**The Number Glitch that can lead to Catastrophe**

Chris Baraniuk

About the author

Chris Baraniuk is a science and technology journalist based in London. He writes for New Scientist, Wired and The Economist among others and has a website at [**chrisbaraniuk.com**](http://chrisbaraniuk.com). His Twitter handle is [**@machinestarts**](https://twitter.com/machinestarts).

(Credit: Getty Images)

**A surprisingly simple bug afflicts computers controlling planes, spacecraft and more – they get confused by big numbers. As Chris Baraniuk discovers, the glitch has led to explosions, missing space probes and more.**

**Related**

**[Press me! The buttons that lie](http://www.bbc.com/future/story/20150415-the-buttons-that-do-nothing)**

**[Why the fax machine won’t die](http://www.bbc.com/future/story/20150224-why-the-fax-machine-wont-die)**

**[The hidden ways traffic flows](http://www.bbc.com/future/story/20150115-the-hidden-ways-traffic-flows)**

**[The secret world of… beer](http://www.bbc.com/future/story/20141210-the-secret-world-of-beer)**

**[Junk mail's subliminal messages](http://www.bbc.com/future/story/20141105-the-secrets-of-junk-mail)**

Tuesday, 4 June 1996 will forever be remembered as a dark day for the European Space Agency (Esa). The first flight of the crewless Ariane 5 rocket, carrying with it four very expensive scientific satellites, [ended after 39 seconds in an unholy ball of smoke and fire](http://youtu.be/gp_D8r-2hwk?t=5s). It’s estimated that the explosion resulted in a loss of $370m (£240m).

Why is the number 2,147,483,647 important? **— Anything larger confuses many computers**

What happened? It wasn’t a mechanical failure or an act of sabotage. No, the launch ended in disaster thanks to a simple software bug. A computer getting its maths wrong – essentially getting overwhelmed by a number bigger than it expected.

How is it possible that computers get befuddled by numbers in this way? It turns out such errors are answerable for a series of disasters and mishaps in recent years, destroying rockets, making space probes go missing, and sending missiles off-target. So, what are these bugs, and why do they happen?

Imagine trying to represent a value of, say, 105,350 miles on an odometer that has a maximum value of 99,999. The counter would “roll over” to 00,000 and then count up to 5,350, the remaining value. This is the same species of inaccuracy that doomed the 1996 Ariane 5 launch. More technically, it’s called “integer overflow”, essentially meaning that numbers are too big to be stored in a computer system, and sometimes this can cause malfunction.

**Failure to launch**

A [full investigation of the Ariane incident](https://www.ima.umn.edu/~arnold/disasters/ariane5rep.html) found that a process left over from software in the previous generation of rockets, Ariane 4, had captured an unexpectedly high reading for the sideways velocity of the newer, faster vehicle – and the Ariane 5 rocket’s software couldn’t handle this high figure. A self-destruct sequence was initiated. [A couple of seconds later, the rocket was history](http://www.around.com/ariane.html), as the video below shows.

Such glitches emerge with surprising frequency. It’s suspected that the reason why Nasa lost contact with the Deep Impact space probe in 2013 was an [integer limit being reached](http://en.wikipedia.org/wiki/Deep_Impact_%28spacecraft%29#Contact_lost_and_end_of_mission).

And just last week it [was reported that Boeing 787 aircraft](http://www.theregister.co.uk/2015/05/01/787_software_bug_can_shut_down_planes_generators/) may suffer from a similar issue. The control unit managing the delivery of power to the plane’s engines will automatically enter a failsafe mode – and shut down the engines – if it has been left on for over 248 days. Hypothetically, the engines could suddenly halt even in mid-flight. The [Federal Aviation Administration’s directive on the matter](https://s3.amazonaws.com/public-inspection.federalregister.gov/2015-10066.pdf) states that a counter in the control unit’s software will “overflow” after this specific period of time, causing an error. Although scant details have been released – the FAA and Boeing declined to comment for this article – [some amateur observers have pointed out](https://twitter.com/FioraAeterna/status/594110518203260929) that 248 days (when counted in 100ths of a second) is equal to the number 2,147,483,647 – which is significant.

