NETEQUALIZER

# The Story of the NetEqualizer

**By Art Reisman CTO**

www.netequalizer.com

## How I got my start...

In the spring of 2002, I was a systems engineer at Bell Labs in charge of architecting Conversant - an innovative speech-processing product. Revenue kept falling quarter by quarter, and meanwhile upper management seemed to only be capable of providing material for Dilbert cartoons, or perhaps helping to fine-tune the script for **The Office**. It was so depressing that I could not even read Dilbert anymore - those cartoons are not as amusing when you are living them every day.

Starting in the year 2000, and continuing every couple of months, there was a layoff somewhere in the company (which was Avaya at the time). Our specific business unit would get hit every six months or so. It was like living in a hospice facility. You did not want to get to know anybody too well because you would be tagged with the guilt of still having a job should they get canned next week. The product I worked on existed only as a cash cow to be milked for profit, while upper management looked to purchase a replacement. I can't say I blamed them; our engineering expertise was so eroded by then that it would have been a futile effort to try and continue to grow and develop the product.

*Cash Cows?*

*Gwynant Valley, Wales, ca. 1890-1900.*

Mercifully, I was laid off in June of 2003.

## How I got the idea...

Prior to my pink slip, I had been fiddling with an idea that a friend of mine, Paul Harris, had come up with. His idea was to run a local wireless ISP. This initially doomed idea spawned from an article in the local newspaper about a guy up in Aspen, CO that was beaming wireless Internet around town using a **Pringles can** - I am not making this up. Our validation consisted of Paul rigging up a Pringles can antenna, attaching it to his laptop's wireless card (we had external cards for wireless access at the time), and then driving a block from his house and logging in to his home Internet. Amazing!

The next day, while waiting around for the layoff notices, we hatched a plan to see if we could set up a tiny ISP from my neighborhood in northern Lafayette, CO. I lived in a fairly dense development of single-family homes, and despite many of my neighbors working in the tech industry, all we could get in our area was dial-up Internet. Demand was high for something faster.

So, I arranged to get a 1/2 T1 line to my house at the rate of about $1,500 per month, with the idea that I could resell the service to my neighbors. Our take rate for service appeared to be everybody I talked to. And so, Paul climbed onto the roof and set up some kind of pole attached to the top of the chimney, with a wire running down into the attic where we had a $30 Linksys AP. The top of my roof gave us a line-of-sight to 30 or 40 other rooftops in the area. We started selling service right away.

In the meantime, I started running some numbers in my head about how well this 1/2 T1 line would hold up. It seemed like every potential customer I talked to planned on downloading the Library of Congress, and I was afraid of potential gridlock. I had seen gridlock many times on the network at the office – usually when we were beating the crap out if it with all the geeky things we experimented on at Bell Labs.

*I*

*It seems like everyone wants to download the entire Library of Congress. Actual Library of Congress, circa 1900.*

We finally hooked up a couple of houses in late March, and by late April the trees in the area leafed out and blocked our signal. Subsequently, the neighbors got annoyed and stopped paying. Most 802.11 frequencies do not travel well through trees.

I was also having real doubts about our ability to make back the cost of the T1 service, especially with the threat of gridlock looming once more people came online - not to mention the line-of-sight being blocked by the trees.

Being laid off was a blessing in disguise. Leaving Bell Labs was not a step I would have taken on my own. Not only did I have three kids, a mortgage, and the net worth of a lawnmower, my marketable technical skills had lapsed significantly over the past four years. Our company had done almost zero cutting-edge R&D in that time. How was I going to explain that void of meaningful, progressive work on my resume? It was a scary realization.

*And then the trees leafed out and blocked our signal...*

*Fig Tree Avenue, circa 1900.*

## Making it a project...

Rather than complain about it, I decided to learn some new skills, and the best way to do that is to give yourself a project. I decided to spend some time trying to figure out a way to handle the potential saturation on our T1 line. I conjured up my initial solution from my computer science background. In any traditional operating systems' course, there is always a lesson discussing how a computer divvies up its resources. Back in the old days, when computers were very expensive, companies with computer work would lease time on a shared computer to run a "job". Computing centers at the time were either separate companies, or charge-back centers in larger companies that could afford a mainframe. A job was the term used for your computer program. The actual computer code was punched out on cards. The computer operator would take your stack of cards from behind a cage in a special room and run them through the machine. Many operators were arrogant jerks that belittled you when your job kicked out with an error, or if it ran too long and other jobs were waiting. Eventually computer jobs evolved so they could be submitted remotely from a terminal, and the position of the operator faded away. Even without the operator, computers were still very expensive, and there were always more jobs to run than the amount of leased time on the computer. This sounds a lot like a congested Internet pipe, right?

