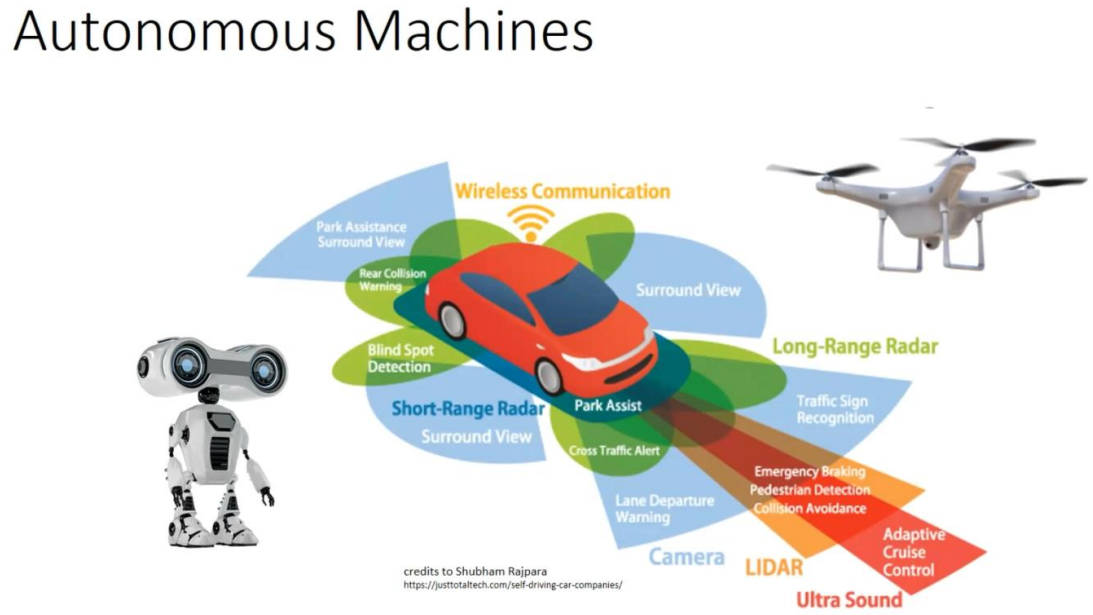
RTAS 2021 Panel Discussion
May 21, 2021

Moderator: Nan Guan, The Hong Kong Polytechnic University
Panelists:
Shinpei Kato, The University of Tokyo; Tier IV, Inc.
Andrei Kholodnyi, Wind River Systems
Shaoshan Liu, PerceptIn
Jan Staschulat, Research Engineer, Robert Bosch GmbH
Robert Bosch GmbH Rich West, Boston University

---Introduction to the panel---

[Nan Guan]

The topic of panel discussion is **RTOS for autonomous machines.**



Just a few words of that background. Everybody knows and agrees autonomous machines are very important, and there are many different kinds of autonomous machines. If we want to extract some common futures, they are really smart machines which can perceive the surrounding world and have the intelligence to decide and to react correspondingly. I think, in the very near future we will live with all those autonomous machines in our life.

Today, we want to discuss about the real-time operating system for autonomous machines and by real-time operating systems, I mean the wide sense. Because when people usually talk about RTOS, it could be the very thin real-time scheduling kernel. Today we are talking about the real time runtime software infrastructure, including different layers, the middleware and so on.

I am very happy to invite five experts in this area, and now I invite them to introduce themselves.

--- Self-introduction of the panelists---

Shinpei Kato
The University of Tokyo
Tier IV, Inc.

Andrei Kholodnyi
Wind River Systems

Shaoshan Liu
PerceptIn

Jan Staschulat
Robert Bosch GmbH

Rich West
Boston University

[Shinpei Kato]

It has been a while since I attended RTAS, which as almost 10 years ago. I am still doing a lot of r&d in real time systems, but I am more interested in open source project today. I am really happy to share my experience with the problem of real-time system and scheduling in the real world.

[Andrei Kholodnyi]

My name is Andrei Kholodnyi. I am from Wind River Systems, and we produce real-time operating system VxWorks. I am also a member of the ROS2 technical steering committee and I lead also ROS2 real-time working group. I am deeply involved in real-time operating systems and other technologies surrounded.

[Shaoshan Liu]

My name is Shaoshan Liu and from PerceptIn. We focus on autonomous machines, to be more specific, for intelligent transportation. My personal stand is that the age of autonomous machines is upon us, but yet we don't have a very clear idea how to build the software system and the hardware system for these autonomous machines. I am really glad to have these panel to discuss the future directions of these fields.

[Jan Staschulat]

My name is Jan and I am working at Bosch in the area of robotics and autonomous driving. My background is real-time computing and adapting these techniques to ROS2 and developing deterministic execution mechanisms for ROS2 on microcontrollers. My main interest is determinism and real-time systems.

[Rich West]

I am Richard and I am actually a chief software architect for DRAKO MOTORS, as well as a professor at Boston university. I am currently developing a system for next-generation electric vehicles, where we're trying to consolidate all the electronic control unit functionality of a traditional vehicle onto a centralized operating system.

--- Research challenge and opportunities for RTOS for autonomous machines---

[Nan Guan]

Today, the ultimate goal is we try to identify some research opportunities and challenges on this topic. But I understand this may be a very big question, so it is difficult to start. Let's try to start with a little bit of brainstorming style: people can just try to give their opinions and all the panelists are welcome to comment on others' opinions and to discuss. Along the way, there will be some specific topics popping up, then we will catch them and we will go deeper. I invite everybody to say something about what's the unique research challenge and opportunities for RTOS for autonomous machines.

[Shinpei Kato]

I used to work on scheduling algorithms on Linux kernel implementation, which actually leds me to start developing the platform for autonomous vehicles, and I made it open source which today we call Autoware.

I started Autoware using ROS, because it was the most deployed open source runtime in a compositional system. I realize that we didn't really have real-time capabilities in ROS, and its original version also lacks in the capability of real time. Even I had a great experience in real time systems community with a lot of useful technologies to ensure safety or security of such autonomous machines, but first of all, it is very difficult to still employ such useful technologies in real systems. For my interest, I know that we have so many useful technologies from this community, so my goal is to employ those pieces of useful technologies into real systems as much as possible.

What I am really interested in today is a graph analysis in real time aspect, because today you can't build your system by yourself, so you would like to work with other components. ROS or ROS2 are such platforms that allow you to develop your components and plug them into other components which could be developed by other companies or individuals. Now the system exactly looks like a direct acyclic graph. It is a pity that a great experience in this community how to make such graph-based systems to be real time capable, but today even I can't really design and implement a real system with DAG real time scheduling. There are actually similar problems to me for other real time technology.

My observation is still, it is quite difficult to employ useful technologies from this community to implement in actual real system, I like to know, from other people, from ROS community how you are identifying your core challenges in real systems in terms of the technologies that developed from this community.

[Shaoshan Liu]

I have several observations, for the last few years, when we ship commercial products autonomous machines and so on.

The first thing is that it seems to have a very deep processing pipeline unlike previous computing we have encountered. In mobile computing, the computation pipeline is much shorter. For mobile computing you are really running one foreground APP at one

time, as for robotic computing you have to optimize the whole graph, which the previous speaker brought up that it is a DAG graph and every node has to be optimized so that none of those nodes can become the bottleneck, otherwise, your system is screwed. But on the scheduling side, we don't seem to have things to take care of that. So how to deal with this challenge when people build their own ad hoc solution, software and hardware, to make sure their whole system is meeting the deadline end-to-end. We did that as well. We had a micro paper to describe our system which uses FPGA for sensor processing, GPU for the heavy lifting, and CPU for planning and control. But essentially, it is an ad hoc system, and it is really hard to transfer to other autonomous machines.

