the Availability Digest

The Great 2003 Northeast Blackout and the $6 Billion Software Bug

March 2007

August 11, 2003. The Blaster worm wreaked havoc among PCs around the world. Terrorism?

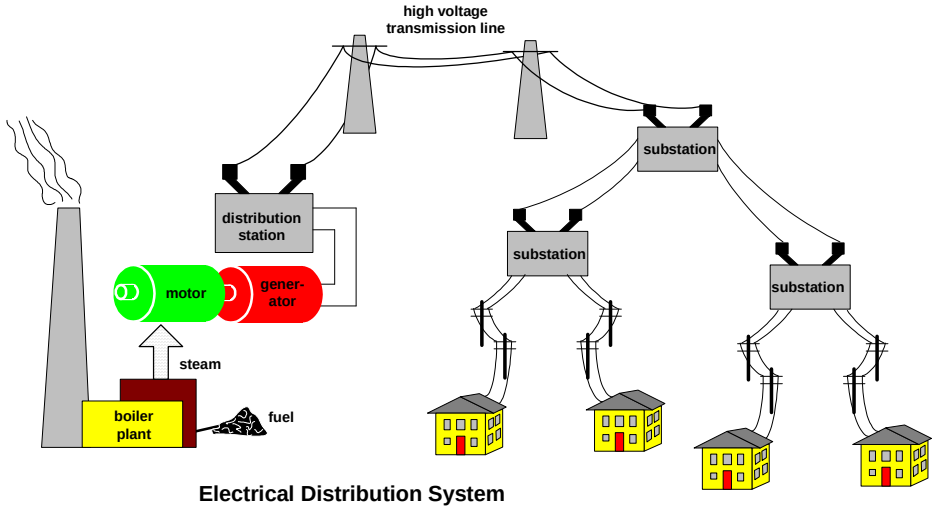
Three days later, on August 14, Northeast North America went dark. Was this a continuation of the terrorist cyber attack? It certainly looked that way.

However, as it turned out, the cause of the great 2003 Northeast Blackout was anything but sinister. The Blackout was, in fact, triggered on a hot day by an untrimmed tree in Ohio and was aided by a hung alarm system.

Power Grids

Most of the world's electric generation systems belong to some sort of power grid from which they can draw power if their demand exceeds their capacity. In addition, they can supply power to the grid if they are generating an excess amount of power.

A typical power plant generates electricity by coal-fired, oil-fired, or nuclear steam-generating plants. The steam produced is used to drive motors, which in turn drive the generators that produce electricity. (Other techniques involve driving motors with water power, wind power, or geological thermal power. Large solar arrays generate power without moving parts.)



Electrical Distribution System

The power generated by the generating station is distributed via high-voltage transmission lines to substations around the utility's service area. The substations step down the high voltage to that required by the utility's consumers and distribute this power to homes, businesses, and industries.

A generation and distribution facility is connected to its neighbors to share in power demands from its customers. This interconnection forms the electrical power distribution grid.[1]

Each power generating system is heavily protected with devices such as circuit breakers, which can trip to isolate failed or overloaded components such as generators, transformers, or transmission lines. The status of all components, including the state of the circuit breakers, is monitored by sophisticated SCADA (Supervisory Control and Data Acquisition) computer systems. These systems are typically redundant systems. In addition to status monitoring, they also incorporate sophisticated models to predict the proper response to fault conditions.

If all else fails, a power generating plant will go into "safe mode" to prevent overload damage. In this mode, it can be shut down in an orderly manner. It typically takes a day or more to return a generating plant to service.

There are strict protocols in place which govern the procedures to be followed should a power company encounter problems. Specifically, if they have to shut down a generating plant, whether for planned maintenance or due to an unplanned failure, they must coordinate this action with their neighboring grid members since this will impose additional load on their neighbors' generating capabilities, which must be managed. If there is not good coordination, a failed generating plant can overload a neighboring plant and take it down, which will overload its neighbors and take them down, and on and on until there is a massive area electrical outage. That is what happened on August 14, 2003.

The All-Important Monitoring System

Our story centers on FirstEnergy, a major producer of electric power in Ohio. FirstEnergy uses a Unix-based redundant GE XA/21 SCADA transmission management system written in C and C++ to control power generation and distribution via its high-voltage transmission system.[2] The XA/21 provides reliable management of generated power in response to system disturbances. It monitors system conditions against operating limits and develops corrective and preventative strategies.

