curl user survey 2017 analysis

“Argh!!!!!!!!”

summary and analysis by Daniel Stenberg

version 1.0, May 30, 2017

About curl

Curl is an open source project that produces the curl tool and the libcurl library. We are a small project, with few core contributors, with little commercial backing and yet we're over 19 years old and we have gathered help from over 1500 named contributors through the years. Our products run in a vast amount of internet connected devices, tools and services on the current Internet.

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

Background

We do this user survey annually in an attempt to catch trends and longer running changes in the project, its users and in 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 and the vast majority of users and curl developers never get to hear about it and never get an opportunity to respond.

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.

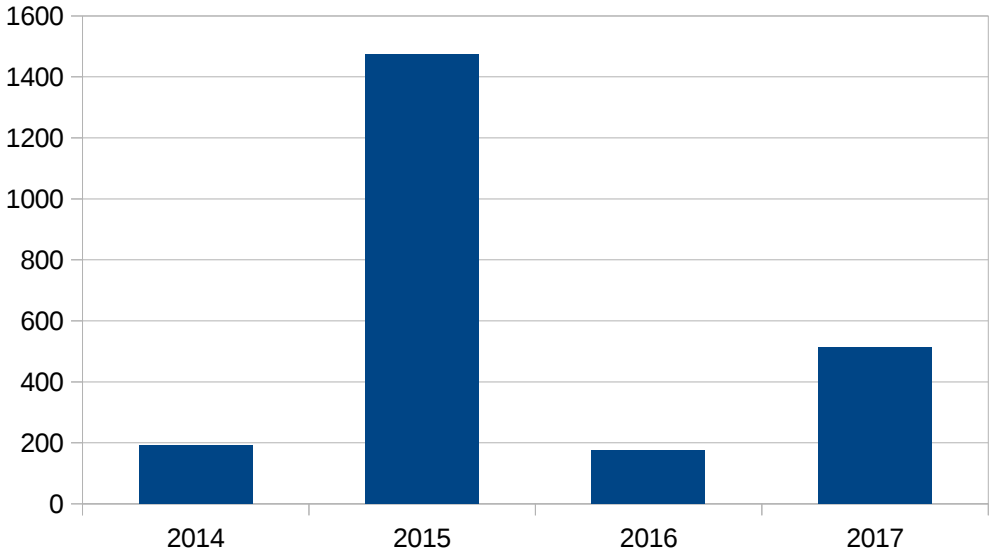
This year, the survey was up 16 days from May 12 to and including May 27.

It was announced on the curl-users and curl-library mailing lists (with two reminders), numerous times on Daniel's tweeter feed (@bagder) and on Daniel's blog (https://daniel.haxx.se/blog). The last few days of the polling period, the survey was announced widely on the curl web site with an "alert style" banner on most pages on the site that made it hard to miss.

Number of responses

Possibly as a result of the slightly harder push for the survey this year as compared to last, we received almost three times the amount of answers as last year: 513, compared to 176. As always, one of the biggest problems with these surveys is that it is answered by a different subset of people every year, so all analyses or drawn conclusions have to be read with that huge caveat.

The last four years' participation levels is shown in the diagram on the right here →



Please check the protocols you use curl/libcurl for

n = 506

Are there supported protocols in curl that aren't used that we could consider removing support for?

- No

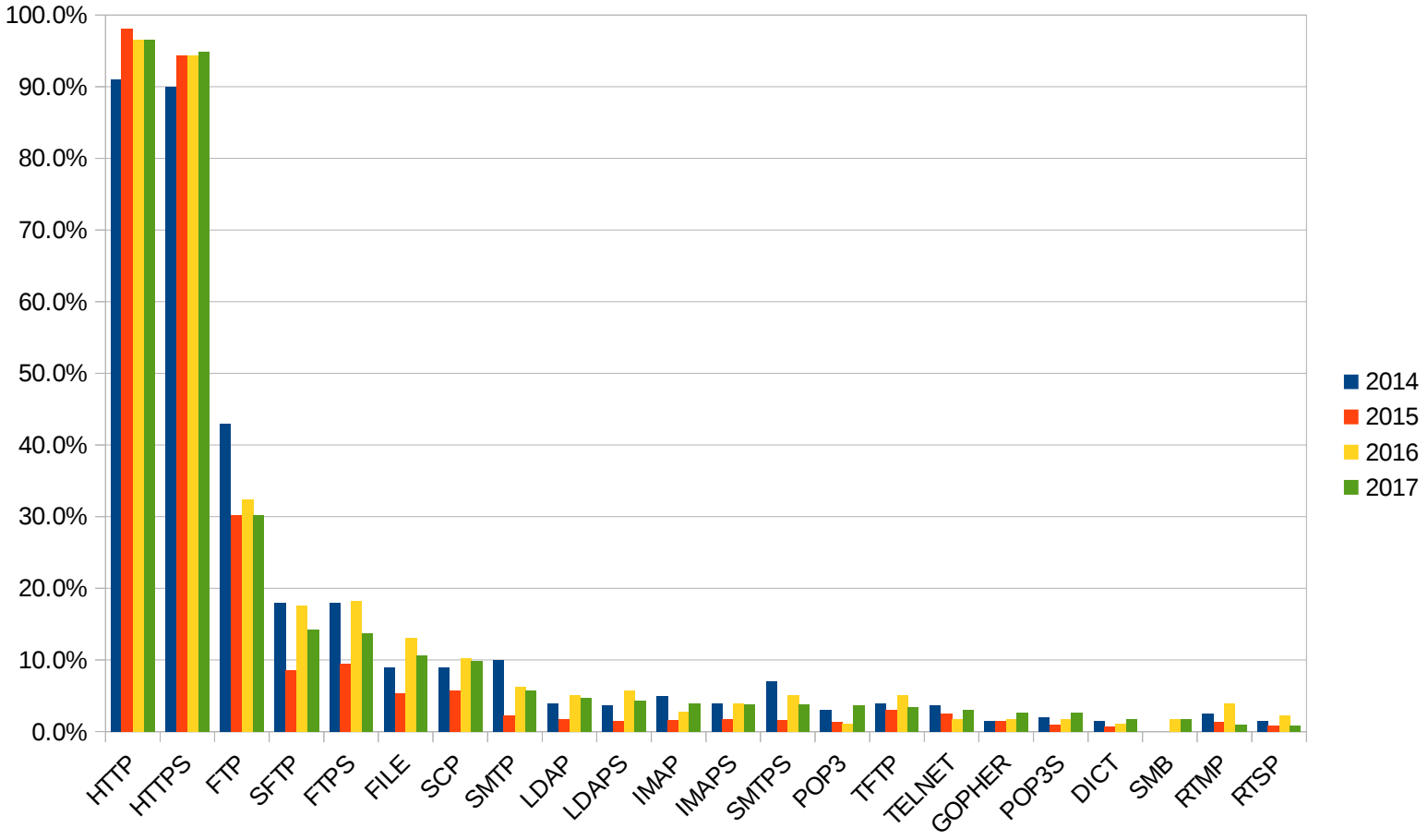
Can we detect a clear trend among protocol usage?

- No. There are possibly minor changes among some protocols, but the recorded movements are still too small to be significant.

This year's results plotted out next to the previous three years shows very clearly that the same protocols are widely used and the same suspects are the not so well used ones.

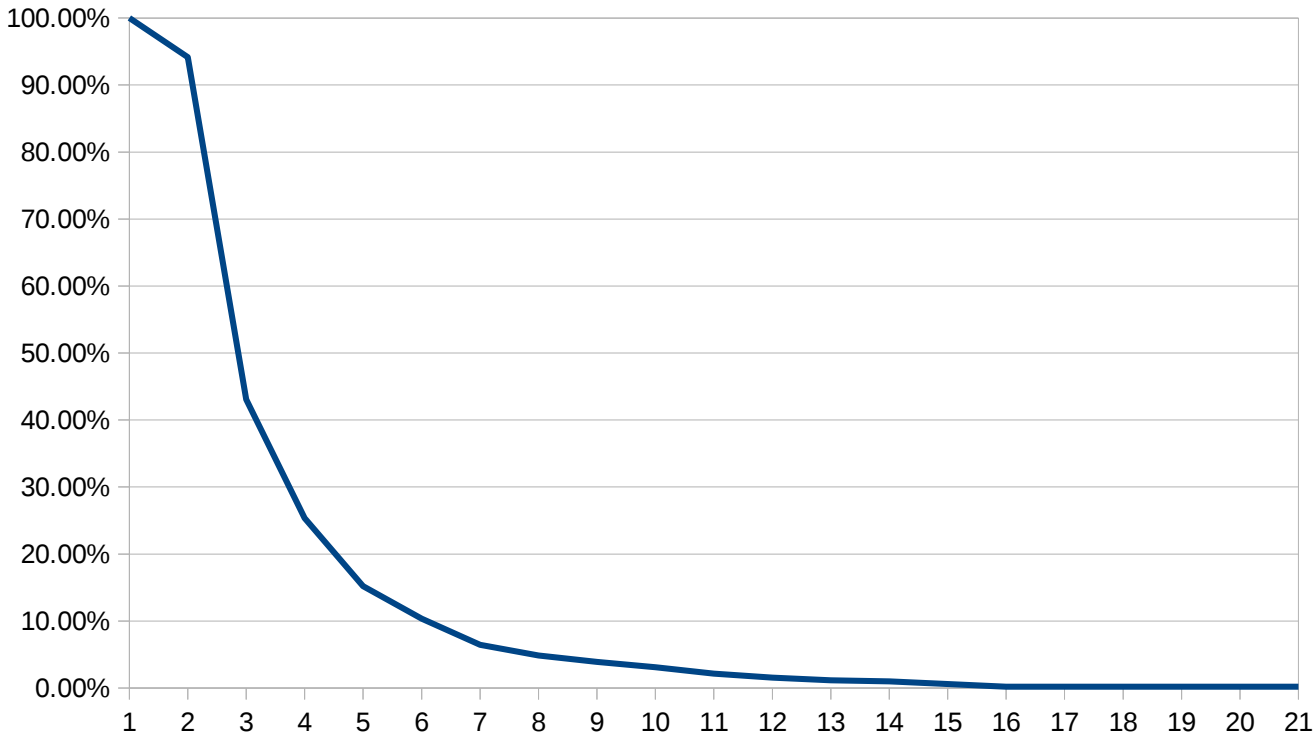
HTTP and HTTPS dominate (96.6% and 94.9%), as always, with FTP being a clear third (30.2%). SFTP (14.2%), FTPS (13.8%), FILE (10.7%) and SCP (9.9%) are all in the one in seven to one in ten range. The seven protocols quite commonly used. Then comes the rare ones. SMTP (5.7%), LDAP (4.7%), LDAPS (4.3%) and IMAP (4%) are on or above four percent. One user out of 25.

The number of protocols that landed on less than 4% usage, is half the number of listed supported protocols: 11. RTSP clocked in last at 0.8%. Down from 2.3% last year.



This year I also decided to count the amount of protocols each respondent filled in and this is what I learned: half of all users (262) use exactly two protocols, no users selected all 22 protocols but the maximum number of protocols checked was 21.

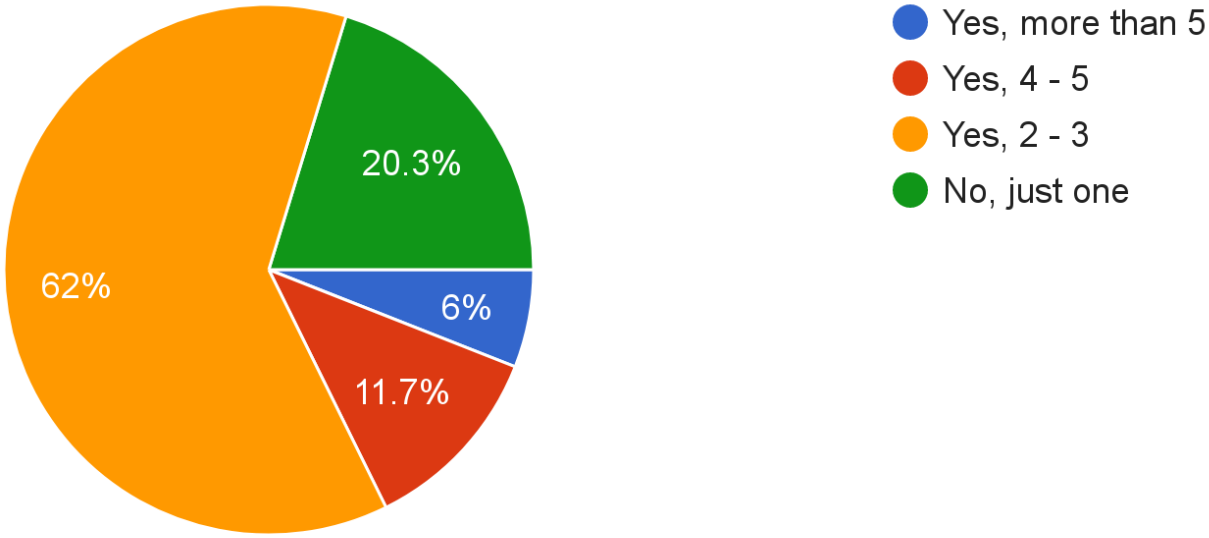
Looking at it differently, only 10% of the users use 6 protocols or more. We have to go to the 99th percentile to find users using 13 protocols...



