If you're looking for a low-maintenance IP communications network, peer-to-peer SIP might be just the thing.

# Decentralizing

DAVID A. BRYAN AND BRUCE B. LOWEKAMP, SIPEERIOR TECHNOLOGIES

# SIP

SIP (Session Initiation Protocol) is the most popular protocol for VoIP in use today.[1] It is widely used by enterprises, consumers, and even carriers in the core of their networks. Since SIP is designed for establishing media sessions of any kind, it is also used for a variety of multimedia applications beyond VoIP, including IPTV, videoconferencing, and even collaborative video gaming.

In the past three years, interest in decentralized, peer-to-peer SIP (P2PSIP) has increased. P2PSIP removes or reduces the number of centralized servers needed in a SIP deployment.[2,3,4,5] There has been much speculation that this interest can be attributed to Skype, the popular pseudo-P2P communications service (Skype still tightly centralizes authentication, billing, and admission control). Although one potential use is to build a SIP-based, low-cost, server-less worldwide network, much of the interest has to do with enabling SIP to operate in deployments where conventional server-based SIP isn't well suited.

One major deployment scenario where P2PSIP appears superior to a server-based solution is for small-office deployments where users may have little or no technical support capacity. The self-organizing aspects of P2P lend themselves to systems that are far easier to configure and manage than even simple centralized servers. For small-office applications, configuration may consist of nothing more than typing an extension number into each device. P2PSIP is also being considered to provide highly reliable systems because of its lack of a single point of failure, and as a mechanism for sharing information between existing SIP servers in larger deployments. Outside of the obvious telephony applications, P2PSIP shows promise for disconnected or ad hoc communications environments, emergency responder networks, and even clusters of consumer electronics devices streaming media to one another.

In this article, we explain what P2PSIP is, how it differs from conventional SIP, where this technology is being used, efforts toward standardization, and the future of P2PSIP. First, we take a look at the basics of P2P technology, particularly which types of P2P are being used or considered for P2PSIP.

## WHAT IS P2P?

A variety of definitions exist for P2P systems, which can even be P2P to a greater or lesser degree. At the most basic level, a P2P system is one where multiple software applications interact directly with one another as peers to accomplish a task. The group of peers as a whole is often referred to as an overlay. This is in contrast to the more traditional client-server model, where one centralized piece of software (the server) processes requests from numerous clients. Choosing P2P or client-server is an architectural decision about where the processing of information takes place. For deployment scenarios such as disconnected networks of devices, a P2P solution may be the only option available. In other deployments, the choice is dictated by economic or configuration considerations, and the end user may be unaware of, and perhaps not even care, where the processing takes place.

The belief that P2P is a fundamentally new idea is a common misconception. Many common network protocols, including BGP (Border Gateway Protocol) and even SMTP, are arguably P2P, but between instances of servers or routers. Like many aspects of the Internet, we are now seeing this architecture moving to the edge. Increasingly peer groups are made up of end-user applications, in contrast to groups of managed servers. When people refer to something as P2P today, they generally mean P2P between end-user applications.

A P2P architecture doesn't necessarily imply that every peer must provide every service or store all the available data. Collectively, all the peers in the overlay must provide all services, but any one particular peer may provide only a fraction. For example, a collection of peers replicating a database might each store a small number of entries from the database. If a very large group of peers splits up the task, the odds of one particular entry being on a given peer are quite low, but at least one peer in the overlay will store a given entry, guaranteeing that as a group the peers provide the full database service.

In contrast to centrally managed servers, in many P2P systems peers are assumed to be ephemeral in nature. Because the peer software may be running on unmanaged end-user machines, they may be available only while the software is running and may disconnect at any time for a variety of reasons. The constant change in the makeup of the peers in the overlay, referred to as *churn*, is an important consideration in developing P2P applications.

In some P2P architectures, a subset of the peers provides more services than the others. These peers, often called *super-peers*, may even be the only peers providing services. In such architectures, the super-peers may collectively replace the servers, with the remaining peers behaving essentially as clients communicating with the super-peers for their services. This approach is often used in the presence of NATs (network address translators), where peers behind NATs may be unable to receive requests and therefore can't fully participate as peers.

Yet another common variation of P2P is the hybrid architecture. Hybrid P2P systems use a centralized server to locate a particular peer offering a service, but the service takes place directly between the peers. The best-known use of P2P, online file sharing, often worked this way. Each peer sharing files published a list to a central server. A user looking for a specific file searched on the central server to locate a peer with that file, then transferred the file directly from the peer that was storing it. Today's SIP systems can be thought of as hybrid P2P systems.