The XA/21 is a proven system with over 100 worldwide installations and millions of hours of operational experience. It is a system respected by the power generating utilities.

The Cascading Power Failure

Power lines sag in hot weather. They also sag due to the heat generated by the electrical current which they are carrying. The high-voltage transmission lines (up to 500,000 volts) can blast a tree to its roots. This takes a tremendous amount of power and instantly overloads the transmission

[1] Electrical System Overview, *GlobalSecurity.org*; April 27, 2005.

[2] www.gepower.com.

lines. Therefore, it is imperative that trees under these transmission lines be kept trimmed so that they will not come in contact with the transmission lines.

The policy of FirstEnergy was to trim trees every five years. However, they did not always stick to this schedule; and the result was that some trees under its transmission lines had grown too tall.

August 14 was a hot day, pushing 90 degrees Fahrenheit in the Ohio area. Air conditioners and fans were imposing heavy demand on generating capacity. Between the heat of the day and the heat generated by the electrical current, transmission lines were seriously sagging. System logs showed the following sequence of faults:[3]

At 1:31 p.m., the power generating plant at Eastlake, Ohio, shut down. This plant had a history of maintenance problems.

At 2:02 p.m., the first 345 kilovolt transmission line came into contact with a tree and failed.

At 2:14 p.m., the GE XA/21 SCADA alarm subsystem failed. However, the SCADA system was otherwise working; and the controllers were not aware of this failure. Therefore, it was not repaired.

At 2:27 p.*m.,* a second transmission line fell into the trees and was shut down. The controllers received no notification of this nor of any of the problems which were to follow due to the errant SCADA system.

At 3:32 p.m., power shifted by the first failure caused another transmission line to sag into the trees and fail.

The power controllers were trying to understand these failures and failed to inform system controllers in nearby states. Therefore, FirstEnergy's power neighbors were not given the opportunity to plan for increased power demands caused by the Ohio failures.

At 3:39 p.m., another transmission line failed.

At 3:41 p.m. and 3:46 p.m., two circuit breakers connecting FirstEnergy's grid with the outside world tripped while sixteen more transmission lines failed. This turned out to be the last chance that controllers had to save their grid if they had they cut power to Cleveland at that time.

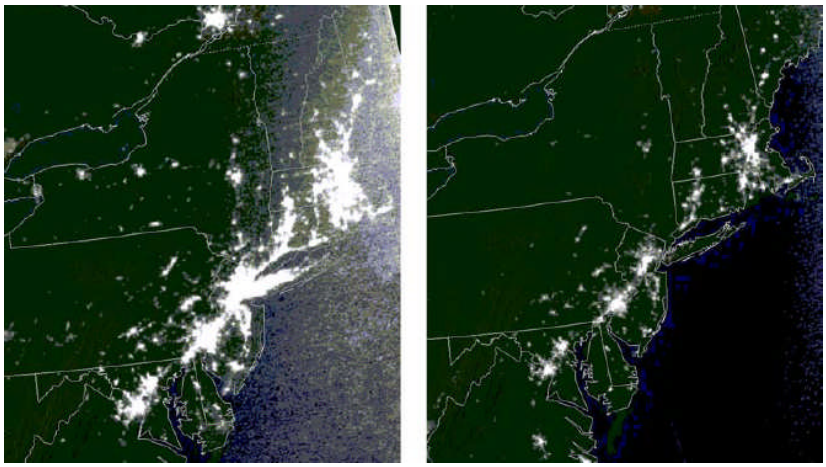
At 4:06 p.m., after another transmission line failed, an uncontrollable power surge was created; and chaos ensued. Ohio drew two gigawatts of power from Michigan. By 4:10 p.m., the eastern Michigan grid disconnected from the western part of the state; and Cleveland separated from the Pennsylvania grid. Many more transmission lines failed in Ohio and Michigan, blocking the eastward flow of power. Generators went down, creating huge power deficits. To compensate, 3.7 gigawatts of power surged from the East through Ontario to Michigan, a sudden tenfold increase in power transmission. This tripped the East Coast breakers, and the blackout was on.

Within a minute, the flow reversed to send two gigawatts eastward from Michigan through Ontario. It flipped again in just a half a second.

By 4:10 p.m., international connections began failing. Western Ontario separated from the East, and the first Ontario plants went offline in response to the unstable system. New York separated from New England, Ontario separated from the western New York grid, and the last lines between Michigan and Ohio failed. Windsor and Ontario dropped off the grid.