SMBS was mistakenly left out from the list of protocols...

Do you use curl/libcurl on multiple platforms?

n = 503



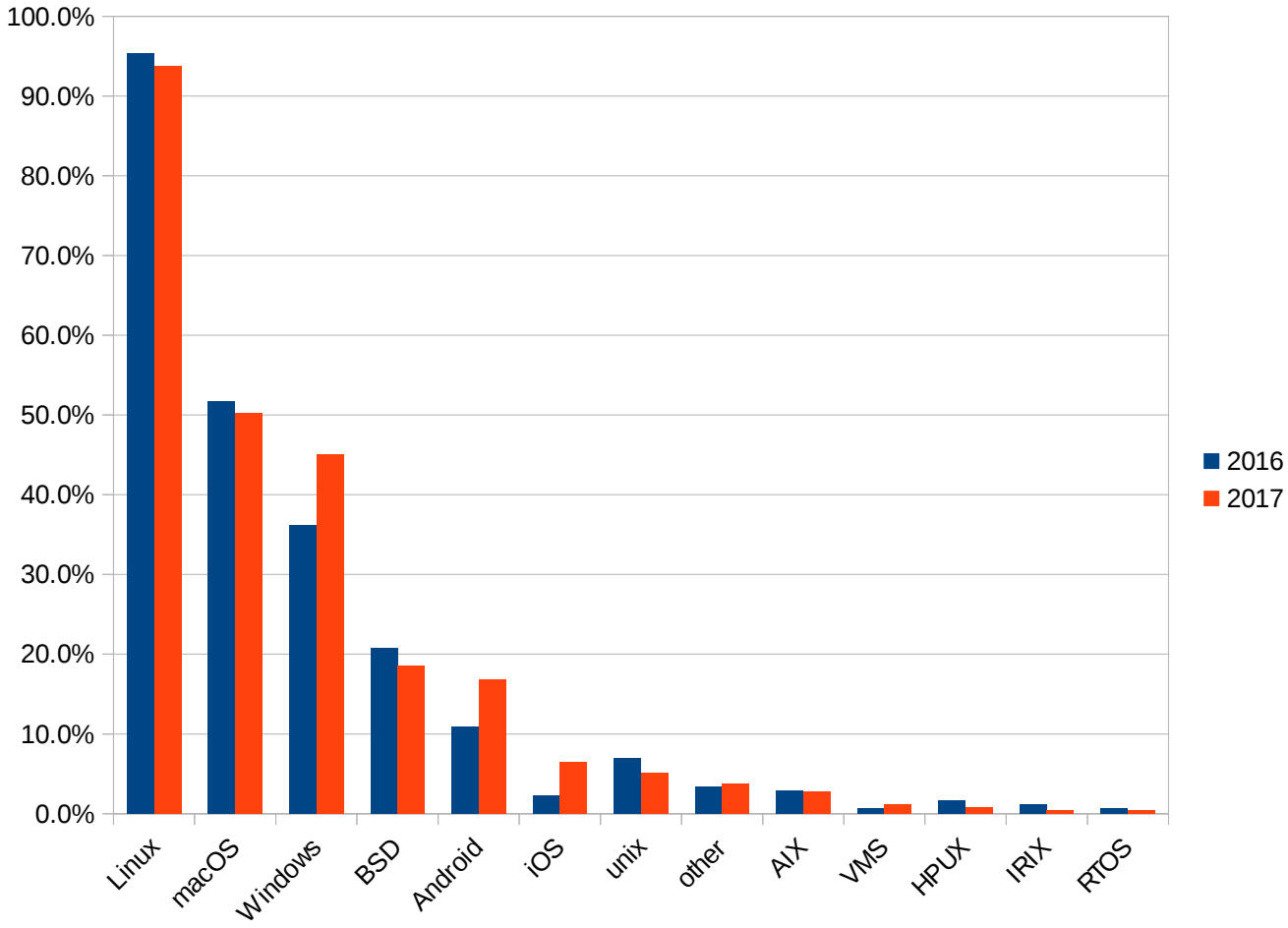
This distribution keeps looking almost identical year after year. See the distribution the last four years in the chart below and the stripes are close to horizontal:

You use curl/libcurl on which platforms?

n = 506

Here's a multiple select answer to a question that was new last year. The distribution was mostly the same this time. The notable changes were for Windows that climbed from 36.2% to 45.1% of the users, Android took off from 10.9% to 16.8% and iOS fired away from 2.3% to 6.5%. The amount of VMS users also rose 100% from 0.6% to 1.2% but with those small amounts of users (6 this year), that could just be the result of us reaching out to a more VMS heavy audience this time.

93.7% of curl users run curl or libcurl on Linux in 2017.



(The fact that Solaris was left out as a named platform is a mistake and should be corrected next year.)

Do you typically build curl/libcurl yourself?

n = 504

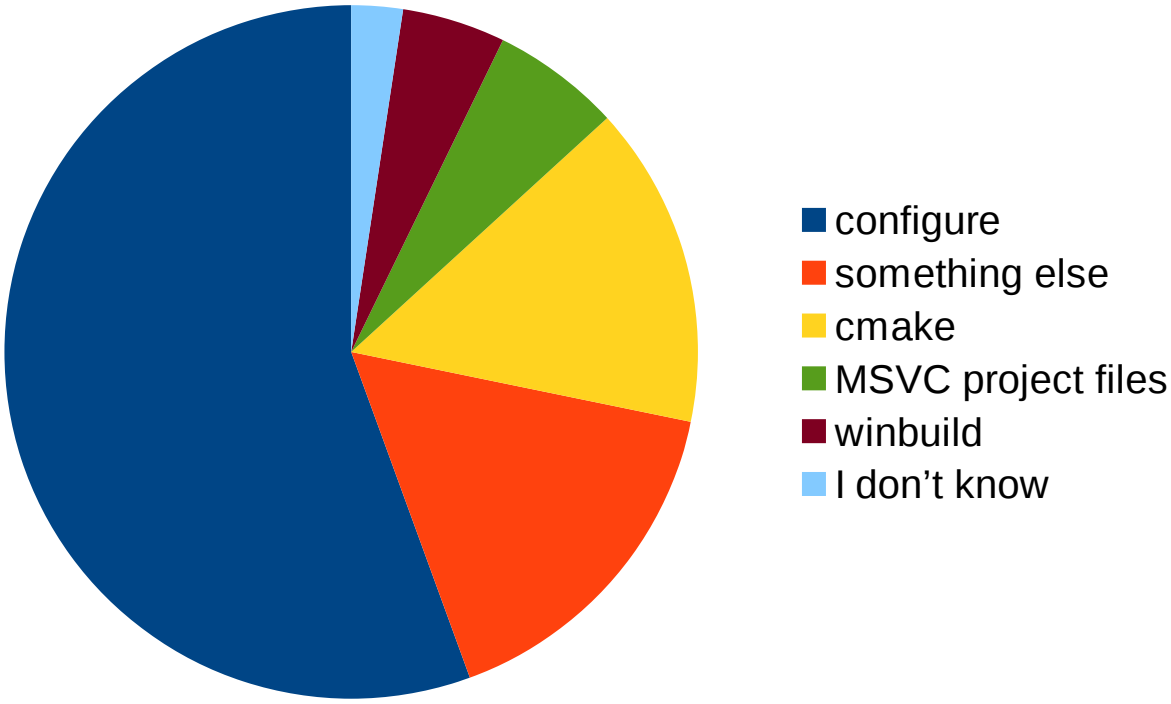
We introduced this question last year, but only as a yes/no question. This year we wanted to also get a better feel for what build system people are using if they actually do build curl themselves.

Last year 67.4% answered no, this year 66.9% did it.

Out of the 33.1% that didn't answer no, the configure based build system crushed the others with 55.9% of the yeses. A little confusing, the "something else" answer came at 16.3% ahead of cmake at 15.1%.

It was pointed out to me that people are in fact sometimes using different build systems on different platforms so the question should probably be made into a multiple choice in the future.

Put in pie chart, it looks like this:



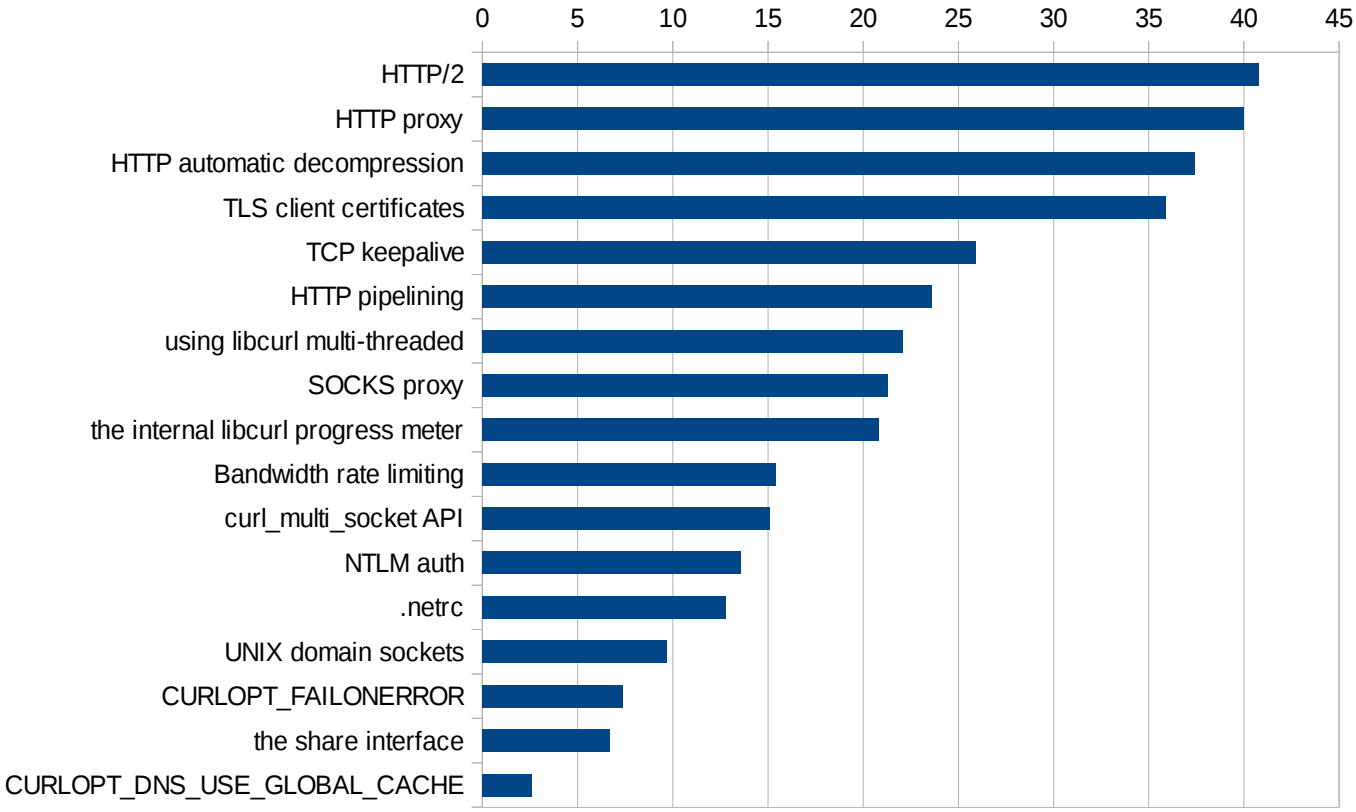
Tell us which libcurl features you use?