We step back and try to summarize what is next to that. The first thing is that the so-called real time system today, ROS itself is a middleware, it pretty much has no influence on how we schedule things down there, has no heterogeneous computing support, for example, to bridge two FPGA is very hard, and then to use the traditional scheduling algorithms on a heterogeneous system is very hard.

So, the result is that we have to perform a lot of manual work to try to construct these ad hoc solutions to map different computing onto different substrates, and it takes a long time, a lot of iterations. Most of the companies today in this field, we don't have google's resource, we don't have apple’s resource to deal with this kind of hassles. So if there were a very good real time system, the whole field will iterate much, much faster.

The second problem is a more recent problem that we have observed is that all these machines are connected machines, so we have not only to deal with computation on machine, we have to deal with communications across machines, and that kind of support in software system is missing or not enough today.

To summarize, we need better scheduling support from the ROS level, what we call the RTOS for robots. But today, ROS is a bit disconnected with the underlying real operating system on scheduling. Second, we need much better support for mapping the workload to heterogeneous hardware, such as domain specific architecture, or FPGA, or GPU, while ROS was built on top of CPU today. Autonomous machines, when you try to think of it, its abstraction is beautiful, for example, you have a function call to the perception, you can make that a very high-level abstraction of software. But today we are going into a lot of details to optimize the fine-grained stuff. But if the software layer can provide more or higher levels of abstraction will make programming much easier.

[Andrei Kholodnyi]

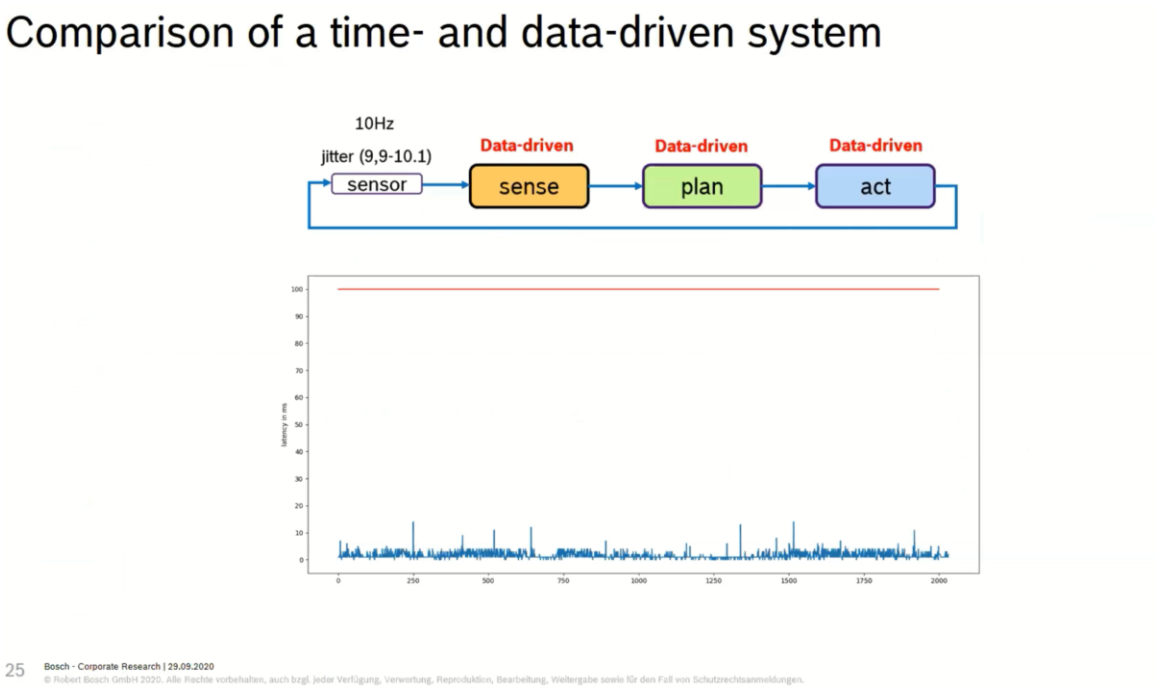
As a real-time operating system manufacturer, I would only support what you have said. In regards of autonomous machines, internet is moving from a human Internet to machine-machine Internet where all these real-time computing probably will move to the edge or a lot of computing will move to the edge. What would be also interesting is in terms of time sensitive network, it would also move from this heterogeneous system real time into some kind of distribute real time, it is also pretty interesting topics to address for this community.

Besides that, since there will be more and more autonomous machines, I think all these self-X properties, such as self-sensing, self-maintenance and so on, for the autonomous machines would get more and more attraction. So, in this regard, probably what would be also a very interesting area for the real time systems is some kind of self-optimization in terms of real time properties, like better layers and determinism and some kind of self-improvement scheduling.

This kind of stuff would be also very interesting to see what kind of research can people bring into these areas. And if we touch RTOS for ROS2, I totally agree with previous speakers that there is a lack of real time in ROS2. As the chief real time working group, I just could admit this, and I think there is the effort in ROS2 community to work on this.

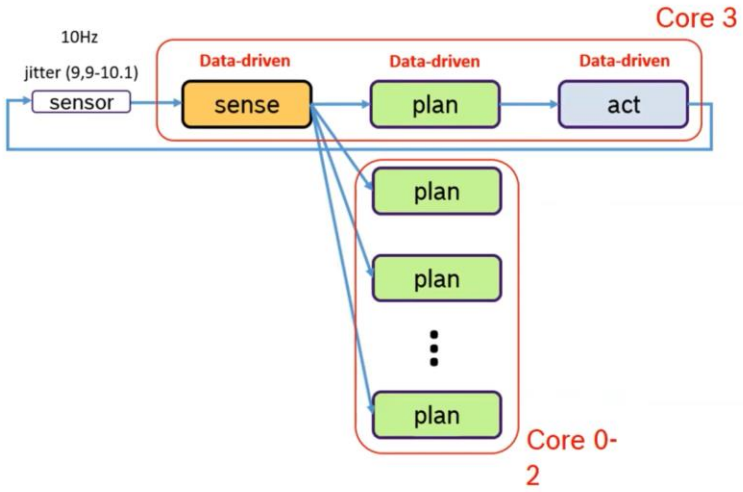
[Jan Staschulat]

I have two slides to describe some problems what we see here at Bosch.