[3] Northeast Blackout of 2003, *Wikipedia*.

By 4:13 p.m., the cascading failures ended. 508 generating stations at 256 power plants, including 22 nuclear power plants in the U.S. and Canada, had gone offline. Over 40 million people in the U.S. and Canada were without power. The financial impact ran into billions. It wasn't until the next evening that partial power was restored.



The Northeast Before the Blackout | **The Northeast After the Blackout**

(NOAA satellite photographs. The remaining lights are communities, residences, and businesses with local power generation capabilities.)

This blackout was more than a lights-out inconvenience. Sewage spilled into waterways. Train service in the Northeast Corridor, including those provided by Amtrak, Long Island Railroad, and Metro-North, was shut down. Planes couldn't fly because passenger screening equipment was down, baggage couldn't be delivered, and electronic ticketing systems could not be accessed. Gas stations couldn't pump fuel. Oil refineries shut down. Cell phones and laptops quit working when their batteries ran out. Miners were marooned underground. The U.S./Canadian border shut down because of the lack of electronic border check systems. The backup diesel generator for the 1,900 room Marriott Hotel in New York wouldn't start even though it had been tested weekly. Guests had to walk down and sleep under the stars.[4] It is thought that the Ontario government fell in October elections because of the blackout. Financial losses were estimated to be six billion dollars.

The Monitoring System that Wasn't

All this disaster took was a hot day, a tall tree, and a balky monitoring system. If any one of these fault links had not occurred, the failure chain would have been broken; and the Northeast Blackout of 2003 would not have occurred (at least, not on August 14).

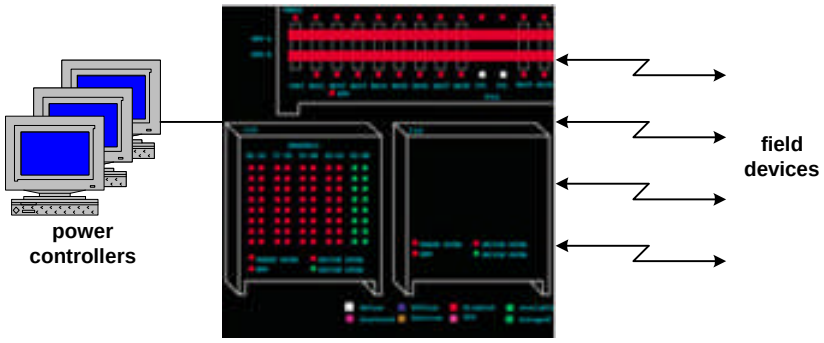
We can't control hot days. FirstEnergy could have trimmed their trees. But why did the redundant GE XA/21 system fail?

As logs were reviewed, it was found that the first thing that had happened was that alarm conditions were no longer being processed. This created an ever-growing queue of alarm conditions awaiting processing. Response times for operator requests grew from one second to one minute. Within thirty minutes, this queue became so long that it caused the primary system to

[4] R. LaPedis, Lessons learned from the 2003 northeastern blackout, *The Connection*; March/April 2004.

shut down and pass control to its backup system. Equally overwhelmed by the large queue of alarm events, the backup system itself eventually failed.

Where should one look to find the problem? There were four millions of lines of code in the system, and millions of hours of operation had never induced this problem. GE Energy made the tracking and solution of this problem its highest priority.



GE XA/21 SCADA System

The focus of the code search was the million lines of code comprising the alarm system.[5] It took a half-dozen experts eight weeks, but eventually they were able to recreate the problem by slowing the system way down. It turned out to be a race condition with an opportunity window measured in milliseconds or less. Two programs were able to get write access to the same data structure simultaneously. Their updates contaminated the structure. The result was that the alarm program was put into an infinite loop. It could neither process alarms nor report that it was in failure.

Because alarms could not be processed, the queue of events that had to be monitored for alarm conditions could not be processed. The queue grew without limit, and after about thirty minutes it had consumed all available memory. At this point, the primary server crashed; and the backup took over. It, too, was overwhelmed by the monster queue; and it crashed. At this point, the operators realized that there was a problem with their SCADA system; but it was far too late.

This software bug cost power customers an estimated six billion dollars.

GE Energy has since sent a patch correcting this bug to all of its customers around the world.

Lessons Learned

There are many lessons to be learned from this disaster. Some had to do with power generation, but many had to do with data processing, which is our focus.

