FlashPatch: Spreading Software Updates over Flash Drives in Under-connected Regions

Henry Corrigan-Gibbs
Stanford University
henrycg@stanford.edu

Jay Chen
NYU Abu Dhabi
jchen@cs.nyu.edu

ABSTRACT

Computers in developing regions often lack the Internet connectivity and network bandwidth necessary to consistently download and apply software updates and security patches. However, even unconnected computers contract viruses and malware through the sharing of USB flash drives and other removable media. This paper introduces FlashPatch, a system for distributing software updates to computers in such areas by having software updates "piggy-back" on the existing flow of flash drives in rural regions. FlashPatch requires no changes in user behavior once the software has been installed. We implemented a proof-of-concept FlashPatch prototype and evaluated it in a field trial in Ghana. We present data on the prevalence and spread of viruses at our study site and offer experimental evidence of FlashPatch's effectiveness from a nine-month field trial. We found that FlashPatch provided additional antivirus protection to 30% of the machines in our study without imposing any tangible burdens on the system owners.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection; H.1.2 [**Models and Principles**]: User/Machine Systems

General Terms

Human Factors, Security

Keywords

delay-tolerant networking; developing region; flash drive; malware; software update; virus

1. INTRODUCTION

Keeping software up to date with the latest security patches is a challenge even in well-resourced corporate environments. In emerging regions, where Internet connectivity is often unreliable, slow, and prohibitively expensive, downloading software updates is even more of a struggle. Although these areas may not have reliable Internet access, computer viruses and malware still spread quickly via the physical circulation of USB flash storage drives used for file sharing [3]. Computers in developing regions are thus in a "worst of both worlds" situation with respect to computer security: they are *not* networked enough to download routine security patches but they *are* networked enough to be infected with viruses, worms, and other malware (which we refer to collectively as "viruses").

The proliferation of malware in developing regions is not just a problem of academic interest: computer virus infections have tangible human costs. Viruses cause Internet café owners to lose business, render users' data unrecoverable, and inconvenience computer owners by consuming scarce system and network resources [5].

The difficulty of getting software updates to computers in emerging regions is one example of how computer security "best practices" break down in places lacking broadband Internet access and reliable power. In this work, we attempt to reconcile the assumptions of developers in infrastructure-rich countries with the realities of computer users in infrastructure-poor countries. We do so by introducing a novel technique for distributing software updates in under-connected regions.

This paper presents the design and implementation of FlashPatch, a system that *exploits* the ubiquity of USB drives—normally a vector for virus transmission—to facilitate the distribution of software updates to computers with unreliable, expensive, or non-existent Internet access. FlashPatch spreads software updates to offline machines by having the updates "piggy-back" on the existing circulation of USB flash drives. When a user plugs a flash drive into a computer running our software, the computer will, with the user's consent, copy a number of software updates onto the drive. When the user later connects that same drive to a different machine, which lacks those particular updates, the machine will automatically copy the updates from the USB drive and install them locally.

The driving design principle behind FlashPatch is that the system should require *minimal* user intervention in the update process. To use FlashPatch, a user or system administrator need only install the FlashPatch daemon on the computer—FlashPatch requires no other change in user behavior. Streamlining the user experience in this way is especially important in developing regions, where users may be less comfortable with computer technology and more resistant to change [6]. We envision a system like FlashPatch being distributed as a part of an operating system or antivirus software.

To test the potential benefits of a system like FlashPatch, we implemented a prototype virus scanner that uses a peer-to-peer software update mechanism to keep the scanner's virus definition database current when network connectivity is unavailable. The prototype scanner we developed automatically searches for

ACM DEV-5 (2014), December 5–6, 2014, San Jose, CA, USA.

ACM 978-1-4503-2936-1/14/12 ...$15.00.
http://dx.doi.org/10.1145/2674377.2674384.

infected files on flash drives inserted into every machine running our software.

We deployed our prototype antivirus in Internet cafés, schools, and graphic design centers in a small town in Ghana during an evaluation period of nine months. The 63 computers involved in our study all had some form of Internet access, though connectivity across our study site was unreliable.

We gathered quantitative data on the prevalence and nature of computer virus infections at our study site, which support our hypothesis that even offline computers need periodic software updates to be protected against flash-drive-borne viruses. In particular, we found that the virus population at our field site was surprisingly "young"—out of 674 infected files observed, 63 files contained a virus that had been discovered by anti-virus vendors fewer than six months before our scanner detected it.

Over the course of our pilot deployment, FlashPatch provided additional antivirus protection to more than 30% of the 63 machines running the software. Out of a total of 10,907 computer-hours tracked in our study, FlashPatch-provided updates accounted for a total of 1,721 computer-hours of virus protection (15.8% of the total). Moreover, FlashPatch provided protection to 15 computers for the majority of the time that they were *offline* and could not download software updates over the network.

The key contributions of this work are:

- a novel technical architecture for peer-to-peer software updates in developing regions (Section 4),
- quantitative data on flash-drive-borne computer virus infections in computing centers in rural Ghana (Section 6.3), and
- a field evaluation of a flash-drive-based update mechanism as implemented in an antivirus scanner (Section 6.4).

This work provides preliminary evidence for the utility and practicality of a flash-drive-based update system. To put the ideas behind FlashPatch into a larger computer security context, we discuss in Section 7 how the ideas behind FlashPatch could be incorporated into a commercial antivirus product or operating system.

2. RELATED WORK

Prior work has demonstrated that computer viruses are an economic and social problem in developing regions [3]. Bhattacharya and Thies investigate the problem of computer viruses in Indian telecenters and find that 80% of telecenters suffer from "regular" or "highly detrimental" virus infections [5]. Johnson et al. observe that over half of active IP addresses in a rural network in Zambia were engaged in port scanning, indicative of a malware infection [13]. The experiences of Brewer et al. with computer security problems in developing regions corroborate the environmental findings we outline in the following section.

Paik describes a system for gathering detailed information about computer virus prevalence in developing regions [16]. Our work focuses on the distribution of software updates rather than on measurement of virus infection rates, though we do provide some local data on virus "epidemiology." Others have considered the idea of spreading updates or patches via high-latency links [19]. Our application of these techniques to the context of software updates in developing regions is novel, to the best of our knowledge.

The delay-tolerant networking literature focuses on the problem of running unicast traffic over a large network of unreliable links [9, 12]. We address a much simpler version of the delay-tolerant networking problem, since in our network of USB drives, a single sender (the software vendor) *broadcasts* the same message (the software updates) to all users. In the future, we may apply more sophisticated delay-tolerant routing techniques to our system for collecting log files over the network of flash drives.