n = 390

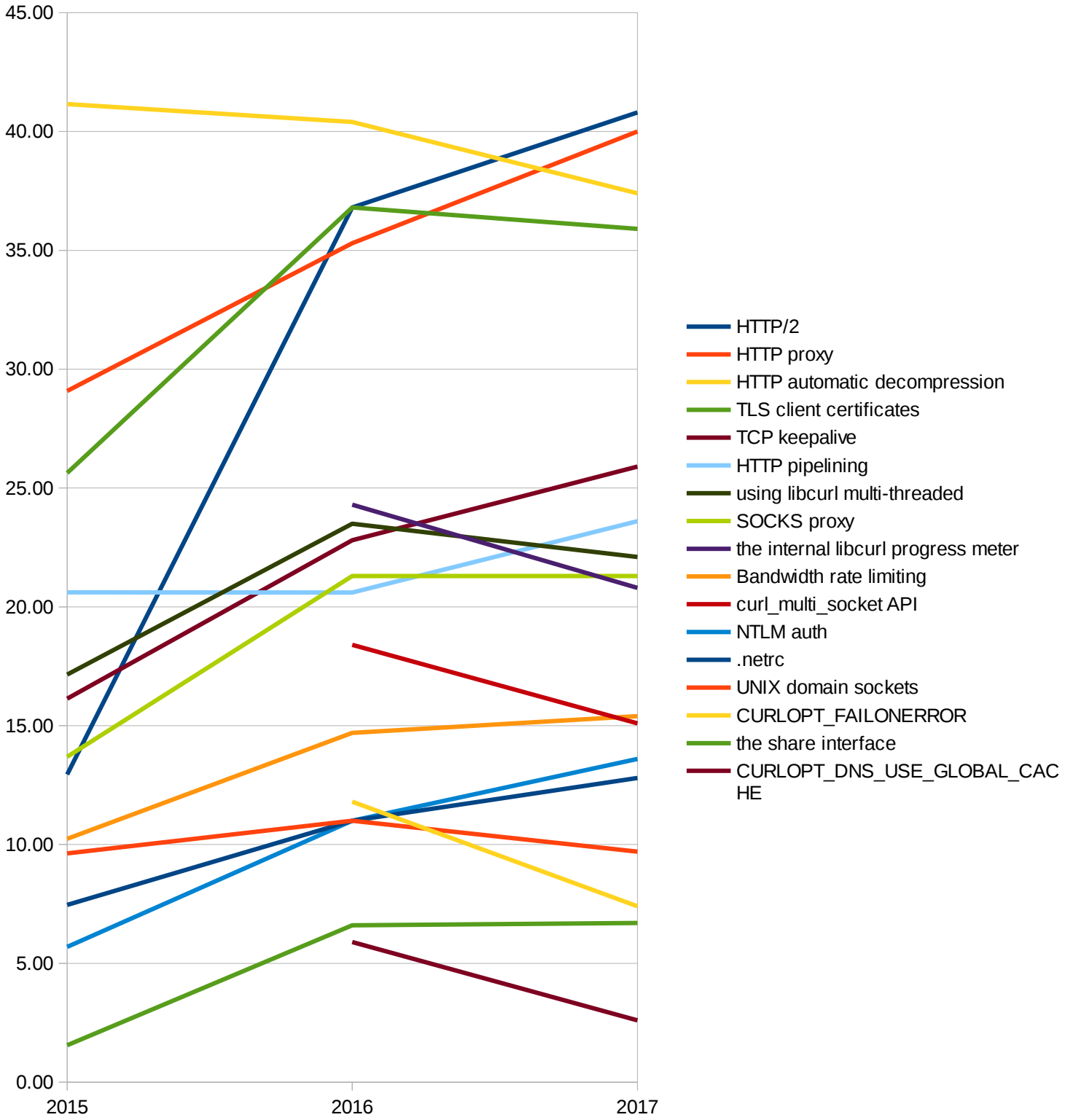
One of my favorite questions. Again it proves that whatever we provide, there is always some amount of users out there using it!

HTTP/2 rose to the top feature among the ones we ask about. The second graph shows the feature usage development for 2015, 2016 and 2017. For some to me unexplainable reason, HTTP proxy use has also risen notably over the last years. We added features to ask about in 2016, so some lines start first in 2016 and not in 2015.

The 2017 “I use these features” distribution looks like this:



As always, I question the amount of HTTP Pipelining (23.6%) and TLS client certificate (35.9%) users. Basically because those are rather niche features and I would expect us to get a lot more bug reports to the project if the usage levels really were this high. I suspect they get a lot of checkboxes marked by users who don’t fully know what the features mean.

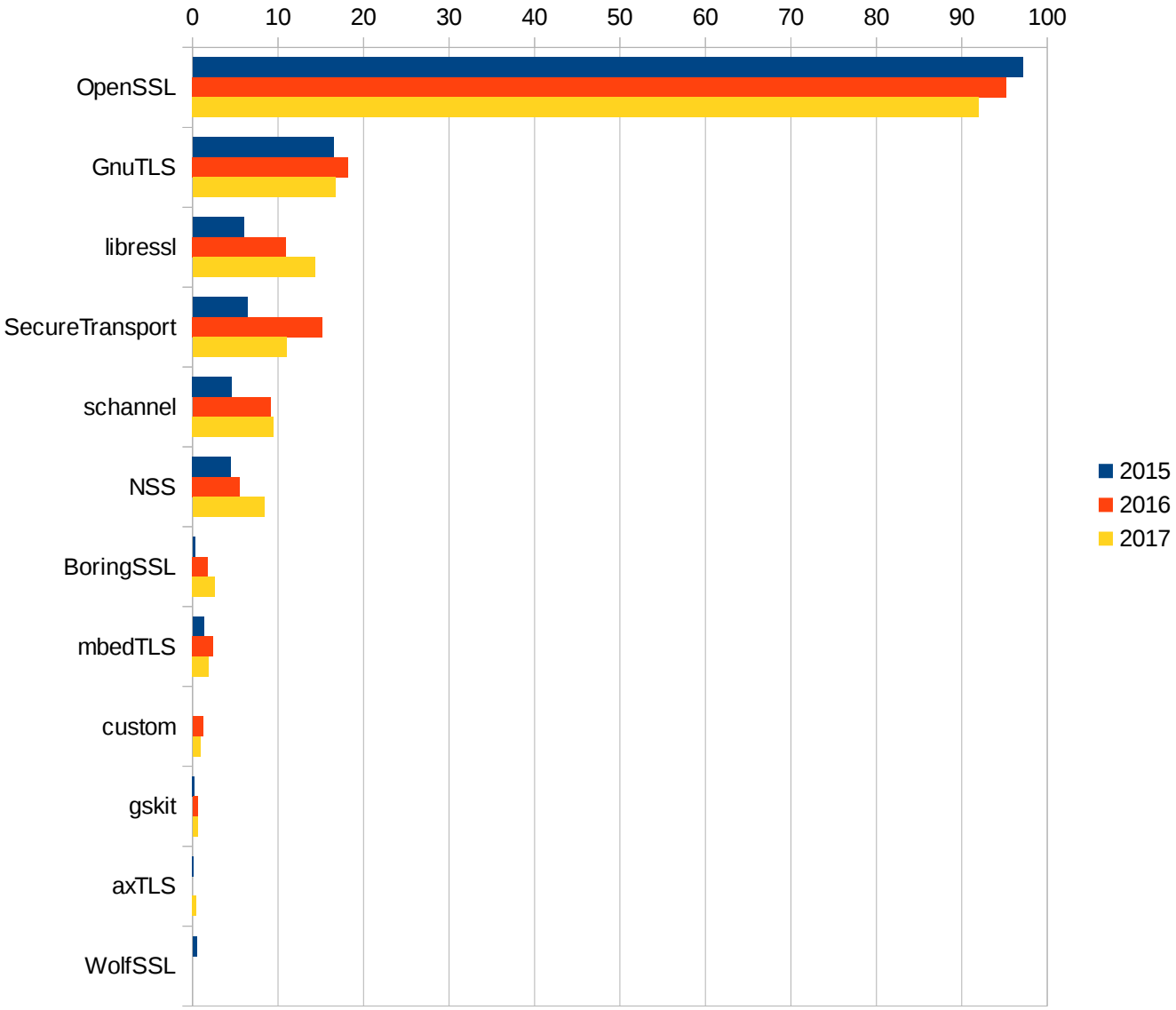

45.00
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.00
0.00
2015
2016
2017
HTTP/2
HTTP proxy
HTTP automatic decompression
TLS client certificates
TCP keepalive
HTTP pipelining
using libcurl multi-threaded
SOCKS proxy
the internal libcurl progress meter
Bandwidth rate limiting
curl_multi_socket API
NTLM auth
.netrc
UNIX domain sockets
CURLOPT_FAILONERROR
the share interface
CURLOPT_DNS_USE_GLOBAL_CACHE

Which SSL backend(s) do you typically use the most?

n = 462

curl is a world leader as the client which support the most number of different TLS libraries. Which libraries do users actually run with?

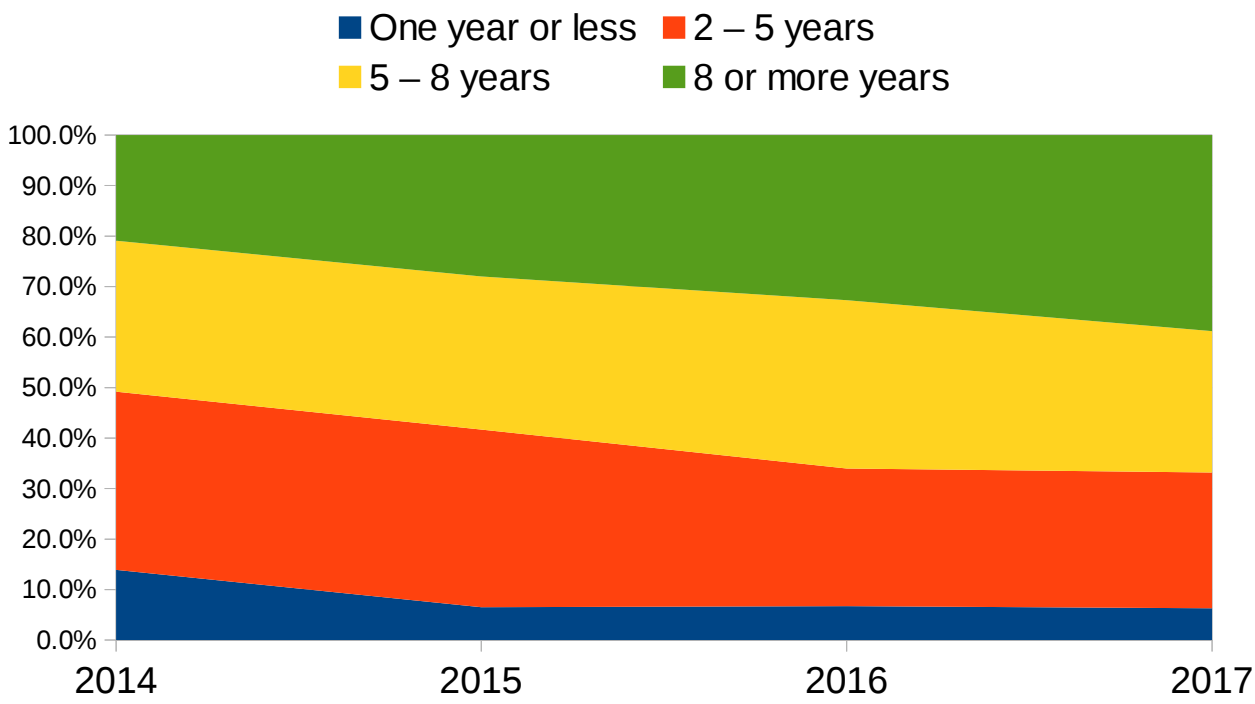
Answer: **OpenSSL**. At a really strong 92% it has stilled dropped now for two years in a row, even if only very marginally. From 97.1% in 2015 and 95.2% last year. GnuTLS remains the stable second most popular TLS library for curl at 16.7% but has now gotten a real competitor in libressl which has climbed the chart now two years and landed at 14.3% of the users in 2017. It also made it surpass Secure Transport (the macOS and iOS native TLS library), which took a hit and went from last years 15.2% down to 11% this year. Not a single user marked WolfSSL this year (again).



