable. The “BSD” label will probably be removed in next year’s report.)



The full distribution of platforms, compared to last year is shown in this table. The “BSD” amount was counted and shown wrong (much too large) in last year’s analysis since I manged to add up the wrong percentages.

Platform	2018	2019	18 => 19
Linux	90.9%	93.0%	2.10%
Windows	49.6%	48.6%	-1.00%
macOS	44.0%	47.4%	3.40%
Android	16.0%	18.0%	2.00%
BSD	14.3%	13.3%	-1.00%
FreeBSD	12.9%	10.6%	-2.30%
iOS	6.4%	7.2%	0.80%
OpenBSD	5.8%	4.8%	-1.00%
Solaris	4.9%	4.3%	-0.60%
unix	2.7%	3.9%	1.20%
AIX	1.2%	2.2%	1.00%
NetBSD	2.4%	2.1%	-0.30%
other	3.0%	1.8%	-1.20%
RTOS	0.4%	1.2%	0.80%
HPUX	1.3%	1.1%	-0.20%
VMS	0.3%	1.0%	0.70%
IRIX	0.0%	0.6%	0.60%

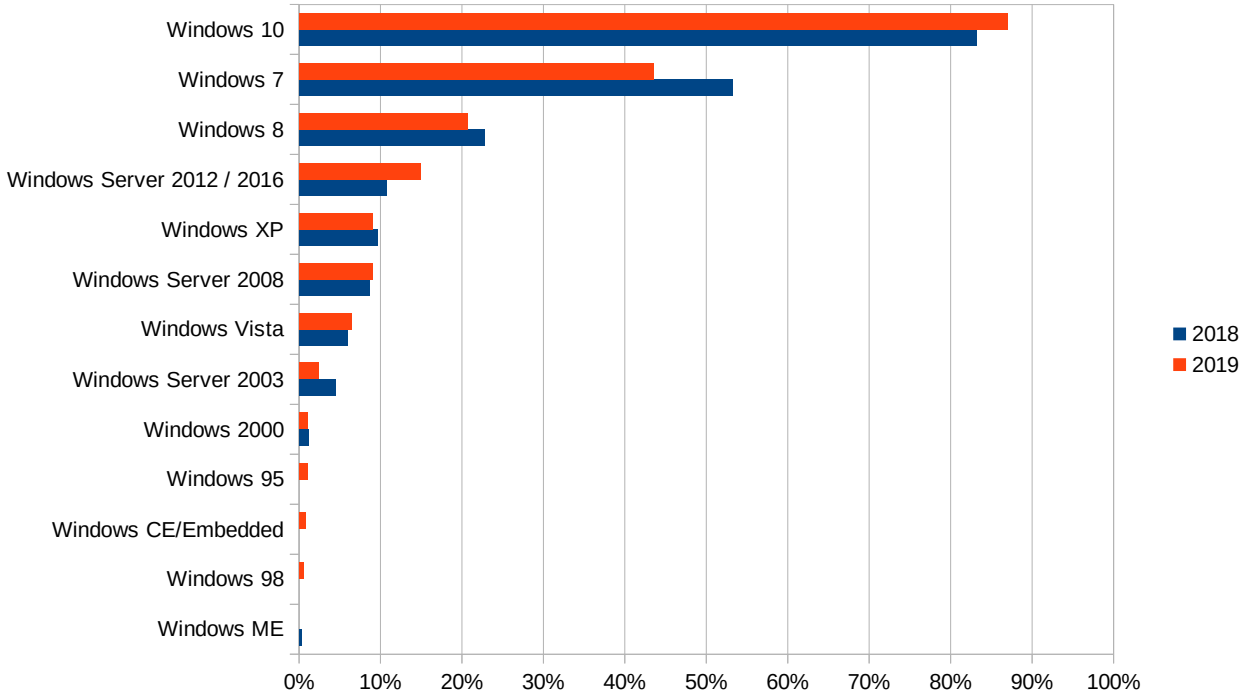
Among the write-ins, we found Illumos and z/OS but also “Ubuntu” and “GNU/Linux” ...

Which Windows versions

N = 368

(First, in the question prior to this, 353 users stated they use curl on Windows. Now, on this question that asks what Windows version they use it on, 15 users that apparently didn't use it on Windows in the previous question now filled in an answer here…)

The year since last survey shows that we have fewer users on Windows 7 (43.5%) and Windows 8 (20.7%) now and even more on Windows 10 (87.0%). Among the older Windows versions there are still a significant share of users. 9% are using curl on Windows XP, 6.5% are on Windows Vista...



Interesting write-ins were "Server 2019" and "Windows Subsystem for Linux", which we probably should consider having as choices in next year's survey…

Building curl

N = 725

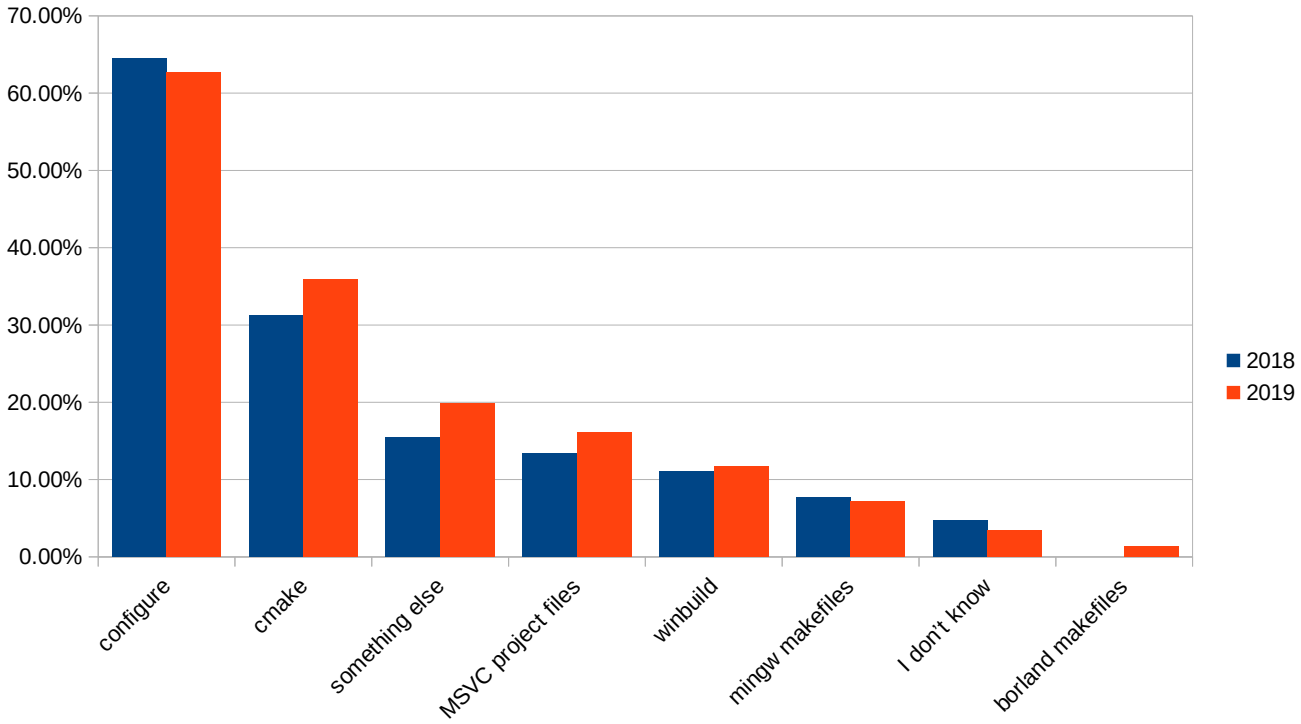
This year, more than one user told me that this question was ambiguous. What does “typically” mean and several users said they use several methods and different methods on different platforms. I suppose the feedback here is that this question can be improved.

70.8% of the users don’t build curl themselves. This number remains fairly stable for this, the third year this question was asked.

This year the cmake build system climbed even further. Now 36% (up from 31.2%) of the users who build curl themselves say they use it, while 62.7% (down from 64.4%) use configure.

It is also notable that “something else” is the third most common build option (19.9%). Presumably custom build solutions for users’ applications or specific environments.

Another interesting note could be that the Borland makefiles have been removed from the curl source repository since almost a year back so the 1.4% of the users who use those to build curl are a bit out of date...



What features are used?

N = 590

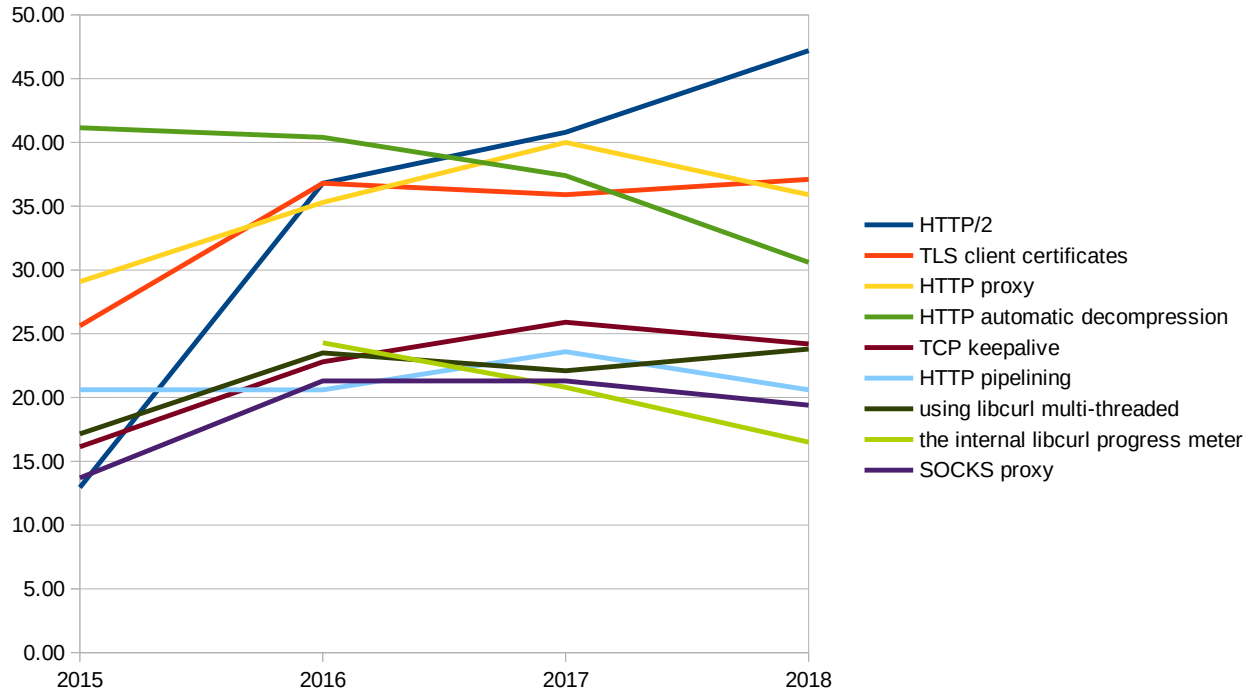
To get a clue about how popular and used some of curl's vast array of features are. It can also shed some light on feature and protocol trends if we look at development compared to previous years.

