



# The Next Wave

*The National Security Agency's Review of Emerging Technologies*

*Vol 18 No 3 • 2010*

## **Mapping Out Faster, Safer Networks**

**Clumps, Hoops, and Bubbles**

**How Akamai Maps the Net**

**Compressed Sensing and  
Network Monitoring**

**Revealing Social Networks  
of Spammers**

**Challenges in Internet  
Geolocation**



## Letter from the Editor

This issue of *The Next Wave* is largely derived from talks given at the 2008 and 2009 Network Mapping and Measurement Conferences (NMMCs), held at the Laboratory for Telecommunications Sciences (LTS) in College Park, Maryland. These conferences evolved from the NetTomo workshops I-IV, which grew out of research on network tomography sponsored by the Information Technology Industry Council (ITIC). By 2007, it became obvious that a broader scope was needed than strictly network tomography, and the name change was instituted in 2008.

Network tomography and mapping are closely related fields. To explain the difference between tomography and mapping, here are two simple definitions. *Network tomography* is the study of a network's internal characteristics using information derived from end-point data. *Network mapping* is the study of the physical connectivity of the Internet, determining what servers and operating systems are running and where. A deeper explanation of tomography follows. For a longer discussion of mapping, please see the article "Mapping Out Faster, Safer Networks."

Network tomography is generally of two types—both of them massive inverse problems. The first type uses end-to-end data to estimate link-level characteristics. This form of tomography often is active in nature, using many pings, traceroutes, and other mapping tools to obtain the necessary data. Due to the large amount of undesirable traffic experienced by many networks, routers or other network equipment may not respond to ping or traceroute requests. This deficiency has led to a second form of network tomography that is sometimes called inferential network tomography. This form of network tomography uses individual router- or node-level measurements to recover path-level information. This data can be obtained passively, and it does not create a traffic burden that has the potential to change the logical network structure. The study of network tomography includes network topology (both logical and physical), the origin-destination traffic matrix, and quality of service parameters such as loss rates or delay characteristics. Accurate and timely information about traffic flows are necessary for good network management.

Network tomography research leads to other topics of interest.

1. How do you measure the network?
2. What kind of networks do these techniques apply to?
3. Does it matter if you test parts of the network individually, and then put them all together, or does the entire network need to be in the test? (integration testing)
4. What sensing techniques are best to use?
5. Exactly what kind of data do you need to gather?
6. What about techniques from other disciplines, such as social networking?  
Will they apply to the networks you are interested in?
7. How does industry do their network mapping?
8. What about attribution?

Some of these questions were addressed at the NMMC sessions, and, therefore, are addressed in the following articles. (See "Compressed Sensing and Network Monitoring," for example, regarding question number four above.) Many more questions arise in the study of network mapping and measurement. The NMMC series has been a huge success, with participants from different countries, federal agencies, universities, and industry. NMMC 2010 will be held August 9-11 at McGill University in Montreal, Canada.



The Next Wave is published to disseminate significant technical advancements in telecommunications and information technologies. Mentions of company names or commercial products do not imply endorsement by the US government. Articles present views of the authors and not necessarily those of NSA or the TNW staff.  
For more information, please contact us at [TNW@tycho.ncsc.mil](mailto:TNW@tycho.ncsc.mil)

---

The graph that appears on the cover of this issue of *The Next Wave* shows the router level connectivity of the Internet as measured by the Internet Mapping Project. The work is being commercially developed by Lumeta Corporation.

**Credit:**  
Patent(s) pending and copyright © Lumeta Corporation 2009. All rights reserved.



# CONTENTS

## FEATURES

- 4 Mapping Out Faster, Safer Networks
- 8 How Akamai Maps the Net:  
*An Industry Perspective*
- 16 Compressed Sensing and Network Monitoring
- 26 Revealing Social Networks of Spammers
- 35 Challenges in Internet Geolocation,  
*or Where's Waldo Online?*
- 39 Clumps, Hoops, and Bubbles—Moving  
Beyond Clustering in the Analysis of Data



# Mapping Out Faster, Safer Networks

Maps. We use them every day. Your GPS guides you to that new restaurant you've wanted to try. The information map in the mall points out where HERE is. Online gamers pull up battle maps to navigate virtual worlds. The social network of your friends and your friends' friends weaves a cat's cradle of intertwined relationships. Your computer files are stored in folders that are displayed hierarchically. Site maps lay out how web pages link up. And think how much easier life would be if you had a map of the labyrinth of telephone options you need to navigate—"Press 1 for hours and locations"... "Press 2 to report a problem"... "Press 3 for account information"—when you try to pay your electric bill over the phone.

Maps don't just show how things are connected. They can also identify trouble spots and weak points you need to be aware of. GPS maps are able to alert you to traffic tie ups due to accidents or lane closures so you can adjust your route. Your security system might display a floor plan that shows which windows and doors are unlocked so you can protect your property.

Network mapping does the same things for the Internet, helping to direct traffic and expose vulnerabilities. Network mapping can happen at different layers of the Internet, including applications, routing, or physical infrastructure, or in different parts of the Internet. Because the Internet changes constantly, any map of any variety—there are many Internet maps and no two agree—addresses a moving target.

## Tracing network routes

Network maps track the routes information packets take across an IP (Internet protocol) network to reach a remote host. Network routing is opportunistic, assigning packets to the first available router. This approach means traffic can be directed along different paths to reach a destination, and the number of hops needed to get there can vary. Network mapping makes it easy to visualize what routes are being taken.

The traceroute network utility was introduced on Unix operating systems in 1987 to map network traffic. Variants of the traceroute program are used on other operating systems—tracert and ping utilities are used on Windows operating systems, and tracepath is the network tool used on current Linux installations.

Network technicians use the traceroute utility to troubleshoot network problems. Knowing a packet's traceroute can help identify failed routers or firewalls that are obstructing traffic. Traceroute can also be used for penetration testing, to hunt for network entry points that could pose a security risk.

Hackers are especially interested in finding back doors into networks, and they have readily

adopted traceroute as an easy way to exploit network vulnerabilities. It didn't take cybercriminals long to discover that not only can the utility be used to locate a network's weak points, initiating traceroute from multiple systems can flood a network to launch a denial-of-service attack.

## The Internet Mapping Project

Traceroutes were initially used by network administrators to troubleshoot and tune local networks, but the utility would eventually be applied on a much larger scale. As the World Wide Web rapidly grew in popularity during the 1990s, the need for a world-wide map was realized. Efforts to map network traffic globally began in earnest with the Internet Mapping Project, started by Bill Cheswick and Hal Burch at Bell Labs in 1998. Every day for eight years, the project recorded traceroutes for trillions of packets traveling across hundreds of thousands of IP networks. The network map that emerged painted a picture resembling a sky filled with fireworks on the Fourth of July. (See the cover image for an example.)

Now managed by the Lumeta Corporation, which spun off from Bell Labs in 2000, the Internet Mapping Project continues to chart the back roads

and thoroughfares of Internet traffic. The goal of the project has been to provide global network visibility through the accurate measurement of four factors: (1) network topology, (2) address space, (3) leaks, and (4) device fingerprints. Independent discovery processes are used to reveal these four components that define a network.

## Network topology

Network topology describes the flow of network traffic and the bottlenecks that slow it down. A computer's network discovery setting affects whether it can see other computers on the network, or be seen by them. A computer can operate in stealth mode by setting its network discovery setting to off. Such "dark" components, when discovered, can add details to a network map for a clearer picture of the network's topology.

Visualizing network topology makes it easier to find ways to accelerate network traffic. The Internet highway is getting clogged with streaming videos, mountains of emails, music downloads, online photo albums, high-definition movies, and epic battles in virtual worlds.

Grid computing may someday usher in an age when bandwidth is virtually unlimited, but in the meantime, pressure is on to squeeze the broadband tube a little harder.

An obvious solution for speeding up network traffic is to increase the bandwidth it travels on. Service providers worldwide have been challenged to roll out 100 Mbps broadband over the next decade, and trials for achieving speeds twice that are already underway. But another approach to moving network traffic faster is to move it smarter. By mapping out a comprehensive route-based topology, the true perimeter of the network is defined—a first step in understanding network limitations. Network maps can then be used to identify bottlenecks and chart shortcuts, making it possible to devise more efficient ways to move packets to their destinations.

## Address space

As enterprises and government agencies try to balance the forces for network change with the requirements for risk management and compliance initiatives, IT security managers are faced with the formidable task of securing what they aren't even aware of. The solution lies partly in discovering all of a network's entities—those that are authorized as well as those that are unauthorized. Network host discovery is used to conduct a census of IP addresses across protocols and reveal known and previously undetected network entities. Host discovery is one of the earliest phases of network reconnaissance.

Address space determines the amount of memory allocated to a computational entity such as a networked computer, a file, a server, or some other device. A unique number assigned by the Internet Assigned Numbers Authority (IANA) identifies individual network nodes. IPv4 (Internet Protocol Version 4) address space is limited to a 32-bit field, yielding a maximum 4,294,967,296 unique addresses. But the supply of available IPv4 addresses is rapidly running out. The move to IPv6, with a 128-bit field, should extend the availability of new addresses well into the future. As the number of IP addresses increases, the need to identify hosts that are active and then focus on them becomes even more important for securing a network.

## Network leaks

Network leaks occur at nodes that inadvertently let information packets pass from a local network to the Internet, or, more important, that let packets from the outside get in. Leak discovery tools identify unauthorized or previously undetected inbound and outbound network traffic and the nodes that passed them through. This information is vital for setting up network defenses.

A common way to probe for network leaks is by tracking the routes of IP packets that use a forged source address. When the targeted machine responds to a traceroute request, logs from these spoofed IP requests reveal which routers passed the

packets on to their destination. Tests from outside a network that turn up inside indicate a firewall leak.

Once network leaks have been detected, IT managers can plug the holes by deleting links to the Internet that shouldn't be there and eliminating unauthorized devices. Lumeta's Cheswick considers taking such precautions simply as good "network hygiene."

## Device fingerprints

Even with the most rigorous network defenses, leaks are likely to persist. Knowing the IP addresses of potential attackers isn't enough to protect a network. Operators can and do change IP addresses, often intentionally to avoid identification. But most devices also carry a code that serves as a fingerprint, making their identities harder to conceal.


Many vendors assign a unique CDI (client device identification) code to the products they manufacture. These device IDs, or device fingerprints, make it possible to challenge off-site computers and other computational devices trying to access a local network. And, as with human fingerprinting, a device fingerprint can be a valuable forensics tool. Device fingerprint discovery provides a summary of the software and hardware settings collected from remote computing devices to identify the source of new attacks or other hosts of interest.

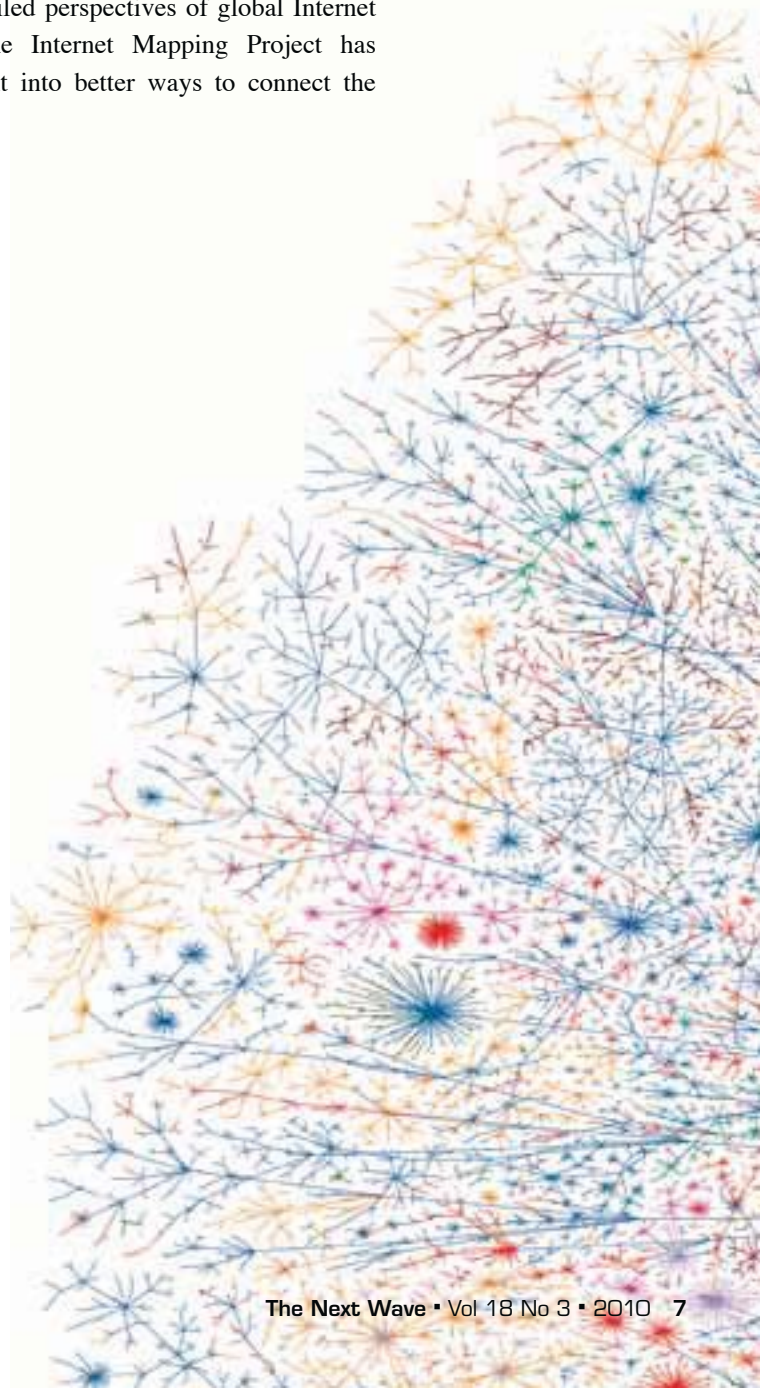
Although a sophisticated attacker can spoof a CDI, there are ways to know if the code has been tampered with. Identifying a device with a fingerprint that has been altered or even removed can tip off nefarious activity, providing another source of information that can be used to enhance network security.

## Faster, more secure networks

Network maps are useful tools for improving existing networks, and they can be crucial for the evolution of IPv6 and beyond. Network mapping makes it possible to visualize network topology, identify address spaces, find network leaks, and

match device fingerprints. What's more, as a network's topology is filled in, a clearer picture of the characteristics of the network is revealed, providing for even more detailed analyses. The analysis of network maps can lead to moving traffic faster, keeping information safer, and finding cyber criminals easier.

The Internet Mapping Project has been a driving force behind the development of effective network mapping tools and practices. By producing varied and detailed perspectives of global Internet connections, the Internet Mapping Project has provided insight into better ways to connect the world. 





# How Akamai Maps the Net: *An Industry Perspective*



**Figures 1 and 2:**

These two snapshots show where Akamai is physically deployed, with each spire representing a city where Akamai has servers. (Akamai can have ten or twenty datacenters in major cities.) The size and color of the spires represent the capacity and load for that region.

In 2010, everyone uses the Internet. Even if you don't browse the Web, your computer, DVD player, and other appliances try to pull software and firmware updates without your interaction. Your cell phone uses the Web to pull updates, ringtones, and video, and soon your car will talk wirelessly to the gas station while it's filling up. Akamai is behind the scenes of much of this networked activity, so whether you know it or not, you probably use Akamai every day.

Akamai Technologies operates a delivery platform for the world's Web content and video, delivering tens of thousands of websites and hundreds of billions of transactions every day. Its customers include the top retailers, portals, media and entertainment firms, advertisers, software and hardware vendors, software-as-a-service (SaaS) providers, marketing organizations, and more. Its range of customers includes Amazon, Apple (iTunes), Microsoft, Yahoo!, ESPN, Ticketmaster, General Motors, Travelocity, MySpace, Adobe, Verizon wireless, Voice of America, NASA, and the US Air Force. Akamai estimates it handles about 25 percent of the world's Web traffic, over two terabits per second most of the day. As a result of its daily operations, Akamai needs to map the Internet, and it can do so from a very unique perspective.

Akamai's platform is a truly distributed architecture, comprising over 60,000 servers deployed in about 1,400 datacenters on about 900 networks worldwide. Geographically, these datacenters are in about 650 cities in 76 countries around the world. (See Figures 1 and 2 for locations.) Akamai does not own any facilities—the company puts its hardware mostly in public Internet collocation facilities on providers such as Qwest, Sprint, France Telecom, NTT, Telia, Comcast, Verizon, or other global backbones, regional providers, and major ISPs. Akamai's primary mission is to improve the overall quality of delivery by situating content, media, and applications delivery capabilities close to the user.

Akamai's design principles were born alongside the algorithms its founders developed at MIT, and they rely heavily on the general notion of *mapping*. Akamai's software platform was built from the ground up and



to scale in ways traditional IT systems do not. Since it's a truly distributed system, multiple components operate physically separate from each other, yet they are interdependent. Akamai Mapping tackles the need to map resources to one another across the network. [1]

Internally to Akamai, a *map* simply expresses how two or more groups are related. Akamai calculates thousands of maps continuously. This article describes three major types of mapping that Akamai performs. The first and most common type is end-user request mapping. The second is mapping connections between two different points on the Internet *through a third point*. And the third is mapping the *geographic location* of a network address.

### End-user request mapping

The most heavily used map translates domain name service (DNS) requests for a resource to network addresses where the resource can be located. In other words, when someone requests some Web content, such as a download or a video, the response tells the user's machine which Akamai server would provide the best connection. The location of the optimal server is calculated based on the structure of the Internet and how it is performing, the user's location, where Akamai servers are located, and how much load exists on Akamai's individual servers at the time.

The basic workflow, depicted in Figure 3, is as follows:

- A user types a website hostname such as `www.nfl.com` into their browser.
- The machine's operating system uses DNS to lookup the IP address for that website.
- The request is redirected to Akamai, transparent to the user, when the DNS name `www.nfl.com` is aliased to an Akamai hostname.

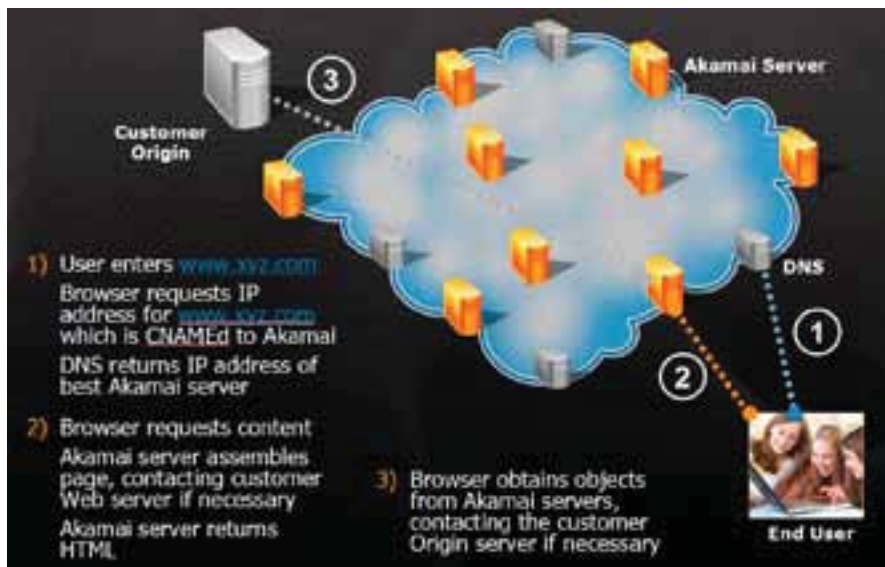


Figure 3: DNS Workflow.

- Behind the scenes, Akamai maps the structure and performance of the Internet. Akamai's DNS responds to the user with a list of network addresses that will provide the best performance at that time.
- The user's computer connects to the Akamai server address and downloads NFL content and videos from a nearby server, usually on the same ISP used to access the Internet.

To map requests to resources on the Internet, Akamai needs to know both the structure and the characteristics of the network between any two relevant points. The first step is to understand the structure of the Internet from the vantage points of Akamai's servers.

For purposes of mapping requests to resources, Akamai tries to develop objective observations of every possible path a particular end user could take in communicating with an Akamai server. If, for example, Akamai has servers located in three different datacenters on the end user's ISP, it makes sense to direct

the user to the datacenter that has the best performance when communicating to the user's location.

To simplify the computational complexity of the problem, Akamai uses the notion of a *core point* on the edge of the network. Multiple end users often come into the network through a particular piece of infrastructure that acts as a gateway for a group of network addresses. As an example, all users at a major corporation may be forwarded to headquarters before their internal network touches the Internet, or users connecting over DSL may all pass through a specific network node before their communications reach the public Internet.

Akamai uses two pieces of information to map the topological structure of the network: BGP and traceroutes.

### Border Gateway Protocol data

BGP, or Border Gateway Protocol, is a protocol used by routers to identify where IP addresses exist on the Internet.

BGP dictates which way networks will send traffic with a particular destination, and it is the underlying mechanism through which the macroscopic Internet maintains interconnectivity.