How many years have you been using libcurl?

n = 490

Do we get new users or are we all just stuck with curl since ages? The trend now shows that we're growing more and more users who have been using curl for 8 years or more as 38.8% of the answers were in that group this year. The distribution of the age groups over time is as follows below:



Select the channels you are subscribed to or participate in

n = 181

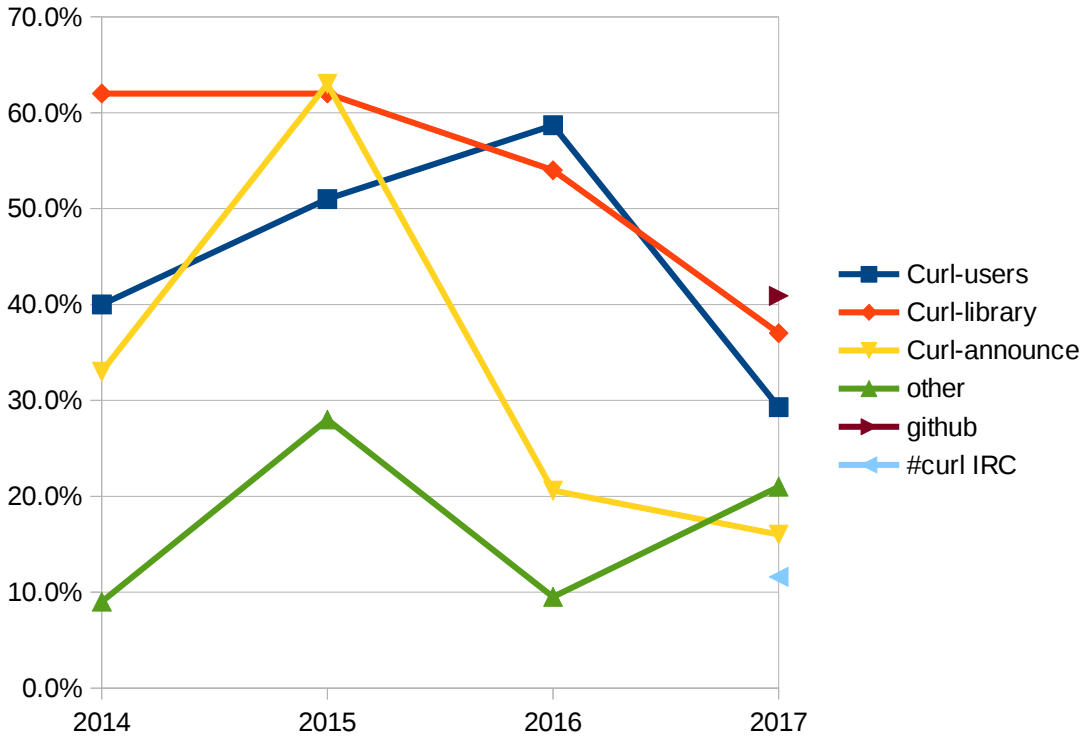
Just over a third of the total answered this question, which is telling. It was a similar fall-out in last year's survey. This year I added the github repo and #curl IRC channel as alternatives.

While this question seems to show a decreasing interest in the mailing lists, it is interesting to note that the number of subscribers to them has not shown any decline, but instead have slowly increased over the years. curl-users has almost 1,000 subscribers, curl-library approaches 1,600 subscribers and curl-announce has roughly 550 subscribers. It could indicate that the audience we reach with the surveys has shifted a bit away from the mailing list people over the years?

40% said they subscribe to github – which we know has 440 "watchers" now, and 5,060 "stars", but surely just starring a repository there can't be counted as "subscribed to" ? The least used channel according to the answers is the #curl IRC channel at 11.6%. Not bad I think since the channel is typically visited by only around 100 persons ordinary days.

The "other" choice was the only growing answer this year. I wonder what that can be. Stackoverflow?

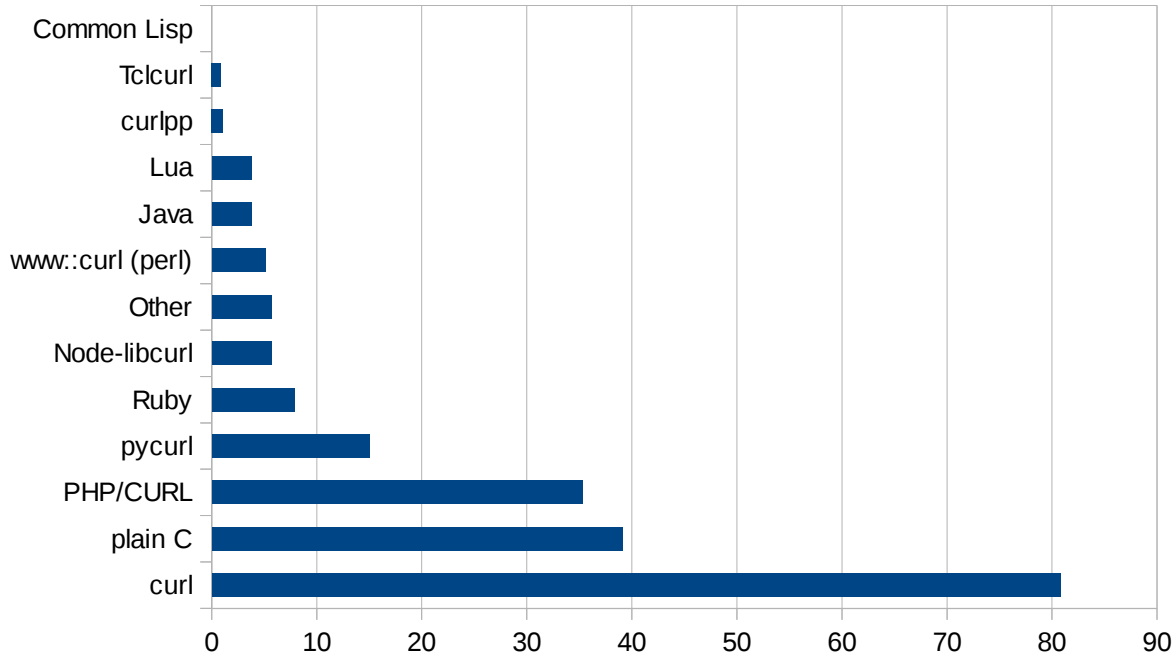
The fluctuations in how users have answered this question over the years:



If you're using a libcurl binding, tell us which!

n = 453

The binding popularity distribution remains almost static. The curl tool wins by far, the C API is the



most popular one and the PHP binding is the most used non-native interface. Nothing new here.

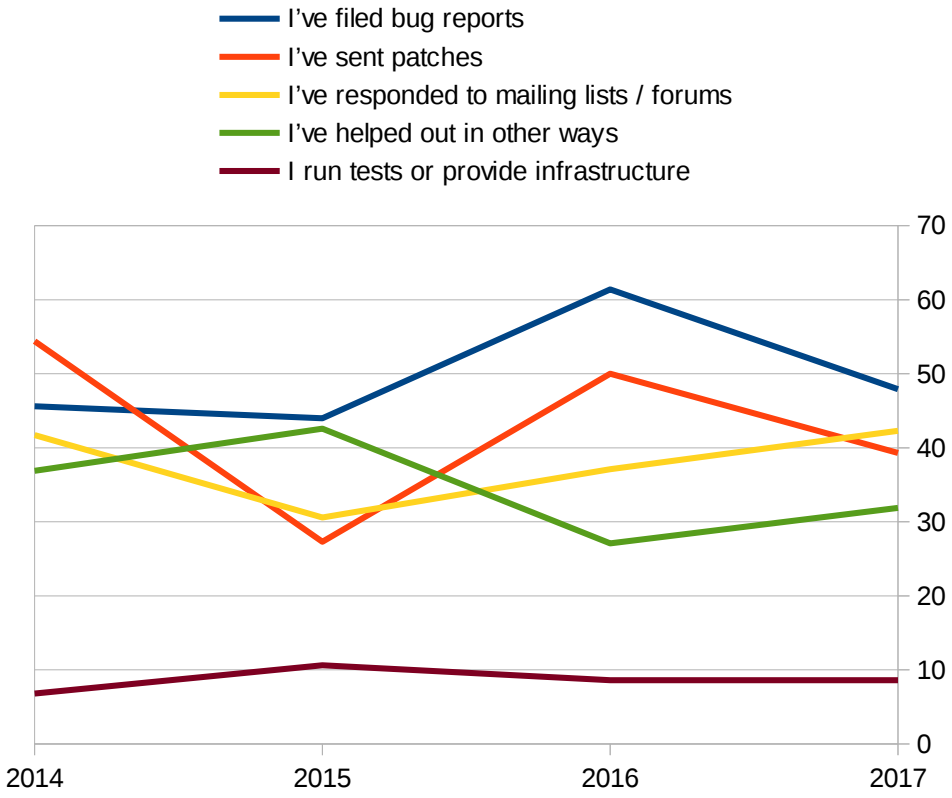
The “other” has grown to 5.7% this year, so I had to give it a look to see if there’s any runner-up that we should ask for next year (rust-curl and go-curl have already been suggested for that).

The other write-ins included the following bindings: Zabbix, Xojo, wxCurl, R curl, Stata, Rust-Curl, RCurl, PHP/Guzzle, perl5 code, pecl-http, Nim, .NET Core's System.Net.Http implementation, backed by the C API, https://github.com/stil/CurlThin, ILE/RPG, Pascal binding, haskell, https://github.com/andelf/go-curl, erlang github.com/puzza007/katipo, custom c++, wrapped with SWIG for python, Custom, Curl Plugin (Squeak), Curl for People: https://github.com/whoshuu/cpr, curlcpp, C# (HttpClient) – a rather impressive list.

How have you contributed to the curl project?

n = 163

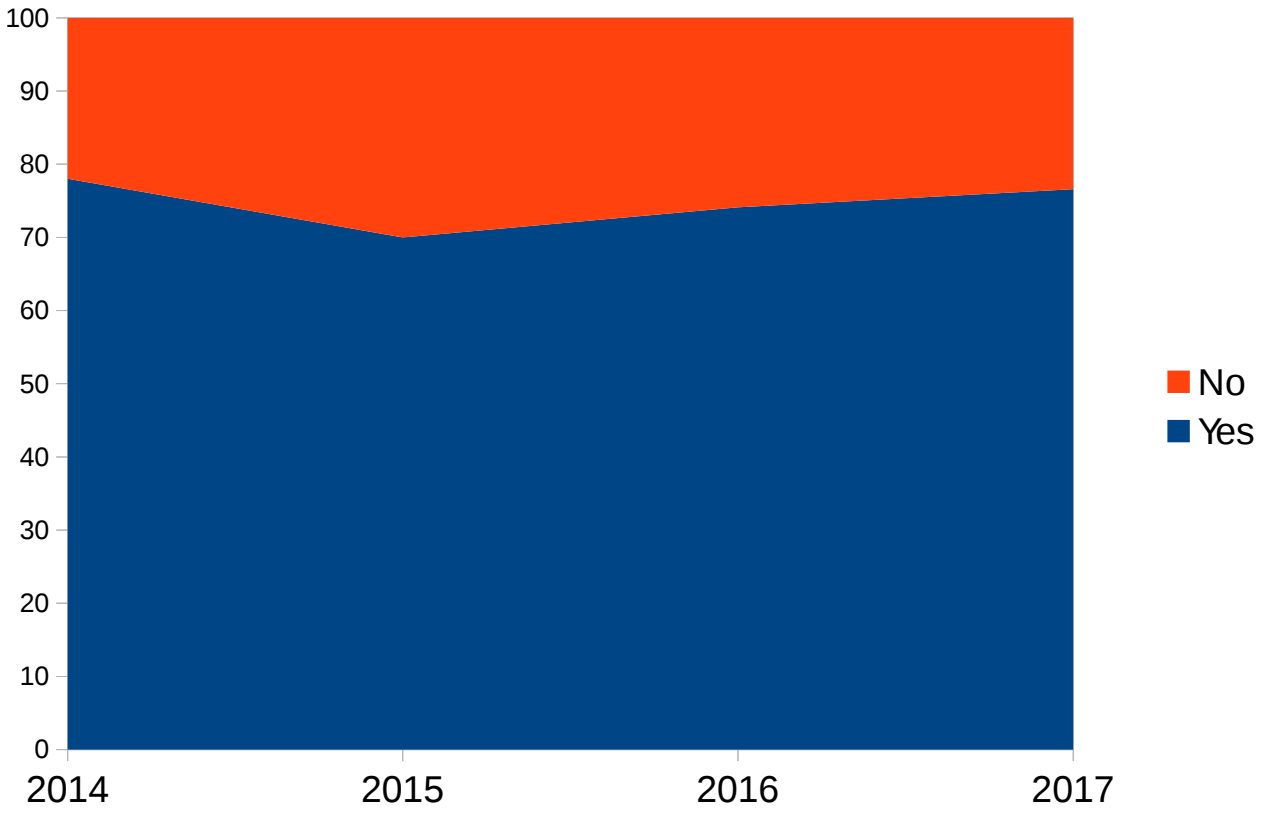
Again less than a third answered this question, which I suppose is because the other two thirds haven't contributed.



