

Ruminations of a Software Man

Essay on the Nature of Software Infrastructure—the Run to Ubiquity

Craig Burton-August, 2008

Look into the mirror of history to see the history of tomorrow.
— .D. LuCxeed

Introduction

This essay briefly covers how Novell went about creating a new software category—*software infrastructure*. Software infrastructure was different than hardware, software or firmware. It was priced differently and was viewed differently and behaved differently in the market than its other three siblings.

This historical view is meant to create a discussion on how we might treat the emerging software infrastructure.

One of the core theses of this essay is that Novell did something that no one had done before. Novell sold software infrastructure as a separate product without hardware. Up until 1983, software infrastructure was never sold separately, it came with hardware. It was almost considered hardware. As I will explain, shifting away from providing software infrastructure coupled with hardware to providing software infrastructure independently changed things forever and is about to change things again as we gear up like never before for the run to ubiquity.

The role of software and hardware

Circa 1984, software and hardware divisions were simple—especially when it came to PCs. The only two applications driving personal computer business use were Lotus 1-2-3 and WordStar. The biggest game software was Microsoft Flight Simulator. In these days, there was no abstraction layer between the application and the hardware. MS Flight Simulator and Lotus 1-2-3 did not use the operating system to display anything; these applications wrote directly to the graphics cards to gain maximum performance and minimum intrusion.

There were three classes of wares that made up personal computer systems; hardware, software, and firmware (I am not going to go into the mini-computer discussion here at all.). This of course was the impetus for Novell naming its product “NetWare.” Before NetWare, the software category was divided into two big categories—below the API and above the API. Below the API was operating system stuff. Above the API was application stuff. As I explained above, the applications of the day did not stick to working above the API all of the time, it was common to break the rules and go directly to the hardware.

As I write this, I imagine that most of my readers know all of this and have heard it so many times that they are wondering what screw has come loose that would cause me to hash through this all again. Just hang on a bit longer and it will become apparent.

A new “Ware” in the house

As most people know, before NetWare, Novell was a hardware company. It built a very expensive workstation and a centralized server that connected all of these workstations together. The advent of the PC and the incredible revolution of mass storage caused Novell to quickly abandon the hardware business.

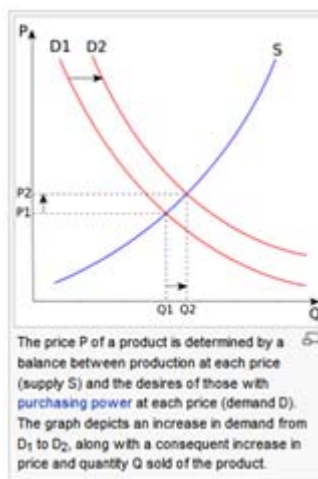
But I have to tell you, trying to figure out what NetWare really was, and how to price it and think about it created some significant dilemmas. I won't take you through the whole thing other than to say that NetWare was priced and sold as a hybrid between hardware and software. We have since come to call it “software infrastructure.”

Software infrastructure has the unique characteristic that sometimes it acts like hardware, and sometimes it acts like software. Software—both below the API and above the API—had then and has now some different characteristics than software infrastructure.

While there are other issues, I will focus on how software infrastructure relates to the laws of supply and demand.

Software Infrastructure and the Laws of Supply and Demand

When you look at software infrastructure, the traditional laws of supply and demand are reversed. Traditional economics says that if something is scarce, there is greater demand for the item and the price increases. As the item becomes ubiquitous, demand decreases as does the price.



With software infrastructure, ubiquity is what creates demand. Price and elasticity are not subject to scarcity. In other words, you want to have software infrastructure as ubiquitous as possible as fast as possible. Price is sustainable by delivering the next—never ending—component of infrastructure, not by traditional protectionism tactics.

A lot of people I discuss this with still argue with me about how important protectionism is. Again, the case of Novell comes up. From one point of view, Microsoft forced the price of NetWare down by “embracing and extending” NetWare with its own networking products. This is just not the case.

Novell, in a fit of protectionism decided that it was going to drag its feet in providing networking support for Windows 95—seriously. The

thinking was that Novell was so powerful that somehow it could slow down the adoption of Windows. This protectionism brain fart just gave Microsoft the opportunity to get the credibility it needed and to become the main supplier of NetWare-like connectivity.

The point of this essay isn't to give Novell grief; this inverted economic model on software infrastructure is not readily evident and is a hard concept to get.

So here is my point about the inverted supply and demand model; today's core software infrastructure is made up of a core set of services. Roughly, file, print, web, database, directory, security, and the Internet protocol suite. Anything that artificially restricts the growth of this infrastructure compounds growth limitation on almost all technology across the board.

Anything that increases ubiquity of these core services helps the market mature and get better.

This is why net neutralization is so dangerous. Not just because it limits freedom, but because it threatens to artificially impact the run to ubiquity. *The run to ubiquity*—I can't emphasize how important this is.

Let's put it other terms. Technology is on an evolutionary track similar to all of the other animal kingdoms. According to some, like [Kevin Kelly](#), it is exactly like an animal kingdom. In these terms, think of the Internet as a terra forming process—the creation of a technology planet, as it were. For this planet to really be a place that lets technology evolve and be used like some of us predict, there need to be ubiquitous core infrastructure components. Think of software infrastructure as air. Software infrastructure needs to be as ubiquitous as air. This will not diminish its value, but make its value elemental. Can you place a value on the presence of air? Probably not. It is so ubiquitous in fact, that you only consider its value when you can't get any. Or, when you see the air quality in Beijing, whichever comes first.

Summary

Software infrastructure is not like a dial tone. It is not a highway that data “travels” on and thus to be thought of as a goods delivery system that should be taxed and regulated. Software infrastructure is technology that gives life to the planet of discovery, invention, and progress. If you artificially limit infrastructure, you artificially limit discovery, invention, and progress. Can you imagine if the government tried to tax and regulate air?