**<http://futureoftheinternet.org/files/2013/06/ZittrainTheFutureoftheInternet.pdf>**

**BUILDING NETWORKS ON A NETWORK**

Returning to a threshold question: if we wanted to allow people to use information technology at home and to be able to network in ways beyond sending floppy diskettes through the mail, how can we connect homes to the wider world? A natural answer would be to piggyback on the telephone network, which was already set up to convey people’s voices from one house to another, or between houses and institutions. Cyber law scholar Tim Wu and others have pointed out how difficult it was at first to put the telephone network to any new purpose, not for technical reasons, but for ones of legal control—and thus how important early regulatory decisions forcing an opening of the network were to the success of digital networking.2

In early twentieth-century America, AT&T controlled not only the telephone network, but also the devices attached to it. People rented their phones from AT&T, and the company prohibited them from making any modifications to the phones. To be sure, there were no AT&T phone police to see what customers were doing, but AT&T could and did go after the sellers of accessories like the Hush-A-Phone, which was invented in 1921 as a way to have a conversation without others nearby overhearing it.3 It was a huge plastic funnel enveloping the user’s mouth on one end and strapped to the microphone of the handset on the other, muffling the conversation. Over 125,000 units were sold.

As the monopoly utility telephone provider, AT&T faced specialized regulation from the U.S. Federal Communications Commission (FCC). In 1955, the FCC held that AT&T could block the sale of the funnels as “unauthorized foreign attachments,” and terminate phone service to those who purchased them, but the agency’s decision was reversed by an appellate court. The court drolly noted, “[AT&T does] not challenge the subscriber’s right to seek privacy. They say only that he should achieve it by cupping his hand between the transmitter and his mouth and speaking in a low voice into this makeshift muffler.”

Cupping a hand and placing a plastic funnel on the phone seemed the same to the court. It found that at least in cases that were not “publicly detrimental”—in other words, where the phone system was not itself harmed—AT&T had to allow customers to make physical additions to their handsets, and manufacturers to produce and distribute those additions. AT&T could have invented the Hush-A-Phone funnel itself. It did not; it took outsiders to begin changing the system, even in small ways.

Hush-A-Phone was followed by more sweeping outside innovations. During the 1940s, inventor Tom Carter sold and installed two-way radios for companies with workers out in the field. As his business caught on, he realized how much more helpful it would be to be able to hook up a base station’s radio to a telephone so that faraway executives could be patched in to the front lines. He invented the Carterfone to do just that in 1959 and sold over 3,500 units. AT&T told its customers that they were not allowed to use Carterfones, because these devices hooked up to the network itself, unlike the Hush-APhone, which connected only to the telephone handset. Carter petitioned against the rule and won.5 Mindful of the ideals behind the Hush-A-Phone decision, the FCC agreed that so long as the network was not harmed, AT&T could not block new devices, even ones that directly hooked up to the phone network.

These decisions paved the way for advances invented and distributed by third parties, advances that were the exceptions to the comparative innovation desert of the telephone system. Outsiders introduced devices such as the answering machine, the fax machine, and the cordless phone that were rapidly adopted.6 The most important advance, however, was the dial-up modem, a crucial piece of hardware bridging consumer information processors and the world of computer networks, whether proprietary or the Internet.

With the advent of the modem, people could acquire plain terminals or PCs and connect them to central servers over a telephone line. Users could dial up whichever service they wanted: a call to the bank’s network for banking, followed by a call to a more generic “information service” for interactive weather and news.

The development of this capability illustrates the relationships among the standard layers that can be said to exist in a network: at the bottom are the physical wires, with services above, and then applications, and finally content and social interaction. If AT&T had prevailed in the Carterfone proceeding, it would have been able to insist that its customers use the phone network only for traditional point-to-point telephone calls. The phone network would have been repurposed for data solely at AT&T’s discretion and pace. Because AT&T lost, others’ experiments in data transmission could move forward. The physical layer had become generative, and this generativity meant that additional types of activity in higher layers were made possible. While AT&T continued collecting rents from the phone network’s use whether for voice or modem calls, both amateurs working for fun and entrepreneurs seeking new business opportunities got into the online services business.

**THE PROPRIETARY NETWORK MODEL**

The first online services built on top of AT&T’s phone network were natural extensions of the 1960s IBM-model minicomputer usage within businesses: one centrally managed machine to which employees’ dumb terminals connected. Networks like CompuServe, The Source, America Online, Prodigy, GEnie, and MCI Mail gave their subscribers access to content and services deployed solely by the network providers themselves.