Are you involved in other open source projects?

n = 496

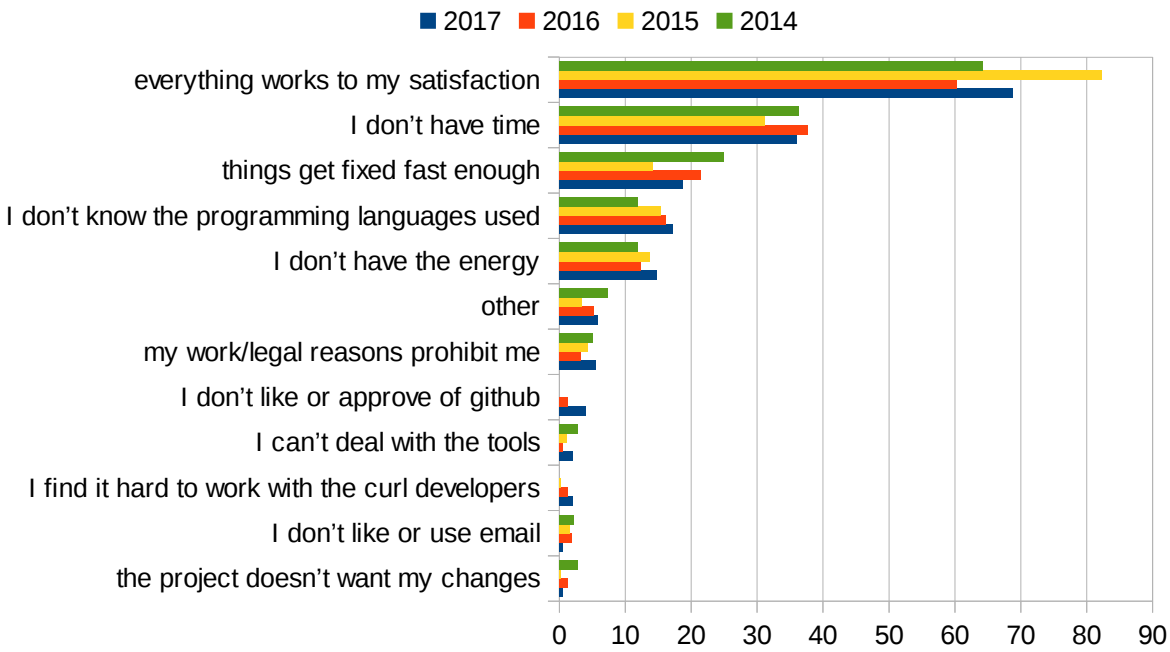
This year’s 76.6% yes is pretty much in line with previous years, as this graph shows clearly. Curl users are typically involved involved in other projects too.



What are the primary reasons you haven't contributed or don't contribute more to the project?

n = 481

The primary reasons for why people don’t contribute (more) to the project are still the ones that from a project’s perspective are pretty good. The people who don’t like github grew this year (4%), and unfortunately so did the “I find it hard to work with the curl developers” (2.1%)



The write-ins for “other” included:

- work on packaging
- Too many SSL backends to deal with to implement something
- Not sure where to start.
- Not experienced enough
- Mutually exclusive requirements. Developer doesn't want one big patch implementing a single feature, nor wants smaller patches which individually don't provide new features. I don't know how to accomplish these mutually exclusive goals.
- Most of my work belongs to the companies I work for and contributing to OSS is kind of annoying from within private firms
- limited use of curl so don't know much about it
- Lack of domain-specific knowledge
- Just started learning it.
- i use curl because old computers have it installed. new development doesn't help much
- It's hard to start working on a big project
- it's a bloody difficult problem that no-one wants to touch.
- It is hard to notice any bugs, I fix only the little ones which I am able to find.
- I never considered to contribute to curl, I consider myself an end user of it

- I'm not good enough in coding
- I have not yet felt the need to participate, while I am already participating in other OSS projects
- I have not been able to figure out git.
- I don't feel good enough
- I don't enjoy development
- I do not see the reason to contribute.
- I am unsure how to help, it seems very mature already
- I am nowhere near expert enough in most of the technologies covered, to compare with the people who maintain curl currently.
- Haven't contributed to OSS before
- don't understand fully how secure connections work
- don't have a chance yet
- curl just "works" for everything I need it for.
- C is annoying to do safe memory management. C++ STL objects would make things easier.
- building on multiple platforms scares me

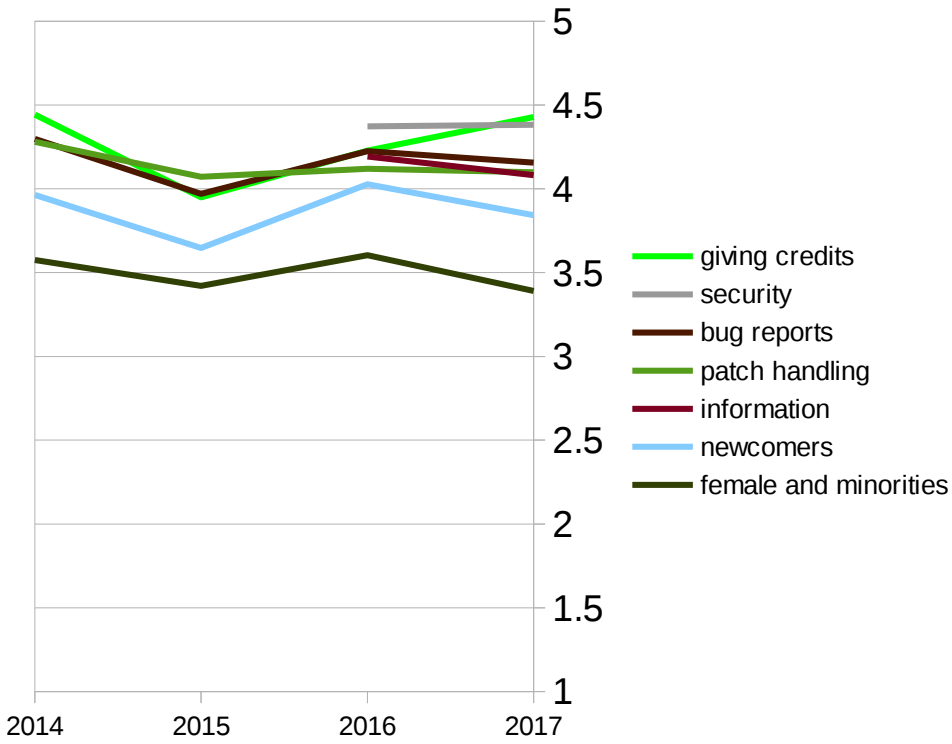
How good is the project and its members to handle…

Users are asked to grade the project and give us a score on 1 to 5 for seven different areas. 5 means very good, 1 means very bad. All fives would give an average of 5.0, all ones would give a 1.0.

The seven different areas the users were asked to grade are:

1. Patches and pull-requests (score: 4.1)
2. Bug reports (score 4.2)
3. Female contributors and other minorities (score: 3.4)
4. Attribution and giving credits (score: 4.4)
5. Helping newcomers to the project (score: 3.8)
6. Information about what's going on (score: 4.1)
7. Handling security-related issues (score: 4.4)

The total average score 2017 is 4.05, down from 4.11 last year. Not a very big change, and looking at the scores‘ development over time they seem to not move very much. Either we don’t improve or degrade much in the project, or even if we do, the users think we’re doing pretty much the same as before:

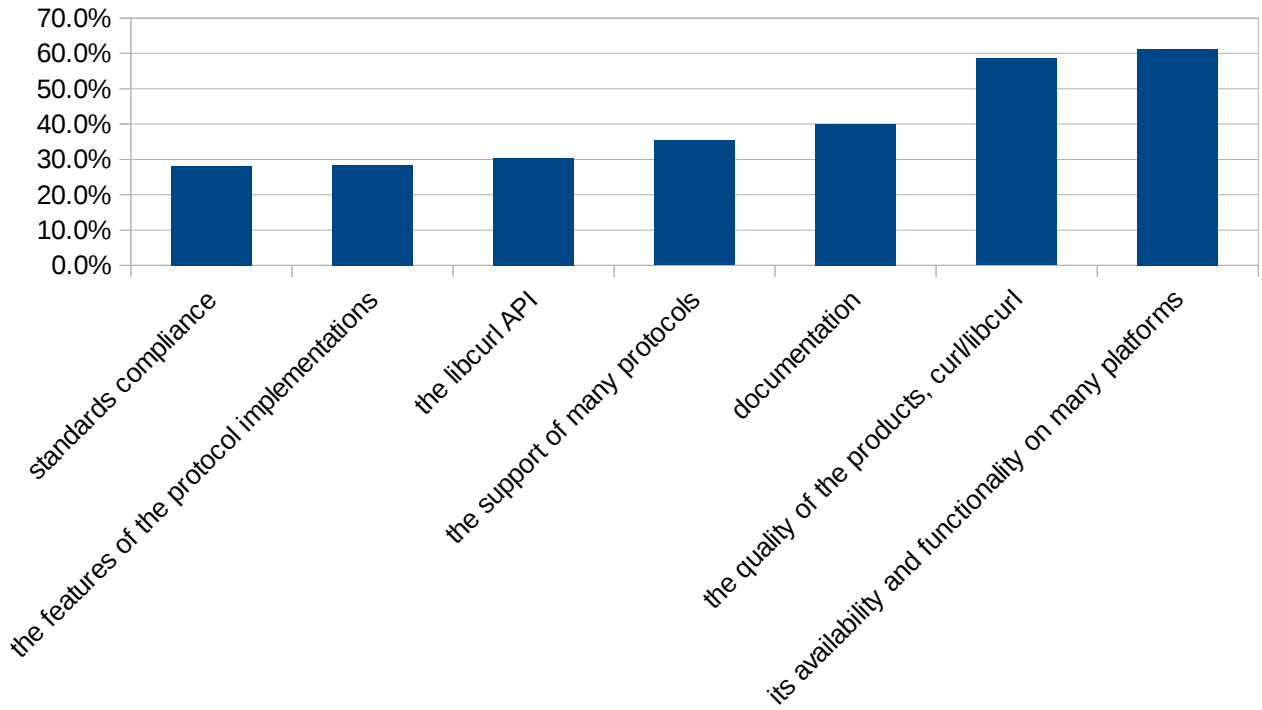


There can be little doubt that our weak areas include catering for minorities and newcomers, but in general I think we do fairly well.

Which are the curl project's best areas?

n = 447

Asked to pick the three best areas of the curl project, these are the top areas 2017:



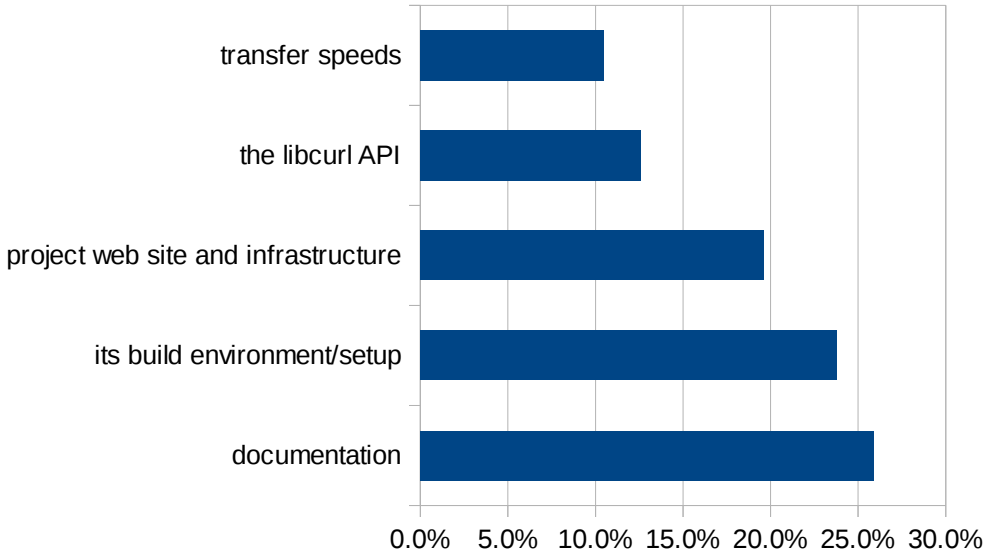
Since 2014, documentation has risen steadily and reached 40% this year. The "support of many protocols" has on the other hand shrunk every year from 55% in 2014 down to 35.3% 2017.

This question is of course very interesting when put next to the following question, which asks for the "worst areas" with the same areas being offered.