As P2P technology has evolved, the mechanisms to distribute and locate data have fallen into two broad categories, unstructured and structured.

## UNSTRUCTURED P2P

The earliest P2P systems were unstructured (figure 1). In these unstructured systems, the peers are organized in a haphazard way. Each new peer locates and connects to one or a few other peers in the overlay. There is no mechanism for selecting to which peers a new peer connects; any available peer will do. As a result, some peers may be connected to only a few other peers, while others may be connected to many. The data stored or services provided by each peer are similarly randomly distributed. For data storage, this means any peer in the overlay can store a given piece of information, and that the information may not be well distributed among the peers. The connection between the peers is a logical structure, meaning that the connections don't have to be related to the underlying physical network in any way.

Such an arrangement is easy to form and requires little overhead to maintain, but searching can be very difficult. Because any peer can store the information, every peer must be queried to be certain the data isn't present in the overlay. Because the structure is random, it is impossible to know how many other peers a peer will be connected to, or how many hops away the farthest peer will be. Peers can be easily split from the overlay, since there is no structure ensuring redundant links between portions of the overlay. Exhaustive searches can be very time consuming in large networks, and limiting the search by capping the depth of the searches results in nondeterministic searches, since not every peer is consulted.

For these reasons, unstructured P2P has fallen out of favor for use in large or Internet-scale deployments. For smaller deployments, or deployments where the underlying network itself may have an unstructured arrangement (particularly sensor networks and other ad hoc or wireless network arrangements), this approach has proven to be well suited and is still widely used.

## STRUCTURED P2P AND DHTS

In contrast, in a structured P2P architecture (figure 2) the peers are connected to one another in a defined, logical structure—for example a ring, tree, or grid. Many arrangements are possible, but in most, peers are assigned a (hopefully) unique identifier when they join the overlay. This peer identifier, or PeerID, could be assigned by some out-of-band mechanism, selected randomly, or most commonly, determined by hashing a property of the peer such as the IP address. The PeerID determines which other peers the new peer makes connections with. For example, the new peer may connect to peers with identifiers that are "close" in some mathematical sense such as numerical value or number of matching binary digits.

Since the connections between peers are carefully controlled, a well-designed structured P2P algorithm can ensure that each peer is connected to several others, preventing partitioning when a single peer fails. Because the structure of the overlay is controlled, the total distance between any two peers can be controlled, limiting the number of hops between them.

One particular flavor of structured P2P that is widely deployed is the DHT (distributed hash table). Some of the most widely discussed DHT algorithms today include Chord, Kademlia, and Bamboo.[6,7,8] In a DHT, not only is the structure of connectivity between the peers controlled in a mathematical way, but the placement of resources onto the peers is as well. Each resource is assigned an identifier, or ResourceID, in the same identifier space as the PeerID. That is, the range of values a PeerID or ResourceID can take on are the same. The ResourceID is the hash of a property of the resource such as a filename or keyword. A resource's keyword is hashed to produce a ResourceID, and the peer with the "closest" PeerID stores the resource. Both the definition of *close* and provisions for redundancy are dependent on the particular DHT algorithm used.

For example, if a ring-like structure is used as the logical structure for connecting the peers, a resource with a ResourceID of *n* might be stored on the peer with the

