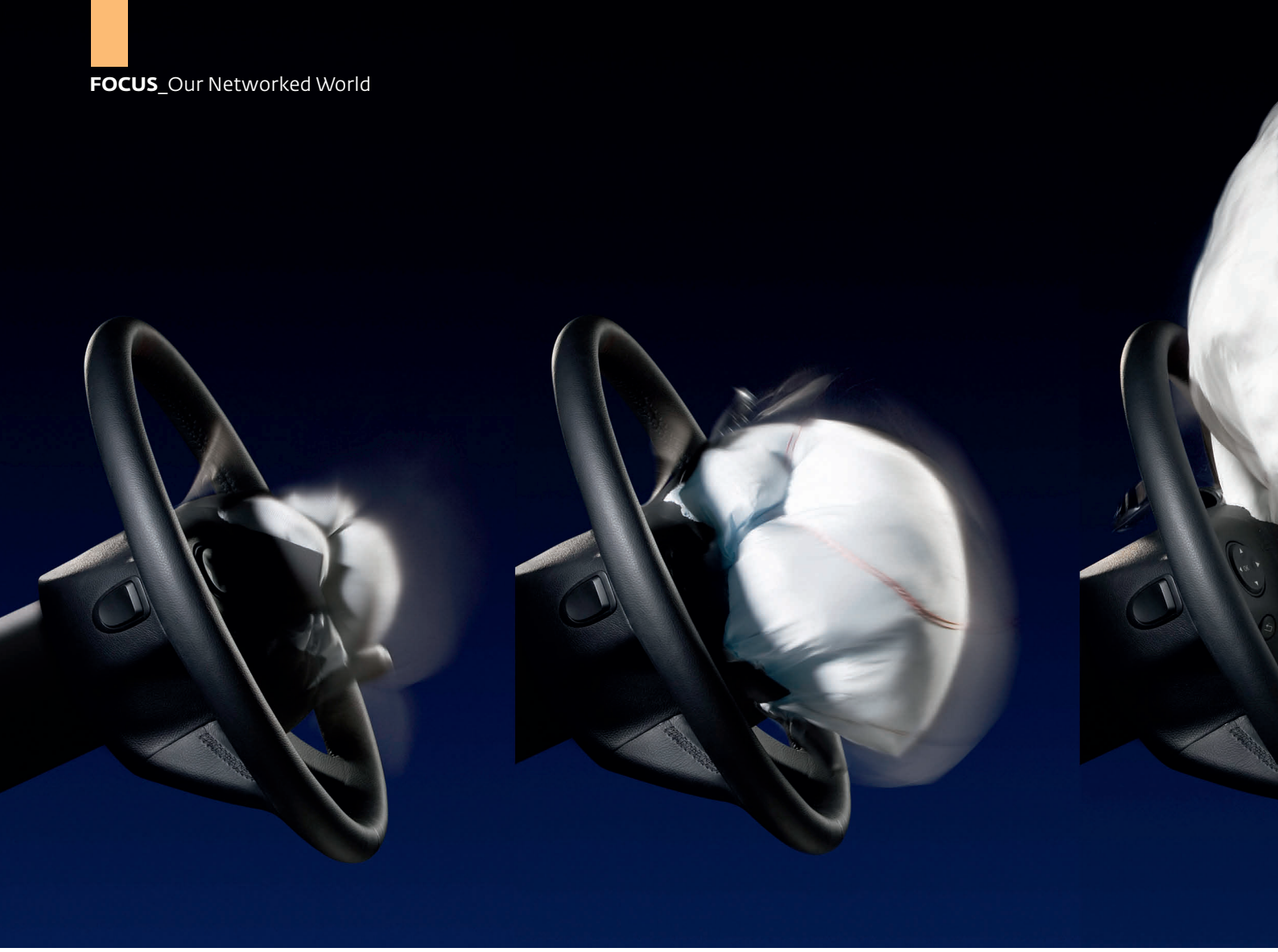


On Time – For Sure

When a computer takes forever to load a website, it may be annoying, but it is nothing more serious than that. If, however, the electronics in a car or a plane don't process commands exactly when they are supposed to, the consequences can be fatal. **Björn Brandenburg** and his team at the **Max Planck Institute for Software Systems** in Kaiserslautern and Saarbrücken study how to construct real-time systems in such a way that it can be proven that they always react on time.