When Akamai deploys to a network provider, it usually negotiates access to that network's BGP data. The data is obtained through a passive peering session to receive an un-aggregated view of the network from one or more of its routers. Understanding BGP data benefits the network, because it allows Akamai to more effectively minimize the amount of traffic that network must handle, and hence increase its overall efficiency.

With BGP data from a network, Akamai obtains a local view of how the Internet's IP address space, on the whole, is broken out topologically, and what paths exist between different blocks of addresses. This data can change dynamically, as well, so having a direct feed from the routers in a locality allows Akamai to rapidly react to those changes.

While BGP serves to provide a good view on how the Internet is connected, a much more granular view of the network's topology is needed. Akamai's traceroute process fills in a much higher level of detail than the BGP data alone.

## Traceroute data

A traceroute is a network probe between two points that attempts to identify all the infrastructure nodes between those points. A traceroute provides a much higher level of detail than BGP, as BGP only provides coarse information on connectivity and where IP addresses *should be* on the network. Traceroutes provide a more direct and detailed way to measure where IP addresses *really are*. (See "Globe at a Glance" for a representation of a set of traceroutes.) The process is similar to using MapQuest.com for driving

directions between two street addresses to determine all the streets in between.

Although traceroutes operate between two network locations, the request must be initiated from one of the locations, which causes some difficulty. Conducting traceroutes from just one or two network locations will not provide a good sense of the overall interconnectivity of the network, just as running MapQuest from Chicago to every major US city will not reveal any of the roads between, say, New York and Florida or California and Oregon.

Because Akamai is deployed in 900 network providers around the world, it can easily conduct a much deeper look into the world's local Internet connectivity. Akamai also leverages its delivery data to determine where to target its traceroutes—Akamai's commercial services only need to map where end users are located, so Akamai more heavily weights network addresses that have connected to Akamai at some time in the past. Due to the size of Akamai's customer base, however, it is likely that every active machine on the Internet has connected to it at some time—Akamai sees hits from over 300 million unique IP addresses each day, and over 400 million in three months.

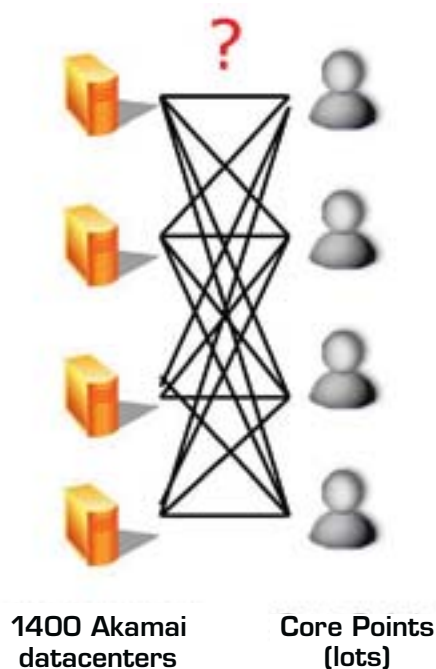
## Measuring the network

Once the mapping function takes into account both BGP and the results of its traceroutes, it has developed a very good map of the edges of the network, and what IP addresses should be considered "core points," or specific network locations beyond which multiple users connect to the Internet. The next step is measuring the characteristics of the network, which occurs on a much higher frequency—every few seconds.

For each core point, Akamai needs to identify information about nearby

Akamai datacenters and how well each datacenter can communicate to the core points. To do this, Akamai conducts specific measurements designed to measure the latency, loss, capacity, and overall availability of the connection. For this calculation, Akamai *no longer cares about the structure of the network*. Because Akamai tries to optimize the quality of delivery, which is dictated strictly by factors that impact the service protocol's operation, the structure of the network can be ignored. (Note this is not "Quality of Service (QoS)" because this is the public Internet; a better term might be "best available QoS.")

To optimize a service protocol's operation, Akamai models how that protocol operates. For normal Web delivery, including HTTP and HTTPS, there are two major considerations. First, the network connection must be somewhat available—high levels of loss will impact performance, though small levels will be corrected through the TCP's operation. After this, however, latency is the biggest factor to consider. The TCP operates through the notion of a *window size*, which is how much data can be in transit before an acknowledgement is received. If the latency is high, the acknowledgements will not be received fast enough, and the overall throughput will be significantly limited. Throughput impacts the speed the user sees a Web page or download, so latency is heavily weighted for a HTTP or HTTPS "map." For streaming media, on the other hand, the primary communication to the user is a continuous stream of data. While there is some interaction between the client and the server, the stream will begin to look choppy and low-quality if too much loss occurs. As a result, for its video "maps," Akamai optimizes for lower loss versus latency.



**Figure 4:**

The basic Akamai mapping problem is a bipartite graph optimization, as shown. End users, aggregated behind “Core Points” on the right, are mapped to the best Akamai datacenter, on the left, based on real-world network conditions.

and processing for each machine, Akamai conducts a secondary calculation local to *regions* of the Internet. Regions of the Internet are used because the previous calculations should have identified what subset of the Internet the core points and datacenters reside in. The use of smaller, region-specific calculations reduces the computational complexity and hence the turnaround time of recalculations.

This secondary calculation uses a bipartite graph model as well, but maps more information. Taking into account the optimal mappings of requests for particular applications between a core point and datacenters, information on existing load, expected load (based on DNS responses already provided), and historic load variation, this calculation’s results are overlaid with a *consistent hashing* technique.

For Web content, Akamai caches data on servers, and each server only has a limited storage capacity. Given that Akamai delivers tens of thousands of websites, it is inefficient to use every server indiscriminately for caching every site. Thus, the notion of *consistent hashing* for this purpose was developed at MIT as part of the algorithms devised before Akamai’s inception [2]. Each customer is associated with a “bucket” of content, referenced via a hash index. In each datacenter, a minimum number of machines are dedicated to handling that hash, to minimize the impact on overall storage resources in the datacenter. When load escalates, however, new servers are recruited consistently as needed, and

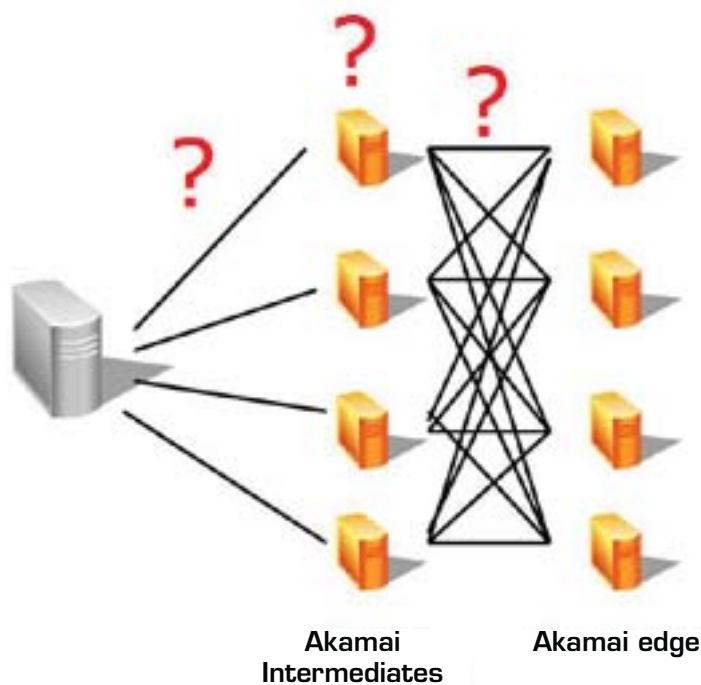
released consistently when load drops. Thus, storage resources are used more only when load escalates and overall efficiency of scale exists, even in a world of unpredictable supply and demand. Consistent hashing means that the mapping processes, which run separate from the delivery servers, can operate without knowing how many “buckets” exist—hashes are consistently arrived at regardless of the size of the space being mapped using the hash.

Akamai uses variations of this mapping mechanism very heavily for a variety of purposes beyond just the mapping of end users to Akamai machines. For example, some Akamai customers host their static websites on distributed persistent storage facilities Akamai has deployed around the world. Mapping allows Akamai to load-balance and optimize the performance between multiple storage centers when an “edge” server needs to fetch content. Other edge servers forward requests through a hierarchy of machines, and the various levels of the hierarchy use maps to automatically identify the next node in the request chain. Live streaming feeds are replicated through the Internet to reduce packet loss, but the selection of replication nodes is done using a similar mapping mechanism to optimize reliable streaming media delivery. Akamai allows customers to run on its platform Java applications, which have their own unique load balancing parameters and timing considerations fed into their maps. Some of Akamai’s customers even use a similar

Once Akamai measures the latency and loss of the network between each nearby datacenter and the core point, it treats the problem as a bipartite graph problem.

As shown in Figure 4, the two groups on the graph are Akamai datacenters on the left and core points on the right. Between each node on the left and each node on the right, a cost is assigned based on the characteristics of the protocol to be optimized—for example, a combination of availability, loss, and latency specific to the application, including historically expected values and passive measurements from previously delivered traffic. The costs are optimized between servers and core points, allowing a high-level mapping of the best and second-best datacenter for a given block of end-user IP space, represented by a core point.

Because it would be inefficient to randomly use servers within a given datacenter, and Akamai must account for other factors such as the load on a given server and the limits on storage



**Figure 5:**

To map through intermediate nodes, Akamai calculates a more complex graph, where paths between Akamai servers, on the right, and some central infrastructure, on the left, are optimized by finding the best intermediate nodes to forward traffic through.

mapping mechanism to load-balance their own datacenters using their own combination of criteria, allowing them to not only load balance but optimize performance, cost, and define how they want failover scenarios to be handled, as well.

### Mapping through intermediate nodes

Traffic on the US highway system is unpredictable, and it can change at any minute on any road. Sending a delivery by truck from one city in the US to another can be done over a variety of routes, but picking the best route is not always straightforward. To provide the best possible speed of delivery, the sender might look at MapQuest.com to determine an initial delivery path, and then pick two or three alternate paths using different highways and possibly different cities as intermediate stopping points to the destination. Multiple copies of the delivery can be sent over each path, and whichever arrives first is delivered. This concept of improving delivery speed through taking multiple, possibly faster, indirect routes is the same concept Akamai

uses to speed long-haul communications on the Internet.

In Akamai's early days, some of the network engineers noticed that it was very difficult to reliably connect to some remote machines for some manual diagnostic checks. For example, it was far easier to connect to machines in South Korea from another Akamai machine in Japan, versus directly from the US. This basic approach uses *intermediates* for optimizing performance, and has a uniquely interesting variation to the traditional Akamai mapping problem: it optimizes the path between two points *by using a third point as an intermediate*.

For this technique, the mapping model switches from being a bipartite graph to being a tripartite graph. See Figure 5 for a visual representation. As an example, consider Taleo, an Akamai customer that provides Software as a Service. Their Web application is secure and highly dynamic, and they primarily use Akamai to make performance of the site consistently fast worldwide. Using Akamai's technique of mapping through intermediate nodes helps them maintain performance without distributing their

application and database infrastructure geographically.

End users make a request for Taleo's application, and they connect to the best Akamai server, as determined by the results of the previous mapping calculation. This Akamai server, however, may be located in Australia, and it needs to access Taleo's central infrastructure, which is located in another continent. The three parts of the graph calculation for this mapping are Taleo's central infrastructure, Akamai's edge datacenter (in this case, the example in Australia), and a set of Akamai nodes that should be considered as intermediates to improve performance and reliability.

The set of intermediates is chosen using a calculation that relies on global BGP data. Since wide-area network communications are dictated by BGP, a direct connection will travel across the path BGP has in place already. BGP does not accommodate performance, however, and in many cases provides a sub-optimal path across the Internet. The only way around is to "trick" BGP by forwarding communications between different network addresses, causing

the communications to take a different network path in the process. Akamai looks at the possible nodes it can use as intermediates, chosen from its global population of 1,400 datacenters, and picks nodes that are likely to provide a diversity of paths in comparison to the existing direct BGP path, and which may also provide lower-latency communications between the two endpoints. As part of its continuous mapping processes, Akamai measures the performance between its possible intermediate nodes and Taleo's infrastructure, between the intermediate nodes and the Akamai edge datacenter, and between the endpoints directly.

The basic principle is to choose the best set of intermediates that provides the lowest overall latency for endpoint-to-endpoint communications, if possible. In many cases, the performance of the direct path is fine, but in many other cases, using an intermediate can provide dramatic performance improvements, sometimes more than two or three times faster than

the direct route. Using intermediates can also bypass network routing problems, which BGP does not always react to effectively.

Since this technique does not rely on caching to operate, Akamai has been able to provide optimizations to both TCP traffic and raw IP communications. Each optimization has its own criteria, and these all are built into custom "flavors" of mapping.

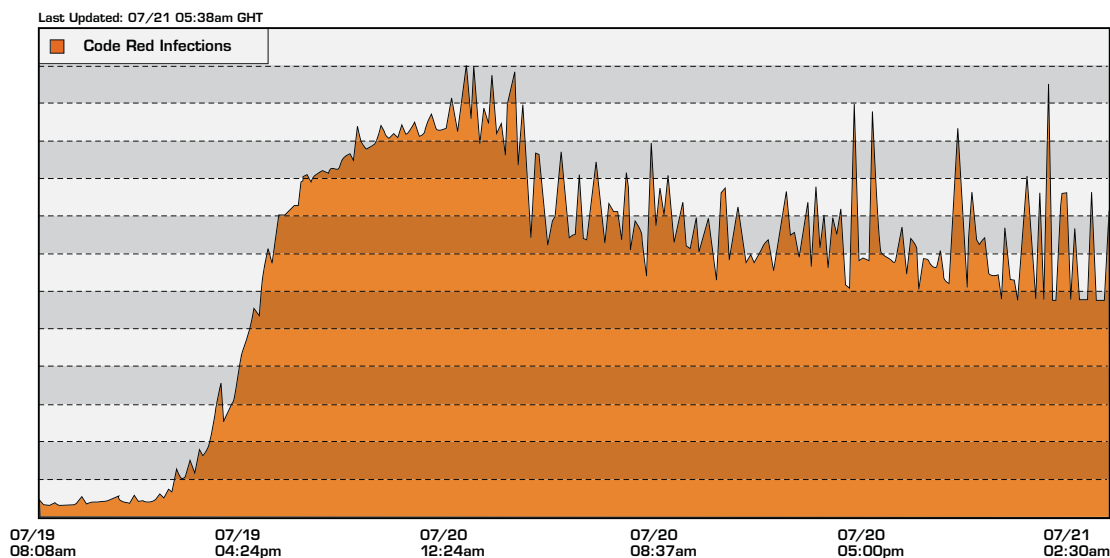
### Mapping data

In addition to mapping its services, Akamai conducts mapping on data. A good example is Akamai's commercial IP geolocation service, which provides geolocation and other information for IP addresses. Due to the nature of the information desired, different data is collected and processed than in other Akamai techniques.

Akamai looks at all available sources of possible geolocation data for inference. The first step in geolocation

mapping is to take into account the structure of the network.

First, using its BGP feeds, Akamai maps the breakdown of IP address blocks and the associated registry information, which is not too accurate in general, but which can be useful if no other information is available. Akamai also examines registries to determine further subdivisions in the network space implicit in how blocks have been registered and assigned. Akamai then performs traceroute queries, which are targeted at identifying the path to each portion of the blocks identified on the Internet, conducted over a variety of periodicities and different levels of coverage of IP space. DNS reverse-lookups of IP addresses are added to the data available, sometimes indicating geographic information on the path identified through a traceroute to an IP address, or relating to the IP address itself. The data is combined with manual information continuously entered by Akamai's interactions with



Graph showing aggregate level of observed activity on the first few days of the Code Red virus outbreak in 2001. Akamai observed the surge connection attempts from infected machines from around the world.

users on the system, with the networks its servers are deployed in, and from ongoing manual investigations of geolocation data. Lastly, Akamai leverages some passive TCP latency data extracted from real-world interactions between end user IP addresses and its edge servers. The end result is derived by applying heuristics to all of these sources to determine an accurate geolocation product at both the country and city level.

Akamai is currently conducting research on using passively observed latency measurements in larger volume, higher fidelity, and with greater rigor to improve the overall accuracy of geolocation.

The geolocation service also provides two other interesting pieces of data about an address that are determined via alternate ways of “mapping” the network: throughput data and proxy data. Whenever Akamai delivers a piece of content to an end user, be it a picture from a news site, an antivirus patch, or a video downloaded from iTunes, Akamai records the amount of time taken to download the content. Using this passive observation, Akamai can very accurately model the throughput and connection speed available at that IP address on the network, and provides the result. While not strictly a “map,” this modeling is more a mapping of characteristics onto the IP address space, providing a greater level of detail.

Proxy data is also inferred using passively collected observations from the hundreds of billions of transactions Akamai serves each day. Specific to geolocation, Akamai looks for the presence of a particular HTTP header that is passed by well-configured proxies, **X-Forwarded-For**. This header indicates that the IP address issuing the HTTP request is doing so on behalf of another IP address. If the identified secondary IP

address in the header is for a public IP address, then that IP is flagged as a proxy.

## Mapping other characteristics of the network: Attacks, proxies, performance

To track the spread of certain viruses across the network, Akamai has deployed a “darknet” of servers in about 200 different networks that passively observe attempts to connect to it. Using the darknet, Akamai can passively observe the spread of different types of virus or worm outbreaks. To heighten the awareness of intrusions specifically targeting a specific organization, not the Internet as a whole, Akamai also models the baseline level of virus and worm intrusions as “background noise” that should not be perceived as targeted intrusion attacks.

Akamai is conducting research into determining more information about proxies, as well. Outside of current techniques, Akamai is looking into ways of modeling what appear to be proxies from other characteristics, such as an abnormal amount of traffic over time, or the number of unique entities identified behind an IP address.

Akamai possesses other information that may be useful for passive analysis of the Internet’s IP space. Through the sheer volume of normal Internet traffic it delivers, estimated to be about 25 percent of the Web, Akamai can model what parts of the Internet appear to have certain types of activity patterns. For example, some parts of the world may be active at certain times of day, and may have software installed that automatically identifies the time zone of the user’s machine or localized software in place. Some parts of the Internet may also have very specific interest in categories of content over time, such as online shopping, media,

or music downloads. Akamai provides some aggregated views into this data, an example of which is shown in Figure 7. In this view, activity across about 100 different news-related websites is aggregated, allowing users to determine if a big news story has hit—or allowing a news website to determine if its traffic fluctuations are in line with the rest of the industry.

Akamai provides some customers with a specific view of what Akamai sees with respect to symptoms of activity on the network relative to what that customer sees. As an example of the type of data available, Akamai can measure if hourly performance to core points in a particular geography is changing as the result of a network-based or physical event, such as a natural disaster, and how it impacts performance. It can also be used to determine if a local view of what is coming in and out of network gateways, and the performance thereof, is similar to what’s happening in the rest of the Internet, or just a localized issue.

## Summary


The dynamic nature of Akamai’s scalable and flexible distributed systems design relies heavily on, and benefits greatly from, the rigorous efforts invested in network mapping. Akamai’s notion of network mapping is relatively broad, and is crafted into several specific methods for real-time service operation or long-term data analysis. Akamai’s network presence and access to traffic provides a very unique vantage point to understand the Internet and how it is operating; these examples provide a sampling of how Akamai takes advantage of this information for very specific purposes. Whatever shapes the Internet morphs into in the future, you can bet that Akamai will be present and will have new ways of mapping it. 



Figure 7:

Snapshot of Akamai's Net Usage Index for News sites, providing a view into the overall usage of about 100 of the Web's top news sites.

of the described methods are patented.

[1] M. Afegan, J. Wein, and A. LaMeyer. Experience with some principles for building an internet-scale reliable system. WORLDS 05: Second Workshop on Real, Large Distributed Systems.

[2] D. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin, R. Panigrahy. Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web.

## References

Many of Akamai's technical publications can be found online at <http://www.akamai.com/publications>. Several



# Compressed Sensing & Network Monitoring

---

*Reprinted with permission of IEEE. Originally published in IEEE Signal Processing Magazine, pp 92-101, March 2008, J. Haupt, W. U. Bajwa, M. Rabbat, and R. Nowak, Compressed Sensing for Networked Data. (©2008 IEEE)*

## Introduction

Network monitoring and inference is an increasingly important component of intelligence gathering, from mapping the structure of the Internet, to discovering clandestine social networks, as well to information fusion in wireless sensor networks. Indeed, several international conferences are dedicated to the nascent field of network science. This article considers a particularly salient aspect of network science that revolves around large-scale distributed sources of data and their storage, transmission, and retrieval. The task of transmitting information from one point to another is a common and well-understood exercise. But the problem of efficiently sharing information from and among a vast number of distributed nodes remains a great challenge, primarily because we do not yet have well developed theories and tools for distributed signal processing, communications, and information theory in large-scale networked systems.

The problem is illustrated by a simple example. Consider a network of  $n$  nodes, each having a piece of information or data  $x_j, j=1, \dots, n$ .