Related systems address the problem of replicating a data store over unreliable network links [11, 18, 8]. The complexity of these systems arises because different users can make concurrent writes to the data store. Since only one user in our system (the software vendor) publishes updates, we do not need the full power of these replication techniques.

Other work has used physical transportation of digital storage media in place of always-on network links. DakNet used public buses to transport a digital storage device to provide asynchronous network access to remote areas [17]. In deploying the Ca:sh electronic medical records system on handheld computers, the designers relied on physical transport of flash drives to avoid the unreliable telecommunications network [1].

This work takes inspiration from a proposal of Nowlan and Ford to create a "viral" operating system, in which data, applications, and updates are transparently shared between physical machines [15]. While Nowlan and Ford offer a general vision for operating system design, we focus on the very specific problem of providing software updates to computers in developing regions. We also contribute a field-tested implementation and experimental results.

3. BACKGROUND

This work proposes a system for distributing software updates (antivirus updates, in particular) to computers in developing regions. The design of such a system must take into account the particular environmental, cultural, and technological constraints that arise in computing centers in regions with poor Internet and power infrastructure.

3.1 Computing Environment

In this section, we enumerate a number of the most important constraints that shaped the design of FlashPatch. We also briefly outline the effect that each of these constraints has on the traditional model for distributing software updates, in which computers periodically download software updates over the network.

Since we conducted our field work in a particular region of rural Ghana, some of the design constraints may be unique to that particular cultural and geographical setting. Even so, prior work on computing centers in India [5], Zambia [13, 14], and elsewhere [6] demonstrates that many of the design constraints we observed in Ghana likely also hold in other developing regions.

Unreliable Internet Access. One of the most salient features of many rural computing environments is the lack of reliable and affordable Internet access. Although 3G Internet coverage now reaches into some rural areas, large swaths of many developing regions—including portions of our field site—have no reliable options for Internet access. A secondary school near our field site in Ghana, for example, had a 40-computer technology lab and a budget for Internet connectivity. Even this exceptionally well-resourced school was unable to procure Internet access because the mobile data signal at their site was too weak to be usable.

The lack of fast and reliable Internet access is a primary barrier to the traditional model of downloading software updates over the network. Savvy computer users often prevent their antivirus software, Flash player, or operating system from downloading updates to save precious network bandwidth for applications that are more apparently useful. Computer users whose machines have no network access at all have no means to download updates, so

their machines will remain out of date until the owner upgrades the software using physical media.

Pay Per MB. Another feature of many rural computing environments is that Internet service providers typically bill customers according to the number of bytes transferred. For example, a major telecommunication company in Ghana offers a package that costs roughly 6 USD that entitles the user to transfer 1 GB of data within a 30-day period.

Since the total file size of Windows updates for a month could be tens or hundreds of megabytes, simply keeping the operating system up to date could consume almost all of the user's monthly data transfer allocation. In addition, the cost per megabyte is often high relative to income in developing regions, which may make downloading software updates prohibitively expensive. For example, downloading the full 420 MB Symantec virus definition database over a popular mobile network would cost 2.26 USD in Ghana today. Ghana's daily GDP per capita (2013) is 5.07 USD [20].

Ubiquity of Flash Drives. Another prominent feature of the computing environment in many developing regions is the use of portable USB flash drives (or "pen drives"). The high cost of network bandwidth means that users rely heavily on flash drives to share software and personal files. In addition, some lower-income users do not own their own computer and so they use a low-cost flash drive as their primary medium for long-term file storage, in place of a personal computer hard drive. These users access their personal files at Internet cafés or, less commonly, at a computer in their workplace.

The pervasiveness of flash drives means that even computers *without* Internet access may frequently be exposed to viruses and malware carried by infected flash drives. Offline computers are particularly vulnerable to compromise, since they will never receive OS or antivirus updates over conventional network update channels.

Windows XP. The vast majority of computers in Internet cafés at our field site ran the Windows XP operating system. Café owners and operators reported that their customers had become accustomed to the Windows XP user interface and were resistant to learning a new operating system. Although offering Windows XP provides usability benefits to café customers, the now-antiquated operating system lacks many of the security features common in current OSes. For example, early versions of XP do not use data execution prevention (DEC/NX) and no versions of Windows XP use address space layout randomization.

Windows XP does separate user account types into "limited" and "administrator" account types, but only one Internet café we visited, out of six, took advantage of user accounts to prevent users (or malware) from installing new software on the machines. In all other cases, café clients used "administrator"-type accounts and thus viruses or trojan horses inadvertently run by clients could completely overrun the machine. Indeed, we saw a number of severely compromised machines. In the worst case, a piece of malware had disabled the Windows "control panel," making it impossible for the café manager to change system settings or uninstall software.

Frequent Reformatting. The commonality of catastrophic computer virus infections in Internet cafés and other rural computing centers means that computers frequently become unusable as the result of virus infection. Since removing viruses from an infected machine—particularly one with a compromised administrator account—is time-consuming and often practically

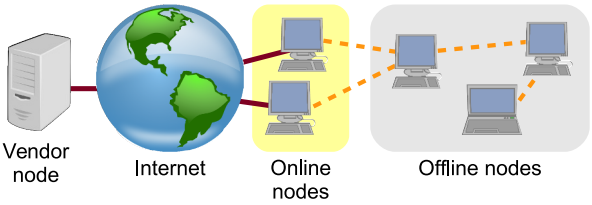


Figure 1: Updates flow over network links (solid lines) to online nodes, and over flash drives (dashed orange lines) to offline nodes.

impossible, computer owners and Internet café managers reformat their machines with relative frequency. The Internet café managers we interviewed formatted their hard drives and reinstalled the operating system every seven weeks, on average.

The process of reinstalling the operating system increases the burden of downloading updates: a newly reformatted machine will need to download all of the updates made available since the production of the operating system's installation CD. In many cases, fully updating a newly formatted computer would require downloading many years worth of software updates and virus definitions. In areas with poor Internet connectivity, computer users may decide it not worth the time or expense to download the large update files after every reformat.

3.2 Problem Statement

The goal of this work is to design a system for distributing software updates to as many machines as possible as quickly as possible in regions with poor Internet infrastructure. We are particularly interested in update distribution systems that are practical *given the existing computing environment and user behavior* in developing regions.

For purposes of this paper, we simplify the problem somewhat by focusing on the distribution of software updates for *one particular* software package, rather than trying to distribute updates for every software package on the machine. We also assume that the software package has a single *vendor* (e.g., a corporate entity) which publishes versioned and digitally signed versions of a data file to a public Web site. The digital signature scheme should satisfy the now-conventional definition of security [10]. Standard digital signature schemes (e.g., DSA) satisfy this property under appropriate cryptographic assumptions.