Turns out the distribution on features are mostly like previous years, with some exceptions:

Feature	2019
HTTP/2	55.80
TLS client certificates	40.30
HTTP proxy	34.70
HTTP automatic decompression	30.20
TCP keepalive	22.50
HTTP pipelining	20.20
using libcurl multi-threaded	20.00
the internal libcurl progress meter	19.00
SOCKS proxy	18.50
curl_multi_socket API	13.20
Bandwidth rate limiting	12.90
UNIX domain sockets	11.00
NTLM auth	10.30
.netrc	8.60
DNS-over-HTTPS (DoH)	7.30
CURLOPT_FAILONERROR	6.80
the share interface	4.20
HTTP/0.9	4.10
Metalink	1.40

New feature for this year is DoH, that immediately got a 7.3% user base. Quite notable I think. I also added Metalink as an answer option this year although the feature has been around for a long time, but I was curious if it is actually used. At 1.4% of the users I think we can safely say that it is *barely* used.

The development over the four last years for the top-9 most used features:

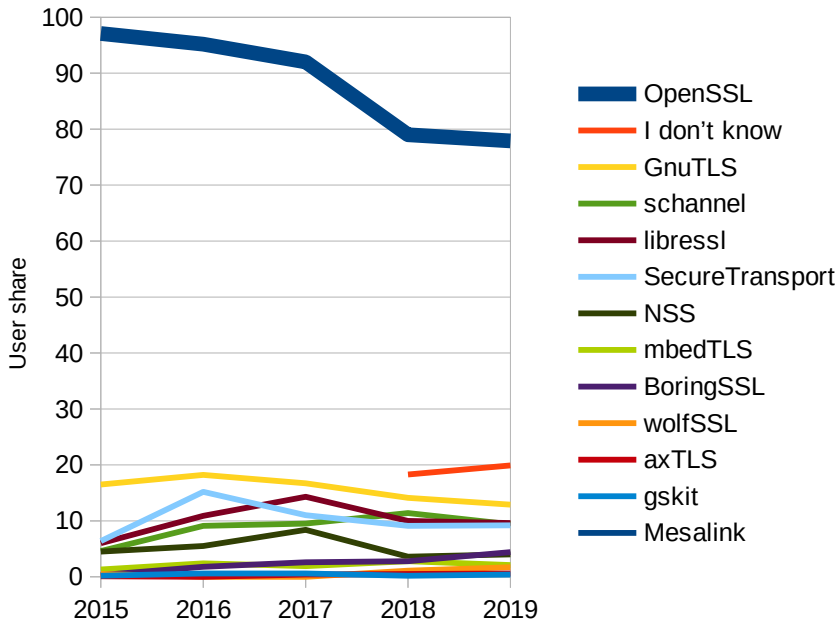


SSL backends

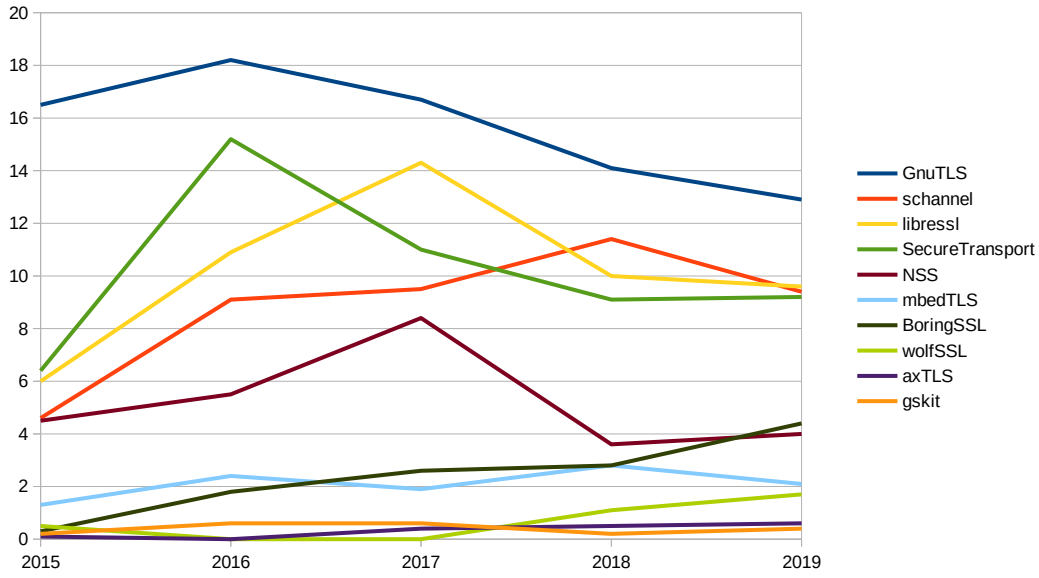
N = 705

OpenSSL remains the king of SSL backends in curl. At 77.9% of the users, there a loooong way down to the second most used backend - GnuTLS at 12.9%. The trend is however that both the OpenSSL and GnuTLS shares are shrinking slightly over time.

Redhat has dropped NSS as backend for curl and as it has traditionally been the only provider that builds curl with NSS by default, that backend share is likely do drop further going forward. axTLS is no longer supported and will vanish over time. In Apple's version of curl that ships with macOS, they've stopping building with Secure Transport and use libressl these days. Microsoft ships curl in Windows 10 built with the Schannel backend.



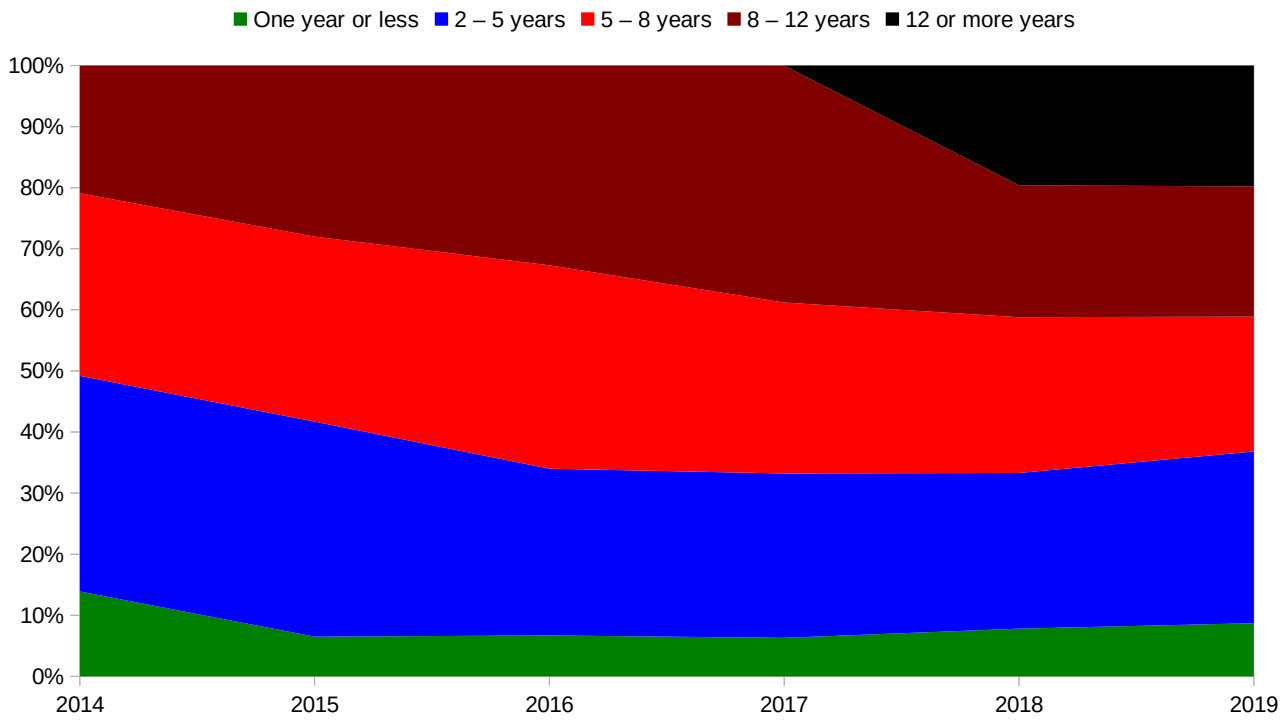
Zoomed in on the bottom part of the above graph (without Mesalink since it was new this year and we have no trend for it). BoringSSL grows to 4.4%, larger than ever.



Years of curl use

N = 723

How’s the user growth of newcomers and how’s the retention of keeping old users? Turns out the distribution of users among the different answer options is fairly stable.



The “12 years or more” option was added in the 2018 survey.