TEXT **ALEXANDER STIRN**

Photo: Daimler AG

When fractions of seconds are crucial for survival: The airbag must be inflated before the driver's head hits the steering wheel. Its control unit must therefore trigger exactly when it is supposed to.

Punctuality is undervalued – at least in computer science. If the airbag in a car is triggered a few milliseconds too late because its control unit is preoccupied with other tasks, the driver or other occupants are at risk. If a cell phone misses the instant in which it has permission from its cell tower to communicate, the data connection fails. If a pilot wants to land a plane but his commands don't reach the engines or landing flaps in time, this can have fatal consequences.

"The physical world doesn't stop just because a computer fails to keep up," explains Björn Brandenburg, head of a junior research group at the Max Planck Institute for Software Systems in Kaiserslautern and Saarbrücken. The computer scientist deals with "real-time systems," as they are known – applications for which unpredictable delays aren't an option.

The theoretical and practical implementation of such systems presents computer science with major challenges: a program is typically deemed correct if it produces the correct output for a given input. "For us, on the other hand, a system is considered correct only if it delivers the right output at the right time," says Brandenburg.

As head of the research group on real-time systems, the 30-year-old therefore studies how such systems can be guaranteed to operate safely and reliably in an increasingly complex and networked environment. While experiments and intuition still play an important role in practice, Brandenburg relies on hard mathematics: "For safety-critical applications, we need analysis methods that are mathematically sound and that accurately prove that a system always operates as required," says the Max Planck researcher.

The demands on real-time systems are thus fundamentally different than everyday IT concerns: no mathematical models are needed to determine wheth-

Advocates for punctuality: Björn Brandenburg (right) and Alexander Wieder develop methods to prove that safety-critical systems perform the job that they are expected to – on time, every time.

er the program window with the e-mail that you just read closes as quickly as desired. Even if absolutely nothing happens sometimes, the user might curse, but the world certainly won't come to an end. Developers have to trust their experience and their ideas when it comes to improving these types of systems. They look at the program code and test the improvements extensively. "If it usually works, that's completely okay for general-purpose systems," says Brandenburg.

However, usual or average values aren't sufficient in safety-critical real-time systems such as a car's airbag. The product needs to work. Period. "Generally speaking, systems have now become so complex that humans can't reliably anticipate worst-case behavior using their intuition alone," says Brandenburg.

MORE THAN A HUNDRED MICROCONTROLLERS IN A CAR

A modern luxury car contains more than a hundred microcontrollers. These control each and every system, from the airbag to the engine control unit to the radio. Most of these microcontrollers perform several tasks, which is what makes the mathematical description so complicated: "The more components interact with each other, the more difficult it is to rule out something going wrong," says Björn Brandenburg.

The computer scientist compares the situation with an office. The employee working in the office, who in this scenario represents a processor, is under pressure. His boss constantly wants him to do something – those jobs need to be performed right away. But his colleagues also have questions and don't want to hang around all day waiting for answers to their concerns.

The office works only if all jobs are executed within the required timeframe, and if the poor employee doesn't put off a mountain of work when it's time to leave the office for the day. Above all, however, the individual demands, particularly those of the boss, mustn't be left hanging for so long that the next request already shows up with the same priority. Otherwise it will all end in chaos. Computer scientists refer to this as non-linear behavior, which suddenly leads to jumps in response time.

"To be able to describe such mechanisms mathematically, we need models that represent the world correctly and that we can also use to prove that the systems fulfill the requirements they were designed to fulfill," says Brandenburg. Therefore, in the case of the office – or a simple real-time system – computer scientists use mathematical equations to deal with the frequency of the individual jobs, the work needed to process them and the required timeframe within which a response is needed. The formula system often turns out to be so complicated that it can no longer be resolved analytically. In such cases, computer scientists need to look for a solution step by step, by assuming a start value and consistently refining it – a standard procedure that mathematicians refer to as fixed-point iteration.

