Sources of Failure in the Public Switched Telephone Network

What makes a distributed system reliable? A study of failures in the US Public Switched Telephone Network shows that human intervention is one key to this large system's reliability.

D. Richard Kuhn
National Institute of Standards and Technology

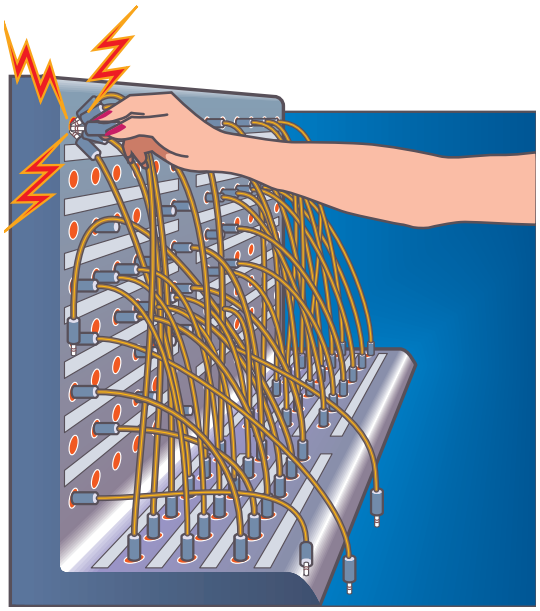
To operate successfully, most large distributed systems depend on software, hardware, and human operators and maintainers to function correctly. Failure of any one of these elements can disrupt or bring down an entire system.

One such distributed system, the US Public Switched Telephone Network (PSTN), is the US portion of possibly the largest distributed system in existence.[1] Like all telephone switching networks, the PSTN performs a fairly simple task: It connects point A with point B. Paradoxically, this seemingly trivial task requires some of the most complex and sophisticated computing systems in existence. Software for a switch with even a relatively small set of features may comprise several million lines of code.

The PSTN contains thousands of switches. Switches include redundant hardware and extensive self-checking and recovery software. For several decades, AT&T has expected its switches to experience not more than two hours of failure in 40 years,[2] a failure rate of 5.7×10^{-6}.

Since 1992, telephone companies have been required to notify the US Federal Communications Commission (FCC) of outages affecting more than 30,000 customers. I used these outage records to determine the principal causes of PSTN failures. To account for the possible effects of seasonal fluctuations in call-processing volume, I analyzed failures over two years, from April 1992 to March 1994, beginning with the earliest FCC reports. I made quantitative measures of how each failure source affects system dependability, in an effort to shed some light on the dependability of different components (including software).

The PSTN's dependability stems from a design that successfully exploits the loose coupling of system components. Because the PSTN has many similarities with other types of distributed systems, the analysis may suggest factors to consider in the design of distributed systems in general.



Major sources of failure were human error (on the part of both telephone company personnel and others), acts of nature, and overloads. Overloads caused nearly half of all downtime (44 percent) in terms of outage minutes.

An unexpected finding, given the complexity of the PSTN and its heavy reliance on software, was that software errors caused less system downtime (2 percent) than any other source of failure except vandalism. Hardware and software failures were similar in terms of average number of customers affected (96,000 and 118,000) and duration of outage (160 and 119 minutes).

Errors on the part of telephone company personnel and acts of nature caused similar amounts of downtime (14 and 18 percent).

0018-9162/97/$10.00 © 1997 IEEE

Table 1. Failure categories.

Category	Source	Examples
Human error—company	Errors made by telephone company personnel	Errors in • cable maintenance • power supply maintenance • power monitoring • facility or hardware board maintenance • software versions (mismatches) • following software maintenance procedures (such as errors in patch installations and configuration changes; does not include source code changes) • data entry
Human error—others	Errors made by people other than telephone company personnel	Cable cuttings Accidents (for example, cars striking telephone poles or equipment)
Acts of nature	Major and minor natural events	Cable, power supply, or facility damage from burrowing animals or lightning
	Natural disasters	Earthquakes, hurricanes, or floods
Hardware failures	Hardware component failures	Failures of cable components, power supplies, or facility components, clock or clock synchronization failures
Software failures	Internal errors in the software	Software errors under normal operation or in recovery mode
Overloads	Service demand exceeds the designed system capacity	
Vandalism	Sabotage or other intentional damage	

Table 2. Failure effects by categories and sources, for outages from April 1992 to March 1994.

