- **Regan Robertson** Mod • 14 days ago
  The video will start on this page at 11:00am, and you may have to hit play or unmute. As a reminder this site works best in a Chrome Browser. You can write in comments through this feature to continue the conversation or ask questions. Please login or sign up for a Disqus account to participate in the discussion boards.
  - Reply
  - Share ›

- 
  Stuart Card • 14 days ago
  Welcome all to the first talk in Session 10 -- Assured Systems III. We all have heard about the important work Kestrel is doing to bring trustworthy networking to seL4, so now we can hear the latest right from the source. I hope to learn both about the work product and the methods used to produce it.
  - Reply
  - Share ›

- 
  Renato Levy • 14 days ago • edited
  looking forward for this presentation. This capability is very much needed. I hope it can be easily virtualized!
  - Reply
  - Share ›

Aleksey Nogin • 14 days ago

What is the underlying semantics that you use to prove the transformations correct? Does it handle concurrency?

Reply

Share ›

Eric Smith Aleksey Nogin • 14 days ago

Hi Aleksey,

Most of APT operates on simple ACL2 functions, where the semantics is given by the functions' definition in ACL2. For the final step (to C code), we'll use the semantics we are developing for C (also defined in the logic of ACL2). We currently have little or no handling of concurrency. Not sure how much will be necessary for this project.

Reply

Share ›

Aleksey Nogin Eric Smith • 14 days ago

Once you go from functional specifications to optimized implementation, I would imagine that a lot of things that are stateless per-packet in the specification become stateful operations on some shared data structures. Are you using some existing formalism for the memory model (e.g. based on a separation logic approach), or are you creating your own? If latter, is this something that is already done, or still being worked on?

Reply

Share ›

Lennart Beringer Aleksey Nogin • 14 days ago

Also, are there any plans to relate (semiformally, modulo the difference in the proof assistant used eg) your semantics of C against the one used by seL4 (Autocorres), CompCert, or similar ones?

1

Reply

Share ›

Eric Smith Lennart Beringer • 14 days ago

That would be good, but we don't have explicit plans to do it in the short term. Our C formalization is being written directly from the standard(s) that define C. It would be good to cross check it with other formalizations. I think the difference in proof assistants would make any formal connection hard (though there is a formal

connection between ACL2 and HOL), so semiformal, as you suggest, would seem to be the way to go.

Reply

Share ›

Lennart Beringer Eric Smith • 14 days ago

at least some mileage could likely be gotten from using tools like lem & ott, or just by using same style of semantics (similar notions of states, corresponding judgement forms and rules,...)

Reply

Share ›

Eric Smith Aleksey Nogin • 14 days ago

This is still being worked on and will depend on how Alessandro implements ATC. I don't think he has explicit plans to use separation logic. Let me know if you'd like me to put you in touch with him.

Reply

Share ›

Aleksey Nogin Eric Smith • 14 days ago

Another question on specification - what is the shape of your top-level specification? Is it some sort of an I/O automata model for the whole stack (e.g. one that receives incoming packet and API call events, and produces packet transmission, API call returns, and API callback events), or something else?

- Reply
- Share ›

Eric Smith Aleksey Nogin • 14 days ago

Yes, that's essentially what we plan to do, at least for stateful protocols like TCP and probably even for things like UDP where we want to provide a socket interface. But we haven't created the spec yet, so things may change.

- Reply
- Share ›

Paul Pazandak • 14 days ago

Will the resulting C code incorporate support for UDP as well?

- Reply

Share ›

Eric Smith Paul Pazandak • 14 days ago

Hi Paul, Yes, we plan to cover UDP as well. In fact, we may handle it before TCP since it's so much simpler.

Reply

Share ›

Paul Pazandak Eric Smith • 14 days ago • edited

That would be great. DDS uses UDP (it can use TCP) and implements better optimized functions on top than TCP for real-time pub/sub comms. We'd love to try it out when it's ready.

Reply

Share ›

Renato Levy Paul Pazandak • 14 days ago

this could be an important layer for a DDS version for seL4. RTI?

- Reply
- Share ›

Jason H Li • 14 days ago

Like what I heard - no undefined behaviors since they don't exist in ACL2 hence not in C!

- Reply
- Share ›

Jerry Dussault • 14 days ago

Really needed, like the apaproach. What are your plans for licensing and long-term support?