These data could be files to be shared, or simply scalar values corresponding to node attributes or sensor measurements. Let us assume that each  $x_j$  is a scalar quantity for the sake of this illustration. Collectively these data  $\mathbf{x}=[x_1, \dots, x_n]^T$ , arranged in a vector, are called *networked data* to emphasize both the distributed nature of the data and the fact that they may be shared over the underlying communications infrastructure of the network. The networked data vector may be very large;  $n$  may be a thousand, a million, or more. Thus, even the process of gathering  $\mathbf{x}$  at a single point is daunting (requiring  $n$  communications at least). Yet this global sense of the networked data is crucial in applications ranging from network security to wireless sensing. Suppose, however, that it is possible to construct a highly compressed version of  $\mathbf{x}$ , efficiently and in a decentralized fashion. This would offer many obvious benefits, provided that the compressed version could be processed to recover  $\mathbf{x}$  to within a reasonable accuracy.

There are several decentralized compression strategies that could be utilized. One possibility is that the correlations between data at different nodes are known a priori. Then distributed source



coding techniques, such as Slepian-Wolf coding, can be used to design compression schemes without collaboration between nodes. (See [1] and the references therein for an excellent overview of such approaches.) Unfortunately, in many applications, prior knowledge of the precise correlations in the data is unavailable, making it difficult or impossible to apply such distributed source coding techniques. This situation motivates collaborative, in-network processing and compression, in which unknown correlations and dependencies between the networked data can be learned and exploited by exchanging information between network nodes. However, the design and implementation of effective collaborative processing algorithms can be quite challenging, since they too rely on some prior knowledge of the anticipated correlations and depend on somewhat sophisticated communications and node processing capabilities.

This article describes a very different approach to the decentralized compression of networked data. Specifically, consider a compression of the form  $y = Ax$ , where  $A = \{A_{i,j}\}$  is a  $k \times n$  “sensing” matrix with far fewer rows than columns (i.e.,  $k \ll n$ ). The compressed data vector  $y$  is  $k \times 1$ , and therefore is much easier to store, transmit, and retrieve compared to the uncompressed networked data  $x$ . The theory of compressed sensing guarantees that, for certain matrices  $A$ , which are non-adaptive and often quite unstructured,  $x$  can be accurately recovered from  $y$  whenever  $x$  itself is compressible in some domain (e.g., frequency, wavelet, time) [2]–[5].

To carry the illustration further, and to motivate the approaches proposed in this article, let us look at a very concrete example. Suppose that *most* of the network nodes have the same nominal data value, but the few remaining nodes have different values. For instance, the values could correspond to security statistics or sensor readings at each node. The networked data vector in this case is mostly constant, except for a few deviations in certain locations. This minority may be of most interest in security or sensing applications. Clearly  $x$  is quite compressible; the nominal value plus the locations and values of the few deviant cases suffice for its specification.

Consider a few possible situations in this networked data compression problem. First, if the nominal value were known to all nodes, then the desired compression is accomplished simply

by the deviant nodes sending that notification. Second, if the nominal value were not known, but the deviant cases were assumed to be isolated, then the nodes could simply compare their own values to those of their nearest neighbors to determine the nominal value and any deviation of their own. Again, notifications from the deviant nodes would provide the desired compression. There is a third, more general, scenario in which such simple local processing schemes can break down. Suppose that the nominal value is unknown to the nodes a priori, and that the deviant cases could be isolated or clustered. Since the deviant nodes may be clustered together, simply comparing values between neighboring nodes may not reveal them all, and perhaps not even the majority of them, depending on the extent of clustering. Indeed, distributed processing schemes in general are difficult to design without prior knowledge of the anticipated relations among data at neighboring nodes. This serves as a motivation for the theory and methods discussed here.

Compressed sensing offers an alternative measurement approach that does not require any specific prior signal knowledge and is an effective (and efficient) strategy in each of the situations described above. The values of all nodes can be recovered from the compressed data  $y = Ax$ , provided its size  $k$  is proportional to the number of deviant nodes. As we shall see,  $y$  can be efficiently computed in a distributed manner, and by virtue of its small size, it is naturally easy to store and transmit. In fact, in certain wireless network applications (see *Wireless Sensor Networks in the Networked Data Compression in Action* section of this article for details),  $y$  can be computed in the air itself, rather than in silicon! Thus, compressed sensing offers two highly desirable features for networked data analysis. The method is *decentralized*, meaning that distributed data can be encoded without a central controller, and *universal*, in the sense that sampling does not require a priori knowledge or assumptions about the data. For these reasons, the advantages of compressed sensing have already caught on in the research community, as evidenced by several recent works [6]–[10].

### Compressed sensing basics

The theory of compressed sensing (CS) extends traditional sensing and sampling systems to a much broader class of signals. According to CS

theory, any sufficiently compressible signal can be accurately recovered from a small number of non-adaptive, randomized linear projection samples. For example, suppose that  $x \in \mathbb{R}^n$  is  $m$ -sparse (i.e., it has no more than  $m$  nonzero entries) where  $m$  is much smaller than the signal length  $n$ . Sparse vectors are very compressible, since they can be completely described by the locations and amplitudes of the non-zero entries. Rather than sampling each element of  $x$ , CS directs us to first precondition the signal by operating on it with a diversifying matrix, yielding a signal whose entries are mixtures of the non-zero entries of the original signal. The resulting signal is then sampled  $k$  times to obtain a low-dimensional vector of observations. Overall, the acquisition process can be described by the observation model  $y = Ax + \epsilon$ , where the matrix  $A$  is a  $k \times n$  CS matrix that describes the joint operations of preconditioning and subsampling, and  $\epsilon$  represents errors due to noise or other perturbations. The main results of CS theory have established that if the number of CS samples is a small integer multiple greater than the number of non-zero entries in  $x$ , then these samples sufficiently “encode” the salient information in the sparse signal and an accurate reconstruction from  $y$  is possible. These results

are very promising because at least  $2m$  pieces of information (the location and amplitude of each nonzero entry) are generally required to describe any  $m$ -sparse signal, and CS is an effective way to obtain this information in a simple, non-adaptive manner. The next few subsections explain, in some detail, how this is accomplished.

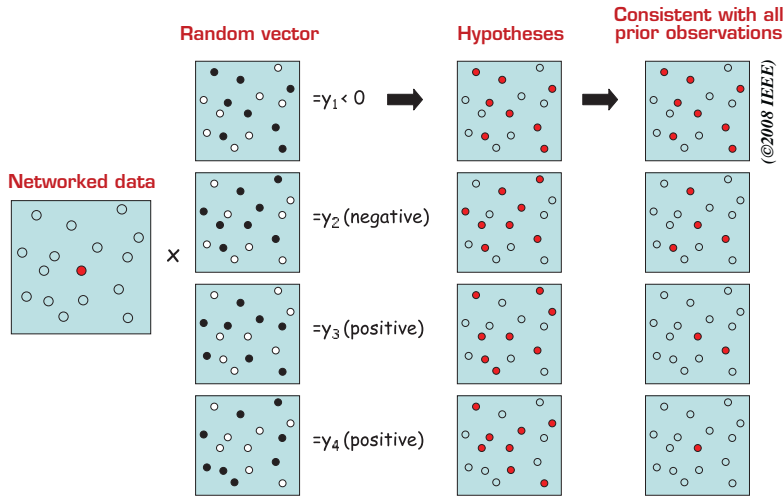
### Compressed sensing for networked data

To illustrate the CS random projection encoding and reconstruction ideas, consider the simple reconstruction example (Figure 1). Suppose that in a network of  $n$  sensors, only one of the sensors is observing some positive value, while the rest of the sensors observe zero. The goal is to identify which sensor measures the nonzero value using a minimum number of observations. Consider making random projection observations of the data, where each observation is the projection of the sensor readings onto a random vector having entries  $\pm 1$  each with probability  $1/2$ . The value of each observation, along with knowledge of the random vector onto which the data was projected, can be used to identify a set of about  $n/2$  hypothesis sensors that are consistent with that particular observation. The estimate of the anomalous sensor given  $k$  observations is simply the intersection of the hypothesis sets consistent with each of the  $k$  observations. It is easy to see that, on average, about  $\log n$  observations are required before the correct (unique) sensor is identified. Define the  $\ell_0$  quasi-norm  $\|z\|_0$  to be equal to the number of nonzero entries in the vector  $z$ . Then this simple procedure can be thought of as the solution of the optimization problem

$$\arg \min_z \|z\|_0 \text{ subject to } y = Az. \quad (1)$$

### Encoding requirements

Suppose that for some observation matrix  $A$  there is a nonzero  $m$ -sparse signal  $x$  such that the observations  $y = Ax = 0$ . Recovery of  $x$  is impossible in this setting, since the observations provide no information about the specific signal being observed. Matrices that are resilient to such ambiguities are those that satisfy the Restricted Isometry Property (RIP) [2], [11], [12]. Essentially, a  $k \times n$  sensing matrix  $A$  with unit-normed rows (i.e.,  $\sum_{j=1}^n A_{i,j}^2 = 1$  for  $i = 1, 2, \dots, k$ ) is said to satisfy a RIP of order  $s$  whenever  $\|Ax\|_2^2 \approx k\|x\|_2^2/n$  holds simultaneously for all  $s$ -sparse vectors  $x \in \mathbb{R}^n$ . The RIP is so-named because it describes matrices that impose



**Figure 1:** A simple reconstruction example on a network of  $n = 16$  nodes. One distinguished sensor observes a positive value while the remaining  $n - 1$  observe zero. The task is to identify which sensor is different by using as few observations as possible. In the CS approach, the data are projected onto random vectors, such as those depicted in the second column (where nodes indicated in black multiply their data value by  $-1$  and those in white by  $+1$ ). The third column shows that about  $n/2$  hypothesis sensors are consistent with each random projection observation, but the number of hypotheses that are simultaneously consistent with all observations (shown in the fourth column) decreases exponentially with the number of observations. The random projection observations are approximately performing binary bisections of the hypothesis space, and only about  $\log n$  observations are needed to determine which sensor reads the nonzero value.

near-isometry (approximate length preservation) on a restricted set of subspaces (the subspaces of  $s$ -sparse vectors). In simpler terms, a matrix satisfies RIP if and only if vectors that are sufficiently sparse are not in the null space of the matrix.

In practice, sensing matrices that satisfy the RIP are easy to generate. It has been established that  $k \times n$  matrices whose entries are independent and identically distributed realizations of certain zero-mean random variables with variance  $1/n$  satisfy a RIP with very high probability when  $k \geq \text{const} \cdot \log n \cdot m$  [2], [3], [13]. Physical limitations of real sensing systems motivate the unit-norm restriction on the rows of  $A$ , which essentially limits the amount of “sampling energy” allotted to each observation.

### Decoding: Algorithms and bounds

Compressed sensing is a form of subsampling, so aliasing is present, and needs to be accounted for in the reconstruction process. The same compressed data could be generated by many  $n$ -dimensional vectors, but the RIP implies that only one of these is sparse. This might seem to require that any reconstruction algorithm must exhaustively search all sparse vectors, but fortunately the process is much more tractable. Given a vector of (noise-free) observations  $y = Ax$ , the unknown  $m$ -sparse signal  $x$  can be recovered exactly as the unique solution to

$$\arg \min_z \|z\|_1 \text{ subject to } y = Az, \quad (2)$$

where  $\|z\|_1 = \sum_{i=1}^n |z_i|$  denotes the  $\ell_1$ -norm, provided  $A$  satisfies RIP of order  $2m$  [12]. The recovery procedure can be cast as a linear program, so solution methods are tractable even when  $n$  is very large.

Compressed sensing remains quite effective even when the samples are corrupted by additive noise, which is important from a practical point of view since any real system will be subjected to measurement inaccuracies. A variety of reconstruction methods have been proposed to recover (an approximation of)  $x$  when observations are corrupted by noise. For example, estimates  $\hat{x}$  can be obtained as the solutions of either

$$\arg \min_z \|z\|_1 \text{ subject to } \|A^T(y-Az)\|_\infty \leq \lambda_1, \quad (3)$$

where  $\|z\|_\infty = \max_{i=1, \dots, n} |z_i|$  [5], or the penalized least squares minimization

$$\arg \min_z \left\{ \|y-Az\|_2^2 + \lambda_2 \|z\|_0 \right\} \quad (4)$$

as proposed in [4], for appropriately chosen regularization constants  $\lambda_1$  and  $\lambda_2$  that each depend on the noise variance. In either case, the reconstruction error  $E[\|x-\hat{x}\|_2^2/n]$  decays at a rate of  $(m \log n/k)$ . In practice, the optimization (3) can be solved by a linear program, while (4) is often solved by convex relaxation—replacing the  $\ell_0$  penalty with the  $\ell_1$  penalty. The appeal of CS is readily apparent from the error rate which (ignoring the logarithmic factor) is proportional to  $m/k$ , the variance of an estimator of  $m$  parameters from  $k$  observations. In other words, CS is able to both identify the locations and estimate the amplitudes of the non-zero entries without any specific prior knowledge about the signal except the assumption of sparsity.

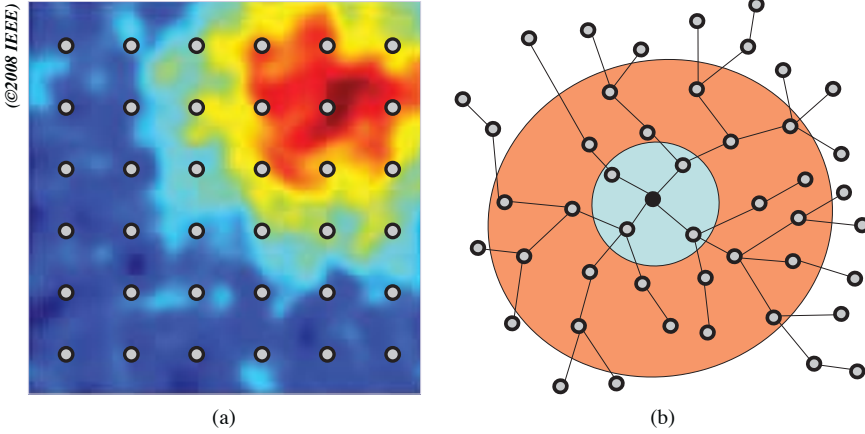
### Transform domain sparsity

Suppose the observed signal  $x$  is not sparse, but instead a suitably transformed version is. Specifically, let  $T$  be a transformation matrix, and assume that  $\theta = Tx$  is sparse. The CS observations can be written as  $y = Ax = AT^{-1}\theta$ . If  $A$  is a random CS matrix satisfying the RIP, then in many cases so is the product matrix  $AT^{-1}$  [13]. Consequently, the CS observation process does not require prior knowledge of the domain in which the data are compressible. The sparse vector  $\theta$  (and hence  $x$ ) can be accurately recovered from  $y$  using the reconstruction techniques described above. For example, in the noiseless setting one can solve

$$\hat{\theta} = \arg \min_z \|z\|_1 \text{ subject to } y = AT^{-1}z, \quad (5)$$

to obtain an exact reconstruction of the transform coefficients of  $x$ . Note that, while the samples do not require selection of an appropriate sparsifying transform, the reconstruction does.

Often, signals of interest will not be exactly sparse, but instead most of the energy is concentrated on a relatively small set of entries while the remaining entries are very small. The degree of effective sparsity of such signals can be quantified with respect to a given basis. Formally, for a signal  $x$  let  $x^s$  be the approximation of  $x$  formed by retaining the  $s$  coefficients having largest magnitude in the transformed representation  $\theta = Tx$ . Then  $x$  is called  $\alpha$ -compressible if the approximation error obeys



**Figure 2:** Sparsifying transformation techniques depend on network topologies. The smoothly varying field in (a) is monitored by a network of wireless sensors deployed uniformly over the region, and standard transform techniques can be used to sparsify the networked data. For more abstract topologies, graph wavelets can be effective. In (b), the graph (Haar) wavelet coefficient at the location of the black node and scale three is given by the difference of the average data values at the nodes in the red and blue regions.

$$\frac{\|x-x^s\|_2^2}{n} \leq \text{const} \cdot s^{-2\alpha} \quad (6)$$

for some  $\alpha = \alpha(x, T) > 0$ . This model describes, for example, signals whose ordered (transformed) coefficient amplitudes exhibit power-law decay. Such behavior is associated with images that are smooth or have bounded variation [3], [11], and is often observed in the wavelet coefficients of natural images. In this setting, CS reconstruction techniques can again be applied to obtain an estimate of the transformed coefficients directly. For example, the solutions of optimizations analogous to (3) and (4) yield estimates whose estimation error decays at the rate  $(\log n/k)^{2\alpha/2\alpha+1}$ , quantifying the simultaneous balancing of the errors due to approximation and estimation [4]. This result guarantees that even when signals are only approximately sparse, consistent estimation is still possible.

## Sparsifying networked data

Compressed sensing can be very effective when  $x$  is sparse or highly compressible in a certain basis or dictionary. But, while transform-based compression is well-developed in traditional signal and image processing domains, the understanding of sparsifying/compressing bases for networked data is far from complete. There are, however, a few promising new approaches to the design of transforms for networked data, some of which are described below.

### Spatial compression

Suppose a wireless sensor network is deployed to monitor a certain spatially-varying phenomenon, such as temperature, light, or moisture. The physical field being measured can be viewed as a signal or image with a degree of spatial correlation or smoothness. If the sensors are geographically placed in a uniform fashion,

such as in Figure 2(a), then sparsifying transforms may be readily borrowed from traditional signal processing. In these settings, the sensor locations can be viewed as *sampling locations* and tools like the discrete Fourier transform (DFT) or discrete wavelet transform (DWT) may be used to sparsify the networked data. In more general settings, wavelet techniques can be extended to also handle nonuniform distribution of sensors [14].

### Graph wavelets

Standard signal transforms cannot be applied in more general situations. For example, many network monitoring applications rely on the analysis of traffic levels at the network nodes. Changes in the behavior of traffic levels can be indicative of variations in network usage, component failures, or malicious activities. There are strong correlations between traffic levels at different nodes, but the topology and routing affect the nature of these relationships in complex ways. Graph wavelets, developed with these challenges in mind, adapt the design principles of the DWT to arbitrary networked data [15].

To understand graph wavelets, it is useful to first consider the Haar wavelet transform, which is the simplest form of DWT. The Haar wavelet coefficients are essentially obtained as digital differences of the data at different scales of aggregation. The coefficients at the first scale are differences between neighboring data points, and those at subsequent spatial scales are computed by first aggregating data in neighborhoods (dyadic intervals in one dimension and square regions in two dimensions) and then computing differences between neighboring aggregations.

Graph wavelets are a generalization of this construction, where the number of hops between nodes in a network provides a natural distance

measure that can be used to define neighborhoods. The size of each neighborhood (with radius defined by the number of hops) provides a natural measure of scale, with smaller sizes corresponding to finer spatial analysis of the networked data. Graph wavelet coefficients are then defined by aggregating data at different scales, and computing differences between aggregated data, as shown in Figure 2(b). Further details and generalizations of this can be found in [15].

### Diffusion wavelets

Diffusion wavelets provide an alternative approach to constructing a multi-scale representation for networked data. Unlike graph wavelets, which produce an overcomplete dictionary, diffusion wavelets produce an orthonormal basis tailored to a specific network by analyzing eigenvectors of a diffusion matrix derived from the network adjacency matrix (hence the name “diffusion wavelets”). The resulting basis vectors are generally localized to neighborhoods of varying size and may also lead to a sparsifying representation of networked data. A thorough treatment of this topic can be found in [16].

One example of sparsification using diffusion wavelets is shown in Figure 3, where the node data correspond to traffic rates through routers in a computer network. There are several highly localized regions of activity, while most of the remaining network exhibits only moderate levels of traffic. The traffic data are sparsely represented in the diffusion wavelet basis, and a small number of coefficients can provide an accurate estimate of the actual traffic patterns.

### Networked data compression in action

This section describes two techniques for obtaining projections of networked data onto general vectors, which can be thought of as the rows of the sensing matrix  $A$ . The first approach described below assumes that the network is any general multihop network. This model could explain, for example, wireless sensor networks, wired local area networks, or even portions of the Internet. In the multihop setting, the projections can be computed and delivered to every subset of nodes in the network using gossip/consensus techniques, or they might be delivered to a single point using clustering and aggregation. The second, more

specific, approach described below is motivated by many wireless sensor networking applications in which explicit routing information is difficult to obtain and maintain in real time. In this setting, sensors contribute their measurements in a joint fashion by simultaneous wireless transmissions to a distant processing location, and the observations are accumulated and processed at that (single) destination point.