Categories and sources	No. of outages	Average no. of customers affected	Average outage duration (minutes)	Customer minutes (in millions)
Human error—company	77	182,060	149.4	2,349.3
Cable maintenance	8	66,900	168.9	61.3
Power supply maintenance	19	292,980	150.4	879.1
Power monitoring	4	71,000	185.2	36.5
Facility or hardware board maintenance	15	169,370	134.7	242.7
Software versions (mismatches)	13	127,020	176.5	189.2
Following software maintenance or upgrade	8	225,960	204.2	871.2
Data entry	10	163,300	60.6	69.3
Human error—others	73	83,936	360.1	2,415.8
Cable cuttings	64	78,690	355.6	1,852.5
Accident	9	121,240	392.0	563.3
Acts of nature	32	159,000	828.2	3,124.0
Cable	13	13,000	717.6	784.8
Power supply	7	201,000	236.0	532.5
Facility	10	111,820	1,064.7	312.9
Natural disaster	2	1,200,000	2,437.0	1,493.8
Hardware failures	56	95,690	159.8	1,210.8
Cable component	2	125,000	46.0	5.7
Power supplies	14	112,580	103.9	369.9
Facility component	34	80,840	201.6	748.1
Clock or clock synchronization	6	130,670	91.0	87.1
Software failures	44	118,200	119.3	355.5
Normal operation	13	93,020	187.5	102.6
Recovery mode	31	124,940	86.8	252.9
Overloads	18	276,760	1,123.7	7,527.2
Vandalism	3	85,930	456.0	110.5

FAILURE CLASSIFICATION

Table 1 lists the failure classification scheme I used, a scheme that is general enough for comparisons with failures in other large distributed systems. In the case of the human error category, I separated errors made by telephone company personnel from those made by nonemployees because the companies have direct control over employees only. Overload conditions are accounted for separately because they represent failures accepted as an engineering trade-off between dependability and cost.

ANALYSIS PROCEDURE

FCC outage reports include the company name and the location of the facility where the outage occurred. They also include the date, time, and duration of each outage as well as the number of affected customers. They conclude with a descriptive summary of the outage.

Three of these parameters directly measure a failure's effects: the number of outages, their duration, and the number of customers affected. Using these parameters, I calculated a *customer minutes value*—the number of customers affected multiplied by the outage duration in minutes. Customer minutes are a more realistic measure of a disruption's magnitude as a basis for comparing failure data than outage duration alone. For example, a 20-minute outage affecting 10,000 customers (200,000 customer minutes) is considered more severe than a 30-minute outage affecting 1,000 customers (30,000 customer minutes). I did not use an industry measure of outages, *user lost erlangs* (ULE),[3] because I did not have access to some of the data necessary for computing ULEs. In addition, ULEs are more useful for statistically predicting the duration of future failures, and I wanted to identify and compare the underlying causes of past failures.

I assumed that the FCC reports recorded date and time values at the location where the outage occurred. There is some ambiguity in the times reported: Companies sometimes omitted the time zone, whether it was daylight savings or standard time, and whether the time recorded was an a.m. or p.m. time. The values for customers affected refer to the number of customers served by the failed facility, rather than the customers who were actively using the telephone system at the time of the failure. There may be some variations in the way companies report this value.

I encountered one significant interpretation problem. On April 21, 1992, the Alaskan ocean fiber-cable repeater failed. According to the report, service was restored when the company switched to satellite communications. The company reported the outage duration as two weeks, but it is not clear from the report if it took two weeks to repair the repeater or to switch to satellite communication, although the former appears more probable. Because this report was unclear, I did not use it in calculating total values or averages.

Including overloads as a failure category is somewhat problematic. When an overload occurs, the calls in progress do not fail, but it does prevent the system from accepting additional calls. Since the FCC reports list the number of customers served by an overloaded facility, rather than only the affected customers, these numbers are somewhat misleading. Other types of failure (such as cable cuttings) do prevent service to all customers of an affected facility. Thus, the numbers of customers affected are, in this sense, not directly comparable. The FCC reports do not include the number of customers normally using the system at the time of the outage.

