Unplanned Obsolescence: Hardware and Software After Collapse

Esther Jang
University of Washington
infrared@cs.uw.edu

Matthew Johnson
University of Washington
matt9j@cs.uw.edu

Edward Burnell
M.I.T.
eburn@mit.edu

Kurtis Heimerl
University of Washington
kheimerl@cs.uw.edu

ABSTRACT

In a setting of economic and infrastructural collapse, the inability to manufacture and maintain computing resources will be an enormous limitation on the continued use of technology. The concept of "rot" exists for both hardware and software, referring to a slow loss of functionality over time. Given a desire to maintain technological capability, we raise a variety of questions about technology use in such a scenario. How long will current hardware last through repair, robust construction, and good maintenance practices? What would software development and maintenance entail without today's Internet infrastructure? What can be done to keep our software stable and usable for as long as possible in the face of viruses, storage degradation, and other threats? We present rough estimates of the expected longevity of desktop and laptop hardware for various levels of maintenance, and argue that software and hardware degradation together jointly limit how long devices will remain usable for computing tasks, especially those involving any exposure to external files or networks. We propose both physical and social strategies to guard against both modes of degradation.

CCS CONCEPTS

•**Security and privacy** → *Human and societal aspects of security and privacy;* •**Hardware** → *Aging of circuits and systems;* •**Software and its engineering** → *Software creation and management;*

KEYWORDS

longevity; hardware; software; security; malware;

ACM Reference format:
Esther Jang, Matthew Johnson, Edward Burnell, and Kurtis Heimerl. 2017. Unplanned Obsolescence: Hardware and Software After Collapse. In *Proceedings of LIMITS '17, June 22–24, 2017, Santa Barbara, CA, USA, ,* 9 pages.
DOI: http://dx.doi.org/10.1145/3080556.3080566

LIMITS '17, June 22–24, 2017, Santa Barbara, CA, USA

DOI: http://dx.doi.org/10.1145/3080556.3080566

1 INTRODUCTION

Computing resources are integral to the fabric of our modern society. Medical records are stored and accessed electronically, weather is predicted using computational models, and people have access to high-bandwidth long-distance communications infrastructure at their fingertips. In an infrastructural collapse, computing and all of the services which rely on its affordances would be put in jeopardy. In the event that large-scale electronics manufacturing were to suddenly halt, or a region were to be cut off from the global supply chain, computing devices would become scarce resources. Furthermore, without reliable power generation and distribution, long distance communication over the Internet as we know it today would likely not exist, even if networking hardware could be maintained. Lack of connectivity would render all modern network-based services and software maintenance infrastructure defunct. To retain the functional benefits of computing, we would need to conserve existing hardware and software resources until we learned as a society to recreate their functional equivalents in a more sustainable way, or do without them.

1.1 Assumptions

The production of computing resources currently rests on massive technical, social, and economic infrastructures. Tomlinson et. al.'s 2012 paper Collapse Informatics proposes the idea of "Peak ICT," wherein the decline of fossil fuels ("Peak Oil") lead to reduced ICT creation and operation [58]. In this work, we assume a slightly more general "Peak ICT" scenario where, whether due to a lack of raw materials, access to production facilities or energy to run them, a disrupted supply chain, or any combination which we believe likely in the event of collapse: 1) the manufacture or acquisition of new integrated circuits (ICs) is prohibitively difficult and 2) long distance networking and information sharing becomes difficult with the decay of Internet infrastructure.

In their work discussing a minimal set of devices and protocols required to reproduce the functionality of the Internet, Raghavan and Hasan detail the extensive network of resource dependencies involved in hardware device manufacture, and recommend reducing these dependencies [42]. However, we assume most communities will not have specially-architected computing devices designed for the loss of present-day manufacturing infrastructure. Most people's only recourse upon failure of hardware components will be to repair them or procure replacements from those manufactured before collapse.

Hardware alone does not a modern computing platform make. We also anticipate a slew of challenges related to maintaining the correctness of software and user data, especially due to malware infections and "bit rot" on storage media not designed for decades of integrity.

Evidence from developing regions today suggests that we should expect malware to remain an issue as long as computers engage in networking and file transfer over any medium, even (perhaps especially) in the absence of global connectivity. Without connectivity, vulnerable USB drives or direct ad-hoc wireless connections are the file transfer mediums of choice; without access to official distribution channels, the only way to acquire software and media is often through the illegal downloading and sharing [8, 51]. Cracked software and digital rights management stripped media is frequently contaminated and becomes a vector for malware transmission [16]. Many computer users in developing regions cope with malware-infected systems at substantial cost to productivity and security [7, 11, 24, 35]. The contemporary experiences of users in these conditions inform our expectations of a future collapse computing scenario.

Furthermore, long-term connectivity loss and lack of a centralized trust infrastructure break many processes fundamental to modern software design, development, distribution, and verification. We hypothesize a dramatic reduction in the authoring and dissemination of software after collapse, to the point where patches and security fixes are no longer widely available.

Finally, we suggest technologies, practices, and social infrastructures yet to be developed that could mitigate the risks collapse imposes on keeping both software and the hardware it runs on functioning in an environment adversarial to users and developers.

2 MITIGATING HARDWARE RISK

2.1 Computing Usage and Environment

We consider two usage scenarios which may characterize either end of a spectrum of computer lifetimes. In scenario one, dedicated computers are set aside for the operation of critical services, such as weather modeling or database accesses, and are kept in a controlled environment such as a clean room to consciously maximize longevity. Scenario two is that of a personal computer, probably a portable laptop, used as is typical today without any special protection from the elements. We use the two scenarios to separately reflect on the inherent effects of computational load and external environmental effects such as impact, water damage, or particle intrusion.

Our motivation for this separation is that many types of damage come from the external environment and can be almost entirely prevented through stringent environmental control and limited device mobility. For example, dust and dirt on electrical components can prevent proper cooling, increasing their chance of failure. Humidity or spilled water can corrode circuits or cause shorts that lead to component damage. Accidental impact due to dropping or jostling during transport may result in mechanical damage to the screen, keyboard, ports, fans, and the chassis, opening additional entry points for dust, dirt, and water. Strict control of the material computing environment and avoidance of machine transport mitigate many of these risks. We argue that environmental control can increase device longevity at the cost of losing some of the social functions of computing permitted by mobility today.