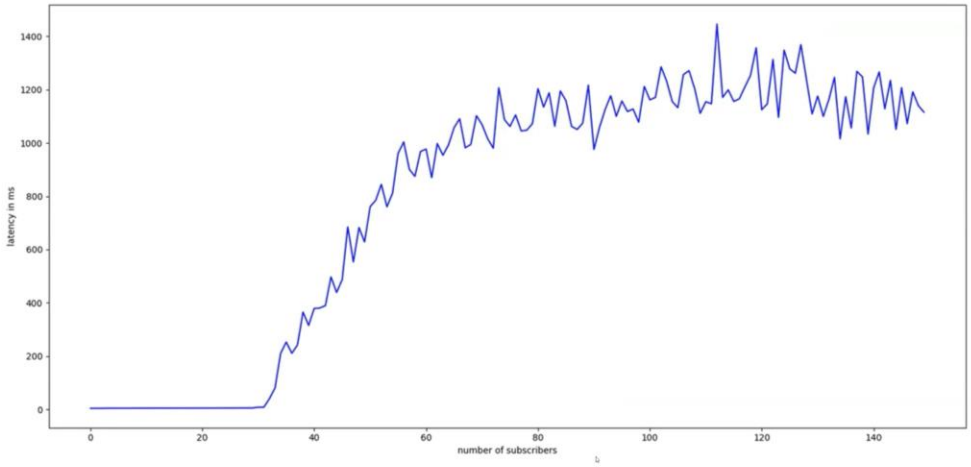
This is the work we are working in real time working group for ROS2, and it was presented last year, where the papers also available. What I want to highlight as a problem or as a as an experiment that we did, we are looking at end-to-end latency, which is very important in the robotics but also in autonomous driving. You have sense, plan, act, typical control loops where sensor data comes in, and in this case, it is data driven, so like in ROS2, every element is executed when the message is available. And now the end-to-end latency from the very beginning to the end has been measured on a Raspberry Pi. All executed on a single core, and we see the end-to-end latency is two to three milliseconds, kind of.

Experimental Setup



And now we were interested, what happens if we add some node on this system, and we have made this experiment on one core. We had the sense plan act loop, and then, on the other three cores, we just have some additional listeners, so another planner or something else will just listen to the messages.

Results
Cause effect chain latency over subscribers



And we were just interested what happens to the end-to-end latency of this safety critical control loop. And this is the result. On the X axis, you see the number of subscribers on the other cores. On the y axis, you see the end-to-end latency. You see

an increase of three orders of magnitude. Because you have changed nothing on your single core, that these are the only processes running, and you have just added some additional workload on the other cores. And this is what I want to highlight, we're at this conference, we are looking at real time systems and scheduling, but putting it all together is a big challenge.

And you might ask yourself what went wrong here. Of course, you have shared memory and because you are sharing all the messages need to be passed from one core to the others. That's why the safety critical part cannot access or does not get the access fast enough. And the second thing is the communication stack in ROS2. We have not really looked into details what was the real cause, just shared memory, or is it part of ROS2.

Similar to a paper yesterday, with virtual reality they looked into network stack and optimize there. The message that I want to convey is: doing real time analysis based on singer values of execution times it is not really practical anymore. you have to really look at the big picture, and this basically fails when you integrate things.

I have been working for a long time in the worst-case execution time analysis community and now I am on industrial side. I have to say that this assumption, when you do scalability analysis that you first determine one value the worst-case execution time of a task and then do a very advanced scheduling analysis or scheduling policy on top of it, does not meet the current setup of systems anymore. You do have to look at shared memory access time which you don't know how long it takes. Of course, you can stay always inside of your theory and advanced this theory, but it does not address the problems we have.

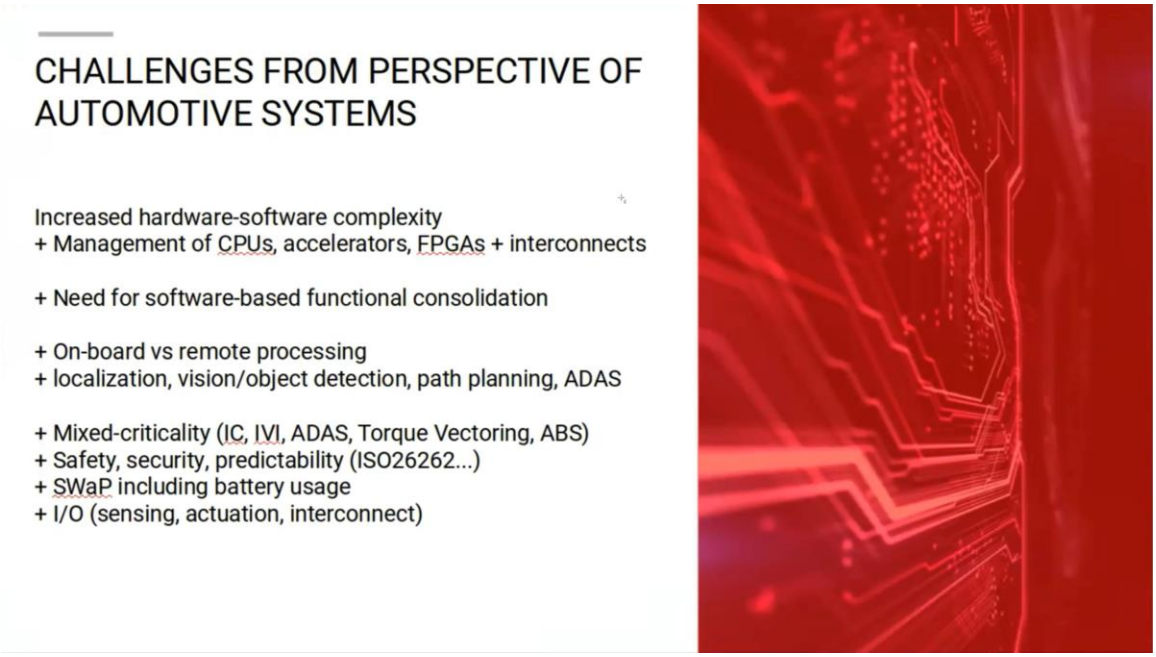
It is very difficult to apply these kind of scaling techniques to a multicore platform for autonomous driving. What I am interested in is approaches that can handle uncertainties, for example with fault models, and not so much about probabilistic ones, because probabilistic modeling approaches always assume randomization, I think randomization and determinism is not really a good mix.

Another way would be to control the interference between different cores, either in terms of hardware, operating system or analytical models or analysis techniques. As the final research challenge, I would see is the integration of CPU processing in cause effect chains and the end-to-end analysis.

[Rich West]

To follow on from what Jan just said, I thought that was pretty interesting, you mentioned several things that are you mentioned this idea of pipeline processing from sensing to sensor data processing to activation, and we've also heard people talk about things like ROS and ROS2 and the need for the underlying operating system beneath that.

Now I have recently started working with a partner company Drako Motors, as the chief software architect for that company. And we're building this vehicle management system called Drive-alas, so I am going to tell you some of the things that I see from the perspective of the actual operating system itself underneath all of these other higher level abstractions like ROS and so forth.



So, first of all we're seeing a ground change in the hardware that traditionally has been in the sphere of vehicle management systems. In the past we've had these very basic electronic control units and now we're going to a much more complex hardware, software a hybrid of CPUs, hardware accelerators, FPGAs, and interconnection networks.

And I said here that we need one of the things I think is a very important thing is how do we do this functional consolidation. How do we stop the growth of electronic control units getting out of hand and actually consolidating the traditional functionality is targeted at separate hardware on to actually a centralized platform. I will say more about that in the moments.

And then obviously there's the trade-off between what how much you do processing on EG a vehicle versus on the edge network in the vicinity of a vehicle. And then you've got all your traditional challenges your mix criticality challenges you've got to manage your vehicle, from the point of view of the user interface to the occupants of the vehicle, the instrument cluster the in-vehicle Infotainment features and so forth.

Through to the more critical features that are timing sensitive, which relate to the power train and control of the vehicle talk factoring Advanced Driver Assistance Systems and so forth. And i've mentioned the usual safety, security predictability challenges size weight and power, including battery usage, and then things such as IR, and I think IR, we heard Andre mentioned this before about things like time sensitive networking thing

we're going to move away from traditional canbus networks in vehicles to higher bandwidth networks, so let me just show you a couple of other slides.