Which are the curl project's worst areas?

n = 143

Only a third as many answering on this as for the "best areas" question. The top-5 voted worst are:

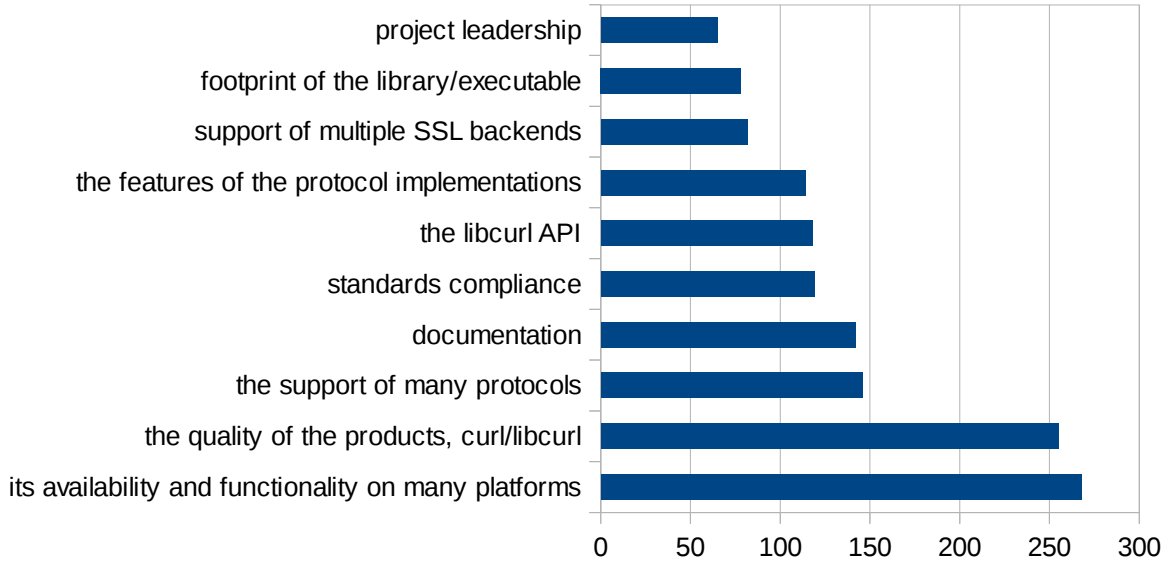


Interesting changes over the years: documentation is now voted worst at a lower rate than in the past, on an all time low at 25.9%. Hopefully this means the last years' hard work on improving the docs has paid off.

The last years' work of fixing the build environments have however not paid off to the same extent since it has become a growing worst. It was down at 9.7% worst in 2015 and has now bounced up to 23.8% in 2017. The web site remains at this level – hinting that the last years' site improvements didn't do much to people.

It remains interesting that documentation is the top-worst but at the same time ranked the third best area. Of course, counted in plain numbers, 37 users voted documentation as worst while 179 marked it for best.

I did an exercise and subtracted the "worst" votes from the "best" points for each area (by absolute number of users who marked it) and the top-results of that are still almost the same as the best top list:



There are only two areas that received more worst votes than best votes by absolute numbers:

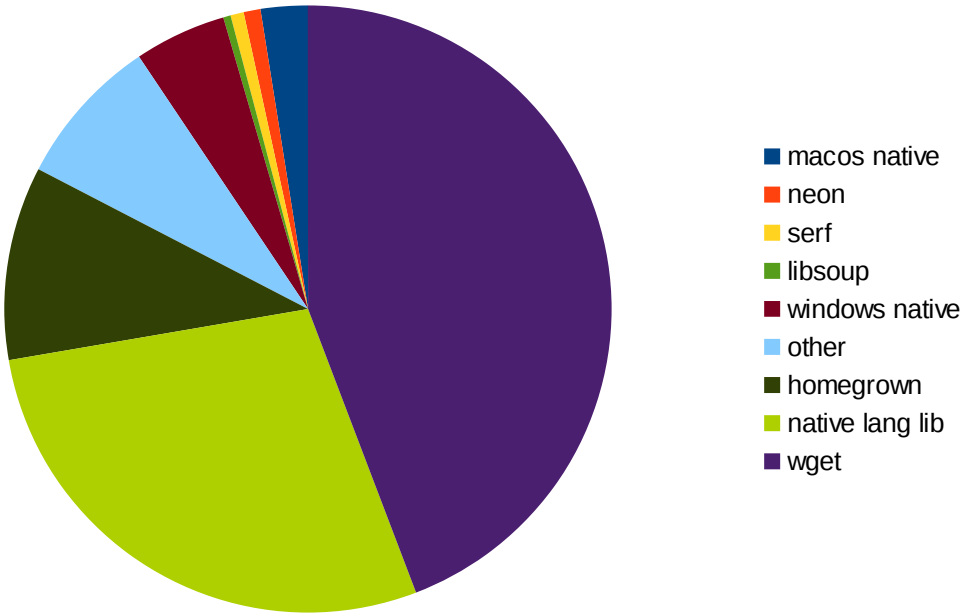
Area	**Best – Worst = score**
project web site and infrastructure	15 - 28 = -13
its build environment/setup	13 – 34 = -21

Our two least favorite areas…

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

n = 448

The answer proportions to this answer remain very similar to past years.



Wget code remains the clear winner: 44.2%, native lang lib at 28.1% and homegrown at 10.3%

The “others” category at 8.0% is interesting so I had a look at what people wrote in for that to see if we should add some of those as choices for next year’s survey:

- Anything written in memory-safe languages
- Argh!!!!!!!!
- aria2
- BSD's ftp if only it supported TLS it already supports HTTP
- cpp-netlib, asio, Qt, POCO
- Don't know
- Go net/http
- h2o
- httoop, netcat
- httpie
- Hyper
- I'd try pretty hard to make libfetch work. If not, I'd sigh and roll my own.
- is there alternatives?
- I would be in deep shit
- I would be in despair
- i would cry
- native unix clients assuming TLSv1.2+ support
- neon, httpie for cmdline

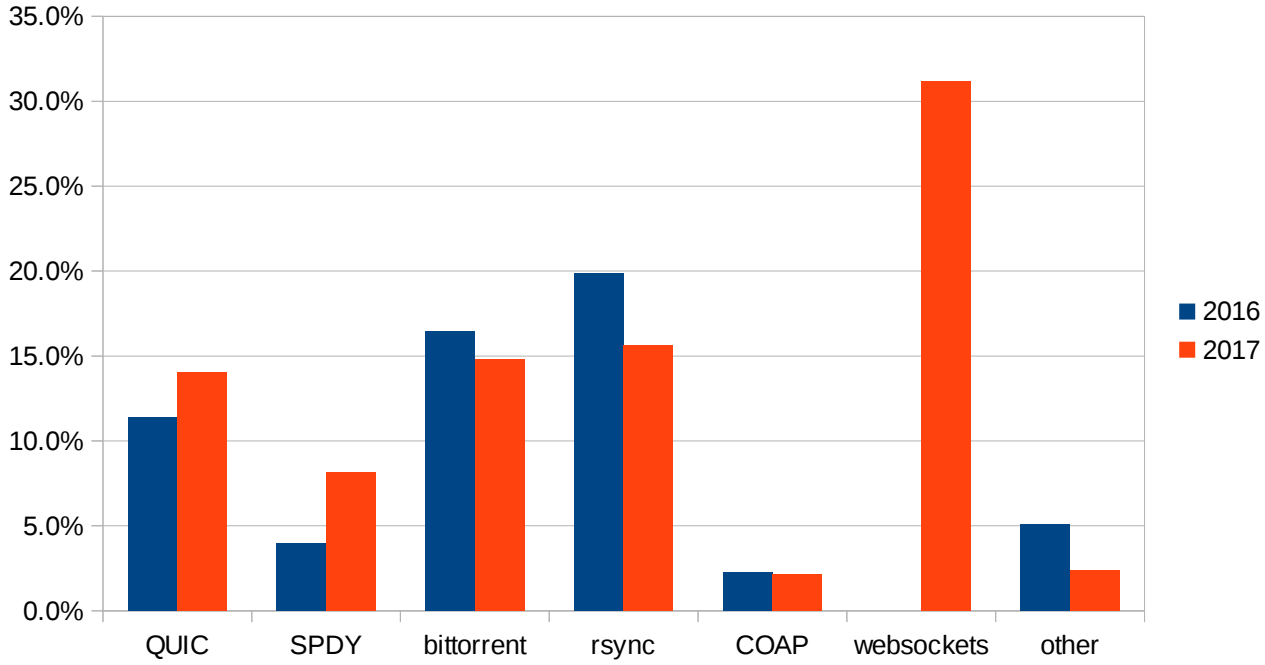
- .NET System.Net or System.Web
- none, they all suck! (well typically don't support h2)
- POCO C++ library
- Postman
- prefer native, but a lot of the time protocol features are missing Qt
- Qt Networking Classes
- Really Vince
- Requests (python library)
- SSH and its scp interface
- telnet, netcat
- Totally blown out of the water
- write it myself

If you miss support for a protocol, tell us which!

n = 281

Ah, one of my favorite questions. This question ran last year as well, with the addition of websockets an alternative this time. Last year 38.6% of the respondents marked at least one protocol they want added, and this year 54.8% did. Clearly an increased number of people. The graph below shows percentages as a total of the entire respondent population. An amazing 31.2% of the 513 respondents actually marked that they miss websockets.

rsync (15.6%) and bittorrent (14.8%) are less requested this year, but I still consider that a fairly high demand for both of them.



The write-ins this year included:

- webdav
- SNMP
- sieve
- Plain TCP like netcat
- multicast QUIC
- metalink (concurrent download)
- I've basically never used rsync directly (not much anyway), but if curl had it, I would! :-)
- IPFS
- IETF equavalant of QUIC would be useful when it's ready and gets adoption in browsers, webservers, etc.
- HTTP(s) over SCTP
- HTTP Alternative Services
- CAN

What feature/bug fix would you like to see the project implement next?

n = 76

This is a free form text field so this is not easy to summarize. It is however a “let it all out” opportunity for people. This time I’ll try to comment on requests.

As a security maintainer, I'd like more info on security issues in a timely manner... or permission to join oss-distros

The whole purpose of oss-distros is to get early warnings about pending security advisories. We’ve been considering our own pre-notify list but we’re not happy of the maintenance burden that brings, or even setting up the definition of who would be allowed to join and how to keep track of that. Not easy. We typically notify oss-distros two weeks ahead of a security release.

Keep the project alive :-)

This is probably our most important feature so let’s try to do this very good going forward!

Better reporting of transfer codes/status/errors to scripts using the command-line tool.

In my mind I was under the impression we already do report errors back pretty decently so I would need this explained further. We’re always up to improving our code, error reporting and the curl tool!

I would like to see shorter (more compact) syntax.

For what? The command line tool? I don’t think we can do it much more compact and still be backwards compatible but if you have ideas on how, just bring them on and we can work on it!

You guys really need to add support for digest authentication with qop=auth-int. My company's APIs use it for all their services, and it's difficult that we are unable to use curl for testing or to provide example usage.

Also finally completing the HTTP digest support with auth-int in all cases, getting info about rspauth validity, and additional hash algorithms supported.

5.4? (HTTP Digest using SHA-256)

Patches accepted!

Specifying SSL backends at run-time

That feels like opening up a very large Pandora’s box that I rather keep closed. I would like to first understand what the benefits of this would be and why not working on fixing the problems with specific SSL backends isn’t the better answer. Or just switching to the SSL backend you think work better...

detect raw binary data and don't dump it on my interactive terminal

There’s work in progress on exactly this feature right now. Might land in a curl coming release in not too distant future.

Replaceable transport backend like for SSL. For example to replace sockets transport with winapi-based pipe transport and then to tunnel http packets over the winapi pipe.

An API which is independent of sockets. Meaning I can just push binary data into libcurl and receive from libcurl and will handle the sending via socket (or something else) myself. This is similair of using UNIX-Daomain-Sockets, but thise are still sockets. I want to have just arbitrary Memory-Buffers

That sounds spaced out. Bring your ideas on how to do this to the curl-library list and we'll talk!

Some sort of easier OAuth/SAML/OpenID Connect integration for API authentication

Some sort of patch that improves the situation accordingly will be appreciated!

Feature-complete CMake build

I think everyone wants that. It "just" takes for people to fix the remaining issues and missing pieces. You can help!

Better documentation on which API calls can/cannot be used in which callback contexts

Agreed. We have heaps of documentation already but it's like there is never enough and it can always be improved. As a general rule we don't call libcurl APIs at all from within libcurl callbacks...

Make test suite all green everywhere / 100% false-positive free test suite for Solaris

Believe me, we all want that. It's a never-ending fight and really hard work. We can certainly use all the help we can get with this task.

Use a single TLS session ID for FTPS control and data

It mostly needs to the internal session ID store and its internal API to be modified a bit. It'd be lovely if someone who has this need and a server demanding this could grab this task...

Increased focus on security and code size reduction.