- Reply
- Share ›

Nathan Studer Jerry Dussault • 14 days ago

Adding on. Will the licensing be the same for the source code, specification, and the formalized RFC specs?

Reply

Share ›

Eric Smith Nathan Studer • 14 days ago

Probably. I expect/hope that we can be pretty flexible here.

Reply

Share ›

Eric Smith Jerry Dussault • 14 days ago

Hi Jerry, We plan to make the code open source as well as the specs and the tools used to generate it (some are already open source). So if a change is needed, one can re-generate the code from the (changed) spec with, we hope, effort proportional to the size of the change. The license will be a permissive one, like 3-clause BSD. Ideally, the network stack would be just the first of several formally synthesized components that we develop and support over time.

Reply

Share ›

Renato Levy Eric Smith • 14 days ago

sounds like a must have component for CoE distros

Reply

Share ›

Jerry Dussault Eric Smith • 14 days ago

That's super! Thanks.

Reply

Share ›

Jason H Li Jerry Dussault • 14 days ago

Hey something the CoE can help host down the road!

Reply

Share ›

June Andronick • 14 days ago

Nice to see the work ramping up with the start of phase 2. Thanks Eric.

Reply

Share ›

Eric Smith June Andronick • 14 days ago

Thanks, June! We are excited to get started!

Reply

Share ›

Ihor Kuz • 14 days ago

Regarding NIC driver in the same component as the stack or not? Including untrustworthy NIC driver code in the same component as the verified/trustworthy network stack code will potentially compromise the network stack code and make it untrustworthy.

Reply

Share ›

Renato Levy Ihor Kuz • 14 days ago

correct! some NICs can leak at the SoC level

Reply

Share ›

Eric Smith Ihor Kuz • 14 days ago

Yes, we know. Ideally, it would all be verified/synthesized, but we had to pick and choose what we'd work on first. It sounds like you think it would be better to have the NIC driver in a separate component. That seems fine with me, but I need to really get my head into the seL4 issues.

1

Reply

Share ›

Jerry Dussault • 14 days ago

Top down is the way to go, but has anyone looked at trying to verify a TCP/IP stack bottom-up, using one of the approaches presented earlier in the Summit. That would be a fascinating comparison.

Reply

Share ›

Carl Nerup Jerry Dussault • 14 days ago

Cog has done it.

1

Reply

Share ›

Ihor Kuz Carl Nerup • 14 days ago

Do you have details?

Reply

Share ›

Carl Nerup Ihor Kuz • 14 days ago • edited

Sure - let's set up a call and our CTO can walk you through it. We have done it for multiple customers. Pulling the Comms stack is a de facto requirement when someone considers isolation for a high assurance system.

Reply

Share ›

Eric Smith Ihor Kuz • 14 days ago

Interesting! Any published papers or other write-ups available?

Reply

Share ›

Eric Smith Jerry Dussault • 14 days ago

I'm not aware of any such work but would be very interested if there has been such an effort!

Reply

Share ›

Stuart Card • 14 days ago

UDP support would also enable layering the NACK Oriented Reliable Multicast (NORM) protocol, developed by NRL, for group communications, inc. to receivers under EMCON.

- Reply
- Share ›

Paul Pazandak Stuart Card • 14 days ago

Assuming that the UDP is fully implemented... is that the plan? Multicast is an important feature for DDS pub/sub.

- Reply
- Share ›

Eric Smith Paul Pazandak • 14 days ago

We're planning to do as much as we can. It sounds like multicast is a priority to the community, and that's very helpful to know!

- Reply
- Share ›

Ihor Kuz • 14 days ago

I may have missed it, but what kind of properties would be verified about the stack?

Reply

Share ›

Eric Smith Ihor Kuz • 14 days ago

The main one would be that it implements the spec, in terms of basic stuff like parsing the packets, the TCP state machine, etc. That would imply also the lack of undefined behavior of the C code. We think separation (of data belonging to different processes using the stack) may also be of interest, but on the other hand all the data gets mixed together when it goes out over the same network interface, so maybe that's not critical. Thoughts? What else would be valuable?

Reply

Share ›

Stuart Card Eric Smith • 14 days ago

Whether separation is important depends upon where the crypto is implemented and upon whether we are concerned with e.g. Denial of Service attacks by starvation of one process for bandwidth by another flooding the stack.

1

Reply

Share ›

Aleksey Nogin Stuart Card • 14 days ago

This is a really good point - I think some soft of fairness / anti-DoS guarantee would be really great to have.