THE GROWTH OF VEHICLE ELECTRONICS

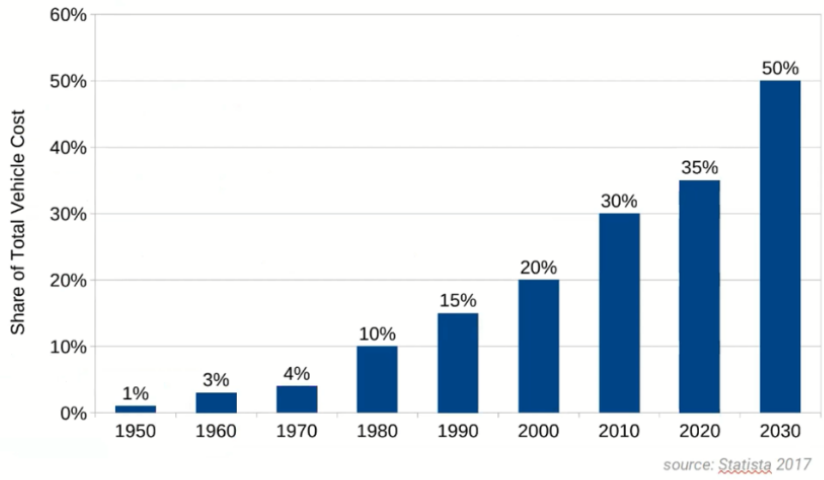
Modern luxury vehicles have 50-150 ECUs
source: Strategy Analytics, IHS Markit

Global ECU market $63.6 billion (2018)
source: grandviewresearch.com

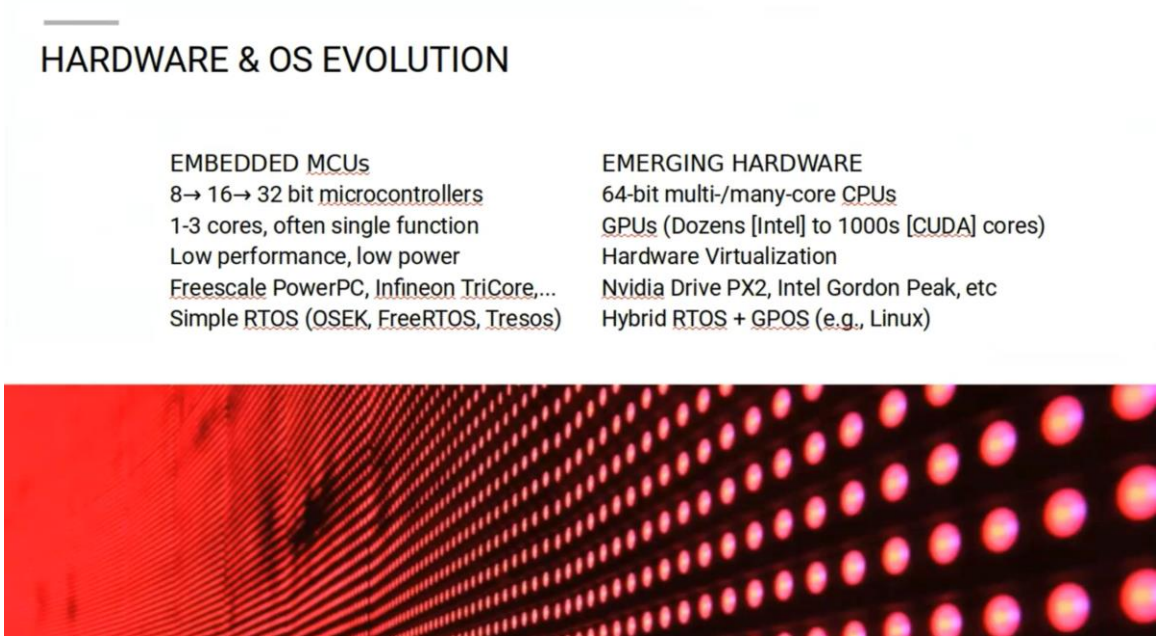
ADAS, HEVs and BEVs driving costs of electronics in vehicles

HEV + BEV ECUs 3% market in 2018
+ Potential rise to 15% by 2030
+ Continental & Bosch have 28% ECU market
source: eenewsautomotive.com 2018

ELECTRONICS SHARE OF TOTAL VEHICLE COST



If you look at a modern luxury vehicle you'll see anything from 50 to 150 electronic control units in that vehicle. This market is growing enormously it is a 10s of billion dollar markets. If you look at statistics as data they predict that by 2030, the cost of a vehicle will be 50% based upon the electronics, implemented on that vehicle. So now we're starting to see new challenges from a software perspective, how we manage that hardware.



So I said here hardware and operating system evolution going back to traditional cars that were not autonomous you'd have everything from eight to 32 bit microcontrollers running at 10s to hundreds of megahertz with anything from one to potentially three cores. These microcontrollers would all be part of separate electronic control units distributed around the vehicle controlling everything from your chassis your body your power train your Infotainment functionality. And, each ECU will be dedicated to a separate function typically and the low power and low performance.

And so, in vehicles that we run today you'll have simple real time operating systems you'll have OSEK operating systems FreeRTOS you might have Tresos operating systems. But as we go towards more complex hardware marrying CPUs GPUs FPGAs and high bandwidth sensor technologies. We're going to have to come up with new operating system designs that are a hybrid of traditional real time operating system type features, and more general purpose features that you'd find in systems such as Linux. And the reason for this is simple, we can't just rewrite all those RTOS from scratch, it will take just many, many years to do so. And the verification and certification challenges would be prohibitive here, so we want to keep the art of small, but we want to integrate it with a general purpose system.

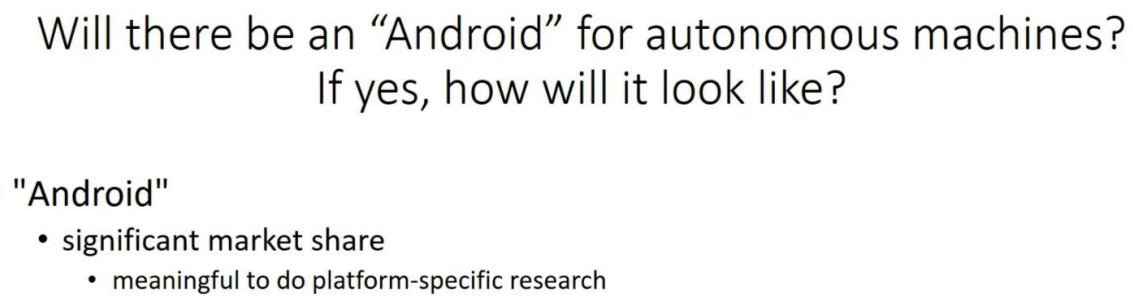
And this is something that I am looking at working with my company now at Drako Motors, we're trying to build a system that consolidates all these ECU functions a software defined functions on a centralized platform. So I am going to stop there, but that's really all I wanted to say at this point.

---Discussion around “Android” for autonomous machines---

[Nan Guan]

The time goes so fast, so we are we just started, but we already it is already half hour. So I get impression that this is really a complex problem, so in the statements, we have mentioned everything, I hear a lot of keywords like heterogeneous, communication, multicore, distributed, everything.