For our purposes, an *update* consists of a three-tuple

`version_num`, `data_blob`, `signature`

where the version number is an integer incremented with each published update, the data blob is binary data representing the software, and the signature is a digital signature on the first two elements of the tuple using the vendor's secret signing key. For an antivirus program, for example, the data blob would consist of a list of virus signatures. For a chat application, the data blob would consist of the application binary itself.

All client nodes (machines with the software installed) have a copy of the software vendor's public signature verification key, and thus can verify the provenance of a particular update.

4. SYSTEM ARCHITECTURE

In this section, we present the technical architecture of FlashPatch. When using FlashPatch to upgrade many different pieces of software, there can be many different vendors operating at once (one per application), but for simplicity we consider the case of a single vendor updating a single software application.

4.1 Update Distribution

As depicted in Figure 1, a deployment of FlashPatch involves three classes of nodes:

- a single **vendor node**, which publishes updates for the piece of software in question,
- **online nodes**, which are able to download software updates from the vendor, and
- **offline nodes**, which have no network connectivity and can communicate with the online nodes only via flash drives.

A node can oscillate between being online and offline if, for example, the node's network connection is function for only a few days each week.

Whenever the vendor publishes (e.g., to a public website or FTP server) an updated version of its software, the vendor assigns a version number to the new version and digitally signs the entire update package with its private signing key. Online nodes periodically check the vendor's servers for updates, download new versions of the software when available, verify the digital signature on the newly downloaded package, and then install it locally.

Once an online node has downloaded the updated version of the software, it will make this new version of the software available to offline nodes via our flash drive file transfer mechanism. Whenever a user later inserts a flash drive into a node (either online or offline), the node will check to see if the flash drive has a newer signed version of the software than the node does by comparing the version numbers. If the flash drive's version is newer, the machine will install the updated software locally. If the machine's version of the software is newer, the machine will copy the software to the flash drive.

Whenever a user plugs a flash drive into an up-to-date online machine and then into an out-of-date offline machine, the out-of-date machine will be able to retrieve the latest version of the software from the flash drive. Figure 2 depicts this update process. In this way, users who move flash drives amongst collections of online and offline machines (e.g., between online Internet cafés and an offline school computer lab) inadvertently spread software updates.

FlashPatch provides the following safety and liveness properties:

Security. Correct (i.e., not faulty) nodes will only ever apply updates produced by the vendor. That is, at any time period, a correct node will be running a version of the software signed by the vendor, even if the software is somewhat out of date.

Even if an adversary can intercept a flash drive an overwrite its contents with a maliciously formed update, the adversary would be unable to forge the vendor's signature on the malicious update. Since correct nodes will only apply signed updates, the adversary will be unable to coerce an honest node into applying a forged update without breaking the security of the signature scheme.

Eventual Consistency. FlashPatch provides *eventual consistency* [2]: if the vendor issues no new updates and the network of flash drives is not partitioned (every node can communicate with every other node via the network of flash drives), then every node in the system will eventually have the latest update from the vendor.

Ideally, we could provide a stronger consistency guarantee—e.g., that every node in the network would receive every update within one week of the update's publication. Providing such a guarantee, however, would require much stronger assumptions about rate of movement of the flash drives around the network of nodes.

4.2 Log Collection

To facilitate the evaluation of FlashPatch, each node running the FlashPatch client software generates a log file that contains a record of all of the important FlashPatch-related events at that node. Although online machines can upload their log files periodically our server, offline machines have no network connectivity, so we uploading the log files is not an option. To collect log files from offline nodes, we built a system that is the complement of the FlashPatch update distribution system: offline nodes transfer their log files back to online nodes via flash drives, and the online nodes upload these log files to our research group's server.

Unfortunately, deployment constraints meant that we were *not* able to evaluate this log collection mechanism in our field trial. To work around this limitation, our evaluation focuses on the computers for which we do have log files: machines that had Internet access at some point in time during the study. For the benefit of future researchers, we sketch our log collection system design here without making claims about its robustness.

The technical challenge of the log collection scheme is to make sure that the log files eventually reach an online node without distributing so many copies of the log files that the flash drives run out of space. When a user inserts a flash drive into a computer running FlashPatch, the log collection algorithm running on the FlashPatch node has to make two decisions. First, if there are log files on the inserted flash drive, the algorithm must decide whether to copy these log files onto the local machine. Second, the algorithm must decide whether to copy the machine's log files onto the flash drive.

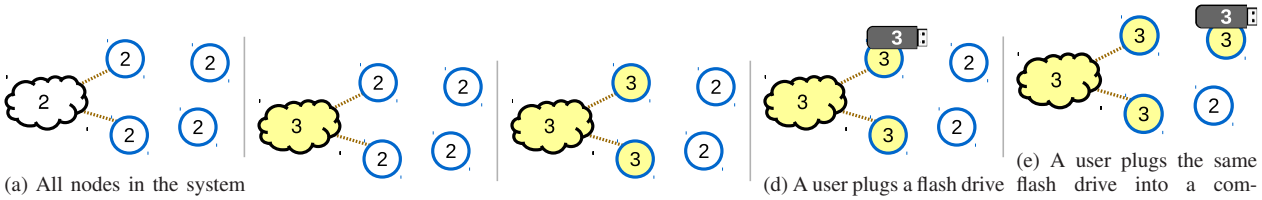
In our log collection algorithm, the computer running FlashPatch tries to decide whether it is "closer to" or "farther from" our research group's server than the recently inserted flash drive. If the computer is closer to our server, then the computer copies the logs *from* the flash drive onto its local disk. If the computer is farther from our server, then the computer copies to log files *to* the flash drive. As this process continues, the log files should eventually move closer and closer to our server until they are at a networked node and can be updated.

The next challenge is to define the distance between a particular node (or flash drive) and our server. We compute the distances using a distributed algorithm. All nodes in the network start out at distance ∞ from our server. When a node connects to the Internet, we set its distance to 1. When user inserts a flash drive at distance d_{flash} into a computer at distance d_{comp}, the FlashPatch software first checks which distance is smaller. If, for example, $d_{\text{flash}} < d_{\text{comp}}$ (the flash drive is closer), then the software sets the computer's distance to $d_{\text{comp}} = d_{\text{flash}} + 1$. If the computer is closer, then the software sets the flash drive's distance to $d_{\text{flash}} = d_{\text{comp}} + 1$. The FlashPatch software stores the flash drive's distance in a data file on the drive itself.

To account for the fact that nodes that are online (at distance 1) may go offline permanently (e.g., due to an equipment failure), we "expire" the distances after one week. That is, if a computer or flash drive has not connected to any other node in the system in one week, the FlashPatch software resets its distance to ∞. To account for the fact that a given flash drive may never reach an online machine, each computer running FlashPatch copies each chunk of its logfiles to three distinct flash drives, when possible.