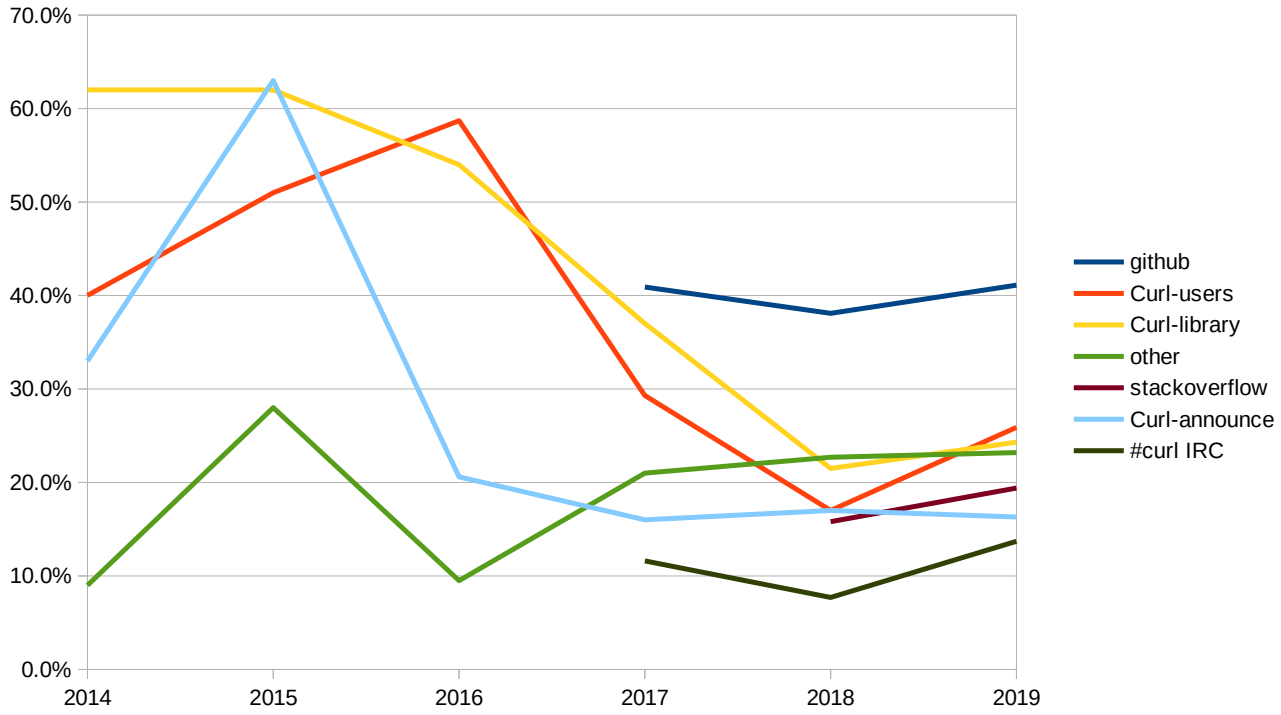
Participating channels

N = 263

Notably fewer responses to this question.

In which communication channels do survey respondents participate? Here's one of the answers with the most fluctuating results over the years. This year seems to be the year when the curl-users mailing list bounced back up a little bit, but the github repository is still the primary place for users to follow the project, at 41.%. Interestingly enough the curl IRC channel grew to 13.7% - that's also the only official real-time chat communication channel for the project and it typically has 100-110 persons joined any given time.

The fact that less that no single option gets more than 40% is probably due to the simple fact that the vast majority of users have no real interest or need to participate in these.



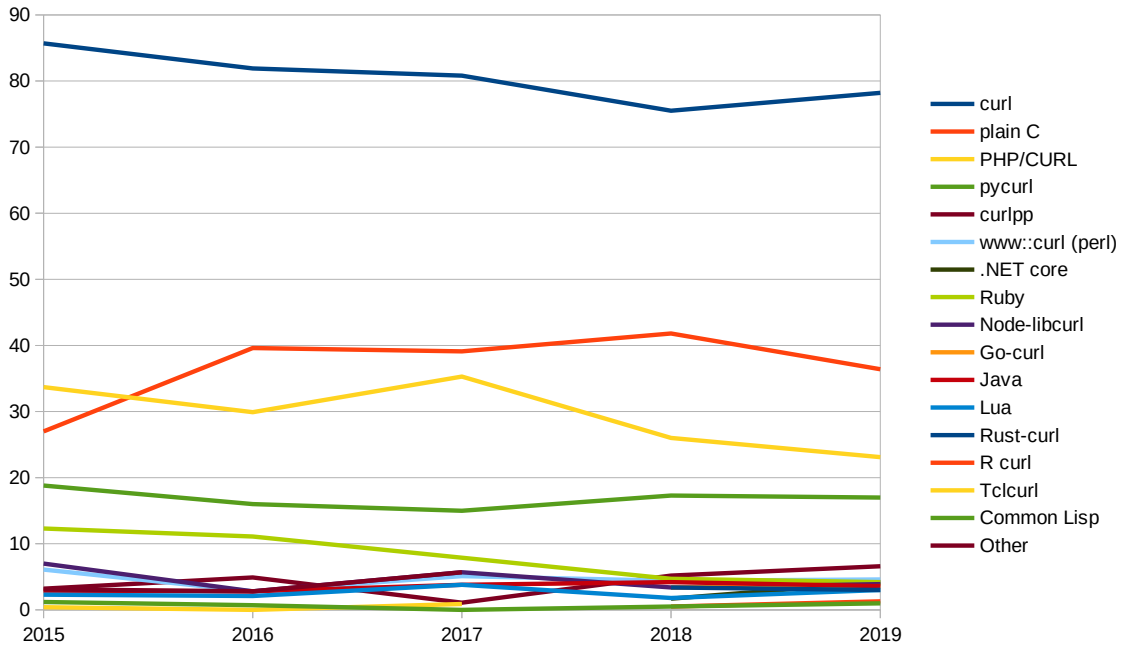
How do you “access” libcurl

N = 671

Basically a complicated way to ask which libcurl binding that’s used. Turns out the distribution is almost identical to previous years and no big surprises.

binding	2019
curl	78.2
plain C	36.4
PHP/CURL	23.1
pycurl	17
curlpp	6.6
www::curl (perl)	4.6
.NET core	4.2
Ruby	4.2
Node-libcurl	3.9
Go-curl	3.6
Java	3.6
Lua	3
Rust-curl	3
R curl	1.3
Tclcurl	1.2
Common Lisp	1

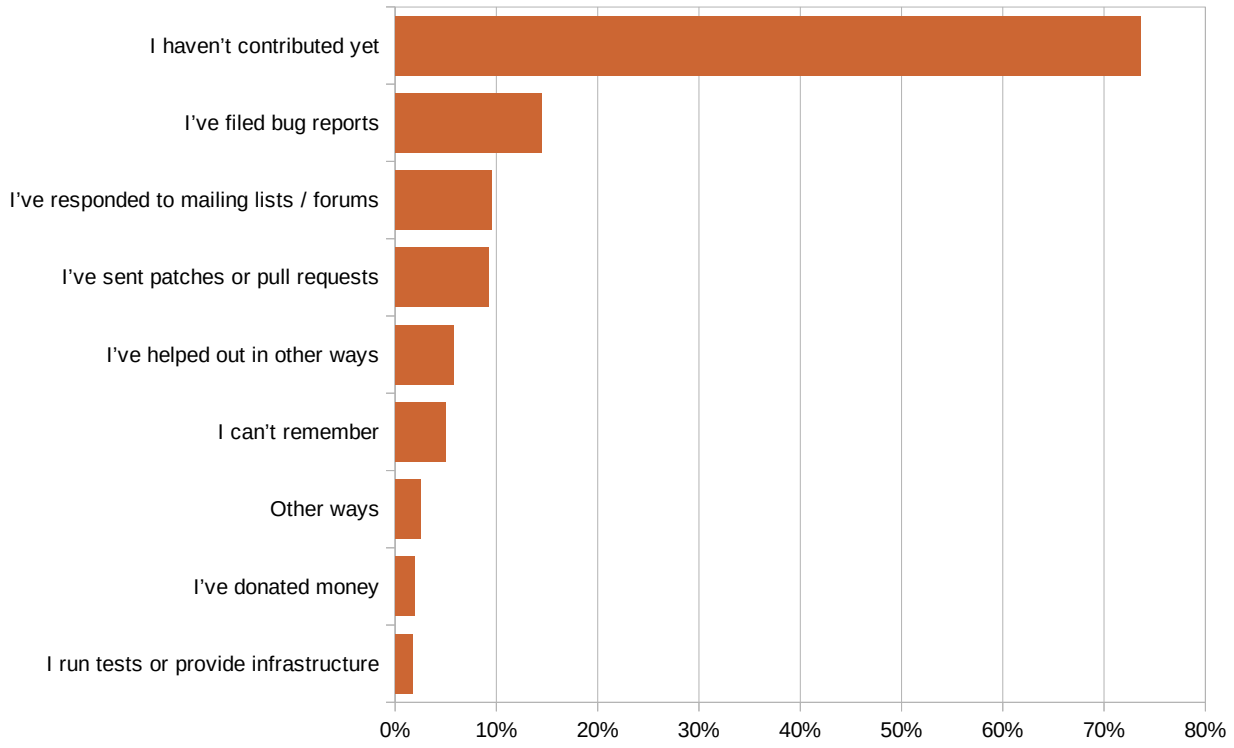
Drawn in a graph, you can see how they’re virtually identically distributed over the years.



Contributions

N= 640

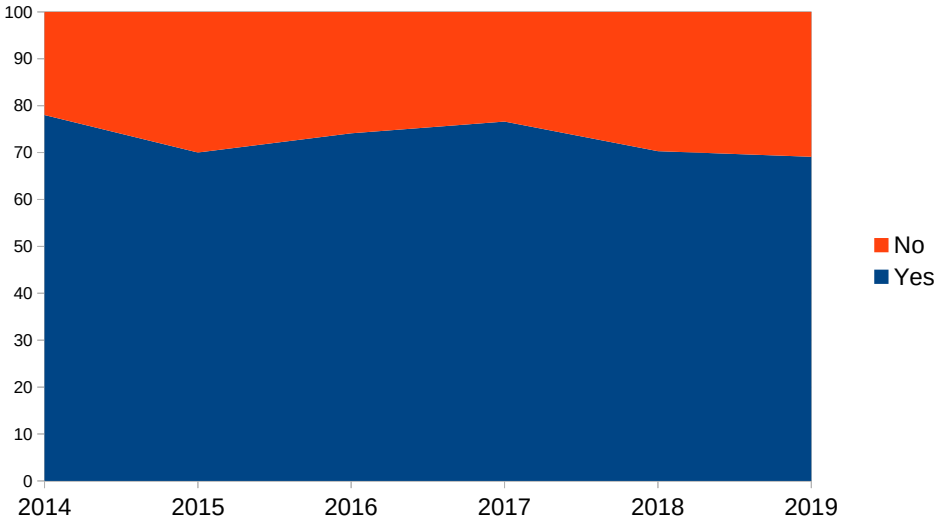
73.6% of the users have not contributed yet. That’s still 26.4%, more than every forth users, who have contributed in one way or another.