Reply

Share ›

Ihor Kuz Eric Smith • 14 days ago

"separation (of data belonging to different processes using the stack)" was what I was wondering about. In the absence of that I would use separate IP stack components for separate data streams.

Reply

Share ›

Stuart Card • 14 days ago

I am eager to begin developing a formally verifiable implementation of the Host Identity Protocol (HIPv2) and Drone Remote Identification Protocol (DRIP) atop this. What are your plans for working with developers of other network protocol stack components?

- Reply
- Share ›

Eric Smith Stuart Card • 14 days ago

Cool! Those sound like the kind of things that our stack should support. Is it enough that we provide a POSIX-style socket interface to higher level protocols like those?

- Reply
- Share ›

Ihor Kuz Eric Smith • 14 days ago

a "POSIX-style socket interface" may not be the most appropriate if you want to run the stack as a separate component (due to it being a "copying interface). Of course the POSIX interface is useful if you are porting code. So tradeoffs...

- Reply

Share ›

Eric Smith Ihor Kuz • 14 days ago

Ihor, Can you say more? Are you thinking something with shared memory would be needed for better performance?

Reply

Share ›

Ihor Kuz Eric Smith • 14 days ago

Curtis might say more about it this afternoon in his talk (I can't remember if he touches on this).

You definitely need shared memory between components for packets.

But the usual read/write syscalls assume that data to be sent is in an arbitrary buffer in the sender app. If that's in the same address space as the stack/driver, then they can access that directly. But if it's in a separate component/address space, it likely requires a copy across the component boundary. It can be avoided, but leads to awkward requirements and restrictions placed on top of the POSIX interface.

Having a different API may make this less awkward and less prone to requiring copy operations.

Reply

Share ›

Eric Smith Ihor Kuz • 14 days ago

Thanks, Ihor! This is very helpful. I need to get my head into these interfacing issues.

Reply

Share ›

Jacob Saina • 14 days ago

How do you deal with cases where a spec is ambiguous, or potentially even inherently vulnerable?

Reply

Share ›

Eric Smith Jacob Saina • 14 days ago

Ambiguities in the spec (e.g., the RFC) should be caught as we formalize it in ACL2, in the sense that the ACL2 formalization will be well-defined. As in most formal methods projects, there is always the chance that the spec is just wrong. But the expectation is that the spec is so much smaller and simpler than the code that errors are much less likely. This risk can be mitigated somewhat by proving various properties of the spec, such as type-correctness of all functions and any other desired properties (e.g., that parsing and serialization are inverses).

- 1
- Reply
- Share ›

Jacob Saina Eric Smith • 14 days ago • edited

Right, I'm wondering what one would do once ambiguity is discovered. There's a disconnect for me between ambiguous spec and generated code - do you have to manually make decisions about how to constrain ambiguities? Is this difficult?

- Reply
- Share ›

Eric Smith Jacob Saina • 14 days ago

It could well be that the spec leaves some behavior under-constrained. In that case, we should be able to pick any of the possible behaviors allowed by the spec and still be spec compliant. That choice would be made during the refinement process. I'm thinking of things like which error you return if there are multiple errors. If anyone does care which behavior is chosen,

maybe that should be part of the spec! Contradictions in the spec will have to be resolved since no implementation can satisfy a contradictory spec. Other ambiguities ("Does it mean A or B?") may have to be resolved by looking at real implementations, but we hope there's not too much of that (it may indicate a problem with the RFC). In the past we've found that just the process of trying to formalize something can flush out a lot of spec issues (missing cases, contradictory and unclear stuff). Not sure if that will be the case here as these things have been specified fairly carefully in the RFCs.

- 1
- Reply
- Share ›

Jacob Saina Eric Smith • 14 days ago

Thanks!

- Reply
- Share ›

Olin Sibert • 14 days ago

I would very much like to see the L2 (Ethernet) driver as a separate partition, both to facilitate porting to different driver hardware and to facilitate sharing of the network (which may be virtual) by multiple instances of the IP stack. Such sharing has complexities of its own, but having a well-defined architectural boundary at which it can be controlled seems like an important principle.

- 1
- Reply

Share ›

Eric Smith Olin Sibert • 14 days ago

Thanks, Olin. This kind of feedback is what we were hoping for from the talk/summit.

Reply

Share ›