5. IMPLEMENTATION

This section describes our prototype implementation of a virus scanner which uses FlashPatch to keep its virus definition database



(a) All nodes in the system run version 2 of the software.

(b) The vendor publishes version 3 of the software.

(c) Networked nodes download the update to version 3.

(d) A user plugs a flash drive into a computer running version 3.

(e) A user plugs the same flash drive into a computer running version 2—updating it.

Figure 2: A toy example showing how updates propagate from the vendor (the cloud) to computers running FlashPatch (the circles).

up to date. The core component of our FlashPatch implementation is a daemon that:

- scans inserted flash drives for viruses,
- scans inserted flash drives for virus database updates,
- copies updates to inserted flash drives, and
- periodically tries to download virus database updates over the network.

We implemented the FlashPatch client daemon as a C# application that wraps the open-source ClamAV antivirus software package. Our source code for the client runs to 3,374 lines of code (excluding machine-generated files). Our FlashPatch implementation targeted the Windows XP operating system, since that was the most common OS at our field site.

ClamAV stores the frequently updated portion of its virus definition database in a file called `daily.cvd`. The `cvd` file is an archive containing a version number, the list of virus definitions, other metadata, and a digital signature on the entire archive.

Since the `cvd` file format already incorporates versioning and a signature, the FlashPatch daemon passes around the `daily.cvd` file directly. That is, when a user inserts a flash drive into a machine running FlashPatch, the FlashPatch daemon first checks whether a file called `daily.cvd` exists on the flash drive (in a particular subdirectory allocated for FlashPatch). If the `cvd` file exists, the daemon checks that the file contains a valid signature by the ClamAV maintainers, compares the versions of the local and remote `cvd` files, and transfers the `daily.cvd` file to or from the flash drive, depending on which of the two versions is newer.

Whenever a user inserts a flash drive into a computer running the FlashPatch daemon, the FlashPatch software invokes the ClamAV scanner to search the flash drive's entire filesystem for viruses. When the software encounters a virus, it alerts the user with a dialog box but, to avoid mistakenly erasing valuable user data, the daemon *never* deletes the infected file.

The FlashPatch daemon attempts to upload its log files to our server every three hours and it tests its Internet connectivity by making a short HTTP request every hour. The daemon attempts to download a ClamAV virus definition database update from the network every seven days, to avoid consuming an excessive amount of network bandwidth. After a failed network update attempt, the daemon tries to download the update every hour. The ClamAV database files were roughly 35 MB in size.

6. EVALUATION

This section describes our experimental evaluation of the FlashPatch prototype. We field-tested our software in Hohoe, a town of roughly 50,000 people in Ghana's Volta region, beginning in July 2013. At the time of our study, there were six large Internet cafés in Hohoe's town center. All six of the café owners agreed to have the FlashPatch prototype installed on one or more of the computers in their café, and we also obtained permission to install the software on computers at a graphic design shop and in a secondary-school computer lab.

The goal of our evaluation was to gain some insight into the following questions:

- **Environment.** What is the general state of the computing infrastructure in rural Internet cafés and computing centers? Can we quantify the prevalence of flash drives and the level of computer-to-computer sharing?
- **Viruses.** Which types of computer viruses are present in rural computing environments? Which viruses are the most common? How quickly do viruses spread?
- **FlashPatch.** To what extent can FlashPatch's update mechanism aid in keeping antivirus software up to date in rural computing environments? What are the limitations of FlashPatch in such environments?

6.1 Ethical Considerations

We considered a number of ethical issues prior to the deployment of FlashPatch. We highlight a few of these considerations here to highlight the types of challenges involved in performing this kind of field evaluation.

Informed Consent. Since human subjects were involved in our field evaluation of FlashPatch, we sought and obtained IRB approval for our study methodology and we obtained written consent from the owner of each computer running the software. Internet café owners rely on their computers for income, so we took precautions to prevent the prototype from adversely affecting the host computer's usability.

Before installing the FlashPatch software on a user's computer, we explained the purpose of the software and its limitations (e.g., that it could not detect all viruses). We explained that the software would copy files to any flash drives inserted into the machine.

We also offered the computer owners the option to have a dialog box appear whenever the machine running FlashPatch detected a new flash drive. This dialog box would let the flash drive owner choose whether or not to have the FlashPatch updates copied to the drive on a per-insertion basis. Only one of the café owners (out of six) requested that this feature be enabled.

Resource Use. We limited the amount of space that our software could consume to 20% of the capacity of the USB drive, and the software did not copy anything onto the drive if doing so would have filled up the drive. In practice, the prototype used around 35 MB of space on a flash drive. Before copying data files to the flash drive, the software copied a `README` file to the drive, which explained how to remove the update files from the drive and also how to uninstall the prototype completely from the machine.

Although we experimented with having the FlashPatch daemon scan the user's entire hard disk periodically, the computational load of continuous scanning consumed nearly all of the processing power of the machines and rendered the computer largely unusable

during the time of the scan. Scanning hard disks overnight was not an option because all of the café managers in our field site shut down their computers at night to save power.

Distribution. The frequency with which café managers reformat their computers meant that FlashPatch would have to be reinstalled on machines to allow them to continue participating in our study. In only a few cases, we were able to reinstall the FlashPatch daemon after these reformattings. After we left the field site, however, we were unable to continue reinstallation.

It would have been possible to distribute the FlashPatch prototype to café owners to allow them to reinstall the software in our absence. We did not take that step, however, to avoid the possibility that owners would share copies of our software and that our prototype software could spread to cafés and computing centers outside of our study site. Since we would not be able to obtain informed consent from "second-hand" participants in other regions and we wanted to limit our study to a known set of computers, we decided against distributing the binary to café owners and managers.

Opting Out. After installation, we explained to the computer owner how to uninstall the software and we left the computer owner with our contact information.

It would have been possible to track each flash drive using its unique serial number. This approach, however, would not have given flash drive owners a way to "opt out" of having their drives tracked in our study. Instead, when a user inserts a flash drive into a machine running our FlashPatch daemon, the daemon writes a random unique identifier into a data file on the flash drive. We use this identifier to track each drive as it moves from computer to computer. Since formatting the drive or deleting the data file erases the drive's identifier, we do not perfectly capture the movement of drives.

Privacy. To protect the privacy of participants in our study, our prototype software did not record file names, IP address, MAC addresses, or other information that would have made it easy to identify a computer owner. Even though our prototype's log files contained no particularly sensitive information, we took the extra precaution of having our prototype software encrypt its log files using our research group's public key before the software sent the logs over the Internet or over our network of USB drives.