I would like to drive the following discussion a little bit to a specific question. You know, we have a lot of researchers in real time community and for a lot of issues you just mentioned, we have people work on those issues. But it is it is difficult to directly apply the theoretical work or even a bit less theoretical work to practice, this is really the case.

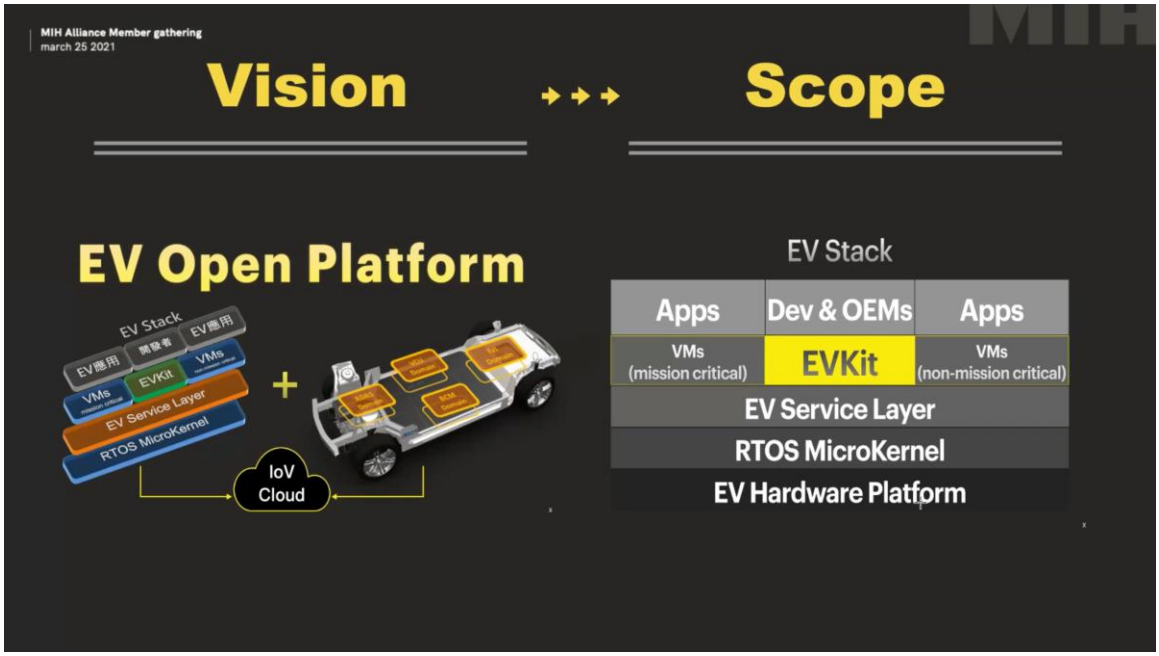


So I wonder is there a possibility that many people can work on standard thing, for example today, if you say I am working with the operating system for mobile phones, if you do not work on android or you cannot work on ios, probably you still can work on that our first year, you can create new things, but the impact would be not as large as working on, for example, android directly.

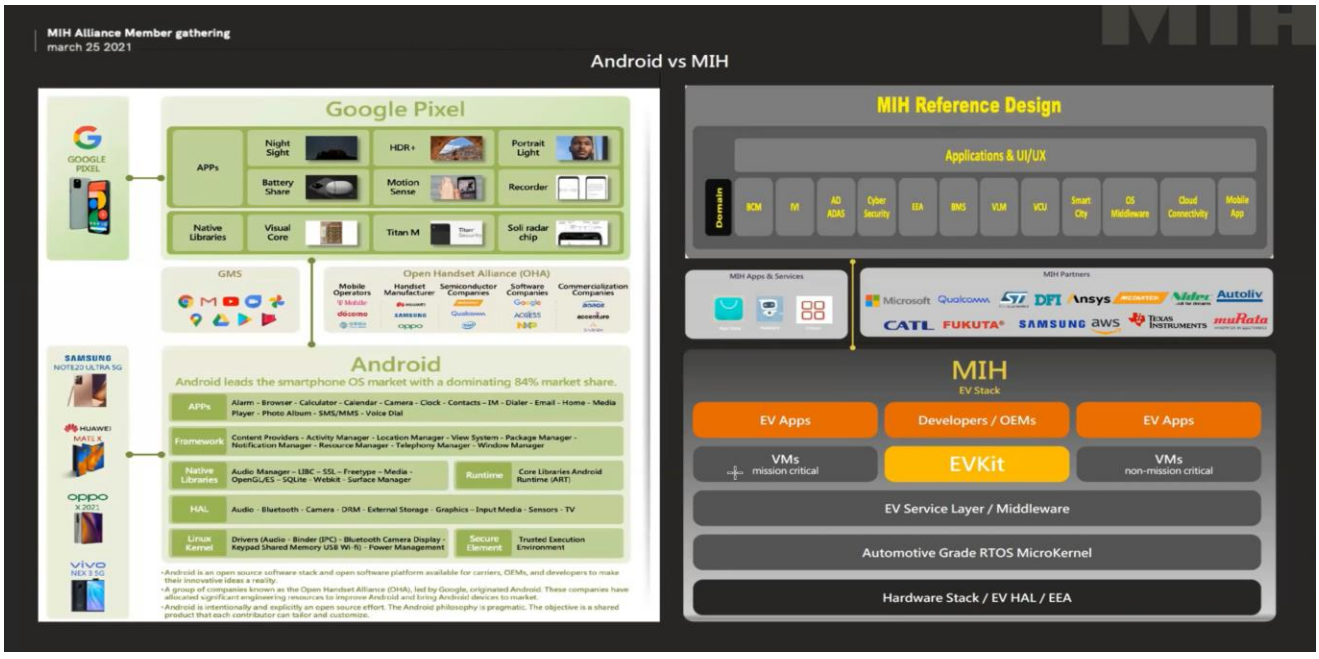
My question is that, what's your opinion about will there be something like android for autonomous machine, at least one thing that we can benefit. It is really meaningful to do this platform specific research, so our research is not so fragmented. So what's your opinion about this.

[Shinpei Kato]

I think the answer is definitely. Let me share some slides here.



This is exactly what my technical goal to make android for autonomous machines, so we recently launched MIH, which is an EV open platform initiated by Foxconn. I believe this is a way to the future of autonomous machine. We can actually open up not only software, but also hardware components. The reason why android is actually really deployed in the market is, yes android is open source, but the Community initiated by Google, they clearly defined the reference design how android can be deployed in the real hardware.