In 1983, a home computer user with a telephone line and a CompuServe subscription could pursue a variety of pastimes8—reading an Associated Press news feed, chatting in typed sentences with other CompuServe subscribers through a “CB radio simulator,” sending private e-mail to fellow subscribers, messaging on bulletin boards, and playing rudimentary multiplayer games.9 But if a subscriber or an outside company wanted to develop a new service that might appeal to CompuServe subscribers, it could not automatically do so. Even if it knew how to program on CompuServe’s mainframes, an aspiring provider needed CompuServe’s approval. CompuServe entered into development agreements with outside content providers10 like the Associated Press and, in some cases, with outside programmers, but between 1984 and 1994, as the service grew from one hundred thousand subscribers to almost two million, its core functionalities remained largely unchanged.

Innovation within services like CompuServe took place at the center of the network rather than at its fringes. PCs were to be only the delivery vehicles for data sent to customers, and users were not themselves expected to program or to be able to receive services from anyone other than their central service provider. CompuServe depended on the phone network’s physical layer generativity to get the last mile to a subscriber’s house, but CompuServe as a service was not open to third-party tinkering.

Why would CompuServe hold to the same line that AT&T tried to draw? After all, the economic model for almost every service was the connect charge: a per-minute fee for access rather than advertising or transactional revenue.13 With mere connect time as the goal, one might think activity-garnering user-contributed software running on the service would be welcome, just as user-contributed content in the CB simulator or on a message board produced revenue if it drew other users in. Why would the proprietary services not harness the potential generativity of their offerings by making their own servers more open to third-party coding? Some networks’ mainframes permitted an area in which subscribers could write and execute their own software, but in each case restrictions were quickly put in place to prevent other users from running that software online. The “programming areas” became relics, and the Hollerith model prevailed.

Perhaps the companies surmised that little value could come to them from user and third-party tinkering if there were no formal relationship between those outside programmers and the information service’s in-house developers. Perhaps they thought it too risky: a single mainframe or set of mainframes running a variety of applications could not risk being compromised by poorly coded or downright rogue applications.

Perhaps they simply could not grasp the potential to produce new works that could be found among an important subset of their subscribers—all were instead thought of solely as consumers. Or they may have thought that all the important applications for online consumer services had already been invented—news, weather, bulletin boards, chat, e-mail, and the rudiments of shopping.

In the early 1990s the future seemed to be converging on a handful of corporate-run networks that did not interconnect. There was competition of a sort that recalls AT&T’s early competitors: firms with their own separate wires going to homes and businesses. Some people maintained an e-mail address on each major online service simply so that they could interact with friends and business contacts regardless of the service the others selected. Each information service put together a proprietary blend of offerings, mediated by software produced by the service. Each service had the power to decide who could subscribe, under what terms, and what content would be allowed or disallowed, either generally (should there be a forum about gay rights?) or specifically (should this particular message about gay rights be deleted?). For example, Prodigy sought a reputation as a family-friendly service and was more aggressive about deleting sensitive user-contributed content; CompuServe was more of a free-for-all.

But none seemed prepared to budge from the business models built around their mainframes, and, as explained in detail in Chapter Four, works by scholars such as Mary Benner and Michael Tushman shed some light on why. Mature firms can acquire “stabilizing organizational routines”: “internal biases for certainty and predictable results [which] favor exploitative innovation at the 24 The Rise and Stall of the Generative Net expense of exploratory innovation.” And so far as the proprietary services could tell, they had only one competitor other than each other: generative PCs that used their modems to call other PCs instead of the centralized services. Exactly how proprietary networks would have evolved if left only to that competition will never be known, for CompuServe and its proprietary counterparts were soon overwhelmed by the Internet and the powerful PC browsers used to access it. But it is useful to recall how those PC-to-PC networks worked, and who built them.

**AN UNTENABLE STATUS QUO**

The Internet and its generative machines have muddled along pretty well since 1988, despite the fact that today’s PCs are direct descendants of that era’s unsecured workstations. In fact, it is striking how few truly disruptive security incidents have happened since 1988. Rather, a network designed for communication among academic and government researchers appeared to scale beautifully as hundreds of millions of new users signed on during the 1990s, a feat all the more impressive when one considers how demographically different the new users were from the 1988 crowd. However heedless the network administrators of the late ’80s were to good security practice, the mainstream consumers of the ’90s were categorically worse. Few knew how to manage or code their generative PCs, much less how to rigorously apply patches or observe good password security.

The threat presented by bad code has slowly but steadily increased since 1988. The slow pace, which has let it remain a back-burner issue, is the result of several factors which are now rapidly attenuating. First, the computer scientists of 1988 were right that the hacker ethos frowns upon destructive hacking.40 Morris’s worm did more damage than he intended, and for all the damage it did do, the worm had no payload other than itself. Once a system was compromised by the worm it would have been trivial for Morris to have directed the worm to, for instance, delete as many files as possible.41 Morris did not do this, and the overwhelming majority of viruses that followed in the 1990s reflected similar authorial forbearance. In fact, the most well-known viruses of the ’90s had completely innocuous payloads. For example, 2004’s Mydoom spread like wildfire and affected connectivity for millions of computers around the world. Though it reputedly cost billions of dollars in lost productivity, the worm did not tamper with data, and it was programmed to stop spreading at a set time.