2.2 Computation-limited Components

Computational loads themselves contribute to physical wear on many components, leading to performance degradation and eventual failure with regular use of the computing resource. Storage drives are one such component. A casual study of Internet forums on computer repair suggests that hard drive replacement is one of the most common repairs performed on consumer machines, for a variety of reasons ranging from mobility-related damage to performance deterioration from component wear over time.

The industry standard for manufacturers to provide estimated lifetimes for HDDs has historically been Mean Time Between Failures (MTBF) or Mean Time To Failure (MTTF), measured in hours of uptime. Common MTTF ratings for modern consumer-grade hard drives range from 100,000 to 1,000,000 hours, which represents roughly 100 years of continuous use. In reality, however, real world data has shown that modern consumer HDDs fail at rates of around 2-5% per year, with an observed acceleration to around 10% after the first four to six years [4, 5, 50]. A generous estimate at the original 2-5% puts the half-life of a HDD at 13.5 to 34 years; with the increased failure rate from 5% to 10% after the first 6 years, the half-life is 9.7 years. Furthermore, these empirical failure rates were measured in datacenters, where the drives would have been largely protected from unpredictable power fluctuations and physical damage. Power outages are known to cause "head crashing" in HDDs, where the mechanical disk head snaps back to a starting position upon loss of power and potentially scratches the disk platter [30]. Since HDDs are considered very difficult to repair with common tools, we propose that when worn out or damaged (perhaps every 10-20 years), they will need to be replaced.

SSD manufacturers typically provide lifetimes in terms of number of writes to the drive, since molecular wear occurs with each write on the flash memory gate storing the written value. For example, one 120 GB Samsung SSD has a lifetime of 100 terabytes written. At the typically cited estimated "average" workstation usage of 10 GB per day, this SSD has a lifetime of about 28 years, with lifetime scaling roughly linearly with the size of the drive [1]. Therefore, we propose that a SSD will only need to be replaced every 20-30 years at this stock workload, though performance will decrease steadily throughout the drive's lifetime as cells fail, and may drop below that required by the user. Write intensive workloads will naturally lead to much faster SSD failure depending on the nature of the workload.

Parts with moving components other than HDDs, such as optical drives and fans, are also susceptible to wear over time, but have been less well studied. MTBF values for consumer CPU fans are typically specified in the 30-50,000 hour range, or 3.4-5.7 years, though high-end CPU fans can be found with listed MTBFs of 28 years [37]. However, unlike HDDs, fans are amenable to cleaning, lubrication, and repair, and may not need to be replaced as often with regular maintenance [13].

Finally, some components age over time via chemical processes. One common repair is the replacement of electrolytic capacitors in a power supply unit (PSU) or on a motherboard, due to the slow

evaporation of the electrolyte resulting in decreased capacitance. Typical consumer electrolytic capacitors are rated to run for 2000 hours at either 85C or 105C; depending on the type, at a working temperature of 45C they will have a lifetime of around 3.7 or 14.6 years of continuous use, respectively, with the lifetime highly dependent on temperature [15]. Unfortunately, unused electrolytic capacitors have a shelf life of only 2-3 years, due to degradation of the aluminum oxide layer insulating the capacitor foil. They may be usable after "reformation," in which a DC voltage is applied to the capacitor over a period of days or weeks to restore the aluminum oxide layer [43]. A better solution might be to replace the electrolytics with a few smaller but longer-lasting ceramic capacitors (lifetime 100+ years) in parallel and a resistor in series to mimic the properties of the electrolytic capacitor [56].

Also, after just a few years depending on environmental conditions such as temperature, thermal grease applied between a CPU and heatsink may solidify and crack, introducing air gaps that decrease the effectiveness of cooling. It is unclear from our research exactly when this happens or whether it can be prevented; however, if detected before any damage occurs to the CPU, the hardened grease can be removed with an organic solvent and reapplied. If damage does occur to the CPU, a replacement chip must be procured and substituted, which may be possible or prohibitively difficult depending on whether the CPU is socketed or soldered directly to the motherboard.

2.3 Environmental Management

In order to maintain a longevity-friendly environment for computers in scenario one, the units would ideally be kept in a clean-room-like environment, with air filtering, rigorous entry and exit protocols, low humidity, and cool temperatures to avoid overheating [22]. Regular maintenance, such as cleaning of parts vulnerable to dust such as fans, could also prevent avoidable damage. Finally, one of the most important features of this environment would be a clean, reliable source of power to prevent surges and outages that would damage either the computing devices or the equipment being used to maintain favorable environmental conditions for its survival.

2.3.1 Power Management. Computing will only be possible with some power source, whether via intermittent grid electricity or an off-grid solution. An exploration of the space of power systems that could provide clean, reliable power for computing devices is out of the scope of this paper, but we describe one such minimal, off-grid system to show that it would be feasible to build and maintain.

The following system is based on current solutions for off-grid power used in RVs and boats: A constant-voltage DC power source such as a solar panel charges a 12V battery system, either a 12V car/marine lead-acid battery or pairs of 6V go-kart/motorcycle lead-acid batteries, with a simple low voltage indicator (made from LEDs and resistors, with no IC). A 12V DC car/marine PSU draws power from the batteries, and powers the computer. When the sun is shining, the solar panels charge the batteries up to their "full" voltage via constant-voltage (CV) charging; as the computer runs, it drains the batteries until the low voltage indication, at which point the user should turn the system off until the sun is shining again.

Each part of this system is essential: the solar panels produce power, the batteries handle input dropouts, and the PSU takes the slightly-fluctuating DC input and produces clean power at multiple voltages. Common warranties on modern solar panels guarantee an output of no less than 80% of the rated power over the first 25 years of use. However, with a typical degradation rate of 0.5% a year, the output should not fall below 80% for the first 44.5 years [32]. Typical lead-acid car batteries last 0.5-4 years inside a car depending on usage, but would last longer in more favorable temperatures and avoiding deep discharge while attached to a solar panel [27]. Sealed lead-acid (or VRLA) batteries last up to 10 years without maintenance, and even after sulfation are regularly revived and reused [40]. We expect commonly available DC/DC PSUs to also have electrolytic capacitors, and therefore similar lifetimes to AC/DC PSUs; to extend their lifetime, the same capacitor replacements as described above would be required. Therefore, we conclude that computing would likely not be limited by a lack of mains power; it would be feasible to maintain a power system that would last the lifetime of a computer and inflict minimal damage on its hardware.

