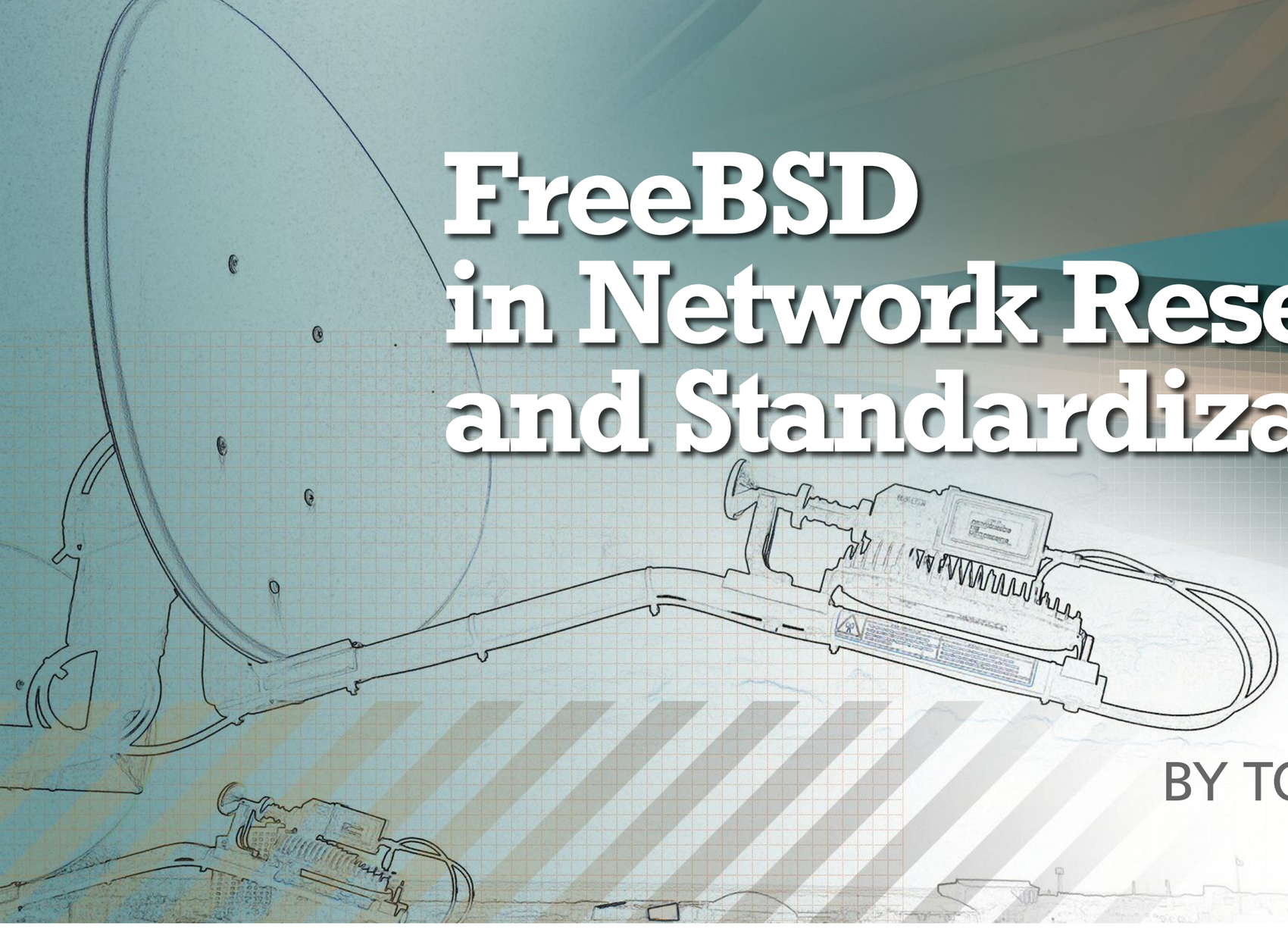
FreeBSD in Network Research and Standardization



BY TOM JONES

Our research looks at Internet protocols and how they can be improved to make the Internet better. We perform measurements on the network to get a picture of how things really are deployed, and from these measurements we develop new mechanisms, enhancements, and improvements to Internet protocols. Our work outputs into and advises Internet standards with the Internet Engineering Task Force (IETF), and we publish our work publicly so that the entire Internet community can benefit.