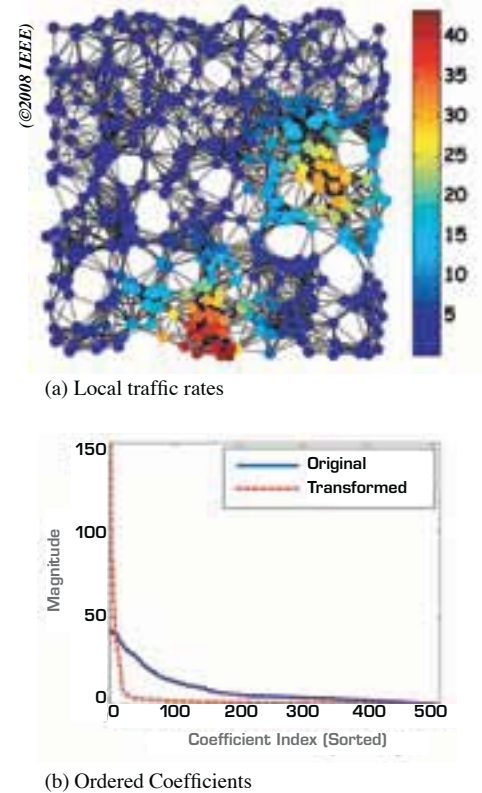
### Compressed sensing for networked data storage and retrieval

In general multihop networks, two simple steps can be used for the decentralized computation and distribution of each CS observation of the form  $y_i = \sum_{j=1}^n A_{i,j} x_j$ ,  $i = 1, \dots, k$ :

**Step 1:** Each of the  $n$  sensors,  $j = 1, \dots, n$ , locally computes the term  $A_{i,j} x_j$  by multiplying its data with the corresponding element of the sensing matrix. The sensing matrix can be generated in a distributed fashion by letting each node locally generate a realization of  $A_{i,j}$  using a pseudo-random number generator seeded with its identifier. (In this example, the integers  $j = 1, \dots, n$  serve as the identifiers.) Given the identifiers of the nodes, the destination node(s) can also easily generate the random vectors  $\{A_{i,j}\}_{i=1}^k$  for each sensor  $j = 1, \dots, n$ .

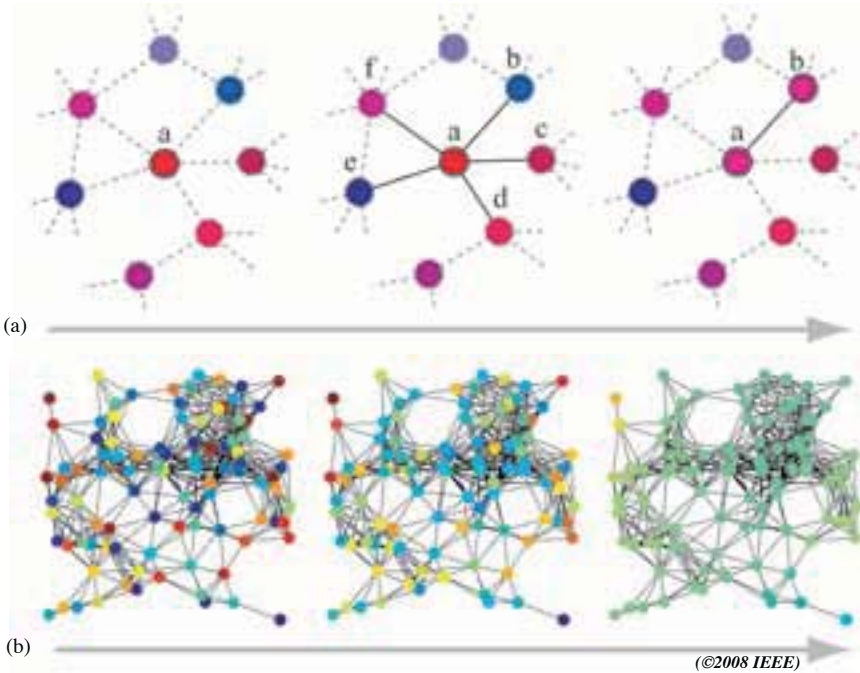
**Step 2:** The local terms  $A_{i,j} x_j$  are simultaneously aggregated and distributed across the network using randomized gossip, which is a simple iterative decentralized algorithm for computing linear functions such as  $y_i = \sum_{j=1}^n A_{i,j} x_j$  (see Figure 4). Note that gossip algorithms are highly resilient to node failures because: (i) each node only exchanges information with its immediate neighbors, and (ii) when they terminate, the value of  $y_i$  is available at every node in the network.

Since the above procedure ensures that the networked data projections are known at every node, a user can query any node in the network and compute  $\hat{x}$  via one of the reconstruction methods



**Figure 3:**

An illustration of the compressibility of spatially correlated networked data using diffusion wavelets. The actual networked data shown in (a) are not sparse, but can be represented with a small number of diffusion wavelet coefficients, as seen in (b).



**Figure 4: Randomized gossip:** (a) depicts one iteration, in which the color of a node corresponds to its local value. To begin, the network is initialized to a state where each node has a value  $x_i(0)$ ,  $i = 1, \dots, n$ . Then in an iterative, asynchronous fashion, a random node  $a$  “activates” and chooses one of its neighbors  $b$  at random. The two nodes then “gossip” by exchanging their values  $x_a(t)$  and  $x_b(t)$ , or in the CS setting the values multiplied by pseudo-random numbers, and perform the update  $x_a(t+1) = x_b(t+1) \leftarrow (x_a(t) + x_b(t))/2$ , while the data at all the other nodes remains unchanged. In (b), we have an example network of 100 nodes with: (i) random initial values (left), (ii) after each node has communicated five times with each of its neighbors (middle), and (iii) after each node has communicated 50 times with each of its neighbors (right). It can be shown that for this simple procedure,  $x_i(t)$  converges to the average of the initial values,  $1/n \sum_{j=1}^n x_j(0)$ , at every node in the network as  $t$  tends to infinity.

outlined in the Compressed Sensing Basics section. Further, this can be quite an efficient procedure in many scenarios. For example, in networks with power-law degree distributions such as the Internet, an optimized gossip algorithm uses on the order of  $kn$  transmissions to compute  $k$  CS observations [17], generally  $k \ll n$ . So this is much more efficient than exhaustively exchanging raw data values, which would take about  $n^2$  transmissions. Of course, if the location of the node to be queried is fixed a priori—and if the network provides reliable routing service—then it may be more efficient to replace gossip computation with aggregation up a spanning tree or around a cycle. For more on using gossip algorithms to compute/distribute compressed data in multihop networks, see [7].

### Compressed sensing in wireless sensor networks

A typical wireless sensor network, as in Figure 5, consists of a large number of small, inexpensive wireless sensors, spatially distributed over a region of interest that can sense the physical environment in a variety of modalities. The essential task in many applications of sensor networks is to extract some relevant information from distributed data and then wirelessly deliver it to a distant destination, called the fusion center (FC). While this task can be accomplished in a number of ways, one particularly attractive technique corresponds to delivering random projections of the sensor networked data to the FC by exploiting recent results on uncoded (analog) coherent transmission schemes in wireless

sensor networks [18]–[21]. The proposed distributed communication architecture—introduced in [6], [8], and refined in [22]—requires only one (network) transmission per random projection and is based on the notion of so-called “matched source-channel communication” [20], [21]. Here, the CS projection observations are simultaneously calculated (by the superposition of radio waves) and communicated using amplitude-modulated coherent transmissions of randomly weighted sensed values directly from the sensor nodes to the FC via the air interface. Algorithmically, sensor nodes sequentially perform the following steps in order to communicate  $k$  random projections of the networked data to the FC:

**Step 1:** Each of the  $n$  sensors locally draws  $k$  elements of the random projection vectors  $\{A_{i,j}\}_{i=1}^k$  by using its network address as the seed of a pseudo-random number generator. Given the network addresses of the nodes, the FC can also easily reconstruct the random vectors  $\{A_{i,j}\}_{i,j=1}^{k,n}$ .

**Step 2:** The sensor at location  $j$  multiplies its measurement  $x_j$  with  $\{A_{i,j}\}_{i=1}^k$  to obtain a  $k$ -tuple

$$v_j = (A_{1,j} x_j, \dots, A_{k,j} x_j)^T, j = 1, \dots, n, \quad (7)$$

and all the nodes coherently transmit their respective  $v_j$ 's in an analog fashion over the network-to-FC air interface using  $k$  transmissions. Because of the additive nature of radio waves, the corresponding received signal at the FC at the end of the  $k$ -th transmission is given by

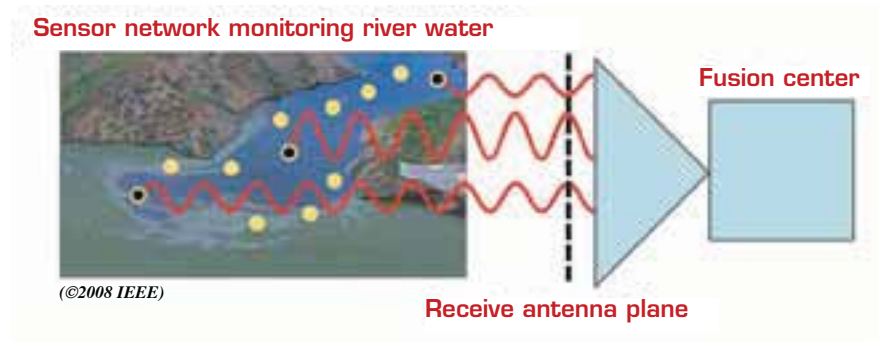
$$y = \sum_{j=1}^n v_j + \epsilon = Ax + \epsilon, \quad (8)$$

where  $\epsilon$  is the noise generated by the communication receiver circuitry of the FC. The steps above correspond to a completely decentralized way of delivering  $k$  random projections of the networked data to the FC by employing  $k$  (network) transmissions. The final estimate  $\hat{x}$  can be computed at the FC via any of the methods outlined earlier. As noted earlier, the main advantage of using this approach for computing random projections is that it can be implemented *without* any complex routing information and as a result might be a more suitable and scalable option in many sensor networking applications.

## Conclusions and extensions

This article has described how compressed sensing techniques could be utilized to reconstruct sparse or compressible networked data in a variety of practical settings, including general multihop networks and wireless sensor networks. Compressed sensing provides two key features, universal sampling and decentralized encoding, making it a promising new paradigm for networked data analysis. The focus here was primarily on managing resources during the encoding process, but it is important to note that the decoding step also poses a significant challenge. Indeed, the study of efficient decoding algorithms remains at the forefront of current research [23]–[25].

In addition, specialized measurement matrices, such as those resulting from Toeplitz-structured matrices [26] and the incoherent basis sampling methods described in [27], lead to significant reductions in the complexity of convex decoding methods. Fortunately, the sampling matrices inherent to these methods can be easily implemented using the network projection approaches described above. For example, Toeplitz-structured CS matrices naturally result when each node uses the same random number generation scheme and seed value, in which each node advances its own random sequence by its unique (integer) identifier at initialization. Similarly, random samples from any orthonormal basis (the observation model described in [27]) can easily be obtained in the settings described above if each node is preloaded with its weights for each basis element in the corresponding orthonormal transformation matrix. For each



**Figure 5:** An illustration of a wireless sensor network and fusion center.

A number of sensor nodes monitor the river water for various forms of contamination and periodically report their findings over the air to the fusion center. CS projection observations are obtained by each sensor transmitting a sinusoid with amplitude given by the product of the sensor measurement and a pseudo-random weight. When the transmissions arrive in phase at the fusion center, the amplitude of the resulting received waveform is the sum of the component wave amplitudes.

observation, the requesting node (or fusion center) broadcasts a random integer between 1 and  $n$  to the nodes to specify which transform coefficient from the predetermined basis should be obtained, and the projection is delivered using any suitable method described above.

Finally, it is worth noting that matrices satisfying the RIP also approximately preserve additional geometrical structure on subspaces of sparse vectors, such as angles and inner products, as shown in [28]. A useful consequence of this result is that an ensemble of CS observations can be “data mined” for events of interest [29], [30]. For example, consider a network whose data may contain an anomaly that originated at one of  $m$  candidate nodes. An ensemble of CS observations of the networked data, collected without any a priori information about the anomaly, can be analyzed “post-mortem” to accurately determine which candidate node was the likely source of the anomaly. Such extensions of CS theory suggest efficient and scalable techniques for monitoring large-scale distributed networks, many of which can be performed without the computational burden of reconstructing the complete networked data.  $\square$

## References

- [1] S. S. Pradhan, J. Kusuma, and K. Ramchandran. Distributed compression in a dense microsensor network. *IEEE Signal Processing Mag.*, 19(2):51–60, March 2002.
- [2] E. J. Candès and T. Tao. Decoding by linear programming. *IEEE Trans. Inform. Theory*, 51(12):4203–4215, December 2005.
- [3] D. L. Donoho. Compressed sensing. *IEEE Trans. Inform. Theory*, 52(4):1289–1306, April 2006.
- [4] J. Haupt and R. Nowak. Signal reconstruction from noisy random projections. *IEEE Trans. Inform. Theory*, 52(9):4036–4048, September 2006.
- [5] E. Candès and T. Tao. The Dantzig selector: Statistical estimation when  $p$  is much larger than  $n$ . *Annals of Statistics*, 35(6):2313–2351, December 2007.
- [6] W. U. Bajwa, J. Haupt, A. M. Sayeed, and R. Nowak. Compressive wireless sensing. In *Proc. IPSN'06*, pages 134–142, Nashville, TN, April 2006.



- [7] M. Rabbat, J. Haupt, A. Singh, and R. Nowak. Decentralized compression and predistribution via randomized gossiping. In *Proc. IPSN'06*, pages 51–59, Nashville, TN, April 2006.
- [8] W. U. Bajwa, J. Haupt, A. M. Sayeed, and R. Nowak. A universal matched source-channel communication scheme for wireless sensor ensembles. In *Proc. ICASSP'06*, pages 1153–1156, Toulouse, France, May 2006.
- [9] D. Baron, M. B. Wakin, M. F. Duarte, S. Sarvotham, and R. G. Baraniuk. Distributed compressed sensing. pre-print. [Online]. Available: <http://www.ece.rice.edu/~drorb/pdf/DCS112005.pdf>
- [10] W. Wang, M. Garofalakis, and K. Ramchandran. Distributed sparse random projections for refinable approximation. In *Proc. IPSN'07*, pages 331–339, Cambridge, MA, April 2007.
- [11] E. J. Candès and T. Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE Trans. Inform. Theory*, 52(12):5406–5425, December 2006.
- [12] E. J. Candès. The restricted isometry property and its implications for compressed sensing. In *C. R. Acad. Sci., Ser. I*, vol. 346, pages 589–592, Paris, 2008.
- [13] R. Baraniuk, M. Davenport, R. A. DeVore, and M. B. Wakin. A simple proof of the restricted isometry property for random matrices. *Constructive Approximation*, 28(3):253–263, December 2008.
- [14] R. Wagner, R. Baraniuk, S. Du, D. Johnson, and A. Cohen. An architecture for distributed wavelet analysis and processing in sensor networks. In *Proc. IPSN'06*, pages 243–250, Nashville, TN, April 2006.
- [15] M. Crovella and E. Kolaczyk. Graph wavelets for spatial traffic analysis. In *Proc. IEEE Infocom.*, vol. 3, pages 1848–1857, March 2003.
- [16] R. Coifman and M. Maggioni. Diffusion wavelets. *Applied Computational and Harmonic Analysis*, 21(1):53–94, July 2006.
- [17] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized gossip algorithms. *IEEE Trans. Inform. Theory*, 52(6):2508–2530, June 2006.
- [18] M. Gastpar and M. Vetterli. Source-channel communication in sensor networks. In *Proc. IPSN'03*, pages 162–177, Palo Alto, CA, April 2003.
- [19] K. Liu and A. M. Sayeed. Optimal distributed detection strategies for wireless sensor networks. In *Proc. 42nd Annual Allerton Conference on Commun., Control and Comp.*, October 2004.
- [20] M. Gastpar and M. Vetterli. Power, spatio-temporal bandwidth, and distortion in large sensor networks. *IEEE Journal Select. Areas Commun.*, 23(4):745–754, April 2005.
- [21] W. U. Bajwa, A. M. Sayeed, and R. Nowak. Matched source-channel communication for field estimation in wireless sensor networks. In *Proc. IPSN'05*, pages 332–339, Los Angeles, CA, April 2005.
- [22] W. U. Bajwa, J. Haupt, A. M. Sayeed, and R. Nowak. Joint source-channel communication for distributed estimation in sensor networks. *IEEE Trans. Inform. Theory*, 53(10):3629–3653, October 2007.
- [23] A. C. Gilbert and J. Tropp. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Trans. Inform. Theory*, 53(12):4655–4666, December 2007.
- [24] M. Figueiredo, R. Nowak, and S. Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE Journal Select. Topics in Signal Processing*, 1(4):586–597, 2007.
- [25] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky. An interior point method for large-scale  $\ell_1$ -regularized least squares. *IEEE Journal Select. Topics in Signal Processing*, 1(4):606–617, 2007.
- [26] W. Bajwa, J. Haupt, G. Raz, and R. Nowak. Toeplitz-structured compressed sensing matrices. In *Proc. SSP'07*, pages 294–298, Madison, WI, August 2007.
- [27] E. Candès and J. Romberg. Sparsity and incoherence in compressive sampling. *Inverse Problems*, 23(3):969–985, 2006.
- [28] J. Haupt and R. Nowak. *A generalized restricted isometry property*. University of Wisconsin - Madison, Tech. Rep. ECE-07-1, May 2007.
- [29] J. Haupt and R. Nowak. Compressive sampling for signal detection. In *Proc. ICASSP'07*, Honolulu, HI, April 2007.
- [30] J. Haupt, R. Castro, R. Nowak, G. Fudge, and A. Yeh. Compressive sampling for signal classification. In *Proc. 40th Asilomar Conference on Signal, Systems, and Computers*, pages 1430–1434, Pacific Grove, CA, October 2006.



# Revealing Social Networks of Spammers

Spam doesn't really need an introduction—anyone who owns an email address likely receives spam emails every day. However, spam is much more than just an annoyance. Spam's hidden economic cost for companies in wasted storage, bandwidth, technical support, and most important, the loss of employee productivity, is astronomical. The annual cost of spam for a company with 12,000 employees is approximately \$2.4 million, according to a study conducted by *Windows & .NET Magazine* in 2003 [1]. Since then, the amount of spam received has only increased. According to estimates from MessageLabs, over 80 percent of emails received from 2005 to 2008 were spam [2].



The magnitude of the spam problem has not gone unnoticed by the US government. In 2003, the United States government drafted the Controlling the Assault of Non-Solicited Pornography and Marketing (CAN-SPAM) Act to address the issue. CAN-SPAM provided guidelines on unsolicited email practices and specified how unsolicited email could be sent legally. Unfortunately, compliance has been extremely low; therefore, the act has had virtually no effect on lowering the volume of spam.

On the other hand, CAN-SPAM allowed Internet service providers (ISPs) and web site owners to file lawsuits against spammers, resulting in fines and occasional jail sentences for convicted spammers. While lawsuits are certainly a way to fight back against spammers, given the vast number of spammers, suing an individual has a negligible effect on reducing the overall volume of spam, especially when lawsuits are brought regardless of the impact of the offense. Unfortunately, spammers have responded by taking greater measures to conceal their identities to avoid being detected. Clearly, other mechanisms are necessary to combat spam effectively.

One type of spam that represents a significant threat to individuals and companies alike is *phishing* spam. Phishing is an attempt to fraudulently acquire sensitive information by appearing to represent a trustworthy entity. Phishing spam often takes the form of emails appearing to be from a trusted financial institution with which the recipient does business. These emails are written to persuade the recipient to reveal confidential information such as online banking passwords, credit card numbers, or a social security number. Many victims of identity theft have been fooled into revealing sensitive information by phishing emails.

Current methods to combat spam before it reaches a user include *content-based filtering* at the recipient's email server as well as *blacklisting* email servers known to send only spam emails. Both measures reduce the annoyance of spam and the loss of employee productivity by decreasing spam emails arriving at employee inboxes. However, these strategies can also backfire. For example, content-based filtering has the unintended side effect of misclassifying legitimate email as spam.

Furthermore, filtering does nothing to reduce the volume of spam that is sent. When spammers know that a smaller percentage of emails are getting

past the spam filters to the intended recipients, they might compensate by sending more spam emails. Thus, content-based filtering may even increase the volume of spam sent!

Email servers that send only spam can be blacklisted to filter out all emails sent from them. Blacklisting differs from content-based filtering in that the filtering is done on email servers instead of on individual emails. Blacklisting is a more efficient filtering approach, but the disadvantage to blacklisting is that many email servers send both legitimate email as well as spam; blacklisting such a server would result in legitimate emails being misclassified as spam.

Current anti-spam methods share one common weakness—they are local; that is, they detect and filter out spam at a single location, which is the recipient's email server. Local anti-spam solutions are easy to maintain because a single administrator, usually the information technology group of the company or ISP, manages the process. But what could an analyst discern by examining how spam operates on a greater network level?

In this article, we investigate the spam problem using a global approach, which requires detection and monitoring of an entire network or at multiple locations within a network. By taking a global approach, an analyst can correlate data over multiple email servers, times, and locations to infer the behavior of spammers on a large scale, which can then be used to combat spam nearer to its source.

The best defense spammers have against anti-spam techniques is to send spam emails without being detected. So how do they do this? Consider the path of spam, illustrated in Figure 1. First, a spammer acquires email addresses on a web page using a *harvester*, which is a piece of software designed to visit web sites and extract email addresses from the HTML source code. Next, spam servers send emails to the acquired addresses. These can be servers that belong to the spammers, or they can be *zombie* computers, computers compromised by viruses or other malware that end up sending spam without their owners' knowledge. Finally, these spam emails make their way to the recipients' inbox or junk mail folder.