An Unstructured P2P Overlay

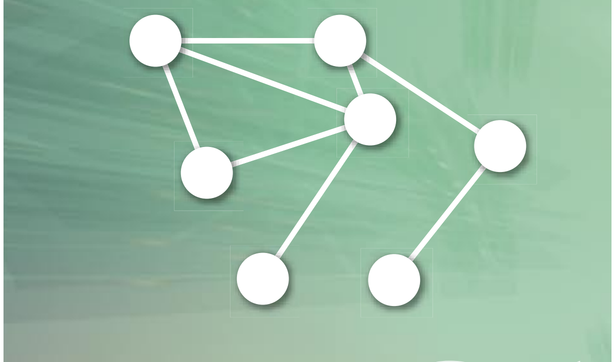

FIG 1

A Structured P2P Overlay

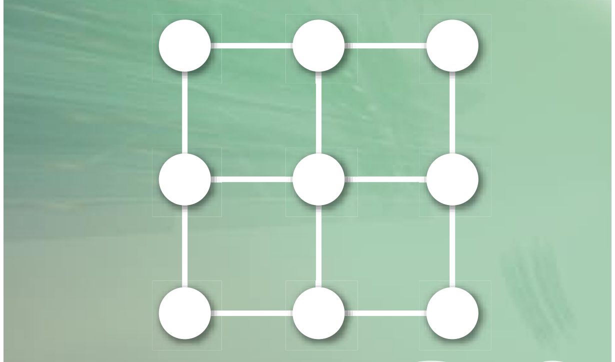

FIG 2

PeerID closest to but larger than *n* (see figure 3). In this case, we show five peers (circles), with PeerIDs of 100, 200, 300, 400, and 500. The system stores two resources, with ResourceIDs of 345 and 444, shown by squares. The arrows indicate which peer is responsible for storing each resource in the system.

When some other peer wants to locate a resource later, it hashes the distinguishing name of that resource and uses the overlay to contact the peer with the nearest PeerID. That peer provides the resource if present, or it can report that the resource doesn't exist if it is not stored by the overlay. This mechanism requires fewer messages to be sent to locate data and provides deterministic search, ensuring that the unique responsible peer is queried for the resource.

Since each peer connects directly only to some subset of the entire overlay, the search may still take more than one hop. A peer may ask a neighbor that is closer to the desired PeerID, which then asks another still closer neighbor, etc. Most of the DHTs used today are structured in such a way that they can guarantee that at most log(n) peers, where *n* is the total number of peers in the overlay, must be consulted to locate a particular peer. Although this is obviously a higher search cost than the direct query and response of a client-server network, it results in relatively low search times even for overlays with a large number of peers, while eliminating central servers.

## SIP IS ALREADY (MOSTLY!) DECENTRALIZED

Why all the interest in trying to make SIP P2P? Why not create a new protocol? Starting with an existing standard allows leveraging work that has already been done—several years of effort in the case of SIP. Additionally, SIP has many features that make it attractive for conversion to P2P. SIP was designed to move as much functionality as possible to the edges. SIP endpoints are intelligent in that most call signaling is handled between the endpoints. When a call is established, the messages initiating the call, indicating call progress (ringing and answering, for example), and terminating the call all originate on the endpoints. Capabilities such as feature sets and available media codecs are negotiated directly between the endpoints as well. Once the call is established, media flows directly between the endpoints. In fact, if two SIP phones are configured so they each believe the other phone is the SIP server, they will communicate and use all features between themselves perfectly. No servers required!

Resource Storage in a Ring-like DHT

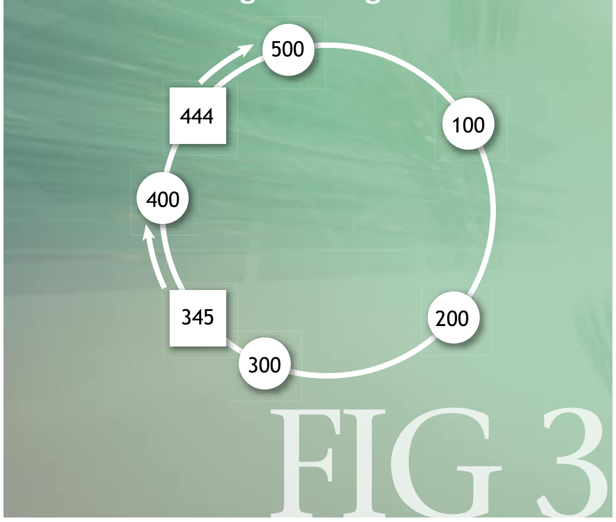

FIG 3

What is centralized in SIP is resource location. Connecting two phones by convincing each the other is a server proves that the features and protocol work between them, but it doesn't prove much else. The primary purpose of the SIP server is to locate other endpoints.

In a conventional SIP network, each phone communicates with the central SIP registrar server. Each user has an AoR (address of record), typically a username or phone number. The registrar maintains a mapping between the AoR and the IP address of that user's phone. Each entry mapping one AoR to an IP address is called a registration. When one user wishes to call another, the first user's phone contacts the SIP proxy, which contacts the registrar. The registrar looks up the requested AoR and returns the IP address to the proxy. The proxy then uses the address to proxy the call to the destination phone.