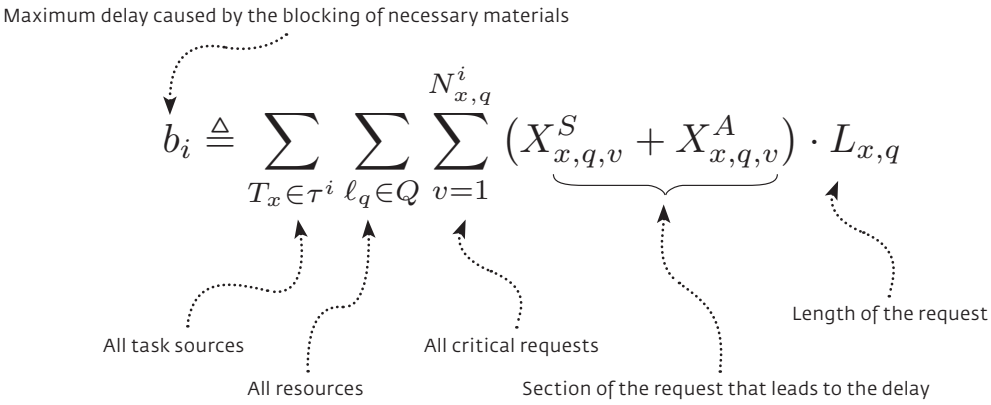
Another problem is finding realistic minimum and maximum values for the work involved, the frequency, and the response time permitted. The formulas can be put through their paces only if such information is available. "In practice, a very experienced engineer often performs this task, doing some testing and then adding in a safety margin," says Brandenburg. "In many cases, that can provide a sufficiently good estimate."

MODERN PROCESSORS ARE LESS PREDICTABLE

In the case of the airbag, the maximum permitted period of computation is derived from the time that elapses from the moment the crash happens to the moment the driver's head hits the steering wheel. The airbag must be completely inflated by then, and this information can then be used to deduce how quickly the software needs to respond. "Ultimately, the deadline is always derived from the physical demands on the system," says Brandenburg.

The engine is no exception. The permitted engine speed dictates the minimum time between two processes. If, for example, the exhaust gas concentration needs to be read once every revo-

Photo: Thomas Hartmann



A formula for delay: The Max Planck researchers study real-time systems whose processors share other resources. The systems might perform a safety-critical job too late because they can't access a particular resource. The maximum delay is derived from the sum (Σ) of all sections of requests that lead to delays, and the respective length of these requests. The sections of requests may belong to various tasks that originate from different sources and that rely on several resources. In order to be able to calculate the delay, it is crucial to identify which sections of the requests $X^S_{x,q,v}$ and $X^A_{x,q,v}$, lead to the delay. To do this, Björn Brandenburg and his colleagues use a method known as linear optimization.

lution, a maximum engine speed of 6,000 revs per minute (or 100 revs per second) gives a smallest-possible break of one hundredth of a second.

It's considerably more complicated to determine the actual computing time of a modern system and thus estimate whether the work can also be done within the specified timeframe. It was comparatively easy to do this with previous processor generations. In those days, engineers could tell from the machine code how many computing steps were necessary to execute a job. As the processor's operation cycle was known, it could be used to calculate the time required.

Modern processors are much less predictable: they predict the control flow and try to pre-compute the intermediate results that will likely be needed. They can scale their clock frequency. They employ a hierarchy of caches. A distinct field of research in computer science, known as worst-case execution time analysis, deals solely with this challenge – determining the maximum computing time for a specific program and a specific piece of hardware in a worst-case scenario.

"It's incredibly difficult to precisely bound the worst case," says Brandenburg. "That's why it can be useful to monitor real systems and extract samples from them." For example, a software program logs all the commands that are processed during a test drive in a car. Engineers can derive data from this information, such as the maximum execution time. A small safety buffer is then built into this to give the output values for the computerized analysis of the programs. "The purists in our field would say that this involves a measurement. It can't be used to prove with strict mathematical precision that it really is the worst-case scenario," says Brandenburg. "I take a more pragmatic view. In any case, an analysis of response times is better than some spreadsheet calculation into which a few more or less arbitrary figures have been typed."