Other projects

N = 719

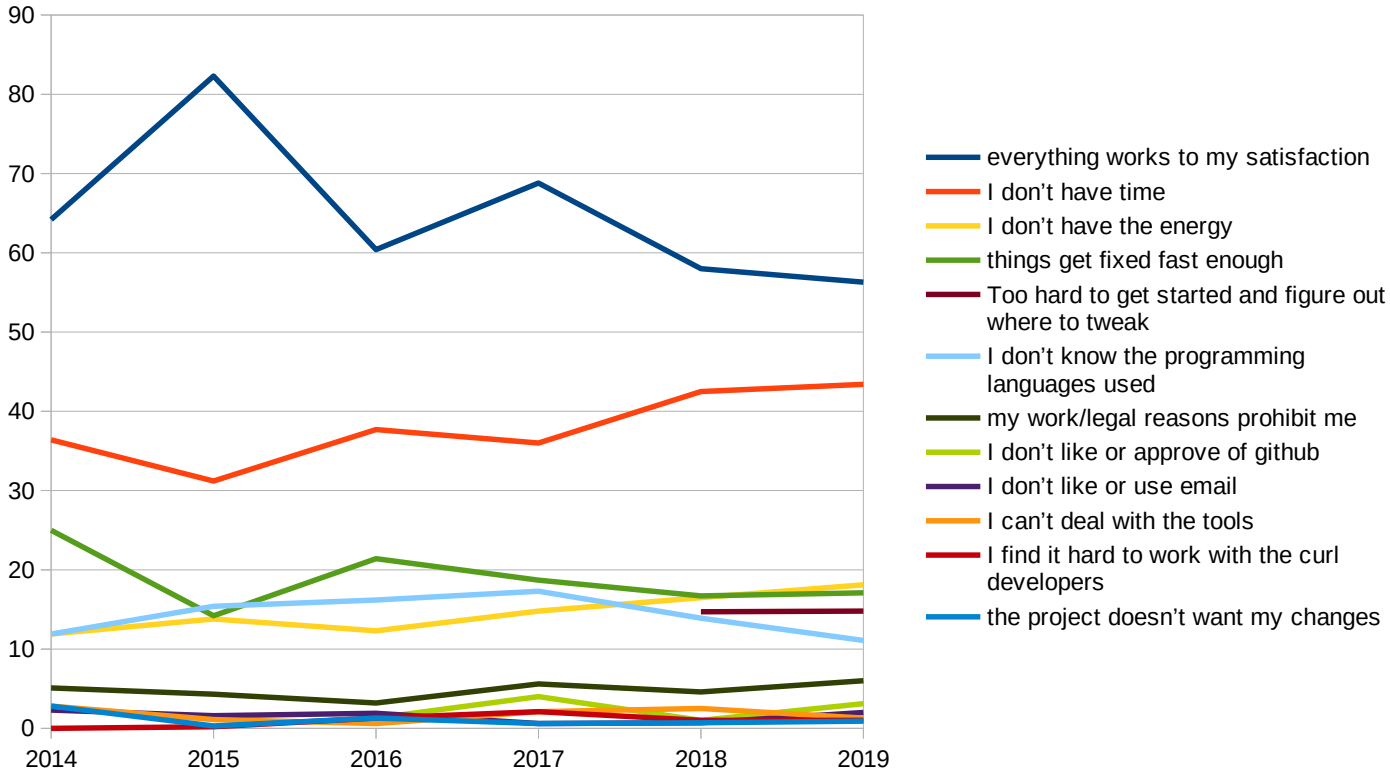
69.1% of the users are involved in other open source projects. A tiny decline can possibly be noticed year-over-year:



Reasons not to contribute to the project

N = 684

It seems there are two primary “excuses” why not to contribute more to the curl project. “Everything works to my satisfaction” (56.3%) is at an all time low but still the biggest. “I don’t have time” (43.4%) is conversely at an all time high and all the others are sub 20%. No major changes over time but all the reasons are used and there’s also a large creativity among the write-ins as to why you don’t contribute (more) but usually variations of the existing options.



“I don’t like or use email” (2%) seems like a really strange excuse these days since most contributions are done via github now and you can avoid using emails to a really large extent.

How to get (more) contributions from you?

N = 101

This was 101 free form text replies and I can't include them all here. I'll try to sum them up. Over all they were much in similar vein as last year.

"I don't know", "have more bugs", "invent more time", "it's already good"

Lots of variations of these.

"More docs" and "tagged issues"

We have so much documentation already that I honestly can't imagine that lack of docs is the problem. Possibly lack of the exact right docs that these users would like, but without very clear and specific feedback on this I don't think we can do much about it. Someone suggested "provide PDF documentation", as if **Everything curl** isn't already available with *plenty* of documentation in PDF and other formats.

Tagging issues with "help wanted" is something we've tried for quite some time and I don't think it has helped – *not even one single time*. The idea to mark issues with "easy fix" or "newbie friendly" is floated by several (and is in general hardly a new idea) but the problem with that is that we don't want bugs to linger around and if bugs are truly easy to fix then someone will fix them really quickly. They won't be kept around open for a fresh contributor to possibly find in a future.

Can we do more and other things to lower the bar for newcomers? I'm sure.

"Tweet about new features" and "promote features via social media/blogs"

This is already being done! I can only encourage more people to do it.

"Speak in GraphQL for POST requests instead of some bloated JSON"

Uh? Right… Some sort of feature-request I suppose but I don't understand it!

"Don't use the haxx.se domain - managers don't let us recommend that to customers (we are prohibited from mentioning haxx.se in our documentation, even though our product has use cases in which a customer must download curl on their own). They would allow curlproject.org (for example)."

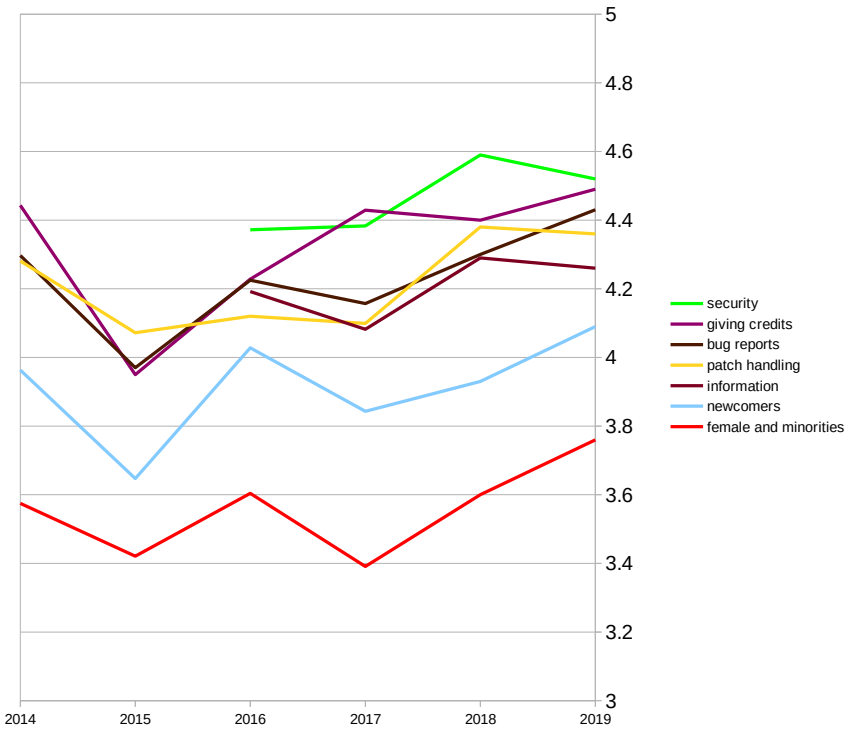
I would not mind having a domain like "curl.TLD" but there is none available and there hasn't been for a long time. curl.haxx.se works fine and is well established. We also offer mirrors from where curl can be downloaded and since most of the site is already available in a git repository, it would be very easy for someone to run a full copy of the site on another domain.

Additionally, I find it so silly that some organizations would avoid a site just due to it having "haxx" in the domain name so I consider sticking to this name also works as a tiny educational effort.

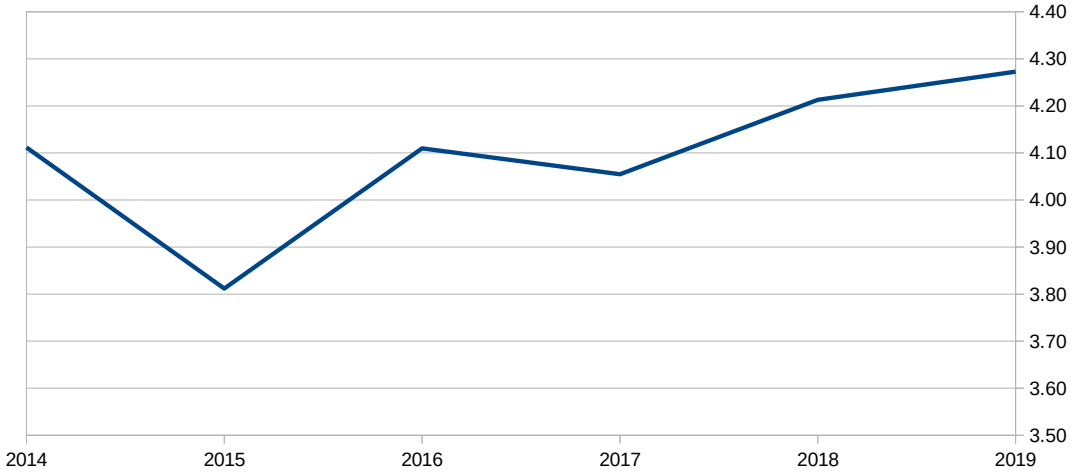
How good is the project to handle…

These questions require some insight or at knowledge about the project itself that maybe not everyone has. The individual scores at each of these are probably less important than the general deltas and trends of where they're going. They should tell us what users' perception of the project is in these various areas. This year we got slightly higher scores than 2018 in general.

Handling of female and other minorities remains a weak area (3.76).