6.2 Computing Environment

Our experimental deployment of FlashPatch targeted computers running Windows XP, since the majority of machines in Internet cafés in Hohoe were running some version of Windows XP.

Over the course of the deployment, we added a feature to the FlashPatch prototype scanner which allowed us to gather basic statistics about the host machine's operating system performance and hardware capabilities. We have this hardware data from 31 computers at our field site. The remaining computers ran a prior version of the FlashPatch client that did not collect hardware and operating system data.

Of the 31 machines for which we have OS data, 26 were running Windows XP Service Pack 2 (released in 2004), four were running XP Service Pack 3 (released in 2008), and one was running XP Service Pack 1 (released in 2002). Five of the inspected machines had a dual-core CPU and the remaining machines were running single-core processors.

Every machine we encountered had at least one commercial antivirus product installed. These included AVG Internet Security, Avast, Avira, Rising Antivirus, and zbshareware USB Disk Security.

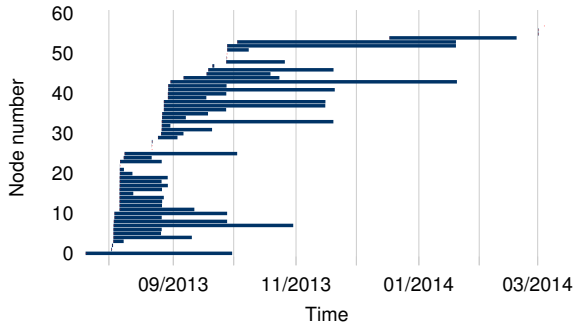


Figure 3: Nodes in our field deployment with their first and last times online (excluding three nodes with faulty clocks).

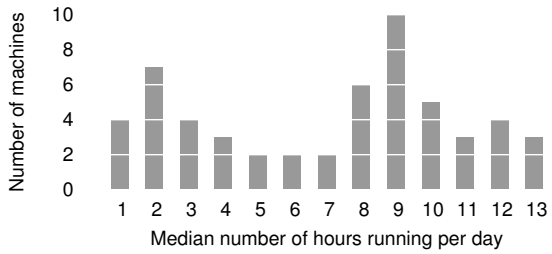


Figure 4: A histogram showing the number of hours per day for which each machine at our site was online.

Figure 3 shows the first and last time that a given computer running our FlashPatch client reported back to our research group's server. The figure excludes three computers with internal clocks that were not set to update over the network and which reported times outside of our study period (e.g., January 1, 2000). The lifetime of a FlashPatch node ranged from less than a single day to over four months, with a median time-to-silence of 21.2 days.

The first reason for a FlashPatch-enabled node to "go silent" was that the computer owner uninstalled the software intentionally. On one occasion, we obtained consent from a café owner to install the software on the café's computers. When we returned to the café the next day to check that the software was functioning correctly, we discovered that the software had been uninstalled from all of the computers in the café. This could be because the café owner had, we suppose, preferred to not participate in our study but was hesitant to directly refuse consent. This behavior may account for the number of nodes with very short recorded lifetimes. These participants' unwillingness to confront or disappoint a foreign researcher echoes findings of prior work [7] and underscores the limitations of the "informed consent" process in developing regions [4].

The second reason for a node to "go silent" was that the computer owner reformatted the machine's hard drive. Since we did not distribute the FlashPatch installation software to computer owners, reformatting effectively removed the computer from our study. If we exclude nodes with a lifetime of less than one day, the median lifetime of a FlashPatch node was 24.0 days. Since, on average, we would have installed the FlashPatch software at the midpoint of period between reformattings, we can extrapolate to conclude that the median time between reformattings is roughly 48 days. This figure corresponds with Bhattacharya and Thies' finding that the bulk of café owners (in India) reformat their computers every 2–3 months [5].

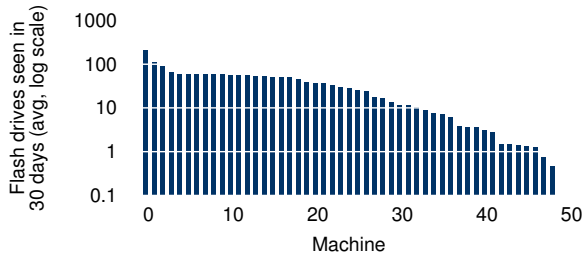


Figure 5: The average number of flash drives inserted into each machine in our study in a 30-day period (14 machines saw no drives at all).

Figure 4 demonstrates, for each computer in our study, the median number of hours it was online per day. The peak in the chart at nine hours accounts for the fact that many computers at our field site were online during business hours only. There is a smaller peak at two hours, which may account for extra computers at a café which are only switched on during "peak" hours—when other customers are using all of the café's primary computers. Computers at graphic design shops, we noticed, are often only booted up when a customer comes to the shop. They remain offline otherwise.

6.2.1 *Flash Drives*

Of the 63 participating nodes in our study, 49 saw at least one flash drive over the course of the study period. Of these, 47 machines saw at least one flash drive per month on average and 33 machines saw at least 10 flash drives per month on average. There were 16 extraordinarily active nodes, which had over 50 flash drives insertions monthly, on average. Figure 5 presents these figures in the form of a bar chart.

Finally, Figure 6 is a representation of the connectivity graph of the FlashPatch network created by the movement of flash drives. The data from this chart is drawn from the 34 machines running a version of the FlashPatch daemon which could track the movement of individual flash drives. (Earlier versions of the software did not have this feature.) In the figure, each vertex represents a computer running FlashPatch. The graph contains an edge (i, j) if the computers represented by vertices i and j saw a common flash drive during the course of the study. The area of each vertex is proportional to the total number of flash drives inserted into the corresponding machine during the course of the study. The width of each edge is proportional to the number of flash drives the machines saw in common.

We tracked flash drives across machines by copying a file with a unique identifier to the drive. When a user deleted this identifying file or reformatted the flash drive, we lost the ability to track the drive. Because of this, Figure 6 represents a lower bound on the connectivity of the machine-to-machine network.

The node adjacency graph (Figure 6) gives a few pieces of information about the network of machines on which FlashPatch runs. First, the machines which are connected to each other in the graph form a single connected component: there is a path from every connected machine to every other connected machine. Second, there are many machines (17 out of 34) that never saw a common flash drive during the duration of our study. Unsurprisingly, the FlashPatch nodes which saw few flash drives are less likely to form a part of the large connected component.