Our research group has been involved with the IETF for over three decades, contributing to a wide range of standards. Two of our efforts in recent years have been trying to adapt the Internet to support large packet sizes and making sure that new protocols work well in satellite networks.

FreeBSD is a vehicle for a lot of our work. By using FreeBSD both for research test beds and as a target for new features, we are able to complete work that is on par with what can be performed by a much larger organization.

FreeBSD has become an important part of our research, and in the last few years, we have used FreeBSD in three distinct ways:

- FreeBSD with ZFS makes a core component of the storage network for our research group.
- FreeBSD and dummynet are used as a core component in network experiments we perform.
- The FreeBSD network stack is a high-performance venue for research on new protocol features.

Our research group is very lucky to have our own network within the University of Aberdeen. This is becoming more and more rare as the IT service departments have taken over running computers from computer science and engineering. We run all of the infrastructure in our labs and for our test beds, using the university as an upstream provider to access the UK research network JANET.

With our own separate network, we are able to host and run experiments from IP address space that we manage. This gives us a great amount of flexibility when putting together an experiment, and we are able to bring up a test network in just the time it takes to configure the machines. In many other institutions, any network operation requires months of liaison and interaction with an IT department in which a very strong case needs to be made before anything out of the ordinary is considered. We maintain a strong relationship with network operations at the university and they are key to helping us maintain our independence.

From our network, we are able to perform large-scale measurements of the Internet, and these measurements feed into protocol design work and standardization. Without our vantage point and a friendly upstream, we would have to do scans of the entire Internet from cloud providers. The complexities of setting up a measurement from a cloud provider is hard to imagine, whereas we are able to spin up an experiment to answer a question we have in short order.

Internet Standardization in the IETF

The Internet Engineering Task Force is the standards body that defines the protocols with which the Internet is built. The IETF differs from other standards bodies (such as ISO or IEEE) in many ways, and the deliberately open process and volunteer core really stand out.

The IETF process is open to anyone who can subscribe to a mailing list—the only barrier to entry to take part is taking part. This is quite dramatically different from other organizations, as this open model allows universities like ours to be part of enhancing the Internet on the same level as billion-dollar companies and countries.

The standards the IETF defines go through an intense authorship process. They are started as Internet drafts written or curated by a group of volunteer authors. These documents may be adopted by a working group where they are reviewed and commented on, and the ideas within are improved through a process that can take a couple of months or several years. Once the idea has settled and has been thoroughly reviewed, the document may be published as an RFC with its own RFC number.

Rough consensus and running code is one of the core precepts of the IETF. Working implementations is one of the metrics used to evaluate a new idea, with the IETF preferring multiple interoperable implementations of a draft while it is being developed toward becoming an RFC. There really are only two open-source operating systems that have a mature, high-quality, high-performance network stack, FreeBSD and Linux.

It can be very helpful for a draft to see implementation so that it can be tested in the environment in which it will be deployed and so that the actual standards language that defines the protocol can be carefully checked.

We have worked on implementations of many different protocols in both Linux and FreeBSD at all stages of their life cycles, including documents we have written and documents from others. Some of these ideas continue through to becoming RFCs while others are not proven in the IETF process and are put to the side.

We have worked on new transport protocols (DCCP, UDP Lite, and UDP Options), enhancements to existing protocols such as TCP (TCP ABE RFC 8511, NewCWV RFC 7661), and new network layer mechanisms (Hop by Hop MTU) as well as developed new protocol mechanisms (Datagram PLPMTUD) and given advice on defaults to help in satellite networks.

We have also worked on documenting and improving the Sockets API (RFC8304) and have been involved in the TAPS working group to create a next generation transport framework. We are working on improvements to the existing socket API that arise from the development of transport protocols that run in user space or on top of UDP.