In many networks, the proxy and the registrar are implemented as one software package, called simply a SIP server, or confusingly, sometimes just a proxy. The central server does little for the call other than ensuring that the endpoints are able to locate each other. As previously noted, SIP in many ways already qualifies as a hybrid P2P system. The intelligence and services reside in the endpoints, and a centralized server is used simply to allow the peers—in this case SIP phones—to locate each other and communicate.

## P2PSIP: REMOVING THE REGISTRAR

P2PSIP can be thought of as taking the last vestigial centralized functions in SIP and distributing them among the phones. The most important centralized aspect of a SIP system is the registrar, and the core of P2PSIP is a fully decentralized, server-less registrar replacement. In P2PSIP, the peers store the registrations, rather than the centralized server.

Some early commercial attempts at P2P telephone systems took a "replicate everything everywhere" approach, using a broadcast mechanism to exchange data. Each phone periodically broadcasts user registration information. Every other phone sees the broadcasted registration and stores the mapping. If a phone drops off the network, it no longer periodically broadcasts, and the registration is eventually removed from the other phones' registration tables. To place a call a phone simply looks up the number it wishes to call in the local table of registrations and places the call directly between the phones.

While this approach works for small-office systems, it has a number of drawbacks. First, the broadcast mechanism is poorly suited for overlays that span multiple networks. Broadcast traffic (in general) does not survive crossing routers, precluding overlays of distributed users—for example, a group of consumers on the broader Internet. Scalability is also a problem for such systems, since each peer needs to store information about every other peer.

More recently, a number of companies, as well as the IETF (Internet Engineering Task Force), have focused on using a structured P2P approach that works for both large and small systems. In particular, designs build on a DHT. Each phone serves as a peer in the overlay. The phone is assigned a PeerID, either the hash of an IP address or a value assigned by a centralized security server. Once the server assigns a PeerID, the server never needs to be consulted again during basic operation. The AoR of the user of that phone is hashed into the same space to produce a ResourceID, and the nearest peer stores the registration. Each peer is responsible for some portion of the identifier space; therefore, some of the registrations are stored on each peer.

As with any DHT, each peer knows about some fixed number of peers distributed across the overlay. To route a message, the peer with the closest PeerID to the requested ResourceID is selected from the list of known peers. The message is then sent to this closest known peer. The process repeats, converging on the peer with the nearest value.

When a phone joins the overlay, it determines its PeerID and exchanges some messages to place itself into the overlay. This process is called *peer joining*, since it is the process by which the phone becomes one of the responsible peers in the overlay. The peer learns about other peers in the overlay so it can route messages, inserts itself into the overlay, and becomes responsible for some portion of the identifier space. Since some other peer used to be responsible for storing the registrations in that portion of the identifier space, those registrations are transferred to the new peer during the join process. At this stage, the phone itself has become part of the overlay and can participate in storing and locating resources, but the registration of the user of the phone has not yet been accomplished. As peers leave, they should hand stored resources to other devices, but various redundancy schemes are employed to prevent loss of data in the event that one or more phones fails and takes some information with it.

The next step is to hash the username or phone extension to produce a ResourceID for that user. A message is constructed, containing a mapping between the AoR and the IP address of the phone. In conventional SIP, this message would be sent to the centralized registrar. In P2PSIP, the registration message is instead routed through the hops of the overlay. Once the message reaches the responsible peer, it stores the mapping, responding to the sender to indicate success. As with conventional SIP, these registrations last for a finite period of time and must be refreshed periodically to prevent stale registrations. A well-behaved peer will request that any registrations for its associated user be removed in the case of a clean shutdown.

When a user wants to call another party, the AoR is again hashed, producing a ResourceID for the party to be called. A query message (or, alternatively, the message to initiate the call directly) containing the AoR is routed through the hops of the overlay to the peer responsible for the ResourceID. If the message is a query, the receiving peer looks in its local table of mappings and returns the IP address of the requested username or phone extension, or it returns a not-found if the value isn't stored. If the message received is to initiate the call directly, the peer sends back a not-found if the value isn't stored; otherwise, it forwards the message to the IP address stored in the registration map.

Once the destination peer is reached, the call occurs between the two peers without any further involvement of the overlay. The beauty of P2PSIP is that the interaction between the peers at this stage is pure SIP. Implementers who are building on existing SIP deployments can reuse existing code for features, enhancements, etc. Existing SIP devices can also be easily incorporated simply by pushing a registration into the overlay of the conventional SIP device.