Of the 301 flash drives inserted into the 34 machines running the latest version of the FlashPatch daemon, 17% were inserted into two or more machines and the remaining 83% were only

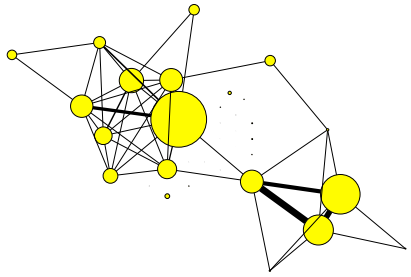


Figure 6: Node adjacency—edges indicate that two nodes saw a common flash drive.

Number of machines seen	1	2	3	4	5	6
Frequency (flash drives)	249	38	9	3	1	1

Table 1: Frequency table showing that 95% of flash drives were inserted into fewer than three machines running the FlashPatch daemon.

ever inserted into a single machine running the FlashPatch daemon (Table 1). The figures in Table 1 provide a *lower bound* on the total number of machines each drive saw during the evaluation period, since they do not include computers which did not have the FlashPatch daemon installed.

6.3 Viruses

Our FlashPatch-enabled virus scanner searched for infected files on every flash drive inserted into machines running our prototype software. By design, the virus scanner did *not* scan the computer's hard drive for viruses, so the figures in this section refer only to viruses found on removable flash drives at our study site.

During the evaluation period, the prototype software scanned a total of 1,242 flash drives for viruses and found 233 drives with at least one infected file. The largest number of infected files on a single drive was 52 and 10% of drives scanned had six or more infected files. As would be expected, the number of viruses detected on a single machine was roughly proportional to the total number of flash drives inserted into that machine (Figure 7).

Our FlashPatch-enabled virus scanner uncovered a surprisingly diverse population of computer viruses at our field site. Many viruses appeared just once, though a few viruses were extremely prevalent. Figure 8 illustrates the virus population encountered in the course of our evaluation. The scanner detected 48 distinct viruses, with the most common (`Win.Worm.Autorun-3638`) appearing 130 times and the least common (`W32.Sality-83` among others) appearing only once. The median number of

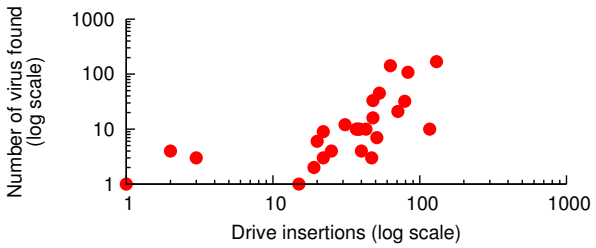


Figure 7: The number of viruses each computer in the study detected is roughly proportional to the number of flash drives inserted into the computer.

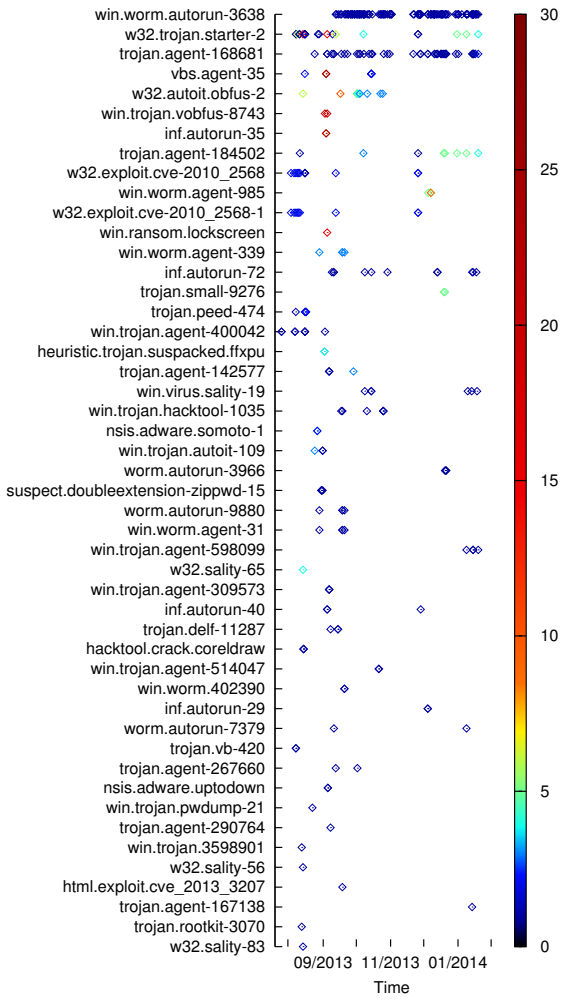


Figure 8: Time at which our FlashPatch-enabled virus scanner detected each virus. The coloring of each point indicates the number of files infected with that virus found during a particular scan.

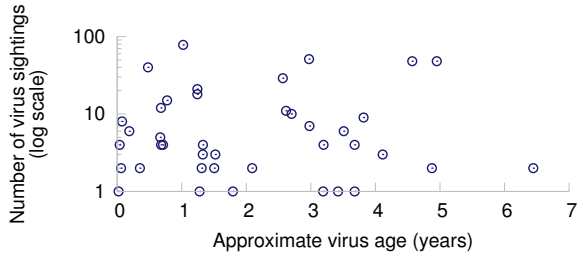


Figure 9: Number of sightings of each virus over the virus age.

sightings for a particular virus was four. In a number of instances, depicted using colors in Figure 8, the same flash drive carried many files infected with the same virus.

Figure 9 depicts the number of times the scanner detected each virus over the "age" of the virus. We define the *age* of the virus as the number of days between (a) the date ClamAV maintainers incorporated a signature for the virus into ClamAV's virus definition database and (b) the date at which our scanner detected the virus on a flash drive. Surprising to us was that our scanner detected a number of "young" viruses during our evaluation run. The scanner detected 63 files infected with viruses that were less than six months old.

Our scanner's detection of so many youthful viruses indicates that flash drives in less connected computing centers can carry viruses that even somewhat out-of-date antivirus software would not catch. If a computer had antivirus software that was last updated in June 2013 (2 months prior to the start of our deployment), the software would have never detected 23% of the infected files found by our FlashPatch-equipped virus scanner. Since all of these viruses traveled via flash drives, they affect computers with or without Internet access.

The prevalence of these young flash-drive-borne viruses indicates that the premise behind FlashPatch is sound: being able to distributed antivirus updates (or other security patches and software updates) to offline machines would provide a tangible computer security benefit.

6.4 FlashPatch Effectiveness

Computers running our FlashPatch-enabled antivirus software periodically tried to download software updates over the network. When a machine successfully downloaded an update, we consider that machine to be "online" until it tried and failed to download an update. Whenever a machine failed to download an update—either because the network connection was down or the download was interrupted—we consider that machine to be "offline" until it successfully downloaded an update over the network. When an offline machine received a software update via a flash drive through FlashPatch, we considered that machine to be "protected by FlashPatch." The intuition behind this definition is that, when an offline machine receives an update via a flash drive, it is getting protection against new viruses that it would not have otherwise have received.