The final response time calculation comes into play only after all requirements and potential sources of delay have been factored into the model. "If we can then show that, even in the worst-case scenarios, the response time never exceeds the specified deadline, we know that it's okay, the system is safe," says Alexander Wieder, a doctoral student in Brandenburg's research group.

ENGINEERS DEFINE THE REQUIREMENTS

Mathematicians usually prove something like this using a method they refer to as "proof by contradiction" or "indirect proof". The researchers assume that the specified response time has been exceeded – in other words, that the event they wanted to rule out has actually occurred. Then they see what conclusions they can draw. If the job wasn't performed as planned, there are two options: either it took longer than expected to process a job, or the processor must have done something different while it should have been performing the safety-critical job – so it must have been busy working on higher-priority requests.

The computer scientists then look more closely at all the processes and analyze how much work they involve. The method is gradually refined until, eventually, there are a few processes left that, under the given assumptions, involve more work than the model allows for. A contradiction. "For us, it means that either our model is wrong, or something like this is impossible," says Brandenburg. The specified response time would therefore never be exceeded. The system is 100 percent safe – at least under the given model assumptions.

The models are based on demands that are defined by engineers. Beyond these specifications – if the engine runs at more than 6,000 rpm, for example – the mathematical proof doesn't guarantee safety. The software may still function, but it isn't guaranteed do so. The engineers therefore need to apply their expertise to define safe and complete demands on the system.

Further, a type of "volatile boss", who assigns several jobs within a short time and then just retreats into his office, is frequently simulated. Computer scientists refer to this phenomenon as "bursty arrivals". It is easily expressed

Graphic: MPI for Software Systems

left: Interpreting between two vernaculars: Björn Brandenburg understands both the engineers who develop the real-time systems and the theorists who want to prove their reliability. That is why he wants to liaise between the two disciplines.

right page: In order for a plane to be able to land safely, the electronics must communicate the pilot's control commands to the engines or landing flaps within a specified timeframe.

in algorithms, even if the proof becomes more involved as a result.

However, real-time applications really become complicated when they are processed on a processor that needs to share other resources with additional processors. For example, multiple processors may jointly access a message buffer that stores sensor data until it can be processed further. Research at the Max Planck Institute for Software Systems focuses heavily on systems with shared resources.

To use the office analogy, where an employee represents a processor, this means that the office employees who normally work undisturbed side by side now have only one photocopier or one telephone between them. This inevitably leads to discussions. Some employees urgently need to finish a job; others actually have more time to spare, but don't want to wait. "Whether all deadlines can be met can depend on the sequence in which each person takes a turn," says Björn Brandenburg.

In practice, computer scientists pursue a variety of approaches: The processors may use the resource in the order in which they request access, in order of priority, or just in a random order determined by the underlying hardware. When it comes to the question of how the individual processors should spend the waiting time, several approaches can be taken: They can ask incessantly: "Can I? Can I? Can I?" until it is finally their turn, or they can wait to be invited to use the photocopier.

Theoretically, waiting is the better solution, as the employees can use the time to tackle other jobs that aren't quite as important. In practice, however, constantly switching from photocopying to talking on the phone can entail a considerable amount of additional work, with nothing really getting done. In critical systems, such as a car, the method based on "busy-waiting" – that is, the one that involves constantly asking "Can I?" – is typically preferred.

Brandenburg and his group aren't the first to address this problem. To date, however, computer scientists have usually analyzed the shared resources manually: they have considered possible delays and calculated a maximum value for the response time based on their considerations. But with this method, it takes just one incorrect assumption to arrive at a result that is no longer certain. Furthermore, the estimates turn out to be incredibly pessimistic. The results are similarly unrealistic. "The engineers say: 'Nice that you have a safe bound, but in practice, the response time will never be that high. It's useless,'" says Brandenburg.

