LENS: Location-based Emergency Notification Service

Jian Wang, Tim Yardley, Himanshu Khurana and Liying Wang
University of Illinois at Urbana-Champaign
{jwang32, yardley, hkhurana}@illinois.edu, jessa.wang.liying@gmail.com

ABSTRACT

University campuses and municipalities are currently spending large sums of money to acquire systems that allow dissemination of information in emergency situations. The majority of these are mass notification systems that first register multiple contacts for community residents (email, phone, pager, etc.) and then deliver information to those residents at the push of a button to leave a message. Motivated by the limitations of such approaches, in this work we explore the use of existing metropolitan network infrastructures to design a new Location-Based Emergency Notification Service (LENS). LENS selectively redirecting residents to safety information using existing communication channels (e.g., Web browsing over HTTP). LENS eliminates the need for registration, provides minimal interruption to users and involves a low-cost setup. We prototype LENS using off-the-shelf components and demonstrate efficiency and scalability for a 60,000 user campus environment.

Keywords

Emergency notification, network, redirection, location-based computing, web browser.

INTRODUCTION

Physical security and safety of residents of a community such as a University campus or a municipality is paramount to administrators of the community. Consequently, University campuses and municipalities are currently spending large sums of money to acquire systems that allow dissemination of information in emergency situations. The majority of these are mass notification systems that first register multiple contacts for community residents (email, phone, pager, etc.) and then deliver information to those residents at the push of a button to leave a message. Other notification technologies being deployed today include sirens and indoor/outdoor speakers. Such emergency notification systems are intended for use in a variety of situations such as health alerts (e.g., contagious diseases, water contamination), weather alerts and crime alerts (e.g., armed intruder in the vicinity) (Virginia Tech Report, 2007; Wei and Pearson, 2006). While these systems may have already been instrumental in saving lives on campuses, considerable research is needed to first understand the desirable properties of emergency notification systems and then developing ones that satisfy these properties (Gow, Townsend, McGee and Anderson, 2008). Furthermore, a range of systems will likely be needed, as different properties are of interest in different communities and in different emergencies.

In this work we propose a novel computer network based emergency notification technology that has distinct properties from previously proposed techniques. We believe that these distinct properties will motivate the use of our approach as a complementary tool to systems already being deployed. This work is motivated by the observation that the current mass notification approaches are significantly limited because of their registration requirements, high costs, limited performance and intrusive nature. It is not always possible to populate a registry of residents and maintain it, given the privacy issues associated with personal information and that residents can change contact information as well as enter and leave the community. In addition, they fail to reach out to nonregistered users such as visitors and those that opt-out of the registration process. The supporting software and hardware systems are typically very costly to setup and maintain. This is especially true when high performance is demanded such as being able to reach an entire campus community within minutes. Furthermore, they are highly intrusive as users are interrupted from their tasks multiple times (on their office phones, cellular phones, etc.) leaving a strong and lasting impression of the warning (Bambenek and Klus, 2008). While such intrusiveness can be very useful in extreme emergencies, such as an active shooter in the vicinity, they are vulnerable to significant consequences in case of false positive or negatives.

Reviewing Statement: This paper has been fully double blind peer reviewed.

In contrast, we propose a notification system that circumvents these other limitations. The idea is to use metropolitan network infrastructures, such as those that exist on University campuses or in municipalities, for selectively redirecting residents to safety information using communication channels that they engage with on a regular basis, namely, Web, telephone, and mobile devices. In an emergency or when a notification is needed, whenever a resident uses an existing communication channel (e.g., visits a Website, makes a call from the office or from their cellular phone, sends a text message), they are automatically redirected to information about the emergency. Users are already used to such redirections, for example, when authenticating with passwords to access the Internet and listening to customer services messages prior to dialing a phone number. In this work we focus exclusively on Web-based redirection and argue that this approach can also be extended for other communication technologies such as phone systems.

Based on this approach we developed a Location-based Emergency Notification Service (LENS) with the following salient properties. First, it is location based as opposed to registry based so it reaches out to all residents (including visitors) connected via a particular network (wired or wireless) regardless of whether they have registered with a community-wide database. Our approach offers a range of granularity for the location depending on the set of intended recipients of a given emergency; in particular, location is decided based on IP addresses so filtering subnets offer this flexible capability. Second, it offers high performance in that it provides access to emergency information within milliseconds (as opposed to current approaches that take several minutes). In certain emergencies saving seconds can be crucial. Furthermore, the costs are very low as enhancements to a single exit router can be sufficient to provide service to an entire campus at any desired level of location granularity. Third, such redirections are significantly less intrusive than an interrupt-based system that calls the resident on their office phone, home phone, cellular phone, sends an e-mail, and sets off their pager.