The address acquisition process, known as *harvesting*, is an often overlooked part of the spam problem. Malicious spammers typically take

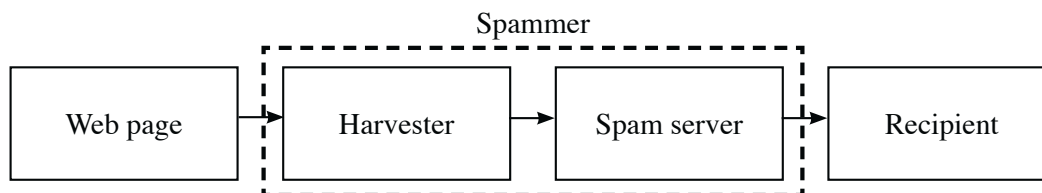


Figure 1: The path of spam from an email address on a web page to your inbox

measures to conceal their identities when sending spam. One common method is to use massive networks of compromised computers, known as *botnets*. However, studies have indicated that spammers do not take comparable precautions when harvesting [3], perhaps because harvesting is seen as a safer and more acceptable activity than sending spam. Hence, monitoring harvesting activity and tracking harvesters can be useful for identifying spammers. This is one of the goals of Project Honey Pot, created by anti-spam company Unspam Technologies, Inc. [4].

### Project Honey Pot

Project Honey Pot was started in 2004 to monitor harvesting and spamming activity via a network of decoy web pages set with trap email addresses, known as *honey pots*. These honey pots are embedded in the HTML source code of a web page and are invisible to human visitors. Harvesters looking for email addresses in HTML source code sometimes stumble across the trap addresses and acquire them. Harvesters can also be directed to trap addresses by links to honey pots from legitimate web sites that they also scan for email addresses.

Each time a honey pot is visited, the centralized Project Honey Pot server generates a unique trap email address. The visitor's IP address is associated with the trap email address and then recorded on the server. The email address embedded in the honey pot is unique, so only the visitor to that honey pot could have collected it. Because these trap email addresses are not published anywhere besides the honey pot, all emails received at these addresses are assumed to be spam.

Project Honey Pot provides a unique opportunity to investigate the social structure of spammers. It is normally very difficult to uncover anything at the spammer level because we cannot associate a spam email with a particular spammer. The "from" address can be easily spoofed, and the

spam served from a compromised computer has little association with the spammer. With Project Honey Pot each spam email is associated with the harvester that acquired the recipient's email address. When spammers fail to conceal their identities while harvesting, the IP address of the harvester is likely to be closely related to the actual location of the spammers.

Because each email received at a trap email address is associated with the harvester that acquired it, the identity of the spammer is revealed. As of March 2010, Project Honey Pot comprised over 48 million honey pots distributed all over the world [4]. The data collected by Project Honey Pot provides a global perspective on spam and makes it possible to investigate correlations over many spam servers and time periods.

### Discovering communities of spammers

As mentioned earlier, understanding the behavior of spammers on an expanded scale is one of the benefits of a global approach for fighting spam. But what do the social networks of spammers look like? In particular, how well organized are spammers? Do they operate alone, or in groups? Are there meaningful communities or organizations of spammers? Sending spam emails is profitable for spammers; otherwise, there wouldn't be so much spam. Can a business model be derived from the community structure of spammers? These questions can be answered using the data collected by Project Honey Pot and a technique known as *spectral clustering* [5].

The social network of spammers can be represented as a graph consisting of nodes and edges, as shown in Figure 2. The nodes correspond to spammers, and an edge between two nodes corresponds to a social relationship between the corresponding spammers. A social relationship can be inferred by the use of common resources or by similar behavior patterns over time. Communities

in a social network emerge by partitioning the graph into groups of nodes. Sets of nodes in the same group are highly similar and sets of nodes in different groups are not similar. Spectral clustering aims to minimize the normalized cut between groups, which is defined by

$$\text{Normalized cut} = \frac{\text{Sum of all edge weights between groups}}{\text{Sum of all edge weights within groups}}$$

For example, spectral clustering divides the graph shown in Figure 2 into the two communities indicated by the blue and green nodes, respectively. The groups revealed by spectral clustering correspond to communities in the social network. For these communities to be meaningful, the graph must be constructed so the edges between nodes correspond to actual relationships between spammers.

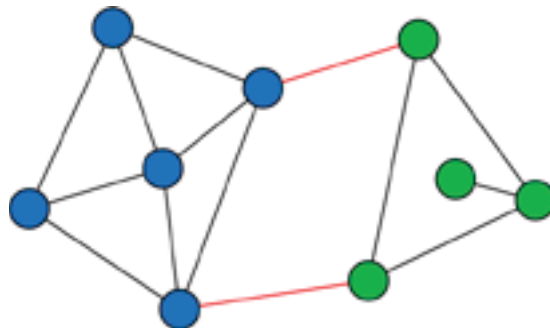


Figure 2: An example of a graph and its separation into two communities by spectral clustering

The main challenge in constructing the graph is choosing the edges and edge weights, because we cannot observe relationships among spammers. This problem does not arise in most other community detection studies. For example, in friendship or collaboration networks, users willingly participate in the study, and information on relationships among members of the network is readily available. However, for spammer network discovery, relationships between spammers are only inferred through correlations between behavior patterns. Two spammers who have high behavioral correlation are likely to be collaborating. This likelihood, which is treated as the strength of the relationship between these two spammers, can be used as the weight of the edge between the two corresponding nodes in the graph. For this research,

we investigate two types of behavioral correlation between spammers: correlation in spam server usage and temporal correlation.

### Correlation in spam server usage

Correlation in spam server usage between two spammers corresponds to common usage of a set of spam servers. Spammers typically try to conceal their identity by using spam servers that aren't traceable back to them, such as botnets. Thus spam servers can be viewed as resources for spammers, and common usage of a set of spam servers between two spammers translates into resource sharing, which suggests that the two spammers are collaborating. By constructing the graph using correlation in spam server usage between all active spammers over a period of time, many interesting communities of spammers are revealed, as shown in Figure 3.

Each node in the graph corresponds to a spammer, and the color and shape of a node indicates the community to which he or she belongs. Note that the majority of spammers belong in a large, loosely-connected community identified by the red nodes. These are the spammers who do not exhibit extremely high correlation with other spammers. Hence it is not a true community, but a collection of spammers who appear to be operating alone. The interesting communities are the smaller, tightly-connected ones surrounding the large red community. We believe that these nodes correspond to actual social communities of spammers working together and sharing substantial email server resources.

Reinforcing our belief is the observation that the discovered communities tend to divide into phishing and non-phishing communities, as shown in Figure 4. The shade of each node corresponds to the phishing level of each spammer, which is defined by

$$\text{Phishing level} = \frac{\text{Number of phishing emails sent}}{\text{Total number of emails sent}}$$

We denote spammers with high phishing levels as *phishers* and the rest as *non-phishers*. Notice that phishers tend to form communities with other phishers, and that non-phishers tend to form communities with other non-phishers. This is also evident from looking at the most frequent subject

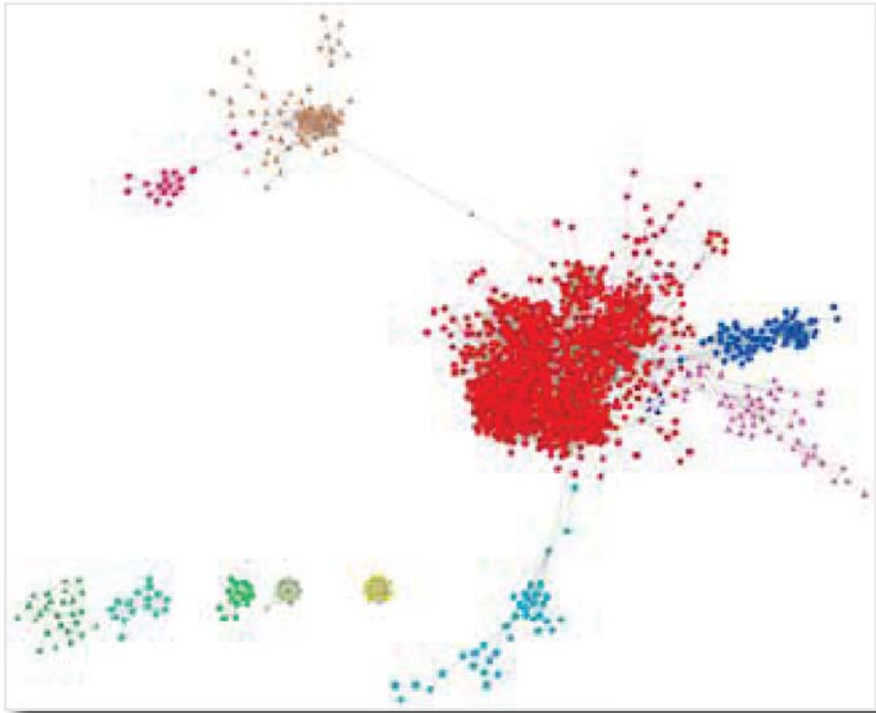


Figure 3: Community structure of spammers inferred by correlation in spam server usage in October 2006

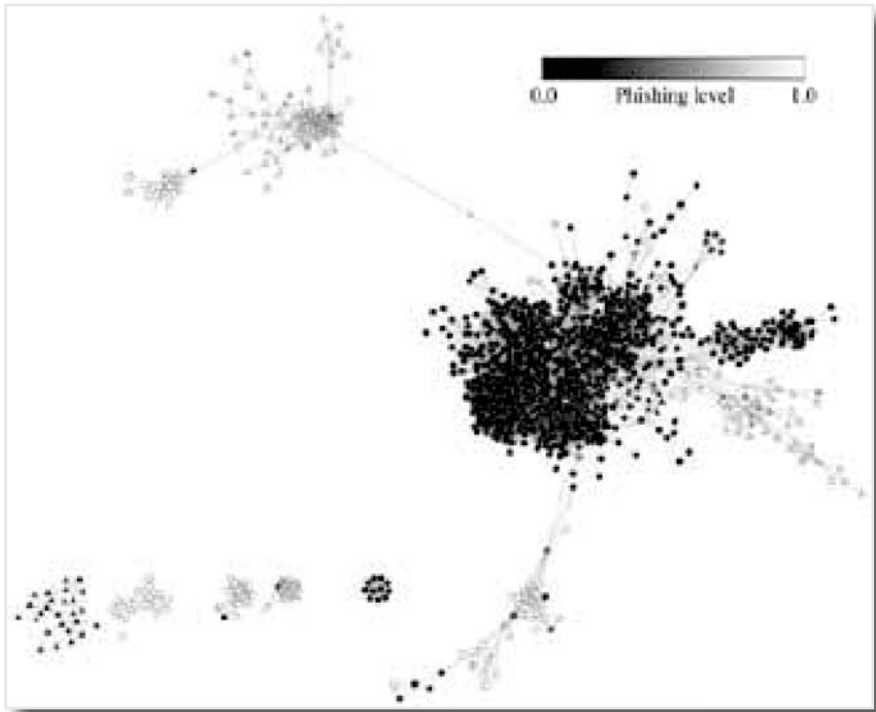


Figure 4: Alternate view of the same social network shown in Figure 3, shaded by phishing level

Table 1: Most common subject lines from a phishing and a non-phishing community (truncated to 50 characters by the Project Honey Pot database)

Phishing Community	Non-Phishing Community
Password Change Required	Make Money by Sharing Your Life with Friends and F
Question from eBay Member	Premiere Professional & Executive Registries Invit
Credit Union Online® \$50 Reward Survey	Texas Land/Golf is the Buzz
PayPal Account	Keys to Stock Market Success
PayPal Account - Suspicious Activity	An Entire Case of Fine Wine plus Exclusive Gift to

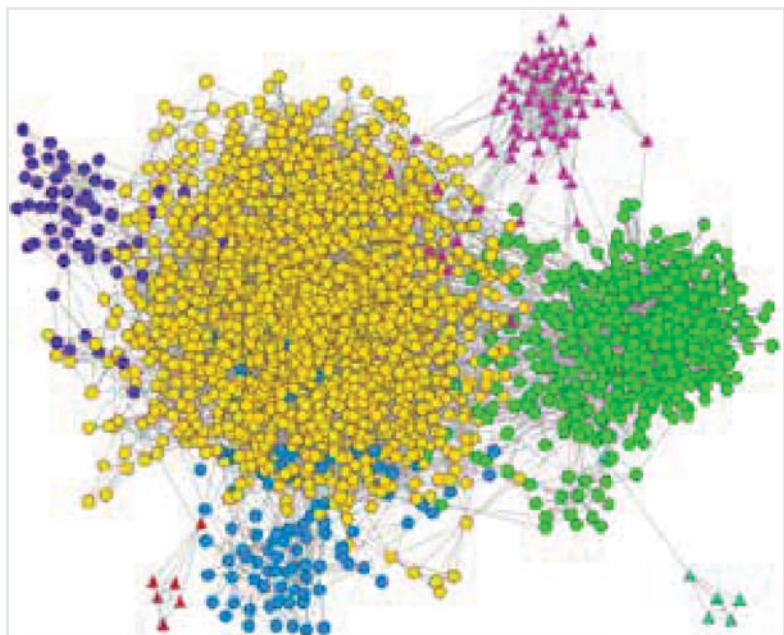
lines of emails from all spammers in a community. For example, the most frequent subject lines from both a phishing community, namely the orange community of triangular nodes at the top of Figure 3, and a non-phishing community, namely the blue community of circular nodes on the right of Figure 3, are listed in Table 1. Notice the distinct separation between phishing subject lines and non-phishing subject lines. The subject headings were not provided to the clustering algorithm and therefore confirm that server usage patterns alone can provide evidence of coordinated phishing behavior. We note that phishers tend to concentrate in small, tightly-connected communities. This observation provides empirical evidence that communities of phishing spammers are sharing resources, namely spam servers, among the community. This suggests that phishers tend to exist in isolated, well-organized social communities or teams.

### Temporal correlation

*Temporal correlation* refers to correlation of the times when emails were sent. High correlation is expected among spammers who are working together. Because we do not know the times when emails were sent, we correlate the times when emails were received. The community structure as revealed using temporal correlation is shown in Figure 5.

Again, the shape and color of a node represent the community that a particular spammer belongs to. Two large communities appear, and as before, they can be interpreted as loosely-connected communities of individuals who do not exhibit much correlation with each other. However, in the smaller communities, some interesting patterns emerge. In particular, we discover groups of spammers with nearly coherent temporal spamming behavior. Consider the group of ten spammers

Figure 5: Community structure of spammers inferred by temporal correlation in October 2006





whose temporal spamming behavior is shown in Figure 6, in which the horizontal axis corresponds to days in a month and the vertical axis corresponds to the number of emails sent each day. The figure consists of ten lines overlaid onto the same plot, with each line corresponding to the temporal spamming behavior of one spammer in the group.

How striking that the ten spammers in Figure 6 are sending almost identical numbers of emails over time! And how probable that they are working together and belong to an actual social community. These ten spammers, found in the community of dark-blue colored nodes in the top left of Figure 5, are especially interesting because they are among the heaviest spammers in the Project Honey Pot data set, where a heavy spammer denotes someone who sends a large number of spam emails. In addition to their highly coherent temporal behavior, these spammers also have IP addresses in the same block, indicating that they are operating from the same physical location, perhaps in the same building. Furthermore, these ten spammers' IP addresses are in the IP address range of a known rogue ISP, McColo Corp., which had been hosting and providing services for cybercriminals until it was taken down in November 2008 [6]. All of the abovementioned observations point to this group of spammers being very well-organized, and thus we conclude that they form a tight social community.

## Conclusions

Current methods of fighting spam are local and take place at the receiving end, which does not help to reduce the amount of network traffic consumed by spam emails. By studying spam from a global perspective using the data collected by Project Honey Pot, we were able to correlate the behavior of spammers, allowing us to identify different communities of spammers. We found that the majority of spammers appeared to be working alone, but a significant number of them appear to form communities or organizations. In particular, we discovered many small communities of spammers who predominantly sent phishing emails, likely attempting to acquire sensitive information to engage in identity theft. We also discovered several communities of spammers operating from the same physical location, suggesting strong social connections between these spammers.

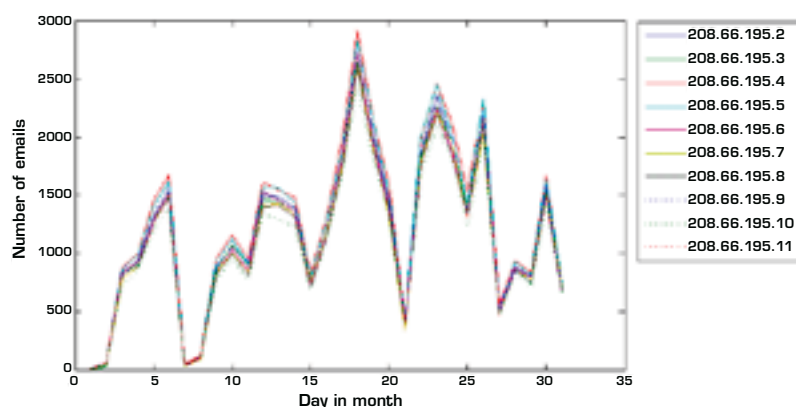


Figure 6: Temporal spamming behavior of group of ten spammers over the month of October 2006, by IP

By analyzing spam and spammer behavior from a global perspective, we were able to identify meaningful communities of spammers. The next step would be to use these findings to combat spam. Several avenues that could be pursued include identifying social cliques that could perhaps be linked to an organization and identifying important members of the social network who could be sued, which would have a much greater effect than randomly targeting spammers. There is also potential for online detection of communities; that is, updating the detected communities as emails are received. This would allow for a new method of spam filtering, not by content or blacklisting, but by behavioral patterns of spammers, which are less variable. Thus filtering by behavioral patterns has the potential to be more effective than existing filtering methods.

Although the problem of spam does not appear to be going away anytime soon, methods and tools for combating it are improving. Spectral clustering and network discovery can lead to insights into how spammers operate by revealing their social networks. The methods described in this paper might also be applied to discovery of illicit behavior patterns in other applications, such as financial transaction networks or chat room interaction networks. For additional details on our methods, the reader is referred to “Revealing Social Networks of Spammers Through Spectral Clustering” [7].

## References:

- [1] The Secret Cost of Spam. *Windows & .NET Magazine*. [Online]. Available: <http://www.itmanagement.com/whitepaper/the-secret-cost-of-spam/>, 2003.
- [2] MessageLabs Intelligence: 2008 Annual Security Report. Symantec Corp. [Online] Available: [http://www.messagelabs.com/download.get?filename=MLIReport\\_Annual\\_2008\\_FINAL.pdf](http://www.messagelabs.com/download.get?filename=MLIReport_Annual_2008_FINAL.pdf).
- [3] M. Prince, L. Holloway, E. Langheinrich, B. M. Dahl, and A. M. Keller. Understanding How Spammers Steal Your E-Mail Address: An Analysis of the First Six Months of Data from Project Honey Pot. In *Proc. 2nd Conf. Email and Anti-Spam*, 2005.
- [4] Project Honey Pot. Unspam Technologies Inc. [Online] Available: <http://www.projecthoneypot.org/>, 2009.
- [5] S. Yu and J. Shi. Multiclass Spectral Clustering. In *Proc. 9th IEEE Int. Conf. Computer Vision*, 2003.
- [6] J. Nazario. Third 'Bad ISP' Disappears—McColo Gone. Arbor Networks. [Online] Available: <http://asert.arbornetworks.com/2008/11/third-bad-isp-dissolves-mccolo-gone/>, 2008.
- [7] K. S. Xu, M. Kliger, Y. Chen, P. J. Woolf, and A. O. Hero III. Revealing Social Networks of Spammers Through Spectral Clustering. In *Proc. IEEE Int. Conf. Communications*, 2009.

# Challenges in Internet Geolocation, or *Where's Waldo Online?*



As more and more of our daily lives are conducted online, conventional tools and techniques used by law-enforcement and intelligence organizations for surveillance and monitoring have become less effective. The decentralization that distinguishes the Internet from traditional telephony networks and makes the internet so robust and resilient also makes it very difficult to identify its users. Unlike real-world interactions, participants of online transactions are often very hard to locate. A poster on a message board, a participant of an online funds transfer, or a sender for a particular email message could be connected to the Internet from anywhere, and it is difficult to determine the computer's precise location when that information is needed.

The process of determining the location of a network participant on the globe is called

## Challenges to geolocation

Geolocation on the Internet would be substantially less difficult if Internet protocol (IP) addresses corresponded to physical locations, much as area codes in phone numbers do. However, the Internet was designed to be fundamentally decentralized, without the rigid region-based routing hierarchy that existed in the original phone networks. Instead, Internet service providers (ISPs) cover overlapping geographic regions that, in some cases, can span entire continents. Hence an IP address, even when narrowed down to its issuing ISP, only provides very coarse-grain geographic information.