Figure 10 shows, for each machine in our study, how many hours that machine was booted up (in gray) and how many hours that machine was receiving virus protection from FlashPatch (shaded in blue). Even though all of the machines listed had Internet connectivity, many of the machines still gained some antivirus protection benefit from FlashPatch. As far as we can tell, these machines received updates over FlashPatch at times when their Internet connection was down or when some other issue prevented them from reaching the ClamAV update servers (e.g., failure to reach a DNS resolver).

There was one exceptional node that received 1,127 hours of protection from FlashPatch (out of 1,229 hours turned on). Although this machine had a relatively reliable Internet connection, a file permission problem, perhaps caused by a commercial antivirus product, prevented it from installing virus database updates downloaded from the Internet. This machine received almost all of its database updates from other FlashPatch nodes.

This one exceptional machine gives some insight into FlashPatch's effectiveness in protecting offline nodes. Even though this machine was not able to download *any* software updates over the network, it received ClamAV updates **only 11.3 days** (on average) after ClamAV published the updates online. Using FlashPatch meant that this computer's antivirus software was only two weeks out of date, instead of many months out of date.

In judging the efficacy of FlashPatch, a particularly relevant statistic is the fraction of *offline* hours for which FlashPatch provided antivirus protection to a given machine. For each machine in our study, Figure 11 shows this statistic. The figure indicates that a number of machines received updates from FlashPatch during their offline hours, and that FlashPatch afforded additional virus

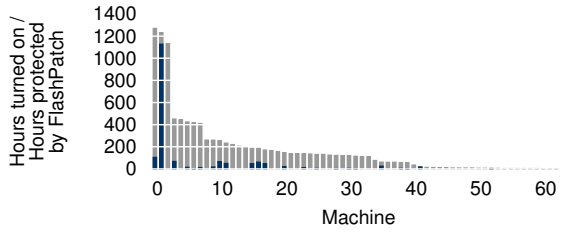


Figure 10: Number of hours each node was turned on / protected by FlashPatch (shaded blue).

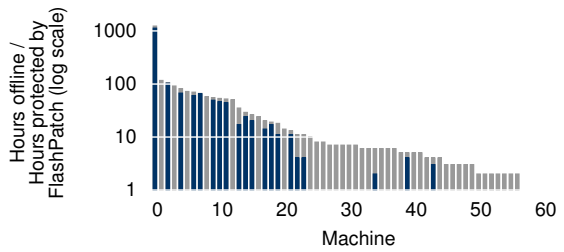


Figure 11: Number of hours each node was offline / protected by FlashPatch (shaded blue).

protection to each of these machines for almost all of their hours offline. Out of the 63 nodes in the study, 15 nodes were protected by FlashPatch for a majority of their time offline.

The number of hours that a node received protection from FlashPatch varied roughly in proportion to the number of flash drives that the node saw during our study period (Figure 12).

Combining Figures 7 and 12 into Figure 13, we find that machines which see more flash drive insertions see more viruses but also see a greater benefit from FlashPatch, in terms of virus protection. We can say then that, roughly, FlashPatch protects most the machines *that need the most protection*—those which see the greatest number of flash-drive-borne viruses.

As Figure 14 depicts, when a machine running our FlashPatch daemon saw a flash drive with a copy of the virus database, roughly 25% of the time the drive had a newer version of the database than did the machine. The remainder of the time, the machine had a newer version of the database.

Finally, Figure 15 demonstrates that the version of the ClamAV database on flash drives at our study site closely tracked the latest version of the ClamAV virus definition database. If a FlashPatch-enabled offline computer at our field site came into contact with a random flash drive containing a version of the ClamAV database, the offline computer would, with high likelihood, receive a recent

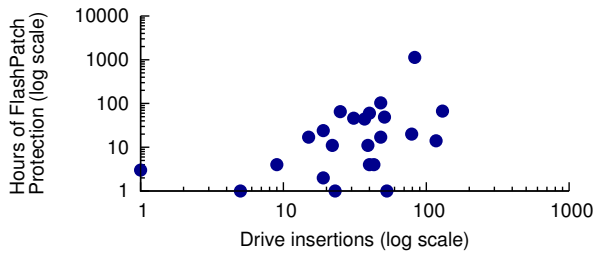


Figure 12: Hours of FlashPatch protection over number of flash drive inserts.

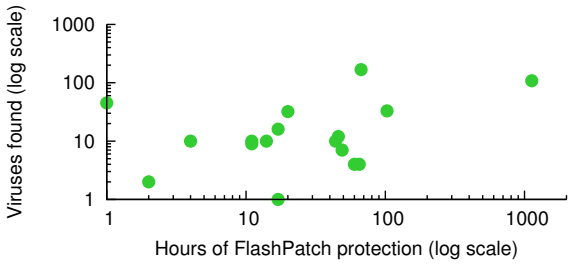


Figure 13: Viruses detected over hours of FlashPatch protection.

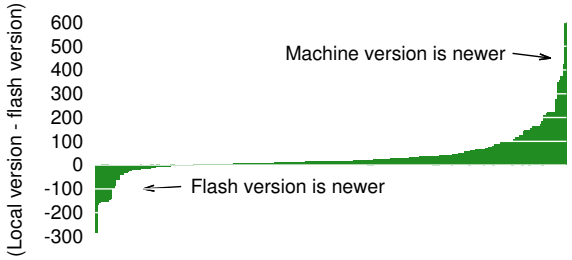


Figure 14: The difference between the virus database version carried on each inserted flash drive and the virus database version of the computer to which the drive was inserted.

copy of the ClamAV database. Since 53% of the flash drives seen at our field site contained some version of the ClamAV database, FlashPatch-enabled offline machines in the proximity of our field site would have a good chance of being updated in this way.

7. DISCUSSION

Our core design principle was that the system should require minimal user interaction to function: rather than trying to modify the technological ecosystem to accommodate our software, designed the software to fit the existing ecosystem. This conviction led us to build our update distribution system on top of the existing network of flash drives—using a common vector for virus transmission as a vector for virus control. Our experimental results validate the core design idea of FlashPatch: that having software updates travel over flash drives is an effective means to improve computer security without imposing *any tangible burdens* on the systems' owners.