In order to be practical LENS uses off-the-shelf tools (e.g., routers, open-source redirectors, and standard core functionality), provides a simple and secure interface, resides on a non-critical network path for fault tolerance, and most importantly requires no additional software or service on the client side. We have implemented a prototype and validated the effectiveness and performance of the solution using real data gathered from the University of Illinois at Urbana and Champaign (UIUC) network. In particular, we demonstrate that the necessary components can scale to meet the demands of a 60,000 user campus environment.

The rest of this paper is organized as follows. In Section 2 we discuss requirements and present our approach. In Section 3 we discuss the design of LENS and in Section 4 its implementation. In Section 5 we present an experimental analysis. In Section 6 we discuss related work and conclude in Section 7.

REQUIREMENTS AND APPROACH

The idea that in-network web-based redirection of community residents to emergency notification holds potential, already addresses several limitations of existing schemes and also provides design constraints. It eliminates the need for registration and builds on familiar mental models of web browsing. In terms of design, it requires the ability to capture and redirect web traffic during an emergency for users affected by the emergency. IT networks in university campuses and municipalities are typically decentralized, heterogeneous and support a large population. For example, users own desktops and laptops with a variety of operating systems and often administer their own machines, and system administrators manage distinct parts of campus networks using disparate networking technologies. In such environments effective redirection technologies must provide the following desirable features:

Flexible location granularity. Different emergencies will require informing different users. For example, water-boil alerts need to be sent to dorms in the affected parts of a campus while information on contagious diseases (such as the recent spread of the meningococcal disease) or armed intruder alerts may need to be sent to the entire campus community. Redirection technologies must support a range of location granularity.

Efficiency and scalability. Emergencies such as severe weather or armed intruders require notifications to be disseminated within seconds or minutes. Furthermore, the notifications may need to be disseminated to tens of thousands of community residents. Redirection technologies must provide adequate efficiency and scalability to match these needs.

Minimal infrastructure changes. Given the decentralized nature of campuses, requiring significant changes (e.g., modifying network paths, DNS systems) will incur prohibitive costs.

No client-side changes. One seemingly easy way to disseminate emergency messages is to require that all users install notification software that will produce pop-up messages on demand. However, given the way campus networks and machines are managed, ensuring that this software is installed and always online for all users will again incur high costs. In general, an approach that requires *no* changes to client-side machines has highest chance of success.

Minimal interruption. The redirection system should ensure that users view the emergency notification but should not disrupt networking operations. This is because networked communication tools need to be available during emergencies as they are fast becoming an important response and recovery component (Jaeger, Schnederman, Flischmann, Preece, Qu and Wu, 2007; Palen and Liu, 2007; Pau and Simonsen, 2008). Similarly, the redirection system should be failsafe in that its failure should not result in loss of network service to the users.

In trying to achieve these desirable properties several existing redirection techniques need to be considered. For example, IP-based redirection, URL rewriting and DNS-based redirection are all commonly used to provide load balancing in Content Delivery Networks (CDNs) (Vakali and Pallis, 2003). Captive portals use firewalls and HTTP redirection to provide authentication in wireless networks. Another related technique is the use of web portals (Staab, Angele, Decker, Erdmann, Hotho, Maedche, Schnurr, Studer and Sure, 2000) that provide aggregated content delivery to a community of users; e.g., with inline browser frames.

Given the requirements for emergency notifications we need a technique that can intercept traffic for a large community and then redirect selectively. This rules out techniques like URL re-writing because they are intended for use at destination web servers. We now discuss three potential techniques, namely, portals, DNS-based redirection and IP-based redirection. 1) Portals such as those providing inline frames and aggregated content can potentially be effective for emergency notification. For example, the outer frame can be used to send messages. However, use of portals for an entire campus requires the presence of a highly provisioned proxy service (to ensure adequate scalability) that manages content delivery for all users. Such proxy services are very rare in campuses and municipalities and developing them would require significant infrastructure changes and would be very costly. 2) DNS-based redirection techniques can also potentially work by dynamically changing DNS servers in a campus at the time of the emergency to redirect all requests to a server (or set of servers) that furnish the emergency message. However, the ability to change all DNS servers and clear caches on cache servers and all clients (something generally out of scope for network control) in a centralized manner would require significant infrastructure changes at high costs or not even be possible. Other techniques are possible such as dynamically loading zone files specific to the clients to be redirected with extremely low time-to-live records, but these suffer from the same caching issues just on a set scale. Furthermore, DNS redirection would potentially result in disruption of *all* network traffic (e.g., email, web services) that is undesirable and could hamper response and recovery operations. 3) IP-based techniques are the most promising and adopted by LENS. They work by selectively redirecting only HTTP requests at routers to deliver emergency messages. With proper design they can address all of the desirable properties. The one drawback is that because redirection takes place at the IP layer the emergency message content does not match the URL, however, we argue that through regular use and appropriate education a campus community can accommodate this discrepancy.