Since IP addresses are mostly opaque identifiers that provide little information on the location of a node, geolocation on the Internet requires going back to first principles. The basics of locating physical objects on the globe have been worked out in great detail over the last few centuries, and comprise triangulation (where bearings to known landmarks are used to determine location), multilateration (where time difference of arrival from a common emitter are used), and trilateration (where distances to known landmarks are used to determine location). Applying these approaches to wired wide-area computer networks poses significant challenges. First, bearings are not applicable, as wired networks do not support traditional notions of angles, grids, or even a Cartesian space. Second, measurements are inherently imprecise, as latencies on a network depend not only on the circuitous paths that packets follow over fiber networks (instead of a straight line from a lighthouse or satellite) but also on the queuing delays encountered in routers along the way. Finally, unlike the extensive lighthouse network or the GPS satellites that provide an almost ubiquitous coverage for navigation, the Internet lacks well-placed, well-known landmarks whose positions are known precisely. As a result, the Internet geolocation problem is akin to navigating

on a map where most lighthouses are not marked, in a world where light does not necessarily travel in a straight line or at a constant speed. Consequently, a naïve application of navigational geolocation techniques to Internet geolocation does not yield accurate or precise locations.

## Octant framework

Our group has been developing a general-purpose, comprehensive framework for geolocation called Octant. The key insight behind Octant is to view the geolocation process as solving a system of geographic constraints.

Octant aggressively extracts these constraints from network measurements, attaches a weight corresponding to the confidence associated with that constraint, and determines a feasible region in which the node of interest is expected to reside. This approach gains its accuracy through three novel techniques. Firstly, Octant can take advantage of negative information—information on where a node is not—in addition to positive information—information on where the node might be. Secondly, Octant utilizes available structural information about the network to extract additional geographic constraints from routers on the network path, thus compensating for the indirect and circuitous nature of routing paths on the Internet. Finally, Octant can reason in the presence of uncertainty by deriving constraints from landmarks whose positions are not known precisely, but are instead computed by Octant itself. The result is a system that can extract and combine all available position-related information to geolocate lighthouses and nodes alike.

## Geographic constraints

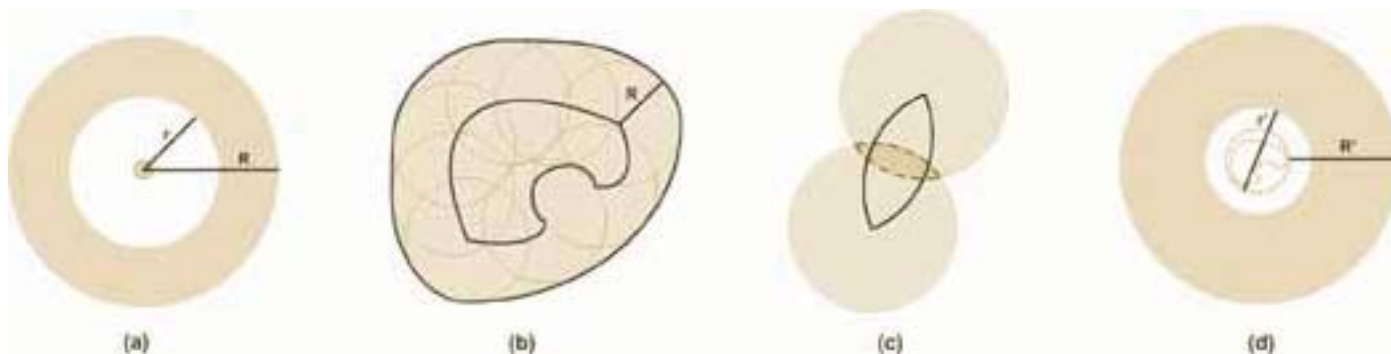
Constraints in Octant are geographic rules that describe where a node can or cannot be, relative to a landmark on the globe. The constraints are derived from network measurements between nodes and landmarks. These constraints can not only be of the positive form “node

$A$  is within  $x$  miles of Landmark  $L_1$ ,” but also encompass negative information of the form “node  $A$  is further than  $y$  miles from Landmark  $L_1$ .” Both kinds of constraints carry valuable information, and a comprehensive framework must be able to take advantage of both kinds of information.

Establishing suitable landmarks on the Internet is essential for generating precise constraints. Landmarks with known locations, such as nodes at universities and data-centers with well-established position, can serve as a basis for precise constraints, but are relatively few in number and distributed unevenly throughout the globe. To compensate for this, Octant co-opts nodes within the network fabric and uses them as additional landmarks. Since the positions of these nodes are not known, Octant first uses the well-established landmarks to geolocate these additional nodes, which are used in turn to geolocate the final node of interest. Extracting and using constraints based on these additional landmarks is non-trivial because the position of the landmark is typically uncertain, rather than a precise point.

In the simple case, where the location of a landmark is known with pinpoint accuracy, the two types of constraints combine to form an annulus centered on the landmark that describes the possible location of the node of interest. This case is illustrated in Figure 1a.

Octant enables meaningful extraction of constraint regions even when the position of the landmark is uncertain and consists of an irregular region. For a landmark  $k$  whose position estimate is  $\beta_k$ , a constraint that places the node within distance  $d$  from the landmark defines a region that consists of the union of all circles of radius  $d$  at all points inside  $\beta_k$ , as shown in Figure 1b. In contrast, a constraint that places the node further than distance  $d$  from the landmark can only safely rule out the intersection of all circles of radius  $d$  from



**Figure 1 (a – d): Comprehensive use of constraints in Octant.** The exact location of an IP address is usually available from the ISP that dynamically assigned that IP address to a particular machine. However, acquiring IP addresses from the ISPs leads to a scalability problem when monitoring many IP addresses: special relationships with potentially thousands of ISPs around the world could be required, or court orders from multiple jurisdictions—which might not even be available—could critically delay proceedings.

all points inside of  $\beta_k$ , regardless of where the landmark might actually be within  $\beta_k$ . This condition is illustrated in Figure 1c. A scalable Octant implementation may decide to approximate certain complex  $\beta_k$  with a simple bounding circle in order to keep the number of curves per region in check and thus gain scalability at the cost of modest error. Figure 1d illustrates the constraint approximation.

Given a set of constraints, a precise region can be efficiently computed geometrically by taking the intersection of positive constraints and subtracting the negative constraints. There are, however, many issues to solve before this approach can be used for practical geolocation on the Internet. In this solution, all constraints are weighted equally and the computed region is discrete; a point is either part of the solution space or it is not. A discrete solution strategy leads to a brittle system, as a single erroneous constraint can collapse the estimated location region down to the empty set. One strategy is to use only highly conservative constraints derived from the speed of light, bounding the maximum distance a packet can theoretically travel in a given time. We show later how to compute robust solutions that are resilient to error and measurement noise.

### Mapping latencies to distances

The network latency between a node and a landmark physically bounds their maximum geographical distance. A round-trip latency measurement of  $d$  milliseconds between a landmark and

a node can be translated into a distance constraint using the propagation delay of light in fiber, approximately  $2/3$  the speed of light. This yields a conservative constraint on node locations that can then be solved using the Octant framework to yield a sound estimated position for the node; such an estimate will never yield an infeasible ( $\emptyset$ ) solution. In practice, however, Internet paths deviate so much from great-circle distances that such constraints are so loose that they lead to very low precision.

Yet the correlation between latency measurements and real-world distances is typically better and tighter than constraints based on the speed of light. Octant calibrates each landmark by measuring its latencies to other landmarks when it is initialized, as well as periodically, to determine the correlation between network measurements derived from that landmark and real-world distances. The goal of the calibration step is to compute two tight bounds,  $R_L(d)$  and  $r_L(d)$ , for landmark  $L$  and latency measurement  $d$ . For a node  $A$  whose ping time to landmark  $L$  is  $d_A$ , Octant can derive the constraint  $r_L(d_A) \leq \|loc(L) - loc(A)\| \leq R_L(d_A)$ , bounding the node's distance from the landmark. In practice, this approach yields good results when there are sufficient landmarks that inter-landmark measurements approximate landmark-to-node measurements.

### Indirect routes

The preceding discussion made the simplifying assumption that route

lengths between landmarks and the node are proportional to great-circle distances. However, this is often not the case in practice, due to ISPs' use of policy routing based on business agreements. A geolocation system with a built-in assumption of proportionality would not be able to achieve good accuracy. Specifically, nodes might choose unexpectedly long and circuitous routes for certain IP addresses. This occurs often enough in practice that accurate geolocation requires a mechanism to compensate for its effects.

Octant addresses indirect routes by performing piecewise localization; that is, localizing routers on the network path from the landmarks to the node of interest in series and using routers localized on previous steps as additional landmarks. This approach yields much better results than using just end-to-end latencies because single-hop paths tend to be less circuitous than multi-hop paths. Since Octant can perform localization based solely on round-trip timings, localizing routers does not require any additional code to be deployed within the network.

### Handling uncertainty

With the many avenues to extract geographic constraints from the network, a mechanism to handle and filter out erroneous constraints is critical for maintaining high localization accuracy. Octant uses a weight assignment mechanism to characterize the confidence of different constraints. A constraint's relative weight value amplifies or

dampens its contribution in estimating the location region of the node of interest.

For latency-based constraints, landmarks farther from a node are less trustworthy than those that are nearby. The simple intuition behind this relationship is that latency in far-away nodes increases due to the higher probability of data packets traversing indirect, meandering routes or highly congested paths. In Octant, every constraint has an associated confidence level, which is tracked through the constraint-satisfaction process. This process yields not only the set of feasible points where the node can potentially lie, but also be the associated probability for the node residing at each point.


In the absence of weights, regions can be combined via intersection operations, leading to a discrete solution for a location estimate—the node is either within a region, or it lies outside. The introduction of weights changes the implementation of location estimates. When combining two regions, Octant determines all possible resulting regions via intersections, and overlapping regions are assigned the sum of their component weights. Non-overlapping regions are retained with their original weights. This condition is illustrated in Figure 2. The final estimated location region is computed by taking the union of all regions, sorted by weight, such that they

exceed a desired weight or region size threshold.

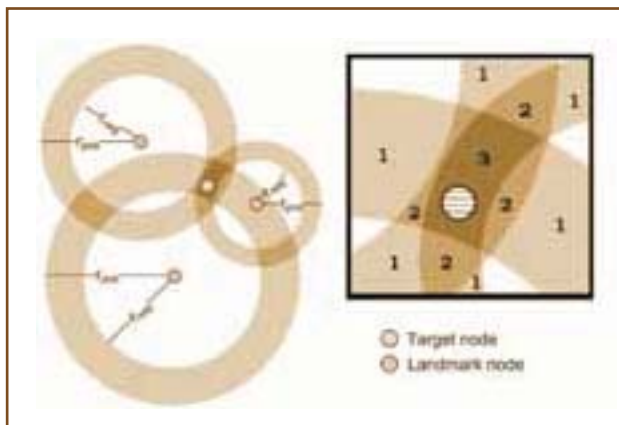
### Future directions

The existing Octant framework is accurate and comprehensive, but it was designed to perform on-demand network measurements to geolocate a single node at a time. Consequently, Octant relies almost entirely on active network measurements performed on demand; it does not perform any pre-computation, and it does not take into account any long-term network effects or perform long-term measurements to aid geolocation. On-demand probing in large-scale deployments poses some additional challenges. Firstly, given the security consciousness of modern network management policies, on-demand network probing is considered highly undesirable when performed at a large scale, in bursts, and to arbitrary clients. Such probes can be misclassified, and their delivery may be intentionally delayed or dropped in response. Secondly, the expensive constraint evaluation computation is dependent on the constraints extracted from the on-demand network measurements. The constraint evaluation must therefore be performed at runtime. Finally, on-demand probing is observable, warning the node of interest of possible surveillance.

ing a global mapping of IP address ranges to these key points to provide a location estimate of every IP address. We can perform more precise geolocation on specific nodes with fewer on-demand probes by leveraging this global mapping, or use the global mapping to identify neighbors that can be probed as proxies if the node is sensitive to probing.

There are, of course, associated privacy concerns with geolocation. Such technologies point to potential threats for unauthorized personnel to identify the location of sensitive assets, suggesting the need for further work on understanding the theoretical limits of geolocation techniques and developing countermeasures where necessary. 

We are currently investigating geolocation techniques based on passive measurements. Our research, conducted in collaboration with researchers at Duke University and Akamai Technologies, centers on identifying key ingress points in the network where end-users are connected to the Internet core, accurately locating these points on the globe using periodic measurements, and creat-



**Figure 2:** Octant assigns weights to constraints based on their inherent accuracy. Overlapping regions are given the sum of the weights of their components.



## Clumps, Hoops, and Bubbles— Moving Beyond Clustering in the Analysis of Data

**E**xploratory data analysis is the search for structure in complex data. In many cases the origins and statistical properties of a data set may be poorly understood. Data may also be incomplete or noisy. How can we understand the information contained in such a data set when we don't even know what questions to ask?

One of the most common first steps in data analysis is to cluster the data points—that is, to look for groups of data points that appear to have some set of characteristics in common. Data clusters are statistical features that can be discovered by a computer and then investigated further. In a topological sense, however, data clusters are the simplest type of structure—a connected mass of points.

The rapidly developing new field of topological data analysis has given us algorithms that can be thought of as a higher-dimensional analogue to data clustering. This paper will explain some of these topological methods and give examples of how they have been used.



**Figure 1:** This squirrel will kill your plants and not feel bad about it

## Motivating example: Bushy-tailed tree rats

Let’s face it—squirrels are the terrorists of the animal kingdom. Given acres and acres of dirt to dig in, they will invariably jump into the nearest flowerpot and uproot your beloved geraniums. In the fall, they sit in trees and throw half-eaten acorns at you.<sup>1</sup>

Suppose a naturalist marks all the squirrels in a wildlife preserve with electronic tags that record their locations every hour. (Experiments like this have been done with zebras and whales [1], [2]). Each data point would consist of the geographic coordinates and ID of the squirrel. If there were two populations of squirrels living in opposite corners of the preserve (Figure 2), that fact could be discovered by any one of a number of data clustering algorithms.

Now, suppose you want to detect a lack of data points in a region. For example, suppose that a fox lives in the center of the nature preserve and preys on the squirrels. Therefore, no squirrels can be found in that region of the park (Figure 3). How could a data clustering algorithm detect this avoidance behavior?

In fact, what you would hope to observe is a sort of *hole* in the squirrel location data. While data clustering algorithms detect connected components in a data set, a hole is the *absence* of a connected component. Topologists have formalized this notion of “holes” in arbitrary dimensions.

Studying the topological properties of discrete data is difficult for several reasons. The traditional algorithms for analyzing topology tend to be slow, and the computations tend to be sensitive to noise. In recent years, however, mathematicians and computer scientists have developed several new algorithms that are taking exploratory data analysis into new dimensions.

## Algebraic topology in pictures

At a basic level, topology is a more *lenient* version of geometry. Topology is often called “rubber sheet geometry,” because spaces that can be continuously transformed into one another are topologically the same. For example, to a topologist a square and a circle are equivalent (a topologist would say “homeomorphic”) because there is a way to pair every point on a square with a point on a circle in such a way that points that are close together on the square are paired with points that are close together on the circle (and vice versa). On the other hand, a figure-eight is not topologically equivalent to a circle. One way to see this difference is by removing any one point from a circle, which leaves one connected arc. But removing the intersecting point from a figure-eight leaves two disconnected arcs.

Unfortunately, deciding when two spaces are topologically equivalent (homeomorphic) is not easy because it is difficult to find a way to pair the points of two spaces. Even more difficult is to prove that two spaces are *not*

homeomorphic. Algebraic topology was developed to compare the properties of topological spaces without dealing with the spaces directly. Algebraic topologists have defined **topological invariants** that label every topological space with some more concrete mathematical object, such as a number or a group. The key property of an invariant is that spaces that are topologically equivalent must be assigned the same label. If two spaces have different labels, you know for sure they are topologically different. For example, a topological invariant would have to give the same label to a circle and a square.



**Figure 2:** Data points are represented by squirrel shapes. The data points form two distinct clusters. These clusters represent an interesting bit of statistical structure that can be further explored. Refuge map is adapted from [3].



**Figure 3:** In this example, the squirrels seem to avoid the region in the center. This is also an interesting bit of structure in the data. But how do you discover or interpret the absence of data points in a region?

<sup>1</sup> This example is completely made up, and the author has absolutely no expertise in squirrel behavior or biology





**Figure 4:** To a topologist a coffee cup and a donut are equivalent objects. They are both solid three-dimensional objects with one hole.

The simplest topological invariants are the **Betti numbers**. Betti numbers do have a precise mathematical definition, but it is possible to explain the idea in pictures. For now, focus instead in terms of pictures. Suppose you have a topological space, call it  $X$ . Intuitively, the  $k$ th Betti number,  $\beta_k$ , counts the number of  $k$ -dimensional “holes,” and the zeroth Betti number,  $\beta_0$  counts the number of connected components in  $X$ .

For example, a square and a circle both have  $\beta_1 = 1$  since they both

have a single one-dimensional hole. A figure-eight, on the other hand, has  $\beta_1 = 2$  because it has two distinct one-dimensional holes. Moving up a dimension, a sphere has  $\beta_1 = 0$ , but  $\beta_2 = 1$ . It has no one-dimensional holes, but does have a void inside it. Figure 5 shows several common household objects along with their Betti numbers. The zeroth Betti number,  $\beta_0$ , corresponds to the number of connected components in the space,  $X$ . In this sense, computing the zeroth Betti number is analogous to doing data clustering.

### Triangulating spaces

Most interesting topological spaces are continuous objects. A collection of data, on the other hand, is just a bunch of discrete points. Alone, the data points have no interesting topology. To study a data set topologically, one has to assume that the data points are sampled from some underlying space and build a continuous structure on the points. Usually, this amounts to building what topologists call a **simplicial complex**. Figure 6 illustrates the difference between a continuous space, discrete data points, and a triangulation.

**Fact: A computer can compute the Betti numbers of a simplicial complex.**

A simplicial complex is a collection of objects called simplices. A 0-simplex is a point, written  $[a_0]$ . A 1-simplex can be thought of as a line connecting two 0-simplices, written  $[a_0 a_1]$ , and a 2-simplex is a triangle with 0-simplices at each vertex and 1-simplices as edges, written  $[a_0 a_1 a_2]$ . Any  $k + 1$  points  $a_0, a_1, \dots, a_k$  can define a  $k$ -simplex  $[a_0 a_1 \dots a_k]$  whose faces are  $(k - 1)$ -simplices. Figure 7 illustrates some low-dimensional simplices and an example of a simplicial complex.

See references [4] and [5] for more in-depth mathematical background. Readers familiar with algebraic topology are reminded that the  $k$ th Betti number is equal to the rank of the free part of the  $k$ th homology group. As such, for some spaces the  $k$ th Betti number’s value will depend on the ring over which the computation is performed. In this paper all computations are over a field, so no torsion parts ever appear.

### Persistent homology

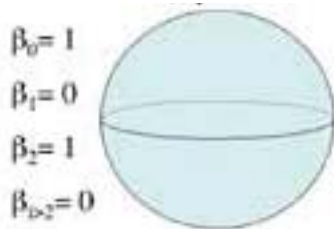
Look at the collection of data points in Figure 8(a). You probably see that the points lie roughly in a circle. Or you could say that there is a hole in the middle of the data. How would you write a computer program that could detect the presence of the hole? You would probably start triangulating the data by drawing lines between pairs of points that are close together and filling in triangles when three points are close together. You could pick some threshold  $T$  and use the rule:

- if  $|x_i - x_j| < T$  then add the 1-simplex  $[x_i x_j]$
- if  $|x_i - x_j|, |x_j - x_k|, |x_i - x_k| < T$  then add the 2-simplex  $[x_i x_j x_k]$

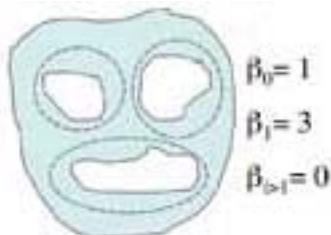
**A solid 2-dimensional blob**



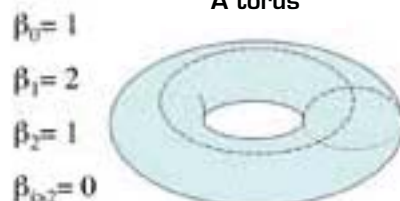
**A sphere**



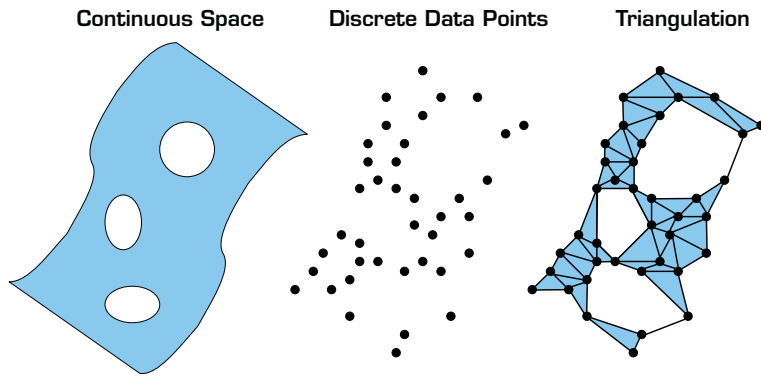
**A 2D blob with three holes**



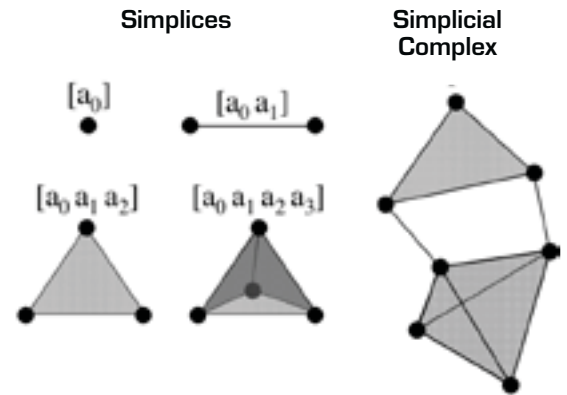
**A torus**



**Figure 5:** Some simple topological spaces and their Betti numbers. Intuitively, the  $k$ th Betti number counts the number of  $k$ -dimensional holes in the space.