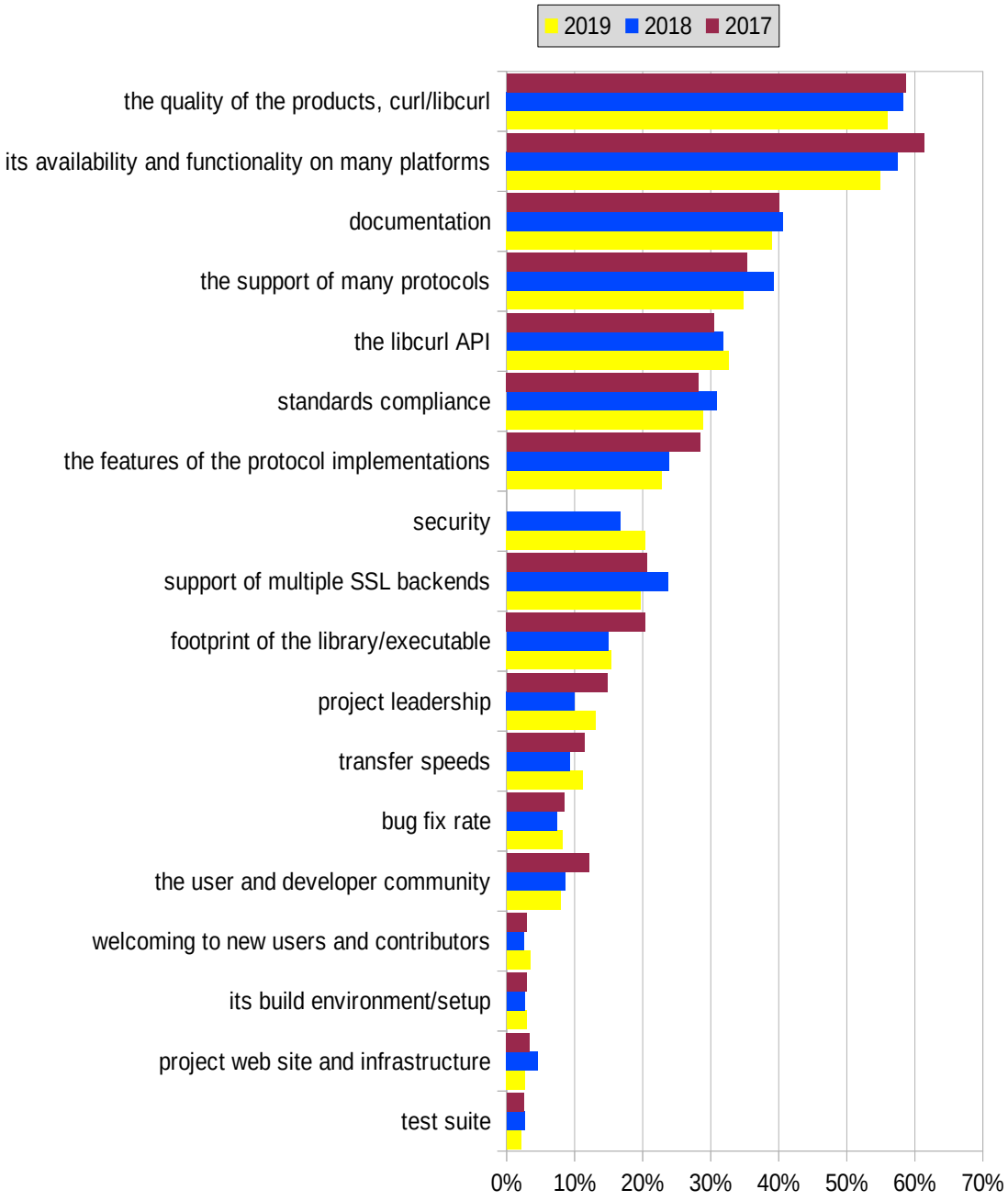
Making the average score (4.27) better than ever:



Which are the curl project's best areas?

N = 623

Another question with very similar answers to previous years. Looking at this year and the two previous in a graph:



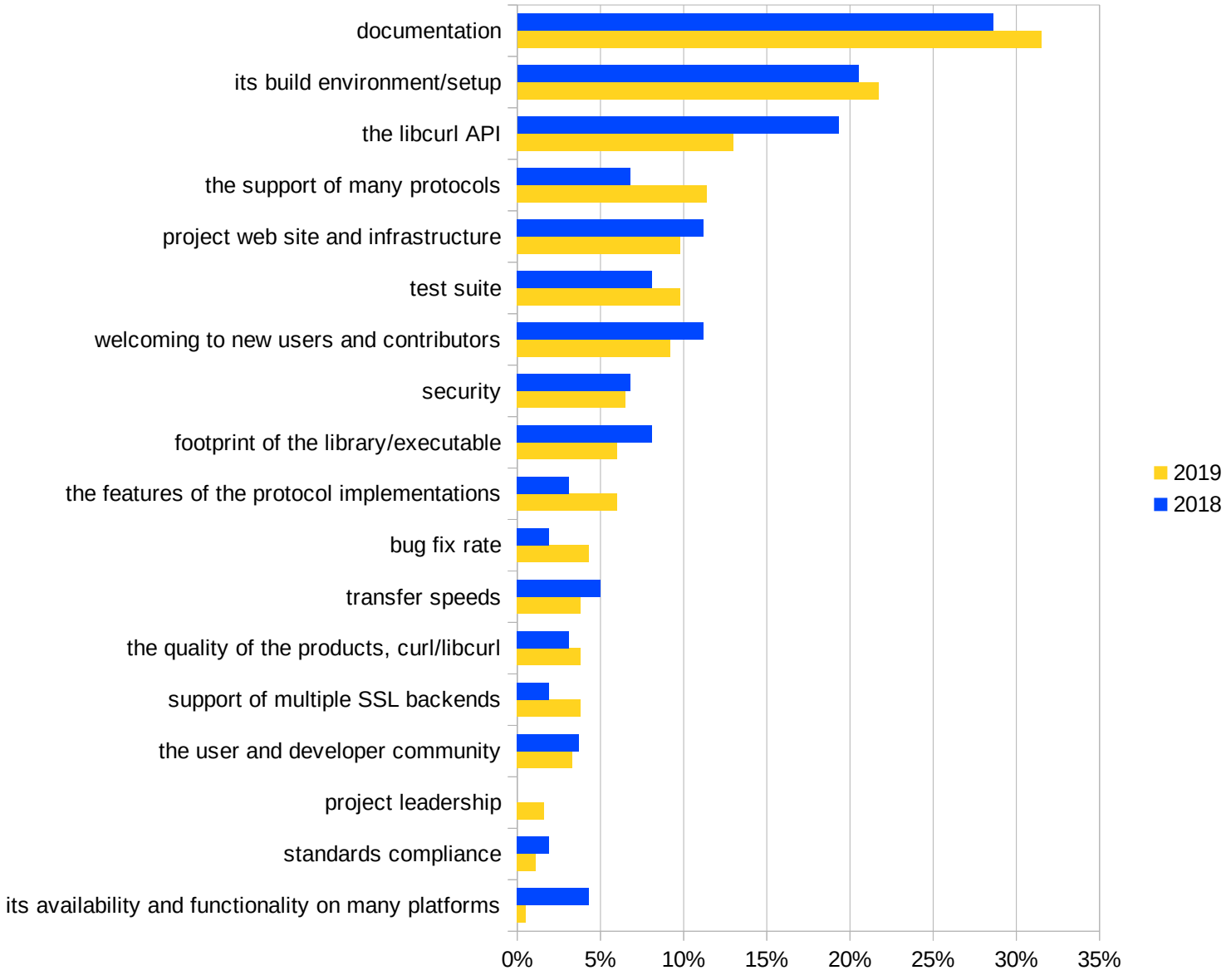
Product quality (56%), our multi platform support (54.9%) and our documentation (39%) are the top-3 qualities in the project according to our users.

Which are the curl project's worst areas?

N = 184

Less than a third as many users answered to this compared to the previous one about *best* areas. Both questions have the exact same answer options.

This is always interesting to see how the top choice of the project's worst area is always "documentation", while at the same time it is one of the top-3 *best* areas. It just proves that it is very hard to get documentation right for everyone and all types of audiences.



If you couldn't use libcurl, what would be your preferred alternative?

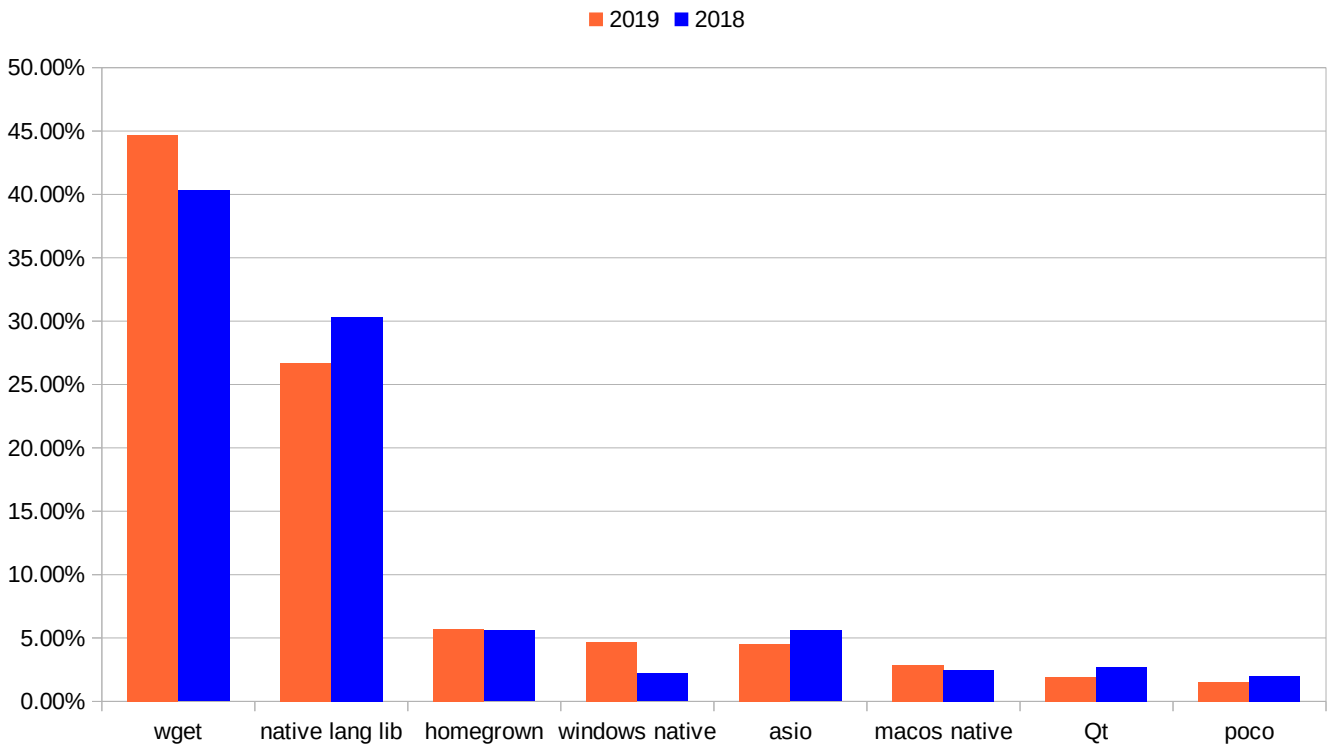
N = 618

Where's the competition? The answers to this question are difficult to interpret in many other ways than that libcurl has no clear or distinct competitor. If there would be no libcurl or if we would do the wrong turn at a fork in the road somewhere in the future, it seems clear that there's no single backup alternative lurking in the shadows. Rather, there seems to be a wide range of choices.