LENS DESIGN: CHALLENGES AND APPROACH

LENS uses four key components in its design of an IP-based redirection system. These components include the network router, a proxy server, a redirection application and a database server. The design is based on a straightforward approach: divert and redirect. The first step is to divert traffic from the router. This is achieved by the router forwarding selected HTTP requests to the proxy server. After the proxy server captures the diverted HTTP request, it forwards the requests to the redirection application. The redirection application validates the request and redirects the user to the emergency page, and then once the user acknowledges having read the page (by clicking a button) the redirection application fetches the originally requested page for the user. The proxy helps with delivery of the fetched page back to the user. Figure 1 below demonstrates the design of LENS.

In this figure, an HTTP request sent by the client is forwarded to the proxy server through path 1. The proxy server forwards the request to the redirection application in path 2. A validation process decision is then made at the redirection application and the emergency message (a web page) is dispatched as necessary by the proxy server through path 4 and 5. Once an acknowledgement from client is received by the proxy server, it fetches the original destination web page from Internet through path 6 and 7, and sends it back to the client.

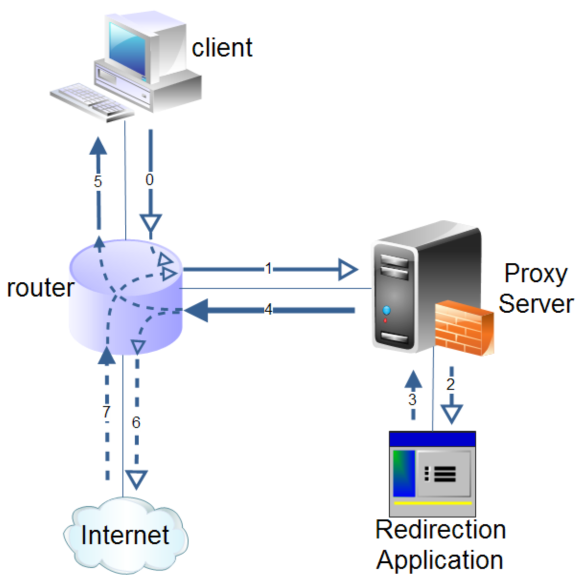

Figure 1: LENS Components and Architecture

In the following Table we summarize how LENS meets the requirements identified above. We then discuss several challenges that emerge with the approach and how we address those challenges.

Desired Property	LENS Design
Flexible location granularity	*Router-based granularity*. LENS can be instantiated at any router to achieve varying granularity; e.g., wing, floor, building and campus.
Efficiency and scalability	*Router-based efficiency and scalability*. By focusing on routers LENS leverages the inherent efficiency and scalability of the router. Our experiments demonstrate this in practice.
Minimal infrastructure changes	*Router-based infrastructure enhancements*. By using commercial routers that provide easy access for redirection and filtering requests and we keep infrastructure changes to a minimum.
No client-side changes	*Browser-based notification.* LENS redirection works on all standard browsers out-of-the-box so no client-side changes are needed.
Minimal interruption	*Familiar Browser-based Interruptions.* LENS uses interruption semantics already familiar to users such as authentication in wireless networks.

Table 1: LENS Design Approach

In designing LENS based on the approach described above we identified three new challenges that had to be addressed.

Exactly once delivery semantics: A simple approach for IP-based redirection would redirect users to the emergency information page over and over again as long as the redirection system is active. This is considered as interruption of network accessibility, thereby would be unacceptable in practice. To address this we extended the basic design to provide exactly-once delivery of the emergency information to each user. Essentially, we maintain state information about each user in the database where each user is identified by the IP address of the request. Every time a user makes an HTTP request the database is queried using the IP address of the request (captured by the proxy server). If a record is found then the redirection process for that request is abandoned and the original requested page is served. If not, the redirection process is executed for the request and a new record is created. The solution is fairly practical but not perfect because if a user makes requests from two different computers (i.e., makes requests originating from multiple IP addresses) that she would be directed twice (once for each originating IP address). However, we argue that this approach is still reasonable. This extended design is discussed in the next section.