If an inverter (with a standard life expectancy 10 years [49]) were added to the system, a standard AC PSU could be used instead of a DC one, and lead-acid batteries could be skipped for an off-the-shelf uninterruptible power supply (UPS) system (with a life expectancy of 3-5 years [52]). It would also be feasible to reconstruct the function of a UPS with a charging circuit, lead-acid battery, and an inverter, which would likely be more robust and have a longer shelf life than a UPS. Many options exist for powering computation according to need and hardware availability at the time.

2.4 Mobility-limited Components

For a baseline failure rate for mobile computing, we refer to a Consumer Reports study in 2015 that claimed modern consumer laptops have a 10-20% chance of failure over the first three years of ownership, with a median of 18% [55]. The median half-life computed from this value is 10.47 years, although as we have explored in previous sections, the annual failure rate of hardware tends to accelerate with age. According to an older study by SquareTrade in 2009 [47], which cited a higher failure rate of 30% over the first three years, about a third of laptop failures were due to accidents as opposed to malfunction. As hardware reliability has improved with SSD proliferation, this proportion has likely risen. Specific repair challenges are detailed below.

Mobile laptops tend to suffer damage from exposure to heat, dust, dirt, and water (especially containing salt). To repair corrosion and/or shorted electronics due to water damage, the corroded metal can be removed using isopropyl alcohol, and the electronics can be replaced by soldering. However, this kind of repair takes considerable care, effort, and expertise, especially with the tight integration and decreasing size of hardware components in modern laptops.

Another limit to longevity is that laptop batteries are consumables with a lifetime of 2-5 years, and need to be replaced for the continuation of mobile use, though said replacement is trivial to perform when the part is available [10].

Table 1: Summary of Recommended Replacement Parts and Estimated Lifetimes

(H) in the Estimated Life column indicates that the value is a half-life computed from other ratings.

Limited by	Part	Estimated Life (yrs)	Notes
Computation	Ceramic capacitors	100+	(MLCCs) Could replace electrolytic capacitors
	SSD	20–30	120 GB SSD at 10 GBW/day
	HDD	9.7–13.5 (H)	At 5% baseline failure/yr
	Electrolytic capacitors	3.7–14.6	Affects PSU, motherboard, AC/DC adapters
	CPU fans	3.4–5.7	Longer life with cleaning/lubrication/repair
	Thermal grease	2+	Depends on temp and conditions
Mobility	Aggregate of device parts	10.47 (H)	Based on Consumer Reports study
	Peripherals	Variable	Screens/monitors, keyboards, mice
	Li-ion batteries	2–5	Computer technically works without batteries
Power	Solar panels	50+	$\geq$ 80% of rated power output for first 25 yrs
	DC/DC PSU	3.7–14.6	Assumed limited by electrolytic capacitors
	Inverter	10	Standard for solar inverters
	Sealed lead-acid battery	10	Easy to repair with standard tools
	UPS	3–5	May also be built w/ lead-acid battery
	Unsealed lead-acid battery	0.5-4	Easy to repair with standard tools

Repetitive physical handling due to mobility can lead to mechanical constraint wear on case screws, tape, and glue (especially after multiple repair-related disassemblies). Laptop form factors tend to differ significantly between models, so if the chassis is cracked or falls apart due to being handled roughly or dropped, a replacement may have to be fabricated from some renewable material such as wood (which has been proposed for laptop chassis in some renewable designs [21]).

On the other hand, while the chassis and peripherals may be flimsy or complicated to replace, laptops are designed to be compatible with a large variety of spare peripherals such as external monitors and USB mice and keyboards. A laptop may theoretically remain usable for computation long after the chassis has been replaced by a box housing just the motherboard, storage drive, and peripheral connectors.

Ruggedization against foreign particle entry might also help mitigate exposure to the elements. Specifically designed ruggedized computing devices are costly but available according to military specifications [59] for applications such as warfighting or construction. At the most basic level of protective design, a HDD in a laptop can be replaced with a SSD before mobile use in order to avoid mechanical damage to the storage drive, and potential errors in stored data.

2.5 Resources for Repair

In order to sustain the repairs mentioned above, replacement parts must either be kept in stock by the device owner, or available through a procurement network. For the mobility-limited scenario, we discussed needing HDDs or SSDs, fans, electrolytic capacitors, PSUs, thermal grease, and possibly CPUs. Additional parts would be desired for the mobile scenario, including Li-ion batteries, screens or external monitors, keyboards, mice and other peripherals, and AC/DC adapters and cords if AC mains power was still available. See Table 1 for a summary of commonly required parts. The question remains whether all of the the replacement components will have shelf lives long enough to be usable for repairs after fifty or a hundred years. For example, SSDs packed for long term storage in a temperate environment, with desiccant, and away from radiation, are likely to remain in good condition after 15 years or more, because integrated circuits are expected to last as long under the same circumstances [31]. However, not much work has been done on measuring their shelf life for longer periods.

Just as important for successful repairs would be people with the skills needed to perform them, such as soldering and using a multimeter. Without intentional teaching and community retention of these skills in a generation of less computing ubiquity than we have today, the skills could be lost to many communities. Social networks or institutions of people interested in computer repair could be invaluable for sourcing parts and maintaining skills needed to keep computing alive until devices and power are no longer scarce.

3 MITIGATING SOFTWARE RISK

Software degradation is less predictable than hardware degradation and subject to different challenges under collapse. While software does not "wear out" like hardware, it can be slowly corrupted over time, often has external dependencies, and is subject to contamination from the outside environment.