This study excludes overloads when computing failures under the control of the phone companies, because overloads are expected failures.

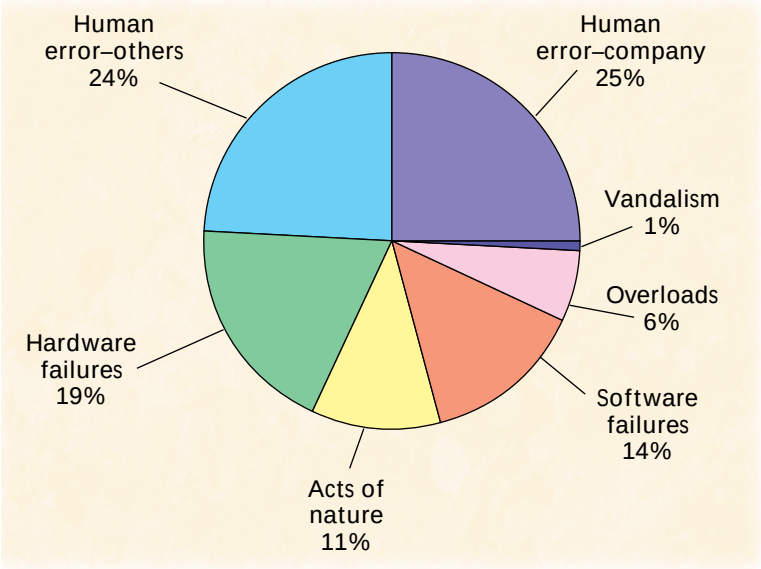


Figure 1. Number of telephone outages by category.

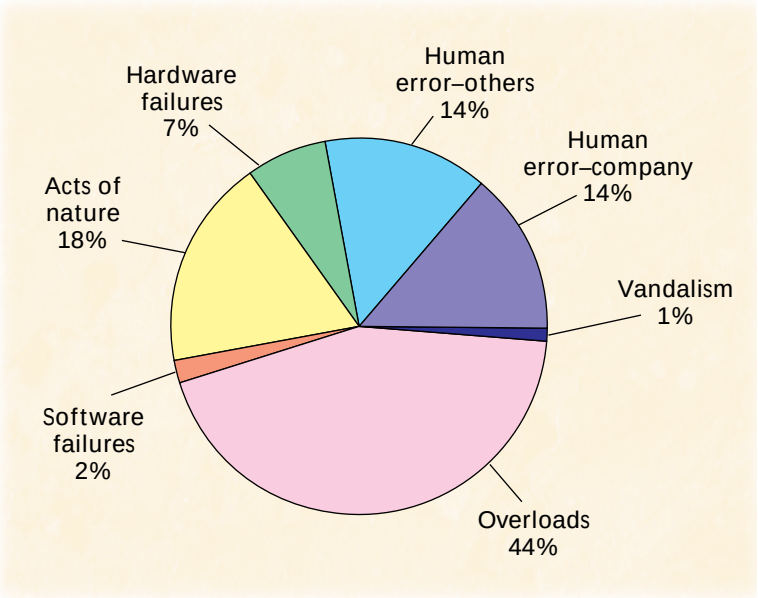


Figure 2. Downtime as measured in customer minutes, by category.

FINDINGS

Table 2 summarizes the number and duration of outages, customers affected, and customer minutes by cause. Figure 1 shows the percentage of outages attributed to each major category; Figure 2, the percentage of customer minutes. The data show that the number and magnitude of outages differs significantly for most failure categories. For example, although overloads caused only 6 percent of the total outages, they accounted for nearly half the total customer minutes.