**Figure 6:** A set of discrete points has no interesting topology. Studying the triangulation leads to finding the interesting topology.



**Figure 7:** Low-dimensional simplices and a Simplicial Complex

Figures 8(b-e) show triangulations constructed using such a procedure for several different threshold values. Since the threshold is increasing, each triangulation in the sequence is a subset of the next. One would hope that computing the Betti numbers of these triangulations would give the Betti numbers of a circle, namely  $\beta_1 = 1$ . As it turns out, every triangulation pictured gives the wrong answer. Due to the noise in the data, as the threshold  $T$  increases, data points get connected, forming loops. The loops generated by the noise are quickly filled in with triangles as  $T$  continues to increase, but new loops form, preventing us from obtaining a space with  $\beta_1 = 1$ .

It would be ideal to distinguish between those loops that form and are quickly filled in and those loops that persist for a long time as  $T$  increases. This is exactly what persistent homology allows us to do. It tracks individual

topological features (like loops) through a nested sequence of spaces and discovers when they appear and when they are filled in. Topological features that form and quickly disappear are considered the product of noise, but the features that persist from early in the sequence to the end are more fundamental and interesting.

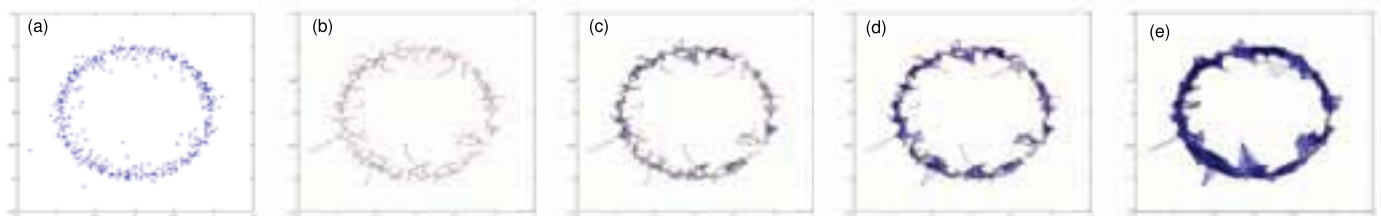
Suppose a loop first forms in the space constructed with threshold value  $T = b$ , and first gets filled in at threshold value  $T = d$ . Then the **persistence interval** for that loop is  $(b, d)$ . The same is true for higher dimensional features.

### Visualizing persistence

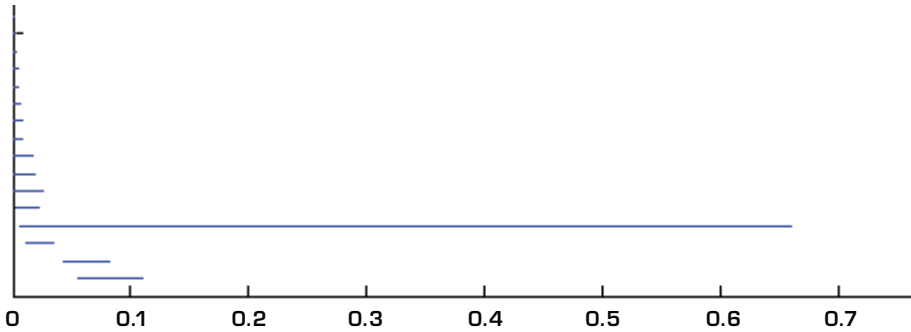
There are several ways to visually represent persistence information. One popular visual representation is called the **persistence barcode**. In a persistence barcode, each topological feature that arises in the sequence of spaces is represented by a horizontal line.

Topological features that persist through most of the sequence are represented by long lines, and features that appear and are quickly filled in (noise) are represented by short lines. The horizontal axis in a persistence barcode is labeled with the threshold values that were used to build the triangulations.

Figure 9 shows the *Betti*<sub>1</sub> persistence barcode for the data in Figure 8. There are several short blue lines corresponding to loops that formed and were quickly filled in as the threshold used to construct the triangulations increased. The long blue line corresponds to the overall circular structure of the data (or rather the hole in the middle of the data). The values on the horizontal axis correspond to threshold values. This example is just the barcode for one-dimensional topological features (loops). For any space there are also persistence barcodes for topological features in dimension 0



**Figure 8(a-e):** A nested sequence of triangulations of a set of data points. Because the data points form what looks like a circle, it seems reasonable to compute the Betti number  $\beta_1 = 1$ . In fact, computing the first Betti number for every triangulation in the sequence gives the wrong answer. This outcome shows how noise can be a problem in topological computations.



**Figure 9:** The persistence barcode for the data pictured in Figure 8. The short blue lines correspond to loops that appeared and were quickly filled in as the threshold increased (noise), and the long line corresponds to the circular structure of the data.

(clumps), dimension 2 (bubbles), and higher dimensions. The barcode for any dimension higher than the dimension of the space will always be empty.

All of the persistence results presented here will be expressed in terms of barcodes, so a simpler example is worth a closer look. Figure 10 shows a popular example of a sequence of nested spaces.

**Computing homology**

If a space can be shown as a simplicial complex, then a computer can compute its Betti numbers. The traditional algorithm for doing this computation

uses linear algebra (a lot of matrices). To compute the  $k$ th Betti number,  $\beta_k$ , of a simplicial complex with  $N$   $k$ -dimensional simplices, the computer needs to deal with an  $N \times N$  matrix. The complexity of the algorithm is polynomial in  $N$ , which presents a problem when working with data sets with millions, or even thousands, of data points.

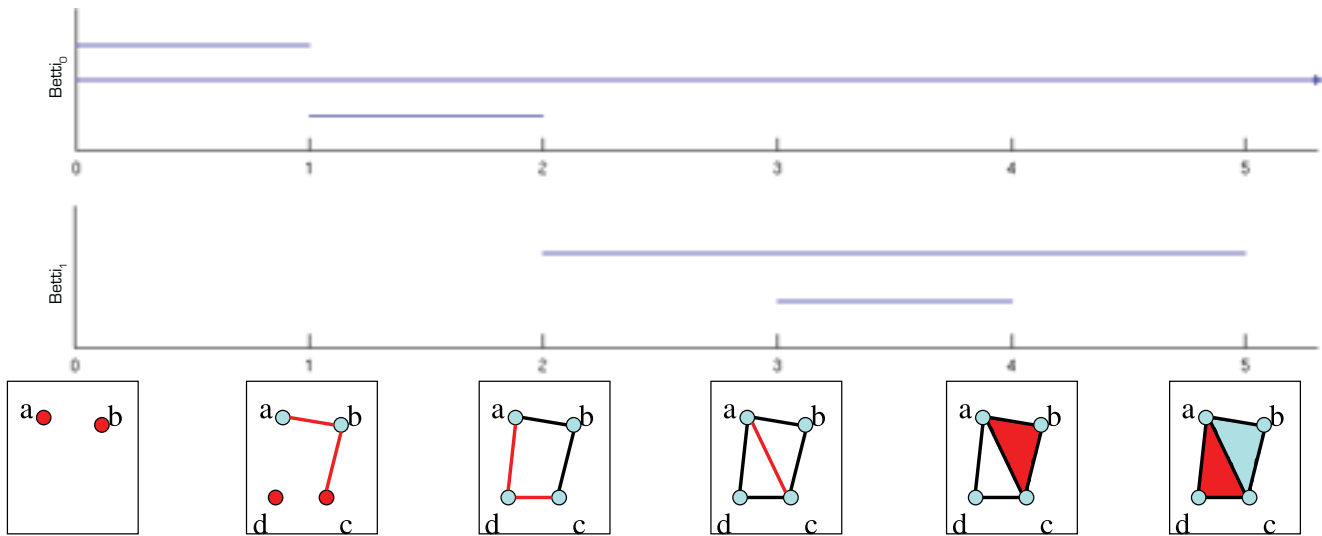
It turns out that the matrices used to compute Betti numbers have a lot of structure, and both they and the simplicial complex itself can be simplified to give more efficient computations. The Computational Homology Project

(CHomP) [6] has developed a variety of algorithms and software for efficiently computing Betti numbers, homology, and functions on homology. Note that CHomP works with cubical simplices instead of simplicial complexes. The theory is equivalent, but sometimes a computer can more naturally represent a space in terms of squares and cubes than as triangles and tetrahedra.

One type of simplification developed by CHomP researchers that has an effective visual interpretation is the use of “reduction.” It turns out that often a lot of the simplices in a simplicial complex can be collapsed without affecting the homology or Betti numbers of the space. Since the homology computation can be polynomial in the number of simplices, pre-processing to reduce the number of simplices has a big payoff.

*Computing persistent homology*

The computations done by CHomP all deal with a single space or function. As already mentioned, Betti number



**Figure 10:** A simple example of a nested sequence of spaces and the corresponding persistence barcodes for dimensions 0 and 1. At  $T = 0$  there are two disconnected points, so there are two lines in the  $Betti_0$  persistence barcode. At  $T = 1$  those two points become connected, so only one of the lines in the barcode persists. A new disconnected point appears, though, so a new line in the  $Betti_0$  barcode begins at  $T = 1$ . At  $T = 2$  the first loop forms, so a line appears in the  $Betti_1$  barcode. All of the points are connected at this point. From  $T = 2$  on there is only one line in the  $Betti_0$  barcode. At  $T = 3$  the loop is divided into two distinct loops, so a new line appears in the  $Betti_1$  barcode. At  $T = 4$  one of the extra loops is filled in, so the second line in the  $Betti_1$  barcode has ended. At  $T = 5$  all loops are filled in, so all lines in the  $Betti_1$  barcode have ended.

computations can be very sensitive to noise, and persistent homology can be used to distinguish between important topological features and features arising from noise. An algorithm to compute persistent homology over the finite field  $\mathbf{Z}_2$  was first published by Edelsbrunner, Letscher, and Zomorodian in 2000 [7]. A more general approach and algorithm along with a more powerful mathematical context for the algorithm was published by Zomorodian and Carlsson in 2004 [8].

The persistence algorithm takes a nested sequence of simplicial complexes and generates a collection of persistence intervals. The persistence intervals can be displayed as barcodes or by using other visualization techniques.

Restricting the computation to homology over fields makes certain shortcuts possible in the linear algebra. Also, instead of performing separate computations for each space in the nested sequence of spaces, the algorithm actually does a single homology computation that encodes all the information about where in the sequence different simplices appear.

The running time of the persistence algorithm tends to grow linearly with the number of simplices. The worst-case complexity is still polynomial, but performance tends to be much better than that in practice. The most serious computational problem is due to the number of simplices that appear as the sequence of triangulations is constructed. Therefore, the most practical use of the persistence algorithm is to employ a traditional data clustering algorithm to identify connected components, and then compute the persistent homology of each cluster individually.

Other approaches have been taken to limit the number of simplices necessary to compute persistence. For example, a type of simplicial complex called a **witness complex** reduces the

number of data points one starts with and builds a more efficient triangulation [9]. Investigations are underway to make the CHomP-style reduction compatible with the persistence algorithm.

### Software tools

The CHomP tools are available from Dartmouth [6]. The persistent homology tool, **Plex**, is a collection of MATLAB modules and scripts released by researchers at Stanford [10]. It supports the computation and visualization of nested sequences of simplicial complexes and persistent homology. The Stanford researchers have recently released a new Java-based implementation called **JPlex**. All of the persistence barcodes and 3D simplicial complexes in this paper were generated using Plex and the associated MATLAB scripts.

### Homology and statistics

A persistence barcode or set of persistence intervals generated from a data set is a statistic. In all of the examples in this paper, this statistic is evaluated in a fairly qualitative way by looking for long lines in the persistence barcode. However, a more objective way to distinguish between interesting topological features and noise is necessary. Questions like, “How long does a persistence interval need to be before being considered interesting?” and “How sensitive is a persistence barcode to noise?” must be answered.

These problems are just beginning to be addressed. For example, in [11] Bubenik and Kim compute the expected persistence barcodes for certain probability distributions on circular and spherical spaces. In [12] Cohen-Steiner, Edelsbrunner, and Harer present results on the stability of persistence diagrams of functions (something not addressed here). In the process they define a function for measuring the “distance”

between persistence diagrams. Also, the properties of a persistence barcode are very dependent on the method used to triangulate the data and generate the nested sequence of simplicial complexes. In [13] de Silva and Ghrist prove a relationship between the Rips and Čech complex of a data set. In [14] Chazal and Oudot study the relationship between Rips, Čech, and witness complexes and their effects on persistence computations.

## Applications and examples

Persistent homology has proven useful for extracting topological information from discrete noisy data. The key properties that make it so useful are its ability to tie together topological features appearing on different scales and the existence of fast algorithms to compute it. Persistent homology techniques have been applied to a number of problems including natural image analysis [15], molecular protein shapes [16], surface description [17], and sensor network coverage [13].

Much of the research in this area has been supported by the Defense Advanced Research Projects Agency (DARPA) Topological Data Analysis (TDA) and Sensor Topology for Minimal Planning (SToMP) programs. Robert Ghrist’s recent article [18] contains a survey of some results from the TDA program.

An example of my own application of persistent homology appears later in this article and in more detail in [19]. The general procedure used in my example is:

- Start with a data set
- Define a metric (distance function) on the data points
- Build a nested sequence of simplicial complexes based on the metric
- Use a persistence algorithm to compute persistence barcodes
- Interpret the results

### Encounter traces

The performance of wireless networks with mobile nodes is influenced by the mobility of the nodes. Unfortunately, node mobility is fantastically complicated. Researchers have focused instead on the node encounter patterns that the mobility produces. To this end, there have been several experiments that tag people or animals with wireless nodes (small short-range Bluetooth radios) that record which other nodes they come in contact with and when. These experiments produce **encounter traces**, a series of data points, each consisting of the encounter time and the IDs of the two nodes involved.

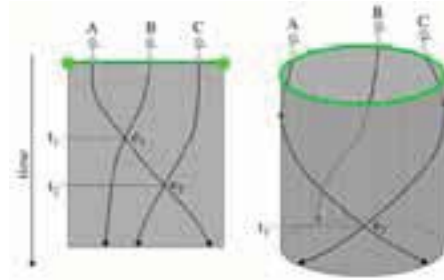
Encounter trace experiments include the famous Hagggle project experiments [20] and a student experiment at UT-Austin [21]. Wireless LAN traces such as the MIT trace [22], the UCSD trace [23], and the Dartmouth trace [24] are also commonly repurposed for use as encounter traces [25].

The data points in an encounter trace consist of the IDs of the two wireless nodes involved and the time of the encounter. For example, a section of the trace could look like:

time	Node ID 1	Node ID 2
9:42:30	20	12
9:47:01	72	31
9:47:21	58	20
10:02:55	64	45
...	...	...

These data points provide very little information. In particular, there is not any explicit information about the locations of the nodes. Persistent homology techniques are used to deduce information about the topology of the space the nodes live in from the encounter trace. The same techniques can be used to detect certain changes in the space.

Surprisingly, physical information can be deduced from data points that



**Figure 11:** The topology of the space affects the type of encounter patterns that are possible. If the space is like a line, node A cannot encounter node C without one of them encountering node B. If the space is like a loop, nodes A and C can encounter each other without encountering node B.

do not even contain relative position information. The concept makes some intuitive sense, however, when considering Figure 11, which illustrates how the topology of a space has an effect on the types of encounter patterns that are possible.

### Defining a metric on an encounter trace

Building a weighted graph on the set of encounters will give rise to a metric. This metric will be different from the Euclidean metric associated with the physical space of the experiment.

Assume that an encounter trace contains  $N$  data points of the form:

$$\begin{aligned}
 e_1 &= (t_1, \text{nodeA}_1, \text{nodeB}_1) \\
 e_2 &= (t_2, \text{nodeA}_2, \text{nodeB}_2) \\
 &\vdots \\
 &\vdots \\
 e_N &= (t_N, \text{nodeA}_N, \text{nodeB}_N)
 \end{aligned}$$

Here  $e_i$  represents the  $i$ th encounter, consisting of  $t_i$ , the time the encounter took place, and  $\text{nodeA}_i$  and  $\text{nodeB}_i$ , the two nodes involved in the encounter. Some maximum node velocity is assumed  $v_{max}$ .

Construct a weighted graph  $G$ , in which the vertices correspond to the encounters  $\{e_i\}_{i=1..N}$ . The idea is summarized in Figure 12. Suppose two encounters,  $e_i$  and  $e_j$ , have a node in

common. These encounters occurred at two particular points in space,  $\bar{x}_i$  and  $\bar{x}_j$ . The locations  $\bar{x}_i$  or  $\bar{x}_j$  are not known, but if the two encounters are close in time, then they must be close in space. It can be deduced:

$$|t_i - t_j| < T \Rightarrow |x_i - x_j| < T \cdot v_{max}$$

Therefore, if two encounters  $e_i$  and  $e_j$  have a node in common, they can be connected with an edge with weight  $|t_i - t_j|$ . This allows us to define a metric on the set of encounters

$$d(e_i, e_j) = \begin{cases} d_G(e_i, e_j) & \text{if } e_i \text{ and } e_j \text{ are connected in } G \\ \infty & \text{otherwise} \end{cases}$$

where  $d_G(e_i, e_j)$  is the minimum distance between the vertices  $e_i$  and  $e_j$  in the weighted graph  $G$ .

The reverse of the implication above is not valid. Two encounters may happen at the same physical spot but be far apart in time. Because of this, it is better to think of the space being studied as the physical space crossed with time. This idea was evident in Figure 12. For example, if the nodes are moving on a circle  $X = S^1 \subseteq \mathbb{R}^2$  with coordinates  $(x,y)$ , then the space whose topology should be reconstructed is a cylinder with coordinates  $(x,y,t)$ .  $X \times \mathbb{R}^+$ , which is homologically equivalent to  $X$ , so the Betti numbers of the product space being reconstructed will be the same as the Betti numbers of  $X$ .

### Building a witness complex

A set of data points  $\{e_i\}_{i=1..N}$  with a metric on them is now in place. There are a variety of ways to build a nested sequence of simplicial complexes from these data, but building the witness complex in the manner of de Silva and Carlsson [9] seems to give the most efficient triangulation and the best results.

The concept of the witness complex is based on the Delaunay triangulation [26]. The first step is to select a subset of **landmark** data points to use in the

analysis. The remaining data points are used to decide which of the landmarks to connect. The construction incorporates a variable threshold,  $T$ , that can be used to build a nested sequence of simplicial complexes.

### Experiments and results

The first step focuses on encounter traces generated from simulations. This access allows for the addition of certain characteristics to the experiment, which will allow for observation of how the changes manifest themselves in the results.

#### One-dimensional experiments

Remember the squirrel example at the beginning of this paper. Since squirrels can move pretty much freely around a two-dimensional area, their encounter traces will be a little complex. To start with something simpler, consider an imaginary example involving the squirrel's harmless cousin the naked mole rat (Figure 13). Naked mole rats spend their lives in networks of underground tunnels. Understanding the naked mole rats' burrowing habits would require either excavating their burrow, and destroying it in the process, or gathering an encounter trace from the mole rats themselves. The topological connectivity of the burrow could be discovered as a result of the encounter trace, and changes could be detected as the mole rats extend

some parts and abandon others. Such research is of particular interest given the prominent role naked mole rats have played in movies [27] and television programs [28].

Studying the encounter traces of naked mole rats is simpler because the tunnels they live in are effectively one-dimensional spaces. Two mole rats cannot avoid each other when passing in a tunnel.

Our first experiments are simulations of nodes (mole rats) doing random walks in one-dimensional spaces. Encounters are recorded the moment two nodes pass each other. A simple event-driven simulator was built to generate these data.

Compare three types of ID experiments:

- A line segment
- A single loop
- A multi-loop

The line segment experiment used 50 nodes following random walks. Topologically, this space has one connected component and no higher-dimensional topological features, so  $\beta_0 = 1$  and  $\beta_k = 0$  for  $k > 0$ . Since the fully contractible topology always results from setting the witness complex threshold high enough, it is difficult to quantify how well the method is working for this type of space.



**Figure 13:** Naked mole rats live in networks of underground tunnels. These tunnels are effectively one-dimensional spaces. This is what the squirrel in Figure 1 would look like without fur.