We already have a very strong focus on security. If you can think of other or better ways we can improve the security of curl and libcurl, then please tell us. Regarding code size we work on two areas: 1) write effective code and remove things that aren't necessary or needed and 2) allow custom builds to disable specific features in order to reduce footprint. But I don't think we have any magic tricks up our sleeves that will let us drastically shrink the libcurl code size going forward.

multiple, parallel connections to a single host to increase transfer speeds

This has to concern the curl tool. We need to convert the curl tool to use the multi interface instead of the easy interface – which really isn't *that* big challenge, and then we can start doing parallel transfers. We've discussed this back and forth before and we mostly need someone to decide to "just do it" and plunge through.

Have an "extra cookie" option so you can add cookies to an existing -b option. The reason is I very often use the "copy as curl" option from browsers' dev tools, but I want to add my own cookies to the request, and I like to be able to add stuff at the beginning/end of that LONG "copy as curl" request.

Sounds like a decent idea. Just do it, and send us a patch!

Auto-naming output filenames like wget maybe

Patches accepted!

stdin command execution

We brain-stormed a bit around this idea at the curl://up meeting and yeah, this would be really cool...

test-suite running in parallel

I'd *love* that!

Strict SCP does not work with Strato HiDrive, while SFTP does work flawlessly

Honestly, that sounds like a libssh2 issue...

keep it stable and reliable. no feature creep pls.

We should indeed keep it stable and reliable. That's our main purpose. But we also need to listen to our users and keep our products relevant and matching the needs and desires "out there".

SON Object Handling

download all web page elements

That implies HTML and CSS parsing...

curl_easy_escape returning string length

Wow, really?

Support for callbacks in non-OpenSSL SSL backends

Everything that makes the different SSL backends behave more homogeneously is good!

x over SCTP

My experience with SCTP tells that it is usually not just a matter of changing the socket type and then everything is merry and dandy so changing protocols from TCP to SCTP is likely to take testing and verifying. And then we land on the lack of specifications for how that should work. Probably also a slightly lack of server side support too. But I wouldn't be totally against such a change if done with care and proper attention to details.

Fine gained credential management, ideally integrated with ssh-agent / gpg-agent

I'm intrigued, but I don't think I fully understand what it means.

Directory listing in file://

Just do it!

Fix blocking in SOCKS proxy code.

Patches welcome!

Better Mac OS X support

Better support of what?

unified universal configuration/build system

I'm not aware of any such system. The one ring that in the darkness binds them?

ability to manage connections directly (force specific connections to close; force new connections to be opened; use specific connections for specific transfers)

That'd be interesting, yes!

Sftp speed

The recent buffer size changes should at least make things a lot better in this department. The curl tool as of 7.54.1 will be notably faster for SFTP than before. But yes, it can be improved much further too from there…

--json, so I don't have to remember the header I need to set

We've discussed various json specific features at times but they've never materialized...

Fix all the binary comparability bugs especially involving the uppercase versioning symbols. Usually curl is transparent to me except when games use it and this happens.

What bugs?

Even better tracing and explanations of SSL connection problems

I'm with you!

curl can almost NEVER download a file with the "right name" with -O and -J

There are certainly still features and functionality lacking there, especially when the Content-Disposition is URL encoded etc. Room for people to help us out!

more user friendly manpage and --help output

We've discussed splitting it up somehow and showing only the parts the user asks for or are interested in at that moment, but nobody has had the energy and will to pull that through. We can use your help here!

I think that test suite should be extended to allow real binary data transfer, independent from the system encoding.

And it doesn't do that already?

Alt-Svc - as a bootstrapper for QUIC

Agreed, when we support QUIC we should probably also allow “bootstrapping” to it via the Alt-Svc header as that’s the official way to do it. Let’s get QUIC working first...

Compressed HTTP POST body

This is a lack of specification problem. There’s no way in standard HTTP to compress a request body.

I would like to see all transfer numbers in unsigned integers, not doubles. CURLINFO_SIZE_DOWNLOAD is annoying.

There’s a pull-request in progress right now doing this.

Support for BearSSL backend

It’s not that hard to add support for a new SSL backend. Go ahead!

off thread async api

I’m interested in venturing into this territory and discuss what can be done and how. I believe one of libcurl’s current bottle necks on modern architectures is exactly that it has no native support for multi-threading but instead the application using libcurl must do it “manually”. I would be interested in seeing what we can do to help here.

easier integration with libevent / other event loops

Easier than multi_socket ? If you can suggest such a way, I’ll gladly listen!

libcurl needs a way to get a filename to use for the downloaded file. It's annoying to add your own parser for Content-Disposition

I can sympathize. Now how would this work?

More examples of usage

I’ve worked hard over the last month or so and I’ve added I believe almost 100 examples to libcurl option man pages. We have almost 100 stand-alone example source codes. But yes, I’m sure we can always add more. This is something everyone can help out with as well. Not only by writing examples, but also by suggesting what sort of examples you think are missing!

Have had issues with Pipelining, so have had to remove support for using it.

We have bugs. HTTP/1 Pipelining is really tricky to get right and has never reached a really stable state in curl. HTTP/2 is the newer, better and more reliable way to achieve the same result. Also, browsers will stop supporting Pipelining.

Maybe bundle an up to date SSL/TLS library like H2O webserver

Sorry, we have enough work as it is. We cannot take on a full TLS library as well. That would be totally irresponsible and wrong. The best TLS libraries are those which a lot of skilled people work on and use.

Improved bandwidth throttling for multi handles (cap across all handles, don't read entire buffer from socket if it would go over bandwidth limit)

Doing bandwidth capping based on more than one connection is a challenge to get “right”, or we need a lot settings to control it. Bring your ideas and descriptions to the mailing list and we can discuss it!

What feature/bug would you like to see the project REMOVE?

n = 28

Only a few people could think of something they want us to remove.

Automatic determination of the features curl gets compiled with: use opt-in instead of opt-out and error out if something's missing instead of dropping the feature silently.

I assume this talks about configure behavior. The features aren't really "dropped silently" since the summary at the end shows exactly what it found and how it will build curl. What should be opt-in and opt-out there is of course subject for discussion and is a matter of personal preference. By a very long standing unix tradition the configure script picks "sensible defaults" by itself if it can. Of course it can be discussed and tweaked going forward. Give us the pros and cons and let's talk about it!

I'm just slightly worried about the wide surface of libcurl. I really only want HTTP(S) requests (GET, POST, multipart/form-data please), but I don't want to make protocol mistakes and that (S) is really challenging to implement. I don't ever want to be in a position where a malicious server can trigger an obscure feature in curl that I've never heard about and compromise my application.

A fair point of view I think. Nobody wants to be "dragged down" by features they don't use and code they don't use. In the curl project we should strive to minimize code amounts and always pay attention to security and wring safe code. When we get security issues reported, we work hard on fixing them in the next pending release and provide clear and accurate information both about the flaw and the cure.

Support for ancient systems (Windows < 2000, GCC < 2.95)

When you read our source code, you'll find that we don't suffer much by supporting old Windows versions. It's not making any contributors' lives miserable or anything. In fact, most of our portability efforts are already done and maintaining the existing level is usually not that big of a deal.

The implicit POST on -d

I'm puzzled. What would -d do instead?

Old, mostly-unused protocols. Less code, simpler, less bugs.

The more a protocol is used, the more code and features it gets and the more bugs (and bugfixes) it gets. The hardly ever used protocols are also the smallest code bases and the code we get the fewest bug reports about. It is sort of how the game of life evens out things for us.

Cmake

I don't think we'd gain anything by removing cmake support. The cmake build is (still) not as feature complete as the configure based one but it has users and removing it won't make anyone happier or magically improve the other build setups.

the requirement to sign up for a github account to report bugs

We need users to register before submitting bugs to fight the spam we get otherwise. But also, submitting a bug report is not a one way communication where you can do a single shot post. You need to interact with the project, answer follow-up questions and often come back with additional information. That requires that you have an account.

We could use another service than github but that would just move the requirement for users to register at another service instead. The upside with github in this regard is that they already have several million developers as registered users and a large portion of modern open source hosts things there, which means that a lot of users already have an account.

Before we required a github account we required a sourceforge account.

If you really want to avoid creating a github account (which you might find useful for other proejcts as well), you can also opt to post your bug report on the suitable mailing list and we can take it from there. You can even post anonymously there (even if you need a working email address to mail from).

The proposal to remove the -k flag

You want us to remove the proposal?

While I don't think that particular proposal is going to be accepted as-is, as it met a fairly hard resistance, I treasure suggestions and proposals from people and I especially appreciate that people are ready and prepared to suggest "crazy" or "far-out" things that questions how we have been doing things in the past. We must keep making sure we're not using blinders.

Closing bugs as ""This works as designed"" without a 2nd thought

This comment clearly refers to some specific issue and indicates we hurt someone's feelings at some point. Of course we shouldn't do anything "without a 2nd thought" - we should take bugs seriously. But, we can only do so much with the time and energy we have. We need to filter out the bugs and some bugs just have to be moved to the `TODO` or `KNOWN_BUGS` documents since there's nobody around to work on them short-term. Having open bugs just linger around in the bug tracker is also bad since it creates false hopes among the submitters and they make it harder to find the important bugs. The best way to get a bug fixed is to engage yourself in the problem. Spend time and effort on helping us to reproduce, to understand the issue and to come up with a proper fix.

Additionally: we're all humans working in the project. Sometimes we make the wrong decision or we express ourselves in a clumsy way. If a bad decision was made, if a wrong thing was said, it is never too late to go back to the old subject and reopen it again for another take on that issue or problem to convince the others about your point or to highlight a mistake that was made before.

A little less black magic (transformations of the input) when making HTTP requests would be helpful

I'd appreciate an elaboration on this. Over the recent years I've also wrote "Everything curl" as a more tutorial like documentation to help users get the full picture of what curl does and how it works, but I'm of course also so deeply involved in curl and HTTP so I'm not seeing things the same way others users may. So please, help me understand what we need to document better!

global state making it thread unsafe

libcurl has no global state that makes it thread unsafe. However, most users build libcurl to use one or more third party libraries, in particular for TLS, and a whole slew of those libraries have parts of their init sequence thread unsafe. So, for libcurl to function with thread unsafe libraries it needs to init them from libcurl's init function that then has to be declared thread unsafe as well.

As those other libraries slowly fix their thread issues over time, libcurl will then subsequently also be able to be made completely thread-safe.

unrestricted auth. callback is better and more secure, not to mention more in your face for those using insecure api.

Feel free to elaborate on this.

remove support for URLs which are not colon-slash-slash (or colon-slash-slash-slash in case of file protocol). We don't need relaxed non-standards stinking up the ecosystem.

1. what is a URL? I blogged about it a while ago: https://daniel.haxx.se/blog/2016/05/11/my-url-isnt-your-url/

2. I think it is fairly important that curl can work with URLs the browsers can work with and work with the sites the browsers can work with. That means we can't totally ignore what and how they do HTTP and handle URLs. Yes, we should fight against silliness and stupid actions, but we also need to know when the fight is lost and we just need to shut up and do it even if we don't like it.

3. The FILE:// URL has **always** had three slashes like that, put in the standards specifications since RFC 1738 so of course we can't remove that or stop supporting that. But don't mistake FILE's three slashes with the browsers' support for >2 slashes for other URL schemes.