FreeBSD and Dummynet to Emulate Satellites

Our research group has a long history working with satellite networks. The added delay from a round-trip out to geostationary orbit can have large effects on how a transport protocol performs. Recently, we have been working with the European Space Agency to look at how well the QUIC transport protocol works in satellite networks.

QUIC is a new UDP-based transport protocol designed to replace http/2. The protocol was originally developed by Google and has been taken on by the IETF as a work item. It has taken

a lot of hard work, but the working group is now approaching the release of QUICv1. QUICv1 has been designed with a new HTTP layer, HTTP3, which is to replace HTTP2 in the web, using a protocol that is comprehensively encrypted. Strong encryption and authentication form the roots of QUIC, to the point that most protocol headers are obscured from the network.

Satellite operators and providers are worried about the impact of the complete encryption. Part of the reason why TCP works well over a satellite network is that a proxy normally runs on the satellite terminal that enhances TCP. One of the ways these proxies make TCP usable with the high and variable satellite delay is by splitting the connection at the home terminal and at the satellite gateway. This split is transparent TCP, but it has a very large beneficial effort on TCP's performance.

Our KA Band satellite dishes that connect to the HYLAS 1 satellite looking over the noted Sir Duncan Rice library building at the University of Aberdeen.

QUIC specifically sets out to make such proxies impossible to implement. With performance enhancing proxies no longer available, QUIC has to be evaluated on its own satellite network cases. The Internet community and satellite operators need to work together to make their network practical for future deployments.

Our test bed network has a satellite link similar to the sort you might get as a DSL replacement for your remote cabin in the mountains. We have in the past used this link to look at how the Internet is developing and changing for people in rural Scotland. We have a limited capacity on this link each day. For us to be able to do repeated experiments on satellite paths, we had to create an emulated link that matched the real satellite as best as possible. Having an emulated link to confirm tests also helps make our work reproducible by others who are not fortunate enough to have a cool satellite to play with.

There are two options for doing high-quality network emulation. Linux offers network emulation module (netem), and FreeBSD has the dummynet framework that works with IPFW. We performed a series of experiments on the link to get a model of its typical characteristics and got numbers for the forward (downstream) and return (downstream) link capacity and the delay of the network. The geostationary satellite system that we use has a typical delay of 600ms, which is huge compared to what you would see on customer DSL line.

We like to be thorough when creating a test bed, and we configure both FreeBSD dummynet and Linux netem test beds. Clock and timing issues make building an emulation network in virtual machines impractical, which is a shame, but we are unsure if there is any way out. Network namespaces in Linux do seem to get around a lot of the timing issues while being able to run many nodes on one host. Unfortunately, dummynet does not work in vnet Jails in FreeBSD yet.

With the requirements for discrete hosts, we tend to use PC ENGINES APU 2 boards, which are supported well in FreeBSD and Linux and are cheap enough that we have a small cluster of them for network experiments.

FreeBSD shines over Linux when it comes to configuring the network for emulation. We are unable to easily get Linux performance in line with what we need due to the design of the netem network emulator. We have found that packets were able to bypass the delay and capacity limits we put on them. It is hard to be sure of the results when occasionally a packet is able to cross the network 100 times faster than normal. The difficultly with configuring netem has hit other people recently, and we have seen other groups default to using dummynet and FreeBSD for satellite network emulation.

FreeBSD Enhancements

Our FreeBSD kernel work started in 2014. At that time, we were working on standardizing a modification to TCP called NewCWV. NewCWV defines new mechanisms for how a TCP acts when traffic is sent in bursts such as when you are watching streaming video.

We had an implementation of NewCWV for Linux that was developed while we were still working on the protocol mechanisms. We made an attempt to upstream this code into Linux, but the Linux networking community is quite closed, and we got a cold response. The IETF likes running code, and the more running code the better. We had a window of time before a project could advance, and I suggested that I port the code we had from Linux to FreeBSD.

Getting code accepted by an open-source project really is as difficult or more difficult than the implementation. An offer from a passerby might be a bug fix or a great new feature, but all code needs to be maintained, and the cost of taking something on always has to be weighed. When the offered code is an experimental feature that touches a core network path, it is much harder to know if it is a good idea to take the code.