How so? It just so happens that 2,147,483,647 is the maximum positive value that can be stored by a “32-bit signed register”, commonly installed on many computer systems. On Ariane, by comparison, the software was using a "16-bit" space, which is much smaller and only capable of storing a maximum value of 32,767.
 **Puzzling limit**

Numbers are infinite, so why choose such limited storage spaces for them? The answer is that computers have traditionally demanded efficiency in all things. Storage space used to be much more costly than it is today and processing larger values took longer. If you kept to certain limits, software was expected to run more smoothly. Rocket guidance systems do a lot of critical number crunching very quickly, so these overheads certainly matter. The problem with that, as the Ariane 5 proved, is that such limitations aren’t always foreseen as problematic.

“We have to recognise that in software we are always approximating reality,” explains Bill Scherlis, a software expert at Carnegie Mellon University. “There’s always an engineering trade-off between the cost of having a more precise representation and the benefit of the efficiency.”

(Credit: Getty Images)

Mathematician Douglas Arnold at the University of Minnesota includes the Ariane 5 incident on a web page entitled “[Some disasters attributable to bad numerical computing](https://www.ima.umn.edu/~arnold/disasters/disasters.html)”. Arnold also notes the 1991 case of a [Patriot missile which failed to intercept an Iraqi Scud attack on a US Army barracks during the Gulf War](https://www.ima.umn.edu/~arnold/disasters/patriot.html). In this case, an overflow error meant that the missile defence system mis-tracked the incoming Scud projectile, which was travelling at 1.7km/s, and instead scanned an area of airspace more than 500 metres from the target.

As a result, the Scud hit the barracks, killed 28 soldiers and injured a further 98 people. The full details of the computer bug in this case are quite complicated, but software engineer Andrew Lum at the University of Sydney has [posted a helpful explanation of what happened, including diagrams of the Patriot system](http://sydney.edu.au/engineering/it/~alum/patriot_bug.html).

**Gangnam glitch**

Not all rollover glitches are as destructive as these examples, but they do frequently create unexpected effects. For example, in the video game Civilization, an unanticipated bug in this vein caused the peaceful character Gandhi to become uncharacteristically hostile. When players chose a certain mode to play in, the value which defined Gandhi’s aggressiveness rolled backwards past zero to the maximum. Consequently, he [would threaten players with nuclear weapons at every turn](http://kotaku.com/why-gandhi-is-such-an-asshole-in-civilization-1653818245) – to the great amusement of many players.

And in December, [it was reported that Gangnam Style, the most popular video of all time on YouTube had “broken” the website’s view counter](http://www.bbc.com/news/world-asia-30288542). The counter had apparently been programmed to only run to 2,147,483,647 – again, the maximum positive value of a 32-bit signed register. It turned into good PR for YouTube, which updated the view count storage while wallowing in worldwide coverage of the site’s most popular ever video. The new maximum is well over nine quintillion.

Psy's Gangnam Style is credited with 'breaking' video-sharing website YouTube (Credit: Getty Images)

Scherlis notes that the previous limitation reveals the expectations of the original programmers who built YouTube. “Certainly, when YouTube’s software was first developed, I think it was probably hard for any developers or designers to imagine that they would overflow [this number],” he says.

It’s often this sort of assumption, which initially may seem reasonable, that causes problems years down the line. The most talked about overflow bug in history, which many will remember, was the [much-hyped Millennium Bug](http://en.wikipedia.org/wiki/Year_2000_problem). Although largely considered a damp squib, the Y2K problem did cause some headaches.

With Y2K, the bug was simple. What happens when you record years by the last two digits? 1900 becomes identical to 2000. Many people realised that this would cause confusion for any computer systems storing year values in this manner. As a result, [a lot of advice was published in advance to programmers](https://books.google.co.uk/books?id=TodvBDXmM_oC&pg=PA128&lpg=PA128&dq=rollover+glitch&source=bl&ots=OYk52Ohfd7&sig=wH1es7krfcVxkglhL4fNM5t9iGQ&hl=en&sa=X&ei=pavQVP75CMreao6HgMAG&ved=0CFAQ6AEwCTgK#v=onepage&q=rollover%20glitch&f=false) so that they could update systems before or on 1 January 2000. Planes did not fall from the sky, but there were [some interesting consequences](http://en.wikipedia.org/wiki/Year_2000_problem#Documented_errors). For instance, radiation detection equipment in the Japanese prefecture of Ishikawa crashed at midnight; 150 slot machines at a race track in Delaware failed; and several websites gave the new date as “1 January 19100”.