3.1 Limits on Development and Distribution

In a scenario where power and manufacturing infrastructure are unavailable, the Internet would likely cease to exist as well, which poses an enormous number of threats to modern software functionality. Firstly, software distribution would mostly cease, as would the distribution of bug fixes and security patches, which would still be needed given malware's ability to survive outside of the Internet in regions with low connectivity [11, 19]. Secondly, cloud infrastructure would not be available, which would suspend all web services immediately, and render renewable-license software void

with no means to renew at the end of the license period. Finally, software development would slow to a crawl without the current development ecosystem, which has co-evolved with increasing societal connectivity. Unfortunately, software development would be needed as a crucial line of defense against malicious software (malware) infection, another significant threat to computing discussed below.

Rapid innovation in software today depends heavily on web-based tools for easy long-distance discussion, technical search, and software distribution. Without communication tools, online documentation, and cloud-hosted code repositories such as github and npm to facilitate collaboration, developers would have to work and learn individually through time consuming experimentation, likely replicating each other's code [38].

3.1.1 Solutions for Developer Collaboration. Collaboration through distributed version control tools would be possible without the Internet, but would require either co-location of developers or the establishment a highly reliable developer network. From our discussion on hardware above, we believe that tightly integrated mobile computing platforms like modern laptops will fail faster than stationary desktops and servers with easily replaceable components. The eventual depletion of mobile computing resources will make it increasingly difficult to gather people and their computers in a single location. Therefore, it may be crucial to establish communication channels, file sharing practices, and communities for maintaining software engineering knowledge before the breakdown of mobile computing.

These communications could be as simple as broadcasting code over radio, which was done in Finland in 1985 as part of an effort to stimulate interest in computing [29]. Another strategy could be to establish decentralized communication networks over sneakernet with cryptographically assured messaging, or point to point wireless systems as inspired by community networks and ham radio [14]. Regardless of communication medium, person to person networking will be an important part of post-collapse computing, without centralized Internet communities to establish reputation and put developers in initial contact with one another. When remote collaboration becomes infeasible, computing centers could be established to bring software engineers to the same physical location to allow in-person collaboration.

3.2 Data Decay over Time

High barriers to verified file sharing also create challenges to maintaining correct copies of data, including software. All data is vulnerable to subtle faults of the underlying hardware it is stored on, including in-memory bit flips [17, 54] and on-disk file corruption [18]. The widely deployed Windows operating system does not implement error correction codes in its default filesystem, and commodity consumer hardware eschews error correction-enabled memory for lower cost and higher performance. Over time flaws will accumulate; while corruption to non-essential files could be harmless, corruption of key files in the operating system or critical user applications could cause irrecoverable failure of the overall computing resource. In our well-resourced world we can ignore these issues because it is easy to reinstall an application, and software lifetimes are relatively short. However, in a collapse scenario, everyday users must take on the burdens of data management and preservation that are left to data center administrators and archivists today.

3.3 Trust Breakdown

A fundamental but relatively invisible piece of modern software infrastructure is the ubiquitously available public key infrastructure (PKI). Centralized certificate authorities sign and validate website secure socket layer (SSL) credentials, software packages, and system updates to give end users a reasonable way to validate their authenticity. While nothing in the cryptographic principles of PKI requires centralization, it does require a root of trust upon which chains of trust can be built to validate third parties. In an environment with extremely limited connectivity, it will be difficult for content creators to obtain digital signatures that will be trusted by all the end users that content may eventually reach. Most SSL certificates distributed with browsers and operating systems have expiration dates, beyond which key invalid errors will be thrown by the validating software. As seen in Figure 1, all root certificates on a currently up to date system will be invalid in 30 years. While users can continue to rely on expired keys, they will have to override warnings and run the risk of long-held keys being compromised with no way to get replacements.

A systematic breakdown of the current signing infrastructure will further complicate the problem of software authenticity verification and increase the chances that normal users encounter malware through compromised content. Without an understanding of how PKI operates users will have a difficult time handling the remnants of the current implementation and making the right choices with regards to trust and system security that are handled transparently today.

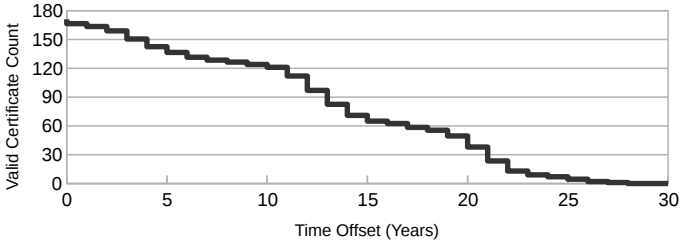

Figure 1: Valid Certificates vs. Time
Measured from expiration dates on installed SSL root certificates on an up to date Ubuntu 16.10 system.

3.4 Malicious Software

Finally, we see malware as potentially the single largest threat to productive computing after collapse given its volatile nature and high risk of harm. Malware can cause varying levels of disruption to a computing system, ranging from passive non-interference up to catastrophic data loss or even irreparable hardware damage [3, 28, 33]. In a collapse scenario where new, trusted copies of data cannot be easily retrieved and systems cannot just be wiped and reinstalled, users may have to cope with the effects of malware infections indefinitely.

While a collapse event significant enough to impact computing capability may also diminish incentives to create malware due to decreased computer usage, in some scenarios they may actually be enhanced. For example, in a collapse triggered by warfare, cyber weapons may be intentionally developed and deployed by opposing factions to harm critical infrastructure[26, 48]. Collateral damage from such weapons could spread unchecked through the software ecosystem if no countermeasures are in place. Malware authors also write for a variety of other personal motivations, such as boredom, which may not disappear after collapse [53]. It only takes one developer to create and release a piece of malware, but containing it requires coordinated effort to update the systems of a large number of vulnerable users. Furthermore, current malware in the wild will not cease to exist, and users will have to contend with any malicious code deployed but not yet patched at the time centralized update services fail.

Many types of malware have the advantage of spreading virally through incidental contact with other systems, while patches, not commonly spread peer to peer, will be slowed by the destruction of centralized distribution channels [9]. Without the Internet, users would have to rely on peer-to-peer file transfers to productively exchange megabytes of information [51]. Direct file transfers provide no way to verify the authenticity of received files before opening, and without updates to malware signature databases users will have no way to identify new malware in received files [16].