To prevent the spread of the FlashPatch software beyond our study site, we did not distribute the FlashPatch installation files to computer owners. As computer owners reformatted their machines, the number of active FlashPatch nodes in the area decreased, and this interfered with the system's ability to spread updates and

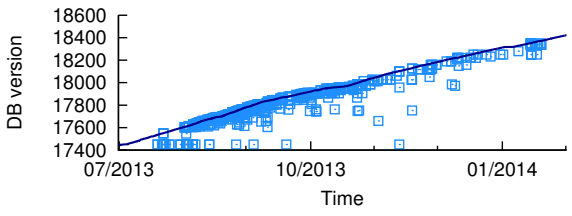


Figure 15: The latest version of the ClamAV database (line) and the version of the ClamAV database seen on inserted flash drives. *Excludes* drives with no database at all.

collect log files. Had we allowed reinstallation, we might have been able to magnify the impact of FlashPatch in the area. The fact that all computer owners in our study installed commercial antivirus software after reformatting their machines suggests that integrating FlashPatch into existing antivirus products would be a viable route to deployment, even in environments in which reformatting is common.

There are a number of potential areas of improvement for our FlashPatch prototype:

First, our client daemon used the open-source ClamAV virus scanner, which was slower and less effective than the common commercial antivirus products (e.g., AVG Internet Security, Rising Antivirus) in our deployment setting. Commercial products are often designed with the peculiarities of Microsoft Windows in mind, so they are able to combat Windows-specific tricks that viruses use to evade detection.

Second, our software lacked the ability to perform *incremental updates*, for reasons relating to the implementation of ClamAV's update system. It is feasible, in principle, to add incremental updates to FlashPatch, and doing so would lead to considerable savings in bandwidth consumption for users of the software.

There are a few other technical challenges that could arise in a widespread deployment of FlashPatch. For example, if there are many different software packages which all require software updates, the combined update files may not all fit on a standard 2 GB USB flash drive. Even storing Windows update files, which can run to hundreds of megabytes, may require too much space for users to tolerate. A FlashPatch-aware operating system could mediate between applications, wanting to share updates over flash drives, and users, wanting to keep as much free space on their flash drives as possible. We leave these challenges open for future work.

8. CONCLUSION

This paper introduced FlashPatch, a system for spreading software updates in developing regions where network connectivity is unreliable or bandwidth is expensive. FlashPatch uses the existing flow of USB flash drives in developing regions as a distribution channel for software updates or security patches. We demonstrated the practical benefits of FlashPatch through a nine-month field evaluation in Ghana. In our field trial, we gathered data on the prevalence and spread of viruses at our study site and found that FlashPatch provided additional antivirus protection to 30% of the machines running our prototype without requiring any change in user behavior. The idea of spreading updates automatically via flash drives may be widely applicable in developing regions, especially when incorporated into existing antivirus tools and operating systems.

Acknowledgements

We would like to thank Anthony Julius Yaw and Kelly McCabe for their help in deploying FlashPatch and we would like to thank Aaron Lynch, Aditya Vashistha, and the anonymous reviewers for their suggestions on how to improve the presentation of this work. This work was partially supported by an NSF Graduate Research Fellowship under Grant No. DGE-114747.

9. REFERENCES

[1] Vishwanath Anantraman, Tarjei Mikkelsen, Reshma Khilnani, Vikram S Kumar, Rao Machiraju, Alex Pentland, and Lucila Ohno-Machado. Handheld computers for rural healthcare, experiences in a large scale implementation. In *Development by Design*, 2002.

[2] Peter Bailis and Ali Ghodsi. Eventual consistency today: limitations, extensions, and beyond. *Communications of the ACM*, 56(5):55–63, 2013.

[3] Yahel Ben-David, Shaddi Hasan, Joyojeet Pal, Matthias Vallentin, Saurabh Panjwani, Philipp Gutheim, Jay Chen, and Eric A. Brewer. Computing security in the developing world: A case for multidisciplinary research. In *5th NSDR*, pages 39–44, 2011.

[4] Solomon R Benatar. Reflections and recommendations on research ethics in developing countries. *Social science & medicine*, 54(7):1131–1141, 2002.

[5] Prasanta Bhattacharya and William Thies. Computer viruses in urban Indian telecenters: Characterizing an unsolved problem. In *5th NSDR*, pages 45–50, 2011.

[6] Eric Brewer, Michael Demmer, Melissa Ho, RJ Honicky, Joyojeet Pal, Madelaine Plauche, and Sonesh Surana. The challenges of technology research for developing regions. *Pervasive Computing, IEEE*, 5(2):15–23, 2006.

[7] Nicola Dell, Vidya Vaidyanathan, Indrani Medhi, Edward Cutrell, and William Thies. Yours is better!: participant response bias in HCI. In *CHI*, pages 1321–1330. ACM, 2012.

[8] Michael J Demmer, Bowei Du, and Eric A Brewer. TierStore: A distributed filesystem for challenged networks in developing regions. In *FAST*, volume 8, pages 1–14, 2008.

[9] Kevin Fall. A delay-tolerant network architecture for challenged internets. In *SIGCOMM*, pages 27–34, 2003.

[10] Shafi Goldwasser, Silvio Micali, and Ronald L Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

[11] Richard Guy, Peter Reiher, Michial Gunter, Wilkie Ma, and Gerald Popek. Rumor: Mobile data access through optimistic peer-to-peer replication. In *Workshop on Mobile Data Access*, 1998.

[12] Sushant Jain, Kevin Fall, and Rabin Patra. Routing in a delay tolerant network. *SIGCOMM*, 34(4), August 2004.

[13] David L. Johnson, Veljko Pejovic, Elizabeth M. Belding, and Gertjan van Stam. Traffic characterization and internet usage in rural Africa. In *20th WWW Companion*, pages 493–502, 2011.

[14] K. W. Matthee, Gregory Mweemba, A. V. Pais, Gertjan Van Stam, and Marijn Rijken. Bringing internet connectivity to rural Zambia using a collaborative approach. In *ICTD*, pages 1–12, 2007.

[15] Michael F. Nowlan and Bryan Ford. Embrace your inner virus. In *"Vision Session" at 9th USENIX OSDI*, 2010.

[16] Michael Paik. Gotta catch'em all! Innoculous: enabling epidemiology of computer viruses in the developing world. In *5th NSDR*, pages 51–56, 2011.

[17] Alex Pentland, Richard Fletcher, and Amir Hasson. DakNet: Rethinking connectivity in developing nations. *Computer*, 37(1):78–83, 2004.

[18] Karin Petersen, Mike J Spreitzer, Douglas B Terry, Marvin M Theimer, and Alan J Demers. Flexible update propagation for weakly consistent replication. In *SOSP*, 1997.

[19] Ian H. Witten, Rarold W. Thimbleby, George Coulouris, and Saul Greenberg. Liveware: A new approach to sharing data in social networks. *International journal of man-machine studies*, 34(3):337–348, 1991.

[20] World Bank. GDP per capita (current US$). Retrieved 30 July 2014.