User specific redirection: In general, IP-based redirection would lead to redirection of all HTTP requests in the system. This would include redirection of many applications that communicate using HTTP and likely lead to process and application failures. This would be undesirable and the right approach is to only redirect interactive user requests. To achieve this goal we leverage the HTTP header field “User-agent”. We do this based on the simple observation that real users use browsers that are easily identifiable while applications have their own identifiers. This field allows for easy identification of all standard browsers (e.g., IE, Firefox, Chrome, Safari). We then use filtering capabilities at routers to exclude all other requests from the redirection process.

Redirection of users behind a NAT: Our approach of identifying users for selective redirection requires that each user can be uniquely identified by the IP address in the requests. This requirement is relatively easy to achieve for wired networks as users machine typically have publicly addressable unique IP addresses. This assumption is likely to strengthen as system move to IPv6 from IPv4. However, often wireless networks (e.g., 802.11 networks) and even some wired networks use NAT whereby a single IP address identifies all users outside the NAT. This limits our ability to provide exactly-once delivery to each user. To address this challenge we propose a mitigating approach that the LENS redirection process should be implemented in association with the NAT where unique IP addresses are visible. This is relatively easy to do for wireless networks that have the ability to force users for authentication purposes. A better and more general solution remains an open challenge.

LENS DESIGN: SECURITY

With an emergency notification system, the security of the system itself is of paramount importance. LENS addresses security by leveraging the architecture design and focusing on the realistic threats and attack models. First, there are two primary design principles for security that need to be emphasized. The LENS notification system should be treated as a critical asset both during active use and when sitting idle. As such, physical access to the machines should be protected and the machines should be actively monitored at all times from a system stability and network monitoring point of view. Secondly, access control mechanisms should be utilized such that any reconfiguration of these machines requires direct physical access to prevent the possibility of tampering. With these primary design principles in place, there are two primary attack vectors for LENS. We will detail each of them and discuss our approach to address them.

One vector of attack is through the administration interface whereby an adversary might want to use the interface to either institute an emergency notification that isn’t approved by administration or to prevent authorized notifications from being instituted. One reason for the former attack would be to cause panic while one reason for the latter would be to cause a denial of service. To combat this, we propose a separation between the application administration interface used to manage notifications, and the backend activation of that notification. The backend process to activate the notification would require a human with appropriate access (e.g., network administrator) to approve the notification and enable the redirection on the router (or set of routers). The activation of the redirection on the router would be protected by strong access controls and authentication measures that would essentially prevent the adversary from being able to launch these attacks without first compromising credentials of authorized personnel that are typically well protected. Further, this activation could be restricted to require console access to the router itself, requiring an attacker to gain physical access to the networking equipment in addition to the credentials.

A second attack vector would be from potential malicious traffic originating from the campus or external networks. There are two types of attacks here; an attack on the infrastructure itself and an attack via payload content or URL. With the LENS architecture, the only traffic that is allowed to the backend infrastructure is port 80 HTTP traffic. This limits the attack space to HTTP traffic, so an attack on the infrastructure would likely involve an abnormal amount of requests to overwhelm resources and cause a denial-of-service while an attack via payload/URL would likely involve exploiting application layer vulnerabilities (for example, SQL injection).

The infrastructure attack can be adequately addressed via network monitoring and other standard network protection mechanisms (such as rate limiting) that are typically deployed on large-scale networks such as campus networks targeted by LENS. This leaves the remaining attack of payload/URL as the concern. LENS addresses this by never sending the URL or payload content to any process and all decisions being made purely off the headers. Of the headers, the only ones inspected are IP and UserAgent, neither of which is utilized by any process other than the redirection itself. An interesting case here would involve an attack to the URL processor of the proxy system (SQUID) itself. In order to mitigate the impact, the system should be actively monitored. The impact of such an attack could also be minimized by the packet and content inspection provided by the router for redirection, leveraging those features to block potentially malformed payloads. This type of attack would only be feasible during an active redirection period, but is still an important risk to consider.

We argue that the proposed design principles and details of implementation appropriately address the threat model and major attack vectors for the notification system. In the case that the notification system is being attacked or detected as compromised, the systems should be immediately disabled and other methods of notification should be leveraged to support emergency notification.