Separate Your Backup System

Perhaps the greatest lesson that was learned by the 2003 Northeast Blackout is the same lesson that was learned after 9/11. Provide a great deal of space between your systems, whether you are running active/backup, active/hot standby, or active/active. If your active system was in New York City and your backup system was in Cleveland, you were out of luck. If your backup system was in Chicago or San Francisco, you were okay.

[5] Tracking the Blackout Bug, *Security Focus*; April 7, 2004.

How far is enough? One never knows. After 9/11, the regulatory bodies wanted to mandate a minimum separation of 300 miles for financial institutions. This recommendation hasn't been rigorously followed and may not have been enough anyway to avoid an outage with a blackout this size. The only thing that is certain is that the further your systems are separated, the safer you are.

Mind Your Backup Power

An alternative to system separation is to provide backup power at your sites. These UPS (uninterrupted power supply) systems are usually diesel motor generators. However, there are countless stories of UPS systems not starting. Though frequent testing of these systems is a must, the Marriott experience mentioned above shows that even that may not be enough. Others have found that their diesel fuel has congealed if it is not replaced periodically. Still others have found that they did not have enough fuel on hand to operate for the duration on the power outage. If the Northeast Blackout had lasted for days, could you get your generators refueled?

Make sure that you have enough battery backup to carry your system through the minutes that it may take to start a balky generator. Finally, with all of these problems, if your system is really mission-critical and depends on diesel generator backups, consider having two such systems in case one won't start.

Don't Trust Your Systems

Another key lesson is "Don't trust your systems." In this case, the power controllers trusted their systems to the extent that they ignored phone calls from the field and other centers warning about worsening conditions in the field. As a result, they did not know of the extensive tripping of electrical facilities in their grid; and they made no emergency alternative procedures to monitor their grid. Even worse, perhaps, was that they did not alert their neighboring grids of the problems they were facing.

Computers fail. Hardware fails. Software fails. Event notification fails. Failover fails. A computer monitioring system is a great aid, and in many cases it is required in order to be able to effectively run a facility such as a power utility. However, there must be procedures in place to alert operators to a failure of their system *independent* of the system itself. That is, one cannot depend upon a sick system to tell you that it is sick. Furthermore, there must be procedures in place to allow operations to continue in the event of a monitoring failure or to smoothly terminate operation if necessary.

Software is Never Bug-Free

Another lesson is that software is never bug-free. Here is a system with millions of hours of operational experience, and it still had a lurking bug in it. Is this the last bug? It reminds us of the software bug detector once promoted. It is a little box with a green light that plugs into your computer. When the last software bug has been fixed, the green light illuminates. Upon further inspection, opening up the box finds it empty – the green light is wired to nothing.

Accepting that software is never bug-free, it then becomes incumbent upon the programming team to write defensive code. Defensive code defends against situations that can cause damage, such as multiple write-access to the same data structure. This particular case is a good example of the benefits of object-oriented programming. If the system data structures had been encapsulated in an object which could protect its data, this kind of problem could have been effectively eliminated.

Defensive code also continually checks the health of the system components. If there had been built into the system a monitor that periodically queried the various software modules, such as the alarm module, a lack of response would have identified this problem almost immediately. Though the alarm system was looping, the Unix operating system provided occasional time slices to the rest of the system (thus, the increase in operator response time from one second to one minute). The system could have sounded an alarm.

In a companion article, we described microrebooting[6] as a technique for quickly repairing a software system by restarting only the failed submodule. This is an excellent example of a situation in which this technique would have been useful.

Postscript

As bad as this blackout was, it was not the worst that could happen. Just a month later, on September 28 at 3 a.m., a larger blackout in Italy affected 56 million people for nine hours. This blackout was unaided by computer error. It was caused when a storm took out two high-voltage transmission lines feeding power to Italy from Switzerland. The cascading effect of this failure caused the transmission lines to Italy from France to trip, plunging Italy into darkness.[7]

Within a year, other major blackouts occurred in London, Denmark and Sweden, Greece, Bahrain, Chile, Croatia, and Jordan.[8] These support the predominant lesson learned as described above – separate your systems for reliable backup.

[6] Microrebooting for Fast Recovery, *Availability Digest*; March, 2007.

[7] 2003 Italy Blackout, *Wikipedia*.

[8] Resources for Understanding Electric Power Reliability, *PSERC*; undated.