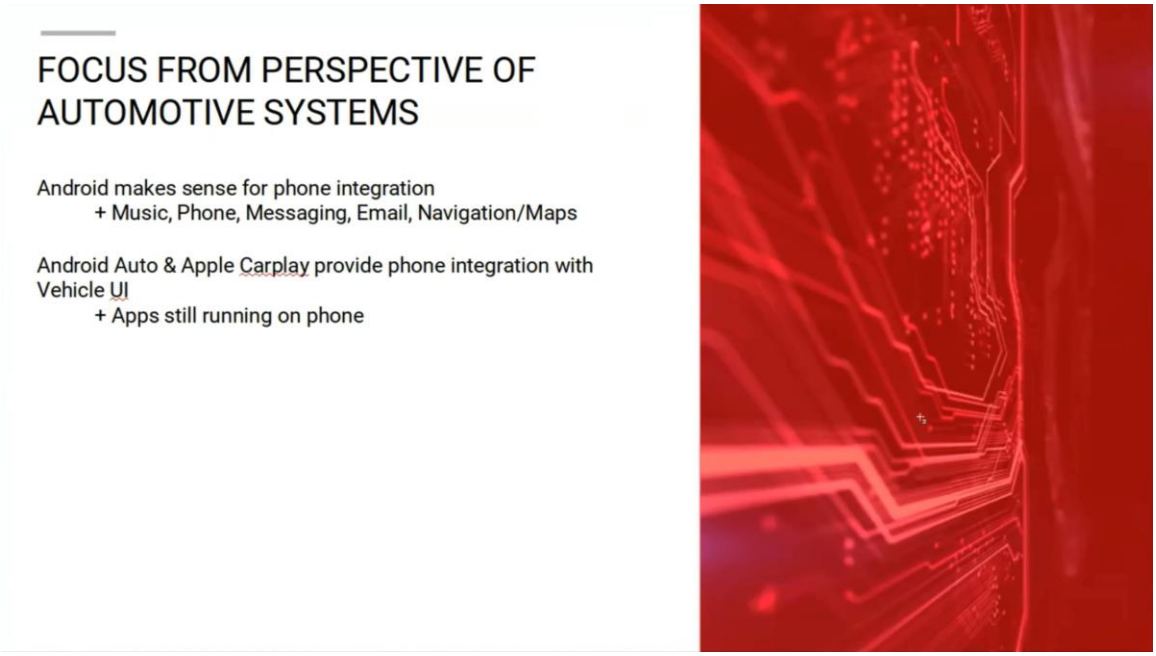


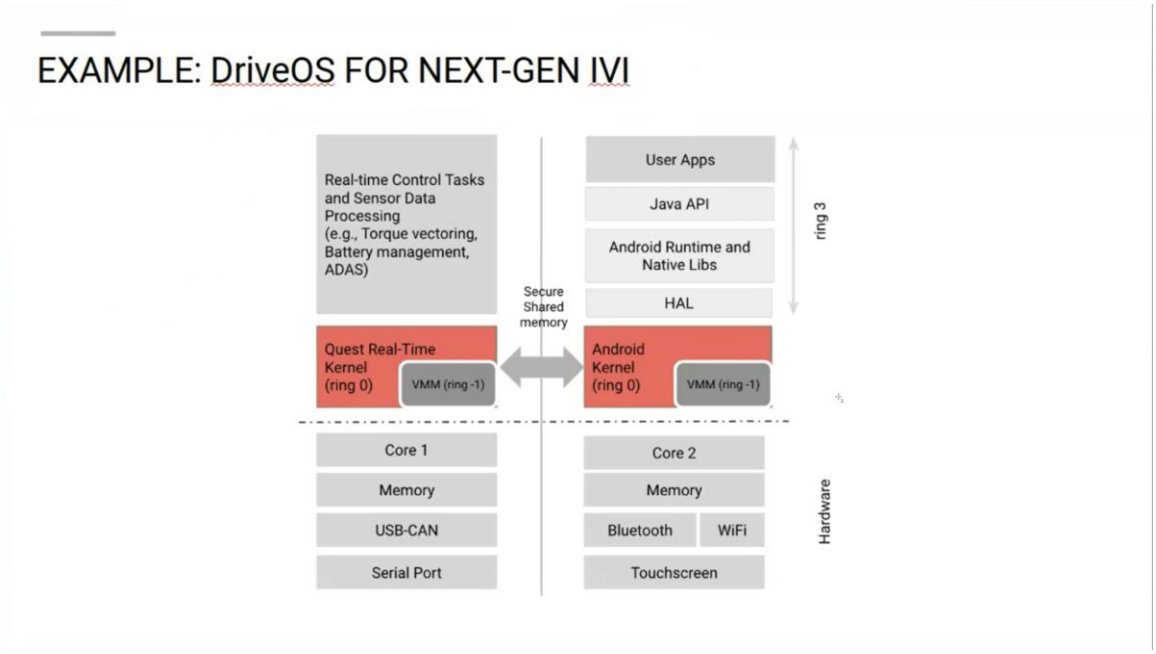
If we consider similar market, so this is a very interesting activity for my carrier today actually so to build the same ecosystem as what android has a lot, but we can make it

for an electrical vehicles or market, so I believe that the answer is yes, and I believe that it should look like this. Now the challenge is to build an ecosystem or Community so that's the most challenging part actually is not a technical perspective. So for me, the challenge is how we can make that community, so Linux is a Community android is a Community, how we can make this as a Community, this is my current understanding to the challenge in the real world to make and drive for autonomous machines, we need a community.

[Rich West]

So let me just share something else, so we have a HotMobile paper on exactly this and the company that I am working with has a group out in Finland that's basically been building an instrument cluster and in-vehicle Infotainment system for the vehicle that we're developing and we started out using android integrating android as a guest operating system.



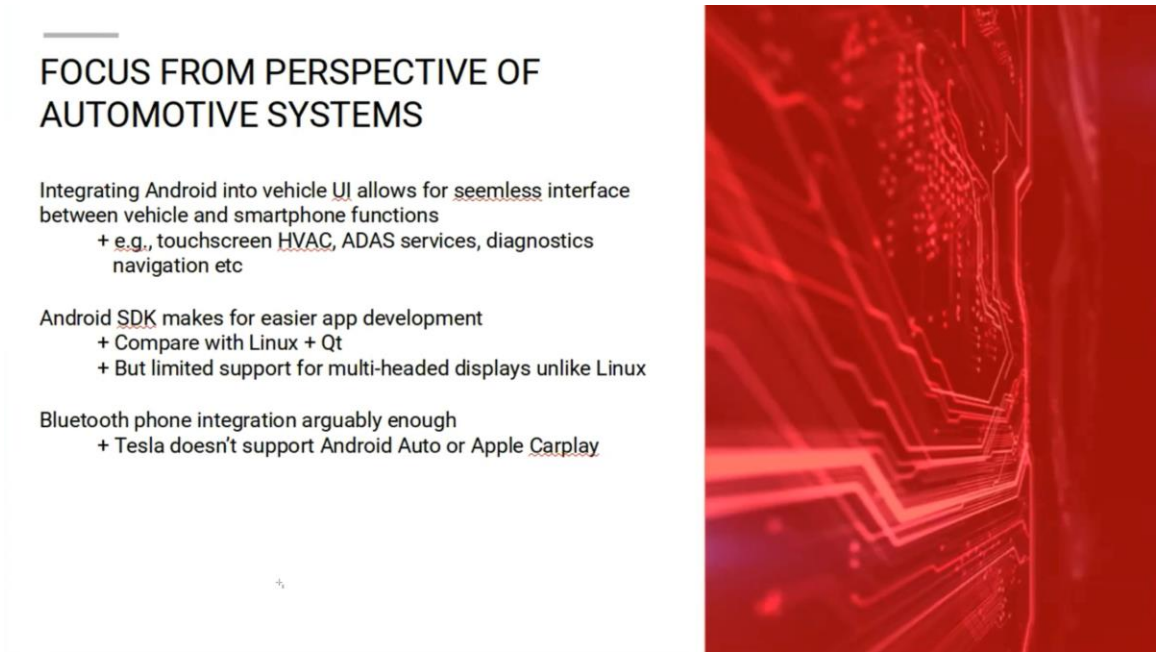


In our vehicle management system, so here's an example of just a very, very simple version, this is just a two core platform, but you can imagine many core platform and we've actually got a many core incarnation of our system. So on the left here we've got a sandbox petition hypervisor setup where you've got your real time operating system talking to your critical devices, particularly the CAN bus interface. And it is doing all your real time sensing and control functions. And, on the right hand side we've got a hand and android stack which basically provides the user interface to the user.