Fears of a global meltdown from the 'Millennium Bug' turned out to be unfounded (Credit: Getty Images)

Twelve years later, in a similar incident, a 105 year-old woman in Sweden named Anna Eriksson [received a letter inviting her to start pre-school](http://www.huffingtonpost.com/2012/11/20/anna-eriksson-105-invited-to-preschool_n_2167995.html) because software had been designed to contact individuals born in “07”. The designers intended this for people with a birthdate in 2007, not 1907 as was the case for Ms Eriksson. An inability to correctly recognise the year even led to [millions of German credit and debit cards becoming unusable on New Year’s Day in 2010](http://content.time.com/time/business/article/0%2C8599%2C1952305%2C00.html).

**The year 2038**

About 15 years ago programmer William Porquet had the idea of thinking ahead to yet another crucial date – GMT 3.14.07am on Tuesday 19 January 2038. This is the moment when the number of seconds since 1 January 1970 will exceed one of the maximum values [of many computers’ date and time registers nowadays](http://en.wikipedia.org/wiki/Unix_time). Like the Millennium Bug, failure to prepare for this could result in computer crashes.

“It was in 1999 that I first wrote about this,” comments Porquet. “I acquired the domain name [2038.org](http://www.2038.org/) and at first it was very tongue-in-cheek. It was almost a piece of satire, a kind of an in-joke with a lot of computer boffins who say, ‘oh yes we’ll fix that in 2037…’ But then I realised there are actually some issues with this.”

Will a January morning in 2038 see computers crashing all over the world? (Credit: Getty Images)

Porquet is concerned about old bits of software that nobody tends to anymore – on long-established networks, or on old hardware being used in remote parts of the world. How many of them will still be in use 23 years from now and what consequences that could have is anybody’s guess.

“A lot of computer systems,” notes Porquet, “can be caused to fail in a predictable manner. But this is failure in an unpredictable manner.”

**Glitch in time**

Markus Kuhn, a computer scientist at the University of Cambridge explains that time related bugs create interest partly because their consequences are unpredictable, but also because they are “not unexpected” and that people are able to speculate about what will happen when the fateful date arrives.

Kuhn thinks that the 2038 problem will be less significant than Y2K because the Millennium Bug has prepared the computer industry to make the necessary fixes. Indeed, that’s all part of William Porquet’s plan. “I hope it’s something that will take me out of semi-retirement for a very large sum of money,” he says, only half joking.

The speed of Earth's rotation may also cause a slight time change that could crash computers (Credit: Getty Images)

For Kuhn, the interesting time problem for computers of the moment is not an overflow glitch per se, but it is one this coming June. The year 2015 will be a second longer than 2014 thanks to a move to correct the discrepancy between astronomical time (the time on Earth based on our planet’s rotation) and atomic time (the most accurate known time-keeping method, in terms of counting seconds). While atomic time, which is to be adjusted with this year’s leap second, is [mind-bogglingly precise](http://www.nist.gov/pml/div688/nist-f2-atomic-clock-040314.cfm), it is actually slightly out of sync with astronomical time because the Earth’s rotation is very gradually decelerating. [Geological events such as earthquakes](http://www.nature.com/news/2004/041229/full/news041229-6.html) can cause changes in the speed of this rotation meaning that the addition of leap seconds, unlike leap years, is variable. The last one was in 2012 and [crashed many computers](http://www.wired.com/2012/07/leap-second-glitch-explained/all/). Fortunately, says Kuhn, we’ll hopefully be more prepared than we were in the past.

It seems like no matter what we do, certain numbers and calculations will always confuse computers, causing malfunction – or worse. “We’ve learned a lot from the Y2K experience and other similar events,” notes Scherlis. “But the reality in which we are always making approximations and having to navigate an engineering trade off? That is with us forever.”