LENS IMPLEMENTATION

Based on discussion in Section 3, our implemented system is showed in the following diagram.

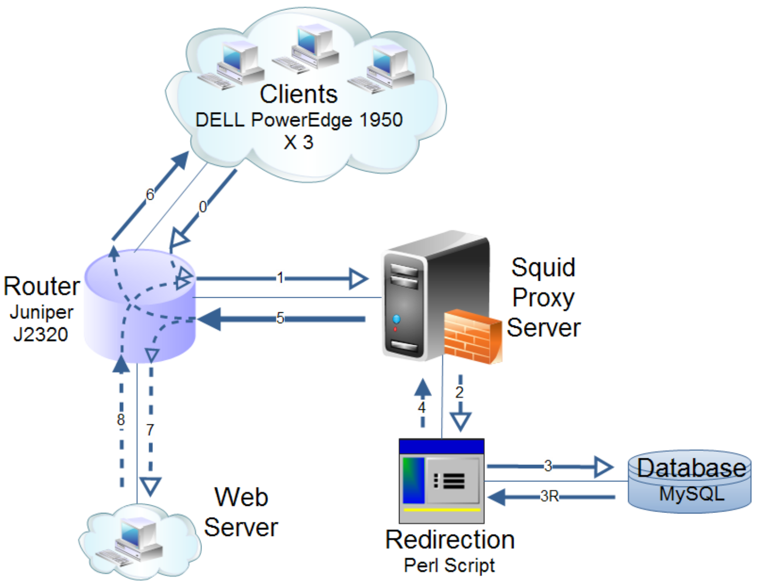

Figure 2: LENS Implementation

LENS was prototyped leveraging open-source tools and developed to initially run on a Linux based system to meet the design goals stated in Section 3. Here we detail the breakdown of implementation of each of the stated components.

Experiment Machines:
Dell PowerEdge 1950 servers with the following specifications were selected to run various components of our experimental system: two 1.6GHZ Dual-core processors; bus Speed: 1066MHZ; L2 cache size: 4MB; system memory: 8GB; system memory speed: 667MHZ; operating system: Fedora core8/Ubuntu8.04. They were utilized for simulated client requests, the proxy server, external web server, and application/database server.

Network Router:
In selecting a commercial router for the project we explored two choices, Cisco and Juniper. With either selection, appropriate hardware is able to provide the necessary diversion and redirection of HTTP requests. The only variant is the method by which they are configured to do so. With Cisco routers a solution called Web Cache Communication Protocol (WCCP) can be configured to handle redirection to the proxy server. With Juniper routers, application based identification rules (e.g. Filter Based Forwarding-FBF) can be used to facilitate the same redirection. Both methods have been deployed successfully on our test bed. In this paper, we will only present the test result from Juniper router as it represents the more general way of router redirection. We selected a Juniper Model J2320 router.

Proxy Server:
For handling the redirected HTTP requests, we employ the commonly used Squid[1] server. Squid is a software web cache engine that primarily serves HTTP traffic. In addition, Squid has several advanced features that are desirable to LENS, those features include high efficiency, multi-threaded and customizable redirectors, external

[1] http://www.squid-cache.org

ACLs, and interception caching. Squid version 3.0 was selected for our experiment and configured accordingly to facilitate the rest of the system design.

Redirection Application:
The redirection application was written in Perl, and configured to have the necessary identification information (client IP addresses) sent to it via standard mechanisms in Squid. This application is the core logic of the redirection, operating under a single-hit semantic so that users only have to see and acknowledge the alert one time. As such, the code was structured to be as fast as possible for anyone that should not be redirected, and only making queries out to the database when necessary.

Database Server:
The database schema for this experiment was kept simple for optimization reasons as well as the simple nature of what it needs to perform. More preciously, since client IP information and information regarding whether that IP has acknowledged the alert message are necessary for redirection application, several megabytes of query cache memory can serve thousands of users in the community. As such, the database acts more as an event based persistent store than anything else. MySQL was chosen for the implementation as the DB requirements were straightforward and it generally provides proven scalability at the level we needed.

External Web Server:
An external web server was utilized to provide a destination location when a user acknowledges an alert, or has already been redirected and is therefore being sent to the requested location. To work towards this being eliminated as a bottleneck in the experiment, *nginx* was chosen to provide the services of the external web server. A simple HTML page (file size 25KB) was utilized as the content.