3.4.1 Software Recovery. Presently only two main models currently exist for the recovery of systems compromised by malware. The first involves expert security researchers and developers characterizing malware infections, designing a tailored removal tool, and deploying that removal tool to infected users to restore their systems. Experts also generate signatures of the malware to detect and prevent future infections. Severe collapse scenarios preclude usage of this model due to a lack of connectivity for experts to gather malware samples from the broader user base and then distribute fixes. Isolated groups of users will likely not have access to the expert resources and time required to solve problems in this manner.

The other more extreme model, completely wiping and restoring the computer from a new OS image, is often used as a last resort in developed countries against rootkits or sophisticated malware, and as a regular cleansing operation in developing contexts where tailored fixes may not be available [7, 19]. The source image for the new operating system install can come either from a restricted partition on the user's hard drive or from a dedicated piece of external installation media. On new machines commonly provided without disk drives today, the partition approach is favored for most users to decrease costs on the manufacturer and simplify recovery. However, the partition approach presents several notable disadvantages: since the partition is always physically present on the computer sophisticated attacks could bypass OS security measures and modify data on the partition to infect the recovery image. Similarly, since the partition is tied to the same physical disk as the running OS, failure or corruption of that disk could damage the image. Lastly, the image will still be vulnerable to the original exploit and reinstalling it will not prevent future infections.

In a collapse scenario long term maintenance of reliable backup data becomes both much more important for system longevity and much more difficult to achieve with limited resources. Present day solutions rely on software to manage backup images, but secure hardened backup stores grounded in hardware would provide more assurance that software bugs could not be exploited to gain access. Physical switches allowing read, append, or write access to hardware isolated storage would help users take control of their backup data storage reliably and explicitly.

New sophisticated attacks have been recently uncovered that target low level device firmware on the system's hardware itself, persisting across a complete OS level restore [20, 61]. Recovering from these attacks requires either acquiring new hardware or having access to low level firmware flash tools as a part of the recovery process. Without planning for such a contingency prior to collapse, users infected with this type of malware could be unable to restore their systems to working condition [46].

3.4.2 Sustainable Malware Inoculation. While malware has the advantage of self-replication and contact spread, the same principles could be applied by trusted software sources to distribute patches organically. As demonstrated in Ghana by the FlashPatch project [11], it is possible to piggyback antivirus definitions onto regular file transfers over USB, reaching machines otherwise cut off from network connectivity. The same system could be used to transport signed OS packages or core firmware updates which could be incorporated and passed on automatically by end users who trust the original signing authority. Such a system would require a distributed web of trust based on strong cryptography to allow software packages to be validated securely on remote machines that may have never directly communicated with the creating entity. Such a web could be built with primitives that exist today, such as PGP signatures or another certificate framework. Additionally, long term viral distribution of OS patches would require a user-friendly way to manage patch conflicts (imagine two disconnected developers fixing the same bug) and a way to condense layered patches to keep the space required for their distribution in check over time. While storage is relatively large and inexpensive relative to the size of required packages, there is currently no approach for managing patch conflicts at any level higher than the source code. Further research would be required to enable distributed updates in a transparent and user-friendly manner.

3.4.3 Malware-Tolerant Systems. An important aspect of sustainable defense against malware will be not only preventing infection (as it will become increasingly unavoidable), but containing the damage that follows. One approach to increasing system resilience involves sandboxing different parts of the computer system at a low level to provide high assurance of isolation and better user visibility into system behavior. In security-oriented operating systems like Qubes [41], virtualization technology separates small parts of the operating system into isolated zones with well defined communication permissions and protocols between them [45]. This minimizes the attack surface exposed by each component while allowing users to catch anomalies in communication through intelligent monitoring. Action can then be taken to replace compromised zones before the infection spreads to the entire system and user data is compromised. Replacing a single zone of the system is much

easier and lower-cost than restoring the entire OS. Additionally, hypervisors enforcing virtualization security policy can be simple and minimalistic enough to be formally verified and guaranteed to meet security specifications [6, 25, 39].

Other resilience models are possible as well, potentially drawing from existing concepts in fault-tolerant computing or the design of secure information systems for classified data [23, 36]. Approximate computing techniques could even be applied where multiple runs of a computation are attempted in corrupted environments and results are combined intelligently to catch and repair introduced errors. The high performance computing community is already exploring such techniques for large scale computing at the limits of error correcting code memory [17, 54]. As long as the "viral load" and corruption introduced into the computation was low enough, useful information could still be extracted from compromised compute resources.

4 DISCUSSION

4.1 Designing Systems for Collapse

Emphasizing longevity and repairability instead of up front cost, maximum initial performance, or low size, weight, and power significantly changes the tradespace in designing a computing machine [44]. Present day conditions without limits incentivize design and construction of machines, which while capable in the present environment, may not be adequate for sustainable computing in an extremely limited collapse environment. Notably, IT professionals often focus on hardware longevity in planning for overall system longevity, assuming the availability of valid software, global connectivity, standardized architectures, and a strong network of software developers. However, in a collapse scenario, access to both replacement hardware and up-to-date, uncorrupted software will become limiting system constraints.

4.1.1 User-Mediated Security. An important fundamental paradigm for collapse computing will be putting control of system security back into the hands of users, with human factors in mind. Cut off from centralized services of security researchers and patches, users will need the tools to take system and network security into their own hands. Permissions based systems, like User/Group permissions in Unix derivatives, provide some security; however, they are often difficult for users to understand and configure correctly deprived of context, and present too many uninformative, ignorable prompts [57, 62]. General purpose monitoring tools like file system monitors, registry watchers, or network traffic classifiers increase system transparency at the risk of overwhelming users with false positive warnings and drowning attack signals in noise from nominal system operation [62]. Ongoing work on privilege elevation triage and system security transparency could make systems better able to detect threats from noise by adapting to expected usage patterns and local states. Once updates cease, malware that works around rigid security paradigms will probably proliferate, but well designed human-in-the-loop security paradigms could continue to function as non-technical end users modify their best practices in response to threats evolving in the wild.