Now you know Shinpei is right, I agree with him about trying to build an ecosystem around android just like we're building around Linux and clearly it makes sense to integrate android into let's say a vehicle.

To do all the traditional things that you would have on a smartphone music messaging making phone calls exchanging emails and doing navigation. We've already seen the Android auto and Apple CarPlay allow the integration of a smartphone into a vehicle where you basically use the vehicles head unit it is display for displaying an interface to the phone applications, but the applications are still running on the phone.

Where we need to go is, we need to go to the point where you can go beyond the applications that you would have on the phone, and maybe use the android stack with it is convenient software development kit to define an integrated interface to work with core vehicle functions, such as your heating ventilation and air conditioning or through a touchscreen or a voice activation feature to enable ADAS services, for example.



Now these, the question I have, though, is why do we need android to do this, we started using android but we ended up scrapping it and we've gone with Linux plus qt to build our interface.

And one of the reasons we ended up doing this is because android that very poor functionality to support multiple headed display. If you want to support things like an instrument cluster or read only instrument cluster with a touch screen in vehicle interface to the side and android is very poor at doing that, but Linux is perfectly capable of managing multiple screens at same time.

And the telling point here is, if you look at tesla, tesla doesn't even support android or apple car play and they've gone and implemented their own user interface using Linux.

So right now I am not sure that android is really going to give you anything at the end of the day, that's just my statement.

[Jan Staschulat]

Maybe I jump in when you're talking about an android having something similar like android in like an automotive OS, I asked myself the question you know what are the specific features of android that are beneficial for automotive operating system, so of course it is open source and it is based on existing well-established technology like Linux. But automotive, one question you can ask yourself in terms of reliability if you have an open source and you're running a car with it that that runs basically your control algorithms and all that stuff it is real time capable, and if that basically fails who's going to be responsible for this.

A second thing, if a company would develop this what's kind of the business model behind it, but I don't want to go into that kind of discussion because I guess here I mean the audience here, I would say, I would rather focus on the technical feasibility.

And in my opinion, I agree that having a kind of android for automotive it is very beneficial to have an easy integration and also to do easier development of new features, but where I see the key difference is that while android is based on Linux and way do scheduling for phone, if you want to do real time applications in a car, you do not only need a real time operating system, and I would argue there's so many open issues with this complex hardware, we have shared memory, you have GPUs and all these things have not been addressed sufficiently by academic community, then you cannot just make an operating system out of it.

That's why I would let's say invite or we need here definitely a closer collaboration between academia and business and then product development here stronger collaboration to get to address these open issues.

[Andrei Kholodnyi]

Okay, so let me probably continue.

You know, when I look at this question will be android and quotes rights for an autonomous machines, I think android is, by definition, an mobile operating system and the based on Linux kernel, and I think if you want to try some sense, so it exists already and I mean it is ROS.

And this is there, I think, if we has a strong ecosystem, it has various flavors and it really depends on the industry where you want to try, because if you look at it, for instance, and this is actually approved by Android, Android is very successful in the mobile devices, but if you go and try to apply this to car industry, it hits its limits, for instance, like an automotive you don't have support for that for that for automotive buses, or for multiple displays, because it was not designed for that.

For ROS, in different industries, there are different domain specifics. You can speak about, for instance, after where was dedicated created based on ROS to address automotive autonomous. There are other big communities where there is also specifics like there is a ROS agriculture where people implement specific domain language more or less, there is drawings, where you have a different like PX4 not ROS operating system.

There is also like it could be like a real time and non real time depends on what kind of use cases you want to address. But I think ROS is a good platform to try real time in various areas, and then, depending on the areas to find the real time application examples where you can imply there are different requirements for different industries.

[Shaoshan Liu]

I think there will be an Android but far away, you don't have Linux or unix for robotic yet you don't have the Intel arm for robotics yet. We don't even have a solid end-to-end time model or generic time model for robotics yet, so I think in the next few years, we should focus on building up those foundations before we push very hard for the android route commercially.

Same thing happened in cell phone agents well there are different kinds of operating system different kind of cell phones in the market for about 10-15 years, smartphone market, and then Android came along, ios came along, and became a consolidated market, but then that builds on the foundation that the hardware is ready, the basic software already that we are still far from yet today.

So that's my take, I will spend more time, focusing on how to build the computer architecture for different kinds of autonomous machines to build the basic like the unix like basic operating system or RTOS for different types of autonomous machines yet, but the general Android user interfacing operating system, I think we're not there yet, we're still far from it.

[Nan Guan]

So we already see different opinions, so we don't have much time, but I want you guys to not be nice to each other, just say your different opinions.

[Rich West]

First of all, I don't necessarily buy any of this ROS stuff, so I am going to completely you know start by talking all of my panelists here. I think ROS is just a bunch of middleware we've seen all this before, by the way we saw CORBA in the late 1990s with middleware and the object management group developed Cobra and so forth now we've got ROS to and so forth, but at the end of the day, until you build the underlying operating system to have the real time requirements, you've got a problem.

And also just to talk about android at the end of the day, with android it is a user interface environment really it basically builds on top of Linux to give you a convenient user interface for certain applications that you might want to integrate into your autonomous vehicle.

I am not convinced it is the right way, given my experience and we started out actually playing with android I just wanna let you know that we actually started integrating it into our system and backed away.

Anyway, i'll just start by that by sort of antagonizing my panelists with my concerns about ROS and ROS2.

[Jan Staschulat]

I have been working in the scope of micro ROS basically putting ROS2 on microcontrollers and addressing lots of real time issues, and actually we are working in the real time working group very hard to connect basically the two worlds, I mean ROS2 and real time operating system. And the idea is to have an interface in ROS2 abstraction that allows you to access all features of the real time operating system.

So it allows you to at user level to use all the fancy algorithms, fixed-priority preemptive scheduling or reservation based scheduling and so forth, and as easy to use on ROS.

Because we see also in Bosch that we cannot develop everything from scratch, again we see community around ROS2 with all the simulation tools and environments and every

company has so many drivers available for all of that. Having a similar community for automotive makes absolutely sense. I also agree ROS2 is not there yet, but we're working on this.

[Rich West]

So, Jan, I have a few questions for you as an employee of Bosch. I think Germany is a big influence over certain standards in the suddenly the automotive world and the embedded world at large. But when I look at companies like tesla, tesla very disruptive and they've sort of thrown the rulebook out of the window and they're coming up with their own designs. Isn't there something for an opportunity to now rethink how we do everything from scratch, because we're taking on a new challenge now, particularly with autonomous vehicles.

And one of the things I do want to say is that I really do appreciate with what you've been saying, and that is you've been talking about pipeline processing. The sensing through to the actuation, I think we need a framework which can do real time end-to-end pipeline processing. If we can make that work in ROS, great. But that to me is the grand challenge right now.

[Jan Staschulat]

So I cannot say what we're working currently on but I recently that service oriented data driven activation scheme what you're saying, you have sense plan act, you have graphs that you're working on, and these kind of concepts are as independent of what kind of middleware you're using to express these kind of exclusion patterns, so to say.

ROS2 is well established in robotics and in automotive autonomous driving, you have a very similar application design or software architecture.