Client Machines:
A simple Perl script is executed on client machines and sends HTTP requests during the test. Three identical machines are deployed to run the same script simultaneously in order to insure that all requests are processed concurrently.

Scaling and other considerations:
In order to effectively scale to the desired magnitude, components were carefully selected and coupled together in a manner that attempted to optimize efficiency. In doing so, optimizations were made where necessary for the various applications installed. This included Linux sysctl.conf tweaks, leveraging advanced features and realistic settings in squid.conf, tweaks to the nginx configuration for external web serving, and domain focused design of the redirector application.

In the experiment itself, the components were configured as shown in Figure 2. Three client machines were set up to make multiple requests from varying IP addresses through the router, which then redirected them to the proxy server based on current configuration. If redirected, the clients were handed off to the proxy server which then sent information to the redirection application to process the request. The redirection application determined if the client had been redirected before: if not, sent them the appropriate alert messages by returning a crafted HTML page and then updated the database accordingly; if they had already been redirected and acknowledged the alert, they were sent back the originally requested page which was served by the external web server.

EXPERIMENTS AND RESULTS

In order for LENS to be practical it must achieve the following performance goals. First, the instantiation time of the redirection process in the presence of an emergency must be very small. Second, the latency observed by each user in the community must be reasonable. Third and most importantly the overall throughput must scale to support the entire user community. Fourth, LENS must provide selective dissemination of emergency information to users only. Fifth, LENS must have user acceptance and be effective in case of emergency. In this section we address the first four performance requirements and leave a comprehensive usability study to future work. In addition, to extensive experimentation on our prototype implementation we use two data sets from University of Illinois at Urbana and Champaign (UIUC), one from the National Center of Supercomputing Application (NCSA) and the other from the campus network to validate our results. First, we obtained HTTP request rate information for the campus exit router for a period of one week to help established scalability goals. Second, we obtained HTTP request information for a period of several hours to analyze HTTP headers used in practice.

Instantiation. LENS can be activated by simply activating the filter based forwarding rules at the router. This involves logging into the router to change policies or executing a pre-defined script. This can be done very easily and quickly as soon as an emergency is determined and notification is needed.

Latency and throughput. The main concern for the feasibility of LENS is the latency our approach will introduce to existing network and its overall throughput. We measure the "event latency", that is, the time from when the first diverted request leaves the router and ends when the last request gets back to router, namely, the time interval between path 1 and path 5 in Figure 2 *for all requests in the event*. Our experiments focus on measurement of that event latency, but the diversion delay of router is not taken into account because that delay is typically orders of magnitude smaller as it is a hardware operation.