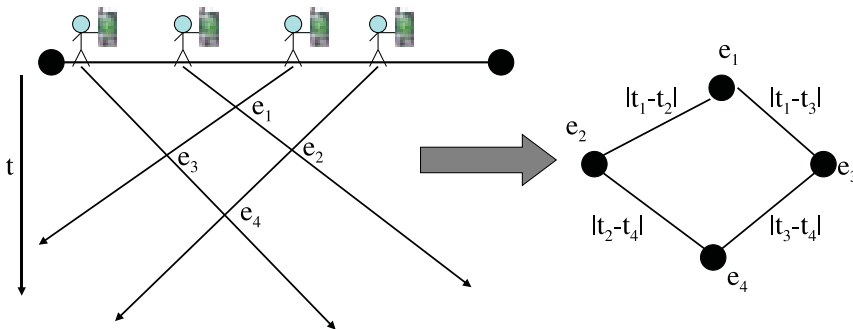
In the single-loop experiments, 50 nodes followed random walks in a circular space. This space also has one connected component, so  $\beta_0 = 1$ , but it has a single non-trivial 1-cycle, so  $\beta_1 = 1$ . By building a filtered witness complex and computing the persistent homology, the correct Betti numbers are recovered 100 percent of the time.

In the multi-loop experiments, adding 50 nodes per extra loop tends to generate enough encounters to reconstruct the spatial topology. The node mobility is the same as before. The correct Betti numbers are  $\beta_0 = 1$ , since there is one connected component, and  $\beta_1 = l$ , where  $l$  is the number of loops. The same technique as before correctly recovered this information for all two- and three-loop examples attempted. Figure 14 shows a witness complex and persistence barcode for a two-loop experiment.

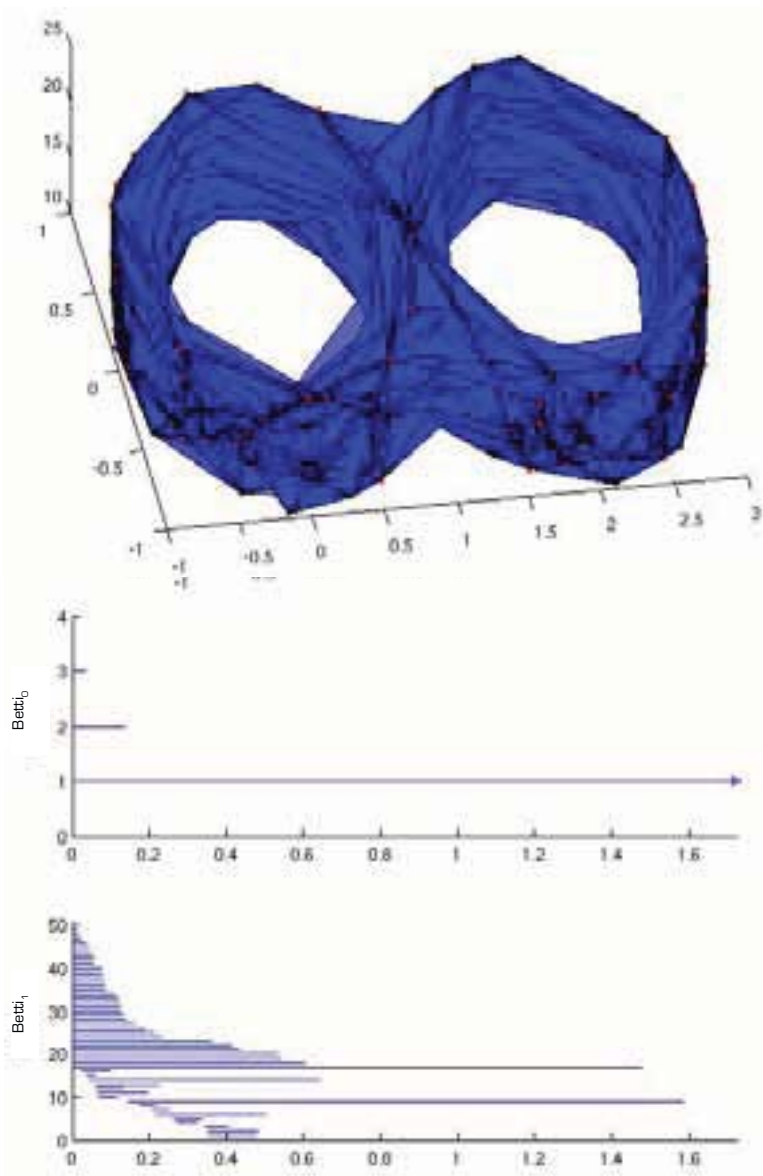
### Interpreting the results

Since these data were generated from controlled simulations, it was known in advance how to interpret the results. The one-dimensional topological features discovered correspond to loops in the space the nodes live in.

In general, interpreting persistence results is not so straightforward. Just as



**Figure 12:** An example of four encounters involving four nodes and the resulting weighted graph. Each encounter becomes a vertex in the graph, and encounters with nodes in common are weighted with the time difference.



**Figure 14:** The witness complex and persistence barcode for an experiment in the two-loop space. The x and y coordinates in the witness complex plot correspond to the physical location of the encounters and are used to visualize the results, but are not used in the computation. The z-axis corresponds to the encounter time.

with data clustering, this analysis technique can find structure in a data set, but it cannot explain what that structure means. It can be used in exploratory analysis to better focus any further investigation.

The examples in the following sections provide some cases where interpretation of the results is not so straightforward.

### Detecting changes in a space

Persistent homology can detect certain types of changes in a space, for

example, a single loop that starts out small, enlarges to a certain extent, and then shrinks down again. Assuming the time for a node to circumnavigate the shrunken loop is comparable to the times between encounters on the expanded loop, the shrunken loop will appear contractible. The persistent homology of the encounter complex should approximate that of a sphere; that is,  $\beta_0 = 1$ ,  $\beta_1 = 0$ , and  $\beta_2 = 1$ .

This experiment was performed using the same simulator as before. At time  $t_{start}$  the graph was scaled down by a factor of 10, relative to the regular size,

while the node velocities were the same as usual. Then the graph was scaled up at a constant rate, reaching normal size (edge length 1) at time  $t_{mid}$ . Finally the graph was scaled back down to the initial size at the opposite rate that it was expanded, ending at  $t_{end}$ .

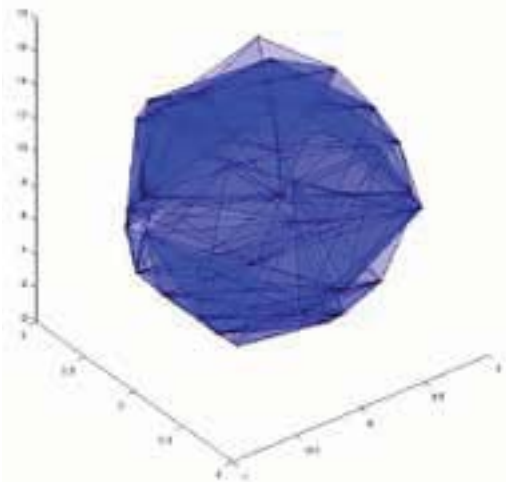
The persistent homology algorithm does recover the correct Betti numbers. Figure 15 shows the witness complex recovered from this experiment. Since the attempt was to recover the second Betti number, it was necessary to fill in 3-simplices in the witness complex. The number of simplices in a complex tends to increase rapidly with dimension, which was the case here. There were 56 0-cells, 1,074 1-cells, 11,703 2-cells, and 86,568 3-cells.

### Detecting changes in a 2D space

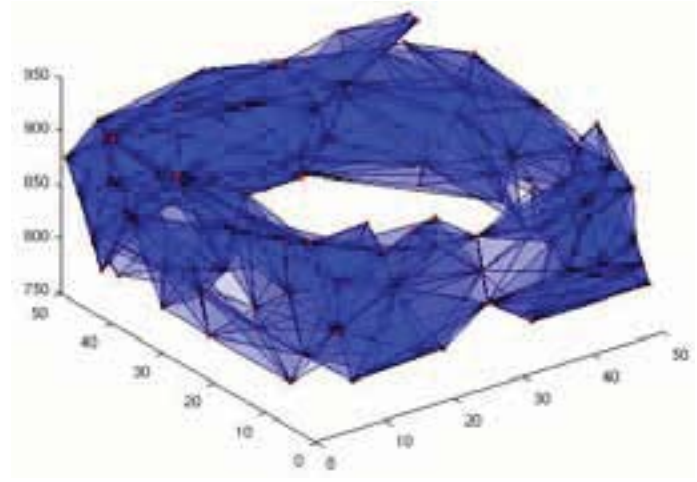
Let us return to the squirrel example from the introduction. Squirrels are not restricted to linear and loop-shaped spaces. It is important to determine if recovery of topological information from nodes living in a more general space is possible.

An experiment took place in which 50 simulated squirrels performed discrete random walks on a bounded two-dimensional grid. After 5,000 simulation steps, the squirrels' mobility model is changed so that the squirrels are repelled by the center of the grid. This repulsive force causes them to congregate near the boundary of the space. Then, after another 5,000 steps, the repulsive force is removed and the squirrels randomly fill up the grid again.

The encounter complex during the random walk phase of the experiment would be expected to have no real topological features. On the other hand, the encounters during the middle phase of the experiment, when the squirrels are repelled by the center of the grid, should have the homology of a loop (see Figure



**Figure 15:** The witness complex obtained from the expanding/contracting loop experiment. The Betti numbers of a sphere were recovered, demonstrating that persistent homology can detect certain types of changes in the physical space.



**Figure 16:** The witness complex obtained for the repulsion phase of the 2D random walk experiment. The number of simplices made it intractable to compute the homology of the entire data set at once.

16). Amazingly, these features for both phases were recovered. Combining all three phases into one computation should yield a non-zero second Betti number much like the expanding/contracting loop experiment. Unfortunately the number of simplices involved made the combined computation intractable with the MATLAB modules.

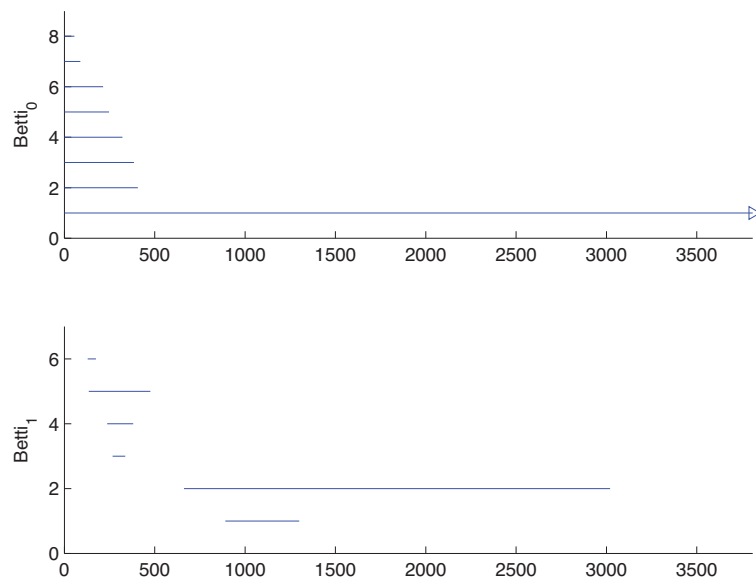
### Experiments with real encounter data

The Haggie Project encounter data includes data from three experiments. The same methods were used in the previous sections were applied to analyze the data from the Cambridge Computer Lab experiment.

According to the documentation, the experiment was conducted over seven days in January 2005 at the University of Cambridge Computer Lab. Nineteen iMotes were carried by graduate students from the System Research Group. Only 12 of the mobile motes yielded usable data, and an additional 210 external Bluetooth devices appeared in the traces. For the analysis, encounters with external devices were filtered out and the data sorted by encounter time; trusting that the iMote clocks were sufficiently well synchronized that this sorting made sense.

Persistent Betti numbers were computed for dimensions 0 and 1 for each day's data individually. The persistence diagrams for the first day are shown in Figure 17. A single fairly persistent 1-cycle on the scale of 50-65 minutes was observed in the results for each day. The cycle was particularly prominent on day one, but similar features appear in days two and three.

In this case, the observed features are probably not due to the space the experiment was conducted in, though that cannot be ruled out, either. Based on the simulation experiments, many more mobile nodes would be required to reliably identify a spatial cycle. The most likely explanation for these results is some sort of scheduling. For example, a group of mote carriers may encounter




**Figure 17:** The persistence barcodes from the first day of the Haggie Cambridge Computer Lab experiment. A fairly persistent 1-cycle on the scale of 50-65 minutes was observed. These results cannot conclusively be explained, but they do demonstrate how persistent homology can reveal topological structure in real encounter trace data.



another group at a meeting or class, and then another encounter may take place at lunch. Unfortunately, not enough is known about the particular experiment to draw a definitive conclusion. It would be worth performing a similar experiment that records more details and is perhaps more tightly controlled to see how different behaviors affect the persistent homology results.

These results show that topological analysis methods such as persistent homology can find structure in real encounter data that may not have been accessible via traditional statistical methods.

## Conclusions

Algebraic topology gives us powerful tools for uncovering features that may not be accessible through traditional statistical methods, and powerful and elegant ways of describing these phenomena. These tools will be useful for discovering structure in complex data. 

## References

- [1] T. Small and Z. J. Haas. The shared wireless infostation model: a new ad hoc networking paradigm (or where there is a whale, there is a way). In *MobiHoc '03: Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking & Computing*, pages 233–244, New York, NY, USA, 2003. ACM Press.
- [2] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet. In *ASPLOS*, San Jose, CA, October 2002.
- [3] U.S. Fish and Wildlife Service. Patuxent research refuge visitor brochure. [Online]. Available: <http://www.fws.gov/northeast/Patuxent/prinfopage.html>.
- [4] G. E. Bredon. *Topology and Geometry*. Springer, 1993.
- [5] A. Hatcher. *Algebraic Topology*. Cambridge University Press, 2002.
- [6] K. Mischaikow and et al. Chomp: Computational homology project. [Online]. Available: <http://chomp.rutgers.edu/>.
- [7] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. In *FOCS '00: Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, page 454, Washington, DC, USA, 2000. IEEE Computer Society.
- [8] A. Zomorodian and G. Carlsson. Computing persistent homology. In *SCG '04: Proceedings of the Twentieth Annual Symposium on Computational Geometry*, pages 347–356, New York, NY, USA, 2004. ACM.
- [9] V. de Silva and G. Carlsson. Topological estimation using witness complexes. In *Sympos. Point-Based Graphics*, pages 157–166, 2004.
- [10] Plex: Persistent Homology Computations. [Online]. Available: <http://comptop.stanford.edu/>
- [11] P. Bubenik and P. T. Kim. A statistical approach to persistent homology. *Homology, Homotopy, and Applications*, 9(2):337–362, 2007.
- [12] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Stability of persistence diagrams. In *SCG '05: Proceedings of the Twenty-first Annual Symposium on Computational Geometry*, pages 263–271. ACM, 2005.
- [13] V. de Silva and R. Ghrist. Coverage in sensor networks via persistent homology. *Algebraic & Geometric Topology*, 7:339–358, 2007.
- [14] F. Chazal and S. Y. Oudot. Towards persistence-based reconstruction in euclidean spaces. In *SCG '08: Proceedings of the Twenty-fourth Annual Symposium on Computational Geometry*, pages 232–241. ACM, 2008.
- [15] G. Carlsson, T. Ishkhanov, V. de Silva, and A. Zomorodian. On the local behavior of spaces of natural image. *International Journal of Computer Vision*, 7:339–358, 2007.
- [16] H. Edelsbrunner and J. Harer. Persistent homology - a survey. In *Twenty Years After*. AMS, 2007.
- [17] E. Carlsson, G. Carlsson, and V. D. Silva. An algebraic topological method for feature identification. *Int. J. Comput. Geometry Appl.*, 16(4):291–314, 2006.
- [18] R. Ghrist. Barcodes, The persistent topology of data. *AMS Current Events Bulletin*, 45(1):61–75, 2008.
- [19] B. Walker. Using persistent homology to recover spatial information from encounter traces. In *MobiHoc '08: Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 371–380, New York, NY, USA, 2008. ACM Press.
- [20] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot. Pocket switched networks and human mobility in conference environments. In *WDTN '05: Proceeding of the 2005 ACM SIGCOMM Workshop on Delay-tolerant Networking*, pages 244–251, New York, NY, USA, 2005. ACM Press.
- [21] S. Shakkottai. Technical report on Bluetooth mote encounter experiments. Technical report, UT-Austin, in preparation.
- [22] M. Balazinska and P. Castro. Characterizing mobility and network usage in a corporate wireless local-area network. In *1st International Conference on Mobile Systems, Applications, and Services (MobiSys)*, San Francisco, CA, May 2003.
- [23] M. Mcnett and G. M. Voelker. Access and mobility of wireless PDA users. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9(2):40–55, April 2005.
- [24] T. Henderson, D. Kotz, and I. Abyzov. The changing usage of a mature campus-wide wireless network. In *MobiCom '04: Proceedings of the 10th Annual International Conference on Mobile Computing and Networking*, pages 187–201, New York, NY, USA, 2004. ACM Press.
- [25] W.J. Hsu and A. Helmy. On nodal encounter patterns in wireless LAN Traces. In *Proceedings of the Second Workshop on Wireless Network Measurements (WinMee 2006)*.
- [26] V. de Silva. A weak characterization of the Delaunay triangulation. [Online]. Available: <http://pages.pomona.edu/~vds04747/public/papers/deSilva WeakDelaunay.pdf>
- [27] E. Morris. *Fast, Cheap & Out of Control*. [DVD]. Columbia TriStar Home Entertainment, 1997.
- [28] Walt Disney Television Animation. *Disney's Kim Possible*. Disney Channel, 2002. [Online]. Available: <http://tv.disney.go.com/disneychannel/kimpossible/>



# Host and Network Integrity *through* Trusted Computing

## Take Control of your network

### Overview

One of the biggest challenges facing computer network administrators today is keeping track of the hosts on their networks. Without this knowledge, it is impossible to keep all hosts patched, up-to-date, and protected from infection and exploitation by malware.

Trusted computing technologies can help administrators take control of their networks so that they can begin to address security problems. Products that leverage these technologies are becoming more and more widely available. Network owners should position themselves to take full advantage of these new products by making sure that they purchase hosts that support the full range of trusted computing technologies.

### Trusted Computing Group

The Trusted Computing Group (TCG) is an industry and government consortium formed to develop and promote standards for trusted computing technologies. They have produced specifications and guidance for—among other things—the hardware TPM, the measured boot and launch of PC operating systems, and the TNC network security architecture.

### Trusted Platform Module

Trusted Computing Technologies are included in most PC desktop systems sold today. The most common is the Trusted Platform Module (TPM). The TPM is a motherboard-based cryptoprocessor with capabilities that include secure generation and storage of cryptographic keys, and generation of random numbers.

An important capability of the TPM with respect to host integrity is the accumulation and secure storage of system measurements. Measurements are hashes of host software computed by the host and

accumulated within the TPM. If the same components are measured at a later time and the measurements have changed, then the components have changed. This mechanism can be used to detect whether system software has been infected with malware.

### Measured Boot and Measured Launch

Measurement is a powerful capability for generating information about the integrity of software and data. Many hosts that support a TPM include a Trusted Computing Group (TCG)-compliant BIOS that automatically measures the host's pre-boot environment. When compared with prior measurements, this measurement indicates whether the BIOS, boot loader, and other low-level system components have been modified since the last system boot.

Many modern microprocessors support a measured launch capability that can be leveraged to ensure the integrity of a post-boot software environment—such as an operating system kernel or virtual machine hypervisor. The measured launch may be used in conjunction with pre-boot measurements to provide reasonable assurance that critical system components have not been modified since the last launch. This potentially powerful capability is provided by microprocessors that support Intel Trusted Execution Technology (TXT) and AMD-V virtualization.

### Network Access Control

Simply measuring pre- and post-boot environments is not enough to ensure network integrity. In order to actually improve the security of a network, the measurements computed for individual hosts must be collected and acted upon. At the very least, measurements should be reported to system administrators, who can then decide whether action is needed. Ultimately, systems can attest their integrity to a centralized network access-control point using an architecture such as Trusted Network Connect (TNC). The control

point can decide whether the host should be allowed on the network.

### Recommendations

Trusted Computing Technologies can provide network administrators with basic information about host integrity without expensive hardware or excessive administrative overhead.

The potential benefits of trusted computing are well worth the minimal investment. While today it is hard to buy a PC that does not come with a TPM, hosts that support measured launch are less common. When purchasing new hosts, system owners should look for desktops and servers that include a TPM and support for measured launch and protected execution—such as Intel's Trusted Execution Technology (TXT) or AMD-V virtualization technology.

Hosts that support TPMs should have their TPMs turned on and activated from the BIOS. This enables measurement of the pre-boot environment, and is necessary for measured launch. For more information on trusted computing and taking advantage of the TPM, see "How to Use the TPM: A Guide to Hardware-Based Endpoint Security," on the TCG website.

[www.trustedcomputing.org](http://www.trustedcomputing.org)

For more information,  
Email: [hostintegrity@tycho.nsa.gov](mailto:hostintegrity@tycho.nsa.gov)



**Enable  
your Trusted  
Platform Module**