## SECURING THE OVERLAY

The proposed mechanism in the IETF, as well as the mechanism chosen for some commercial deployments, is to issue each user in the overlay a certificate. A centralized server issues this certificate, but that server does not have to be consulted again. Instead, the new phone, when first joining, presents the public portion of the certificate to the overlay, which stores it. Since the same central server that issued the other phones' certificates has signed the certificate, it can easily determine that the new phone's certificate is valid. The public part of the certificate remains in the overlay and can be retrieved by a caller and used to encrypt traffic sent to the new phone. In this way, the certificates provide security both for the integrity of the overlay and for media between users of the overlay.

## INTEROPERABILITY AND ADVANCED FEATURES

Most of the features in SIP can be used in a P2PSIP manner unmodified. SIMPLE (SIP for Instant Messaging and Presence Leveraging Extensions) is a set of IM and presence extensions for SIP. SIMPLE-based IM will work in a P2P deployment with essentially no modification. Presence requires some modification, however. In a conventional SIMPLE presence environment, each user publishes information about his or her status to a centralized server, and interested parties subscribe to the presence information. Since a P2PSIP deployment has no centralized servers, the presence information must be stored on the peers. The same mechanism used for locating the registration information for a peer can be used to locate a responsible peer for presence updates. Although this requires each peer to implement presence server behavior, the additional overhead required has not proven to be large.

One of the driving reasons to use SIP for a new serverless communications protocol is a desire for interoperability. While registration is distributed in a P2PSIP deployment, basic call signaling is performed using conventional SIP. This means that interoperability is reduced to the problem of locating the remote party. P2PSIP endpoints that fail to locate a remote party in the overlay can fall back to conventional DNS-based lookup of the SIP address. Static registrations can be stored into the overlay by a helper application, allowing P2PSIP systems to locate and directly communicate with SIP endpoints, gateways, or even application servers. Conventional SIP calls to a P2PSIP system are performed in a similar way. Either a static route to the P2PSIP domain is configured into the conventional SIP server, or one or more peers is consistently available and associated with a DNS-resolvable address, allowing the P2PSIP overlay to be reached using conventional SIP DNS-based resolution techniques.

Voicemail is another feature that must be handled differently in a P2PSIP deployment. In conventional SIP systems, voicemail is stored on a centralized voicemail system. P2PSIP systems can still be configured to use a voicemail server, of course, but since much of the motivation behind P2PSIP involves distributing as much intelligence as possible, this isn't a practical approach.

Storing voicemail in the overlay is essentially the same as using an overlay for file sharing, since voicemail messages are typically small sound files. The peer responsible for the ResourceID of a particular user can store these files. When the user comes online and attempts to register, he or she will find the messages waiting. The sender uses the security certificates to encrypt voicemail messages left for a user.

## REDUNDANCY

Since the devices in the network might come and go, particularly in the case of a global network of home users, redundancy is a critical feature. In most cases, each item of information (registrations, voicemail, etc.) is stored on at least three alternate servers. Various mechanisms have been employed for this, including the three nearest to the hash value, or using three different hash mechanisms to produce three different locations for redundancy. Commercial P2PSIP systems all incorporate some form of redundancy, as will the IETF standard that emerges.

## STANDARDIZATION EFFORTS

Meetings on standardization have been ongoing at the IETF since March 2005. In the beginning these meetings were ad hoc and informal, and most recently led to a formal BoF (birds-of-a-feather) session at IETF-67 in November 2006. Interest among IETF participants is strong; the P2PSIP BoF was the best attended of all the sessions at IETF-67. As of this writing, the IETF is in the final stages of forming a full working group to evaluate P2PSIP technology and develop a standard.

In the meantime, the participants in the informal meetings at IETF have been very busy. Attendance at these meetings has approached 200, making them better attended than nearly any other meetings at the IETF. A community Web page (http://www.p2psip.org) has documented the efforts of the work to date. Nearly two dozen drafts have been written and are being considered by the informal group, and the mailing list for discussion of P2PSIP technology has hundreds of messages a month.