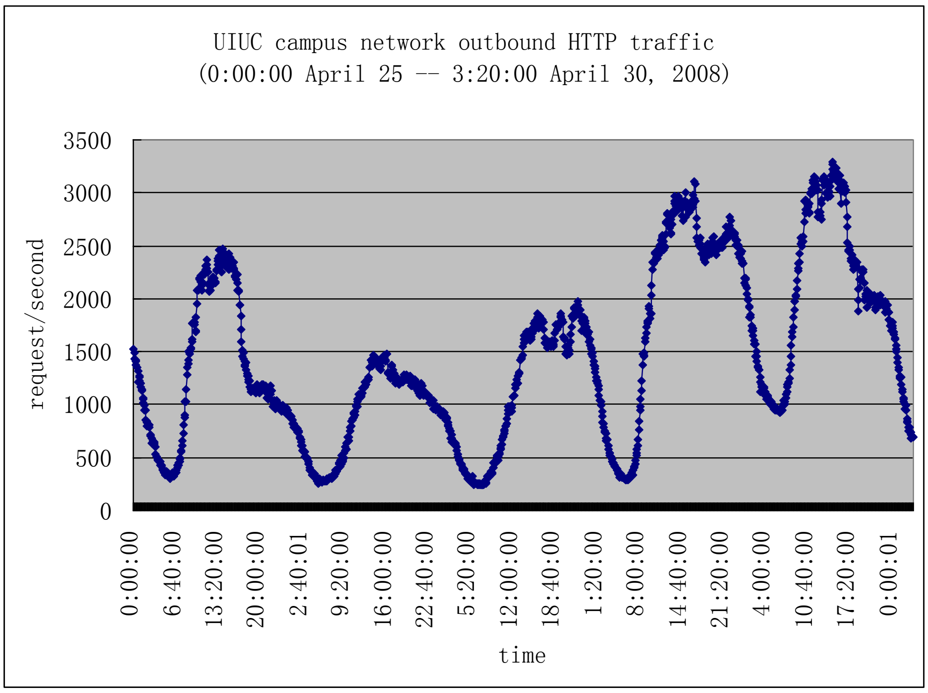

Figure 3: UIUC campus network's outbound HTTP traffic request rate distribution

Key to these experiments is defining the number of requests in an event that LENS should be able to handle. In other words given a certain period of time (for example 5 minutes), the number of requests LENS is able to handle for a community affected by the emergency event. Such a community can be a small one (e.g. a building) or a large one (e.g. a municipality). The goal of our experiment is to show that the hardware described in Section 4 can handle a mid-sized community and LENS can be easily scaled up to cope with large sized community. To determine the desired request rate of a mid-sized community, we study network traffic flow of UIUC campus network with about 60,000 residents. On April 25 2008, we gathered one-week worth of HTTP traffic that went through the exit router of the whole campus. The raw data includes time (every 5 minutes), number of HTTP connections, and data size in bytes. From the raw data, we derived the diagram of request rate distribution, max-min-average request rate for each day as well as for all days, and average data size for each request. All results are shown in Figure 3, Table 1 and Table 2.

From these figure and tables, we see that the average requests rate is about 1,500 request/second and average data size for each request is about 25KB. As discussed below some of these requests will be applications as opposed to users and, therefore, will be ignored by LENS. We set the goals of the emergency message being delivered within 4 to 5 minutes to the entire community that is expected to be online and sending HTTP requests; that is, the number of total requests being handled during the event should be about 360,000 to 450,000. In our experiment, we choose 360,000 as total requests sent by clients and observed the event delay to ensure LENS does not create undue delays to the network.

Since whether a destination web page is cache-hit for Squid proxy server affects the performance of Squid itself, we create two identical HTML pages (file size 25KB) on external web server and configure our test bed such that one of the pages is always a cache-hit in the Squid server and the other is always a cache-miss. The likelihood of a cache-hit is predefined as 35%, which is a reasonable number for many real world network communities (Ari 2004).

UIUC campus network's outbound HTTP traffic, April 25 -- April 30, 2008					
Date	**Max request rate(req/sec)**		**Min request rate(req/sec)**		**Avg request rate(req/sec)**
April, 25, Fri	2476		305		1331
April, 26, Sat	1479		253		903
April, 27, Sun	1974		238		1089
April, 28, Mon	3107		285		1889
April, 29, Tue	3289		922		2118
April, 30, Wed	1777		685		1153
Overall	**Max_{Max}=*3289***	**Avg_{Max}=*2350***	**Min_{Min}=*253***	**Avg_{Min}=*448***	**Avg_{Avg}=*1413***

Table 2: Statistical Results for UIUC Campus Traffic Flow Data

Data size (KB per request)	Max	Min	Average
	39.94KB	7.134KB	23.071KB

Table 3: Data Size Statistics for UIUC Campus Traffic Flow Data[2]

The latency for each request to go through the Squid server and redirection application is logged to a file on the proxy server. Statistical results are shown in table 4 below:

Total requests send	360,000
Cache-hit count	126,000
Cache-hit average request latency (in millisecond)	90
Cache-miss count	234,000
Cache-miss average request latency (in millisecond)	150
Event latency (in seconds)	320
Requests processed per second	1125

Table 4: Experimental Results

During the experiment, CPU usage of the router becomes almost 100%, which points to it as the bottleneck of our test bed. In contrast, the proxy server, on which Squid, the Redirection Application and Database are operating, maintains a CPU usage of 40%. Further, we believe that this router hardware limitation also leads to outliers. For example, we observe in the log file that more than 90 percent of the cache-hit requests are processed in less then 1 millisecond and others' processing time varies from several milliseconds to 20 seconds; and for the cache-miss requests, more than 90 percent of them are processed in 8 milliseconds and others can take up to 100 seconds. Those outliers decrease the throughput of our experiment, and we believe that the throughput will be significantly increased if experiments can utilize a higher performance router.

Selective dissemination. By leveraging User-agent filtering, we specifically minimize the impact to applications and only redirect sessions that are interactive with the user. To verify this we studied several hours of HTTP request data from the dataset we obtained from NCSA for HTTP traffic recorded for a 12 hour period on September 17, 2008. In the data set we found 22 unique User-agents with clear use of standardized nomenclature to identify browsers and other applications. This dataset of users actively browsing highly supports this practice of filtering based on User-agent. However, we note that applications that mimic the User-agent field of browsers would still get redirected using this method, but this is an acceptable tradeoff as these applications are practicing atypical behavior. Further, various tricks can be played to make this false-positive hit less likely by leveraging inspection of the URL or other methods of identification. Combining the User-agent filtering with URL inspection leads to very few false positives for interactive sessions.