Users value and treasure libcurl's support for many platforms and many protocols (as per "best areas" above), and yet there's virtually no established competitor offering that.

Perhaps users in general tend to downplay the challenges of doing a solid a transfer library which could explain why "wget – *or code from wget*" is the number one choice (44.7%) and homegrown is at number three (5.7%).

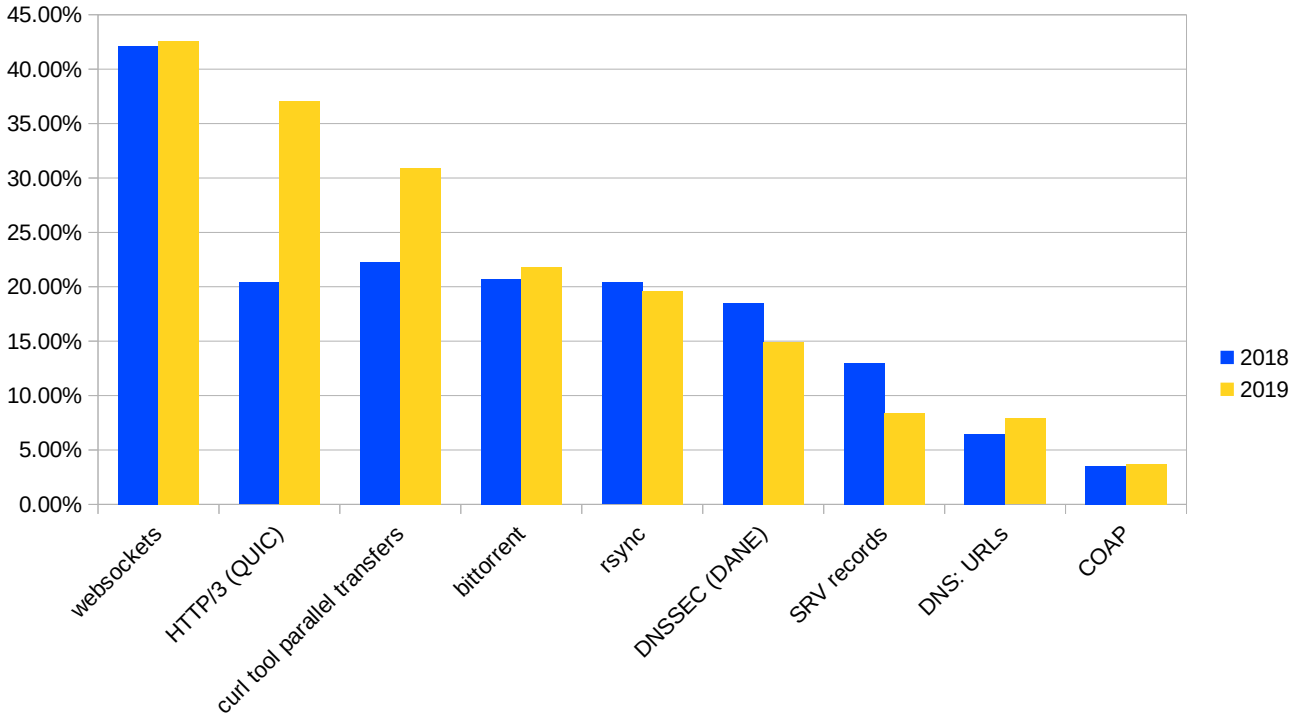
Several of the write-ins stated variations of "I don't know" and "I'd panic".



If you miss support for something, tell us what!

N = 404

websockets (42.6%) remain the number one missed feature as it has been since 2017 when it first was added as an available answer to this question. The biggest change this year is however HTTP/3 (which formerly was labeled “QUIC” in this question) that almost doubled from 20.4% to 37.1% this year. Presumably because people are starting to realize that HTTP/3 is coming “soon” as a real thing. Even at the third place “curl tool parallel transfers” (30.9%) grew substantially – which is fun since there’s already very code for this feature in a separate branch that will soon land in master to start being available for testing and experimenting with!



There were a significant number of “others” suggested for this question among which two even reached >=1% levels: MQTT (1.2%) and “web mirroring” (1%). For the former one, there’s an early pull-request available and given enough energy and focus it could certainly soon become reality.

Many write-ins suggested new things to add support for as listed below. My comments to some of them are added using italics within parentheses. I plan to make sure the best ones of these (according to me) are added to our TODO document.

1. a one-line C API (*I’m very curious on how that would look*)
2. Ability to change tor identity in line something like curl 'newIdentityTorProxy' URL Destination
3. AWS SigV3 auth

4. Better RTSP (*this needs users of RTSP to step forward and highlight what "better" means in this aspect*)
5. Better share and multi documentation (*please file bugs/issues on what's missing, in my view our documentation on this is already pretty complete so I need help to spot the bad parts*)
6. Can't stress this enough – websockets
7. checksum validation for download scripts
8. command line tool access to the URL parser (*I'll appreciate ideas and thoughts on how exactly this should be used as the actual implementation should be easy enough*)
9. connect to all resolved Ips (*I like this idea*)
10. Connection keep-alive across multiple curl instances based on UNIX sockets (*a "master and slave" mode for the curl command line tool has been discussed before and it could enable something like this*)
11. dat
12. ESNI (*yes!*)
13. Exposing crypto stuff (*what crypto stuff and how?*)
14. GPG encrypted .netrc (*if there's a (defacto?) standard for this, it sounds interesting!*)
15. GraphQL (*please explain how we can support this better than we already do!*)
16. gRPC
17. GUI support (*this seems far off but do explain, what kind of GUI would that be?*)
18. HTTP/2 multiplexing using cmd (*that would require parallel transfers so it is bound to come when we ship parallel transfer support. Soon!*)
19. Make API more generic for different protocols like HTTP and RTSP (*I need help to get this clarified and elaborated!*)
20. More examples for multi socket action (*yes please!*)
21. MPTCP (*yes please*)
22. OS integration like a browser - authentication UI (username/pass, client cert auth, ...), system proxy settings (PAC support included)
23. plugins (*plugins for what?*)
24. PortableApps compatible command-line tool
25. S3, Microsoft Azure Blob Storage, other cloud file stores
26. SMB 2 (*seems suitable so please join in and make it happen!*)
27. SPECS to generate rpms in different linux (*curl packaging is normally left to the experts and not by the curl project itself, but if someone wants to work on this I can't see any harm in us hosting or providing such scripts*)

28. unicode download filenames (*agreed, we should improve in this area. In particular on Windows.*)
29. Windows Integration for Internet Options (*I don't know what this means, please explain!*)
30. wget-like progress bars
31. better support to input json from cli inside scripts without double escaping quotes all the time
32. parallel execution of tests (something as easy as make check -jN) (*not so easy to fix, but we're many who'd like this*)
33. Mini curl using proprietary friendly SSL backend (e.g. mbedTLS as possible alternative to WolfSSL)
34. I'd prefer better notification and communication regarding security problems (*I'm all ears in learning what this can mean!*)
35. Callbacks for client certificate selection.
36. Ability to specify a callback to provide username/password for HTTP/Proxy auth.
37. Programmatic error handling for curl (e.g., based on response code, not just server-side errors)
38. Protocol sanity checks for "curl" cli tool; "Warning: server sent broken HTTP header value in response" (on by default)
39. non-silent version of --fail option
40. Mask usernames and passwords from process listing in e.g. urls and options specified on command line (*command line options for passwords are already masked*)
41. Option to represent entire response including metadata (e.g. HTTP status, headers) in JSON format like { status: 200, statusText: OK, headers: ["Content-Length": 34567, ...], body: "foobar" }
42. Maybe an automatic header formatting feature? (that would allow JSON content from debug header to be displayed nicely)
43. A libcurl-API which is independent of sockets. Meaning I provide a stream of bytes from the outside to libcurl and libcurl gives me back a stream of bytes instead of using a socket directly.
44. Pcap dump of cURL generated traffic
45. Don't clobber existing files when downloading a range
46. Simpler handling for REST/OAuth2.0
47. Better SSL/TLS handshake info so I didn't have to use OpenSSL s_client and could do everything with one tool!

48. Main use case is for HTTP(S) support on embedded devices (Linux, <16MB flash) so any effort to reduce the size of libcurl (e.g., tinycurl) would be welcome.
49. CMake support still has some issues, and it took a bit of effort / modification of the CMakeLists.txt files to embed libcurl as a subproject in our CMake build system. Other than that, I absolutely love libcurl, it has been a blast to use, it deals with all the HTTP standard cruft I don't want to, it is really well documented and there are so many useful features. Keep up the great work!
50. HTTP Streaming (*afaik, curl already supports HTTP streaming perfectly well, what's missing?*)
51. Greater support for ftp commands... I.e. native ftp on some platforms don't support ftps, but -Q isn't always enough to do what was previously done via .netrc with ftp
52. Bug with https certificates (*please file an issue on github and explain what the problem is!*)
53. Thread-safe initialization, where supported by the underlying TLS library.
54. Dynamically loading the libraries for less common protocols (libssh, librtmp, etc...). (*dynamically loading libraries is a huge pain in the … so this is not likely to happen any time soon.)*
55. NTLM domain authentication is not described well in the documentation, just f.x. in the mailing list here: https://curl.haxx.se/mail/lib-2005-01/0234.html
56. Certificate revocation check
57. User's control of TLS session reuse
58. Per-multi speed limiting
59. in-memory client certs and private keys
60. a function to clear idle connections in a multi (to forget the in-memory private keys no longer in use)
61. a callback which is called for new SSL_SESSIONs
62. Example for dynamic / static linking; Visual Studio builds;
63. Curl pop3 write_callback sometimes answer with empty 2 char (spaces?). (*please file an issue on github and explain what the problem is!*)
64. Put CURL version into source code (*huh? The version number is generated and stored in a header file in every release.*)
65. source code needs some improvements and cleaning support both client/servers for top protocols (*winner of the "most vague suggestion" award this year*)
66. Integrated support for Windows Certificate Store when using the OpenSSL backend.
67. ASIO-style transfer API
68. Allow us to enable features of curl without recompiling

69. I simpy can't install rubyTyphoeus on Win 10 because of curl
70. Windows IOCP
71. improved CMake support
72. I sometimes need to install curl on an older or obscure platform that does not yet have any https client. The problem is that curl.haxx.se redirects http to https. This is a chicken-and-egg bootstrap problem. Because curl.haxx.se is often the best way to obtain the first ever https client on these machines, it is important for curl.haxx.se to support (cleartext) http downloads.