So that's the kind of a natural way to start looking for these kinds of concepts.

Okay that's all I can say now.

[Andrei Kholodnyi]

Because you are researchers and you want to go from some kind of you know, a paperwork and try it out, I think it is also pretty important to consider that Linux like ROS. There are a lot of developers, who knows Linux and doesn't know any other stuff like any other, you know operating system, and this is exactly what happens with unix and the BSD and all this stuff, and this would happen also to real time operating systems.

I think the majority will be then real time Linux, and if you consider your research, you also need to keep this in mind to try to work with the open source community on that and I think the times where proprietary operating system, so they still have their niche and the good reason to be there, but this wealth of community is really behind probably real-time Linux. So, for other applications like it could be that you would use FreeRTOS safer. It could be but, I think the 99% or 90% you'll be real time Linux.

[Rich West]

I have a question about that, I mean you are in Wind River, vxworks, Wind River hypervisor, there's other competing technologies, such as the QNX hypervisor, etc., there are lots of different real time operating systems out there and hypervisor technologies.

So there's an opportunity to integrate traditional general purpose operating systems such a Linux with custom RTOS on the side, we don't need to take Linux and make it only real time on its own, we can augment it with other.

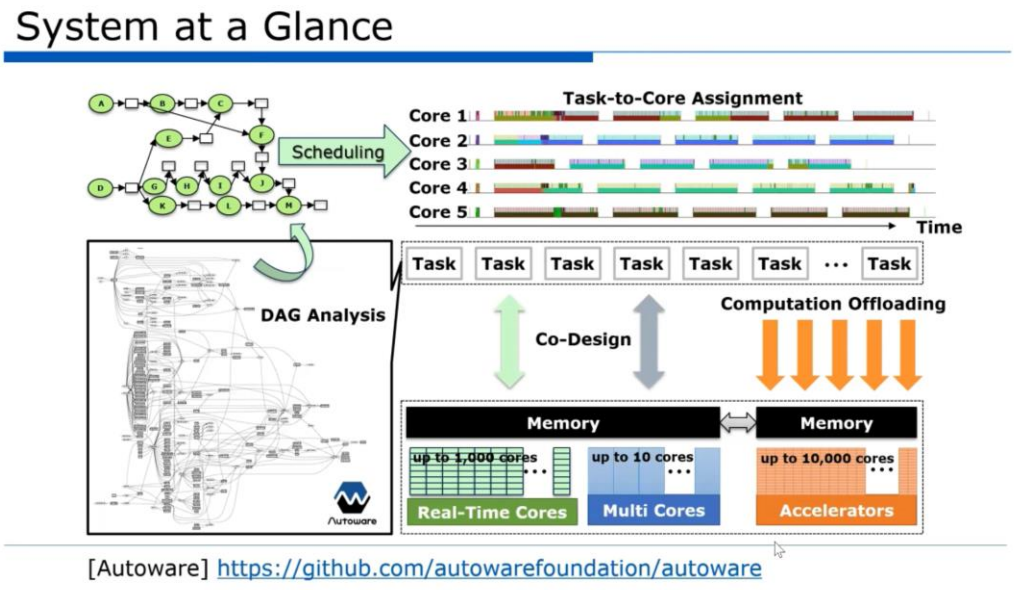
[Andrei Kholodnyi]

That's right, I mean this one of the possibilities and I would like to mention that I mean there are different types of RTOSes, so there is a micro-kernel versus monolithic. I think there is also a good research area that everything probably will move into some kind of uni-kernel design or uni-kernel real time operating systems and then you would have probably non real time environment computation environment, and within this computation environment, you would have islands of real time computation workloads which would run. It would be not like everything would be real time is not realistic, keeping in mind what we have discussed, very complicated heterogeneous systems with fpga gpus everything you know, like on one soc it is completely unrealistic to make a real time operating system in the aggregate of this.

It will be like computation islands, of course, but what I am trying to say if people would go in and compare I would use real time Linux instead of proprietary real time operating system if there is no good reason of using real time operating system proprietary operating system, people would go and use real time Linux instead, right so. And there are good reasons for using proprietary operating systems, real time operating systems, I could say as a manufacturer of such a operating system and our system now on Mars running on Mars, as well as also Linux but you need to keep in mind that you're so what was the first operating system you would go and try would be Linux.

[Shinpei Kato]

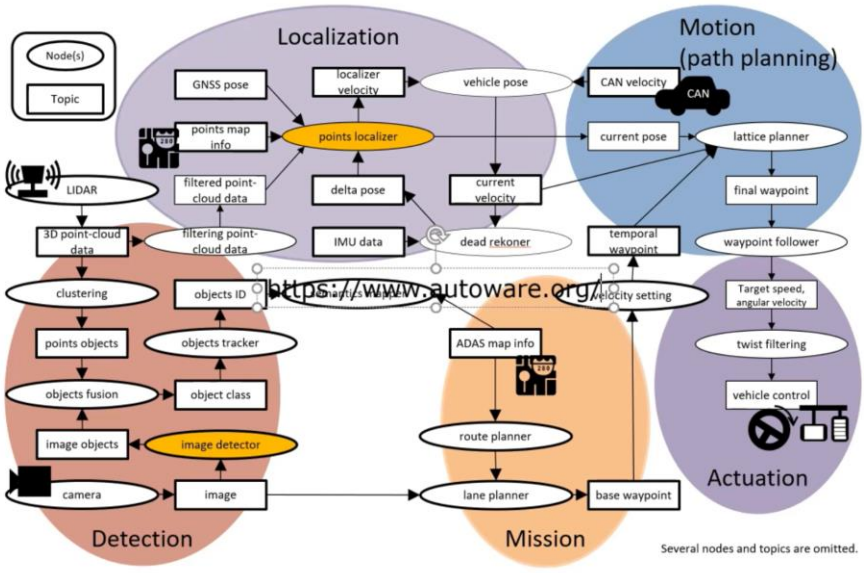
I actually have some proposals to the community.



Let me be a bit theoretical here. Taking the conversations from today's panel discussion I really enjoy so we show that, yes, the computing platform is heterogeneous and the task model process set look like a graph.

Now we have to solve the problem of many competing resource management, but, for example, scheduling. I believe that most of people from this community is very good at solving, for example, if you look at the system as a graph.

If somebody say hey do you know how to derive the upper bound on the latency between two nodes highlighted by yellow. I believe that this Community has some answer.



I am running open source community, which is called Autoware Foundation, which is built on top of ROS2, which will be actually used for an MIH open platform. We all have questions about how to derive such latency issues or schedulability issues, so if some of you are interested in contributing to the open source community by solving such problems, I am really happy to coordinate the working group at Autoware foundation, probably as well as ROS committee. I actually used to be, also at a ROS Steering Committee I just stepped down because I wanted to make room for Sony.

I want to organize a working group, where students or researchers can join to give their ideas about how to solve some problems which will be very, very useful for the open source community, because they actually lack theoretical background. They are very good at engineering technology, but sometimes we don't have the specific domains theoretical solution. Just want to let you know I run open source community, so if you're interested in, you can just come around, there are many working groups which for free, you can join and very soon we're going to launch a new working group, which is a real time for automotive autonomous vehicles.

If some of you are interested in contributing to open source community I am really happy to talk to you guys probably as an extension to this panel.

I want to organize a working group a foundation for real time issues, so if some of you are interested in contributing to those Open Source ecosystem, you are very encouraged. thanks.