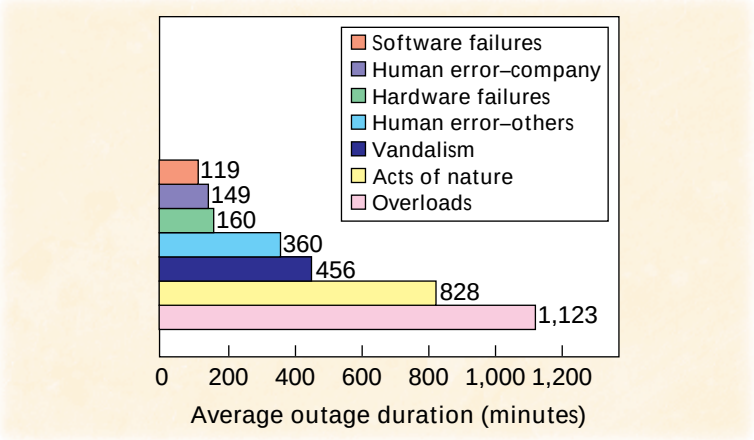


Figure 3. Average duration of outages.

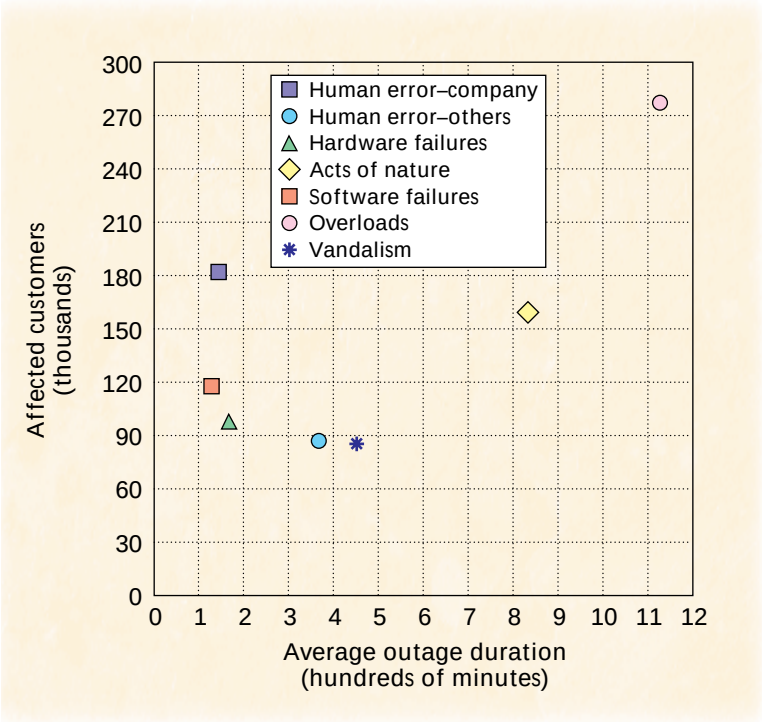


Figure 4. Plot of outage duration against customers affected.

Human error caused nearly half of the outages, but only about a quarter of the downtime.

Figure 3 illustrates the outage durations for the different failure categories and reveals part of the reason number and magnitude measures differ. Software, hardware, and human error by company personnel caused the shortest duration outages. Figure 4 compares the duration and customers affected for the major failure categories. The *x* axis displays outage duration, while the *y* axis displays the number of customers affected. Only overloads and acts of nature (in the upper right corner) are extended and widespread. Failures due to the errors of telephone company personnel (upper left) are brief but have widespread effects. Hardware and software failures were similar in terms of outage duration and customers affected. Vandalism and human errors caused by others were also similar in their effects.

OBSERVATIONS

Figure 2 shows that nearly half of the downtime is caused by overloads, which are expected outages. Because of economic and technical constraints, telephone companies do not expect service to be available all the time. For example, Bellcore's availability objective for local exchange networks in its client companies is 99.93 percent.[4] Larger capacity networks could probably eliminate most of this downtime but increase cost. Through decades of experience, the telephone industry has established a balance between benefits and the cost their consumers find acceptable.

Although the errors attributed to telephone employees are not the major source of outages, they are the major source of failure among those operational aspects under the companies' control. Human error by company personnel accounted for only 25 percent of outages and 14 percent of downtime. But failure sources controllable by the telephone companies (human error plus hardware and software failures) accounted for 58 percent of outages and 23 percent of downtime. So human errors by company personnel contributed nearly half of these outages (25 divided by 58) and nearly two-thirds of customer minutes of downtime (14 divided by 23).