More research could also be done towards establishing strong user data protection in the face of system compromise. Hardware enforced filesystem access could protect critical data stores for keys or recovery images by requiring explicit physical action from the user to enable reads or writes. Hardware could also enforce backup policies, ensuring that recovery copies of data always remain available, or that attempts to destroy or modify backups are brought to the user's attention.

4.2 Social Mechanisms for Maintenance

Per the above analysis, we might expect computing hardware and software to persist in well-maintained environments for several generations, and in mobile forms for approximately one generation. This multigenerational effort relies upon a knowledge and culture of maintenance, and so may fail for cultural reasons; just as we have considered the obsolescence of computing hardware and software, so too must we consider the obsolescence of computing culture, and how it might persist or rot.

History offers many examples of infrastructural maintenance after a collapse, but two interestingly divergent ones are the Chinese and Roman road networks built from around the second century BC to the third century AD, and decaying thereafter. While the Roman network decayed rapidly, contributing to cultural disconnects of the early Middle Ages, the Chinese road network was maintained, albeit reduced from wide roads that could handle drawn carts to narrow ones designed for wheelbarrows [12]. This maintenance was performed by cultural organizations such as the Taoist Yellow Turbans and Buddhist fraternities as a component of their training and service. Perhaps computing could continue similarly after collapse, as public enclaves maintained by semi-ascetic cultural organizations whose primary focus may or may not be computing. Such a situation might lead to a kind of software and hardware monoculture designed for application by non-technical adherents.

For a social model more preserving of technical development effort we can look to the history of early personal computing. As hardware began to enter the mainstream, enthusiast groups maintained and created many of the shared understandings and technologies that allowed individuals to engage with computing [60]. Were post-collapse computing to follow this framework, much of our current technical knowledge, computing heterogeneity, and software development ecosystem might be maintained, but with informal software distribution channels malware could be quite a burden.

Even further back in the history of computing, we recall the development of LISP, whose fundamental lambda calculus was specified in the mathematics literature [2] two decades before it was used for computing [34]. Even as computing collapses, a rich body of computer science literature could survive. New results in encryption, compilers, and other immediately applicable research could be argued mathematically before being input to rare computing resources. Computing could be reserved to polish and finish work already peer-reviewed, maintaining a capable and trusted but highly restricted computing resource for the academic community.

In the discussion of PKI infrastructure above, the importance of trusted transportation was mentioned; historical analogues for this might include the early postal systems of Europe and the Pony Express. Such logistical businesses could of course benefit heavily from computing themselves; one could even imagine overlapping competitive transportation networks offering computing services

and software patches from afar, an environment which would fully explore both hardware longevity after collapse and the dangers of malware.

Taken together, these historical examples make it clear that along with the analyses of hardware and software, the roles that computing might take in society are important factors for the continuation of computing after collapse. What groups will have access to what computing resources? Will these resources be captured and centralized by groups with power, or maintained in a decentralized fashion? How will the education and training necessary to fully utilize and adapt computing to new societies be passed down from generation to generation? These questions call for the study and creation of sustainable and resilient modern computing cultures.

5 CONCLUSIONS AND FUTURE WORK

Collapse scenarios present existential challenges to the preservation of computing capability in the post-collapse context. Hardware, software, and user data all face threats to survival in an environment with limited replacement part availability, limited communications and power infrastructure, and limited software development capabilities. While there are challenges to maintaining hardware in such a constrained scenario, they are relatively well-understood. With sufficient replacement parts and care, a commodity computer may be maintained and powered for the duration of a temporary collapse of several decades. Software, however, presents a set of challenges that are harder to mitigate, as the detrimental effects of long term disconnection, software data corruption, and malware are numerous and potentially devastating.

Further research on computing within these limits could directly benefit users in today's collapse scenarios while improving the survivability of computing as a whole. Significant areas for future work include: further investigation into the longevity and care of hardware in use and storage, to improve the overall environmental sustainability of computing; development of flexible user-centric security paradigms so systems can adapt to changing threats without regular software updates; computing systems designed for secure full recovery in the face of malware infection; and design of distribution technologies to allow secure development and deployment of software without a global Internet.

6 ACKNOWLEDGMENTS

We would like to thank our labmates at the University of Washington and the wider ICTD community for their help and inspiration in the creation of this work.

REFERENCES

[1] SSD Endurance Test - Live Testing Samsung EVO, SanDisk, Intel and Kingston. (????). http://ssdendurancetest.com/

[2] Alonzo Church. 1941. *The calculi of lambda-conversion,*. Princeton University Press; HMilford, Oxford University Press, Princeton, London.

[3] AVTest. *Security Report 2015/2016.* Technical Report. AV Test, Magdeburg German. https://www.av-test.org/fileadmin/pdf/security_report/AV-TEST_Security_Report_2015-2016.pdf

[4] Backblaze. 2016. 2016 Hard Drive Failure Rates for 2TB - 8TB Drives. (Nov. 2016). https://www.backblaze.com/blog/hard-drive-failure-rates-q3-2016/

[5] Backblaze. 2017. 2016 Hard Drive Reliabilty Benchmark Stats. (Jan. 2017). https://www.backblaze.com/blog/hard-drive-benchmark-stats-2016/

[6] Gilles Barthe, Gustavo Betarte, Juan Diego Campo, and Carlos Luna. 2011. Formally verifying isolation and availability in an idealized model of virtualization. In *International Symposium on Formal Methods*. Springer, 231–245. http://link.springer.com/10.1007%2F978-3-642-21437-0_19

[7] Prasanta Bhattacharya and William Thies. 2011. Computer viruses in urban Indian telecenters: Characterizing an unsolved problem. In *Proceedings of the 5th ACM workshop on Networked systems for developing regions*. ACM, 45–50. http://dl.acm.org/citation.cfm?id=1999940

[8] Jay Chen, Michael Paik, and Kelly McCabe. 2014. Exploring Internet Security Perceptions and Practices in Urban Ghana.. In *SOUPS*. 129–142. https://www.usenix.org/sites/default/files/soups14_proceedings.pdf#page=136

[9] L.-C. Chen and K.M. Carley. 2004. The Impact of Countermeasure Propagation on the Prevalence of Computer Viruses. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)* 34, 2 (April 2004), 823–833. https://doi.org/10.1109/TSMCB.2003.817098

