AT&T crush

AT&T is a worldwide company leader in communications, media and entertainment, and technology. As mentioned on its website, in 2017, the company’s consolidated revenue was more than $160.5 billion. However, in 1990 there was a huge collapse on its system. The company who by the time had built a reputation and a large advertising campaign base on its reliability and security suffered with a bug on the software causing several problems for 9 hours.

According to Dennis Burke (1995), at 2:25pm on Monday, January 15th, an alarming number of red warning signals from all the different parts of their world-wide network began to be noticed. That caused almost 50% of the calls placed through AT&T to fail to go through.

Larry Seese, AT&T’s director of technology development at the time, said “The fault was in the code” (Neumann, P. G., 1990). Firstly let’s understand how the software worked. The whole system was based on a 114 computer-operated electronic switches (4ESS) spread across the United States. Each switch, which was capable of manage up to 700,000 calls per hour, were linked via a cascading network called Common Channel Signalling System No.7. A switch would verify a list of 14 different possible routes to complete a call, when a telephone call was received by the network. While the telephone number was passed to a parallel signalling network, alternate routes were checked to ascertain if the destination switch was able to deliver the call to its local company. If the destination switch was busy, a busy signal was sent from the original switch and the line was released. If it was available, a reservation would be made by a computer at the destination switch and ordered it to pass the call. This whole process took four to six seconds (Burke, D., 1995).

A team of 100 technicians identified that the problem started in New York City, while the New York switch had performed a routine self-test that indicated it was reaching its limits. As a measure of maintenance, the switch performed a reset operation and sent a message as a “congestion signal” over all the 114 switches that it would take no more calls until further notice. This reset process last only four seconds and when it finished, it dispersed a signal that it had had backed up during the off-line time. Thus, a cascade effect started when another switch received a message that a call from New York was on its way and began to update its records to show that the New York switch was back on line. But a second message from the New York switch arrived and because the first message had not yet been handled, the second message should have been saved until later. A software defect made the second message to be written and the software in the receiving switch detected the overwrite and activated a backup link while it reset itself, but “another pair of closely timed messages triggered the same response in the backup processor causing it to shut down also”. The problem repeated throughout the 114 switches within the network, which blocked over 50 million calls in the nine hours it took to stabilize the system (Burke, D., 1995).

"We made an improvement in the way we react to those messages so we can react more quickly. The first common channel signalling system 7 initial address message (caused by a call attempt) that switch B receives from switch A alerts B that A is back in service. Switch B then resets its internal logic to indicate that A is back in service," said Seese. This improvement is referred to an upgrade on the software, made months before, to speed the processing of certain types of messages.

A pseudocode by Burke (1995), show how the code is read.

|  |  |
| --- | --- |
|  |  |

The defect was a C program that featured a *break* statement within an if clause, nested with a switch clause. On line 7, if the switch is not empty, the program should have dropped out the if clause, processed the incoming message, and set up the pointer to the database on line 11. However, because of the break statement in the else clause on line 10, the program dropped out the case statement entirely and began doing optional parameter work on line 13.

The upgraded code had been tested rigorously, but the biggest problem with this bug, besides the one-line feature, was the use of C programs and compilers. (artigo) says that complete networks (C program) may fail, thus they recommend a more aggressive approach by having a fully independent alternative.