Effects of human error were about the same for hardware and software maintenance. Human error for maintenance of cable and hardware components and for power monitoring accounted for about 15 percent of outages and 7 percent of downtime. Software-related human errors included mismatched versions, incorrect data entry, and procedural errors during upgrades. These errors accounted for 10 percent of outages and 7 percent of downtime.

Software errors caused a significant number of moderate outages. Although software errors caused approximately 14 percent of the outages, they accounted for only 2 percent of the customer minutes. Excluding human error by others, acts of nature, and overloads, however, software accounted for 24 percent of outages and 9 percent of customer minutes (downtime). Two factors probably cause software outages to be short: the incorporation of human intervention capabilities in the PSTN and the use of extensive error detection and recovery software.

WHY SO RELIABLE?

Despite its enormous size and complexity, the PSTN averaged an availability rate better than 99.999 percent in the time period studied. Why should perhaps the world's largest and most complex computerized distributed system also be among the most reliable?

Reliable software

To begin with, telephone switch manufacturers are among the world's leaders in computing technology.[5] They focus much of their research on developing highly reliable systems. Their software development processes typically incorporate the most sophisticated practices, supplemented by elaborate quality assurance functions. The PSTN software's low failure rate demonstrates that we can develop highly reliable software using the best practices.

Dynamic rerouting

But other factors add to the PSTN's dependability. In particular, telephone network designers appear to have exploited some aspects of the network's nature to compensate for complexities introduced by the dependability requirements.

By its very nature, the telephone network is highly distributed, so localized failures are more likely, and switches can reroute traffic dynamically to avoid a failed network node. More important, intermittent failures are usually not catastrophic. Other systems face much greater risks from a failure, no matter how brief. For example, failures of a few seconds in some fly-by-wire avionics software may result in the aircraft's destruction. A brief failure in one network component has relatively little impact on the availability figures for the entire PSTN across the US. However, for the PSTN to reroute calls, it must keep a good deal of information globally. Maintaining consistent distributed databases can require complex interactions among system components.

In his book, *Normal Accidents*, Charles Perrow identified two factors—interactions and coupling—that are significant in determining a system's safety properties.[6] Interactions refer to the dependencies between components, while coupling refers to the flexibility in a system. He characterized interactions as linear or complex, while coupling is loose or tight. Systems with simple, linear interactions have components that affect only other components that are functionally downstream. Complex system components interact with many other components in different parts of the system. Loosely coupled systems have more flexibility in time constraints, operation sequencing, and assumptions about the environment than do tightly coupled systems. Systems with complex interactions and tight coupling are likely to promote accidents. Complex interactions allow for more complications to develop and make the system hard to understand and predict. Tight coupling also means that the system has less flexibility in recovering when things go wrong.

John Rushby applied Perrow's analysis of failures in large physical systems to computer systems.[7] In such systems, interactions can, for example, take the form of signaling that coordinates processes or keeps distributed databases consistent. Coupling refers to constraints on timing, operation sequencing, acceptable input data ranges, and other aspects of system flexibility. Control systems with non-negotiable, real-time deadlines are tightly coupled, while the Internet, with multiple paths to route packets, is a loose-coupling example. Systems that require frequent updating of a distributed database are likely to have complex interactions to exchange messages among components and maintain the database's global consistency. A simple update and reporting system, which updates a database and writes files for input to report programs, is an example of linear interaction.

Loose coupling

In most systems, a trade-off can be made between simplicity of interactions and looseness of coupling.[7] We can consider the PSTN a loosely coupled system because it can dynamically reroute calls along many paths. However, it achieves this loose coupling at the cost of some complex interactions between components. These include the need for end-to-end acknowledgments, interactions among many systems, and the maintenance of some globally consistent databases. Major switching centers store information on alternative paths and exchange data on traffic patterns and switch status throughout the day. Such complex interactions can contribute to failures by making system behavior difficult to analyze.

The most spectacular example of a failure due to complex interactions in the PSTN is the 1990 nationwide AT&T network failure. This failure resulted from interactions between systems attempting to maintain consistent information about a failed switch. On the other hand, the PSTN's distributed database of routing information promotes loose coupling, which contributes to system dependability.