It appears likely that the IETF will adopt a standard for P2PSIP in the near future. Current technical points that are being debated include which DHT is to be used and whether messages for maintaining the DHT are passed as SIP messages or some other protocol. The most mature of the proposals uses SIP messages to convey DHT information. One motivation for using SIP is simplicity of implementation. Because the device already speaks SIP, the need for a second protocol stack is eliminated. Additionally, SIP as a transport for the DHT maintenance information seems the logical choice, as SIP already incorporates many features that would otherwise need to be defined in a new protocol, including security primitives, NAT traversal, and both redirect and proxy routing mechanisms.

In the meantime, vendors are implementing their own flavors and variations, which will provide deployment experience for the emerging standard. Most of the interested parties have expressed an interest in moving to an IETF standard as it emerges.

## THE FUTURE OF P2PSIP

P2PSIP won't be a replacement for SIP and was never intended as such. It is an enhancement and companion to SIP, enabling SIP to be used in scenarios where it might not have otherwise been easily deployed. It leverages and extends the work and the deployed infrastructure of the SIP community. Conventional client-server SIP will continue to be the preferred choice for many deployments, and in some locations, systems that incorporate both conventional and P2P portions are likely to emerge.

P2PSIP is available today from a number of vendors, but we are at the very beginning of the adoption curve. Today, the various implementations are still proprietary, causing interoperability issues, but the standardization work of the IETF will improve the situation in the future. P2PSIP appears poised to become a powerful tool in the application developer's arsenal. It is likely to find widespread adoption in VoIP for small-enterprise systems, disconnected and ad hoc deployments, and global, decentralized deployments.

It also seems likely to be a strong contender for a protocol in IPTV and between consumer electronics devices in the home, where it can be used to cluster components and allow them to establish multimedia sessions between themselves. If current trends hold, there may be a number of P2PSIP-enabled devices in your future. Q

## REFERENCES

1. Rosenberg, J., et al. 2002. SIP: Session Initiation Protocol. RFC 3261 (June).
2. Bryan, D. A., Lowekamp, B. B., Jennings, C. 2005. SOSIMPLE: A server-less, standards-based P2P SIP communications system. *First International Workshop on Advanced Architectures and Algorithms for Internet Delivery and Applications* (June).
3. Singh, K., Schulzrinne, H. 2005. Peer-to-peer Internet telephony using SIP. *15th International Workshop on Network and Operating Systems Support for Digital Audio and Video* (June).
4. Bryan, D., Lowekamp, B., Jennings, C. 2006. A P2P approach to SIP registration and resource location. *IETF Internet Draft* (work in progress, October).
5. Willis, D., Bryan, D., Matthews, P., Shim, E. 2006. Concepts and terminology for peer-to-peer SIP. *IETF Internet Draft* (work in progress, November).
6. Stoica, I., et al. 2003. Chord: A scalable peer-to-peer lookup protocol for Internet applications. *IEEE/ACM Transactions on Networking* (February).
7. Maymounkov, P., Mazieres, D. 2002. Kademlia: A peer-to-peer information system based on the XOR metric. *Proceedings of the 1st International Workshop on Peer-to-Peer Systems* (March).
8. Rhea, S., Geels, D., Roscoe, T., Kubiatowicz, J. 2003. Handling churn in a DHT. *Technical Report UCB//CSD-03-1299, University of California, Berkeley* (December).

**LOVE IT, HATE IT? LET US KNOW**
feedback@acmqueue.com or www.acmqueue.com/forums

**DAVID BRYAN** plays an active role in IETF P2PSIP efforts and has published numerous IETF drafts, academic papers, and industry trade articles on the subject. He is active in the SIP community, including heading up www.p2psip.org, the leading community Web site for P2PSIP, and is involved with SIPFoundry, the reSIProcate project, and www.vovida.org. He was co-founder and CTO of Jasomi Networks, a pioneer in the SIP Session Border Controller market, and he previously worked for Cisco Systems and Vovida Networks. He holds bachelor's degrees in computer science and physics from Richard Stockton College and a master's degree in computer science from the College of William and Mary, where he is completing his Ph.D.

**BRUCE LOWEKAMP** works with SIPeerior and is an assistant professor at the College of William and Mary. His research covers a wide range of topics in distributed systems and applications. He has published numerous articles on performance monitoring, peer-to-peer communications, adaptive applications, virtual machines for realtime distributed visualization, and simulation of ad hoc mobile networks. He has served as co-chair of the GGF's (Global Grid Forum's) Network Measurements working group and is a contributor to several GGF recommendations and IETF Internet drafts. He received his B.S. from Virginia Tech and his Ph.D. in computer science from Carnegie Mellon University.