The bad code of the ’90s merely performed attacks for the circular purpose of spreading further, and its damage was measured by the effort required to eliminate it at each site of infection and by the burden placed upon network traffic as it spread, rather than by the number of files it destroyed or by the amount of sensitive information it compromised. There are only a few exceptions. The infamous Lovebug worm, released in May 2000, caused the largest outages and damage to Internet-connected PCs to date.43 It affected more than just connectivity: it overwrote documents, music, and multimedia files with copies of itself on users’ hard drives. In the panic that followed, software engineers and antivirus vendors mobilized to defeat the worm, and it was ultimately eradicated. Lovebug was an anomaly. The few highly malicious viruses of the time were otherwise so poorly coded that they failed to spread very far. The Michelangelo virus created sharp anxiety in 1992, when antivirus companies warned that millions of hard drives could be erased by the virus’s dangerous payload. It was designed to trigger itself on March 6, the artist’s birthday. The number of computers actually affected was only in the tens of thousands—it spread only through the pre-Internet exchange of infected floppy diskettes— and it was soon forgotten.45 Had Michelangelo’s birthday been a little later in the year—giving the virus more time to spread before springing—it could have had a much greater impact. More generally, malicious viruses can be coded to avoid the problems of real-world viruses whose virulence helps stop their spread. Some biological viruses that incapacitate people too quickly can burn themselves out, destroying their hosts before their hosts can help them spread further.46 Human-devised viruses can be intelligently designed—fine-tuned to spread before biting, or to destroy data within their hosts while still using the host to continue spreading.

Another reason for the delay of truly destructive malware is that network operations centers at universities and other institutions became more professionalized between the time of the Morris worm and the advent of the mainstream consumer Internet. For a while, most of the Internet’s computers were staffed by professional administrators who generally heeded admonitions to patch regularly and scout for security breaches. They carried beepers and were prepared to intervene quickly in the case of an intrusion.

Less adept mainstream consumers began connecting unsecured PCs to the Internet in earnest only in the mid-1990s. At first their machines were hooked up only through transient dialup connections. This greatly limited both the amount of time per day during which they were exposed to security threats, and the amount of time that, if compromised and hijacked, they would themselves contribute to the problem.

Finally, there was no business model backing bad code. Programs to trick users into installing them, or to bypass users entirely and just sneak onto the machine, were written only for fun or curiosity, just like the Morris worm. There was no reason for substantial financial resources to be invested in their creation, or in their virulence once created. Bad code was more like graffiti than illegal drugs. Graffiti is comparatively easier to combat because there are no economic incentives for its creation. The demand for illegal drugs creates markets that attract sophisticated criminal syndicates.

Today each of these factors has substantially diminished. The idea of a Netwide set of ethics has evaporated as the network has become so ubiquitous. Anyone is allowed online if he or she can find a way to a computer and a connection, and mainstream users are transitioning to always-on broadband. In July 2004 there were more U.S. consumers on broadband than on dial-up,49 and two years later, nearly twice as many U.S. adults had broadband connections in their homes than had dial-up.50 PC user awareness of security issues, however, has not kept pace with broadband growth. A December 2005 online safety study found 81 percent of home computers to be lacking first-order protection measures such as current antivirus software, spyware protection, and effective firewalls.51 The Internet’s users are no longer skilled computer scientists, yet the PCs they own are more powerful than the fastest machines of the 1980s. Because modern computers are so much more powerful, they can spread malware with greater efficiency than ever.

Perhaps most significantly, there is now a business model for bad code—one that gives many viruses and worms payloads for purposes other than simple reproduction.52 What seemed truly remarkable when it was first discovered is now commonplace: viruses that compromise PCs to create large “botnets” open to later instructions. Such instructions have included directing the PC to become its own e-mail server, sending spam by the thousands or millions to e-mail addresses harvested from the hard disk of the machine itself or gleaned from Internet searches, with the entire process typically unnoticeable to the PC’s owner. At one point, a single botnet occupied 15 percent of Yahoo’s entire search capacity, running random searches on Yahoo to find text that could be inserted into spam e-mails to throw off spam filters.53 One estimate pegs the number of PCs involved in such botnets at 100 to 150 million, or a quarter of all the computers on the Internet as of early 2007,54 and the field is expanding: a study monitoring botnet activity in 2006 detected, on average, the emergence of 1 million new bots per month.