What would you like to see the project REMOVE?

N = 37

1. Get rid of the banner on the cli. Or make '-Ss' the default.
2. Output to stderr when piping output without silent
3. Drop support for older windows versions like Windows 2000 and older. The documentation is limited for those older versions and I don't think it's worth our time to work around API limitations for much older versions. XP is old still has some usage.
4. HTTP/0.9 (*support for HTTP/0.9 will become opt-in very soon*)
5. CLI progress meter when trying to grep so I have to use silent and 2>&1 I would prefer to have no progress unless I specify it (*the progress goes to stderr precisely so that it won't interfere with grep*)
6. Everything that I don't use :P But seriously, it [at least the command line tool] feels as if it's trying to do way too much, and is burdened by the legacy decisions at this point. It's probably no big deal in practice though.
7. making ~90% of the features into plugins would be cool (*I'm not convinced of this coolness*)
8. I'd like to see more secure defaults. For example, HTTP without TLS should require an explicit flag. (*An interesting suggestion but I don't think the world is ready for such a drastic move. I'm prepared to entertain adding a build-time option that enables this, but which would be disabled by default.*)
9. DNS over HTTP (DoH) (*can be disabled at build-time already and will be even easier to disable at build-time in the future*)
10. autotools should be replaced by cmake :-) (*this has been suggested many times but so far nobody has made the cmake build even close to the autools version feature and capability wise and as long this is true cmake will remain a less preferred build option for curl*)
11. Drop support for (various combinations of) protocols (*dropping support for something we once added is sensitive and as can be seen in the beginning of this document, all protocols curl supports have users.*)
12. The OS X native ssl doesn't work well with client certificates, at least a couple of years ago. (*Secure Transport support can probably use more love in general, but note that not even Apple themselves ship curl built that way anymore…*)
13. C89 Compliance is outdated. Please consider using modern standards. (*I would argue that we've already paid most of that price and that there's not a lot for us to gain by upping the level to C99 or similar. On the contrary, we have users who are stuck on older compilers and appreciate our dinosaur age compliance*)
14. Keep everything as it is, but please provide more compile-time option flags to disable unused features (cf. tinycurl). That way, we can enable/disable features on a case-by-case

basis, and if we *do* need a feature later on, we can still enable it. (*This is perfectly inline with current thinking and plans going forward. We will get more knobs and more fine-grained such options in the future*)

15. Redirects from HTTP to non-HTTP protocols
16. Remove the CMake and other build files, package libcurl for distribution as an amalgamation (see sqlite)
17. Downstream debian (ubuntu?) should build libcurl/curl with libssh2 = SFTP enabled (*I'm afraid that's something we cannot affect or change. I'd urge you to file issues and communicate directly with the linux distros you want to change*)
18. No reported memory leaks after a https request with openssl backend, e.g. possibility to cleanup all statics. (*done! curl doesn't leak any memory and if you just use a reasonably modern TLS library there will be no valgrind reports at all.*)
19. C (*See further down in the "rewrite curl?" question)*

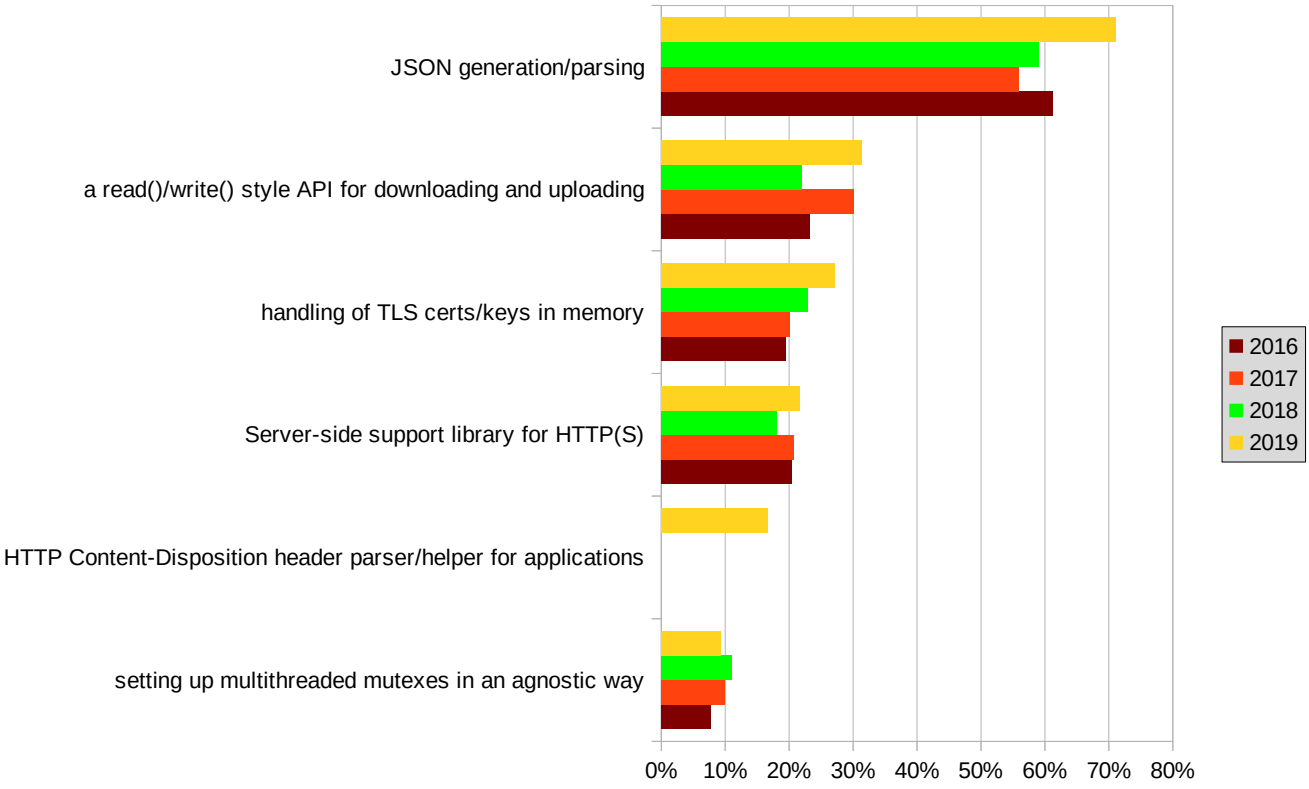
Which of these API(s) would you use if they existed?

N = 419

This year I removed the question about an URL parser API that were among the most frequently chosen answers for several year since it has now been implemented and introduced in libcurl! (Even though a fair amount of write-ins asked for a URL parser API this year anyway...)

Instead there was a new question about Content-Disposition header parsing that ended up ranked number 5. Over all the answers got roughly the same answer distributions as they've gotten other years.

The second most answered choice about read()/write() API (31.3%) is what is already offered in the fcurl[1] repository and yet that one has yet to see any particular use. And before that gets used and we get feedback on that, I don't think we can expect any serious effort of trying to get that merged into libcurl proper. I think this rather shows that it is easy to select a few checkboxes in a survey but an entirely different thing to actually use what eventually gets implemented...



1. Parser for other HTTP headers (*HTTP headers are notoriously fickle so I understand where this is coming from, but they are also fickle for a reason so generically parsing them and returning info from them is… complicated*)
2. Websockets (*again*)
3. GraphQL API (*how would that work?*)

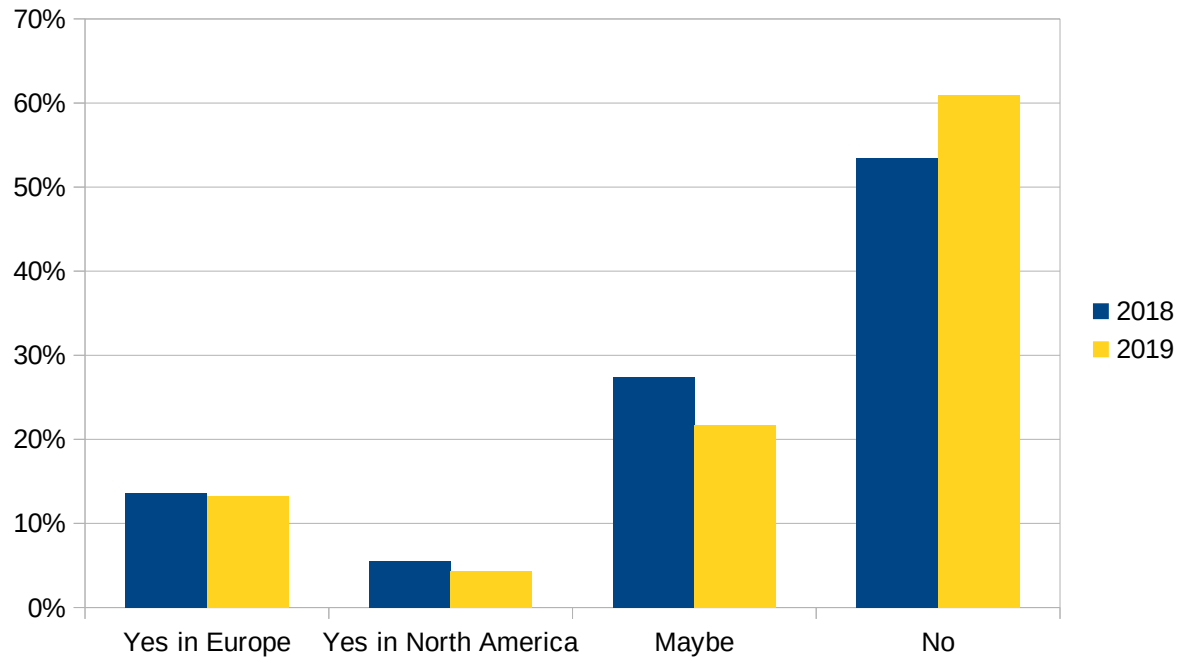
1 https://github.com/curl/fcurl

4. Automatic javscript authentication (*I don't know what this is. PAC?*)
5. Specify HTTP headers as key, value pairs. Since I allocate memory to make a header line, then slist_append makes another copy of it. (*sounds like an alternative API that could be easy to provide?*)
6. API for URL encoding/decoding
7. IPv6/IPv4 address string validity check

Do you wish to attend the next curl://up meeting/conference?