Running code is very important in the IETF process—an idea is much more likely to get truncation if someone has put in the effort to do an implementation. When an idea is in its early stages, it is very easy to have a large impact by prototyping. The accessibility of FreeBSD and the welcoming nature of the community mean that we can spend some time playing with an implementation in FreeBSD. If the idea doesn't pan out (maybe the approach is not quite right, or it isn't the right time), we can give feedback to the authors of the idea. This allows us to have a very active review part in the IETF process.

This can be seen in the 6MAN working group in the IETF. The 6MAN working group maintains IPv6, and it is where extensions to the protocol are developed. Recently there has been an interest in 6MAN to fix path MTU discovery (something that is problematic for all network operators and sysadmins). In 2018 and 2019, there were six or seven different ideas to try different methods to make path MTU probes. We were able to prototype some of these and find the implementation difficulties. This work is still in development, but we are safe in investing a small amount of time to try something out, because if the idea does mature, we know it will get a fair review by the FreeBSD community.

Over the past three years, we, along with Muenster University of Applied Sciences, have been working on a new method to use larger packet sizes on the Internet. Our new algorithm Datagram Packetization Layer Path MTU Discovery (my favorite standards-based tongue twister) is designed to work with datagram protocol whether in user space or in the kernel. The standards part of this algorithm is approaching maturity (entering the final stages of working group process at the start of 2020), and we are proud to say that there are many implementations, including one for FreeBSD's SCTP procotol stack by Julius Flohr while he was visiting our group at the end of 2019.

The FreeBSD community is much easier to understand and approach when you have something new you would like to share. Once you find the right set of people, the FreeBSD community is very friendly and there are developers that will help you progress a patch or a new idea into something great.

The Future

FreeBSD has a long lineage in the research community as both a target and research vehicle. I believe that FreeBSD will continue to be a target for transport and network protocol advancement.

As a platform for research, I do think we are falling behind the zeitgeist. The devops bandwagon rolled in with Docker, and it has had a big impact in the way that experiments and measurement campaigns can be put together. We partnered in a research test bed as part of H2020 Fire project that used docker images as a core component for deploying experiments.

Jails and vnet might have been both nicer and easier to use for the platform, but the mind share is with Docker, and it is hard to justify building out a new system. There are a number of projects to make something more Docker-like for FreeBSD, and we hope these can develop to the point where they can provide the same feature set that is available in the Linux container ecosystem.

We aim to make our experiments reproducible, and there is a lot to be said for devops style pipelines as a method for deploying and running an experiment. Docker has caused a large number of packages and APIs to be available that make programmable tooling possible. While it is possible to hack together a similar system, too often we are required to shell out to manage an interface or configure a firewall.

Dummynet has been around for a very long time, and it is core to our usage of FreeBSD in test bed networks. Dummynet has not seen active development for several years, and while there have been recent changes to allow greater than 2Gb/s rules, there has not been the attention required for it to support traffic shaping and emulation at 10 and 100 gigabit speeds. Dummynet still works well for our satellite use cases as the capacity required is quite low. For other groups that might need very high-capacity networks, however, I am not sure that dummynet will be available as a choice.

Conclusion

For us, FreeBSD is both part of our lab infrastructure and a target for science. Using FreeBSD as a target for our work has given us a high familiarity with it, and that continued exposure has made us want to use FreeBSD as a platform for experiments. FreeBSD's base makes it easy for us to build test beds and it actively helps with our experimentation, but there is still more that could be done to make it the perfect platform for science.

The FreeBSD community is welcoming and encouraging of new work, and this fact makes it reasonable for us to propose taking a new idea into an operating system. Without this community, it is likely that we wouldn't experiment with implementing new ideas from the IETF. FreeBSD is an important part of our work, and we hope to keep using it for many years to come.

TOM JONES is a researcher in the Electronics Research Group in the School of Engineering at the University of Aberdeen in the North East of Scotland. His work looks at Internet transport protocols and their performance with a focus on satellite networks. He has worked on standardization in the IETF trying to define next generation APIs and algorithms to help the Internet grow.