4. Unfortunately, a URL is not a standard to the level we would like. It's broken and there's no fix in sight.

curl_formadd can be a pain for bindings to languages that don't allow variadic foreign functions

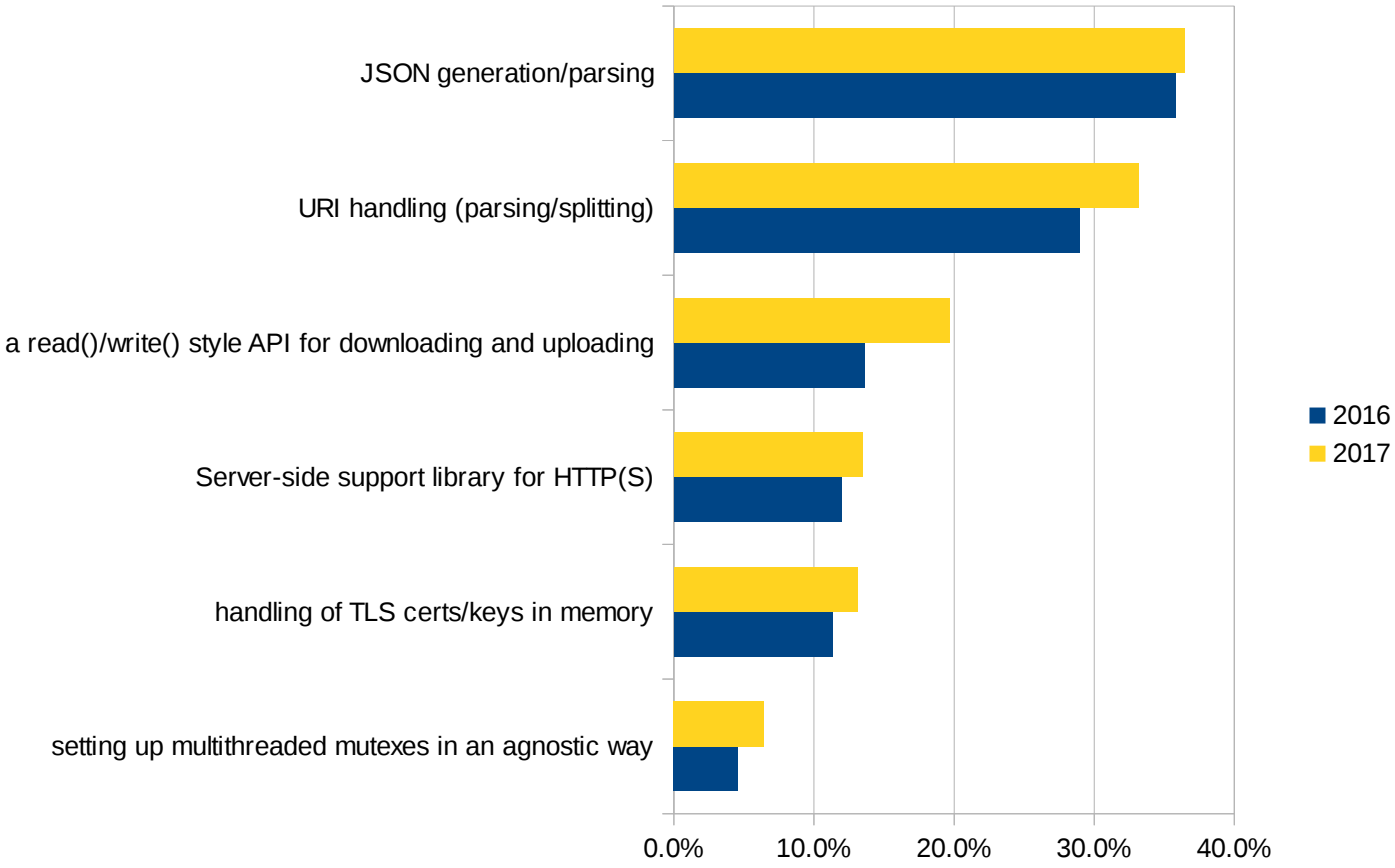
Agreed. It is in fact a generally very cumbersome and unfortunate API that I would like to deprecate and simplify. I've posted some ideas on the curl wiki on how we should or could introduce an improved formpost API, but there's been very little attention or shown interest so it hasn't moved much. I'll welcome your ideas and thoughts on the suggestion and your help in making it progress!

See https://github.com/curl/curl/wiki/formpost-API-redesigned

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

n = 334

65.1% of all respondents filled in this question. In the graph below, I decided to show the amounts as share of the total population and not only as a share amount the ones who answered this single question. Compared to last year's answers, the levels are almost identical.



Again I would like to highlight that I already last year wrote up a take at the third most popular API (according to these results) here: the read()/write() style API and I presented it in this blog post: https://daniel.haxx.se/blog/2016/04/24/fcurl-is-fread-and-friends-for-urls/

The feedback on that API has been an astounding **nothing**. For an API that every 5th user says they'd use if it existed. If there's no feedback or usage at all for an API that 20% says they'd like to use, the APIs with a lower share might make even less sense to put any work into…

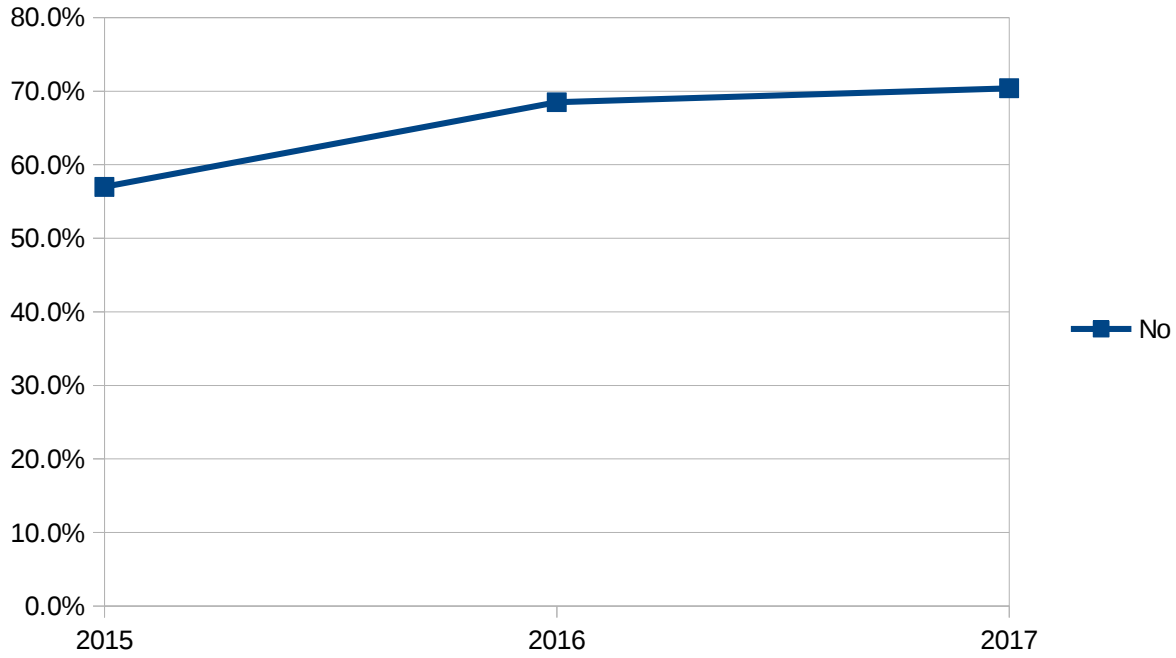
URI parsing and JSON handling are even more requested this year, with basically every third user saying they would use such a libcurl API if it existed!

Should curl join an umbrella project?

n = 368

I keep thinking that as long as we're doing good in this project, the answer to this question will remain basically the same, but if we'd somehow degrade or start to act badly, people will start favoring move to an "umbrella project" …

This question was introduced in the 2015 survey. "No" got 70.4% of the votes this year, slightly more than the 68.5% last year.



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

n = 435

Question premiere this year. A strong 12.6% of the 435 said yes and 38.2% stated “maybe”. In real numbers, that’s 55 persons who would seriously consider joining curl://up next year and a whopping 166 persons who might!

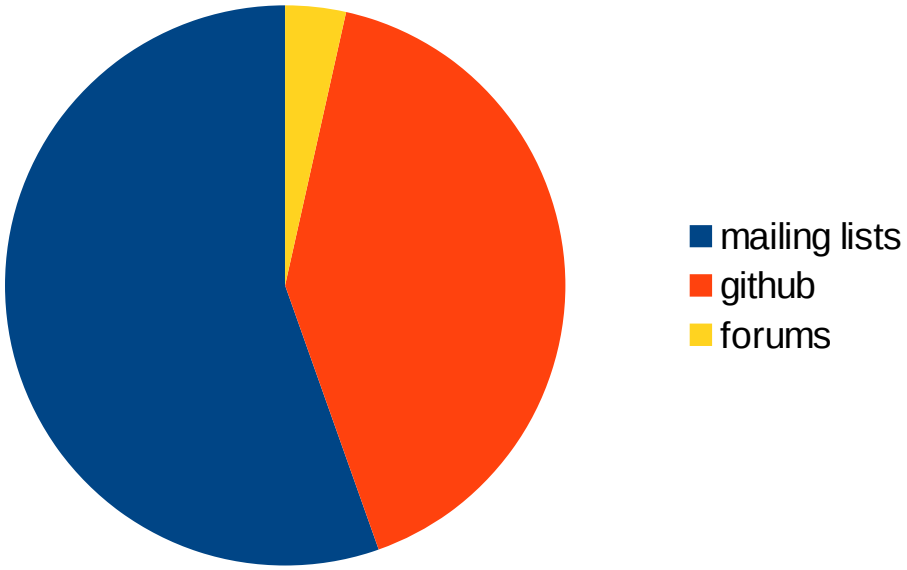
We handle discussions about bugs in the github issue tracker. For issues other than bugs, we use the mailing lists. Which would be your preference for conversations about issues other than bugs in the future

n = 413

The longest question in the survey got “no opinion” as the winner at 36.6%, but I would also count the 100 persons who didn’t answer it to that category. So I instead went through the “other” category write-ins and manually split up the answers between mailing lists, github and “forum”. It turned out several people mentioned they’d prefer a forum such as gitter, discourse or phpBB.

The users with opinions on this question were divided like this:

55.4% prefers staying on mailing lists, but it wasn’t a very big win over the 41.1% who say they’d prefer to discuss matters in github issues. The “forums” people managed to get 3.5% even though there was no direct answer for it.



Home work for us in the project: how do we deal with this. Should we open up a separate discussion repository on github and allow people to discuss matters there that aren’t strictly issues or pull requests? I’m a little concerned it will split up our little community more than it will actually help…

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

The final question is new for this year and got 34 responses. What can we learn and how can we improve the survey next time around?

Not answering your questions because "Female contributors and other minorities" is un-fucking-important. It's a goddamn open source software project, get fucking over it.

We get some responses like this every year about this question and that is probably reason alone to keep it. Our project is far from a diverse set of humans and I think most projects benefit from having a wide range of different people to get the most amount of viewpoints, varied ideas and feedback. The more homogeneous we are, the more do we think in the same way and come to the same conclusions. I believe the curl project would be a better project if we were more diverse and therefore, I feel the question is motivated.

If a single harmless question in the survey makes you offended to this level, then I think we're already better off not having your responses.

"Do you think curl should integrate more complex HTTP authentication schemes into it's API?" (Oauth, JWT, Macaroons, SAML, etc)

While probably interesting to a few, I think it is much too specific to be answerable to most users.

The "How good is the project and its members" section should be optional, I wasn't able to provide any useful information about it.

It already is! You can just skip it.

How many times you caught yourself thinking along the lines of 'phew, glad i have curl handy'!

Haha. And asking for a numerical value?

Do you have trust in the future of this project/lib? -> my answer: totally! :-)

Not a bad idea. I think it would need some qualification on what "trust" really means in this context.

"Why should memory-unsafe software like curl still exist?"

Asking why curl "should exist" in a curl user survey seems weird. I think a lot of the questions already answer why people are using curl so whether that motivates why it "should" exist I leave to others. Nobody is forcing anyone to use curl. You something else if you prefer something else!

Are you male or female?

I've deliberately resisted this question and also similar ones like how old the user is or on what continent he/she lives. Mostly to enforce the anonymity but also because I'm not really sure what I would do with that information. I'm already positive a vast majority of the users filling in the survey are male.

Split the survey into curl and libcurl sections; the two user bases are very different

True, but I would be afraid of dividing the audience like that and risk missing out users who don't feel in either camp or in both etc. And splitting the community in curl vs libcurl is also rather arbitrarily as those are not two "natural" groups either.

Add "Security" item to "Which are the curl project's worst areas?"

Yes, security is not in the best/worst question. That's a mistake I'll fix for next year.

How have we done since last year's survey?

I try to figure that out instead by asking a lot of absolute questions and then I draw conclusion based on how the answers differ from last time. I figure most users won't have any specific reason to do thumbs up or down since curl has just been there and performed as usual for the last year.

"What should curl's defaults be?"

A really hard question to ask I think. I think it should then possible ask about specific things and what should be default and what should be necessary to enable/disable. If you have specific suggestions for this, please let us know!

Source code related ones?

Like what? On source code style?

I'd like to see more questions related to security, regarding what users think can be improved/added.

Examples?

I note that this goes thru Google. That will probably double my junk mail for a while. Why not ask curl to do the delivery?

I run this survey on Google forms simply because it is an easy way to build a survey and get the answers back in a nice and convenient format. I realize this makes some amount of users drop out and never fill in their feedback and I think that's in their right and I totally accept and understand that.

more specific questions regarding how to improve contributions, be more friendly to those who want to contribute. more questions about features missing from important protocols

I'd like that, but I need to come up with the questions...

Which authentication methods do you make use of.

I like that!

Did you ever modify the curl source code before you could use curl in a project

A very specific question. A little too specific I think, as we can be fairly sure there's only going to be a small fraction saying yes to that. Or it would include users who run old curl versions and the modification they had to do was to backport changes from newer versions of curl, which I suppose this question probably isn't intended to cover...

How often (monthly) do you get confused by an IETF RFC, and refer to the libcurl source code for the authoritative spec / implementation?

:-)