AN UPPER BOUND FOR POTENTIAL BLOCKING

The Max Planck researchers have thus chosen a different approach. First, they create a complete set of workflows that are theoretically not impossible in an office – or a real-time system – and therefore can't immediately be ruled out. Then they gradually identify all the scenarios that can't happen in practice. For example, it's inconceivable that an office employee will want to use all of the equipment simultaneously.

In this way, dozens or even hundreds of constraints are ultimately accumulated. All of them can be expressed as linear inequalities. The number of scenarios that aren't ruled out gradually shrinks until an upper bound on the maximum delay encountered by the planned workflows is found. This bound can then be incorporated as another factor in the algorithms that prove the punctuality of the real-time systems.

The underlying method is known as "linear optimization," and has been explored in mathematics for more than 60 years. Computer scientists have since developed a number of fast analysis systems whose algorithms the Max Planck researchers can adapt to real-time systems. Brandenburg says: "We can now analyze significantly more complicated systems, and we have made considerable progress in terms of accuracy."

All of this is particularly important in practice because many real-time systems now use multi-core processors. With these, the processing tasks are shared among two or more independent central processing units, or cores. The cores access shared resources and perform their individual tasks in parallel. "Multi-core processors represent a

Photo: Thomas Hartmann

huge challenge for real-time systems," says Björn Brandenburg.

Computer science is lagging behind here. For a long time, in the context of real-time systems, multi-core processors were considered to be primarily of theoretical interest, and not to be installed in time-critical applications. Now, the requirements even in cars are so high that multi-core processors have become almost standard. The proven algorithms no longer work here; everything needs to be developed from scratch. "The argument previously used, that if a processor focuses on one job, then other, potentially disruptive activities can't take place at the same time, is an incredibly powerful element in the reasoning," says Brandenburg. "That no longer works in multi-core systems – I never know what the other cores are doing at any given time."

In particular, the maximum execution time, which is very hard to determine even on modern single-core processors, presents computer scientists with major challenges. The question of determining exact response times, and not just safe upper bounds, remains open. What it looks like, how it can be described in formulas and how the analysis can be improved as a result, remains unresolved.

Still another – considerably more human – problem is troubling the Max Planck researchers. Many theorists who puzzle over analysis algorithms for multi-core processors are math experts, but rarely do they sit in a lab and write programs. Engineers in industry, on the other hand, are highly specialized and can use the tricks of their trade to operate even the most obscure systems. But they aren't necessarily the best mathematicians.

Instead, they try to squeeze their experience into several thousand pages of comprehensive rules, which – as in automotive engineering, for instance – specify in detail the demands made on every system. This, in turn, doesn't please the theorists, who prefer specifications to be as simple as possible. "Part of my research work consists in sitting in the middle and saying: I understand the mathematicians, I understand the practitioners, I'll try to interpret between the two sides," says Brandenburg. This is a timely approach, particularly in the area of real-time systems. ◀

TO THE POINT

- **When it comes to the safety of cars or planes, the electronics must process data with absolute reliability within a specified timeframe. Experiments can't prove with absolute certainty whether such real-time systems actually perform their jobs on time. Mathematical proof is necessary.**
- **Modern electronic processors have several CPU cores, process several tasks simultaneously and vary their clock frequency. This makes the mathematical description of such systems, and the proof that they function correctly, very complicated.**
- **In some ways, Max Planck researchers are breaking the mold in their reasoning, for example by leveraging linear optimization. This allows them to analyze more complicated systems with greater accuracy than before.**

GLOSSARY

Real-time system: A system, such as an electronic control unit or a microcomputer, that needs to handle a process within a specified timeframe. Real-time systems are used in safety-critical tasks, such as controlling an airbag.

Linear optimization: Techniques for finding an optimal solution to a problem that can be described by a linear equation subject to a set of constraints that are represented as linear inequalities.

Proof by contradiction: Indirect proof that proves a proposition by refuting its opposite. The proposition to be refuted is reduced to such an extent that a contradiction emerges as a substantiated proposition.

Photo: istockphoto / Jaromír Chalabala