For a communications system, coupling is probably the more important of the two properties in determining its capacity to tolerate failures. It is directly related to the system's primary function: maintaining connections between points. The PSTN is loosely coupled, allowing for flexibility in recovering from failures. For the PSTN, loose coupling probably more than makes up for the interaction complexity. Designers should consider the trade-off between these factors—linear interactions or loose coupling—to add dependability to any high-integrity system. Two levels

Designers devote about half of the software in telephone switches to error detection and correction.

of recovery mechanisms—automated and manual—exploit the PSTN's loose coupling.

Designers devote about half of the software in telephone switches to error detection and correction. Such a high percentage of self-checking is probably atypical for software systems. Although some researchers note that adding fault-tolerance and fault-avoidance mechanisms to software sometimes decreases dependability because of the recovery mechanisms' added complexity,[8] these mechanisms work with great success in switching systems. Other computer-driven systems might benefit from more extensive use of built-in diagnostic and recovery software.

Human intervention

In addition to built-in self-test and recovery mechanisms, operators monitor telephone switches 24 hours a day and usually have the ability to modify switch software on the fly. Switch manufacturers provide 24-hour support services, usually with a remote maintenance capability that allows them to correct software in a switch thousands of miles away. Human intervention corrected many failures in under one hour. Simply restarting a switch temporarily fixed a significant number of software-caused outages.

Traffic routing also benefits from automated and human operations. Using information on switch status and traffic patterns exchanged by switches, software within a switch will automatically select an alternative route if the preferred route becomes overloaded or unavailable. If the switch exhausts all alternative routes, human intervention can reconfigure the network, sometimes solving the problem in a few minutes.[2] Status data exchanged regularly between switches makes automated and human operations to reconfigure routing possible. PSTN designers made the coupling-interactions trade-off in favor of loose coupling. Loose coupling allows human operators to intervene in the event of failure, rather than relying entirely on computer control.

Software is not the weak link in the PSTN system's dependability. Extensive use of built-in self-test and recovery mechanisms in major system components (switches) contributed to software dependability and are significant design features in the PSTN. The network's high dependability indicates that the trade-off between dependability gains and complexity introduced by built-in self-test and recovery mechanisms can be positive. Likewise, the trade-off between complex interactions and loose coupling of system components has been positive, permitting quick human intervention in most system failures and resulting in an extremely reliable system. ❖

Acknowledgments

I thank Dolores Wallace for discussions that helped greatly in focusing this article. Lisa Carnahan and Roger Martin gave the article thorough reviews, and Jim Lake compiled most of the data. The National Communications System partially supported this work.

References

1. A.V. Aho and N.D. Griffeth, "Feature Interactions in the Global Information Infrastructure," *Proc. 3rd ACM SigSoft Symp. Foundations Software Eng.*, ACM Press, New York, 1995.
2. G. Zorpette, "Keeping the Phone Lines Open," *IEEE Spectrum*, June 1989, pp. 32-36.
3. *A Technical Report on Network Survivability Performance*, T1A1.2/93-001R3, T1A1.2 Working Group on Network Survivability Performance, 1993.
4. C.M. Hamilton, "Telecommunication Network Dependability: A Baseline on Local Exchange Network Availability," *Proc. IEEE Annual Reliability and Maintainability Symp.* IEEE Press, Piscataway, N.J., 1991.
5. C. Jones, *Applied Software Measurement*, McGraw-Hill, New York, 1991.
6. C. Perrow, *Normal Accidents: Living with High Risk Technologies*, Basic Books, New York, 1984.
7. J. Rushby, "Critical System Properties: Survey and Taxonomy," *Reliability Eng. and System Safety*, Vol. 43, No. 2, pp. 189-219.
8. P.G. Neumann, *Computer Related Risks*, Addison-Wesley, Reading, Mass., 1995.

D. Richard Kuhn is a computer scientist and manager of the Software Quality Group at the National Institute of Standards and Technology. His research interests include software analysis, formal methods, and open systems. He is the author of more than 25 papers in these areas. Kuhn received an MS in computer science from the University of Maryland, College Park, and an MBA from the College of William and Mary. He is a member of the IEEE, the IEEE Computer Society, the ACM, and the Beta Gamma Sigma honor society.

Contact Kuhn at the National Institute of Standards and Technology, Gaithersburg, MD 20899; kuhn@nist.gov.