The solution for computers with limited resources was a specialized program called an operating system. Operating systems decided what jobs could run, and how much time they would get, before getting furloughed. During busy times, the operating system would temporarily kick larger jobs out and make them wait before letting them back in. The more time they used before completion, the lower their priority, and the longer they would wait for their turn.

## My big idea...

My idea - and the key to controlling congestion on an Internet pipe - was based on adapting the proven **OS scheduling** methodology used to prevent gridlock on a computer and apply it to another limited resource - bandwidth on an Internet link. But, I wasn’t quite sure how to accomplish this yet.

*I had a Big Idea, applying OS scheduling theory to manage Internet bandwidth.*

*Photographed at the GE exhibit at the State Fair, 1947. The lamp is 50,000 watts.*

**Kevin Kennedy** was a very respected technical manager during my early days at Bell Labs in Columbus, Ohio. Kevin left shortly after I came on board, and eventually rose up to be John Chambers’ number two at Cisco. Kevin helped start a division at Cisco which allowed a group of engineers to migrate over and work with him - many of whom were friends of mine from Bell Labs. I got on the phone and consulted a few of them on how Cisco dealt with congestion on their network. I wondered if they had anything smart and automated, and the answer I got was "yes, sort of." There was some newfangled way to program their IOS operating system, but nothing was fully automated. That was all I needed to hear. It seemed I had found a new niche, and I set out to make a little box that you plugged into a WAN or Internet port that would automatically relieve congestion and not require any internal knowledge of routers and complex customizations.

In order to make an automated fairness engine, I would need to be able to tap into the traffic on an Internet link. So I started looking at the Linux kernel source code and spent several weeks reading about what was out there. Reading source code is like building a roadmap in your head. Slowly over time neurons start to figure it out - much the same way a London Taxi driver learns their way around thousands of little streets with some of them being dead ends. I eventually stumbled into the Linux bridge code. The Linux bridge code allows anybody with a simple laptop and two Ethernet cards to build an Ethernet bridge. Although an Ethernet bridge was not really related in function to my product idea, it solved all of the upfront work I would need to do to break into an Internet connection to examine data streams and then reset their priorities on the fly as necessary - all this with complete transparency to the network.

## Making it a reality...

As usual, the mechanics of putting the concept in my head into working code was a bit painful and arduous. I am not the most adept when it comes to using code syntax and wandering my way around kernel code. A good working knowledge of building tools, compiling tools, and legacy Linux source code is required to make anything work in the Linux kernel. The problem was that I couldn't stand those details. I hated them and would have gladly paid somebody else to implement my idea, but I had absolutely no money. Building and coding in the Linux kernel is like reading a book you hate where the chapters and plot are totally scrambled. But, having done it many times, I slogged through, and out the other side appeared the Linux Bandwidth Arbitrator (LBA) - a set of utilities and computer programs made for Linux open source that would automatically take a Linux bridge and start applying fairness rules.

Once I had the tool working in my small home test lab, I started talking about it on a couple of Linux forums. I needed a real network to test it on because I had no experience running a network. My engineering background up until now had been working with firmware on proprietary telecommunication products. I had no idea how my idea would perform in the wild.

## Going Live!

Eventually, as a result of one of my Linux forum posts, a call came in from a network administrator and Linux enthusiast named Eric who ran a network for a school district in the Pacific Northwest. I thought I had hit the big time. He was a real person with a real network with a real problem. I helped him load up a box with our tool set in his home office for testing. Eventually, we got it up and running on his district network with mixed results. This experiment, although inconclusive, got some serious kinks worked out with my assumptions.