[10] Apple Computer. Determining battery cycle count for Mac notebooks. (????). https://support.apple.com/en-us/HT201585

[11] Henry Corrigan-Gibbs and Jay Chen. 2014. FlashPatch: Spreading Software Updates over Flash Drives in Under-connected Regions. ACM Press, 1–10. https://doi.org/10.1145/2674377.2674384

[12] Kris De Decker. 2011. How to Downsize a Transport Network: The Chinese Wheelbarrow. (Dec. 2011). http://www.lowtechmagazine.com/2011/12/the-chinese-wheelbarrow.html

[13] eBay. 2016. How to Repair a CPU Fan. (March 2016). http://www.ebay.com/gds/How-to-Repair-a-CPU-Fan-/10000000177770703/g.html

[14] Kevin Fall. 2003. A delay-tolerant network architecture for challenged internets. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM, 27–34. http://dl.acm.org/citation.cfm?id=863960

[15] Mark Fortunato. 2013. Ensure long lifetimes from electrolytic capacitors: A case study in LED light bulbs. (April 2013). http://www.edn.com/design/analog/4411475/1/Ensure-long-lifetimes-from-electrolytic-capacitors--A-case-study-in-LED-light-bulbs

[16] John F. Gantz, Pavel Soper, Thomas Vavra, Lars Smith, Victor Lim, and Stephen Minton. 2015. Unlicensed Software and Cybersecurity Threats. (Jan. 2015).

[17] Al Geist. 2016. How To Kill A Supercomputer: Dirty Power, Cosmic Rays, and Bad Solder. (Feb. 2016). http://spectrum.ieee.org/computing/hardware/how-to-kill-a-supercomputer-dirty-power-cosmic-rays-and-bad-solder

[18] Jim Gray and Catharine Van Ingen. 2005. Empirical measurements of disk failure rates and error rates. *arXiv preprint cs/0701166* (Dec. 2005). https://arxiv.org/abs/cs/0701166

[19] Shimin Guo, Mohammad Hossein Falaki, Earl A. Oliver, S. Ur Rahman, Aaditeshwar Seth, Matei A. Zaharia, and Srinivasan Keshav. 2007. Very low-cost internet access using KioskNet. *ACM SIGCOMM Computer Communication Review* 37, 5 (2007), 95–100. http://dl.acm.org/citation.cfm?id=1290181

[20] Alex Hern. 2015. Lenovo does it again as LSE component removed after security fears. *The Guardian* (Aug. 2015). https://www.theguardian.com/technology/2015/aug/14/lenovo-service-engine-pre-installed-security-superfish

[21] Stewart Hickey, Colin Fitzpatrick, Paul Maher, Jose Ospina, Karsten Schischke, Peter Beigl, Itziar Vidorreta, Mona Yang, Ian D. Williams, and Emilia den Boer. 2014. Towards zero waste in industrial networks: A case study of the D4R laptop. In *Proceedings of the Institution of Civil Engineers - Waste and Resource Management*, Vol. 167. 101–108. https://doi.org/10.1680/warm.13.00031

[22] Liberty Industries. Cleanroom Operating & Maintenance Protocol. (????). http://www.liberty-ind.com/pdf/maint_protocol_pdf.pdf

[23] Information Assurance Directorate. 2014. The Community Gold Standard Framework 2.0. (June 2014). https://www.iad.gov/iad/library/ia-guidance/ia-standards/cgs/community-gold-standard-framework.cfm

[24] IT News Africa. 2008. 'Raila Odinga' computer virus routs Malawi's cities. (Sept. 2008). http://www.itnewsafrica.com/2008/09/raila-odinga-computer-virus-routs-malawis-cities/

[25] Gerwin Klein, Kevin Elphinstone, Gernot Heiser, June Andronick, David Cock, Philip Derrin, Dhammika Elkaduwe, Kai Engelhardt, Rafal Kolanski, Michael Norrish, and others. 2009. seL4: Formal verification of an OS kernel. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*. ACM, 207–220. http://dl.acm.org/citation.cfm?id=1629596

[26] Ted Koppel. 2015. *Lights Out: A Cyberattack, A Nation Unprepared, Surviving the Aftermath* (1st edition ed.). Crown, New York.

[27] Blair Lampe. 2016. The Average Car Battery Life: When is it Time for a Change? (March 2016). http://knowhow.napaonline.com/average-car-battery-life-time-change/

[28] R. Langner. 2011. Stuxnet: Dissecting a Cyberwarfare Weapon. *IEEE Security Privacy* 9, 3 (May 2011), 49–51. https://doi.org/10.1109/MSP.2011.67

[29] Matthew Lasar. 2012. Experiments in airborne BASICâĂŤbuzzing' computer code over FM radio. (Aug. 2012). https://arstechnica.com/business/2012/08/experiments-in-airborne-basic-buzzing-computer-code-over-fm-radio/2/

[30] Joel Lee. 2014. The Effects Power Outages Can Have On Your Computer. (Sept. 2014). http://www.makeuseof.com/tag/effects-power-outages-can-computer/

[31] R. R. Madsen. 2008. Component Reliability After Long Term Storage. *Texas Instruments* (2008). https://www.smtnet.com/library/files/upload/Component-Reliability-After-Long-Term-Storage.pdf

[32] Mathias Aarre Maehlum. 2014. The Real Lifespan of Solar Panels. (May 2014). http://energyinformative.org/lifespan-solar-panels/

[33] Malwarebytes Labs. 2017. *The State of Malware: 2017.* Technical Report. Malwarebytes Labs. https://www.malwarebytes.com/pdf/white-papers/stateofmalware.pdf

[34] John McCarthy. 1960. Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I. *Commun. ACM* 3, 4 (April 1960), 184–195. https://doi.org/10.1145/367177.367199

[35] Chris Michael. 2009. Computer viruses slow African expansion. *The Guardian* (Aug. 2009). https://www.theguardian.com/technology/2009/aug/12/ethiopia-computer-virus

[36] National Institute of Standards and Technology. 2001. Security requirements for cryptographic modules. *Federal Information Processing Standards Publication Series* (May 2001).