Analysis. We believe our approach can be applied to both small communities and large communities. First, milliseconds level delay indicates that the processing capacity of our approach can easily scale up to thousands of requests per second to match real-world needs. Second, since CPU usage of proxy server remains at a low level in our experiment, it's possible to add more instances of the redirection mechanism to the same machine since each of them works independently. Assuming the router hardware is not a bottleneck of the system, we can even add multiple proxy servers and attach them to the same router while each server manages one subset of

[2] Each data piece is derived from the average data size of every 5 minutes, unit KB/request.

the community. With appropriate load balancing configuration, multiple instance of the redirection mechanism will be able to handle an even higher request rate.

CONCLUSIONS

In this work we explore the use of existing metropolitan network infrastructures to design a new Location-Based Emergency Notification Service (LENS). LENS selectively redirects residents to safety information using existing communication channels (e.g., Web browsing over HTTP) and can serve as a complementary tool for use along with other emergency notification systems. LENS eliminates the need for registration, provides minimal interruption to users and involves a low-cost setup. We prototype LENS using off-the-shelf components and conduct performance experiments to demonstrate feasibility. Future work in this area involves experimentation with higher performance routers on the technical side, and determining adoption strategies and undertaking usability studies on the policy/usability side. For the latter set of problems interesting questions include the type of situations (shooter-in-the-vicinity, tornado, water boil alerts) that are most suited for LENS, the response by users to LENS notifications, and comparing those responses with alternative approaches.

ACKNOWLEDGEMENTS

We would like to thank John Washington and Aashish Sharma for their help in getting data from the University of Illinois networks. We would like to thank Rakesh Bobba, Jon Marks, Nathan VanHoudnos, Todd Short and Von Welch for comments and discussions. This material is based upon work supported by the Office of Naval Research under award numbers N00014-06-1108 and N00014-07-1-1173, and by the National Science Foundation under grant numbers CNS 07-16626 and CNS-0524695.

REFERENCES

1. Ari, I. (2004) Design and Management of Globally Distributed Network Caches, *PhD Dissertation*, University of California Santa Cruz.
2. Bambenek, J. and Klus, A. (2008) Do Emergency Text Messaging Systems Put Students in More Danger? *EDUCAUSE Quarterly*, vol. 31, no. 3, July–September 2008.
3. Gow, G. A., Townsend, D., McGee, T., and Anderson, P. (2008) Communication technology and campus safety: Critical sociotechnical concerns for emergency messaging at Canadian universities, *IEEE ISTAS International Symposium on Technology and Society*, June 2008, pp.1-5.
4. Jaeger, P. T., Schneiderman, B., Fleischmann, K. R., Preece, J., Qu, Y. and Wu, F.P. (2007) Community response grids: E-government, social networks, and effective emergency response, *Telecommunications Policy*, 31, 2007, 592-604.
5. Palen, L. and Liu, S.B. (2007) Citizen communications in crisis: anticipating a future of ICT-supported public participation, in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, San Jose, California, USA, April 28 - May 03, 2007.
6. Pau, L-F. and Simonsen, P. (2008) Emergency Messaging to General Public via Public Wireless Networks, in *Proceedings of the International community on information Systems for Crisis Response And Management* (ISCRAM), Washington DC, May 2008.
7. Staab, S., Angele, J., Decker, S., Erdmann, M., Hotho, A., Maedche, A., Schnurr, H. -P., Studer, R. and Sure, Y. (2000), Semantic community Web portals, *Computer Networks*, Volume 33, Issues 1-6, June 2000, Pages 473-491.
8. Vakali, A. and Pallis, G. (2003) Content Delivery Networks: Status and Trends, *IEEE Internet Computing*, vol. 07, no 6, pp. 68-74, Nov/Dec, 2003.
9. Virginia Tech Review Panel. (2007, August). Mass Shootings at Virginia Tech April 16, 2007: Report of the Review Panel. Retrieved Oct. 1, 2007. Available at http://www.vtreviewpanel.org/report/index.htm.
10. Wei, L. and Pearson, J. (2006) Final Report of Emergency Communication Systems for Florida University and Community College Campuses, Technical Report, University of Central Florida, 2006. Available at http://ec.creol.ucf.edu/.