N = 631

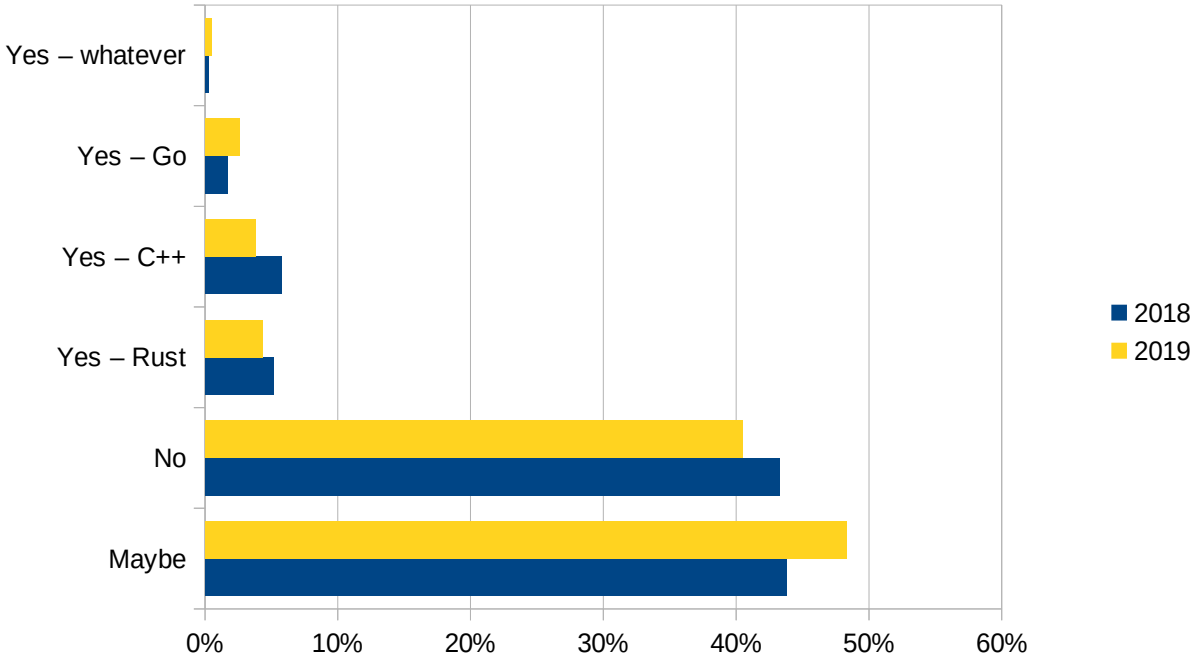
A little fewer No and Maybe but roughly the same responses as last year (which was the first time this question was asked). Given that we ended up a little over 20-something participants on curl up in Prague this spring after about 80 peeps said “Yes if in Europe” last year, these numbers (13.2%, or 83 users said yes if in Europe) would indicate that we might end up the same number in curl up 2020 (date and location for that has not been decided yet).



Should libcurl get rewritten in another language?

N = 654

Admittedly this is not really a question that anyone off the streets should be able to answer, but given that there are a lot of opinions anyway I still think it is interesting to learn how people view this question. I find it comforting that most people either say outright “No” (40.5%) or “Maybe – I leave that decision to the ones in the know” (48.3%). 11.2% said yes of some flavor, and this year Rust (4.3%) become the most popular yes choice, passing C++ (3.8%) which was the highest Yes answer last year. Both still decreased a little.



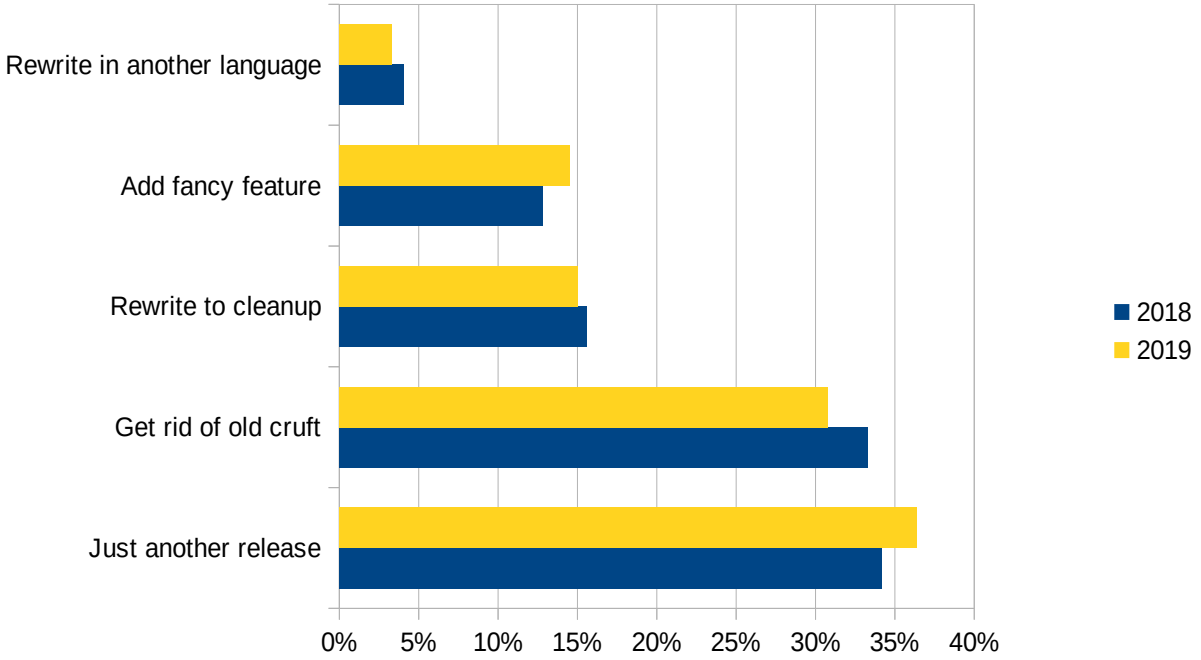
What should "curl 8.0" be?

N = 552

Here’s another question that isn’t terribly important but helps us put a toe in the water and get a sense for what sort of things our users would prefer.

Curl 8 is extremely likely to happen within a five year period and the discussion recently has been around doing “just another release” that bumps the version number and that also happens to be what 36.4% of the users answered. It also happens to be the, by far, easiest solution. The second most answered choice “Get rid of old cruft and barely used protocols” (30.8%) is certainly a much harder route as that risks leading to endless discussions and debate around what exactly qualifies. One man’s old cruft is another man’s must-have.

If there’s something these annual survey has taught us, then it is that *everything* and *every feature* in curl is used by someone.



Which question would you like to see in this survey next year?

N = 49

Edited to remove most jokes, weirdos and duplicates.

1. How often do you use curl?
2. Which build system should curl use (on unix)?
3. Regarding competitors, allow more than one answer. Even with multiple competitors I'd still be at loss.
4. A very open-ended "Is there anything else we should know?" at the end.
5. Don't you think this whole project is amazing?
6. What is the most unexpected place you've found curl? (e.g. car entertainment unit)
7. Primary use cases (what do you use it to accomplish) like checking HTTP headers, downloading files, downloading HTML
8. conference question: Yes, location doesn't matter.
9. Option to disable insecure options such as -k, NULL ciphers
10. Licensing ecosystem in which users deploy libcurl, inferred constraints regarding licensing (mostly of the dependencies, especially SSL backend, since curl license itself is permissive enough) and footprint (also w.r.t dependencies)
11. what's your current curl versions? :)
12. more about UX and making sure the rarely used features are usable and documented
13. If we liked the progress made in the last year and if not if we'd be willing to help to reach that next year
14. Make it shorter, not longer :)
15. "Should we replace autotools by CMake? [Yes]"
16. If you couldn't use 'curl' (the CLI), what would be your preferred alternative/s?
17. How likely are you to recommend curl?
18. "What supplementary utilities/documentation would you like to see installed together with curl?" perhaps as a multi-checkbox with alternatives like "json manipulation/extraction tool", "oauth examples/helpers", "examples how to use to access popular apis like twitter, etc"
19. Asking for GraphQL support. If not implemented yet. The POST GraphQL query converted to JSON is awful to see. You can see the diff using the Insomnia app.

20. Add “Follow Curl developers on Twitter” to channels you subscribe too (it’s how I knew to fill in this survey!). And yes that probably mostly means @bagder but probably better not to explicitly state that (which I guess from what I know of you is why you didn't include it as a channel!).
21. I'd like a way to be more sure old unsupported OS builds still remain a priority as a way of keeping some security even in situations where running the old environment is itself an insecurity.
22. "The number of security issues" and "... bugs" as options in "worst areas" question. The dozens of bugfixes proudly announced in each release makes one wonder how many bugs are still remaining to be found, or added with new features. Maybe I'm being harsh.
23. Should the haxx.se domain be replaced to improve the product's perception by some subsets of non-technical stakeholders?
24. What alternative download utility do you use (wget, aria2, httpie, etc…)
25. Is there any platform you feel is under-supported?
26. What do you think of the XX language binding to curl?
27. About downstream packaging of curl/libcurl
28. possibly some consideration of QUIC
29. curl learning paths
30. "Do you only like curl because it's easier to pipe it into a file than to remember whether wget is -o or -O"
31. I don't know about questions but I'd like to answer that I contributed to curl!
32. should curl.haxx.se allow cleartext http downloads, to support the case where the downloading client device does not yet have https capability?
33. "would attend curl://up as long as it's NOT in the UsofA"
34. How did you get started helping this project?