I went back to the Linux forums with my newfound knowledge. I learned of a site called "**freshmeat.net**" where one could post free software for commercial use. The response was way more than I expected, perhaps a thousand hits or so in the first week. However, the product was not easy to build from scratch and most hits were just curious seekers of free tools. Very few users had built a Linux kernel, let alone had the skill set to build a Linux Bandwidth Arbitrator from my instructions. But, it only took one qualified candidate to further validate the concept.

This person turned out to be an IT administrator from a state college in Georgia. He loaded our system up after a few questions, and the next thing I knew I got an e-mail that went something like this:

*An IT Administrator at a State College gave me my 1st feedback:*

*"All of our congestion has ceased, and the utilization on our main Internet trunk is 20% less.  The students are very happy!"*

*College ca. 1900.*

“Since we installed the LBA, all of our congestion has ceased, and the utilization on our main Internet trunk is 20% less. The students are very happy!”

I have heard this type of testimonial many times since, but I was in total disbelief with this first one. It was on a significant network with significant results! Did it really work, or was this guy just yanking my chain? No. It was real, and it really did work!

I was broke and ecstatic at the same time. The Universe sends you these little messages that you are on the right track just when you need them. To me, this e-mail was akin to 50,000 people in a stadium cheering for you. Queue

## And the rest is history...

Our following on freshmeat.net grew and grew. We broke into the Top 100 projects, which is like making it to Hollywood Week on American Idol to tech geeks, and then broke the Top 50 or so in their rankings. This was really quite amazing because most of the software utilities on freshmeat.net were consumer-based utilities, which have a much broader audience. The only projects with higher rankings in a business-to-business utility product (like the LBA) were utilities like SQL Dansguard, and other very well-known projects.

Shortly after going live on freshmeat.net, I started collaborating add-ons to the LBA utility with Steve Wagor (now my partner at APconnections). He was previously working as a DBA, webmaster, and jack-of-all-trades for a company that built websites for realtors in the southwestern United States. We were getting about one request a week to help install the LBA in a customer network. Steve got the idea to make a self-booting CD that could run on any standard PC with a couple of LAN cards. In August of 2004, we started selling them. Our only current channel was freshmeat.net, which allowed us to offer a purchasable CD as long as we offered the freeware version too.* We sold fifteen CD's that first month. The only bad news was that we were working for about $3.00 per hour. There were too many variables on the customer-loaded systems to be as efficient as we needed to be. Also, many of the customers loading the free CD were as broke as we were and not able to pay for our expertise.

** As an interesting side note, we also had a free trial version that ran for about two hours that could be converted to the commercial version with a key. The idea was to let people try it, prove it worked, and then send them the permanent key when they paid. Genius, we thought. However, we soon realized there were thousands of small Internet cafes around the world that would run the thing for two hours and then reboot. They were getting congestion control and free consulting from us. So in countries where the power goes out once a day anyway, no one is bothered by a sixty-second Internet outage.*

As word got out that the NetEqualizer worked well, we were able to formalize the commercial version and started bundling everything into our own manufacturing and shipping package from the United States. This eliminated all the free consulting work on the demo systems, and also ensured a uniform configuration that we could support.

Today NetEqualizer has become an adjective brand name for bandwidth shaping in growing circles.

*Figure 6: The original LBA has been commercialized and evolved into the NetEqualizer, which we sell directly to thousands of satisfied customers across six continents. NetEqualizer NE2000   ca 2011.*

## Some facts that humble me...

NetEqualizer is a multi-million dollar company.

NetEqualizer's have over ten million users going through them on six continents.

We serve many unique locales in addition to the world's largest population centers. Some of the more interesting places are:

- Malta
- The Seychelles Islands
- The Northern Slopes of Alaska
- Iceland
- Barbados
- Guantanamo Bay
- The Yukon Territory
- The Afghan-American Embassy
- The United States Olympic Training Center
- Multiple NBA arenas
- Yellowstone National Park

Stay tuned for Part II, "From Startup to Multi-National, Multi-Million Dollar Enterprise."

Meanwhile, check out these related articles:

"**NetEqualizer Brand Becoming an Eponym for Fairness and Net-Neutrality Techniques**"

"**Building a Software Company from Scratch**" - Adapted from an entrepreneur.org article.

## The End (for now)....