[37] Orion Fans. 2017. Life Expectancy. (2017). http://orionfans.com/how-to-read-a-data-sheet/life-expectancy.html

[38] Birgit Penzenstadler, Ankita Raturi, Debra J. Richardson, M. Six Silberman, and Bill Tomlinson. 2015. Collapse (and other futures) software engineering. *First Monday* 20, 8 (2015). http://128.248.156.56/ojs/index.php/fm/article/view/6123

[39] Geoffrey Plouviez, Emmanuelle Encrenaz, and Franck WajsbÃijrt. 2013. A Formally Verified Static Hypervisor with Hardware Support for a Many-Core Chip. In *Euro-Par 2013: Parallel Processing Workshops*. Springer, Berlin, Heidelberg, 801–811. https://doi.org/10.1007/978-3-642-54420-0_78

[40] PowerThru. Lead Acid Battery Working Lifetime Study. (????). http://www.power-thru.com/documents/The%20Truth%20About%20Batteries%20-%20POWERTHRU%20White%20Paper.pdf

[41] The Qubes OS Project. Qubes OS. (????). https://www.qubes-os.org/

[42] Barath Raghavan and Shaddi Hasan. 2016. Macroscopically sustainable networking: on internet quines. ACM Press, 1–6. https://doi.org/10.1145/2926676.2926685

[43] Tim Reese. Strategies to Repair or Replace Old Electrolytic Capacitors. (????). https://www.nmr.mgh.harvard.edu/~reese/electrolytics/

[44] Christian Remy and Elaine M. Huang. Sustainable Interaction Design: Obsolescence in a Future of Collapse and Resource Scarcity. (????). https://www.researchgate.net/profile/Christian_Remy/publication/282216102_Limits_and_sustainable_interaction_design_Obsolescence_in_a_future_of_collapse_and_resource_scarcity/links/56d01bcf08ae4d8d64a1bdc4.pdf

[45] Joanna Rutkowska. 2014. Software compartmentalization vs. physical separation. (Aug. 2014). http://invisiblethingslab.com/resources/2014/Software_compartmentalization_vs_physical_separation.pdf

[46] Joanna Rutkowska. 2015. State considered harmful. (2015). https://blog.invisiblethings.org/papers/2015/state_harmful.pdf

[47] Austin Sands and Vince Tseng. 2009. SquareTrade Laptop Reliability. (Nov. 2009). https://www.squaretrade.com/htm/pdf/SquareTrade_laptop_reliability_1109.pdf

[48] David E. Sanger and Mark Mazzetti. 2016. U.S. Had Cyberattack Plan if Iran Nuclear Dispute Led to Conflict. *The New York Times* (Feb. 2016). https://www.nytimes.com/2016/02/17/world/middleeast/us-had-cyberattack-planned-if-iran-nuclear-negotiations-failed.html

[49] Narasimhan Santhanam. 2015. What is the Lifetime of Solar Inverters? (Sept. 2015). http://www.solarmango.com/ask/2015/09/28/what-is-the-lifetime-of-solar-inverters/

[50] Bianca Schroeder and Garth A. Gibson. 2007. Disk failures in the real world: What does an MTTF of 1,000,000 hours mean to you? USENIX, San Jose, CA. https://www.usenix.org/legacy/events/fast07/tech/schroeder/schroeder_html/index.html

[51] Thomas N. Smyth, Satish Kumar, Indrani Medhi, and Kentaro Toyama. 2010. Where there's a will there's a way: mobile media sharing in urban india. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM, 753–762. http://dl.acm.org/citation.cfm?id=1753436

[52] Justin Solis. 2015. Tips to Maximize the Life Expectancy of Your UPS System. (April 2015). http://blog.schneider-electric.com/it-management/2015/04/28/tips-to-maximize-the-life-expectancy-of-your-ups-system/

[53] Eugene H. Spafford. 1991. Computer Viruses and Ethics. (1991). http://docs.lib.purdue.edu/cgi/viewcontent.cgi?article=1900&context=cstech

[54] Vilas Sridharan, Nathan DeBardeleben, Sean Blanchard, Kurt B. Ferreira, Jon Stearley, John Shalf, and Sudhanva Gurumurthi. 2015. Memory Errors in Modern Systems: The Good, The Bad, and The Ugly. ACM Press, 297–310. https://doi.org/10.1145/2694344.2694348

[55] Donna Tapellini. 2015. Survey Results: The Most Reliable Laptops. (Oct. 2015). http://www.consumerreports.org/cro/laptops/LaptopReliability

[56] TDK. Guide to Replacing an Electrolytic Capacitor with an MLCC | Multilayer Ceramic Chip Capacitors. (????). https://product.tdk.com/info/en/products/capacitor/ceramic/mlcc/technote/solution/mlcc03/index.html#qnote_02

[57] The Ponemon Institute. 2015. *The Cost of Malware Containment.* Technical Report. The Ponemon Institute. http://www.ponemon.org/local/upload/file/Damballa%20Malware%20Containment%20FINAL%203.pdf

[58] Bill Tomlinson, Michael Silberman, Donald Patterson, Yue Pan, and Eli Blevis. 2012. Collapse informatics: augmenting the sustainability & ICT4D discourse in HCI. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 655–664. http://dl.acm.org/citation.cfm?id=2207770

[59] US Department of Defense. 2008. Department of Defense Test Method Standard: Environmental Engineering Considerations and Laboratory Tests. (Oct. 2008). http://everyspec.com/MIL-STD/MIL-STD-0800-0899/MIL-STD-810G_12306/

[60] Stephen Wozniak. Homebrew And How The Apple Came To Be. (????). http://www.atariarchives.org/deli/homebrew_and_how_the_apple.php

[61] Jonas Zaddach. 2011. *Implementation and Implications of a Stealth Hard-Drive Backdoor.* http://dl.acm.org/citation.cfm?id=2046707 OCLC: 873035573.

[62] Mary Ellen Zurko. 2005. User-centered security: Stepping up to the grand challenge. In *Computer Security Applications Conference, 21st Annual*. IEEE, 14–pp. http://ieeexplore.ieee.org/abstract/document/1565247/